AAMOT OLE 19780220 OSLO METROPOLITAN UNIVERSITY 20200630 BSc EE
Public Internet Radio Client for Accessing Free Audio Maps in Countries with Free Speech
GNOME Internet Radio Locator (gnome-internet-radio-locator)
GNOME Radio (gnome-radio) http://www.gnomeradio.org/

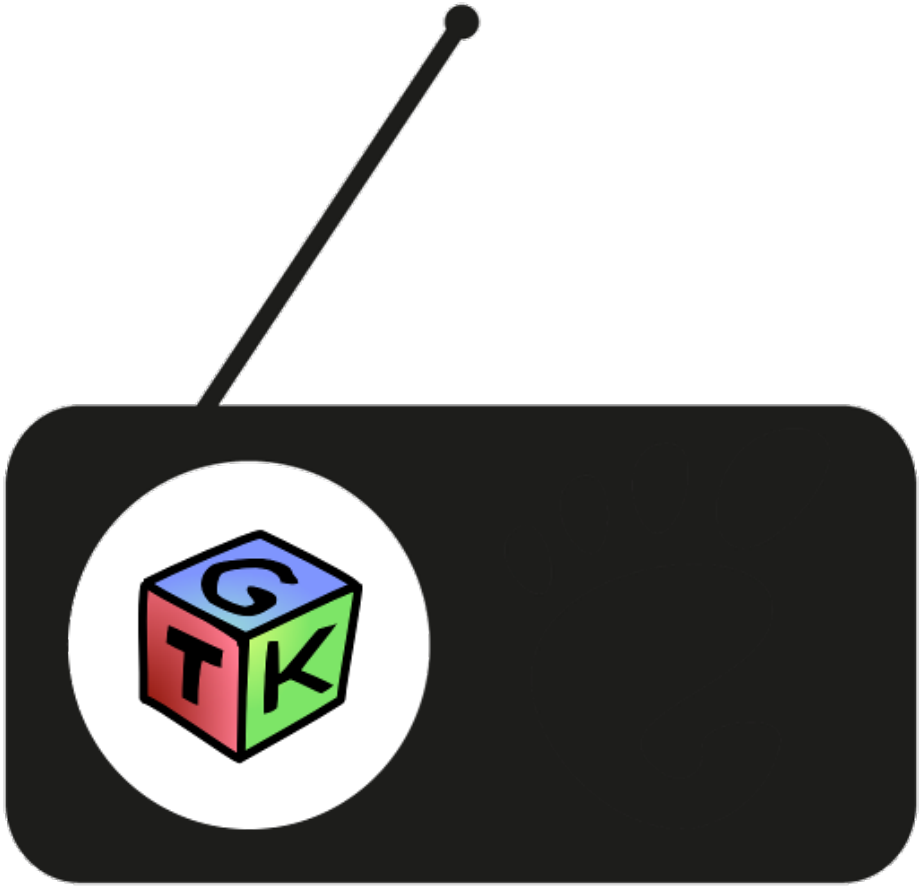
Public Internet Radio Client for Accessing Free Audio Maps in Countries with Free Speech

GNOME Internet Radio Locator (gnome-internet-radio-locator)
GNOME Radio (gnome-radio)

`http://www.gnomeradio.org/~ole/thesis.pdf`

Ole Kr. Aamot

June 24, 2020


G
T
K

Introduksjon

Denne Bachelor-oppgaven innen emnet medisinsk teknologi er dataprogrammet gnome-internet-radio-locator utviklet i løpet av 18 år for datamaskinoperativsystemet GNU/Linux og skrivebordet GNOME som har blitt utviklet siden juli 1997 og kontinuerlig forbedret i løpet av 22 år.

Min hypotese var at mennesker holder seg stabile ved å lytte til radiosendinger på en datamaskin fordi mennesker liker å høre til musikk på radio på datamaskin og stabile mennesker på radio.

Oppgaven er et ingeniørarbeid som tok 18 år å fullføre og er dedikert til Oda (musikkterapeut og musiker i Tøyen Botanical Crush) og mine to foreldre Gunhild (sykepleier) og Helge (ingeniør).

Programmet er implementert for GNOME i programmeringspråket ANSI C og er tilgjengelig for Debian GNU/Linux, Fedora Linux, Ubuntu Linux og MacPorts. Programmet kan lastes ned fra `http://www.gnomeradio.org/`
Du kan følge prosjektet på `https://wiki.gnome.org/Apps/Radio` og `https://gitlab.gnome.org/ole/gnome-radio`

Takk for velvilje fra studenter ved entreprenøruniversitetet Massachusetts Institute of Technology i Cambridge, MA som har utdannet utallige entreprenører i ingeniørvitenskap, Universitetet i Oslo og Høgskolen i Oslo og Akershus og venner, særlig du som hjalp meg og oppmuntret meg til å fortsette utdannelsen etter en arbeidsperiode i 9 år for Domeneshop AS fra mandag til fredag kl. 9 - 17, og pasienter ved Lovisenberg Diakonale Sykehus som deltok i spørreundersøkelsen. Takk til Graham Morrison i spalten FOSSPicks for britisk omtale av GNOME Internet Radio Locator i Linux Magazine 222 i mai 2019, Marius Nestor for omtale på Softpedia, Øyvind Sæther for anmeldelse på linuxreviews.org og Peter Norvig for invitasjon til Google i Mountain View, California, U.S.A.

Takk også til Ranveig, Anja og Veena, Asgeir, Daniel, Kristian, Arnfinn, Kent, Ståle, Dag og Jan som var svært tålmodige arbeidskolleger i Oslo i 9 år og takk til mine gode venner Tarjei og Håkon Tjønn, John Grande, Ami Niemela, Saga, Mio og Jan Ole Kjellesvig, Hans Petter Jansson, Daniel Mikkelsen, Lars Bungum, Karl-Erik Olausen, Geir Kaaresen, Oskar Nakken, Elnaz Asgari, Petter Reinholdtsen, Vetle Berg Abrahamsen, Janan Anes, Patrick Edvard Antonsen, og Hans Petter Solli, Sigrid Sandkjær, og Oda Bjørke Dypvik.

Dataprogrammet startet med frie radiosendinger på IMCbra med Elin og Ellinor sommeren 2002 under protestene mot Verdensbankmøtet i Oslo i 2002 og mennesker har fortsatt å kopiere og installere dataprogrammet til frie datamaskiner over hele verden via Internett via frivillig installasjon og samarbeid fra 2014 etter mitt besøk i New York City, Cambridge og Boston i Massachusetts, U.S.A. i juni 2014 og jeg startet igjen utviklingen 1. november 2014 og videreutviklingen for GNOME Maps og GStreamer etter mitt besøk på UC Berkeley, i Palo Alto og hos Google i Mountain View i California i august og september 2015 og endte med konklusjonen om at radio er stabiliserende og sosialiserende å lytte til i GNOME Internet Radio Locator for mennesker som var i karantene under

Korona-pandemien i verden mellom 12. mars 2020 og 15. juni 2020.

Til slutt stor takk til professor i psykiatri Astrid Nøklebye Heiberg ved Universitetet i Oslo, førsteamanuensis i arkivvitenskap Thomas Sødring ved Oslo Metropolitan University og programleder Fredrik Solvang i NRK som trodde på prosjektet mitt, samt overingeniør Jon Sverre Dischington Hanssen, ingeniør Vetle Berg Abrahamsen og medstudent Adrian Szabo Aabech for 9 Volt FM-senderne som Aabech og Abrahamsen bygget på ingeniørhøgskolen høsten 2017 og som fortsatt sender radio i 30 meter på FM 92.1 MHz.

Med et sterkt ønske om fred og forståelse i verden gjennom radio og utvikling.

Ole Kr. Aamot (BS EE), Oslo, 1. juni 2020

Submitted in coherence with Norwegian Education Law

This dissertation fullfills the requirements of the Bachelor of Science degree in Electrical Engineering at Oslo Metropolitan University, in coordination with the status of the Bachelor of Science program at University of Oslo, Faculty of Mathematics and Natural Sciences, Norway.

Typeset in LaTeX.

Forord

Opphavet til offentlig radio ble første gang uttrykt og forklart av Edwin Howard Armstrong i kjelleren i Philosophy Hall ved Columbia University i New York City. Justin Dove (PhD) ved Massachusetts Institute of Technology er en av de som anbefalte meg å besøke Columbia University i New York City da jeg besøkte Cambridge, MA i juni 2014. Jeg besøkte senere University of Berkeley California i august 2015 hvor jeg møtte Greg Thomas (PhD) som studerte ved Columbia University og besøkte Googleplex, Menlo Park, Hewlett Packard Garage og Stanford University.

Denne oppgaven er derfor inspirert av Justin Dove (PhD) og Greg Thomas (PhD), mine to amerikanske venner fra østkysten og vestkysten av U.S.A. og Peter Norvig (Director of Research, Google, Inc.) som jeg møtte i Mountain View i august 2015.

U.S.A. går gjennom en tung tid når denne oppgaven skrives og radio kan kanskje hjelpe litt med å stabilisere det amerikanske samfunnet.

Radio har reddet verden tidligere. Radioreportere som amerikaneren Edward R. Murrow fra London under blitz-bombingen av London, nordmannen Toralv Øksnevad, "stemmen fra London', ankermann i BBCs sendinger fra Bush House i London på norsk mellom januar 1941 og mai 1945 under Nazi-Tysklands okkupasjon av Norge mellom 9. april 1940 og 8. mai 1945, Gunnar Nygaards norske sendinger fra USA fra september 1940, og rapportene til amerikaneren George Putnam på amerikanske skip på vei mot Normandy og D-dagen 6. juni 1944, den hemmelige, allierte invasjonen av Europa i Frankrike holdt oss informert.

Under 2. verdenskrig og senere har radio tjent offentligheten ved å skape en offentlig mening og adressert problemer som krever offentlighetens lys.

Radio som medium har hatt betydning for å informere mennesker om protester, konflikter mellom land og internasjonale kampanjer for å slippe fri fengslede menneskerettighetsforkjempere og styrte diktatorer i land uten reell ytringsfrihet.

Menneskerettighetsorganisasjonen Amnesty International har i flere land sendt "distress signal"-kampanjer på FM-båndet for å opplyse allmennheten og frigi mennesker som sitter i fengsel uten lov og dom og uten tilgang til advokat.

I enkelte land driver myndighetene i diktaturer med såkalt "jamming" av radiosignaler på FM-båndet for å hindre mennesker fra å lytte til frie radiosendinger. Internett-radio er ikke sårbart for "jamming" av signalene.

Radio som teknologi har utvidet horisonten til mange mennesker siden den irsk-italienske oppfinneren Guglielmo Marconi kom til U.S.A. i 1899 for å vise hvordan den trådløse telegrafen kunne sende Morse-koder uten kabler.

Internett ble oppfunnet av ARPA i 1969 og utbredelsen har vært en viktig faktor i demokratibevegelser i hele verden siden World Wide Web (WWW) ble oppfunnet i 1989 av Tim Berners-Lee og Cascading Style Sheets (CSS) ble foreslått av nordmannen Håkon Wium Lie 10. oktober 1994 ved CERN i Sveits og utviklet som Internett-standardene HTTP, HTML og CSS.

Senere ble Internett-radio oppfunnet i forbindelse med arbeid med gode komprimeringsalgoritmer for lyd som MP3, Advanced Audio Codec (AAC), Ogg Vorbis og FLAC.

Internett-radio gjorde det mulig å kringkaste lyd med høy lydkvalitet.

Denne Bachelor-oppgaven beskriver arbeidet med å kartlegge frie radiostasjoner som kringkaster i verden og programmere et grafisk kart basert på det grafiske

grensesnittet GTK+ og lydstrøm via HTTP over et TCP/IP datanettverk mellom minst to datamaskiner som konverterer lydbølger til digitale bits, og digitale bits til lydbølger.

Denne Bachelor-oppgaven representerer 18 år med arbeid for å kartlegge frie radiostasjoner og 3 år med programmeringsarbeid for å presentere frie radiostasjoner på et grafisk kart presentert på en datamaskin med det grafiske brukergrensesnittet GTK+/GNOME og kontinuerlig lydstreaming over et datanettverk mellom minst to datamaskiner som er kapable til å konvertere digitale bits med lydbølger og stillhet til akustiske lydbølger i et stereoheadsett eller monohøytalere.

Arbeidet med å finne radiostasjoner som sender direkte radio har tatt mest tid. De fleste radiostasjonene er tilknyttet universiteter og høgskoler, noen av radiostasjonene er kommersielle, mens andre er offisielle kringkastingskanaler som National Public Radio (NPR) i U.S.A., British Broadcasting Corporation (BBC) i Storbritannia, Radio Eins (RDS) i Tyskland, Danmarks Radio (DR) i Danmark, Sveriges Radio (SR) i Sverige og Norsk rikskringkasting (NRK) i Norge. Mange av stasjonene i U.S.A. har reklame foran sendingen når en begynner å lytte. Dette gjelder f.eks. American University Radio (WAMU) som har vært den viktigste National Public Radio (NPR) medlemsstasjonen for Washington siden 2007 og Hawaii Public Radio i U.S.A. som kringkaster National Public Radio (www.npr.org) i U.S.A. som ble startet 26. februar 1970 i Washington, D.C., i U.S.A. I 1979 startet lobbyorganisasjonen National Cable Television Association i Washington, D.C. den amerikanske radiostasjonen C-SPAN (Cable Satellite Public Affairs Network - `http://www.c-span.org/`), som kringkaster forhandlingene i Kongressen (United States House of Representatives) og Senatet (United States Senate) i Washington, District of Columbia, U.S.A. På tidspunktet da dette ble skrevet foregikk det en riksrettssak mot President Donald J. Trump som først var diskutert i Kongressen og deretter fremmet i Senatet og en virus-pandemi kjent som Korona/COVID-19 som har rammet mennesker i hele verden.

Prosjektet

Dataskrivebordet GNU Network Object Model Environment (GNOME) ble startet av Miguel de Icaza, Federico Mena og Elliot Lee i august 1997 basert på Gimp Tool Kit (GTK+) fra UC Berkeley i U.S.A. og videreutviklet av Red Hat Software, Inc. i Red Hat Development Labs og Eazel, Inc. GNOME Foundation som ble etablert under GUADEC 2000 ved ENST på Telecom France i Paris 17. mars 2000 og har støttet utviklingen av GNOME og sosiale møter mellom GNOME-utviklere på konferansen GUADEC i 19 byer.

Arbeidet med programvaren GNOME Internet Radio Locator begynte ved Norsk Regnesentral i 2002. Utviklingen ble startet igjen 1. november 2014 med publisering av versjon 0.1.0 og fortsatte i 2015, 2016 og 2017 med tekstsøk og radiostasjoner i XML-fila `http://www.gnome.org/%7eole/gnome-internet-radio-locator/gnome-internet-radio-locator.xml` og direktestrømming av radiostasjoner ved hjelp av GStreamer, publisering av 1.0.0 16. september 2018, 2.0.0 20. februar 2019 og 3.0.0 24. januar 2020.

En ny variant (gnome-internet-radio-locator) med omskrivning for GNOME 3 begynte i 2017 med støtte for OpenStreetMap og kartmarkører. En ny omskrivning for GTK+ 4 begynte i 2018.

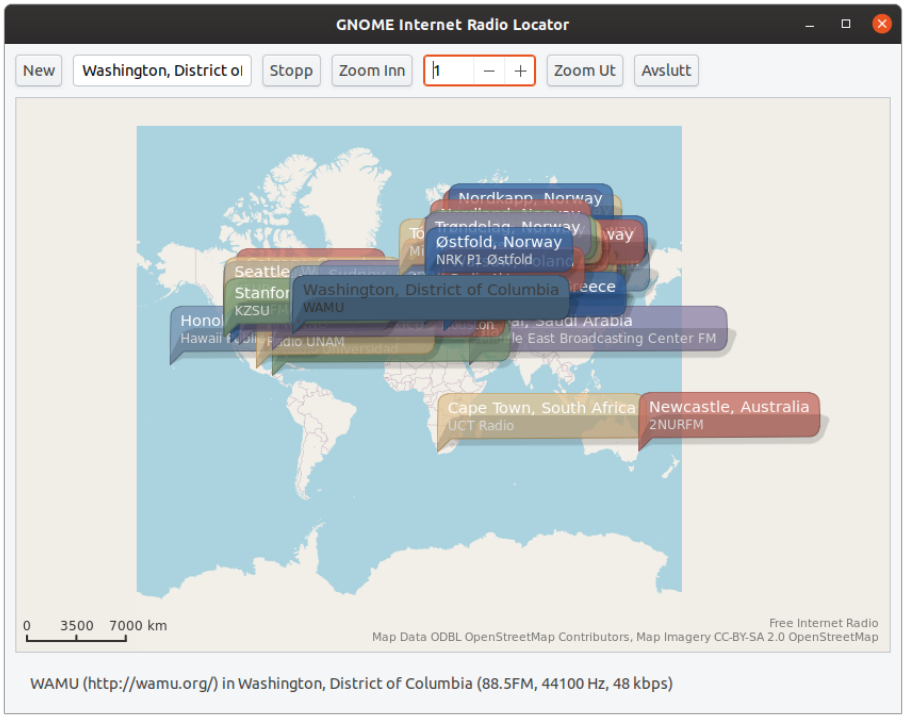
I 2019 la jeg til farger på markørene og publiserte versjon 2.0.4 av gnome-internet-radio-locator som del av GNOME-prosjektet 24. mai 2019.

Jeg publiserte gnome-radio versjon 0.2.0 10. september 2019 og gnome-internet-radio-locator versjon 3.0.1 24. januar 2020.

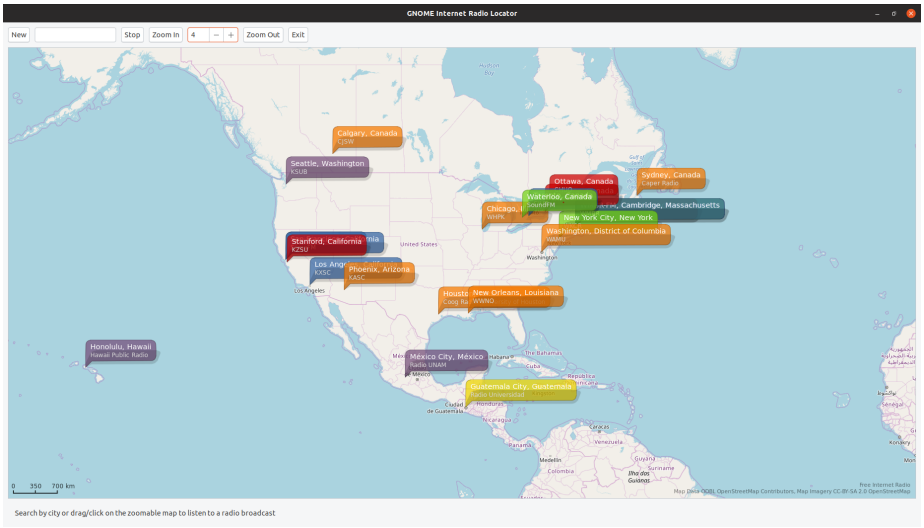
Programmet støtter 124 radiostasjoner i 97 byer i Australia, Sør-Afrika, Saudi Arabia, Catalonia, Belgia, Italia, Frankrike, Storbritannia, Dublin, Irland, Skottland, Færøyene, Tyskland, Danmark, Sverige, Finland, Russland, Mexico, Guatemala, Canada, U.S.A. og Norge.

I Norge er de lokale radiostasjonene til NRK P1 i Trøndelag, Nordland, Finnmark, Innlandet, Buskerud, Rogaland, Møre og Romsdal, Sogn og Fjordane, Hordaland, Sørlandet og Østfold oppmarkert på kartet over Norge.

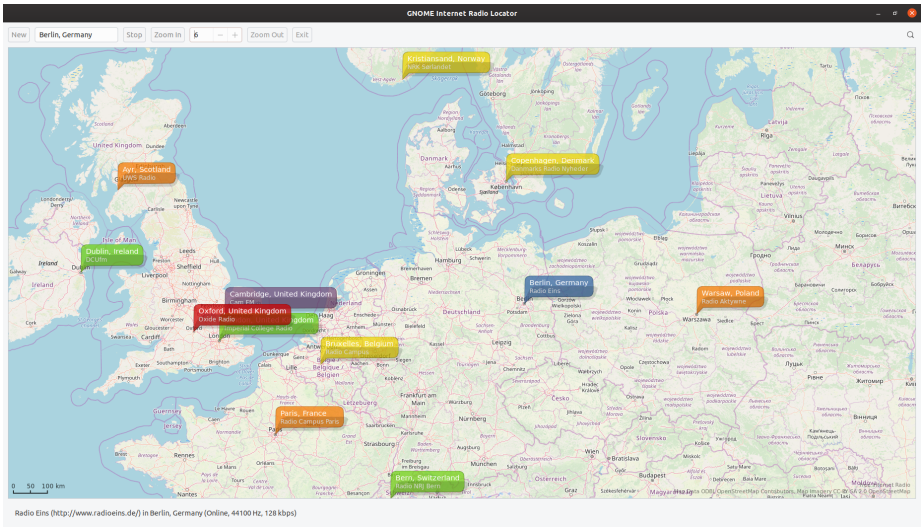
Internasjonale radiostasjoner er oppmarkert på kartet i resten av verden.



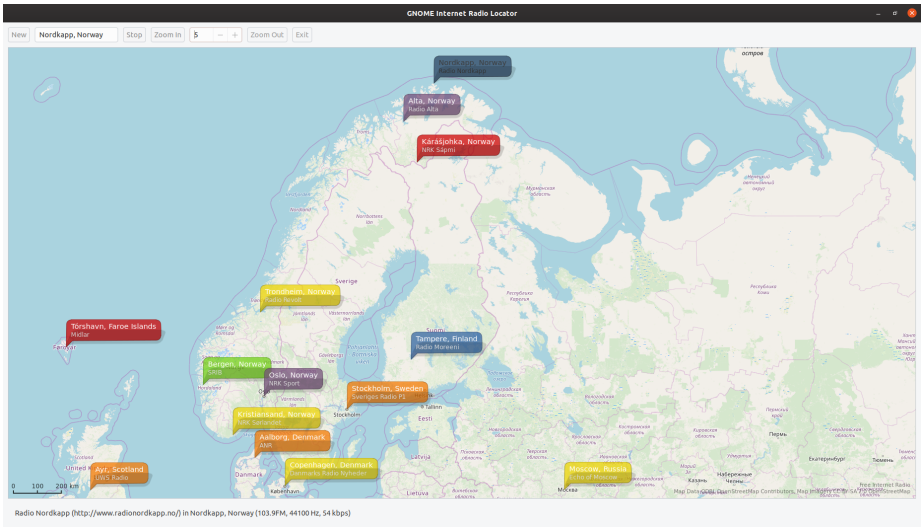
Saudi Arabia and South Africa



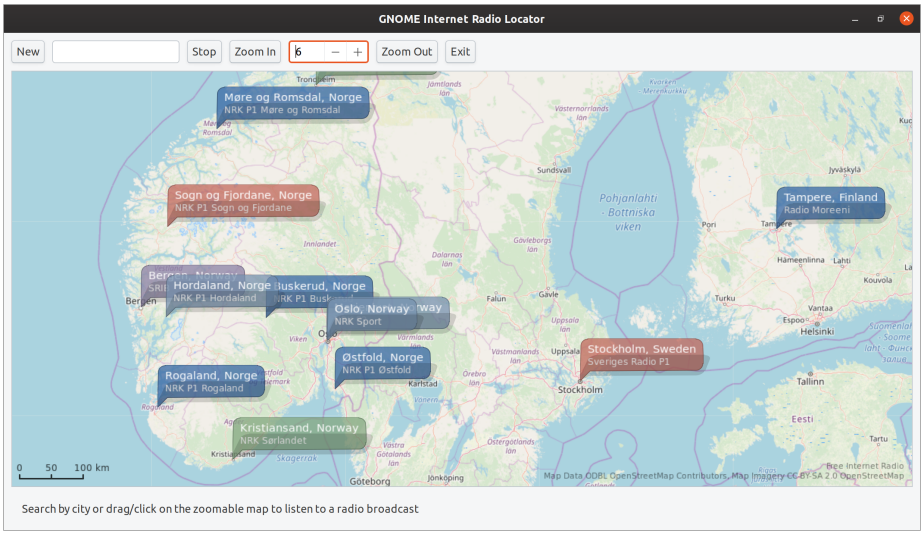
United States of America, Mexico, Guatemala and Canada



Norway, Denmark, Sweden, Finland, Scotland, Ireland, United Kingdom, Poland, Germany, France, Italy, Russia

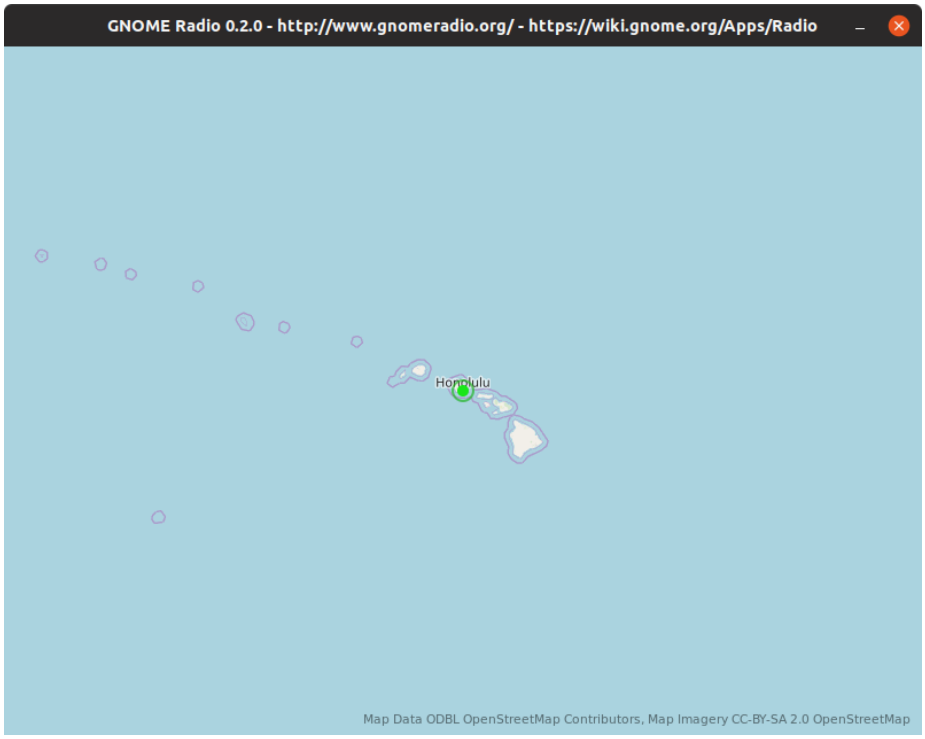


Faroe Islands, Scotland, Norway, Denmark, Sweden, Finland, Russia

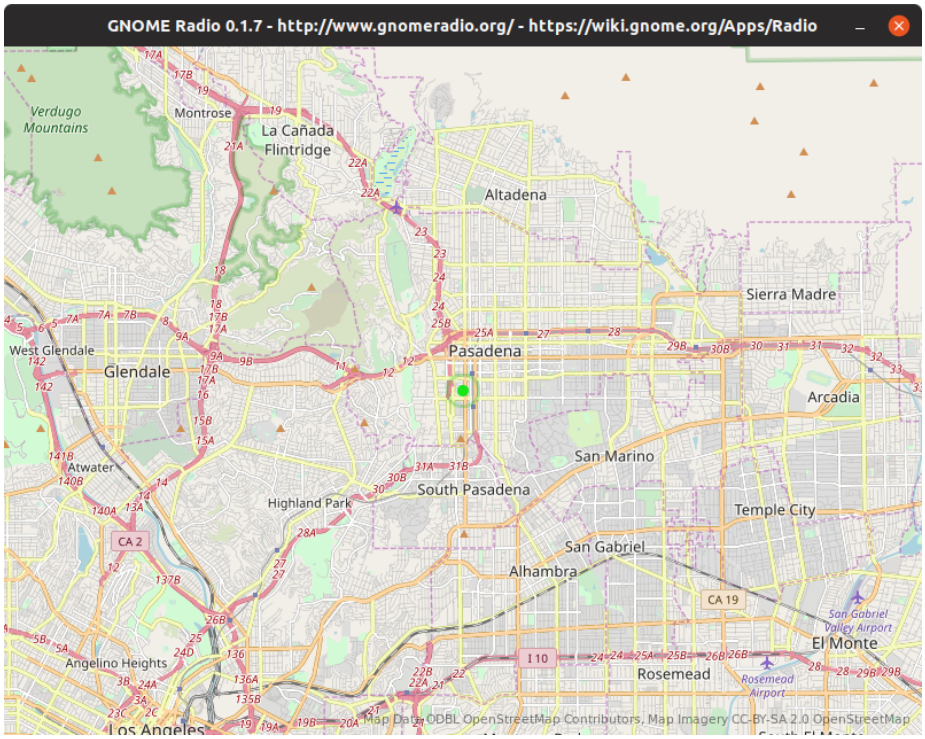


Buskerud, Rogaland, Møre og Romsdal, Sogn og Fjordane, Kristiansand, Hordaland, Østfold

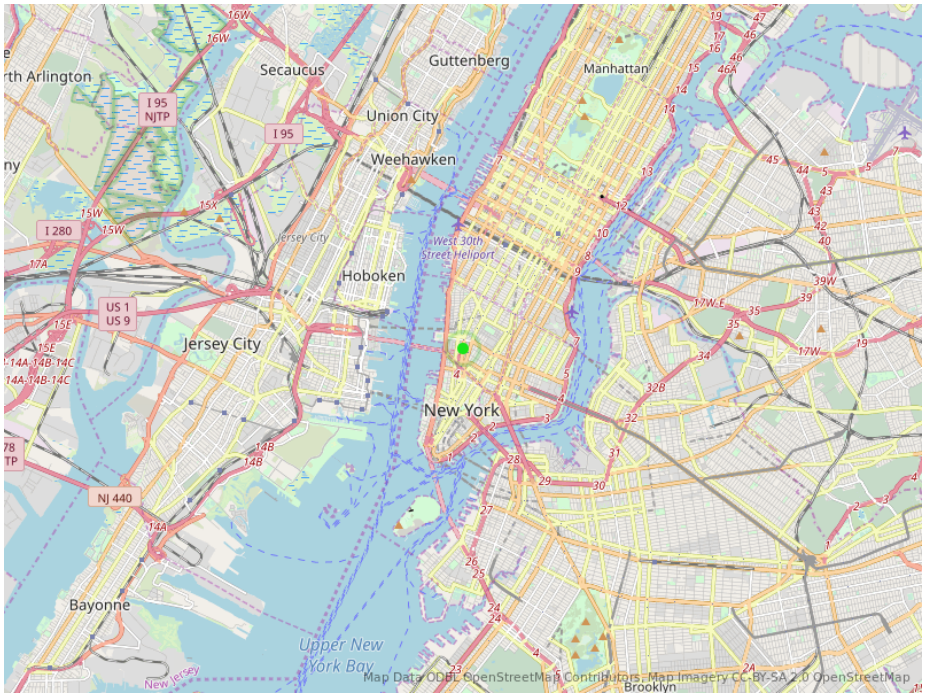
Screenshot of gnome-internet-radio-locator 2.4.0



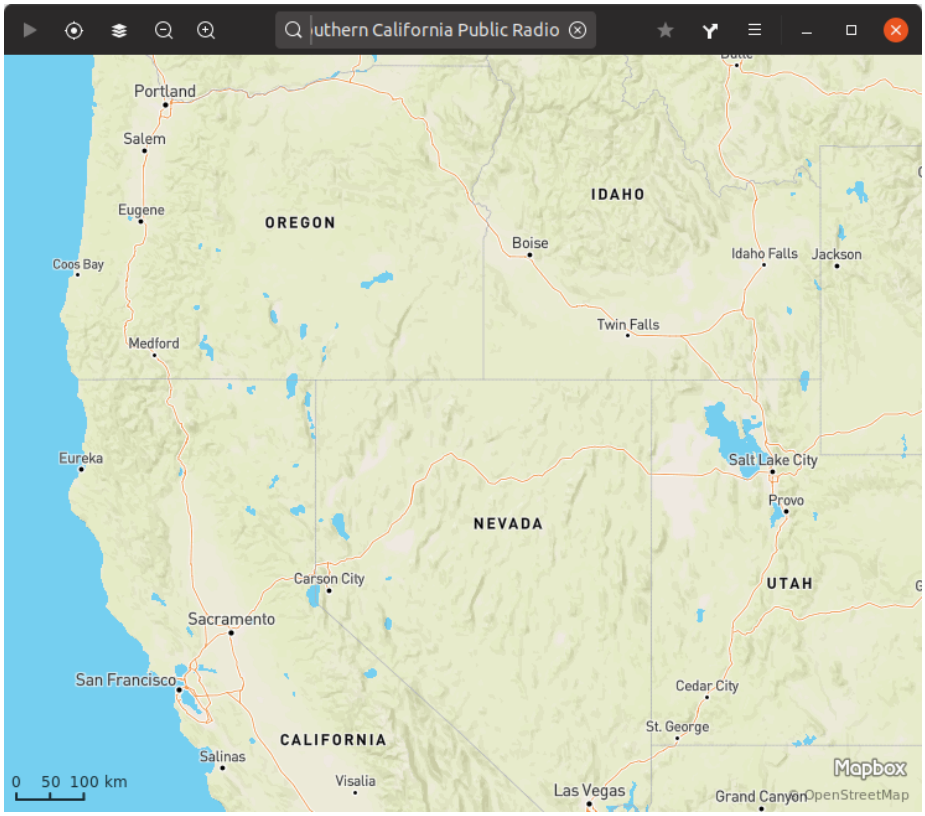
Screenshot of gnome-radio 0.2.0 (Hawaii Public Radio)



Screenshot of gnome-radio 0.1.7 (Southern California Public Radio)



Screenshot of gnome-radio 0.1.5 (New York Public Radio)



Screenshot of gnome-radio 0.1.6 (MapBox)

Arbeidet vil fullføres innen 30. juni 2020.

Contents

Topics

Part I

Electricity

Chapter 1

Electrical Engineering

A. P. Trotter, a leading British electrical engineer, wrote that "electrical engineering was born at the Paris Exhibition" (1889), one of the first long distance radio transmissions happened from the Eiffel tower in Paris from the Paris Exibition in 1889 on January 12, 1908.

As it is shown in this thesis, that primarily documents the computer software implementation in the computer programming language ANSI C of the Free Software radio software packages GNOME Internet Radio Locator (gnome-internet-radio-locator) and GNOME Radio (gnome-radio) implemented and published in this paper, it was possible to visualize some of the modern age Internet radio broadcasters on a world map about 109 years later in 2017 with the help of the free software libraries GTK+/GNOME, GStreamer and libchamplain in the GNOME Radio project (`http://www.gnomeradio.org/`) and automatically begin the reception of a live radio broadcast on a personal computer running the free computer operating systems Debian GNU/Linux, Fedora Core and Ubuntu Linux from free radio stations broadcasting around the world over the Internet.

The work continues in the GNOME Radio project after this thesis was summoned on January 24, 2020, with the release of GNOME Internet Radio Locator version 3.0.0.

Visit `http://www.gnomeradio.org/` for details on how to install software for visualizing the location on a world map and begin the reception of a live radio broadcast on Free Software systems such as GNU/Linux.

Part II

Radio

Chapter 2

American, British, Norwegian, and European Radio History

According to The Oxford Companion to American History, radio was theoretically foreseen in 1865 by the Scottish physicist James Clerk Maxwell. His electromagnetic equations led to experiments by the German physicist Heinrich Hertz, who, by using an electrical discharge, produced the 'hertzian,' or radio, waves (1886). His work was in turn advanced before 1900 by the research of the Italian physicist Guglielmo Marconi, and by the electrical engineer John Fleming, who perceived that a specially constructed light bulb ('diode') could serve to detect radio signals. The related invention in 1906 of the three-electrode vacuum tube by Lee de Forest made radio practicable by its ability to amplify weak signals.

The first successful telegraph was demonstrated by Samuel Finley Breese Morse, with a network linking Washington DC to Baltimore in the United States of America with 35 miles of copper cable. His first transmitted telegraphic message on May 24, 1844 was: "What hath God wrought", transmitted in Morse code on line from Washington, D.C., to Baltimore, according to the books "Principles of Digital Audio" by Ken C. Pohlmann published by McGraw-Hill in 2000 and Webster's Guide To American History published by G. & C. Merriam Company, Publishers in Springfield, Massachusetts, U.S.A. in 1971. Though construction of line was financed by the government, Morse failed to persuade it to buy rights to his invention and formed his own company. His partner Alfred Vail invented telegraphic printer in 1844.

On February 14, 1876, Alexander Graham Bell, a successful teacher of the deaf, filed for a patent on a magnetoelectric telephone device where the human voice could be transmitted in a two-way system. He demonstrated his invention on March 10, 1876 at the Philadelphia Centennial Exhibition of 1876 and attacted considerable attention. The first telephone system, with 21 subscribers, was established two years later in New Haven, Connecticut.

Thomas Alva Edison produced the initial audio wave recording on a sheet of tin foil wrapped around a cylinder that can be turned on December 6, 1877 of him reciting "Mary Had a Little Lamb".

The first telephone communication between cities opened in 1877 between Salem, Massachusetts, and Boston, and Chicago and Milwaukee. By 1880 almost 150 separate telephone companies were operating 34,000 miles of lines.

The successful development of telegraph and telephone in the United States led to near-monopolies by Western Union (telegraph) and American Telephone and Telegraph Company (AT&T) (telephone). At the same time, as electricity was used more after 1880, companies appeared that manufactured electric lights, electric motors and the like. The most important electrical manufacturing firms were Westinghouse, started in 1886, which brought the alternating current power system to the United States, and General Electric (GE), formed in 1892 as an amalgamation of two older firms, including Thomas Alva Edison's. After several years of competition and patent arguments, GE and Westinghouse agreed in 1896 that GE should receive two-thirds of the business growing from their shared patents. This early patent "pool" agreement was an important precedent for the radio manufacturing industry. Another important firm was Western Electric, which specialized in telephone communications equipment and was taken over by American Telephone and Telegraph Company in 1881.

The first important armed-forces tests of radio were conducted independently by the United States Navy and the British Royal Navy in 1899. The American tests were made of Marconi apparatus aboard the U.S. battleships New York and Massachusetts. Although radio signals bridged distances of up to 40 miles, it was clear that the lack of suitable tuning devices led to unacceptable interference.

Radio clearly was such a potential saver of lives and property in peril at sea that otherwise practices that interfered with its maritime operation had to be overcome.

By 1909 the Bell system was so successful that the American Telephone and Telegraph Company (AT&T) was able to purchase Western Union - only to have to resell it in 1914 because of the antitrust laws.

AT&T was split into several parts in 1884. Bell's basic patents expired in 1894.

The Radio Act of 1912 established several key legal and regulatory principles that continue to undergird broadcast regulation

Radio broadcasting was a new electrical communications concept. Radio removed our dependence on wires, and finally "broadcasting" presented a new concept.

Wireless, or radio, is a logical extension of wired telegraphy and telephony. Wires are easily broken and hard to string between distant communities or over physical obstacles. With wireless techniques, communication could take place as rapidly as with wired devices but did not require a physical connection. Distant locations could be contacted quickly, relaying might be unnecessary, and ships could keep in touch with land. The penalty for this was that a radio message would go out in all directions at once and could be picked up by anyone who cared to listen.

The Radio Act of 1912 established several key legal and regulatory principles that continued to undergird broadcast regulation more than three quarters of a century later. In the beginning the federal government would control broadcasting. No one could broadcast without a license from the Secretary of Commerce and Labor. The 1912 Act declared that broadcasting without a license was illegal and that only the government could convey a license.

During World War I, the Navy was able to gain control of all radio stations in the United States except those operated by the Army.

The Radio Corporation of America came into being 17 October 1919.

In March 1919, only a few months after the end of the First World War, the voice of a British wireless operator was heard nearly three thousand miles away across the Atlantic in the United States. This was probably the first external broadcast ever made from the British Isles - the transmitter was in Ballyunion in Ireland - and was look on at the time as uncanny, according to the book "A Skyful of Freedom - 60 Years of the BBC World Service" by Andrew Walker published by Broadside Books Limited in 1992.

Once the wartime restrictions on transmission were lifted in September 1919, the amateur broadcasters were back in force, and because so many had served in the war, they were familiar with the latest technological advances.

Radio remained in its experimental stage until improvements, especially those by E. F. W. Alexanderson, who opened the way for a primitive form of broadcasting in 1920. Soon thereafter notable advances made possible the growth of broadcasting chains and the development of international transmissions.

Following the broadcast of the 1921 World Series between the Yankees and the Giants on WJZ, broadcasting as we know it took off.

In late March 1923 Hoover called a second National Radio Conference.

In 1924 there was 2,500,000 radios in the United States, as opposed to 5,000 sets in 1920.

On November 30, 1924 Radio Corporation of America (RCA) demonstrated wireless telegraph transmission of photographs from New York to London.

The first loudspeaker was developed in 1925 by Rice and Kellog shortly after the valve amplifier.

The British Broadcasting Corporation was established on 1 January 1927. It was the direct successor of the British Broadcasting Company which had been formed in 1922 as a consortium of manufacturers of domestic wireless receiving sets. The newly-established body was now set up as a public corporation under Royal Charter, deriving its authority from the King in Council. This constitutional device was adopted in order to distance the corporation from the government and from parliament, although ultimately it was to be responsible to them. Designed to provide public service broadcasting of a high quality the BBC enjoyed a monopoly until 1955, when commercial broadcasting was introduced. Initially, the BBC was put under the control of the Postmaster General, but in 1974, broadcasting matters were transferred to the Home Office, and the Home Secretary became the minister responsible for broadcasting. As the minister in charge of the BBC at the time of its establishment, the Postmaster General assumed the task of framing its legal basis. And, despite the fact that the BBC was meant to be distanced from the government, he provided himself with overriding formal powers over transmissions and the content of programmes: among other things, he could require the BBC to broadcast or to refrain from broadcasting any matter as he saw fit. (Etzioni-Halevy 1988)

In the United States of America, National Broadcasting Company was organized as first nationwide radio broadcasting network in 1926; National Broadcasting Company (NBC) was established in 1926, Columbia Broadcasting System (CBS) was organized in 1927 and Mutual Broadcasting System (MBS) was founded in 1934.

Federal Radio Commision was created in 1927. The Radio Act of 1927 enacted ideas that had been in the legislative hopper since the first National Radio Conference.

The two key aspects of the Radio Act of 1927 are the declaration that there could be no private ownership in the entire spectrum and the related decision that users of the spectrum would be licensed under the public interest standard. The new Radio Act put first things first. Although the 1912 Act had required a license to use the air, it had been silent on the issue of ownership of the airwaves. The 1927 Act was not. It bluntly declared that there could be no private ownership of the airwaves; they were public and use could occur only with the government's permission. That permission, in the form of a license, would be granted without charge, but for no more than three years.

The first audio broadcasts used amplitude modulation (AM) of the carrier wave in the MF and HF bands. According to the book "The Art of Sound Reproduction" written by John Watkinson and published by Focal Press in 1998, AM broadcasts are prone to interference and do not reproduce the whole frequency range, but AM works for long distance broadcasts. Shortly before the Second World War frequency modulation (FM) was developed.

Edwin H. Armstrong, originally backed by Radio Corporation America, developed a system for transmitting radio based on modulation in the frequency (hence FM, frequency modulation) instead of the amplitude (AM, amplitude modulation) of the radio wave. FM can transmit anything the human ear can hear and more. AM has the advantage that it can transmit farther than FM, but that is its only advantage.

Armstrong was believed to have conceived and invented wideband frequency modulation (FM) on September 1931, while his patent was granted on December 26, 1933.

In November 1934 Federal Communications Commission (FCC) replaced Federal Radio Commission. It is authorized to control all foreign and interstate radio, telegraph, and cable communications.

According to Section 3(0) of the Communications Act of 1934, broadcasting is "the dissemination of radio communications" intended to be received by the public, directly or by means of intermediary relay stations.

We should establish here that broadcasting signifies transmission of music, speech, and/or pictures in forms that the general public can understand, on a regular and announced schedule, on a frequency band for which the general public has receivers, by a station licensed by the government for that purpose (if licensing was then required).

The public interest was a bedrock requirement for the issuing of a radio station license from the very beginning. It is a paramount philosophy of the Communications Act of 1934.

By the late 1930s, radio was woven into the fabric of American life. Public events, from political rallies to sporting events and vaudeville routines, were now enjoyed by millions in private.

The invention in 1939 of a new form of radio transmission and reception, frequency modulation (FM), by E. H. Armstrong, eliminated static and improved fidelity.

In the beginning the Federal Communications Commission (FCC) granted radio station licenses and required radio stations to classify programming into

a minimum of seven categories: Entertainment, Religion, Agricultural, Educational, News, Discussion, and Talks.

Increasingly, Americans got their news from radio, especially news of the expanding war in Europe.

The immediacy and drama of the war news tied people more intimately to unfolding events, it also, apparently, put some on edge.

During World War II listening to radio from the BBC became very important in Norway, where the BBC at first had problems of credibility because of its overoptimism during the military campaign of 1940. It was accused of having put out "fake news" - "just another form of propaganda" much like the U.S. President Donald J. Trump accused the media of in the beginning of his presidency.

The Norwegian democratically elected government and King Haakon escaped to London on April 9th, 1940 and so did several members of the broadcasting service. They were eventually seconded to the BBC at their own request and worked so successfully with their British colleagues that by 1941 a Norwegian who escaped to the United States was reporting: "Nobody reads any news papers or listens to any radio except the BBC."

After Denmark was occupied by Nazi Germany in 1940 a Danish political leader who had escaped to Britain broadcast a call for action over the Danish service when it did report the heroism shown in other occupied countries in 1942. From then on Danes regarded themselves at war and the BBC was with them.

The North American service gradually grew until by 1942 it was on the air for more than seven hours a day. By that time the United States was in the war, and there were programmes linking the American and Canadian forces in Britain with home. American programmes were transmitted to Britain via a BBC studio in New York.

On D-Day, the invasion of Normandy, no fewer than 725 out of 914 radio stations in the U.S.A. carried BBC War Reports. In the first half of 1945 regular rebroadcasting reached a peak. It was estimated that over 15 million people in the States were hearing one or more BBC programmes a week as the war in Europe drew to a close. (Walker 1992)

By 1950 92% of households had radios, and the direction of industry expansion was in the range of offerings (stations) and in the encouraging the ownership of more than one radio. In contract, at that time only 62% of households had a telephone. Over the next 38 years, the role of radio changed, but did not diminish.

From In 1957 FM accounted for only 2 percent of radio ad sales, but increas ed to 15 percent by 1965.

Corporation for Public Broadcasting is a private, nonprofit corporation created by Congress in the Public Broadcasting Act of 1967.

In 1969 the Public Broadcasting System (PBS) was established, which received some federal money to support noncommercial and educational programs. But according to The Reader's Companion to American History, PBS must still rely on viewer support and corporate sponsorship to survive.

In 1970 with the help from Corporation for Public Broadcasting, National Public Radio (NPR), America's first National Public Radio Network was established.

By 1988, the average household had six radios, and four out of five adults listened to the radio every weekday.

Delivery of digital audio files over the Internet had became possible in the early 1990s, soon after the introduction of the World Wide Web and browsers, partly thanks to the new and efficient audio coding system of mp3 (MPEG-1 Audio Layer III), developed at the German Fraunhofer Institute between 1988-1992 simultaneously with audio coding for DAB (Fraunhofer 2009).

First attempts to stream audio over the Internet were also made by that time, and after the introduction of the essential computer software such as Real Audio, streaming started to gain popularity (Priestman 2002; Menduni 2007).

Earlier when one had to first download the audio file and then listen, the stream player made it possible to listen to the audio while also receiving it. The sound quality was not very high at first, but the number of radio stations offering audio streaming services over the Internet grew rapidly in the 1990s, especially in the United States (Lax and Ala-Fossi 2008).

Digital radio is one of the most exciting developments ever in radio. The crystal clear sound, utterly silent background and interference free reception delivers a new level of performance from broadcast sources.

The New York Times welcomed the new digital revolution as the "biggest technological leap" since FM technology was developed in the 1940s and 50s', offering the same "high quality" of sound – free of static and hiss. (O'Neill 2010).

The experience of a digital broadcast of a live symphony orchestra concert is astonishing.

Chapter 3

Sound waves

On radio you can listen to immediate sound waves in the broadcasts.

Longitudinal waves in a medium, usually in air, are called sound waves.

The most general dedinition of sound is a longitudinal wave in a medium.

The wave function for a sound as a longitudinal wave $y(x,t)$ can be described as the interval t with amplitude A with the cosinus function, if the wave is sinusoidal.

$$y(x,t) = Acos(kx - \omega t) \tag{3.1}$$

In the longitudinal wave the displacements are parallel to the directional of travel of the wave, so distances x and y are measuree parallel to each other, not perpendicular as in a transverse wave. The amplitude A is the maximum displacement of a particle in the medium from its equilibrium position.

Sound waves from radio in air are compressed and decompressed pressure fluctuations between the amplitude device and human ears. This field is further described in biology or physics and is not a subject in electrical engineering, but we know that the human ear is sensitive to waves in the frequency range from about 20 to 20,000 Hz, the audible range, that can be reproduced on electronical devices as the pressure differences can be sensed on microphones and simular devices.

For audio in uncompressed form (16-bit linear PCM) with a sample frequency of 48kHz the data transferred is equal to 48000 Hertz that equals 1536000 bits per second or 192 kbits per second.

Waves of shorter wavelength Λ (larger wave number $k = \frac{2\Pi}{\Lambda}$ have greater pressure variations for a given displacement amplitude because the maxima and minima are squeezed closer together.

In electrical engineering an audio receptor is a microphone, while an audio transmitter is a loudspeaker or a headset. The speed of sound in hot air (20 C) is 343 meters/second. The sound wave falls to 0 dB after the initial signal-to-air curve.

With Internet radio there is very little or no delay in sound waves in the broadcasts between the listener and the presenter or musician in a live radio broadcast.

Chapter 4

Electrical fields

Electrical fields are based and formed when a wire is ignited with power from a power generator such as a dynamo, turbin generator or chemical fluid in a battery such as 9 Volt batteries from Duracell, Energizer or VARTA.

The duration of the electrical field is determined by the flow of electrical current and the flow of electrons from negative to positive charge measured in Coloumb.

Chapter 5

Electrical systems

An electrical system consists of a wire, a circuit, a power source and a capacitor.

There are three important kinds of electrical transmissions: conduction, induction and radiation.

5.1 Conduction

Conduction means the sending of impulses through a medium capable of transmitting electricity - a wire, salt water, or the earth.

5.2 Induction

Induction refers to the appearance of a current in one circuit when it is placed near another, already charged, circuit, without a physical connection. Induction can cause cross-talk on a telephone circuit, and induction coils permit the recording of telephone conversations without wire hookups.

5.3 Radiation

Radiation means the generation of electromagnetic waves, generally sent out from an antenna. The radio transmissions of today use this last method, which became practical just before 1900.

The major findings of Scottish mathematician and physicist James Clerk Maxwell, published in 1864, suggested that a signal could be sent out electromagnetically that would be completely detached from the point of origin. Using mathematical equations, he demonstrated that electricity, light and heat are essentially the same and that all radiate at the same speed in free space.

German physicist Heinrich Hertz demonstrated the correctness of Clerk Maxwell's theories in a series of experiments in 1887 and 1888. The fundamental unit of frequency, the Hertz (Hz), is named for him. Hertz measured the speed of electromagnetic radiation (the speed of light), the length of various waves, and similar parameters but did not promote the use of wireless for communication.

In 1894, at the age of 20, Guglielmo Marconi read of Hertz's experiments and aimed to apply this knowledge to communication. By 1896 he could transmit and receive two miles or more on his father's estate near Bologna.

On Christmas Eve 1906, Reginald Fessenden (1866-1932), the inventor of amplitude modulation (AM), transmitted a program of speech and music from Brant's Rock, Massachusetts, in what was probably the first radio broadcast in the United States of America.

Reginald A. Fessenden became a professor of electrical engineering at the University of Pittsburg after having worked for Thomas Alva Edison and with the U.S. Weather Bureau on a system of wireless transmission of forecasts. He wanted to develop a workable system of transoceanic wireless using continous waves rather than Marconi's spark gap technique. Fessenden believed that this method would provide the power necessary for more effective Morse code transmissions and simultaneously create the quieter carrier wave required for voice transmission.

Part III

Audio

Chapter 6

Digital Audio

Digital audio systems spread in the United Kingdom, when BBC research engineers developed a system for combining the television sound signal, known as send in syncs, still used by the BBC and independent UK television companies, according to Michael Talbot-Smith's book "Sound Engineering explained" published by Focal Press in 2002.

In the early 1970s, BBC radio adopted digital audio.

According to Harry Nyquist, the sampling frequency R to reproduce the bandwidth B must be twice as large as the bandwidth B in cycles per second.

$$2R > B \tag{6.1}$$

Nyquist's Law, named in 1933 after scientist Harry Nyquist, states that a sound must be sampled at least twice its highest analog frequency in order to extract all of the information from the bandwidth and accurately represent the original acoustic energy. Sampling at slightly more than twice the frequency will make up for imprecisions in filters and other components used for the conversion.

For example, human hearing ranges from 20Hz to 20,000Hz, so to imprint sound to a CD, the frequency must be sampled at a rate of 40,000Hz to reproduce the 20,000Hz signal. The CD standard is to sample 44,100 times per second, or 44.1 kHz.

The compact discs - the CD - used the principles of digital audio and was released in 1982.

With digital audio, the original analogue signal is converted into a code of much more robust signals. These take the form of pulses of voltage of standard height and duration.

It is possible to reconstruct the height and duration of the pulse as digital audio. The analogue signal is split up into small sample, the 'height' is known as the amplitude which is measured in the next stage. Sampling in CDs and modern digital audio systems is done with a rate of just over 44000 times a second. Each sample has to be measured, known as quantizing, the process of measuring the amplitude of each sample.

Good quality digital audio needs at least 65000 quantizing levels, each measurement is a number between 0 and 65000!

Digital audio systems store numbers thst can be anything up to 65000 and do it at a rate of just over 44000 each second, stored in binary arithmetic of only two digits, 1 and 0.

Chapter 7

Solid state recording

1 second of stereo sound contains about 1.4 megabits without taking into account bits for timing, error detection and correction. 2 megabits per second is a reasonable bare minimum, so a minute of good stereo audio needs about 14 or 15 megabytes as the unit of memory storage in computers is the byte, equivalent to 8 bits.

Chapter 8

Audio compression

Compression of digital audio means reducing the number of bits in an audio sample without a serious reduction in quality.

It's possible by fairly recent development in digital audio. There are many 'sounds' that we cannot hear: frequencies in the region of 30 Hz unless at a high level, for example.

The normal ear is about 70 dB less sensitive to 30 Hz than it is at around 3 kHz and not particularly sensitive to the higher audio frequenciew above about 3 kHz.

Chapter 9

MPEG 1.0 Audio Layer III, Ogg Vorbis and MPEG-4 AAC

By running heavy audio compression algorithms in MPEG-1 Layer III (MP3), Ogg Vorbis and MPEG-4 Advanced Audio Codec (AAC), computers can store recorded audio from solid state recording for considerable durations on to a solid state storage card or in random access memory.

Chapter 10

Streaming audio

The combination of digital audio, solid state recording, data compression and MPEG 2.0 Audio Layer III results in compressed audio streaming in the free and public internet radio client software gnome-internet-radio-locator documented in this thesis.

Part IV

Computers

Chapter 11

Computer systems

A computer system is a digital machine for computing numbers and performing binary arithmetic of the digits 1 and 0, producing printed papers, transmitting data between another computing system, visualizing graphics such as a map on a display system as described in Chapter 13 ("Graphical Display Interfaces") and resonating audio.

Chapter 12

Computer storage

Computers operate with temporary storage and solid-state storage. In this thesis the storage is temporary and the data is a stream of audio between two computers and a stream of map data as tiles. See Chapter 17 ("Mapping Systems").

Computers can store its physical location with GPS. The logical components read a stream of audio from the position of the GPS antenna and displays the public broadcasts nearby.

The saturation will bring resonnance to the public spot and give a good recording of audio from all angels and balcony spots. The playback of the audio recording will be very precise. I propose MLAR, a format for multiple-location audio recording (MLAR). MLAR, built on software from Xiph.org, will enable recording from multiple device audio for playback and entertainment, with radio locator and absolute precision (up to 10 meters).

Chapter 13

Network systems

A network system consists of three or more digital computer systems (see Section 3.1 on "Computer systems") that can connect between separated physical space and fields (see Chapter 4 on "Electrical systems").

Chapter 14

Network adressing

Network adressing in IP is based on Internet Protocol adressing of computer network adressing (physical and logical link).

Chapter 15

Internet systems (Internet Protocol)

The Internet was built upon the Internet Protocol described in RFC 781 (1981) and RFC 8098 (2017), documents known as “Request for Comments”.

Chapter 16

Domain Name Systems for Internet systems

The first Internet domain was symbolics.com introduced in 1985. The rationale behind Internet domains under .com were commercial, not in the public benefit like .org domains.

When computers were set up in networks, the Internet Protocol addresses were at first static and stored in /etc/hosts on Unix. At some point ICANN, ISC and IETF decided that the DNS configuration should be hierarchical.

Today we have a system where domain names expire 1 year after the registration date which promotes the free market dynamics, but does not solve a technical problem, because the domains stop working when the network operators don't receive the annual yearly fee for the domain from the domain registrant and competes against registrars.

The root servers on different Internet servers are hierarchical. There are indications that domains should be dynamically configured, maintained and stored forever. There should be a public archive of domain name system records with historical records.

GNOME Internet Radio Locator is such a public archive for live streams from 124 public radio station in the domain name system with locality annotation.

Chapter 17

Radio microphones

Radio microphones are widely used in film, broadcasting, theater and other industries, and it is not difficult to think of circumstances in which freedom from trailing microphone cables can be a considered advantage in all of the above.

See "Sound and recording" by Francis Rumsey and Tim McCormick published by Focal Press for a discussion and a typical 48 volt phantom powering arrangement in an electronically balanced circuit.

Chapter 18

Audio systems

Audio was previously recorded for the first time by Thomas Alva Edison on tin foil in 1878. Marconi invented long-distance transatlantic radio in 1901 and received the Nobel Prize in Physics for this work in 1909 with Karl Ferdinand Braun.

The gramophone industry was built on recorded music, for private listeners, public spots and playback on radio. A public gramophone player, the jukebox, brought music to the public and private rooms.

Radio kept playing songs on request and some still do. When the jukebox and audio amplification systems in homes became popular, radio still survived, even with the introduction of streaming audio systems in the early 2000 after the introduction of audio codecs with good compression. See Chapter 13 ("Audio codecs").

Chapter 19

Audio codecs

The supported audio codecs in GNOME Internet Radio Locator are based on the implementation of codecs in GStreamer such as MPEG, FLAC, Ogg Vorbis and Advanced Audio Codec (AAC).

MLAR, proposed in Section 3.2 on Computer storage, will store recorded audio in locations, at historical moments and scale rooms from different view points.

MLAR stands for Multi-Location Audio Recording.

Picard from MetaBrainz Foundation is a cross-platform music tagger written in Python that uses AcoustID audio fingerprints, allowing files to be identified by the actual music, even if they have no metadata.

Chapter 20

Audio fingerprinting

- Audio Fingerprinting software written in C .NET by Yvo Nelemans
  https://github.com/nelemans1971/AudioFingerprinting
- Open source audio fingerprinting of broadcast mp3Open source audio fingerprinting of broadcast mp3
  https://stackoverflow.com/questions/38566477/open-source-audio-fingerprint
- Echoprint - An open music identification service
  https://www.ee.columbia.edu/~dpwe/pubs/EllisWP11-echoprint.pdf
- Codegen for Echoprint
  https://github.com/spotify/echoprint-codegen
- Finding duplicate songs in your music collection with Echoprint
  https://musicmachinery.com/tag/echoprint/
- MusicBrainz Fingerprinting
  https://musicbrainz.org/doc/Fingerprinting
- The Echo Nest
  http://the.echonest.com/

Chapter 21

Audio response

The response of audio is vital in any audio system. Radio broadcasts important messages that sometimes are location-sensitive to the public and it is crucial in saving lives with distance between the signal and location. The distance for audio broadcasted on radio is typical limited by the power of the radio transmit in FM. With radio broadcasted over the Internet, the distance is unlimited.

Chapter 22

Visual response

The response of visual elements are essential in any graphical system. Quick visual response makes it easy to operate a computer system with a screen and a navigation device. See Chapter 22 ("Graphical User Interface systems").

Chapter 23

Graphical Display interfaces

The X Window System revolutionized the computing industry when it was introduced at Massachusetts Institute of Technology (M.I.T.) in 1984. GTK+, the GNU Image Manipulation Toolkit, announced at UC Berkeley in 1996, was the first free, LGPL widget toolkit for the X Window System and it made it possible to write Free Software programs with good graphical appearance for the X Window System in networks.

The GNOME project, a free desktop for the X Window System built on the GTK+ library, was announced in 1997 at UNAM in Mexico City. The work on GNOME Internet Radio Locator began in 2002 at Norwegian Computing Center and continued after a visit to M.I.T. in Cambridge, MA in June 2014 as I became aware of OpenStreetMap, libchamplain and GNOME Maps.

The initial 1.0 release was announced on gnome-announce-list@gnome.org and published on https://download.gnome.org/sources/gnome-internet-radio-locator/

Work on the GNOME 3 application gnome-internet-radio-locator began in 2017 and work on the GTK+ 4 application gtk-internet-radio-locator began in 2018.

Work on writing the GNOME Radio application gnome-radio from scratch began on February 17, 2019.

Chapter 24

Operating systems

The introduction of the Linux kernel by Linus Torvalds at University of Helsinki, after the delay of the GNU Hurd microkernel, written from scratch in machine assembler and published under the GNU General Public License, marked the next technical revolution in home computing and eventually saved the society from huge costs as server operating system. With Linux it was finally possible to install Free Unix on a personal computer. Alan Cox wrote large parts of the networking code and sound code at University of Wales Swansea in the early 1990s.

Mac OS X from Apple, Inc. is a proprietary operating system shipped with Macs. The stability of the kernel in Mac OS X is due to work by Jordan Hubbard from the FreeBSD project and the maintenance of the Darwin kernel took many years of work between 2000-2018. The choice of UNIX by Apple made many developers switch from Linux to Mac OS X.

Chapter 25

Installation systems

Debian GNU/Linux and Ubuntu Linux maintains a software installation package repository based on APT and dpkg.

http://www.gnome.org/%7eole/debian/
http://www.gnome.org/%7eole/ubuntu/
Fedora Linux maintains a tree of software in the dhf installation system on top of RedHat Package Manager (RPM) packages.
http://www.gnome.org/%7eole/fedora/
The MacPorts project makes it easy to install open-source/free software on Mac OS X with the port command in Terminal from a package repository based on FreeBSD ports.

The MacPorts packages gnome-internet-radio-locator and gnome-radio contain the packaged software implemented and described in this thesis.
https://ports.macports.org/port/gnome-internet-radio-locator/summary
https://ports.macports.org/port/gnome-radio/summary

Chapter 26

Desktop Systems

The rise of Linux in the late 1990s with the GTK+ and GNOME projects made people aware of the importance of free licenses and the consequences of bad licensing. The Open Source Initiative was launched in 1998 and it was crucial in the public release of the Mozilla Web Browser source code from Netscape, Inc. Some supporters of Open Source were critical of the GNU General Public License. The next generation of the Library GPL was dubbed Lesser General Public License. Free Software developers kept working on GTK+, GIMP and GNOME and GStreamer.

Chapter 27

Mapping systems

In mathematics, the four color theorem, or the four color map theorem, states that, given any separation of a plane into contiguous regions, producing a figure called a map, no more than four colors are required to color the regions of the map so that no two adjacent regions have the same color.

The four color theorem reduces the number of colours that are required to produce maps into four colours. It was proved around 2004 by Steve Coast and the community began work on free geographical maps in the OpenStreetMap.org project. The project produces free maps of geographical sites, buildings and roads in a crowd-sourced project dubbed OSM. In 2013 the GNOME project released GNOME Maps. In 2017 support for OpenStreetMap in GNOME Internet Radio Locator as well as support for libchamplain and gst-player was implemented. In 2018 the work on the GTK+ 4.0 port began as gtk-internet-radio-locator that is published in https://download.gnome.org/sources/gtk-internet-radio-locator/

Chapter 28

Graphical User Interface systems

In 2017 libinput, a Free Software project to build graphical user interface cursors and device input systems for Wayland, the replacement display system after The X Window system, was presented at GUADEC 2017 (see chapter 13: Graphical Display Interfaces). It provides input support for mice, trackballs, and touch pads. Work on documenting GNOME Internet Radio Locator began at GUADEC 2017 at Manchester Metropolitan University in July 2017.

Chapter 29

Experiment

With components from BuildCircuit in Australia the FM transmitter hardware was built by Adrian Szabo Aabech. The software gnome-internet-radio-locator was built by Ole Kr. Aamot. A breadboard with 5 resistors, 8 condensators, 1 inductor and 17 nodes was configured for the experiment Internet-to-FM. The transmitter works from 3 to 9 Volt for FM 92.1 Mhz. The range distance is 10 meters with 30 cm antenna.

29.1 Time-Dilation in Internet-to-FM Audio Signal

An experiment in backward signal processing was performed successfully in 2017 by Adrian Szabo Aabech and Ole Kr. Aamot.

A computer connected to the FM transmitter hardware and Internet WiFi broadcasted international radio broadcasts from gnome-internet-radio-locator to the bandwidth FM 92.1 Mhz from remote audio sources broadcasted with HTTP/1.1 over the network stack TCP/IP, hereby known as "Internet-to-FM".

29.2 Experiment Setup Details

`http://www.gnome.org/%7eole/HIOA2017.pdf`

Chapter 30

Result of experiment

Visual switching with audio feedback delay worked with Internet-to-FM for FM transmitter broadcasting to FM receiver wirelessly connected to a computer connected to WiFi and visually running gnome-internet-radio-locator and broadcasting audio over FM via live audio stream over Wireless Internet.

We could hear the latency delay between FM 92.1 MHz and WiFi 2400 GHz.

Chapter 31

Final production system

In 2001 Norwegian Computing Center built a full audio recording studio in the Kristen Nygaard Building in Forskningsparken and organized 01lab, a conference and meeting place for scientists and artists. A full studio was designed and built by Benum.

In 2002 I began work on a XML DTD and the GNOME Internet Radio Locator program to stream audio from a FM radio connected to the Internet in the studio of Radio NOVA, Radio Tellus and radiOrakel to the GNOME desktop.

Chapter 32

Survey

Two of three patients, the third who is my opponent who thinks that you should only listen to local radio stations, answered that they enjoyed listening to NRK P2 and NRK Alltid Klassisk.

Two of three nurses prefer NRK P3, while the third who is my defender thinks that BBC World Service is the best radio station, because it's from around the world.

The patient subject Geir installed Ubuntu 20.04 and GNOME Internet Radio Locator 3.0.1 on his MacBook Air. The patient subject Anne prefers free walking in nature.

The experiment with the patient was successful.

The Corona epidemic in March 2020 led to isolation, while the nature of radio is social and less people meet each other.

Radio as a medical technology is a tool in certain situations, but meeting other people is important. Radio can be an isolating facot in our lives, but is a social tool for people with social anxiety.

Chapter 33

Result

The Internet-to-FM radio experiments with GNOME Internet Radio Locator software that finalizes the ideas explained in this thesis is available from `https://www.gnome.org/%7eole/gnome-internet-radio-locator` and `https://download.gnome.org/sources/gnome-internet-radio-locator/3.0/gnome-internet-radio-locator-3.0.1.tar.xz`

Chapter 34

Future work

In May 2018 the work on the GTK+ 4.0 port began as gtk-internet-radio-locator that is published in https://download.gnome.org/sources/gtk-internet-radio-locator/ http://www.aamot.org/ole/thesis/aamot-thesis-20180929.zip

In August 2019 I announced the GNOME Radio Project and published a conference talk on GNOME Radio during GUADEC 2019 available from http://www.gnomeradio.org/GUADEC2019.pdf with source code available from http://www.gnomeradio.org/0.2/gnome-radio-0.2.0.tar.gz

Part V

Software

Chapter 35

Position of Radio in the 7 Layers of the OSI Model

A Public Internet Radio Client for accessing Free Audio Maps such as GNOME Radio is positioned in the following 7 Layers of the OSI Model:

- Application (End User Layer): HTTP, DNS
- Presentation (Syntax Layer): MPEG, OGG
- Session (Synchronization): GStreamer
- Transport (End-to-end Connections): TCP, UDP
- Network (Packets): IP, ICMP, IPSec, IGMP
- Data Link (Frames): Ethernet
- Physical (Physical Structure): Coax, Fiber, Wireless, Hubs, Repeaters)

35.1 Application

Application — GUI systems — End User Layer — Protocol Name
Radio — GTK+, GNOME — X, Wayland — HTTP, DNS

35.2 Presentation

Presentation — Mapping systems — Syntax layer — Protocol Name
Audio Map — Champlain, OpenStreetMap — MPEG, OGG — ASN.1

35.3 Session

Session — Desktop systems — Synchronization — Protocol Name
Synch — GStreamer — GstClock — PTP

35.4 Transport

Transport — Operating systems — End-to-end connection — Protocol Name
Linux — GNOME — TCP, UDP — TCP/IP

35.5 Network

Network — Internet systems — Packets — Protocol Name
Internet — INET — IP, ICMP, IPSec, IGMP — TCP/IP

35.6 Data Link

Data Link — Computer systems — Frames — Ethernet
WiFi — 802.11 — MAC — LAN

35.7 Physical

Physical — Network systems — Physical structure — Protocol name
Hardware — PC, Mac, Router, Switch, Patch Panel — PCB — Coax, Fiber, Wireless, Hubs, Repeaters

Chapter 36

Source Code in gnome-internet-radio-locator-3.0.1.tar.xz

http://download.gnome.org/sources/gnome-internet-radio-locator/3.0/gnome-internet-radio-locator-3.0.1.tar.xz

36.1 gnome-internet-radio-locator.c

```
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses
     />.
 */

#include <gtk/gtk.h>
#include <gst/player/player.h>
```

```
#include <champlain/champlain.h>
#include <champlain-gtk/champlain-gtk.h>
#include <clutter-gtk/clutter-gtk.h>
#include <geocode-glib/geocode-glib.h>
#include <glib/gstdio.h>
#include <glib/gi18n.h>
#include <string.h>

#include "gnome-internet-radio-locator.h"
#include "gnome-internet-radio-locator-gui.h"
#include "gnome-internet-radio-locator-markers.h"
#include "gnome-internet-radio-locator-player.h"

#define N_COLS 2
#define COL_ID 0
#define COL_NAME 1

static ChamplainPathLayer *path_layer;
static ChamplainPathLayer *path;
static gboolean destroying = FALSE;

static ChamplainView *champlain_view;
GApplication *app;
GtkWidget *search_selector;
GtkWidget *window;
GtkWidget *statusbar;
GNOMEInternetRadioLocatorData *gnome_internet_radio_locator;

GtkWidget *input;

GtkEntryCompletion *completion;

GList *gnome_internet_radio_locator_archivers;
GList *gnome_internet_radio_locator_listeners;
GList *gnome_internet_radio_locator_programs;
GList *gnome_internet_radio_locator_stations;
GList *gnome_internet_radio_locator_streams;

GtkWidget *archivers_selector = NULL;
GtkWidget *listeners_selector = NULL;
GtkWidget *programs_selector = NULL;
GtkWidget *stations_selector = NULL;
GtkWidget *streams_selector = NULL;
GtkWidget *search_selector = NULL;

gchar *list_item_data_key ="list_item_data";

GtkWidget *gnome_internet_radio_locator_app;
GstPlayer *player;
ChamplainMarkerLayer *layer;
ClutterActor *marker;

gchar *world_station_xml_filename, *local_station_xml_file;

extern GNOMEInternetRadioLocatorStationInfo *stationinfo, *localstation
    ;

extern struct GNOMEInternetRadioLocatorMedia *media;

GStatBuf stats;

ChamplainView *view;
```

```
gchar *
str_channels (GNOMEInternetRadioLocatorChannels type) {
  gchar *channels;
  if (type == GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_MONO) {
    channels = g_strdup("Mono");
  }
  if (type == GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_STEREO) {
    channels = g_strdup("Stereo");
  }
  if (type == GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_5_1) {
    channels = g_strdup("Surround");
  }
  if (type == GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_NONE) {
    channels = g_strdup("None");
  }
  return channels;
}

/*
 * Terminate the main loop.
 */
static void
on_destroy (GtkWidget *widget, gpointer data)
{
    destroying = TRUE;
    gtk_main_quit ();
}


static void
toggle_layer (GtkToggleButton *widget,
          ClutterActor *layer)
{
    if (gtk_toggle_button_get_active (widget))
    {
        champlain_path_layer_set_visible (path_layer, TRUE);
        champlain_path_layer_set_visible (path, TRUE);
        champlain_marker_layer_animate_in_all_markers (
            CHAMPLAIN_MARKER_LAYER (layer));
    }
    else
    {
        champlain_path_layer_set_visible (path_layer, FALSE);
        champlain_path_layer_set_visible (path, FALSE);
        champlain_marker_layer_animate_out_all_markers (
            CHAMPLAIN_MARKER_LAYER (layer));
    }
}

static gboolean
mouse_click_cb (ClutterActor *actor, ClutterButtonEvent *event,
    ChamplainView *view)
{
    GError **error;
    gdouble lat, lon;
    GeocodePlace *place_city, *place_country;
    GeocodeLocation *location_city;
    GeocodeLocation *location_country;
    GeocodeReverse *reverse_city, *reverse_country;
    ClutterColor city_color = { 0x9a, 0x9b, 0x9c, 0x9d };
    ClutterColor text_color = { 0xff, 0xff, 0xff, 0xff };
```

```
    const char *name, *name_city, *name_country;
    /* GeocodeForward *fwd; */
    /* GList *list; */
    /* GError **err; */
    lon = champlain_view_x_to_longitude (view, event->x);
    lat = champlain_view_y_to_latitude (view, event->y);
    /* champlain_view_center_on (CHAMPLAIN_VIEW (view), lat, lon); */
    location_city = geocode_location_new (lat, lon,
        GEOCODE_LOCATION_ACCURACY_CITY);
    location_country = geocode_location_new (lat, lon,
        GEOCODE_LOCATION_ACCURACY_COUNTRY);
    reverse_city = geocode_reverse_new_for_location (location_city);
    reverse_country = geocode_reverse_new_for_location (
        location_country);
    place_city = geocode_reverse_resolve (reverse_city, error);
    place_country = geocode_reverse_resolve (reverse_country, error);
    name_city = geocode_place_get_town (place_city);
    name_country = geocode_place_get_country (place_country);
    if (!g_strcmp0(name_country, "United␣States␣of␣America")) {
        name_country = geocode_place_get_state (place_country);
    }
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    name = g_strconcat(name_city, ",␣", name_country, NULL);
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), (gchar *)name);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_color);
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker), lat,
        lon);
    if (g_strcmp0(name, NULL)) {
        champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (
            marker));
        gtk_entry_set_text(GTK_ENTRY(input),(gchar *)name);
        g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
            G_CALLBACK(marker_function), NULL);
    }
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Mouse␣click␣at:␣%f␣%f␣(%s)\
        n", lat, lon, name);
    return TRUE;
}


static void
map_source_changed (GtkWidget *widget,
            ChamplainView *view)
{
    gchar *id;
    ChamplainMapSource *source;
    GtkTreeIter iter;
    GtkTreeModel *model;

    if (!gtk_combo_box_get_active_iter (GTK_COMBO_BOX (widget), &iter))
        return;

    model = gtk_combo_box_get_model (GTK_COMBO_BOX (widget));

    gtk_tree_model_get (model, &iter, COL_ID, &id, -1);

    ChamplainMapSourceFactory *factory =
        champlain_map_source_factory_dup_default ();
```

```
    source = champlain_map_source_factory_create_cached_source (factory
        , id);
    g_object_set (G_OBJECT (view), "map-source", source, NULL);
    g_object_unref (factory);
}

static void
zoom_changed (GtkSpinButton *spinbutton,
          ChamplainView *view)
{
    gint zoom = gtk_spin_button_get_value_as_int (spinbutton);

    g_object_set (G_OBJECT (view), "zoom-level", zoom, NULL);
}

static void
volume_changed (GtkSpinButton *spinbutton)
{
    gint volume = gtk_spin_button_get_value_as_int (spinbutton);

    g_object_set (G_OBJECT (player), "volume", volume, NULL);
}

static void
map_zoom_changed (ChamplainView *view,
          GParamSpec *gobject,
          GtkSpinButton *spinbutton)
{
    gint zoom;

    g_object_get (G_OBJECT (view), "zoom-level", &zoom, NULL);
    gtk_spin_button_set_value (spinbutton, zoom);
}


static void
view_state_changed (ChamplainView *view,
            GParamSpec *gobject,
            GtkImage *image)
{
    ChamplainState state;

    if (destroying)
        return;

    g_object_get (G_OBJECT (view), "state", &state, NULL);
    if (state == CHAMPLAIN_STATE_LOADING)
    {
        gtk_image_set_from_icon_name (image, "edit-find",
            GTK_ICON_SIZE_BUTTON);
    }
    else
    {
        gtk_image_clear (image);
    }
}


static void
zoom_in (GtkWidget *widget,
     ChamplainView *view)
{
```

```
    champlain_view_zoom_in (view);
}


static void
zoom_out (GtkWidget *widget,
      ChamplainView *view)
{
    champlain_view_zoom_out (view);
}


static void
toggle_wrap (GtkWidget *widget,
         ChamplainView *view)
{
    gboolean wrap;

    wrap = champlain_view_get_horizontal_wrap (view);
    champlain_view_set_horizontal_wrap (view, !wrap);
}

static void
build_combo_box (GtkComboBox *box)
{
    ChamplainMapSourceFactory *factory;
    GSList *sources, *iter;
    GtkTreeStore *store;
    GtkTreeIter parent;
    GtkCellRenderer *cell;

    store = gtk_tree_store_new (N_COLS, G_TYPE_STRING, /* id */
                    G_TYPE_STRING, /* name */
                    -1);

    factory = champlain_map_source_factory_dup_default ();
    sources = champlain_map_source_factory_get_registered (factory);

    iter = sources;
    while (iter != NULL)
    {
        ChamplainMapSourceDesc *desc = CHAMPLAIN_MAP_SOURCE_DESC (iter
            ->data);
        const gchar *id = champlain_map_source_desc_get_id (desc);
        const gchar *name = champlain_map_source_desc_get_name (desc);

        gtk_tree_store_append (store, &parent, NULL);
        gtk_tree_store_set (store, &parent, COL_ID, id,
                    COL_NAME, name, -1);

        iter = g_slist_next (iter);
    }

    g_slist_free (sources);
    g_object_unref (factory);

    gtk_combo_box_set_model (box, GTK_TREE_MODEL (store));

    cell = gtk_cell_renderer_text_new ();
    gtk_cell_layout_pack_start (GTK_CELL_LAYOUT (box), cell, FALSE);
    gtk_cell_layout_set_attributes (GTK_CELL_LAYOUT (box), cell,
                    "text", COL_NAME, NULL);
```

```
}


static void
append_point (ChamplainPathLayer *layer, gdouble lon, gdouble lat)
{
    ChamplainCoordinate *coord;

    coord = champlain_coordinate_new_full (lon, lat);
    champlain_path_layer_add_node (layer, CHAMPLAIN_LOCATION (coord));
}

static void
add_clicked (GtkButton     *button,
         ChamplainView *view)
{
    GtkWidget *window, *dialog, *vbox, *combo;
    GtkResponseType response;

    window = g_object_get_data (G_OBJECT (view), "window");
    dialog = gtk_dialog_new_with_buttons ("Add␣secondary␣map␣source",
                          GTK_WINDOW (window),
                          GTK_DIALOG_MODAL,
                          "Add",
                          GTK_RESPONSE_OK,
                          "Cancel",
                          GTK_RESPONSE_CANCEL,
                          NULL);

    combo = gtk_combo_box_new ();
    build_combo_box (GTK_COMBO_BOX (combo));
    gtk_combo_box_set_active (GTK_COMBO_BOX (combo), 0);

    vbox = gtk_dialog_get_content_area (GTK_DIALOG (dialog));
    gtk_container_add (GTK_CONTAINER (vbox), combo);

    gtk_widget_show_all (dialog);

    response = gtk_dialog_run (GTK_DIALOG (dialog));

    if (response == GTK_RESPONSE_OK)
    {
        GtkTreeModel *model;
        GtkTreeIter iter;
        ChamplainMapSource *source;
        ChamplainMapSourceFactory *factory;
        char *id;

        if (!gtk_combo_box_get_active_iter (GTK_COMBO_BOX (combo), &
            iter))
            return;

        model = gtk_combo_box_get_model (GTK_COMBO_BOX (combo));

        gtk_tree_model_get (model, &iter, COL_ID, &id, -1);

        factory = champlain_map_source_factory_dup_default ();
        source = champlain_map_source_factory_create_memcached_source (
            factory, id);

        champlain_view_add_overlay_source (view, source, 0.6 * 255);
```

```
        g_object_unref (factory);
        g_free (id);
    }

    gtk_widget_destroy (dialog);
}

static void
new_station(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    g_print(_("New␣Internet␣Radio␣Station\n"));
  return;
}

static void
search_station(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    g_print(_("Search␣Internet␣Radio␣Station\n"));
  return;
}

static void
listen_station(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    gnome_internet_radio_locator_player_stop(player);
    player = gst_player_new (NULL,
        gst_player_g_main_context_signal_dispatcher_new(NULL));
    /* g_object_set_data(G_OBJECT(widget), "station_uri",
        g_value_get_string(&value)); */
    if (!g_strcmp0(gnome_internet_radio_locator->selected_station_uri,
        NULL)) {
        gnome_internet_radio_locator_player_new(player, _("http://fm939
            .wnyc.org/wnycfm"));
    } else {
        gnome_internet_radio_locator_player_new(player,
            gnome_internet_radio_locator->selected_station_uri);
    }
    gst_player_play(player);
    return;
}

static void
stop_station(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    guint context_id;
    gnome_internet_radio_locator_player_stop(player);
    context_id = gtk_statusbar_get_context_id (GTK_STATUSBAR (statusbar
        ), "Station␣Name");
    gtk_statusbar_pop (GTK_STATUSBAR (statusbar), GPOINTER_TO_INT (
        context_id));
    gtk_statusbar_push (GTK_STATUSBAR (statusbar), GPOINTER_TO_INT (
        context_id), _("Search␣by␣city␣or␣drag/click␣on␣the␣zoomable␣
        map␣to␣listen␣to␣a␣radio␣broadcast"));
    return;
}

static void
pause_station(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
  /* FIXME: Removing Pause.  Can't quit after gst_player_pause is
      called. */
#if 0
```

```
    gnome_internet_radio_locator_player_pause(player);
#endif
    return;
}

static void
prev_station(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    g_print(_("Previous␣Internet␣Radio␣Station\n"));
    return;
}

static void
stations_all(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    return;
}

static void
next_station(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    return;
}

static void
about_station_cb(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    return;
}

static void
about_program_cb(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
    return;
}

static void
quit_program(GSimpleAction *simple, GVariant *parameter, gpointer
    user_data) {
  /* gnome_internet_radio_locator_player_stop(player); */
  /* g_application_quit(app); */
    gst_player_stop(player);
    g_object_unref (player);
    gst_deinit();
    gtk_main_quit ();
    return;
}

void on_new_station_changed(GtkWidget * a, gpointer user_data)
{
    GNOMEInternetRadioLocatorStationInfo *stationinfo = NULL;
    /* GList *l = g_list_first(gnome_internet_radio_locator_stations);
        */
    /* stationinfo = l->data; */

    gnome_internet_radio_locator->selected_station_band =
        g_strdup(g_object_get_data(G_OBJECT(a), "station_band"));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_new_station_changed:␣%s\
        n",
        gnome_internet_radio_locator->selected_station_band);
    gnome_internet_radio_locator->selected_station_description =
```

```
        g_strdup(g_object_get_data(G_OBJECT(a), "station_description"))
            ;
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_new_station_changed:␣%s\
        n",
        gnome_internet_radio_locator->selected_station_description);
    gnome_internet_radio_locator->selected_station_location =
        g_strdup(g_object_get_data(G_OBJECT(a), "station_location"));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_new_station_changed:␣%s\
        n",
        gnome_internet_radio_locator->selected_station_location);
    gnome_internet_radio_locator->selected_station_name =
        g_strdup(g_object_get_data(G_OBJECT(a), "station_name"));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_new_station_changed:␣%s\
        n",
        gnome_internet_radio_locator->selected_station_name);
    gnome_internet_radio_locator->selected_station_uri =
        g_strdup(g_object_get_data(G_OBJECT(a), "station_uri"));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_new_station_changed:␣%s\
        n",
        gnome_internet_radio_locator->selected_station_uri);
    gnome_internet_radio_locator->selected_station_website =
        g_strdup(g_object_get_data(G_OBJECT(a), "station_website"));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_new_station_changed:␣%s\
        n",
                            gnome_internet_radio_locator->
                                selected_station_website);

    /* appbar_send_msg(_("Selected the radio station %s in %s: %s"), */
    /*      gnome_internet_radio_locator->selected_station_name, */
    /*      gnome_internet_radio_locator->selected_station_location, */
    /*      selected_station_uri, */
    /*      gnome_internet_radio_locator->selected_station_band); */
    gnome_internet_radio_locator_station_update(stationinfo,
                             gnome_internet_radio_locator->
                                 selected_station_band,
                             gnome_internet_radio_locator->
                                 selected_station_description,
                             gnome_internet_radio_locator->
                                 selected_station_name,
                             gnome_internet_radio_locator->
                                 selected_station_location,
                             gnome_internet_radio_locator->
                                 selected_station_uri,
                             gnome_internet_radio_locator->
                                 selected_station_website);
}

void on_stations_selector_changed(GtkWidget * a, gpointer user_data)
{
    GNOMEInternetRadioLocatorStationInfo *station = NULL;

    /* if (gnome_internet_radio_locator->selected_station_uri != NULL)
        */
    /*  g_free(gnome_internet_radio_locator->selected_station_uri); */

    gnome_internet_radio_locator->selected_station_uri = g_strdup(
        g_object_get_data(G_OBJECT(a), "station_uri"));

    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_station_select_changed:␣
        %s\n", gnome_internet_radio_locator->selected_station_uri);

    gnome_internet_radio_locator->selected_station_name =
```

```
        g_strdup(g_object_get_data(G_OBJECT(a), "station_name"));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_station_select_changed:␣
        %s\n",
        gnome_internet_radio_locator->selected_station_name);

    gnome_internet_radio_locator->selected_station_location =
        g_strdup(g_object_get_data(G_OBJECT(a), "station_location"));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_station_select_changed:␣
        %s\n",
        gnome_internet_radio_locator->selected_station_location);

    gnome_internet_radio_locator->selected_station_band =
        g_strdup(g_object_get_data(G_OBJECT(a), "station_band"));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_station_select_changed:␣
        %s\n",
        gnome_internet_radio_locator->selected_station_band);

    gnome_internet_radio_locator->selected_station_description =
        g_strdup(g_object_get_data(G_OBJECT(a), "station_description"))
            ;
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("on_station_select_changed:␣
        %s\n",
        gnome_internet_radio_locator->selected_station_description);

    /* appbar_send_msg(_("Selected the radio station %s in %s: %s"), */
    /*      gnome_internet_radio_locator->selected_station_name, */
    /*      gnome_internet_radio_locator->selected_station_location, */
    /*      selected_station_uri, */
    /*      gnome_internet_radio_locator->selected_station_band); */

    gnome_internet_radio_locator->selected_station_name = g_strdup(
        g_object_get_data(G_OBJECT(a), "station_name"));
    /* gnome_internet_radio_locator_history = g_list_add(GLIST(
        gnome_internet_radio_locator_history), (
        GNOMEInternetRadioLocatorStationInfo *)station); */
    /* gnome_internet_radio_locator_helper_main(selected_station_uri);
        */
}

static void
gnome_internet_radio_locator_window_cb (GtkApplication *app,
        gpointer user_data)
{
    GtkWidget *widget, *grid, *toolbar, *new, *search, *listen, *stop,
        *prev, *stations, *next, *station, *program, *quit;

    window = gtk_application_window_new (app);
    widget = gtk_champlain_embed_new();
    toolbar = gtk_toolbar_new();
    input = gtk_entry_new();

#if 0
    search = gtk_tool_button_new(gtk_image_new_from_icon_name(NULL,
        GTK_ICON_SIZE_BUTTON), _("Search"));
    gtk_tool_item_set_is_important(GTK_TOOL_ITEM(search), TRUE);
    gtk_toolbar_insert (GTK_TOOLBAR (toolbar), GTK_TOOL_ITEM(search),
        1);
    gtk_widget_show (GTK_WIDGET(search));
    gtk_tool_item_set_tooltip_text (GTK_TOOL_ITEM(search), _("Search␣
        Internet␣Radio␣Station"));
    g_signal_connect(search, "clicked", G_CALLBACK (search_station),
        GTK_WINDOW (window));
```

```
#endif

#if 0
    prev = gtk_tool_button_new(gtk_image_new_from_icon_name(NULL,
        GTK_ICON_SIZE_BUTTON), _("Prev"));
    gtk_tool_item_set_is_important(GTK_TOOL_ITEM(prev), TRUE);
    gtk_toolbar_insert (GTK_TOOLBAR (toolbar), GTK_TOOL_ITEM(prev), 5);
    gtk_widget_show (GTK_WIDGET(prev));
    gtk_tool_item_set_tooltip_text (GTK_TOOL_ITEM(prev), _("Prev␣
        Internet␣Radio␣Station"));
    g_signal_connect(prev, "clicked", G_CALLBACK (prev_station),
        GTK_WINDOW (window));

    stations = gtk_tool_button_new(gtk_image_new_from_icon_name(NULL,
        GTK_ICON_SIZE_BUTTON), _("Stations"));
    gtk_tool_item_set_is_important(GTK_TOOL_ITEM(stations), TRUE);
    gtk_toolbar_insert (GTK_TOOLBAR (toolbar), GTK_TOOL_ITEM(stations),
         6);
    gtk_widget_show (GTK_WIDGET(stations));
    gtk_tool_item_set_tooltip_text (GTK_TOOL_ITEM(stations), _("
        Stations"));
    g_signal_connect(stations, "clicked", G_CALLBACK (stations_all),
        GTK_WINDOW (window));

    next = gtk_tool_button_new(gtk_image_new_from_icon_name(NULL,
        GTK_ICON_SIZE_BUTTON), _("Next"));
    gtk_tool_item_set_is_important(GTK_TOOL_ITEM(next), TRUE);
    gtk_toolbar_insert (GTK_TOOLBAR (toolbar), GTK_TOOL_ITEM(next), 7);
    gtk_widget_show (GTK_WIDGET(next));
    gtk_tool_item_set_tooltip_text (GTK_TOOL_ITEM(next), _("Next␣
        Internet␣Radio␣Station"));
    g_signal_connect(next, "clicked", G_CALLBACK (next_station),
        GTK_WINDOW (window));

    station = gtk_tool_button_new(gtk_image_new_from_icon_name(NULL,
        GTK_ICON_SIZE_BUTTON), _("About␣Station"));
    gtk_tool_item_set_is_important(GTK_TOOL_ITEM(station), TRUE);
    gtk_toolbar_insert (GTK_TOOLBAR (toolbar), GTK_TOOL_ITEM(station),
        8);
    gtk_widget_show (GTK_WIDGET(station));
    gtk_tool_item_set_tooltip_text (GTK_TOOL_ITEM(station), _("About␣
        Station"));
    g_signal_connect(station, "clicked", G_CALLBACK (about_station_cb),
         GTK_WINDOW (window));

    program = gtk_tool_button_new(gtk_image_new_from_icon_name(NULL,
        GTK_ICON_SIZE_BUTTON), _("About␣Program"));
    gtk_tool_item_set_is_important(GTK_TOOL_ITEM(program), TRUE);
    gtk_toolbar_insert (GTK_TOOLBAR (toolbar), GTK_TOOL_ITEM(program),
        9);
    gtk_widget_show (GTK_WIDGET(program));
    gtk_tool_item_set_tooltip_text (GTK_TOOL_ITEM(program), _("About␣
        Program"));
    g_signal_connect(program, "clicked", G_CALLBACK (about_program_cb),
         GTK_WINDOW (window));
#endif
#if 0
    grid = gtk_grid_new();
    gtk_grid_attach (GTK_GRID(grid), GTK_WIDGET(toolbar), 0, 0, 1, 1);
    gtk_grid_attach (GTK_GRID(grid), GTK_WIDGET(widget), 0, 1, 1, 1);
        champlain_view = gtk_champlain_embed_get_view (
            GTK_CHAMPLAIN_EMBED(widget));
```

```
        gtk_widget_set_size_request(GTK_WIDGET(widget), 740, 580);
    gtk_container_add (GTK_CONTAINER(window), GTK_WIDGET(grid));
    g_signal_connect (window, "destroy", G_CALLBACK(gtk_main_quit),
        NULL);
#endif
    gtk_window_set_title (GTK_WINDOW(window), _("GNOME␣Internet␣Radio␣
        Locator"));
    gtk_window_set_default_size (GTK_WINDOW(window), 740, 580);
    gtk_window_maximize (GTK_WINDOW (window));
    gnome_internet_radio_locator_app =
        create_gnome_internet_radio_locator_app();
    gtk_widget_show(gnome_internet_radio_locator_app);

    /* stations_selector = create_stations_selector(
        selected_station_uri, "gnome_internet_radio_locator.xml"); */

    /* g_object_add_weak_pointer(G_OBJECT(stations_selector), */
    /*            (void **) &(stations_selector)); */

    gtk_widget_show_all (window);

}

void on_new_station_clicked(GtkWidget *a,
                gpointer user_data)
{
    GtkWidget *station;
    GNOMEInternetRadioLocatorStationInfo *stationinfo = NULL;
    /* GList *l = g_list_first(gnome_internet_radio_locator_stations);
        */
    gchar *selected_station_uri, *selected_station_band, *
        selected_station_description, *selected_station_name, *
        selected_station_location, *selected_station_website;
    /* stationinfo = l->data; */
    gint result;
    // appbar_send_msg(_("New radio station"));
    station = create_new_station_selector(user_data);
    result = gtk_dialog_run (GTK_DIALOG(station));
    switch (result)  {
    case GTK_RESPONSE_ACCEPT:
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Squeak!\n\n");
        selected_station_band = g_strdup(g_object_get_data(G_OBJECT(
            station), "station_band"));
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
            on_new_station_select_changed:␣%s\n", selected_station_band
            );
        selected_station_description = g_strdup(g_object_get_data(
            G_OBJECT(station), "station_description"));
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
            on_new_station_select_changed:␣%s\n",
            selected_station_description);
        selected_station_location = g_strdup(g_object_get_data(G_OBJECT
            (station), "station_location"));
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
            on_new_station_select_changed:␣%s\n",
            selected_station_location);
        selected_station_name = g_strdup(g_object_get_data(G_OBJECT(
            station), "station_name"));
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
            on_new_station_select_changed:␣%s\n", selected_station_name
            );
```

```
        selected_station_uri = g_strdup(g_object_get_data(G_OBJECT(
            station), "station_uri"));
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
            on_new_station_select_changed:␣%s\n", selected_station_uri)
            ;
        selected_station_website = g_strdup(g_object_get_data(G_OBJECT(
            station), "station_website"));
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
            on_new_station_select_changed:␣%s\n",
            selected_station_website);
        gnome_internet_radio_locator_station_update (stationinfo,
            selected_station_band, selected_station_description,
            selected_station_name, selected_station_location,
            selected_station_uri, selected_station_website);
        break;
    default:
        g_print (_("Nothing\n\n"));
        break;
    }
    gtk_widget_destroy(station);
    /* gtk_widget_show(station); */
}

#if 0
static gboolean
on_location_matches(GtkEntryCompletion *widget,
            GtkTreeModel *model,
            GtkTreeIter *iter,
            gpointer user_data)
{
    GValue value = {0, };
    gtk_tree_model_get_value(model, iter, STATION_LOCATION, &value);
    gnome_internet_radio_locator->selected_station_location = g_strdup(
        g_value_get_string(&value));
    g_value_unset(&value);

    /* appbar_send_msg(_("Found location %s"), */
    /*      gnome_internet_radio_locator->selected_station_location);
        */
    /* gnome_internet_radio_locator_helper_run(selected_station_uri, */
    /*      gnome_internet_radio_locator->selected_station_name, */
    /*      GNOME_INTERNET_RADIO_LOCATOR_STREAM_SHOUTCAST, */
    /*      GNOME_INTERNET_RADIO_LOCATOR_STREAM_PLAYER); */
    return FALSE;
}
#endif

gboolean
on_search_matches(GtkEntryCompletion *widget,
          GtkTreeModel *model,
          GtkTreeIter *iter,
          gpointer user_data)
{
    /* GeocodeNominatim *geocode_nominatim; */
    GeocodePlace *place;
    GeocodeLocation *geocode_location;

    GeocodePlace *place_city, *place_country;
    GeocodeLocation *location_city;
    GeocodeLocation *location_country;
    GeocodeReverse *reverse_city, *reverse_country;
```

```
    glong lat, lon;
    GValue city = {0, };
    GValue value = {0, };
    GValue station_name = {0, };
    gchar *location;
    gchar *town;
    gchar *country;
    gchar *state;
    GError **err;
    guint context_id;

    gtk_tree_model_get_value(model, iter, STATION_LOCATION, &city);
    gtk_tree_model_get_value(model, iter, STATION_URI, &value);
    gtk_tree_model_get_value(model, iter, STATION_NAME, &station_name);
/*  g_print ("on_search_matches: %s\n", (gchar *)g_value_get_string(&
    city)); */
/*  location = (gchar *)g_value_get_string(&city); */
/*  town = strtok(location, ", "); */
/*  country = strtok(NULL, " "); */
/* /\* Handle U.S. states */
/*  geocode_place_set_country (place, "United States of America"); */
/*  country = geocode_place_get_state (country); */
/* *\/ */
/*  place = geocode_place_new((gchar *)g_value_get_string(&station_name
    ), GEOCODE_PLACE_TYPE_MISCELLANEOUS); */

/*  g_print ("geocode_place_new:town: %s\n", geocode_place_get_town(
    place)); */
/*  g_print ("geocode_place_new:country: %s\n",
    geocode_place_get_country(place)); */

/*  geocode_place_set_town (place, town); */
/*  geocode_place_set_country(place, country); */

/*  geocode_nominatim = geocode_nominatim_new ("https://nominatim.gnome
    .org/", "ole@gnome.org"); */

/*  g_print ("geocode_place_get_town: %s\n", geocode_place_get_town(
    place)); */
/*  g_print ("geocode_place_get_country: %s\n",
    geocode_place_get_country(place)); */
/*  /\* g_print ("%s\n", geocode_nominatim_get_city(geocode_nominatim))
    ; *\/ */

/*  reverse_city = geocode_reverse_new_for_location(geocode_location);
    */
/*  place = geocode_reverse_resolve(reverse_city, err); */

/*  geocode_location = geocode_place_get_location(place); */
/*  g_print ("%f7.3\n", geocode_location_get_latitude(geocode_location)
    ); */
/*  lat = geocode_location_get_latitude(geocode_location); */
/*  lon = geocode_location_get_longitude(geocode_location); */
/*  g_print ("lat: %ld\n", lat); */
/*  g_print ("lon: %ld\n", lon); */

/*
    location_city = geocode_location_new (lat, lon,
        GEOCODE_LOCATION_ACCURACY_CITY);
    location_country = geocode_location_new (lat, lon,
        GEOCODE_LOCATION_ACCURACY_COUNTRY);
    reverse_city = geocode_reverse_new_for_location (location_city);
```

```
    reverse_country = geocode_reverse_new_for_location (
        location_country);
    place_city = geocode_reverse_resolve (reverse_city, error);
    place_country = geocode_reverse_resolve (reverse_country, error);
    name_city = geocode_place_get_town (place_city);
    name_country = geocode_place_get_country (place_country);
*/
    /* champlain_view_center_on (CHAMPLAIN_VIEW (view),lat,lon); */
    gnome_internet_radio_locator_player_stop(player);
    player = gst_player_new (NULL,
        gst_player_g_main_context_signal_dispatcher_new(NULL));
    /* g_object_set_data(G_OBJECT(widget), "station_uri",
        g_value_get_string(&value)); */
    gnome_internet_radio_locator_player_new(player, g_value_get_string
        (&value));
    stationinfo = gnome_internet_radio_locator_station_load_from_file(
        localstation, world_station_xml_filename);
    while (stationinfo != NULL) {
            if (strcasecmp(stationinfo->stream->uri, g_value_get_string
                (&value))==0) {
            gchar *statusmsg = g_strconcat(stationinfo->name, "␣(",
                stationinfo->uri, ")␣in␣", stationinfo->location, "␣(",
                 stationinfo->band, ",␣", g_strdup_printf("%li",
                stationinfo->stream->samplerate), "␣Hz,␣",
                g_strdup_printf("%li", stationinfo->stream->bitrate), "
                ␣kbps)", NULL);
            context_id = gtk_statusbar_get_context_id (GTK_STATUSBAR (
                statusbar), "Station␣Name");
            gtk_statusbar_pop (GTK_STATUSBAR (statusbar),
                GPOINTER_TO_INT (context_id));
            gtk_statusbar_push (GTK_STATUSBAR (statusbar),
                GPOINTER_TO_INT (context_id), statusmsg);
        }
        stationinfo = stationinfo->next;
    }
    gst_player_play(player);
    return FALSE;
}

int
main (int argc,
      char **argv)
{
    GtkWidget *window;
    GtkWidget *widget, *vbox, *bbox, *button, *viewport, *image;
    ClutterActor *scale;
    ChamplainLicense *license_actor;
    GtkListStore *model;
    GtkTreeIter iter;
    GNOMEInternetRadioLocatorStationInfo *stationinfo, *localstation;
    guint context_id;
    bindtextdomain (GETTEXT_PACKAGE,
        GNOME_INTERNET_RADIO_LOCATOR_LOCALEDIR);
    bind_textdomain_codeset (GETTEXT_PACKAGE, "UTF-8");
    textdomain (GETTEXT_PACKAGE);
    if (gtk_clutter_init (&argc, &argv) != CLUTTER_INIT_SUCCESS)
        return 1;
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    /* give the window a 10px wide border */
    gtk_container_set_border_width (GTK_CONTAINER (window), 10);
    /* give it the title */
```

```
gtk_window_set_title (GTK_WINDOW (window), _("GNOME␣Internet␣Radio␣
    Locator"));
/* Connect the destroy event of the window with our on_destroy
    function
 * When the window is about to be destroyed we get a notificaiton
     and
 * stop the main GTK loop
 */
g_signal_connect (G_OBJECT (window), "destroy", G_CALLBACK (
    on_destroy),
          NULL);

vbox = gtk_box_new (GTK_ORIENTATION_VERTICAL, 10);

widget = gtk_champlain_embed_new ();
view = gtk_champlain_embed_get_view (GTK_CHAMPLAIN_EMBED (widget));
clutter_actor_set_reactive (CLUTTER_ACTOR (view), TRUE);
g_signal_connect (view, "button-release-event", G_CALLBACK (
    mouse_click_cb), view);

g_object_set (G_OBJECT (view),
          "kinetic-mode", TRUE,
          "zoom-level", 2,
          NULL);

g_object_set_data (G_OBJECT (view), "window", window);

scale = champlain_scale_new ();
champlain_scale_connect_view (CHAMPLAIN_SCALE (scale), view);
/* champlain_view_ensure_visible(G_OBJECT (view), NULL, TRUE); */
/* champlain_view_set_keep_center_on_resize(G_OBJECT (view), TRUE);
     */
/* align to the bottom left */
clutter_actor_set_x_expand (scale, TRUE);
clutter_actor_set_y_expand (scale, TRUE);
clutter_actor_set_x_align (scale, CLUTTER_ACTOR_ALIGN_START);

clutter_actor_set_y_align (scale, CLUTTER_ACTOR_ALIGN_END);
clutter_actor_add_child (CLUTTER_ACTOR (view), scale);

license_actor = champlain_view_get_license_actor (view);
champlain_license_set_extra_text (license_actor, "Free␣Internet␣
    Radio");
champlain_view_center_on (CHAMPLAIN_VIEW (view), 42.3617430,
    -71.0839082);
layer = create_marker_layer (view, &path);
champlain_view_add_layer (view, CHAMPLAIN_LAYER (path));
champlain_view_add_layer (view, CHAMPLAIN_LAYER (layer));

path_layer = champlain_path_layer_new ();
/* Cheap approx of Highway 10 */
append_point (path_layer, 45.4095, -73.3197);
append_point (path_layer, 45.4104, -73.2846);
append_point (path_layer, 45.4178, -73.2239);
append_point (path_layer, 45.4176, -73.2181);
append_point (path_layer, 45.4151, -73.2126);
append_point (path_layer, 45.4016, -73.1926);
append_point (path_layer, 45.3994, -73.1877);
append_point (path_layer, 45.4000, -73.1815);
append_point (path_layer, 45.4151, -73.1218);
champlain_view_add_layer (view, CHAMPLAIN_LAYER (path_layer));
```

```
    gtk_widget_set_size_request (widget, 640, 520);

    bbox = gtk_box_new (GTK_ORIENTATION_HORIZONTAL, 10);

    button = gtk_button_new();
    image = gtk_image_new_from_icon_name("media-playback-start",
        GTK_ICON_SIZE_BUTTON);
    gtk_button_set_image (GTK_BUTTON (button), image);
    gtk_button_set_label (GTK_BUTTON (button), _("New"));
    g_signal_connect(button, "clicked", G_CALLBACK (
        on_new_station_clicked), view);
    gtk_container_add (GTK_CONTAINER (bbox), button);

    memset(&stats, 0, sizeof(stats));

    input = gtk_entry_new();

    completion = gtk_entry_completion_new();
    gtk_entry_completion_set_text_column(completion, STATION_NAME);
    gtk_entry_completion_set_text_column(completion, STATION_LOCATION);
    gtk_entry_set_completion(GTK_ENTRY(input), completion);
    g_signal_connect(G_OBJECT(completion), "match-selected",
             G_CALLBACK(on_search_matches), NULL);
    model = gtk_list_store_new(11, G_TYPE_STRING, G_TYPE_STRING,
        G_TYPE_STRING, G_TYPE_STRING, G_TYPE_STRING, G_TYPE_STRING,
        G_TYPE_STRING, G_TYPE_STRING, G_TYPE_STRING, G_TYPE_STRING,
        G_TYPE_STRING, G_TYPE_STRING);
    world_station_xml_filename = g_strconcat(
        GNOME_INTERNET_RADIO_LOCATOR_DATADIR, "/gnome-internet-radio-
        locator.xml", NULL);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("world_station_xml_filename␣
        =␣%s\n",
        world_station_xml_filename);

    if (world_station_xml_filename == NULL) {
        g_warning(("Failed␣to␣open␣%s.␣␣Please␣install␣it.\n"),
              world_station_xml_filename);
    }

    local_station_xml_file =
        g_strconcat(g_get_home_dir(), "/.gnome-internet-radio-locator/
            gnome-internet-radio-locator.xml", NULL);

    if (!g_stat(local_station_xml_file, &stats)) {
        localstation =
            gnome_internet_radio_locator_station_load_from_file(NULL,
            local_station_xml_file);
    } else {
        localstation = NULL;
    }

    if (localstation == NULL) {
        printf(_("Failed␣to␣open␣%s\n"), local_station_xml_file);
    }

/*   g_free (local_station_xml_file); */

    stationinfo = gnome_internet_radio_locator_station_load_from_file(
        localstation, world_station_xml_filename);

    gnome_internet_radio_locator_stations = NULL;
```

```
    while (stationinfo != NULL) {

        gtk_list_store_append(model, &iter);
        gtk_list_store_set(model,
                   &iter,
                   STATION_NAME,
                   stationinfo->name,
                   STATION_LOCATION,
                   stationinfo->location,
                   STATION_URI,
                   stationinfo->stream->uri,
                   STATION_DESCRIPTION,
                   stationinfo->description,
                   STATION_FREQUENCY,
                   stationinfo->frequency,
                   STATION_BAND,
                   stationinfo->band,
                   STATION_TYPE,
                   stationinfo->type,
                   STATION_RANK,
                   stationinfo->rank,
                   STATION_BITRATE,
                   stationinfo->bitrate,
                   STATION_SAMPLERATE,
                   stationinfo->samplerate,
                   STATION_ID,
                   stationinfo->id,
                   -1);

        stationinfo = stationinfo->next;
    }

    gtk_entry_completion_set_model(completion, GTK_TREE_MODEL(model));

    gtk_widget_show(input);

    gtk_container_add (GTK_CONTAINER (bbox), input);

#if 0
    button = gtk_button_new();
    image = gtk_image_new_from_icon_name("media-playback-start",
        GTK_ICON_SIZE_BUTTON);
    gtk_button_set_image (GTK_BUTTON (button), image);
    gtk_button_set_label (GTK_BUTTON (button), "Listen");
    g_signal_connect(button, "clicked", G_CALLBACK (listen_station),
        view);
    gtk_container_add (GTK_CONTAINER (bbox), button);
#endif

    button = gtk_button_new();
    image = gtk_image_new_from_icon_name("media-playback-stop",
        GTK_ICON_SIZE_BUTTON);
    gtk_button_set_image (GTK_BUTTON (button), image);
    gtk_button_set_label (GTK_BUTTON (button), _("Stop"));
    g_signal_connect(button, "clicked", G_CALLBACK (stop_station), view
        );
    gtk_container_add (GTK_CONTAINER (bbox), button);
    button = gtk_button_new ();
    image = gtk_image_new_from_icon_name ("zoom-in",
        GTK_ICON_SIZE_BUTTON);
    gtk_button_set_image (GTK_BUTTON (button), image);
    gtk_button_set_label (GTK_BUTTON (button), _("Zoom␣In"));
```

```
    g_signal_connect (button, "clicked", G_CALLBACK (zoom_in), view);
    gtk_container_add (GTK_CONTAINER (bbox), button);
    button = gtk_spin_button_new_with_range (0, 20, 1);
    gtk_spin_button_set_value (GTK_SPIN_BUTTON (button),
                   champlain_view_get_zoom_level (view));
    g_signal_connect (button, "changed", G_CALLBACK (zoom_changed),
        view);
    g_signal_connect (view, "notify::zoom-level", G_CALLBACK (
        map_zoom_changed),
              button);
    gtk_container_add (GTK_CONTAINER (bbox), button);

    button = gtk_button_new ();
    image = gtk_image_new_from_icon_name ("zoom-out",
        GTK_ICON_SIZE_BUTTON);
    gtk_button_set_image (GTK_BUTTON (button), image);
    gtk_button_set_label (GTK_BUTTON (button), _("Zoom␣Out"));
    g_signal_connect (button, "clicked", G_CALLBACK (zoom_out), view);
    gtk_container_add (GTK_CONTAINER (bbox), button);

#if 0
    button = gtk_toggle_button_new_with_label (_("Markers"));
    gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (button), TRUE);
    g_signal_connect (button, "toggled", G_CALLBACK (toggle_layer),
        layer);
    gtk_container_add (GTK_CONTAINER (bbox), button);
#endif
    /* button = gtk_combo_box_new (); */
    /* build_combo_box (GTK_COMBO_BOX (button)); */
    /* gtk_combo_box_set_active (GTK_COMBO_BOX (button), 0); */
    /* g_signal_connect (button, "changed", G_CALLBACK (
        map_source_changed), view); */
    /* gtk_container_add (GTK_CONTAINER (bbox), button); */

#if 0
    button = gtk_button_new ();
    image = gtk_image_new_from_icon_name ("list-add",
        GTK_ICON_SIZE_BUTTON);
    gtk_button_set_image (GTK_BUTTON (button), image);

    g_signal_connect (button, "clicked", G_CALLBACK (add_clicked), view
        );
    gtk_container_add (GTK_CONTAINER (bbox), button);
#endif
    button = gtk_button_new();
    image = gtk_image_new_from_icon_name("stop", GTK_ICON_SIZE_BUTTON);
    gtk_button_set_image (GTK_BUTTON (button), image);
    gtk_button_set_label (GTK_BUTTON (button), _("Exit"));
    g_signal_connect(button, "clicked", G_CALLBACK(quit_program), view)
        ;
    gtk_container_add (GTK_CONTAINER (bbox), button);

    button = gtk_image_new ();
    gtk_widget_set_size_request (button, 22, -1);
    g_signal_connect (view, "notify::state", G_CALLBACK (
        view_state_changed),
              button);
    gtk_box_pack_end (GTK_BOX (bbox), button, FALSE, FALSE, 0);
    viewport = gtk_frame_new (NULL);
    gtk_container_add (GTK_CONTAINER (viewport), widget);
    gtk_box_pack_start (GTK_BOX (vbox), bbox, FALSE, FALSE, 0);
    statusbar = gtk_statusbar_new ();
```

```
    context_id = gtk_statusbar_get_context_id (GTK_STATUSBAR (statusbar
        ), "Station␣Name");
    gtk_statusbar_pop (GTK_STATUSBAR (statusbar), GPOINTER_TO_INT (
        context_id));
    gtk_statusbar_push (GTK_STATUSBAR (statusbar), GPOINTER_TO_INT (
        context_id), _("Search␣by␣city␣or␣drag/click␣on␣the␣zoomable␣
        map␣to␣listen␣to␣a␣radio␣broadcast"));
    gtk_box_pack_end (GTK_BOX (vbox), statusbar, FALSE, FALSE, 0);
    gtk_widget_show (statusbar);
    gtk_container_add (GTK_CONTAINER (vbox), viewport);

    /* and insert it into the main window  */
    gtk_container_add (GTK_CONTAINER (window), vbox);

    /* make sure that everything, window and label, are visible */
    gtk_widget_show_all (window);

    gst_init(&argc, &argv);

    /* start the main loop */
    gtk_main ();

    return 0;
#if 0

    app = gtk_application_new ("org.gnome.gnome-internet-radio-locator"
        , G_APPLICATION_FLAGS_NONE);

    g_signal_connect (app, "activate", G_CALLBACK (
        gnome_internet_radio_locator_window_cb), NULL);

    status = g_application_run (G_APPLICATION (app), argc, argv);

    g_object_unref (app);

    return status;
#endif
}
```

36.2 gnome-internet-radio-locator-gui.c

```
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
```

```
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses
     />.
 */

#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <locale.h>
#include <sys/stat.h>

#include <config.h>
#include <gtk/gtk.h>
#include <gtk/gtkcombobox.h>
#include <gst/player/player.h>
#include <glib/gstdio.h>
#include <glib/gi18n.h>
#include <gio/gio.h>
#include "gnome-internet-radio-locator.h"
#include "gnome-internet-radio-locator-gui.h"
#include "gnome-internet-radio-locator-keys.h"
#include "gnome-internet-radio-locator-player.h"
#include "gnome-internet-radio-locator-program.h"
#include "gnome-internet-radio-locator-station.h"
#include "gnome-internet-radio-locator-streams.h"
#include "gnome-internet-radio-locator-tz.h"

extern GtkWidget *gnome_internet_radio_locator_app;
extern GtkWidget *search_selector;
extern GtkWidget *input;
extern GtkWidget *statusbar;
extern GstPlayer *player;

GNOMEInternetRadioLocatorStationInfo *stationinfo, *localstation;

GtkWidget *create_stations_selector(char *selected_station_uri,
                    char *filename)
{
    GtkWidget *stations_selector;
    GtkWidget *align, *menu, *drop_down, *item;
    gchar *station_uri, *station_name, *station_location, *station_band
        , *station_description, *station_website;
    gchar *label, *world_station_xml_filename, *local_station_xml_file;
    int i = 0, selection = -1;
    GStatBuf stats;
    memset(&stats, 0, sizeof(stats));
    /* The Stations dialog */
    /* stations_selector = gtk_dialog_new_with_buttons("Select a
        station", */
    /*                       GTK_WINDOW(gnome_internet_radio_locator_app
        ), */
    /*                       0,  /\* flags *\/ */
    /*                       NULL, */
    /*                       GTK_RESPONSE_ACCEPT, */
    /*                       NULL); */
    /* gtk_container_set_border_width */
    /*     (GTK_CONTAINER(GTK_DIALOG(stations_selector)->vbox), 6); */
```

```
    /* align = gtk_alignment_new(0.5, 0.5, 0, 0); */
    /* gtk_container_add(GTK_CONTAINER */
    /*         (GTK_DIALOG(stations_selector)->vbox), align); */
    /* gtk_container_set_border_width(GTK_CONTAINER(align), 6); */
    /* gtk_widget_show(align); */
    menu = gtk_menu_new();
    gtk_widget_show(menu);
    /* creating the menu items */
    /* world_station_xml_filename = gnome_program_locate_file(NULL, */
    /*                              GNOME_FILE_DOMAIN_APP_DATADIR, */
    /*                              "gnome-internet-radio-locator/gnome-
        internet-radio-locator.xml", */
    /*                              FALSE, */
    /*                              NULL); */

    /* world_station_xml_filename = g_strdup("https://people.gnome.org
        /~ole/gnome-internet-radio-locator/gnome-internet-radio-locator
        .xml"); */
    world_station_xml_filename = g_strconcat(
        GNOME_INTERNET_RADIO_LOCATOR_DATADIR, "/gnome-internet-radio-
        locator.xml", NULL);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("world_station_xml_filename␣
        =␣%s\n",
        world_station_xml_filename);
    if (world_station_xml_filename == NULL) {
        g_warning(_("Failed␣to␣open␣%s\n"),
              world_station_xml_filename);
    }
    local_station_xml_file =
        g_strconcat(g_get_home_dir(), "/.gnome-internet-radio-locator/
            gnome-internet-radio-locator.xml", NULL);

    if (!g_stat(local_station_xml_file, &stats)) {
        localstation =
            gnome_internet_radio_locator_station_load_from_file(NULL,
            local_station_xml_file);
    } else {
        localstation = NULL;
    }
    if (localstation == NULL) {
        g_warning(_("Failed␣to␣open␣%s\n"), local_station_xml_file);
    }
/*   g_free (local_station_xml_file); */
    stationinfo =
        gnome_internet_radio_locator_station_load_from_file(
            localstation,
                    world_station_xml_filename);
    gnome_internet_radio_locator_stations = NULL;
    while (stationinfo != NULL) {

        label =
            g_strconcat(stationinfo->name, "␣(",
                stationinfo->location, ")", NULL);
        station_uri = g_strdup(stationinfo->stream->uri);
        station_name = g_strdup(stationinfo->name);
        station_location = g_strdup(stationinfo->location);
        station_band = g_strdup(stationinfo->band);
        station_description = g_strdup(stationinfo->description);
        station_website = g_strdup(stationinfo->uri);
```

```
        gnome_internet_radio_locator_stations = g_list_append(
            gnome_internet_radio_locator_stations,(
            GNOMEInternetRadioLocatorStationInfo *)stationinfo);

        if (label != NULL) {
            item = gtk_menu_item_new_with_label(label);
            gtk_menu_shell_append(GTK_MENU_SHELL(menu), item);
            g_signal_connect(G_OBJECT(item), "activate",
                     G_CALLBACK
                     (on_stations_selector_changed),
                     NULL);
            g_object_set_data(G_OBJECT(item), "station_uri",
                       (gpointer) station_uri);
            g_object_set_data(G_OBJECT(item), "station_name",
                       (gpointer) station_name);
            g_object_set_data(G_OBJECT(item),
                       "station_location",
                       (gpointer) station_location);
            g_object_set_data(G_OBJECT(item),
                       "station_band",
                       (gpointer) station_band);
            g_object_set_data(G_OBJECT(item),
                       "station_description",
                       (gpointer) station_description);
            g_object_set_data(G_OBJECT(item),
                       "station_website",
                       (gpointer) station_website);
            gtk_widget_show(item);
            g_free(label);

            /* selection */
            if (selected_station_uri != NULL &&
                !strcmp(selected_station_uri, station_uri))
                selection = i;
        } else {
            g_free(station_uri);
            g_free(station_name);
            g_free(station_location);
            g_free(station_band);
            g_free(station_description);
        }
        i++;
        stationinfo = stationinfo->next;
    }

    /* drop_down = gtk_combo_box_new(); */
    /* gtk_widget_show(drop_down); */
    /* gtk_combo_box_popdown(GTK_OPTION_MENU(drop_down), menu); */
    /* gtk_container_add(GTK_CONTAINER(align), drop_down); */

    if (selection != -1)
        gtk_combo_box_set_active(GTK_COMBO_BOX(drop_down), selection);

    g_signal_connect(G_OBJECT(stations_selector), "response",
              G_CALLBACK(gtk_widget_hide),
              (gpointer) stations_selector);
    g_signal_connect(G_OBJECT(stations_selector), "delete-event",
              G_CALLBACK(gtk_widget_hide),
              (gpointer) stations_selector);

    return stations_selector;
}
```

```
static gboolean
on_location_matches(GtkEntryCompletion *widget,
            GtkTreeModel *model,
            GtkTreeIter *iter,
            gpointer user_data)
{
    GValue value = {0, };

    gtk_tree_model_get_value(model, iter, STATION_LOCATION, &value);
    gnome_internet_radio_locator->selected_station_location = g_strdup(
        g_value_get_string(&value));
    g_value_unset(&value);

    /* appbar_send_msg(_("Found location %s"), */
    /*      gnome_internet_radio_locator->selected_station_location);
        */
    /* gnome_internet_radio_locator_helper_run(
        gnome_internet_radio_locator->selected_station_uri, */
    /*      gnome_internet_radio_locator->selected_station_name, */
    /*      GNOME_INTERNET_RADIO_LOCATOR_STREAM_SHOUTCAST, */
    /*      GNOME_INTERNET_RADIO_LOCATOR_STREAM_PLAYER); */
    return FALSE;
}

void
save_cb (GtkWidget *widget, gpointer data) {

    /* g_print("%s\n", data); */
        gint context_id;
        char *nameentry, *locationentry, *urientry, *websiteentry, *
            descriptionentry, *statusmsg;
    nameentry = g_object_get_data(G_OBJECT(widget), "station_name");
    locationentry = g_object_get_data(G_OBJECT(widget), "
        station_location");
    urientry = g_object_get_data(G_OBJECT(widget), "station_uri");
    websiteentry = g_object_get_data(G_OBJECT(widget), "station_website
        ");
    descriptionentry = g_object_get_data(G_OBJECT(widget), "
        station_description");
    player = gst_player_new (NULL,
        gst_player_g_main_context_signal_dispatcher_new(NULL));
    gnome_internet_radio_locator_player_new(GST_PLAYER(player),
        urientry);
    context_id = gtk_statusbar_get_context_id (GTK_STATUSBAR (statusbar
        ), "Station␣Name");
    statusmsg = g_strconcat(_("Added␣"), nameentry, _("␣in␣"),
        locationentry, NULL);
    gtk_statusbar_push (GTK_STATUSBAR (statusbar), GPOINTER_TO_INT (
        context_id), statusmsg);
    gst_player_stop(player);
    gst_player_play(player);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("%s\n", nameentry);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("%s\n", locationentry);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("%s\n", urientry);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("%s\n", websiteentry);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("%s\n", descriptionentry);
}

GtkWidget *create_new_station_selector(gchar *location) {

    GtkWidget *station_selector, *content_area;
```

```
GtkWidget *align;
GtkWidget *bandentry, *descriptionentry, *nameentry, *locationentry
    , *urientry, *websiteentry;
GtkEntryCompletion *completion;
GtkListStore *location_model;
GtkTreeIter iter;
TzDB *db;
GPtrArray *locs;
guint i;
char *pixmap_dir = NULL;
gchar *path = NULL;
GtkWidget *item = NULL;
setlocale (LC_ALL, "C");

gchar *world_station_xml_filename, *local_station_xml_file;
gint retval;

/* int i = 0, search_selection = -1; */

GStatBuf stats;

memset(&stats, 0, sizeof(stats));

/* The Stations dialog */
station_selector = gtk_dialog_new_with_buttons(_("New␣Internet␣
    Radio␣Station"),
                           GTK_WINDOW(
                               gnome_internet_radio_locator_app),
                           0,
                           (_("_Save")),
                           GTK_RESPONSE_ACCEPT,
                           NULL);
content_area = gtk_dialog_get_content_area (GTK_DIALOG (
    station_selector));

g_signal_connect(G_OBJECT(station_selector), "response", G_CALLBACK
    (save_cb), G_OBJECT(station_selector));
/* gtk_container_set_border_width */
/*  (GTK_CONTAINER(GTK_DIALOG(station_selector)->vbox), 6);  */

/* align = gtk_alignment_new(0.5, 0.5, 0, 0);  */
/* gtk_container_add(GTK_CONTAINER */
/*        (GTK_DIALOG(station_selector)->vbox), align);  */
/* gtk_container_set_border_width(GTK_CONTAINER(align), 6); */
/* gtk_widget_show(align); */

bandentry = gtk_entry_new();
nameentry = gtk_entry_new();
locationentry = gtk_entry_new();
urientry = gtk_entry_new();
websiteentry = gtk_entry_new();
descriptionentry = gtk_entry_new();

gtk_entry_set_text(GTK_ENTRY(nameentry), _("Station␣name"));
gtk_entry_set_text(GTK_ENTRY(bandentry), _("Bandwidth"));
if (!g_strcmp0(gtk_entry_get_text(GTK_ENTRY(input)),"")) {
    gtk_entry_set_text(GTK_ENTRY(locationentry), _("City␣name"));
} else {
    gtk_entry_set_text(GTK_ENTRY(locationentry), (gpointer)
        gtk_entry_get_text(GTK_ENTRY(input)));
}
```

```
gtk_entry_set_text(GTK_ENTRY(urientry), _("http://uri-to-stream/"))
    ;
gtk_entry_set_text(GTK_ENTRY(descriptionentry), _("Description"));
gtk_entry_set_text(GTK_ENTRY(websiteentry), _("http://uri-to-
    website/"));
completion = gtk_entry_completion_new();
gtk_entry_completion_set_text_column(completion, STATION_LOCATION);
gtk_entry_set_completion(GTK_ENTRY(locationentry), completion);
g_signal_connect(G_OBJECT(completion), "match-selected",
         G_CALLBACK(on_location_matches), NULL);
location_model = gtk_list_store_new(2, G_TYPE_STRING, G_TYPE_STRING
    );
world_station_xml_filename = g_strconcat(
    GNOME_INTERNET_RADIO_LOCATOR_DATADIR, "/gnome-internet-radio-
    locator.xml", NULL);
GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("world_station_xml_filename␣
    =␣%s\n",
    world_station_xml_filename);
if (world_station_xml_filename == NULL) {
    g_warning((_("Failed␣to␣open␣%s.␣␣Please␣install␣it.\n")),
          world_station_xml_filename);
}
local_station_xml_file =
    g_strconcat(g_get_home_dir(), "/.gnome_internet_radio_locator/
        gnome_internet_radio_locator.xml", NULL);
if (!g_stat(local_station_xml_file, &stats)) {
    localstation =
        gnome_internet_radio_locator_station_load_from_file(NULL,
        local_station_xml_file);
} else {
    localstation = NULL;
}
if (localstation == NULL) {
    g_warning(_("Failed␣to␣open␣%s\n"), local_station_xml_file);
}
stationinfo = gnome_internet_radio_locator_station_load_from_file(
    localstation,
                                  world_station_xml_filename);
gtk_container_add(GTK_CONTAINER(content_area), nameentry);
gtk_container_add(GTK_CONTAINER(content_area), bandentry);
gtk_container_add(GTK_CONTAINER(content_area), locationentry);
gtk_container_add(GTK_CONTAINER(content_area), urientry);
gtk_container_add(GTK_CONTAINER(content_area), descriptionentry);
gtk_container_add(GTK_CONTAINER(content_area), websiteentry);
gtk_widget_show(nameentry);
gtk_widget_show(bandentry);
gtk_widget_show(locationentry);
gtk_widget_show(urientry);
gtk_widget_show(descriptionentry);
gtk_widget_show(websiteentry);
/* g_signal_connect(G_OBJECT(station_selector), GTK_RESPONSE_ACCEPT
    , */
/*       G_CALLBACK(on_new_station_selector_changed), */
/*       NULL); */
g_object_set_data(G_OBJECT(station_selector), "station_name",
          (gchar *) gtk_entry_get_text(GTK_ENTRY(nameentry)));
g_object_set_data(G_OBJECT(station_selector), "station_band",
          (gchar *) gtk_entry_get_text(GTK_ENTRY(bandentry)));
g_object_set_data(G_OBJECT(station_selector), "station_location",
          (gchar *) gtk_entry_get_text(GTK_ENTRY(locationentry)));
GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("LOCATIONENTRY:␣%s\n", (
    gchar *) gtk_entry_get_text(GTK_ENTRY(locationentry)));
```

```
    g_object_set_data(G_OBJECT(station_selector), "station_uri",
              (gchar *) gtk_entry_get_text(GTK_ENTRY(urientry)));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("URIENTRY:␣%s\n", (gchar *)
        gtk_entry_get_text(GTK_ENTRY(urientry)));
    g_object_set_data(G_OBJECT(station_selector), "station_description"
        ,
              (gchar *) gtk_entry_get_text(GTK_ENTRY(descriptionentry))
                  );
    g_object_set_data(G_OBJECT(station_selector), "station_website",
              (gchar *) gtk_entry_get_text(GTK_ENTRY(websiteentry)));
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("WEBSITEENTRY:␣%s\n", (gchar
         *) gtk_entry_get_text(GTK_ENTRY(websiteentry)));
#if 0 /* FIXME: Add input fields */
    g_object_set_data(G_OBJECT(station_selector), "station_description"
        ,
              (gchar *) station_description);
    g_object_set_data(G_OBJECT(station_selector), "station_website",
              (gchar *) station_website);
#endif
    // gtk_widget_show(station_selector);
    // g_free(label);
    g_signal_connect(G_OBJECT(station_selector), "response",
             G_CALLBACK(gtk_widget_hide),
             (gpointer) station_selector);
    g_signal_connect(G_OBJECT(station_selector), "delete-event",
             G_CALLBACK(gtk_widget_hide),
             (gpointer) station_selector);
    /* tz_db_free (db); */
    /* g_free (pixmap_dir); */
    /* g_free (filename); */
    /* g_free (path); */
    return station_selector;
}

GtkWidget *create_gnome_internet_radio_locator_app(void)
{
    GtkWidget *gnome_internet_radio_locator_app;
    GtkWidget *vbox1;
    GtkWidget *gnome_internet_radio_locator_pixmap;
    GtkWidget *appbar;
    GtkWidget *progress;
    gsize length;
    const gchar *selected_station, *selected_station_uri, *
        selected_station_name, *selected_station_location, *
        selected_station_description;
    GNOMEInternetRadioLocatorData *gnome_internet_radio_locator_data =
        g_new0(GNOMEInternetRadioLocatorData, 1);
    char *pmf;
    gtk_window_set_title(GTK_WINDOW(gnome_internet_radio_locator_app),
        "GNOME␣Internet␣Radio␣Locator");
    gnome_internet_radio_locator = gnome_internet_radio_locator_data;
    gnome_internet_radio_locator_data->settings = g_settings_new(
        GNOME_INTERNET_RADIO_LOCATOR_UI);
    selected_station_uri = g_variant_get_string(g_settings_get_value (
        gnome_internet_radio_locator_data->settings, "
        selected_station_uri"), &length);
    selected_station = g_variant_get_string(g_settings_get_value(
        gnome_internet_radio_locator_data->settings, "station"), &
        length);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG ("SELECTED␣STATION␣=␣%s\n",
        selected_station);
```

```
    /* gnome_internet_radio_locator->selected_station_uri =
        selected_station_uri; */
    selected_station_name = g_variant_get_string(g_settings_get_value (
        gnome_internet_radio_locator_data->settings, "
        selected_station_name"), &length);
    /* gnome_internet_radio_locator->selected_station_name =
        selected_station_name; */
    selected_station_location = g_variant_get_string(
        g_settings_get_value (gnome_internet_radio_locator_data->
        settings, "selected_station_location"), &length);
    /* gnome_internet_radio_locator->selected_station_location =
        selected_station_location; */
    selected_station_description = g_variant_get_string(
        g_settings_get_value (gnome_internet_radio_locator_data->
        settings, "selected_station_description"), &length);
    /* gnome_internet_radio_locator->selected_station_description =
        selected_station_description; */
#if GNOME_INTERNET_RADIO_LOCATOR_CFG_GNOME_CONFIG
    gnome_config_push_prefix("/gnome-internet-radio-locator/General/");
    gnome_internet_radio_locator->selected_listener_uri =
        gnome_config_get_string("selected_listener_uri=");
    gnome_internet_radio_locator->selected_listener_name =
        gnome_config_get_string("selected_listener_name=");
    gnome_internet_radio_locator->selected_listener_location =
        gnome_config_get_string("selected_listener_location=");
    gnome_internet_radio_locator->selected_listener_description =
        gnome_config_get_string("selected_listener_description=");
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_listener_uri:␣%s\n",
           gnome_internet_radio_locator->selected_listener_uri);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_listener_name:␣%s\n",
           gnome_internet_radio_locator->selected_listener_name);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_listener_location:␣%s\n"
        ,
           gnome_internet_radio_locator->selected_listener_location);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_listener_band:␣%s\n",
           gnome_internet_radio_locator->selected_listener_band);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_listener_description:␣%s
        \n",
           gnome_internet_radio_locator->selected_listener_description)
               ;
    gnome_internet_radio_locator->selected_station_uri =
        gnome_config_get_string("selected_station_uri=");
    gnome_internet_radio_locator->selected_station_name =
        gnome_config_get_string("selected_station_name=");
    gnome_internet_radio_locator->selected_station_location =
        gnome_config_get_string("selected_station_location=");
    gnome_internet_radio_locator->selected_station_description =
        gnome_config_get_string("selected_station_description=");
    gnome_internet_radio_locator->selected_station_name =
        gnome_config_get_string("selected_station_name=");
    gnome_internet_radio_locator->selected_station_location =
        gnome_config_get_string("selected_station_location=");
    gnome_internet_radio_locator->selected_station_description =
        gnome_config_get_string("selected_station_description=");
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_station_uri:␣%s\n",
           gnome_internet_radio_locator->selected_station_uri);
```

```
GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
    gnome_internet_radio_locator->selected_station_name:␣%s\n",
       gnome_internet_radio_locator->selected_station_name);
GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
    gnome_internet_radio_locator->selected_station_location:␣%s\n",
       gnome_internet_radio_locator->selected_station_location);
GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
    gnome_internet_radio_locator->selected_station_description:␣%s\
    n",
       gnome_internet_radio_locator->selected_station_description);
if (strcmp(gnome_internet_radio_locator->selected_station_uri,"")
    ==0) {
    gnome_internet_radio_locator->selected_station_uri = g_strdup(_
        ("http://fm939.wnyc.org/wnycfm"));
}
if (strcmp(gnome_internet_radio_locator->selected_station_name,"")
    ==0) {
    gnome_internet_radio_locator->selected_station_name = g_strdup(
        _("WNYC"));
}
if (strcmp(gnome_internet_radio_locator->selected_station_location,
    "")==0) {
    gnome_internet_radio_locator->selected_station_location =
        g_strdup(_("New␣York␣City,␣NY"));
}
if (strcmp(gnome_internet_radio_locator->selected_station_band,"")
    ==0) {
    gnome_internet_radio_locator->selected_station_band = g_strdup(
        _("Online"));
}
if (strcmp(gnome_internet_radio_locator->
    selected_station_description,"")==0) {
    gnome_internet_radio_locator->selected_station_description =
        g_strdup(_("WNYC␣93.9␣FM␣and␣AM␣820␣are␣New␣York's␣flagship
        ␣public␣radio␣stations,␣broadcasting␣the␣finest␣programs␣
        from␣NPR,␣American␣Public␣Media,␣Public␣Radio␣International
        ␣and␣the␣BBC␣World␣Service,␣as␣well␣as␣a␣wide␣range␣of␣
        award-winning␣local␣programming."));
}
gnome_internet_radio_locator->selected_streams_uri =
    gnome_config_get_string("selected_streams_uri=");
gnome_internet_radio_locator->selected_streams_mime =
    gnome_config_get_string("selected_streams_mime=");
gnome_internet_radio_locator->selected_streams_codec =
    gnome_config_get_string("selected_streams_codec=");
gnome_internet_radio_locator->selected_streams_bitrate =
    gnome_config_get_string("selected_streams_bitrate=");
gnome_internet_radio_locator->selected_streams_samplerate =
    gnome_config_get_string("selected_streams_samplerate=");
gnome_internet_radio_locator->selected_streams_channels =
    (Gnome_Internet_Radio_LocatorChannels)gnome_config_get_string("
        selected_streams_channels=");
GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
    gnome_internet_radio_locator->selected_streams_uri:␣%s\n",
       gnome_internet_radio_locator->selected_streams_uri);
GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
    gnome_internet_radio_locator->selected_streams_mime:␣%s\n",
       gnome_internet_radio_locator->selected_streams_mime);
GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
    gnome_internet_radio_locator->selected_streams_codec:␣%s\n",
       gnome_internet_radio_locator->selected_streams_codec);
```

```
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_streams_bitrate:␣%s\n",
           gnome_internet_radio_locator->selected_streams_bitrate);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_streams_samplerate:␣%s\n
        ",
           gnome_internet_radio_locator->selected_streams_samplerate);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        gnome_internet_radio_locator->selected_channels:␣%0x\n",
           gnome_internet_radio_locator->selected_streams_channels);
    gnome_config_pop_prefix();
#endif
    return gnome_internet_radio_locator_app;
}
```

36.3 gnome-internet-radio-locator-listener.c

36.4 gnome-internet-radio-locator-markers.c

```
/* $id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses
     />.
 */

/*
 * Copyright (C) 2008 Pierre-Luc Beaudoin <pierre-luc@pierlux.com>
 *
```

```
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
    02110-1301  USA
 */

#include <string.h>
#include <gtk/gtk.h>
#include <gst/player/player.h>
#include <champlain/champlain.h>
#include "gnome-internet-radio-locator.h"
#include "gnome-internet-radio-locator-markers.h"
#include "gnome-internet-radio-locator-player.h"

extern GtkWidget *statusbar;
extern GtkWidget *input;
extern GtkEntryCompletion *completion;
extern GNOMEInternetRadioLocatorStationInfo *stationinfo, *localstation
    ;
extern gchar *world_station_xml_filename;
extern GstPlayer *player;
extern ChamplainMarkerLayer *layer;
extern GtkWidget *window;
extern ChamplainView *view;

typedef struct
{
    ChamplainView *view;
    ChamplainMarker *marker;
} LocationCallbackData;

static gboolean
location_callback (LocationCallbackData *data)
{
    /* champlain_view_center_on (data->view, lat, lon); */
    /* champlain_location_set_location (CHAMPLAIN_LOCATION (data->
        marker), lat, lon); */
    g_print("%s\n", __FUNCTION__);
    return TRUE;
}

void
marker_function (ChamplainMarker *self,
         gdouble          dx,
         gdouble          dy,
         ClutterEvent    *event,
         gpointer         user_data)
{
        gchar *station, *station_link;
    gchar *markup;
    guint context_id;
```

```
    station = (gchar *)champlain_label_get_text (CHAMPLAIN_LABEL (self)
        );
    station_link = strtok(station, "\n");
    gtk_entry_set_text(GTK_ENTRY(input),(gchar *)station_link);
    gst_player_stop(player);
    player = gst_player_new (NULL,
        gst_player_g_main_context_signal_dispatcher_new(NULL));
    stationinfo = gnome_internet_radio_locator_station_load_from_file(
        localstation, world_station_xml_filename);
    while (stationinfo != NULL) {
             if (strcasecmp(stationinfo->location, station_link)==0) {
            gchar *statusmsg = g_strconcat(stationinfo->name, "␣(",
                stationinfo->uri, ")␣in␣", stationinfo->location, "␣(",
                 stationinfo->band, ",␣", g_strdup_printf("%li",
                stationinfo->stream->samplerate), "␣Hz,␣",
                g_strdup_printf("%li", stationinfo->stream->bitrate), "
                ␣kbps)", NULL);
            gnome_internet_radio_locator_player_new(GST_PLAYER(player),
                 stationinfo->stream->uri);
            context_id = gtk_statusbar_get_context_id (GTK_STATUSBAR (
                statusbar), "Station␣Name");
            gtk_statusbar_pop (GTK_STATUSBAR (statusbar),
                GPOINTER_TO_INT (context_id));
            gtk_statusbar_push (GTK_STATUSBAR (statusbar),
                GPOINTER_TO_INT (context_id), statusmsg);
        }
        stationinfo = stationinfo->next;
    }
    gst_player_play(player);
    if (user_data != NULL) {
        champlain_label_set_text (CHAMPLAIN_LABEL (self), user_data);
    }
    /* g_signal_connect(CHAMPLAIN_LOCATION(self), "button-press",
        G_CALLBACK (on_new_station_clicked), user_data); */
    return;
}

ChamplainMarkerLayer *
create_marker_layer (G_GNUC_UNUSED ChamplainView *view,
    ChamplainPathLayer **path)
{
    ClutterActor *marker;
    ClutterActor *layer_actor;
    ClutterColor city_color = { 0xf3, 0x94, 0x07, 0xbb };
        ClutterColor city_a_color = { 0x83, 0xa6, 0x7f, 0xbb };
        ClutterColor city_b_color = { 0xc1, 0x66, 0x5a, 0xbb };
        ClutterColor city_c_color = { 0x88, 0x7f, 0xa3, 0xbb };
        ClutterColor city_d_color = { 0x34, 0x65, 0xa4, 0xbb };
        ClutterColor city_e_color = { 0x75, 0x90, 0xae, 0xbb };
        ClutterColor city_f_color = { 0xe0, 0xc3, 0x93, 0xbb };
    ClutterColor text_color = { 0xff, 0xff, 0xff, 0xff };
    LocationCallbackData callback_data;
    gchar *station = NULL;
    *path = champlain_path_layer_new ();
    layer = champlain_marker_layer_new_full (CHAMPLAIN_SELECTION_SINGLE
        );
    layer_actor = CLUTTER_ACTOR (layer);
    /* Create callback that updates the map periodically */
    /* callback_data.view = CHAMPLAIN_VIEW (view); */
    /* callback_data.marker = CHAMPLAIN_MARKER (layer); */
    /* g_timeout_add (1000, (GSourceFunc) location_callback, &
        callback_data); */
```

```
    /* marker = champlain_point_new (); */
    /* champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        37.873093, -122.303769); */
    /* champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        37.873093, -122.303769); */
    /* champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (
        marker)); */
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
#if 0
    marker = champlain_label_new_with_text ("Norway\n<span␣size=\"small
        \">Oslo</span>", "Helvetica␣14", NULL, NULL);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_alignment (CHAMPLAIN_LABEL (marker),
        PANGO_ALIGN_RIGHT);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        37.873093, -122.303769);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker));
#endif
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("Aalborg,␣Denmark\n<span␣size=\"small\">ANR</
        span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        57.0482206, 9.9193939);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
    g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
        G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("Alta,␣Norway\n<span␣size=\"small\">Radio␣Alta</
        span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        70.04962755, 23.0825349565332);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
```

```
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Ayr,␣Scotland\n<span␣size=\"small\">UWS␣Radio</
    span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    55.4594119, -4.6326702);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Barcelona,␣Catalonia\n<span␣size=\"small\">
    Barcelona␣City␣FM</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    41.3828939, 2.1774322);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Bergen,␣Norway\n<span␣size=\"small\">SRIB</span
    >");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    60.3943034, 5.3258117);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
```

```
    station = g_strdup("Berkeley,␣California\n<span␣size=\"small\">KALX
        </span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        37.873093, -122.303769);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
    g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
        G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("Berlin,␣Germany\n<span␣size=\"small\">Radio␣
        Eins</span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        52.5170365, 13.3888599);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
    g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
        G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("Bern,␣Switzerland\n<span␣size=\"small\">Radio␣
        NRJ␣Bern</span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        46.9482713, 7.4514512);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
    g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
        G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("WBUR,␣Boston,␣Massachusetts\n<span␣size=\"small
        \">WBUR</span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
```

```
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    42.3516603, -71.1226348);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("WTBU,␣Boston,␣Massachusetts\n<span␣size=\"small
    \">WTBU</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    42.3602534, -71.0582912);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Bruxelles,␣Belgium\n<span␣size=\"small\">Radio␣
    Campus</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    50.84404145, 4.36720169448285);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Buskerud,␣Norway\n<span␣size=\"small\">NRK␣P1␣
    Buskerud</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
```

```
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    60.2497876, 8.96278676790604);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Calgary,␣Canada\n<span␣size=\"small\">CJSW</
    span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    51.0534234, -114.0625892);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("WHRB-FM,␣Cambridge,␣Massachusetts\n<span␣size
    =\"small\">WHRB</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    42.3723191, -71.1186638);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("WMBR-FM,␣Cambridge,␣Massachusetts\n<span␣size
    =\"small\">WMBR</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    42.3617430, -71.0839082);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
```

```
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Cambridge,␣United␣Kingdom\n<span␣size=\"small
    \">Cam␣FM</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    52.2033051, 0.124862);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Cape␣Town,␣South␣Africa\n<span␣size=\"small\">
    UCT␣Radio</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    -33.928992, 18.417396);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Copenhagen,␣Denmark\n<span␣size=\"small\">
    Danmarks␣Radio␣Nyheder</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    55.6867243, 12.5700724);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
```

```
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Chicago,␣Illinois\n<span␣size=\"small\">WHPK</
    span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    41.8755546, -87.6244212);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Dubai,␣Saudi␣Arabia\n<span␣size=\"small\">
    Middle␣East␣Broadcasting␣Center␣FM</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    21.4146056, 39.8227432);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
/* champlain_path_layer_add_node (path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Dublin,␣Ireland\n<span␣size=\"small\">DCUfm</
    span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    53.3497645, -6.2602732);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
/* champlain_path_layer_add_node (path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
```

```
station = g_strdup("Finnmark, Norway\n<span size=\"small\">NRK P1
    Finnmark</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    69.86993469999999, 21.827278548625912);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.
        png", NULL);
station = g_strdup("Guatemala City, Guatemala\n<span size=\"small
    \">Radio Universidad</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    14.6417889, -90.5132239);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Honolulu, Hawaii\n<span size=\"small\">Hawaii
    Public Radio</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    21.304547, -157.8556764);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Hordaland, Norway\n<span size=\"small\">NRK P1
    Hordaland</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
```

```
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    60.2570766, 6.06249778348651);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("K r  johka,␣Norway\n<span␣size=\"small\">NRK
    ␣S pmi</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    68.7718259, 24.2803624);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Kingston,␣Canada\n<span␣size=\"small\">CFRC</
    span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    44.230687, -76.481323);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Kristiansand,␣Norway\n<span␣size=\"small\">NRK␣
    S rlandet</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
```

```
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    58.14615, 7.9957333);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Houston,␣Texas\n<span␣size=\"small\">Coog␣Radio
    ,␣University␣of␣Houston</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    29.7589382, -95.3676974);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Innlandet,␣Norway\n<span␣size=\"small\">NRK␣P1␣
    Innlandet</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    61.26810555, 10.485458802304304);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.
        png", NULL);
station = g_strdup("London,␣United␣Kingdom\n<span␣size=\"small\">
    Imperial␣College␣Radio</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    51.5073219, -0.1276474);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
```

```
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Los␣Angeles,␣California\n<span␣size=\"small\">
    KXSC</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    34.1430079, -118.14176172581);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Lyon,␣France\n<span␣size=\"small\">Radio␣Brume
    </span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    45.7544734, 4.8122242);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("M xico␣City,␣M xico\n<span␣size=\"small\">
    Radio␣UNAM</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    19.647012, -101.22900565);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
```

```
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Moscow,␣Russia\n<span␣size=\"small\">Echo␣of␣
    Moscow</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    55.4792046, 37.3273304);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("M re␣og␣Romsdal,␣Norway\n<span␣size=\"small\">
    NRK␣P1␣M re␣og␣Romsdal</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    62.8452777, 7.51819407263736);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Newcastle,␣Australia\n<span␣size=\"small\">2
    NURFM</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    -32.9272881, 151.7812534);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("New␣Orleans,␣Louisiana\n<span␣size=\"small\">
    WWNO</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
```

```
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    29.9499323, -90.0701156);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("New␣York␣City,␣New␣York\n<span␣size=\"small\">
    WKCR</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    40.7306458, -73.9866136);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Nordkapp,␣Norway\n<span␣size=\"small\">Radio␣
    Nordkapp</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    71.1699506, 25.7858893);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Nordland,␣Norway\n<span␣size=\"small\">NRK␣P1␣
    Nordland</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
```

```
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    67.27564050000001, 13.862360639913621);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.
        png", NULL);
station = g_strdup("Norway\n<span␣size=\"small\">Radio␣Norwegian</
    span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    60.133054099999995, 7.531103018917516);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Ontario,␣Canada\n<span␣size=\"small\">
    RadioWaterloo</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    43.466874, -80.524635);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Oslo,␣Norway\n<span␣size=\"small\">NRK␣Sport</
    span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    59.9132694, 10.7391112);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
```

```
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Oxford,␣United␣Kingdom\n<span␣size=\"small\">
    Oxide␣Radio</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    51.7520131, -1.2578499);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Ottawa,␣Canada\n<span␣size=\"small\">CHUO</span
    >");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    45.421106, -75.690308);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Paris,␣France\n<span␣size=\"small\">Radio␣
    Campus␣Paris</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    48.8566101, 2.3514992);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
```

```
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("Phoenix,␣Arizona\n<span␣size=\"small\">KASC</
        span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        33.4485866, -112.0773456);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
    g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
        G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("Pisa,␣Italy\n<span␣size=\"small\">Radio␣Eco</
        span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        43.7159395, 10.4018624);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
    g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
        G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("Portland,␣Oregon\n<span␣size=\"small\">KPSU</
        span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
    champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        45.5202471,-122.6741949);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
    g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
        G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
    station = g_strdup("Rogaland,␣Norway\n<span␣size=\"small\">NRK␣P1␣
        Rogaland</span>");
    champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
```

```
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    58.93631375, 5.80587864304024);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("San␣Francisco,␣California\n<span␣size=\"small
    \">SomaFM</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    37.7792808, -122.4192363);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Seattle,␣Washington\n<span␣size=\"small\">KSUB
    </span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    47.6038321, -122.3300624);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Sogn␣og␣Fjordane,␣Norway\n<span␣size=\"small\">
    NRK␣P1␣Sogn␣og␣Fjordane</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
```

```
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    61.53093745, 6.10242908066871);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Stanford,␣California\n<span␣size=\"small\">KZSU
    </span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    37.4248398, -122.1677058);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Stockholm,␣Sweden\n<span␣size=\"small\">
    Sveriges␣Radio␣P1</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    59.3251172, 18.0710935);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Sydney,␣Canada\n<span␣size=\"small\">Caper␣
    Radio</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    46.1654766, -60.1735637935906);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
```

```
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Tampere,␣Finland\n<span␣size=\"small\">Radio␣
    Moreeni</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    61.4980214, 23.7603118);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Thessaloniki,␣Greece\n<span␣size=\"small\">1431
    ␣AM</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    40.6403167, 22.9352716);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Toronto,␣Canada\n<span␣size=\"small\">CIUT</
    span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    43.653963, -79.387207);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
```

```
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("T rshavn,␣Faroe␣Islands\n<span␣size=\"small\">
    Midlar</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    62.012, -6.768);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Troms,␣Norway\n<span␣size=\"small\">NRK␣P1␣
    Troms</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    64.5731537, 11.52803643954819);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.
        png", NULL);
station = g_strdup("Trondheim,␣Norway\n<span␣size=\"small\">Radio␣
    Revolt</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    63.4305658, 10.3951929);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.
        png", NULL);
station = g_strdup("Tr ndelag,␣Norway\n<span␣size=\"small\">NRK␣P1
    ␣Tr ndelag</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
```

```
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    63.87759235, 10.195050547141093);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.
        png", NULL);
station = g_strdup("Warsaw,␣Poland\n<span␣size=\"small\">Radio␣
    Aktywne</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_b_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    52.2319237, 21.0067265);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup("Washington,␣District␣of␣Columbia\n<span␣size=\"
    small\">WAMU</span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_c_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
    38.8949549, -77.0366456);
champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
    );
champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
/* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
    )); */
g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
    G_CALLBACK(marker_function), station);
marker = champlain_label_new_from_file ("icons/emblem-generic.png",
     NULL);
station = g_strdup(" stfold ,␣Norway\n<span␣size=\"small\">NRK␣P1␣
     stfold </span>");
champlain_label_set_text (CHAMPLAIN_LABEL (marker), station);
champlain_label_set_use_markup (CHAMPLAIN_LABEL (marker), TRUE);
champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_d_color)
    ;
champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
    text_color);
```

```
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        59.20313315, 10.9535037948529);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_marker_animate_in(CHAMPLAIN_MARKER (marker));
    /* champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker
        )); */
    g_signal_connect(CHAMPLAIN_LOCATION(marker), "button-press",
        G_CALLBACK(marker_function), station);
    marker = champlain_label_new_from_file ("icons/emblem-generic.png",
         NULL);
#if 0
    marker = champlain_label_new_from_file ("icons/emblem-important.png
        ", NULL);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_e_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        37.873093, -122.303769);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker));
    marker = champlain_point_new ();
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_f_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        37.873093, -122.303769);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker));
    marker = champlain_label_new_from_file ("icons/emblem-favorite.png"
        , NULL);
    champlain_label_set_draw_background (CHAMPLAIN_LABEL (marker),
        FALSE);
    champlain_label_set_color (CHAMPLAIN_LABEL (marker), &city_a_color)
        ;
    champlain_label_set_text_color (CHAMPLAIN_LABEL (marker), &
        text_color);
    champlain_location_set_location (CHAMPLAIN_LOCATION (marker),
        37.873093, -122.303769);
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (marker)
        );
    champlain_path_layer_add_node (*path, CHAMPLAIN_LOCATION (marker));
#endif
    champlain_marker_layer_set_all_markers_draggable (layer);
    clutter_actor_show (layer_actor);
    return layer;
}
```

36.5 gnome-internet-radio-locator-player.c

```
/* GStreamer command line playback testing utility
 *
 * Copyright (C) 2013-2014 Tim-Philipp M ller <tim centricular net>
 * Copyright (C) 2013 Collabora Ltd.
 * Copyright (C) 2014 Sebastian Dr ge <sebastian@centricular.com>
 * Copyright (C) 2015 Brijesh Singh <brijesh.ksingh@gmail.com>
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the
 * Free Software Foundation, Inc., 51 Franklin St, Fifth Floor,
 * Boston, MA 02110-1301, USA.
 */

#include <locale.h>

#include <gst/gst.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

#include <gst/player/player.h>
#include "gnome-internet-radio-locator.h"
#include "gnome-internet-radio-locator-player.h"
#include "gnome-internet-radio-locator-player-kb.h"

#define VOLUME_STEPS 20

GST_DEBUG_CATEGORY (play_debug);
#define GST_CAT_DEFAULT play_debug

extern GstPlay *player;
#if 0
static gboolean play_next (GstPlay * play);
static gboolean play_prev (GstPlay * play);
static void play_reset (GstPlay * play);
static void play_set_relative_volume (GstPlay * play, gdouble
    volume_step);
#endif

void
gnome_internet_radio_locator_player_main(gchar *streamuri, gchar *name)
{

    gchar *uri = g_strdup(streamuri);

    /* if (argc > 1) { */
    /*   if (g_strrstr (argv[1],"rtsp://") || g_strrstr (argv[1],"http
        ://") || g_strrstr (argv[1],"file://")) */
    /*     uri = g_strdup (argv[1]); */
    /*   else */
    /*     uri = g_strdup_printf ("file://%s", argv[1]); */
```

```
    /* } */
    /* else { */
    /*   g_message ("Specify the media file name..<
        gnome_internet_radio_locator-playerplayer filename>"); */
    /*   return 0; */
    /* } */

    gnome_internet_radio_locator->player_loop = g_main_loop_new (NULL,
        FALSE);
    g_main_loop_run (gnome_internet_radio_locator->player_loop);
    gst_deinit();
    gnome_internet_radio_locator->player_status =
        GNOME_INTERNET_RADIO_LOCATOR_PLAYER_TRUE;
    exit (0);
}

#if 0
static void
end_of_stream_cb (GstPlayer * player, GstPlay * play)
{
    g_print ("\n");
    /* and switch to next item in list */
    if (!play_next (play)) {
        g_print ("Reached␣end␣of␣play␣list.\n");
        g_main_loop_quit (play->loop);
    }
}
#endif
#if 0
static void
error_cb (GstPlayer * player, GError * err, GstPlay * play)
{
    g_printerr ("ERROR␣%s␣for␣%s\n", err->message, play->uris[play->
        cur_idx]);
    /* if looping is enabled, then disable it else will keep looping
        forever */
    play->repeat = FALSE;
    /* try next item in list then */
    if (!play_next (play)) {
        g_print ("Reached␣end␣of␣play␣list.\n");
        g_main_loop_quit (play->loop);
    }
}
#endif
static void
position_updated_cb (GstPlayer * player, GstClockTime pos, GstPlay *
    play)
{
    GstClockTime dur = -1;
    gchar status[64] = { 0, };
    g_object_get (play->player, "duration", &dur, NULL);
    memset (status, '␣', sizeof (status) - 1);
    if (pos != -1 && dur > 0 && dur != -1) {
        gchar dstr[32], pstr[32];
        /* FIXME: pretty print in nicer format */
        g_snprintf (pstr, 32, "%" GST_TIME_FORMAT, GST_TIME_ARGS (pos))
            ;
        pstr[9] = '\0';
        g_snprintf (dstr, 32, "%" GST_TIME_FORMAT, GST_TIME_ARGS (dur))
            ;
        dstr[9] = '\0';
        g_print ("%s␣/␣%s␣%s\r", pstr, dstr, status);
```

```
    }
}

static void
state_changed_cb (GstPlayer * player, GstPlayerState state, GstPlay *
    play)
{
    g_print ("State␣changed:␣%s\n", gst_player_state_get_name (state));
}

static void
buffering_cb (GstPlayer * player, gint percent, GstPlay * play)
{
    g_print ("Buffering:␣%d\n", percent);
}

static void
print_one_tag (const GstTagList * list, const gchar * tag, gpointer
    user_data)
{
    gint i, num;
    num = gst_tag_list_get_tag_size (list, tag);
    for (i = 0; i < num; ++i) {
        const GValue *val;
        val = gst_tag_list_get_value_index (list, tag, i);
        if (G_VALUE_HOLDS_STRING (val)) {
            g_print ("␣␣␣␣%s␣:␣%s␣\n", tag, g_value_get_string (val));
        } else if (G_VALUE_HOLDS_UINT (val)) {
            g_print ("␣␣␣␣%s␣:␣%u␣\n", tag, g_value_get_uint (val));
        } else if (G_VALUE_HOLDS_DOUBLE (val)) {
            g_print ("␣␣␣␣%s␣:␣%g␣\n", tag, g_value_get_double (val));
        } else if (G_VALUE_HOLDS_BOOLEAN (val)) {
            g_print ("␣␣␣␣%s␣:␣%s␣\n", tag,
                 g_value_get_boolean (val) ? "true" : "false");
        } else if (GST_VALUE_HOLDS_DATE_TIME (val)) {
            GstDateTime *dt = g_value_get_boxed (val);
            gchar *dt_str = gst_date_time_to_iso8601_string (dt);
            g_print ("␣␣␣␣%s␣:␣%s␣\n", tag, dt_str);
            g_free (dt_str);
        } else {
            g_print ("␣␣␣␣%s␣:␣tag␣of␣type␣'%s'␣\n", tag,
                G_VALUE_TYPE_NAME (val));
        }
    }
}

static void
print_video_info (GstPlayerVideoInfo * info)
{
    gint fps_n, fps_d;
    guint par_n, par_d;
    if (info == NULL)
        return;
    g_print ("␣␣width␣:␣%d\n", gst_player_video_info_get_width (info));
    g_print ("␣␣height␣:␣%d\n", gst_player_video_info_get_height (info)
        );
    g_print ("␣␣max_bitrate␣:␣%d\n",
         gst_player_video_info_get_max_bitrate (info));
    g_print ("␣␣bitrate␣:␣%d\n", gst_player_video_info_get_bitrate (
        info));
    gst_player_video_info_get_framerate (info, &fps_n, &fps_d);
    g_print ("␣␣framerate␣:␣%.2f\n", (gdouble) fps_n / fps_d);
```

```
    gst_player_video_info_get_pixel_aspect_ratio (info, &par_n, &par_d)
        ;
    g_print ("␣␣pixel-aspect-ratio␣␣%u:%u\n", par_n, par_d);
}

static void
print_audio_info (GstPlayerAudioInfo * info)
{
    if (info == NULL)
        return;
    g_print ("␣␣sample␣rate␣:␣%d\n",
         gst_player_audio_info_get_sample_rate (info));
    g_print ("␣␣channels␣:␣%d\n", gst_player_audio_info_get_channels (
        info));
    g_print ("␣␣max_bitrate␣:␣%d\n",
         gst_player_audio_info_get_max_bitrate (info));
    g_print ("␣␣bitrate␣:␣%d\n", gst_player_audio_info_get_bitrate (
        info));
    g_print ("␣␣language␣:␣%s\n", gst_player_audio_info_get_language (
        info));
}

static void
print_subtitle_info (GstPlayerSubtitleInfo * info)
{
    if (info == NULL)
        return;
    g_print ("␣␣language␣:␣%s\n", gst_player_subtitle_info_get_language
         (info));
}

static void
print_all_stream_info (GstPlayerMediaInfo * media_info)
{
    guint count = 0;
    GList *list, *l;
    g_print ("URI␣:␣%s\n", gst_player_media_info_get_uri (media_info));
    g_print ("Duration:␣%" GST_TIME_FORMAT "\n",
         GST_TIME_ARGS (gst_player_media_info_get_duration (media_info)
             ));
    g_print ("Global␣taglist:\n");
    if (gst_player_media_info_get_tags (media_info))
        gst_tag_list_foreach (gst_player_media_info_get_tags (
            media_info),
                      print_one_tag, NULL);
    else
        g_print ("␣␣(nil)␣\n");
    list = gst_player_media_info_get_stream_list (media_info);
    if (!list)
        return;
    g_print ("All␣Stream␣information\n");
    for (l = list; l != NULL; l = l->next) {
        GstTagList *tags = NULL;
        GstPlayerStreamInfo *stream = (GstPlayerStreamInfo *) l->data;
        g_print ("␣Stream␣#␣%u␣\n", count++);
        g_print ("␣␣type␣:␣%s_%u\n",
             gst_player_stream_info_get_stream_type (stream),
             gst_player_stream_info_get_index (stream));
        tags = gst_player_stream_info_get_tags (stream);
        g_print ("␣␣taglist␣:␣\n");
        if (tags) {
            gst_tag_list_foreach (tags, print_one_tag, NULL);
```

```
        }
        if (GST_IS_PLAYER_VIDEO_INFO (stream))
            print_video_info ((GstPlayerVideoInfo *) stream);
        else if (GST_IS_PLAYER_AUDIO_INFO (stream))
            print_audio_info ((GstPlayerAudioInfo *) stream);
        else
            print_subtitle_info ((GstPlayerSubtitleInfo *) stream);
    }
}

static void
print_all_video_stream (GstPlayerMediaInfo * media_info)
{
    GList *list, *l;
    list = gst_player_get_video_streams (media_info);
    if (!list)
        return;
    g_print ("All␣video␣streams\n");
    for (l = list; l != NULL; l = l->next) {
        GstPlayerVideoInfo *info = (GstPlayerVideoInfo *) l->data;
        GstPlayerStreamInfo *sinfo = (GstPlayerStreamInfo *) info;
        g_print ("␣%s_%d␣#\n", gst_player_stream_info_get_stream_type (
            sinfo),
             gst_player_stream_info_get_index (sinfo));
        print_video_info (info);
    }
}

static void
print_all_subtitle_stream (GstPlayerMediaInfo * media_info)
{
    GList *list, *l;
    list = gst_player_get_subtitle_streams (media_info);
    if (!list)
        return;
    g_print ("All␣subtitle␣streams:\n");
    for (l = list; l != NULL; l = l->next) {
        GstPlayerSubtitleInfo *info = (GstPlayerSubtitleInfo *) l->data
            ;
        GstPlayerStreamInfo *sinfo = (GstPlayerStreamInfo *) info;
        g_print ("␣%s_%d␣#\n", gst_player_stream_info_get_stream_type (
            sinfo),
             gst_player_stream_info_get_index (sinfo));
        print_subtitle_info (info);
    }
}

static void
print_all_audio_stream (GstPlayerMediaInfo * media_info)
{
    GList *list, *l;
    list = gst_player_get_audio_streams (media_info);
    if (!list)
        return;
    g_print ("All␣audio␣streams:␣\n");
    for (l = list; l != NULL; l = l->next) {
        GstPlayerAudioInfo *info = (GstPlayerAudioInfo *) l->data;
        GstPlayerStreamInfo *sinfo = (GstPlayerStreamInfo *) info;
        g_print ("␣%s_%d␣#\n", gst_player_stream_info_get_stream_type (
            sinfo),
             gst_player_stream_info_get_index (sinfo));
        print_audio_info (info);
```

```
    }
}

static void
print_current_tracks (GstPlay * play)
{
    GstPlayerAudioInfo *audio = NULL;
    GstPlayerVideoInfo *video = NULL;
    GstPlayerSubtitleInfo *subtitle = NULL;
    g_print ("Current␣video␣track:␣\n");
    video = gst_player_get_current_video_track (play->player);
    print_video_info (video);
    g_print ("Current␣audio␣track:␣\n");
    audio = gst_player_get_current_audio_track (play->player);
    print_audio_info (audio);
    g_print ("Current␣subtitle␣track:␣\n");
    subtitle = gst_player_get_current_subtitle_track (play->player);
    print_subtitle_info (subtitle);
    if (audio)
        g_object_unref (audio);
    if (video)
        g_object_unref (video);
    if (subtitle)
        g_object_unref (subtitle);
}

static void
print_media_info (GstPlayerMediaInfo * media_info)
{
    print_all_stream_info (media_info);
    g_print ("\n");
    print_all_video_stream (media_info);
    g_print ("\n");
    print_all_audio_stream (media_info);
    g_print ("\n");
    print_all_subtitle_stream (media_info);
}

static void
media_info_cb (GstPlayer * player, GstPlayerMediaInfo * info, GstPlay *
     play)
{
    static int once = 0;
    if (!once) {
        print_media_info (info);
        print_current_tracks (play);
        once = 1;
    }
}

#if 0
static GstPlay *
play_new (gchar ** uris, gdouble initial_volume)
{
    GstPlay *play;
    play = g_new0 (GstPlay, 1);
    play->uris = uris;
    play->num_uris = g_strv_length (uris);
    play->cur_idx = -1;
    play->player =
        gst_player_new (NULL,
            gst_player_g_main_context_signal_dispatcher_new
```

```
                (NULL));
    g_signal_connect (play->player, "position-updated",
              G_CALLBACK (position_updated_cb), play);
    g_signal_connect (play->player, "state-changed",
              G_CALLBACK (state_changed_cb), play);
    g_signal_connect (play->player, "buffering", G_CALLBACK (
        buffering_cb), play);
    g_signal_connect (play->player, "end-of-stream",
              G_CALLBACK (end_of_stream_cb), play);
    g_signal_connect (play->player, "error", G_CALLBACK (error_cb),
        play);
    g_signal_connect (play->player, "media-info-updated",
              G_CALLBACK (media_info_cb), play);
    play->loop = g_main_loop_new (NULL, FALSE);
    play->desired_state = GST_STATE_PLAYING;
    play_set_relative_volume (play, initial_volume - 1.0);
    return play;
}

static void
play_free (GstPlay * play)
{
    play_reset (play);
    gst_object_unref (play->player);
    g_main_loop_unref (play->loop);
    g_strfreev (play->uris);
    g_free (play);
}
#endif
/* reset for new file/stream */
static void
play_reset (GstPlay * play)
{
}

static void
play_set_relative_volume (GstPlay * play, gdouble volume_step)
{
    gdouble volume;
    g_object_get (play->player, "volume", &volume, NULL);
    volume = round ((volume + volume_step) * VOLUME_STEPS) /
        VOLUME_STEPS;
    volume = CLAMP (volume, 0.0, 10.0);
    g_object_set (play->player, "volume", volume, NULL);
    g_print ("Volume:␣%.0f%%␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣\n", volume * 100);
}

static gdouble
get_volume (GtkWidget *widget, GstPlay *play)
{
    gdouble volume;
    g_object_get (player, "volume", &volume, NULL);
    return volume;
}

#if 0
static gchar *
play_uri_get_display_name (GstPlayer * player, const gchar * uri)
{
    gchar *loc;
    if (gst_uri_has_protocol (uri, "file")) {
    loc = g_filename_from_uri (uri, NULL, NULL);
```

```
    } else if (gst_uri_has_protocol (uri, "pushfile")) {
        loc = g_filename_from_uri (uri + 4, NULL, NULL);
    } else {
        loc = g_strdup (uri);
    }
    /* Maybe additionally use glib's filename to display name function
        */
    return loc;
}

void
play_uri (GstPlayer *player, const gchar * next_uri)
{
    gchar *loc;
    play_reset (player);
    loc = play_uri_get_display_name (player, next_uri);
    g_print ("Now␣playing␣%s\n", loc);
    g_free (loc);
    g_object_set (player, "uri", next_uri, NULL);
    gst_player_play (player);
}

/* returns FALSE if we have reached the end of the playlist */
#if 0
static gboolean
play_next (GstPlay * play)
{
    if ((play->cur_idx + 1) >= play->num_uris) {
        if (play->repeat) {
            g_print ("Looping␣playlist␣\n");
            play->cur_idx = -1;
        } else
            return FALSE;
    }
    play_uri (play, play->uris[++play->cur_idx]);
    return TRUE;
}

/* returns FALSE if we have reached the beginning of the playlist */
static gboolean
play_prev (GstPlay * play)
{
    if (play->cur_idx == 0 || play->num_uris <= 1)
        return FALSE;
    play_uri (play, play->uris[--play->cur_idx]);
    return TRUE;
}

static void
do_play (GstPlay * play)
{
    gint i;
    /* dump playlist */
    for (i = 0; i < play->num_uris; ++i)
        GST_INFO ("%4u␣:␣%s", i, play->uris[i]);
    if (!play_next (play))
        return;
    g_main_loop_run (play->loop);
}

static void
add_to_playlist (GPtrArray * playlist, const gchar * filename)
```

```
{
    GDir *dir;
    gchar *uri;
    if (gst_uri_is_valid (filename)) {
        g_ptr_array_add (playlist, g_strdup (filename));
        return;
    }
    if ((dir = g_dir_open (filename, 0, NULL))) {
        const gchar *entry;
        /* FIXME: sort entries for each directory? */
        while ((entry = g_dir_read_name (dir))) {
            gchar *path;
            path = g_strconcat (filename, G_DIR_SEPARATOR_S, entry,
                NULL);
            add_to_playlist (playlist, path);
            g_free (path);
        }
        g_dir_close (dir);
        return;
    }
    uri = gst_filename_to_uri (filename, NULL);
    if (uri != NULL)
        g_ptr_array_add (playlist, uri);
    else
        g_warning ("Could␣not␣make␣URI␣out␣of␣filename␣'%s'", filename)
            ;
}

static void
shuffle_uris (gchar ** uris, guint num)
{
    gchar *tmp;
    guint i, j;
    if (num < 2)
        return;
    for (i = 0; i < num; i++) {
        /* gets equally distributed random number in 0..num-1 [0;num[
            */
        j = g_random_int_range (0, num);
        tmp = uris[j];
        uris[j] = uris[i];
        uris[i] = tmp;
    }
}

static void
restore_terminal (void)
{
    gst_play_kb_set_key_handler (NULL, NULL);
}

static void
toggle_paused (GstPlay * play)
{
    if (play->desired_state == GST_STATE_PLAYING) {
        play->desired_state = GST_STATE_PAUSED;
        gst_player_pause (play->player);
    } else {
        play->desired_state = GST_STATE_PLAYING;
        gst_player_play (play->player);
    }
}
```

```
static void
relative_seek (GstPlay * play, gdouble percent)
{
    gint64 dur = -1, pos = -1;
    g_return_if_fail (percent >= -1.0 && percent <= 1.0);
    g_object_get (play->player, "position", &pos, "duration", &dur,
        NULL);
    if (dur <= 0) {
        g_print ("\nCould␣not␣seek.\n");
        return;
    }
    pos = pos + dur * percent;
    if (pos < 0)
        pos = 0;
    gst_player_seek (play->player, pos);
}

static void
keyboard_cb (const gchar * key_input, gpointer user_data)
{
    GstPlay *play = (GstPlay *) user_data;
    switch (g_ascii_tolower (key_input[0])) {
    case 'i':
    {
        GstPlayerMediaInfo *media_info = gst_player_get_media_info (
            play->player);
        if (media_info) {
            print_media_info (media_info);
            g_object_unref (media_info);
            print_current_tracks (play);
        }
        break;
    }
    case '␣':
        toggle_paused (play);
        break;
    case 'q':
    case 'Q':
        g_main_loop_quit (play->loop);
        break;
    case '>':
        if (!play_next (play)) {
            g_print ("\nReached␣end␣of␣play␣list.\n");
            g_main_loop_quit (play->loop);
        }
        break;
    case '<':
        play_prev (play);
        break;
    case 27:                    /* ESC */
        if (key_input[1] == '\0') {
            g_main_loop_quit (play->loop);
            break;
        }
        /* fall through */
    default:
        if (strcmp (key_input, GST_PLAY_KB_ARROW_RIGHT) == 0) {
            relative_seek (play, +0.08);
        } else if (strcmp (key_input, GST_PLAY_KB_ARROW_LEFT) == 0) {
            relative_seek (play, -0.01);
        } else if (strcmp (key_input, GST_PLAY_KB_ARROW_UP) == 0) {
```

```
            play_set_relative_volume (play, +1.0 / VOLUME_STEPS);
        } else if (strcmp (key_input, GST_PLAY_KB_ARROW_DOWN) == 0) {
            play_set_relative_volume (play, -1.0 / VOLUME_STEPS);
        } else {
            GST_INFO ("keyboard␣input:");
            for (; *key_input != '\0'; ++key_input)
                GST_INFO ("␣␣code␣%3d", *key_input);
        }
        break;
    }
}
#endif
#endif

void
gnome_internet_radio_locator_player_new (GstPlayer * player, const
    gchar * next_uri)
{
    gst_player_set_uri(player, next_uri);
}

void
gnome_internet_radio_locator_player_stop (GstPlayer *player)
{
    if (player != NULL) {
        gst_player_stop(GST_PLAYER(player));
    }
}

void
gnome_internet_radio_locator_player_pause (GstPlayer *player)
{
    /* FIXME: Unable to quit after pause is called for the first time.
        */
#if 0
    gst_player_pause(player);
#endif
}

#if 0
int
main (int argc, char **argv)
{
    GstPlay *play;
    GPtrArray *playlist;
    gboolean print_version = FALSE;
    gboolean interactive = FALSE; /* FIXME: maybe enable by default? */
    gboolean shuffle = FALSE;
    gboolean repeat = FALSE;
    gdouble volume = 1.0;
    gchar **filenames = NULL;
    gchar **uris;
    guint num, i;
    GError *err = NULL;
    GOptionContext *ctx;
    gchar *playlist_file = NULL;
    GOptionEntry options[] = {
        {"version", 0, 0, G_OPTION_ARG_NONE, &print_version,
         "Print␣version␣information␣and␣exit", NULL},
        {"shuffle", 0, 0, G_OPTION_ARG_NONE, &shuffle,
         "Shuffle␣playlist", NULL},
        {"interactive", 0, 0, G_OPTION_ARG_NONE, &interactive,
```

```
      "Interactive control via keyboard", NULL},
    {"volume", 0, 0, G_OPTION_ARG_DOUBLE, &volume,
      "Volume", NULL},
    {"playlist", 0, 0, G_OPTION_ARG_FILENAME, &playlist_file,
      "Playlist file containing input media files", NULL},
    {"loop", 0, 0, G_OPTION_ARG_NONE, &repeat, "Repeat all", NULL},
    {G_OPTION_REMAINING, 0, 0, G_OPTION_ARG_FILENAME_ARRAY, &
        filenames, NULL},
    {NULL}
};
g_set_prgname ("gst-play");
ctx = g_option_context_new ("FILE1|URI1 [FILE2|URI2] [FILE3|URI3] 
    ...");
g_option_context_add_main_entries (ctx, options, NULL);
g_option_context_add_group (ctx, gst_init_get_option_group ());
if (!g_option_context_parse (ctx, &argc, &argv, &err)) {
    g_print ("Error initializing: %s\n", GST_STR_NULL (err->message
        ));
    g_clear_error (&err);
    g_option_context_free (ctx);
    return 1;
}
g_option_context_free (ctx);
GST_DEBUG_CATEGORY_INIT (play_debug, "play", 0, "gst-play");
if (print_version) {
    gchar *version_str;
    version_str = gst_version_string ();
    g_print ("%s version %s\n", g_get_prgname (), "1.0");
    g_print ("%s\n", version_str);
    g_free (version_str);
    g_free (playlist_file);
    return 0;
}
playlist = g_ptr_array_new ();
if (playlist_file != NULL) {
    gchar *playlist_contents = NULL;
    gchar **lines = NULL;
    if (g_file_get_contents (playlist_file, &playlist_contents,
        NULL, &err)) {
        lines = g_strsplit (playlist_contents, "\n", 0);
        num = g_strv_length (lines);
        for (i = 0; i < num; i++) {
            if (lines[i][0] != '\0') {
                GST_LOG ("Playlist[%d]: %s", i + 1, lines[i]);
                add_to_playlist (playlist, lines[i]);
            }
        }
        g_strfreev (lines);
        g_free (playlist_contents);
    } else {
        g_printerr ("Could not read playlist: %s\n", err->message);
        g_clear_error (&err);
    }
    g_free (playlist_file);
    playlist_file = NULL;
}
if (playlist->len == 0 && (filenames == NULL || *filenames == NULL)
    ) {
    g_printerr ("Usage: %s FILE1|URI1 [FILE2|URI2] [FILE3|URI3] ...
        ",
            "gst-play");
    g_printerr ("\n\n"),
```

```
            g_printerr ("%s\n\n",
                    "You must provide at least one filename or URI to
                        play.");
        /* No input provided. Free array */
        g_ptr_array_free (playlist, TRUE);
        return 1;
    }
    /* fill playlist */
    if (filenames != NULL && *filenames != NULL) {
        num = g_strv_length (filenames);
        for (i = 0; i < num; ++i) {
            GST_LOG ("command line argument: %s", filenames[i]);
            add_to_playlist (playlist, filenames[i]);
        }
        g_strfreev (filenames);
    }
    num = playlist->len;
    g_ptr_array_add (playlist, NULL);
    uris = (gchar **) g_ptr_array_free (playlist, FALSE);
    if (shuffle)
        shuffle_uris (uris, num);
    /* prepare */
    play = play_new (uris, volume);
    play->repeat = repeat;
    if (interactive) {
        if (gst_play_kb_set_key_handler (keyboard_cb, play)) {
            atexit (restore_terminal);
        } else {
            g_print ("Interactive keyboard handling in terminal not
                available.\n");
        }
    }
    /* play */
    do_play (play);
    /* clean up */
    play_free (play);
    g_print ("\n");
    gst_deinit ();
    return 0;
}
#endif
```

36.6 gnome-internet-radio-locator-program.c

36.7 gnome-internet-radio-locator-runners.c

36.8 gnome-internet-radio-locator-station.c

```
/* $id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses
     />.
 */

#include <config.h>
#include <stdio.h>
#include <string.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <sys/resource.h>
#include <glib.h>
#include <glib/gstdio.h>
#include <gio/gio.h>
#include <gtk/gtk.h>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include "gnome-internet-radio-locator.h"
#include "gnome-internet-radio-locator-station.h"

extern GNOMEInternetRadioLocatorData *gnome_internet_radio_locator;
extern GNOMEInternetRadioLocatorStationInfo *stationinfo, *localstation
    ;

extern gchar *world_station_xml_filename, *local_station_xml_file;
```

```
extern GStatBuf stats;

extern GtkEntryCompletion *completion;
GtkListStore *model = NULL;
GtkTreeIter iter;

void gnome_internet_radio_locator_helper_run(gchar *url, gchar *name,
    GNOMEInternetRadioLocatorStreamType type,
    GNOMEInternetRadioLocatorHelperType helper)
{
    GError *err = NULL;
    GTimeVal mtime;
    /* const char *mime_info = NULL; */
    /* GnomeVFSMimeApplication *app; */
    char *app = NULL, *command = NULL, *msg = NULL;
    char **argv = NULL;
    gint argc;
    gboolean    ret;
    GError *error = NULL;
    GCancellable *player_cancellable;
    GCancellable *record_cancellable;

    if (signal(SIGCHLD, SIG_IGN) == SIG_ERR) {
        perror(0);
        exit(1);
    }

    g_return_if_fail(url != NULL);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("%s", url);

    /* mime_info = gnome_vfs_get_mime_type(url); */

    g_get_current_time(&mtime);

    /* app = (gchar *)gnome_vfs_mime_get_default_application (mime_info
        ); */

    // gnome_internet_radio_locator_player_main(url,
        gnome_internet_radio_locator->selected_station_name);

    if (helper == GNOME_INTERNET_RADIO_LOCATOR_STREAM_PLAYER) {
        /* gnome_internet_radio_locator_player_main(url,
            gnome_internet_radio_locator->selected_station_name); */
    }

    if (helper == GNOME_INTERNET_RADIO_LOCATOR_STREAM_RECORD) {
        /* gnome_internet_radio_locator_record_main(url,
            gnome_internet_radio_locator->selected_station_name); */
    }

    return;

#if 0
    if (g_strcmp0(app,"no")!=0) {
        command = g_strconcat(app, "␣", url, NULL);
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Helper␣application␣is␣%
            s\n", command);
        if (type == GNOME_INTERNET_RADIO_LOCATOR_STREAM_SHOUTCAST) {
            if (helper == GNOME_INTERNET_RADIO_LOCATOR_STREAM_PLAYER) {
                command = g_strconcat(app, "␣", url, NULL);
                /* argv[0] = g_strdup(app); */
                /* argv[1] = g_strdup(url); */
```

```
}
if (helper == GNOME_INTERNET_RADIO_LOCATOR_STREAM_RECORD) {

    GtkWidget *dialog;
    char *filename, *default_filename;
    GDate gdate;
    time_t t;
    struct tm *tmp;
    char outstr[200];
    t = time(NULL);
    tmp = localtime(&t);
    strftime(outstr, sizeof(outstr), "%a, %d %b %Y %T %z",
        tmp);
    dialog = gtk_file_chooser_dialog_new ("Save Recorded
        Broadcast File",
                          GTK_WINDOW(
                              gnome_internet_radio_locator_app
                              ),
                          GTK_FILE_CHOOSER_ACTION_SAVE,
                          GTK_STOCK_CANCEL,
                              GTK_RESPONSE_CANCEL,
                          GTK_STOCK_SAVE,
                              GTK_RESPONSE_ACCEPT,
                          NULL);
    default_filename = g_strconcat(
        gnome_internet_radio_locator->selected_station_name
        , " - Broadcast Recording - ", outstr, ".mp3", NULL
        );
    gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (
        dialog), default_filename);
    if (gtk_dialog_run (GTK_DIALOG (dialog)) ==
        GTK_RESPONSE_ACCEPT)
    {
        filename = gtk_file_chooser_get_filename (
            GTK_FILE_CHOOSER (dialog));
        gnome_internet_radio_locator->selected_archive_file
             = g_strconcat(filename, NULL);
    }
    gtk_widget_destroy (dialog);

    /* printf("Archiving program at %s\n", archive); */

    appbar_send_msg(_("Recording from %s in %s to %s"),
            gnome_internet_radio_locator->
                selected_station_name,
            gnome_internet_radio_locator->
                selected_station_location,
            gnome_internet_radio_locator->
                selected_archive_file);

    command = g_strconcat(app, " ", url, " -d ",
        g_get_home_dir(), "/.gnome_internet_radio_locator/
        -D %D", NULL);
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("%s\n", command)
        ;

    // gnome_internet_radio_locator_archive_new("Archive",
        gnome_internet_radio_locator->selected_archive_file
        );

    /* " -d ", g_get_home_dir(), "/.
        gnome_internet_radio_locator -D \"", name, "\" -s -
```

```
                    a -u gnome_internet_radio_locator/", VERSION, NULL)
                    ; */
                /* command = g_strconcat(command, " -d ",
                    g_get_home_dir(), "/.gnome_internet_radio_locator
                    /", name, " -D %S%A%T -t 10 -u
                    gnome_internet_radio_locator/", VERSION, NULL); */
            }
                GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Helper␣command␣
                    is␣%s\n", command);
        }
        /* gnome_vfs_mime_application_free (app); */
    } else {
        if (helper == GNOME_INTERNET_RADIO_LOCATOR_STREAM_PLAYER) {
            msg = g_strdup_printf(_("An␣error␣happened␣trying␣to␣play␣"
                        "%s\nEither␣the␣file␣doesn't␣exist,␣or␣you␣"
                        "don't␣have␣a␣player␣for␣it."),
                          url);
        }
        if (helper == GNOME_INTERNET_RADIO_LOCATOR_STREAM_RECORD) {
            msg = g_strdup_printf(_("An␣error␣happened␣trying␣to␣record
                ␣"
                        "%s\nEither␣the␣file␣doesn't␣exist,␣or␣you␣"
                        "don't␣have␣a␣recorder␣for␣it."),
                          url);
        }
        if (msg != NULL) {
            show_error(msg);
            g_free(msg);
        }
        return;
    }

    if (helper == GNOME_INTERNET_RADIO_LOCATOR_STREAM_PLAYER) {
#if 0
        g_shell_parse_argv(command,
                   &argc,
                   &argv,
                   NULL);
        ret = g_spawn_async_with_pipes (".",
                        argv,
                        NULL,
                        G_SPAWN_SEARCH_PATH|G_SPAWN_STDOUT_TO_DEV_NULL|
                            G_SPAWN_STDERR_TO_DEV_NULL|
                            G_SPAWN_DO_NOT_REAP_CHILD,
                        NULL,
                        NULL,
                        &gnome_internet_radio_locator->player_pid,
                        NULL,
                        NULL,
                        NULL,
                        &err);
        if( ! ret )
        {
            msg = g_strdup_printf(_("Failed␣to␣run␣%s␣(%i)\n"), command
                , gnome_internet_radio_locator->player_pid);
            show_error(msg);
            g_free(msg);
            return;
        }
        /* Add watch function to catch termination of the process. This
             function
         * will clean any remnants of process. */
```

```
        g_child_watch_add( gnome_internet_radio_locator->player_pid, (
            GChildWatchFunc)cb_child_watch_player,
            gnome_internet_radio_locator);

        /* gnome_internet_radio_locator->player_pid = pid; */
        gnome_internet_radio_locator->player_status =
            GNOME_INTERNET_RADIO_LOCATOR_PLAYER_TRUE;
        /* Install timeout fnction that will move the progress bar */
        gnome_internet_radio_locator->timeout_id = g_timeout_add(100,(
            GSourceFunc)cb_timeout,gnome_internet_radio_locator);
/* #endif */
/* #if 0 */
        ret = g_spawn_async_with_pipes( NULL, /* command */ argv, NULL,
                        G_SPAWN_DO_NOT_REAP_CHILD|G_SPAWN_DEFAULT, NULL
                            ,
                        NULL, &gnome_internet_radio_locator->player_pid
                            , NULL, &out, &error, NULL );
        if( ! ret )
        {
            msg = g_strdup_printf(_("Failed␣to␣run␣%s␣(%i)\n"), command
                , gnome_internet_radio_locator->player_pid);
            show_error(msg);
            g_free(msg);
            return;
        }
        /* Add watch function to catch termination of the process. This
             function
         * will clean any remnants of process. */
        g_child_watch_add( gnome_internet_radio_locator->player_pid, (
            GChildWatchFunc)cb_child_watch_player,
            gnome_internet_radio_locator );
        /* Create channels that will be used to read
            gnome_internet_radio_locator from pipes. *

#ifdef G_OS_WIN32
        out_ch = g_io_channel_win32_new_fd( out );
        err_ch = g_io_channel_win32_new_fd( error );
#else
        out_ch = g_io_channel_unix_new( out );
        err_ch = g_io_channel_unix_new( error );
#endif
        */

                /* Add watches to channels */
        g_io_add_watch( out_ch, G_IO_IN | G_IO_HUP, (GIOFunc)
            cb_out_watch, gnome_internet_radio_locator );
        g_io_add_watch( err_ch, G_IO_IN | G_IO_HUP, (GIOFunc)
            cb_err_watch, gnome_internet_radio_locator );
        /* Install timeout fnction that will move the progress bar */
        gnome_internet_radio_locator->timeout_id = g_timeout_add( 100,
            (GSourceFunc)cb_timeout, gnome_internet_radio_locator );
/* #endif */
#if 0
        if (!g_spawn_command_line_sync(command, stdout, stderr, status,
             &err)) {
            msg = g_strdup_printf(_("Failed␣to␣open␣URL:␣'%s'\n"
                        "Status␣code:␣%i\n"
                        "Details:␣%s"), url, status, err->message);
            show_error(msg);
            g_error_free(err);
            g_free(msg);
        } else {
```

```
            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Launching␣%s\n",
                command);
        }
        /* Add watch function to catch termination of the process. This
             function
         * will clean any remnants of process. */
        g_child_watch_add( gnome_internet_radio_locator->player_pid, (
            GChildWatchFunc)cb_child_watch_player,
            gnome_internet_radio_locator );
#endif
        /* Original async player code */

        gnome_internet_radio_locator->player_launcher =
            g_subprocess_launcher_new (G_SUBPROCESS_FLAGS_NONE);
        gnome_internet_radio_locator->player_subprocess =
            g_subprocess_launcher_spawn (gnome_internet_radio_locator->
            player_launcher, &error, app, url, NULL);
        if (gnome_internet_radio_locator->player_subprocess == NULL) {
            msg = g_strdup_printf(_("Failed␣to␣open␣URL:␣'%s'\n"
                        "Details:␣%s"), url, error->message);
            show_error(msg);
            g_error_free(error);
            g_free(msg);
            goto quit_player;
        } else {
            gnome_internet_radio_locator->player_status =
                GNOME_INTERNET_RADIO_LOCATOR_PLAYER_TRUE;
            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Launching␣%s␣player
                \n", command);
        }

#endif
#if 0
        if (!g_spawn_command_line_async(command, &err)) {
            msg = g_strdup_printf(_("Failed␣to␣open␣URL:␣'%s'\n"
                        "Details:␣%s"), url, err->message);
            show_error(msg);
            g_error_free(err);
            g_free(msg);
        } else {
            gnome_internet_radio_locator->player_status =
                GNOME_INTERNET_RADIO_LOCATOR_PLAYER_TRUE;
            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Launching␣%s␣player
                \n", command);
            g_subprocess_wait_async (gnome_internet_radio_locator->
                player_subprocess,
                         player_cancellable,
                         (GAsyncReadyCallback)cb_subprocess_player,
                         NULL);
        }
#endif

    }

    if (helper == GNOME_INTERNET_RADIO_LOCATOR_STREAM_RECORD) {

/* #if 0 */
        /* gchar *argv[] = { command, NULL }; */
        /* Spawn child process */
        gchar *recording_path_name = g_strconcat(g_get_home_dir(), "/.
            gnome_internet_radio_locator/", NULL);
        gchar *formatting_argument = g_strdup("-D␣%D");
```

```
        gnome_internet_radio_locator->record_launcher =
            g_subprocess_launcher_new (G_SUBPROCESS_FLAGS_NONE);
        /* gnome_internet_radio_locator->record_subprocess =
            g_subprocess_launcher_spawn (gnome_internet_radio_locator->
            record_launcher, &error,
            GNOME_INTERNET_RADIO_LOCATOR_HELPER_RECORD, url, "-d",
            recording_path_name, "-D", "%D", NULL); */
        if (gnome_internet_radio_locator->record_subprocess == NULL) {
            /* msg = g_strdup_printf(_("Failed to run '%s'\n" */
            /*          "Details: %s"),
                GNOME_INTERNET_RADIO_LOCATOR_HELPER_RECORD, error->
                message); */
            show_error(msg);
            g_error_free(error);
            g_free(msg);
            goto quit_record;
        } else {
            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Launching␣%s\n",
                command);
            g_subprocess_wait_async (gnome_internet_radio_locator->
                record_subprocess,
                         record_cancellable,
                         (GAsyncReadyCallback)cb_subprocess_record,
                             NULL);
        }

#if 0
        g_shell_parse_argv(command,
                   &argc,
                   &argv,
                   NULL);
        ret = g_spawn_async_with_pipes (".",
                        argv,
                        NULL,
                        G_SPAWN_SEARCH_PATH|G_SPAWN_STDOUT_TO_DEV_NULL|
                            G_SPAWN_STDERR_TO_DEV_NULL|
                            G_SPAWN_DO_NOT_REAP_CHILD,
                        NULL,
                        NULL,
                        &gnome_internet_radio_locator->record_pid,
                        NULL,
                        NULL,
                        NULL,
                        &err);
        if( ! ret )
        {
            msg = g_strdup_printf(_("Failed␣to␣run␣%s␣(%i)\n"), command
                , gnome_internet_radio_locator->record_pid);
            show_error(msg);
            g_free(msg);
            return;
        }
        /* Add watch function to catch termination of the process. This
             function
         * will clean any remnants of process. */
        g_child_watch_add( gnome_internet_radio_locator->record_pid, (
            GChildWatchFunc)cb_child_watch_record,
            gnome_internet_radio_locator);

        /* gnome_internet_radio_locator->record_pid = pid; */
        gnome_internet_radio_locator->record_status =
            GNOME_INTERNET_RADIO_LOCATOR_RECORD_TRUE;
```

```
        gnome_internet_radio_locator_archive_new("Archive",
            gnome_internet_radio_locator->selected_archive_file,
            gnome_internet_radio_locator->selected_streams_codec);

        /* Install timeout fnction that will move the progress bar */
        gnome_internet_radio_locator->timeout_id = g_timeout_add(100,(
            GSourceFunc)cb_timeout,gnome_internet_radio_locator);
/* #endif */
#if 0
        ret = g_spawn_async_with_pipes( NULL, /* command */ argv, NULL,
                        G_SPAWN_DO_NOT_REAP_CHILD|G_SPAWN_DEFAULT, NULL
                            ,
                        NULL, &gnome_internet_radio_locator->record_pid
                            , NULL, &out, &error, NULL );
        if( ! ret )
        {
            msg = g_strdup_printf(_("Failed␣to␣run␣%s␣(%i)\n"), command
                , gnome_internet_radio_locator->record_pid);
            show_error(msg);
            g_free(msg);
            return;
        }
        /* Add watch function to catch termination of the process. This
             function
         * will clean any remnants of process. */
        g_child_watch_add( gnome_internet_radio_locator->record_pid, (
            GChildWatchFunc)cb_child_watch_record,
            gnome_internet_radio_locator );
        /* Create channels that will be used to read
            gnome_internet_radio_locator from pipes. */
#ifdef G_OS_WIN32
        out_ch = g_io_channel_win32_new_fd( out );
        err_ch = g_io_channel_win32_new_fd( error );
#else
        out_ch = g_io_channel_unix_new( out );
        err_ch = g_io_channel_unix_new( error );
#endif
        /* Add watches to channels */
        g_io_add_watch( out_ch, G_IO_IN | G_IO_HUP, (GIOFunc)
            cb_out_watch, gnome_internet_radio_locator );
        g_io_add_watch( err_ch, G_IO_IN | G_IO_HUP, (GIOFunc)
            cb_err_watch, gnome_internet_radio_locator );
        /* Install timeout fnction that will move the progress bar */
        gnome_internet_radio_locator->timeout_id = g_timeout_add( 100,
            (GSourceFunc)cb_timeout, gnome_internet_radio_locator );
#endif
#if 0
        if (!g_spawn_command_line_sync(command, stdout, stderr, status,
             &err)) {
            msg = g_strdup_printf(_("Failed␣to␣open␣URL:␣'%s'\n"
                        "Status␣code:␣%i\n"
                        "Details:␣%s"), url, status, err->message);
            show_error(msg);
            g_error_free(err);
            g_free(msg);
        } else {
            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Launching␣%s\n",
                command);
        }
        /* Add watch function to catch termination of the process. This
             function
```

```
         * will clean any remnants of process. */
        g_child_watch_add( gnome_internet_radio_locator->record_pid, (
            GChildWatchFunc)cb_child_watch_record,
            gnome_internet_radio_locator );
#endif
#endif
    }

quit_player:

        if (gnome_internet_radio_locator->player_subprocess)
        g_object_unref(gnome_internet_radio_locator->player_subprocess)
            ;
        if (gnome_internet_radio_locator->player_launcher)
        g_object_unref(gnome_internet_radio_locator->player_launcher);
quit_record:
        if (gnome_internet_radio_locator->record_subprocess)
        g_object_unref(gnome_internet_radio_locator->record_subprocess)
            ;
        if (gnome_internet_radio_locator->record_launcher)
        g_object_unref(gnome_internet_radio_locator->record_launcher);
#endif
}

static void
gnome_internet_radio_locator_station_parser(
    GNOMEInternetRadioLocatorStationInfo * station, xmlDocPtr doc,
            xmlNodePtr cur)
{
    xmlNodePtr sub;
    char *chans;

    g_return_if_fail(station != NULL);
    g_return_if_fail(doc != NULL);
    g_return_if_fail(cur != NULL);

    station->id = (gchar *)xmlGetProp(cur, (const xmlChar *)"id");
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->id = %s\n",
        station->id);
    station->name = (gchar *)xmlGetProp(cur, (const xmlChar *)"name");
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->name = %s\n",
        station->name);
    station->rank = (gchar *)xmlGetProp(cur, (const xmlChar *)"rank");
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->rank = %s\n",
        station->rank);
    station->type = (gchar *)xmlGetProp(cur, (const xmlChar *)"type");
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->type = %s\n",
        station->type);
    station->band = (gchar *)xmlGetProp(cur, (const xmlChar *)"band");
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->band = %s\n",
        station->band);

    sub = cur->xmlChildrenNode;

    while (sub != NULL) {

        if ((!xmlStrcmp(sub->name, (const xmlChar *) "frequency"))) {
            station->frequency = (gchar *)
                xmlNodeListGetString(doc, sub->xmlChildrenNode,
                        1);
            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->frequency
                = %s\n",
```

```
        station->frequency);
    }

    if ((!xmlStrcmp(sub->name, (const xmlChar *) "location"))) {
        station->location = (gchar *)
            xmlNodeListGetString(doc, sub->xmlChildrenNode,
                    1);
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->location␣=
            ␣%s\n", station->location);

    }

    if ((!xmlStrcmp
         (sub->name, (const xmlChar *) "description"))) {
        station->description = (gchar *)
            xmlNodeListGetString(doc, sub->xmlChildrenNode,
                    1);
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->
            description␣=␣%s\n", station->description);
    }

    if ((!xmlStrcmp(sub->name, (const xmlChar *) "uri"))) {
        station->uri = (gchar *)
            xmlNodeListGetString(doc, sub->xmlChildrenNode,
                    1);
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->uri␣=␣%s\n
            ", station->uri);
        /* fprintf(stdout, "<a href=\"%s\">%s (%s)</a>\n\n",
            station->uri, station->name, station->location); */
    }

    if ((!xmlStrcmp(sub->name, (const xmlChar *) "stream"))) {

        GNOMEInternetRadioLocatorStreamInfo *stream = g_new0(
            GNOMEInternetRadioLocatorStreamInfo, 1);

        /* gnome_internet_radio_locator->stream_count++; */

        station->stream = stream;

        station->stream->mimetype = (gchar *)
            xmlGetProp(sub, (const xmlChar *)"mime");
        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->stream->
            mimetype␣=␣%s\n",
            station->stream->mimetype);
        if (xmlGetProp(sub, (const xmlChar *)"bitrate") != NULL) {
            station->stream->bitrate =
                (glong)atol((char *)xmlGetProp(sub, (const xmlChar
                    *)"bitrate"));
            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->stream
                ->bitrate␣=␣%li\n",
                station->stream->bitrate);
        }

        if (xmlGetProp(sub, (const xmlChar *)"samplerate") != NULL)
             {
            station->stream->samplerate = (glong) atol((char *)
                xmlGetProp(sub, (const xmlChar *)"samplerate"));
        }

        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->stream->
            samplerate␣=␣%li\n",
```

```
                station->stream->samplerate);
            station->stream->uri = (gchar *)xmlGetProp(sub, (const
                xmlChar *)"uri");
            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->stream->
                uri␣=␣%s\n",
                station->stream->uri);

            chans = (gchar *)xmlGetProp(sub, (const xmlChar *)"stations
                ");

            if (chans != NULL) {
                if (strcmp(chans, "stereo") == 0) {
                    station->stream->channels =
                        GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_STEREO;
                    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->
                        stream->channels␣=␣%d\n", station->stream->
                        channels);
                } else if (strcmp(chans, "mono") == 0) {
                    station->stream->channels =
                        GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_MONO;
                    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->
                        stream->channels␣=␣%d\n", station->stream->
                        channels);
                } else if (strcmp(chans, "5:1") == 0) {
                    station->stream->channels =
                        GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_5_1;
                    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("station->
                        stream->channels␣=␣%d\n", station->stream->
                        channels);
                }
                g_free(chans);
            }

        }
        /* if... "stream" */
        sub = sub->next;
    }
    /* gnome_internet_radio_locator->station_count++; */
    return;
}

GNOMEInternetRadioLocatorStationInfo *
    gnome_internet_radio_locator_station_load_from_http(
    GNOMEInternetRadioLocatorStationInfo * head,
                         gpointer data)
{
    GNOMEInternetRadioLocatorStationInfo *gstation;
    gstation = gnome_internet_radio_locator_station_load_from_file (
        head, "https://people.gnome.org/~ole/gnome-internet-radio-
        locator/gnome_internet_radio_locator.xml");
    return gstation;
}

GNOMEInternetRadioLocatorStationInfo *
    gnome_internet_radio_locator_station_load_from_file(
    GNOMEInternetRadioLocatorStationInfo * head,
                         char *filename)
{
    xmlDocPtr doc = NULL;
    xmlNodePtr cur = NULL;
    GNOMEInternetRadioLocatorStationInfo *curr = NULL;
    char *version;
```

```
GNOMEInternetRadioLocatorStationInfo *mem_station;

g_return_val_if_fail(filename != NULL, NULL);

doc = xmlReadFile(filename, NULL, 0);

if (doc == NULL) {
    perror("xmlParseFile");
    xmlFreeDoc(doc);
    return NULL;
}

cur = xmlDocGetRootElement(doc);

if (cur == NULL) {
    fprintf(stderr, "Empty␣document\n");
    xmlFreeDoc(doc);
    return NULL;
}

if (xmlStrcmp(cur->name, (const xmlChar *) "
    gnome_internet_radio_locator")) {
    fprintf(stderr,
        "Document␣of␣wrong␣type,␣root␣node␣!=␣
            gnome_internet_radio_locator\n");
    xmlFreeDoc(doc);
    return NULL;
}

version = (gchar *)xmlGetProp(cur, (const xmlChar *)"version");

GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Valid␣
    GNOMEInternetRadioLocator␣%s␣XML␣document...␣Parsing␣stations
    ...\n",
    version);

free(version);

cur = cur->xmlChildrenNode;

while (cur != NULL) {

    if ((!xmlStrcmp(cur->name, (const xmlChar *) "station"))) {

        GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Found␣a␣new␣station
            .\n");

        curr = g_new0(GNOMEInternetRadioLocatorStationInfo, 1);
        mem_station = g_new0(GNOMEInternetRadioLocatorStationInfo,
            1);

        gnome_internet_radio_locator_station_parser(curr, doc, cur)
            ;

        curr->next = head;

        head = curr;

        mem_station = head;

        gnome_internet_radio_locator_stations = g_list_append(
            gnome_internet_radio_locator_stations, (
```

```
                GNOMEInternetRadioLocatorStationInfo *)mem_station);

            GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Done␣with␣parsing␣
                the␣station.\n");

        }
        cur = cur->next;
    }

    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("Finished␣parsing␣XML␣
        document.\n");

    xmlFreeDoc(doc);

    return curr;
}

gint gnome_internet_radio_locator_station_update (
    GNOMEInternetRadioLocatorStationInfo *head, gchar *station_band,
    gchar *station_description, gchar *station_name, gchar *
    station_location, gchar *station_uri, gchar *station_website) {
    gchar msg[256];
    gchar *msg1, *msg2;
    /* Open ~/.gnome_internet_radio_locator/
        gnome_internet_radio_locator.xml.  Parse structure.  Insert new
         item.  Save structure. */
    GNOMEInternetRadioLocatorStationInfo *new_station;
    GNOMEInternetRadioLocatorStationInfo *stationinfo;
    /* GList *gnome_internet_radio_locator_local_stations = NULL; */
    gchar *local_station_uri, *local_station_name, *
        local_station_location, *local_station_band, *
        local_station_description, *local_station_website;
    gchar *stations = g_strconcat(g_get_home_dir(), "/.gnome-internet-
        radio-locator/gnome-internet-radio-locator.xml", NULL);

    gboolean local_gnome_internet_radio_locator_file = g_file_test (".
        gnome-internet-radio-locator/gnome-internet-radio-locator.xml",
         G_FILE_TEST_EXISTS);

    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("
        local_gnome_internet_radio_locator_file␣=␣%i\n",
        local_gnome_internet_radio_locator_file);

    if (local_gnome_internet_radio_locator_file == 0) {

        gchar *local_gnome_internet_radio_locator_directory =
            g_strconcat(g_get_home_dir(), "/.gnome-internet-radio-
            locator", NULL);
        g_mkdir_with_parents (
            local_gnome_internet_radio_locator_directory, 0700);

    }

    FILE *fp;

    stationinfo = gnome_internet_radio_locator_station_load_from_file(
        NULL, stations);

    new_station = g_new0(GNOMEInternetRadioLocatorStationInfo, 1);
    new_station->name = g_strdup(station_name);
    new_station->band = g_strdup(station_band);
    new_station->description = g_strdup(station_description);
```

```
new_station->location = g_strdup(station_location);
new_station->stream = g_new0(GNOMEInternetRadioLocatorStreamInfo,
    1);
new_station->stream->uri = g_strdup(station_uri);
new_station->uri = g_strdup(station_website);
fp = g_fopen(stations, "w+");
g_fprintf(fp, "<?xml␣version='1.0'␣encoding='UTF-8'?>\n<!DOCTYPE␣
    gnome_internet_radio_locator␣SYSTEM␣'
    gnome_internet_radio_locator-0.1.dtd'>\n<
    gnome_internet_radio_locator␣version='%s'>\n", VERSION);
// stationinfo-> = l->data;
while (stationinfo != NULL) {
    local_station_uri = g_strdup(stationinfo->stream->uri);
    local_station_name = g_strdup(stationinfo->name);
    local_station_location = g_strdup(stationinfo->location);
    local_station_band = g_strdup(stationinfo->band);
    local_station_description = g_strdup(stationinfo->description);
    local_station_website = g_strdup(stationinfo->uri);
    /* FIXME: Save mime='audio/mp3' uri='%s' codec='MPEG 1 Audio,
        Layer 3 (MP3)' samplerate='24000 Hz' channels='Mono'
        bitrate='32 kbps' */
    g_fprintf(fp, "␣␣<station␣band=\"%s\"␣id=\"%s\"␣lang=\"en\"␣
        name=\"%s\"␣rank=\"1.0\"␣type=\"org\">\n␣␣␣␣<frequency␣uri
        =\"%s\">%s␣in␣%s</frequency>\n␣␣␣␣<location>%s</location>\n
        ␣␣␣␣<description␣lang=\"en\">%s</description>\n␣␣␣␣<stream␣
        uri=\"%s\"␣/>\n␣␣␣␣<uri>%s</uri>\n␣␣</station>\n",
        local_station_band, local_station_name, local_station_name,
         local_station_website, local_station_band,
        local_station_location, local_station_location,
        local_station_description, local_station_uri,
        local_station_website);
    stationinfo = stationinfo->next;

}
g_fprintf(fp, "␣␣<station␣band=\"%s\"␣id=\"%s\"␣lang=\"en\"␣name
    =\"%s\"␣rank=\"1.0\"␣type=\"org\">\n␣␣␣␣<frequency␣uri=\"%s\">%
    s␣in␣%s</frequency>\n␣␣␣␣<location>%s</location>\n␣␣␣␣<
    description␣lang=\"en\">%s</description>\n␣␣␣␣<stream␣uri=\"%s
    \"␣/>\n␣␣␣␣<uri>%s</uri>\n␣␣</station>\n", new_station->band,
    new_station->name, new_station->name, new_station->uri,
    new_station->band, new_station->location, new_station->location
    , new_station->description, new_station->stream->uri,
    new_station->uri);
g_fprintf(fp, "</gnome_internet_radio_locator>\n");
fclose(fp);
gnome_internet_radio_locator_stations = g_list_append(
    gnome_internet_radio_locator_stations, (
    GNOMEInternetRadioLocatorStationInfo *)new_station);

g_free(stations);
g_free(new_station);
g_free(stationinfo);

model = gtk_list_store_new(11, G_TYPE_STRING, G_TYPE_STRING,
    G_TYPE_STRING, G_TYPE_STRING, G_TYPE_STRING, G_TYPE_STRING,
    G_TYPE_STRING, G_TYPE_STRING, G_TYPE_STRING, G_TYPE_STRING,
    G_TYPE_STRING, G_TYPE_STRING);

world_station_xml_filename = g_strconcat(
    GNOME_INTERNET_RADIO_LOCATOR_DATADIR, "/gnome-internet-radio-
    locator.xml", NULL);
```

```
    GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG("world_station_xml_filename␣
        =␣%s\n",
        world_station_xml_filename);

    if (world_station_xml_filename == NULL) {
        g_warning(("Failed␣to␣open␣%s.␣␣Please␣install␣it.\n"),
              world_station_xml_filename);
    }

    local_station_xml_file =
        g_strconcat(g_get_home_dir(), "/.gnome-internet-radio-locator/
            gnome-internet-radio-locator.xml", NULL);

    if (!g_stat(local_station_xml_file, &stats)) {
        localstation =
            gnome_internet_radio_locator_station_load_from_file(NULL,
            local_station_xml_file);
    } else {
        localstation = NULL;
    }

    if (localstation == NULL) {
        printf("Failed␣to␣open␣%s.\n", local_station_xml_file);
    }

/*   g_free (local_station_xml_file); */

    stationinfo = gnome_internet_radio_locator_station_load_from_file(
        localstation, world_station_xml_filename);

    gnome_internet_radio_locator_stations = NULL;

    while (stationinfo != NULL) {

        gtk_list_store_append(model, &iter);
        gtk_list_store_set(model,
                   &iter,
                   STATION_NAME,
                   stationinfo->name,
                   STATION_LOCATION,
                   stationinfo->location,
                   STATION_URI,
                   stationinfo->stream->uri,
                   STATION_DESCRIPTION,
                   stationinfo->description,
                   STATION_FREQUENCY,
                   stationinfo->frequency,
                   STATION_BAND,
                   stationinfo->band,
                   STATION_TYPE,
                   stationinfo->type,
                   STATION_RANK,
                   stationinfo->rank,
                   STATION_BITRATE,
                   stationinfo->bitrate,
                   STATION_SAMPLERATE,
                   stationinfo->samplerate,
                   STATION_ID,
                   stationinfo->id,
                   -1);

        stationinfo = stationinfo->next;
```

```
    }

    gtk_entry_completion_set_model(completion, GTK_TREE_MODEL(model));

    return (0);
}
```

36.9 gnome-internet-radio-locator-stations-map.c

36.10 gnome-internet-radio-locator-streams.c

36.11 gnome-internet-radio-locator-tz.c

```
/* -*- Mode: C; tab-width: 8; indent-tabs-mode: t; c-basic-offset: 8
    -*- */
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2015, 2018, 2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses
    />.
```

```
 */

/*
 * Essential parts of the source code below was based on
 * gnome-control-center/panels/datetime/tz.c
 *
 */

/* Generic timezone utilities.
 *
 * Copyright (C) 2000-2001 Ximian, Inc.
 *
 * Authors: Hans Petter Jansson <hpj@ximian.com>
 *
 * Largely based on Michael Fulbright's work on Anaconda.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, see <http://www.gnu.org/licenses/>.
 */


#include <config.h>
#include <glib.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <math.h>
#include <string.h>
#include "gnome-internet-radio-locator-tz.h"

/* Forward declarations for private functions */

static float convert_pos (gchar *pos, int digits);
static int compare_country_names (const void *a, const void *b);
static void sort_locations_by_country (GPtrArray *locations);
static gchar * tz_data_file_get (void);
static void load_backward_tz (TzDB *tz_db);

/* ---------------- *
 * Public interface *
 * ---------------- */
TzDB *
tz_load_db (void)
{
    gchar *tz_data_file;
    TzDB *tz_db;
    FILE *tzfile;
    char buf[4096];

    tz_data_file = tz_data_file_get ();
    if (!tz_data_file) {
```

```
        g_warning ("Could not get the TimeZone data file name");
        return NULL;
    }
    tzfile = fopen (tz_data_file, "r");
    if (!tzfile) {
        g_warning ("Could not open *%s*\n", tz_data_file);
        g_free (tz_data_file);
        return NULL;
    }

    tz_db = g_new0 (TzDB, 1);
    tz_db->locations = g_ptr_array_new ();

    while (fgets (buf, sizeof(buf), tzfile))
    {
        gchar **tmpstrarr;
        gchar *latstr, *lngstr, *p;
        TzLocation *loc;

        if (*buf == '#') continue;

        g_strchomp(buf);
        tmpstrarr = g_strsplit(buf,"\t", 6);

        latstr = g_strdup (tmpstrarr[1]);
        p = latstr + 1;
        while (*p != '-' && *p != '+') p++;
        lngstr = g_strdup (p);
        *p = '\0';

        loc = g_new0 (TzLocation, 1);
        loc->country = g_strdup (tmpstrarr[0]);
        loc->zone = g_strdup (tmpstrarr[2]);
        loc->latitude  = convert_pos (latstr, 2);
        loc->longitude = convert_pos (lngstr, 3);

#ifdef __sun
        if (tmpstrarr[3] && *tmpstrarr[3] == '-' && tmpstrarr[4])
            loc->comment = g_strdup (tmpstrarr[4]);

        if (tmpstrarr[3] && *tmpstrarr[3] != '-' && !islower(loc->zone)
            ) {
            TzLocation *locgrp;

            /* duplicate entry */
            locgrp = g_new0 (TzLocation, 1);
            locgrp->country = g_strdup (tmpstrarr[0]);
            locgrp->zone = g_strdup (tmpstrarr[3]);
            locgrp->latitude  = convert_pos (latstr, 2);
            locgrp->longitude = convert_pos (lngstr, 3);
            locgrp->comment = (tmpstrarr[4]) ? g_strdup (tmpstrarr[4])
                : NULL;

            g_ptr_array_add (tz_db->locations, (gpointer) locgrp);
        }
#else
        loc->comment = (tmpstrarr[3]) ? g_strdup(tmpstrarr[3]) : NULL;
#endif

        g_ptr_array_add (tz_db->locations, (gpointer) loc);

        g_free (latstr);
```

```
        g_free (lngstr);
        g_strfreev (tmpstrarr);
    }

    fclose (tzfile);

    /* now sort by country */
    sort_locations_by_country (tz_db->locations);

    g_free (tz_data_file);

    /* Load up the hashtable of backward links */
    load_backward_tz (tz_db);

    return tz_db;
}

static void
tz_location_free (TzLocation *loc)
{
    g_free (loc->country);
    g_free (loc->zone);
    g_free (loc->comment);

    g_free (loc);
}

void
tz_db_free (TzDB *db)
{
    g_ptr_array_foreach (db->locations, (GFunc) tz_location_free, NULL)
        ;
    g_ptr_array_free (db->locations, TRUE);
    g_hash_table_destroy (db->backward);
    g_free (db);
}

GPtrArray *
tz_get_locations (TzDB *db)
{
    return db->locations;
}


gchar *
tz_location_get_country (TzLocation *loc)
{
    return loc->country;
}


gchar *
tz_location_get_zone (TzLocation *loc)
{
    return loc->zone;
}


gchar *
tz_location_get_comment (TzLocation *loc)
{
    return loc->comment;
```

```
}


void
tz_location_get_position (TzLocation *loc, double *longitude, double *
    latitude)
{
    *longitude = loc->longitude;
    *latitude = loc->latitude;
}

glong
tz_location_get_utc_offset (TzLocation *loc)
{
    TzInfo *tz_info;
    glong offset;

    tz_info = tz_info_from_location (loc);
    offset = tz_info->utc_offset;
    tz_info_free (tz_info);
    return offset;
}

TzInfo *
tz_info_from_location (TzLocation *loc)
{
    TzInfo *tzinfo;
    time_t curtime;
    struct tm *curzone;
    gchar *tz_env_value;

    g_return_val_if_fail (loc != NULL, NULL);
    g_return_val_if_fail (loc->zone != NULL, NULL);

    tz_env_value = g_strdup (getenv ("TZ"));
    setenv ("TZ", loc->zone, 1);

#if 0
    tzset ();
#endif
    tzinfo = g_new0 (TzInfo, 1);

    curtime = time (NULL);
    curzone = localtime (&curtime);

#ifndef __sun
    /* Currently this solution doesnt seem to work - I get that */
    /* America/Phoenix uses daylight savings, which is wrong    */
    tzinfo->tzname_normal = g_strdup (curzone->tm_zone);
    if (curzone->tm_isdst)
        tzinfo->tzname_daylight =
            g_strdup (&curzone->tm_zone[curzone->tm_isdst]);
    else
        tzinfo->tzname_daylight = NULL;

    tzinfo->utc_offset = curzone->tm_gmtoff;
#else
    tzinfo->tzname_normal = NULL;
    tzinfo->tzname_daylight = NULL;
    tzinfo->utc_offset = 0;
#endif
```

```
    tzinfo->daylight = curzone->tm_isdst;

    if (tz_env_value)
        setenv ("TZ", tz_env_value, 1);
    else
        unsetenv ("TZ");

    g_free (tz_env_value);

    return tzinfo;
}


void
tz_info_free (TzInfo *tzinfo)
{
    g_return_if_fail (tzinfo != NULL);

    if (tzinfo->tzname_normal) g_free (tzinfo->tzname_normal);
    if (tzinfo->tzname_daylight) g_free (tzinfo->tzname_daylight);
    g_free (tzinfo);
}

struct {
    const char *orig;
    const char *dest;
} aliases[] = {
    { "Asia/Istanbul",  "Europe/Istanbul" },    /* Istanbul is in both
        Europe and Asia */
    { "Europe/Nicosia", "Asia/Nicosia" },       /* Ditto */
    { "EET",            "Europe/Istanbul" },    /* Same tz as the 2
        above */
    { "HST",            "Pacific/Honolulu" },
    { "WET",            "Europe/Brussels" },    /* Other name for the
        mainland Europe tz */
    { "CET",            "Europe/Brussels" },    /* ditto */
    { "MET",            "Europe/Brussels" },
    { "Etc/Zulu",       "Etc/GMT" },
    { "Etc/UTC",        "Etc/GMT" },
    { "GMT",            "Etc/GMT" },
    { "Greenwich",      "Etc/GMT" },
    { "Etc/UCT",        "Etc/GMT" },
    { "Etc/GMT0",       "Etc/GMT" },
    { "Etc/GMT+0",      "Etc/GMT" },
    { "Etc/GMT-0",      "Etc/GMT" },
    { "Etc/Universal",  "Etc/GMT" },
    { "PST8PDT",        "America/Los_Angeles" },    /* Other name for
        the Atlantic tz */
    { "EST",            "America/New_York" },   /* Other name for the
        Eastern tz */
    { "EST5EDT",        "America/New_York" },   /* ditto */
    { "CST6CDT",        "America/Chicago" },    /* Other name for the
        Central tz */
    { "MST",            "America/Denver" },     /* Other name for the
        mountain tz */
    { "MST7MDT",        "America/Denver" },     /* ditto */
};

static gboolean
compare_timezones (const char *a,
           const char *b)
{
```

```
    if (g_str_equal (a, b))
        return TRUE;
    if (strchr (b, '/') == NULL) {
        char *prefixed;

        prefixed = g_strdup_printf ("/%s", b);
        if (g_str_has_suffix (a, prefixed)) {
            g_free (prefixed);
            return TRUE;
        }
        g_free (prefixed);
    }

    return FALSE;
}

char *
tz_info_get_clean_name (TzDB *tz_db,
            const char *tz)
{
    char *ret;
    const char *timezone;
    guint i;
    gboolean replaced;

    /* Remove useless prefixes */
    if (g_str_has_prefix (tz, "right/"))
        tz = tz + strlen ("right/");
    else if (g_str_has_prefix (tz, "posix/"))
        tz = tz + strlen ("posix/");

    /* Here start the crazies */
    replaced = FALSE;

    for (i = 0; i < G_N_ELEMENTS (aliases); i++) {
        if (compare_timezones (tz, aliases[i].orig)) {
            replaced = TRUE;
            timezone = aliases[i].dest;
            break;
        }
    }

    /* Try again! */
    if (!replaced) {
        /* Ignore crazy solar times from the '80s */
        if (g_str_has_prefix (tz, "Asia/Riyadh") ||
            g_str_has_prefix (tz, "Mideast/Riyadh")) {
            timezone = "Asia/Riyadh";
            replaced = TRUE;
        }
    }

    if (!replaced)
        timezone = tz;

    ret = g_hash_table_lookup (tz_db->backward, timezone);
    if (ret == NULL)
        return g_strdup (timezone);
    return g_strdup (ret);
}

/* ----------------- *
```

```
 * Private functions *
 * ----------------- */

static gchar *
tz_data_file_get (void)
{
    gchar *file;

    file = g_strdup (TZ_DATA_FILE);

    return file;
}

static float
convert_pos (gchar *pos, int digits)
{
    gchar whole[10];
    gchar *fraction;
    gint i;
    float t1, t2;

    if (!pos || strlen(pos) < 4 || digits > 9) return 0.0;

    for (i = 0; i < digits + 1; i++) whole[i] = pos[i];
    whole[i] = '\0';
    fraction = pos + digits + 1;

    t1 = g_strtod (whole, NULL);
    t2 = g_strtod (fraction, NULL);

    if (t1 >= 0.0) return t1 + t2/pow (10.0, strlen(fraction));
    else return t1 - t2/pow (10.0, strlen(fraction));
}


#if 0

/* Currently not working */
static void
free_tzdata (TzLocation *tz)
{

    if (tz->country)
      g_free(tz->country);
    if (tz->zone)
      g_free(tz->zone);
    if (tz->comment)
      g_free(tz->comment);

    g_free(tz);
}
#endif


static int
compare_country_names (const void *a, const void *b)
{
    const TzLocation *tza = * (TzLocation **) a;
    const TzLocation *tzb = * (TzLocation **) b;

    return strcmp (tza->zone, tzb->zone);
}
```

```
static void
sort_locations_by_country (GPtrArray *locations)
{
    qsort (locations->pdata, locations->len, sizeof (gpointer),
           compare_country_names);
}

static void
load_backward_tz (TzDB *tz_db)
{
  GError *error = NULL;
  char **lines, *contents;
  guint i;

  tz_db->backward = g_hash_table_new_full (g_str_hash, g_str_equal,
      g_free, g_free);

  if (g_file_get_contents (GNOME_INTERNET_RADIO_LOCATOR_DATADIR "/
      stations/backward", &contents, NULL, &error) == FALSE)
    {
      g_warning ("Failed to load 'backward' file: %s", error->message);
      return;
    }
  lines = g_strsplit (contents, "\n", -1);
  g_free (contents);
  for (i = 0; lines[i] != NULL; i++)
    {
      char **items;
      guint j;
      char *real, *alias;

      if (g_ascii_strncasecmp (lines[i], "Link\t", 5) != 0)
        continue;

      items = g_strsplit (lines[i], "\t", -1);
      real = NULL;
      alias = NULL;
      /* Skip the "Link<tab>" part */
      for (j = 1; items[j] != NULL; j++)
        {
          if (items[j][0] == '\0')
            continue;
          if (real == NULL)
            {
              real = items[j];
              continue;
            }
          alias = items[j];
          break;
        }

      if (real == NULL || alias == NULL)
        g_warning ("Could not parse line: %s", lines[i]);

      /* We don't need more than one name for it */
      if (g_str_equal (real, "Etc/UTC") ||
          g_str_equal (real, "Etc/UCT"))
        real = "Etc/GMT";
```

```
      g_hash_table_insert (tz_db->backward, g_strdup (alias), g_strdup
          (real));
      g_strfreev (items);
    }
  g_strfreev (lines);
}
```

36.12 gnome-internet-radio-locator-gui.h

```
/* $Id$ */

#ifndef GNOME_INTERNET_RADIO_LOCATOR_GUI_H
#define GNOME_INTERNET_RADIO_LOCATOR_GUI_H

GtkWidget *create_gnome_internet_radio_locator_app(void);

GtkWidget *create_listeners_selector(char *selected_listener_uri,
                     char *filename);
GtkWidget *create_stations_selector(char *selected_station_uri,
                    char *filename);
GtkWidget *create_streams_selector(char *selected_streams_uri,
                   char *filename);
GtkWidget *create_search_selector(void);

GtkWidget *create_new_station_selector(char *location);

GtkWidget *create_gnome_internet_radio_locator_app(void);

#endif /* GNOME_INTERNET_RADIO_LOCATOR_GUI_H */
```

36.13 gnome-internet-radio-locator.h

```
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
```

```
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 */

#ifndef GNOME_INTERNET_RADIO_LOCATOR_H
#define GNOME_INTERNET_RADIO_LOCATOR_H 1

#include <config.h>
#include <glib.h>

typedef enum {
    GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_MONO = 0x0001,
    GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_STEREO = 0x0002,
    GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_5_1 = 0x0005,
    GNOME_INTERNET_RADIO_LOCATOR_CHANNELS_NONE = 0xffff
} GNOMEInternetRadioLocatorChannels;

typedef enum {
    GNOME_INTERNET_RADIO_LOCATOR_STREAM_SHOUTCAST = 0x0001,
    GNOME_INTERNET_RADIO_LOCATOR_STREAM_OGG = 0x0002,
    GNOME_INTERNET_RADIO_LOCATOR_STREAM_AAC = 0x0003
} GNOMEInternetRadioLocatorStreamType;

typedef enum {
    GNOME_INTERNET_RADIO_LOCATOR_STREAM_PLAYER = 0x0001,
    GNOME_INTERNET_RADIO_LOCATOR_STREAM_RECORD = 0x0002
} GNOMEInternetRadioLocatorHelperType;

#include "gnome-internet-radio-locator-listener.h"
#include "gnome-internet-radio-locator-program.h"
#include "gnome-internet-radio-locator-runners.h"
#include "gnome-internet-radio-locator-streams.h"
#include "gnome-internet-radio-locator-station.h"

#if GNOME_INTERNET_RADIO_LOCATOR_DEBUG == 0
#define GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG(x...) g_message(x)
#else
#define GNOME_INTERNET_RADIO_LOCATOR_DEBUG_MSG(x...)
#endif

#define GNOME_INTERNET_RADIO_LOCATOR_RECORD_TRUE 1
#define GNOME_INTERNET_RADIO_LOCATOR_RECORD_FALSE 0
#define GNOME_INTERNET_RADIO_LOCATOR_PLAYER_TRUE 1
#define GNOME_INTERNET_RADIO_LOCATOR_PLAYER_FALSE 0

gchar *str_channels (GNOMEInternetRadioLocatorChannels type);

void show_error(gchar * msg);
void statusbar_send_msg(const char *a, ...);

void appbar_send_msg(const char *a, ...);
gint gnome_internet_radio_locator_archive_new(gchar *title, gchar *file
    , gchar *codec);

/* Fetcher for the channels */
```

```
void gnome_internet_radio_locator_helper_run(char *url, char *name,
    GNOMEInternetRadioLocatorStreamType type,
    GNOMEInternetRadioLocatorHelperType gnome_internet_radio_locator);
void on_search_button_clicked(GtkWidget * button, gpointer user_data);
void on_listen_button_clicked(GtkWidget * button, gpointer user_data);
void on_record_button_clicked(GtkWidget * button, gpointer user_data);
void on_stop_button_clicked(GtkWidget *a, gpointer user_data);
void on_next_station_click(GtkWidget *, gpointer user_data);
void on_new_station_clicked(GtkWidget *a, gpointer user_data);
void on_new_station_selector_changed(GtkWidget *a, gpointer user_data);
void on_previous_station_click(GtkWidget *, gpointer user_data);
void on_listeners_selector_button_clicked(GtkWidget *, gpointer
    user_data);
void on_listeners_selector_changed(GtkWidget * a, gpointer user_data);
void on_programs_selector_button_clicked(GtkWidget *, gpointer
    user_data);
void on_programs_selector_changed(GtkWidget * a, gpointer user_data);
void on_stations_selector_button_clicked(GtkWidget *, gpointer
    user_data);
void on_stations_selector_changed(GtkWidget * a, gpointer user_data);
void on_streams_selector_button_clicked(GtkWidget *, gpointer user_data
    );
void on_streams_selector_changed(GtkWidget * a, gpointer user_data);
void quit_app(GtkWidget *, gpointer user_data);
void about_app(GtkWidget *, gpointer user_data);
void about_listener(GtkWidget *, gpointer user_data);
void about_station(GSimpleAction *simple, GVariant *parameter, gpointer
     user_data);
void about_program(GSimpleAction *simple, GVariant *parameter, gpointer
     user_data);

gboolean on_search_matches(GtkEntryCompletion *widget,
               GtkTreeModel *model,
               GtkTreeIter *iter,
               gpointer user_data);

struct _GNOMEInternetRadioLocatorData {
    GtkImage *pixmap;
    GtkProgressBar *progress;
    GSettings *settings;
    GtkAboutDialog *window;
    GtkStatusbar *statusbar;
    GtkWidget *player_window;
        GtkWidget *about_station;
    gchar *selected_archive_file;
    GNOMEInternetRadioLocatorListenerInfo *selected_listener;
    gchar *selected_listener_uri;
    gchar *selected_listener_name;
    gchar *selected_listener_location;
    gchar *selected_listener_band;
    gchar *selected_listener_description;
    GNOMEInternetRadioLocatorProgramInfo *selected_program;
    gchar *selected_program_uri;
    gchar *selected_program_name;
    gchar *selected_program_location;
    gchar *selected_program_band;
    gchar *selected_program_description;
    GNOMEInternetRadioLocatorRunnersInfo *selected_runners;
    gint timeout_id;
    gint current_station_number;
    GNOMEInternetRadioLocatorStationInfo *previous_station;
    GNOMEInternetRadioLocatorStationInfo *selected_station;
```

```
    gchar *selected_station_uri;
    gchar *selected_station_name;
    gchar *selected_station_location;
    gchar *selected_station_band;
        gchar *selected_station_description;
    gchar *selected_station_website;
    gint selected_bitrate;
    GNOMEInternetRadioLocatorStreamsInfo *selected_streams;
    gchar *selected_streams_mime;
    gchar *selected_streams_uri;
    gchar *selected_streams_codec;
    gchar *selected_streams_samplerate;
    gchar *selected_streams_stations;
    gchar *selected_streams_bitrate;
    GNOMEInternetRadioLocatorChannels selected_streams_channels;
    gint selected_samplerate;
    GdkPixbuf *icon;
    gint player_status;
    GPid player_pid;
    GSubprocess *player_subprocess;
    GSubprocess *record_subprocess;
    GSubprocessLauncher *player_launcher;
    GSubprocessLauncher *record_launcher;
    gint record_status;
    GPid record_pid;
    gint station_count;
    gint stream_count;
    GMainLoop *player_loop;
    GMainLoop *record_loop;
};

typedef struct _GNOMEInternetRadioLocatorData
    GNOMEInternetRadioLocatorData;

void about_station(GSimpleAction *simple, GVariant *parameter, gpointer
     user_data);

extern GNOMEInternetRadioLocatorData *gnome_internet_radio_locator;
extern GList *gnome_internet_radio_locator_listeners;
extern GList *gnome_internet_radio_locator_programs;
extern GList *gnome_internet_radio_locator_stations;
extern GList *gnome_internet_radio_locator_streams;

#endif /* GNOME_INTERNET_RADIO_LOCATOR_H */
```

36.14 gnome-internet-radio-locator-keys.h

```
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
```

```
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 */

#ifndef GNOME_INTERNET_RADIO_LOCATOR_KEYS_H
#define GNOME_INTERNET_RADIO_LOCATOR_KEYS_H

#define GNOME_INTERNET_RADIO_LOCATOR_DOMAIN "org.gnome.gnome-internet-
    radio-locator"
#define GNOME_INTERNET_RADIO_LOCATOR_UI
    GNOME_INTERNET_RADIO_LOCATOR_DOMAIN".ui"
#define GNOME_INTERNET_RADIO_LOCATOR_STATION "station"

#endif /* GNOME_INTERNET_RADIO_LOCATOR_KEYS_H */
```

36.15 gnome-internet-radio-locator-listener.h

```
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 */
```

```
#ifndef GNOME_INTERNET_RADIO_LOCATOR_LISTENER_H
#define GNOME_INTERNET_RADIO_LOCATOR_LISTENER_H

#include <gtk/gtk.h>
#include <glib.h>
#include "gnome-internet-radio-locator.h"

typedef struct _GNOMEInternetRadioLocatorListenerInfo
    GNOMEInternetRadioLocatorListenerInfo;
typedef struct _GNOMEInternetRadioLocatorLocationInfo
    GNOMEInternetRadioLocatorLocationInfo;

struct _GNOMEInternetRadioLocatorListenerInfo {
    gchar *id;
    gchar *location;
    gchar *mail;
    gchar *name;
    gchar *pass;
    gchar *uri;
    gchar *band;
    gchar *description;
    GNOMEInternetRadioLocatorLocationInfo *locationinfo;
    GNOMEInternetRadioLocatorListenerInfo *next;
};

struct _GNOMEInternetRadioLocatorLocationInfo {
    gchar *name;
    gchar *link;
    gchar *glat;
    gchar *glon;
    gchar *grad;
    gchar *vote;
    gchar *rack;
};

GNOMEInternetRadioLocatorListenerInfo *
    gnome_internet_radio_locator_listener_load_from_file(
    GNOMEInternetRadioLocatorListenerInfo * head,
                            char *filename);
GNOMEInternetRadioLocatorListenerInfo *
    gnome_internet_radio_locator_listener_load_from_http(
    GNOMEInternetRadioLocatorListenerInfo * head,
                            gpointer data);

#endif /* GNOME_INTERNET_RADIO_LOCATOR_LISTENER_H */
```

36.16 gnome-internet-radio-locator-markers.h

```
/* $id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2019  Aamot Software
 *
```

```
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses
     />.
 */

/*
 * Copyright (C) 2008 Pierre-Luc Beaudoin <pierre-luc@pierlux.com>
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
     02110-1301  USA
 */

#include <champlain/champlain.h>

#ifndef GNOME_INTERNET_RADIO_LOCATOR_MARKERS_H
#define GNOME_INTERNET_RADIO_LOCATOR_MARKERS_H 1

ChamplainMarkerLayer *create_marker_layer (ChamplainView *view,
    ChamplainPathLayer **path);

void marker_function (ChamplainMarker *self, gdouble dx, gdouble dy,
    ClutterEvent *event, gpointer user_data);

#endif
```

36.17 gnome-internet-radio-locator-player.h

```
/* $Id$
 *
 * GNOME Internet Radio Locator
```

```
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 */

#ifndef GNOME_INTERNET_RADIO_LOCATOR_PLAYER_H
#define GNOME_INTERNET_RADIO_LOCATOR_PLAYER_H

#include <gst/gst.h>
#include <gst/player/player.h>

typedef struct
{
  gchar **uris;
  guint num_uris;
  gint cur_idx;

  GstPlayer *player;
  GstState desired_state;

  gboolean repeat;

  GMainLoop *loop;
} GstPlay;

void play_uri (GstPlayer *player, const gchar * next_uri);

void gnome_internet_radio_locator_player_new (GstPlayer * player, const
     gchar * next_uri);

void gnome_internet_radio_locator_player_new (GstPlayer * player, const
     gchar * next_uri);

void gnome_internet_radio_locator_player_quit (GstPlayer *player);

void gnome_internet_radio_locator_player_pause (GstPlayer *player);

void gnome_internet_radio_locator_player_stop (GstPlayer *player);

static gdouble get_volume (GtkWidget *widget, GstPlay *play);

#endif /* GNOME_INTERNET_RADIO_LOCATOR_PLAYER_H */
```

36.18 gnome-internet-radio-locator-kb.h

```
/* GStreamer command line playback testing utility - keyboard handling
    helpers
 *
 * Copyright (C) 2013 Tim-Philipp M ller <tim centricular net>
 * Copyright (C) 2013 Centricular Ltd
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the
 * Free Software Foundation, Inc., 51 Franklin St, Fifth Floor,
 * Boston, MA 02110-1301, USA.
 */
#ifndef __GST_PLAY_KB_INCLUDED__
#define __GST_PLAY_KB_INCLUDED__

#include <glib.h>

#define GST_PLAY_KB_ARROW_UP    "\033[A"
#define GST_PLAY_KB_ARROW_DOWN  "\033[B"
#define GST_PLAY_KB_ARROW_RIGHT "\033[C"
#define GST_PLAY_KB_ARROW_LEFT  "\033[D"

typedef void (*GstPlayKbFunc) (const gchar * kb_input, gpointer
    user_data);

gboolean gst_play_kb_set_key_handler (GstPlayKbFunc kb_func, gpointer
    user_data);

#endif /* __GST_PLAY_KB_INCLUDED__ */
```

36.19 gnome-internet-radio-locator-player-renderer.h

```
/* GStreamer
 *
 * Copyright (C) 2015 Sebastian Dr ge <sebastian@centricular.com>
 *
 * This library is free software; you can redistribute it and/or
```

```
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the
 * Free Software Foundation, Inc., 51 Franklin St, Fifth Floor,
 * Boston, MA 02110-1301, USA.
 */

#ifndef __GTK_VIDEO_RENDERER_H__
#define __GTK_VIDEO_RENDERER_H__

#include <gst/player/player.h>
#include <gtk/gtk.h>

G_BEGIN_DECLS

typedef struct _GstPlayerGtkVideoRenderer
    GstPlayerGtkVideoRenderer;
typedef struct _GstPlayerGtkVideoRendererClass
    GstPlayerGtkVideoRendererClass;

#define GST_TYPE_PLAYER_GTK_VIDEO_RENDERER             (
    gst_player_gtk_video_renderer_get_type ())
#define GST_IS_PLAYER_GTK_VIDEO_RENDERER(obj)          (
    G_TYPE_CHECK_INSTANCE_TYPE ((obj),
    GST_TYPE_PLAYER_GTK_VIDEO_RENDERER))
#define GST_IS_PLAYER_GTK_VIDEO_RENDERER_CLASS(klass)  (
    G_TYPE_CHECK_CLASS_TYPE ((klass),
    GST_TYPE_PLAYER_GTK_VIDEO_RENDERER))
#define GST_PLAYER_GTK_VIDEO_RENDERER_GET_CLASS(obj)   (
    G_TYPE_INSTANCE_GET_CLASS ((obj),
    GST_TYPE_PLAYER_GTK_VIDEO_RENDERER, GstPlayerGtkVideoRendererClass)
    )
#define GST_PLAYER_GTK_VIDEO_RENDERER(obj)             (
    G_TYPE_CHECK_INSTANCE_CAST ((obj),
    GST_TYPE_PLAYER_GTK_VIDEO_RENDERER, GstPlayerGtkVideoRenderer))
#define GST_PLAYER_GTK_VIDEO_RENDERER_CLASS(klass)     (
    G_TYPE_CHECK_CLASS_CAST ((klass),
    GST_TYPE_PLAYER_GTK_VIDEO_RENDERER, GstPlayerGtkVideoRendererClass)
    )
#define GST_PLAYER_GTK_VIDEO_RENDERER_CAST(obj)        ((
    GstPlayerGtkVideoRenderer*)(obj))

GType gst_player_gtk_video_renderer_get_type (void);

GstPlayerVideoRenderer * gst_player_gtk_video_renderer_new (void);
GtkWidget * gst_player_gtk_video_renderer_get_widget (
    GstPlayerGtkVideoRenderer * self);

G_END_DECLS

#endif /* __GTK_VIDEO_RENDERER_H__ */
```

36.20 gnome-internet-radio-locator-player-resourcer.h

```
#ifndef __RESOURCE_as_H__
#define __RESOURCE_as_H__

#include <gio/gio.h>

extern GResource *as_get_resource (void);
#endif
```

36.21 gnome-internet-radio-locator-program.h

```
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 */

#ifndef GNOME_INTERNET_RADIO_LOCATOR_PROGRAM_H
#define GNOME_INTERNET_RADIO_LOCATOR_PROGRAM_H

#include <gtk/gtk.h>
#include "gnome-internet-radio-locator.h"

typedef struct _GNOMEInternetRadioLocatorProgramInfo
    GNOMEInternetRadioLocatorProgramInfo;
typedef struct _GNOMEInternetRadioLocatorArchiveInfo
    GNOMEInternetRadioLocatorArchiveInfo;

struct _GNOMEInternetRadioLocatorProgramInfo {
```

```
    GtkWidget *widget;
    gchar *id;
    gchar *name;
    gchar *rank;
    gchar *type;
    gchar *band;
    gchar *frequency;
    gchar *location;
    gchar *description;
    gchar *bitrate;
    gchar *samplerate;
    gchar *uri;
    /*   gchar *category; */
    GNOMEInternetRadioLocatorArchiveInfo *archive;
    GNOMEInternetRadioLocatorProgramInfo *next;
    GNOMEInternetRadioLocatorProgramInfo *prev;
};

struct _GNOMEInternetRadioLocatorArchiveInfo {
    gchar *id;
    gchar *name;
    GNOMEInternetRadioLocatorArchiveInfo *next;
    gchar *mimetype;
    glong bitrate;
    glong samplerate;
    GNOMEInternetRadioLocatorChannels channels;
    gchar *uri;
};

GNOMEInternetRadioLocatorProgramInfo *
    gnome_internet_radio_locator_program_new (
    GNOMEInternetRadioLocatorProgramInfo * head,
                  gchar *id,
                  gchar *name,
                  gchar *date,
                  gchar *time,
                  gchar *file);

void gnome_internet_radio_locator_program_free(
    GNOMEInternetRadioLocatorProgramInfo * info);
GNOMEInternetRadioLocatorProgramInfo *
    gnome_internet_radio_locator_program_load_from_file(
    GNOMEInternetRadioLocatorProgramInfo * head,
                        char *filename);
GNOMEInternetRadioLocatorProgramInfo *
    gnome_internet_radio_locator_program_load_from_http(
    GNOMEInternetRadioLocatorProgramInfo * head,
                        gpointer data);

#endif /* GNOME_INTERNET_RADIO_LOCATOR_PROGRAM_H */
```

36.22 gnome-internet-radio-locator-runners.h

```
/* $Id$
```

```
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 */

#ifndef GNOME_INTERNET_RADIO_LOCATOR_RUNNERS_H
#define GNOME_INTERNET_RADIO_LOCATOR_RUNNERS_H

typedef struct _GNOMEInternetRadioLocatorRunnersInfo
    GNOMEInternetRadioLocatorRunnersInfo;

struct _GNOMEInternetRadioLocatorRunnersInfo {
    GtkWidget *widget;
    GMutex *mutex;
    pid_t pid;
    gchar *name;
    gchar *date;
    gchar *time;
    gchar *file;
    gchar *command;
};

GNOMEInternetRadioLocatorRunnersInfo *
    gnome_internet_radio_locator_runners_new (pid_t pid,
                    gchar *name,
                    gchar *date,
                    gchar *time,
                    gchar *file);

void gnome_internet_radio_locator_runners_free(
    GNOMEInternetRadioLocatorRunnersInfo * info);
void gnome_internet_radio_locator_runners_mutex_get(
    GNOMEInternetRadioLocatorRunnersInfo *info);
void gnome_internet_radio_locator_runners_mutex_release(
    GNOMEInternetRadioLocatorRunnersInfo *info);
void gnome_internet_radio_locator_runners_mutex_lock(
    GNOMEInternetRadioLocatorRunnersInfo *info);
void gnome_internet_radio_locator_runners_mutex_unlock(
    GNOMEInternetRadioLocatorRunnersInfo *info);

void cb_record_execute(GtkButton *button);

#endif /* GNOME_INTERNET_RADIO_LOCATOR_RUNNERS_H */
```

36.23 gnome-internet-radio-locator-station.h

```
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 */

#ifndef GNOME_INTERNET_RADIO_LOCATOR_STATION_H
#define GNOME_INTERNET_RADIO_LOCATOR_STATION_H

#include "gnome-internet-radio-locator.h"

typedef struct _GNOMEInternetRadioLocatorStationInfo
    GNOMEInternetRadioLocatorStationInfo;
typedef struct _GNOMEInternetRadioLocatorStreamInfo
    GNOMEInternetRadioLocatorStreamInfo;

typedef enum {
    STATION_LANG = 11,
    STATION_ID = 10,
    STATION_NAME = 0,
    STATION_RANK = 7,
    STATION_TYPE = 6,
    STATION_BAND = 5,
    STATION_FREQUENCY = 4,
    STATION_LOCATION = 1,
    STATION_DESCRIPTION = 3,
    STATION_BITRATE = 8,
    STATION_SAMPLERATE = 9,
    STATION_URI = 2
} GNOMEInternetRadioLocatorStationColumn;

struct _GNOMEInternetRadioLocatorStationInfo {
    GtkWidget *widget;
    gchar *id;
    gchar *name;
    gchar *rank;
    gchar *type;
```

```
    gchar *band;
    gchar *frequency;
    gchar *location;
    gchar *description;
    gchar *bitrate;
    gchar *samplerate;
    gchar *uri;
    gchar *lang;
    /*   gchar *category; */
    GNOMEInternetRadioLocatorStreamInfo *stream;
    GNOMEInternetRadioLocatorStationInfo *next;
    GNOMEInternetRadioLocatorStationInfo *prev;
};

struct _GNOMEInternetRadioLocatorStreamInfo {
    gchar *id;
    gchar *name;
    GNOMEInternetRadioLocatorStreamInfo *next;
    gchar *mimetype;
    glong bitrate;
    glong samplerate;
    GNOMEInternetRadioLocatorChannels channels;
    gchar *uri;
};

void show_error(gchar * msg);
void gnome_internet_radio_locator_station_free(
    GNOMEInternetRadioLocatorStationInfo * info);
GNOMEInternetRadioLocatorStationInfo *
    gnome_internet_radio_locator_station_load_from_file(
    GNOMEInternetRadioLocatorStationInfo * head,
                        char *filename);
GNOMEInternetRadioLocatorStationInfo *
    gnome_internet_radio_locator_station_load_from_http(
    GNOMEInternetRadioLocatorStationInfo * head,
                        gpointer data);
void gnome_internet_radio_locator_helper_run(char *url, char *name,
    GNOMEInternetRadioLocatorStreamType type,
    GNOMEInternetRadioLocatorHelperType gnome_internet_radio_locator);

gint gnome_internet_radio_locator_station_update (
    GNOMEInternetRadioLocatorStationInfo *head, gchar *station_band,
    gchar *station_description, gchar *station_name, gchar *
    station_location, gchar *station_uri, gchar *station_website);

#endif /* GNOME_INTERNET_RADIO_LOCATOR_STATION_H */
```

36.24 gnome-internet-radio-locator-stations-map.h

```
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2015-2019  Aamot Software
```

```
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 *
 */

/*
 * Essential parts of the source code below was based on
 * gnome-control-center/panels/datetime/cc-timezone-map.h
 *
 * Copyright (C) 2010 Intel, Inc
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, see <http://www.gnu.org/licenses/>.
 *
 * Author: Thomas Wood <thomas.wood@intel.com>
 *
 */


#ifndef _GNOME_INTERNET_RADIO_LOCATOR_STATIONS_MAP_H
#define _GNOME_INTERNET_RADIO_LOCATOR_STATIONS_MAP_H

#include <gtk/gtk.h>
#include <gtk/gtkenums.h>

#include "gnome_internet_radio_locator-tz.h"

G_BEGIN_DECLS

#define GNOME_INTERNET_RADIO_LOCATOR_TYPE_STATIONS_MAP
    gnome_internet_radio_locator_stations_map_get_type()

#define GNOME_INTERNET_RADIO_LOCATOR_STATIONS_MAP(obj) \
  (G_TYPE_CHECK_INSTANCE_CAST ((obj), \
  GNOME_INTERNET_RADIO_LOCATOR_TYPE_STATIONS_MAP,
      GNOMEInternetRadioLocatorStationsMap))
```

```
#define GNOME_INTERNET_RADIO_LOCATOR_STATIONS_MAP_CLASS(klass) \
  (G_TYPE_CHECK_CLASS_CAST ((klass), \
  GNOME_INTERNET_RADIO_LOCATOR_TYPE_STATIONS_MAP,
      GNOMEInternetRadioLocatorStationsMapClass))

#define GNOME_INTERNET_RADIO_LOCATOR_IS_STATIONS_MAP(obj) \
  (G_TYPE_CHECK_INSTANCE_TYPE ((obj), \
  GNOME_INTERNET_RADIO_LOCATOR_TYPE_STATIONS_MAP))

#define GNOME_INTERNET_RADIO_LOCATOR_IS_STATIONS_MAP_CLASS(klass) \
  (G_TYPE_CHECK_CLASS_TYPE ((klass), \
  GNOME_INTERNET_RADIO_LOCATOR_TYPE_STATIONS_MAP))

#define GNOME_INTERNET_RADIO_LOCATOR_STATIONS_MAP_GET_CLASS(obj) \
  (G_TYPE_INSTANCE_GET_CLASS ((obj), \
  GNOME_INTERNET_RADIO_LOCATOR_TYPE_STATIONS_MAP,
      GNOMEInternetRadioLocatorStationsMapClass))

typedef struct _GNOMEInternetRadioLocatorStationsMap
    GNOMEInternetRadioLocatorStationsMap;
typedef struct _GNOMEInternetRadioLocatorStationsMapClass
    GNOMEInternetRadioLocatorStationsMapClass;
typedef struct _GNOMEInternetRadioLocatorStationsMapPrivate
    GNOMEInternetRadioLocatorStationsMapPrivate;

struct _GNOMEInternetRadioLocatorStationsMap
{
  GtkWidget parent;

  GNOMEInternetRadioLocatorStationsMapPrivate *priv;
};

struct _GNOMEInternetRadioLocatorStationsMapClass
{
  GtkWidgetClass parent_class;
};

GType gnome_internet_radio_locator_stations_map_get_type (void)
    G_GNUC_CONST;

GNOMEInternetRadioLocatorStationsMap *
    gnome_internet_radio_locator_stations_map_new (void);

gboolean gnome_internet_radio_locator_stations_map_set_location (
    GNOMEInternetRadioLocatorStationsMap *map,
                     const gchar   *timezone);
void gnome_internet_radio_locator_stations_map_set_bubble_text (
    GNOMEInternetRadioLocatorStationsMap *map,
                                     const gchar   *text);

TzLocation *gnome_internet_radio_locator_stations_map_get_location (
    GNOMEInternetRadioLocatorStationsMap *map);

G_END_DECLS

#endif /* _GNOME_INTERNET_RADIO_LOCATOR_STATIONS_MAP_H */
```

36.25 gnome-internet-radio-locator-streams.h

```c
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2014-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
     02110-1301 USA.
 */

#ifndef GNOME_INTERNET_RADIO_LOCATOR_STREAMS_H
#define GNOME_INTERNET_RADIO_LOCATOR_STREAMS_H

#include "gnome-internet-radio-locator.h"

typedef struct _GNOMEInternetRadioLocatorStreamsInfo
    GNOMEInternetRadioLocatorStreamsInfo;
typedef struct _GNOMEInternetRadioLocatorEncoderInfo
    GNOMEInternetRadioLocatorEncoderInfo;

struct _GNOMEInternetRadioLocatorStreamsInfo {
    GtkWidget *widget;
    gchar *mime;
    gchar *uri;
    gchar *codec;
    gchar *samplerate;
    gchar *streams;
    gchar *bitrate;
    GNOMEInternetRadioLocatorChannels channels;
    GNOMEInternetRadioLocatorEncoderInfo *encoder;
    GNOMEInternetRadioLocatorStreamsInfo *next;
    GNOMEInternetRadioLocatorStreamsInfo *prev;
};

struct _GNOMEInternetRadioLocatorEncoderInfo {
    gchar *id;
    gchar *name;
    GNOMEInternetRadioLocatorEncoderInfo *next;
    gchar *mimetype;
    glong bitrate;
    glong samplerate;
    GNOMEInternetRadioLocatorChannels channels;
    gchar *uri;
```

```
};

GNOMEInternetRadioLocatorStreamsInfo *
    gnome_internet_radio_locator_streams_new (
    GNOMEInternetRadioLocatorStreamsInfo * head,
                  gchar *mime,
                  gchar *uri,
                  gchar *codec,
                  gchar *samplerate,
                  gchar *channels,
                             gchar *bitrate);

void gnome_internet_radio_locator_streams_free(
    GNOMEInternetRadioLocatorStreamsInfo * info);
GNOMEInternetRadioLocatorStreamsInfo *
    gnome_internet_radio_locator_streams_load_from_file(
    GNOMEInternetRadioLocatorStreamsInfo * head,
                          char *filename);
GNOMEInternetRadioLocatorStreamsInfo *
    gnome_internet_radio_locator_streams_load_from_http(
    GNOMEInternetRadioLocatorStreamsInfo * head,
                          gpointer data);

#endif /* GNOME_INTERNET_RADIO_LOCATOR_STREAMS_H */
```

36.26 gnome-internet-radio-locator-tz.h

```
/* -*- Mode: C; tab-width: 8; indent-tabs-mode: t; c-basic-offset: 8
    -*- */
/* $Id$
 *
 * GNOME Internet Radio Locator
 *
 * Copyright (C) 2015-2019  Aamot Software
 *
 * Author: Ole Aamot <ole@gnome.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
    02110-1301 USA.
 *
 */
```

```
/*
 * Essential parts of the source code below was based on
 * gnome-control-center/panels/datetime/tz.h
 *
 */

/* Generic timezone utilities.
 *
 * Copyright (C) 2000-2001 Ximian, Inc.
 *
 * Authors: Hans Petter Jansson <hpj@ximian.com>
 *
 * Largely based on Michael Fulbright's work on Anaconda.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, see <http://www.gnu.org/licenses/>.
 */


#ifndef _E_TZ_H
#define _E_TZ_H

#include <glib.h>

#ifndef __sun
#  define TZ_DATA_FILE "/usr/share/zoneinfo/zone.tab"
#else
#  define TZ_DATA_FILE "/usr/share/lib/zoneinfo/tab/zone_sun.tab"
#endif

typedef struct _TzDB TzDB;
typedef struct _TzLocation TzLocation;
typedef struct _TzInfo TzInfo;


struct _TzDB
{
    GPtrArray  *locations;
    GHashTable *backward;
};

struct _TzLocation
{
    gchar *country;
    gdouble latitude;
    gdouble longitude;
    gchar *zone;
    gchar *comment;

    gdouble dist; /* distance to clicked point for comparison */
};
```

```
/* see the glibc info page information on time zone information */
/*  tzname_normal    is the default name for the timezone */
/*  tzname_daylight  is the name of the zone when in daylight savings
    */
/*  utc_offset       is offset in seconds from utc */
/*  daylight         if non-zero then location obeys daylight savings
    */

struct _TzInfo
{
    gchar *tzname_normal;
    gchar *tzname_daylight;
    glong utc_offset;
    gint daylight;
};


TzDB      *tz_load_db                (void);
void       tz_db_free                (TzDB *db);
char *     tz_info_get_clean_name    (TzDB *tz_db,
                       const char *tz);
GPtrArray *tz_get_locations          (TzDB *db);
void       tz_location_get_position  (TzLocation *loc,
                       double *longitude, double *latitude);
char      *tz_location_get_country   (TzLocation *loc);
gchar     *tz_location_get_zone      (TzLocation *loc);
gchar     *tz_location_get_comment   (TzLocation *loc);
glong      tz_location_get_utc_offset (TzLocation *loc);
gint       tz_location_set_locally   (TzLocation *loc);
TzInfo    *tz_info_from_location     (TzLocation *loc);
void       tz_info_free              (TzInfo *tz_info);

#endif
```

Chapter 37

Source Code in gnome-radio-0.2.0.tar.gz

http://www.gnomeradio.org/0.2/gnome-radio-0.2.0.tar.gz

37.1 gnome-radio-file.c

```
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include "gnome-radio-file.h"

extern GList *radio_stations;

#if 0
static void
gnome_radio_file_parser (RadioInfo *info,
             xmlDocPtr doc,
             xmlNodePtr cur)
{
    xmlNodePtr sub;
    g_return_if_fail (info != NULL);
    g_return_if_fail (doc != NULL);
    g_return_if_fail (cur != NULL);
    info->uri = (gchar *) xmlNodeGetProp (cur, (const xmlChar *)"uri");
    sub = cur->xmlChildrenNode;
    while (sub != NULL) {
        if ((!strcasecmp (sub->name, (const xmlChar *) "uri"))) {
            info->uri = (gchar *) xmlNodeListGetString (doc, sub->
                xmlChildrenNode, 1);
            g_print ("station:%s\n", info->uri);
        }
        if ((!strcasecmp (sub->name, (const xmlChar *) "location"))) {
            LocationInfo *location = g_new0 (LocationInfo, 1);
            info->location = location;
            info->location->city = (gchar *) xmlNodeListGetString (doc,
                 sub->xmlChildrenNode, 1);
```

```
            info->location->lat = (gchar *) xmlNodeGetProp (doc, sub->
                xmlChildrenNode);
            info->location->lon = (gchar *) xmlNodeGetProp (doc, sub->
                xmlChildrenNode);
            g_print ("location:city:%s\n", info->location->city);
            g_print ("location:lat:%d\n", info->location->lat);
            g_print ("location:lon:%d\n", info->location->lon);
        }
        if ((!strcasecmp (sub->name, (const xmlChar *) "stream"))) {
            StreamInfo *stream = g_new0 (StreamInfo, 1);
            info->stream = stream;
            info->stream->uri = (gchar *) xmlNodeGetProp (doc, sub->
                xmlChildrenNode);
        }
        sub = sub->next;
    }
    return;
}

RadioInfo *
gnome_radio_file_loader (RadioInfo *head,
             char *filename)
{
    xmlDocPtr doc = NULL;
    xmlNodePtr cur = NULL;
    RadioInfo *curr = NULL;
    RadioInfo *list = NULL;
    g_return_val_if_fail (filename != NULL, NULL);
    doc = xmlReaderForFile (filename, NULL, 0);
    if (doc == NULL) {
        perror ("xmlParseFile");
        xmlDocFree (doc);
        return NULL;
    }
    cur = xmlDocGetRootElement (doc);
    if (cur == NULL) {
        fprintf (stderr, "Empty␣document\n");
        xmlDocFree (doc);
        return NULL;
    }
    if (strcasecmp (cur->name, (const xmlChar *) "radio")) {
        fprintf (stderr,
             "Document␣of␣wrong␣type,␣root␣node␣!=␣radio\n");
        xmlDocFree (doc);
        return NULL;
    }
    cur = cur->xmlChildrenNode;
    while (cur != NULL) {
        if ((!strcasecmp (cur->name, (const xmlChar *) "station"))) {
            curr = g_new0(RadioInfo, 1);
            gnome_radio_file_parser (curr, doc, cur);
            curr->next = head;
            head = curr;
            list = head;
            radio_stations = g_list_append (radio_stations, (RadioInfo
                *)list);
            g_free (curr);
        }
        curr = curr->next;
    }
    xmlDocFree (doc);
    return curr;
```

```
}
#endif
```

37.2 gnome-radio-file.h

```
#ifndef GNOME_RADIO_FILE_H
#define GNOME_RADIO_FILE_H 1

#include <glib.h>
#include <gtk/gtk.h>
#include <gst/player/player.h>

typedef struct _LocationInfo LocationInfo;
typedef struct _RadioInfo RadioInfo;
typedef struct _StreamInfo StreamInfo;
typedef struct _RadioWindow RadioWindow;
typedef struct _RadioOscilloscope RadioOscilloscope;

struct _RadioOscilloscope {
    GtkWidget *window;
        struct timeval *tv;
        struct timezone *tz;
        GstPlayer *player;
    gchar *window_title;
    gchar *text;
    gchar *uri;
};

struct _RadioWindow {
    GtkWidget *window;
    gchar *window_title;
    gchar *text;
    gchar *uri;
};

struct _LocationInfo {
    double *lat;
    double *lon;
    gchar *city;
};

struct _RadioInfo {
    gchar *uri;
    LocationInfo *location;
    StreamInfo *stream;
    RadioInfo *next;
    RadioInfo *prev;
};

struct _StreamInfo {
    gchar *uri;
    StreamInfo *next;
    StreamInfo *prev;
};
```

```
#endif /* GNOME_RADIO_FILE_H */
```

37.3 gnome-radio-main.c

```
#include <config.h>
#include <gtk/gtk.h>
#include <gst/player/player.h>
#include <champlain/champlain.h>
#include <math.h>
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include "gnome-radio-file.h"
#include "gnome-radio-vosc.h"
#define RADIO_MARKER_SIZE 10

#if 0
static void
radio_window_init (RadioWindow *window)
{
    gtk_widget_init_template (GTK_WIDGET (window));
}

static void
radio_window_class_init (RadioWindowClass *class)
{
    gtk_widget_class_set_template_from_resource (GTK_WIDGET_CLASS (
        class),
                            "/org/gtk/gnome-radio/window.ui");
}

static void
search_text_changed (GtkEntry *entry, RadioWindow *window)
{
    RadioWindow *priv;
    const gchar *text;
    GtkWidget *tab;
    GtkWidget *view;
    GtkTextBuffer *buffer;
    GtkTextIter start, match_start, match_end;
    text = gtk_entry_get_text (entry);
    if (text[0] == '\0')
        return;
    priv = radio_window_get_instance_private (window);
    tab = gtk_stack_get_visible_child (GTK_STACK (priv->stack));
    view = gtk_bin_get_child (GTK_BIN (tab));
    buffer = gtk_text_view_get_buffer (GTK_TEXT_VIEW (view));
    /* Very simple-minded search implementation */
    gtk_text_buffer_get_start_iter (buffer, &start);
    if (gtk_text_iter_forward_search (&start, text,
        GTK_TEXT_SEARCH_CASE_INSENSITIVE,
                    &match_start, &match_end, NULL))
    {
```

```
        gtk_text_buffer_select_range (buffer, &match_start, &match_end)
            ;
        gtk_text_view_scroll_to_iter (GTK_TEXT_VIEW (view), &
            match_start,
                        0.0, FALSE, 0.0, 0.0);
    }
}

static void
init_radio_window (RadioWindow *window) {
    gtk_widget_class_bind_template_callback (GTK_WIDGET_CLASS (class),
        radio_search_changed);
}
#endif

/*
  gnome-radio draws the radio_marker wth Cairo composed of 1 static
  filled circle and 1 stroked circle animated as echo.
 */
static ClutterActor *
create_radio_marker (void)
{
    ClutterActor *radio_marker;
    ClutterActor *bg;
    ClutterTimeline *timeline;
    cairo_t *cr;
    /* Create the marker */
    radio_marker = champlain_custom_marker_new ();
    /* Static filled circle ----------------------------------------
        */
    bg = clutter_cairo_texture_new (RADIO_MARKER_SIZE,
        RADIO_MARKER_SIZE);
    cr = clutter_cairo_texture_create (CLUTTER_CAIRO_TEXTURE (bg));
    cairo_set_operator (cr, CAIRO_OPERATOR_CLEAR);
    cairo_paint(cr);
    cairo_set_operator (cr, CAIRO_OPERATOR_OVER);
    /* Draw the circle */
    cairo_set_source_rgb (cr, 0, 0, 0);
    cairo_arc (cr, RADIO_MARKER_SIZE / 2.0,
           RADIO_MARKER_SIZE / 2.0,
           RADIO_MARKER_SIZE / 2.0, 0, 2 * M_PI);
    cairo_close_path (cr);
    /* Fill the circle */
    cairo_set_source_rgba (cr, 0.1, 0.9, 0.1, 1.0);
    cairo_fill (cr);
    cairo_destroy (cr);
    /* Add the circle to the radio_marker */
    clutter_container_add_actor (CLUTTER_CONTAINER (radio_marker), bg);
    clutter_actor_set_anchor_point_from_gravity (bg,
        CLUTTER_GRAVITY_CENTER);
    clutter_actor_set_position (bg, 0, 0);
    /* Echo circle ------------------------------------------------ */
    bg = clutter_cairo_texture_new (2 * RADIO_MARKER_SIZE,
                    2 * RADIO_MARKER_SIZE);
    cr = clutter_cairo_texture_create (CLUTTER_CAIRO_TEXTURE (bg));
    /* Draw the circle */
    cairo_set_source_rgb (cr, 0, 0, 0);
    cairo_arc (cr, RADIO_MARKER_SIZE, RADIO_MARKER_SIZE,
           0.9 * RADIO_MARKER_SIZE, 0, 2 * M_PI);
    cairo_close_path (cr);
    /* Stroke the circle */
    cairo_set_line_width (cr, 2.0);
```

```
    cairo_set_source_rgba (cr, 0.1, 0.7, 0.1, 1.0);
    cairo_stroke (cr);
    cairo_destroy (cr);
    /* Add the circle to the radio_marker */
    clutter_container_add_actor (CLUTTER_CONTAINER (radio_marker), bg);
    clutter_actor_lower_bottom (bg); /* Ensure it is under the previous
         circle */
    clutter_actor_set_position (bg, 0, 0);
    clutter_actor_set_anchor_point_from_gravity (bg,
                              CLUTTER_GRAVITY_CENTER);
    /* Animate the echo circle */
    timeline = clutter_timeline_new (1000);
    clutter_timeline_set_loop (timeline, TRUE);
    clutter_actor_set_opacity (CLUTTER_ACTOR (bg), 255);
    clutter_actor_set_scale (CLUTTER_ACTOR (bg), 0.5, 0.5);
    clutter_actor_animate_with_timeline (CLUTTER_ACTOR (bg),
                         CLUTTER_EASE_OUT_SINE,
                         timeline,
                         "opacity", 0,
                         "scale-x", 2.0,
                         "scale-y", 2.0,
                         NULL);
    clutter_timeline_start (timeline);
    return radio_marker;
}

double lat = 21.293352;
double lon = -157.839583;

typedef struct
{
    ChamplainView *view;
    ChamplainMarker *radio_marker;
} GpsCallbackData;

typedef struct
{
        RadioOscilloscope *oscilloscope_visual;
} OscilloscopeCallbackData;

static gboolean
gps_callback (GpsCallbackData *data)
{
    champlain_view_center_on (data->view, lat, lon);
    champlain_location_set_location (CHAMPLAIN_LOCATION (data->
        radio_marker), lat, lon);
    return TRUE;
}

gint
main (gint argc, gchar **argv)
{
    GstPlayer *player;
    GtkWidget *window;
    ChamplainView *view;
    ClutterActor *actor, *radio_oscilloscope, *radio_marker, *
        oscilloscope_visual, *stage;
    ChamplainMarkerLayer *layer;
    RadioInfo *radioinfo;
    GpsCallbackData callback_data;
    /* OscilloscopeCallbackData oscilloscope_data; */
    /* VOSCWindow *vosc; */
```

```
    if (clutter_init (&argc, &argv) != CLUTTER_INIT_SUCCESS)
        return 1;
    /* vosc = (VOSCWindow *)g_new0(VOSCWindow, 1); */
    stage = clutter_stage_new ();
    clutter_stage_set_title (stage, g_strconcat("GNOME␣Radio", "␣",
        VERSION, "␣-␣", "http://www.gnomeradio.org/", "␣-␣", "https://
        wiki.gnome.org/Apps/Radio", NULL));
    clutter_actor_set_size (stage, 800, 600);
    g_signal_connect (stage, "destroy", G_CALLBACK (clutter_main_quit),
         NULL);
    /* Create the map view */
    actor = champlain_view_new ();
    clutter_actor_set_size (CLUTTER_ACTOR (actor), 800, 600);
    clutter_container_add_actor (CLUTTER_CONTAINER (stage), actor);
    /* Create the radio_marker layer */
    layer = champlain_marker_layer_new_full (CHAMPLAIN_SELECTION_SINGLE
        );
    clutter_actor_show (CLUTTER_ACTOR (layer));
    champlain_view_add_layer (CHAMPLAIN_VIEW (actor), CHAMPLAIN_LAYER (
        layer));
    /* Create a radio_marker */
    radio_marker = create_radio_marker ();
    champlain_marker_layer_add_marker (layer, CHAMPLAIN_MARKER (
        radio_marker));
    /* Create a oscilloscope_visual */
    /* oscilloscope_visual = create_oscilloscope_visual (); */
        /* gnome_radio_add_visual_oscilloscope (layer,
            GNOME_RADIO_MARKER (oscilloscope_visual)); */
    /* Finish initialising the map view */
    g_object_set (G_OBJECT (actor), "zoom-level", 1,
              "kinetic-mode", TRUE, NULL);
    champlain_view_center_on (CHAMPLAIN_VIEW (actor), lat, lon);
    /* Create callback that updates the map periodically */
    callback_data.view = CHAMPLAIN_VIEW (actor);
    callback_data.radio_marker = CHAMPLAIN_MARKER (radio_marker);
    /* oscilloscope_data.view = GNOME_RADIO_VIEW (radio_oscilloscope);
        */
        /* oscilloscope_data.oscilloscope_visual = GNOME_RADIO_MARKER (
            oscilloscope_visual); */
    /* Create the radio player */
    player = gst_player_new (NULL,
        gst_player_g_main_context_signal_dispatcher_new(NULL));
    /* gnome_radio_file_loader (radioinfo, "gnome-radio.xml"); */
    gst_player_set_uri (GST_PLAYER (player), "http://khpr-ice.
        streamguys1.com:80/khpr2");
    gst_player_stop (GST_PLAYER (player));
    /* Visual Oscillator */
        /* vosc->window = gtk_window_new (GTK_WINDOW_TOPLEVEL); */
        /* gtk_widget_show_all (vosc->window); */
        /* gnome_radio_real(GST_PLAYER (player), CLUTTER_ACTOR (
            radio_oscilloscope)); */
        /* clutter_container_add_actor (CLUTTER_CONTAINER (stage),
            CLUTTER_ACTOR (radio_oscilloscope)); */
    gst_player_play(GST_PLAYER (player));
    g_timeout_add (1000, (GSourceFunc) gps_callback, &callback_data);
    /* g_timeout_add (1000, (GSourceFunc) gnome_radio_real, &
        oscilloscope_data); */
    clutter_actor_show (stage);
        /* clutter_actor_show (radio_oscilloscope); */
    clutter_main ();
    return (0);
}
```

```
<radio>
  <station name="WNYC" uri="http://www.wnyc.org/">
    <location lat="40.7265342" lon="-74.0054459">New York City</
        location>
    <stream uri="http://fm939.wnyc.org/wnycfm" />
  </station>
  <station name="SCPR" uri="https://www.scpr.org/">
    <location lat="34.1376417" lon="-118.1507543">Southern California</
        location>
    <stream uri="http://live.scpr.org/kpcclive/" />
  </station>
  <station name="KHPR" uri="https://www.hawaiipublicradio.org/">
    <location lat="21.293352" lon="-157.839583">Hawaii</location>
    <stream uri="http://khpr-ice.streamguys1.com:80/khpr2" />
  </station>
</radio>
```

Part VI

Software Packages for i386, x86_64 and amd64

Debian GNU/Linux, Fedora and Ubuntu software installation packages of GNOME Internet Radio Locator for the computer hardware architecture i386, x86_64, and amd64.

https://www.gnome.org/~ole/debian/gnome-internet-radio-locator_3.0.0-1_i386.deb

https://www.gnome.org/~ole/fedora/RPMS/x86_64/gnome-internet-radio-locator-3.0.0-1.fc31.x86_64.rpm

https://www.gnome.org/~ole/ubuntu/gnome-internet-radio-locator_3.0.0-1_amd64.deb

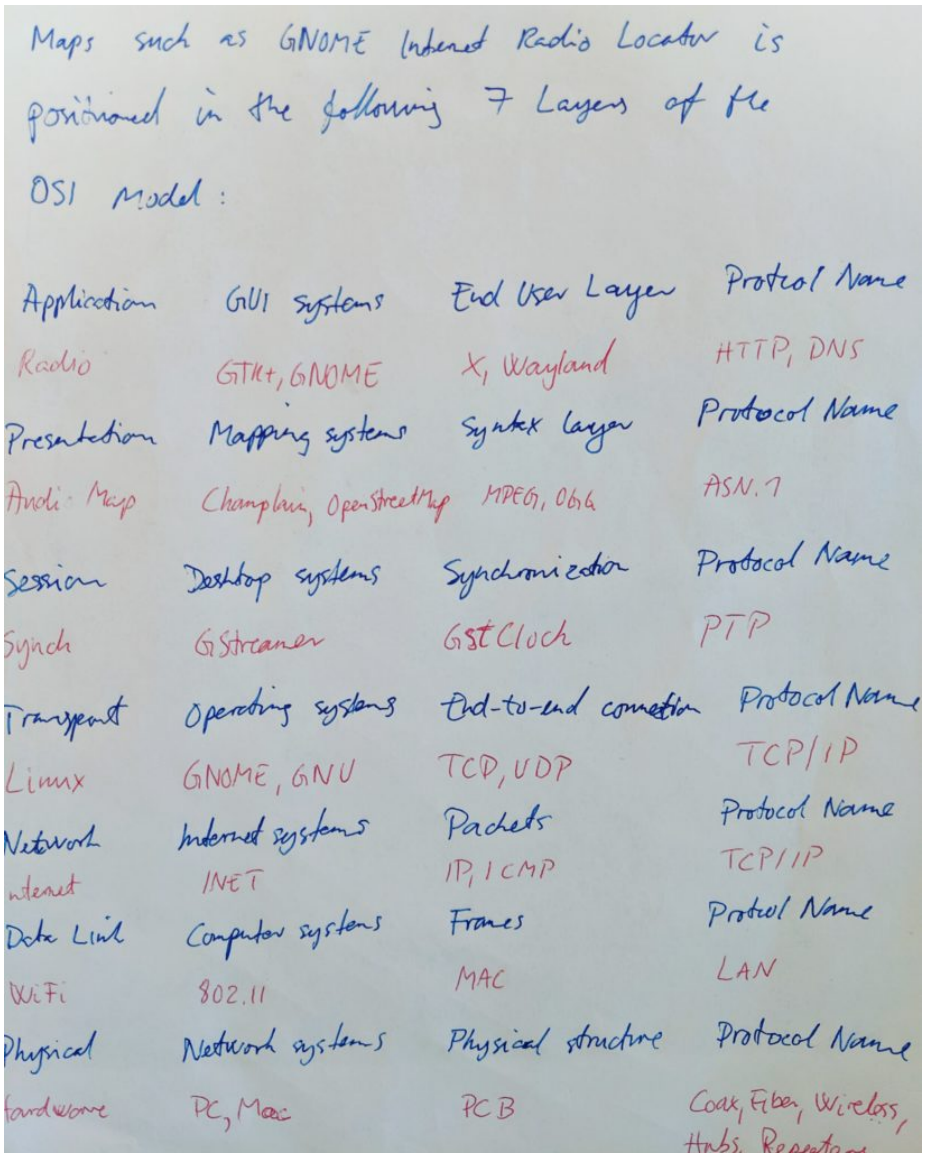
Part VII

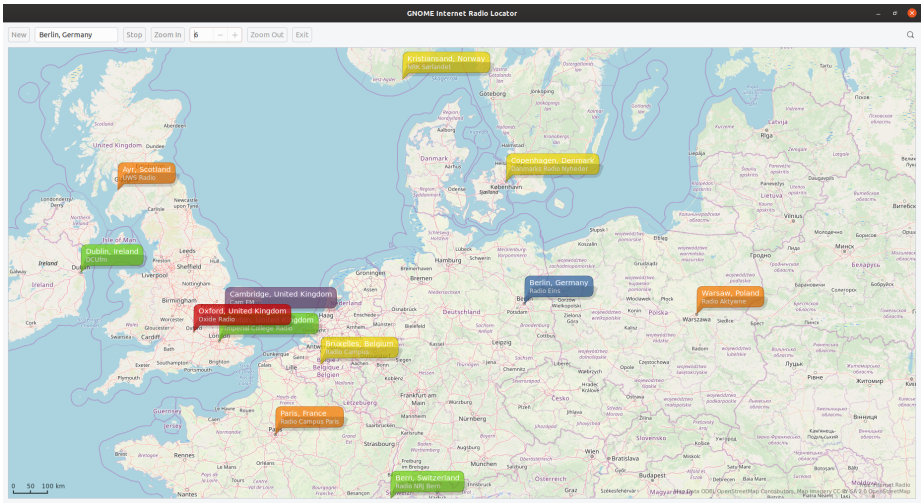
Figures and Layouts

Chapter 38

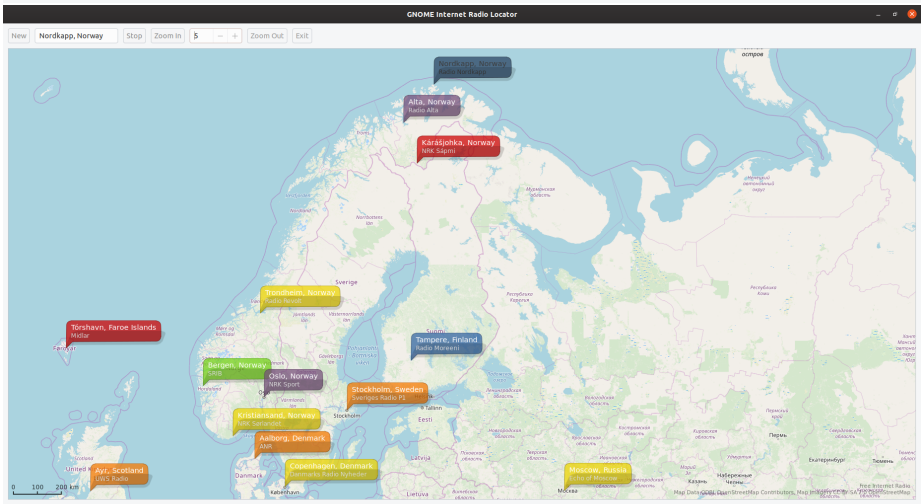
Illustrations

The following illustrations document the position of radio in the open systems interconnection (OSI) model, and its free implementation as a graphical user interface software package GNOME Internet Radio Locator (gnome-internet-radio-locator) and GNOME Radio (gnome-radio), implemented as Free Software for Debian GNU/Linux, Fedora Linux, Ubuntu Linux, and MacPorts.


Maps such as GNOME Internet Radio Locator is positioned in the following 7 Layers of the OSI Model:
Application
GUI systems
End User Layer
Protocol Name
Radio
GTK+, GNOME
X, Wayland
HTTP, DNS
Presentation
Mapping systems
Syntax Layer
Protocol Name
Audio Map
Champlain, OpenStreetMap
MPEG, OGG
ASN.1
Session
Desktop systems
Synchronization
Protocol Name
Synch
GStreamer
GstClock
PTP
Transport
Operating systems
End-to-end connection
Protocol Name
Linux
GNOME, GNU
TCP, UDP
TCP/IP
Network
Internet systems
Packets
Protocol Name
Internet
INET
IP, ICMP
TCP/IP
Data Link
Computer systems
Frames
Protocol Name
WiFi
802.11
MAC
LAN
Physical
Network systems
Physical structure
Protocol Name
Hardware
PC, Mac
PCB
Coax, Fiber, Wireless,
Hubs, Repeaters



Radio Eins (http://www.radioeins.de/) in Berlin, Germany (Online, 44100 Hz, 128 kbps)



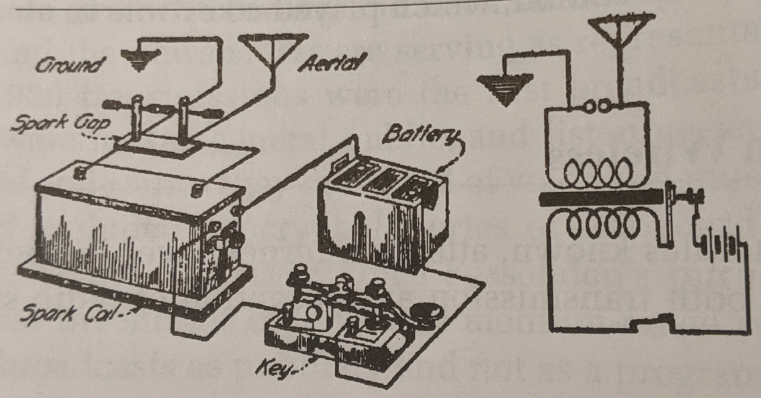
Radio Nordkapp (http://www.radionordkapp.no/) in Nordkapp, Norway (103.9FM, 44100 Hz, 54 kbps)

Chapter 39

Electronical Layouts

39.1 Wireless Transmission

Typical Early Wireless Transmitter Taken from a typical book of the time intended for radio experimenters, this is both a drawing and an electrical schematic of a spark-gap transmitter (to the left)—the standard wireless transmitter from the time of Hertz in the 1880s through World War I. That it was used for wireless telegraphy (code) and not telephony (voice and music) is evident by the telegraph key. With a battery, the system was self-contained, not requiring plug-in sources of power. The receiving end might consist of a coherer (in the earliest days), and after 1900 a crystal detector as is shown in the box on page 90.



Source: Yates and Pacent (1922), page 61.

39.2 Radio Transmission

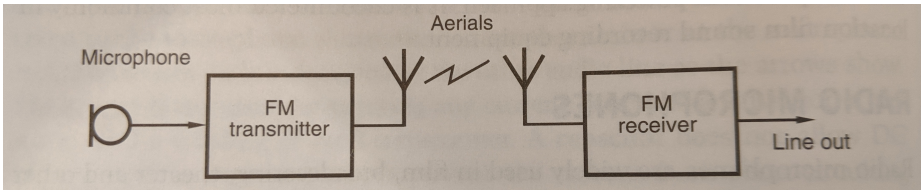


FIGURE 3.15 *A radio microphone incorporates an FM transmitter, resulting in no fixed link between microphone and mixer.*

39.3 Microphone

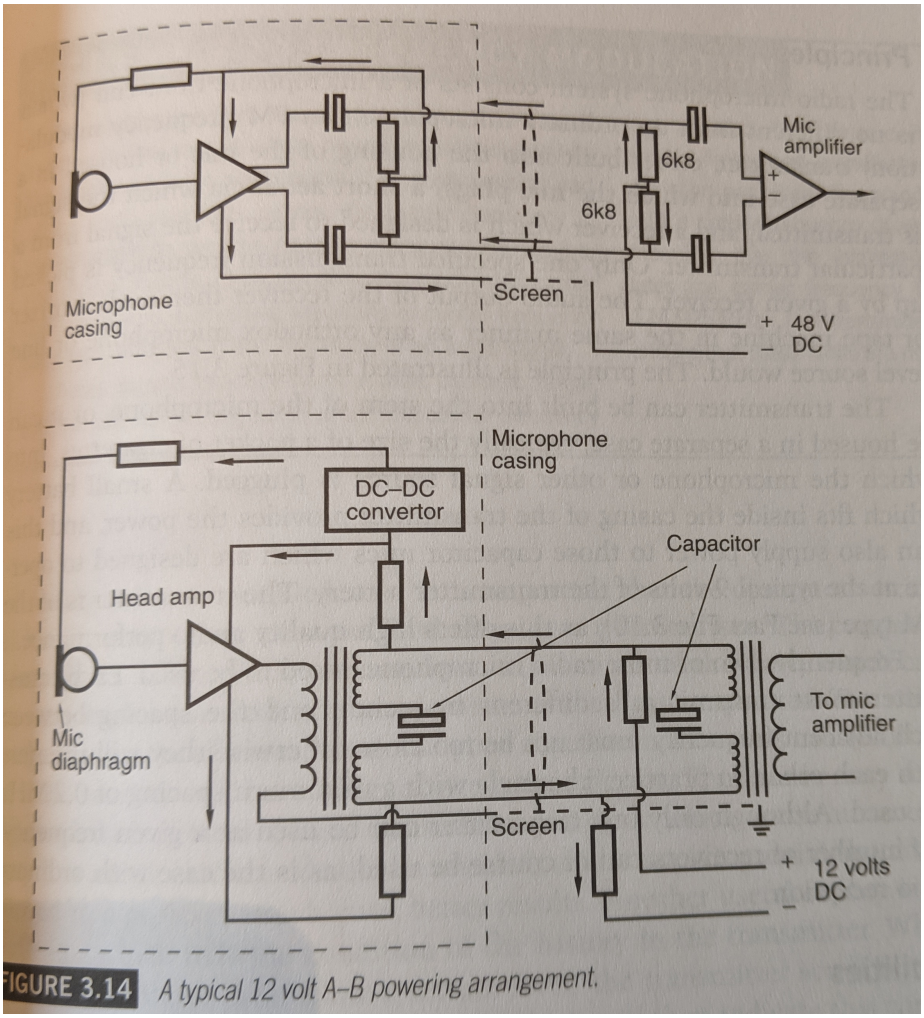
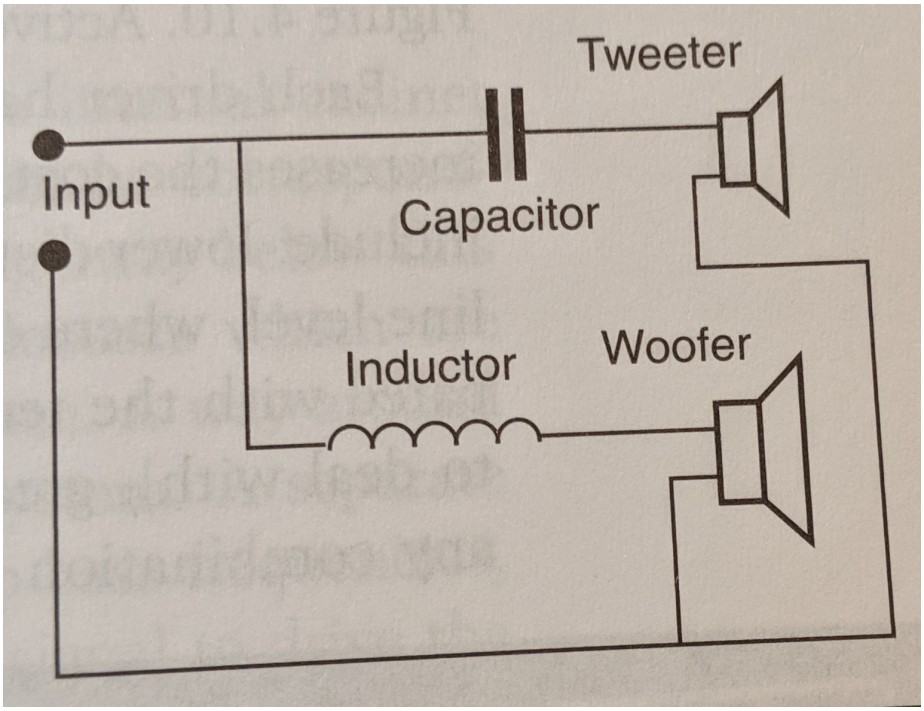


FIGURE 3.14 *A typical 12 volt A–B powering arrangement.*

39.4 Loudspeaker



Part VIII

Classification of radio

When you publish music, photographs or video digitally, the physical representation of a scenery or humans visible in the exposured digital media can be labeled as places or with personal names and locations. Apple, Facebook, Google, Spotify and Twitter are software for classified information. This is more difficult with live radio in GNOME Radio, but the near exact location of the radio station broadcasting live voices of humans can't be recorded and categorized. In GNOME Internet Radio Locator no humans are identified on the broadcast, but the characteristic human voices of the presenters and debaters are differentiated by the listener.

Part IX

Human vs. algorithmic curation of music on radio

In this Bachelor thesis I have advocated for Free Software and human music curation in Free Internet Radio in my thesis.

Google Music and Spotify Radio is using a proprietary software algorithm to select what music to play. With Google Music and Spotify Radio the user will listen to the same music she has been listening to because the algorithms lack human curation. Apple Music is promoting Beats 1, a human curated, but monopolistic radio station.

Support for human curation from Spotity is discussed in this interview: https://www.theverge.com/2015/9/30/9416579/spotify-discover-weekly-online-music-curation-interview

> "This past June, legendary producer and major label insider Jimmy Iovine unveiled Apple Music as the grand finale of the tech titan's semi-annual product showcase. "It's a revolutionary music service curated by the leading music experts who we helped handpick," he declared, placing Apple firmly in the human curation camp. Apple's curation service, Iovine promised, would match the song you hear to the mood and the moment. "Algorithms alone can't do that emotional task. You need a human touch."

Free Internet Radio is still a better alternative for listeners than Apple Music, Google Music and Spotify Radio, because of the human curation of music that is played on the air on a computer running Free Software is supported or funded by its listeners.

Many stations included in gnome-internet-radio-locator have a audio message that says "You are listening to Free Internet Radio sponsored by listeners like yourself. To donate, visit www.¡STATION SIGN¿.org". Allthough these recorded messages repeats every time you switch station, most Internet radio stations are operated by humans.

We support the element of human influence and consideration of music and debates in Free Internet Radio being funded and supported by the People and for the People, not by a billion dollar company such as Apple with human curation of a monopolistic station or by Google and Spotify with proprietary computer curation algorithms that makes you listen to the same music over and over again.

Hypothesis

Music on radio shouldn't be curated by a computer algorithm, but by humans.

Conclusion

Wrong at times, since it's on a computer, perhaps correct at Trump rallying events, but in the end he will be forgotten on the radio.

Part X

Rescue and emergency communication over public radio

In the case of evacuation of a physical area, due to Force Majeure events such as hurricanes, fires and wars, where a life-saving police, ambulance and hospital team would save lives during such events, the rescue and emergency work is coordinated and communicated over public radio broadcasts.

Listen to radio for safety information about COVID-19 and other crisis.

Part XI

Essay: Democratic voices on public radio in Norway as Free Software (gnomeradio.org)

Ole Kristian Aamot (BS EE)
olekaa@math.uio.no
Department of Mathematics
University of Oslo, Norway

Norway is a democratic country with a public government and a minister cabinet choosen in democratic elections.

Hospitals, kindergartens, schools and universities prepare and support the voters for the Norwegian inclusive work and welfare state.

Several free and independent radio stations in Norway broadcast democratic debates on the progress of our country and civil society in peace.

The radio stations in Norway that are available in GNOME Internet Radio Locator (gnome-internet-radio-locator) are NRK P1, NRK P2, NRK P3, NRK P11, NRK Alltid Nyheter, as well as regional broadcasts in NRK Østfold, NRK Vestfold, NRK Rogaland, NRK Hordaland, NRK Sørlandet, NRK Sogn og Fjordane, NRK Møre og Romsdal, NRK Trøndelag, NRK Nordland, NRK Troms og Finnmark that broadcast daily radio programs such as the hourly, daily and weekly news reports NRK Dagsnytt, NRK Morgennytt, NRK Formiddagsnytt, NRK Ettermiddagsnytt, NRK Dagsrevyen and NRK Kveldsnytt as well as the debate programs NRK Dagsnytt 18 and NRK Debatten. In addition to NRK, there is support for 3 Public University student radio stations such as Radio NOVA at University of Oslo, Studentradioen i Bergen at University of Bergen, Radio Revolt at NTNU in Trondheim, Norway in the public Internet Radio client GNOME Internet Radio Locator described in the Bachelor thesis "Public Internet Radio Client for Accessing Free Audio Maps in Countries with Free Speech."

http://folk.uio.no/olekaa/thesis/bachelor/AAMOT-2020.pdf

The Norwegian democracy is upheld and protected by equal opportunities for moderate and often radical voices on radio through sound arguments between peers such as petitions from the left to the right in the Norwegian democracy.

The elected party leader representatives such as Lan Marie Nguyen Berg (Miljøpartiet De Grønne), Erna Solberg (Høyre), Trine Skei Grande (Venstre), Jonas Gahr Støre (Arbeiderpartiet), Audun Lysbakken (Sosialistisk Venstreparti), Bjørnar Moxnes (Rødt), Kjell Ingolf Ropstad (Kristelig Folkeparti), Siv Jensen (Fremskrittspartiet) and Trygve Slagsvold Vedum (Senterpartiet) must argue in public debates on NRK Radio and TV, as representatives for the public opinion, as they argue for political beliefs and convictions on the best progress for the voting population in Norway.

Part XII

Binary Delivery

http://folk.uio.no/olekaa/thesis/bachelor/AAMOT-2020.zip

Part XIII

Installation Instruction for Public Internet Radio Client

Chapter 40

Installation on GNU/Linux

40.1 Debian GNU/Linux 10 i386

dpkg -i https://www.gnome.org/ ole/debian/gnome-internet-radio-locator$_3$.0.0−1$_i$386.*deb*

40.2 Debian GNU/Linux 9 amd64

dpkg -i https://www.gnome.org/ ole/debian/gnome-internet-radio-locator$_2$.8.0−1$_a$*md*64.*deb*

40.3 Fedora Core 31 x86_64

rpm -Uvh https://www.gnome.org/ ole/fedora/RPMS/x86$_6$4/*gnome*−*internet*−*radio* − *locator* − 3.0.0 − 1.*fc*31.*x*86$_6$4.*rpm*

40.4 Fedora Core 30 x86_64

rpm -Uvh https://www.gnome.org/ ole/fedora/RPMS/x86_64/gnome-internet-radio-locator-2.1.1-1.fc30.x86_64.rpm

40.5 Fedora Core 29 x86_64

rpm -Uvh https://www.gnome.org/ ole/fedora/RPMS/x86_64/gnome-internet-radio-locator-2.0.0-1.fc29.x86_64.rpm

40.6 Ubuntu 19.10 amd64

dpkg -i https://www.gnome.org/ ole/ubuntu/gnome-internet-radio-locator_3.0.0-1_amd64.deb

40.7 Ubuntu 19.04 amd64

dpkg -i https://www.gnome.org/ ole/ubuntu/gnome-internet-radio-locator_3.0.0-1_amd64