PiPtr Documentation.
Rev 0.0 April 14, 2017
James Lee N1DDK

Revision	Date	Who	Description
0.0	April 14,2017	James Lee	Initial document

Notes:

Table of Contents

Introduction

This repeater control project is more than yet another open source ham radio repeater cotroller.

Take a moment to read at least th enext few paragraphs before diving into and of the latter detailed sections.

Also donations are welcome through PayPal to jml@jmlzone.com or through mmra.org.

Why?

The Minute Man repeater Association (www.mmra.org) maintains a network of over 20 repeaters throughout eastern and metro west Massachusetts. These repeaters are controlled by a variety of different controllers with obscure programming and control methods. While the technical committee for these repeaters has tried to minting a constant model, very few control operators are able to easily configure and maintain the system. When using commercial repeater controllers, the controllers tend to be over $150 per port and have limited expansion capabilities for large sites or are over kill with more ports than needed for a remote site.
Other issues seen on the MMRA network are:

- Difficulties with clock synchronization on the controllers on the network.
- Occasional random behavior from aging EEPROMs, lighting struck hardware and other controller hardware flakiness.
- Lack of source control and documentation for the repeater programming.
- long lag times when network comes up due to cascaded slow controller response.
- Limited internet control

Goals

- Open source hardware and software for long term wide spread support.
- Cost effective goal of $50 per port.
- Fast and flexible networking
- Extensive internet and non internet control
- Accurate clocks
- Software upgradable.

Why not?

Why not use open repeater or some of the other open source projects out there?

- The may not be as “open” as the name suggests.
- The may not be designed for linking and some of our other goals.

Basic Architecture

The basic design is a add on board for a Raspberry Pi 2, 3 or beyond with minimal hardware to provide a full high end repeater controller.
Schematics will be captured in eagle.
Software in Python and C primarily. The 2 most common current programming languages.

There are 2 radio ports per controller and the controllers have side cascading ports to build up in groups of 2. See example network topologies later in the document a site with 7 repeaters, a remote base and 2 link radios would be configured with 5 of these controllers and all linking combinations are possible!

Custom Hardware

The custom hardware consists of the radio interface and the audio path. It include environmental monitoring to see if you left the lights on in the shelter or if the temperature or humidity is out of bounds. Built in measuring of the 12v repeater power supply and several available user analog channels.
The main rule of thumb is zero jumper and zero mechanical potentiometers. All adjustments are made via software controllable elements.

Audio

Audio is routed in and out of the computer via standard inexpensive USB sound cards.

- There is an analog audio path for a repeater to stand alone.
- There is an analog audio path for the 2 repeaters on a controller to link.
- There are 2 analog audio paths for controller to controller linking.
- Additionally since the audio goes in and out of the computer, it is possible to do digital audio formats and or digital repeater linking all in software. This is TBD and currently beyond the scope of the base project.

Digital Audio Functions

Base digital audio decoding functions include:

- Touch tone decoding
- Soft CTCSS decoding

Basic Audio sending functions include:

- Morse generation
    - ASCII to morse
    - Any tone frequency (specified in Hertz)
    - Any speed (specified in Words per minute)
    - Any amplitude (specified in arbitrary volume units)
- Tone generation
    - Any simple tone or sequence of tones
    - Specified as groups of three numbers:
        - Tone in Hertz

- Duration in mili-Seconds
- Amplitude
- Delays between beeps are just tones of zero amplitude

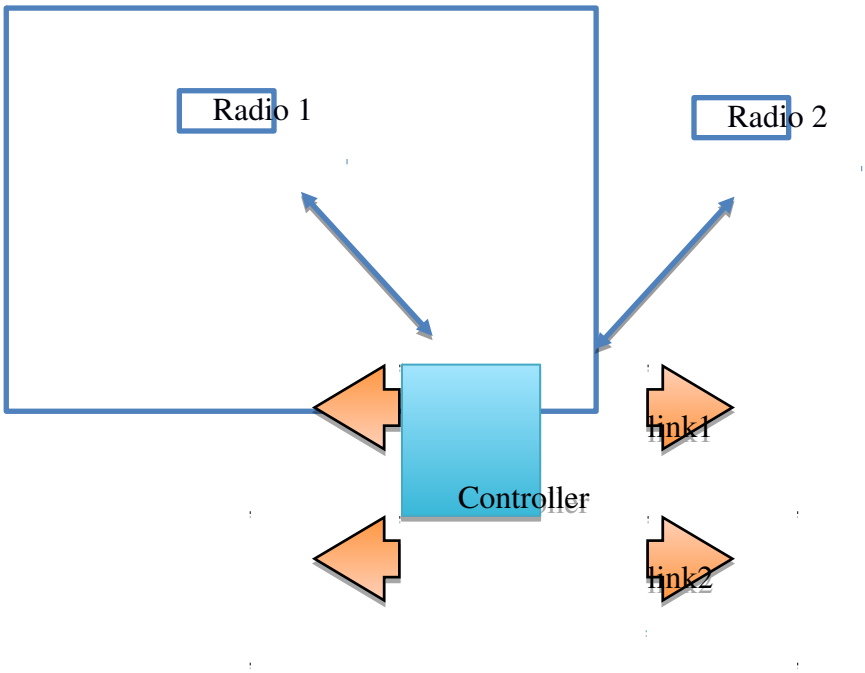
- Speech Generation
  - Text to speech using one available open source programs
- Wave file play back

Soft modem capabilities allow nodes at remote sites to form an ad-hoc data network a link radio to allow remote controlling, updating and time synchronization.

Example Network Topologies

A Basic Node

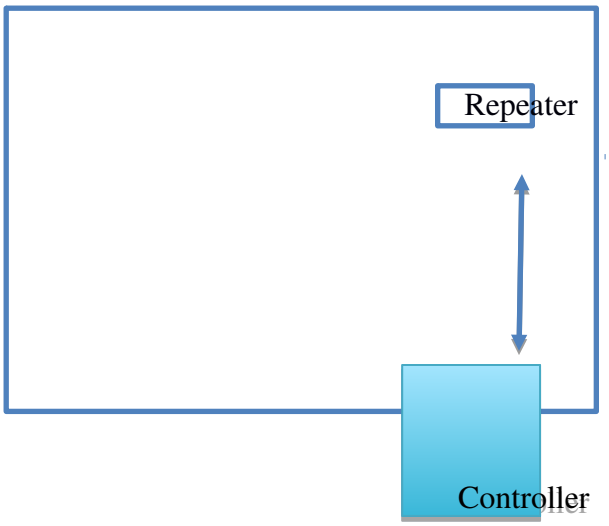
The basic topology of a single repeater node is 2 radio ports.
The radio ports may be either full duplex repeaters or link/remote base radios.
There are also 2 link ports to link to other controllers.



Drawing 1: Basic Node

A Single Repeater

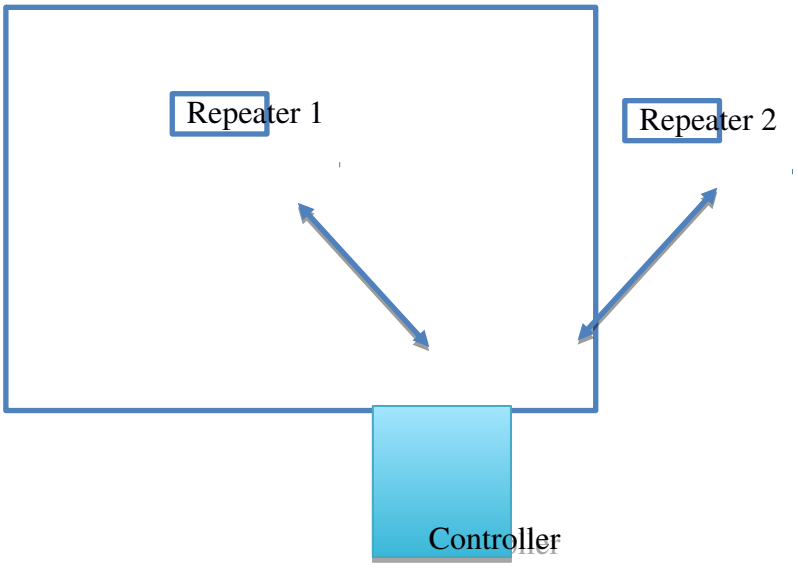
A single repeater is the simplest configuration. The repeater would be connected to radio port 1 or 2. thats all.



Drawing 2: A single repeater configuration

2 Repeaters

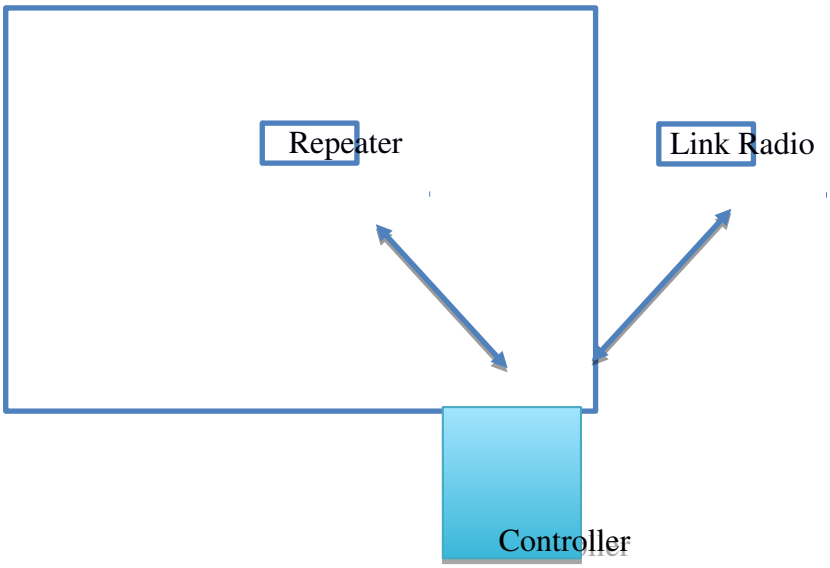
With 2 repeaters a repeater would be connected to each radio port. By default they will be independent repeaters; however the repeaters are completely independently controllable or easily linkable.



Drawing 3: 2 repeater configuration

A Repeater and a link radio (or remote base)

A link or remote base is connected similarly to a repeater, except it is used half duplex and tail hangs and messaging on the link or remote base radio is minimized.

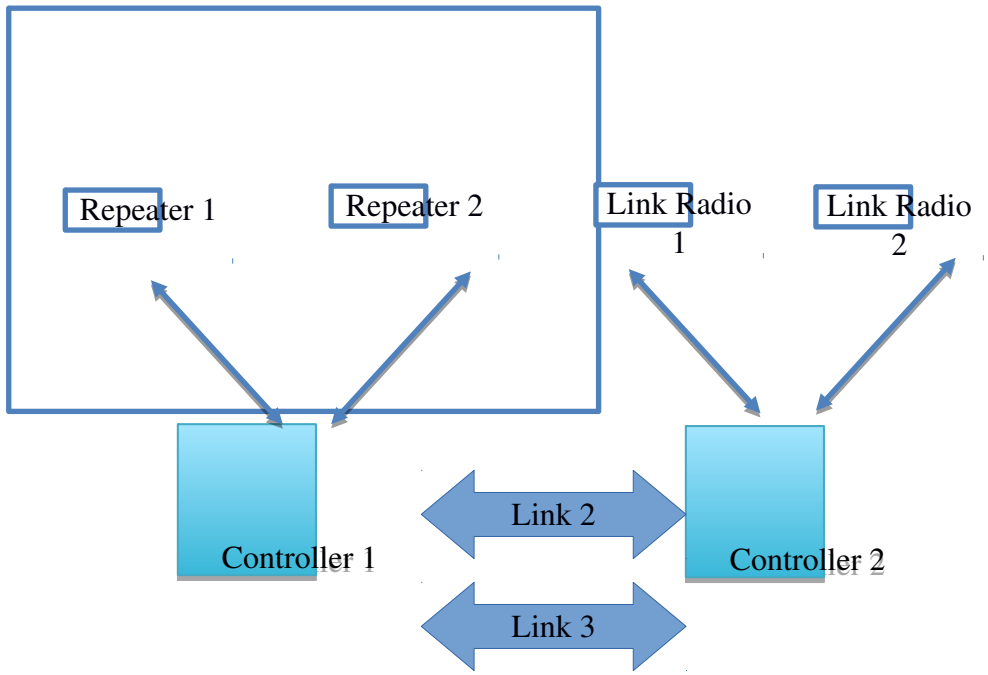


Drawing 4: Single repeater with link

2 Repeaters With 2 Link Radios

In this configuration it is possible to group a repeater and link on a single controller like above, or you could put both repeaters on once controller and both link radios on the second controller. In the recommended configuration shown below, the controllers would be configured as follows:

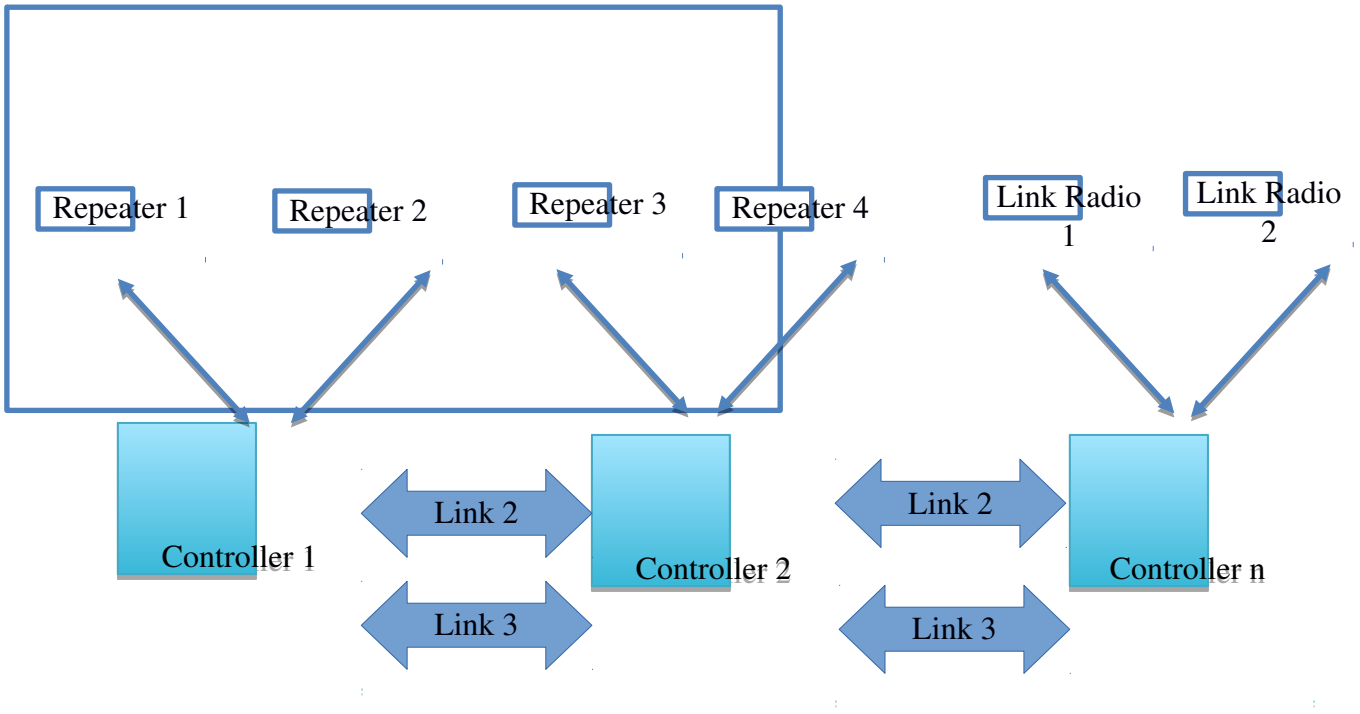
- On the repeater controller
  - Both radio ports configured as repeaters
  - The repeaters can be cross linked by connecting both to link 1, the internal link.
  - Either (or both) repeater can be connected to link 2 or link 3 to use the link radios
- On the link radio controller
  - Both ports configured as Links
  - Link radio 1 configured to link 2
  - Link radio 2 configured to link 1



Drawing 5: 2 repeaters, 2 controllers and 2 links

More

The examples shown in this section are just examples. You can have as many controllers connected to each other as you like. Also it is not required to connect all link2's to link2's and link 3's to link 3's so you can have additional logs and global links as makes sense for your network.



Drawing 6: A bigger configuragtion

Remember:

- All radio ports can act as fully independent repeaters.
- Link 1 exists to allow the 2 ports on a controller to link.
- Link 2 and Link 3 are simple connections to physically co-located radios and repeaters.

Hardware Description

Radio Port

The radio port comes out on a 10 pin header (only 9 pins are used). This may be cabled with a simple ribbon cable to a 9 pin D connector DE9F to be compatible with radios bailed for an Scom 7330 controller.

Header Pin	De 9 pin	Function
1	1	Receiver audio
2	6	GND (RX audio)
3	2	COR (logic)
4	7	GND (CTCSS audio)
5	3	CTCSS decode (logic / or discriminator audio)
6	8	Transmitter (CTCSS audio or logic control)
7	4	PTT
8	9	GND (Transmit Audio)
9	5	Transmitter audio
10	NA	Physical key

However there is more capability than a basic Scom controller.

Receiver Logic Connections

The COR and CTCSS inputs can be logic level from 1.6V to 30V for a logic ‘1’ and less than 0.6V for a logic ‘0’ The polarity of the logic does not matter since it can be inverted in the configuration. There are no jumpers to set. If they are not used, the pins can be left un-connected.

CTCSS

If a hardware CTCSS decoder is not part of the radio hardware it is not needed. Discriminator audio may be brought in on the CTCSS pin or of the receiver does not high pass the audio to filter out the CTCSS tones, the CTCSS Tones will be decoded off the receive audio.

DTMF

The DTMF tone will be decoded off the discriminator or receive audio, which every was used for the CTCSS.

Transmitter Connections

PTT

The push to talk output is an open drain control camaple of sinking 200mA at up to 35 volts.

CTCSS

The CTCSS output can be either an open drain output or an audio output under software contol.

Environmental Monitor

The environmental monitor consists of a DHT22 (AM2302) temperature and relative humidity sensor and a light depending resistor (CDS cell). These are both mounted to the board and are monitored periodically by the software.

User IO

The user IO convicts of TBD logic IO pins and TBD analog input pins. The digital pins can be configured for input or output. The analog pins are by default scaled for 0-25 volts

Host Port

The host port is the mapping from physical bus pins on the host board to the functions used on the controller.

PI pin assignments

pin #	Pi Name	Pi Alt	function	Note:
1	3.3v			
2	5v			
3	GPIO2	SDA1		
4	5v			
5	GPIO3	SCL1		
6	gnd			
7	GPIO4	GPIO_GCLK		
8	GPIO14	TDX0	dctcss1	
9	gnd			
10	GPIO15	RXD0	dctcss2	

PI pin assignments

pin #	Pi Name	Pi Alt	function	Note:
11	GPIO17	GPIO_GEN0	cor1	
12	GPIO18	GPIO_GEN1	pwm0	Was Ptt earlier
13	GPIO27	GPIO_GEN2	hwctcss1	
14	gnd			
15	GPIO22	GPIO_GEN3	ptt1	
16	GPIO23	GPIO_GEN4	COR2	
17	3.3v			
18	GPIO23	GPIO_GEN5	CTCSS2	
19	GP10	SPI_MOSI		
20	gnd			
21	GPIO9	SPI_MISO		
22	GPIO25	GPIO_GEN6	ptt2	
23	GPIO11	SPI_SCLK		
24	GPIO8	SPI_CE0_N	SPIG2A	
25	gnd			
26	GPIO7	SPI_CE1_N	SPIG1	
27	ID_SD			
28	ID_SC			
29	GPIO5		SPIA	
30	gnd			
31	GPIO6		SPIB	
32	GPIO12		SPIC	
33	GPIO13		LINK1O	
34	gnd			
35	GPIO19		pwm1	
36	GPIO16		LINK1C	
37	GPIO37		liknc2o	

PI pin assignments

pin #	Pi Name	Pi Alt	function	Note:
38	GPIO20		link2c	
39	gnd			
40	GPIO21			

I2C bus

There is a 3.3v and 5v shared I2C bus available for expansion such as extra GPIO or specific control functions. The 5V I2C is used on the board for slave addresses: so be careful when adding I2C devices.

Audio Path Theory of Operation

The incoming audio from the receiver is gained up to be 2V peak to peak coming out of the input buffer. When the main audio is used to drive the soft CTCSS and DTMF decoder, there is a fixed attenuation before this level goes out to the computer, This path can be enabled for initial tune up. Software can be used to monitor this point and help adjust the base receiver level.

From the first level receiver input amplifier the signal level is adjustable to any of the 4 link / mixing busses:

- Link 0, local transmitter only.
- Link 1, either transmitter on this node.
- Link 2, an external link
- Link 3, an external link

The link connection and level are controllable via the software controllable potentiometers. Remember that these soft pots also contain switches and the connection to any of these points is disconnected when not in use.

From the link / mixing busses the signal can be picked up by the programmable gain amplifier which has the ability to select any of the 4 sources as its input. If needed gain can be applied here. This output is then mixed with the sound card output buffer before being sent to the transmitter.

From the USB sound card a programmable gain buffer is used to buffer the sound card output to a fixed impedance. This is then mixed with the receive audio from the programmable gain amplifier above.

The transmit audio is a programmable gain buffer which gets the sum of the selected receive audio and

the sound card audio.

When discriminator output is used to drive the CTCSS decoder the buffer from eh CTCSS input is buffered to 2V peak to peak and then attenuated before going out to the USB sound card to be used for the soft CTCSS and DTMF decoding.

All the audio circuits are reference to Vref at 2.5 V and supplied from the 5V supply. The 5V analog supply is just a low pass filter version of the 5V supply bucked from 12V supply.

Quick Start Guide

Since we are currently in "developer mode" it is highly recommended that you have a HDMI monitor and USB keyboard and mouse on your Raspberry Pi and to have your Raspberry Pi on the internet, hardwired or wireless with a high speed data connection for all the software you will install.

Install Rasbian.

Jessie is what is being used for the initial bring up. Follow the directions one the raspberry pi web site to install the base operating system.

Post configure your raspberry pi

Run raspi-config:

- Set your host name to be unique; Suggestion callsign_frequency: ie n1ddk_224_080
- Enable spi (under advanced options)
- Strongly suggested to set your internationalization options, timezone, keyboard, etc.

Update your Raspberry Pi

sudo aptget-update
sudo aptget-upgrade

Install supplemental packages

sudo apt-get install -y python-dev
sudo apt-get install -y espeak

Bring in the git repository for the repeater control software

git clone …

give it a test spin

sudo python/main.py

Software Description

Configuration XML

The config.xml file contains a template and sample configuration for the controller. Although intended to be machine readable it is generally human readable. The Controller contains a GUI with limited capability to dump the file, however a full functioning editor for this file could be developed.

Once the host name four your RaspberryPi node is set, a node specific version of this file will be loaded (and saved) using the naming rules defined below.

File Loading and Overriding

The Python and config files are by default in the python directory. Once your host name is set to something other then the default of 'raspberrypi' if a directory matching your host name exists files in that directory will have precedence over files in the python directory. Additionally the file <hostname>/<hostname>.xml (where <hostname> is replaced with your host name) will be the highest precedence configuration file and will be the file name used when saved from the GUI.

If you want to override any of the python files with your user code, you can copy the file from the python directory and edit it and place it in the <hostname> directory. However note that if you get an update to the code, it will not be reflected in your edited files.

Primary C programs

These program reside in the bin directory; C source, headers and makefile are provided in the c directory. They are called and run directly from the main python code; however you can trunthem and test them form the command line if you wish to experiment with them.

multimon

Multimon was written in 1996 by Thomas Sailer (sailer@ife.ee.ethz.ch, hb9jnx@hb9w.che.eu) and released with GNU general public license. It has been significantly enhanced by James Lee N1DDK in 2016 and 2017 to include ALSA input and CTCSS decoding. Additionally the DTMF decoding has been enhanced to decode larger windows for less glitching and a 'mute' output was added to the DTMF decoding.

Multimon includes the capability to decode many other formats such as bell202 for APRS, those capabilities exist within the program but are not currently used by the repeater controller.

mout

mout is the Morse output routine based on ascii to morse code written by James Lee (KA1KUB in 1983) and ported to C in 2016 and added ALSA output in 2017 by James Lee N1DDK. This code take a command line of exactly 5 arguments:
mout <device> <wpm> <freq> <amplitude> <message>
i.e.:
mout sysdefault:CARD=Device 22 660 1000 "this is a test "

tout

Tout is the general purpose tone generator. It take inputs in sets of 3 and generates one or more tones in a sequence.
tout <device> <freq> <amplitude> <duration> [<freq> <amplitude> <duration>...]
i.e.:
tout sysdefault:CARD=Device 660 1000 100
tout sysdefault:CARD=Device 660 1000 100 100 0 10 440 900 48

System Executables

aplay

Aplay is the ALSA play file utility used to play wave files. It is called (from the system) by the controller code when a wave file is needed to be played.

espeak

Espeak which was one of the additional packages you needed to load is a text to speech routine which can be used in messaging and is used int he talking clock announcements.

Python Files

main.py

As the name describes this the the main entry and launch point for the code. It brings in the other files and instantiates the 2 repeater ports, 2 command queues the GUI and (TBD other network control ports)
Once all the objects are instantiated, it called “load XML” to read in the configuration file. Next it starts threads for the 2 ports and the 2 command processors and the GUI.

radioPort.py

This file has the place holders for all the configurable items on each radio port. It defines the RX and TX classes and the radioPort class. The radioPort creates multiple timers and instantiates the RX and TX classes and the FSM for each port.

rptfsm.py

This file is the state machine that controls the behavior of a repeater or link. it is separated from the radioPort file since it is likely to undergo significant enhancement.

cmd.py

This file houses both the command look up table (touch tome codes) for both ports and the code that is executed when a command is entered. It is likely to be greatly enhanced, as well as customized by for each repeater.

Repeater Control Flow

state machines

timers

command flow

queues

Example of Queuable “messages”

Alternname Messages