PiTone_V30

PL Tone Generating Software for the Raspberry Pi

PiTone is a program that runs on a Raspberry Pi and produces a highly accurate CTCSS tone (also known as a PL tone) using a MCP4725 digital to analog converter (DAC) to generate a stepped sinewave approximation. The software is intended for use in a repeater transmitter where the repeater transmitter or controller does not generate its own CTCSS transmit tone. CTCSS transmit tones can be particularly beneficial in sites using the Yaesu DR-1X/2X repeater, inserting a CTCSS transmit tone so that analog-only FM stations remain squelched without having to hear the “digital noise” during Yaesu Fusion’s C4FM digital transmissions.

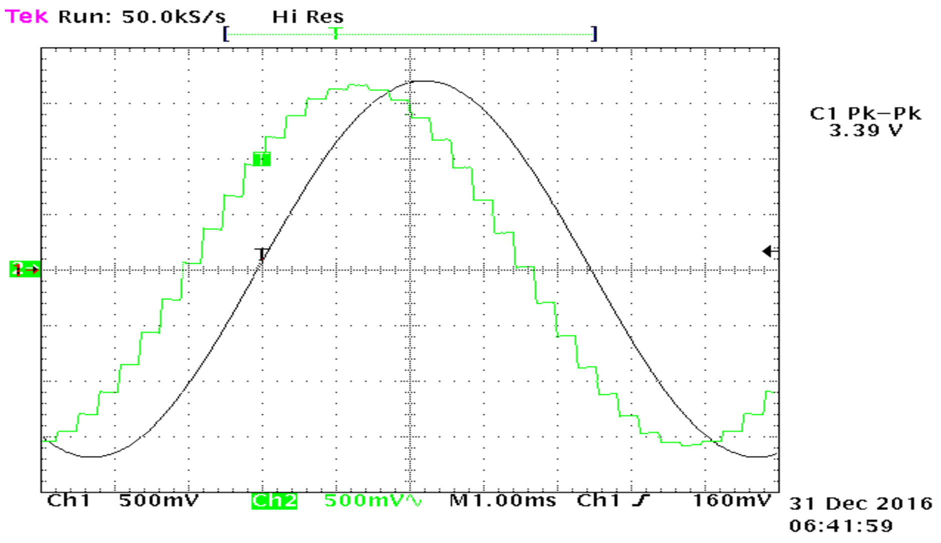


Figure 1: PiTone Waveforms using 32 steps (unfiltered and 3 pole low pass filtered)

PiTone is now in its third revision, currently at revision level 3.0.   PiTone_v30 can be configured to:

- Generate a continuous CTCSS tone
- Generate a CTCSS tone which follows the received CTCSS tone
- Generate a CTCSS tone which follows the received CTCSS tone with a settable on-time delay at the end of the received CTCSS tone.

PiTone_v30 is configured by reading a *pitone.ini* file at runtime. The following parameters can be configured.

- Frequency
- Number of steps per cycle
- Amplitude
- Type of output
    - Continuous
    - Follows received CTCSS signal
    - Follows received CTCSS signal with settable turn-off delay

Copyright

This software is copyrighted freeware. You can use, modify, and distribute the software provided that you offer it and any derivative works as freeware. Any commercial use must be approved in writing by the author.

Disclaimer

This software controls equipment that could be damaged by said software. You are responsible for installing, configuring, testing and ensuring that the software performs properly in your environment. The author cannot be held liable for any direct, indirect, consequential or incidental damages to other pieces of software, equipment, goods or persons arising from the use of this software.

By downloading this software you accept the above terms of copyright and disclaimer.

Release Notes

RELEASE	DATE	CHANGES
Version 1.0	02/01/2017	Initial release
Version 2.0	11/18/2019	Rewritten for Version 2 of PiTone
Version 3.0	12/16/2019	Rewritten for Version 3 of Pitone

Table of Contents

Overview

This document describes using PiTone_v30 with the Raspian operating system. Of course, PiTone_v30 can be used with other Raspberry Pi operating systems. For instance, Pitone_v30 can be used as part of the repeater controller system known as MABEL for the Yaesu DR-1X/2X repeater which commonly uses the Allstar HamVOIP distribution running under ArchLinux. The MABEL DR-1X Interface board hardware also includes a low pass filter and a connector for a DAC module. More information about this system is available at http://www.customcomms.net/products.

Specifications

- Output frequency from 66 Hz to 254 Hz CTCSS frequency
- Output voltage configurable from 10% to 100% of VCC (typically .5 to 5 volts peak-to-peak)
- Frequency stability within .05%
- Output fidelity configurable by creating waveforms from 8 – 92 steps per cycle

Key Features

- Can be configured to be “always on” or “gated” by an external GPIO input
- “Headless” operation. Can be remotely controlled over a TCP/IP connection
- No DIP switches, diodes, solder-bridges. All configuration is via a *pitone.ini* file
- Inexpensive, readily available components
- Highly accurate frequency generator
- Flexible design allows for installation-specific customization
- Open-source software written in standard C++(14)

Required Materials

- Raspberry Pi ($5 Pi Zero, $19 - $50 for Pi 2,3 or 4 from various sources)
    - Model 3b or newer recommended for shared installations. For stand-alone installations, the inexpensive Pi Zero or an unused Pi2 or B+ is sufficient
        - If using a Pi Zero, you may want to purchase a 40 pin header (available from Mouser, Digikey, Newark and many other sources) to facilitate connections between the Pi Zero and the MCP4725 DAC
    - 8GB (or larger) microSD card with Raspbian operating system. Fast card (class 10) recommended
    - Raspberry Pi micro-USB power cable with 5 vdc power supply
- Digital to Analog Converter (DAC) ($1.13 - $4.95 from various sources)
    - MCP4725 DAC breakout board
    - 5 conductor ribbon interconnect cable
    - Low value (~ 1 mfd) DC blocking capacitor
- Optional – Wave shaping circuit to smooth the stepped waveform
    - RC Circuit. See text
- Setup and Configuration Equipment – Two Choices

- **Easy** Direct connections – recommended for beginners. The easiest way to get started and configure your system
    - USB keyboard and mouse for configuring the Raspberry Pi
    - HDMI cable and video monitor for viewing the Raspberry Pi console during configuration
- *Optional*: **Less Easy** Remote connections – needed for headless / remote configuration. Freeware Windows programs
    - Terminal (command line interface - CLI) access: PuTTY
    - X window GUI pair: VNCserver (on RPi) and XRDP on RPI with RDP on Windows
    - File management: FTP access - FileZilla, or explorer access – WinSCP
    - MobaXterm

Step 1. Preparing the Raspberry Pi

Installing the Raspbian Operating System

There are many excellent guides to assist in preparing the Raspbian operating system on your Raspberry Pi. Some recommendations:

Books:
Raspberry Pi in Easy Steps by Mike McGrath
Raspberry Pi User Guide by Gareth Halfacree
Websites:
Setting Up Your Raspberry Pi http://www.raspberrypi.org/help/quick-start-guide/
Setting Up a Raspberry Pi with NOOBS https://learn.adafruit.com/setting-up-a-raspberry-pi-with-noobs

Use the guides to install the most recent Raspbian operating system on your Raspberry Pi. After configuring the Raspbian operating system on your RPi, use the Command Line Interface (CLI) console to enter the commands: (press <enter> after each command)

```
sudo apt-get update
sudo apt-get upgrade - y
```

Those commands ensure that you have the latest version of Raspbian and its installed applications.

Enabling the Raspberry Pi's SSH and I2C Functions

By default neither the Raspbian SSH nor I2C interface is enabled. The easiest way to configure these interfaces is from the GUI. With your mouse, select >Start (the Raspberry Pi Icon), >Preferences >Raspberry Pi Configuration. In the Configuration dialog window, select the >Interfaces tab, and enable the SSH and I2C options. *While you're here, you should also set the language, timezone, keyboard, and WiFi country settings on your RPi.*

Step 2. Software - Configuring the PiTone Software

Downloading and Installing the program

The latest version of PiTone can be downloaded from www.hamprojects.info/pitone as *pitone_30_release.zip*. The ZIP file includes several files: this document - PiToneManual.pdf , PiTone_v30 executable, PiTone.ini file and the GPL licensed PiTone_v30 source code.

- **Easy Method:** Direct Connect. Unzip the zip archive to a USB flash drive. Plug the drive into a USB port on the Raspberry Pi, which will automatically detect its presence and launch the RPi File Manager. Copy/paste the files from the USB flash drive to the Raspberry Pi's **`/home/pi`** directory (the home directory for the user named "pi"). You will need to set the `pitone` binary file's attribute to *Execute*, which is accomplished by right-clicking on the `pitone_v30` binary executable in the **`/home/pi`** directory, selecting >Properties >Permissions, and setting Execute = Owner.

- **Less Easy Method:** Remote Connect. Unzip the zip archive to a temporary directory on your PC. Transfer the files to the Raspberry Pi's **`/home/pi`** directory using either an FTP client (FileZilla), an explorer client (WinSCP) or MobaXterm. You will need to set the PiTone_v30 binary file's attribute to *Execute*, which can be accomplished several ways:
  - Within the FileZilla or WinSCP programs, right-click on the file and set >Properties
  - or, Login to the RPi GUI, start the file manager, and right-click on the file and set >Properties
  - or, Login to the RPi console, and from the CLI set its properties with: **sudo chmod +x pitone_v3**

Configuring the Raspbian I2C Baudrate

By default the Raspbian I2C interface runs at baudrate = 400000, which is too slow for PiTone operation. To modify the I2C baudrate, edit the file `/boot/config.txt`. From the command line, type **sudo nano /boot/config.txt** to open the file in a simple text editor with administrative read/write privileges. Down-arrow in the file, looking for the line reading `dtparam=i2c_arm=on` (somewhere around line 45 – this line was inserted for you when you enabled the I2C interface earlier). Beneath that line, add the following text on a separate line:

```
dtparam=i2c_baudrate=1200000
```

Close the file by typing **`<crtl>X`** to exit, **`Y`** to save the changes, then **`<Enter>`**

This statement sets the baud rate to 1.2 MBPS (ensure that you have 5 zeroes 00000!). Reboot the RPI with `sudo reboot`

Determining the DAC I2C Address

At the command line (CLI), enter **sudo apt-get install –y i2c-tools**. Once the tools are installed, from the command line, enter **i2cdetect –y 1**. Make note of the file descriptor (I2C address), you will need to enter it into the pitone.ini file.

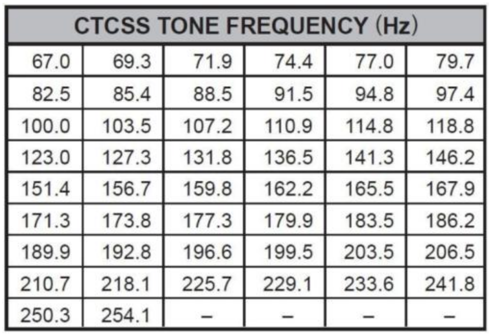
Configuring PiTone

PiTone is controlled by parameters in the pitone.ini file when the program is launched. The pitone.ini file must be located in the same directory as the executable. Here is the contents of the pitone.ini file:

```
;======================================================================
; pitone.ini  - settings for pitone_v30 CTCSS Tone generator
;           2019-12-11, by K8UT
; NOTE: See pitone.cpp source code for Raspberry Pi setup instructions
;======================================================================
; plTone frequency.  66Hz to 254 Hz. for example: 110.9
plTone = 110.9
;
; stepCount to build sine wave.  8 to 92. for example: 32
stepCount = 32
;
; percentVolts. 10 to 100 percent of Vcc. for example: 75
percentVolts = 75
;
; i2cAddress of DAC. 60, 61, or 62. for example: i2cAddress = 62
i2cAddress = 62
;
; useTxCTCSS to trigger sinewave on/off. optional. Y or N. for example: useTxCTCSS = N
; NOTE: Custom setup needed for useTxCTCSS with a Raspberry Pi-4. See documentation
useTxCTCSS = N
;
; delayCycles. optional. Trailing number of cycles after Tx with useTxCTCSS.
; 0 = OFF (no delay) max = 1000. for example: delayCycles = 220
delayCycles = 0
;======================================================================
```

Explanation of PiTone-V30 configuration inputs

plTone – This is the desired CTCSS (PL) tone in hertz. Values from 66 to 254 are accepted. Standard CTCSS frequencies are shown in the table. Note that PiTone_v30 does not require the plTone input to be one of these frequencies. Any decimal value between 66 and 254 is acceptable.



CTCSS TONE FREQUENCY (Hz)					
67.0	69.3	71.9	74.4	77.0	79.7
82.5	85.4	88.5	91.5	94.8	97.4
100.0	103.5	107.2	110.9	114.8	118.8
123.0	127.3	131.8	136.5	141.3	146.2
151.4	156.7	159.8	162.2	165.5	167.9
171.3	173.8	177.3	179.9	183.5	186.2
189.9	192.8	196.6	199.5	203.5	206.5
210.7	218.1	225.7	229.1	233.6	241.8
250.3	254.1	–	–	–	–

Figure 2 - Standard CTCSS Frequencies

stepCount – This is the number of steps that pitone_v30 uses to create one cycle of the tone frequency. Values from 8 to 92 are accepted. There is a tradeoff between CPU useage, the number of steps and the distortion of the sinewave. In the following figures, the green trace shows the ouput voltage for a stepCount of 16, 32 and 64. The black trace shows the output after passing the signal through a 3 pole low pass active filter.

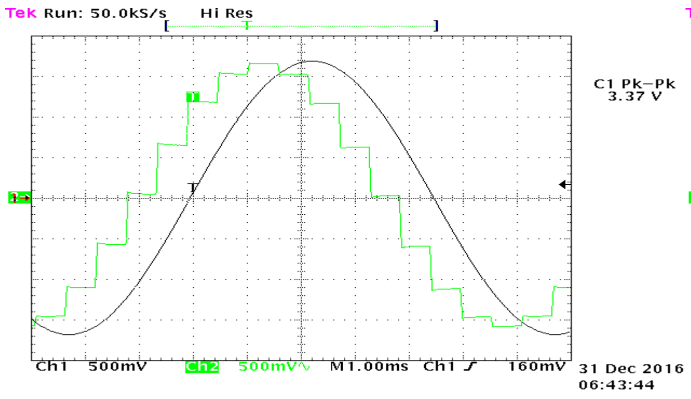


Figure 3 - 16 Step Sine Wave

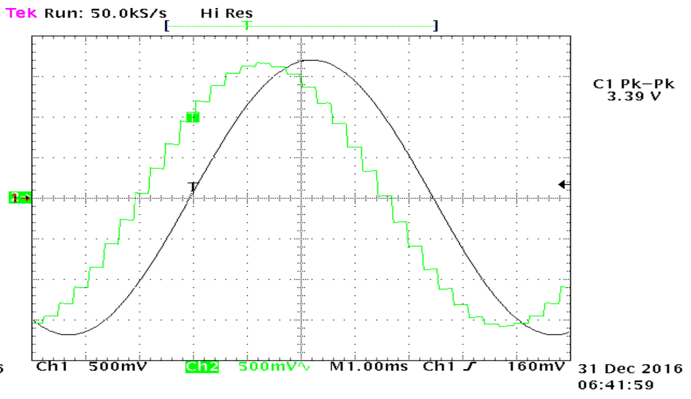


Figure 4 - 32 Step Sine Wave

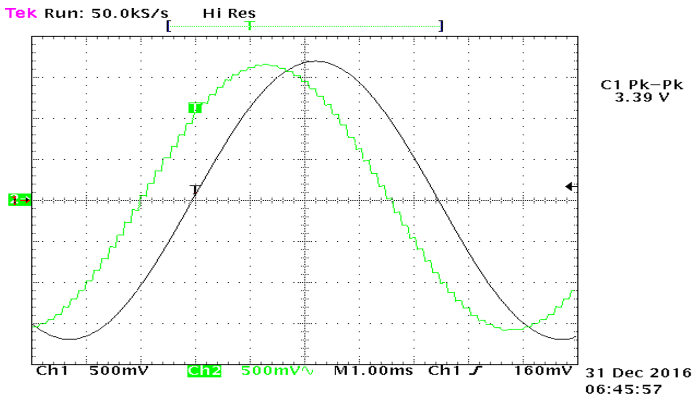


Figure 5 - 64 Step Sine Wave

Clearly, the greater the number of steps, the greater the fidelity (and thus lower distortion) of the sine wave. However, a larger number of steps requires more CPU time. PiTone_v30 using 32 steps requires about 15% of the CPU time of one core (out of 4) on a Raspberry Pi 3 B+. As shown in Figures 3, 4 and 5, the steps (distortion) in the stepped sine wave can be removed very nicely by a low pass filter. The MABEL DR-1X/2X interface board design includes a 3 pole active filter for this purpose. Note that the frequency accuracy and stability of the generated signal is not affected by the number of steps as the time for one cycle remains the same.

The next figure shows a schematic of this filter.

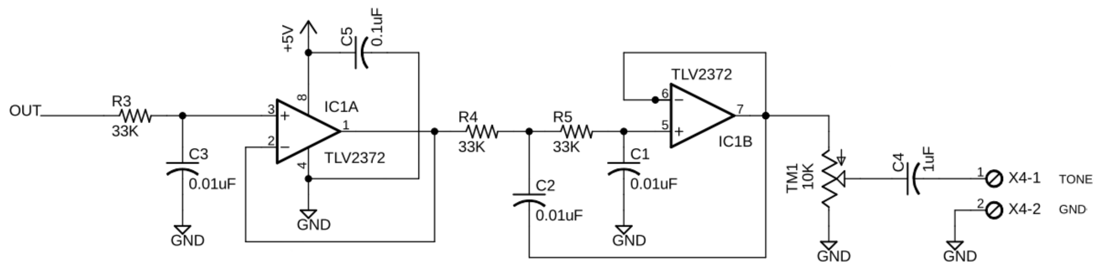


Figure 6 – 3 Pole Low Pass Active Filter

percentVolts – The peaks of the sinewave available out of the DAC vary from a maximum slightly less than Vcc of the DAC to slightly greater than ground. So for a 3.3 volt Vcc, the maximum peak-to-peak value of the waveform will be slightly less than 3.3 volts. The percentVolts parameter allows you to specify a lower output voltage – from 100% down to 10% of the peak-to-peak voltage. The peak-to-peak output is always centered at VCC/2

CAUTION – The ouput of the DAC is DC coupled. You may need an AC coupling capacitor to inject the CTCSS signal into your transmitter.

i2cAddress – This is the I2C address of the MCP4725 DAC. Experience shows that it varies between from 60 to 62 depending on the source of the DAC IC. The I2C address of the your MCP4725 DAC can be found by entering `i2cdetect -y 1` at the command line.

useTxCTCSS – This input specifies if you want the CTCSS tone to always be on (N) or follow an external input signal (Y) connected to Pin 22 of the Raspberry Pi 40 pin connector. If you are using the MABEL DR-1X Interface board, you will connect the TxCTCSS signal from a GPIO pin on your USB audio IC in your RA-35 radio interface to the third opto-isolator on the MABEL interface board.

SPECIAL NOTE - If you are wiring a TxCTCSS signal that you create or tap into from your repeater you need to be aware of the following. PiTone_v3 uses the WiringPi library for the GPIO interface. Unfortunately, due to a bug in the library, when using a Raspberry Pi4, WiringPi does not “connect” a pull-up resistor on the TxCTCSS input pin as programmed in the PiTone_v30 software. Thus, for the Raspberry Pi 4 only, if you are using an open collector or open drain circuit to drive pin 22, you must provide a 56K ohm pullup resistor to 3.3 VDC on the Pi.

delayCycles – If you have chosen to use the TxCTCSS signal, you have the option of delaying the turn-off of the PiTone signal output when the TxCTCSS signal turns off. This delay can be from 0 to 1000 cycles of your CTCSS frequency. For example, at 100 Hz, 200 delayCycles would be 2 seconds (200 x 10 millisec). Of course, "0" is no delay. The following figure shows a delay of 200 cycles at a CTCSS frequency of 110.9 Hz ((1/110.9)x200=1.8 seconds). The top trace is TxCTCSS and the bottom trace is the CTCSS output.

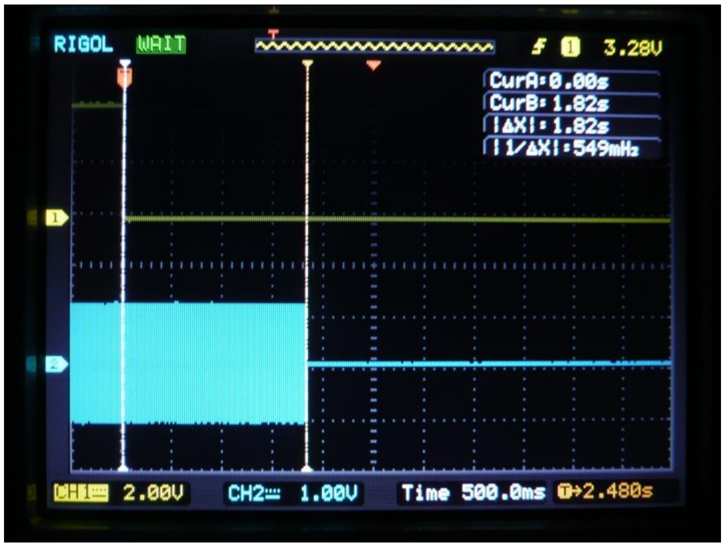
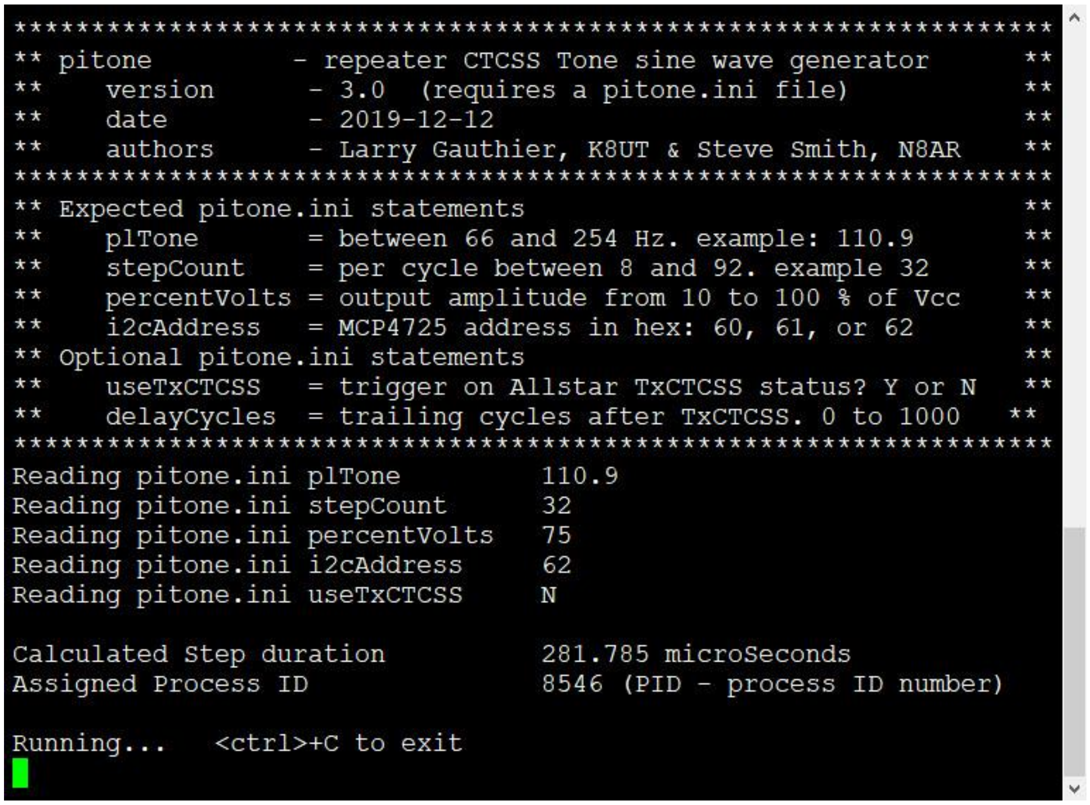


Figure 7 – CTCSS Turn-off Delayed by 1.8 Seconds

Running PiTone

You can start PiTone_v30 from the command line. At the command line, type `cd /home/pi` and press **<enter>** to navigate to the Pi folder. Then type `./pitone_v30.` When PiTone launches, a splash screen displays the program's status on the console.



```
*******************************************************************
** pitone          - repeater CTCSS Tone sine wave generator     **
**    version       - 3.0  (requires a pitone.ini file)          **
**    date          - 2019-12-12                                 **
**    authors       - Larry Gauthier, K8UT & Steve Smith, N8AR   **
*******************************************************************
** Expected pitone.ini statements                                **
**    plTone        = between 66 and 254 Hz. example: 110.9      **
**    stepCount     = per cycle between 8 and 92. example 32     **
**    percentVolts = output amplitude from 10 to 100 % of Vcc    **
**    i2cAddress    = MCP4725 address in hex: 60, 61, or 62      **
** Optional pitone.ini statements                                **
**    useTxCTCSS    = trigger on Allstar TxCTCSS status? Y or N  **
**    delayCycles   = trailing cycles after TxCTCSS. 0 to 1000  **
*******************************************************************
Reading pitone.ini plTone          110.9
Reading pitone.ini stepCount       32
Reading pitone.ini percentVolts    75
Reading pitone.ini i2cAddress      62
Reading pitone.ini useTxCTCSS      N

Calculated Step duration           281.785 microSeconds
Assigned Process ID                8546 (PID - process ID number)

Running...   <ctrl>+C to exit
```

PiTone_v30 will start and run until you press **<ctrl> C** or logout of Raspbian. If you want PiTone_v3 to continue running after you log out just type `./pitone_v30 &` to start it.

Improving Frequency Stability with the CHRT Command Line Option

Occasionally, a cycle or two of PiTone will be distorted when the Raspbian operating system gets interrupted to perform other tasks. A typical occurrence of this can be seen in Figure 8.

Promoting PiTone to a real-time task prevents this occasional distortion and delivers rock-solid output frequency.

Rather than entering `./pitone_v30`, enter the following command: **`chrt -r 99 ./pitone_v30`** This will invoke the Linux CHRT (Change to Real Time) scheduling policy and promote PiTone to the highest level, 99, ahead of other processes.

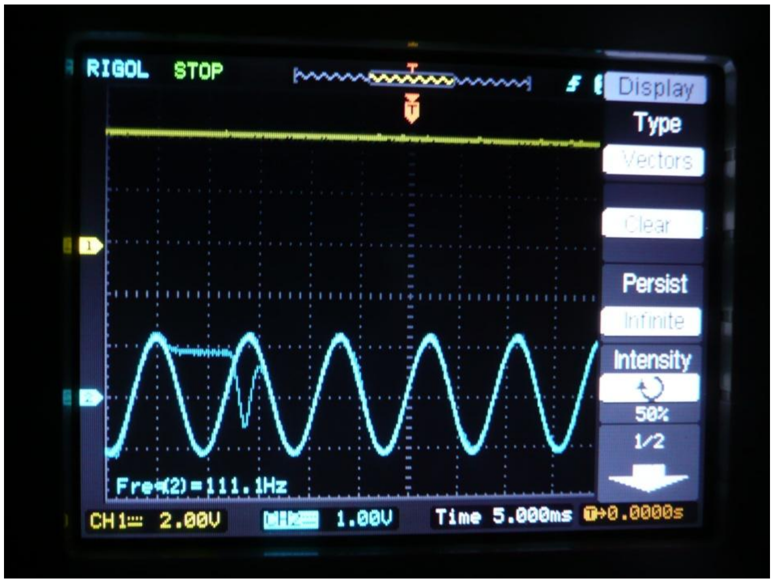


Figure 8 – One Distorted Cycle of the CTCSS Sine Wave

Figure 9 shows a histogram of a 110.9 Hz PiTone sine wave running with Real Time priority.

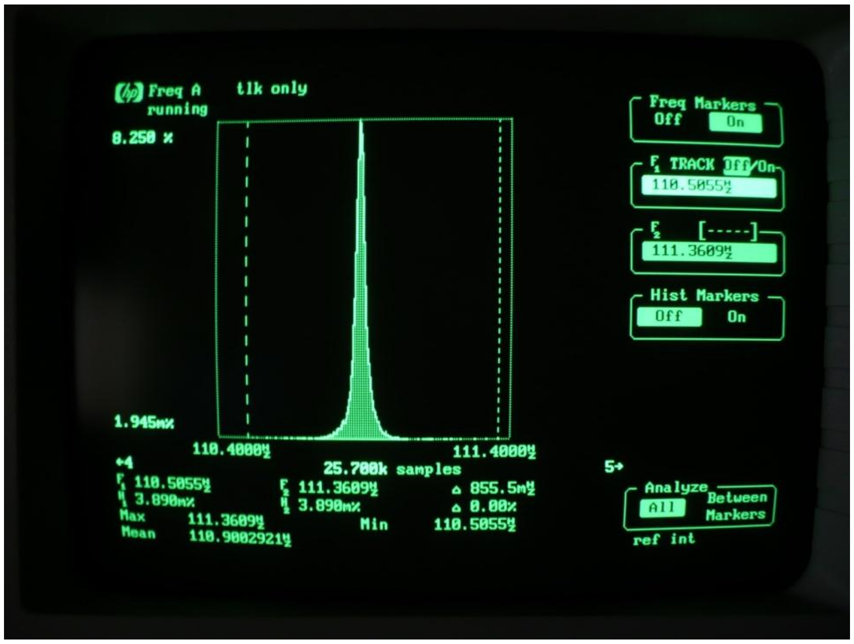


Figure 9 – Histogram of a CTCSS Tone Running in Real Time

Adding Boot Auto-Start to PiTone

Perhaps you remember `using AUTOEXEC.BAT` files to automatically start MS-DOS programs on an IBM PC? The Raspbian operating system has a similar feature. You can configure your Raspberry Pi to automatically launch PiTone when the RPi reboots. Raspberry Pi's boot commands are located in a file called **`rc.local`** in the **`/etc`** directory. Editing this file requires root user (sudo) privileges. From the Raspberry Pi CLI, type the command **`sudo nano /etc/rc.local`** . When ***nano*** (a text editor) opens, use the down-arrow key to move near the bottom of the file and insert the following text on a line immediately above the **`exit 0`** statement:

```
cd /home/pi
sleep 30
sudo chrt  -r 99 ./pitone_v30 &
```

After adding those lines, press **`<crtl>X`** to exit, **`Y`** to save the changes, then **`<Enter>`**. Thereafter, PiTone will automatically start whenever the Raspberry Pi boots.

3. Connecting the DAC to the Raspberry Pi

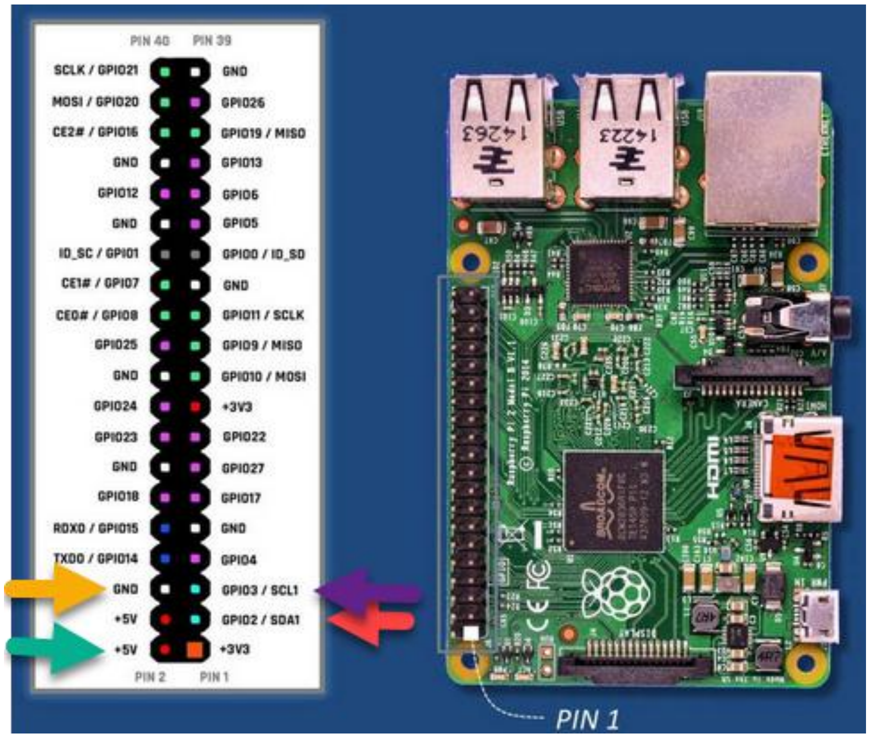
Solder the 5 pin in-line connector to 5 pins of the MCP4725 breakout board labeled **`OUT, GND, SCL, SCA, and VCC.`** (Depending on the model MCP4725 board you purchased, this may leave a second GND position unconnected. See photo). Connect four conductors of the ribbon cable between the following pins of the MCP4725 and the GPIO pins of the Raspberry Pi.

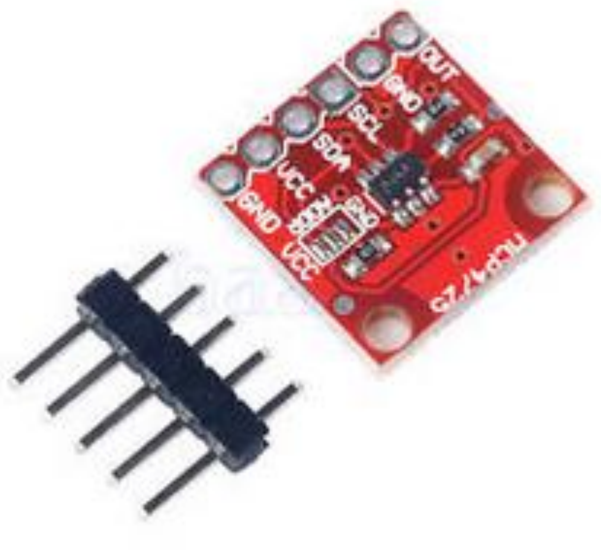
Raspberry Pi connections

•	MCP4725	VCC	to	GPIO pin 2 (green arrow)
•	MCP4725	SDA	to	GPIO pin 3 (red arrow)
•	MCP4725	SCL	to	GPIO pin 5 (purple arrow)
•	MCP4725	GND	to	GPIO pin 6 (orange arrow)

Repeater input connections

•	MCP4725	OUT	to	DC blocking capacitor, optional RC filter, and repeater Tone In
•	MCP4725	2nd GND	to	repeater chassis ground


PIN 40
PIN 39
SCLK / GPIO21
GND
MOSI / GPIO20
GPIO26
CE2# / GPIO16
GPIO19 / MISO
GND
GPIO13
GPIO12
GPIO6
GND
GPIO5
ID_SC / GPIO1
GPIO0 / ID_SD
CE1# / GPIO7
GND
CE0# / GPIO8
GPIO11 / SCLK
GPIO25
GPIO9 / MISO
GND
GPIO10 / MOSI
GPIO24
+3V3
GPIO23
GPIO22
GND
GPIO27
GPIO18
GPIO17
RXD0 / GPIO15
GND
TXD0 / GPIO14
GPIO4
GND
GPIO3 / SCL1
+5V
GPIO2 / SDA1
+5V
+3V3
PIN 2
PIN 1
PIN 1


OUT
GND
SCL
SDA
VCC
GND
ADDR
GND
VCC
MCP4725