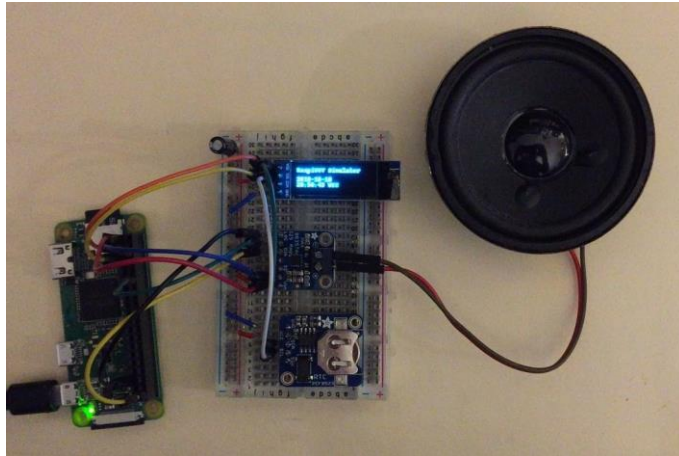


RaspiWWV

2018-10-11 R.Grockett



Remember the days when you would sit listening to WWV time signals on your Shortwave radio (*tick, tick, tick... At the tone, the time will be...*)?

Oh! You missed out on that? Now you can (re-)experience those moments and have your own WWV clock, no Shortwave Radio and no Internet connection required. Isn't that what you always wanted?

Actually, this is a project to help show you how to connect a tiny OLED screen, a Real Time Clock (RTC) and an Audio Amplifier all to a single Raspberry Pi Zero! All with the bonus of being able to listen to WWV time "signals" anytime you wish.

Unlike the real WWV, the audio time signals as well as the admittedly tiny clock display will be only accurate to about one second of drift per day using the typical low-cost RTC module. You can improve this by using a higher quality (more expensive) RTC, or by just leaving the Raspberry Pi connected to the Internet, but this would never replace the atomic clocks of the real WWV. (*See precision time keeping with Raspberry Pi articles, such as <https://www.satsignal.eu/ntp/Raspberry-Pi-NTP.html>)*

Otherwise, for most uses, the accuracy is probably sufficient. Most uses? Well, besides falling to sleep to the droning sound of WWV, I used the shortwave radio version to timestamp astronomical observations; using a tape recorder (remember those?), I would record meteor observations, lunar occultations or other events I saw, with WWV time signals in the background. "Mark!" The recording could then be transcribed with the sightings and timestamps in the comfort of home.

For those with such nostalgia, be sure to dig up an old transistor radio (and remember those?) to use as a case for your project. Note that the case will need to be large enough to install a battery, if you want portability!

The “WWV” software, written in Python, displays the time and plays the appropriate audio. The Raspberry Pi clock and RTC will resync to Network Time Protocol (NTP) servers whenever it connects to the Internet. (Anytime it is within range of your WiFi, if you are using a Raspberry Zero W.)

Hardware Parts

- Raspberry Pi Zero W
- MakerFocus 0.91 Inch I2C SSD1306 OLED Display Module
Amazon <http://a.co/d/ioakKen>
(or other SSD1306 128x32 OLED display)
- Real Time Clock (RTC) module PCF8523
<https://www.adafruit.com/?q=PCF8523>
- I2S 3W Class D Amplifier Breakout - MAX98357A
<https://www.adafruit.com/product/3006>
- Small Speaker (I salvaged a 2” speaker from old PC speakers)
- Breadboard, Pi Breakout cable/connector to breadboard, wire
- 5v 1amp battery (used to charge a cellphone)
Such as Anker PowerCore 5000 Portable Charger <http://a.co/d/1utjWPg>
Alternately, you can use a LiPo battery + 5v converter/charger
<https://www.adafruit.com/product/2011>
<https://www.adafruit.com/product/2465>
- CR1220 battery for RTC
- Case – old transistor radio

OS Setup

Install **Raspbian Stretch Lite** onto a 4GB or larger SD card. (steps below)

Did you know?

If you install Raspbian on an SD card using a PC, you can create two files on the card to configure WiFi and SSH access before you boot it on a Raspberry?

For this, assume your SD card is currently mounted as K: on your PC:

- 1) Install the latest Raspbian Lite image to the SD.
<https://www.raspberrypi.org/downloads/raspbian/>

- 2) With notepad, create a file called just “ssh” and use Save As “All files” to K:\ssh
The file can contain anything. It’s the **filename** that is important. Must **NOT** be “ssh.txt”!!!
- 3) With notepad, create a second file called “wpa_supplicant.conf” with following:

```
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1

network={
    ssid="mySSID"
    psk="mypassword"
    key_mgmt=WPA-PSK
}
```

Use Save As “All files” to K:\wpa_supplicant.conf

Again, do not let Notepad change it to “wpa_supplicant.conf.txt”!!

When you boot the Raspberry the first time, Raspbian will look for these and connect to your Wifi. You will have to look on your Router for the IP address, though, since its auto assigned.

1. If you haven’t already, install Raspbian Lite version onto a 4GB or larger microSD card. You DO NOT need the GUI version, as this project does not use a monitor or keyboard.
<https://www.raspberrypi.org/downloads/raspbian/>
2. You will need to access the Raspberry remotely via SSH. On Windows, you can use PUTTY SSH terminal program. On Mac, just bring up a command terminal window.
<https://www.putty.org/>
3. Insert the microSD card into the Pi and plug in the power now. It will take a few minutes to boot.
4. To remotely log in to your Raspberry Pi, you will need to find its IP address.
You can try:

\$ **ssh** [pi@raspberrypi.local](https://www.raspberrypi.org/)

(Or from Putty, enter hostname **pi@raspberrypi.local**

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address) Port

Connection type:

☐ Raw ☐ Telnet ☐ Rlogin ☒ SSH ☐ Serial

Load, save or delete a stored session

Saved Sessions

Otherwise, you will need to see if your Router will show the IP addresses of your local devices.

Default id/passwd is **“pi/raspberry”**

Once booted and logged in, start by updating your Pi’s OS, change the “pi” password, and turn on I2C interface:

```
sudo apt-get update
sudo apt-get upgrade
sudo raspi-config
```

Select:

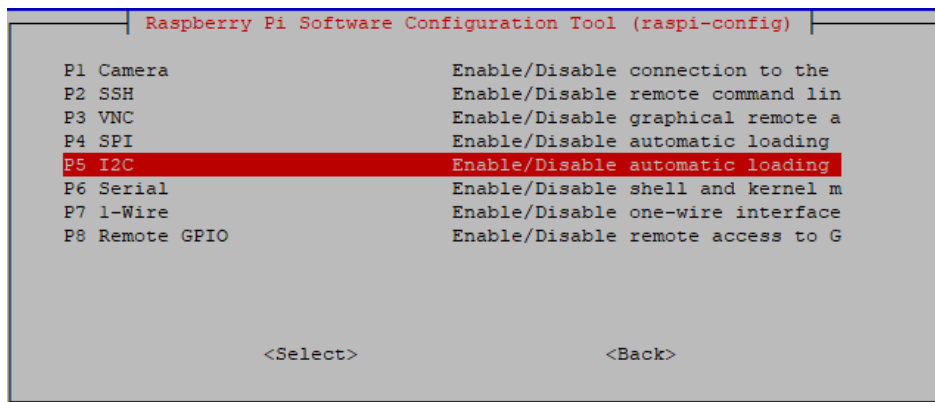
Change user password

Interfacing Options -> I2C Enable -> Yes

Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password	Change password for the current u
2 Network Options	Configure network settings
3 Boot Options	Configure options for start-up
4 Localisation Options	Set up language and regional sett
5 Interfacing Options	Configure connections to peripher
6 Overclock	Configure overclocking for your P
7 Advanced Options	Configure advanced settings
8 Update	Update this tool to the latest ve
9 About raspi-config	Information about this configurat

<Select> <Finish>



Be sure to leave the **Locale** and **Timezone** (UTC) unchanged.

WWV runs in UTC time.

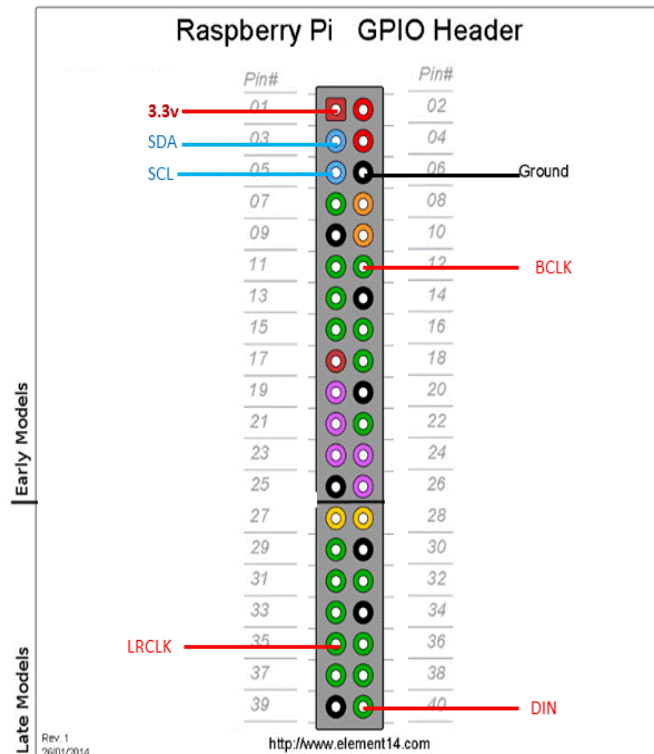
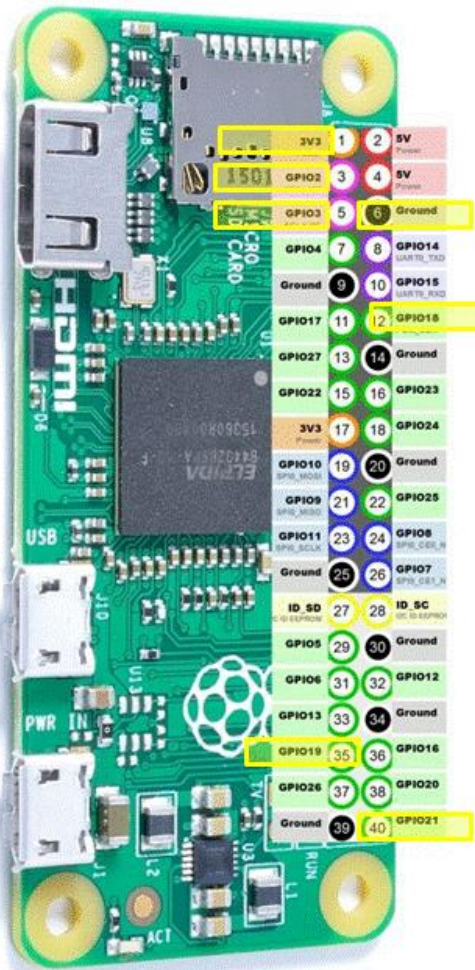
You can wait to reboot after the next software install:

RaspiWWV Software Install

```
cd /home/pi
sudo apt install git
git clone https://github.com/rgrokkett/RaspiWWV.git
sudo shutdown now
```

Power down and unplug the Pi to do wiring next, before trying the software.

TODO – Wiring diagram for all modules



“On/Off” Button Setup

To add a safe shutdown button to your Pi, add a momentary pushbutton between pin 11 (GPIO 17) and Ground.

Pi	Desc
Pin 11 (GPIO 17)	Safely Shutdown/Start up Raspberry (Does not remove 5V power, must be done usually via the 5V USB battery on/off switch)
Pin 9 (GND)	Ground side of momentary button

OLED Display Setup

(from <https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/overview>)

Wire in SSD1306 OLED display



OLED	Pi
SCL	SCL pin 5 (GPIO 3)
SDA	SDA pin 3 (GPIO 2)
VCC	3.3v pin 1
GND	GND pin 6

```
sudo apt-get install -y python-smbus
sudo apt-get install -y i2c-tools
```

Power up the Raspberry and log back in and run:

```
sudo i2cdetect -y 1
```

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  3c  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi:~ $
```

You should see a “3C” address show up.

If not, check your wiring and be sure i2c is turned on in raspi-config.

Next, run the OLED install script to install the Adafruit Python SSD1306 library:

```
cd /home/pi/RaspiWWV
sh installOLED.sh
```

Once Finished, run the OLED test program:

```
python testOLED.py
```

Run the following commands:

```
sudo apt-get install python-dev python-pip
```

```

sudo apt-get install python-imaging
sudo apt-get install git
git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git
cd Adafruit_Python_SSD1306
sudo python setup.py install
cd examples
sudo python stats.py

```

If you get a display, then continue to installing the Real Time Clock (RTC)

If not, check your wiring and the Adafruit tutorial (url above) for more info.

Real Time Clock (RTC) Setup

(from : <https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/wiring-the-rtc>)

```
sudo shutdown now
```

Again, power off the Raspberry after it shuts down.

Wire in the RTC in parallel with the OLED, so now both the OLED and the RTC modules are connected to the same GPIO pins on the Pi. (The I2C bus allows for multiple devices, as long as they have different addresses)

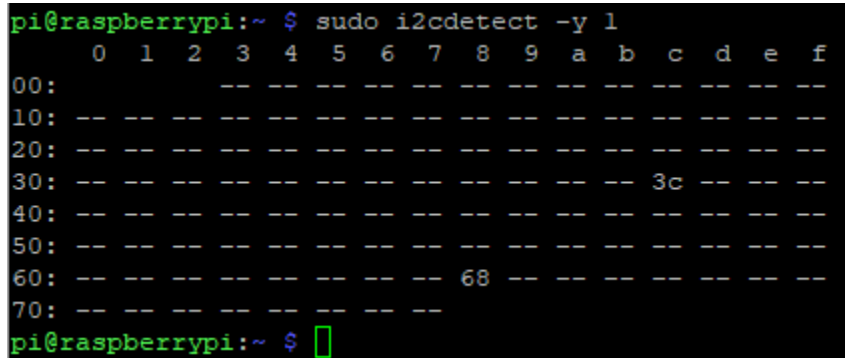


RTC	OLED	Pi
SCL	SCL	SCL pin 5 (GPIO 3)
SDA	SDA	SDA pin 3 (GPIO 2)
VCC	VCC	3.3v pin 1
GND	GND	GND pin 6

Put a CR1220 battery into the RTC

Plug in the power to the Pi and log back in again and run:

```
sudo i2cdetect -y 1
```



```
pi@raspberrypi:~$ sudo i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~$
```

You should see a 68 address as well as the 3C address.

Run the RTC install script:

```
cd /home/pi/RaspiWWV
sh installRTC.sh
```

Answer the question concerning which RTC chip type you have. The default is the pcf8523 chip listed above.

~~Run one of the following to append to the end of the boot config file
whichever matches your RTC chip:~~

```
sudo bash -c 'echo "dtoverlay=i2c-rtc,ds1307" >> /boot/config.txt'  
or  
sudo bash -c 'echo "dtoverlay=i2c-rtc,pcf8523" >> /boot/config.txt'  
or  
sudo bash -c 'echo "dtoverlay=i2c-rtc,ds3231" >> /boot/config.txt'
```

```
sudo reboot
```

```
ssh pi@raspberrypi
```

```
sudo i2cdetect -y 1
```

You should see UU instead of 0x68

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- UU -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

Verify the Pi shows the correct time:

```
date
```

```
Tue  9 Oct 20:51:40 UTC 2018
```

Note that the time MUST be in UTC time.

If not correct, be sure the Pi can access the Internet so it can set its time.

Now run the RTC install script again:

```
cd /home/pi/RaspiWWV
sh installRTC.sh
```

Answer with the same RTC chip you have.

This time, the script will disable the old fake hardware clock and replace with the RTC.
It will also set the time of the RTC clock to match the system clock.

~~Disable the “fake hwclock” and activate the “hardware clock”~~

```
sudo apt-get -y remove fake-hwclock
sudo update-rc.d -f fake-hwclock remove
sudo systemctl disable fake-hwclock
sudo mv /run/systemd/system /run/systemd/system.BAK
```

You can also manually set the RTC clock to the Pi’s current time:

```
sudo hwclock -w
```

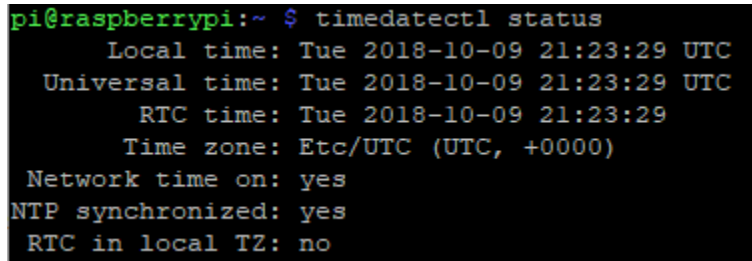
To read the RTC clock’s time:

```
sudo hwclock -r
```

You should only have to set it once, unless there is a problem with the battery.

Check the time for the System clock vs the hardware RTC clock:

```
timedatectl status
```



```
pi@raspberrypi:~ $ timedatectl status
    Local time: Tue 2018-10-09 21:23:29 UTC
    Universal time: Tue 2018-10-09 21:23:29 UTC
        RTC time: Tue 2018-10-09 21:23:29
        Time zone: Etc/UTC (UTC, +0000)
    Network time on: yes
    NTP synchronized: yes
    RTC in local TZ: no
```

If wifi is available, the Raspi will update the system clock and the RTC clock to the Internet Network Time Protocol (NTP) servers so as to correct any drift.

For this to occur, the Pi must be rebooted while connected to a WiFi network.

TODO cron script?

```
sudo hwclock -w --update-drift
```

```
Ping -c 1 -W 5 8.8.8.8 | grep '0% packet loss'
```

```
if true, then sleep 5 min and test again, if good then set hwclock above
```

Audio Setup

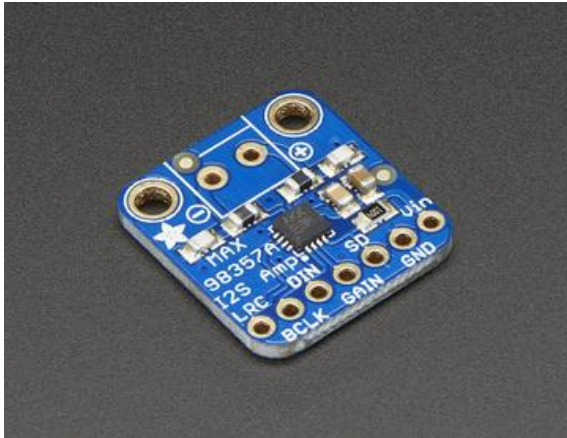
(from <https://learn.adafruit.com/adafruit-max98357-i2s-class-d-mono-amp/raspberry-pi-usage>)

NOTE: THIS ASSUMES YOU ARE USING THE ADAFRUIT MAX98357 I2S AMPLIFIER

[HTTPS://WWW.ADAFRUIT.COM/PRODUCT/3006](https://www.adafruit.com/product/3006)

Shutdown and unplug power from the Pi and install the audio card:

```
sudo shutdown now
```



Wire as follows:

- **Amp Vin** to Raspi Zero Pi **Pin 1 +3.3V**
- **Amp GND** to Raspi Zero Pi **GND**
- **Amp DIN** to Raspi Zero Pi **Pin 40 (GPIO 21)**
- **Amp BCLK** to Raspi Zero Pi **Pin 12 (GPIO 18)**
- **Amp LRCLK** to Raspi Zero Pi **Pin 35 (GPIO 19)**
-
- **Speaker** to Amp Audio connector

For wiring simplicity sake, I have used the +3.3v lead to power all the modules.

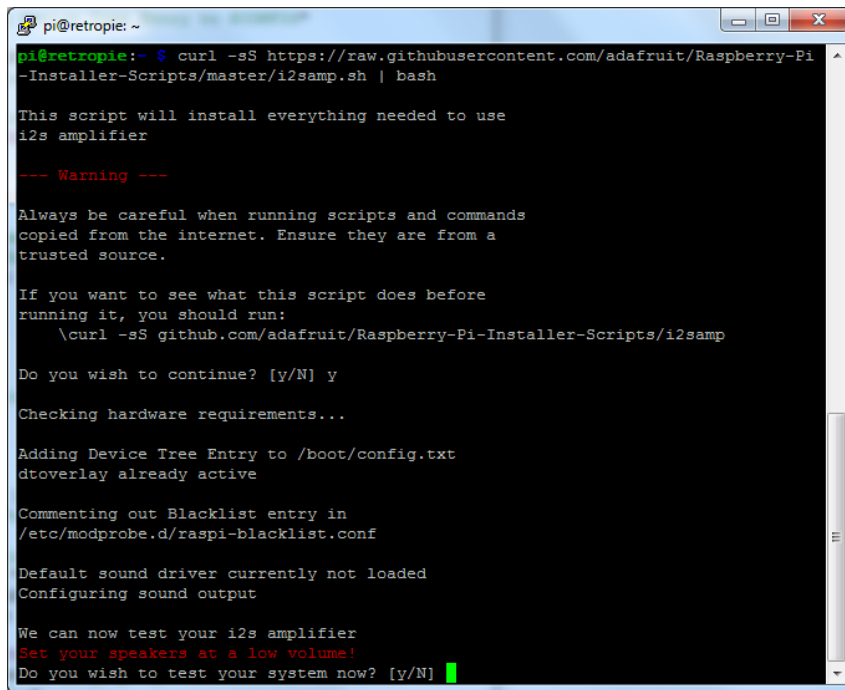
The audio amp can also be run from Pin 2, +5V instead, if you need higher volume.

Next, run the Audio amp install script:

```
cd /home/pi/RaspiWWV  
sh installAMP.sh
```

Next, run the Adafruit/Pimoroni install script:

```
curl -sS https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2samp.sh | bash
```

A screenshot of a terminal window titled 'pi@retropie: ~'. The terminal shows the execution of the command `curl -sS https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2samp.sh | bash`. The script's output includes a warning about running scripts from the internet, a prompt to continue (answered 'y'), and a series of configuration steps: checking hardware requirements, adding a Device Tree entry to `/boot/config.txt` (noting dtoverlay is active), commenting out a blacklist entry in `/etc/modprobe.d/raspi-blacklist.conf`, and configuring sound output. It concludes by asking if the user wants to test the system now, with a green cursor at the end of the line.

```
pi@retropie:~$ curl -sS https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2samp.sh | bash

This script will install everything needed to use
i2s amplifier

--- Warning ---

Always be careful when running scripts and commands
copied from the internet. Ensure they are from a
trusted source.

If you want to see what this script does before
running it, you should run:
  \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N] y

Checking hardware requirements...

Adding Device Tree Entry to /boot/config.txt
dtoverlay already active

Commenting out Blacklist entry in
/etc/modprobe.d/raspi-blacklist.conf

Default sound driver currently not loaded
Configuring sound output

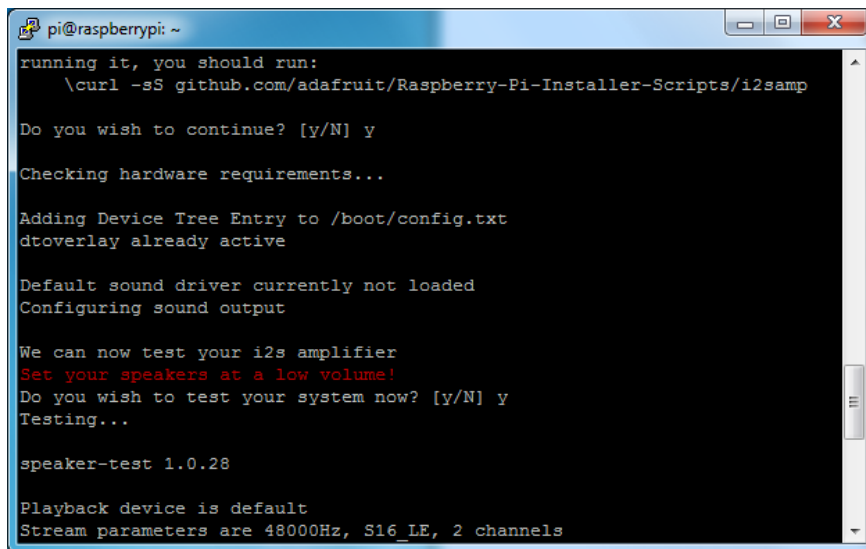
We can now test your i2s amplifier
Set your speakers at a low volume!
Do you wish to test your system now? [y/N] █
```

After rebooting, log back in and re-run the script again.

```
cd /home/pi/RaspiWWV
sh installAMP.sh
```

```
curl -sS https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2samp.sh | bash
```

It will ask to test the speaker. Answer Y

A terminal window titled 'pi@raspberrypi: ~' with standard window controls. The terminal output shows the execution of a script that checks hardware requirements, adds a device tree entry, configures sound output, and tests the i2s amplifier. The user has responded 'y' to the prompts to continue and test the system. The script concludes with the version 'speaker-test 1.0.28' and playback details: 'Playback device is default' and 'Stream parameters are 48000Hz, S16_LE, 2 channels'.

```
pi@raspberrypi: ~
running it, you should run:
  \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp
Do you wish to continue? [y/N] y
Checking hardware requirements...

Adding Device Tree Entry to /boot/config.txt
dtoverlay already active

Default sound driver currently not loaded
Configuring sound output

We can now test your i2s amplifier
Set your speakers at a low volume!
Do you wish to test your system now? [y/N] y
Testing...

speaker-test 1.0.28

Playback device is default
Stream parameters are 48000Hz, S16_LE, 2 channels
```

You can also run the following to test your speaker:

```
sudo amixer sset PCM,0 95%

aplay /usr/share/sounds/alsa/Front_Center.wav
```

~~Install MP3 player software~~

```
sudo apt-get install -y mpg123
```

If you don't have any sound or need more information, again check out the Adafruit Tutorial URL at the beginning of the Audio section, above.

```
cd /home/pi/RaspiWWV
sh ./installWWV.sh
```

```
sh test.sh
```

```
sudo amixer sset PCM,0 95%
```

```
sudo alsamixer
```

Esc key to exit.

[illegible]

Reboot your Pi and after about a minute, you should start hearing the “WWV” time signal.

Sources (and MANY THANKS!)

- WWV Audio – from Mxsmanic
<https://freesound.org/people/Mxsmanic/sounds/139323/>
- OLED SSD1306 Tutorial and Library
<https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/overview>
https://github.com/adafruit/Adafruit_Python_SSD1306
- RTC Clock Tutorial
<https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/wiring-the-rtc>
- Adafruit MAX98357 I2S Class-D Mono Amp Tutorial
<https://learn.adafruit.com/adafruit-max98357-i2s-class-d-mono-amp/raspberry-pi-usage>