Digital Waveform Analysis Monitor
User Manual

Table of Contents

Disclaimer:

All software of the Digital Waveform Analysis Monitor (DWAM) is downloaded, written, compiled, and run using an Orange Pi 3 LTS, and other microcontrollers, whether it be different versions of the Orange Pi or another type of microcontroller, could vary in set up.

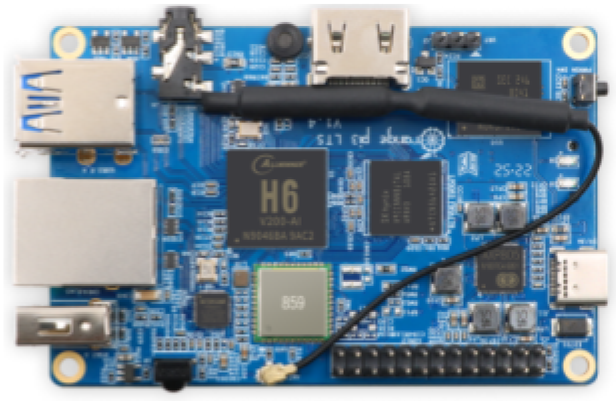
Figure 1: Orange Pi 3 LTS

Image Setup

Linux-Based Debian Image:

To set up the image environment for the Orange Pi, follow the following steps.

1. Download the debian image on your computer https://drive.google.com/drive/folders/1ctuKgHNN9r517tiAv9GGGaR7UYQgZiXP
2. Unzip the image and extract the files
3. Use a disk image burner to burn the img file that was extracted onto the microSD
    a. A recommended image burner is Win32
4. Plug the micro SD into Orange Pi

Open OPi Interface:

Required Software

Terminal Emulator:

To begin downloading and writing software for the DWAM, the user should begin by opening the Terminal Emulator, which can be found by right-clicking the desktop and navigating to Applications>Terminal Emulator.

Sudo Commands:

Everything must be run with sudo for the commands. Sudo allows users to run commands as the root user. In order for the SPI and I2C to work, everything needs to be run with sudo since these are only allowed by the root user permissions. Also, files to be edited are restricted to only the root user. If prompted, the password is ***orangepi*** as default. In order to download packages, internet is required. After everything is setup, disable networking on the device.

Python 3:

The program used to read squelch, convert the waveforms into a .wav file, and save the DateTime are all written using Python3.
First, begin by installing python3 with the following command:

sudo apt-get install python3-dev

Next, install pip3 running:

sudo apt install python3-pip

Finally, install wheel by running the following:

sudo pip3 install wheel

SPI and I2C Interface

Overlays:

In order for SPI and I2C to be enabled, overlays need to be added to the orangepiEnv.txt. This file acts as a boot configuration file for the Orange Pi.
To add these overlays:

1. Navigate to the root directory: *cd /*
2. Navigate to the etc folder: *cd boot/*
3. Open a code editor on the file orangepiEnv.txt: *sudo nano orangpiEnv.txt*
4. Add I2C and SPI to the overlays: *overlays=spi-spidev1 i2c0*
5. Reboot the Orange Pi in order to configure the overlays: *sudo reboot*

SPI:

SPI allows the Orange Pi to interface with the analog-to-digital converter. This will send the device the digital version of the signal from the receiver.

The only step is to install spidev (python's spi library):

sudo pip3 install spidev

You can test SPI by

1. Navigate to Capstone: *cd ~/Capstone*
2. Run spi_test.py: *sudo python3 spi_test.py*

I2C:

In order for the RTC to work, we need to add a couple modules to the device. Follow these steps to do so.

1. Navigate to the / directory: *cd /*
2. Navigate to etc directory: *cd etc/*
3. Edit the modules file: *sudo nano modules*
4. Add these to the end of the file:

   i2c-dev
5. Reboot the Orange Pi: *sudo reboot*

Since I2C is added to the overlays, there is no additional setup needed to run the DWAM. However, for debugging purposes, if an issue arises, we recommend downloading tools for debugging.

1. Install i2c-tools: *sudo apt install -y i2c-tools*
2. Run check on i2c0: *sudo i2cdetect -y 0*

Typically, the real-time clock will display as UU on address 0x68. If the clock does not attach correctly, it will display as 68 on 0x68.

GPIO Read

Install wiringOP:

For debugging purposes, users can install wiringOP and read all the gpio pins and their modes.

1. Clone wiringOP: *git clone https://github.com/orangepi-xunlong/wiringOP*
2. Move to wiringOP directory: *cd wiringOP*
3. Build wiringOP clean: *sudo ./build clean*
4. Build wiringOP: *sudo ./build*
5. Read pins: *gpio readall*

OPi.GPIO:

For the DWAM to run, the python library for reading GPIO pins must be installed.

sudo pip3 install OPi.GPIO

Squelch Break:

Squelch break is a GPIO read. When the read is 1 (high), the squelch is broken. When the read is 0 (low), the squelch is unbroken.

This can be tested by running the gpio_test.py.

1. Navigate to Capstone: *cd ~/Capstone*
2. Run spi_test.py: *sudo python3 gpio_test.py*

MCP ADC Setup

Spidev:

Spidev is the Python library used to interface with the MCP3008. The Spidev library requires some setup in order to be able to be used.

1. Import the libraries and open the SPI bus
2. Read the channel
3. Convert to volts (if necessary)

All this code can be referenced in the spi_test.py file and in the Appendix. This same code is used in the main.py file but is used when called upon.

Saving Data

Open File:

Open the file with the file name and set the number of channels, sample width, and frame rate parameters.

Read Data:

All data is being saved locally first in a list for speed. All data in the list is shifted down by 511 to have equally negative and positive parts since the range of the data from the ADC is 0 to 1023. It is multiplied by 60 in order to amplify the signal to be louder.

Write wav:

The data is then formatted to be added to the wave file and then added.

RTC Date/Time Setup

Pi-hats:

To set up the RTC ds1307 external date/time module, the following steps must be strictly followed to not damage the modules I2C.

1. Clone the following repo:

   cd; git clone https://github.com/Seeed-Studio/pi-hats.git
2. Open the tools directory:

   cd ~/pi-hats/tools
3. Install the drivers

   sudo ./install.sh -u rtc_ds1307
4. Sync the settings

   sync

Setting Time on RTC:

To set the time for the first time, use the following steps.

1. Disconnect the RTC from the Orange Pi
2. Boot the Pi
3. Have the time set to local time through wifi
4. In the terminal, switch to root user: *sudo -i*
5. Set the rtc device: *echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new device*
6. Set the time of the rtc: *hwclock -w -f /dev/rtc1*
7. Reboot: *sudo reboot*

Modules:

Finally, to finish setting up the ds1307, the user must add the ds1307 into the modules. To edit the modules file, simply use the following command:

sudo nano /etc/modules

Then add the following line to the end of the modules file:

Rtc-ds1307

As long as I2C has been set up correctly, according to the prior I2C Interface setup, the ds1307 should be ready to keep the time with no network connection.

DWAM on Startup

Rc.local:

The rc.local file allows the Orange Pi, and, subsequently, the DWAM to run scripts on startup. This is extremely important for the DWAM to run without a user interface. For if any of the lines within rc.local fail the subsequent lines will not run.

To Navigate to rc.local file, follow these steps:

1. Navigate to the root directory: *cd /*
2. Navigate to the etc folder: *cd etc/*
3. Open a code editor on the rc.local file: *sudo nano rc.local*

Note: In the following sections on how to set up rc.local, the command “echo” will be seen multiple times This command allows the system to send a response to the console, making it possible to sign into sudo commands without a user interface. In these examples, the password to log into sudo commands is *orangepi* but this can be replaced by whatever password is chosen by the user.

Get DateTime:

When turning on the DWAM, some software needs to be set up before the main DWAM script can be run. One of these is the software to connect and read the proper date and time from the external RTC DS1307 module.

To set up the RTC date and time, the device must first be declared on the I2C system by typing the following command in rc.local:

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new device
```

Then the date and time can be set using the pi-hats “set” command for the DS1307. Do this by adding the following command directly below the previous:

```
Echo orangepi | sudo -S hwclock -s -f /dev/rtc1
```

These two lines will create a new device on the I2C interface and set the system clock to match the rtc date and time.

Mount Flash:

To ensure that the DWAM writes the WAV files to the correct flash drive without a user interface, the flash drive must be dynamically discovered and mounted to the Orange Pi 3. To do this, the following line must be added to rc.local:

```
Echo orangepi | sudo -S mount /dev/sda1 /mnt/
```

This line will mount the first external drive found as the /mnt file path within the system and will allow the program to save files directly to the flash drive.

Running the Program:

Finally, to run the main file that performs all of the DWAM tasks, a simple python3 run command:

```
Echo orangepi | sudo -S python3 /home/orangepi/Capstone/main.py &
```

This command will run the main.py file that the user created earlier. If the file path to main.py is different than the one above, just delete the given file path and replace it with the correct one starting from the root directory. Finally, the *&* sign is imperative within the command as it will fork a new process to run the file on. Since the main.py file runs on an indefinite loop, the forked process will allow the orangepi to finish setting up while simultaneously beginning to run the python file.

Using the DWAM

The DWAM runs automatically after being plugged in as long as the SD card, RTC, and Flash Drive are all properly plugged into the board.

Power On/Off:
To power the DWAM on and off after plugging the Orange Pi in, there is a simple power button on the same side as the power cord plugin. To turn the Orange Pi off, simply click this button once. To turn it back on, hold the button down until the power button turns green.

Removing Flash:
The flash drive can be removed at any time. However, after plugging the flash drive back into the Orange Pi, the DWAM power must be cycled so that the flash drive can be properly mounted.

Date/Time Battery:
The external Date and Time RTC DS1307 has its own button battery attached that allows for date and time to remain without power. Therefore the button battery must be replaced about every year. If this battery is replaced while still connected to a powered DWAM, then there should be no need for the DS1307 to be reset. However, if unplugged while switching the battery, the date and time are subject to change, and the steps discussed in **RTC Date/Time Setup** may need to be redone.

Appendix

main.py:

```
# !/usr/bin/python
import OPi.GPIO as GPIO
import wave, struct, math
import spidev
import time
import os
from pydub import AudioSegment
from datetime import datetime
# datetime object containing current date and time
now = datetime.now()


# Open SPI bus
spi = spidev.SpiDev()
spi.open(1, 0)
spi.max_speed_hz = 10000000
# Define sensor channels
audio_pos_channel = 0
# Setup GPIO Pins
GPIO.cleanup()
GPIO.setmode(GPIO.SUNXI)
GPIO.setup('PD15', GPIO.IN)

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def read_channel(channel):
        adc = spi.xfer2([1, (8 + channel) << 4, 0])
        data = ((adc[1] & 3) << 8) + adc[2]
        return data

# Gets the current time and adds it to mnt path as string
def get_time():
        # datetime object containing current date and time
        now = datetime.now()
        dt_string = now.strftime("%m-%d-%Y_%H.%M.%S")
        return '/mnt/' + dt_string
```

```
def main():
        while True:
                # Wait until Squelch Breaks
                if GPIO.input('PD15'):
                        # get the file name which is the time and date
                        filename = get_time() + ".wav"
                        # setup for data reading
                        datalist_audio = []
                        obj = wave.open(filename, 'w')
                        obj.setnchannels(1)   # mono
                        obj.setsampwidth(2)
                        start = time.perf_counter()
                        # read until squelch is unbroken
                        while GPIO.input('PD15'):
                                audio_pos =
(read_channel(audio_pos_channel)-511)*60
                                datalist_audio.append(audio_pos)


                        final =  time.perf_counter() - start
                        print(f"final time is {final:0.4f}")
                        sample_rate = len(datalist_audio)/final
                        obj.setframerate(sample_rate)
                        # write data to wave file
                        for number in datalist_audio:
                                data = struct.pack('<h', number)
                                obj.writeframesraw(data)

                        obj.close()


                        print(filename + ' is made and saved!')


if __name__ == "__main__":
        main()
```

gpio_test.py:

```python
import OPi.GPIO as GPIO

GPIO.setmode(GPIO.SUNXI)
GPIO.setup('PL10', GPIO.IN)

if GPIO.input('PL10'):
    print("high")
else:
    print("low")


GPIO.cleanup()
```

spi_test.py

```python
#!/usr/bin/python
import spidev
import time
import os

# Open SPI bus
spi = spidev.SpiDev()
spi.open(1,0)
spi.max_speed_hz=1000000

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data

# Function to convert data to voltage level,
# rounded to specified number of decimal places.
def ConvertVolts(data,places):
  volts = (data * 5) / float(1023)
  volts = round(volts,places)
  return volts
```

```
# Function to calculate temperature from
# TMP36 data, rounded to specified
# number of decimal places.
def ConvertTemp(data,places):

  # ADC Value
  # (approx)  Temp  Volts
  #    0       -50    0.00
  #   78       -25    0.25
  #  155         0    0.50
  #  233        25    0.75
  #  310        50    1.00
  #  465       100    1.50
  #  775       200    2.50
  # 1023       280    3.30

  temp = ((data * 330)/float(1023))-50
  temp = round(temp,places)
  return temp

# Define sensor channels
temp_channel  = 0

# Define delay between readings
delay = 1

while True:

  # Read the temperature sensor data
  temp_level = ReadChannel(temp_channel)
  temp_volts = ConvertVolts(temp_level,2)
  temp       = ConvertTemp(temp_level,2)

  # Print out results
  print("--------------------------------------------")
  print("Temp : {} ({}V) {} deg C".format(temp_level,temp_volts,temp))
```