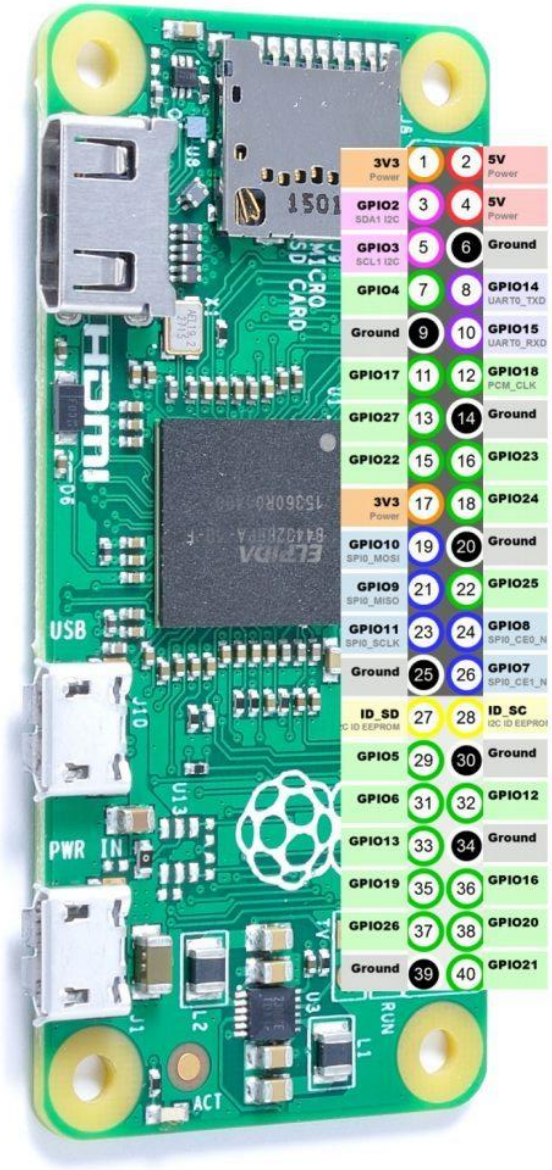
Pins on the Pi Zero W



Notes on the installation

1. SPI (“Spy”) and I2C (“Eye Squared See”) – These are two different types of communications bus. Devices are daisy-chained along the communication wires, making it easy to hook up many devices at once. The bus allows devices to connect to the controller and communicate both ways. Each allows only one device at a time to talk, so they take turns communicating.
2. Python – the controller will get its instructions from a python script that will run automatically when the controller starts up. You will be using Python version 3, so make sure the commands you use below include a ‘3’ at the end.
    a. pip3 – Pip installs add-on modules into Python. So when you need a library that someone else created, you can use pip3 to install it for you.

   b. idle3 – Idle is the program used to edit python script files.
   c. python3 – Python is the application used to run your python script files (always use the extension '.py' for these python files).
3. Blinka – This is a version of python that runs on the Raspberry Pi controller.
4. GPIO – General Purpose Inputs/Outputs. If you look at the pins on your Pi controller, you will notice several named GPIO#. These pins can be used as either inputs or outputs. Unlike the communications bus above, these pins typically connect to one device only.

Hardware

(part numbers & names come from adafruit.com)

Pi Zero W - https://www.adafruit.com/product/3400

3 pole micro switch - https://www.amazon.com/Uxcell-a12013100ux0116-Position-Vertical-Switch/dp/B007QAJUUS/ref=sr_1_11?keywords=3+pole+micro+switch&qid=1583447732&sr=8-11

Power Boost 500 - https://www.adafruit.com/product/1944

Battery (500-1200 mAh) - https://www.adafruit.com/product/258

Pi Zero Spy Camera - https://www.adafruit.com/product/3508

SD Card with NOOBS 3.0 - https://www.adafruit.com/product/1583

USB Microphone - https://www.amazon.com/gp/product/B078J9BTMF/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&psc=1

Inertial Measurement Unit (LSM6DS33+LIS3MDL) - https://www.adafruit.com/product/4485

Altitude Sensor (MPL3115A2) - https://www.adafruit.com/product/1893

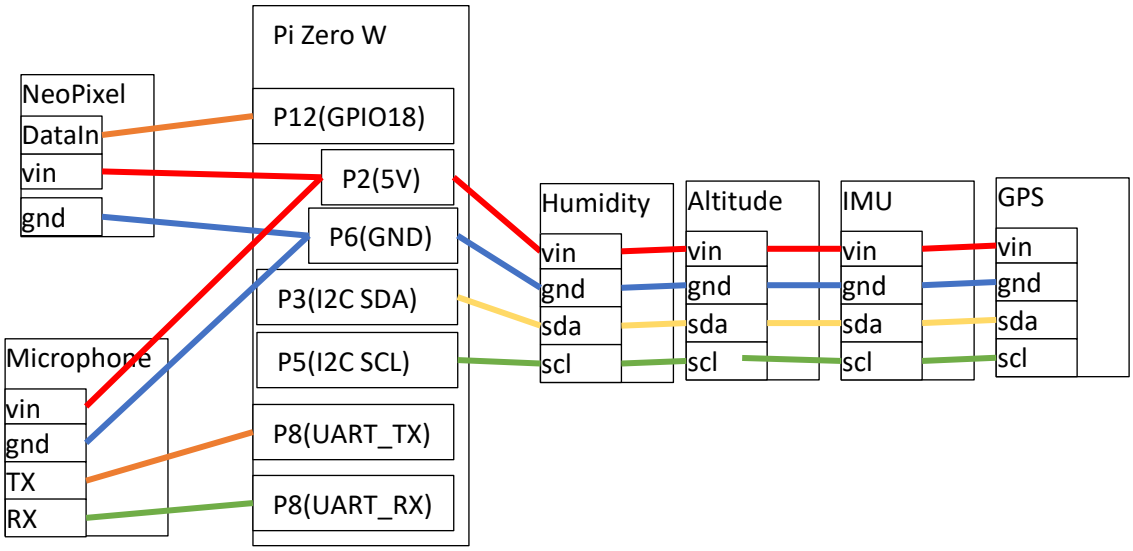
Temperature & Humidity Sensor (Si7021) - https://www.adafruit.com/product/3251

Adafruit Jewel NeoPixels - https://www.adafruit.com/product/2226

Adafruit Mini GPS PA1010D - https://www.adafruit.com/product/4415

Circuit board spacers - https://www.amazon.com/gp/product/B07D78PFQL/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1

Wiring

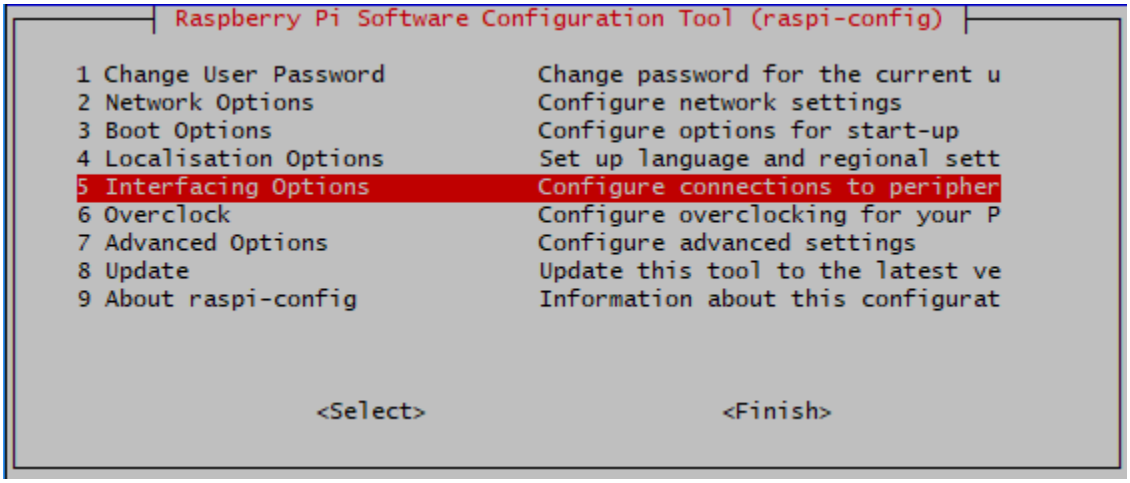


Enable the Interfaces

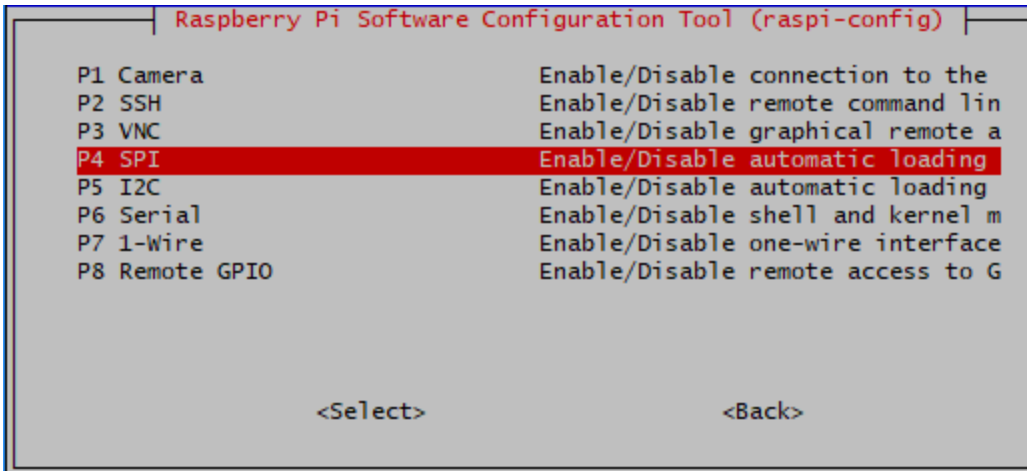
Open a command prompt and run the configuration application

```
sudo raspi-config
```

Select Interfacing Options



Enable the Camera, SPI, and I2C



Save your configuration, exit, and reboot the Pi by running the following in the command line.

```
sudo reboot
```

Update Your Pi Software

Open a command prompt and type the following

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo reboot
```

Install Circuit Python onto your Linux Python

https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/installing-circuitpython-on-raspberry-pi

```
sudo pip3 install --upgrade setuptools
```

Enable SPI

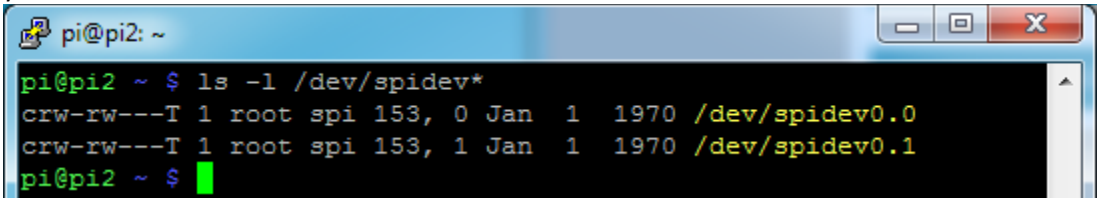
https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-spi

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

```
sudo reboot
```

```
ls -l /dev/spidev*
```

you should see two 'devices' one for each SPI bus



Enable I2C

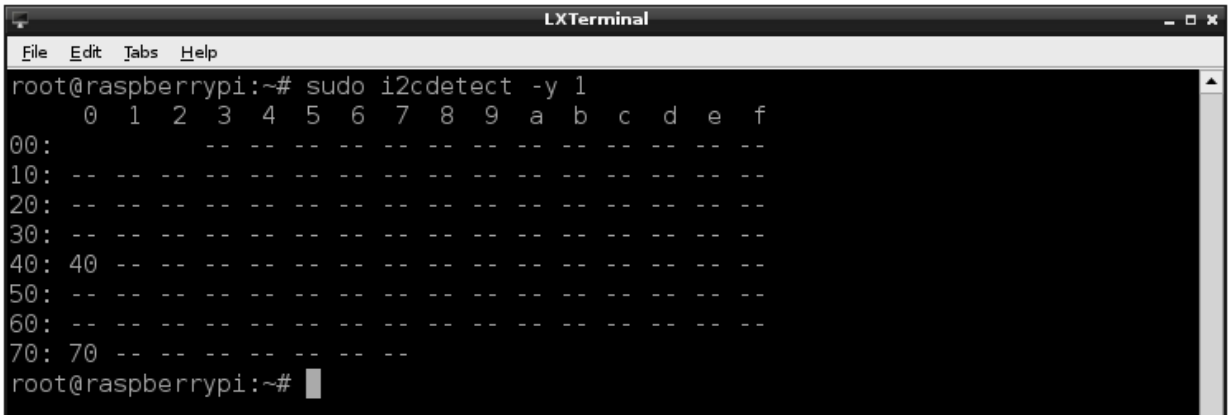
https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c

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

```
sudo reboot
```

```
sudo i2cdetect -y 1
```

If you don't have any devices hooked up yet, you will see all 0's here. Once you connect your devices up to the I2C bus, they will appear in the table below (like the 40 and 70).



Install PI GPIO

```
Sudo pip3 install RPI.GPIO
```

Install adafruit_blinka

```
Sudo pip3 install adafruit-blinka
```

Test your install of blinka

```
Idle3 blinkatest.py
```

Add the following python code to your file (do not include the line numbers)

```
1. import board
2. import digitalio
3. import busio
4.
5. print("Hello blinka!")
6.
7. # Try to great a Digital input
8. pin = digitalio.DigitalInOut(board.D4)
9. print("Digital IO ok!")
10.
11.# Try to create an I2C device
12.i2c = busio.I2C(board.SCL, board.SDA)
13.print("I2C ok!")
14.
15.# Try to create an SPI device
```

```
16. spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)
17. print("SPI ok!")
18.
19. print("done!")
```

From the **File** menu select **Save**, then from the **Run** menu select **Run**. If it runs without error, things are working correctly.

SI7021 Temp & Humidity

https://github.com/adafruit/Adafruit_CircuitPython_SI7021

https://learn.adafruit.com/adafruit-si7021-temperature-plus-humidity-sensor/circuitpython-code

```
sudo pip3 install adafruit-circuitpython-si7021
```

Test your install of the Si7021 temperature sensor

```
Idle3 temperature.py
```

Add the following python code to your file (do not include the line numbers)

```
1. import board
2. import busio
3. import adafruit_si7021
4. i2c = busio.I2C(board.SCL, board.SDA)
5. sensor = adafruit_si7021.SI7021(i2c)
6. print('Temperature: {} degrees C'.format(sensor.temperature))
7. print('Humidity: {}%'.format(sensor.relative_humidity))
```

```
>>> print('Temperature: {} degrees C'.format(sensor.temperature))
Temperature: 21.6142 degrees C
>>> print('Humidity: {}%'.format(sensor.relative_humidity))
Humidity: 48.0605%
>>>
```

Altimeter barometric pressure

https://github.com/adafruit/Adafruit_CircuitPython_MPL3115A2

Run the following in the command line (note that the name of the device MPL, so that is a lower case "L" before the number '3' and not the number '1').

```
sudo pip3 install adafruit-circuitpython-mpl3115a2
```

For altimeter you must set the value as close as possible at that moment for most accurate reading.

https://forecast.weather.gov/product.php?issuedby=BOU&product=OSO&site=bou

Test your install of the MPL3115A2 altitude sensor

```
Idle3 altitudetest.py
```

Add the following python code to your file (do not include the line numbers, you don't have to include the comments that start with '#')

```
1.  import time
2.  import board
3.  import busio
4.  import adafruit_mpl3115a2
5.
6.  # Initialize the I2C bus.
7.  i2c = busio.I2C(board.SCL, board.SDA)
8.
9.  # Initialize the MPL3115A2.
10. sensor = adafruit_mpl3115a2.MPL3115A2(i2c)
11.
12. # You can configure the pressure at sealevel to get better
    altitude estimates.
13. # This value has to be looked up from your local weather
    forecast or meteorlogical
14. # reports.  It will change day by day and even hour by hour
    with weather
15. # changes.  Remember altitude estimation from barometric
    pressure is not exact!
16. # Set this to a value in pascals:
17. sensor.sealevel_pressure = 102250
18.
19. pressure = sensor.pressure
20. print('Pressure: {0:0.3f} pascals'.format(pressure))
21. altitude = sensor.altitude
22. print('Altitude: {0:0.3f} meters'.format(altitude))
23. temperature = sensor.temperature
24. print('Temperature: {0:0.3f} degrees
    Celsius'.format(temperature))
```

Magnetometer (compass direction)

https://github.com/adafruit/Adafruit_CircuitPython_LIS3MDL

```
sudo pip3 install adafruit-circuitpython-lis3mdl
```

Test your install of the LIS3MDL magnetometer sensor

```
Idle3 compasstest.py
```

Add the following python code to your file (do not include the line numbers, you don't have to include the comments that start with '#')

```
1. import time
2. import board
3. import busio
4. import adafruit_lis3mdl
5.
6. i2c = busio.I2C(board.SCL, board.SDA)
7. sensor = adafruit_lis3mdl.LIS3MDL(i2c)
8.
9. while True:
10.       mag_x, mag_y, mag_z = sensor.magnetic
11.
12.       print('X:{0:10.2f}, Y:{1:10.2f}, Z:{2:10.2f}
   uT'.format(mag_x, mag_y, mag_z))
13.       print('')
14.       time.sleep(1.0)
```

IMS (gyroscope and accelerometers):

https://github.com/adafruit/Adafruit_CircuitPython_LSM6DS

```
sudo pip3 install adafruit-circuitpython-lsm6ds
```

Test your install of the LSM6DS rotation and acceleration sensor

```
Idle3 acceltest.py
```

Add the following python code to your file (do not include the line numbers, you don't have to include the comments that start with '#')

```
1.  import time
2.  import board
3.  import busio
4.  import adafruit_lsm6ds
5.
6.  i2c = busio.I2C(board.SCL, board.SDA)
7.
8.  sox = adafruit_lsm6ds.LSM6DSOX(i2c)
9.
10. while True:
11.     print("Acceleration: X:%.2f, Y: %.2f, Z: %.2f
    m/s^2"%(sox.acceleration))
12.     print("Gyro X:%.2f, Y: %.2f, Z: %.2f
    degrees/s"%(sox.gyro))
13.     print("")
14.     time.sleep(0.5)
```

PI CAM

https://github.com/iizukanao/picam

Make sure the camera has been enabled on your Pi.

Install dependencies

```
sudo apt-get install libharfbuzz0b libfontconfig1
```

Create a file that will set up the directories and environment for recording video.

```
idle3 make_dirs.sh
```

Add the following to your file, then save and close the file.

```
#!/bin/bash
DEST_DIR=~/picam
SHM_DIR=/run/shm

mkdir -p $SHM_DIR/rec
mkdir -p $SHM_DIR/hooks
mkdir -p $SHM_DIR/state
mkdir -p $DEST_DIR/archive

ln -sfn $DEST_DIR/archive $SHM_DIR/rec/archive
ln -sfn $SHM_DIR/rec $DEST_DIR/rec
ln -sfn $SHM_DIR/hooks $DEST_DIR/hooks
ln -sfn $SHM_DIR/state $DEST_DIR/state
EOF
```

Make the file above executable and run it.

```
chmod +x make_dirs.sh
```

```
./make_dirs.sh
```

Install picam application

```
wget https://github.com/iizukanao/picam/releases/download/v1.4.7/picam-1.4.7-binary.tar.xz
```

```
tar xvf picam-1.4.7-binary.tar.xz
```

```
cp picam-1.4.7-binary/picam ~/picam/
```

Test your installation

Start picam

```
cd ~/picam
./picam --alsadev hw:1,0
```

In another command line window

Start recording

```
touch hooks/start_record
```

Stop recording

```
touch hooks/stop_record
```

Play your recording

```
sudo apt-get install vlc
cd ~/picam/archive
ls *.ts
vlc <name>.ts
```

Test your picam install in python. Edit the make_dirs.sh file to include the line that starts your camera.

```
idle3 make_dirs.sh
```

Add the very last line to your file, then save and close the file.

```
#!/bin/bash
DEST_DIR=~/picam
SHM_DIR=/run/shm

mkdir -p $SHM_DIR/rec
mkdir -p $SHM_DIR/hooks
mkdir -p $SHM_DIR/state
mkdir -p $DEST_DIR/archive

ln -sfn $DEST_DIR/archive $SHM_DIR/rec/archive
ln -sfn $SHM_DIR/rec $DEST_DIR/rec
ln -sfn $SHM_DIR/hooks $DEST_DIR/hooks
ln -sfn $SHM_DIR/state $DEST_DIR/state
```

```
./picam --alsadev hw:1,0

EOF
```

The create a test python script for the camera.

Idle3 camtest.py

Add the following python code to your file (do not include the line numbers, you don't have to include the comments that start with '#')

```
1. import board
2. import subprocess
3. from pathlib import Path
4. import time
5. import os
6.
7. #start the camera
8. home_dir = '/home/pi/picam'
9. os.chdir(home_dir)
10. camera=subprocess.Popen('./make_dirs.sh')
11.
12. #give it time to wake up
13. time.sleep(3)
14.
15. #Start recording
16. Path('./hooks/start_record').touch()
17.
18. print('started recording')
19. subs = './hooks/subtitle'
20. start_time=time.time()
21. while (time.time() - start_time)< 15:
22.     if os.path.isfile(subs):
23.         os.remove(subs)
24.     With open(subs) as subtitle:
25.         subtext = 'text={}'.format(time.time())
26.         subtitle.write(subtext)
27.         print(subtext)
28.     time.sleep(1)
29.
30. #Stop recording
31. Path('./hooks/stop_record').touch()
32. camera.kill()
```

NeoPixels

https://learn.adafruit.com/neopixels-on-raspberry-pi/python-usage

```
sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel
```

Test your install of the LSM6DS rotation and acceleration sensor

```
Idle3 pixeltest.py
```

Add the following python code to your file (do not include the line numbers, you don't have to include the comments that start with '#')

```
33. import board
34. import neopixel
35. pixels = neopixel.NeoPixel(board.D18, 7)
36. pixels[0] = (255, 0, 0)
37. pixels.fill((0, 0, 255))
38. pixels.fill((255, 0, 0))
39. pixels.fill((255, 255, 255))
```

Logging your data

Add the following to your python code at the top of your python file.

```
1. import logging
2. import time
3. logging.basicConfig(filename='Launch-
   {}.log'.format(time.time()),format='%(asctime)s,%(relativeCreated
   )s,%(message)s',level=logging.DEBUG)
```

Example use case for logging your data

```
logging.info('{},{}'.format('name',42))
```

Final Python Program (note you have to run this as sudo to make the pixels work). Save this file in /rocket/rocket.py

```
import logging
import time
import board
import neopixel
import digitalio
import busio
import adafruit_si7021
import adafruit_mpl3115a2
import adafruit_lis3mdl
import adafruit_lsm6ds
from pathlib import Path
import subprocess

logging.basicConfig(filename='Launch-{}.log'.format(time.time()),
format='%(asctime)s,%(relativeCreated)s,%(message)s', level=logging.DEBUG)
pixels = neopixel.NeoPixel(board.D18, 7)

# Flash the pixels to show we've started
```

```
pixels.fill((0, 0, 255))
time.sleep(0.5)
pixels.fill((255, 0, 0))
time.sleep(0.5)
pixels.fill((255, 255, 255))
time.sleep(0.5)

logging.info('Log your data {}'.format(42))

# Try to get a Digital input
pin = digitalio.DigitalInOut(board.D4)
logging.info("IO ok!")

# Try to create an I2C device
i2c = busio.I2C(board.SCL, board.SDA)
tempSensor = adafruit_si7021.SI7021(i2c)
altSensor = adafruit_mpl3115a2.MPL3115A2(i2c)
altSensor.sealevel_pressure = 102250
magSensor = adafruit_lis3mdl.LIS3MDL(i2c)
soxSensor = adafruit_lsm6ds.LSM6DSOX(i2c)
logging.info("I2C ok!")

# Try to create an SPI device
spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)
logging.info("SPI ok!")

# Green light for launch
logging.info("All systems go! Ready for launch")
pixels.fill((0, 255, 0))

## Log initial values ##
# Log temperature and humidity
logging.info('Temperature1, {}, C'.format(tempSensor.temperature))
logging.info('Humidity, {}, %'.format(tempSensor.relative_humidity))

# Log pressure, altitude, and temperature
logging.info('Pressure, {0:0.3f}, pascals'.format(altSensor.pressure))
logging.info('Altitude, {0:0.3f}, meters'.format(altSensor.altitude))
logging.info('Temperature2, {0:0.3f}, C'.format(altSensor.temperature))

# Log magnetic compass settings
mag_x, mag_y, mag_z = magSensor.magnetic
logging.info('Compass, X, {0:10.2f}, Y, {1:10.2f}, Z, {2:10.2f}, uT'.format(mag_x, mag_y,
mag_z))

# Log the acceleration and gyro
logging.info("Acceleration, X, %.2f, Y, %.2f, Z, %.2f, m/s^2"%(soxSensor.acceleration))
logging.info("Gyro, X, %.2f, Y, %.2f, Z, %.2f, degrees/s"%(soxSensor.gyro))

picam_dir = '/home/pi/picam/'
startCamera = './make_dirs.sh'
startRecord = './hooks/start_record'
sub_path = "./hooks/subtitle"
stopRecord = './hooks/stop_record'

# Start picam
os.chdir(picam_dir)
camera = subprocess.Popen([startCamera])
time.sleep(3)

# Start recording the video
```

```
Path(startRecord).touch()
startTime = time.time()
pixels.fill((255, 255, 255))

# video will record for 10 minutes
while (time.time() - startTime) < 600:
    pixels.fill((0, 255, 0))

    # Log temperature and humidity
    logging.info('Temperature1, {}, C'.format(tempSensor.temperature))
    logging.info('Humidity, {}, %'.format(tempSensor.relative_humidity))

    # Log pressure, altitude, and temperature
    logging.info('Pressure, {0:0.3f}, pascals'.format(altSensor.pressure))
    altitude = altSensor.altitude
    logging.info('Altitude, {0:0.3f}, meters'.format(altitude))
    logging.info('Temperature2, {0:0.3f}, C'.format(altSensor.temperature))

    # Log magnetic compass settings
    mag_x, mag_y, mag_z = magSensor.magnetic
    logging.info('Compass, X, {0:10.2f}, Y, {1:10.2f}, Z, {2:10.2f}, uT'.format(mag_x, mag_y,
mag_z))

    # Log the acceleration and gyro
    logging.info("Acceleration, X, %.2f, Y, %.2f, Z, %.2f, m/s^2"%(soxSensor.acceleration))
    logging.info("Gyro, X, %.2f, Y, %.2f, Z, %.2f, degrees/s"%(soxSensor.gyro))

    # set the subtitle on the video with time and height
    pixels.fill((255, 255, 255))
    if os.path.isfile(sub_path):
        os.remove(sub_path)
    with open(sub_path, 'a') as subtitle:
        subtitle.write('text={} seconds {} meters'.format((time.time()-startTime), altitude))


# Stop recording the video
Path(stopRecord).touch()
camera.kill()

# flash the lights so we can be found
while True:
    pixels.fill((255, 255, 255))
    time.sleep(0.25)
    pixels.fill((255, 0, 0))
    time.sleep(0.25)
    pixels.fill((0, 255, 0))
    time.sleep(0.25)
    pixels.fill((0, 0, 255))
    time.sleep(0.25)
```

GPS

https://github.com/adafruit/Adafruit_CircuitPython_GPS

```
sudo pip3 install adafruit-circuitpython-gps
```

```
# Simple GPS module demonstration.
# Will print NMEA sentences received from the GPS, great for testing connection
# Uses the GPS to send some commands, then reads directly from the GPS
```

```
import time
import board
import busio

import adafruit_gps

# Create a serial connection for the GPS connection using default speed and
# a slightly higher timeout (GPS modules typically update once a second).
# These are the defaults you should use for the GPS FeatherWing.
# For other boards set RX = GPS module TX, and TX = GPS module RX pins.
uart = busio.UART(board.TX, board.RX, baudrate=9600, timeout=10)

# for a computer, use the pyserial library for uart access
#import serial
#uart = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=10)

# If using I2C, we'll create an I2C interface to talk to using default pins
#i2c = busio.I2C(board.SCL, board.SDA)

# Create a GPS module instance.
gps = adafruit_gps.GPS(uart)     # Use UART/pyserial
#gps = adafruit_gps.GPS_GtopI2C(i2c)  # Use I2C interface

# Initialize the GPS module by changing what data it sends and at what rate.
# These are NMEA extensions for PMTK_314_SET_NMEA_OUTPUT and
# PMTK_220_SET_NMEA_UPDATERATE but you can send anything from here to adjust
# the GPS module behavior:
#   https://cdn-shop.adafruit.com/datasheets/PMTK_A11.pdf

# Turn on the basic GGA and RMC info (what you typically want)
gps.send_command(b'PMTK314,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0')
# Turn on just minimum info (RMC only, location):
#gps.send_command(b'PMTK314,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0')
# Turn off everything:
#gps.send_command(b'PMTK314,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0')
# Tuen on everything (not all of it is parsed!)
#gps.send_command(b'PMTK314,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0')

# Set update rate to once a second (1hz) which is what you typically want.
gps.send_command(b'PMTK220,1000')
# Or decrease to once every two seconds by doubling the millisecond value.
# Be sure to also increase your UART timeout above!
#gps.send_command(b'PMTK220,2000')
# You can also speed up the rate, but don't go too fast or else you can lose
# data during parsing.  This would be twice a second (2hz, 500ms delay):
#gps.send_command(b'PMTK220,500')

# Main loop runs forever printing data as it comes in
timestamp = time.monotonic()
while True:
    data = gps.read(32)  # read up to 32 bytes
    # print(data)  # this is a bytearray type

    if data is not None:
        # convert bytearray to string
```

```
        data_string = ''.join([chr(b) for b in data])
        print(data_string, end="")

    if time.monotonic() - timestamp > 5:
        # every 5 seconds...
        gps.send_command(b'PMTK605')  # request firmware version
        timestamp = time.monotonic()
```

Set up for Auto-Start

https://www.linux.com/tutorials/setting-timer-systemd-linux/

Sometimes it is nice to have your device boot up and start doing something. This will show you how to make your PI run your python script on startup. There are numerous ways to accomplish this, I'm showing you the **systemd** option.

1. Create a "Unit File":
    a. Type in the command prompt "**sudo nano /lib/systemd/system/rocket.service**"
    b. In the editor make sure your file looks like the following:

```
[Unit]
 Description=My Model Rocket Service
 After=multi-user.target

 [Service]
 Type=idle
 ExecStart=/usr/bin/python3 ./rocket/rocket.py

 [Install]
 WantedBy=multi-user.target
```

    c. Exit and save the file
        i. **CTRL x**, **Y** to save the changes, and press **Enter** to write to the unit file.
2. Configure ***systemd***
    a. Enable the unit file via system control
        i. Type "**sudo systemctl daemon-reload**"
        ii. Then type "**sudo systemctl enable rocket.service**"
    b. Reboot the machine by typing "**sudo reboot**"
    c. Upon reboot, your pi should light up and be running your code

Troubleshooting

"TypeError: unsupported operand type(s) for -=: 'Retry' and 'int'"

Means your PI Zero has lost its network connection