Raspberry Pi based Oscilloscope

Installation Manual

One of the most important tools in Electrical/Electronic engineering is **The Oscilloscope**.

An oscilloscope is a laboratory instrument commonly used to display and analyze the waveform of electronic signals. In effect, the device draws a graph of the instantaneous signal voltage as a function of time.

In this project we will seek to replicate the signal visualization capabilities of the oscilloscope using the Raspberry Pi and an analog to digital converter module.

Replicating the signal visualization of the oscilloscope using the Raspberry Pi will require the following steps;

1. Perform Digital to analog conversion of the Input signal
2. Prepare the resulting data for representation
3. Plot the data on a live time graph

Analog Signals → **Signal Processing** → **Visualize using the Graph**

- **Hardware Requirements**

  1. Raspberry Pi Model A/B/B+
  2. ADS1115 ADC
  3. Breadboard
  4. Jumper Wires

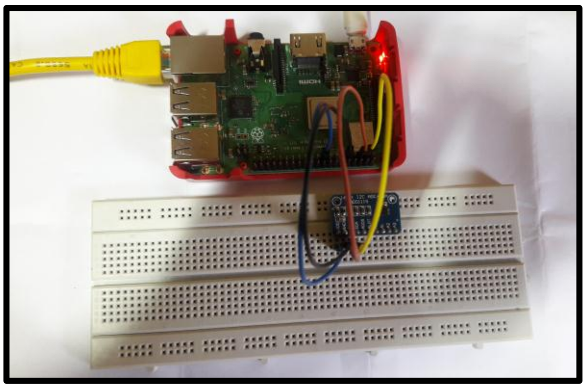
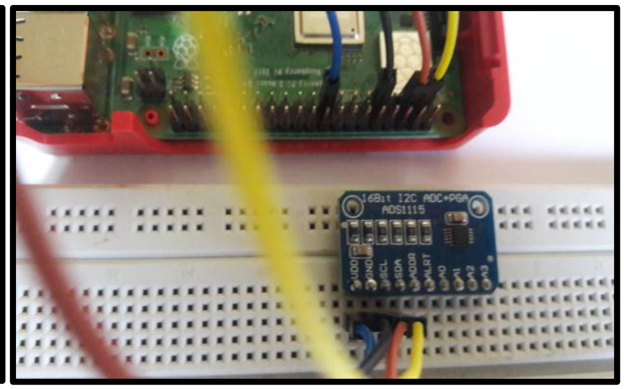
ADS1115 ADC chip is used to convert the analog input signals to digital signals which can be visualized with the Raspberry Pi. This chip is important because the Raspberry Pi does not have an on-board analog to digital converter (ADC).

- **Software Requirements**

  1. Raspbian Stretch OS
  2. Adafruit module for interfacing with the ADS1115 ADC chip
  3. Python Module **matplotlib** used for data visualization

1. **Connect your ADC with Raspberry Pi's GPIO Pins.**

ADS1115 ADC	Pin Number	GPIO Number
VDD	Pin 17	3.3v
GND	Pin 9	GND
SCL	Pin 5	GPIO 3
SDA	Pin 3	GPIO 2

Step 1: Enable Raspberry Pi I2C interface

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ sudo raspi-config
```

Go to Interfacing Options → I2C → Enable (Yes)

Step 2: Update the Raspberry pi

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ sudo apt-get update
```

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ sudo apt-get upgrade
```

Step 3: Install the Adafruit ADS1115 library for ADC

To install the dependencies starting with the Adafruit python module for the ADS115 chip, Ensure you are in the Raspberry Pi home directory ($ cd ~)

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ sudo apt-get install build-essential python-dev python-smbus git
```

Next, clone the Adafruit git folder for the library by running

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ git clone https://github.com/adafruit/Adafruit_Python_ADS1x15.git
```

Change into the cloned file's directory and run the setup file

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ sudo python setup.py install
```

Step 4: Test the library and 12C communication.

Now,it is important to test the library and ensure the ADC can communicate with the raspberry pi over I2C. To do this use an example script that comes with the library.

$ cd examples

$ python simpletest.py

If the I2C module is enabled and connections good, it should display the data as below

If an error occurs, check to ensure the ADC is well connected to the PI and I2C communication is enabled on the Pi.

Step 5: Install *Matplotlib*

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ sudo apt-get install python-matplotlib
```

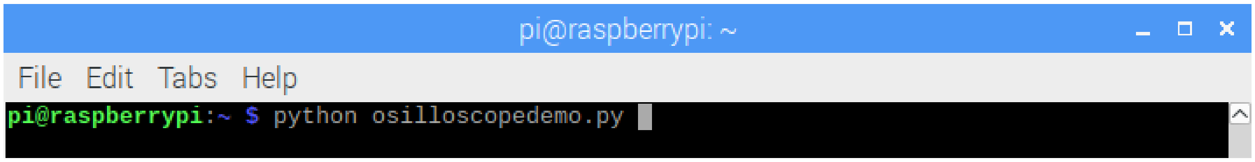
With all the dependencies installed, we are now ready to write the code.

At this stage it is important to switch to a monitor or use the VNC viewer (or Remote Desktop Connection), anything through which you can see your Raspberry Pi's desktop, as the graph being plotted won't show on the terminal.

Step 6 :Python Code for Raspberry Pi Oscilloscope:

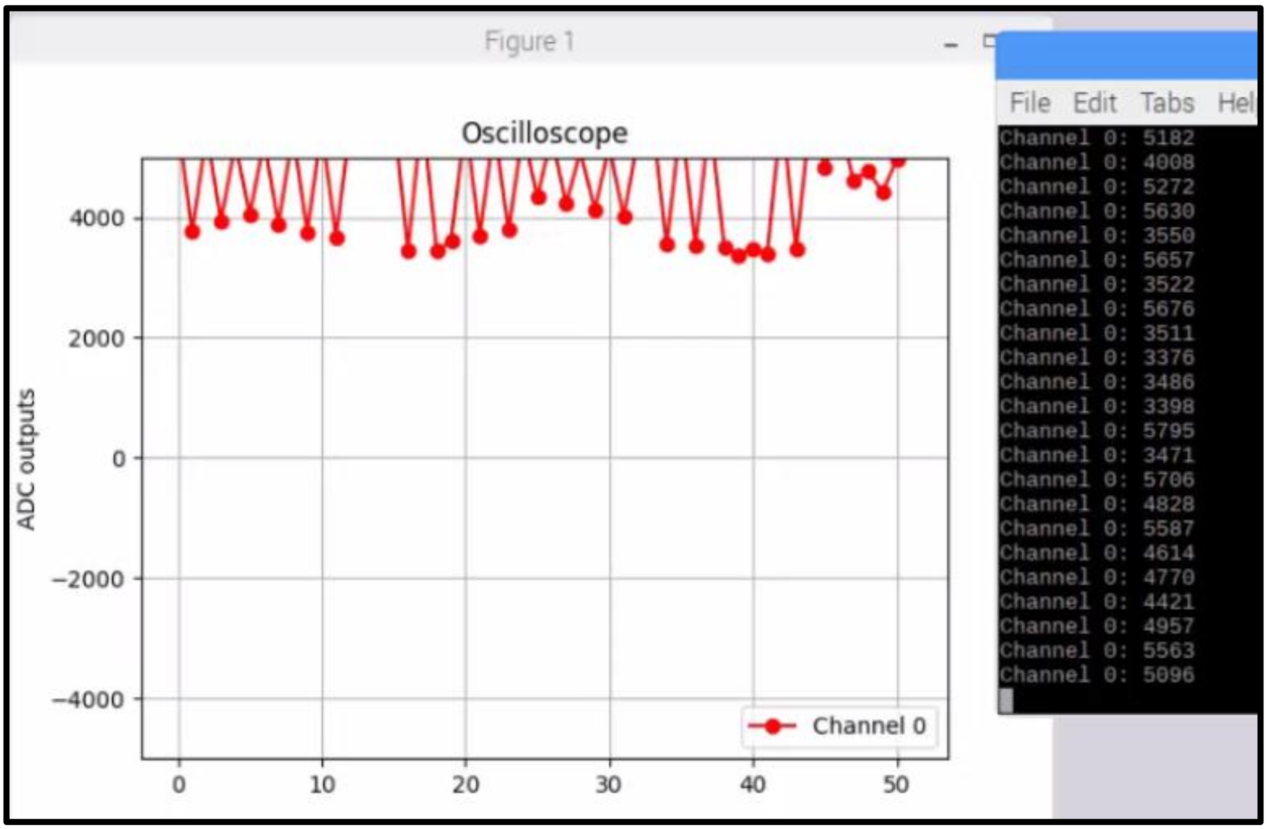
```
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import Adafruit_ADS1x15

# Create an ADS1115 ADC (16-bit) instance.
adc = Adafruit_ADS1x15.ADS1115()

GAIN = 1
val = [ ]


# Start continuous ADC conversions on channel 0 using the previous gain value.
adc.start_adc(0, gain=GAIN)
print('Reading ADS1x15 channel 0')

fig, ax = plt.subplots()
ax.set_ylim(-5000,5000)
ax.set_title('Oscilloscope')
ax.grid(True)
ax.set_ylabel('ADC outputs')

line, = ax.plot([], 'ro-', label='Channel 0')
ax.legend(loc='lower right')

def update(cnt):
    # Read the last ADC conversion value and print it out.
    value = adc.get_last_result()
    print('Channel 0: {0}'.format(value))
    # Set new data to line
    line.set_data(list(range(len(val))), val)
    ax.relim()
    ax.autoscale_view()
    #Store values for later
    val.append(int(value))
    if(cnt>50):
        val.pop(0)

ani = FuncAnimation(fig, update, interval=500)
plt.show()
```

Save the code and run using



That's all !!!

ADC data being printed on the terminal,and related Plot is also visible.



Thank you….