**Practical-1**

Displaying different LED patterns with Raspberry Pi.

**Practical-2**

Displaying Time over 4-Digit 7-Segment Display using Raspberry Pi.

**Practical-3**

Controlling Raspberry Pi with Telegram.

**Practical-4**

Interfacing Raspberry Pi with Pi Camera.

**Practical-5**

Raspberry Pi Based Oscilloscope

### ****Project Requirements****

The requirement for this project can be classified into two:

1. Hardware Requirements
2. Software Requirements

**Hardware requirements**

To build this project, the following components/part are required;

1. Raspberry pi 2 (or any other model)
2. 8 or 16GB SD Card
3. LAN/Ethernet Cable
4. Power Supply or USB cable
5. ADS1115 ADC
6. LDR (Optional as its meant for test)
7. 10k or 1k resistor
8. Jumper wires
9. Breadboard
10. Monitor or any other way of seeing the pi’s Desktop(VNC inclusive)

**Software Requirements**

The software requirements for this project are basically the python modules (**matplotlib and drawnow**) that will be used for data visualization and the Adafruit module for interfacing with the ADS1115 ADC chip. I will show how to install these modules on the Raspberry Pi as we proceed.

While this tutorial will work irrespective of the raspberry pi OS used, I will be using the Raspberry Pi stretch OS and I will assume you are familiar with [setting up the Raspberry Pi](http://circuitdigest.com/microcontroller-projects/getting-started-with-raspberry-pi) with the Raspbian stretch OS, and you know how to SSH into the raspberry pi using a terminal software like putty. If you have issues with any of this, there are tons of [Raspberry Pi Tutorials](https://circuitdigest.com/simple-raspberry-pi-projects-for-beginners) on this website that can help.

With all the hardware components in place, let's create the schematics and connect the components together.

### ****Circuit Diagram:****

To convert the analog input signals to digital signals which can be visualized with the Raspberry Pi, we will be using the **ADS1115 ADC chip**. This chip becomes important because the Raspberry Pi, unlike Arduino and most micro-controllers, does not have an on-board analog to digital converter(ADC). While we could have used any raspberry pi compatible ADC chip, I prefer this chip due to its high resolution(16bits) and its well documented datasheet and use instructions by Adafruit. You can also check our [Raspberry Pi ADC tutorial](https://circuitdigest.com/microcontroller-projects/raspberry-pi-adc-tutorial) to learn more about it.

**ADS1115 and Raspberry Pi Connections:**

VDD – 3.3v

GND – GND

SDA – SDA

SCL – SCL

With the connections all done, power up your pi and proceed to install the dependencies mentioned below.

### ****Install Dependencies for Raspberry Pi Oscilloscope:****

Before we start writing the python script to pull data from the ADC and plot it on a live graph, we need to **enable the I2C communication interface** of the raspberry pi and install the software requirements that were mentioned earlier. This will be done in below steps so its easy to follow:

**Step 1: Enable Raspberry Pi I2C interface**

To enable the I2C, from the terminal, run;

**sudo raspi-config**

When the configuration panels open, select interface options, select I2C and click enable.

**Step 2: Update the Raspberry pi**

The first thing I do before starting any project is updating the Pi. Through this, I am sure every thing on the OS is up to date and I won’t experience compatibility issue with any latest software I choose to install on the Pi. To do this, run below two commands:

**sudo apt-get update**

**sudo apt-get upgrade**

**Step 3: Install the Adafruit ADS1115 library for ADC**

With the update done, we are now ready to install the dependencies starting with the Adafruit python module for the ADS115 chip. Ensure you are in the Raspberry Pi home directory by running;

**cd ~**

then install the build-essentials by running;

**sudo apt-get install build-essential python-dev python-smbus git**

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

**git clone** [**https://github.com/adafruit/Adafruit\_Python\_ADS1x15.git**](https://github.com/adafruit/Adafruit_Python_ADS1x15.git)

Change into the cloned file’s directory and run the setup file;

**cd Adafruit\_Python\_ADS1x1z**

**sudo python setup.py install**

**Step 4: Test the library and 12C communication.**

Before we proceed with the rest of the project, it is important to test the library and ensure the ADC can communicate with the raspberry pi over I2C. To do this we will use an example script that comes with the library.

While still in the Adafruit\_Python\_ADS1x15 folder, change directory to the examples directory by running;

**cd examples**

Next, run the sampletest.py example which displays the value of the four channels on the ADC in a tabular form.

Run the example using:

**python simpletest.py**

**Step 5: Install Matplotlib**

To visualize the data we need to install the matplotlib module which is used to plot all kind of graphs in python. This can be done by running;

**sudo apt-get install python-matplotlib**

**Step6: Install the Drawnow python module**

Lastly, we need to install the drawnow python module. This module helps us provide live updates to the data plot.

We will be installing drawnow via the python package installer; pip, so we need to ensure it is installed.  This can be done by running;

**sudo apt-get install python-pip**

We can then use pip to install the drawnow package by running:

**sudo pip install drawnow**

### ****Python Code for Raspberry Pi Oscilloscope:****

The python code for this **Pi Oscilloscope** is fairly simple especially if you are familiar with the python matplotlib module. Before showing us the whole code, I will try to break it into part and explain what each part of the code is doing so you can have enough knowledge to extend the code to do more stuffs.

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

With the monitor as the interface **open a new python file**. You can call it any name you want, but I will call it scope.py.

**sudo nano scope.py**

With the file created, the first thing we do is import the modules we will be using;

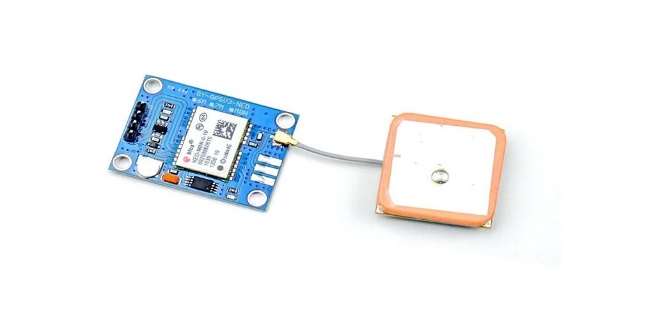
**Code:**

import time  
import matplotlib.pyplot as plt  
#import numpy  
from drawnow import \*  
# Import the ADS1x15 module.  
import Adafruit\_ADS1x15  
# Create an ADS1115 ADC (16-bit) instance.  
adc = Adafruit\_ADS1x15.ADS1115()

GAIN = 1  
val = [ ]  
cnt = 0  
plt.ion()  
# Start continuous ADC conversions on channel 0 using the previous gain value.  
adc.start\_adc(0, gain=GAIN)  
print('Reading ADS1x15 channel 0')  
#create the figure function  
def makeFig():  
    plt.ylim(-5000,5000)  
    plt.title('Osciloscope')  
    plt.grid(True)  
    plt.ylabel('ADC outputs')  
    plt.plot(val, 'ro-', label='Channel 0')  
    plt.legend(loc='lower right')  
while (True):  
    # Read the last ADC conversion value and print it out.  
    value = adc.get\_last\_result()  
    print('Channel 0: {0}'.format(value))  
    # Sleep for half a second.  
    time.sleep(0.5)  
    val.append(int(value))  
    drawnow(makeFig)  
    plt.pause(.000001)  
    cnt = cnt+1  
    if(cnt>50):  
        val.pop(0)

**PRACTICAL 6**

**Raspberry Pi GPS Module Interfacing.**

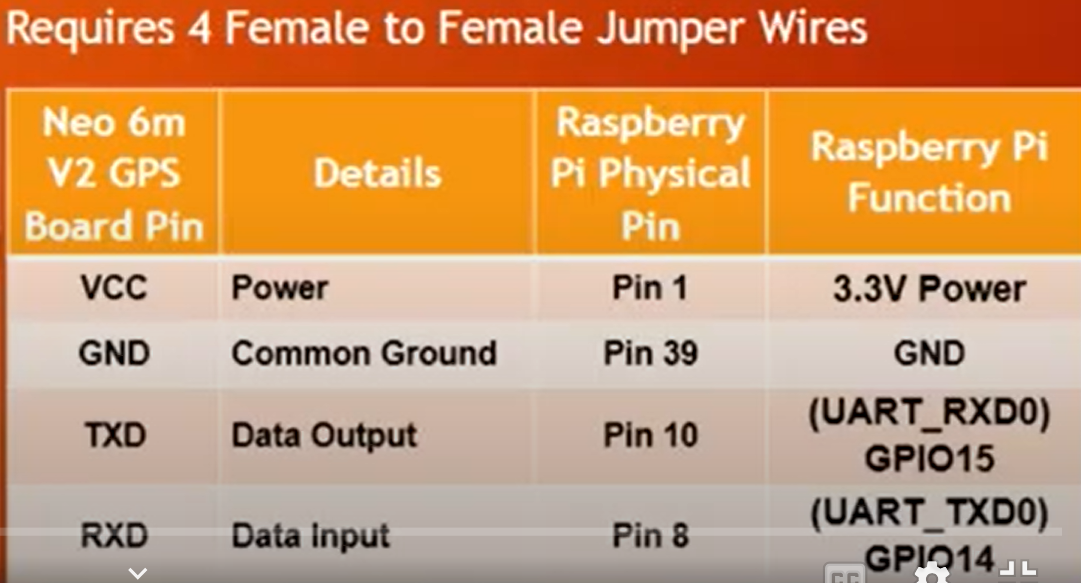
**A Brief Info. About UBlox NEO-M8N GPS Module** 

This is an UBlox NEO-M8N GPS Module with Ceramic Active Antenna. This GPS Module has a 72-channel Ublox M8 engine in the receiver. The module has 4 pins: VCC (Supply Voltage), GND (Ground), Tx (Transmitter), and Rx (Receiver).

This module provides nonstop [NMEA](https://en.wikipedia.org/wiki/National_Marine_Electronics_Association) (National Marine Electronics Association) data strings to the TX pin resulting in GPS information.

For interfacing, make the connections as follows:

1. Connect Vcc of GPS module to Power Supply Pin No.2 (5V) of Raspberry Pi.
2. Connect Tx (Transmitter Pin) of GPS module to Pin No.10 of Raspberry Pi.
3. Connect GND (Ground Pin) of GPS module to Pin No.6 Raspberry Pi.



## Step 1:Set Up the UART in Raspberry Pi

To do this edit the /boot/config.txt file. For this, run the commands below on terminal:

**sudo nano /boot/config.txt**

At the bottom of the config.txt file, add the following lines

**dtparam=spi=on**

**dtoverlay=pi3-disable-bt**

**core\_freq=250**

**enable\_uart=1**

**force\_turbo=1**

ctrl+x to exit and press y and enter to save.

**The second step under this UART setup section is to edit the boot/cmdline.txt**

I will suggest you make a copy of the cmdline.txt and save first before editing so you can revert back to it later if needed. This can be done using;

**sudo cp boot/cmdline.txt boot/cmdline\_backup.txt  
sudo nano /boot.cmdline.txt**

Replace the content with;

**dwc\_otg.lpm\_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles**

Press ctrl+x to exit and press y and enter to save.

Now reboot pi to see the changes

## Step 2:Disable the Raspberry Pi Serial Getty Service

a. If in your output, Serial0 is linked with ttyAMA0, then to disable it use the below command,

**sudo systemctl stop serial-getty@ttyAMA0.service  
sudo systemctl disable serial-getty@ttyAMA0.service**

b. If in your output Serial0 is linked with ttys0, then to disable it use the below command,

**sudo systemctl stop serial-getty@ttys0.service  
sudo systemctl disable serial-getty@ttys0.service**

## Step 3:Activate Ttys0

To enable the ttyso use following command,

**sudo systemctl enable serial-getty@ttys0.service**

## Step 4:Install Minicom and Pynmea2

Use minicom python library to connect with the GPS module and make sense of the data.

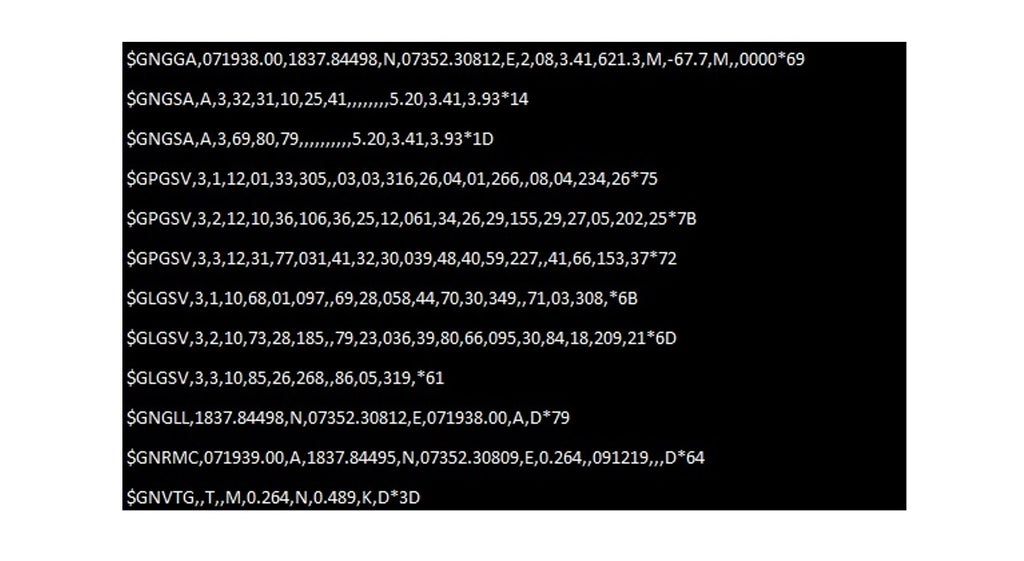
**sudo apt-get install minicom**

Use pynmea2 python library to parse the received NMEA data.

**sudo pip install pynmea2**

## Step 5:Test Output

To test the GPS run the command **sudo cat /dev/ttyAMA0,**You'll get the output as shown below.



## Step 6:Write Python Code

**Now, write the python code for the interfacing of the GPS module with Raspberry pi.**

import time

import serial

import string

import pynmea2

import RPi.GPIO as gpio

gpio.setmode(gpio.BCM)

port = "/dev/ttyAMA0" # the serial port to which the pi is connected.

#create a serial object

ser = serial.Serial(port, baudrate = 9600, timeout = 0.5)

while 1:

try:

data = ser.readline()

# print data

except:

print("loading")

#wait for the serial port to churn out data

if data[0:6] == '$GPGGA':

msg = pynmea2.parse(data)

print msg

time.sleep(2)