Practical No :01

Raspberry Pi 3 Model B

Bluetooth / Wifi model

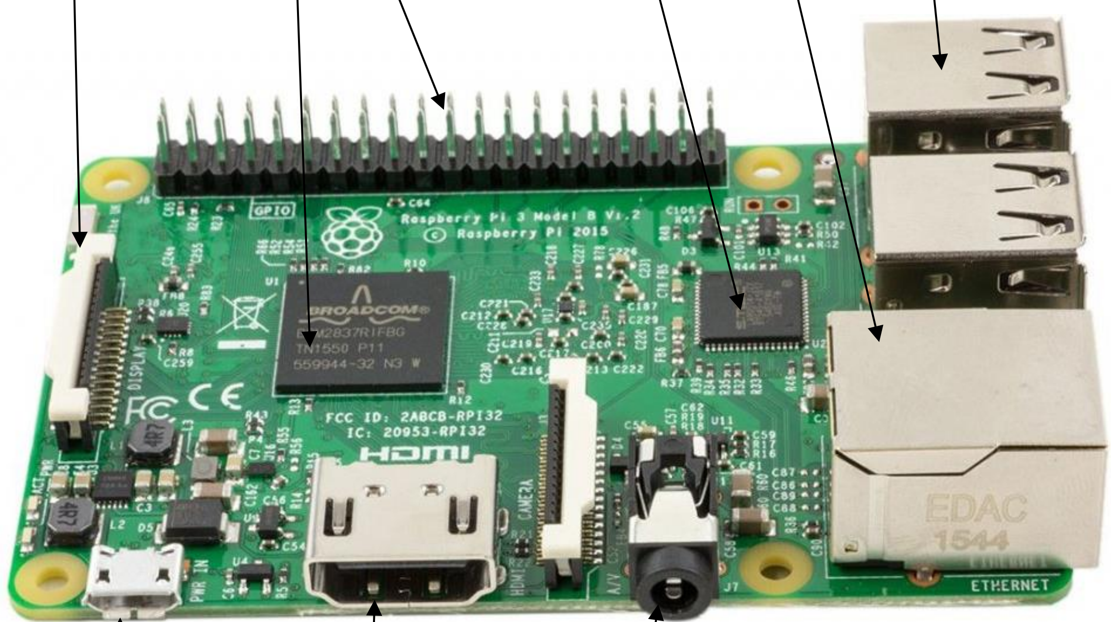
USB chip to connect processor

Broadcom Processor

Ethernet LAN port

4 USB port

GPIO 40 Pin



HDMI port

Backside:

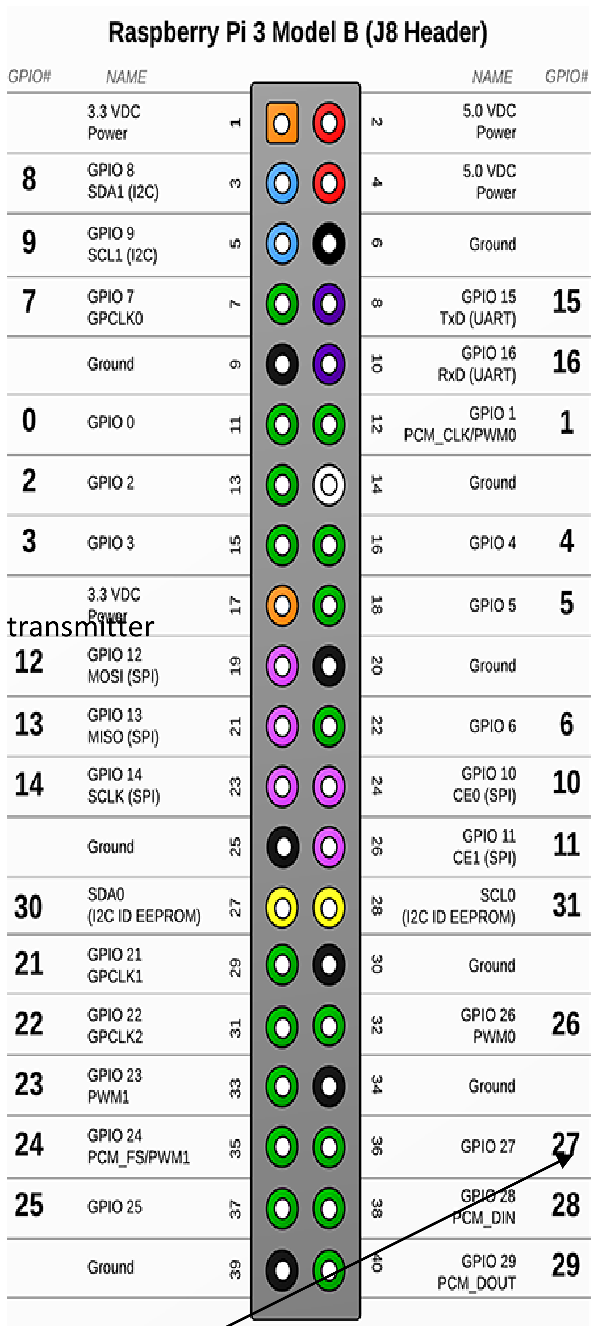
1. Micro SD card slot
2. RAM (min of 4 GB)

Audio/Video jack

Micro USB power Connector

Practical: 1

Used to connect hardware module



1. Power (3.3V,5V)

2. Ground

3. 12C (SDA,SCL)

4. UART – Universal asynchronus transmitter (Txd,RxD)

5. GPIO – Used to turn devices ON or OFF
   EG. – LED

6. SPI – used to connect device using some protocol

7. DNC – Do not connect

Reference pins

DNC

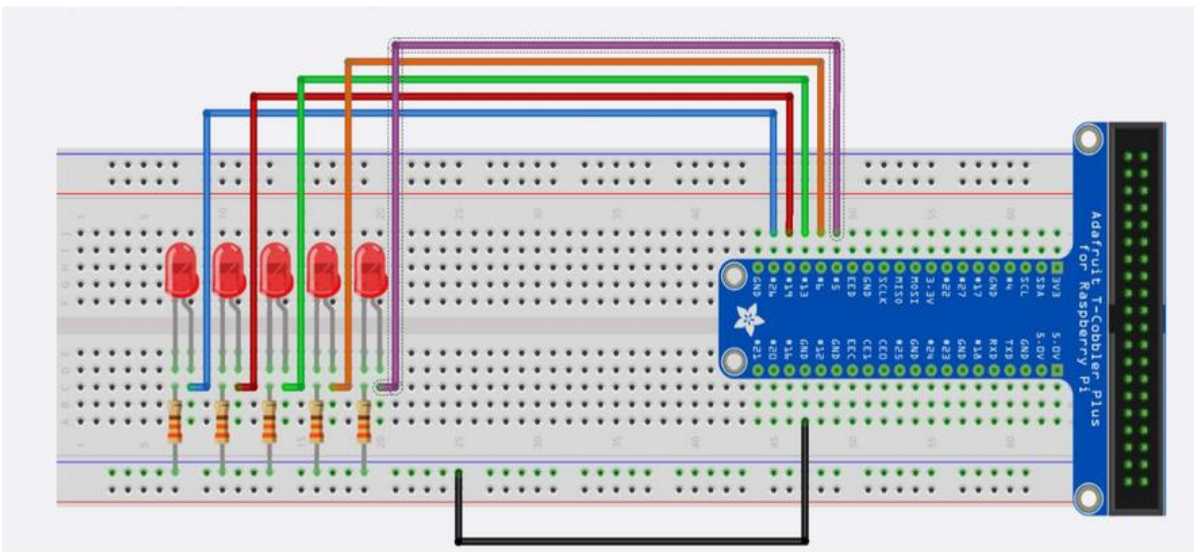
Practical No :02

Aim: Displaying different LED patterns with Raspberry Pi.

Components:

1. LED
2. 330Ω Register
3. 6 m-m jumper Wires

Connection diagram:

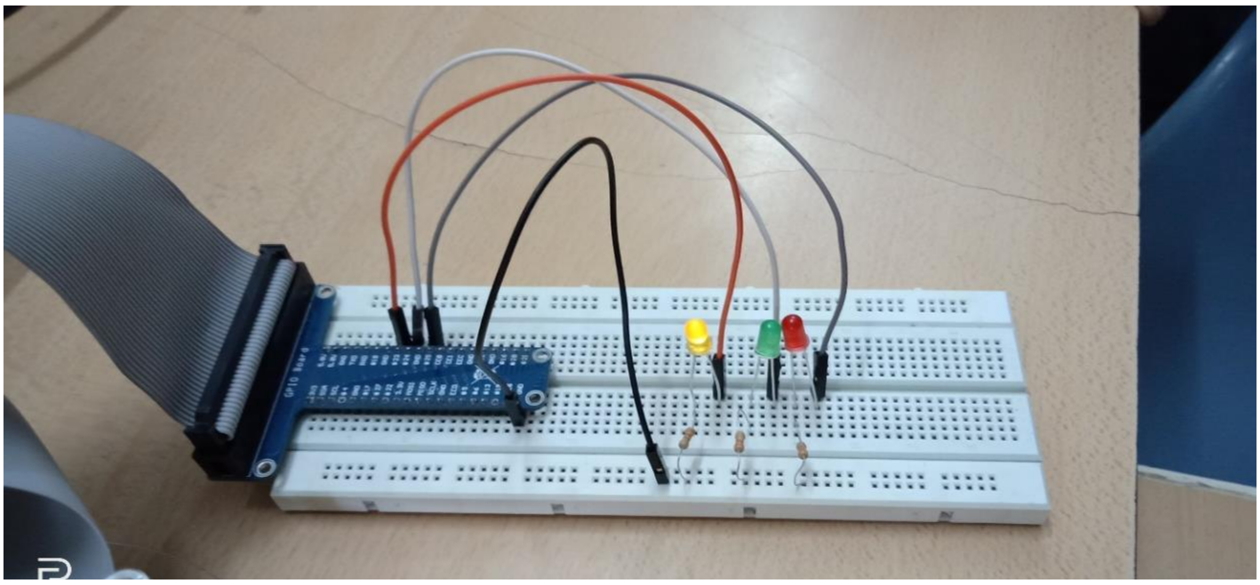


Code:

Centerdash-pattern.py

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(5,GPIO.OUT)
GPIO.setup(6,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(19,GPIO.OUT)
GPIO.setup(26,GPIO.OUT)
list=[5,6,13,19,26]
  for num in range((len(list) / 2)+1):
      GPIO.output(list[num],GPIO.HIGH)
      GPIO.output(list[len(list)-num-1],GPIO.HIGH)
      time.sleep(0.05)
      GPIO.output(list[num],GPIO.LOW)
      GPIO.output(list[len(list)-num-1],GPIO.LOW)
      time.sleep(0.05)
      GPIO.cleanup()
```

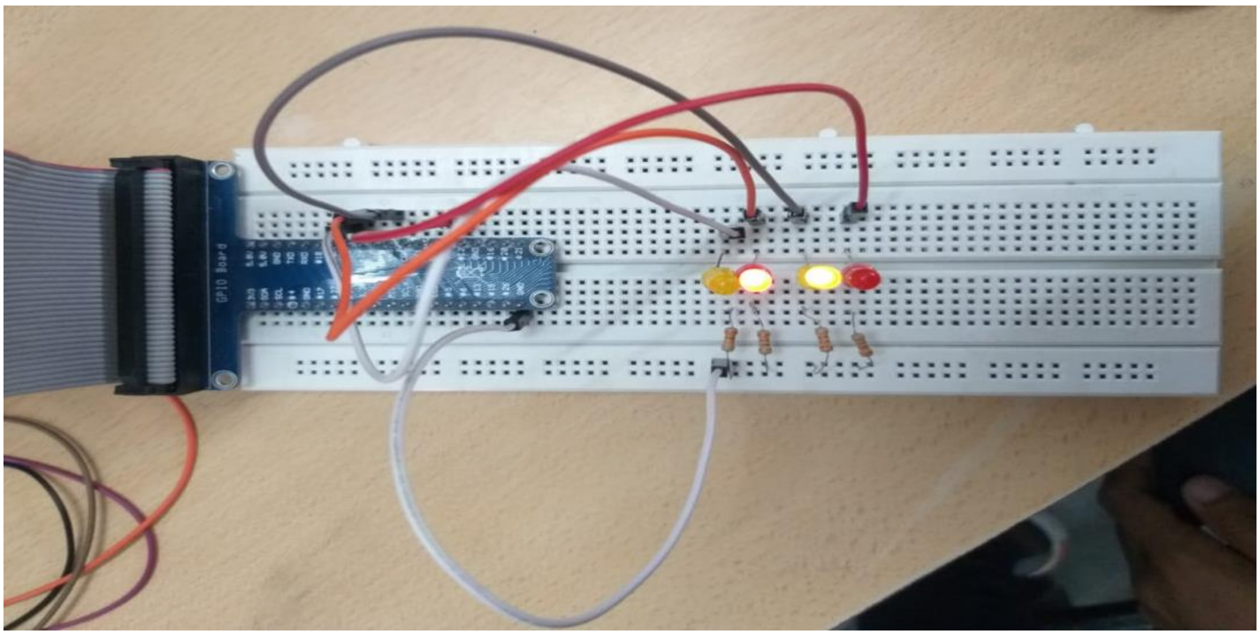
OUTPUT



Running-pattern.py

```
import RPi.GPIO as GPIO
import time
import keyboard

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(5,GPIO.OUT)
GPIO.setup(6,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(19,GPIO.OUT)
GPIO.setup(26,GPIO.OUT)
list=[5,6,13,19,26]
print "Press q to Exit"
try:
    while True:
        if keyboard.is_pressed('q'):
          break;
        for num in range(len(list)):
            GPIO.output(list[num],GPIO.HIGH)
         time.sleep(0.05)
         GPIO.output(list[num],GPIO.LOW)
         time.sleep(0.05)
except KeyboardInterrupt:
    pass
GPIO.output(5,GPIO.LOW)
GPIO.output(6,GPIO.LOW)
GPIO.output(13,GPIO.LOW)
GPIO.output(19,GPIO.LOW)
GPIO.output(26,GPIO.LOW)
GPIO.cleanup()
```

OUTPUT:



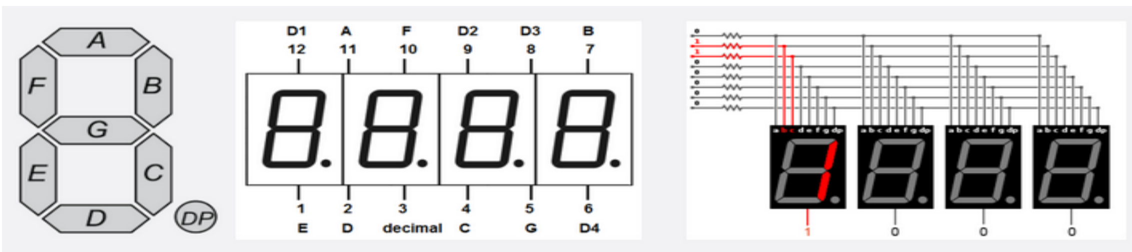
Practical No :03

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

Components:

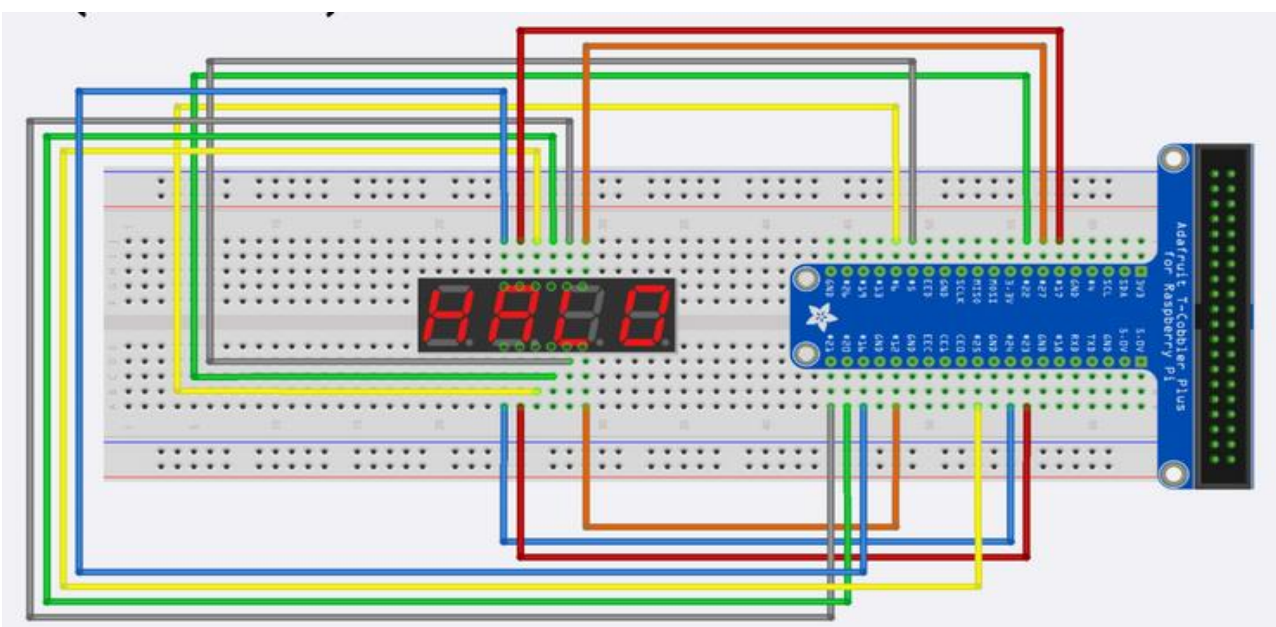
1. 16Bit ADS1115
2. 12 M-M Jumper Wires

Display Description:



Above picture is pretty much self explanatory. Display model included in the kit has 12 pins so as simple as it can be. 8 pins are attached to 7 segments of all digits and single decimal point, other 4 are used as HIGH for every digit. If you connect pin 12 to HIGH, the first digit will activate (9 = second, 8 = third, 6 =fourth). So if we had 12, 9, 8 & 6 all connected to HIGH, and 7 & 4 LOW, all four digits would display the number 1. This HIGH LOW scheme will be exactly opposite of this for Common Cathode displays.

Connection Diagram:

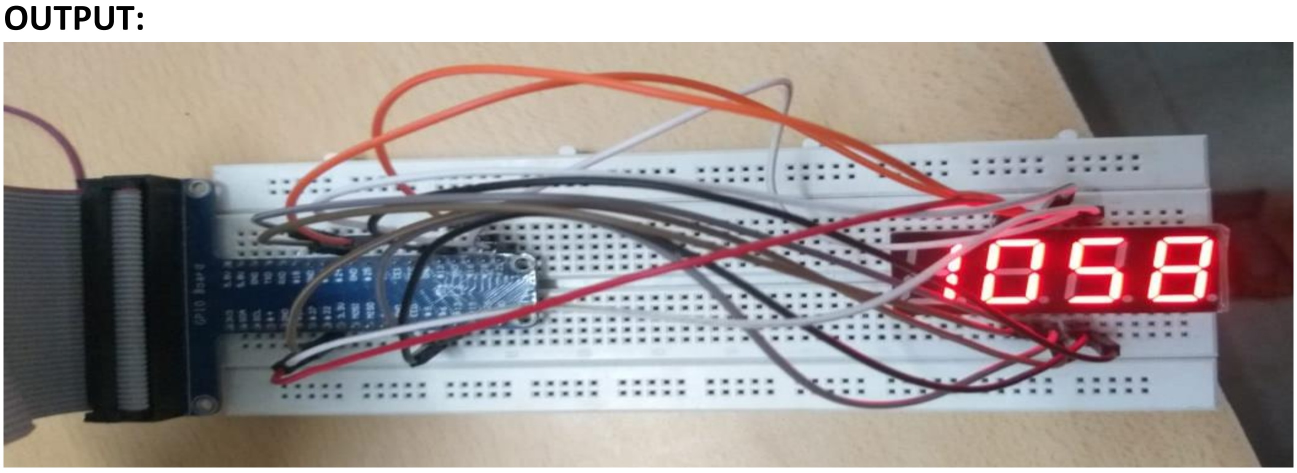


Code:

Seg.py

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
segments = (17,27,22,23,24,25,5,6)
```

```
for segment in segments:
    GPIO.setup(segment, GPIO.OUT)
    GPIO.output(segment, 1)
digits = (16,20,21,12)

for digit in digits:
    GPIO.setup(digit, GPIO.OUT)
    GPIO.output(digit, 1)

num = {' ':(0,0,0,0,0,0,0),
    '0':(1,1,1,1,1,1,0),
    '1':(0,1,1,0,0,0,0),
    '2':(1,1,0,1,1,0,1),
    '3':(1,1,1,1,0,0,1),
    '4':(0,1,1,0,0,1,1),
    '5':(1,0,1,1,0,1,1),
    '6':(1,0,1,1,1,1,1),
    '7':(1,1,1,0,0,0,0),
    '8':(1,1,1,1,1,1,1),
    '9':(1,1,1,1,0,1,1)}

try:
    while True:
        n = time.ctime()[11:13]+time.ctime()[14:16]
        s = str(n).rjust(4)
        for digit in range(4):
            for loop in range(0,7):
                if num[s[digit]][loop] == 0:
                    GPIO.output(segments[loop], 1)
                else:
                    GPIO.output(segments[loop], 0)
                if (int(time.ctime()[18:19])%2 == 0)
and (digit == 1):
                    GPIO.output(6, 0)
                else:
                    GPIO.output(6, 1)

            GPIO.output(digits[digit], 1)
            time.sleep(0.001)
            GPIO.output(digits[digit], 0)

except KeyboardInterrupt:
    GPIO.cleanup()
```
OUTPUT:



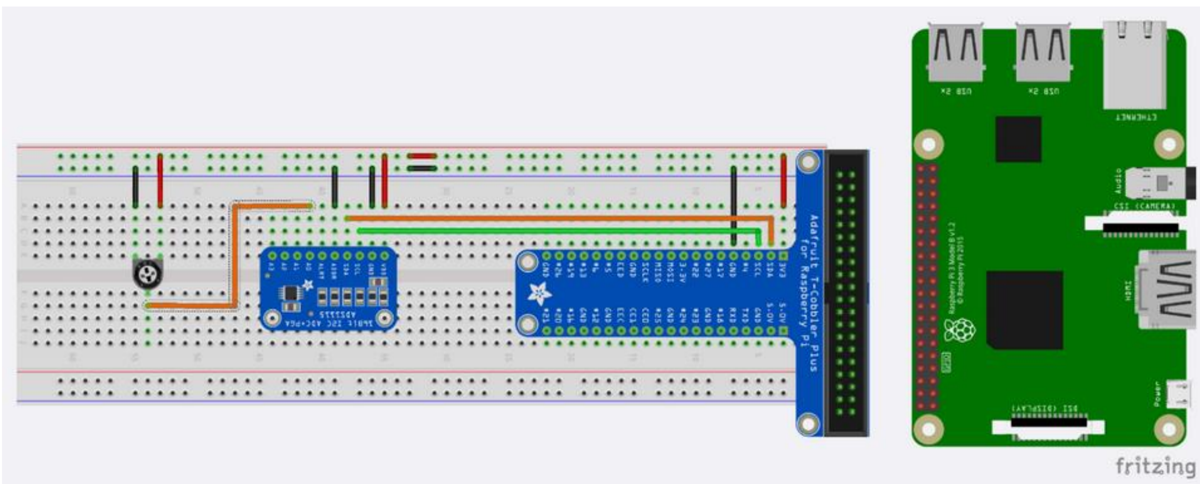
Practical No 4

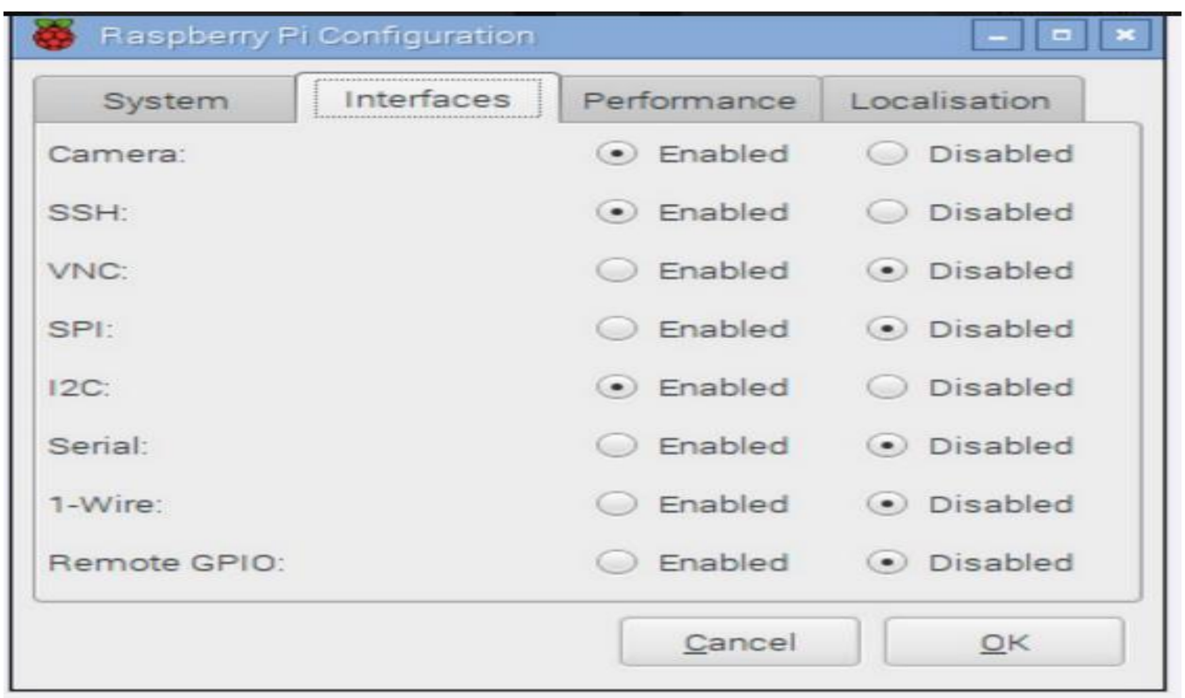
AIM: Raspberry Pi Based Oscilloscope.

Component List:

1. 10K Potentiometer
2. 16Bit ADS1115
3. Jumper Wire

Connection diagram (BCM Pins):

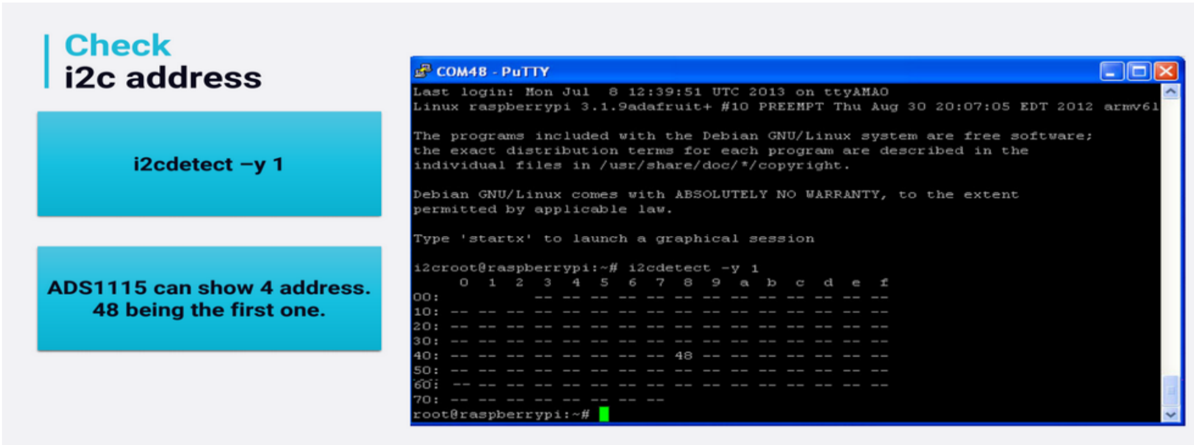




Install dependencies:

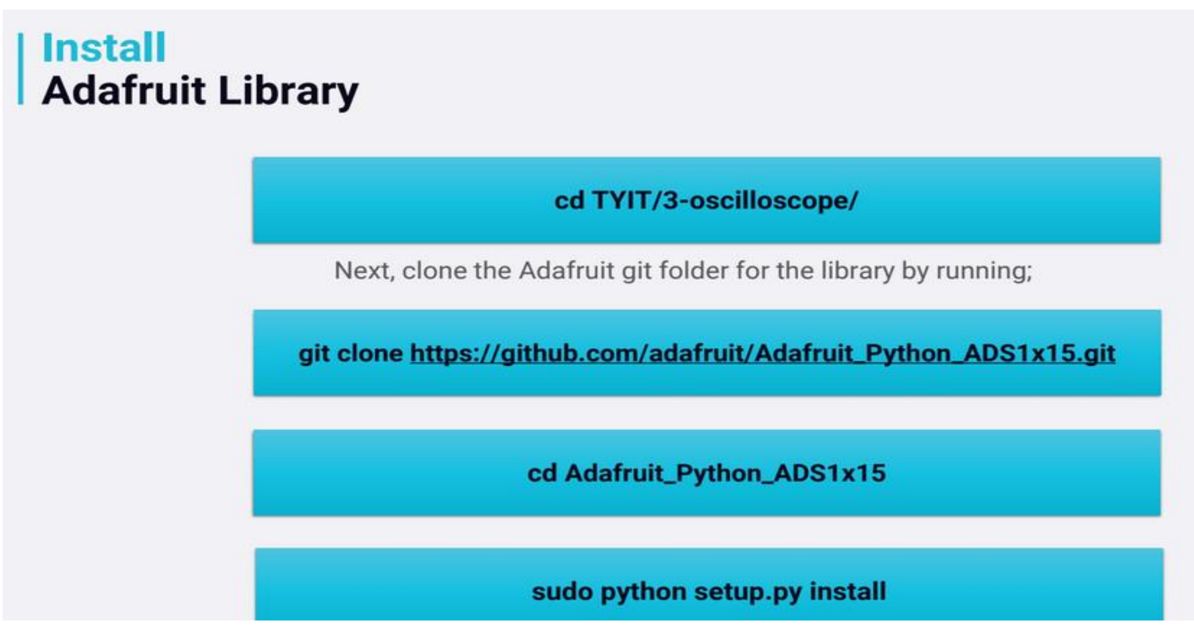
1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt-get install build-essential python-dev python-smbus git
4. sudo apt-get install i2c-tools

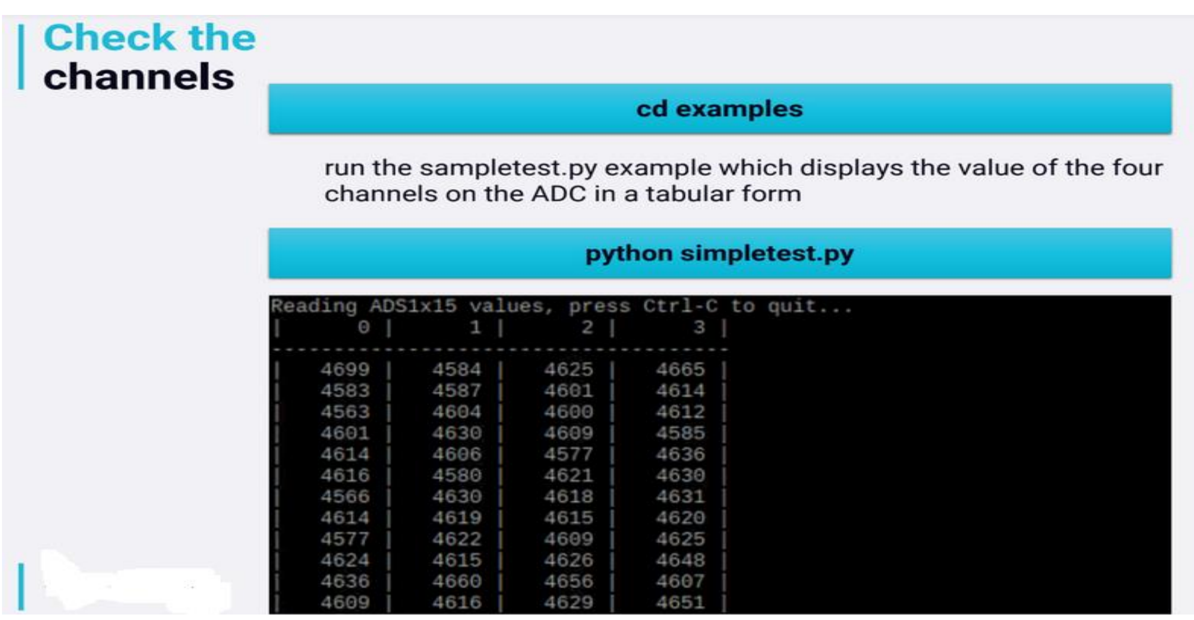
Check i2c Address



Install Adafruit Library:

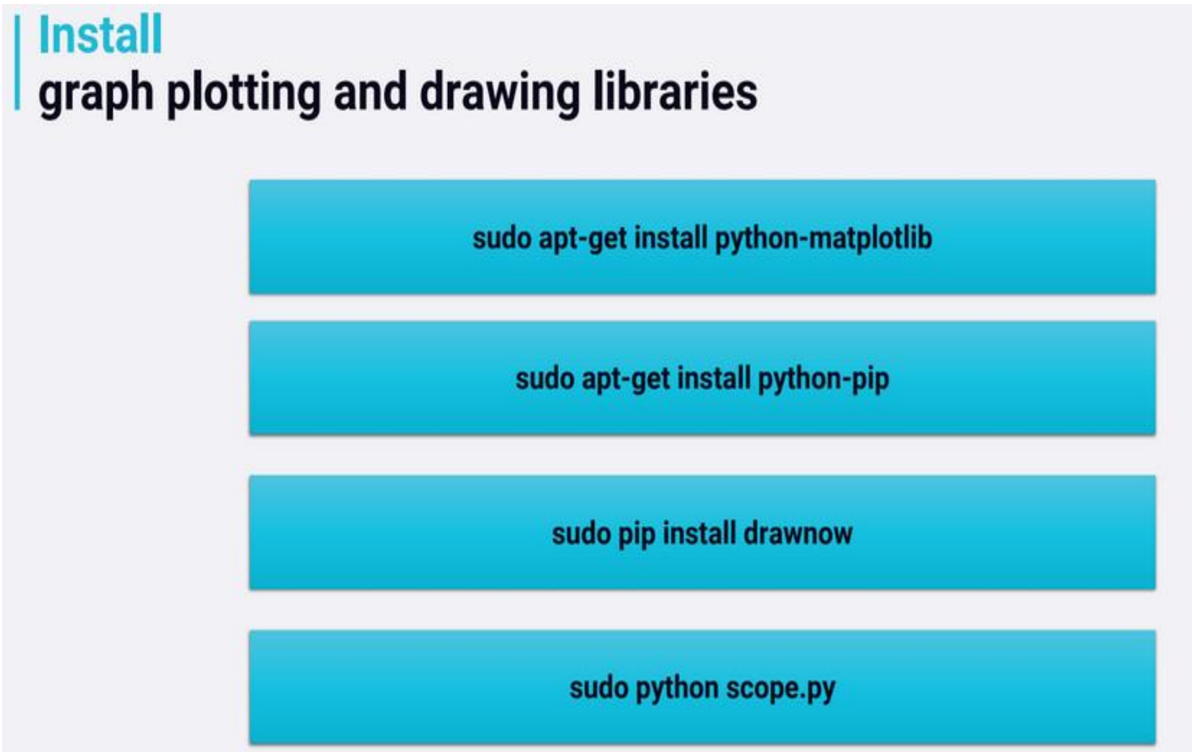
- cd TYIT/3-oscilloscope/
- Next, clone the Adafruit git folder for the library by running.
- Git clone https://github.com/adafruit/Adafruit_Python_ADS1x15
- Cd Adafruit_Python_ADS1x15
- Sudo python setup.py install





Install graph plotting and drawing libraries

- sudo apt-get install python-matplotlib
- sudo apt-get install python-pip
- sudo pip install drawnow
- sudo python scope.py

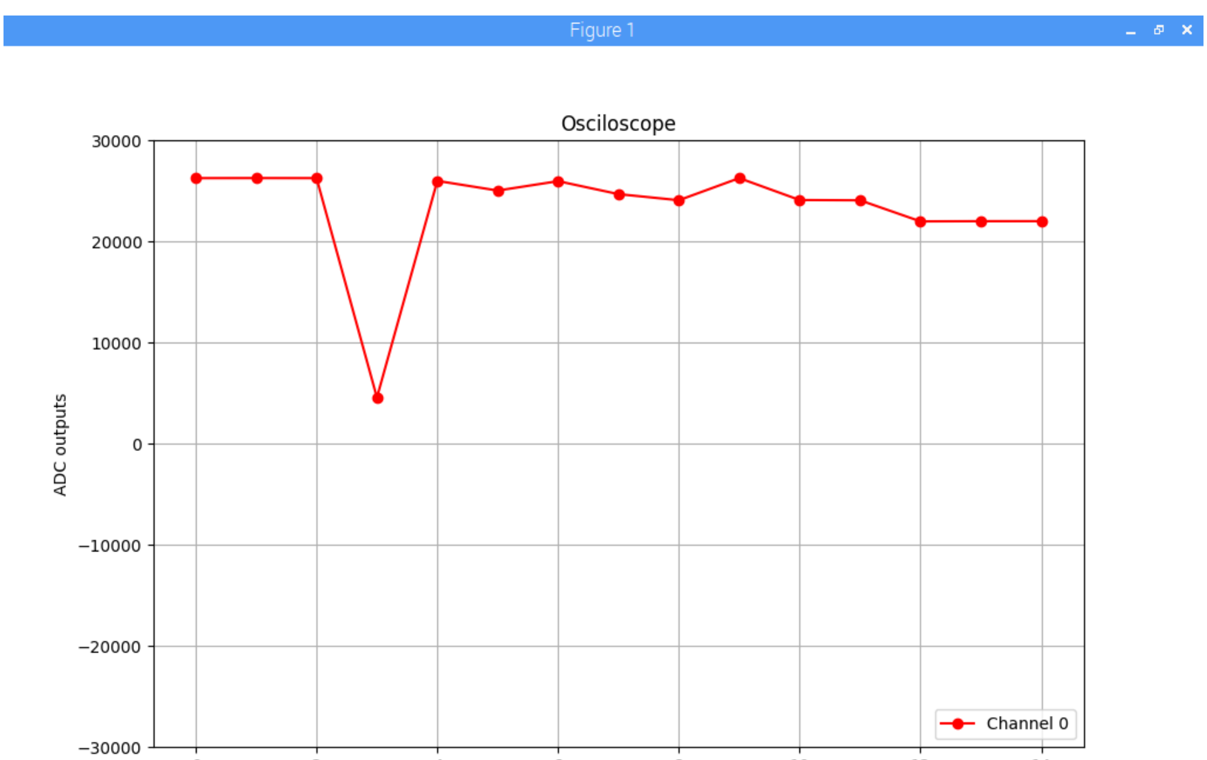


Code:

```
import time
import matplotlib.pyplot as plt
from drawnow import*
import Adafruit_ADS1x15

adc=Adafruit_ADS1x15.ADS1115()
GAIN=1
val=[]
cnt=0
plt.ion()

adc.start_adc(0, gain=GAIN)
print('Reading ADS1x15 Channel 0')

def makeFig():
   plt.ylim(-5000,5000)
   plt.title('Oscilloscope')
```

```
    plt.grid(True)
    plt.ylabel('ADC Ost_resultutputs')
    plt.plot(val, 'ro-', label='Channel 0')
    plt.legend(loc='lower right')

while (True):
    value=adc.get_last_result()
    print('channel 0: {0}'.format(value))

    time.sleep(0.5)
    val.append(int(value))
    drawnow(makeFig)
    plt.pause(0.000001)
    cnt=cnt+1
    if(cnt>50):
        val.pop(0)
```

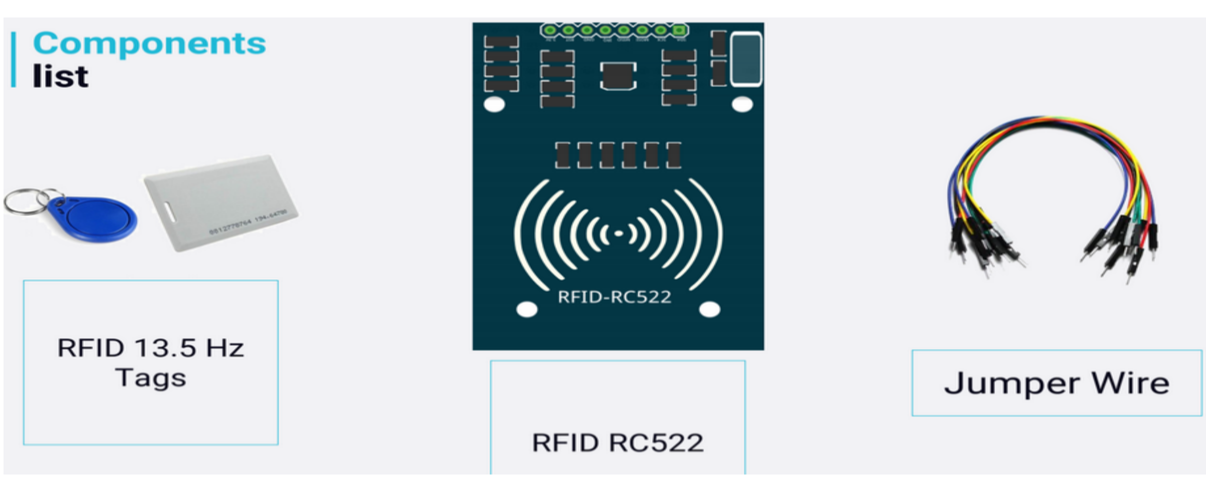
OUTPUT:



Practical No 5

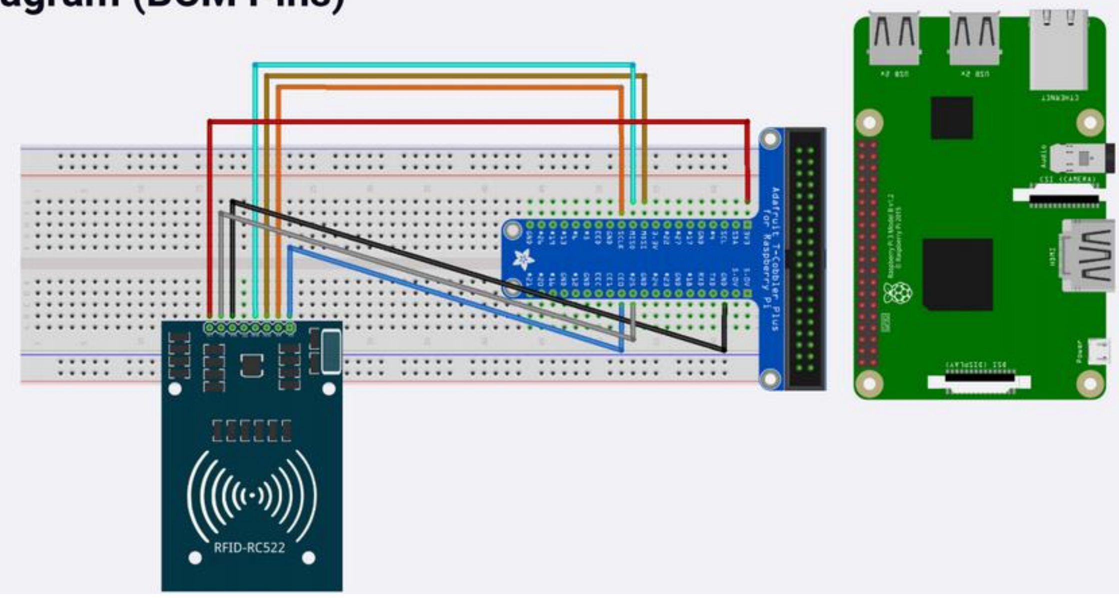
AIM: Interfacing Raspberry Pi with RFID.

Component List:



Connection Diagram:

Connection diagram (BCM Pins)

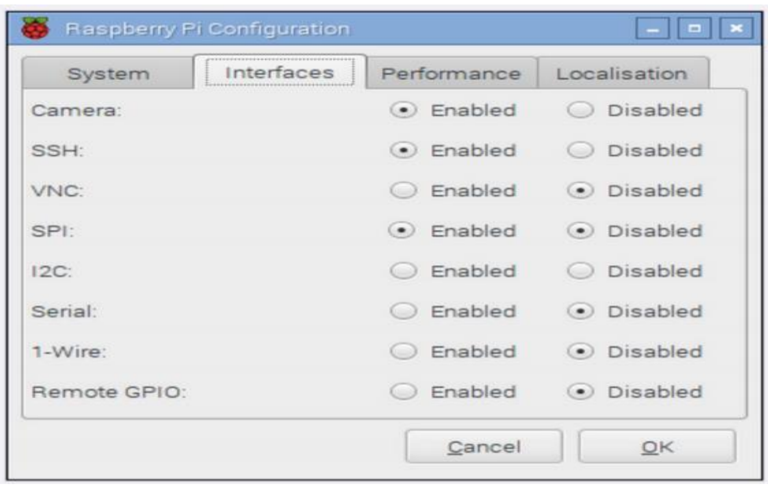


Enable spi

Start Menu -> Preferences -> Raspberry Pi Configuration

Select interfaces Tab
Select SPI Enable
Click OK
Reboot

After Reboot
lsmod | grep spi
It should list spi device as spi_bcm2835



Install SPI py Library

```
cd TYIT/10-RFID/
```

Next, clone the SPI py git folder for the library by running;

```
git clone https://github.com/lthiery/SPI-Py.git
```

```
cd SPI-Py
```

```
sudo python setup.py install
```

Install
Pimylifeup MFRC522 Library

```
cd ../
```

Next, clone the MFRC522 git folder for the library by running;

```
git clone https://github.com/pimylifeup/MFRC522-python.git
```

```
cd MFRC522-python
```

Code
write.py

```
sudo nano write.py
```

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

try:
        text = raw_input('New data:')
        print("Now place your tag to write")
        reader.write(text)
        print("Written")
finally:
        GPIO.cleanup()
```

```
sudo python write.py
```

output
write.py

You will be asked to write in the new data, in our case we are going to just type in **New card001** as its short and simple. Press **Enter** when you are happy with what you have written.

With that done, simply place your RFID Tag on top of your RFID RC522 circuit. As soon as it detects it, it will immediately write the new data to the tag. You should see "**Written**" appear in your command line if it was successful.

You can look at our example output below to see what a successful run looks like.

```
pi@raspberrypi:TYIT/10-RFID/MFRC522-python $ sudo python Write.py
New data:New Card001
Now place your tag to write
Written
```

Code
read.py

sudo nano read.py

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

try:
        id, text = reader.read()
        print(id)
        print(text)
finally:
        GPIO.cleanup()
```

sudo python read.py

output
read.py

With the script now running, all you need to do is place your RFID Tag on top of your RFID RC522 circuit. As soon as the Python script detects the RFID tag being placed on top, it will immediately read the data and print it back out to you.

An example of what a successful output would look like is displayed below.

```
pi@raspberrypi:TYIT/10-RFID/MFRC522-python $ sudo python read.py
827843705425
New Card001
```

OUTPUT

```
pi@raspberrypi: ~/TYIT/MFRC522-python-master
File  Edit  Tabs  Help
pi@raspberrypi:~/TYIT $ cd MFRC522-python-master/
pi@raspberrypi:~/TYIT/MFRC522-python-master $ sudo python write.py
New Data:'Aniket'
Now place your tag to write
Error while writing
written
pi@raspberrypi:~/TYIT/MFRC522-python-master $ sudo python read.py
576202874863
Aniket
pi@raspberrypi:~/TYIT/MFRC522-python-master $ sudo python write.py
New Data:'twinkle'
Now place your tag to write
written
pi@raspberrypi:~/TYIT/MFRC522-python-master $ sudo python read.py
221163856778
twinkle
```

PRACTICAL NO.6

AIM: Camera Connection and capturing Images

APPARATUS: Camera Module

THEORY: The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. The camera is supported in the latest version of Raspbian, the Raspberry Pi’s preferred operating system.

The Raspberry Pi Camera Board Features:
1. Fully Compatible with Both the Model A and Model B Raspberry Pi
2. 5MP Omnivision 5647 Camera Module
3. Still Picture Resolution: 2592 x 1944
4. Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
5. 15-pin MIPI Camera Serial Interface – Plugs Directly into the Raspberry Pi Board
6. Size: 20 x 25 x 9mm
7. Weight 3g
8. Fully Compatible with many Raspberry Pi cases

PROCEDURE:

1. Locate the camera port and connect the camera:
2. Start up the Pi.
3. Open the Raspberry Pi Configuration Tool from the main menu.
4. Ensure the camera software is enabled. If it's not enabled, enable it and reboot your Pi to begin.

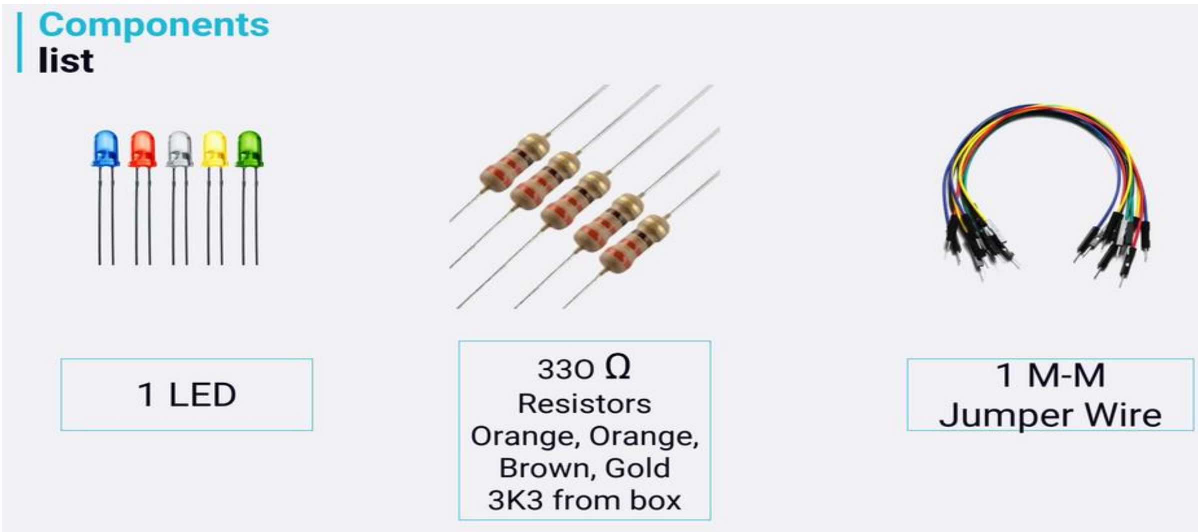
CODE:

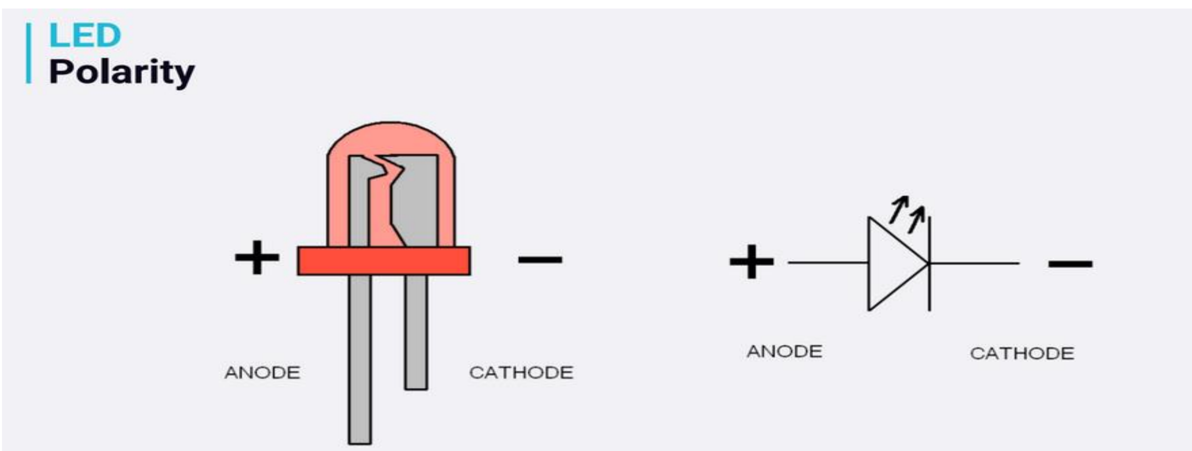
```
 from time import sleep
from picamera import PiCamera
camera = PiCamera()
camera.resolution = (1280, 720) camera.start_preview()
sleep(2)
camera.capture('/home/pi/Pictures/newImage.jpg')
camera.stop_preview()
```

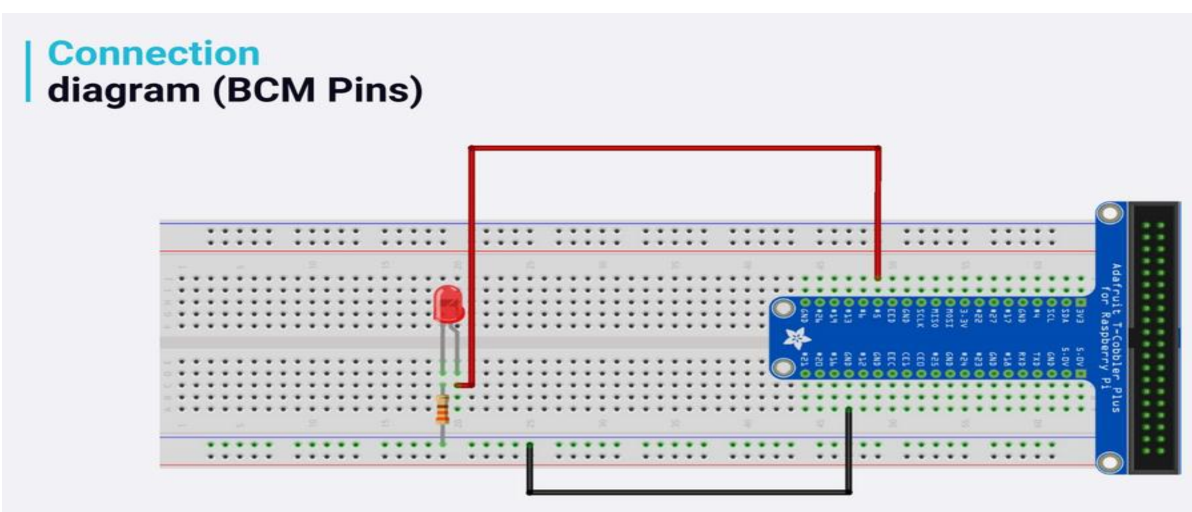
OUTPUT:

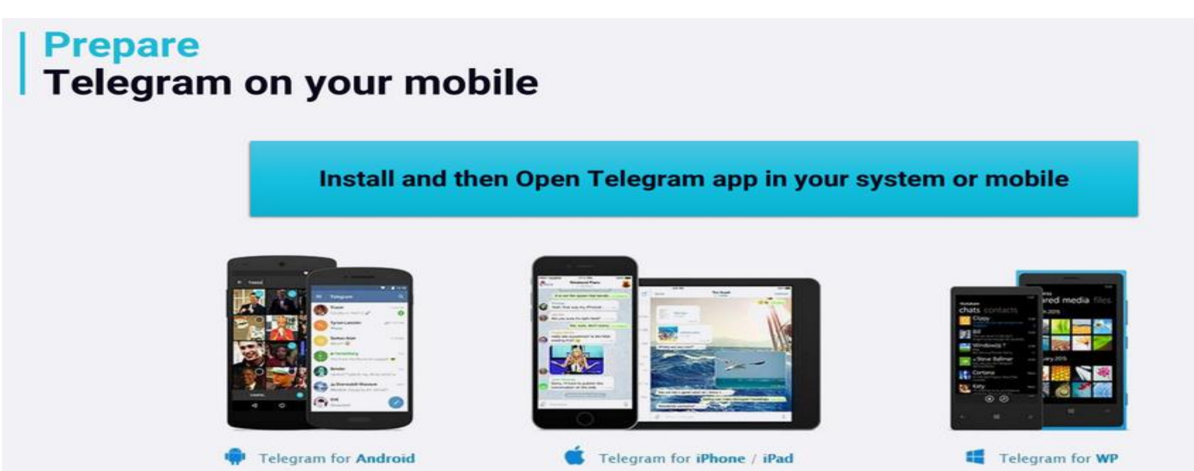
Practical No: 07

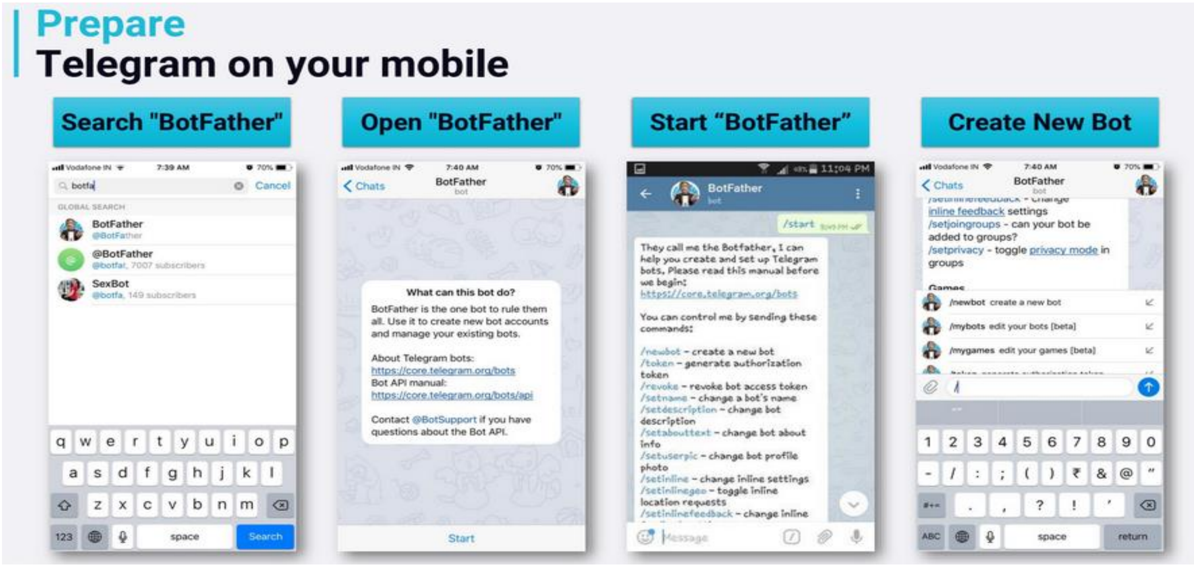
Aim: Controlling Raspberry Pi with Telegram Meesanger.

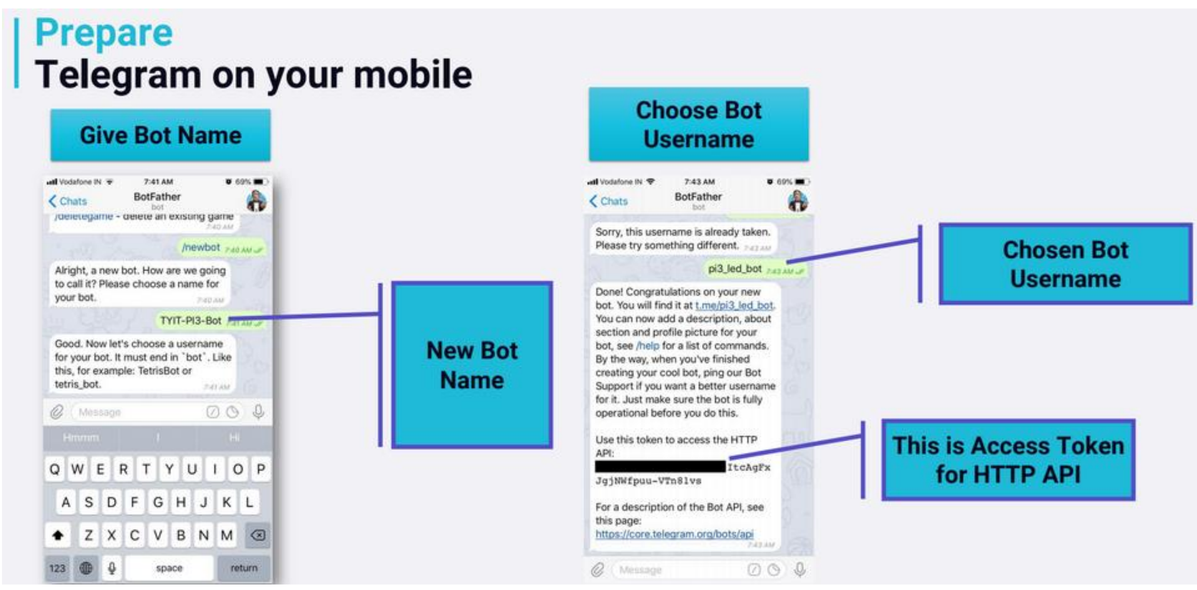


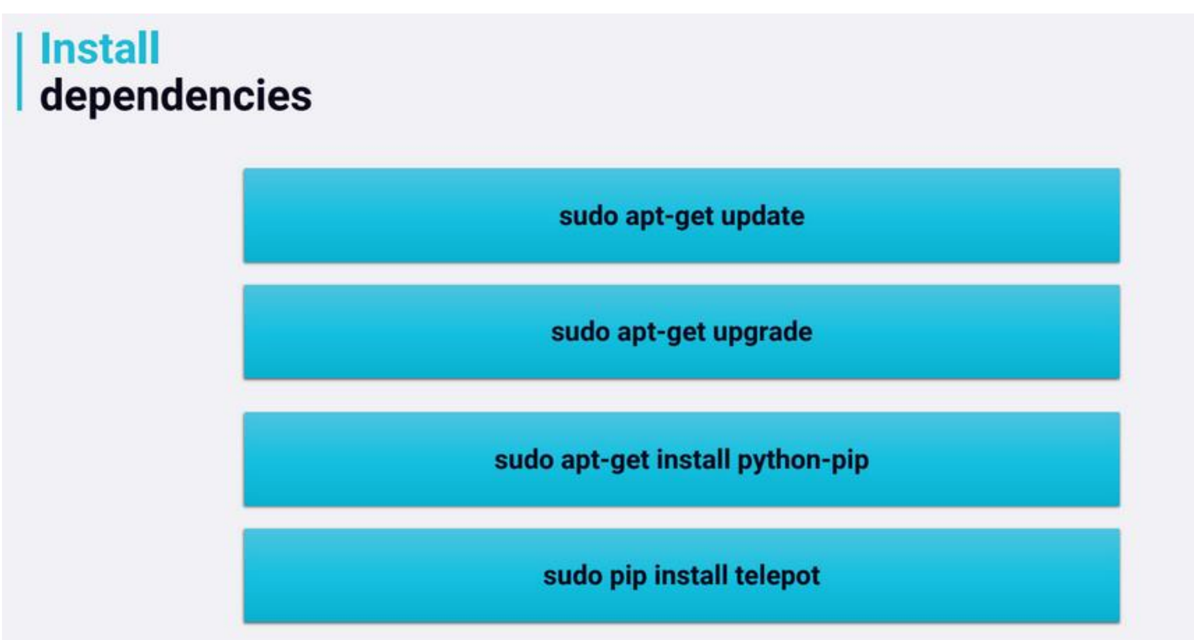










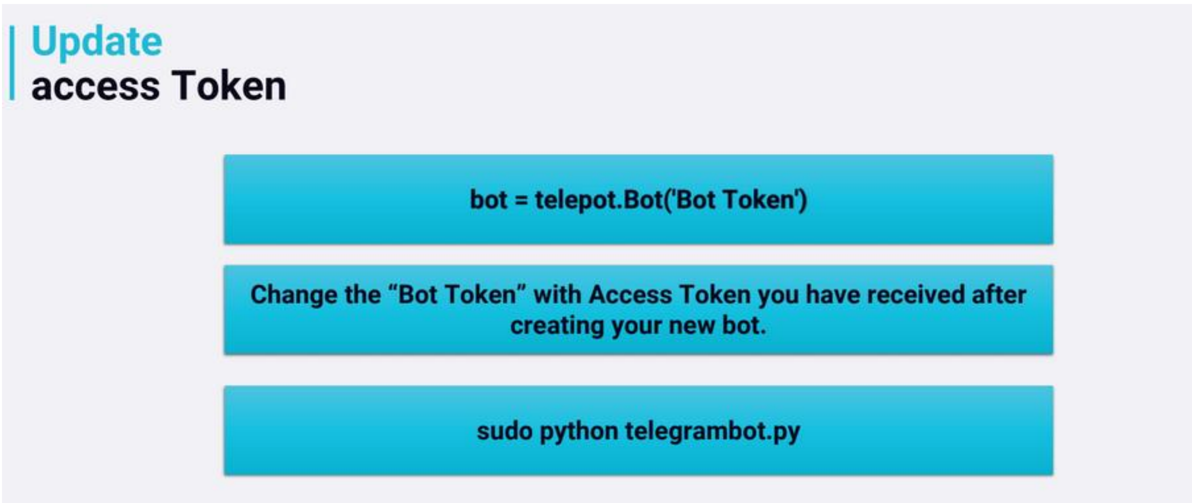


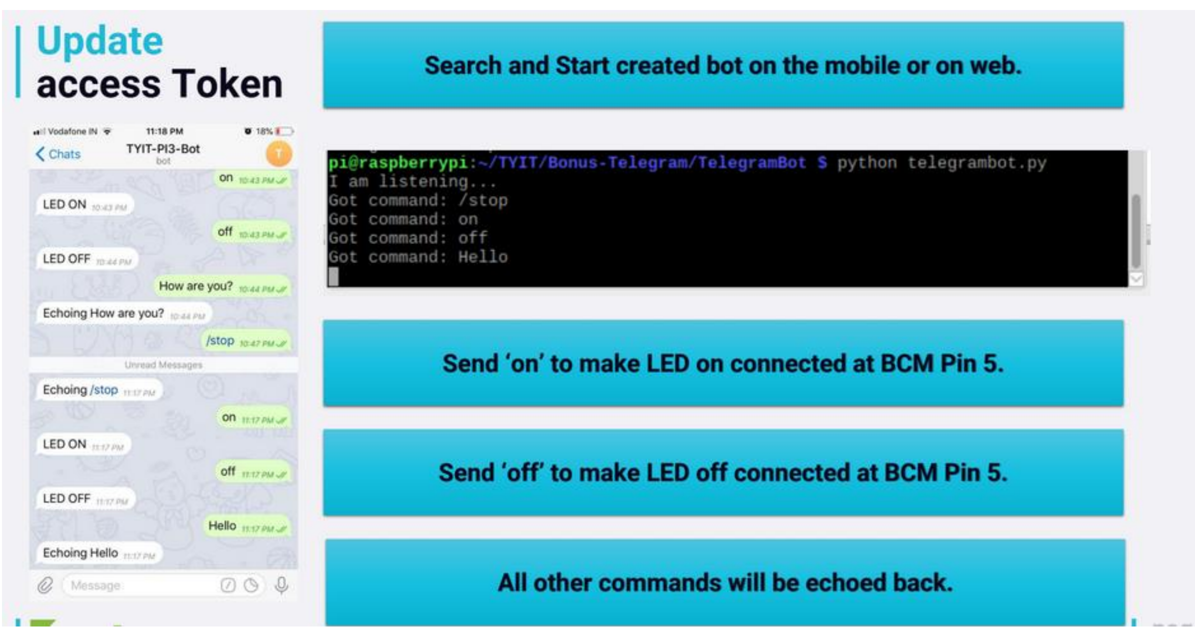
Telegram code:

```
import sys

import time
```

```
import telepot
import RPi.GPIO as GPIO

#LED
def on(pin):
   GPIO.output(pin, GPIO.HIGH)
   return 'LED is On Now on BCM pin 5'

def off(pin):
   GPIO.output(pin, GPIO.LOW)
   return 'LED is Off Now on BCM pin 5'

GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.OUT)

def handle(msg):
   chat_id = msg['chat']['id']
   command = msg['text']

   print('Got Command '+command)
   if command == 'on':
      bot.sendMessage(chat_id, on(5))
   elif command == 'off':
      bot.sendMessage(chat_id, off(5))
   else :
      bot.sendMessage(chat_id, 'Echo:'+command)

bot = telepot.Bot('970040458:AAErVBNY3qrEoOhqqg4IxpKKqpZuQFvq5kY')
bot.message_loop(handle)
print('I am listening...')

while 1:
   try:
      time.sleep(10)
   except KeyboardInterrupt:
      print('')
      print('Program Interrupted')
      GPIO.cleanup()
```

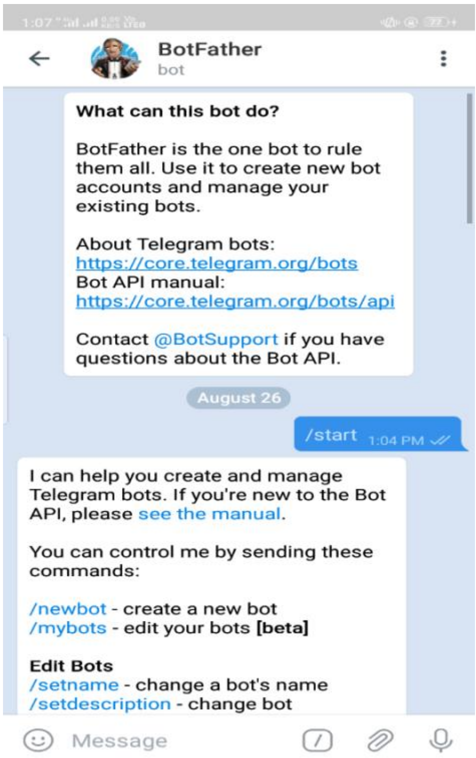
```
    exit()
  except:
    print('other error of exception occured')
    GPIO.cleanup()
```





OUTPUT:

Creating New Bot



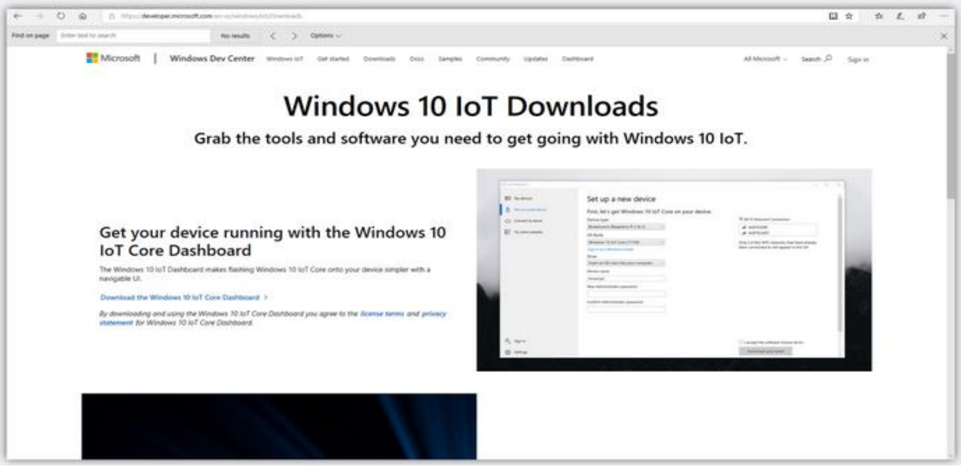
Commands given to bot:

Practical No:08

Aim: Installing Windows 10 IoT Core on Raspberry Pi.

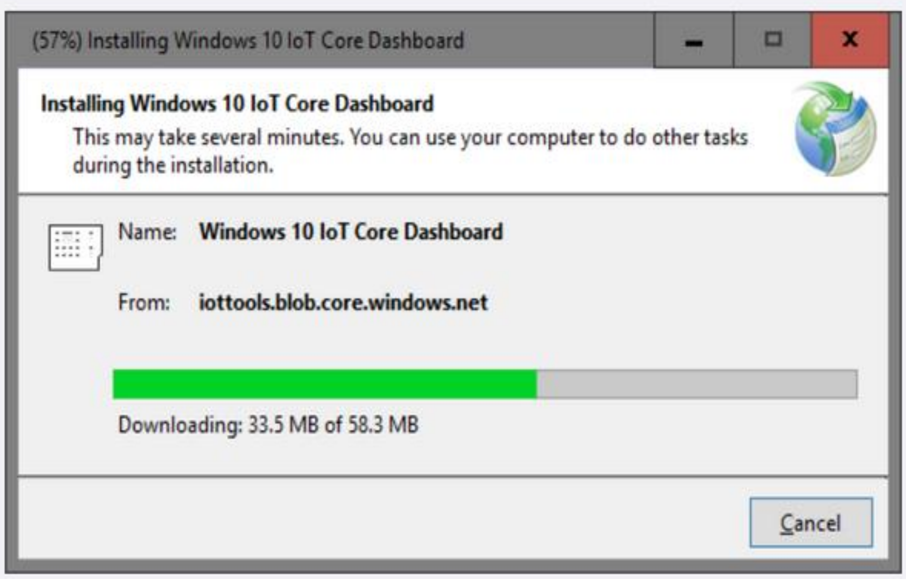
Get ready

To Prepare windows IoT Core installation the SD card is prepared on Windows 10 PC.
Visit https://developer.microsoft.com/en-us/windows/iot/Downloads
And download the windows IoT Core Dashboard.



Get ready

Run the dashboard setup and install Windows IoT Core Dashboard.



Get ready

Run the dashboard after install and select setup a new Device.

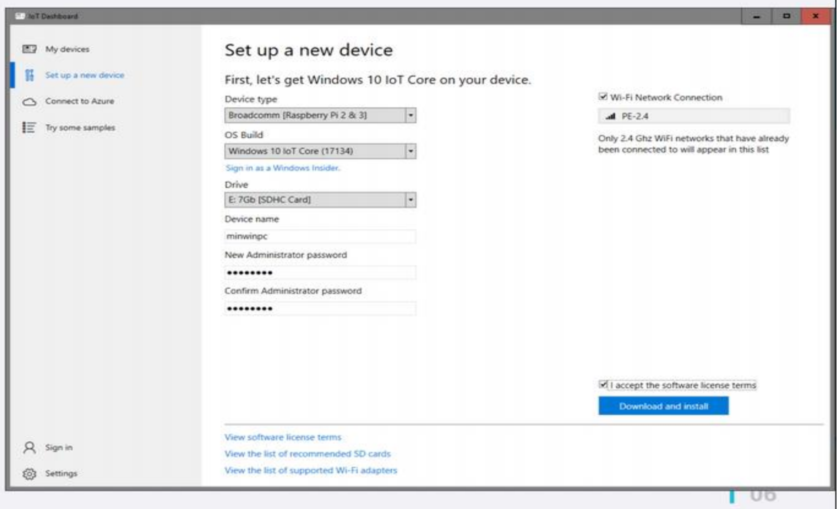
Get ready

Select the Device type as Raspberry Pi 2 & 3.

OS Build and Drive to prepare installation.

Add Administrator password.

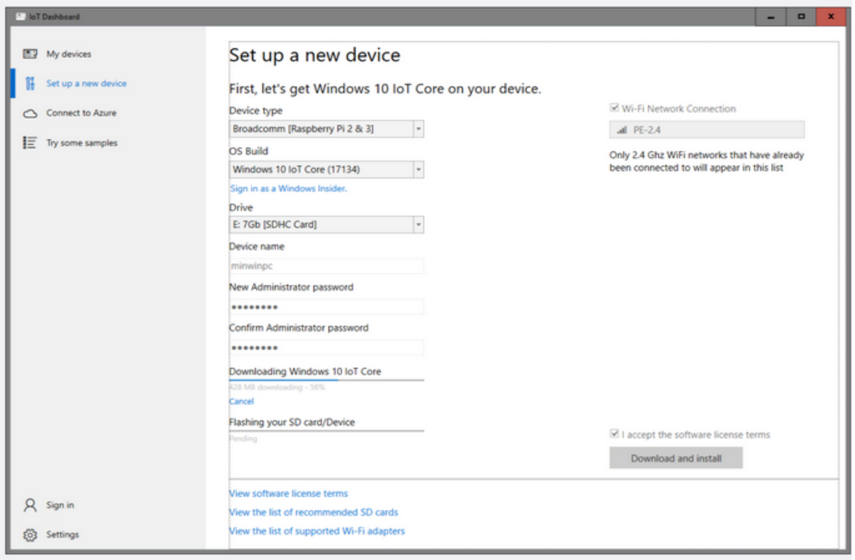
Accept the things and Click on Download and Install.



Get ready

It will download and install the Windows IoT core onto the SD Card.

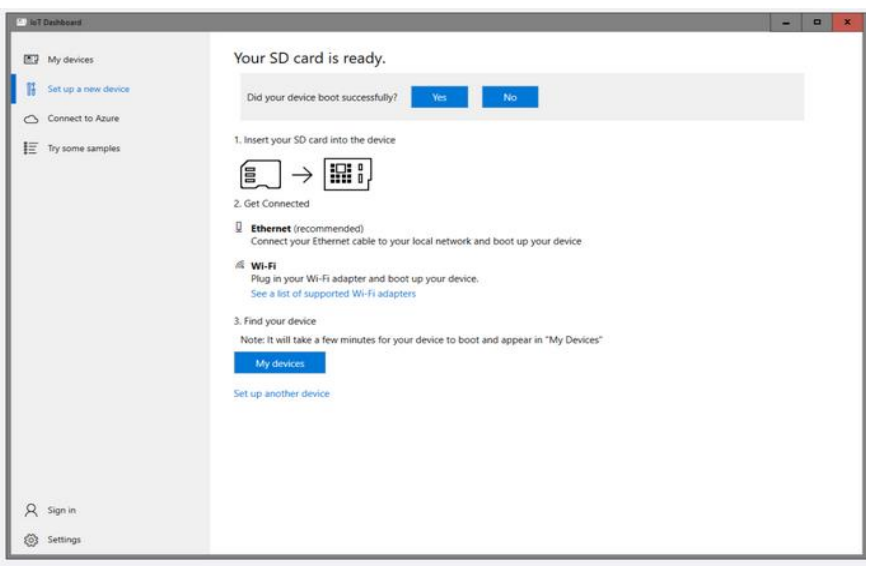
In between the UAC will asks for few permissions.



Get ready

Once card is prepared the Dashboard will ask feedback.

Remove the SD Card and put it in Raspberry Pi.



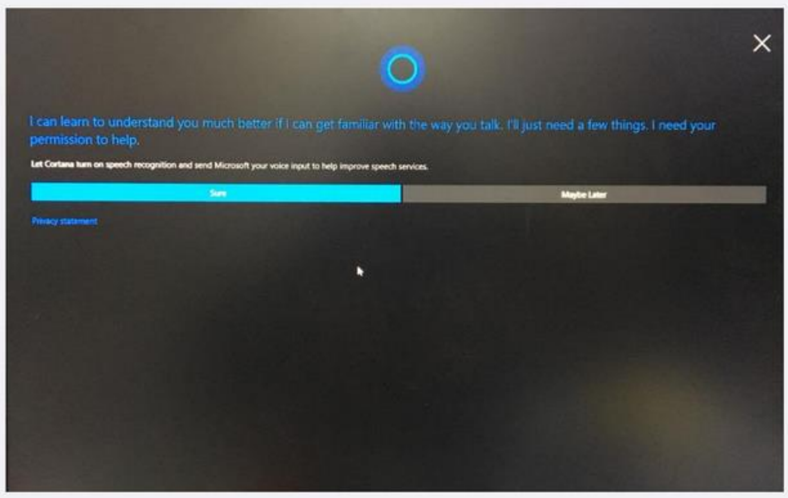
Get
ready

Once power is one, Windows IoT core will do many things in Background and for that time things will look like stopped. Don't worry. For first run it will take time and on successful loading it will show a screen to choose language.

Choose desired language and click Next.

Get
ready

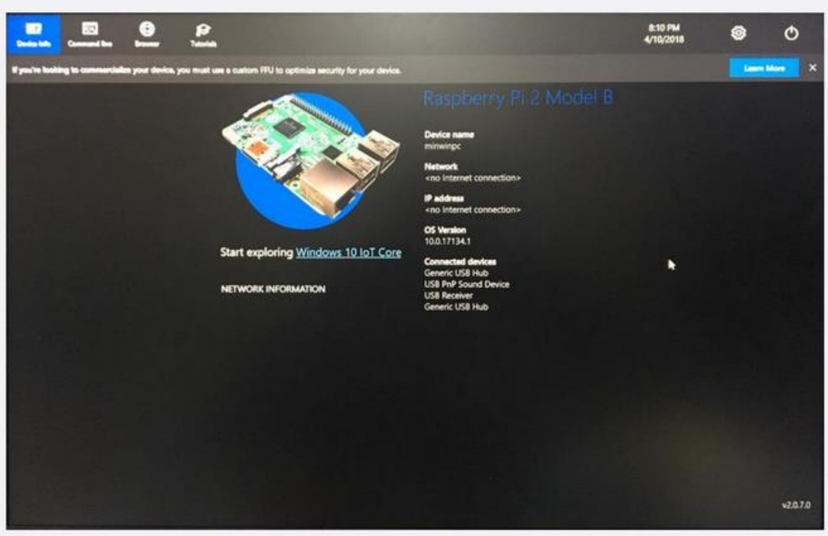
Windows IoT core will also install Cortana.



Get
ready

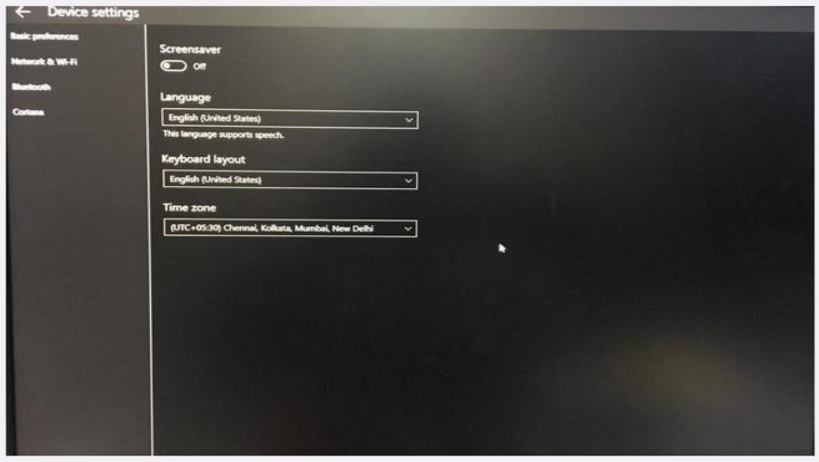
Windows IoT core will boot into desktop like this.

It shows Device Info, Provides Command Line and Browser and Tutorials.



Get ready

We can change the required settings from settings tab.

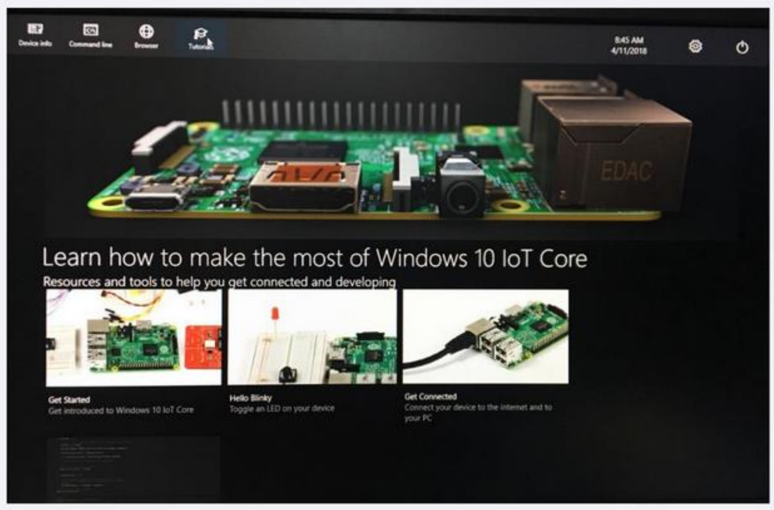


Get ready

Hello Blinky is a small installed program that blinks the Raspberry Pi LED with given ms stoppage.

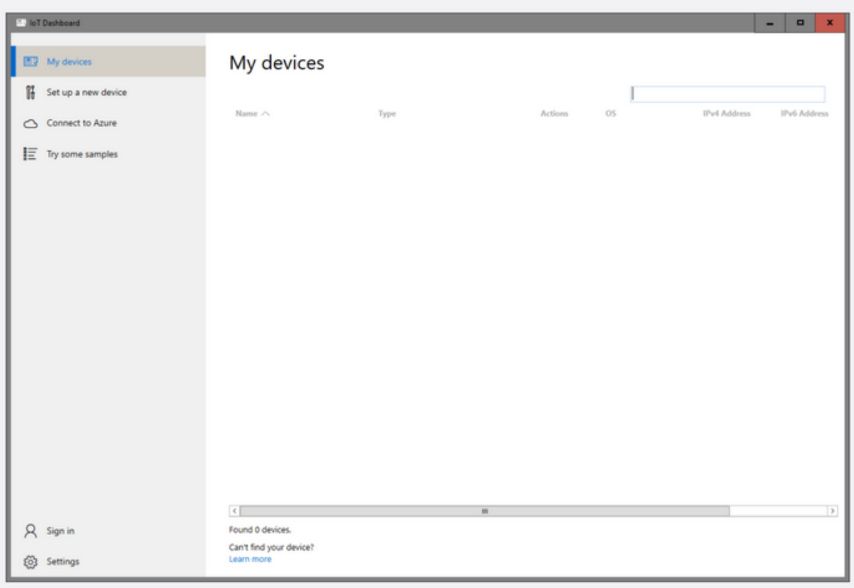
The section also provides the required tutorial section.

Take a look at it.



Get ready

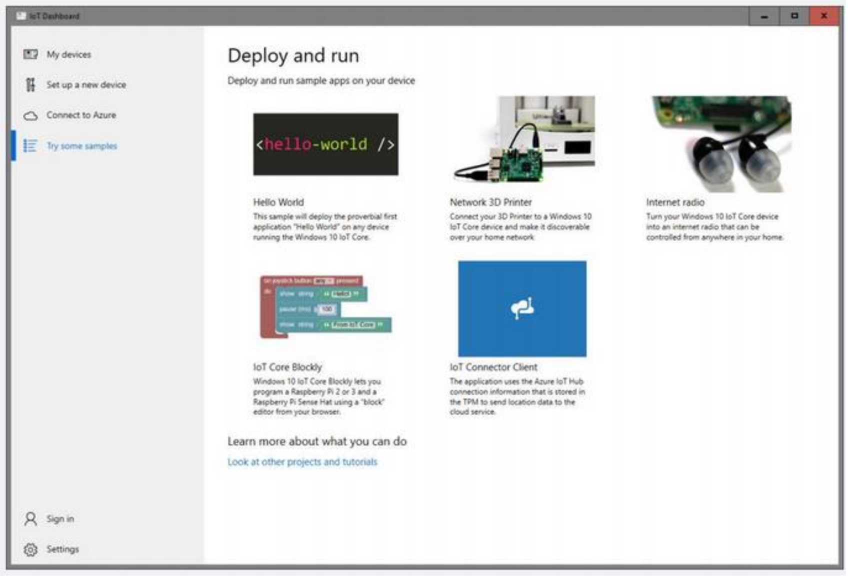
On Windows IoT Core one can login to existing mictosoft account and the device will be registered to the My Devices area. It could be checked from the IoT Core Dashboard.



Get ready

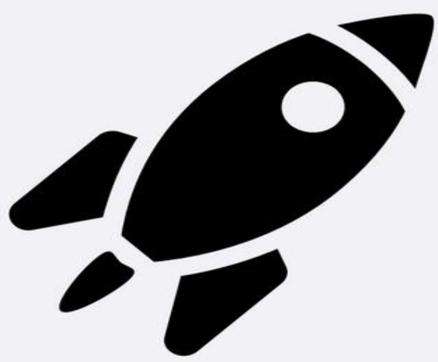
On Windows IoT Core Dashboard also provides few tutorials and deployments of programs on to the IoT devices.

Please take a look.



Running Windows IoT Core.

Hopefully, at this point, you will now have Windows IoT Core successfully set up.



Practical No 9

Aim: Building Google Assistant with Raspberry Pi.

Components list

USB Microphone

Bubble Speaker

Be ready

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Agenda Step by step

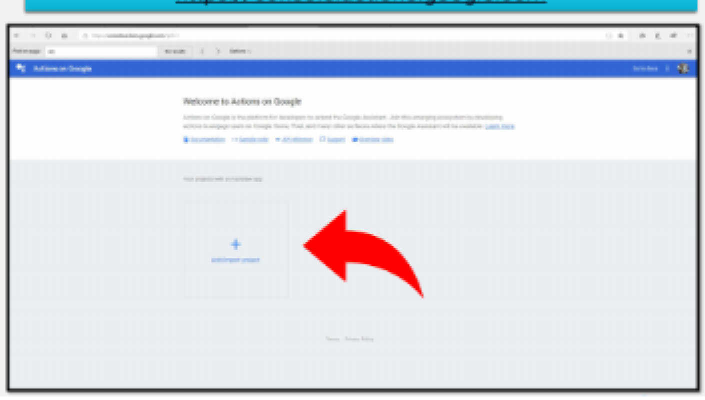
1. Registering for the Google API
2. Setting up your Audio for Google Assistant
3. Checking audio input and output hardware works
4. Downloading and setting up Google Assistant
5. Authorizing your Raspberry Pi for the Google Assistant
6. Using the Google Assistant on the Raspberry Pi
7. Starting Google Assistant API on start up

Registering for the Google API

Before we get started with setting up the Google Assistant code on the Raspberry Pi itself, we must first register and set up a project on the Google Actions Console.
With your Google account ready to go to the URL below which will take you there.

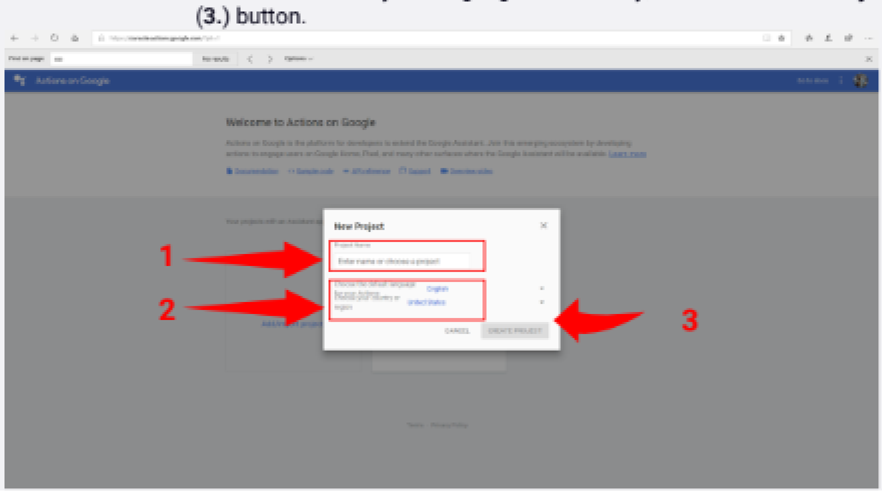
https://console.actions.google.com

Once you have logged into your account, you will be greeted with the screen as shown here
On here you will want to click the "**Add/Import project**" button as shown in our screenshot beside



Registering for the Google API

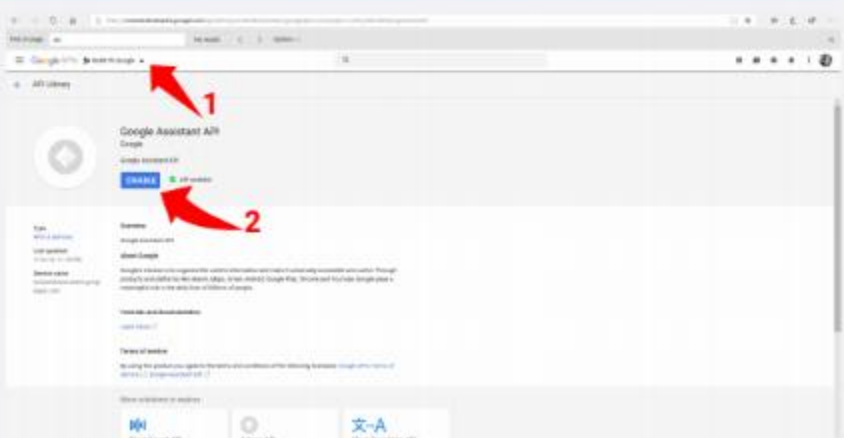
On this next screen, you will be asked to enter a "**Project Name**" (**1.**) In addition to a project name you need to set both your country and your language as shown in the screenshot (**2.**) Once you have set the Project Name and chosen your language and country, click the "**Create Project**" (**3.**) button.



Registering for the Google API

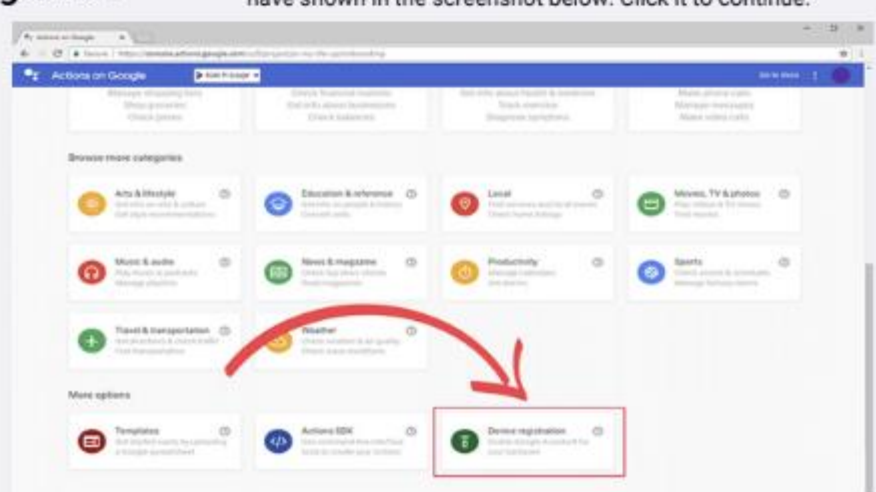
In a **new** tab, go to the [Google developers console]() and enable the Google Embedded Assistant API. Now before you go ahead and press the "**Enable**" button make sure that you have your project selected (**1.**) Once you are sure you have your current project selected, click the "**Enable**" button (**2.**)

https://console.developers.google.com/apis/library/embeddedassistant.googleapis.com



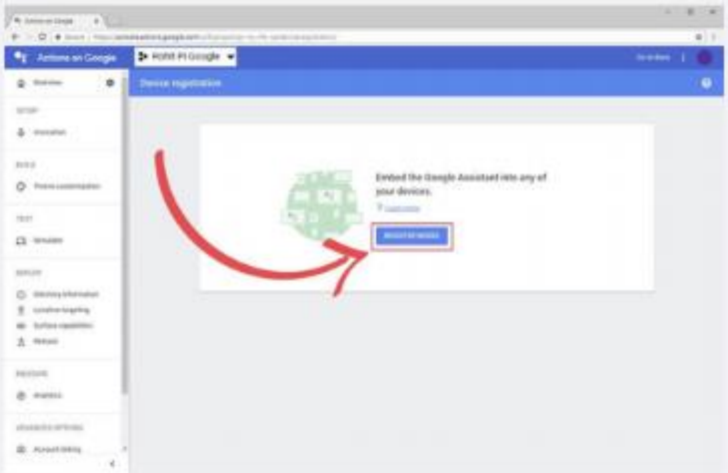
Registering for the Google API

Now back in the other tab where you created the project, scroll down to the bottom of the screen.
You should see a box with the text "**Device Registration**" on it as we have shown in the screenshot below. Click it to continue.



Registering for the Google API

Now back in the other tab where you created the project, scroll down to the bottom of the screen.
You should see a box with the text "**Device Registration**" on it as we have shown in the screenshot below. Click it to continue.



Registering for the Google API

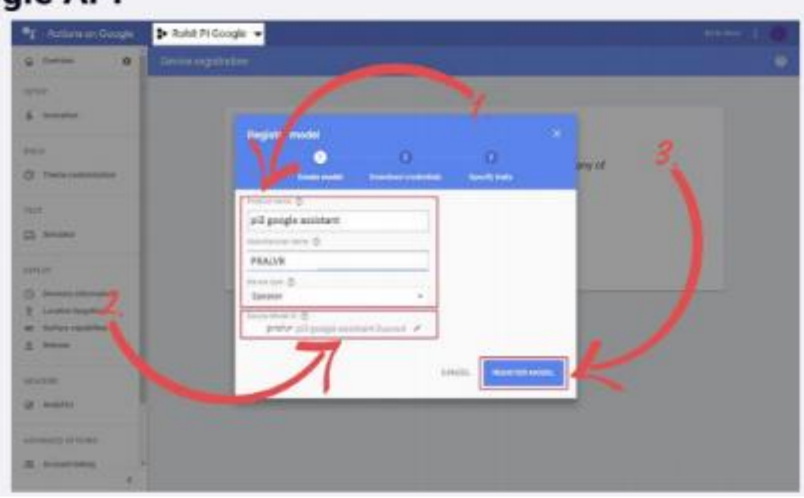
On this screen, you need to set a "**Product Name**", "**Manufacturer name**" and set a "**Device Type**" (**1.**)
Below you can see the data that we entered into it, it doesn't hugely matter what you set here, but all three boxes do need to be set for you to be able to register your model.

For the "**Product Name**" we just set this as a simple descriptor of what we are using this for, which in this tutorials case is simply a "**pi3 Google Assistant**".

"**Manufacturer name**" doesn't hugely matter as we have no intention of this being a widely used device, so we just set this to our websites name "**PRALVR**".

Lastly, we set the "**Device Type**" as "**Speaker**" as we felt it matched best what we intend on using the Google Assistant API for on our Raspberry Pi.
Make sure you write down then "**Device Model ID**" (**2.**) as you will need this later in the tutorial
Finally, once everything is set, and you have written down the "**Device Model ID**" press the "**Register Model**" (**3.**) button to continue.

Registering for the Google API



Registering for the Google API

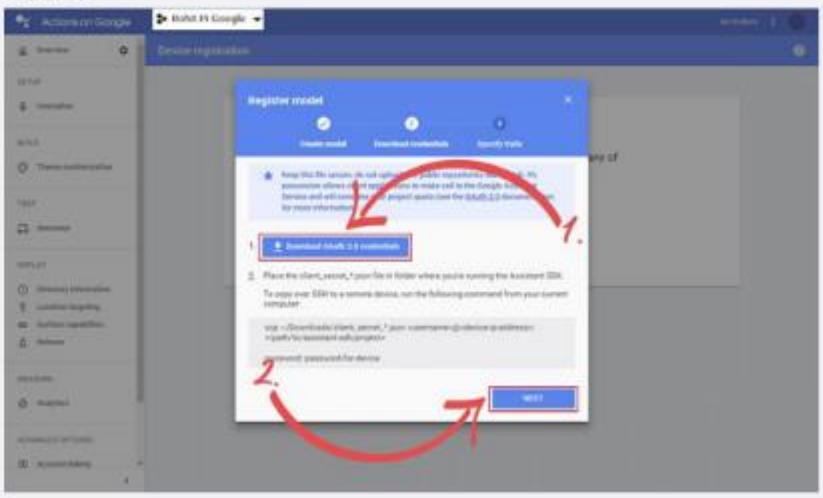
Now that you have registered the model you will now be taken to the "**Download credentials**" screen.
This screen is crucial as the provided credentials file is what we need for our Raspberry Pi 3 based Google Assistant to talk with the server.

To get this credentials file click the "**Download OAuth 2.0 credentials**" (**1.**) button as shown on the screenshot below.

Keep this somewhere safe, as we will the text inside the file to the Raspberry Pi. (Of course, unless you downloaded it directly to your Pi)

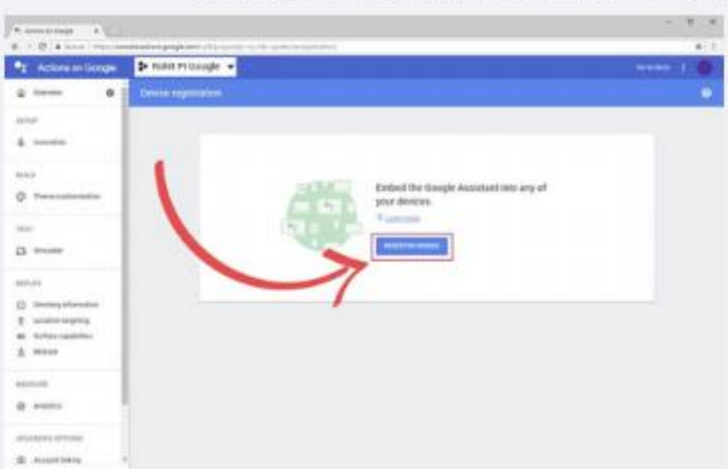
Once you have the credentials safely stored on your on your computer or Raspberry Pi, you need to click the "**Next**" (**2.**) button.

Registering for the Google API



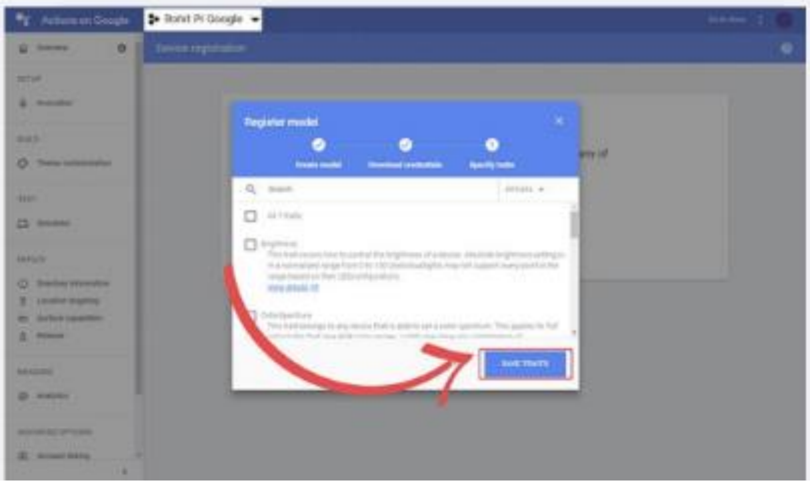
Registering for the Google API

Now back in the other tab where you created the project, scroll down to the bottom of the screen.
You should see a box with the text "**Device Registration**" on it as we have shown in the screenshot below. Click it to continue.



Registering for the Google API

Finally, you can specify any traits that you might need, in our case we don't need any of these so we just clicked the "**Save Traits**" button as shown below.



Registering for the Google API

Once everything is done, you should be shown on this screen. We now only have one last thing we need to do before we can set up the Google Assistant on the Raspberry Pi itself.

Registering for the Google API

Finally, we need to go to the URL displayed below, on here you will need to activate the following activity controls to ensure that the Google Assistant API works correctly.

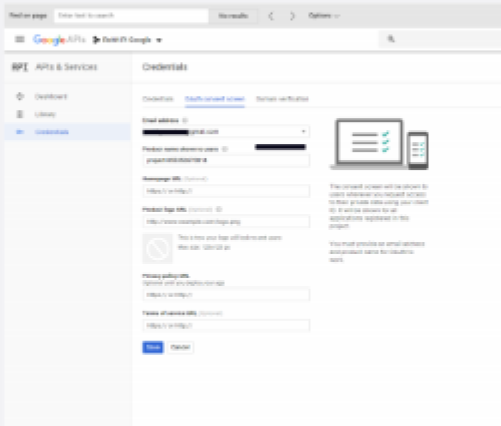
`https://myaccount.google.com/activitycontrols`

1. Web & App Activity
2. Location History
3. Device Information
4. Voice & Audio Activity

Registering for the Google API

Now goto following URL to provide consent with the account you want to access the project and product. If you don't do this then the future registration steps will fail.

https://console.developers.google.com/apis/credentials/consent



Just select the account by which you have created this project. The same credentials you have already downloaded before. So it must match for future steps. Select the same account and save.

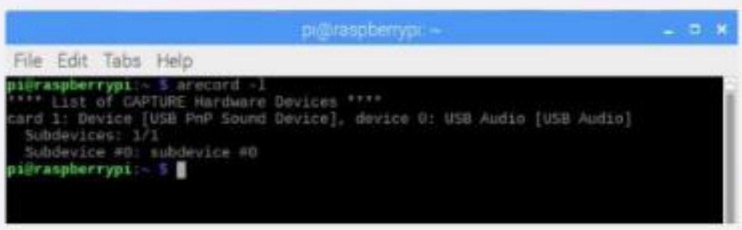
Setting up Audio for Google Assistant

Now that we have set up an account on the Google Actions Console we must configure the audio for it. The Google Assistant SDK that we will be using has some strict requirements for it to work correctly.

To get started with setting up the audio on the Raspberry Pi we must first obtain the card and device numbers for our various inputs and outputs. Our steps below will show you have to get the correct numbers for these devices.

Locate your USB microphone by utilizing the following command. Write down both the card number and the device number for it.

```
arecord -l
```



Checking Audio In/Out

Before we get into all the hard work of setting up our Raspberry Pi Google Assistant and setting up the required API . We will first test to ensure our audio is working.

At this stage, you must have your USB microphone and speakers attached to your Raspberry Pi.

Once you are sure both are connected to the Raspberry Pi, we can test to make sure that the speakers are working correctly by running the following command.

```
speaker-test -t wav
```

You should hear sound from your speakers. This sound will be a person speaking.

If you do not hear anything coming from your speaker's double check they are plugged in correctly and are turned up.

Checking Audio In/Out

Now, let's test our microphone by making a recording, to do this we will run the following command on your Raspberry Pi.

This command will make a short 5-second recording.

```
arecord --format=S16_LE --duration=5 --rate=16000 --file-type=raw out.raw  -D sysdefault:CARD=1
```

If you receive an error when running this command make sure that you have your microphone plugged in, this command will only succeed if it can successfully listen to your microphone.

Checking Audio
In/Out

With our recording done we can now run the following command to read in our raw output file and play it back to our speakers.
Doing this will allow you to test the playback volume and also listen to the recording volume.

Doing this is a crucial task as you don't want your Raspberry Pi picking up every little noise but you also don't want it being able to barely hear you when you say "**Ok Google**".

```
aplay --format=S16_LE --rate=16000 out.raw
```

Checking Audio
In/Out

If you find the playback volume or recording volume is either too high or too low, then you can run the following command to launch the mixer.

This command will allow you to tweak the output volumes for your various output devices. From our tests, we recommend you use a level of at least 70, utilize the command in **Step 1** of this section to check the volume levels.

```
alsamixer
```

Once you have confirmed that your microphone and speakers are working correctly, you can move onto setting up your very own Raspberry Pi Google Assistant!