Experiment No.1

Title: Familiarization with the concept of IOT, Arduino / Raspberry Pi and perform necessary software installation

Objectives: To study IOT, their characteristics of components and basic awareness of Arduino/Raspberry Pi

Outcomes: Students will be able to understand IOT, Arduino/ Raspberry Pi, and also be able to install software setup of Arduino/ Raspberry Pi.

THEORY:

Prerequisites: Fundamentals of Operating systems

Hardware Requirement: Arduino basic kit or Raspberry Pi starter kit

Software Requirement: Can be installed on LINUX and a stripped down IOT version of Windows 10

Introduction:

"The Internet of Things (IOT) is the interconnection of uniquely identifiable embedded computing devices within the existing Internet infrastructure. The Internet of Things connects devices and vehicles using electronic sensors and the Internet".

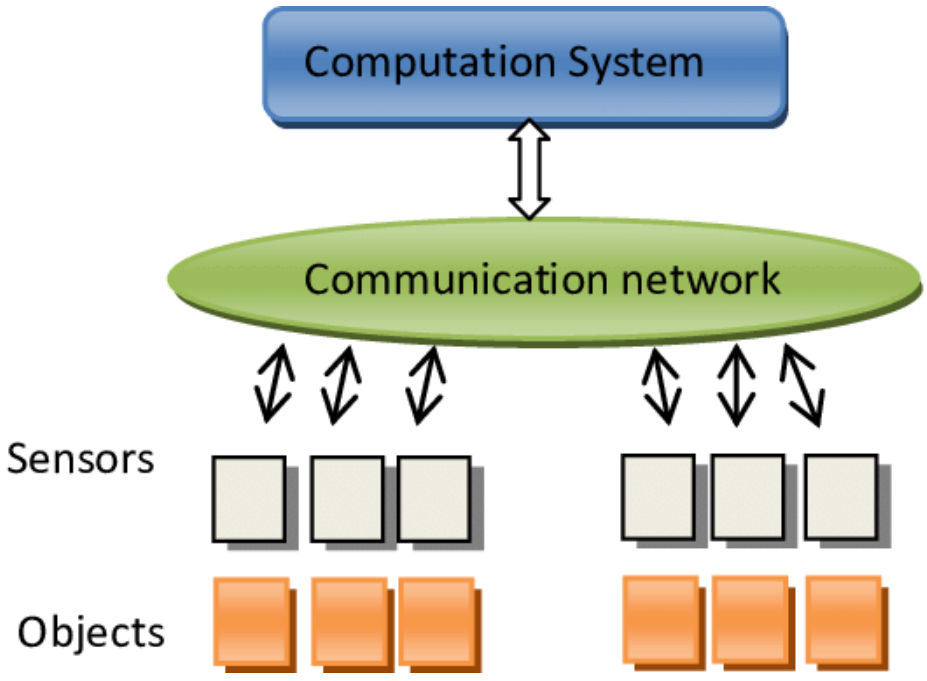


Figure 1: Internet of Things (IOT) basic architecture

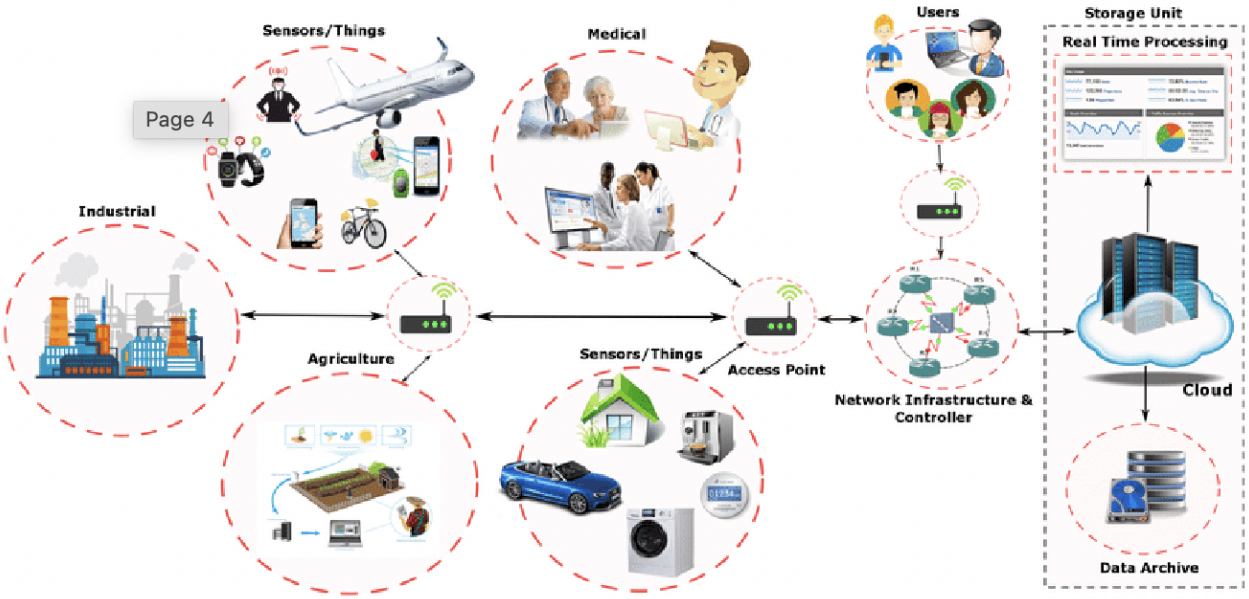


Figure 2: Internet of Things Architecture with applications

The Internet of Things (IOT) is defined as the network of physical objects, things that are embedded
with sensors, software and other technologies for the purpose of connecting and exchanging data with
other devices and systems over the internet (See Figure 1 and Figure 2)

Embedded platform:

What is a microcontroller?
Computer on a single integrated chip – Processor (CPU)
– Memory (RAM / ROM / Flash)
– I/O ports (USB, I2C, SPI, ADC) •Common microcontroller families: – Intel: 4004, 8008, etc.
– Atmel: AT and AVR
– Microchip: PIC
– ARM: (multiple manufacturers) •Used in:
– Cellphones,
–Toys
– Household appliances
– Cars
– Cameras

Arduino:
It is probably the best starting point for embedded based IoT. Based Arduino Boards don't come with Ethernet shield or Wi-Fi shield and for Arduino to be able to work as IoT device, their need to select Arduino with Ethernet shield or Wi-Fi shield. Arduino run on the other hand is a board that comes ported with Ethernet shield.

Raspberry Pi:

It is probably one of the best things to happen in DIY (DO it Yourself) IoT. A wide range of data driven applications like Home Automation Server to Home Multimedia server, File Server can be developed with Pi. Pi like Arduino has general purpose IO pins. But seamless working with sensors is a bit tedious in Pi. Another efficient IoT Board is Intel Edition which has integrated BLE, Wi-Fi among host of other features. It supports wide range of industry standard hardware (over 30) through 70 pin interface (See Figure 2).

Intel Galileo:It is another good offering by Intel which supports the same shielding that of Arduino Uno. So it can be said to be first Intel powered device which is Arduino compatible. It has among other thing a USB host controller like Raspberry Pi which makes this an attractive hardware. Galileo also has Ethernet shield built-in.

Programming with Arduino UNO:

Required Materials

To follow along with Arduino we will need the following materials. You may not need everything
though depending on what you have.

- A computer (Windows, Mac, or Linux)
- An Arduino-compatible microcontroller
- A USB A-to-B cable, or another appropriate way to connect your Arduino-compatible microcontroller to your computer

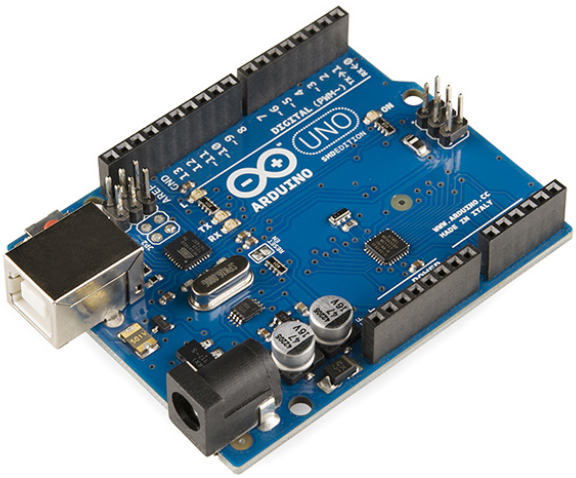


Figure3: Basic Arduino microcontroller

Figure 4: Cable for connection

What is an Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software.Arduino boards are able to read input light on a sensor, a finger on a button.

Downloading the Arduino IDE [visit: https://www.arduino.cc/en/Guide download: https://www.arduino.cc/en/software]

You can download the Arduino IDE from their website. They have installation instructions, but we will also go over the installation process as well. Make sure you download the version that matches your operating system.

The installation procedure is fairly straightforward, but it does vary by OS. Here are some tips to help you along

Troubleshooting Tips:

We recommend using a computer with a full desktop operating system like Windows 10/11,(avoid Windows 8 if you can), Mac OSX, and certain flavors Linux (check the Arduino FAQ page for compatibility).

If you are not a technical or computer savy individual and you have your choice of computers, It is highly recommend to use Windows 7 or 10 computer. You will usually run into the the least issues, if any, with these operating systems.

NOT recommended using a Chromebook, Netbook, tablet, phone, or the Arduino Web IDE in general. You will be responsible for troubleshooting any driver or Arduino Web IDE issues.

Arduino IDE is version 2.0. It is recommended to use that version of the Arduino IDE;

Raspberry Pi users with Raspbian installed should use the Linux ARM download.

It is not recommended using the command line installation. It will install the oldest release of Arduino, which is useless when it comes to installing new boards definitions or libraries.

For additional troubleshooting tips, here is a troubleshooting guide from Arduino

Installation in Windows:

How to install and test the Arduino software with a Windows operating system

Installer:
The Windows version of Arduino is offered in two options: an installer or a zip file.

The installer is the easier of the two options, just download that, and run the executable file to begin the installation.

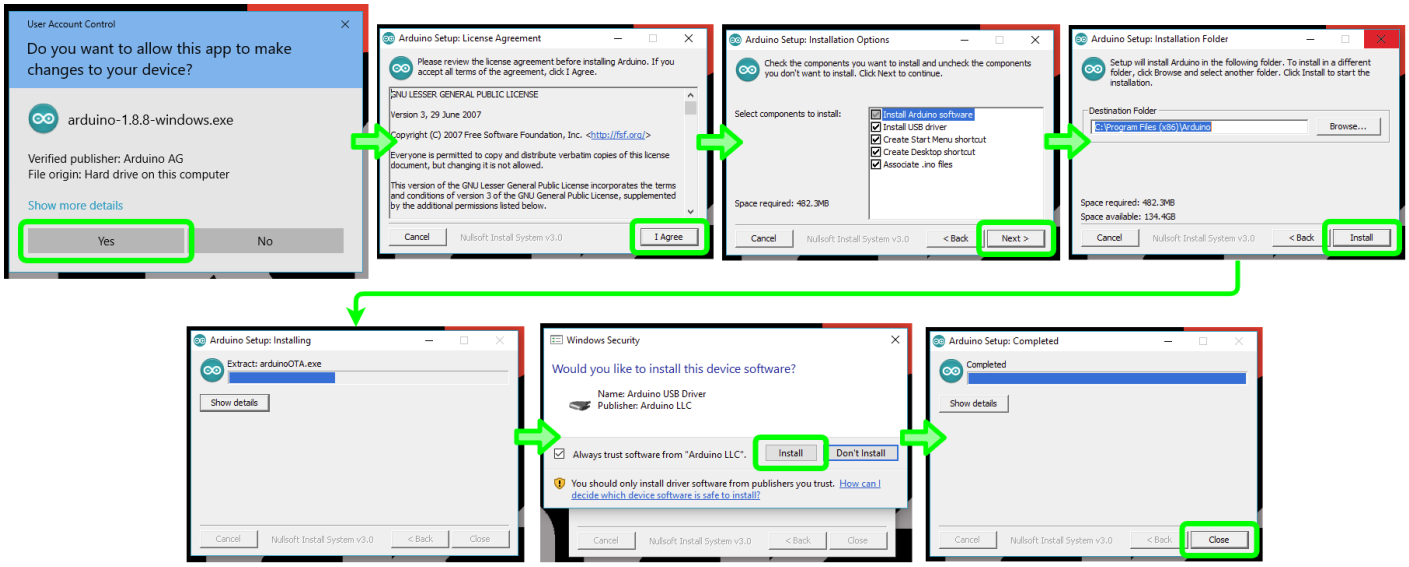


Figure 5: Arduino Installer in Windows

When you're prompted to install a driver during installation, select "Install". This will install drivers for Arduino specific boards (like the Uno, Nano, etc.) that you may use in the future.

ZIP:
If you choose to download the zip file version of Arduino, you'll need to extract the files yourself.
Then run the executable Arduino file in the folder to start the Arduino IDE.

When the download is finished, un-zip it and open up the Arduino folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is important so don't be moving any files around unless you really know what you're doing.

Connecting Arduino:
Power up your Arduino by connecting your Arduino board to your computer with a USB cable (or FTDI cable if you're using an Arduino Pro). You should see the an LED labeled 'ON' light up.

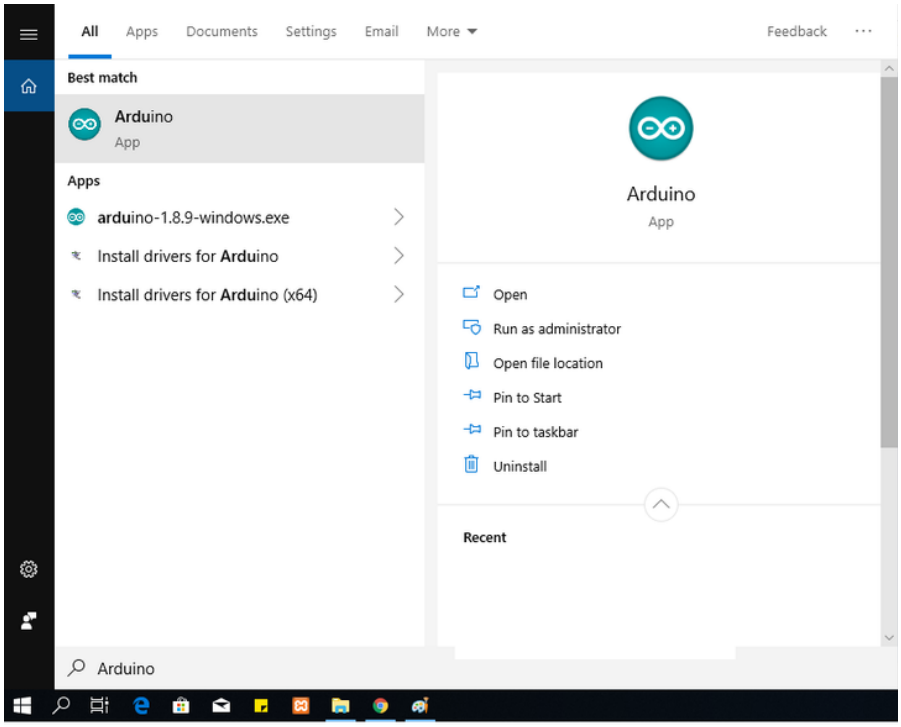


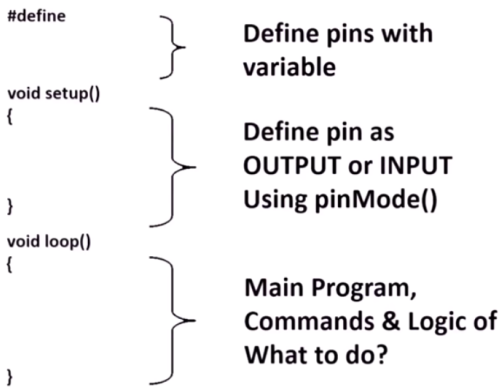
Figure 6 : Open Arduino IDE

After the installation process is complete, there will be an Arduino icon on the Desktop. Or check on the search icon and write "arduino". If you have found the arduino icon, run the application.

Figure 7: Arduino IDE

Arduino Sketch :

The program written in Arduino is called a sketch. The sketch has the following layout:



Once the sketch is written it should be uploaded into the arduino board. This process needs to ensure that the port is enabled in the Arduino IDE. If not then troubleshooting needs to be done. Either by installing the required drivers or by changing the cable (if driver installation doesnot help)

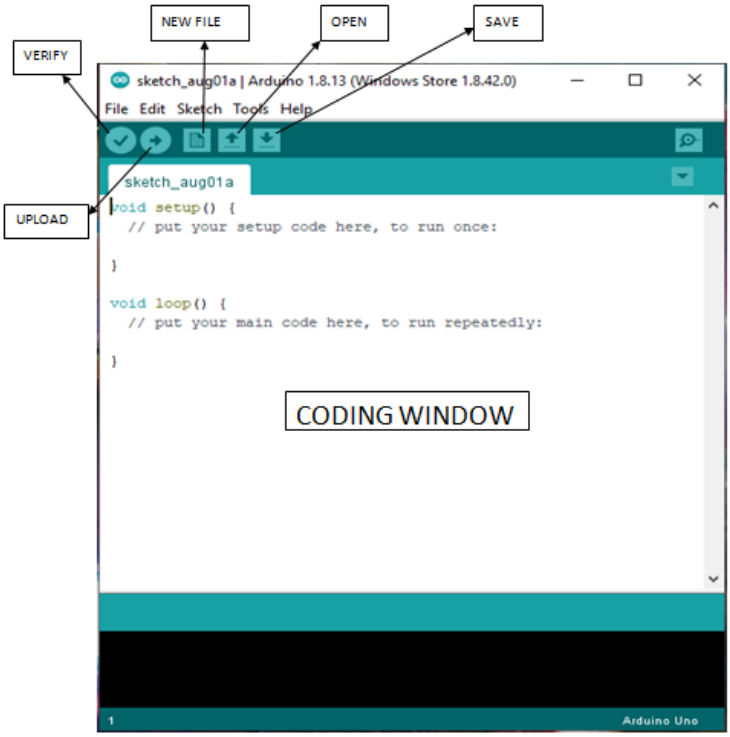


Figure 8: Detailed view of arduino

Select the type of Arduino board you're using: **Tools** > **Board** > **Arduino Uno**.

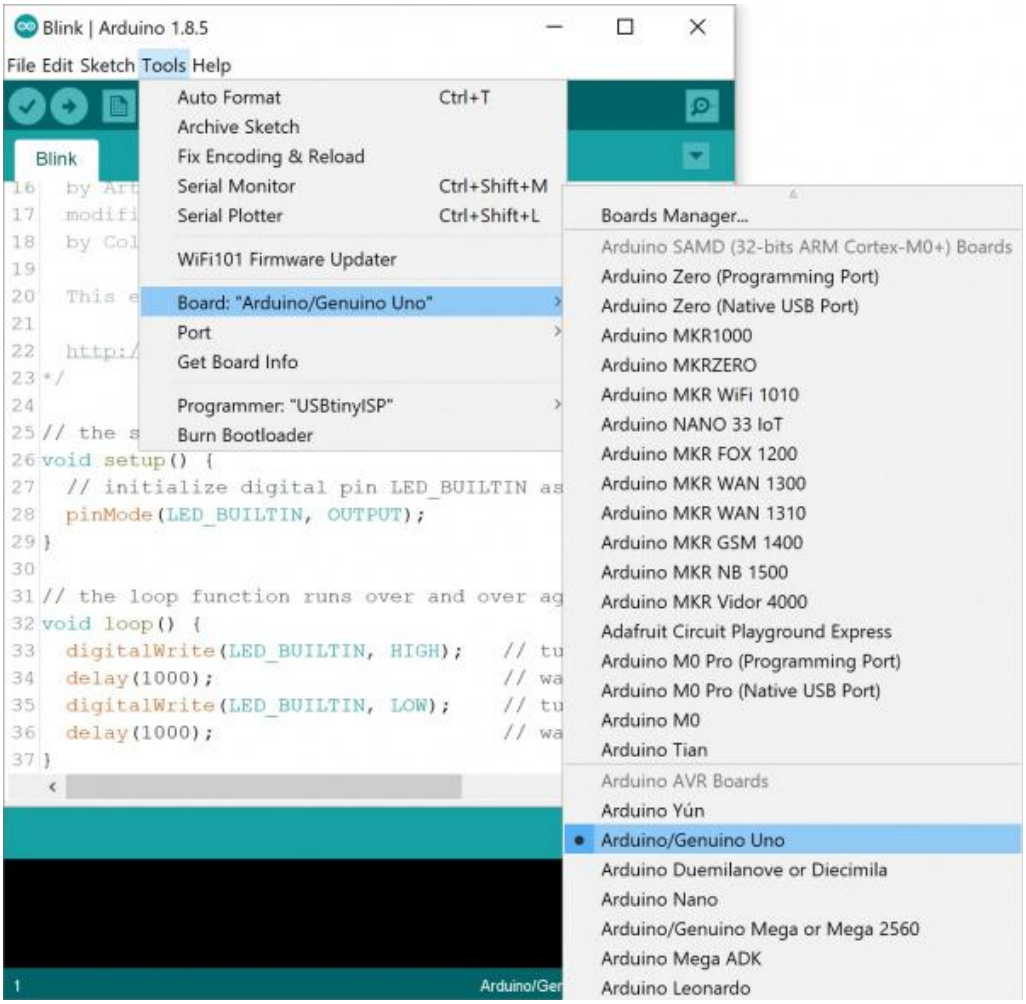


Figure 9: Selecting the board

Select the serial/COM port that your Arduino is attached to: **Tools** > **Port** > **COMxx**.

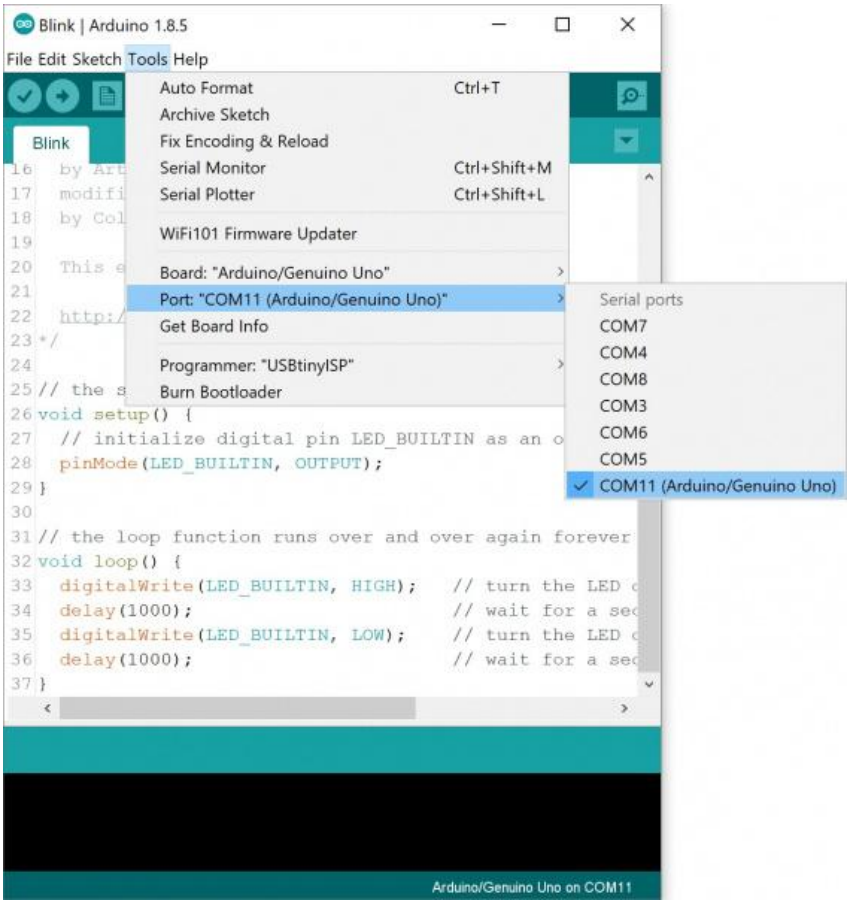


Figure 10: COM Port selection

Open the Blink example sketch by going to: **File** > **Examples** > **01.Basics** > **Blink**.

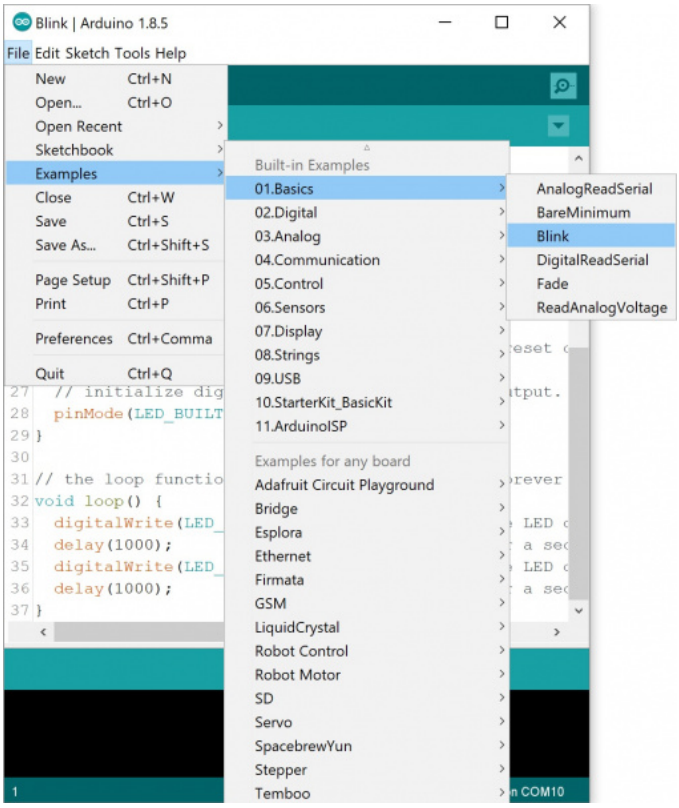


Figure 11 : Select Blink example

With your Arduino board connected, and the Blink sketch open, press the "**Upload**" button.

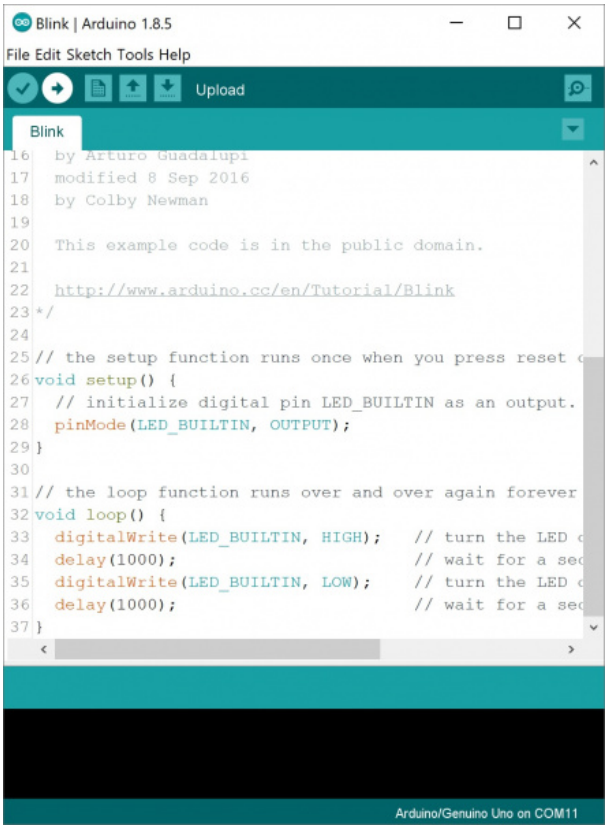


Figure 12 : Blink example

After a second, you should see some LEDs flashing on your Arduino, followed by the message "**Done Uploading**" in the status bar of the Blink sketch.
If everything worked, the onboard LED on your Arduino should now be blinking! You just programmed your first Arduino!

NOTE: Depending on the architecture and development board the built in LED may be defined on a different pin. Ex: LED_BUILTIN or PIN 13. You may change it before uploading the sketch.

Experiment no. 2:

Title: Study of different operating systems for Raspberry-Pi /Beagle board.
Understanding the process of OS installation on Raspberry-Pi /Beagle board

Different operating systems for Raspberry-Pi

The Raspberry Pi was designed for the Linux operating system, and many Linux distributions now have a version optimized for the Raspberry Pi.

Two of the most popular options are Raspbian, which is based on the Debian operating system, and Pidora, which is based on the Fedora operating system. For beginners, either of these two work well; which one you choose to use is a matter of personal preference. A good practice might be to go with the one which most closely resembles an operating system you're familiar with, in either a desktop or server environment.

Lots of other choices. OpenELEC and RaspBMC are both operating system distributions based on Linux that are targeted towards using the Raspberry Pi as a media center. There are also non-Linux systems, like RISC OS, which run on the Pi. Some enthusiasts have even used the Raspberry Pi to learn about operating systems by designing their own.

Raspbian is a Debian-based computer operating system for Raspberry Pi. Since 2015 till now it is officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers.

Raspbian was created by Mike Thompson and Peter Green as an independent project.•The initial build was completed in June 2012.The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARM CPUs.

Raspbian uses PIXEL, Pi Improved Xwindows Environment, Lightweight as its main desktop environment as of the latest update. •It is composed of a modified LXDE desktop environment and the Openbox stacking window manager with a new theme and few other changes. The distribution is shipped with a copy of computer algebra program Mathematica and a version of Minecraft called Minecraft Pi as well as a lightweight version of Chromium as of the latest version.

Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi. Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

Step 1: Raspberry Pi imager can be downloaded for windows, MAC, Linux. The following website has the corresponding links: [Raspberry Pi downloads page]

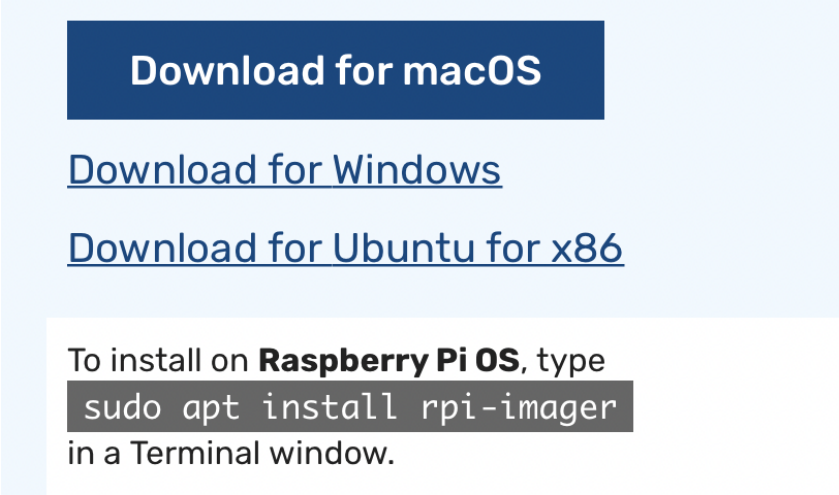


Figure 1:Raspberry Pi OS Download page

Step 2: When the download finishes, click it to launch the installer

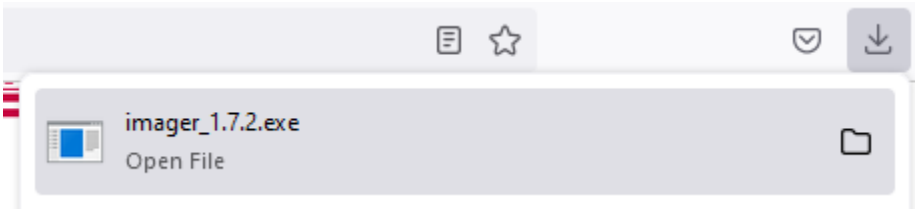


Figure 2: Downloaded imager file

Step 3: Anything that's stored on the SD card will be overwritten during formatting. If your SD card currently has any files on it, e.g. from an older version of Raspberry Pi OS, you may wish to back up these files first to prevent you from permanently losing them. [NOTE: When you launch the installer, your operating system may try to block you from running it. You may choose to run it anyway]. Also chose to format SD card as FAT32 before writing OS to it.

Step 4: Follow the instructions to install and run the Raspberry Pi Imager

- Insert your SD card into the computer or laptop SD card slot
- In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on

Figure 3: Raspberry Pi imager setup for writing to SD card

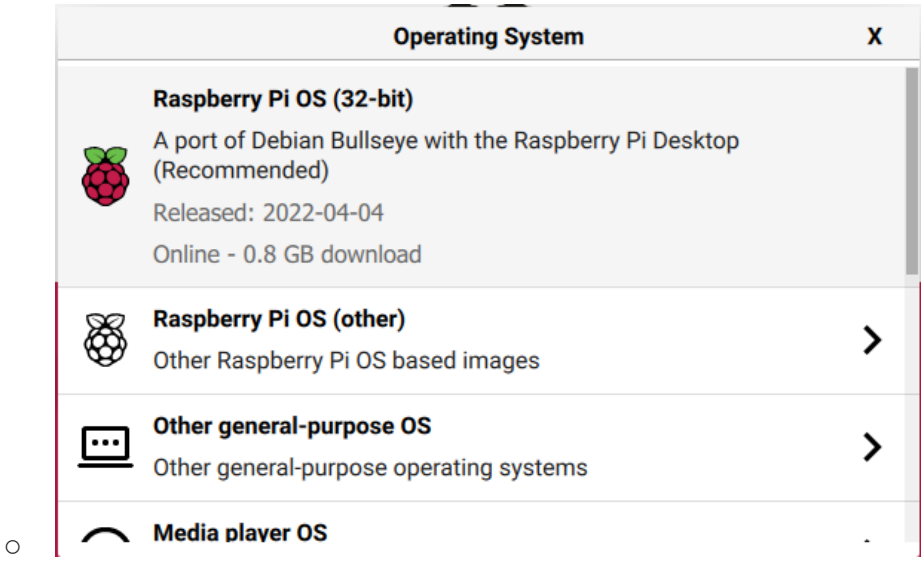


Figure 4: Selection of OS and mass storage

Note: You will need to be connected to the internet the first time for the Raspberry Pi Imager to download the OS that you choose. That OS will then be stored for future offline use. Being online for later uses means that the Raspberry Pi imager will always give you the latest version.

Step 5: You can now insert the SD card into the Raspberry Pi and power it up. When your Raspberry Pi boots for the first time a configuration wizard will run that allows you to set up your Raspberry Pi.

Figure 5: Raspberry Pi connected with keyboard, mouse and monitor

The SD card with Pi OS is inserted into the Raspberry Pi and after powering up the Pi, booting will take place. After the setup wizard, Pi OS launch will look like the below image Figure 2.

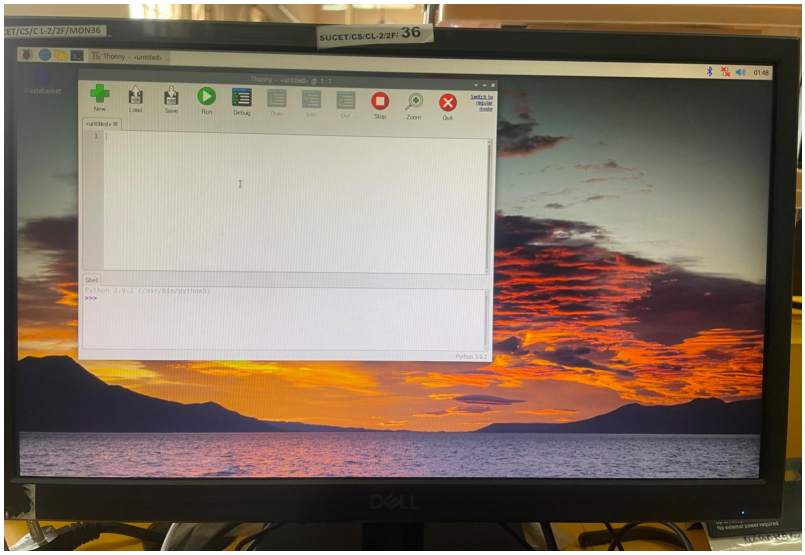


Figure 6: Raspberry Pi OS interface with Thonny Python IDE

Conclusion : Studied the basics of Raspberry Pi and the procedure for installing Raspbian OS and boot Raspberry Pi

Experiment no: 3
Study of Connectivity and configuration of Raspberry-Pi /Beagle board circuit with basic peripherals, LEDS. Understanding GPIO and its use in program.

Problem Statement: Study of Connectivity and configuration of Raspberry-Pi /Beagle board circuit with basic peripherals, LEDS. Understanding GPIO and its use in program

Objective:

1. To understand Connectivity and configuration of Raspberry-Pi
2. To Understanding GPIO and its use in program
3. To understand board circuit with basic peripherals, LEDS

Theory:

Connectivity and configuration of Raspberry-Pi

RPi Remote Connections:

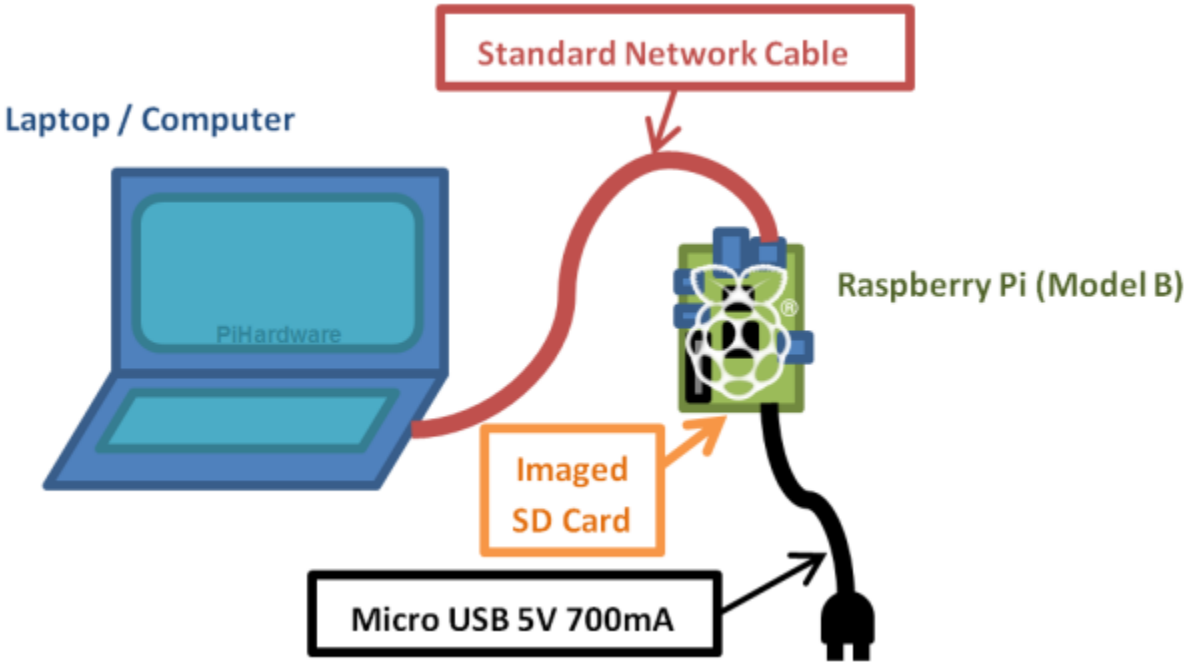


Figure 1: RPi Connectivity

Storage: Secure Digital (SD)
•Form factor
– SD, Mini SD, Micro SD •
Types of Card
– SDSC (SD): 1MB to 2GB – SDHC: 4GB to 32 GB
– SDXD up to 2TB

The card should be at least 2GB in capacity to store all the required files.

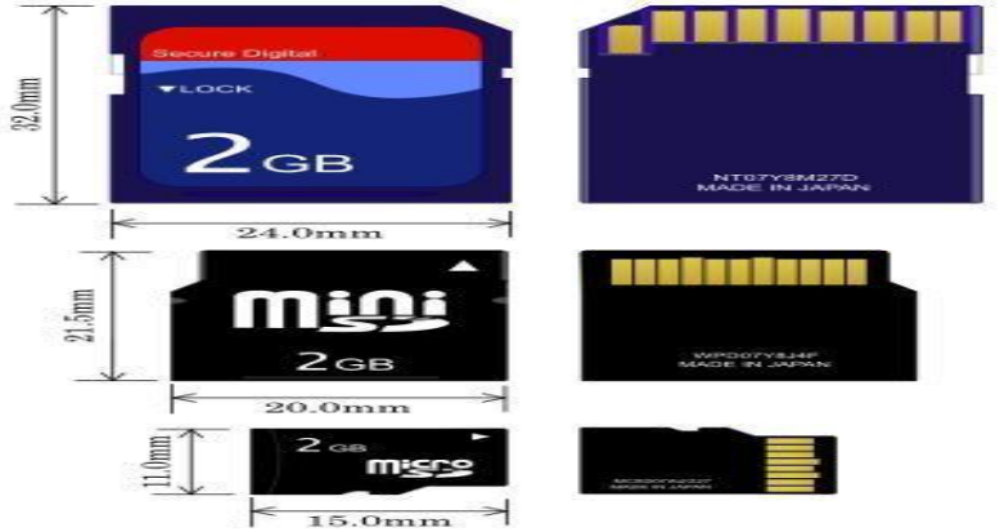


Figure 2: SD cards

Low Speed Peripherals:

General Purpose Input/Output (GPIO)

– Pins can be configured to be input/output

– Reading from various environmental sensors

- Ex: IR, video,temperature, 3-axis orientation, acceleration

– Writing output to dc motors, LEDs for status.

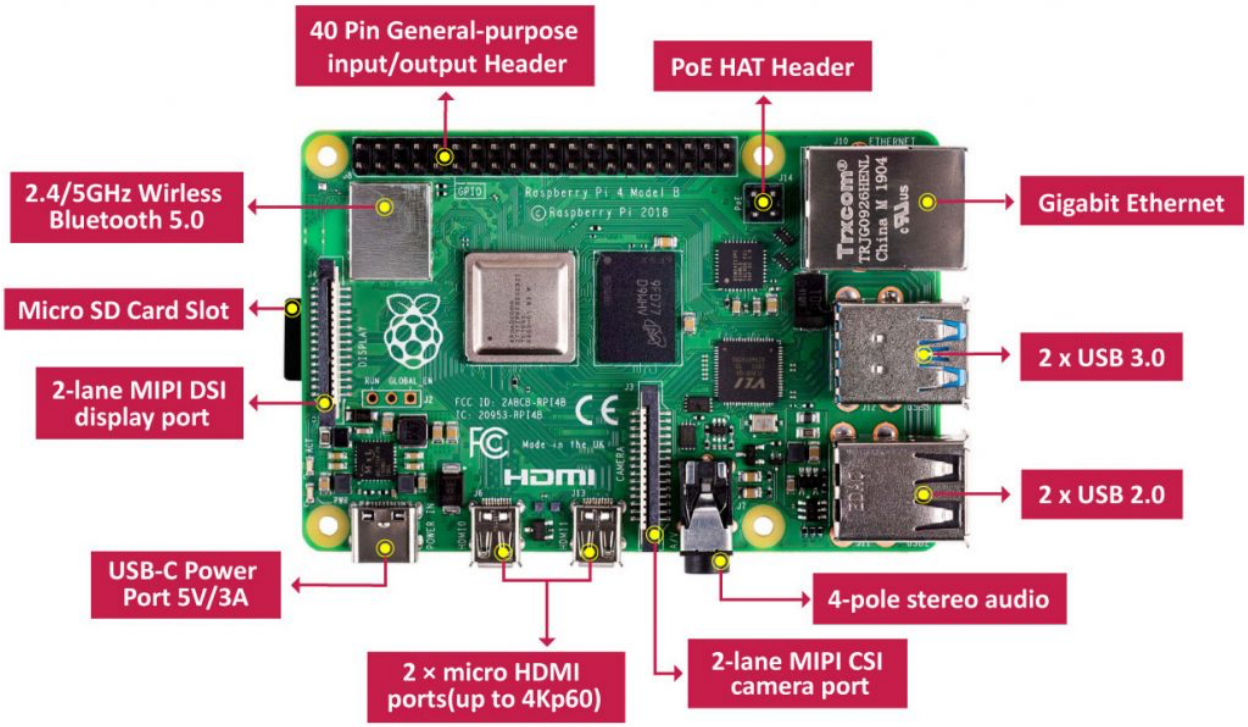
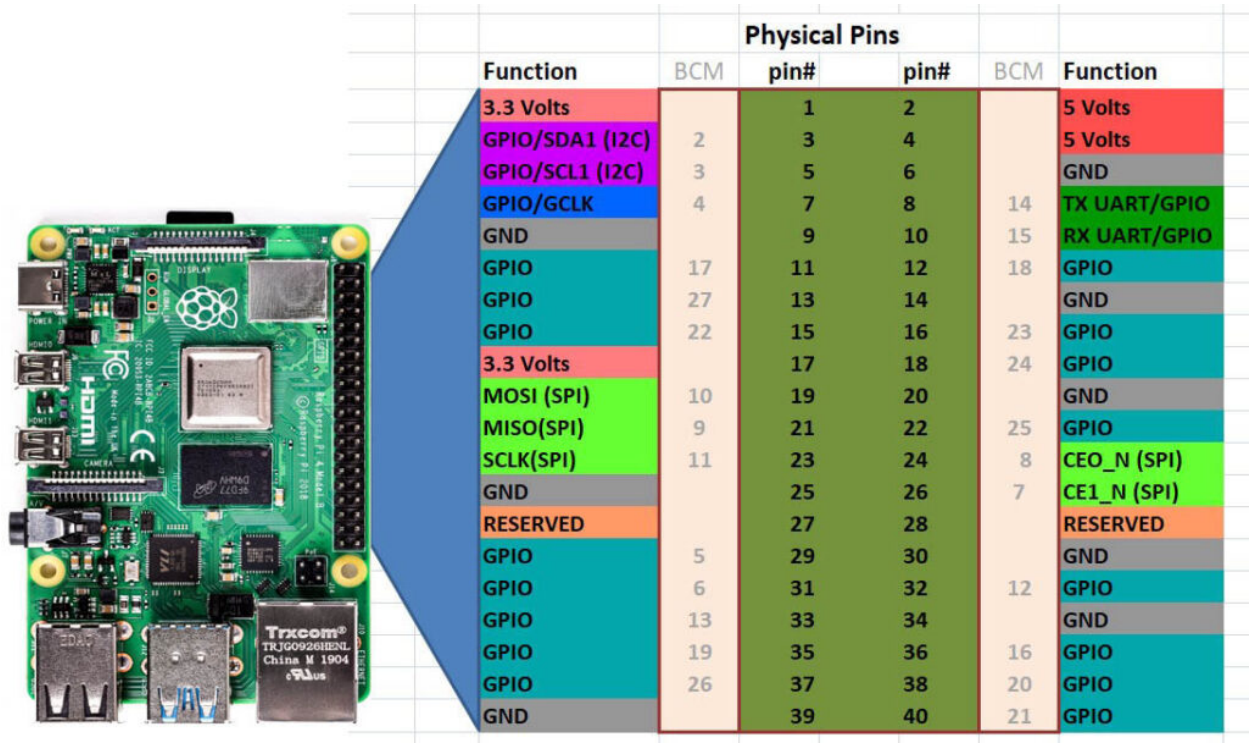


Figure 3: Raspberry Pi 4 layout



Function	BCM	Physical Pins pin#	pin#	BCM	Function
3.3 Volts		1	2		5 Volts
GPIO/SDA1 (I2C)	2	3	4		5 Volts
GPIO/SCL1 (I2C)	3	5	6		GND
GPIO/GCLK	4	7	8	14	TX UART/GPIO
GND		9	10	15	RX UART/GPIO
GPIO	17	11	12	18	GPIO
GPIO	27	13	14		GND
GPIO	22	15	16	23	GPIO
3.3 Volts		17	18	24	GPIO
MOSI (SPI)	10	19	20		GND
MISO(SPI)	9	21	22	25	GPIO
SCLK(SPI)	11	23	24	8	CEO_N (SPI)
GND		25	26	7	CE1_N (SPI)
RESERVED		27	28		RESERVED
GPIO	5	29	30		GND
GPIO	6	31	32	12	GPIO
GPIO	13	33	34		GND
GPIO	19	35	36	16	GPIO
GPIO	26	37	38	20	GPIO
GND		39	40	21	GPIO

Figure 4: GPIO Pin out

Raspberry Pi is a sensational single-board computer (SBC) and development board, which is heavily used for the prototyping of IoT based products. It is the best-selling British computer ever, with more than 10 million pieces sold already. Contrary to common belief, Raspberry Pi is not entirely open source. Yet, it has been used in many open source projects. In this article, we will learn to use it as a development board. For all the programs and circuits discussed here, I have used Raspberry Pi 3. Unless explicitly specified otherwise, from now onwards, the name Pi refers to Raspberry Pi 2 Model B, 3 Model B or Zero.

Raspberry Pi pinout

What makes Raspberry Pi suitable for making IoT projects is its 40-pin expansion header. Pins are arranged in a 2×20 fashion. Figure 1 is a pinout diagram.

Numbering systems

Raspberry Pi pins are numbered in two different ways— physical numbering and Broadcom numbering (BCM). In the first case, the pins are numbered sequentially from one to 40. In the figure, this is shown as *Pin#.* And the image represents what is seen when the Pi is held with the USB ports facing downwards. That is, Pin 2 is the pin in the corner. The Broadcom numbering system is the default option for the SoC (System-on-Chip). Raspberry Pi uses a SoC developed by Broadcom Limited. Raspberry Pi 2 and Zero use BCM2836 and BCM2835, while the Pi 2 version 1.2 and 3 use BCM2837. This is also known as GPIO numbering and is shown as *GPIO#* in the figure.

As you have probably guessed already, all the pins are not programmable. There are eight ground pins and two +5V pins and three +3.3V pins, which are not programmable. There are other dedicated pins too. Most of the pins have alternative functions, as shown in the figure. A majority of the pins are directly connected to the SoC; so while connecting circuits or components, one should be careful to avoid wrong wiring and short circuits. It is always good to have a descriptive pinout diagram printed out for quick reference as well as a multimeter on the work desk.

Programming the pins

Armed with some understanding about the pins, let us move to programming. The Python package used for Raspberry Pi GPIO programming is *RPi.GPIO*. It is already installed in Raspbian, the default operating system for Pi.

If you are using any other operating system, the package can be installed by using the following command:

$ sudo pip install RPi.GPIO

Now, fire up IDLE or the Python console and type the following commands:
import RPi.GPIO as GPIO
print(GPIO.RPI_INFO)

You will get some basic information about your Pi printed as the output. So the ‘Hello world’ of GPIO programming is done!

OR in the Terminal we can type the command : *pinout*

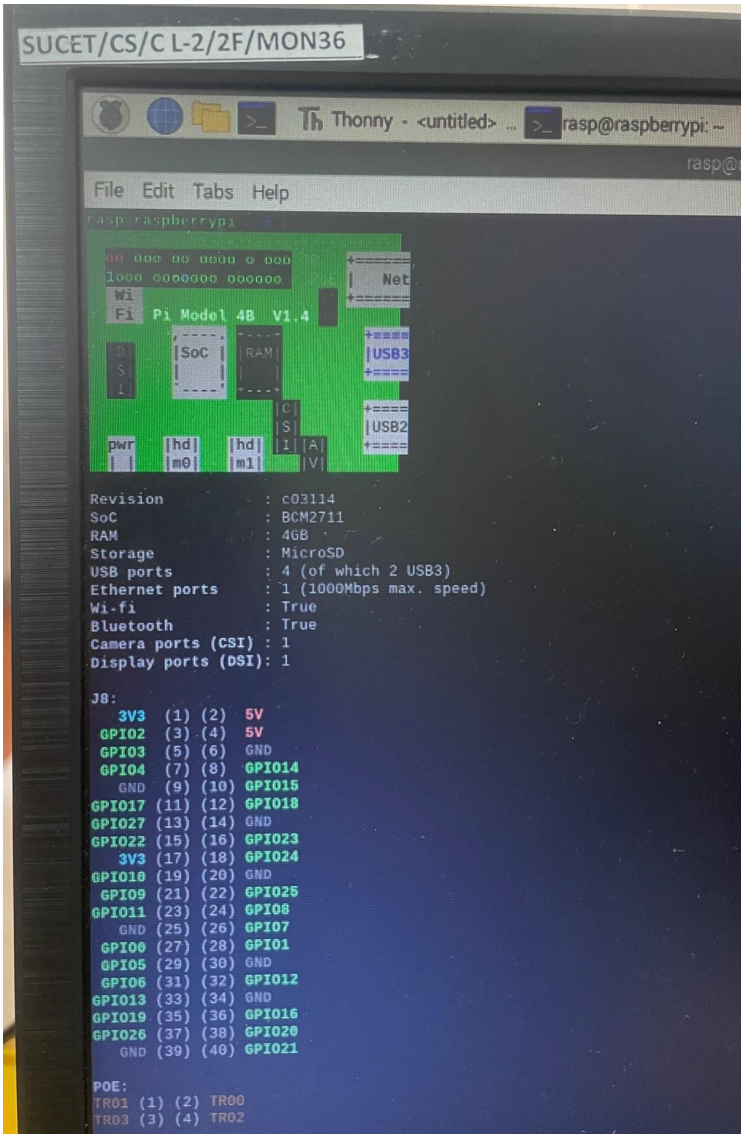


Figure 5: output of *pinout* command

PROCEDURE:

LED Blink Programming using Raspberry Pi:

- In Thonny Python IDE, create a python script. For example : blink.py
- There are two ways to do the blink program.
    - Using gpiozero library
        - This library is easiest way to blink LED. At the same time it is simple to understand and code.
    - RPi.GPIO library
        - This is old library and similar to arduino.
- blink.py will contain the following code:

```
#blinking with gpiozero library
from gpiozero import LED
from time import sleep
led = LED(23)
```

```
while True:
    led.on()
    print('LED ON')
    sleep(1)
    led.off()
    print('LED OFF')
    sleep(1)
```

1. First we will import LED from gpiozero library and sleep from time library.
2. create an object called led that refers to GPIO 23. You can see this number on the pinout chart.
3. In the loop, call on() and off() functions to turn the led on and off. A delay of 1 second is introduced

Python code for blinking LED using RPi.GPIO library:

```
#blink with RPi.GPIO library
import RPi.GPIO as GPIO
import time
ledPin = 16
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)    #to avoid default warnings
GPIO.setup(ledPin,GPIO.OUT)
while True:
    GPIO.output(ledPin, GPIO.HIGH)
    print('LED ON')
    time.sleep(1)
    GPIO.output(ledPin, GPIO.LOW)
    print('LED OFF')
    time.sleep(1)
```

1. import Rpi.GPIO as GPIO and import time for sleep
2. Set pin number for LED. It is also depends on which boarding scheme you are using.
3. For example if we use GPIO.BCM (BCM : Broadcom chip-specific pin numbers), On board pin number it is 16 but on BCM number it is 23. So our ledpin number is 23 for GPIO.BCM. If we are using GPIO.BOARD our pin number will be 16.

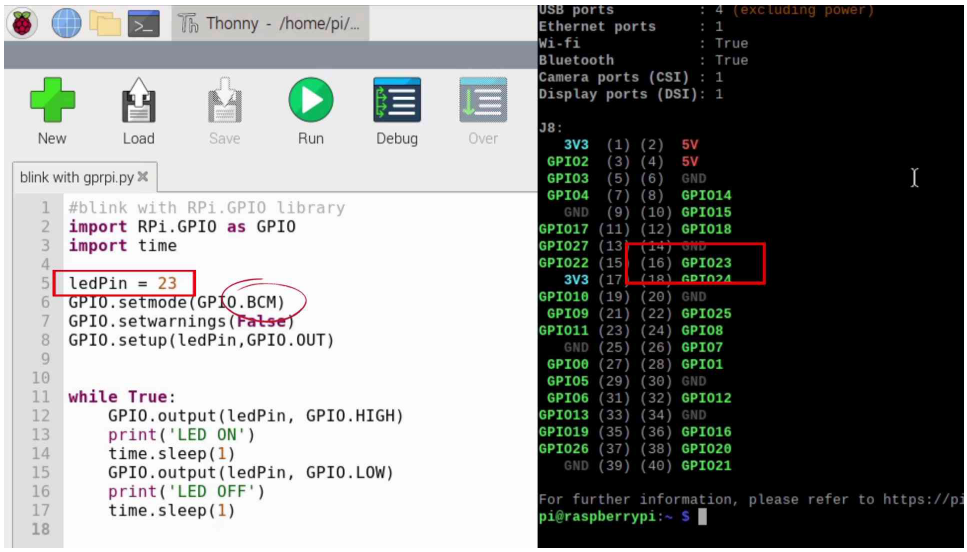


Figure 6: GPIO.BCM and pin numbers

4. Set ledpin as output. Similar to arduino sketch.
5. Finally in the while loop, the ledpin is set to high and low with a sleep delay of 1 second.

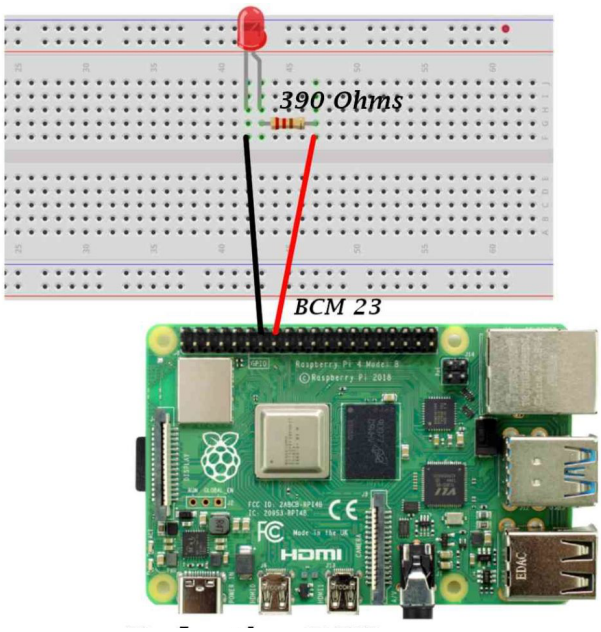


Figure 8: Circuit for LED Blink

Conclusion: We Studied Connectivity and configuration of Raspberry-Pi circuit with basic peripherals, LEDS.

4. Understanding connectivity of Raspberry-Pi /Beagle board with camera

Problem Statement: Understanding the connectivity of Raspberry-Pi /Beagle board circuit with Pi Camera

Theory:

Pi Camera Module (v1.3)

Pi Camera module is a camera which can be used to take pictures and high definition video. Raspberry Pi Board has CSI (Camera Serial Interface) interface to which we can attach Pi Camera module directly. This Pi Camera module can attach to the Raspberry Pi's CSI port using 15-pin ribbon cable.

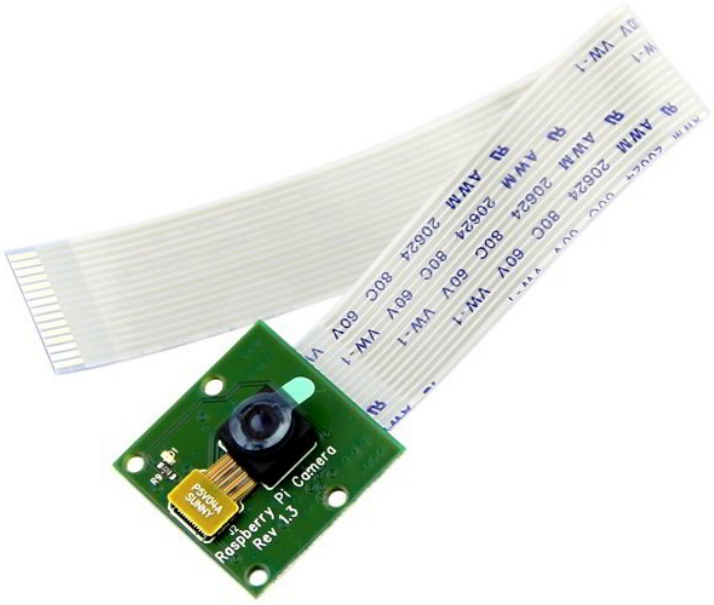


Figure 1: Pi Camera

Features of Pi Camera:

Here, we have used Pi camera v1.3. Its features are listed below,

- Resolution – 5 MP
- HD Video recording – 1080p @30fps, 720p @60fps, 960p @45fps and so on.
- It Can capture wide, still (motionless) images of resolution 2592x1944 pixels
- CSI Interface enabled.

Attaching Pi Camera to Raspberry Pi:

Connect Pi Camera to CSI interface of Raspberry Pi board as shown below,

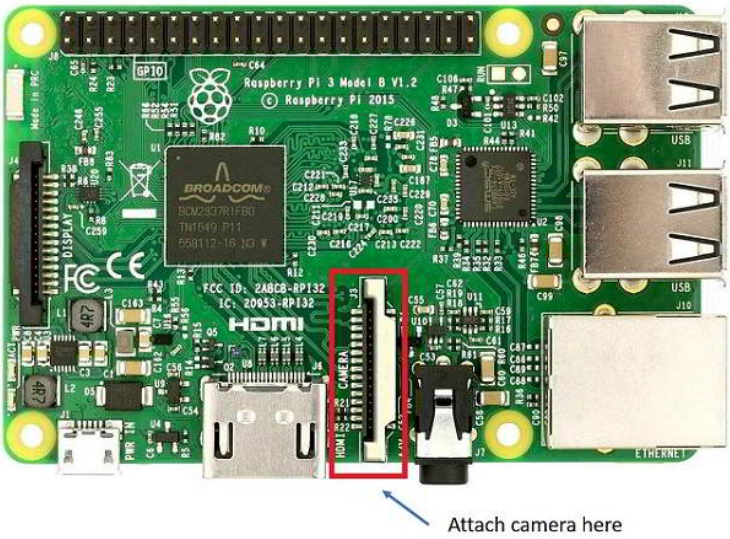


Figure 2: Pi Camera Slot in RPi

Figure 3 : Pi camera connection

Connect the Camera Module

Ensure your Raspberry Pi is turned off.

1. Locate the Camera Module port
2. Gently pull up on the edges of the port's plastic clip
3. Insert the Camera Module ribbon cable; make sure the connectors at the bottom of the ribbon cable are facing the contacts in the port.
4. Push the plastic clip back into place

Turn on RaspberryPi. Now, we can use Pi Camera for capturing images and videos using Raspberry Pi. Before using the Pi Camera, we need to enable the camera for its working.

Enable Camera functionality on Raspberry Pi:

For enabling camera in Raspberry Pi, open raspberry pi configuration using following command,

$ sudo raspi-config

then select **Interface options** in which select **legacy camera** option to enable its functionality. Also enable I2C option in the interfaces.

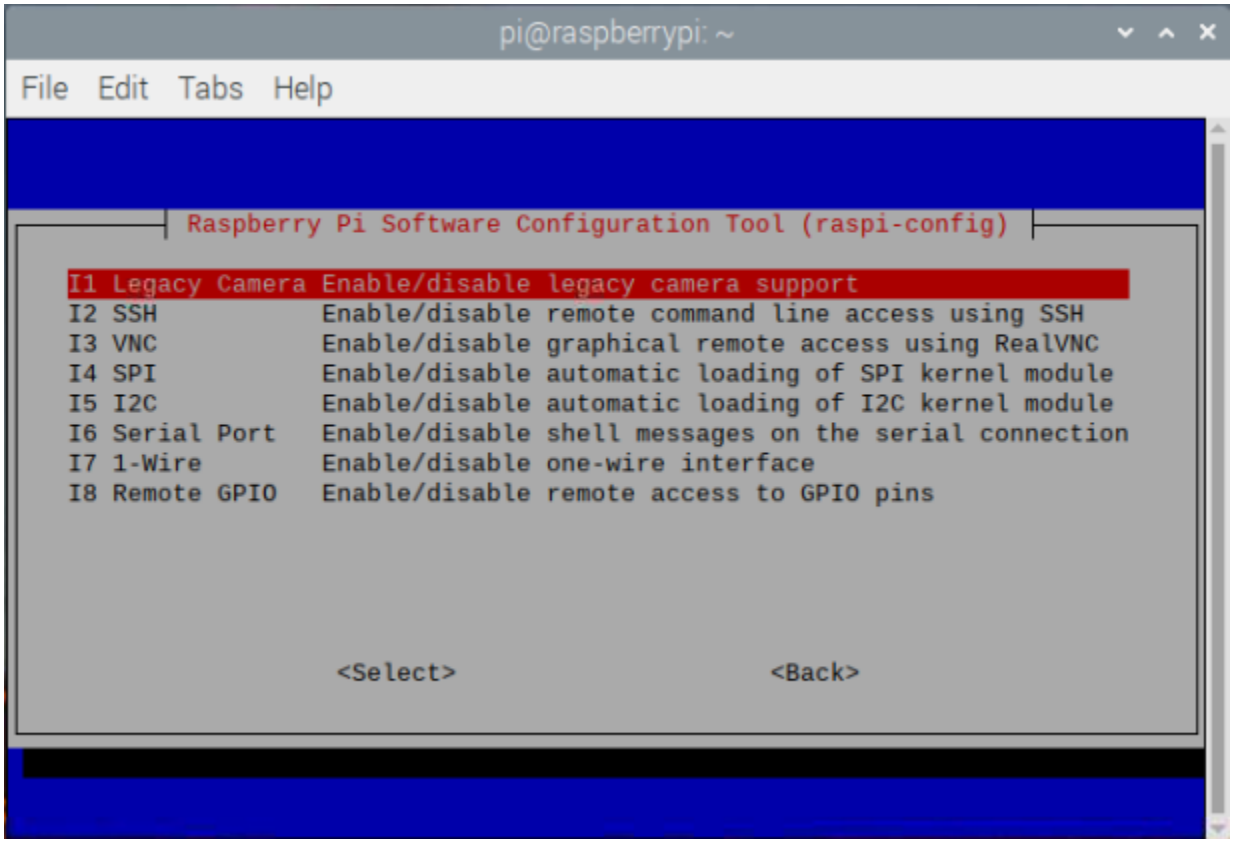


Figure 4: Enable Camera and I2C

Then reboot Raspberry Pi to apply the changes.

The following command captures an image and saves in the specified folder.

$ raspistill -o image.jpg

This will capture the image and save it in image.jpg

We can record a video with the Camera Module by using the following raspivid command:

$ raspivid -o video.mp4

We can capture images using Python.

The Python picamera library allows you to control your Camera Module and create amazing projects.

Python Program for Image Capture

```
from picamera import PiCamera

from time import sleep

camera = PiCamera()

camera.start_preview()

sleep(5)

camera.stop_preview()
```

Functions Used

To use picamera python based library we have to include it in our program as given below

```
import picamera
```

This picamera library has PiCamera class for camera module. So, we have to create object for **PiCamera** class.

PiCamera Class

To use Pi Camera in Python on Raspberry Pi, we can use PiCamera class which has different APIs for camera functionality. We need to create object for the PiCamera class.

E.g. camera = PiCamera()

The above PiCamera class has different member variables and functions which we can access by simply inserting a dot (.) in between object name and member name.

E.g. Camera.resolution = (1080, 648)

It has functions like :

1. capture()-It is used to capture images using Pi Camera.

E.g. camera.capture("/home/pi/image.jpeg")

The capture() function has different parameters which we can pass for different operations like resize, format, use_video_port, etc.

E.g. camera.capture("/home/pi/image.jpeg", resize=(720, 480))

2. resolution= (width,height)

It sets the resolution of camera at which image captures, video records and preview will display. The resolution can be specified as **(width, height)** tuple, as a string formatted **WIDTHxHEIGHT**, or as a string containing commonly recognised display resolution name e.g. "HD", "VGA", "1080p", etc.

E.g.

camera.resolution = (720, 480)

camera.resolution = “720 x 480”

camera.resolution = “720p”

camera.resolution = “HD”

3. Annotate_text = “Text”

It is used to add text on image, video, etc.

E.g. camera.annotate_text = “Hi Pi User”

4. start_preview()

It displays the preview overlay of default or specified resolution.

Example camera.start_preview()

5. stop_preview()

It is used to close the preview overlay.

E.g. camera.stop_preview()

Note: There are various APIs of the PiCamera class. So, to know more API in detail you can refer PiCamera APIs.

Output:

Figure 5: Captured image

Conclusion : Successfully connected RPi camera with Raspberry Pi and capture images and videos.

5. Write an application using Raspberry-Pi /Beagle board/Arduino Board to control the operation of stepper motor

Learn the working of stepper motor and motor speed control using Arduino kit.

Objective:

1. Learn the technicality of DC motor
2. Developing an arduino sketch to run and control the DC motor.
3. Designing the circuit connection using arduino and DC motor.

Hardware Required:

1. Arduino Universal Development Kit
2. DC motor

Theory:

DC motor:

A DC motor is an electrical machine that converts electrical energy into mechanical energy. In a DC motor, the input electrical energy is the direct current which is transformed into the mechanical rotation.

A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

When kept in a magnetic field, a current-carrying conductor gains torque and develops a tendency to move. In short, when electric fields and magnetic fields interact, a mechanical force arises. This is the principle on which the DC motors work.

Experimental Setup and Procedure:

1. Connect DC motor to motor 1 (shown in figure 1) slot of arduino universal development kit as shown in figure 2.
2. Choose the Motor Control sketch available under Sample codes folder in IoT Development Kit folder in C drive.

3. Connect the arduino kit to PC
4. Compile and upload the motor control sketch to the arduino board.
5. You will see the motor rotating in clockwise and anticlockwise directions.
6. In the sketch the pin details are mentioned as for motor 1 slot connects to pin number 10 and 11 of arduino and motor slot 2 connects to pin number 7 and 9 of arduino.

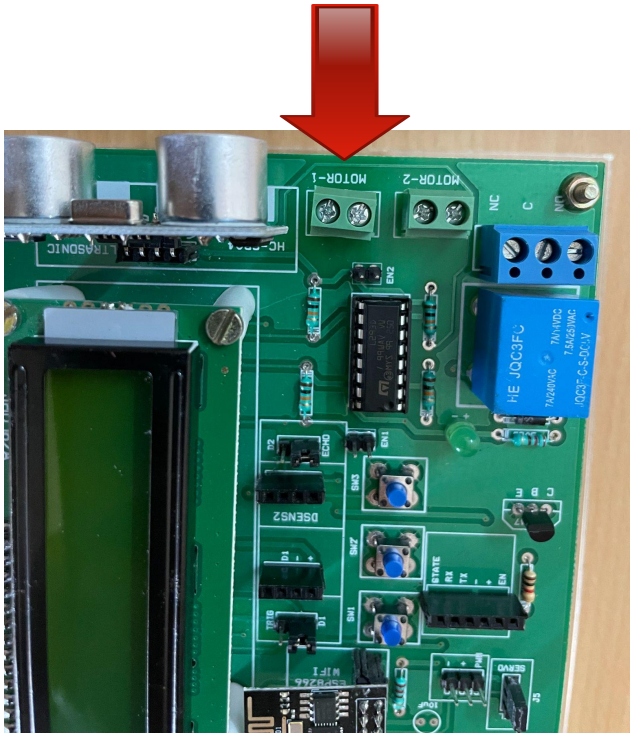


Figure 1: Motor slots in Universal Development Kit

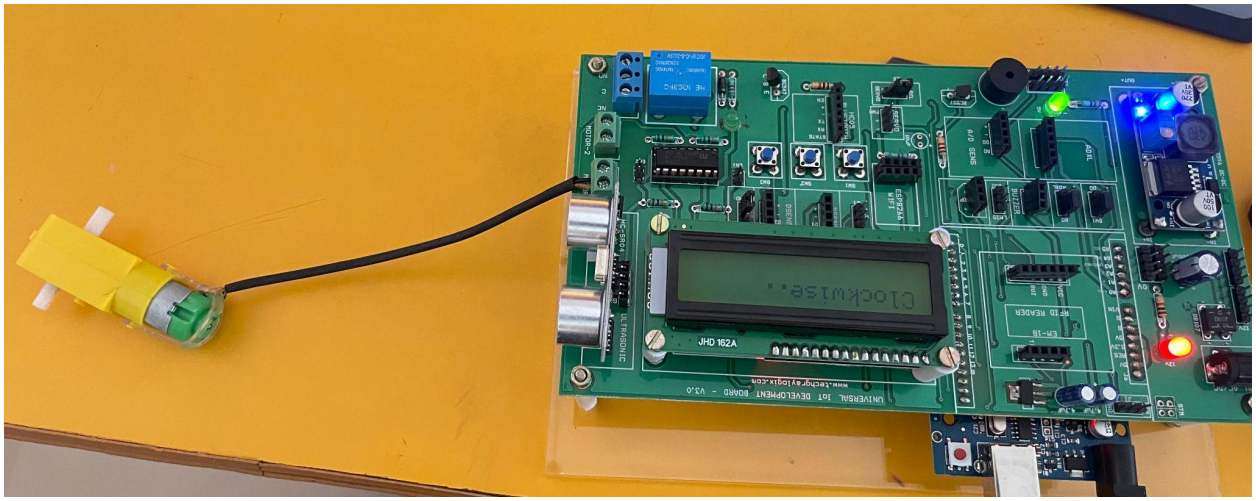


Figure 2: DC motor connection

Conclusion : We learnt the working of a DC motor controlled by Arduino.

6. Write a server application to be deployed on Raspberry-Pi /Beagle board. Write client applications to get services from the server application

Problem Statement: Building a client server application using Raspberry Pi.

Hardware Requirements:

1. Raspberry Pi Board
2. DHT-11 sensor

Software Requirements:

1. Raspberrian OS (IDLE)

Theory :

Sockets: Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.
They are the real backbones behind web browsing. In simpler terms, there is a server and a client.

Socket programming is started by importing the socket library and making a simple socket.

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Here we made a socket instance and passed it two parameters. The first parameter is **AF_INET** and the second one is **SOCK_STREAM**. AF_INET refers to the address-family ipv4. The SOCK_STREAM means connection-oriented TCP protocol.

Now we can connect to a server using this socket.3

Server Socket Methods :

A server has a bind() method which binds it to a specific IP and port so that it can listen to incoming requests on that IP and port. A server has a listen() method which puts the server into listening mode. This allows the server to listen to incoming connections. And last a server has an accept() and close() method. The accept method initiates a connection with the client and the close method closes the connection with the client.

Client Socket Methods :

The client in the client-server architecture requests the server and receives services from the server. For this, there is only one method dedicated for clients − connect(address) : this method actively intimate server connection or in simple words this method connects the client to the server. The argument address represents the address of the server.

General Socket Methods:

Other than client and server socket methods, there are some general socket methods, which are

very useful in socket programming. The general socket methods are as follows −

- socket.recv(bufsize) − As name implies, this method receives the TCP message from socket. The argument bufsize stands for buffer size and defines the maximum data this method can receive at any one time.
- socket.send(bytes) − This method is used to send data to the socket which is connected to the remote machine. The argument bytes will gives the number of bytes sent to the socket.
- socket.recvfrom(data, address) − This method receives data from the socket. Two pair (data, address) value is returned by this method. Data defines the received data and address specifies the address of socket sending the data.
- socket.sendto(data, address) − As name implies, this method is used to send data from the socket. Two pair (data, address) value is returned by this method. Data defines the number of bytes sent and address specifies the address of the remote machine.
- socket.close() − This method will close the socket.
- socket.gethostname() − This method will return the name of the host.
- socket.sendall(data) − This method sends all the data to the socket which is connected to a remote machine. It will carelessly transfers the data until an error occurs and if it happens then it uses socket.close() method to close the socket.

Program for temperature server:

```
import socket
import Adafruit_DHT
s=socket.socket()
s.bind("192.168.43.164",1234)
s.listen(5)
while True:
        print "Waiting for client connection…"
        c,addr=s.accept()
        hum,temp=Adafruit_DHT.read_retry(11,4)
        print "Got connection from address", addr
        c.send(str(temp))
        c.close()
```

Program for Temperature Client:

```
import socket
s=socket.socket()
s.connect (("192.168.43.164",1234))
print " The temperature is :",s.recv(1024
s.close()
```

Temperature Server UDP:

```
import socket,time
Import Adafruit_DHT
sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
while True:
        hum,temp=Adafruit_DHT.read_retry(11,4)
```

```
        print “Broadcasting..”,temp
        sock.sendto(str(temp),(‘192.168.43.164’,1234))
        time.sleep(1)
```

Temperature Client UDP:

```
import socket
sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
sock.bind((‘192.168.43.164’,1234))
While True:
        data,addr=sock.recvfrom(100)
        print “The temperature is:”, data, “from”, addr
```

7. Real Time Intrusion Detection for Smart Home

Problem Statement: Understanding the connectivity of Raspberry-Pi /Beagle board circuit with IR sensor. Write an application to detect obstacle and notify user using LEDs**.**

Objective:

1. **To understand the connectivity of Raspberry-Pi circuit with IR sensor**
2. **To Understanding how to read the Interference of object**
3. **To Understanding how to switch on the LED on detecting interference**

Theory:

IR Sensor

- An infrared sensor is an electronic instrument which is used to sense certain characteristics of its surroundings by either emitting and/or detecting infrared radiation.
- Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion

• Infrared waves are not visible to the human eye. In the electromagnetic spectrum, infrared radiation can be found between the visible and microwave regions.

• The infrared waves typically have wavelengths between 0.75 and 1000μm.

Figure 1: IR Sensor

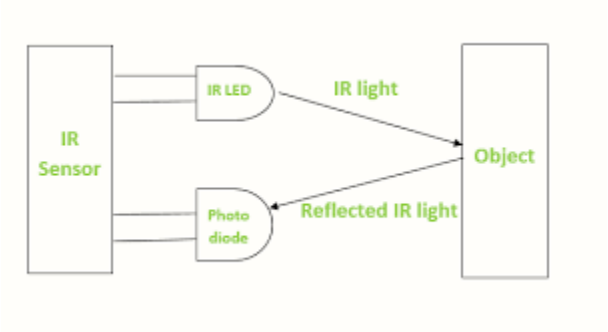


Figure 2: Working of IR sensor

IR Sensor structure

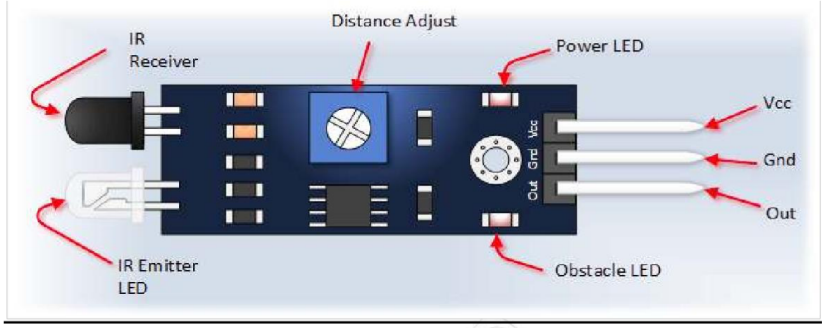


Figure 3: IR sensor structure

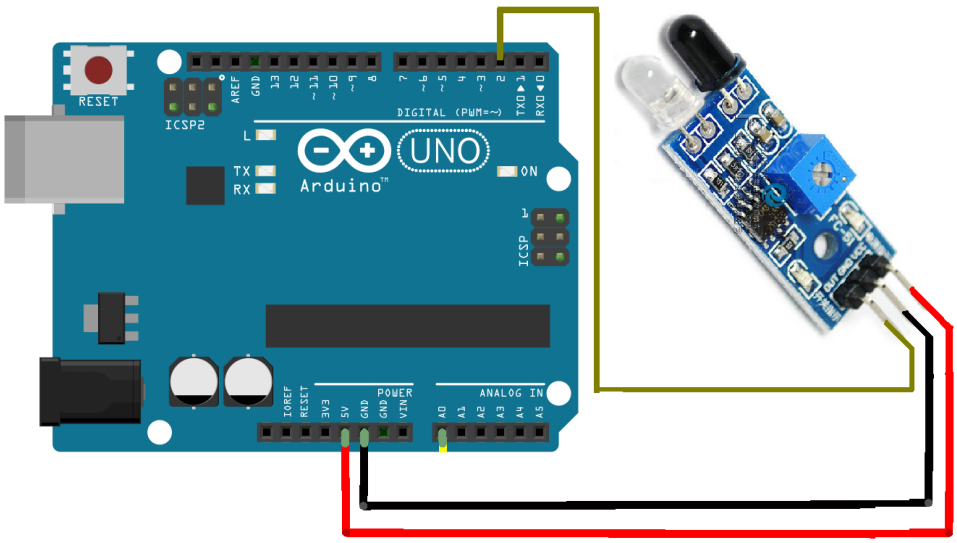


Figure 4: IR sensor connection

Piezo Buzzer: Piezo buzzers are used for making beeps alarms and tones. They can be used in alarm systems, for keypad feedback, or some games. Light weight, simple construction and low price make it usable in various applications like car/truck reversing indicator, computers, call bells etc. There are two types of buzzer: active and passive. In the active model, buzzer starts playing sound only if connected to a power supply, but in the passive model, we need to send a pulse from the microcontroller to play sounds.

The supply voltage of the buzzer is 3 volts, 5 volts and 12 volts. Buzzer has 2 pins: DATA and GND but the buzzer module has an extra VCCpin.

Figure 5: Buzzer

PROCEDURE:

STEPS to setup the circuit:

1) Use the TECHGRAYLOGIX universal development board.
2) Connect IR sensor to DENSE 1 slot
3) Connect Buzzer to the Buzzer slot
4) Place the jumper wires as per the specification mentioned in the sketch.
5) Choose the IR sensor obstacle detection sketch from the sample code.
6) Compile and upload the code to arduino uno
7) Test the circuit by placing obstacle in front of IR sensor and buzzer must turn on when there is obstacle.

Conclusion: IR sensor will sense the obstacle and buzzer will be turned ON on detection of obstacle. Working of IR sensor and buzzer in arduino kit is observed.

8. Create a small dashboard application to be deployed on cloud.

Problem Statement: Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe .

Arduino Uno +ESP8266 Connectivity with Cloud

- The IoT platforms are suites of components those help to setup and manage the internet connected devices
- A person can remotely collect data, monitor and manage all internet connected devices from a single system.
- There are a bunch of IoT platforms available online but building an IoT solution for a company is all depend on IoT platform host and support quality.
  IOT Cloud Platforms
  - Kaa IoT Platform
  - SiteWhere: Open Platform for the Internet of Things
  - ThingSpeak: An open IoT platform with MATLAB analytics
  - DeviceHive: IoT Made Easy
  - Zetta: API-First Internet of Things Platform
  - DSA: Open Source Platform & "Toolkit" for Internet Of Things Devices
  - Thingsboard.io Open-source IoT Platform
  - Thinger.io: The Opensource Platform for Internet of things
  - WSo2- Open source platform for Internet of Things and mobile projects

ThingSpeak:

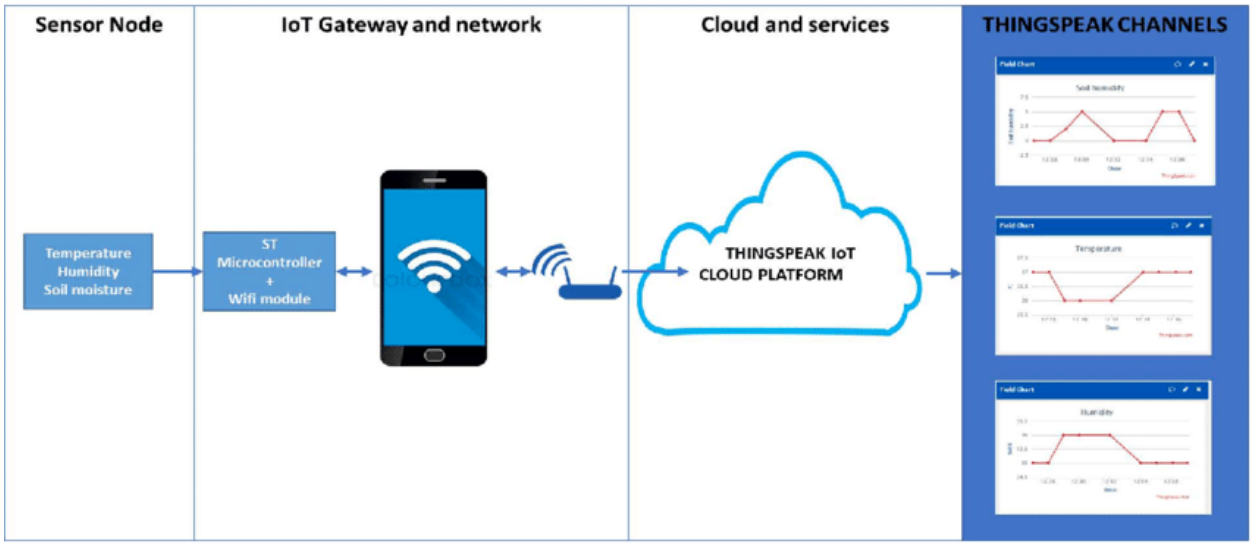


Figure 1: ThingSpeak Cloud Service

ThingSpeak – Features

- Collect data in private channels
- Share data with public channels
- RESTful and MQTT APIs

- MATLAB analytics and visualizations
- Alerts
- Event scheduling
- App integrations
- Worldwide community

NOTE: We need to signup and signin into ThingSpeak (Mathwork account) to obtain the services. We shall use gmail id login.

Channels in ThingSpeak:

A channel is where you send your data to store. Each channel includes 8 fields for any type of data, 3 location fields, and 1 status field. Once you have a ThingSpeak Channel you publish data to the channel, have ThingSpeak process the data, then have your application retrieve the data.

There are public channels available. We can create our own channels for publishing the data from our project.

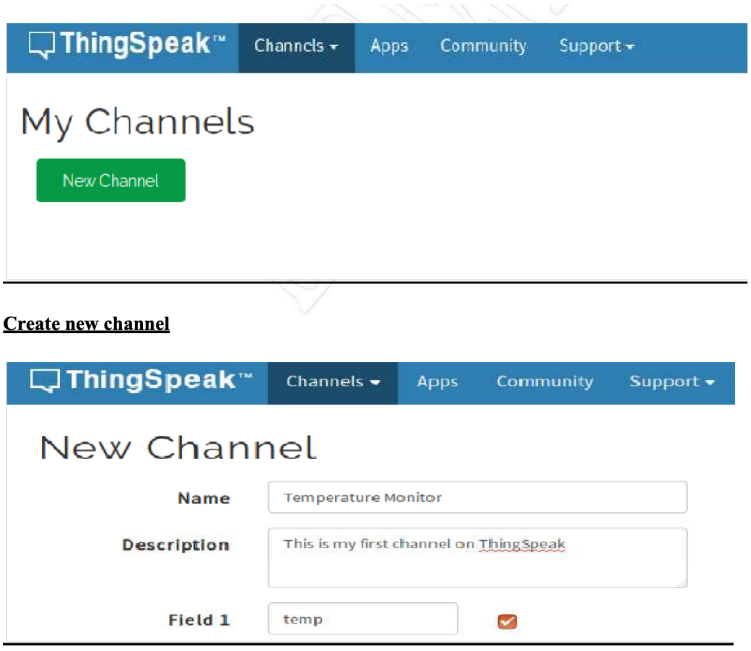
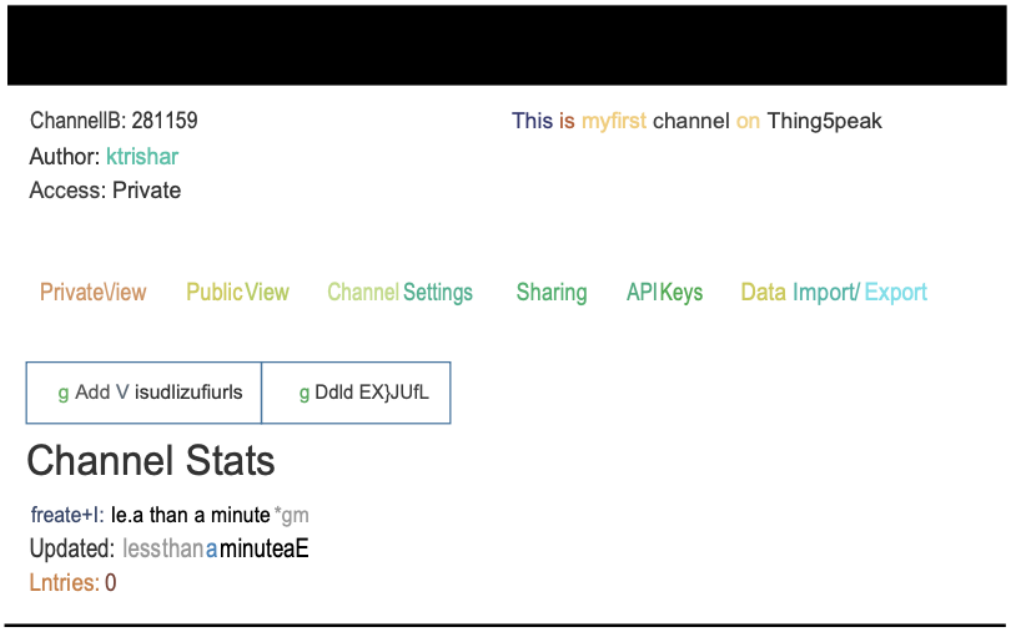


Figure 2: ThingSpeak Channel

First time channel view



Make your channel public

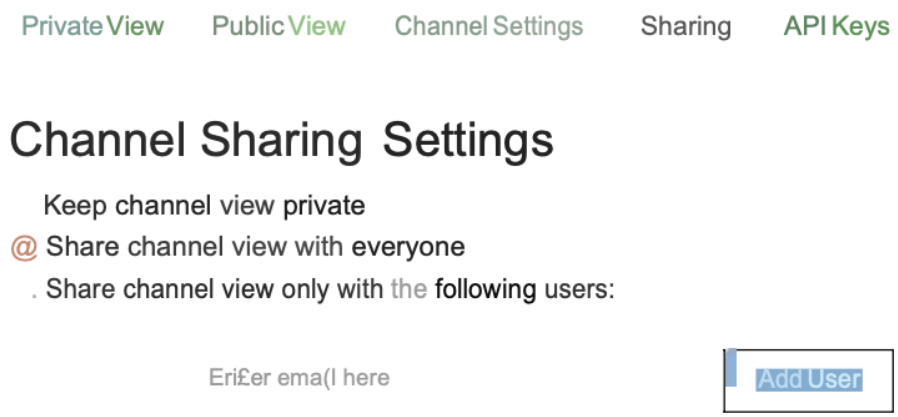
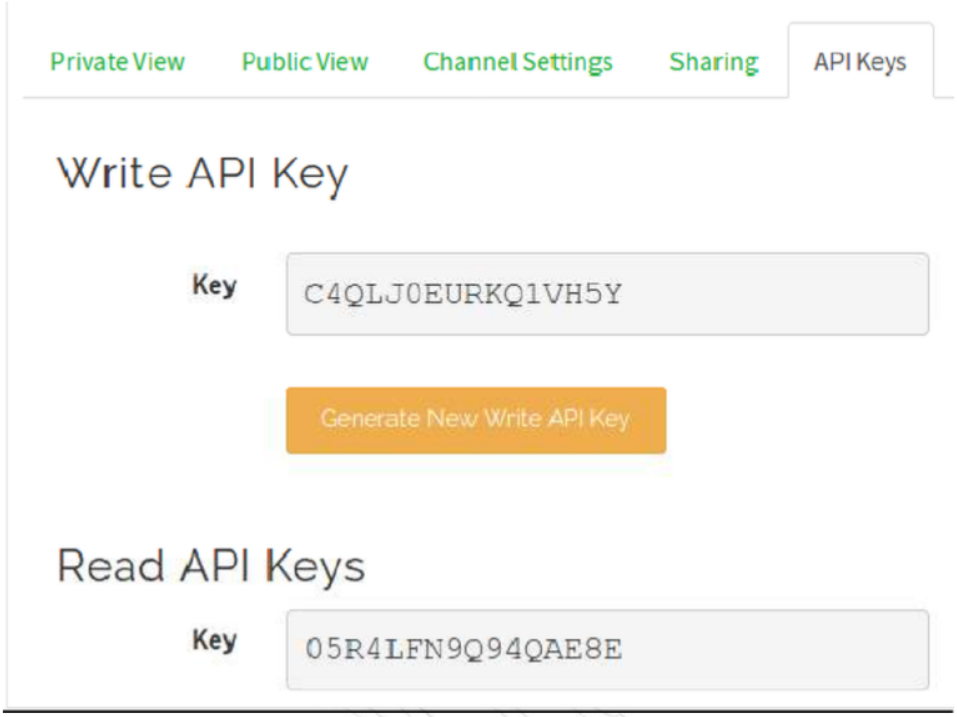


Figure 3: Channel view and settings

The API keys for RW operations



Design IOT App

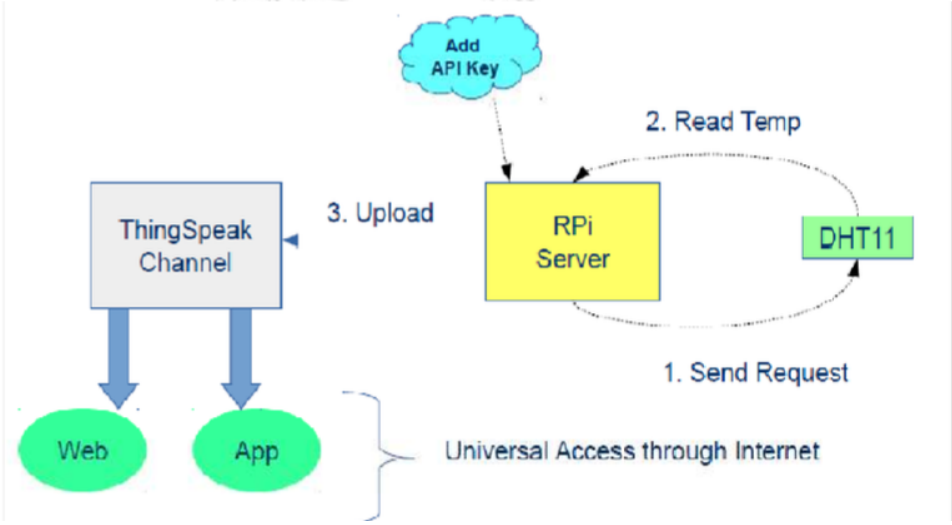


Figure 4: API Keys

API key is required for binding with the channel we created.

We use Write API key to feed data into the channel (write data to channel). For ex: In this example we can send Temperature data to thingspeak channel we created.

Do this in Arduino Board:

Step 1: In Techgraylogix Universal Development Kit, we use DHT11 sensor and ESP8266 module.

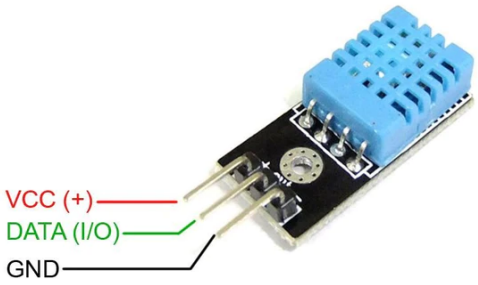


Fig 5: DHT11 sensor

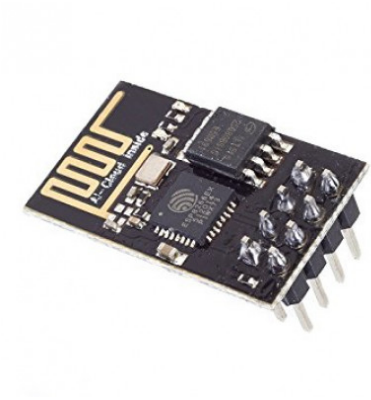


Fig 6: ESP8266 Module

Step 2: Place the DHT11 sensor on LM35 slot in the board and place ESP8266 module into wifi slot.

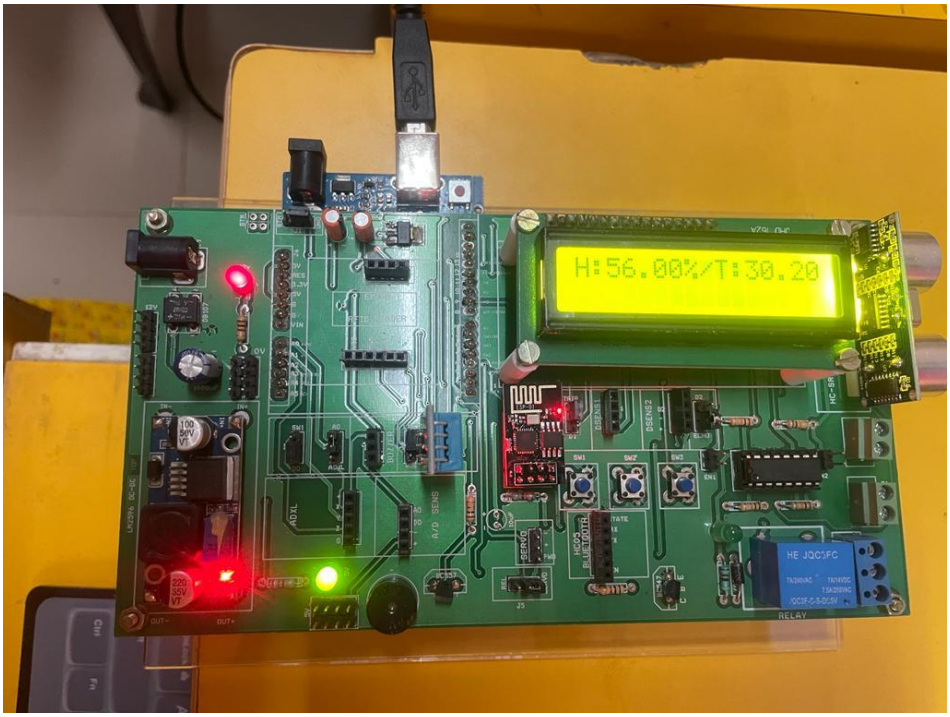


Fig 7: DHT11 and ESP8266 module in Arduino Development Board

Step 3: Open IoT Development Kit folder in C drive of lab PC. Got to IoT ThingSpeak folder. Open the the sketch. Copy and paste the following code into the sketch area (Relace the entire code with this new code:

```
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>

#include "DHT.h"
#define DHTPIN A3     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

SoftwareSerial esp8266(2, 8); // RX, TX

const int startswitchPin = 6;
int startswitchState = 0;

const int fanPin = 3;     // the number of the pushbutton pin
int fanState = 0;

//const int irPin = 13;
//int irState = 0;

LiquidCrystal_I2C lcd(0x27, 16, 2);

int irValue;

#define DEBUG true
#define IP "184.106.153.149"// thingspeak.com
String Api_key = "GET /update?api_key=8JYMP3RN09FCO508";    //change it with your api key like "GET /update?key=Your Api Key"   in thingspeak channel

int error;

//=======================
float h;
float t;
float f;

void setup()
{
  //==================
  lcd.begin(16, 2);
  lcd.backlight();
```

```
  lcd.clear();
  lcd.print("IoT-DEVELOPMENT");
  lcd.setCursor(0, 1);
  lcd.print(" - THINGSPEAK ");
  delay(3000);
  Serial.begin(9600);

  //=====================
  // pinMode(startswitchPin, INPUT_PULLUP);
  //pinMode(irPin, INPUT_PULLUP);
  pinMode(fanPin, OUTPUT);
  // Serial.begin(9600);
  //=======================================

  send_command("AT+RST\r\n", 2000, DEBUG); //reset module
  send_command("AT+CWMODE=1\r\n", 1000, DEBUG); //set station mode
  send_command("AT+CWJAP=\"WIFINETWORKNAME\",\"WIFIPASSWORD\"\r\n", 2000,
DEBUG);
  //=====================================


  dht.begin();
  esp8266.begin(9600);
}

void loop()
{
label0:
  lcd.clear();
  lcd.print("PRESS SWITCH TO");
  lcd.setCursor(0, 1); // Sets cursor at Column 0 and Line 1
  lcd.print("CHECK THE DATA...");
  delay(200);

  startswitchState = digitalRead(startswitchPin);
  if (startswitchState == LOW)
  {
    lcd.clear();
    lcd.print("CHECKING DATA.....");
    delay(3000);
    goto label1;
  }

  goto label0;
```

```
label1:
  fanState = HIGH;
  digitalWrite(fanPin, fanState);
  delay(6000);

  delay(1000);
  h = dht.readHumidity();
  delay(2000);
  t = dht.readTemperature();
  // f = dht.readTemperature(true);
  if (isnan(h) || isnan(t)) // || isnan(f))
  {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  lcd.clear();
  lcd.print("H:");
  lcd.print(h);
  lcd.print("%");
  lcd.print("/T:");
  lcd.print(t);
  //lcd.print("C");

  delay(4000);
  //thingSpeakWrite(t, h, mq3Value, mq5Value, mq5Value);
  updatedata();
  delay(5000);
  fanState = LOW;
  digitalWrite(fanPin, fanState);
  goto label0;
}

//=======================
void updatedata() {
  String command = "AT+CIPSTART=\"TCP\",\"";
  command += IP;
  command += "\",80";
  Serial.begin(9600);
  delay(200);
  esp8266.begin(9600);
  delay(200);
  Serial.println(command);
  delay(200);
  esp8266.println(command);
```

```
 delay(2000);
 if (esp8266.find("Error")) {
  return;
 }
 command = Api_key ;
 command += "&field1=";
 command += String(t);
 //command += "&field2=";
 //command += String(h);
 //command += "&field3=";
 //command += String(irValue);
 // command += "&field4=";
 //  command += String(mq5Value);
 // command += "&field5=";
 // command += String(mq5Value);
 command += "\r\n\r\n\r\n\r\n\r\n\r\n";
 Serial.println(command);
 Serial.print("AT+CIPSEND=");
 esp8266.print("AT+CIPSEND=");
 Serial.println(command.length());
 esp8266.println(command.length());


 if (esp8266.find(">")) {
  delay(200);
  Serial.print(command);
  delay(200);
  esp8266.print(command);
  delay(200);
  delay(5000);
  esp8266.println("AT+RST");
  delay(200);
  esp8266.println("AT");
  delay(200);
  esp8266.println("AT");
  delay(200);
 }


 else {

  Serial.println("AT+CIPCLOSE");
  esp8266.println("AT+CIPCLOSE");
  //Resend...
  error = 1;
```

```
  }
}

String send_command(String command, const int timeout, boolean debug)
{
  esp8266.begin(9600);
  String response = "";
  esp8266.print(command);
  long int time = millis();
  while ( (time + timeout) > millis())
  {
    while (esp8266.available())
    {
      char c = esp8266.read();
      response += c;
    }
  }
  if (debug)
  {
    Serial.print(response);
  }
  return response;
}
```

Step 4: Compile and upload the code. Make sure that your personal hotspot is ON and esp8266 module connects to it. Once it is connected, the Temperature will be written into the channel.

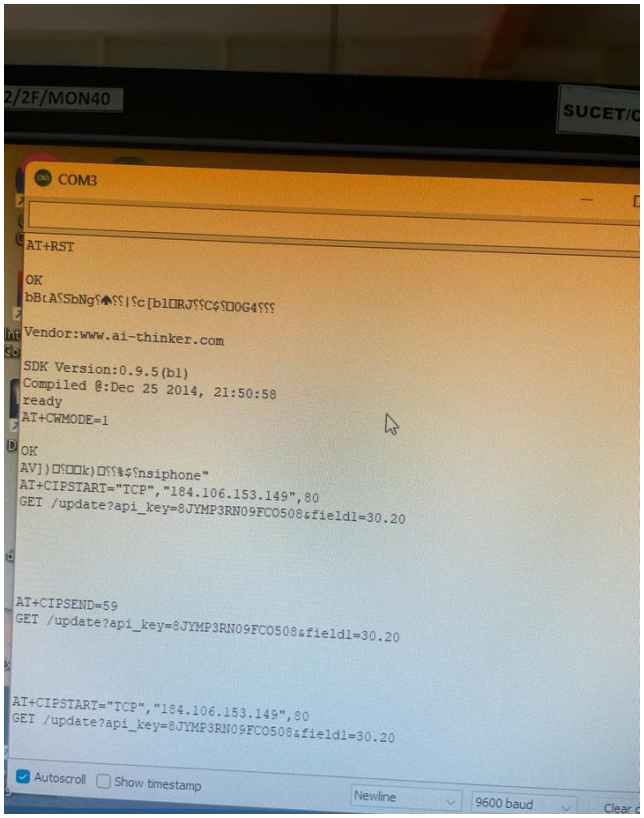


Fig 8 : Output in serial monitor

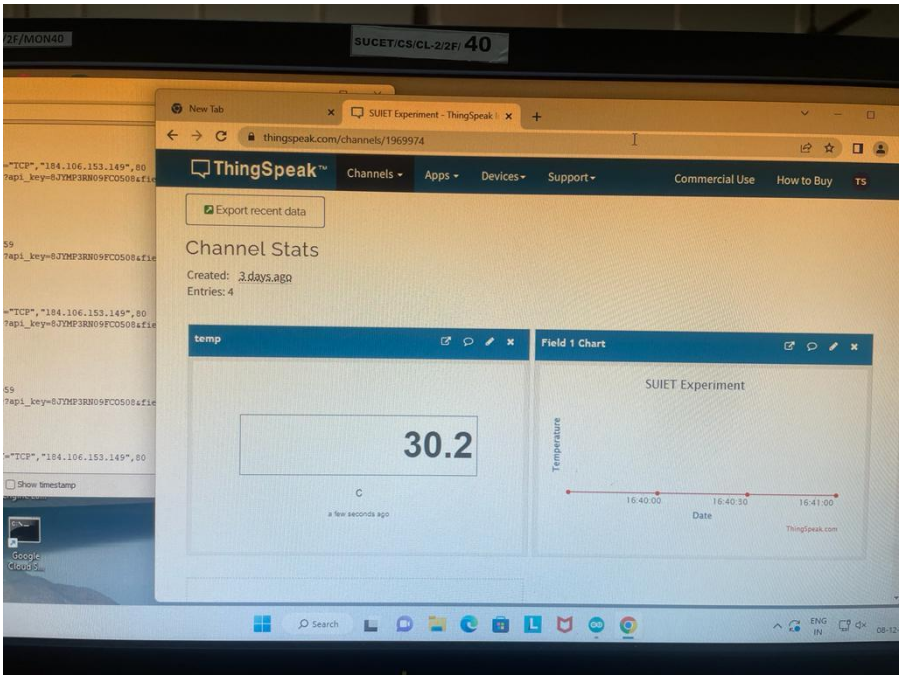


Fig 9: Temperature Data recorded in ThingSpeak

Conclusion: We learned to create a small dashboard application to be deployed on cloud.

Extra Programs:

1. **IoT Parking Sensor Application: Using Ultrasonic Sensor and Buzzer (Instead of RPi camera)**
2. **Understanding actuation using IR sensor and Stepper Motor. (Instead of 6th program Client Server in RPi which is similar to 8th program Thingspeak communication)**