

ARDUINO-BASICS

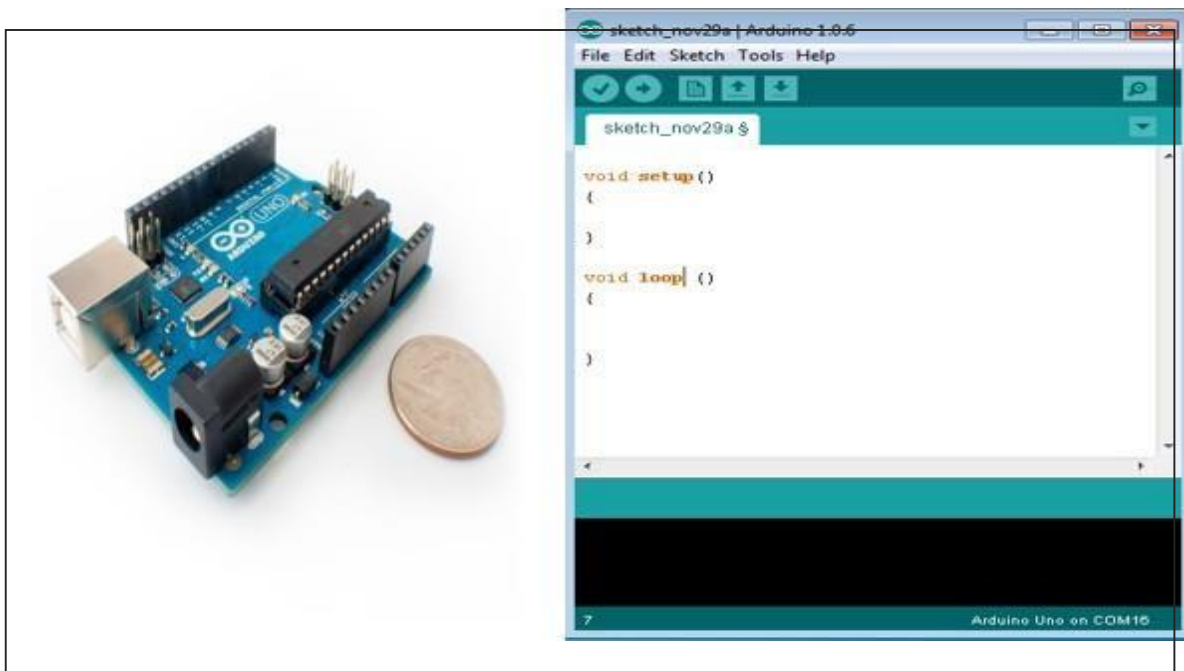
ARDUINO-OVERVIEW

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

The key features are:

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board



Board Types

Various kinds of Arduino boards are available depending on different microcontrollers used. However, all Arduino boards have one thing in common: they are programmed through the Arduino IDE.

The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface (hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V.

Here is a list of different Arduino boards available.

Arduino boards based on ATMEGA328 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino UnoR3	5V	16MHz	14	6	6	1	USB via ATMegal6U2
Arduino UnoR3 SMD	5V	16MHz	14	6	6	1	USB via ATMegal6U2
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro 3.3v/8 MHz	3.3V	8 MHz	14	6	6	1	FTDI-Compatible Header
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino mini05	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 3.3v/8mhz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 5v/16mhz	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Ethernet	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino Fio	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
LilyPad Arduino 328 main board	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
LilyPad Arduino simply board	3.3V	8MHz	9	4	5	0	FTDI-Compatible Header

Arduino boards based on ATMEGA32u4 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1	Native USB
Pro micro 3.3V/8MHz	5V	16MHz	14	6	6	1	Native USB
LilyPad Arduino USB	3.3V	8MHz	14	6	6	1	Native USB

Arduino boards based on ATMEGA2560 microcontroller

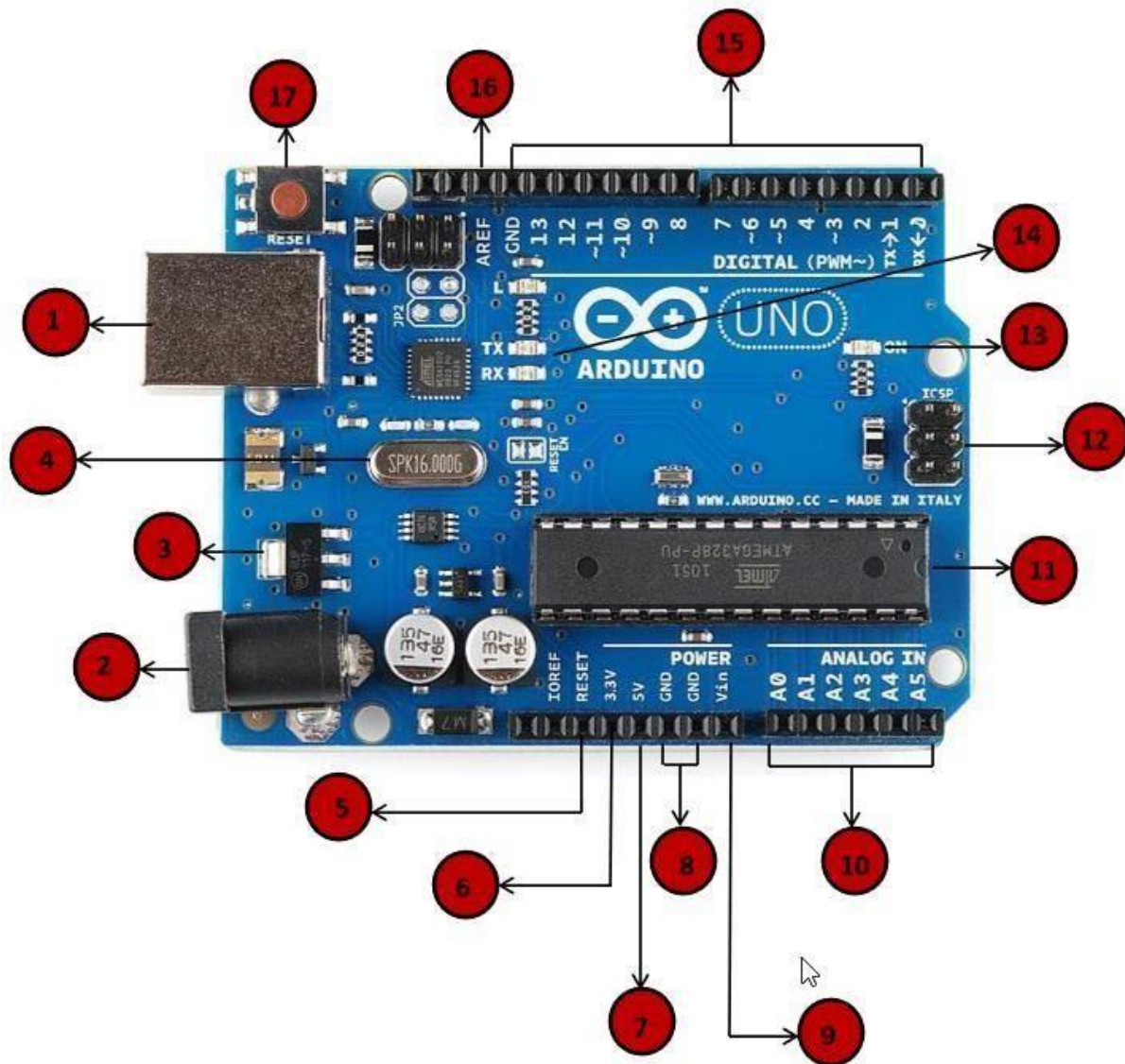
Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560R3	5V	16MHz	54	16	14	4	USB via ATmega16 U2
Mega Pro3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatibl eHeader
Mega Pro5V	5V	16MHz	54	16	14	4	FTDI-Compatibl eHeader
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatibl eHeader

Arduino boards based on AT91SAM3X8E microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Due	3.3V	84MHz	54	12	12	4	USB native

ARDUION-BOARD DISCRIPTION

In this chapter, we will learn about the different components on the Arduino board. We will study the Arduino UNO board because it is the most popular board in the Arduino board family. In addition, it is the best board to get started with electronics and coding. Some boards look a bit different from the one given below, but most Arduinos have majority of these components in common.



1. Power USB

Arduino board can be powered by using the USB cable from your computer.

All you need to do is connect the USB cable to the USB connection (1).

2. Power (Barrel Jack)

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

3. Voltage Regulator

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

4. Crystal Oscillator

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

5. Arduino Reset

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

6. Pins (3.3, 5, GND, Vin)

- 3.3V (6): Supply 3.3 output volt

7. 5V (7): Supply 5 output volt

- Most of the components used with Arduino board works fine with 3.3 volt and 5volt.

8. GND (8) (Ground):

There are several GND pins on the Arduino, any of which can be used to ground your circuit.

9. Vin (9):

This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

10. Analog pins:

The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor

11. Main microcontroller

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

12. ICSP pin

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

13. Power LED indicator

This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

14. TX and RX LEDs

On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led.

15. Digital I / O:

The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM

16. AREF

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins

ARDUINO-INSTALLATION

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1: First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer.

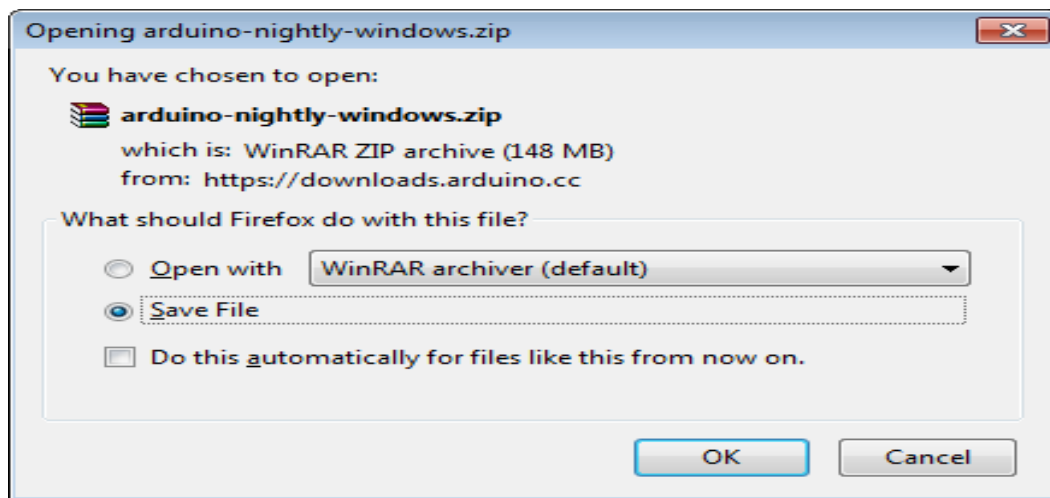


In case you use Arduino Nano, you will need an A to Mini-B cable.



Step 2: Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



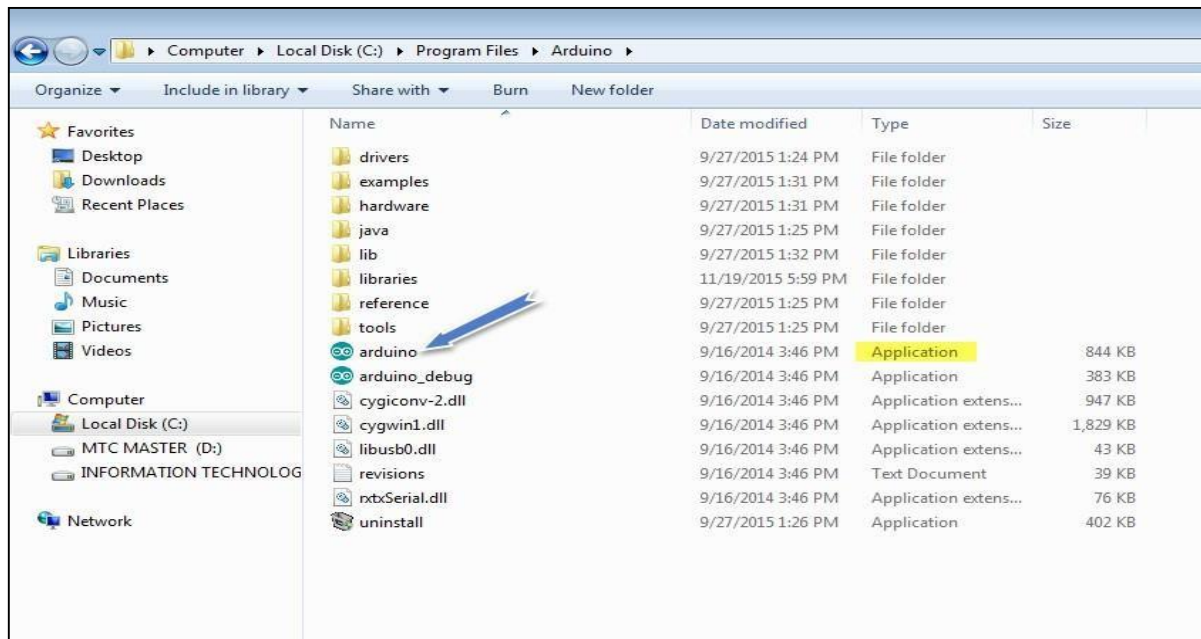
Step 3: Power up your board;

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4: Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE

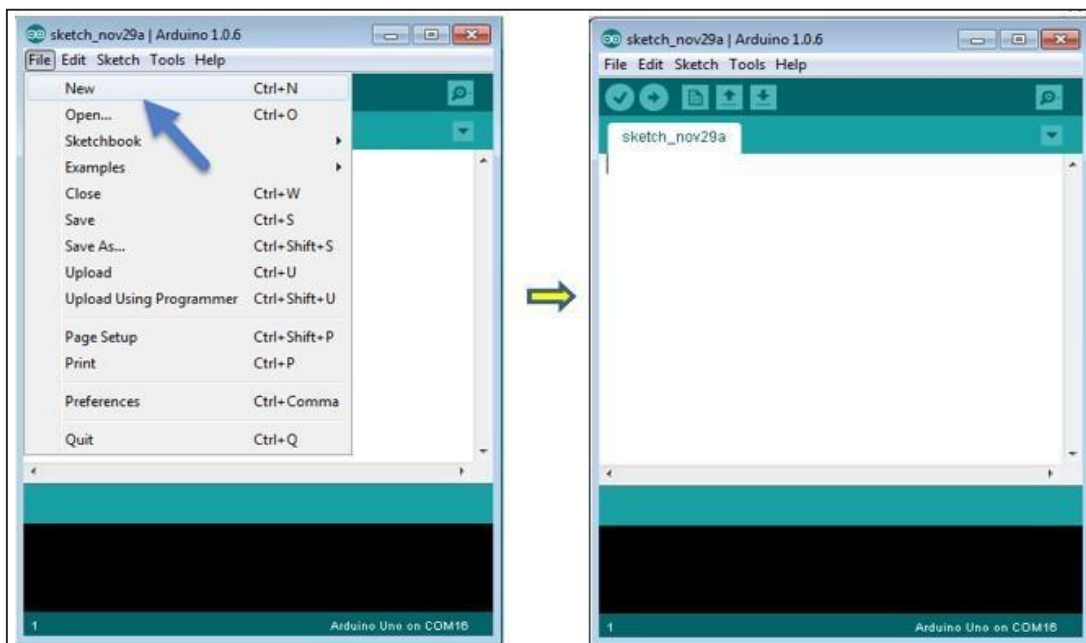


Step 5: Open your first project.

Once the software starts, you have two options:

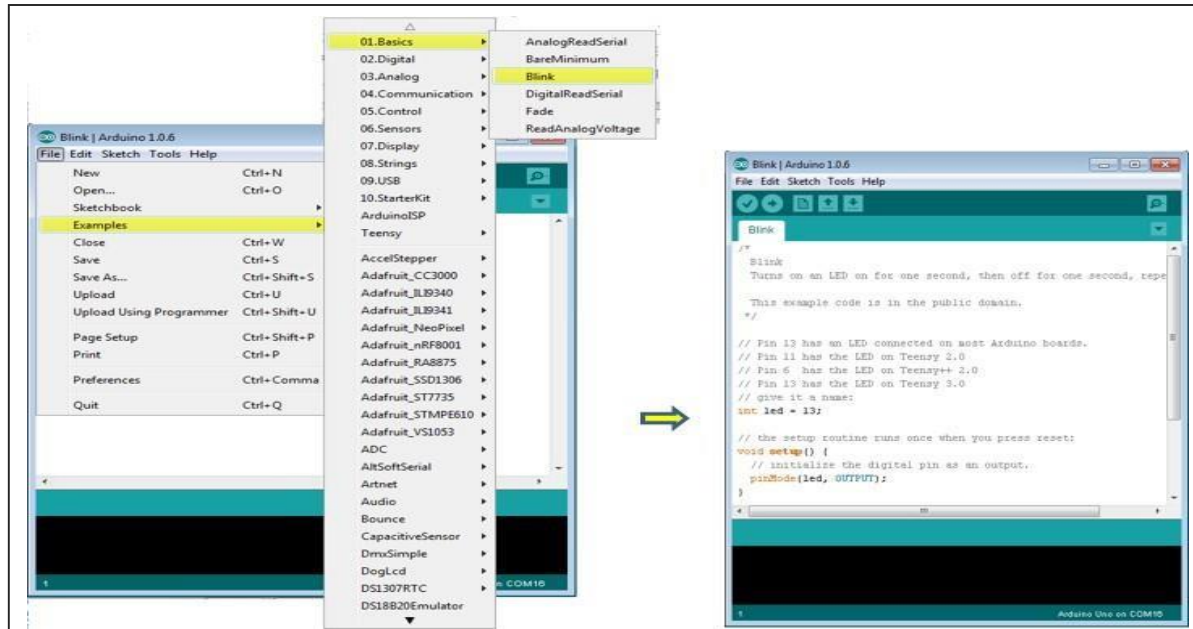
- Create a new project.
- Open an existing project.

To create a new p to create a new project, select File --> New.



To create a new project, select File --> New

To open an existing project example, select File -> Example -> Basics -> Blink.
project, select File --> New



Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6: Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools -> Board and select your board.

What is a Raspberry Pi?

A credit-card-sized computer Runs on sell operating system Raspbian we will use RASPBIAN Windows 10 lot core Retro Pie Open Elec Posverasible to connect with a variety of sensors to interact with the physical world. Make decisions based on processing of gathered sensor data.



Figure 1: Raspberry Pi Board

it is an enhanced motherboard developed in the United Kingdom by the Raspberry Pi foundation, now widely accepted as a part of evolving computer technology. The minicomputer can connect with other peripheral hardware devices such as a keyboard, mouse, and monitor.

Raspberry Pi to learn programming skills, build hardware projects, do home automation, implement Kubernetes clusters and Edge computing, and even use them in industrial applications

Getting started with your Raspberry Pi

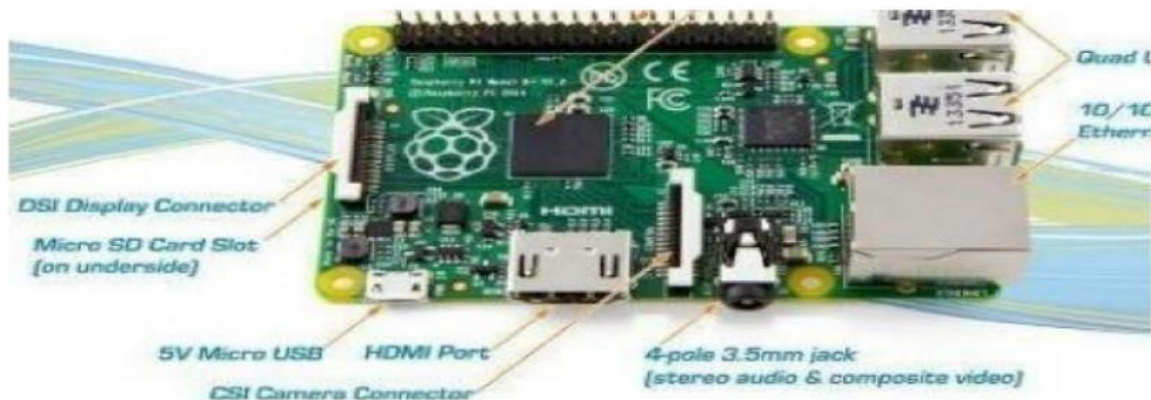


Figure 2: the Raspberry Pi 3 Model

Basic 1-2-3

1. Connect the Raspberry Pi to the keyboard, mouse and the screen.
2. Plug in the power and wait for it to boot.
3. Username pi Password raspberry (should not be needed).

Communicating through the terminal

Start the terminal by going to Accessories and then clicking on the Terminal program. Run through the commands and get a feel for how to change directories, and list its contents.

Basic Command's cheat-sheet	
Command	Description
ls	List contents of directory
cd <dir>	Change to directory. <dir>:t
mkdir <dir>	Make a new directory called <dir>
rmdir <dir>	Delete the empty directory, <dir>
rm, <file>	Delete the file, <file>
nano	Run the <i>nanotext</i> editor program

Figure 3: Commands

GPIO Layout

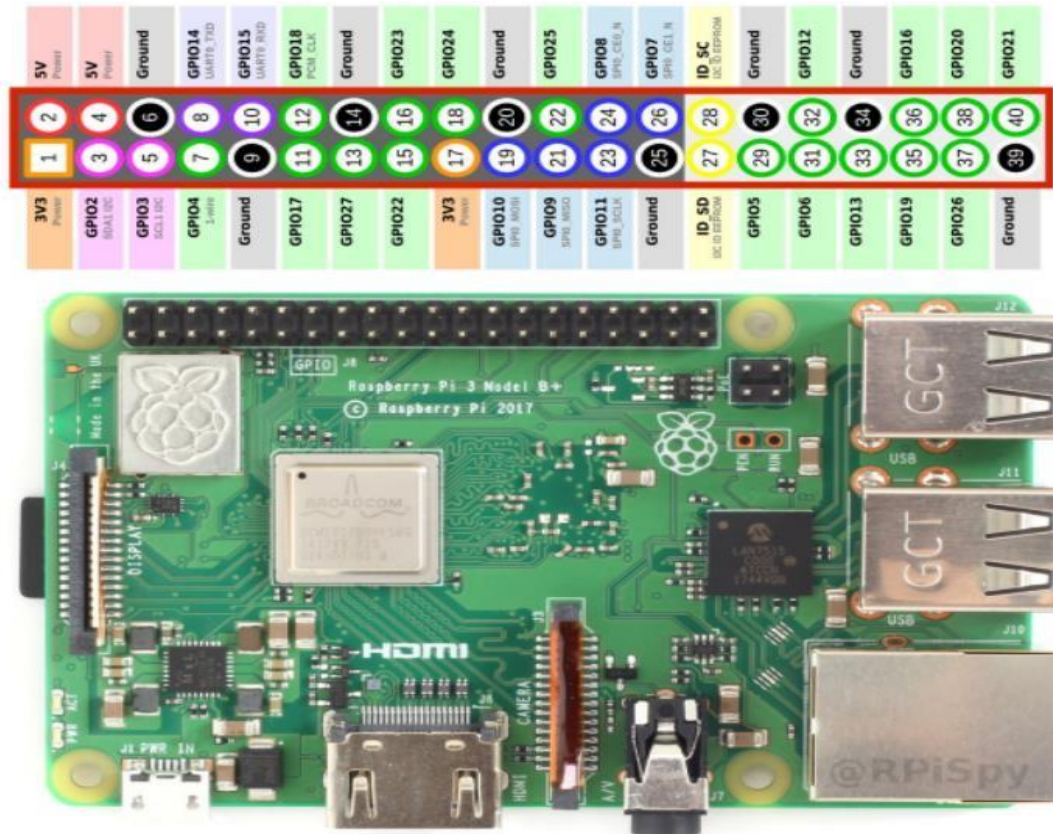


Figure 4: Raspberry Pi 40-pin GPIO Layout

Programming in Python

General-purpose programming language used for scientific and numerical applications.

Open source language that has a lot of online resources for problems you might come across.

We are using Thorny IDE for programming with Python3 in our Raspberry Pi 3

<https://stackoverflow.com> is your best friend for programming troubles.

Hardware PWM is available on these pins only: GPIO12, GPIO13, GPIO18, GPIO19.



Figure 5: Commands

```

24 try:
25     while 1:
26         #NOT PRESSED
27         if GPIO.input(butPin):
28             pwm.ChangeDutyCycle(duty)
29             GPIO.output(ledPin, GPIO.LOW)
30
31         #PRESSED
32         else:
33             GPIO.output(ledPin, GPIO.HIGH)
34             pwm.ChangeDutyCycle(duty)
35             time.sleep(0.5)
36             GPIO.output(ledPin, GPIO.LOW)
37             pwm.ChangeDutyCycle(100-duty)
38             time.sleep(0.5)
39
40 except KeyboardInterrupt:
41     pwm.stop()
42     GPIO.cleanup()
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```

Use the GPIO drive outputs and read inputs.

Common errors and debugging

Python is sensitive to indentation. Make sure that the variables and the function have the same exact . Watch out for small and capital letters. Remember that python starts counting from zero. i.e. when using loops.

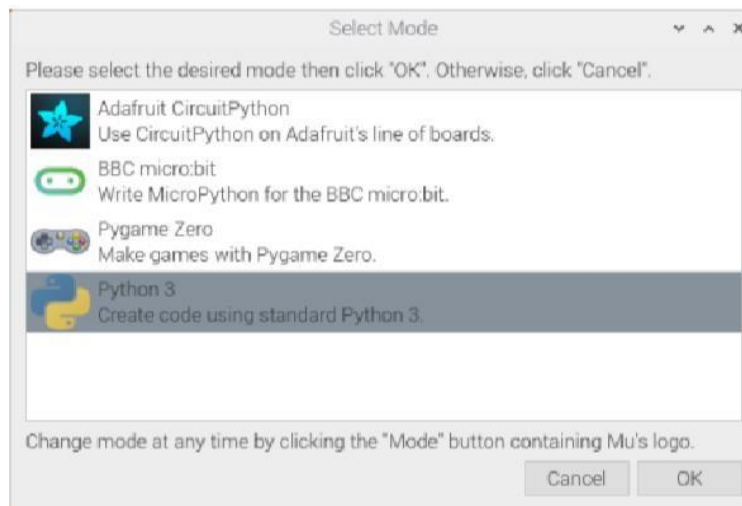
Switching an LED on and off

GPIO Zero is a new Python library which provides a simple interface to everyday GPIO components. It comes installed by default in Raspbian.

- Open Mu.



Then choose the mode in which you want to use Mu. Choose Python 3 if you are creating a new Python script.



First import the GPIO Zero library and tell the Pi which GPIO you are using in this case pin 17.

```
In [1]: from gpiozero import LED
In [2]: led = LED(17)
```

Press **Enter** on the keyboard.

To make the LED switch on, type the following and press **Enter**:

```
In [3]: led.on()
```

To make it switch off you can type:

```
In [4]: led.off()
```

Your LED should switch on and then off again. But that's not all you can do.

Flashing an LED

With the help of the time library and a little loop, you can make the LED flash.

```
from gpiozero import LED
from time import sleep

led = LED(17)

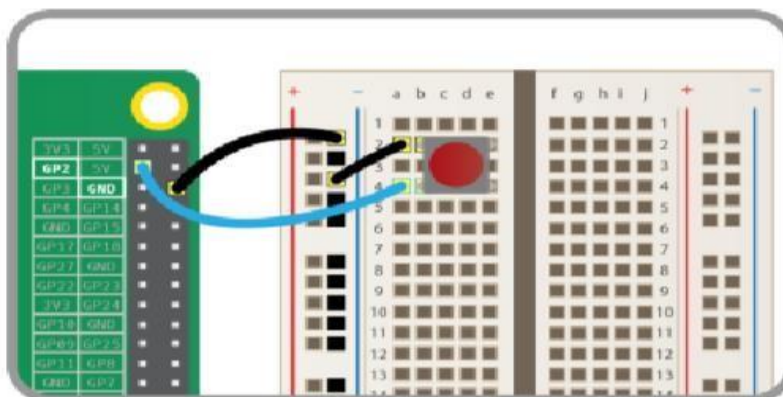
while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

Using buttons to get input

```
from gpiozero import Button
button = Button(2)
```

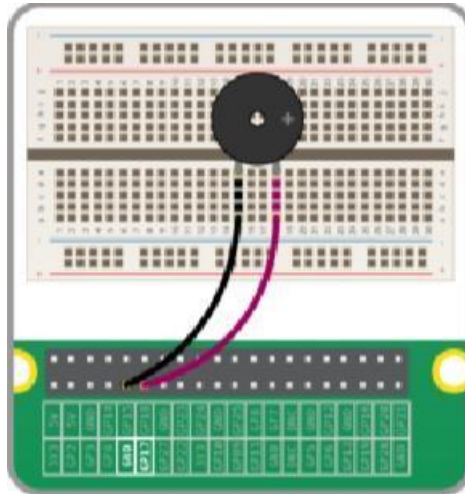
Now you can get your program to do something when the button is pushed. Add these lines:

```
button.wait_for_press()
print('You pushed me')
```



Using a Buzzer

There are two main types of buzzer: Active and passive. A passive buzzer emits a tone when a voltage is applied across it. It also requires a specific signal to generate a variety of tones. The active buzzers are a lot simpler to use, so these are covered here.



Add **Buzzer** to the `from gpiozero import...` line:

```
from gpiozero import Buzzer
from time import sleep
```

Add a line below your creation of **button** and **lights** to add a **Buzzer** object:

```
buzzer = Buzzer(17)
```

In GPIO Zero, a **Buzzer** works exactly like an **LED**, so try adding a `buzzer.on()` and `buzzer.off()` into your loop:

```
while True:
    buzzer.on()
    sleep(1)
    buzzer.off()
    sleep(1)
```

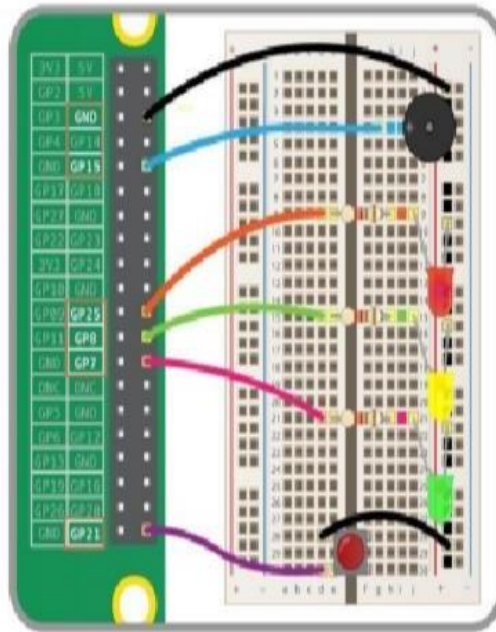
A **Buzzer** has a `beep()` method which works like an **LED**'s `blink`. Try it out:

```
while True:
    buzzer.beep()
```

Making a traffic light

The manufacturers of traffic light systems occur in 3 stages: making signal light heads, making the controller and making the supports.

```
1.from gpiozero import Button, TrafficLights, Buzzer
2.from time import sleep
3.
4.buzzer = Buzzer(15)
5.button = Button(21)
6.lights = TrafficLights(25, 8, 7)
7.
8.while True:
9.    button.wait_for_press()
10.    buzzer.on()
11.    light.green.on()
12.    sleep(1)
13.    lights.amber.on()
14.    sleep(1)
15.    lights.red.on()
16.    sleep(1)
17.    lights.off()
18.    buzzer.off()
```



Using a sensor with the Raspberry Pi

The GPIO in the Raspberry Pi can only handle a maximum of 3.3v, but many of the available sensors can return a signal of 5v to the GPIO. This will damage the Raspberry Pi.

How do we deal with this?

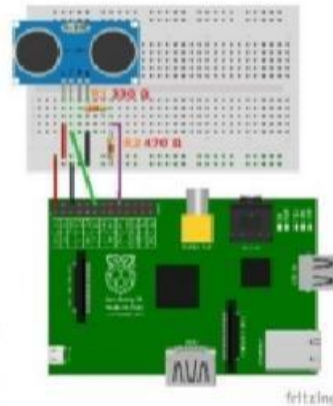
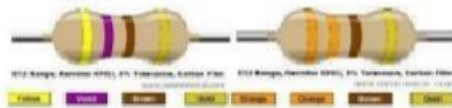
Resistor connected as voltage dividers.

Important to be aware of this.

Measuring distance using Ultrasonic Sensor

Measuring Distance with a Ultrasonic Sensor: HC-SR04

- The Trigger sends a burst of sound that bounces and hits the Echo.
- If we measure the time it takes from sending to receiving we can calculate the distance.
- NOTE: Two different resistors here



Setting up the Raspberry Pi

Setting up the Raspberry Pi



The distance function

Now we need a function to calculate the distance.

With the sensor we can measure the time it takes for a signal to be sent out and reflected back.

We need to convert the elapsed time to distance by using the speed of sound in air and multiplying it with the time divided by two (Because it travels to the object, and back).

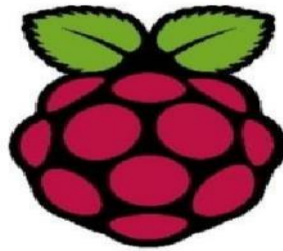
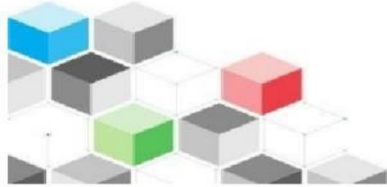
The distance function

```
17 def distance():
18     # set Trigger to HIGH,
19     GPIO.output(GPIO_TRIGGER, True)
20
21     # set Trigger after 0.01ms to LOW
22     # this sends a signal that the ECHO can read in
23     time.sleep(0.00001)
24     GPIO.output(GPIO_TRIGGER, False)
25
26     StartTime = time.time()
27     StopTime = time.time()
28
29     # save StartTime
30     while GPIO.input(GPIO_ECHO) == 0:
31         StartTime = time.time()
32
33     # save time of arrival
34     while GPIO.input(GPIO_ECHO) == 1:
35         StopTime = time.time()
36
37     # time difference between start and arrival
38     TimeElapsed = StopTime - StartTime
39     # multiply with the sonic speed (34300 cm/s)
40     # and divide by 2, because: there and back
41     distance = (TimeElapsed * 34300) / 2
42
43     #Return the Distance that the sensor measured
44     return distance
```

Making it loop until we tell it to stop

```
47 try:
48     while True:
49         #Find the measured distance with distance()
50         dist = distance()
51         print ("Measured Distance = %.1f cm" % dist)
52
53         #Wait one second before a new measurement is made
54         time.sleep(1)
55
56     # Reset by pressing CTRL + C
57 except KeyboardInterrupt:
58     print("Measurement stopped by User")
59     GPIO.cleanup()
```


Making a desktop application



Introducing Tkinter :

A GUI Toolkit for Python. De facto standard GUI for Python Toolkit for GUI programming in Python. Not the only one, but the most commonly used one. Enables you to build GUI's with buttons, sliders, drop downs, and other interactions for the user of the program/device you build.

Event driven programming

When creating a GUI you use event driven programming. In the GUI all buttons, sliders, etc, are known as Widgets. When a widget is used it make an event in you code start. We will create a GUI that enables us to turn the LED on and off with a button in a GUI.

Use the 470 Ohm resistor in the circuit.

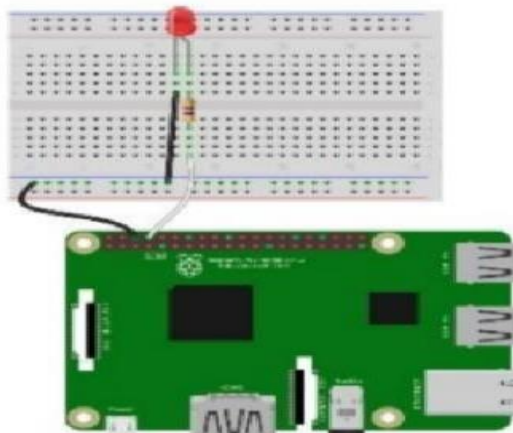


Figure 6: Raspberry Pi connected to breadboard

Building the code for this GUI



Building the GUI

Now we need to decide what the GUI will actually do.

1. We will create a button that will switch the LED on and off
2. We also have to make an exit button so that when we exit the program the LED turns off, and the GPIO port is reset to default as input ports.

Creating the widget triggered event:LedToggle()

```
19 ### Event Functions ###
20 def ledToggle():
21     if led.is_lit:
22         led.off()
23         ledButton["text"]="Turn LED on" # Change only the button text property
24     else:
25         led.on()
26         ledButton["text"]="Turn LED off"
```

Creating the widget triggered event:Close()

```
29 def close():
30     RPi.GPIO.cleanup()
31     win.destroy()
32
```

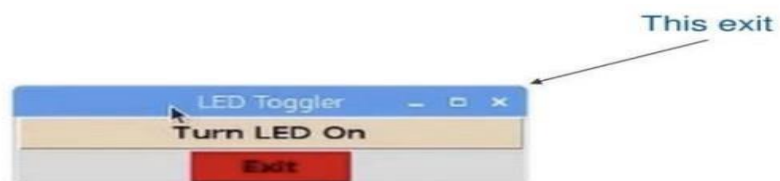
Creating the toggle button, aka a widget

```
35 ### WIDGETS ###
36
37 # Button, triggers the connected command when it is pressed
38 ledButton = Button(win, text='Turn LED on', font=myFont, command=ledToggle, bg='bisque2', height=1, width=24)
39 ledButton.grid(row=0, column=1)
40
```

Creating the exit button

```
41 exitButton = Button(win, text='Exit', font=myFont, command=close, bg='red', height=1, width=6)
42 exitButton.grid(row=2, column=1)
43
```

What happens if we exit the program through the window exit?



Making sure that we have a clean exit

```
44 win.protocol("WM_DELETE_WINDOW", close) # cleanup GPIO when user closes window
45
```

PIR sensor and Buzzer experiment

```
import RPi.GPIO as GPIO
import time
pir_sensor = 11
piezo = 7
GPIO.setmode(GPIO.BOARD)
GPIO.setup(piezo, GPIO.OUT)
GPIO.setup(pir_sensor, GPIO.IN)
current_state = 0
```

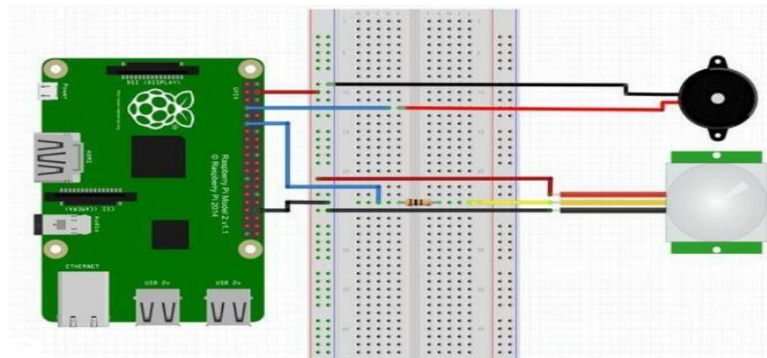


Figure 7: PIR sensor and Buzzer connected to Paspbrry Pi

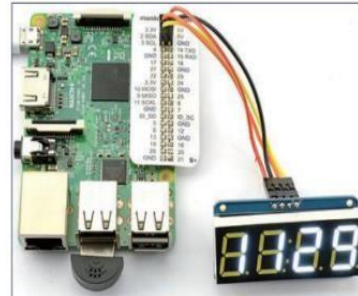
PIR sensor and Buzzer experiment(Contd...)

```
try:
    while True:
        time.sleep(0.1)
        current_state = GPIO.input(pir_sensor)
        if current_state == 1:
            print("GPIO pin %s is %s" % (pir_sensor, current_state))
            GPIO.output(piezo, True)
            time.sleep(1)
            GPIO.output(piezo, False)
            time.sleep(5)
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```

Using a seven segment display with Raspberry Pi

Quantity	Item
1	Raspberry Pi 1, 2 or 3
1	Adafruit seven-segment display w/I2C backpack
4	Female-to-female jumper wires

- VCC on the display to 5V on the Raspberry Pi
- GND to GND
- SCL on the display to SCL on the Raspberry Pi
- SDA on the display to SDA on the Raspberry Pi



Using a seven segment display with Raspberry Pi(Contd...)

```
import time
import datetime

from Adafruit_LED_Backpack import SevenSegment

# =====
# Clock Example
# =====
segment = SevenSegment.SevenSegment(address=0x70)

# Initialize the display. Must be called once before using the display.
segment.begin()

print "Press CTRL+Z to exit"

# Continually update the time on a 4 char, 7-segment display
while(True):
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    second = now.second

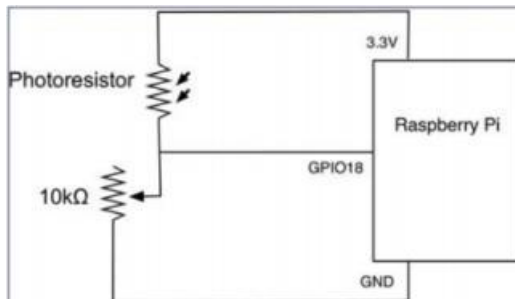
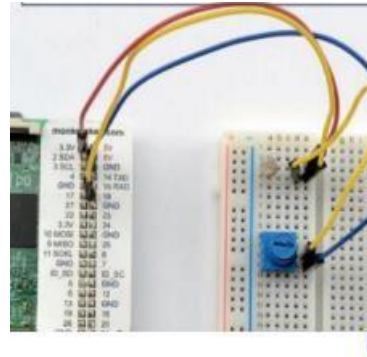
    segment.clear()
    # Set hours
    segment.set_digit(0, int(hour / 10))    # Tens
    segment.set_digit(1, hour % 10)         # Ones
    # Set minutes
    segment.set_digit(2, int(minute / 10))   # Tens
    segment.set_digit(3, minute % 10)       # Ones
    # Toggle colon
    segment.set_colon(second % 2)           # Toggle colon at 1Hz

    # Write the display buffer to the hardware. This must be called to
    # update the actual display LEDs.
    segment.write_display()

    # Wait a quarter second (less than 1 second to prevent colon blinking getting$
    time.sleep(0.25)
```

Threshold sensing with Raspberry Pi

Quantity	Item
1	Raspberry Pi (any model)
1	Photoresistor
1	10k Ω trimpot variable resistor
1	Solderless breadboard
1	Male-to-male jumper wire or solid core wire
3	Male-to-female jumper wires



Threshold sensing with Raspberry Pi(Contd...)

```
import RPi.GPIO as GPIO

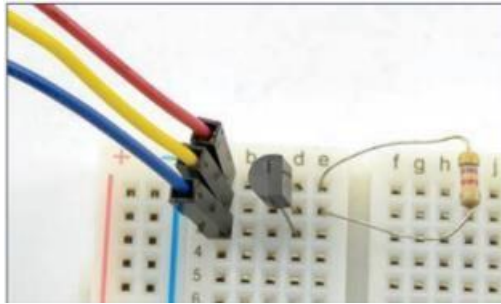
GPIO.setmode(GPIO.BCM)
sensor_pin = 18
GPIO.setup(sensor_pin, GPIO.IN)

was_light = False

try:
    while True:
        is_light = GPIO.input(sensor_pin)
        if (is_light == True) and \
            (was_light == False):
            print("It got light")
            was_light = True
        elif (is_light == False) and \
            (was_light == True):
            print("It went dark")
            was_light = False

finally:
    print("Cleaning up")
    GPIO.cleanup()
```

Using a Digital Temperature Sensor



Using a Digital Temperature Sensor(Contd...)

```
from Tkinter import *
from DS18B20 import *
import time

class App:

    # this function gets called when the app is created
    def __init__(self, master):
        self.master = master
        # A frame holds the various GUI controls
        frame = Frame(master)
        frame.pack()
        label = Label(frame, text='Temp F',
                       font=("Helvetica", 32))
        label.grid(row=0)

        self.reading_label = Label(frame, text='12.34',
                                    font=("Helvetica",
                                           20))

        self.reading_label.grid(row=1)
        self.update_reading()

    # Update the temperature reading
    def update_reading(self):
        temp = read_temp_f()
        reading_str = "{:.2f}".format(temp)
        self.reading_label.configure(text=reading_str)
        self.master.after(500, self.update_reading)

# Set the GUI running, give the window a title, size
# and position
root = Tk()
root.wm_title('Thermometer')
app = App(root)
root.geometry("800x450+0+0")
root.mainloop()
```


Hardware connection of the project

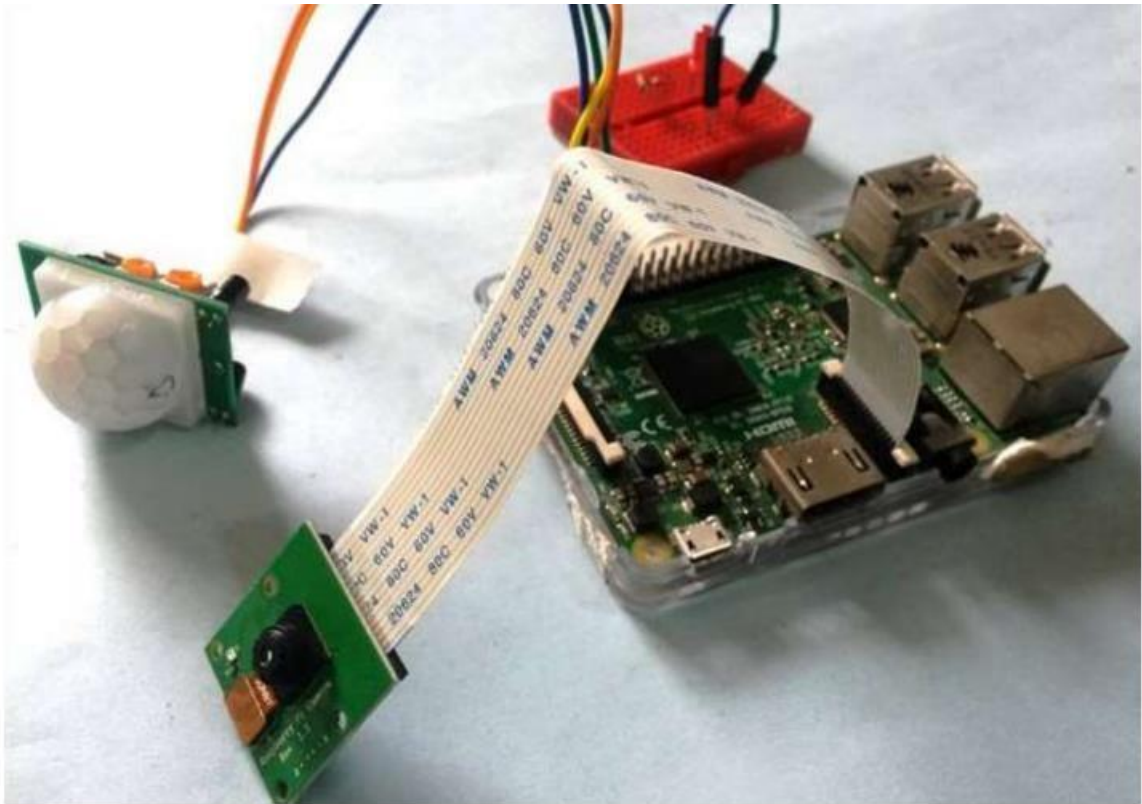


Figure 1: IOT based Raspberry Pi Home Security System with Email Alert

Components Required:

- Raspberry Pi
- Pi Camera
- PIR Sensor
- LED
- Bread Board
- Resistor (1k)
- Connecting wires
- Power supply

Raspberry Pi

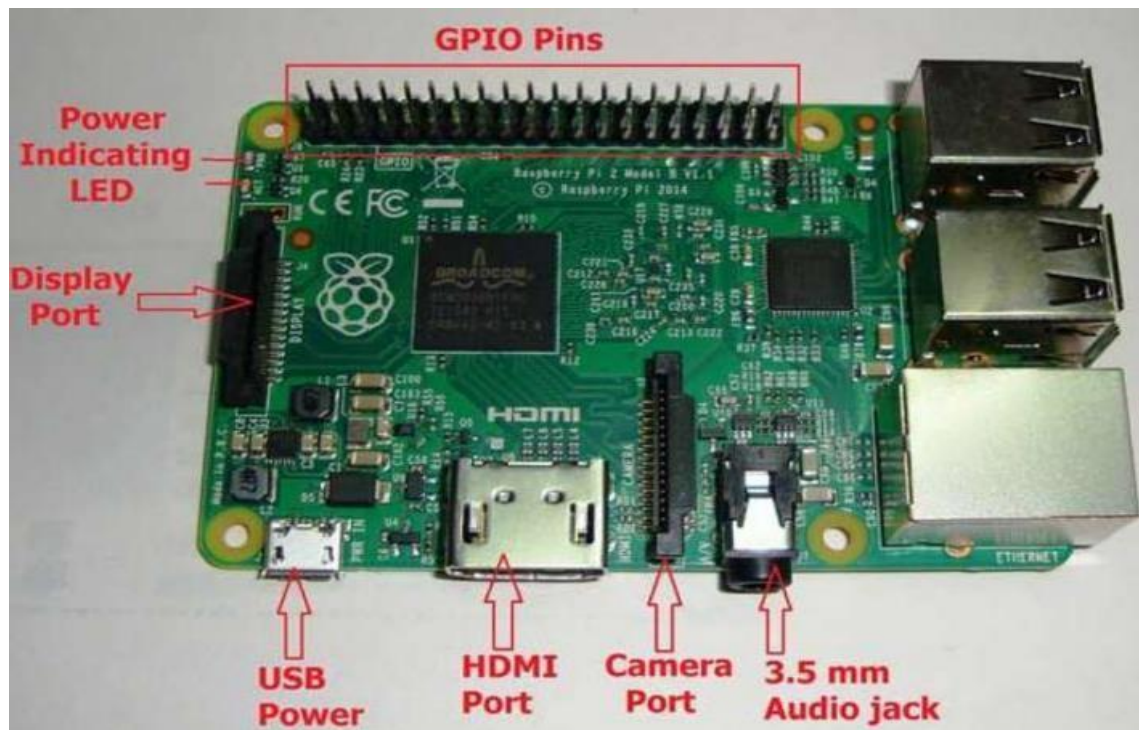


Figure 2: Raspberry Pi

Raspberry Pi(Contd...)

Raspberry Pi is an ARM cortex based single board computer working on low power. With the processing speed and memory, it can be used for performing different functions at a time, like a normal PC, and hence it is called Mini Computer in your palm.

Four USB 2.0 ports and one Ethernet port.

3.5mm jack port for connecting headphones.

It has got port for the camera module and that can be connected to PI without any additional attachments.

PI has a single HDMI port for connecting a LCD/LED screen.

It has micro USB port on the board.

It has on board Wi-Fi and Bluetooth connectivity, more powerful CPU.

There are GPIO (General Purpose Input Output) pins and a couple power ground terminals.

Pi Camera

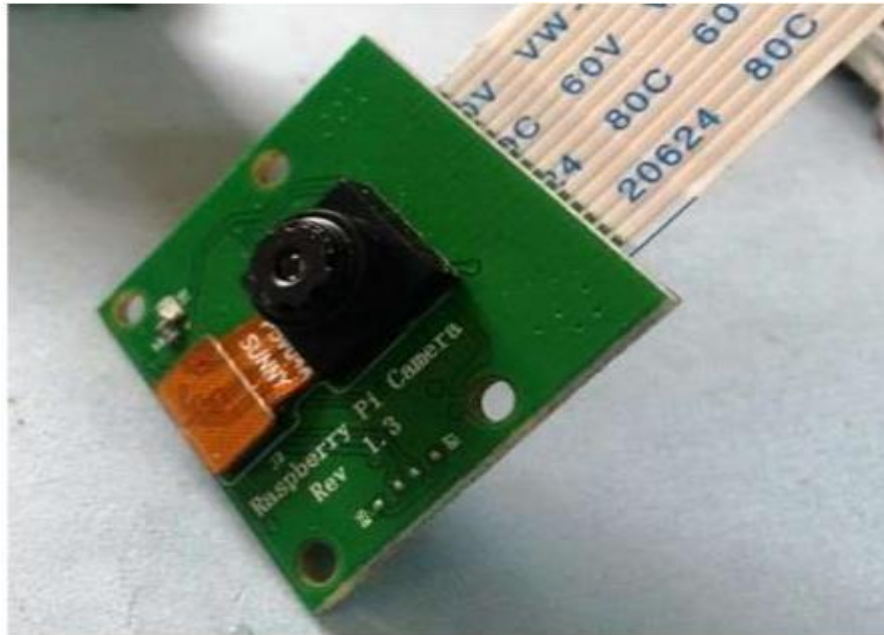


Figure 3: Pi Camera

The camera board attaches to the Raspberry Pi via a ribbon cable.

One end of the ribbon cable goes to the camera PCB and the other end attached to Raspberry Pi hardware itself.

You need to get the ribbon cable connections the right way, or the camera will not work.

On the camera PCB, the blue backing on the cable should be facing away from the PCB, and on the Raspberry Pi hardware it should be facing towards the Ethernet connection.

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.

The sensor has 5 megapixel native resolution in still capture mode.

In video mode it supports capture resolutions up to 1080p at 30 frames per second.

PIR Sensor

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.



Figure 4: PIR Sensor

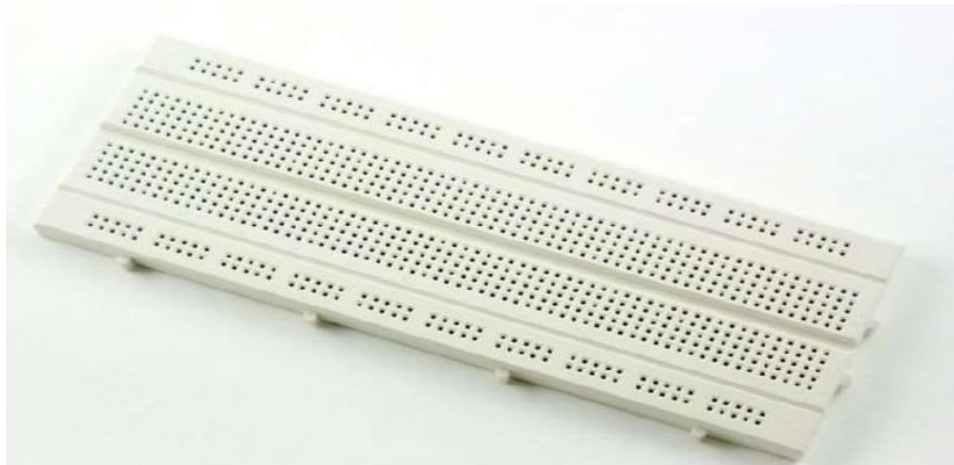
LED

- A Light Emitting Diode (LED) is a semiconductor device, which can emit light when an electric current passes through it.
- LED is a semiconductor light source that produces light when electric current flows through the diode.
- These diodes are widely used due to their effectiveness in terms of low power consumption, longer shelf life, and better illumination.
- There are various types of LEDs made and are classified into different categories.
- LEDs (Light Emitting Diodes) are the latest development in the lighting industry.
- These LEDs are popular by their efficiency, range of color, and long lifespan.
- LED lights are ideal for numerous applications, including night lighting, art lighting, and outdoor lighting.



Figure 5: LED

Breadboard



- A breadboard (sometimes called a plug block) is used for building temporary circuits. It is useful to designers because it allows components to be removed and replaced easily.
- Breadboards are useful when we want to build a circuit to demonstrate its action, then to reuse the components in another circuit.
- A breadboard is a type of prototyping that requires no soldering connections. This will make them less permanent compared to a PCB.
- Breadboards have sockets that you push the components into, allowing you to remove and change them if needed.

Resistor

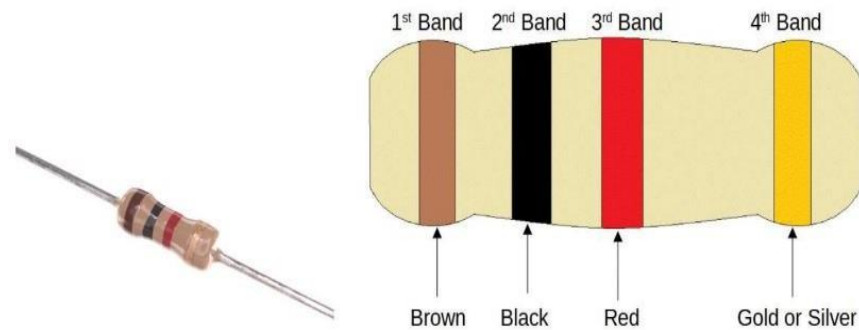


Figure 7: Resistors

- A 1K Ohm resistor can be identified via resistor color codes of Brown-Black-Red-Gold or Brown-Black-Black-Brown-Gold.
- In 1k ohm resistor the term "k" is the abbreviation for the prefix "kilo", meaning 1,000. So, a 1k ohm resistor has a value of 1,000 ohms and the number we will code is 1,000.
- The current flowing in the circuit is approximately 0.002 A.
- By the way, ten 10K resistors in parallel equals 1K total resistance.
- Resistance: 1K Ohm, Power Rating: 1 Watt, Approximate Maximum Current: 31.62mA.

Connecting Wires

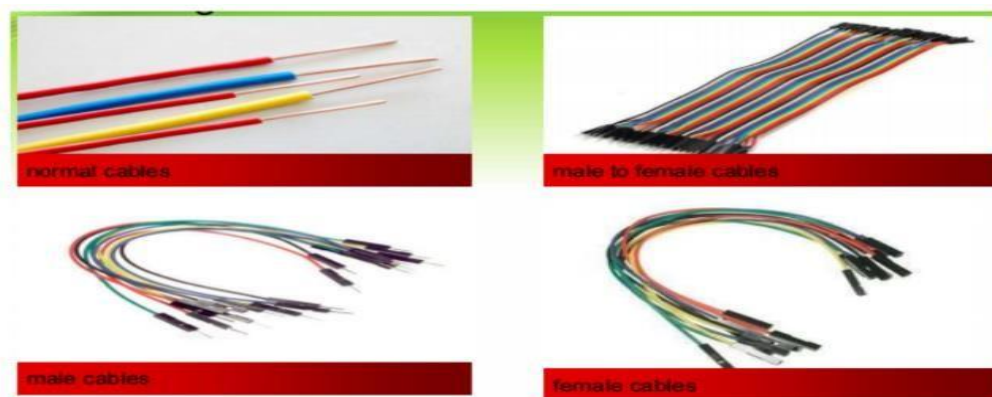


Figure 8: Connecting Wires

Circuit Description

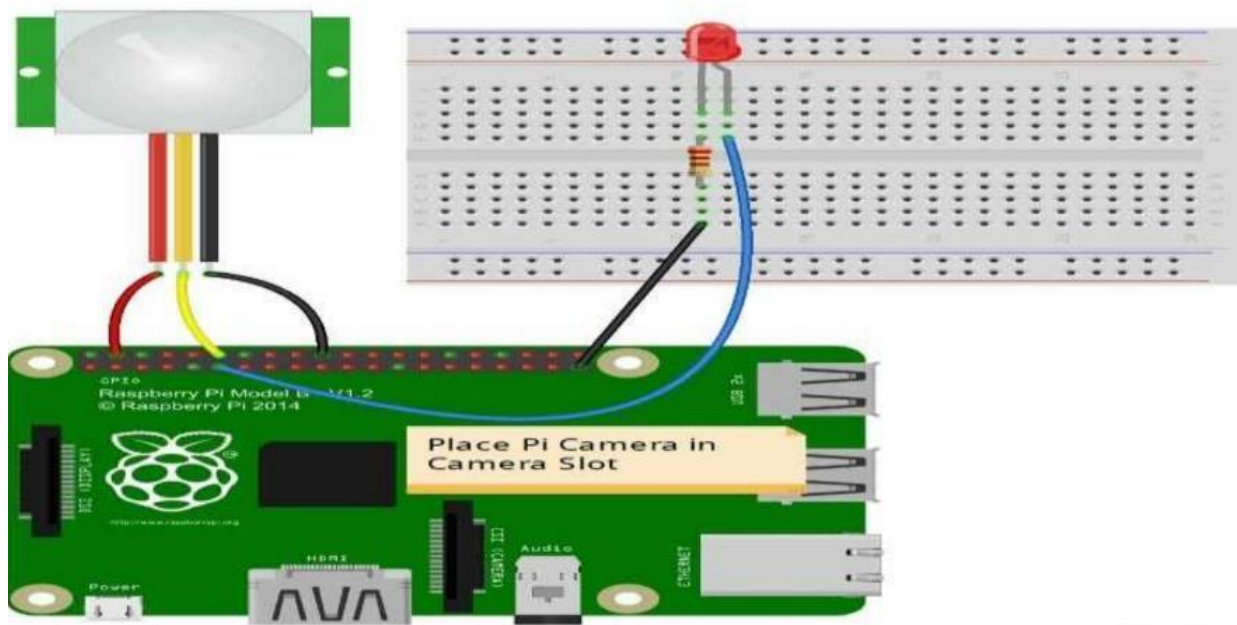


Figure 9: Circuit

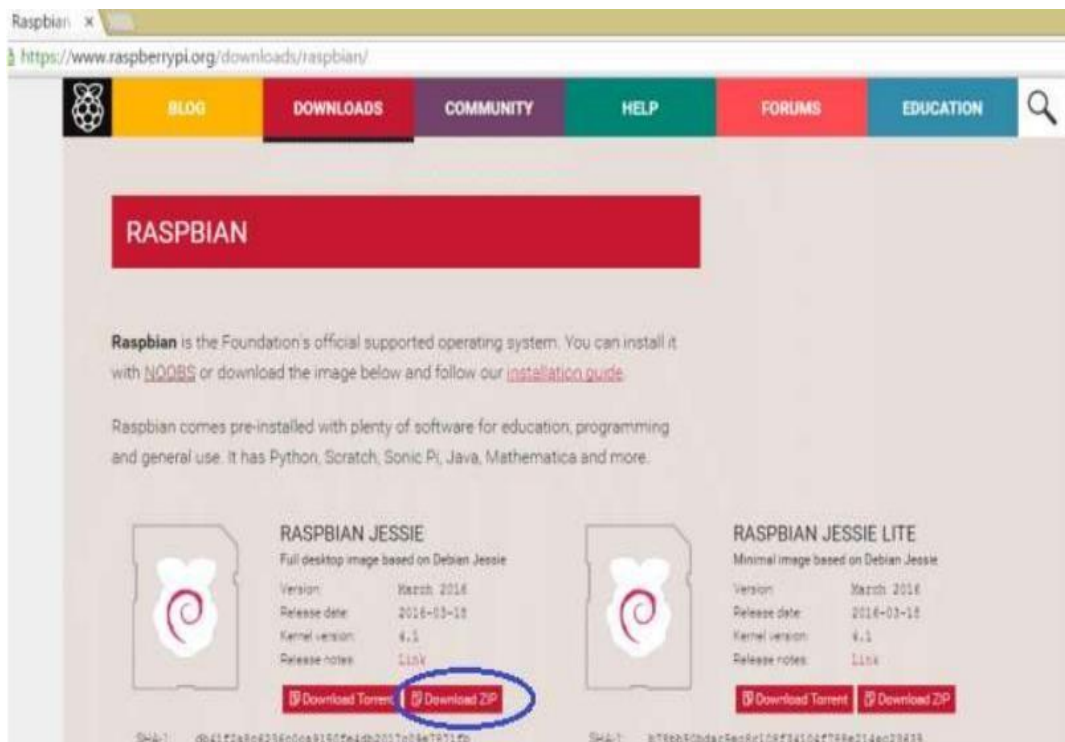
- In this Intruder Alert System, we only need to connect Pi Camera module and PIR sensor to Raspberry Pi 3.
- Pi Camera is connected at the camera slot of the Raspberry Pi and PIR is connected to GPIO pin 18.
- A LED is also connected to GPIO pin 17 through a 1k resistor.

Raspberry Pi Configuration and Programming Explanation:

- We are using Python language here for the Program. Before coding, user needs to configure Raspberry Pi. You should follow below two tutorials for Getting Started with Raspberry Pi and Installing and Configuring Raspbian Jessie OS in Pi:
- Getting Started with Raspberry Pi-Introduction
- Getting Started with Raspberry Pi-Configuration
- After successfully installing Raspbian OS on Raspberry Pi, we need to install Pi camera library files for run this project in Raspberry pi. To do this we need to follow given commands:

Software Requirements:

- First we need the OS (Operating System) for the PI, which can be downloaded from Downloads section of Raspberry Pi website:
*<https://www.raspberrypi.org/downloads/>
- It will show you all the supporting OS for the RASPBERRY PI 2. You can download and install any OS on Pi which is listed there. We are going to download official supported Operating System for Raspberry Pi, which is “Raspbian”. Click on “RASPBIAN”, and download the Full desktop image of Raspbian Jessie. Extract the Raspbian image from the Zip file, using any Zip file extractor like Winrar or Winzip.
- We also need a Image writer software for installing the OS on to the Micro SD card. We have used “win32diskimager” to write the image on Micro SD card, which can be downloaded from below link:
*<https://sourceforge.net/projects/win32diskimager/> Once the download completes, install the software, you will see an icon on the Desktop Screen after installation.



Get started with Raspberry Pi:

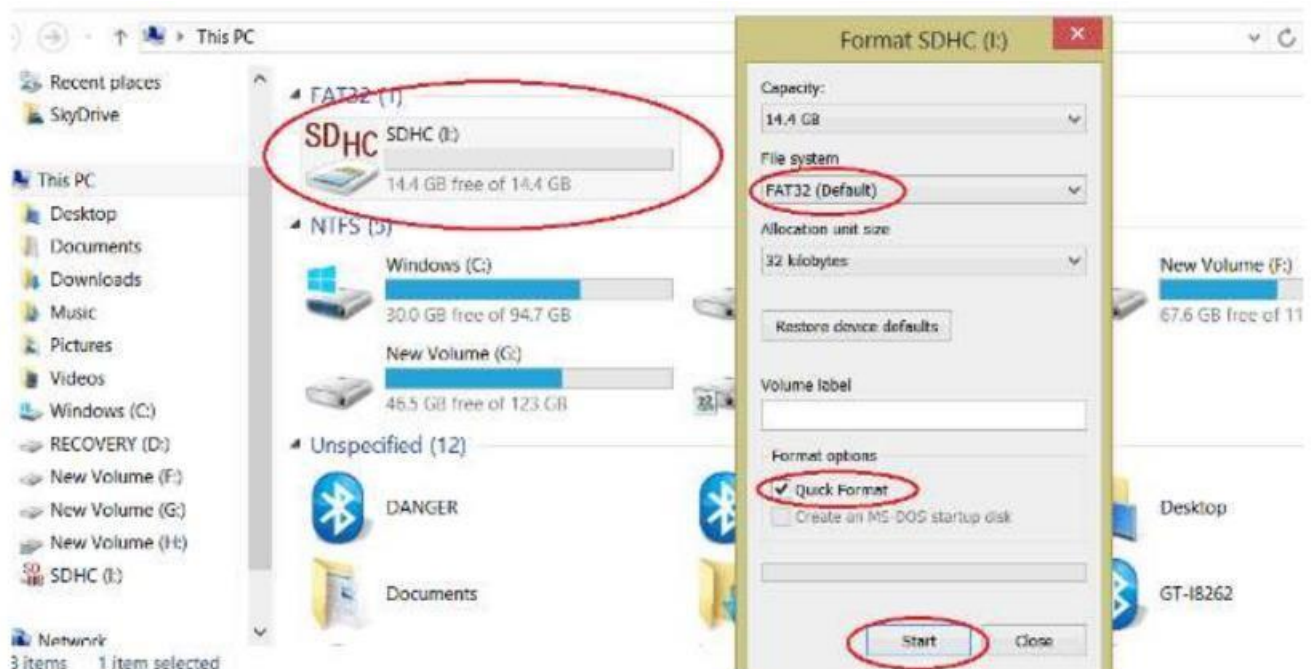
Now, we have all the software and hard ware required to get started with the PI 2.

To install OS on to a SD card follow below steps:

1. UNZIP the ‘Raspbian Jessie’ (OS ZIP file we downloaded from raspberry website) on to the desktop; you will see an Image icon upon extraction on the screen as shown below. Make sure you have at least 5 GB free disk space on ‘C’ drive of your PC. The extraction file size would be greater than 3GB.

2. Insert the SD card into the USB card reader or Card Adapter. Plug the card reader to the PC. You must see the card on the screen as shown below.

3. Format the card drive by Right click on it and select Format. Select File system as ‘FAT32’ and tick on ‘Quick Format’. Finally click on ‘Start’ button to Format the drive.



4. After formatting, run the “win32diskimager” application, which we have downloaded as explained previously

5. Choose the SD card drive, browse for the Raspbian OS image file (which is extracted on the screen) and Click on 'WRITE' icon, to start writing the extracted OS file on to the SD card. This is shown in below figure.

6. After completion of writing, safely remove the SD card from the reader.

Now we have SD card with Raspbian OS installed on it and having all the equipment required to Get Started with Raspberry Pi 2. In the next session we will have the first look at the "PI" OS and we will talk about configuring the BIOS of Raspberry Pi.



In previous session, we have learned about Raspberry Pi board, its ports, its hardware and software requirements and learned to install OS on SD card for PI.

Once the OS (Raspbian Jessie) installed SD card is inserted into the Raspberry Pi with the screen, keyboard and mouse connected, we are ready to boot the Jessie first time. For this particular OS you don't need Ethernet connection.

Once the power is started ON, you will see the power RED LED glowing. The BLUE LED will start blinking at this stage, this means the OS (Operating System) is loading and the PI is checking all the drivers.

By this time you will see data on the screen as shown below,



```
[ 3.206397] [001cf498] (ext4_lookup) from [0014c018] (lookup_real+0x30/0x5c)
[ 3.297203] [0014c018] (lookup_real) from [0014cbb8] (__lookup_hash+0x44/0x4c)
[ 3.308355] [0014cbb8] (__lookup_hash) from [0014ec38] (lookup_slow+0x48/0xb4)
[ 3.319425] [0014ec38] (lookup_slow) from [0014fb8c] (path_lookupat+0x6e8/0x738)
[ 3.330678] [0014fb8c] (path_lookupat) from [0014ff98] (filename_lookup.isra.46+0x30/0x70)
[ 3.342818] [0014ff98] (filename_lookup.isra.46) from [00152130] (user_path_at_empty+0x64/0x8c)
[ 3.355408] [00152130] (user_path_at_empty) from [0015217c] (user_path_at+0x24/0x2c)
[ 3.367063] [0015217c] (user_path_at) from [00141f18] (SyS_faccessat+0xa0/0x1d8)
[ 3.378370] [00141f18] (SyS_faccessat) from [00142078] (SyS_access+0x28/0x2c)
[ 3.389433] [00142078] (SyS_access) from [0000ebc0] (ret_fast_syscall+0x0/0x48)
[ 3.400661] Code: e7934004 e3540000 0a00004c e5963014 (e794e003)
[ 3.410367] ---[ end trace de0305eb0d5102c5 ]---
[ 3.413518] usb 1-1: New USB device found, idVendor=0424, idProduct=9514
[ 3.413525] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 3.414105] hub 1-1:1.0: USB hub found
[ 3.414178] hub 1-1:1.0: 5 ports detected
[ 3.454547] Kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b
[ 3.454547]
[ 3.470602] CPU0: stopping
[ 3.476842] CPU: 0 PID: 0 Comm: swapper/0 Tainted: G      D      3.18.7-ub7+ #755
[ 3.487908] [00016d14] (unwind_backtrace) from [00012c40] (show_stack+0x20/0x24)
[ 3.499161] [00012c40] (show_stack) from [0052efe8] (dump_stack+0x98/0xd0)
[ 3.509881] [0052efe8] (dump_stack) from [0001509c] (handle_IPI+0x234/0x268)
[ 3.520759] [0001509c] (handle_IPI) from [00000618] (do_IPI+0x18/0x1c)
[ 3.531122] [00000618] (do_IPI) from [005349b4] (__irq_svc+0x34/0x14c)
[ 3.541471] Exception stack(0xb07cbf08 to 0xb07cbf50)
[ 3.550003] bf00:      007e9ccc 00000000 ffffffff 00000000 007ca020 007c8d44
```

As told its just PI is loading all the drives. You have to wait until all the drivers are checked, in case of error, turn off and on to restart PI. If there is still trouble try installing the OS on to the SD card again by following the steps described in first session.

If everything goes successfully, you will be asked for authorization. This authorization is predefined, with Username “pi” and password “raspberrypi”,

USER: pi

< Press enter>

PASSWORD: raspberrypi

< Press enter>

Once you enter these details, you will be entered in CLI mode (Command Line) of Raspberry Pi. For entering into the DESKTOP of PI you need to type,

startx

< Press enter>

BIOS settings:

Now you have entered into the screen of Raspberry Pi and you are ready to go. Before going for the programming,

1. You need to configure the BIOS settings of PI
2. You need to configure the keyboard, you have chosen.

If you don't do these two things first you get lot of errors, while programming and operating the PI.

For configuring the BIOS of PI first open the 'LX TERMINAL' of PI and enter this,

```
sudo raspi-config
```

< Press enter>

You will be entered into BIOS after this step.

1. Expand File System: After first start, some memory of the SD will be misplaced and is not considered by PI. You need to select this option to get the files and order and to display the remaining memory of SD card. Once you choose this option, the PI will REBOOT to get everything in order. If you don't expand the file system, you will not be able to use remaining memory of SD card.

2. Change Password: This option changes the login password, its "pi" by default. Just leave it, if you are not doing any important work.

3. Boot To Desktop: On the first start you are entered in CLI mode, as discussed earlier. This option disables that, so that you can enter DESKTOP of Pi with every start.

4. Internationalisation Options: This option is for choosing language. Its ENGLISH by default, leave it you don't want to change the language and date. The DATE may be wrong, we will configure these settings later.

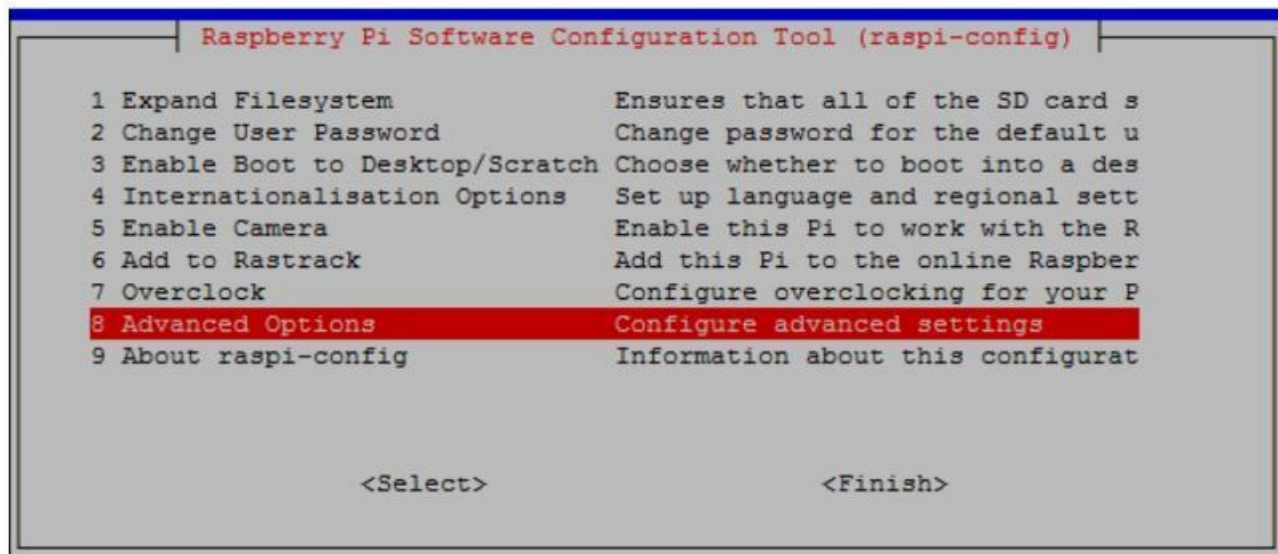
5. Enable Camera: If you have a camera module draws power, so once you enable it the module will be drawing power continuously.

6. Add To Rastack: This option is for connecting your PI online. Just leave it.

7. Over Clock: This option over clocks PI, thus increasing your PI speed and also its power consumption. Over clocking might damage the board, if efficient cooling system is not provided. For basic programming, you need not over clock the PI, just leave it 900MHz.

8. Advanced Options: These options for BOARD devices (like AUDIO, I2C etc) configuration:

We have few options under this, but the important for now is, 'AUDIO'. The PI can output AUDIO either from HEADPHONE jack (on the PI board) or from HDMI port. To find out the ports, check Getting Started with Raspberry Pi. Choose the appropriate one based on you usage. If you won't configure this, you won't get AUDIO. The remaining options are not important for now.



Keyboard Configuring:

We use different type of keyboard all around the world, most of the keys match for all keyboards. But few keys mismatch. This will cause really trouble while programming, as special keys play a crucial part while doing programming in PYTHON and LINUX.

So we need to configure the Raspberry Pi to the keyboard we are using.

I am from INDIA and almost everyone one here uses US (United States) keyboard models.

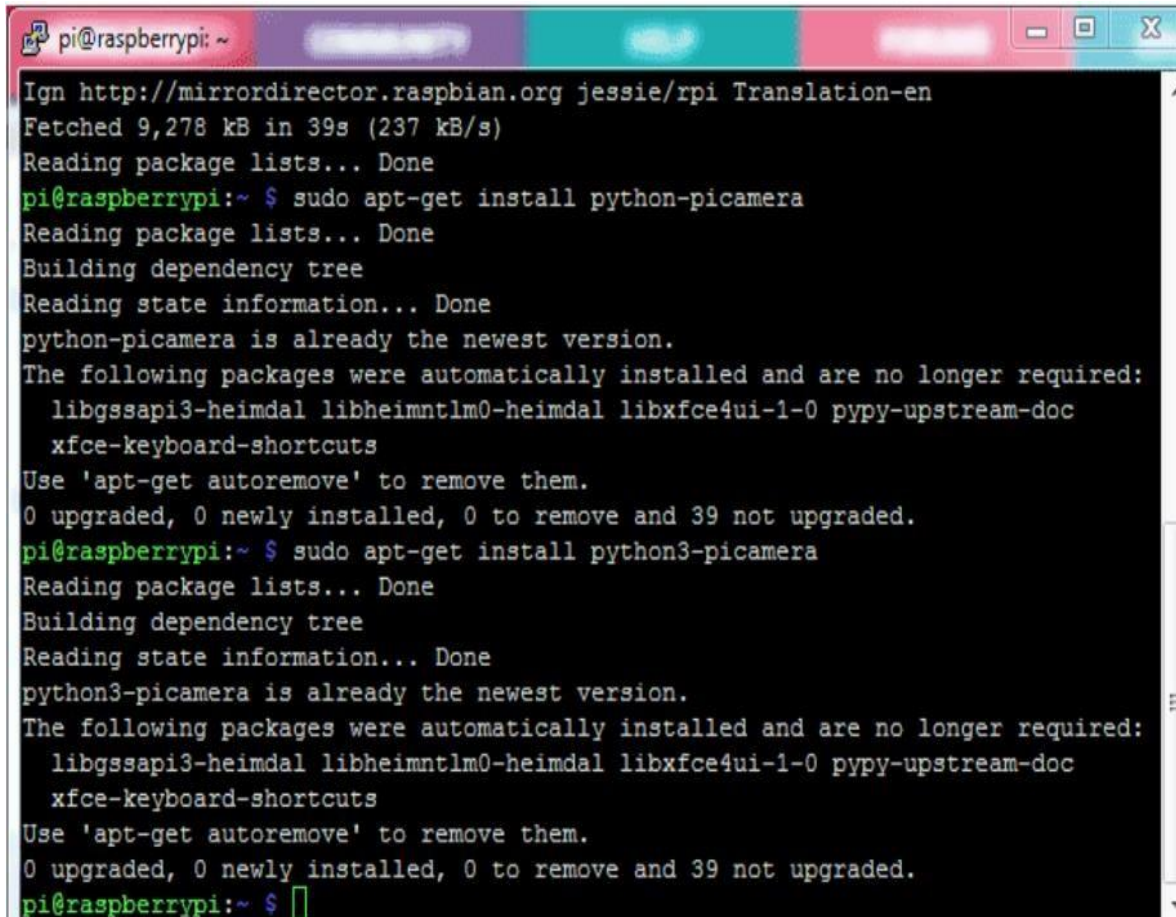


Raspberry Pi Configuration and Programming Explanation:

After successfully installing Raspbian OS on Raspberry Pi, we need to install Pi camera library files for run this project in Raspberry pi. To do this we need to follow given commands:

```
$ sudo apt-get install python-picamera
```

```
$ sudo apt-get installpython3-picamera
```

```
pi@raspberrypi: ~  
Ign http://mirrordirector.raspbian.org jessie/rpi Translation-en  
Fetched 9,278 kB in 39s (237 kB/s)  
Reading package lists... Done  
pi@raspberrypi:~ $ sudo apt-get install python-picamera  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python-picamera is already the newest version.  
The following packages were automatically installed and are no longer required:  
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc  
  xfce-keyboard-shortcuts  
Use 'apt-get autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.  
pi@raspberrypi:~ $ sudo apt-get install python3-picamera  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python3-picamera is already the newest version.  
The following packages were automatically installed and are no longer required:  
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc  
  xfce-keyboard-shortcuts  
Use 'apt-get autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.  
pi@raspberrypi:~ $
```

After it, user needs to enable Raspberry Pi Camera by using Raspberry Pi Software Configuration Tool (raspi-config):

```
$ sudo raspi-config
```

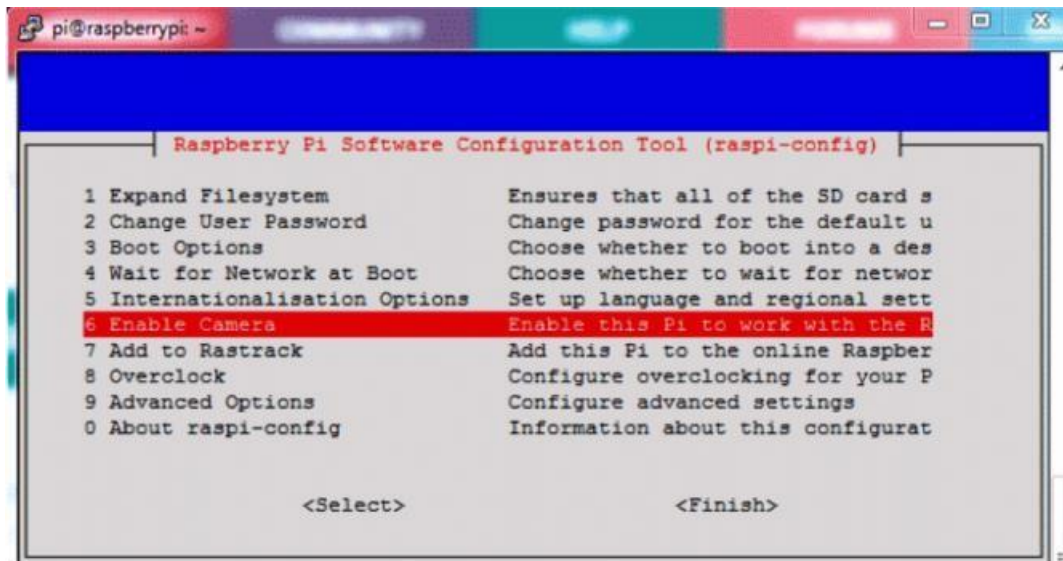
Then select Enable camera and Enable it.

Then user needs to reboot Raspberry Pi, by issuing `sudo reboot`, so that new setting can take. Now your Pi camera is ready to use.

Now after setting up the Pi Camera, we will install software for sending the mail. Here we are using `ssmtp` which is an easy and good solution for sending mail using command line or using Python Script. We need to install two Libraries for sending mails using SMTP:

```
sudo apt-get install ssmtp
```

```
sudo apt-get install mailutils
```



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo apt-get install ssmtp  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
ssmtp is already the newest version.  
The following packages were automatically installed and are no longer required:  
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc  
  xfce-keyboard-shortcuts  
Use 'apt-get autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.  
pi@raspberrypi:~ $ sudo apt-get install mailutils  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
mailutils is already the newest version.  
The following packages were automatically installed and are no longer required:  
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc  
  xfce-keyboard-shortcuts  
Use 'apt-get autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.  
pi@raspberrypi:~ $ sudo nano /etc/ssmtp/ssmtp.conf
```

After installing libraries, user needs to open ssmtp.conf file and edit this configuration file as shown in the Picture below and then save the file.

To save and exit the file, Press 'CTRL+x', then 'y' and then press 'enter'.

```
sudo nano /etc/ssmtp/ssmtp.conf
```

```
root=YourEmailAddress
```

```
mailhub=smtp.gmail.com:587
```

```
hostname=raspberrypi
```

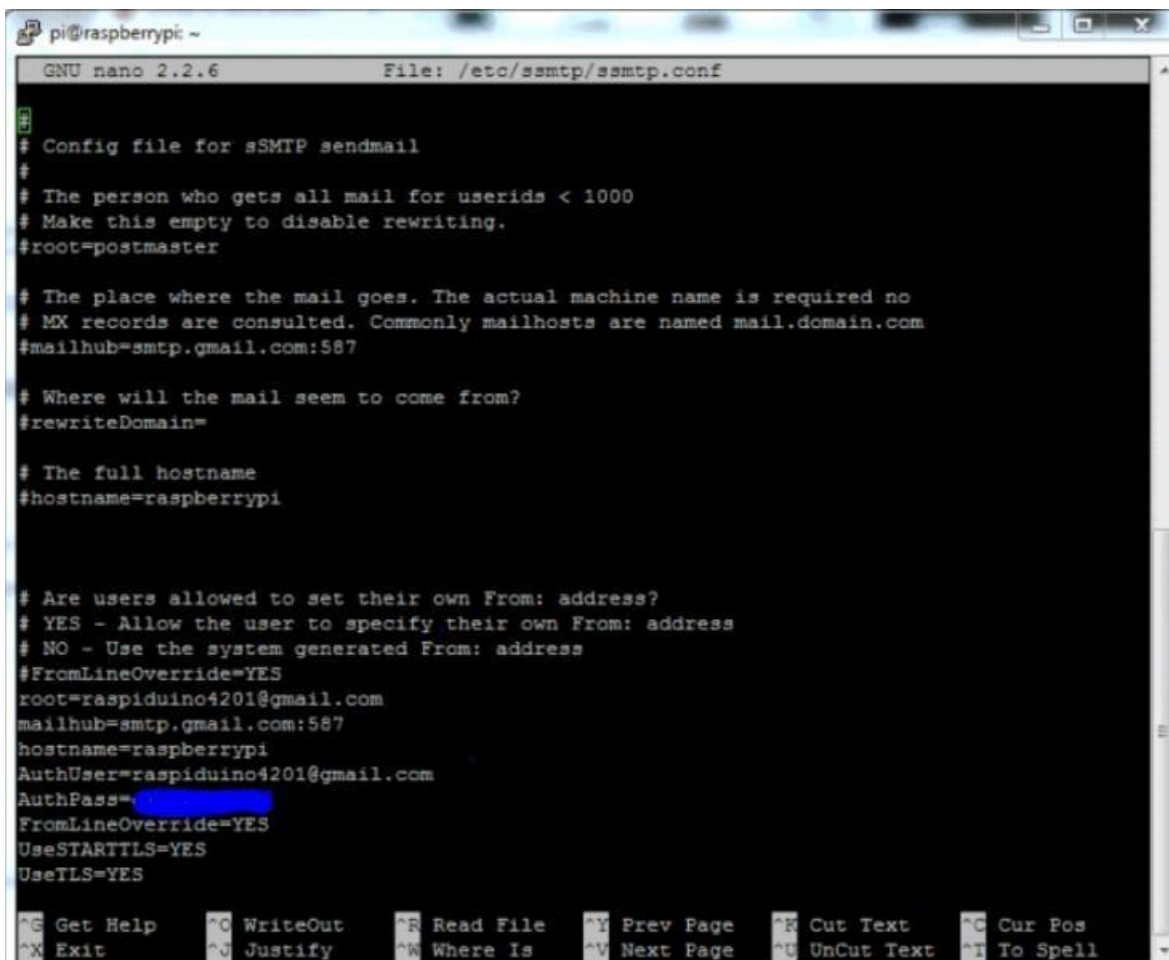
```
AuthUser=YourEmailAddress
```

```
AuthPass=YourEmailPassword
```

```
FromLineOverride=YES
```

```
UseSTARTTLS=YES
```

```
UseTLS=YES
```



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/ssmtp/ssmtp.conf
#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
#root=postmaster
#
# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
#mailhub=smtp.gmail.com:587
#
# Where will the mail seem to come from?
#rewriteDomain=
#
# The full hostname
#hostname=raspberrypi
#
# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
#FromLineOverride=YES
root=raspiduino4201@gmail.com
mailhub=smtp.gmail.com:587
hostname=raspberrypi
AuthUser=raspiduino4201@gmail.com
AuthPass=
FromLineOverride=YES
UseSTARTTLS=YES
UseTLS=YES
^G Get Help ^C WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

We can also test it by sending a test mail by issuing below command, you shall get the mail on the mentioned email address if everything is working fine:

```
echo "Hello saddam" | mail -s "Testing..."saddam4201@gmail.com
```

The Python Program of this project plays a very important role to perform all the operations. First of all, we include required libraries for email, initialize variables and define pins for PIR, LED and other components.

```
import RPi.GPIO as gpio
import picamera
import time
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEBase import MIMEBase
from email import encoders
from email.mime.image import MIMEImage
```

After it, we have initialized mail and define mail address and messages:

```
fromaddr = "raspiduino4201@gmail.com"
toaddr = "saddam4201@gmail.com"
mail = MIMEMultipart()
mail['From'] = fromaddr
mail['To'] = toaddr
mail['Subject'] = "Attachment"
body = "Please find the attachment"
```

Then we have created def sendMail(data) function for sending mail:

```
def sendMail(data):
    mail.attach(MIMEText(body, 'plain'))
    print data
    dat='%s.jpg'%data
    print dat
    attachment = open(dat, 'rb')
    image=MIMEImage(attachment.read())
    attachment.close()
    mail.attach(image)
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
```

```
server.login(fromaddr, "your password")
```

```
text = mail.as_string()
```

```
server.sendmail(fromaddr, toaddr, text)
```

```
server.quit()
```

Function def capture_image() is created to capture the image of intruder with time and date.

```
def capture_image():
```

```
data= time.strftime("%d_%b_%Y|%H:%M:%S")
```

```
camera.start_preview()
```

```
time.sleep(5)
```

```
print data
```

```
camera.capture('%s.jpg'%data)
```

```
camera.stop_preview()
```

```
time.sleep(1)
```

```
sendMail(data)
```

Then we initialized the Picamera with some of its settings:

```
camera = picamera.PiCamera()
```

```
camera.rotation=180
```

```
camera.awb_mode= 'auto'
```

```
camera.brightness=55
```

we have read PIR sensor output and when its goes high Raspberry Pi calls the

capture_image() function to capture the image of intruder and send a alert message with the

picture of intruder as an attachment. We have used sendmail() insdie capture_image() function.

while 1:

```
if gpio.input(pir)==1:
```

```
gpio.output(led, HIGH)
```

```
capture_image()
```

```
while(gpio.input(pir)==1):
```

```
time.sleep(1)
```

```
else:
```

```
gpio.output(led, LOW)
```

```
time.sleep (0.01)
```

Program for IOT based Raspberry Pi Home Security System

```
import RPi.GPIO as gpio
import picamera
import time
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEBase import MIMEBase
from email import encoders
from email.mime.image import MIMEImage
fromaddr = "raspiduino4201@gmail.com" # change the email address accordingly
toaddr = "saddam4201@gmail.com"
mail = MIMEMultipart()
mail['From'] = fromaddr
mail['To'] = toaddr
mail['Subject'] = "Attachment"
body = "Please find the attachment"
led=17
pir=18
HIGH=1
LOW=0
gpio.setwarnings(False)
gpio.setmode(gpio.BCM)
gpio.setup(led, gpio.OUT) #initialize GPIO Pin as outputs
gpio.setup(pir, gpio.IN) #initialize GPIO Pin as input data=""
def sendMail(data):
    mail.attach(MIMEText(body, 'plain'))
    print data
    dat='%s.jpg'%data
    attachment = open(dat, 'rb')
    image=MIMEImage(attachment.read())
```

```

attachment.close()
mail.attach(image)
server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login(fromaddr, "your password")
text = mail.as_string()
server.sendmail(fromaddr, toaddr, text)
server.quit()
def capture_image():
data= time.strftime("%d_%b_%Y|%H:%M:%S")
camera.start_preview()
time.sleep(5)
print data
camera.capture('%s.jpg'%data)
camera.stop_preview()
time.sleep(1)
sendMail(data)
gpio.output(led , 0)
camera = picamera.PiCamera()
camera.rotation=180
camera.awb_mode= 'auto'
camera.brightness=55
while 1:
if gpio.input(pir)==1:
gpio.output(led, HIGH)
capture_image()
while(gpio.input(pir)==1):
time.sleep(1)
else:
gpio.output(led, LOW)
time.sleep(0.01

```

Program for to Drive a given string on Hyper Terminal

```
void setup() {
  Serial.begin(9600); // start serial communication on arduino with pc
  pinMode(A6, INPUT); // SET THE LDR INPUT
}

void loop() {
  Serial.print("Hello! This is a string... without ENTER key pressed! $$$$ ");
  // prints the text under the double quotes without any discrimination
  // no new line and carriage return
  giveDelay(3000); // give a 3 second
  delay
  Serial.println("Hello! This is a string...with ENTER
  pressed!@@@@ "); delay(2000); // give a 2 second delay
  Serial.print("\n"); // this is equivalent to pressing the enter button of keyboard (in software)
  delay(2000); // give a 2 second delay
  Serial.println(" This string gives a carriage return and new line ");
  delay(1000); // give a one second delay
  Serial.println("****LET'S PRINT THE Light SENSOR
  READINGS****"); Serial.println("Trial No.\t\tDarkness\t\tunits");
  // this will print column header with two tab spacing in between
  // if you want to print the light intensity(darkness) in formal way, then use the following lines

  int i=0;
  for(int k=0;k<10;k++){
    int value = analogRead(A6);
    Serial.println(String(i++) + "\t\t" + String(value/10) + "\t\t" + " %
    dark"); delay(500);
  } // close for loop
  Serial.println("****LET'S wait for five
  seconds****"); for(int f=5;f>0;f--){
    Serial.println(f);
    delay(1000);
  }

  Serial.println("****LET'S PRINT THE TEMPERATURE SENSOR READINGS****");
  Serial.println("Trial No.\t\tTemperaure\t\tunits");
  i=0; // Reinitialize i
  to 0 for(int
  k=0;k<10;k++){
    float temp = analogRead(A0); // default pin for Temp sensor LM35
    Serial.println(String(i++) + "\t\t" + String(temp*500.00/1024.00) + "\t\t" + " deg
    Celcius"); delay(500);
  } // close for loop
  Serial.println("****LET'S wait for five
  seconds****"); for(int f=5;f>0;f--){
    Serial.println(f);
    delay(1000);
  }
}
```

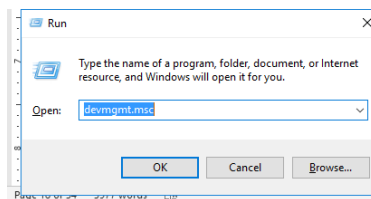
```
}  
}
```

Uploading process: Upload the sketch to Arduino board as described in Pages7-10. After uploading, make sure you close the serial window on the Arduino IDE.

After successful uploading of code in Arduino, Open putty software in the pc and select the “serial” instead of ssh

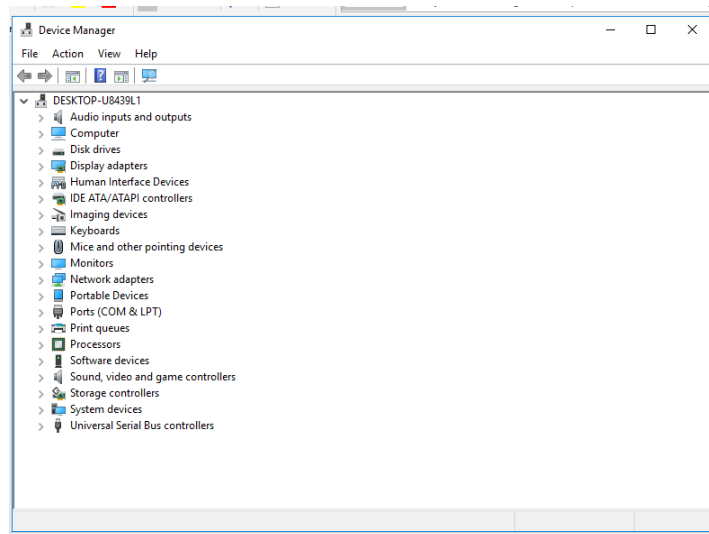
Type the COM port number using your device manager. Press

Windows key +R in the keyboard

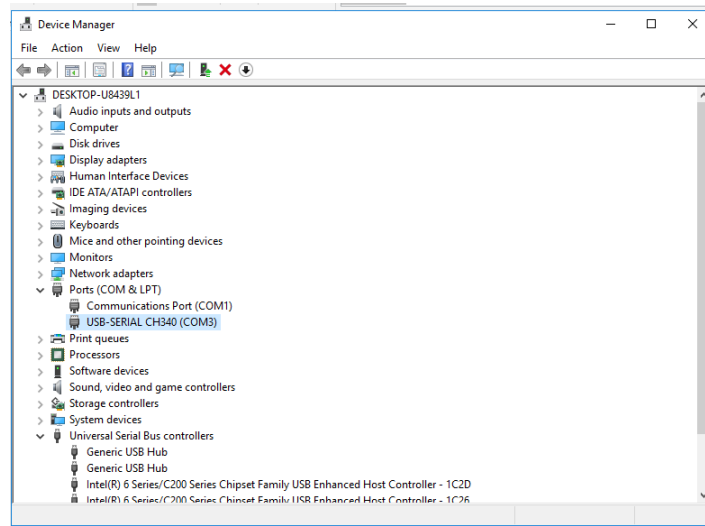


Opens the Run command prompt.

Now type “devmgmt.msc” (without inverted commas) in the dialog box.

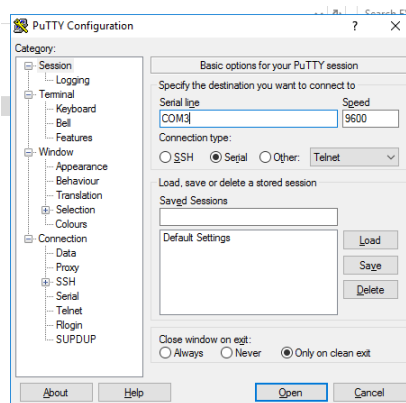


Look at Ports(Com &LPT) and double click on it

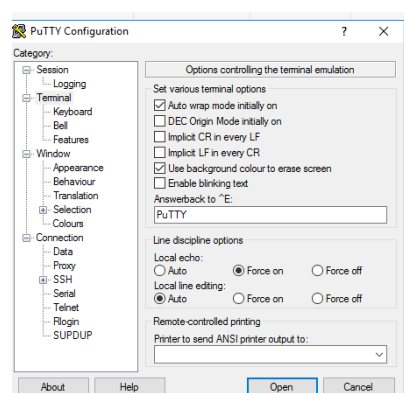


In my case the com number is 3 [USB-SERIAL CH340(COM3)]

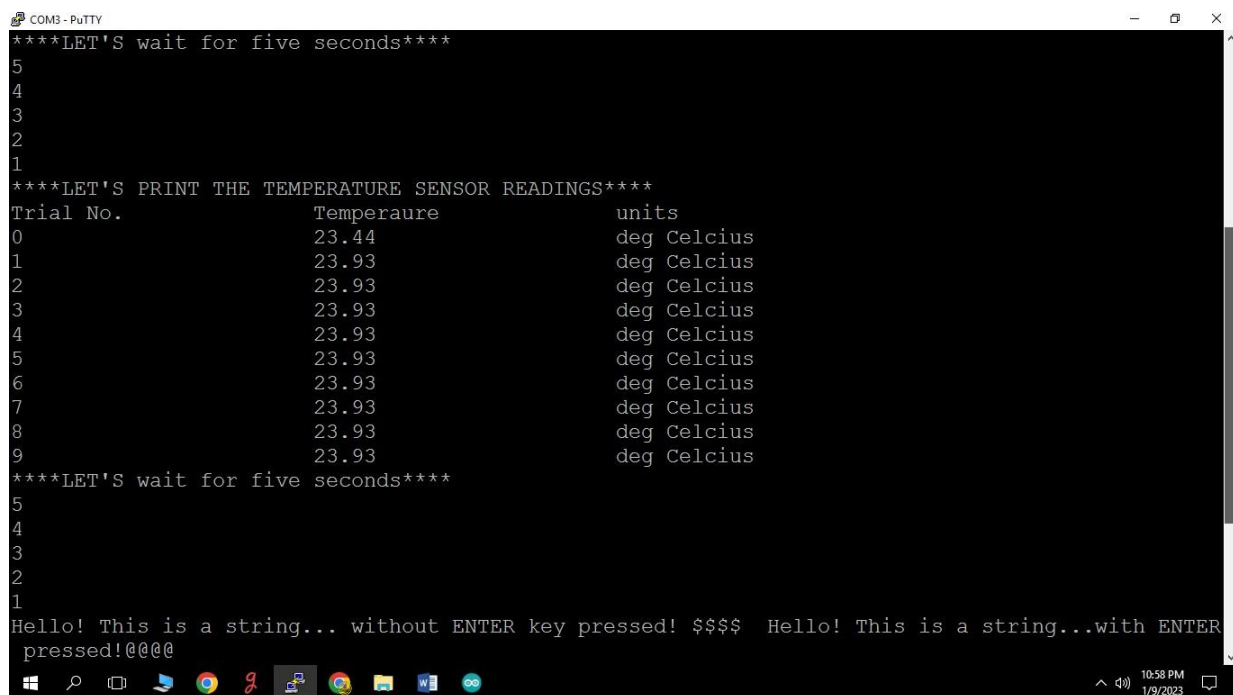
In the putty software enter the number 3 for com port



Select “Terminal” under “category” and select the “Line discipline options” ,“Local Echo”....
Select “Force on”.

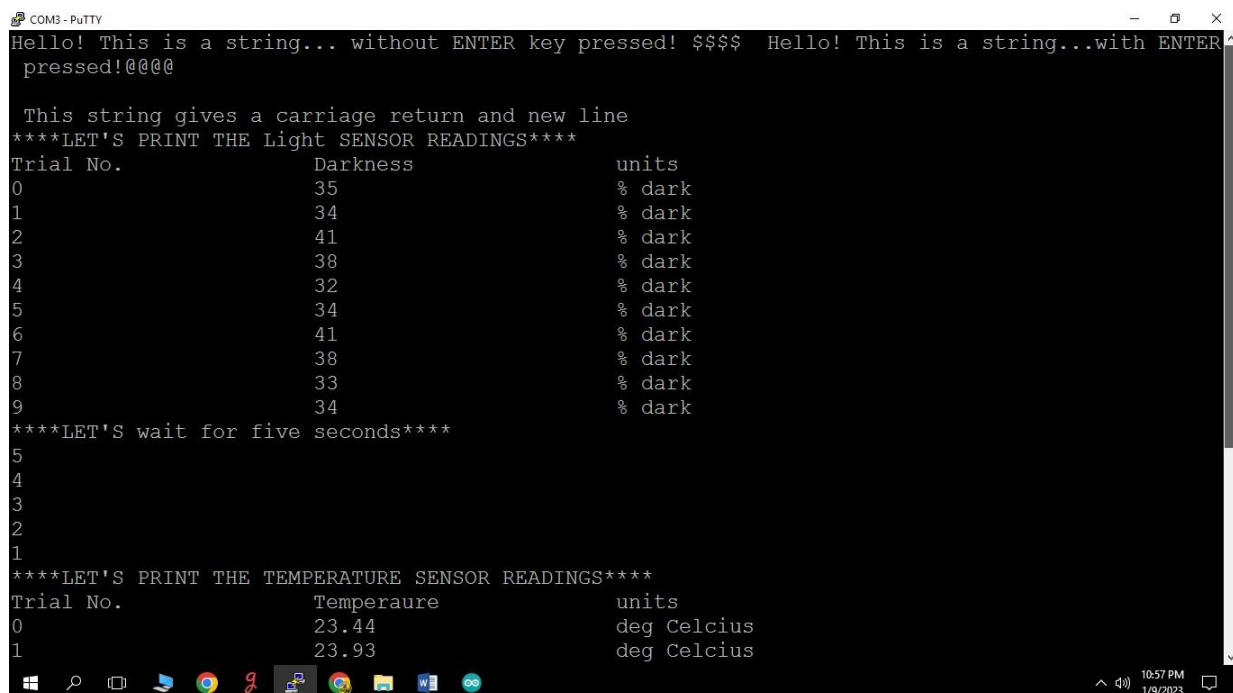


The following screen shots show the output in the terminal window



A screenshot of a terminal window titled 'COM3 - PuTTY'. The output shows a countdown from 5 to 1, followed by a header '***LET'S PRINT THE TEMPERATURE SENSOR READINGS***'. A table of temperature readings is displayed, with columns for Trial No., Temperature, and units. The temperature values are 23.44 for trial 0 and 23.93 for trials 1 through 9. After another countdown, a string is printed without an enter key, followed by a string with an enter key. The Windows taskbar is visible at the bottom.

```
COM3 - PuTTY
***LET'S wait for five seconds***
5
4
3
2
1
***LET'S PRINT THE TEMPERATURE SENSOR READINGS***
Trial No.      Temperaure      units
0              23.44             deg Celcius
1              23.93             deg Celcius
2              23.93             deg Celcius
3              23.93             deg Celcius
4              23.93             deg Celcius
5              23.93             deg Celcius
6              23.93             deg Celcius
7              23.93             deg Celcius
8              23.93             deg Celcius
9              23.93             deg Celcius
***LET'S wait for five seconds***
5
4
3
2
1
Hello! This is a string... without ENTER key pressed! $$$$ Hello! This is a string...with ENTER
pressed!@@@
```



A screenshot of a terminal window titled 'COM3 - PuTTY'. The output shows a message about carriage return and new line, followed by a header '***LET'S PRINT THE Light SENSOR READINGS***'. A table of light sensor readings is displayed, with columns for Trial No., Darkness, and units. The darkness values range from 32 to 41 for trials 1 through 9. After a countdown, a second table of temperature sensor readings is shown for trials 0 and 1. The Windows taskbar is visible at the bottom.

```
COM3 - PuTTY
Hello! This is a string... without ENTER key pressed! $$$$ Hello! This is a string...with ENTER
pressed!@@@

This string gives a carriage return and new line
***LET'S PRINT THE Light SENSOR READINGS***
Trial No.      Darkness      units
0              35            % dark
1              34            % dark
2              41            % dark
3              38            % dark
4              32            % dark
5              34            % dark
6              41            % dark
7              38            % dark
8              33            % dark
9              34            % dark
***LET'S wait for five seconds***
5
4
3
2
1
***LET'S PRINT THE TEMPERATURE SENSOR READINGS***
Trial No.      Temperaure      units
0              23.44             deg Celcius
1              23.93             deg Celcius
```

Program to establish Full duplex Link using Hyper terminal.

Program:

```
// This program makes bidirectional serial
communication void setup() {
Serial.begin(9600);// start serial communication on arduino with pc
Serial.println("Arduino Math table printer");
Serial.println ("Hello! I am Arduino Nano. Please type the number\n for which you want the table");
}
```

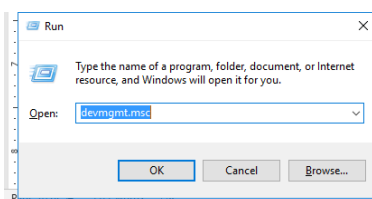
```
void loop() {
if(Serial.available(>0){ // when any serial input is
given int val = Serial.parseInt();
if(val!=0){
for (int i=1;i<13;i++){
Serial.println(String(val)+" X "+String(i)+ " = " +String(val*i));
}
Serial.println("Table printing is over... \nPlease enter another number..");
} // end if (val!=0) loop
} // end if(Serial.available(>0))
} // end main loop
```

Uploading process: Upload the sketch to Arduino board as described in Pages 7-10. After uploading, make sure you close the serial window on the Arduino IDE.

After successful uploading of code in Arduino, Open putty software in the pc and select the “serial” instead of ssh

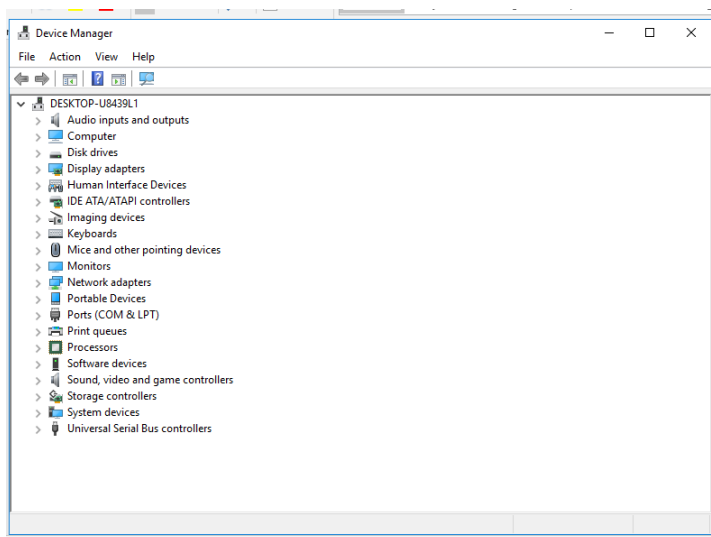
Type the COM port number using your device manager. Press

Windows key +R in the keyboard

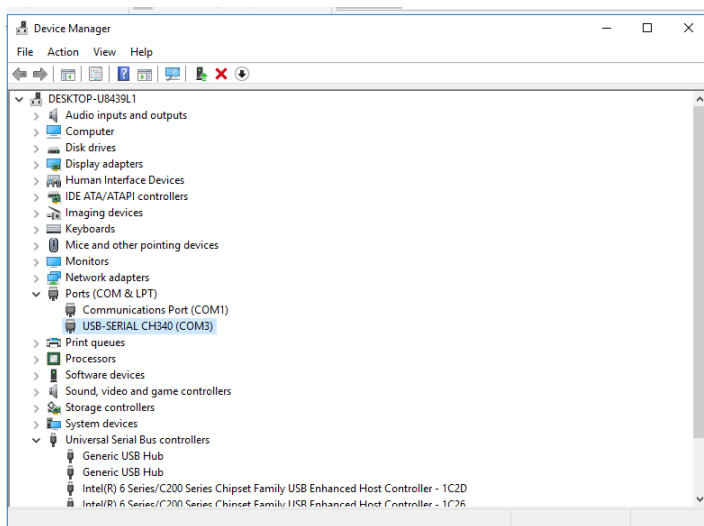


Opens the Run command prompt.

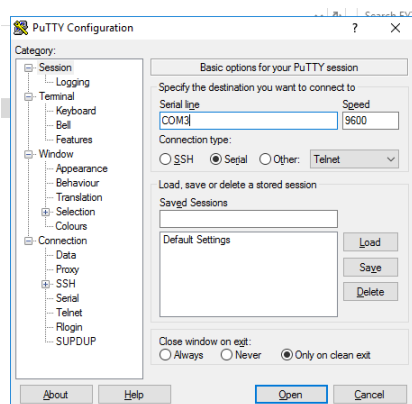
Now type “devmgmt.msc” (without inverted commas) in the dialog box.



Look at Ports(Com &LPT) and double click on it

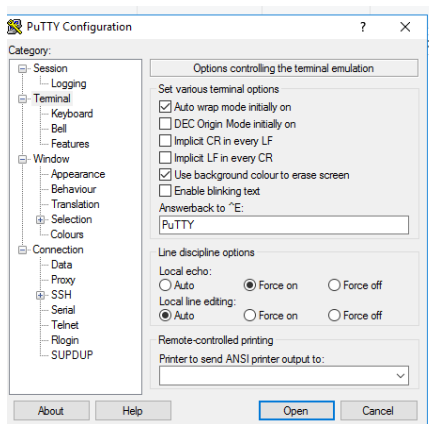


In my case the com number is 3 [USB-SERIAL CH340(COM3)]In the putty software enter the number 3 for com port

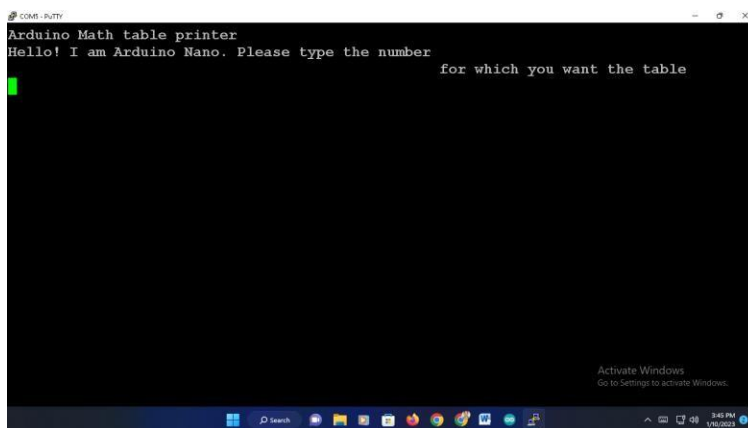


Select “Terminal” under “category” and select the “Line discipline options” ,“Local Echo”....
Select “Force on”

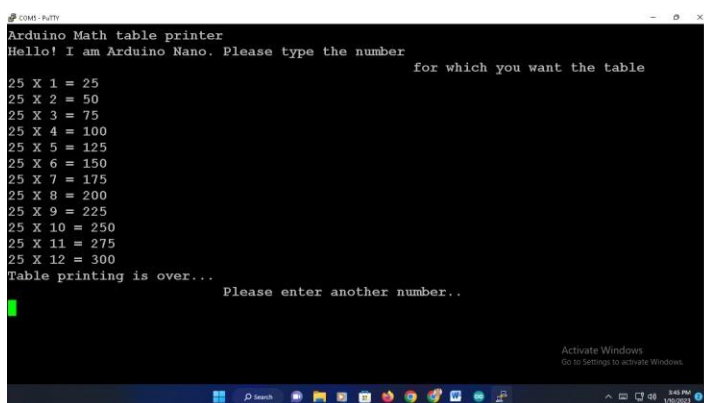
Click open.



The following screen shots will show the output in the terminal window



After typing 25, The output is



Program:2

```
// This program makes bidirectional serial
communication void banner(){// prints the banner
statement
Serial.println("*****")
```

```

); Serial.println("Arduino Math table printer");
Serial.println ("Hello! I am Arduino Nano. \nPlease type the number for following options");

Serial.println ("1.\tLight sensor reading\n2.\tLM35 reading\n3.\tThermistor reading\n");
}

void setup() {
Serial.begin(9600);// start serial communication on arduino with pc
banner();// call banner function
}

void loop() {
if(Serial.available(>0){ // when any serial input is
given int val = Serial.parseInt();// read the number
given by user

if(val==1){// ldr
  int light = analogRead(A6)/10;// read the ldr light
(darkness)in % Serial.println("Darkness in room is "+ String
(light) + "%"); banner();// show options
} // close ldr
loopif
(val==2){//
lm35

int temp35 = analogRead(A0);// read the A0 pin for lm35
o/p float temperature = temp35*500.00/1024.00;
  Serial.println("Temperature in deg Celsius is: " +
String(temperature)); banner();// show options
} // close lm35 loop
if(val==3){//thermistor
float y= analogRead(A1);// thermistor pin A1
float x=(1.00/298.15)+(1.00/3950.00)*log(1.00*y/(1024.00-y));// see below for
calculations float temp=1.000/x;
Serial.println("Thermistor reads"+String(temp-273.15)+String
("C")); banner();// show options
} // close thermistor loop

} // end if(Serial.available(>0))
} // end main loop

```

Refer Appendix for detailed program for thermistor

The output for the above program (after successful uploading of code as said above)

```
COM5 - PuTTY
*****
Arduino Math table printer
Hello! I am Arduino Nano.
Please type the number for following options
1.      Light sensor reading
          2.  LM35 reading
          3.  Thermistor reading
```

Entering 1 gives the following

```
COM5 - PuTTY
*****
Arduino Math table printer
Hello! I am Arduino Nano.
Please type the number for following options
1.      Light sensor reading
          2.  LM35 reading
          3.  Thermistor reading

Darkness in room is 53%
*****
Arduino Math table printer
Hello! I am Arduino Nano.
Please type the number for following options
1.      Light sensor reading
          2.  LM35 reading
          3.  Thermistor reading
```

Darkness in the room is 53%

Entering 2 gives the following:


```
COM5 - PuTTY

                2. LM35 reading
                  3. Thermistor reading

Darkness in room is 53%
*****
Arduino Math table printer
Hello! I am Arduino Nano.

                Please type the number for following options
1.      Light sensor reading
                2. LM35 reading
                  3. Thermistor reading

Temperature in deg Celsius is: 26.37
*****
Arduino Math table printer
Hello! I am Arduino Nano.

                Please type the number for following options
1.      Light sensor reading
                2. LM35 reading
                  3. Thermistor reading

Activate Windows
Go to Settings to activate Windows.
```

Temperature in deg Celsius is: 26.37

Thermistor reads 29.75C

```
COM5 - PuTTY

                2. LM35 reading
                  3. Thermistor reading

Temperature in deg Celsius is: 26.37
*****
Arduino Math table printer
Hello! I am Arduino Nano.

                Please type the number for following options
1.      Light sensor reading
                2. LM35 reading
                  3. Thermistor reading

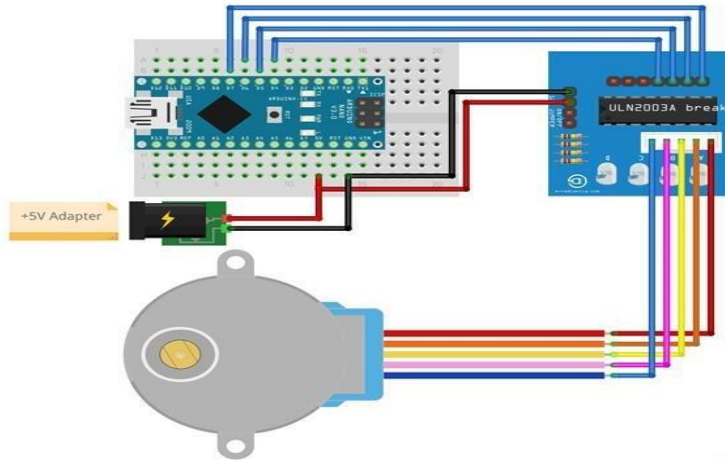
Thermistor reads 29.75C
*****
Arduino Math table printer
Hello! I am Arduino Nano.

                Please type the number for following options
1.      Light sensor reading
                2. LM35 reading
                  3. Thermistor reading

Activate Windows
Go to Settings to activate Windows.
```

Program To Drive Stepper motor using Analog GPIOs

Circuit connections:



Pin (Arduino)	pin (uln2003)	pin (stepper)
D8	O/P 1	COIL 1
D7	O/P 2	COIL 2
D6	O/P 3	COIL 3
D5	O/P 4	COIL 4

Program:

```
#include <Stepper.h>
#define steps_per_rev
32
Stepper stepper(32,5,7,6,8);// THE MOTOR GIVEN HAS 32 STEPS FOR REVOLUTION
//5,7,6,8 ARE THE DIGITAL PINS OF ARDUINO WHICH CONTROL COILS 4 & 2 ,3&1
RESPECTIVELY

int previous = 0;

void setup() {
  Serial.begin(9600);
  stepper.setSpeed(90);
}
float val=0;
void
getval(){
  for (int i=0;i<40;i++){
    val=(val+analogRead());
```

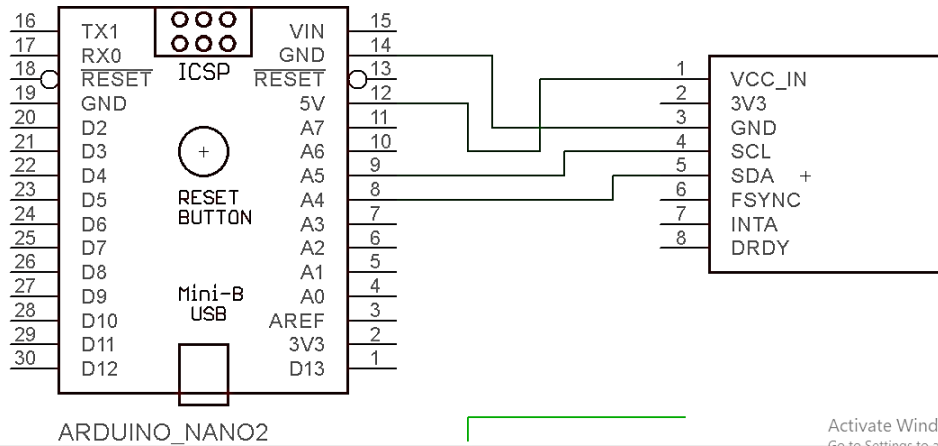
```
    delay(5)
  }
  val/=40;
}

void loop() {
  getval();
  Serial.println(vl)
  stepper.step(val - previous);
  previous = val; // remember the previous value of the sensor
}
```

Uploading process: Upload the sketch to Arduino board as described in Pages7-10.
After successful upload, adjust the pot1 on the board which controls thestepper motor
rotation clockwise/ anticlockwise.

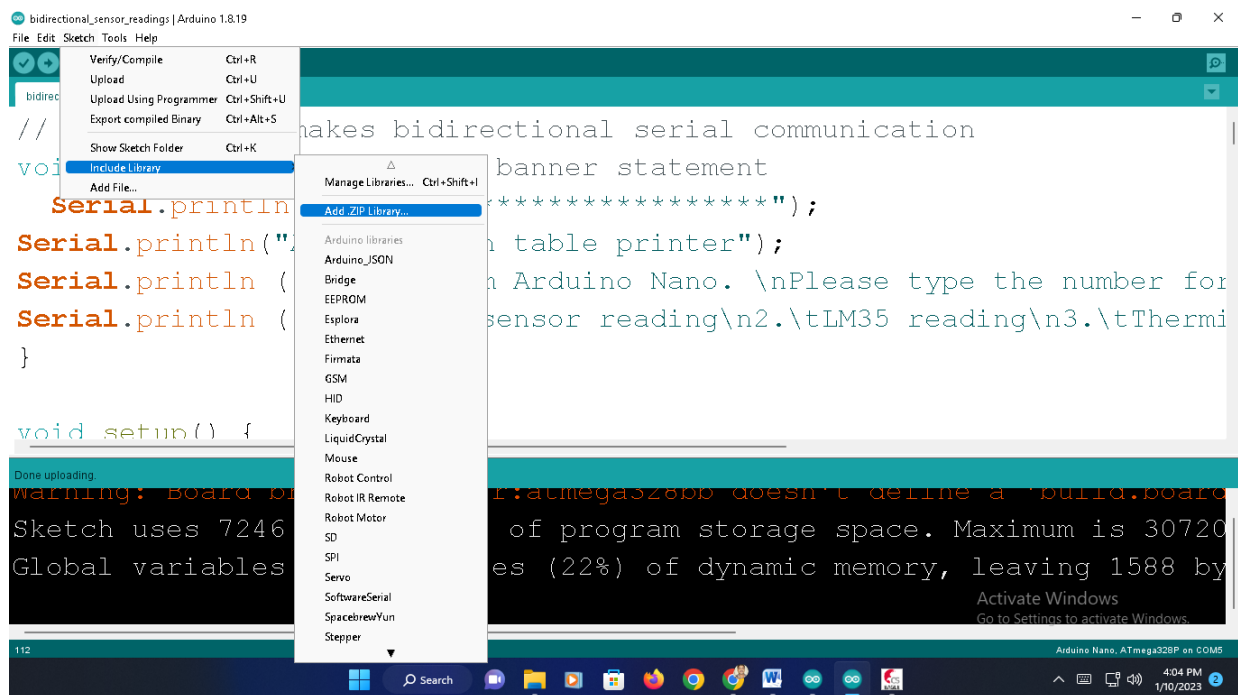
Program To Drive Accelerometer and Display the readings on Hyper Terminal

Circuit diagram:



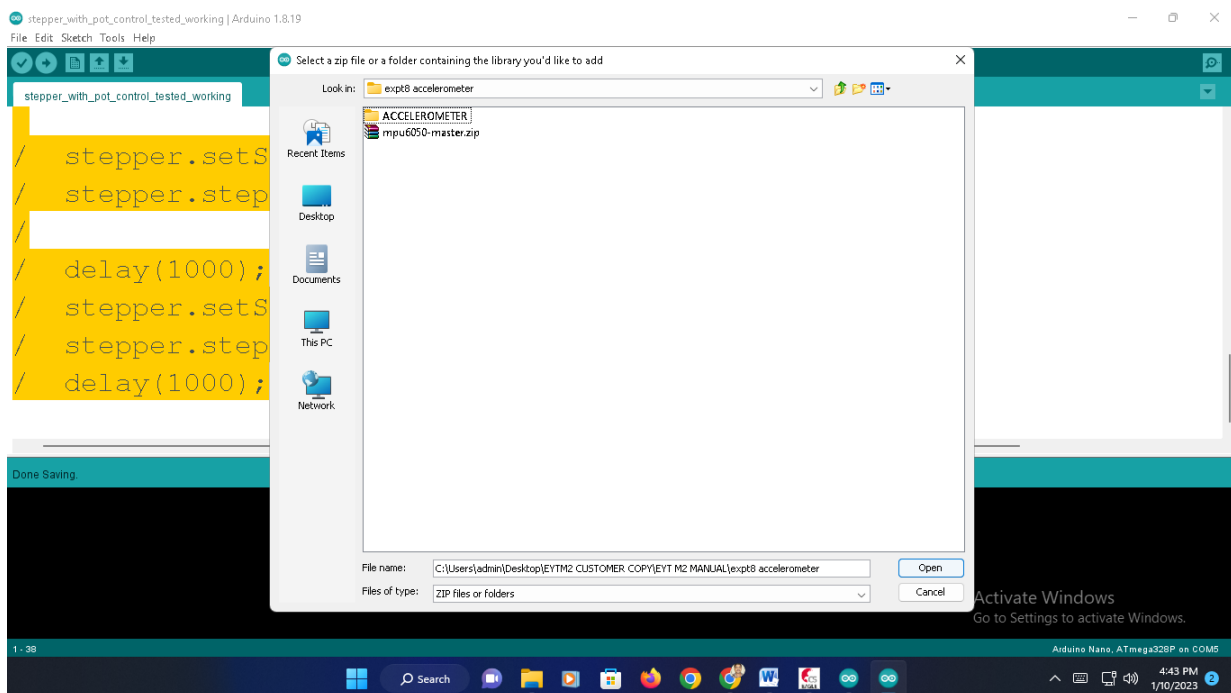
Installation of library:

Go to Tools/Library/Add.Zip Library option



Select the path to your zip file, it will be

EMY2 CUSTOMER COPY\EYT M2 PROGRAMS 9 JAN 2023 FINAL\AS
PER JNTUK SYLLABUS\expt8 accelerometer\mpu6050-master.zip



Click open to add the library to Arduino IDE. Then open the
program expt8/ accelerometer.ino

// INSTALL THE ZIP LIBRARY FOR MPU6050 FROM THE FOLDER

```
//#include "I2Cdev.h"  
#include "MPU6050.h"  
#include "Wire.h"
```

```
MPU6050 accelgyro;  
//MPU6050 accelgyro(0x69);  
int16_t ax, ay, az;  
int16_t gx, gy, gz;  
#define OUTPUT_READABLE_ACCELGYRO  
void setup() {  
  Wire.begin();  
  Serial.begin(9600);  
  Serial.println("Initializing I2C  
  devices..."); accelgyro.initialize();  
}
```

```
void loop() {  
  
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  
  
  Serial.print("a/g:\t");  
  Serial.print(ax);  
  Serial.print("\t");  
}
```

```

Serial.print(ay);
Serial.print("\t");
Serial.print(az);
Serial.print("\t");
Serial.print(gx);
Serial.print("\t");
Serial.print(gy);
Serial.print("\t");
Serial.println(gz);
}

```

Upload the above program to Arduino and open the serial plotter (CTRL+SHIFT+L)

The output will be the following



Tilt and shake the accelerometer, to see the variation of acceleration in the serial plotter. If you want to see raw data in Hyper terminal (Putty), close the serial window. Open putty and choose the correct serial port (COM4,6 etc). choose baud rate of 9600 only. Then the output will be the following

```
COM5 - PuTTY
-2Initializing I2C devices...
a/g: 1988 -1396 15764 -292 -52 -85
a/g: 1920 -1236 15548 -265 -74 -112
a/g: 1984 -1452 15700 -252 -63 -113
a/g: 2028 -1272 15616 -311 -113 -99
a/g: 1940 -1328 15652 -312 -70 -101
a/g: 1912 -1276 15620 -268 -60 -109
a/g: 1992 -1328 15468 -261 -34 -120
a/g: 2008 -1292 15676 -246 -37 -106
a/g: 1880 -1280 15700 -254 -44 -112
a/g: 1932 -1208 15592 -257 -46 -120
a/g: 1940 -1392 15544 -296 -56 -116
a/g: 1752 -1260 15712 -269 -76 -100
a/g: 1808 -1340 15580 -288 -81 -100
a/g: 1976 -1404 15688 -290 -70 -103
a/g: 1828 -1312 15596 -277 -47
```

Activate Windows
Go to Settings to activate Windows.

Search | Taskbar icons: File Explorer, Microsoft Store, Firefox, Google Chrome, Word, Outlook, Teams, PuTTY | System tray: Network, Volume, 4:50 PM, 1/10/2023

