|  |  |  |  |
| --- | --- | --- | --- |
| bitm light | **BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT** | | |
| **Quality Manual**  (ISO 9001:2015) | Doc. No: **WI/L3** | Release No. **5.0**  Date: **01/07/2017** | Section No: **R/WI DSPL/01** |

OMoARcPSD|17071726

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**



INTRODUCTION TO HARDWARE CONTROLLERS

LAB MANUAL

**[ Course Code:21AEC34 ]**

**Prepared By**

**Dr Renuka sagar**

**Assoc Prof**

**Department of Electronics and Communication Engineering**

**List of Experiments**

**Course: Introduction to Hardware controllers**

**Part-A Basic Arduino Programming**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. NO** | **NAME OF THE EXPERIMENT** | | **DATE** | **SIGN** |
| **1** | Getting started with arduino :arduino platform, prototyping environment | |  |  |
| **2** |  | |  |  |
| Arduino IDE: Arduino development Environment, setting up arduino board with electronic components and connections. |  |
| **3** | Arduino First program: Creating sketches, using Libraries,using example codes, Debugging using serial Monitor. | |  |  |
| **4** | Arduino interfaces-Different sensors & Actuators | |  |  |
| **Part-B Basic Raspberrypi Programming** | | | | |
| **5** | Getting started with Raspberrypi Basic functionality of the raspberrypi board and its processor, setting and configuring the board | |  |  |
| **6** | Introduction to Linux: overview of Linux and its terminal commands for operating Raspberrypi | |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **7** | Programming the Raspberrypi: python-Introducing to python programming language & python programming Environment |  |  |
| **8** | Exploring Electronics with Raspberrypi: sensors & Actuator interfacing. |  |  |
| **Part C: open Ended Experiments/Mini-project (only for CIE,not for SEE)** | | | |

**Experiment-1**

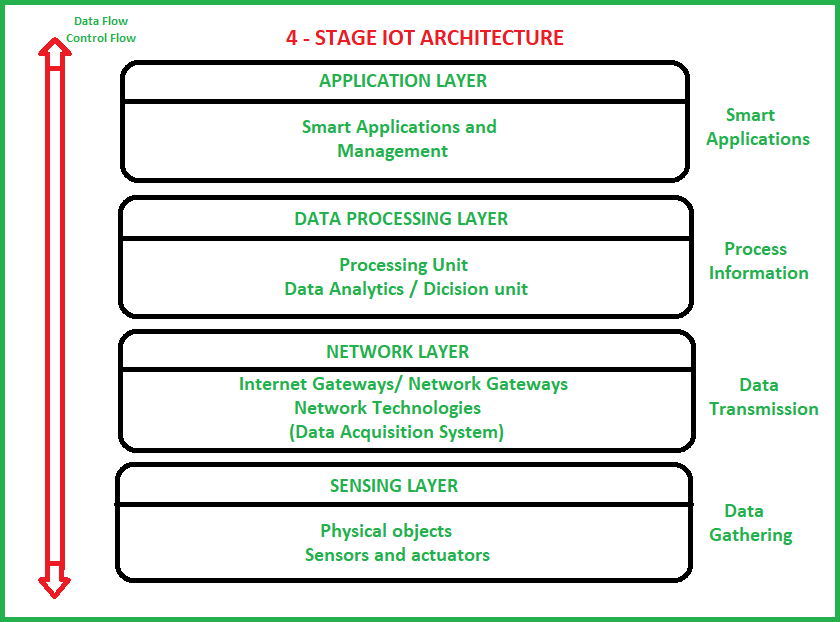
**AIM: Getting started with Arduino: Arduino platform, prototyping environment**

Introduction to IOT Architecture and its components.

# Description:

Internet of Things (IOT) technology has a wide variety of applications and use of Internet of Things is growing so faster.

The architecture of IoT depends upon its functionality and implementation in different sectors. Still, there is a basic process flow based on which IoT is built.Here in this article we will discuss basic fundamental architecture of IoT i.e., 4 Stage IoT architecture.So, from the above image it is clear that there is 4 layers are present that can be divided as follows: Sensing Layer, Network Layer, Data processing Layer, and Application Layer.



These are explained as following below.

# Sensing Layer –

Sensors, actuators, devices are present in this Sensing layer. These Sensors or Actuators accepts data(physical/environmental parameters), processes data and emits data over network.

# Network Layer –

Internet/Network gateways, Data Acquisition System (DAS) are present in this layer. DAS performs data aggregation and conversion function (Collecting data and aggregating data then converting analog data of sensors to digital da[ta etc). Advanced gateways which mainly opens up connection](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=190130107040-ce-a-iot-help) between Sensor networks and Internet also performs many basic gateway functionalities like malware protection, and filtering also some times decision making based on inputted data and data management services, etc.

# Data processing Layer –

This is processing unit of IoT ecosystem. Here data is analyzed and pre-processed before sending it to data center from where data is accessed by software applications often termed as business applications where data is monitored and managed and further actions are also prepared. So here Edge IT or edge analytics comes into picture.

# Application Layer –

This is last layer of 4 stages of IoT architecture. Data centers or cloud is management stage of data where data is managed and is used by end-user applications like agriculture, health care, aerospace, farming, defense, etc.

The best place to start is to define what an architecture is. Fundamentally, an architecture is a diagram or model that comprises two parts:

the key technology components that make it up and the relationship between those components.

**Getting Started**

Download Arduino IDE

The Arduino IDE (Integrated Development Environment) allows you to write programs and

upload them to your Arduino board. Arduino also offers a web editor service called Arduino

Create (https://www.arduino.cc). You can use the web editor if you run into issues with the

IDE, but for this course, we will be using the IDE. Download the latest version of the IDE from

arduino.cc/download. Do not download the Arduino app from the app store.

Windows Installation

1. Download the Windows Installer. Do not perform a non-admin install or download

the Windows app.

2. After finishing the download, navigate to your download path and double-click on

“arduino-1.x.x-rx-windows.exe” (where the x depends on the version you

downloaded). If a security warning window shows up, click on “Run” or “Allow” and

accept the License Agreement. Click on “next” to choose the folder to install the IDE

and click on “Install”.

3. Connect the Arduino board to the computer using a proper USB cable. The board will

automatically draw power from the USB connection of the computer and the green

LED (labeled ON) will turn on.

4. Windows should initiate its driver installation process when the board is plugged in.

In some cases your computer won’t be able to find the drivers by itself. If so, you need

to point it to the proper folder.

 Windows XP: If Windows Update asks about the path for the software, select

“Yes, for this time only” and the “Install from a list or specific location”

 Other: A popup window asks you to install the driver automatically or to look

for it in the computer, choose to look for the driver on your computer.

5. If the installation doesn’t start automatically, click on the Start Menu and open the

device manager.

6. Look for the Arduino device under the category “Other Devices” or “Unknown

Devices” and select “Update Driver” or “Update Driver Software” clicking with the

right button of the mouse.

7. Click on “browse” and select the “Drivers” folder (not the folder “FTDI USB Drivers”)

in the Arduino software IDE folder. Press “OK” and “Next”. If a dialog box about a test

on the Windows Logo shows up, click on “Continue Anyway”. Windows will now

install the driver.

8. In the “Device Manager”, under “Ports (COM & LPT)”, you should see a port similar to

“Arduino Uno (COM4)”.

inside the “Arduino”folder of your PC. If you cannot find the “Arduino”folder in the “Program

Files x86” folder, you may have installed the Arduino IDE incorrectly.

Sample file path: Local Disk (C:) > Program Files (x86) > Arduino > Libraries

This folder is located in a similar location on a Mac. Search your “Library” folder within

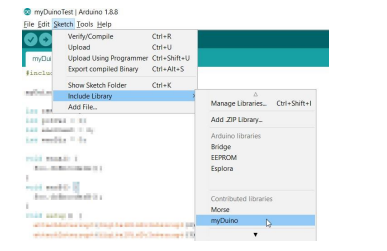
Finder until you find the folder titled “Arduino.” Open this folder and paste the myDuino

folder into the “Libraries” folder inside it.

If the Arduino IDE is already open, you need to close and reopen it for the changes to take

effect. In the toolbar, check Sketch > Include Library for the greyed-out heading “Contributed

libraries” to make sure myDuino shows up under it.

****

**Experiment-2**

**AIM : Setting up the Arduino board with electronic components and connections**

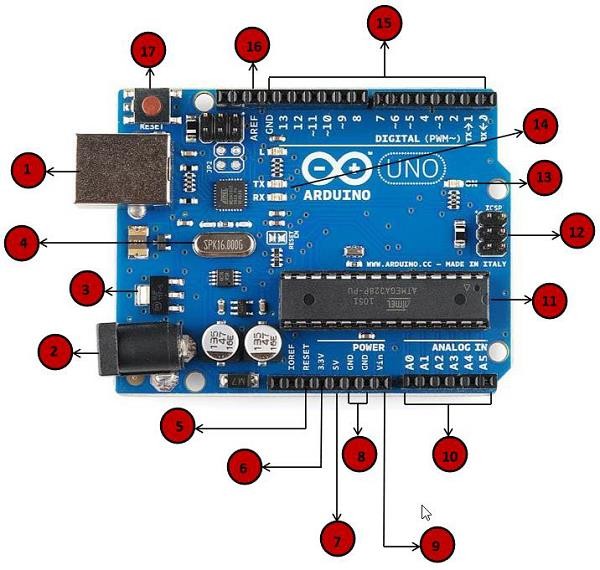
# Description:

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 programed (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.

# Daigram:



# Board Description:

Arduino consists of following components listed below:

## Power USB:

Arduino board can be powered by using the USB cable from your computer.All you need to do is connect the USB.

## Power (Barrel Jack):

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

## 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.

## 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.

## 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

1. **3.3V** :Supply 3.3 output volt
2. **5V :**Supply 5 output volt

## GND:

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

## Vin:

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

## Analog pins:

The Arduino UNO board has six 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.

## Main microcontroller:

Each Arduino board has its own microcontroller . 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.

## 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.

## 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 . The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

## Digital I/O:

The Arduino UNO board has 14 digital I/O pins (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.

## 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.

**Experiment-3**

**Aim: Arduino First program: creating sketches, using libraries, using example codes, Debugging using serial Monitor**.

void setup() {

int a = 2;

int b = 7;

int result;

float result\_fl;

Serial.begin(9600);

Serial.print("Addition (a + b): ");

result = a + b;

Serial.println(result);

Serial.print("Subtraction (10 - 2): ");

result = 10 - 2;

Serial.println(result);

Serial.print("Multiplication (4 \* 3): ");

result = 4 \* 3;

Serial.println(result);

Serial.print("Int Division (5 / 4): ");

result = 5 / 4;

Serial.println(result);

Serial.print("Float Division (5.0 / 4.0): ");

result\_fl = 5.0 / 4.0;

Serial.println(result\_fl);

Serial.print("Remainder (11 % 4): ");

result = 11 % 4;

Serial.println(result);

}

void loop() {

}

**Experiment-4**

**AIM :Arduino Interfaces:- Introduction to various types of Sensors and Actuators.**

# Description:

We live in a World of Sensors. You can find different types of Sensors in our homes, offices, cars etc. working to make our lives easier by turning on the lights by detecting our presence, adjusting the room temperature, detect smoke or fire, make us delicious coffee, open garage doors as soon as our car is near the door and many other tasks.All these and many other automation tasks are possible because of Sensors. Before going in to the details of What is a Sensor, What are the Different Types of Sensors and Applications of these different types of Sensors, we will first take a look at a simple example of an automated system, which is possible because of Sensors (and many other components as well).An Automatic Flight Control System consists of several sensors for various tasks like speed control, height monitoring, position tracking, status of doors, obstacle detection, fuel level, maneuvering and many more. A Computer takes data from all these sensors and processes them by comparing them with pre-designed values.The computer then provides control signals to different parts like engines, flaps, rudders, motors etc. that help in a smooth flight. The combination of Sensors, Computers and Mechanics makes it possible to run the plane in Autopilot Mode.

# Different Types of Sensors

The following is a list of different types of sensors that are commonly used in various applications. All these sensors are used for measuring one of the physical properties like Temperature, Resistance, Capacitance, Conduction, Heat Transfer etc.

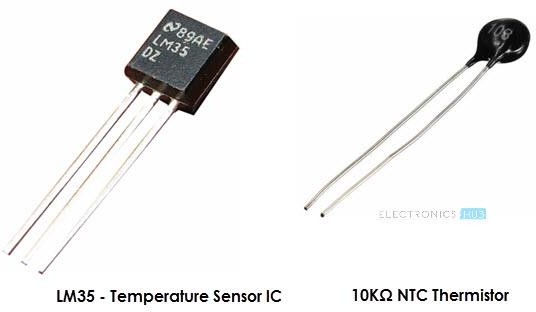
* 1. Temperature Sensor
  2. Proximity Sensor
  3. Accelerometer
  4. IR Sensor (Infrared Sensor)
  5. Pressure Sensor
  6. Light Sensor
  7. Ultrasonic Sensor
  8. Smoke, Gas and Alcohol Sensor
  9. Touch Sensor
  10. Color Sensor
  11. Humidity Sensor
  12. Position Sensor
  13. Magnetic Sensor (Hall Effect Sensor)
  14. Microphone (Sound Sensor)
  15. Tilt Sensor

Flow and Level Sensor

1. PIR Sensor
2. Touch Sensor
3. Strain and Weight Sensor

We will see about few of the above-mentioned sensors in brief. More information about the sensors will be added subsequently. A list of projects using the above sensors is given at the end of the page.

**Temperature Sensor:**One of the most common and most popular sensors is the Temperature Sensor. A Temperature Sensor, as the name suggests, senses the temperature i.e., it measures the changes in the temperature.



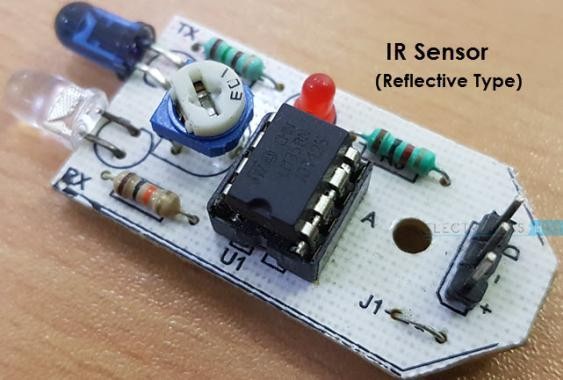
There are different types of Temperature Sensors like Temperature Sensor ICs (like LM35, DS18B20), Thermistors, Thermocouples, RTD (Resistive Temperature Devices), etc.Temperature Sensors can be analog or digital. In an Analog Temperature Sensor, the changes in the Temperature correspond to change in its physical property like resistance or voltage. LM35 is a classic Analog Temperature Sensor.Coming to the Digital Temperature Sensor, the output is a discrete digital value (usually, some numerical data after converting analog value to digital value). [DS18B20 is a simple Digital Temperature Sensor.](https://www.electronicshub.org/arduino-ds18b20-tutorial/)Temperature Sensors are used everywhere like computers, mobile phones, automobiles, air conditioning systems, industries etc.

**Proximity Sensors**:A Proximity Sensor is a non-contact type sensor that detects the presence of an object. Proximity Sensors can be implemented using different techniques like Optical (like Infrared or Laser), Sound (Ultrasonic), Magnetic (Hall Effect), Capacitive, etc.



Some of the applications of Proximity Sensors are Mobile Phones, Cars (Parking Sensors), industries (object alignment), Ground Proximity in Aircrafts, etc.

**Infrared Sensor (IR Sensor):**IR Sensors or Infrared Sensor are light based sensor that are used in various applications like Proximity and Object Detection. IR Sensors are used as proximity sensors in almost all mobile phones.



There are two types of Infrared or IR Sensors: Transmissive Type and Reflective Type. In Transmissive Type IR Sensor, the IR Transmitter (usually an IR LED) and the IR Detector (usually a Photo Diode) are positioned facing each other so that when an object passes between them, the sensor detects the object.The other type of IR Sensor is a Reflective Type IR Sensor. In this, the transmitter and the detector are positioned adjacent to each other facing the object. When an object comes in front of the sensor, the infrared light from the IR Transmitter is reflected from the object and is detected by the IR Receiver and thus the sensor detects the object.Different applications where IR Sensor is implemented are Mobile Phones, Robots, Industrial assembly, automobiles etc.

**Ultrasonic Sensor:**An Ultrasonic Sensor is a non-contact type device that can be used to measure distance as well as velocity of an object. An Ultrasonic Sensor works based on the properties of the sound waves with frequency greater than that of the human audible range.



Using the time of flight of the sound wave, an Ultrasonic Sensor can measure the distance of the object (similar to SONAR). The Doppler Shift property of the sound wave is used to measure the velocity of an object.

**Light Sensor**:Sometimes also known as Photo Sensors, Light Sensors are one of the important sensors. A simple Light Sensor available today is the Light Dependent Resistor or LDR. The property of LDR is that its resistance is inversely proportional to the intensity of the ambient light i.e., when the intensity of light increases, its resistance decreases and vise-versa.By using LDR is a circuit, we can calibrate the changes in its resistance to measure the intensity of Light. There are two other Light Sensors (or Photo Sensors) which are often used in complex electronic system design. They are Photo Diode and [Photo Transistor](https://www.electronicshub.org/basics-of-phototransistor/). All these are Analog Sensors.



There are also Digital Light Sensors like BH1750, TSL2561, etc., which can calculate intensity of light and provide a digital equivalent value.Check out this simple [LIGHT DETECTOR USING LDR](https://www.electronicshub.org/light-detector-using-ldr/) project.

**Smoke and Gas Sensors**:One of the very useful sensors in safety related applications are Smoke and Gas Sensors. Almost all offices and industries are equipped with several smoke detectors, which detect any smoke (due to fire) and sound an alarm.Gas Sensors are more common in laboratories, large scale kitchens and industries. They can detect different gases like LPG, Propane, Butane, Methane (CH4), etc.

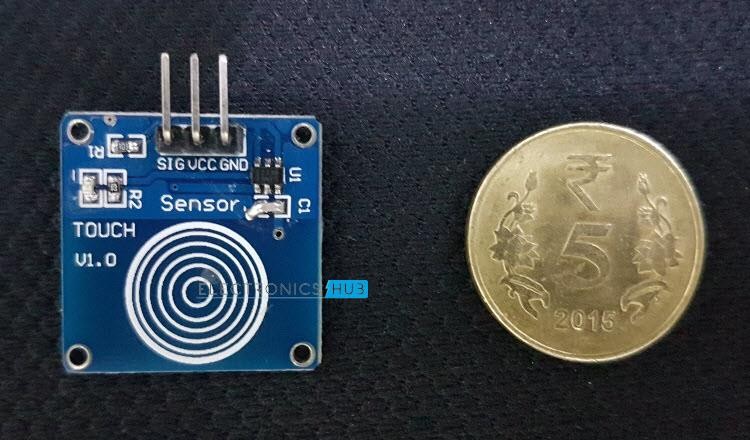


Now-a-days, smoke sensors (which often can detect smoke as well gas) are also installed in most homes as a safety measure.The “MQ” series of sensors are a bunch of cheap sensors for detecting CO, CO2, CH4, Alcohol, Propane, Butane, LPG etc. You can use these sensors to build your own Smoke Sensor Application.

**Alcohol Sensor**:As the name suggests, an Alcohol Sensor detects alcohol. Usually, alcohol sensors are used in breathalyzer devices, which determine whether a person is drunk or not. Law enforcement personnel uses breathalyzers to catch drunk-and-drive culprits.

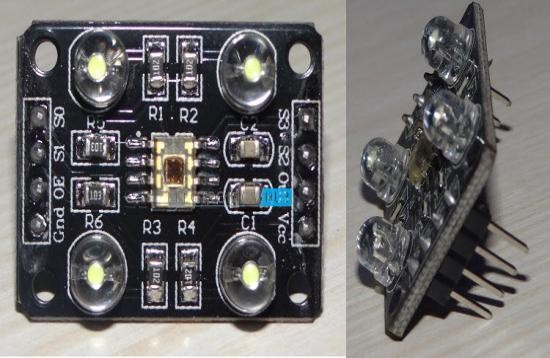


**Touch Sensor:**We do not give much importance to touch sensors but they became an integral part of our life. Whether you know or not, all touch screen devices (Mobile Phones, Tablets, Laptops, etc.) have touch sensors in them. Another common application of touch sensor is trackpads in our laptops.

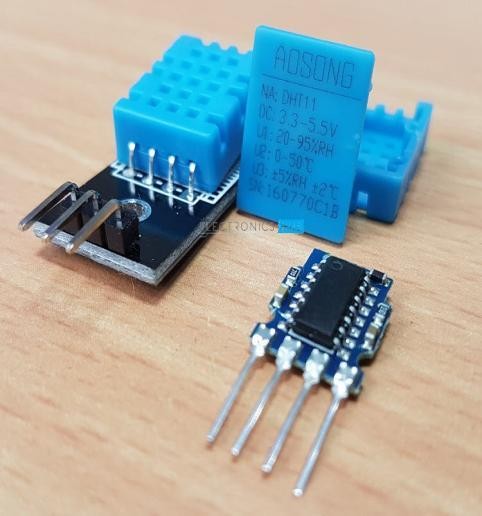


Touch Sensors, as the name suggests, detect touch of a finger or a stylus. Often touch sensors are classified into Resistive and Capacitive type. Almost all modern touch sensors are of Capacitive Types as they are more accurate and have better signal to noise ratio.

**Color Sensor:**A Color Sensor is an useful device in building color sensing applications in the field of image processing, color identification, industrial object tracking etc. The TCS3200 is a simple Color Sensor, which can detect any color and output a square wave proportional to the wavelength of the detected color.

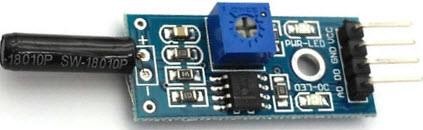


**Humidity Sensor:**If you see Weather Monitoring Systems, they often provide temperature as well as humidity data. So, measuring humidity is an important task in many applications and Humidity Sensors help us in achieving this.Often all humidity sensors measure relative humidity (a ratio of water content in air to maximum potential of air to hold water). Since relative humidity is dependent on temperature of air, almost all Humidity Sensors can also measure Temperature.



Humidity Sensors are classified into Capacitive Type, Resistive Type and Thermal Conductive Type. DHT11 and DHT22 are two of the frequently used Humidity Sensors in DIY Community (the former is a resistive type while the latter is capacitive type).

**Tilt Sensor:**Often used to detect inclination or orientation, Tilt Sensors are one of the simplest and inexpensive sensors out there. Previously, tilt sensors are made up of Mercury (and hence they are sometimes called as Mercury Switches) but most modern tilt sensors contain a roller ball.



AIM : Interface LED with Arduino/Raspberry Pi and write a pogram to turn on LED. LED should be turn on for 1 Second after every 2 Second.

Description:

Components:

The components used are :

1. Arduino Uno R3

2. BreadBoard

3. LED

4. Resistor

Arduino Uno R3:

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Pins used:

Digital pin 13:The pin13 is connected to the Anode terminal of the Led through a Resistor on Breadboard. GND:The Ground is connected to the Cathode of Led.

Breadboard:

A Breadboard is simply a board for prototyping or building circuits on. It allows you to place components and connections on the board to make circuits without soldering. The holes in the breadboard take care of your connections by physically holding onto parts or wires where you put them and electrically connecting them inside the board.

LED:

Led has two terminals Anode and Cathode .The shorter leg toward the flat edge of Led is the Negative terminal of the Led.

Resistor:

LEDs need resistors to help limit the current that passes through them so they do not get damaged. Every LED has a current rating that should not be exceeded and resistors have the ability to limit the current to below the maximum allowable current allowed for the LED.

Components Required:

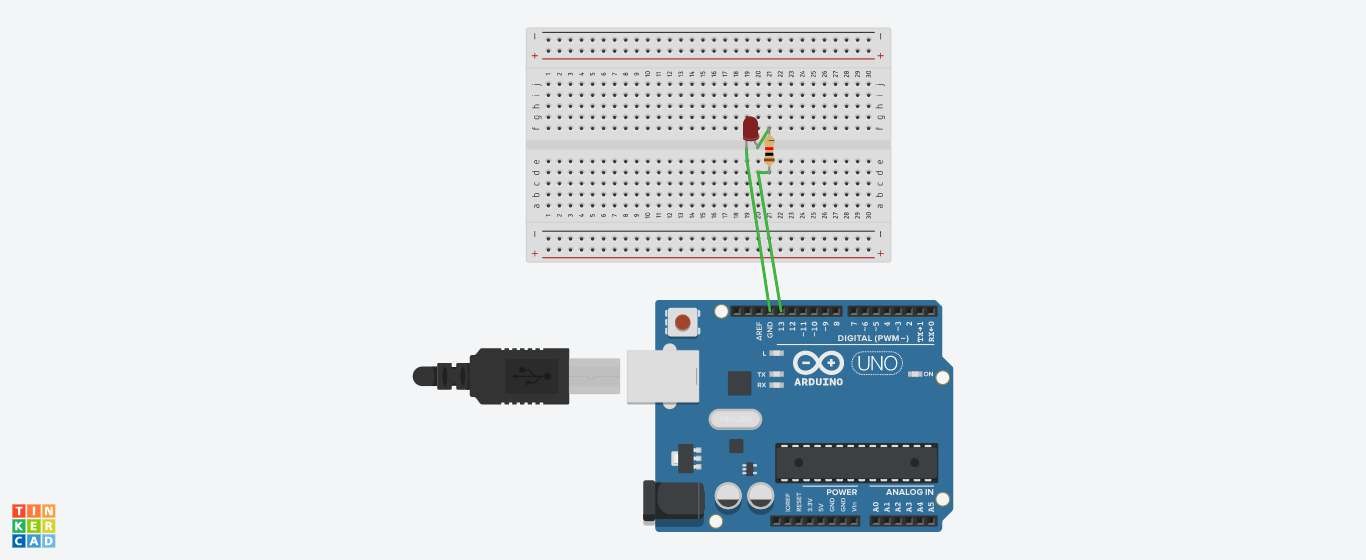
• 1 x Arduino

• 1 x Breadboard

• 1 x Resistor

* 1xLED

**circuit**

****

**code:**

void setup()

{

pinMode(13, OUTPUT);

}

void loop()

{

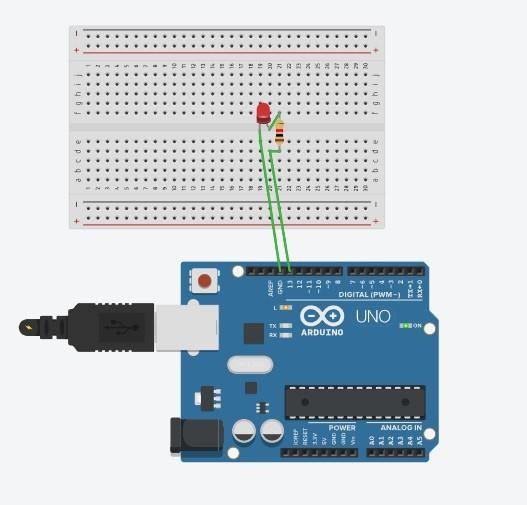
digitalWrite(13, HIGH);

delay(1000); // Wait for 1000 millisecond(s) digitalWrite(13, LOW);

delay(2000); // Wait for 200 millisecond(s)

}

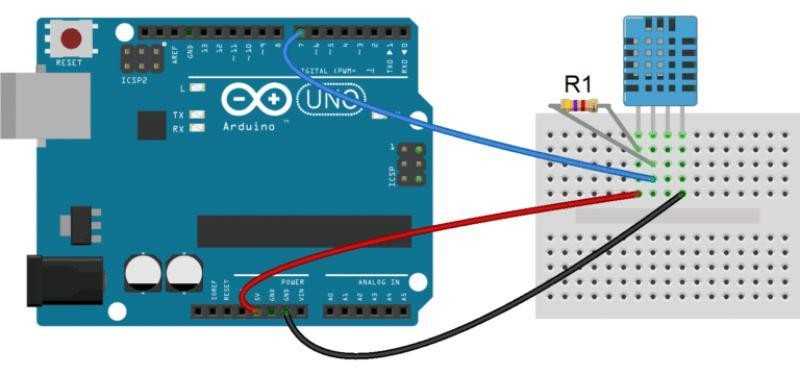
**output:**

****

**AIM : Interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print its readings.**

## Component View :

|  |  |  |
| --- | --- | --- |
| **NAME** | **QUANTITY** | **COMPONENT** |
| U1 | 1 | Arduino Uno R3 |
| U2 | 1 | DHT11 Sensor |
| R1 | 1 | 220 ohm Resistor |



## Code :

#include <dht.h> dh[t DHT;](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=190130107040-ce-a-iot-help)

#define DHT11\_PIN 7 void setup(){

Serial.begin(9600);

}

void loop(){

int chk = DHT.read11(DHT11\_PIN); Serial.print("Humidity = "); Serial.println(DHT.humidity); Serial.print("Temperature = "); erial.println(DHT.temperature); delay(1000);

}

## OUTPUT:-

Humidity = 15.12% Temperature = 40.17 C 104.30 F

Humidity = 15.12% Temperature = 40.17 C 104.30 F

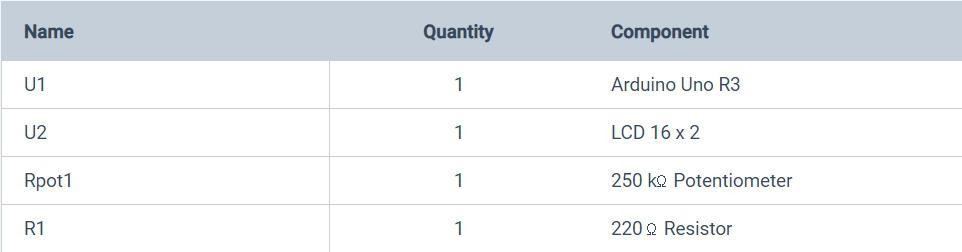
Humidity = 15.15% Temperature = 40.56 C 105.01 F

Humidity = 15.12% Temperature = 40.17 C 104.30 F

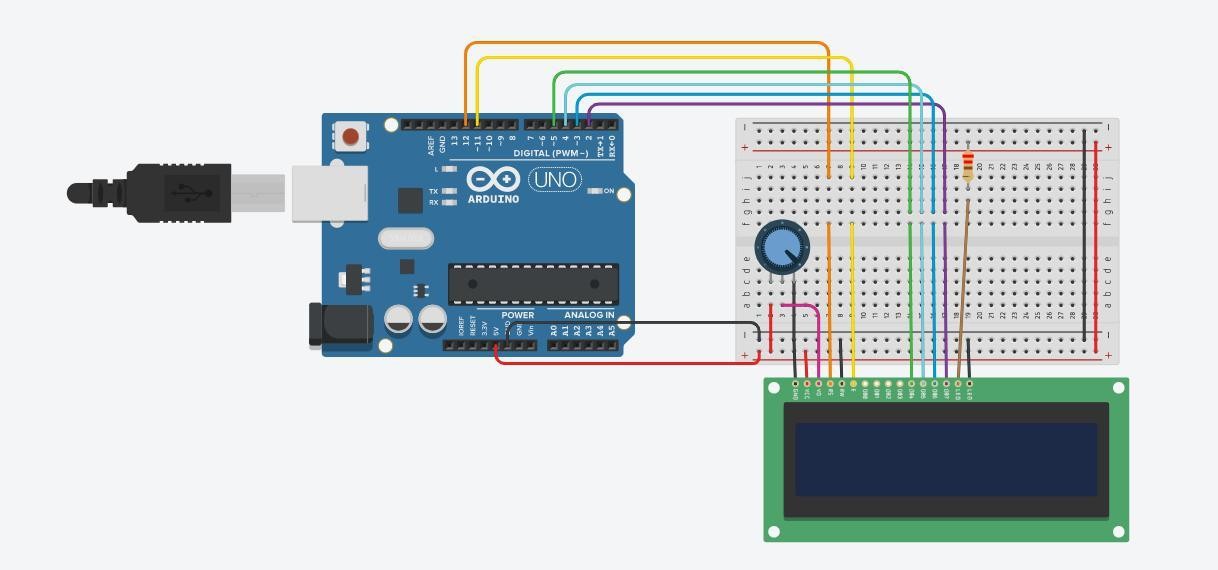
Humidity = 15.12% Temperature = 40.17 C 104.30 F

Humidity = 15.12% Temperature = 40.17 C 104.30 F

**AIM : Interface 16\*2 LCD with Arduino/ Raspberry Pi and write a program to print your name and simulation window second time duration on LCD.**

**Component view:**

**circuit view**

****

## Code :

//include the library code:

#include <LiquidCrystal.h>

//initialize the library with the numbers of the interface pins LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {

//set up the LCD's number of columns and rows: lcd.begin(10, 2);

//Print a message to the LCD. lcd.print("HARSHIKESH");

}

void loop() {

// set the cursor to column 0, line 1

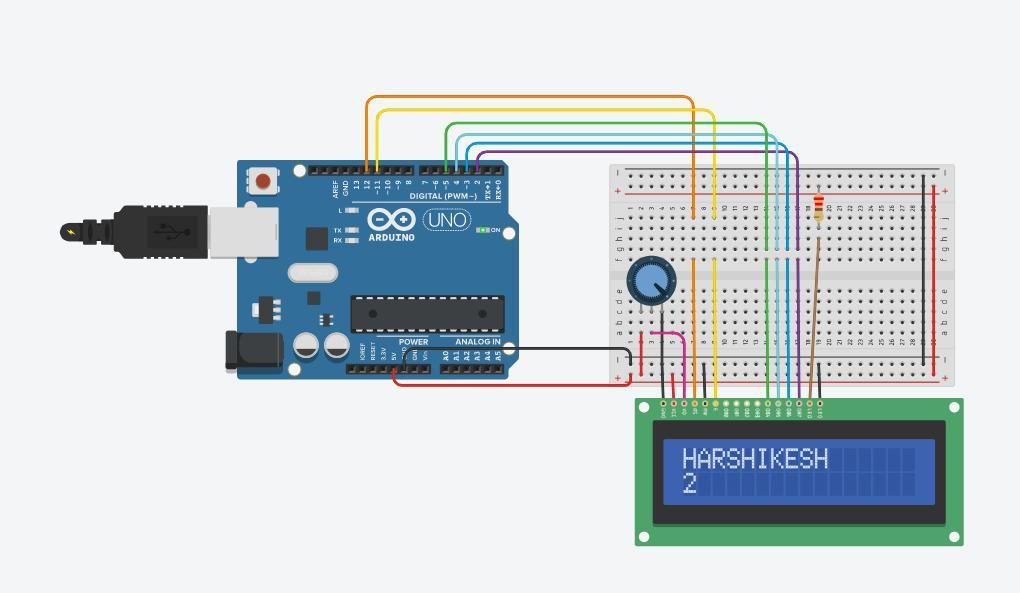
//(note: line 1 is the second row, since counting begins with 0): lcd.setCursor(0, 1);

// print the number of seconds since reset: lcd.print(millis() / 1000);

}

## Simulation :

When you turn the potentiometer , the LCD 16\*2 display the “HARSHIKESH”.

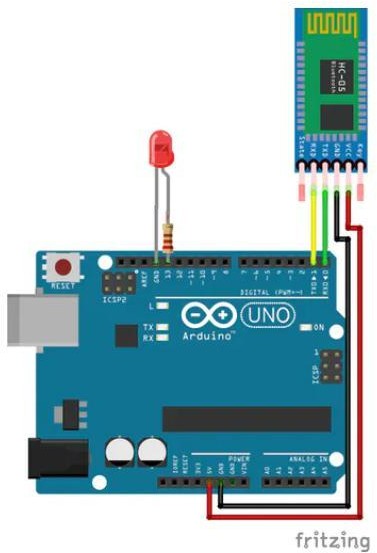


**AIM :To interface Bluetooth with Arduino/Raspberry Pi and write a program to send sensor data to smartphone using Bluetooth.**

## Component View :

|  |  |  |
| --- | --- | --- |
| **NAME** | **QUANTITY** | **COMPONENT** |
| U1 | 1 | Arduino Uno R3 |
| U2 | 1 | HC-05 Bluetooth Module |
| R1 | 1 | 221 OHM Resistor |
| D1 | 1 | Red LED |
| U3 | 1 | Android Device |

**Circuit View**



## Code :

char Incoming\_value = 0; void setup() { Serial.begin(9600); pinMode(13, OUTPUT);

}

void loop(){ if(Serial.available() > 0) { Incoming\_value = Serial.read();

Serial.print(Incoming\_value); Serial.print("\n"); if(Incoming\_value == '1') digitalWrite(13, HIGH);

else if(Incoming\_value == '0') digitalWrite(13, LOW);

}

}

## Simulation :

**How to connect your andriod device with bluebooth module**

* Pair your device with HC 05/06 Bluetooth module1) Turn ON HC 05/06 Bluetooth module2) Scan for availa[ble device3) Pair to HC 05/06 by entering default password](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=190130107040-ce-a-iot-help) 1234 OR 0000
* D application on your android device
* Open the Application



* Press paired devices
* Select your Bluetooth module from the List (HC 05)



* After connecting successfully
* Press ON button to turn ON LED and OFF button to turn OFF the LED



**Part-B Basic Raspberry pi Programming**

**Experiment-5**

**AIM :** Demonstrate Raspberry Pi and its pins.

# Description:

Raspberry Pi is a small-sized computer used Linux operating system. It is mini size computer used mostly to run larger and smart programs to achieve output quickly. Raspberry Pi 4 B+ (RP4) is the lasted model developed by the company, which has all the required latest wired and wireless communications systems used in most of the smart projects. A single Raspberry Pi 4 comes to a Quad-Core processor but it has three different versions which give three different sizes of RAM. Pi 4 uses mini HDMI and it also has two ports for two 4K displays.

# Daigram:



**[Raspberry Pi]**

# Raspberry Pi Pin Configuration:

This section includes information on the pinout diagram and headers pins details with an application of each pin. The Raspberry Pi 4 can be used in the external embedded system to communicate. It has a total of 40 pins from which 28 are GPIO pins and the rest of them are power pins. GPIO pins don’t only perform the simple I/O functions. They could give the UART, SPI, and I2C communications. These communications are specific to every pin and all their function are discussed below:

## Power Supply Pins:

**Power In:** In Raspberry pi, there is two power in method, one is from the USB-C power port and the second one is from any 5V pin. The 5-volt pin is directly connected to the USB-C adapter port. The input on the 5V pin should be stable and according to its specifications. In the case of higher voltage, the device could get burned. 5V input pins will bypass any fuse and regulator in case of power input, so the power supply from 5V should according to its specification to avoid any kind of harm. The power input pin of the Raspberry Pi 4 is given below:

Pin2-6 —> +5V Pin6 —–> GND

**Power Out:** There are two types of power output pin in the Raspberry pi 4 3V3 and 5V. 5V is directly connected to the USB port but 3V3 is connected to through the regulator which gives the stable 3 volts output. All power out pins are given below:

3V3 – Pin1, Pin17

5V – Pin2, Pin6

**Ground:** Raspberry Pi 4 has multiple ground pin which is connected internally and any ground pin can be used by the power supply or external device to make the common ground. The list of the ground pins is given below:

1. Pin6
2. Pin9
3. Pin14
4. Pin20
5. Pin25
6. Pin30
7. Pin34
8. Pin39

## Digital Input/Output Pins:

Almost every device needs to input and output pins to communicate. In this device there are 28 GPIO pins are available these pins can be used for any digital input and output operating. The GPIO pins in the controller have some default values. GPIO pins from 0-9 will be at a HIGH state and from 10 to above the pins will be at a LOW state. All [those pins in the Raspberry Pi 4 are given below:](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=190130107040-ce-a-iot-help)

1. GPIO0 – Pin27
2. GPIO1 – Pin28
3. GPIO2 – Pin3
4. GPIO3 – Pin5
5. GPIO4 – Pin7
6. GPIO5 – Pin29
7. GPIO6 – Pin31
8. GPIO7 – Pin26
9. GPIO8 – Pin24
10. GPIO9 – Pin21
11. GPIO10 – Pin19
12. GPIO11 – Pin23
13. GPIO12 – Pin32
14. GPIO13 – Pin33
15. GPIO14 – Pin8
16. GPIO15 – Pin10
17. GPIO16 – Pin36
18. GPIO17 – Pin11
19. GPIO18 – Pin12
20. GPIO19 – Pin35
21. GPIO20 – Pin38
22. GPIO21 – Pin40
23. GPIO22 – Pin15
24. GPIO23 – Pin16
25. GPIO24 – Pin18
26. GPIO25 – Pin22
27. GPIO26 – Pin37
28. GPIO27 – Pin13

All GPIO pins in Raspberry Pi 4 are not only for input-output function. Each GPIO pin can be used as other functions, which will be specified through programming.

**Raspberry Pi UART Pins:**There are multiple kinds of serial communication and UART is one of them. It is quite popular because of its simple communication system and dependence on most of the software. There is multiple UART communication pin in the Raspberry pi 4 and all of them are given below:

1. TXD1 – GPIO14 – Pin8
2. RXD1 – GPIO15 – Pin10
3. TXD2 – GPIO0 – Pin27
4. RXD2 – GPIO1 – Pin28
5. TXD3 – GPIO5 – Pin29
6. RXD3 – GPIO4 – Pin7
7. TXD4 – GPIO8 – Pin24
8. RXD4 – GPIO9 – Pin21
9. TXD5 – GPIO12 – Pin32
10. RXD5 – GPIO13 – Pin33

## SPI Communication Pins:

Some devices use SPI protocol and it could help the controlling device to control multiple devices using single data transmissions wire. In Raspberry pi 4 there are multiple SPI pins that can be used for SPI communication. The SPI pin of Raspberry Pi 4 is given below:

1. SPI3 CEO N – GPIO0 – Pin27
2. SPI3 MISO – GPIO1 – Pin28
3. SPI3 MOSI – GPIO2 – Pin3
4. SPI3 SCLK – GPIO3 – Pin5
5. SPI4 CEO N – GPIO4 – Pin7
6. SPI4 MISO – GPIO5 – Pin29
7. SPI4 MOSI – GPIO6 – Pin31
8. SPI4 SCLK – GPIO7 – Pin26
9. SPI0 CE1 N – GPIO8 – Pin24
10. SPI0 CE0 N – GPIO9 – Pin21
11. SPI0 MISO – GPIO10 – Pin19
12. SPI0 MOSI – GPIO11 – Pin23
13. SPI5 CEO N/ SPI0 SCLK – GPIO12 – Pin32
14. SPI5 MISO – GPIO13 – Pin33
15. SPI5 MOSI – GPIO14 – Pin8
16. SPI5 SCLK – GPIO15 – Pin10
17. CTS0 – GPIO16 – Pin36
18. RTS0 – GPIO17 – Pin11
19. SPI6 CEO N – GPIO18 – Pin12
20. SPI6 MISO – GPIO19 – Pin35
21. SPI6 MOSI – GPIO20 – Pin38
22. SPI6 SCLK – GPIO21 – Pin40

## I2C Communication Pins:

Raspberry Pi 4 also supports the I2C protocol. It is a type of serial communication used by some sensors and motors to communi[cate. In Pi GPIO pins also gives I2C support. All these pins ar](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=190130107040-ce-a-iot-help)e given below:

1. SDA0/SDA6 – GPIO0 – Pin27
2. SCL0/SCL6 – GPIO1 – Pin28
3. SDA1/SDA3 – GPIO2 – Pin3
4. SCL1/SCL3 – GPIO3 – Pin5
5. SDA3 – GPIO4 – Pin7
6. SCL3 – GPIO5 – Pin29
7. SDA4 – GPIO6 – Pin31
8. SCL4 – GPIO7 – Pin26
9. SDA4 – GPIO8 – Pin24
10. SCL4 – GPIO9 – Pin21
11. SDA5 – GPIO10 – Pin19
12. SCL5 – GPIO11 – Pin23
13. SDA5 – GPIO12 – Pin32
14. SCL5 – GPIO13 – Pin33
15. SDA6 – GPIO22 – Pin15
16. SCL6 – GPIO23 – Pin16

## RPi PWM GPIO Pins:

To generate the desired pulse output signal Raspberry Pi 4 has some PWM pins. Those pins can be used directly with any low voltage external device to get that signal. To generate signal first the pins should get the instructions first. All PWM pins are given below:

1. PWM0 – GPIO12 – Pin32
2. PWM1 – GPIO13 – Pin33
3. PWM0 – GPIO18 – Pin12
4. PWM1 – GPIO19 – Pin35

## Raspberry Pi SDIO Pins:

In Raspberry Pi 4 there is a slot for SD card but GPIO Pins also support the SD card compatibility. SDIO pins on the device can be used for SD card in case of requirement:

1. SD0CLK/SD1 CLK – GPIO22 – Pin15
2. SD0 CMD/SD1 CMD – GPIO23 – Pin16
3. SD0 DATA0/SD1 DAT0 – GPIO24 – Pin18
4. SD0 DAT1/SD1 DAT1 – GPIO25 – Pin22
5. SD1 DAT2/SD1 DAT2 – GPIO26 – Pin37
6. SD0 DAT3/SD1 DAT3 – GPIO27 – Pin13

## Experiment-6

## Introduction to Linux commands and python programming

## Print working directory (pwd)

The pwd command prints your working directory. In other words, it outputs the path of the directory you are currently working in.

## 2. Make directory (mkdir)

Making directories is easy with the mkdir command. The following command creates a directory

## List (ls)

The ls command has many options, including -l to view a long listing of files, displaying the file owner and permissions.

## Change directory (cd)

The cd command's function takes you from your home directory into the Documents directory.

## Copy a file (cp)

Copy files with the cp command.

**Introduction to python programming**

Layers of programming

|  |
| --- |
| High level language(english keywords) |
| Assembly level language(symbols,mnemonics) |
| Machine level(difficult to code) |

**Python command prompt**

* **Print(“Hello world”)**

**Hello world**

**. Age = 10**

**. Name = ‘Raman’**

**. type(age)**

**Class ‘int’**

**. type(name)**

**class ‘str’**

* **print(“Name = % s, Age = % d” % (name,age))**

**Name = Raman, Age = 10**

* **X = [1,2,3,4,3,26,2]**
* **Type(x)**

**Class ‘list’**

* **min(x)**

**1**

* **Max(x)**

**26**

* **Sum(x)**

**41**

**Python Data types**

* **Data types specify 2 things:**
* **what type of data it contains**
* **What operations it performs**

**Eg; a can never be integer, 1 can be integer, float,**

**“1” can be string.**

**1 + 2 = 3(+ is an operation)**

**“ab” + “cd” abcd(concatenation)**

**“1” + “2” = “12”**

**“1” + 2(invalid)**

**python data types: simple, complex**

**simple: Single valued eg: int, float, string**

**complex: Multiple valued eg: list,tuple, dictionary**

**Simple : string(sequence of characters’)**

**s = ‘………………………’**

**s = “………………………”**

**s = “’……………………..”’**

**Eg : s = “ABC”**

**len(s)**

**3**

**t = “pqr”**

**s+t**

**ABCPQR**

* **s1 = ‘This is a string inside single quotes’.**
* **s2 = “This is a string inside double quotes”.**
* **Print(s1)**

**‘This is a string inside single quotes’.**

**. Print(s2)**

**“This is a string inside double quotes”.**

**. Print(s1+s2)**

**‘This is a string inside single quotes’. “This is a string inside double quotes”.**

**List**

**List is an array, it is represented inside a square bracket.[]**

**[1,2,4,..]- set of integers**

**[‘A’,’x’,’z’,….]- set of characters**

**[“ABC”, “XYZ”, ”PQR”]- set of strings**

**Tuple**

**Tuple is a collection of different types, It is represented inside a normal bracket**

**Eg: (1,2,3,…..), (1,’A’,”ABC”)**

**GO TO COMMAND PROMPT**

**Create a list and tuple**

* **X = [1,2,3,1,2]**
* **X[0]**

**1**

* **X[2]**

**3**

**Create a tuple**

* **X = (“ABC”,128,’A’)**
* **X[0]**

**‘ABC’**

* **X[1]**

**128**

**Difference between list & tuple**

* **X = [1,2,3,4]**
* **X[0]**

**1**

* **X[0] = 10**
* **X**

**[10,2,3,4] will be displayed**

* **X = (1,2,3,4)**
* **X[0]**

**1**

* **X[0] = 10**

**error**

**Tuples are immutable, list is mutable**

**Dictionary**

**Dictionary consists of key-value pairs, Dictionary(k : V) pairs, keys are Immutable , value is Mutable**

**(list cannot be used as key), They are represented in {}, Dictionary syntax {K:V; K:V ; K:V}**

* **Eg: x = {“INDIA”: “NEWDELHI”, “AUSTRALIA:CANBERRA”}**
* **X[0] ---------error**
* **X[“INDIA”]**

**NEWDELHI**

**If we want to have list of keys or values , use a function called keys(), values()**

* **Eg2: salary ={“senior”:100, “mid”:50 , “junior” : 20}**
* **Type(salary)**

**Class’dict’**

* **Salary[‘senior’]**

**100**

* **Salary.keys()**

**Dict\_key([‘senior’, ‘mid’, ‘junior’])**

* **Salary.values()**

**Dict\_values([100,50,20])**

**Experiment-7**

### Getting Started with the Interpreter

From a terminal, enter the following commend to start the Python interpreter:

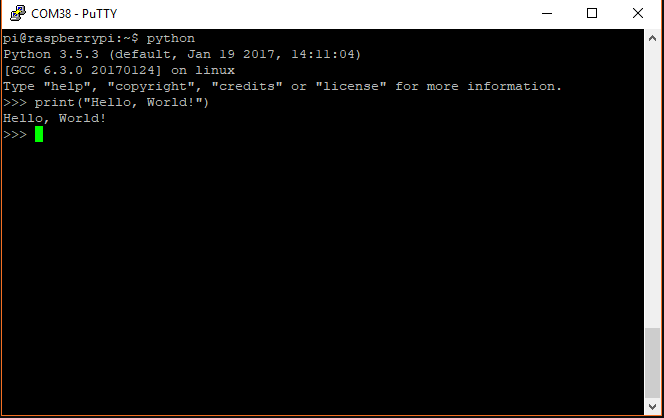
python

You should be presented with a different command prompt, consisting of 3 greater-than signs >>>.

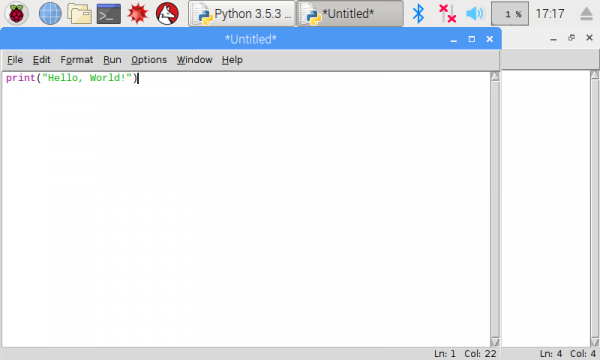
Type the following command:

COPY CODEprint("Hello, World!")

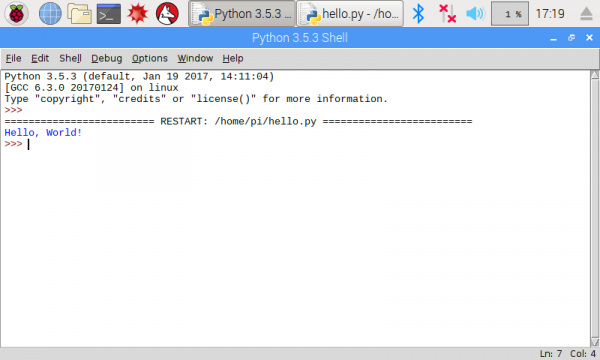
Once you press enter, you should see the phrase Hello, World! repeated back to you.



Open IDLE by selecting the Raspberry Pi logo in the top-left, and click Programming > Python 3 (IDLE). You should be presented with the Python interactive interpreter.



Click File > Save As... to save your code to a Python file (don't forget the .py suffix!). Click Run > Run Module to run your program.

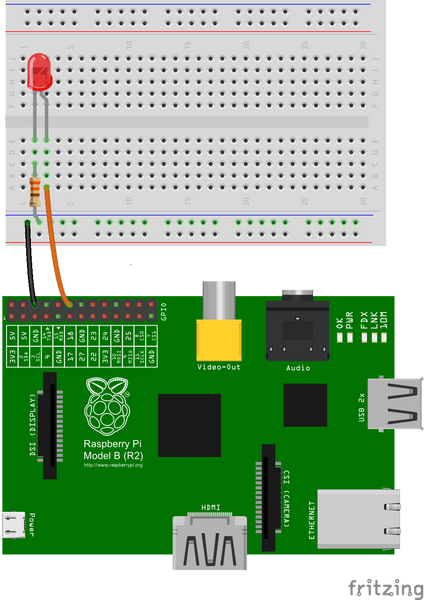


Aim: Interfacing LED /Turning on LED to Raspberrypi

Building the Circuit

The circuit consists of a power supply (the Raspberry Pi), an LED that lights when the power is applied, and a resistor to limit the current that can flow through the circuit.

You will be using one of the ‘ground’ (GND) pins to act like the ‘negative’ or 0 volt ends of a battery. The ‘positive’ end of the battery will be provided by a GPIO pin. Here we will be using pin 18. When they are ‘taken high’, which means it outputs 3.3 volts, the LED will light. Now take a look at the circuit diagram below.



The Code

You are now ready to write some code to switch the LED on. Turn on your Raspberry Pi and open the terminal window.

Create a new text file “LED.py” by typing the following:

nano LED.py

Type in the following code:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(18,GPIO.OUT)

print "LED on"

GPIO.output(18,GPIO.HIGH)

time.sleep(1)

print "LED off"

GPIO.output(18,GPIO.LOW)

Once you have typed all the code and checked it, save and exit the text editor with “Ctrl + x” then “y” then “enter”.

Running the Code

To run this code type:

sudo python LED.py

You will see the LED turn on for a second and then turn off.

If your code does not run and an error is reported, edit the code again using nano LED.py.

**Explanation**

import RPi.GPIO as GPIO

The first line tells the Python interpreter (the thing that runs the Python code) that it will be using a ‘library’ that will tell it how to work with the Raspberry Pi’s GPIO pins. A ‘library’ gives a programming language extra commands that can be used to do something different that it previously did not know how to do. This is like adding a new channel to your TV so you can watch something different.

import time

Imports the Time library so that we can pause the script later on.

GPIO.setmode(GPIO.BCM)

Each pin on the Raspberry Pi has several different names, so you need to tell the program which naming convention is to be used.

print "LED on"

This line prints some information to the terminal.

GPIO.output(18,GPIO.HIGH)

This turns the GPIO pin ‘on’. What this actually means is that the pin is made to provide power of 3.3volts. This is enough to turn the LED in our circuit on.

time.sleep(1)

Pauses the Python program for 1 second

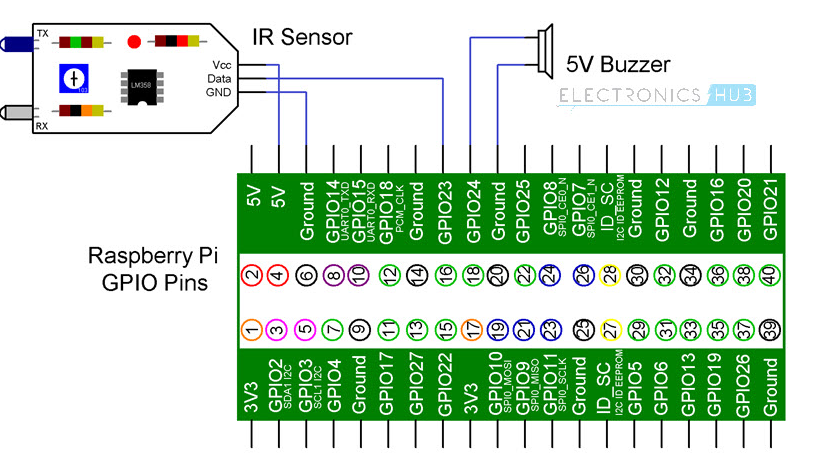
print "LED off"

This line prints some information to the terminal.

GPIO.output(18,GPIO.LOW)

This turns the GPIO pin ‘off’, meaning that the pin is no longer supplying any power.

**Aim: To interface IR sensor to raspberrypi**



circuit design

The IR Sensor Module has only three Pins: VCC, GND and Data. Connect the VCC and GND pins of the IR Sensor to +5V and GND pins of the Raspberry Pi.

Then connect the Data pin of the IR Sensor to GPIO23 i.e. Physical Pin 16 of the Raspberry Pi.

Connect one terminal of the buzzer to GND of Raspberry Pi and the other terminal (usually marked +) to GPIO24 i.e. Physical Pin 18 of Raspberry Pi.

Code

import RPi.GPIO as GPIO

import time

sensor = 16

buzzer = 18

GPIO.setmode(GPIO.BOARD)

GPIO.setup(sensor,GPIO.IN)

GPIO.setup(buzzer,GPIO.OUT)

GPIO.output(buzzer,False)

print "IR Sensor Ready....."

print " "

try:

while True:

if GPIO.input(sensor):

GPIO.output(buzzer,True)

print "Object Detected"

while GPIO.input(sensor):

time.sleep(0.2)

else:

GPIO.output(buzzer,False)

except KeyboardInterrupt:

GPIO.cleanup()