



Darshan
UNIVERSITY

Darshan Institute of Engineering & Technology

Certificate

This is to certify that

Mr./Miss _____

Enrolment No. _____ B.Tech. CSE Semester 6th has satisfactorily completed the course in the Subject– Internet of Things (2101CS631/2301CS593) in this Institute.

Submission Date: __ / __ / __

Staff in Charge

Program Coordinator



योग: कर्मसु कोशलम्

Darshan UNIVERSITY

Department of Computer Science & Engineering

Academic Year 2024-25

B. Tech. Semester – VI

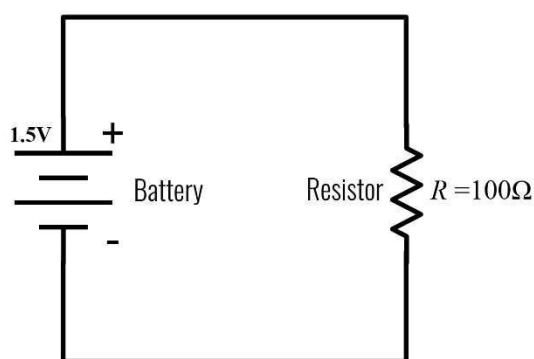
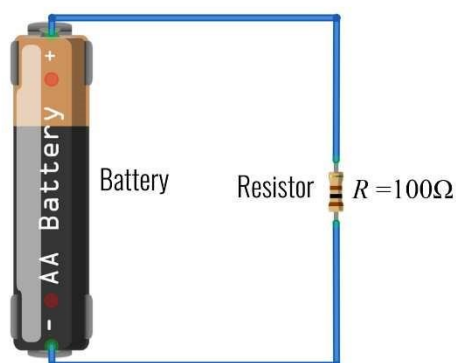
2101CS631/2301CS593 – Internet of Things

Lab Manual

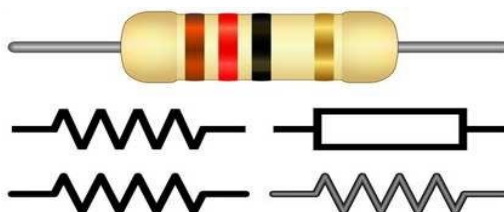
Sr.No	Title	Date	Sign	Remark

1) Basics of Electronics

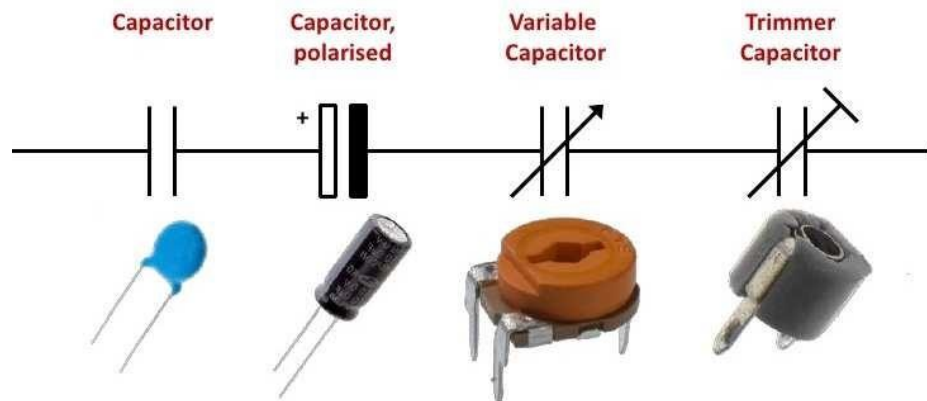
- Electricity is all around us--powering technology like our cell phones, computers, lights, soldering irons, and air conditioners. Electricity is briefly defined as the flow of electric charge or the movement of electrons. It is vital to start by understanding the basics of voltage, current, and resistance. These are the three basic building blocks required to manipulate and utilize electricity. The three basic principles can be explained as below.
- Voltage (V) is the difference in charge between two points.
- Current (I) is the rate at which charge is flowing.
- Resistance (R) is a material's tendency to resist the flow of charge (current). A circuit is a closed loop that allows charge to move from one place to another. Components in the circuit allow us to control this charge and use it to do work.
- Battery/Cell: A battery/cell is a device that converts chemical energy contained within its active materials directly into electric energy by means of an electrochemical oxidation-reduction (redox) reaction. This type of reaction involves the transfer of electrons from one material to another via an electric circuit.



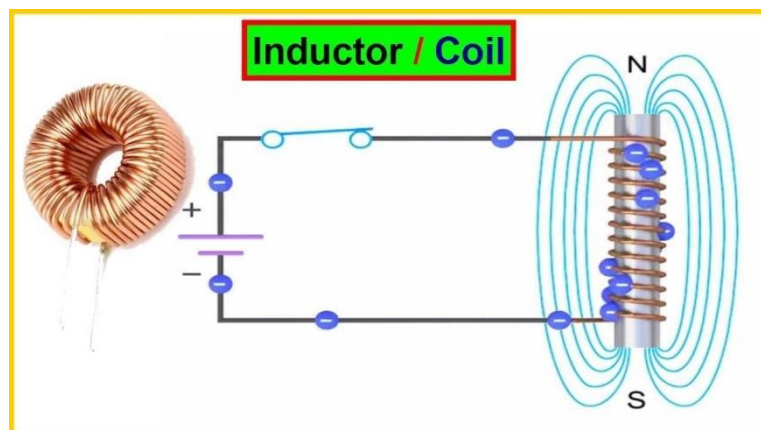
- Resistor: A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element.



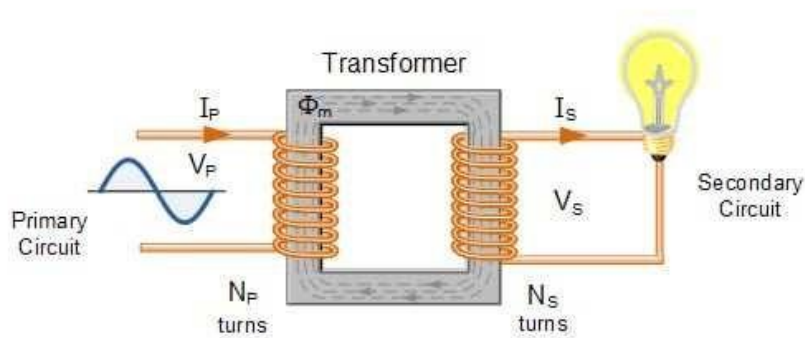
- Capacitor: A capacitor is a device that stores electrical energy in an electric field. It is a passive electronic component with two terminals



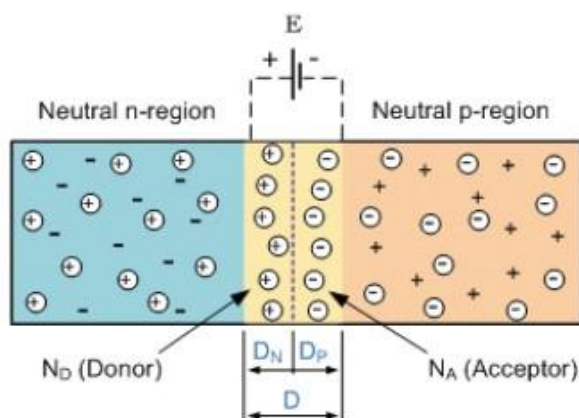
- Inductor: An inductor, also called a coil, choke, or reactor, is a passive two-terminal electrical component that stores energy in a magnetic field when electric current flows through it. An inductor typically consists of an insulated wire wound into a coil.



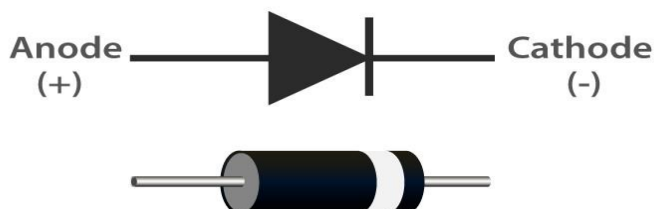
- Transformer: A transformer is a passive component that transfers electrical energy from one electrical circuit to another circuit, or multiple circuits.



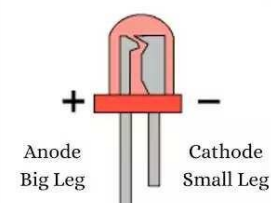
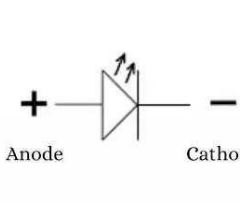
- Semiconductor: A semiconductor material has an electrical conductivity value falling between that of a conductor, such as metallic copper, and an insulator, such as glass. Its resistivity falls as its temperature rises; metals behave in the opposite way. Its conducting properties may be altered in useful ways by introducing impurities ("doping") into the crystal structure.



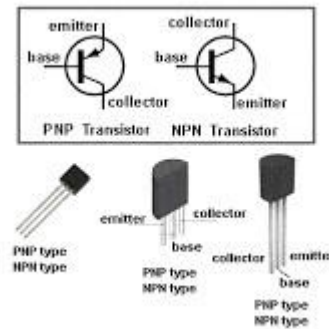
- Diodes: A diode is a two-terminal electronic component that conducts current primarily in one direction; it has low (ideally zero) resistance in one direction, and high resistance in the other.



- LED: A light-emitting diode is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

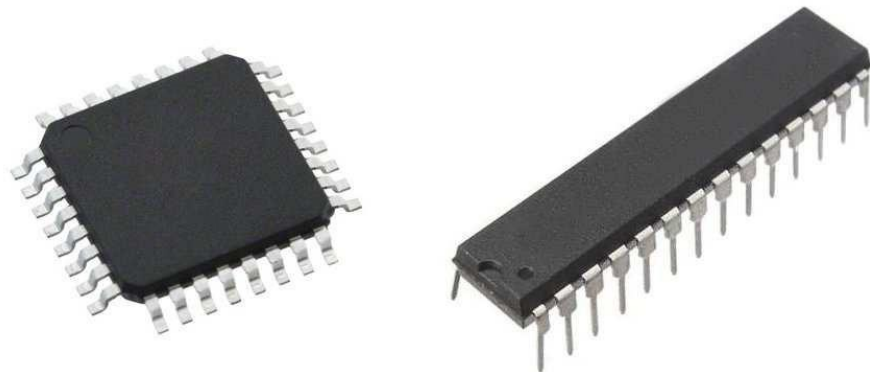
Light Emitting Diode (LED)	Symbol of LED
 <p>Anode Big Leg</p> <p>Cathode Small Leg</p>	 <p>Anode</p> <p>Cathode</p>

- Transistor: A transistor is a semiconductor device used to amplify or switch electrical signals and power. The transistor is one of the basic building blocks of modern electronics. It is composed of semiconductor material, usually with at least three terminals for connection to an electronic

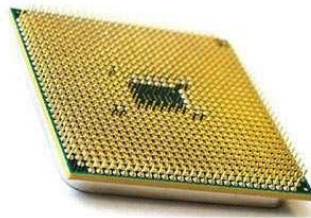


circuit.

- Integrated Circuit: An integrated circuit or monolithic integrated circuit (also referred to as an IC, a chip, or a microchip) is a set of electronic circuits on one small flat piece of semiconductor material, usually silicon. Large numbers of tiny MOSFETs integrate into a small chip. This results in circuits that are orders of magnitude smaller, faster, and less expensive than those constructed of discrete electronic components.

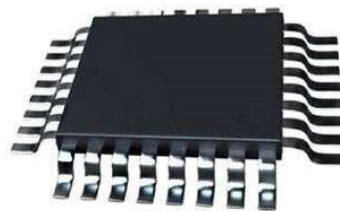


- Microprocessor: A microprocessor is a computer processor where the data processing logic and control is included on a single integrated circuit, or a small number of integrated circuits. The microprocessor contains the arithmetic, logic, and control circuitry required to perform the functions of a computer's central processing unit.



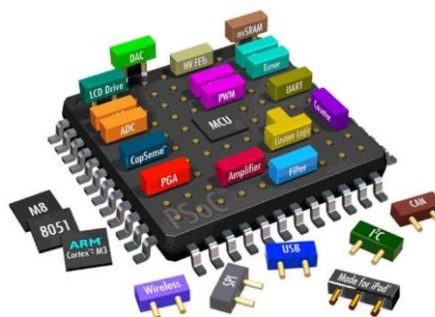
Microprocessor

- Microcontroller: A microcontroller is a small computer on a single metal-oxide-semiconductor integrated circuit chip. A microcontroller contains one or more CPUs along with memory and programmable input/output peripherals.

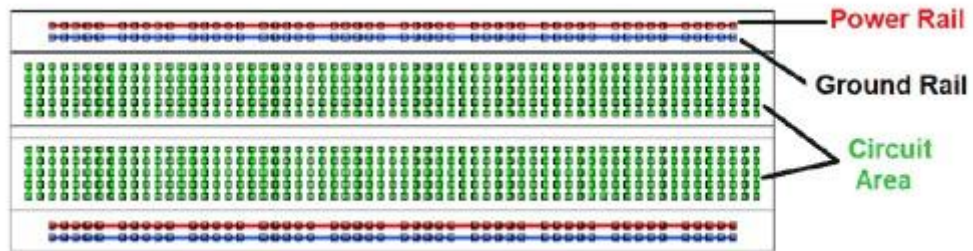


Microcontroller

- SoC: A system on a chip (SoC) is an integrated circuit that integrates all or most components of a computer or other electronic system.

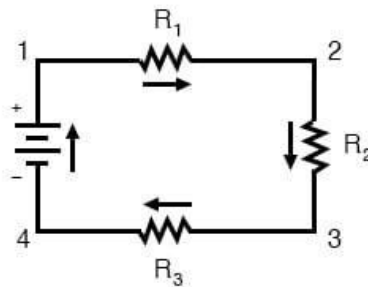


- Breadboard: A breadboard, or protoboard, is a construction base for prototyping of electronics.



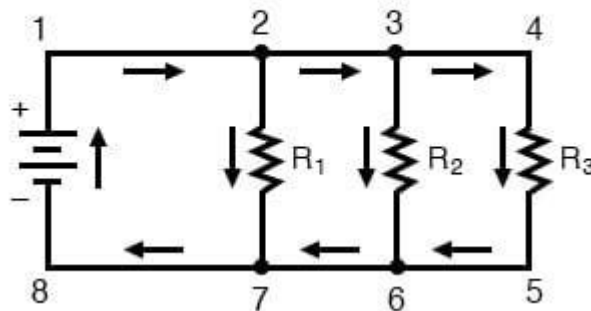
- Series and Parallel concepts
 - Series circuit: The defining characteristic of a series circuit is that there is only one path for current to flow.

Series



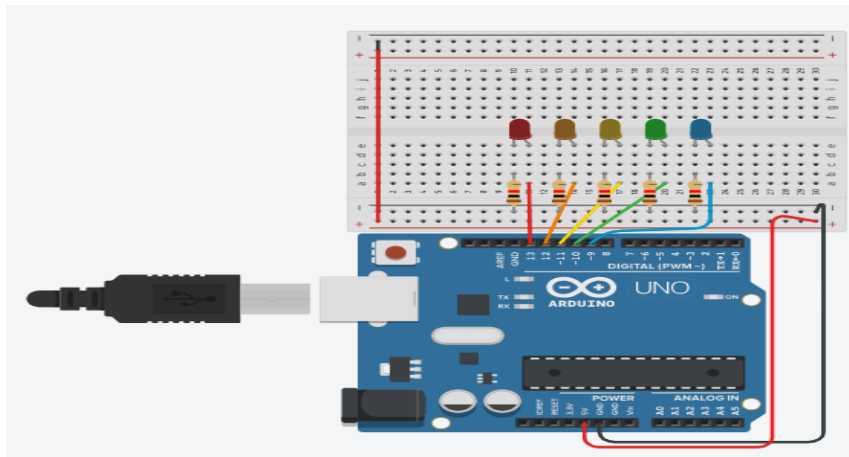
- Parallel circuit: The defining characteristic of a parallel circuit is that all components are connected between the same set of electrically common points.

Parallel



- Tinker Cad
 - Tinkercad offers an online circuit simulation platform where you can connect microcontrollers like Arduino, sensors, motors, LEDs, etc.
 - It allows real-time simulation without using any physical components.
 - You can also write Arduino code (C++) to control the circuits.

- For better understanding of components, wiring and creating circuits Tinkercad is popular for working in an easy simulation environment.
- It helps to learn and prototype various projects using:
 - 3D Design (CAD - Computer-Aided Design)
 - Electronics Circuits (with Arduino simulation)
 - Codeblocks (for block-based programming)
- **The following circuit design created using Tinkercad is to blink LED lights**



2) To study the functionality of the Arduino Uno board and Arduino IDE

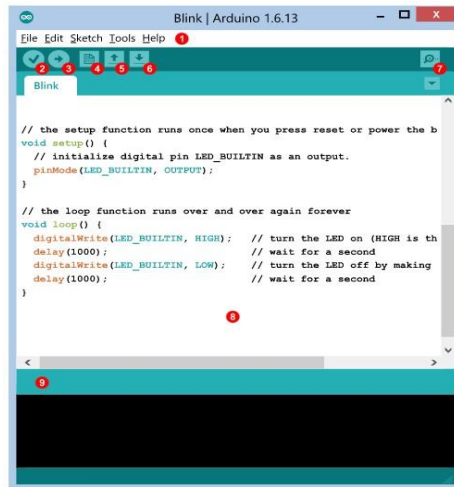
A. 1 Installation of the Arduino IDE

A. 2 Power up Arduino Uno Board

A. 3 Uploading a program to the Arduino UNO

- The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.
- **Writing Sketches**
 - Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension. .ino.
 - All the commands are found within the five menus: File, Edit, Sketch, Tools, and Help.
 - The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.
- **Uploading**
 - Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus.
- **Libraries**
 - Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch.
- **Serial Monitor**
 - This displays serial sent from the Arduino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate the dropdown menu that matches the rate passed to Serial.begin in your sketch.

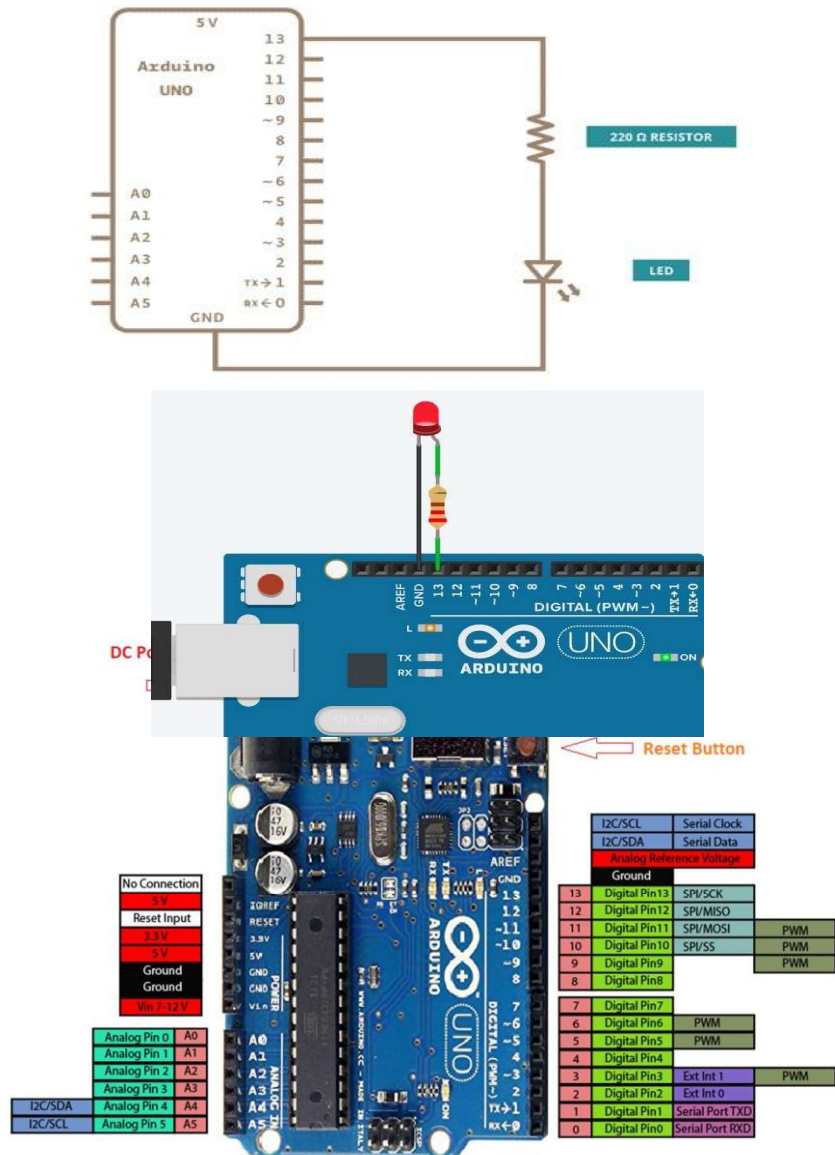
▪ Arduino Uno Board



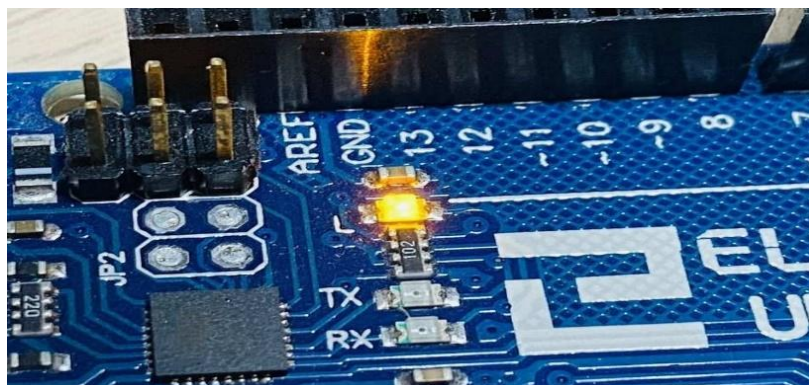
- 1 Menu: Selections of software features.
- 2 Verify: Compiles and verifies your sketch.
- 3 Upload: Send your sketch to STEMtera™ Breadboard.
- 4 New: Opens a new sketch window.
- 5 Open: Open an existing sketch.
- 6 Save: Save current active sketch.
- 7 Monitor: Opens a window to send and receive information.
- 8 Editor: Code editor area. Type your sketch in this area.
- 9 Message: IDE reports success or failure messages here.

▪ Arduino Uno Digital Output

- digitalWrite() : Write a HIGH or a LOW value to a digital pin.
- If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.
- Syntax : digitalWrite(pin, value);
- Parameters pin: the Arduino pin number. value: HIGH or LOW.



- Inbuilt LED control



- LED Blinking

```
// the setup function runs once when you press reset
or power the board
void setup()
{
    // initialize digital pin LED_BUILTIN as an
    output.
    pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs repeatedly
void loop()
{
    digitalWrite(LED_BUILTIN, HIGH);
    // turn the LED on (HIGH is the voltage level)
    delay(1000);                      // wait for a second

    digitalWrite(LED_BUILTIN, LOW);
    // turn the LED off by making the voltage LOW
    delay(1000);                      // wait for a
    second
}
```

- **Two LED Toggling**

```
int led1= 10;
int led2= 11;
void setup()
{
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
}
void loop()
{
    digitalWrite(led1, HIGH);
    digitalWrite(led2, LOW);
    delay(1000);
    digitalWrite(led1, LOW);
    digitalWrite(led2, HIGH);
    delay(1000);
}
```

3) To study the functionality of Node MCU with installation of Arduino packages

A. 1 To study the Pin-diagram of the NodeMCU

A. 2 Arduino IDE package Installation for the ESP8266 support

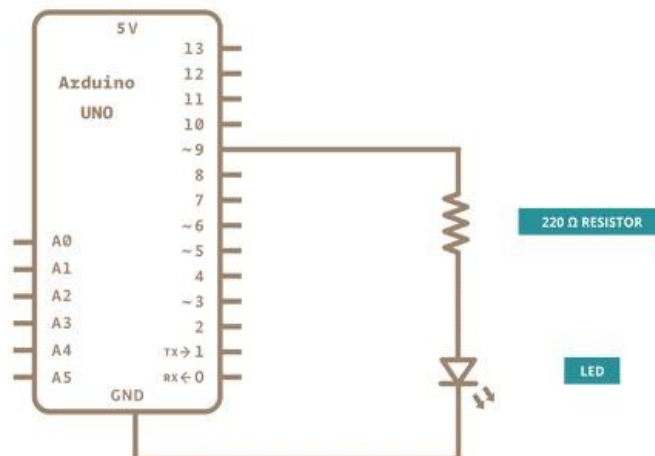
A. 3 Uploading a program to Node MCU

- **analogWrite()**

- Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightness or drive a motor at various speed. After a call to `analogWrite()`, the pin will generate a steady rectangular wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin.
- Syntax: `analogWrite(pin, value)`
- Parameters pin: the Arduino pin to write to. Allowed data types: int.
- value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int.

- **LED Fading**

- This example demonstrates the use of the `analogWrite()` function in fading an LED off and on. `AnalogWrite` uses pulse width modulation (PWM), turning a digital pin on and off very quickly with different ratio between on and off, to create a fading effect.



/*

Fade - This example shows how to fade an LED on pin 9 using the `analogWrite()` function.

The `analogWrite()` function uses PWM, so if you want to change the pin you're using, be sure to use another PWM capable pin. On most Arduino, the PWM pins are identified with a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

*/

```
int led = 9;           // the PWM pin the LED is
                        attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the
                        LED by
// the setup routine runs once when you press reset:
void setup()
{
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}
// the loop routine runs repeatedly
void loop()
{
  // set the brightness of pin 9:  analogWrite(led, brightness);
  // change the brightness for next time through the loop:
  //brightness = brightness + fadeAmount;
  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255)
  {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming
  effect
  delay(30);
}
```

4) To demonstrate digital input of Arduino board with push button

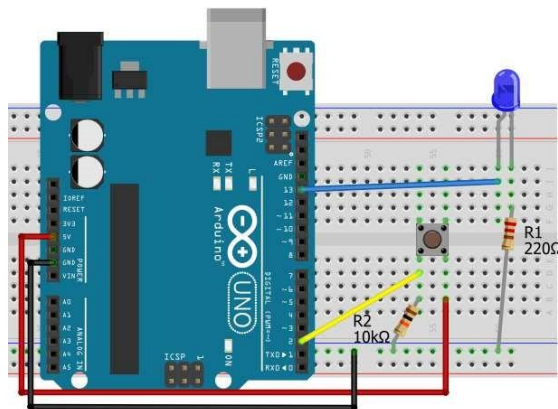
- A. 1 To read status of Push button using NodeMCU
- B. 2 Toggle the state of LED with Push Button
- B. 3 To read multiple digital input simultaneously

- **Arduino Uno Digital Input**

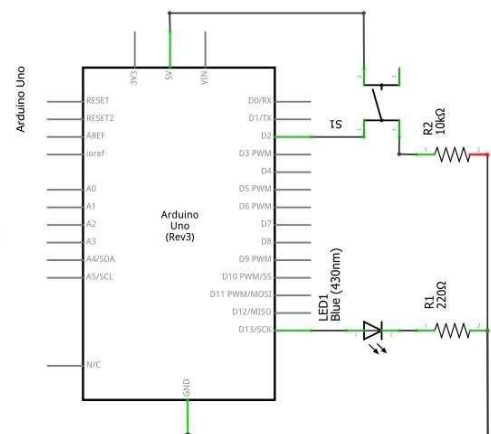
- digitalWrite()
- Reads the value from a specified digital pin, either HIGH or LOW.
- Syntax: digitalWrite(pin)
- Parameters pin: the Arduino pin number you want to read
- Returns HIGH or LOW

- **LED On/Off using a Switch**

Breadboard View



Schematic View



```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 2;   // pushbutton connected to digital
pin 2 int val = 0; // variable to store the read
value
```

```
void setup()
```

```
{
```

```
    pinMode(ledPin, OUTPUT); // sets the digital pin 13 as output
```

```
    pinMode(inPin, INPUT); // sets the digital pin 7 as input
```

```
}
```

```
void loop()
```

```
{
```

```
    val = digitalWrite(inPin); // read the input pin
```

```
    digitalWrite(ledPin, val); // sets the LED to the button's
```

```
value
```

```
}
```


5) To demonstrate digital output of Arduino board with LED control

- A. 1 To control built-in LED
- A. 2 To blink LED with 1 sec delay
- B. 3 To perform Traffic Light signal with Red, Yellow and Green LEDs

- **To blink LED with 1 sec delay**

```
// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000);            // wait for a second
    digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
    delay(1000);            // wait for a second
}
```

- **Traffic Light signal with Red, Yellow and Green LEDs**

```
void setup() {
    // put your setup code here, to run once
    pinMode(D5, OUTPUT);
    pinMode(D6, OUTPUT);
    pinMode(D7, OUTPUT);
}

void loop() {
    // put your main code here, to run the traffic lights repeatedly
    digitalWrite(D5, HIGH);
    delay(1000);
    digitalWrite(D5, LOW);

    digitalWrite(D6, HIGH);
    delay(1000);
    digitalWrite(D6, LOW);

    digitalWrite(D7, HIGH);
    delay(1000);
    digitalWrite(D7, LOW);
    exit(0);
}
```



योग: कर्मसु कौशलम्

Darshan

UNIVERSITY

}

Lab Manual: Internet of Things (2101CS631/2301CS593)

6) To demonstrate Analog Input of Arduino board with potentiometer

- A. 1 To read the value of Potentiometer
- A. 2 To map the value of Analog input into required range
- B. 3 To perform interfacing of Moisture module with Arduino for Soil moisture measurement
- B. 4 To perform interfacing of MQ-XX module with Arduino for gas detection

- **Arduino Uno Analog Input**

- **analogRead()**
- Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().
- It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.
- Syntax: analogRead(pin)
- Parameters - pin: the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)
- Returns int (0 to 1023)

- **To read and map the value of Potentiometer**

```
void setup()
{
    pinMode(A0, INPUT);
    Serial.begin(9600);
    //input pullup
    pinMode(2, INPUT_PULLUP);
}
void loop()
{
    //Analog Input
    //pullup = normally 1
    int pot = analogRead(A0);
    bool pb = digitalRead(2);
    //buttonState pb = 0 (button press)
    if(pb==0) {
        Serial.println(pot); //print potentiometer value
        delay(1000);
    }
}
```

7) To demonstrate Analog Output of Arduino board with LED fading

A. 1 To study the Pulse Width Modulation and PWM pins

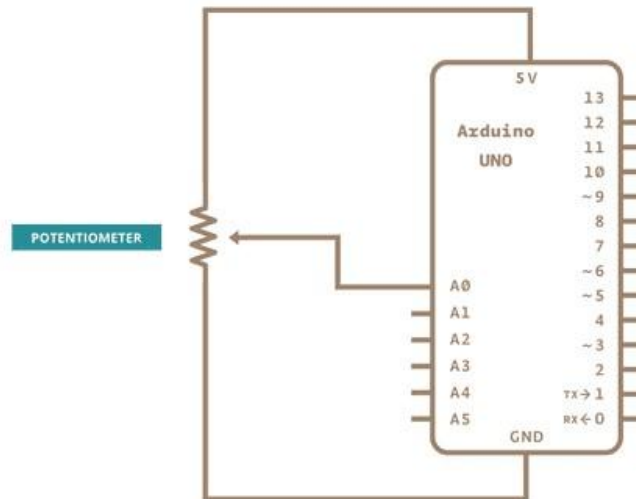
A. 2 To perform Analog Output with LED fading

B. 3 To control the brightness of LED using potentiometer

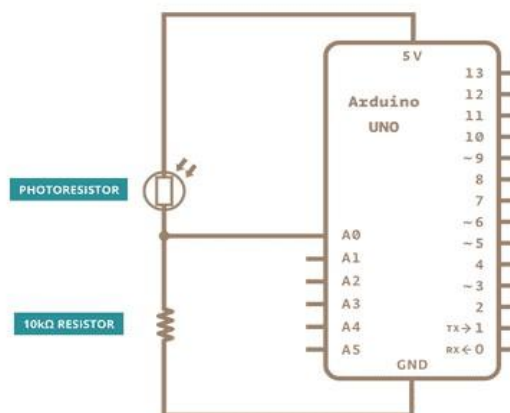
• Brightness Control Using Potentiometer and LDR

- In this example we use a variable resistor (a potentiometer or a photoresistor), we read its value using one analog input of an Arduino board and we change the blink rate of the built-in LED accordingly. The resistor's analog value is read as a voltage because this is how the analog inputs work.

- With Potentiometer



- With LDR



/*

Analog Input Demonstrates analog input by reading an analog sensor on analog pin 0 and Changing light of the connected LED to

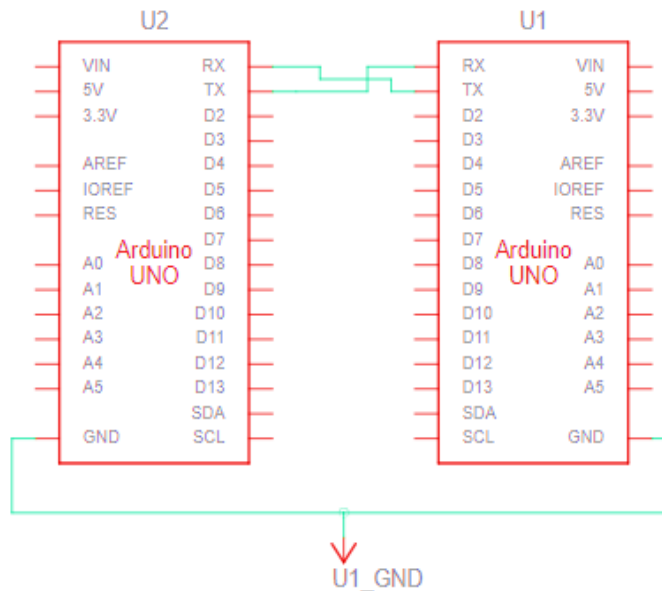
digital pin 9. The amount of changing lights depends on the value obtained by `analogRead()`.

The circuit: - potentiometer - center pin of the potentiometer to the analog input 0 one side pin - (either one) to ground the other side pin to +5V - LED anode (long leg) attached to digital (PWM) output 9 through 220-ohm resistor cathode (short leg) attached to ground

```
*/  
  
int sensorPin = A0;    // select the input pin for the  
potentiometer  
int ledPin = 9;        // select the pin for the LED  
int sensorValue = 0;  
// variable to store the value coming from the sensor  
void setup()  
{  
    //declare the ledPin as an OUTPUT:  
    pinMode(ledPin, OUTPUT);  
}  
void loop()  
{  
    sensorValue = analogRead(sensorPin);  
    analogWrite(ledPin, sensorValue);  
}
```

8) To study and perform Serial communication with Arduino board

- A. 1 To demonstrate serial communication methods for Arduino
- B. 2 To demonstrate sending and receiving of data via serial communication pins



- **Sending and receiving of data via serial communication**

- **Sender Arduino Code**

```
char Mymessage[5] = "Hello"; //String data

void setup() {

    // Begin the Serial at 9600 Baud
    Serial.begin(9600);

}

void loop() {

    Serial.write(Mymessage,5); //Write the serial data

    delay(1000);

}
```

- **Receiver Arduino Code**

```
char Mymessage[10]; //Initialized variable to store received data

void setup() {
```



```
// Begin the Serial at 9600 Baud

Serial.begin(9600);

}

void loop() {

    Serial.readBytes(Mymessage,5); //Read the serial data and store in var

    Serial.println(Mymessage); //Print data on Serial Monitor

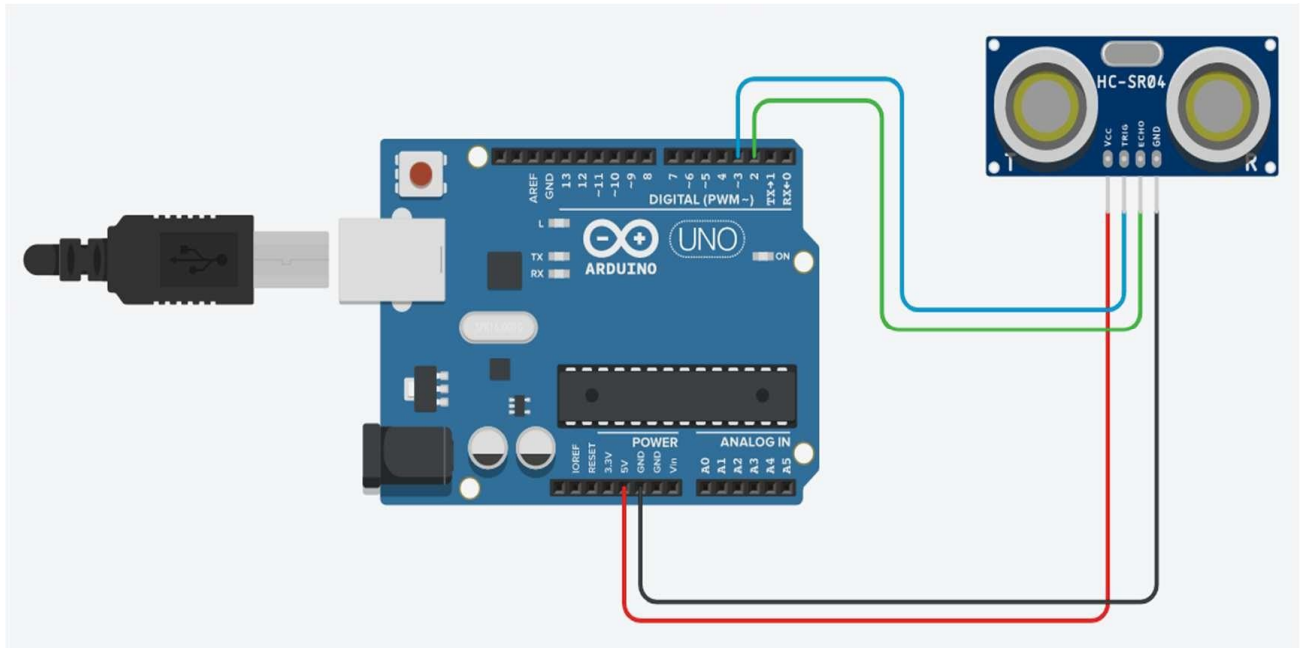
    delay(1000);

}
```

9) Hands on practice for the interfacing of sensor and actuators with Arduino board

- A. 1 To perform interfacing of Ultrasonic sound sensor module with Arduino board for distance measurement
- B. 2 To perform interfacing of LDR module with Arduino board for light intensity measurement
- B. 3 To perform interfacing of DC motor with Arduino board
- B. 4 To perform interfacing of Servo motor with Arduino board
- **Here we are going to interface an Ultrasonic sensor HC-SR 05 with Arduino Uno.**
- **HC - SR 05**
 - It consists of an ultrasonic transmitter and an ultrasonic receiver.
 - First ultrasonic transmitter sends an ultrasonic wave.
 - This signal will collide with the object and reflect the signal.
 - The receiver will receive the reflected signal.
 - The distance calculated by the time taken to receive the reflected.
 - The speed of sound in air at room temperature is 340 Meter/Second or 0.034 centimeter/microsecond.
 - The equation for calculating time is,
$$\text{Time} = \text{Distance} / \text{speed of sound}$$
 - If the object is 10 CM away from the sensor, you will get the time as per the equation is,
$$10 / 0.034 = 294.11 \text{ Microseconds}$$
 - But you will get the value from the Echo pin is 588.22. This is because of the sound wave needs to travel forward and bounce backward. So, we need to divide that value by 2 for get the actual value(time). Here we want to calculate the distance from the time. So, re-arrange the equation we will get,

- Distance=Time x speed of sound



```
#define echoPin 2
#define trigPin 3

long duration;
int distance;
void setup()
{
    Serial.begin(9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
}
void loop()
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration=pulseIn(echoPin, HIGH);
    distance=(duration*0.034/2);
    Serial.print("Distance : ");
    Serial.print(distance);
    Serial.println(" cm ");
    delay(1000);
}
```

- **LDR module with NodeMCU for light intensity measurement**

```
void setup()
{
    pinMode(A0, INPUT);
    Serial.begin(9600);
}
void loop()
{
    int light_intensity = analogRead(A0);
    if (light_intensity > 750)
    {
        Serial.println("Darker");
    }
    else if (light_intensity < 400)
    {
        Serial.println("Too Bright");
    }
    delay(1000);
}
```

- **To interface Servo motor with NodeMCU**

```
#include <Servo.h>
Servo myservo;
void setup() {
    myservo.attach(D4); //PWM
}
void loop() {
    myservo.write(0);
    delay(1000);
    myservo.write(90);
    delay(1000);
    myservo.write(180);
    delay(1000);
}
```

10) To demonstrate Smart Home application with IoT Cloud

- A. 1 To demonstrate the account creation on IoT cloud
- A. 2 To demonstrate the integration of various IoT node with IoT cloud
- B. 3 To design the IoT Dashboard for Smart Home application
- C. 4 To perform the data integration between IoT dashboard, Mobile application and IoT nodes

- **Open an account in blynk cloud**
- <https://blynk.cloud/dashboard/login>
- **Create and get ID, name and Token which are unique for every user.**

- BLYNK_TEMPLATE_ID,
- BLYNK_DEVICE_NAME,
- BLYNK_AUTH_TOKEN
- Upload the following code with necessary changes.
- Control the device using web or mobile app.

// Template ID, Device Name and Auth Token are provided by the Blynk.Cloud // See the Device Info tab, or Template settings

```
#define BLYNK_TEMPLATE_ID "TMPLhRbroS9e1"  
#define BLYNK_DEVICE_NAME "Quickstart Template"  
#define BLYNK_AUTH_TOKEN "1eZcl4gPP081vtOZk3wCQq4DKpPYIYuBY"
```

```
// Comment this out to disable prints and save space  
#define BLYNK_PRINT Serial  
#include <ESP8266WiFi.h>  
#include <BlynkSimpleEsp8266.h>
```

```
char auth[] = BLYNK_AUTH_TOKEN;
```

```
// Your WiFi credentials.  
// Set password to "" for open networks.  
char ssid[] = "SSID";  
char pass[] = "Password";
```

```
BlynkTimer timer;
```

```
// This function is called every time the Virtual Pin 0 state
changes BLYNK_WRITE(V0)
{
    // Set incoming value from pin V0 to a variable
    int value = param.asInt();
    pinMode(2,OUTPUT);
    digitalWrite(2,value);

    // Update state
    Blynk.virtualWrite(V1, value);
}
// This function is called every time the device is connected to
the Blynk.Cloud

BLYNK_CONNECTED()
{
    // Change Web Link Button message to "Congratulations!"
    Blynk.setProperty(V3, "offImageUrl",
        "https://staticimage.nyc3.cdn.digitaloceanspaces.com/gene
        ral/fte/congratulations.png");
    Blynk.setProperty(V3, "onImageUrl", "https://static-
        image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulation
        ns_pressed.png ");

    Blynk.setProperty(V3, "url",
        "https://docs.blynk.io/en/gettingstarted/what-do-i-need-
        to-blynk/how-quickstart-device-was-made"); }

// This function sends Arduino's uptime every second to Virtual Pin 2.
void myTimerEvent()
{
    // You can send any value at any time.
    // Please don't send more that 10 values per second.
    Blynk.virtualWrite(V2, millis() / 1000);
}

void setup()
{
    // Debug console
    Serial.begin(115200);
    Blynk.begin(auth, ssid, pass);
    // Setup a function to be called every second
    timer.setInterval(1000L, myTimerEvent);
}

void loop()
{
    Blynk.run();
    timer.run();
}
```

- **Using Blynk to communicate with sensor**

```
#define BLYNK_TEMPLATE_ID          "TMPL3dUrWlesg"
#define BLYNK_TEMPLATE_NAME        "Quickstart Template"
#define BLYNK_AUTH_TOKEN           "rBw5MD-
uXYpy97bCYL_w0PhHgu9YCiIz"

#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define tri D2//from distance sensor
#define echo D1 //from distance sensor

char ssid[] = " SSID "; //wifi id
char pass[] = " password "; //pwd

BlynkTimer timer;

BLYNK_WRITE(V0)
{
    int value = param.asInt();
    digitalWrite(D4,!value);//line 1 added for builtin led
    Blynk.virtualWrite(V1, value);
}

BLYNK_CONNECTED()
{
    Blynk.setProperty(V3, "offImageUrl", "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.p
ng");
    Blynk.setProperty(V3, "onImageUrl", "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_p
ressed.png");
    Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-
started/what-do-i-need-to-blynk/how-quickstart-device-was-made");
}

void myTimerEvent()
{
    ////////////////Blynk.virtualWrite(V2, millis() / 1000); //cut
this function and pest in loop
}

void setup()
{

    pinMode(D4,OUTPUT);//line 2 added for builtin led,, here D4 or 2
is use for builtin led
```

```
pinMode(tri, OUTPUT); //from distance sensor
pinMode(echo, INPUT); //from distance sensor

Serial.begin(9600); //from distance sensor

Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
timer.setInterval(1000L, myTimerEvent);
}

void loop()
{
    Blynk.run();
    timer.run();

    digitalWrite(tri, LOW); //from distance sensor
    delayMicroseconds(5); //from distance sensor
    digitalWrite(tri, HIGH); //from distance sensor
    delayMicroseconds(10); //from distance sensor
    digitalWrite(tri, LOW); //from distance sensor

    long int duration = pulseIn(echo, HIGH); //from distance sensor
    //speed = 343 m/sec ==> 0.0343 cm/us    us==microsecond
    float cm = 0.0343 * (duration/2); //from distance sensor
    Serial.println(cm); //from distance sensor
    delay(1000); //from distance sensor

    //Blynk.virtualWrite(V2, millis() / 1000); //from myTimerEvent()
    this function
    Blynk.virtualWrite(V2, cm);
}
```

11) To perform smart farming application with MQTT

- A. 1 To connect Node MCU with MQTT server
- B. 2 To send and receive data with MQTT server
- C. 3 To develop IoT Smart farming application

- **We will use hivemq as MQTT Server**

- Open <http://www.hivemq.com/demos/websocket-client/>
- Click on connect
- To publish analog input value with a topic (mainTopic) upload the following code
- Subscribe a topic (mainTopic)
- Publish Code

```
#include<ESP8266WiFi.h>
#include<PubSubClient.h>
#define MSG_BUFFER_SIZE (50)

const char *ssid=" ssid ";
const char *pwd=" pwd ";
const char *mqtt_server="broker.mqtt-dashboard.com";

WiFiClient espClient;
PubSubClient client(espClient);
int value=0;
unsigned long lastMsg=0;

char msg[MSG_BUFFER_SIZE];
void setup_wifi()
{
    delay(10);
    Serial.println();
    Serial.print("Connecting to");
    Serial.println(ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid,pwd);
    while(WiFi.status()!=WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFiconnected");
    Serial.println("IPAddress:");
    Serial.println(WiFi.localIP());
```

```
}  
void setup()  
{  
    Serial.begin(115200);  
    setup_wifi();  
    client.setServer(mqtt_server,1883);  
    while(!client.connected()){  
        Serial.print("Attempting MQTT");  
        String clientId="ESPClient1234";  
        if(client.connect(clientId.c_str())){  
            Serial.println("connected");  
            client.publish("mainTopic","Welcome to IoT");  
        }  
    }  
}  
void loop()  
{  
    client.loop();  
    unsigned long now = millis();  
    if(now-lastMsg>2000)  
    {  
        lastMsg=now;  
        ++value;  
        value = analogRead(A0);  
        snprintf(msg,MSG_BUFFER_SIZE,"helloworld #  
        %ld",value);  
        Serial.print("Publish message:");  
        Serial.println(msg);  
        client.publish("mainTopic",msg);  
    }  
}
```

- Publish and subscribe through controller
- To publish (subTopic) upload the following code
- Publish a message (on) to ON LED and
- Publish a message (off) to OFF LED
- Subscribe Code

```
#include <ESP8266WiFi.h>  
#include <PubSubClient.h>  
const char *ssid=" ssid "; const  
char *pwd=" pwd";  
const char *mqtt_server="broker.mqtt-dashboard.com";  
WiFiClient espClient;  
PubSubClient client(espClient);  
  
void setup_wifi()  
{  
    delay(10);  
    Serial.println();  
    Serial.print("Connecting to ");
```



```
Serial.println(ssid);
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, pwd);

while(WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
void callback(char *topic, byte *payload, unsigned int length)
{
    if(!strncmp((char *)payload, "on", length))
    {
        digitalWrite(2, HIGH);
        // powerState = true;
        //client.publish(PUBTOPIC, "on", true);
    }
    else if (!strncmp((char *)payload, "off", length))
    {
        digitalWrite(2, LOW); // powerState = false;
        //client.publish(PUBTOPIC, "off", true);
    }
}
void setup()
{
    pinMode(D4, OUTPUT);
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);

    while (!client.connected())
    {
        Serial.print("Attempting MQTT");
        String clientId="ESPClient1234";
        if (client.connect(
            clientId.c_str()))
        {
            Serial.println(" connected");
            client.subscribe("subTopic");
        }
    }
}
void loop()
{
    client.loop();
}
```

- **MQTT with Blynk publish-subscribe code using NodeMCU**

- Circuit Connections:

Component	NodeMCU (ESP8266) Pins
Ultrasonic Sensor	VCC → 3.3V / GND → GND
Trig Pin	D5 (GPIO14)
Echo Pin	D6 (GPIO12)
LDR Sensor	One leg to A0 (Analog)
Other leg	GND (with Resistor 10K)

- Ultrasonic Sensor (HC-SR04) reads the distance and sends it to MQTT Broker.
- LDR Sensor reads light intensity and sends it to MQTT Broker.
- The code subscribes to both sensor topics and displays the result in the Blynk App.
- You can monitor the sensor values on the Blynk Dashboard and MQTT Broker.

1. Use Blynk App Dashboard

- Blynk App Setup steps:
 - Open Blynk App → Create a new project → Choose ESP8266 as Device.
 - Use the Authentication Token in the code.
 - Then add:
 - Value Display Widget → Virtual Pin V0 → Name it as Distance.
 - Value Display Widget → Virtual Pin V1 → Name it as Light Intensity.

2. Use MQTT Broker (HiveMQ) to Monitor Values

- You can use HiveMQ Broker to monitor data:
 - Go to: <https://www.hivemq.com/public-mqtt-broker>
 - Topic 1: nodemcu/distance → Shows Distance value.
 - Topic 2: nodemcu/ldr → Shows Light Intensity value.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <BlynkSimpleEsp8266.h>

#define TRIG_PIN D5
#define ECHO_PIN D6
#define LDR_PIN A0

// WiFi and MQTT details
const char* ssid = "YOUR_WIFI_NAME";
const char* password = "YOUR_WIFI_PASSWORD";
const char* mqtt_server = "broker.hivemq.com"; // Public MQTT Broker
```

```
const char* distance_topic = "nodemcu/distance";
const char* ldr_topic = "nodemcu/ldr";

WiFiClient espClient;
PubSubClient client(espClient);

// Blynk Authentication Token
char auth[] = "YOUR_BLYNK_AUTH_TOKEN";

// Variables for sensors
long duration;
int distance;
int ldrValue;

// Function to connect to WiFi
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to WiFi...");

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFi connected");
}

// Callback function for MQTT subscription
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived in topic: ");
    Serial.println(topic);

    // Convert payload to string
    String message = "";
    for (int i = 0; i < length; i++) {
        message += (char)payload[i];
    }

    // Display incoming messages on Serial Monitor
    Serial.println("Message: " + message);
}

// Function to reconnect to MQTT Broker
void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");

        if (client.connect("NodeMCU_Client")) {
```

```
        Serial.println("connected");
        client.subscribe(distance_topic);
        client.subscribe(ldr_topic);
    } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
        delay(5000);
    }
}

void setup() {
    Serial.begin(9600);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);

    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);

    Blynk.begin(auth, ssid, password);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    Blynk.run();

    // Read Distance from Ultrasonic Sensor
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    duration = pulseIn(ECHO_PIN, HIGH);
    distance = duration * 0.034 / 2;

    // Read Light Intensity from LDR
    ldrValue = analogRead(LDR_PIN);

    // Publish both sensor values to MQTT Broker
    String distanceStr = String(distance);
    String ldrStr = String(ldrValue);
    client.publish(distance_topic, distanceStr.c_str());
    client.publish(ldr_topic, ldrStr.c_str());

    // Send values to Blynk App
```



```
Blynk.virtualWrite(V0, distance); // Send Distance to Virtual  
Pin V0  
Blynk.virtualWrite(V1, ldrValue); // Send LDR Value to Virtual  
Pin V1  
  
delay(1000);  
}
```

12) To demonstrate interfacing of Arduino board with wireless Radio Frequency module

- A. 1 To demonstrate Arduino board interfacing with Bluetooth Module
- B. 2 To demonstrate Arduino board interfacing with ZigBee Module
- C. 3 To demonstrate Arduino board interfacing with LoRa Module

- **NodeMCU with Bluetooth Serial Communication**

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(D7, D8); // RX, TX

void setup() {
    Serial.begin(115200);    // Serial Monitor
    BTSerial.begin(9600);    // Bluetooth Communication

    Serial.println("Bluetooth Module
Ready");
}

void loop() {
    // Check if data received from Bluetooth
    if (BTSerial.available()) {
        char received = BTSerial.read();
        Serial.print("Data Received: ");
        Serial.println(received);

        // Send Response back to Bluetooth
        BTSerial.print("Received: ");
        BTSerial.println(received);
    }

    // Check if data entered from Serial Monitor
    if (Serial.available()) {
        char data = Serial.read();
        BTSerial.write(data);
    }
}
```

13) To demonstrate the functionality of General Purpose Input-Output pins of Raspberry Pi

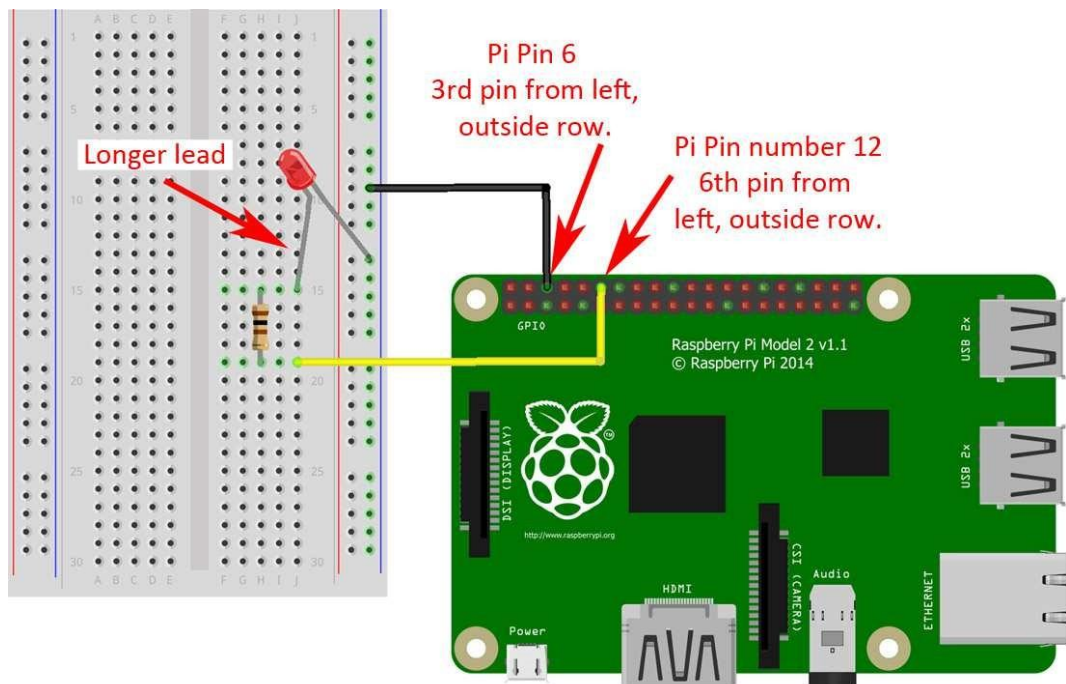
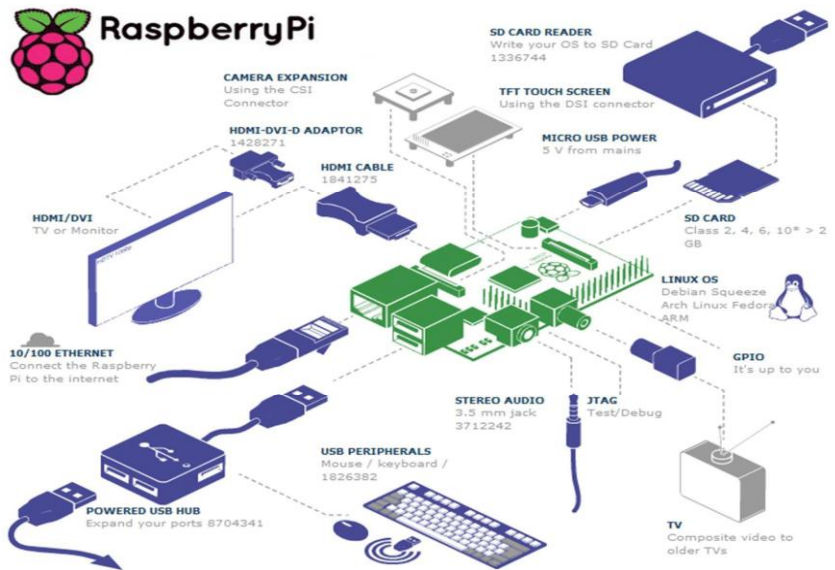
- A. 1 To demonstrate digital Input and Output with Raspberry pi
- B. 2 To demonstrate Analog Input and Output with Raspberry pi

- **Introduction of Raspberry Pi and its Interface**



- Raspberry Pi is a single-board computer, developed by Raspberry Pi Foundation in association with Broadcom and perhaps the most inspiring computer available today.
- Because of its low cost and open design, the model became far more popular than anticipated.
- It is widely used to make gaming devices, fitness gadgets, weather stations, and much more.
- In 2012, the company launched the Raspberry Pi and the current generations of regular Raspberry Pi boards are Zero, 1, 2, 3, and 4.
- Processor
- HDMI Port
- USB Port
- Ethernet Port

- SD Card Slot
- Camera Connector
- Audio Output
- Micro USB Power
- GPIO Pins
- Status LEDs
- Composite Video
- Output



• Controlling GPIO using Python

- Python is a powerful programming language that's easy to use easy to read and write and, with Raspberry Pi
- Python syntax is clean, with an emphasis on readability, and uses standard English keywords.
- To control hardware connected to the Raspberry Pi we will use Python.
- One should be familiar with some of the basics of Python, including literals, variables, operators, control flow, scope, and data structures.
- We'll use the RPi.GPIO module to control all the GPIO of Raspberry Pi

- Import the Rpi.GPIO module

```
import RPi.GPIO as GPIO
```

- **Pin Numbering Declaration**

- After you have included the RPi.GPIO module, the next step is to determine which of the two pin numbering schemes you want to use:
- GPIO.BOARD -- Board numbering scheme. The pin numbers follow the pin numbers on header P1.
- GPIO.BCM -- Broadcom chip-specific pin numbers. These pin numbers follow the lower-level numbering system defined by the Raspberry Pi's Broadcom-chip brain.
- To specify in your code which number-system is being used, use the GPIO.setmode() function.
- GPIO.setmode(GPIO.BCM)
- Both the import and setmode lines of code are required, if you want to use Python.

- **Setting a Pin Mode**

- Like declare a "pin mode" in Arduino before you can use it as either an input or output.
- To set a pin mode, use the setup([pin], [GPIO.IN, GPIO.OUT]) function. So, if you want to set pin 18 as an output, GPIO.setup(18, GPIO.OUT)
- **Outputs: Digital Output**
- To write a pin high or low, use the GPIO.output([pin], [GPIO.LOW, GPIO.HIGH]) function.
- For example, if you want to set pin 18 high, write:
- GPIO.output(18, GPIO.HIGH)
- Writing a pin to GPIO.HIGH will drive it to 3.3V, and GPIO.LOW will set it to 0V.
- Alternative to GPIO.HIGH and GPIO.LOW, you can use either 1, True, 0 or False to set a pin value.

- **PWM ("Analog") Output**

- PWM on the Raspberry Pi is about as limited as can be -- one, single pin is capable of it: 18 (i.e. board pin 12).
- To initialize PWM, use GPIO.PWM([pin], [frequency]) function.
- To make the rest of your script-writing easier you can assign that instance to a variable.
- Then use pwm.start([duty cycle]) function to set an initial value.
- For example...

```
pwm = GPIO.PWM(18, 1000)
```

```
pwm.start(50)
```

- Inputs
- If a pin is configured as an input, you can use the `GPIO.input([pin])` function to read its value.
- The `input()` function will return either a `True` or `False` indicating whether the pin is HIGH or LOW.
- You can use an if statement to test this, will read pin 17 and print whether it's being read as HIGH or LOW.

```
if GPIO.input(17):  
  
    print("Pin 11 is HIGH")  
  
else: print("Pin 11 is LOW")
```

- **Blink.py Blinking LED code in Python**

```
import time  
  
import RPi.GPIO as GPIO  
  
led_pin = 12  
  
# Pin definitions  
GPIO.setmode(GPIO.BCM)  
  
# Use "GPIO" pin numbering  
  
GPIO.setup(led_pin, GPIO.OUT)  
  
# Set LED pin as output  
  
while True:                                     # Blink forever  
  
    GPIO.output(led_pin, GPIO.HIGH)  
  
    # Turn LED on  
  
    time.sleep(1)                                # Delay for 1 second  
  
    GPIO.output(led_pin, GPIO.LOW)  
  
    # Turn LED off  
  
    time.sleep(1)                                # Delay for 1 second
```