DEPARTMENT
OF
COMPUTER SCIENCE

RABBI ZIDNI 'ILMA

YENEPOYA
(DEEMED TO BE UNIVERSITY)

INTERNET OF THINGS LAB MANUAL

(COMMON FOR BCA/BSC)

SEMESTER	**:**	**4**	**CIA MARKS**	**:**	**50**
COURSE TITLE	**:**	**INTERNET OF THINGS PRACTICAL**	**SEE MARKS**	**:**	**50**
COURSE CODE	**:**	**BCA402P/BSC402P**	**TOTAL MARKS**	**:**	**100**
HOURS/WEEK	**:**	**4**	**CREDITS**	**:**	**2**
COURSE COORDINATOR: Mr. Praveen		**Ms. Aysha Dilshad Ms. Prathiksha Ms. Raksha Adyanthaya**			

The Yenepoya Institute of Arts, Science, Commerce and Management

A Constituent Unit of Yenepoya (Deemed to be University)

Balmatta, Mangalore - 575003

PART A	
1	**Controlling LEDs using an Arduino Microcontroller.**
2	**Set Up the DHT 11 Humidity sensor on an Arduino.**
3	**Detecting obstacle with IR sensor and Arduino**
4	**LDR (Light Dependent Resistor) / Light Sensor**
5	**Interfacing LCD_I2C (4 pins) with the Arduino Microcontroller.**
6	**Interfacing Servo Motor with the Arduino.**
7	**Interfacing Ultrasonic Sensor with the Arduino Microcontroller to measure the distance of the object.**
8	**Interfacing Smoke Sensor (MQ2) with the Arduino Microcontroller to detect the smoke from the surrounding.**
Part B	
1	**Controlling led using a Node MCU Esp8266 Microcontroller.**
2	**Detecting obstacle with IR sensor and Node MCU ESP8266.**
3	**LDR (Light Dependent Resistor) / Light Sensor**
4	**Interfacing Servo Motor with the Arduino.**
5	**Interfacing Ultrasonic Sensor with the Node MCU Esp8266 Microcontroller to measure the distance of the object.**

The Yenepoya Institute of Arts, Science, Commerce and Management

A Constituent Unit of Yenepoya (Deemed to be University)

Balmatta, Mangalore - 575003

PART – A

Exercise 1:

1. Controlling LEDs using an Arduino Microcontroller.

a. Blink

Aim:

To control the state of an LED using a microcontroller (Arduino).

Theory:

In the Blink operation, the Arduino board is programmed to turn an LED on and off at predefined intervals. This fundamental program introduces the concept of digital output. The onboard LED (often connected to pin 13 on Arduino Uno) is commonly used for this purpose.

The Arduino sketch for Blink utilizes the digitalWrite function to set the state of a specified digital pin. The LED is turned on by setting the pin to HIGH and turned off by setting it to LOW. delay functions introduce the concept of timing. The delay between turning the LED on and off creates the blinking effect. Understanding timing is crucial for more complex applications.

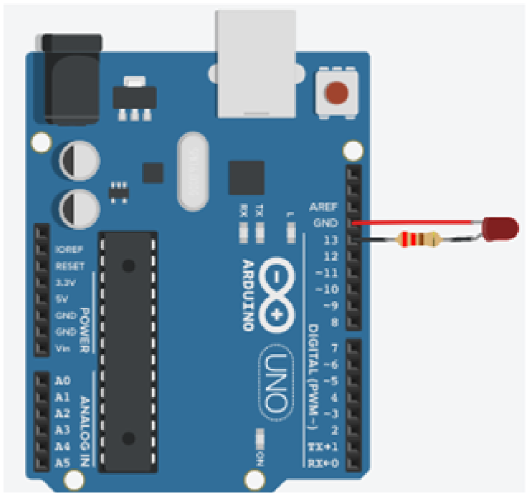
Application:

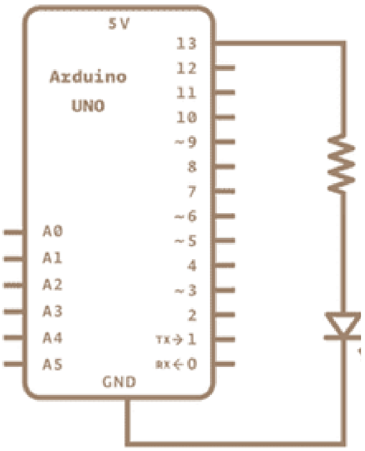
Traffic Light Simulation: The Blink concept can be expanded to simulate a traffic light system. By connecting LEDs to different digital pins, one can create a program that mimics the changing states of a traffic light.

Emergency Indicator: In real-world applications, Blink logic can be used for emergency indicators or warning lights, where an LED blinks to draw attention.

Heartbeat Indicator: In medical devices or wearable technology, a Blink-like operation could be applied to create a heartbeat indicator, providing a visual representation of a simulated heartbeat

Circuit Diagram





Procedure:

[Write the procedure]

Code:

```
int led=12
void setup()
        {
        pinMode(led, OUTPUT);
        }

void loop()
        {
        digitalWrite(led, HIGH);
        delay(1000);
        digitalWrite(led, LOW);
        delay(1000);
        }
```

Result:

- Led turns ON for 1 second.
- Led turns OFF for 1 second.
- The cycle repeats continuously.

Conclusion:

The Blink operation serves as the foundation for understanding digital output, timing, and basic programming concepts using Arduino. These principles become building blocks for more advanced projects, including robotics, automation, and IoT applications.

b. Fade

Aim:

To control the brightness of an LED using a microcontroller (Arduino) through the Fade operation.

Theory:

In the Fade operation, the Arduino board is programmed to smoothly vary the brightness of an LED, creating a fading effect. This program introduces the concept of analog output and Pulse Width Modulation (PWM). PWM is a technique where the LED is rapidly switched on and off at varying duty cycles to control the average power delivered, thereby adjusting the brightness.

The analogWrite function in the Arduino sketch is employed to set the intensity of the LED. This function allows for a range of values between 0 (completely off) and 255 (maximum brightness). The gradual transition between these values generates the fading effect.

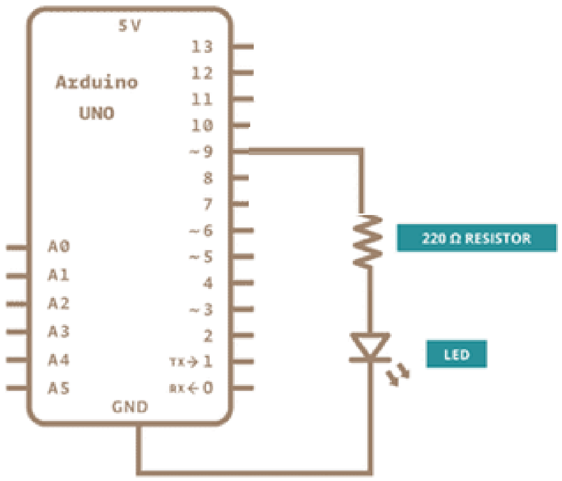
Application:

Dimmable Lighting System: The Fade concept can be extended to simulate a dimmable lighting system. By connecting LEDs to different digital pins and applying Fade logic, one can create a program that mimics the smooth transition of light intensity, providing a customizable ambient lighting experience.

Mood Lighting Controller: In home automation or entertainment systems, Fade logic can be employed to create a mood lighting controller. The LED's gradual change in brightness can be synchronized with different moods or scenarios, enhancing the overall atmosphere.

Sunrise/Sunset Simulator: Fade operation can be applied in projects like sunrise/sunset simulators, where the LED imitates the natural transition of light during dawn and dusk. This can be particularly useful in smart home applications or for creating natural lighting environments.

Circuit Diagram:



Code:

```
int led = 9;
int brightness = 0;
int fadeAmount = 5;
void setup()
        {
         pinMode(led, OUTPUT);
        }
void loop()
        {
        analogWrite(led, brightness);
        brightness = brightness + fadeAmount;
        if (brightness <= 0 || brightness >= 255)
                {
                 fadeAmount = -fadeAmount;
                }
        delay(10);
        }
```

Result:

• The LED connected to pin 9 fades in and out smoothly, creating a visually pleasing transition of brightness.

• The fadeAmount variable controls the rate of brightness change, and the delay(10) introduces a short delay between each intensity adjustment, contributing to the smooth fading effect.

• The LED's brightness transitions from 0 to 255 and vice versa in a continuous loop.

Conclusion:

Just like the Blink operation, the Fade program is a key learning tool. It helps users understand analog output and PWM (Pulse Width Modulation). As users progress, these concepts become crucial for creating various projects, from smart lighting systems to fun mood controllers in the world of microcontroller programming.

2. Set Up the DHT 11 Humidity sensor on an Arduino.

Aim:

To utilize a microcontroller (Arduino) to interface with a DHT (Digital Humidity and Temperature) sensor, and obtain accurate readings of humidity, temperature, and heat index.

Theory:

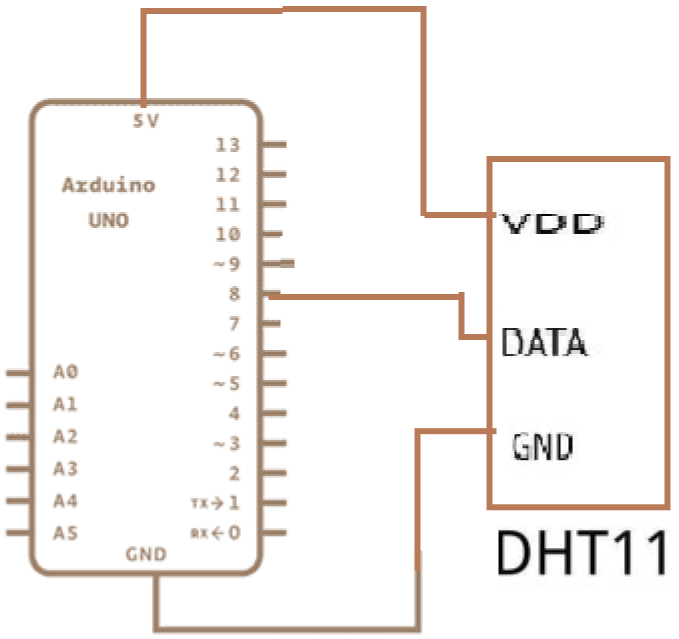
In the DHT Sensor experiment, the Arduino board is programmed to communicate with a DHT sensor (DHT11 in this case) connected to pin 2. The DHT library is employed to read humidity, temperature.

Application:

Climate Monitoring System: DHT sensors are commonly used in climate monitoring systems. The experiment can be extended to create a device that continuously monitors and reports changes in temperature, humidity, and heat index, providing valuable information for environmental control.

Greenhouse Automation: In greenhouse applications, DHT sensors play a crucial role in maintaining optimal growing conditions. The experiment's principles can be applied to automate climate control systems for plants, ensuring the right balance of temperature and humidity.

Circuit Diagram:



Code:

```
#include <dht.h>        // Include library
#define outPin 8        // Defines pin number to which the sensor is connected
dht DHT;                // Creates a DHT object

void setup()
     {
       Serial.begin(9600); // initialize serial communication at 9600 bits per second
     }

void loop()
     {
       int readData = DHT.read11(outPin);
       float t = DHT.temperature;       // Read temperature
       float h = DHT.humidity;          // Read humidity
       Serial.print("Temperature = ");
       Serial.print(t);
       Serial.print("°C | ");
       Serial.print((t*9.0)/5.0+32.0);       // Convert celsius to fahrenheit
       Serial.println("°F ");
       Serial.print("Humidity = ");
       Serial.print(h);
       Serial.println("% ");
```

```
  Serial.println("");
  delay(2000); // wait two seconds
}
```

Result:

• Serial monitor displays humidity, temperature in Celsius and Fahrenheit, and the calculated heat index every 2 seconds.
• The readings provide real-time information about the environmental conditions sensed by the DHT sensor.

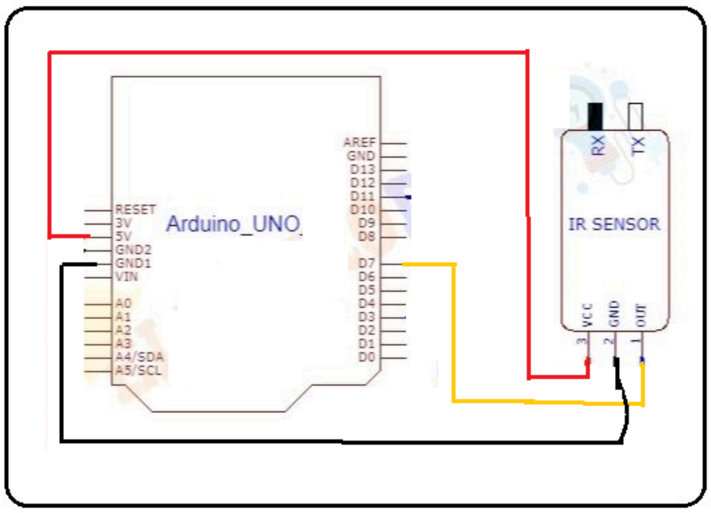
Conclusion:

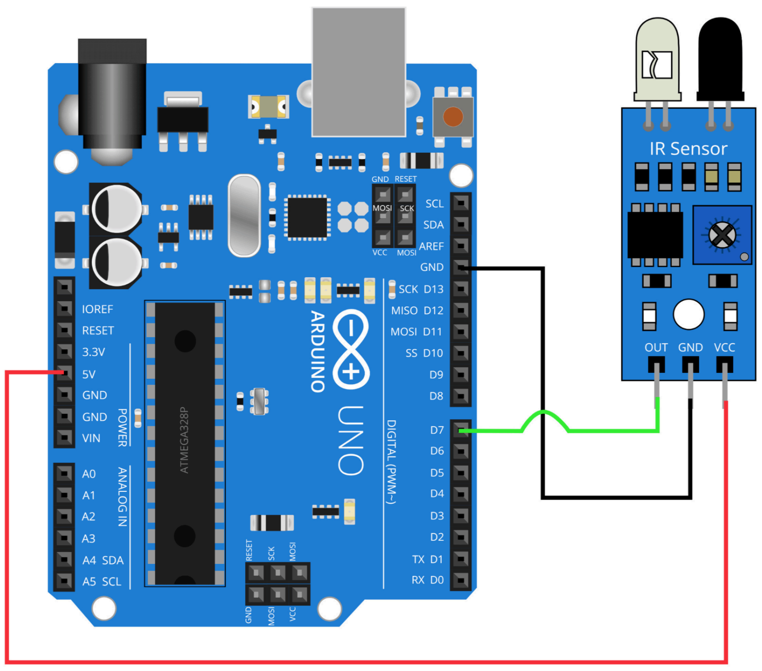
The DHT Sensor experiment provides foundational knowledge in interfacing with digital humidity and temperature sensors. The acquired data, covering humidity, temperature, and heat index, lays the groundwork for a range of applications, including climate monitoring systems and greenhouse automation.

3. Detecting obstacle with IR sensor and Arduino.

Aim:
To interface an Infrared (IR) sensor with a microcontroller (Arduino) and detect the obstacle with it.

Circuit Diagram:





Code:

```
int IRSensor = 7;
void setup()
        {
         pinMode (IRSensor, INPUT); // sensor pin INPUT
         Serial.begin (9600); // Starts the serial communication
        }

void loop()
        {
         //Define a variables for read the IRsensor
         int Sensordata = digitalRead (IRSensor);
         if (Sensordata == HIGH)
                {
                 Serial.print("Stop something is ahed");
                 Serial.println();
                 }
         else
                {
                Serial.print("Path is clear");
                Serial.println();
                 }
        delay(2000);
        }
```

Result:

Whenever an obstacle is present in front of the IR sensor it will detect and display that "Stop something is ahed" on the serial monitor otherwise it will display that " Path is clear".

4.LDR (Light Dependent Resistor) / Light Sensor

Aim: To interface a Light-Dependent Resistor (LDR) sensor with a microcontroller (Arduino), to demonstrate how the LED can be turned on or off based on the amount of light detected by the LDR.

Theory: In the LDR Sensor experiment, the Arduino board is programmed to read digital values from the LDR sensor connected to pin 2.The digital values represent the presence or absence of light.

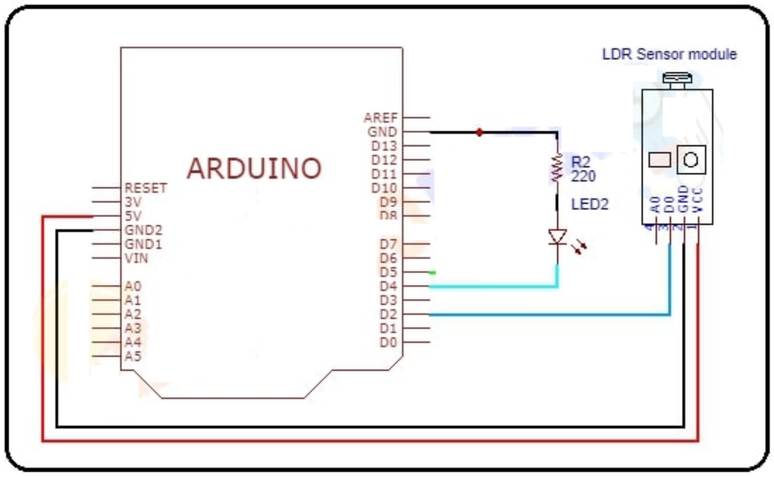
Application:

Ambient Light Control: This experiment serves as the foundation for creating systems that adjust ambient lighting based on the surrounding light intensity. For example, it can be applied to automatically control indoor lighting based on natural light conditions.

Energy-Efficient Systems: LDR sensors can be used in energy-efficient systems to regulate the brightness of displays, streetlights, or outdoor lighting. This contributes to energy conservation by adapting to changing light conditions.

Sunlight Harvesting: In solar energy applications, LDR sensors can be utilized for sunlight harvesting. The information obtained can be used to optimize the positioning of solar panels for maximum exposure to sunlight.

Circuit Diagram:



Code:

```
void setup()
        {
        pinMode(2, INPUT);
        pinMode(7, OUTPUT)
        Serial.begin(9600);
        }
void loop()
        {
        Int resistance = digitalRead(7);
        If (resistance == HIGH){
        Serial.println(" It is dark , LED is turned ON");
        digitalWrite(7,HIGH);
        }
else
        {
        Serial.println(" It is day time , LED is turned OFF");
        digitalWrite(7,LOW);
        }
delay(500);
}
```

Result:

- Serial monitor displays “it is dark time , LED is turned ON” if digital value is HIGH(if it is dark)
- The system continuously monitors the LDR sensor and turns ON or OFF the LED.

Conclusion: The LDR Sensor experiment measures the light intensity based on the resistance. If it is dark time it will turn on the LED else turn off the LED.

5. LCD_I2C (4 pins)

Aim:

To interface a Liquid Crystal Display (LCD) with an I2C connector to a microcontroller (Arduino), and display dynamic content on the screen.

Theory:

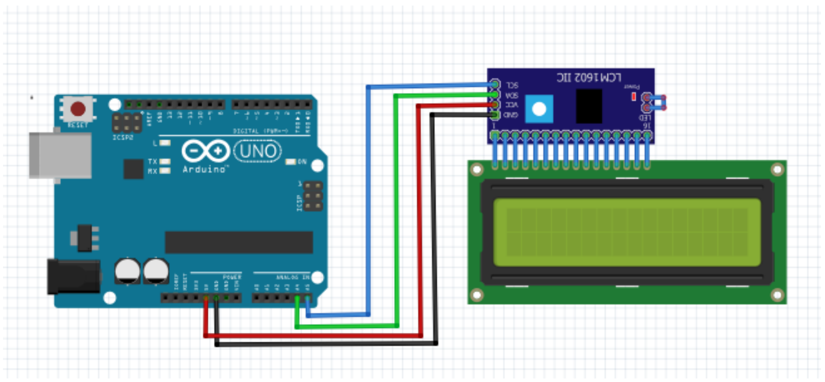
In this LCD with I2C experiment, the Arduino board is programmed to communicate with the LCD screen through the I2C interface. The LCD_I2C library is utilized to simplify the interaction. The LCD is initialized with its specific I2C address (0x3F) and the dimensions of the display (16 columns and 2 rows).

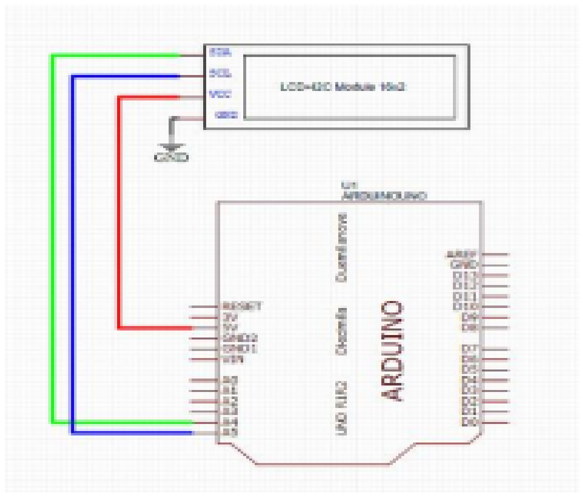
Application:

Information Display System: This experiment serves as the foundation for creating information display systems. The LCD with I2C connectivity can be integrated into projects that require real-time information presentation, such as weather stations, monitoring devices, or smart home control panels.

User Interface in Embedded Systems: The LCD can act as a user interface in embedded systems, offering a means to display relevant information to users interacting with a device.

Circuit Diagram:





Code:

```
#include <LCD_I2C.h>
LCD_I2C lcd(0x3F, 16, 2);
void setup()
        {
         lcd.begin();
         lcd.backlight();
        }
void loop()
        {
         lcd.print(" Hello");
         lcd.setCursor(5, 1);
         lcd.print("World!");
         delay(500);
         for (int i = 0; i < 5; ++i)
                {
                lcd.backlight();
                delay(100);
                lcd.noBacklight();
                delay(0);
                }
         lcd.backlight();
         lcd.clear();
         delay(500);
        }
```

Result:

• The LCD displays "Hello" on the first line and "World!" on the second line. • The backlight blinks for a visually appealing effect.

• The LCD is cleared, and the cycle repeats every 500 milliseconds.

Conclusion:

The LCD with I2C experiment demonstrates the seamless integration of an LCD screen with an Arduino using I2C communication. It provides a foundation for creating diverse projects, from informational display systems to user interfaces in embedded systems.

6. Interfacing Servo Motor with the Arduino.

Aim:
To control the position of a servo motor using a microcontroller (Arduino) through the Servo Motor operation.

Theory:
In the Servo Motor operation, the Arduino board is programmed to precisely control the angular position of a servo motor. A servo motor is a device that incorporates a feedback mechanism, allowing for accurate positioning. It is widely used in robotics and various automation applications.

The `Servo` library in Arduino facilitates the control of servo motors. The `attach` and `write` functions are utilized to assign a pin to the servo motor and set its position, respectively. The servo motor operates within a specified range, typically 0 to 180 degrees.

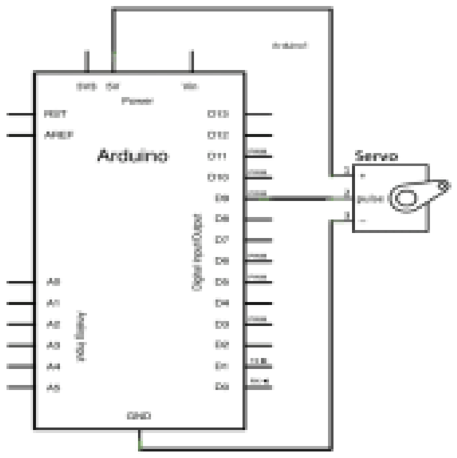
Application:

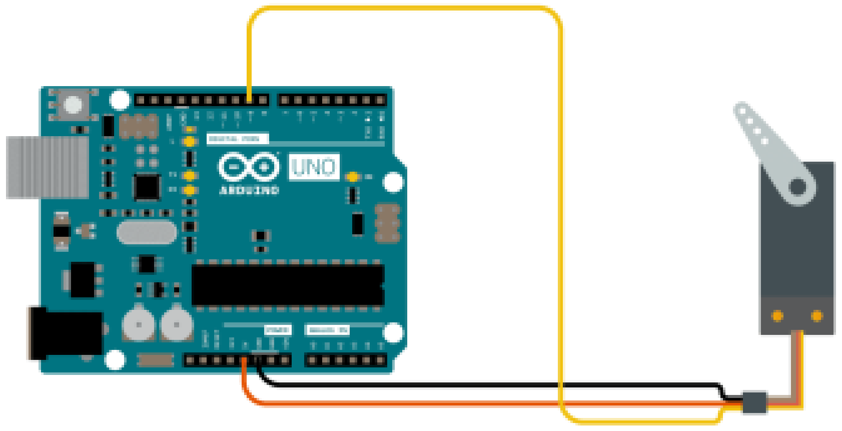
Door/Gate Control System: The Servo Motor concept can be extended to control doors or gates in home automation systems. By connecting the servo to a door or gate mechanism, the Arduino can precisely control the position of the servo motor to open or close the door.

Camera Pan-and-Tilt Mechanism: In robotics or surveillance applications, Servo Motor logic can be employed to create a pan-and-tilt system for a camera. The servo motors control the horizontal and vertical movements, allowing the camera to cover a wide area.

Automated Window Blinds: Servo Motor operation can be applied to automate window blinds. The servo motor adjusts the slats' angle to control the amount of light entering a room, enhancing energy efficiency.

Circuit Diagram:





Code:

```
#include <Servo.h>
Servo myservo;
int pos = 0;
void setup()
        {
        myservo.attach(9);
        }
void loop()
        {
        for(pos = 0; pos <= 180; pos += 1)
                {
                myservo.write(pos);
                delay(15);
                 }
         for(pos = 180; pos >= 0; pos -= 1)
                {
                 myservo.write(pos);
                 delay(15);
                 }
        }
```

Result:

- The servo motor connected to pin 9 smoothly sweeps from 0 to 180 degrees in one direction.
- After reaching 180 degrees, the servo motor smoothly sweeps back from 180 to 0 degrees.
- This sweeping motion repeats continuously, creating a visually perceivable back-and-forth movement.

Conclusion:

The Servo Motor control experiment is crucial in microcontroller programming, offering a foundational understanding of precise motor control. This hands-on experience forms the basis for diverse applications in robotics, automation, and smart home systems.

7. Ultrasonic Sensor

Aim:

To interface an ultrasonic sensor with a microcontroller (Arduino), aiming to measure and display the distance of an object from the sensor.

Theory:

In the Ultrasonic Sensor experiment, the Arduino board is programmed to communicate with the ultrasonic sensor using two pins: trigPin for triggering the sensor and echoPin for receiving the echo signal. The sensor works by emitting ultrasonic waves, and the time taken for the waves to bounce back is used to calculate the distance. The formula distance = duration * 0.034 / 2 converts the time into distance in centimeters.

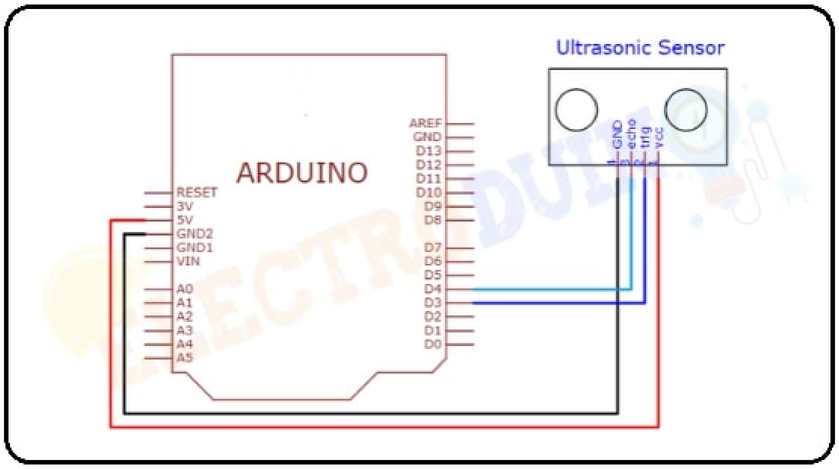
Application:

Distance Measurement System: This experiment serves as the foundation for creating distance measurement systems. Ultrasonic sensors are commonly used in robotics, parking assistance systems, and object detection applications.

Smart Security Systems: Ultrasonic sensors can be applied in smart security systems for proximity detection. This includes alerting or triggering actions based on the distance of an object from the sensor.

Automated Navigation in Robotics: In robotics, ultrasonic sensors are often employed for obstacle avoidance and navigation. The distance data obtained can be used to guide the robot's movements.

Circuit Diagram:



Code:

```
#define echoPin 4
#define trigPin 3
long duration;
int distance;
void setup()
        {
        pinMode(trigPin, OUTPUT);
        pinMode(echoPin, INPUT);
        Serial.begin(9600);
        }
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(500);
        }
```

Result:

• Serial monitor displays "Distance: [Distance in cm]" every 500 milliseconds. • The ultrasonic sensor continuously measures the distance, updating the value based on the object's proximity.

Conclusion:

The Ultrasonic Sensor experiment demonstrates the practical application of an ultrasonic sensor for distance measurement. It forms the basis for creating systems that can intelligently respond to objects in their vicinity.

8. Smoke Sensor (MQ2)

Aim:
To interface a Smoke Sensor (MQ2) with a microcontroller (Arduino), with the aim of monitoring and responding to smoke levels.

Theory:

In the Smoke Sensor (MQ2) experiment, the Arduino board is programmed to read analog values from the smoke sensor connected to pin A1. The analog values represent the concentration of smoke or other gasses in the environment. A threshold value is set, and if the sensor reading falls below this threshold, an external LED connected to pin 2 is turned on.

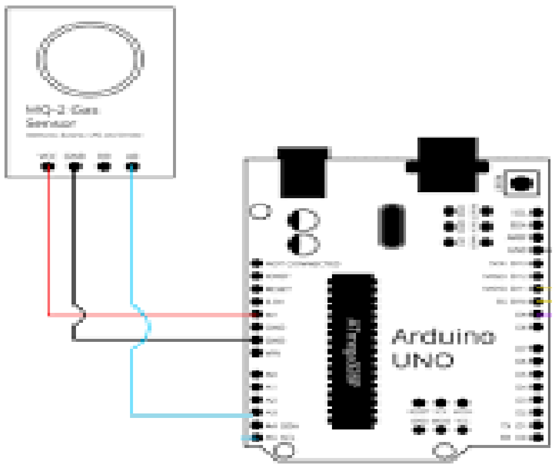
Application:

Smoke Detection System: This experiment serves as the basis for creating a smoke detection system. The Smoke Sensor (MQ2) is commonly used in applications where the detection of smoke or harmful gasses is crucial, such as in fire alarm systems.

Air Quality Monitoring: The MQ2 sensor can be applied to monitor air quality by detecting various gasses. This is useful in indoor environments to ensure a healthy and safe atmosphere.

Security Systems: Smoke sensors can be integrated into security systems to detect unauthorized access, triggering actions such as turning on lights or sounding alarms when smoke is detected.

Circuit Diagram:



Code:

```
int smoke=A3;
void setup()
        {
         Serial.begin(9600);
         pinMode(A3,INPUT);
         pinMode(2,OUTPUT);
        }
void loop()
        {
         int ss=analogRead(A3);
         Serial.println("smoke value is");
         Serial.println(ss);
         if(ss>400)
                 digitalWrite(2,HIGH);
         else
                 digitalWrite(2,LOW);
        }
```

Result:

• Serial monitor displays "smoke value is [Analog Reading]" continuously.

• The external LED turns on if the smoke concentration falls below the set threshold (condition: `ss > 400`), indicating a potential smoke presence.

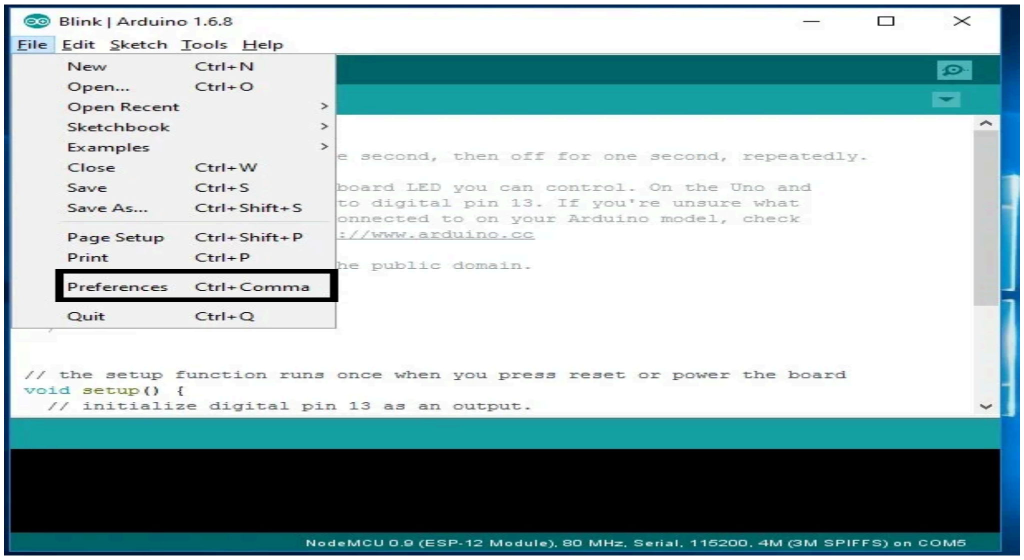
Conclusion:

The Smoke Sensor (MQ2) experiment showcases the practical application of a smoke sensor in monitoring environmental conditions. It provides the foundation for developing smoke detection systems that can be integrated into safety and security solutions.

Node Mcu Controller

Steps to Setup Arduino IDE for NODEMCU ESP8266

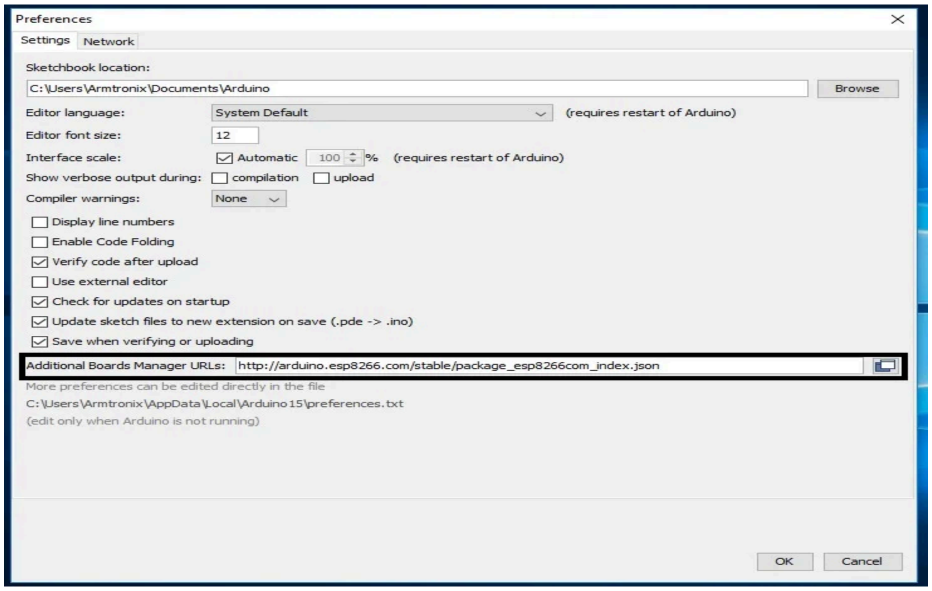
After installing Arduino IDE,
Step:1 Open Arduino IDE, go to the File and click on the Preferences as shown in the figure



```
e second, then off for one second, repeatedly.

board LED you can control. On the Uno and
to digital pin 13. If you're unsure what
onnected to on your Arduino model, check
://www.arduino.cc

he public domain.

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

Step: 2 Adding ESP8266 Board Manager
In the Additional Boards Manager enter below URL and then click OK.

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Preferences

Settings | Network

Sketchbook location: C:\Users\Armtronix\Documents\Arduino [Browse]

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers

☐ Enable Code Folding

☑ Verify code after upload

☐ Use external editor

☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

☑ Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

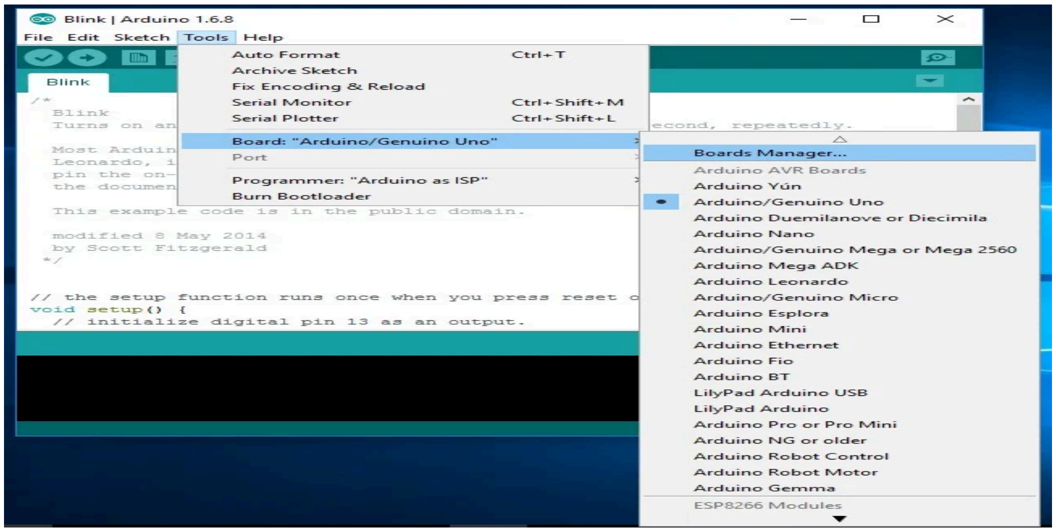
C:\Users\Armtronix\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

[OK] [Cancel]

Step: 3 Selecting Board

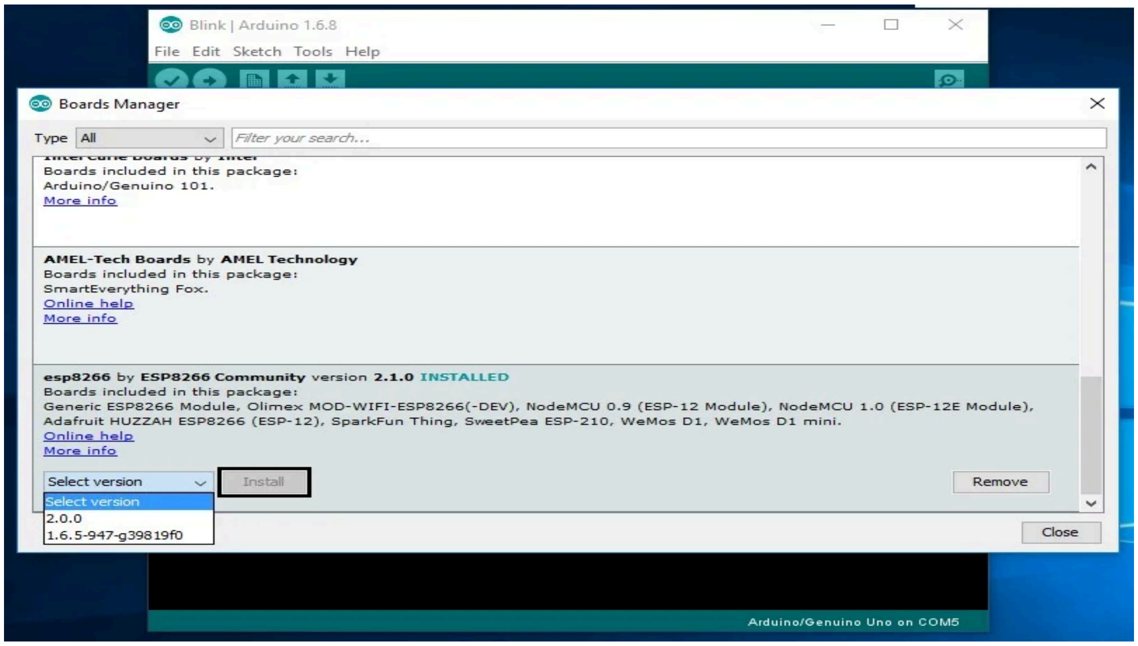
Now open the tools in that select **Board: "Arduino/Genuino Uno"** and click on the **Boards Manager** as shown in the figure



```
/*
  Blink
  Turns on an                          econd, repeatedly.

  Most Arduin
  Leonardo, i
  pin the on-
  the documen

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */

// the setup function runs once when you press reset o
void setup() {
  // initialize digital pin 13 as an output.
```

Step:5 ESP8266 Board Package

The Boards Manager window opens, scroll the window page to bottom till you see the module with the name ESP8266 or else search for ESP8266 by the ESP8266 community. Once we get it, select that module and select version and click on the Install button. When it is installed it shows Installed in the module as shown in the figure and then close the window.



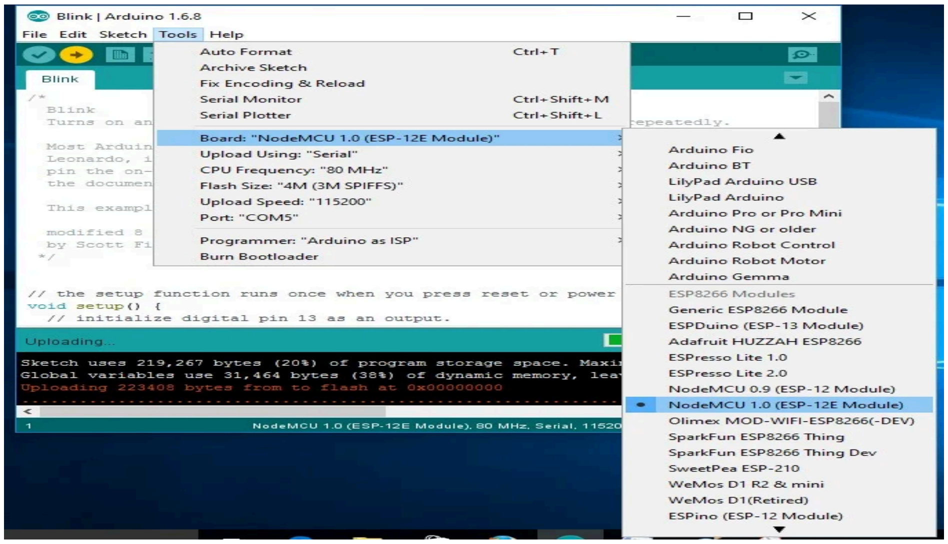
Boards included in this package:
Arduino/Genuino 101.
More info

AMEL-Tech Boards by AMEL Technology
Boards included in this package:
SmartEverything Fox.
Online help
More info

esp8266 by ESP8266 Community version 2.1.0 INSTALLED
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini.
Online help
More info

Step: 6 Selecting ESP8266 Arduino Board

To run the esp8266 with Arduino we have to select the **Board: "Arduino/Genuino Uno"** and then change it to **NodeMCU 1.0 (ESP-12E Module)**



Board: "NodeMCU 1.0 (ESP-12E Module)"
Upload Using: "Serial"
CPU Frequency: "80 MHz"
Flash Size: "4M (3M SPIFFS)"
Upload Speed: "115200"
Port: "COM5"
Programmer: "Arduino as ISP"

```
// the setup function runs once when you press reset or power
void setup() {
  // initialize digital pin 13 as an output.
```

```
Sketch uses 219,267 bytes (20%) of program storage space. Maxi
Global variables use 31,464 bytes (38%) of dynamic memory, lea
Uploading 223408 bytes from to flash at 0x00000000
```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, Serial, 11520

1. Controlling led using a Node MCU Esp8266 Microcontroller.

Aim:

To control the state of an LED using a microcontroller (Node MCU Esp8266).

Theory:

In the Blink operation, the node mcu board is programmed to turn an LED on and off at predefined intervals. This fundamental program introduces the concept of digital output.

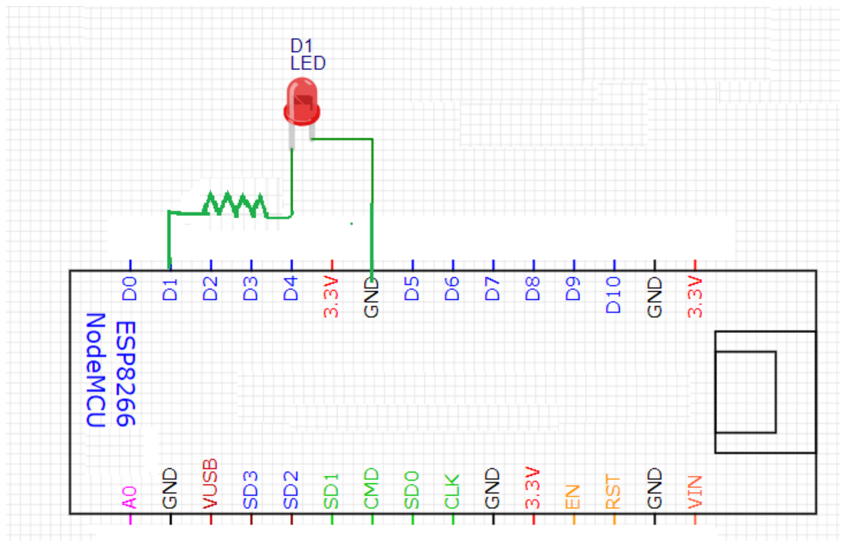
The Arduino sketch for Blink utilizes the digitalWrite function to set the state of a specified digital pin. The LED is turned on by setting the pin to HIGH and turned off by setting it to LOW. delay functions introduce the concept of timing. The delay between turning the LED on and off creates the blinking effect. Understanding timing is crucial for more complex applications.

Application:

Traffic Light Simulation: The Blink concept can be expanded to simulate a traffic light system. By connecting LEDs to different digital pins, one can create a program that mimics the changing states of a traffic light.

Emergency Indicator: In real-world applications, Blink logic can be used for emergency indicators or warning lights, where an LED blinks to draw attention.

Heartbeat Indicator: In medical devices or wearable technology, a Blink-like operation could be applied to create a heartbeat indicator, providing a visual representation of a simulated heartbea



Code :

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Result:

- D1 turns ON for 1 second.
- D1 turns OFF for 1 second.
- The cycle repeats continuously.

Conclusion:

The Blink operation serves as the foundation for understanding digital output, timing, and basic programming concepts using Node MCU. These principles become building blocks for more advanced projects, including robotics, automation, and IoT applications.

2. Detecting obstacle with IR sensor and Node MCU ESP8266.

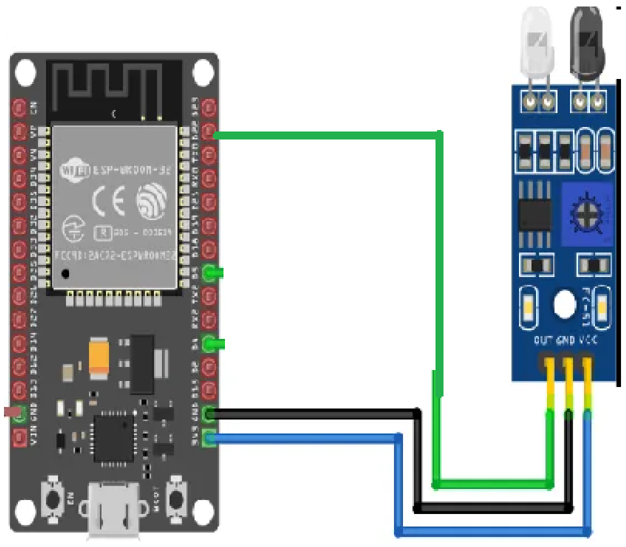
Aim:

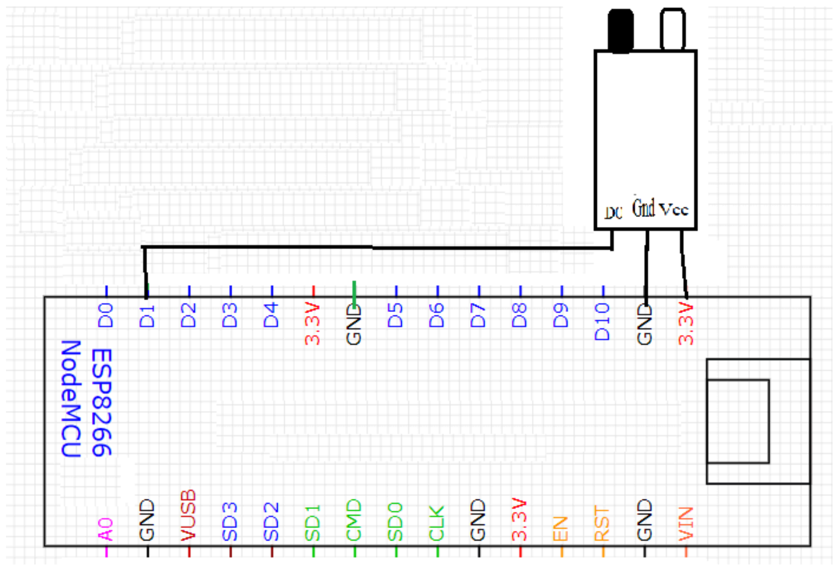
To interface an Infrared (IR) sensor with a microcontroller (Node MCU) and detect the obstacle with it.

Theory:

In the IR Sensor experiment, the Node MCU board is programmed to read digital values from the IR sensor connected to pin D1. The digital values represent the presence or absence of obstacles.

Block Diagram:





Code:

```
void setup()

 {

  pinMode (D1, INPUT); // sensor pin INPUT

  Serial.begin (9600); // Starts the serial communication

 }



void loop()

 {

  //Define a variables for read the IRsensor

  int Sensordata = digitalRead (D1);
  if (Sensordata == HIGH){

    Serial.print("Path is clear");

       Serial.println();

  }

  else{

     Serial.print("Stop something is ahed");

          Serial.println();

  }

  delay(2000);

 }
```

Result:

Whenever an obstacle is present in front of the IR sensor it will detect and display that "Stop something is ahed" on the serial monitor otherwise it will display that " Path is clear".

3. LDR (Light Dependent Resistor) / Light Sensor

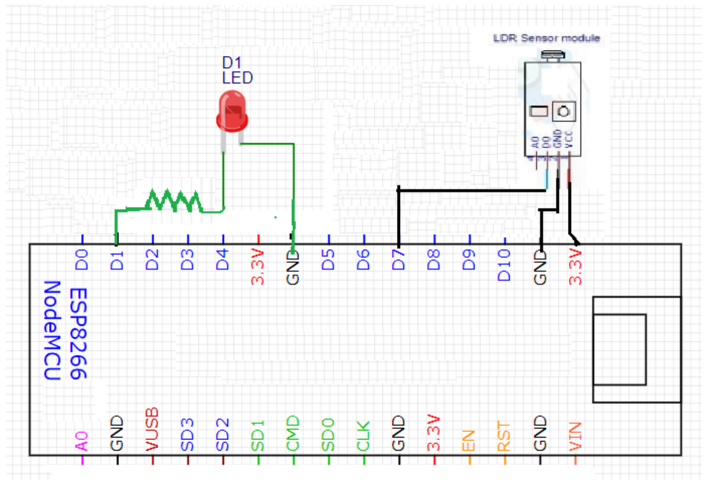
Aim:

To interface a Light-Dependent Resistor (LDR) sensor with a microcontroller (Node MCU), to demonstrate how the LED can be turned on or off based on the amount of light detected by the LDR.

Theory:

In the LDR Sensor experiment, the Node MCU board is programmed to read digital values from the LDR sensor connected to pin D7.The digital values represent the presence or absence of light.

Block Diagram:



Code:

```
void setup() {
 Serial.begin(9600);
pinMode(D1,OUTPUT);
pinMode(D7,INPUT);
}
void loop() {
int Resistance= digitalRead(2);

if (Resistance==HIGH){

println(" it is dark time , LED is turned
ON”);

digitalWrite(D1,HIGH);

}

else

{

println(“ it is day time , LED is turned
OFF”);

digitalWrite(D1,LOW);

}

delay(2000);

}
```

Result:

• Serial monitor displays “it is dark time , LED is turned ON” if digital value is HIGH(if it is dark)

• The system continuously monitors the LDR sensor and turns ON or OFF the LED.

Conclusion:

The LDR Sensor experiment measures the light intensity based on the resistance. If it is dark time it will turn on the LED else turn off the LED.

4. Interfacing Servo Motor with the Arduino.

Aim:

To control the position of a servo motor using a microcontroller (Node MCU) through the Servo Motor operation.

Theory:

In the Servo Motor operation, the Node MCU board is programmed to precisely control the angular position of a servo motor. A servo motor is a device that incorporates a feedback mechanism, allowing for accurate positioning. It is widely used in robotics and various automation applications.

The Servo library in Arduino facilitates the control of servo motors. The attach and write functions are utilized to assign a pin to the servo motor and set its position, respectively. The servo motor operates within a specified range, typically 0 to 180 degrees.

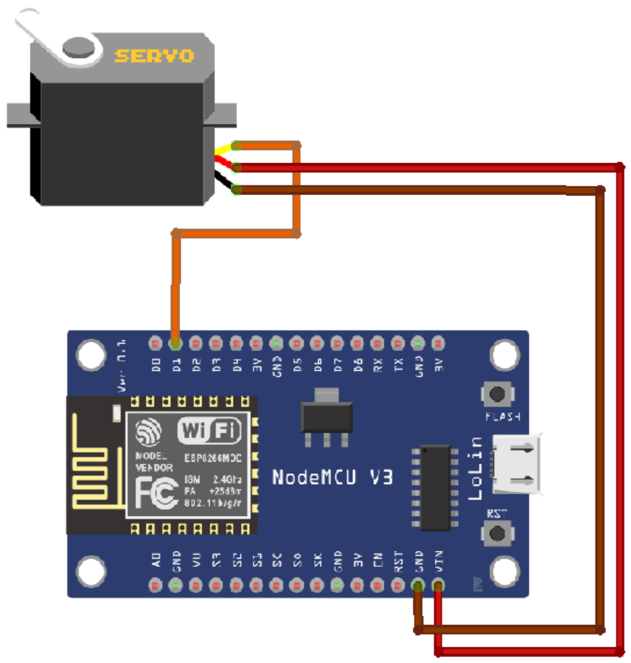
Application:

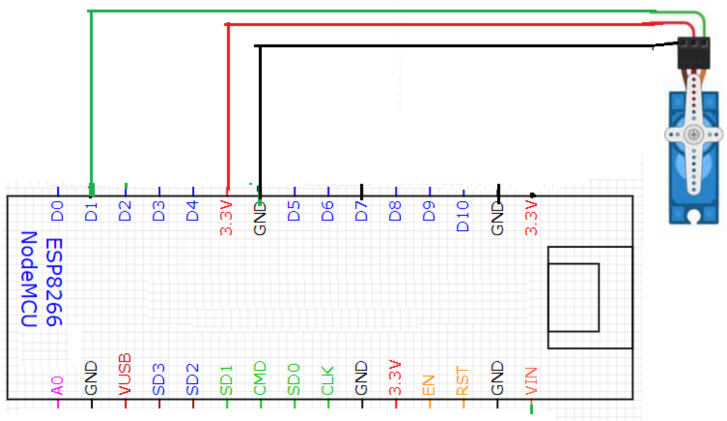
Door/Gate Control System: The Servo Motor concept can be extended to control doors or gates in home automation systems. By connecting the servo to a door or gate mechanism, the Arduino can precisely control the position of the servo motor to open or close the door.

Camera Pan-and-Tilt Mechanism: In robotics or surveillance applications, Servo Motor logic can be employed to create a pan-and-tilt system for a camera. The servo motors control the horizontal and vertical movements, allowing the camera to cover a wide area.

Automated Window Blinds: Servo Motor operation can be applied to automate window blinds. The servo motor adjusts the slats' angle to control the amount of light entering a room, enhancing energy efficiency.

Block Diagram:


SERVO
NodeMCU V3
LoLin
FLASH
RST
D0 D1 D2 D3 D4 3V GND D5 D6 D7 D8 RX TX GND 3V
A0 GND VU S3 S2 S1 SC S0 SK GND 3V EN RST GND VIN


ESP8266
NodeMCU
D0 D1 D2 D3 D4 3.3V GND D5 D6 D7 D8 D9 D10 GND 3.3V
A0 GND VUSB SD3 SD2 SD1 CMD SD0 CLK GND 3.3V EN RST GND VIN

Code:

```
#include <Servo.h>

Servo myservo;

int pos = 0;

void setup() {
 myservo.attach(D1);

}

void loop() {

 for (pos = 0; pos <= 180; pos += 1) {
 myservo.write(pos);

 delay(15);

 }

 for (pos = 180; pos >= 0; pos -= 1) {

 myservo.write(pos);

 delay(15);

 }

}
```

Result:

• The servo motor connected to pin D1 smoothly sweeps from 0 to 180 degrees in one direction.

• After reaching 180 degrees, the servo motor smoothly sweeps back from 180 to 0 degrees.

• This sweeping motion repeats continuously, creating a visually perceivable back-and-forth movement.

Conclusion:

The Servo Motor control experiment is crucial in microcontroller programming, offering a foundational understanding of precise motor control. This hands-on experience forms the basis for diverse applications in robotics, automation, and smart home systems.

5. Ultrasonic Sensor

Aim:

To interface an ultrasonic sensor with a microcontroller (Node MCU), aiming to measure and display the distance of an object from the sensor.

Theory:

In the Ultrasonic Sensor experiment, the Node MCU board is programmed to communicate with the ultrasonic sensor using two pins: trigPin for triggering the sensor and echoPin for receiving the echo signal. The sensor works by emitting ultrasonic waves, and the time taken for the waves to bounce back is used to calculate the distance. The formula distance = duration * 0.034 / 2 converts the time into distance in centimeters.

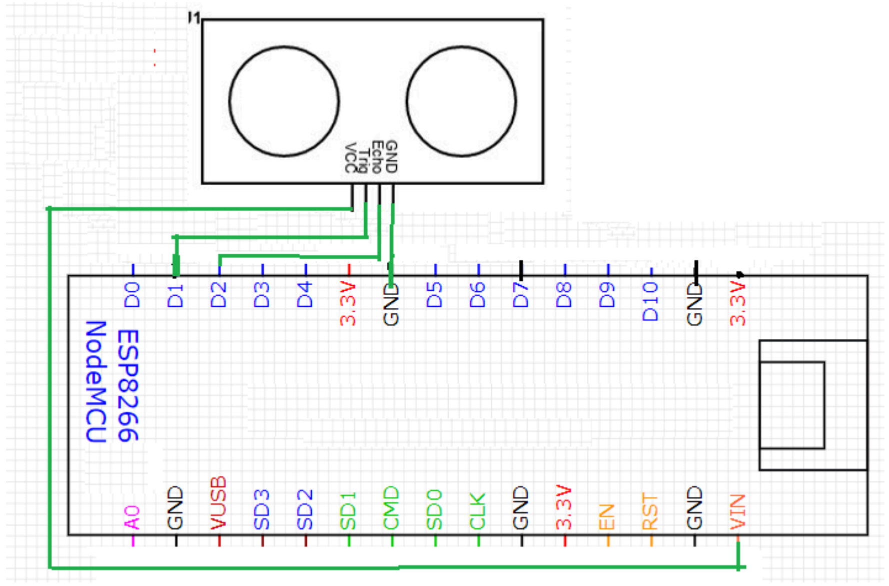
Application:

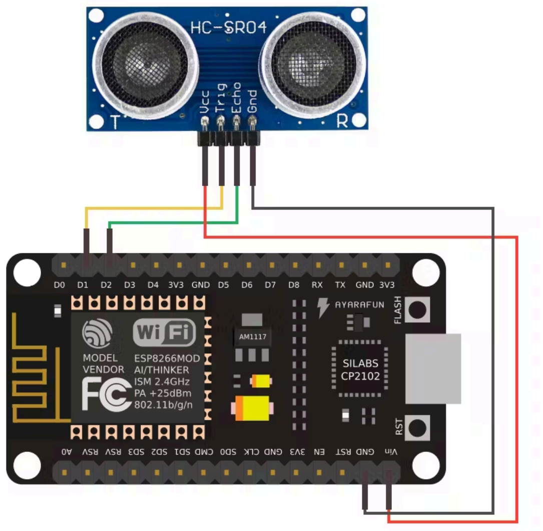
Distance Measurement System: This experiment serves as the foundation for creating distance measurement systems. Ultrasonic sensors are commonly used in robotics, parking assistance systems, and object detection applications.

Smart Security Systems: Ultrasonic sensors can be applied in smart security systems for proximity detection. This includes alerting or triggering actions based on the distance of an object from the sensor.

Automated Navigation in Robotics: In robotics, ultrasonic sensors are often employed for obstacle avoidance and navigation. The distance data obtained can be used to guide the robot's movements.

Block Diagram:




HC-SR04
Vcc
Trig
Echo
Gnd
T
R
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
A0 RSV RSV SD3 SD2 SD1 CMD SD0 CLK GND 3V3 EN RST GND Vin
ESP8266MOD
SILABS CP2102
FLASH
RST

Code:

```
#define echoPin D2

#define trigPin D1

long duration;

int distance;

void setup() {
```

```
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
Serial.begin(9600);
}
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(500);
 }
```

Result:

• Serial monitor displays "Distance: [Distance in cm]" every 500 milliseconds. • The ultrasonic sensor continuously measures the distance, updating the value based on the object's proximity.

Conclusion:

The Ultrasonic Sensor experiment demonstrates the practical application of an ultrasonic sensor for distance measurement. It forms the basis for creating systems that can intelligently respond to objects in their vicinity.