Dr. D. Y. Patil Institute of Technology, Pimpri, Pune

Department of Electronics & Telecommunication Engineering

Academic Year 2024-2024 Semester VII

BE

Subject – MIOT

EXPERIMENT NO.- 1

Title: Study of Raspberry Pi 4, Arduino board and Operating systems for the same. Understand the process of OS installation on the Raspberry Pi.

Hardware Requirements: Arduino Board, Raspberry Pi Board

Software Requirements: Arduino IDE, Noobs OS

Theory: Arduino:

An Arduino board consists of an Atmel 8-bit AVR microcontroller with complementary components that facilitate programming and incorporation into other circuits. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Fig.1 Arduino

Specification of Arduino

- Microcontroller :ATmega328P
- Operating Voltage :5V
- Input Voltage (recommended) :7-12V
- Input Voltage (limit) :6-20V
- Digital I/O Pins :14 (of which 6 provide PWM output)

- Analog Input Pins 6
- DC Current per I/O Pin :20 mA
- DC Current for 3.3V Pin :50 mA
- Flash Memory :32 KB (ATmega328P) of which 0.5 KB used by bootloader
- SRAM :2 KB (ATmega328P)
- EEPROM :1 KB (ATmega328P)
- Clock Speed :16 MHz
- LED_BUILTIN 13

Raspberry Pi :

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.

The original model became far more popular than anticipated, selling outside of its target market for uses such as robotics.

Over 5 million Raspberry Pis have been sold before February 2015, making it the best- selling British computer. By November 2016 they had sold 11 million units.

Fig.2 Raspberry Pi

Featur

- All models feature a Broadcom system on a chip (SoC), which includes an ARM compatible central processing unit (CPU) and an on-chip graphics processing unit (GPU, a VideoCore IV).
- CPU speed ranges from 700 MHz to 1.5 GHz for the Pi 4 and on board memory range from 256 MB to 4 GB RAM.
- Secure Digital (SD) cards are used to store the operating system and program memory in either the SDHC or Micro SDHC sizes.
- Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phone jack for audio.
- Lower level output is provided by a number of GPIO pins which support common protocols like I²C.
- The B-models have an 8P8C Ethernet port and the Pi 3 and Pi Zero W have on board Wi-Fi 802.11n and Bluetooth.

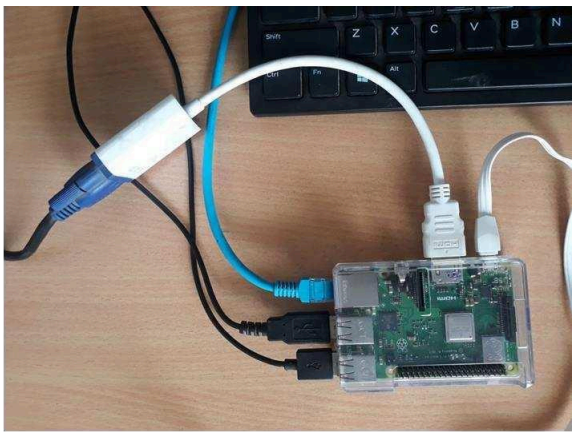
Operating Systems:

- The Foundation provides Raspbian, a Debianbased Linux distribution for download, as well as third party Ubuntu, Windows 10 IOT Core, RISC OS, and specialised media center distributions.
- It promotes Python and Scratch as the main programming language, with support for many other languages.
- The default firmware is closed source, while an unofficial open source is available.

Procedure to install the Operating System:

- Install the SD Formatter software in Computer/Laptop
- Insert the micro SD card into the card reader and connect it to Computer/Laptop
- Download the OS from official web site of raspberry pi
  https://www.raspberrypi.org/downloads/noobs/
- Download zip file
- Extract the zip file into SD card
- Insert the micro SD card into the slot on the underside of the Pi
- Plug in keyboard and mouse
- Plug in monitor using the HDMI port
- The Raspberry Pi doesn't have a power button. It boots up as soon as you plug in the power supply
- If you've completed all the previous steps, plug in the power supply to boot the Raspberry Pi
- Select correct drive & format the SD card

Setup of the Raspberry Pi



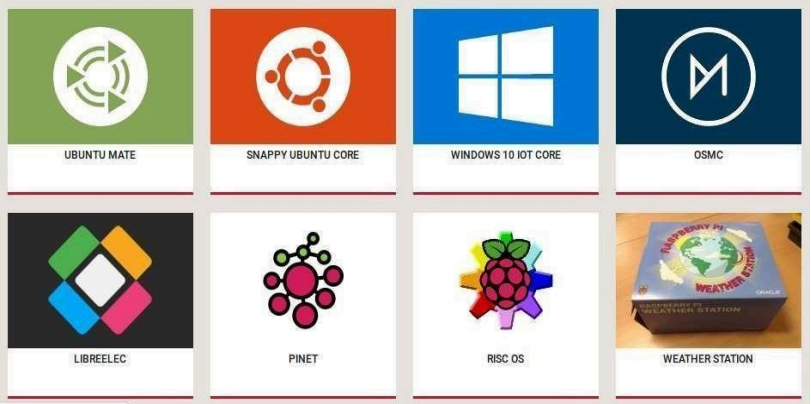


Fig.3 Symbol of

Periphe

Fig.4 Peripherals required for assembling

Conclusion:

Dr. D. Y. Patil Institute of Technology, Pimpri, Pune

Department of Electronics & Telecommunication Engineering

Academic Year 2024-2025 Semester VII

BE

Subject – MIOT

EXPERIMENT NO.- 2

Title: Understand the connection and configuration of GPIO and its use in programming. Write an application of the use of push switch and LEDs.

Aim: study of Open-source prototype platform- for Arduino.

Part A) Write a program for digital read/write using LED and Switch.

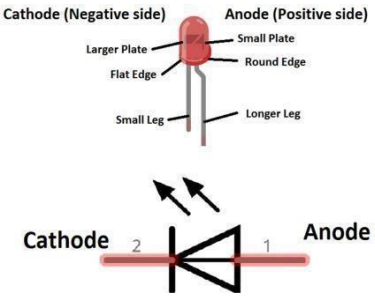
Part B) Write a program Analog read/write using Potentiometer.

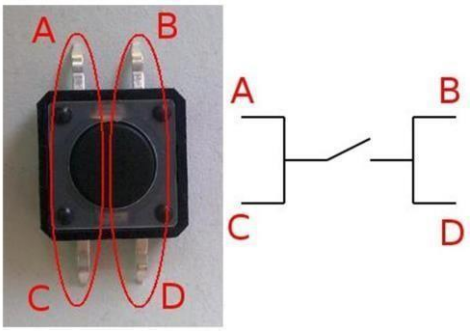
Software Requirements: Open source simulator like Tinkercad, Wowki, Proteus

Theory:

LED:

A light-emitting diode (LED) 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. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light- emitting phosphor on the semiconductor device





Push Button:

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism to control some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal.[1] The surface is usually flat or shaped

to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, although many un-biased buttons (due to their physical nature) still require a spring to return to their un-pushed state. Terms for the "pushing" of a button include pressing, depressing, mashing, slapping, hitting, and punching.

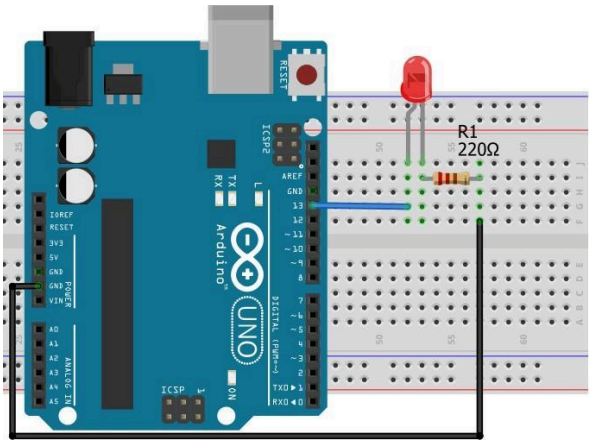


Fig.1 Interfacing of LED

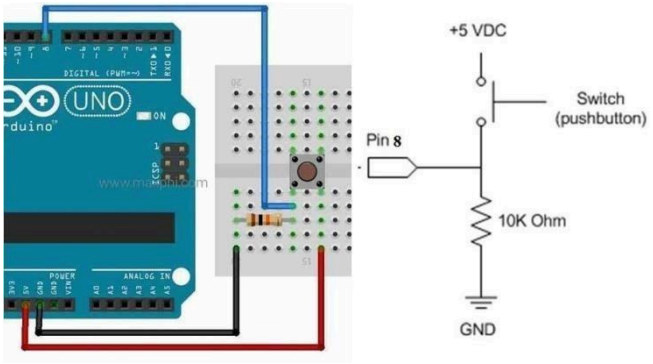


Fig.2 Interfacing of Push Button

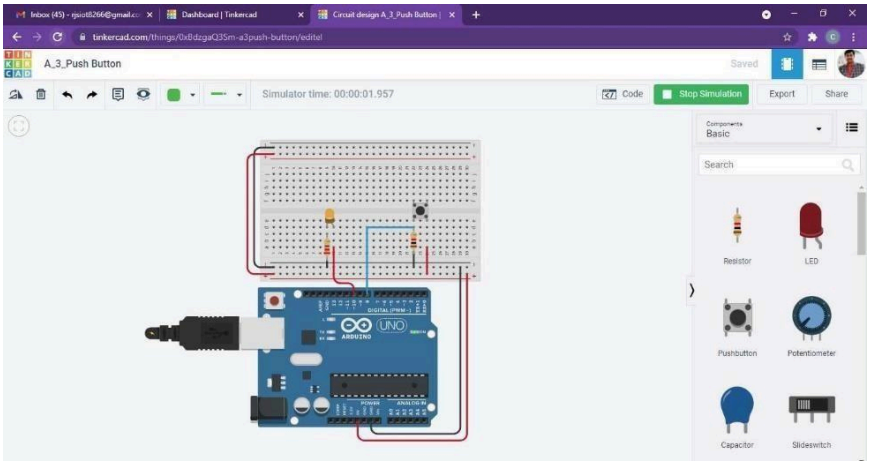


Fig.3 Simulated Result of LED & Push Button

Procedure:

1. Open simulator and sign in.
2. Select Arduino Uno.
3. Make the connection as per circuit diagram.
4. Write code in Editor Window and save file
5. Check for errors if any.
6. Start simulation.

Code:

Part A: Interfacing Arduino with LED & Push button.

```
const int buttonPin = 8; // Digital Pin 8 define as a push Button pin
const int ledPin = 10; // Digital Pin 13 define as LED Pin
int buttonState = 0; // variable for reading the pushbutton status

void setup()
{
pinMode(ledPin , OUTPUT);
pinMode(buttonPin ,
INPUT);
}

void loop()
{
buttonState =
digitalRead(buttonPin); if
(buttonState == HIGH)
{
digitalWrite(ledPin , HIGH);
}
else
{
digitalWrite(ledPin , LOW);
}
}
```

Part B: Interfacing Arduino with Potentiometer.

```
void setup()
{
  Serial.begin(9600);
}

void loop()
 {
  int sensorValue=analogRead(A0);
  int percent=map(sensorValue,0,1023,0,100);
  Serial.println(sensorValue);
  delay(100);
  }
```

Conclusion:

_____
_____
_____
_____

Dr. D. Y. Patil Institute of Technology, Pimpri, Pune

Department of Electronics & Telecommunication Engineering

Academic Year 2024-2025 Semester VII

BE

Subject – MIOT

EXPERIMENT NO.- 3

Title: Study of different sensors: - Temperature sensor, IR sensor, Ultrasonic sensor

Aim: Study of different sensors: - Temperature sensor, Gas sensor.

Hardware Requirements: Arduino Board, LM 35 Temp. Sensor, IR Sensor, UltrasonicSensor

Software Requirements: Arduino IDE

Theory:

LM 35 Temperature Sensor:

- LM35 is a temperature measuring device having an analog output voltage proportional to the temperature.
- It provides output voltage in Centigrade (Celsius). It does not require any external calibration

circuitry.

- The sensitivity of LM35 is 10 mV/degree Celsius. As temperature increases, output voltagealso increases.

E.g. 250 mV means 25°C.

It is a 3-terminal sensor used to measure surrounding temperature ranging from -55 °C to 150 °C.

LM35 gives temperature output which is more precise than thermistor output.

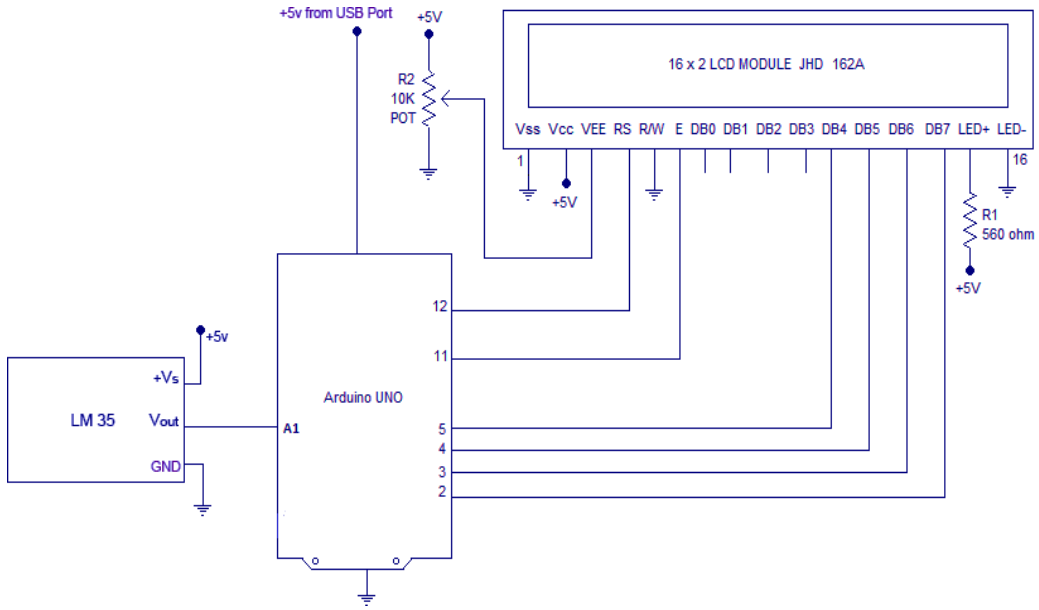


Fig.1 Interfacing of LM 35

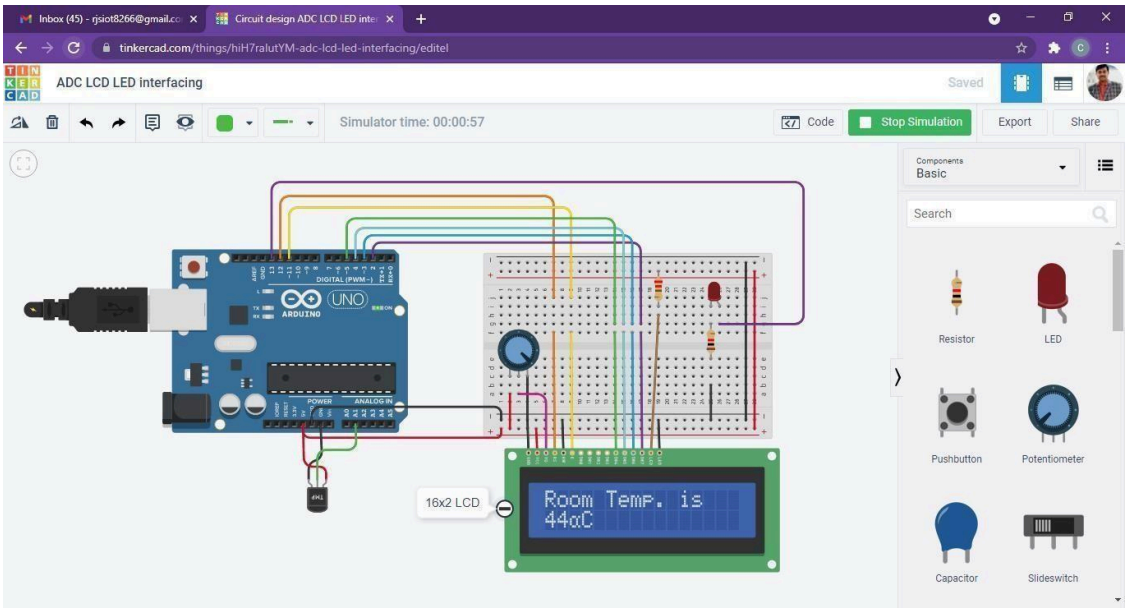


Fig.2 Simulated Result of LM 35

IR Sensor

IR sensor is an electronic device that emits the light in order to sense some object of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations.

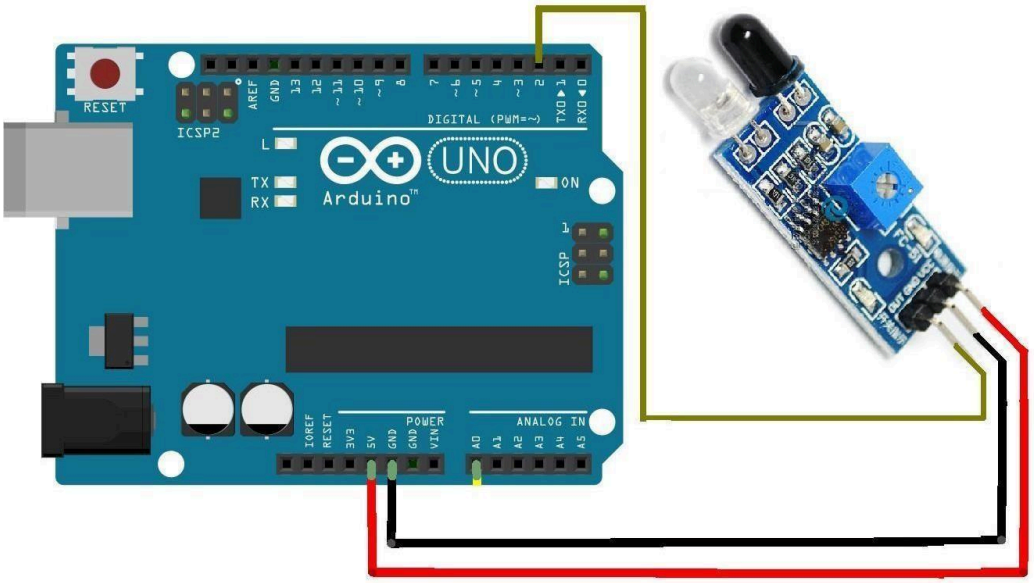


Fig.3 Interfacing of IR Sensor

- **Ultrasonic Sensor:**

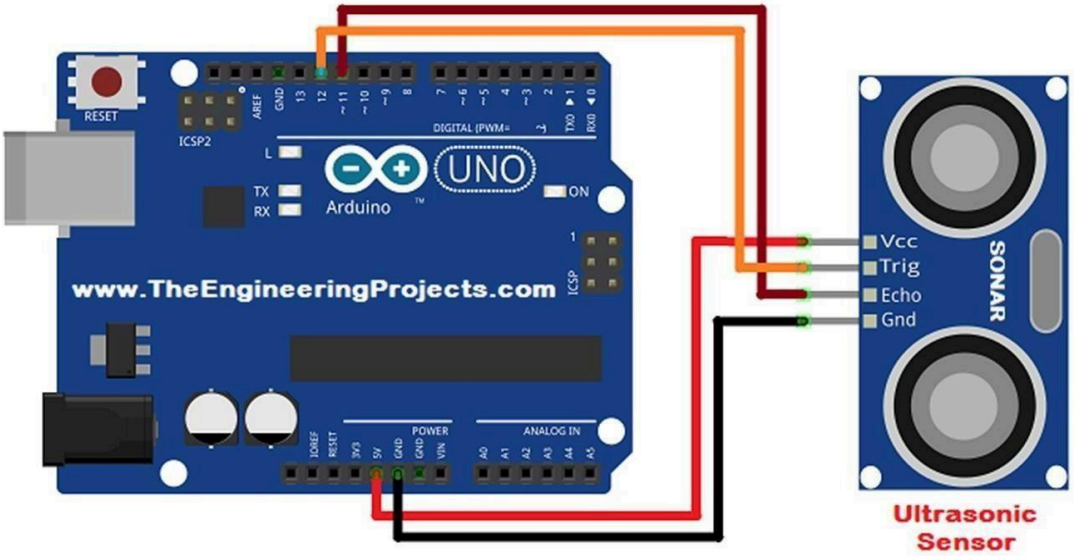


Fig.4 Interfacing of Ultrasonic Sensor

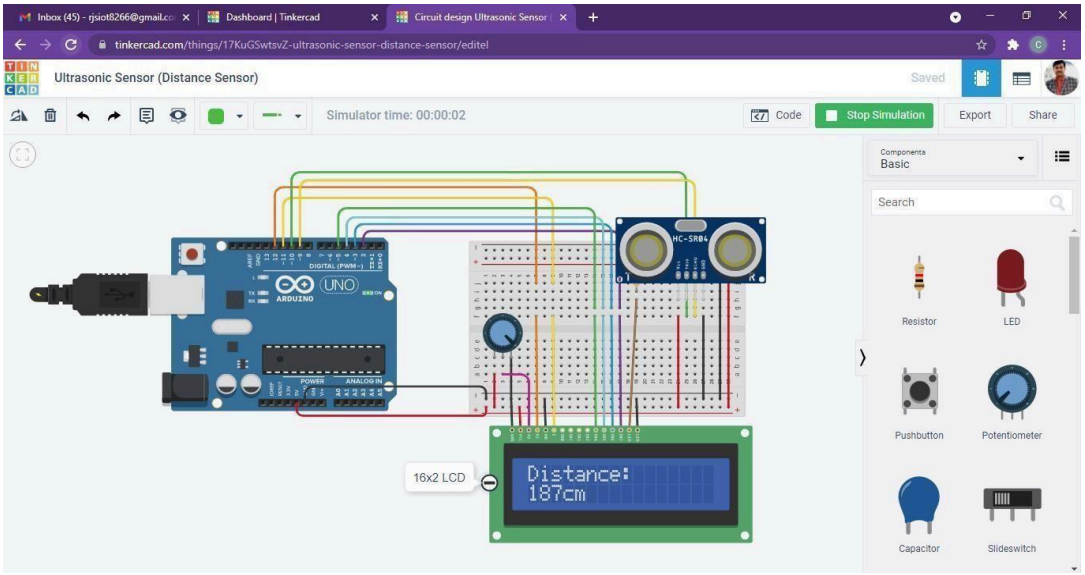


Fig.5 Simulated Result of Ultrasonic Sensor

Procedure:

- Make the connection as per circuit diagram
- Open Arduino IDE
- Select the Arduino Board
- Select the Arduino Port
- Write code in Editor Window and save file
- Compile the code
- Upload the code

Code:

Interfacing of Temperature Sensor (LM35):

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(7,6,5,4,3,2);
void setup()
{
          Serial.begin(19200);
           lcd.begin(16,2);
}
void loop()
{
          int analogvalue = analogRead(A3);
           int pecent = map(analogvalue,0,1023,0,600);
           Serial.print("analogvalue= ");
           Serial.println(analogvalue);
           lcd.setCursor(0,0);
           lcd.print(analogvalue);
           Serial.print("Pecent= ");
           Serial.println(pecent);
           lcd.setCursor(0,1);
           lcd.print(pecent);
           delay(1000);
           lcd.clear ();
}
```

Interfacing of IR Sensor:

```
int ledPin=13;
int inputPin=2;
int val=0;

void setup()
{
           pinMode(13,OUTPUT);
           pinMode( inputPin,
           INPUT);
```

```
  Serial.begin(9600);
}
void loop()
{
        val=digitalRead(inputPin); // check the pin status (High=1/Low=0) //Active Low
        output
        if(val==HIGH)
        {
         Serial.print(“Object Absent\n");
         digitalWrite(13,LOW);
         }
        else
        {
         Serial.print("Object
         Present\n");
         digitalWrite(13,HIGH);
         }
 }
```

Interfacing of Ultrasonic Sensor:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd
(12,11,5,4,3,2);
// defining the pins
const int trigPin =
10; const int echoPin
= 9;
// defining
variables long
duration;
int distance;
void setup() {
        pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
        pinMode(echoPin, INPUT); // Sets the echoPin as an Input
        Serial.begin(9600); // Starts the serial communication
        lcd.begin(16,2);
}
void loop() {// Clears the trigPin digitalWrite(trigPin, LOW);
```

```
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro
seconds digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin,
LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance= duration*0.034/2;
lcd.setCursor (0,0);
lcd.print("Distance: ");
delay(1000);
  lcd.setCursor
  (0,1);
  lcd.print(distance);
  lcd.print("cm");
}
```

Conclusion:

_____
_____
_____
_____

Design Experiments: -

1. **Temperature Alarm**
2. **Push Button and Traffic Lights**

Dr. D. Y. Patil Institute of Technology, Pimpri, Pune

Department of Electronics & Telecommunication

Engineering Academic Year 2024-2025 Semester VII

BE

Subject – MIOT

EXPERIMENT NO.- 4

Title : IoT based DC Motor Control with Arduino.

Aim: Interfacing of **DC motor to Arduino UNO** and control it's speed using **PWM** (Pulse Width Modulation)

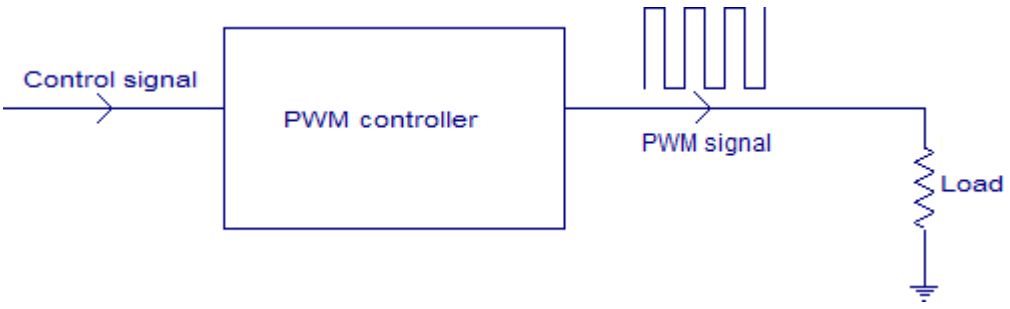
Hardware/ Software : Arduino IDE, Arduino board, DC Motor, L293,

Theory:

PWM control using Arduino

PWM control is a very commonly used method for controlling the power across loads. This method is very easy to implement and has high efficiency. PWM signal is essentially a high frequency squarewave (typically greater than 1KHz). The duty cycle of this square wave is varied in order to vary the power supplied to the load. Duty cycle is usually stated in percentage and it can be expressed using following equation: ***% Duty cycle = (TON/(TON + TOFF)) *100***. Where ***TON*** is the time for which the square wave is high and ***TOFF*** is the time for which the square wave is low.When duty cycle is increased the power dropped across the load increases and when duty cycle is reduced, power across the load decreases.

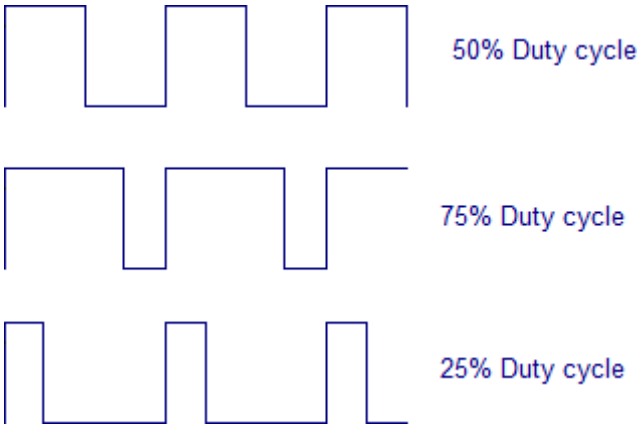
The block diagram of a typical PWM power controller scheme is shown below. Control signal is what we give to the PWM controller as the input. It might be an analog or digital signal according to the design of the PWM controller.



The control signal contains information on how much power hasto be applied to the load. The

PWM controller accepts the control signal and adjusts the duty cycle ofthe PWM signal according to the requirements.

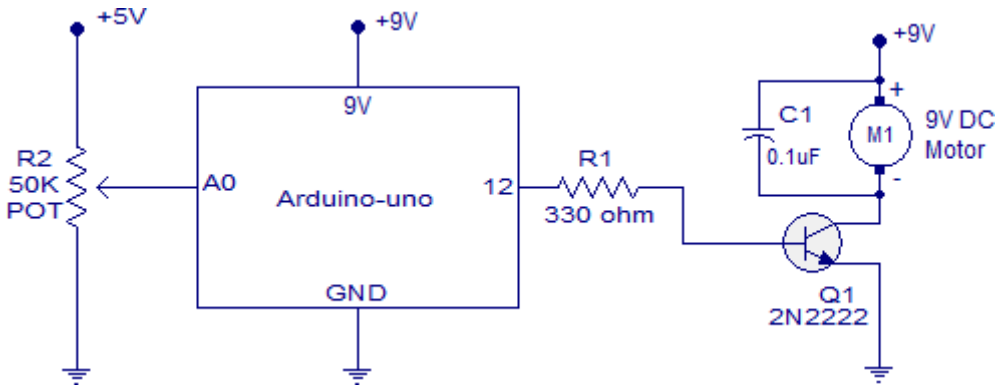
PWM waves with various duty cycle are shown in the figure below.



In the above wave forms you can see that the frequency is same but ON time and OFF time are different.

Motor speed control using arduino.

Circuit diagram of DC motor speed control using arduino is shown in the figure below. The working principle and program of this circuit is same as that of the LED brightness control. Only difference is that and additional motor driver circuit using a transistor is included in the circuit. Each digital pin of the arduino can sink or source only 40mA. DC motors usually consume much more than this and it isnot safe to directly connect a heavy load to the digital pin.



In the circuit diagram, slider of the potentiometer is connected to analog input pin A0 of arduino. Resistor R1 limits the base current of the transistor Q1. Motor is connected as collector load to the transistor. Capacitor C1 by-passes voltage spikes and noises produced by the motor. This filter capacitor is very essential and if it is not there the circuit may not work properly.

Procedure

1. Connect the Arduino through USB and upload the code
2. Open the serial monitor and set the baud rate at 9600
3. Now type any number from 0 to 9.

Code:

```
int pwm = 12; // assigns pin 12 to variable pwm
int pot = A0; // assigns analog input A0 to variable potint t1 = 0;

// declares variable t1
int t2 = 0;          // declares variable t2void setup()
// setup loop
{
   pinMode(pwm, OUTPUT); // declares pin 12 as
   outputpinMode(pot, INPUT); // declares pin A0 as input
}
void loop()
{
   t2= analogRead(pot); // reads the voltage at A0 and saves in t2
   t1= 1000-t2;                  // subtracts t2 from 1000 ans saves the result in t1digitalWrite(pwm,
   HIGH); // sets pin 12 HIGH
   delayMicroseconds(t1);          // waits for t1 uS (high time)
   digitalWrite(pwm, LOW); // sets pin 12 LOW
   delayMicroseconds(t2); // waits for t2 uS (low time)

}
```

Conclusion: