Advanced Embedded Systems Journal

M.Sc Part I Computer Science

Vedant Jadhav 524

March 21, 2023

R.J. College of Arts, Science & Commerce
Advanced Embedded Systems
Supervisor: Mrs. Vaidehi Deshpande
Seat number: 524

Contents

List of Figures

Practical No: 1

Aim: Switching ON and OFF LED using Push button with Arduino UNO.

Description:

1. Arduino :
   Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board and a piece of software, or IDE runs on your computer, used to write and upload computer code to the physical board. Arduino UNO has 14 digital pins and 6 analog pins.
2. Breadboard :
   It is a way of constructing electronics without having to use a soldering iron. Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.
3. Pushbutton :
   The pushbutton is a component that connects two points in a circuit when you press it.
4. LED :
   A light-emitting diode (LED) is a semiconductor device that produces light from electricity. LEDs last a long time and do not break easily.

Hardware Requirement:

1. 1× Arduino
2. 1× Breadboard
3. 1× LED
4. 2× Resistors
5. 1× Push Button
6. Jump Wires

Hardware Configuration:

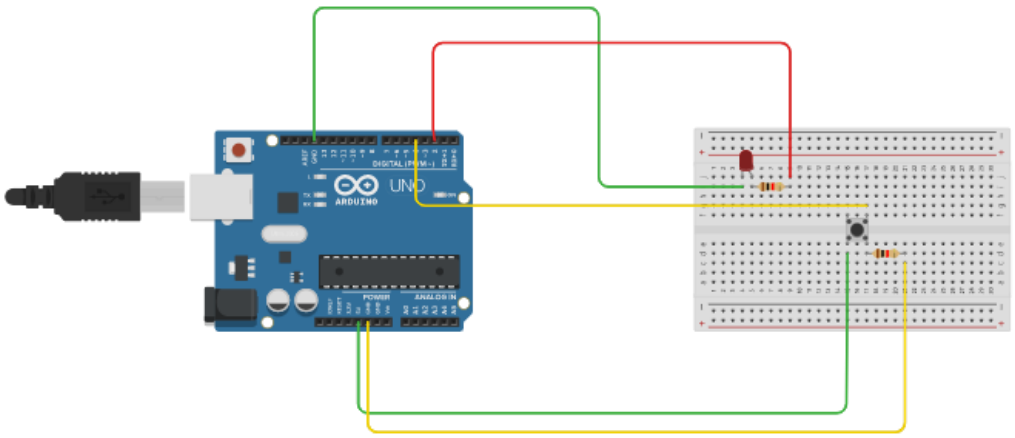


Figure 1: Hardware configuration Switching on and off the LED using a push button with an Arduino Uno

Code:

```
// C++ code
//
const int ledpin = 2;
const int btnpin = 4;
int btnstate = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(ledpin, OUTPUT);
  pinMode(btnpin, INPUT);
}

void loop() {
  btnstate = digitalRead(btnpin);
  if (btnstate == HIGH) {
    digitalWrite(ledpin, HIGH);
```

```
    Serial.println("LED ON");
  } else {
    digitalWrite(ledpin, LOW);
    Serial.println("LED OFF");
  }
}
```

Output :

While pressing the push-button the LED will glow (i.e. ON) and when we release it LED will turn OFF.

Practical No: 2

Aim: Using seven-segment display with Arduino UNO to display numbers from 0 to 9 after specific intervals of time.

Description:

1. Breadboard :
   It is a way of constructing electronics without having to use a soldering iron. Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.

2. Seven-segment :
   The seven-segment display has seven LEDs arranged in the shape of number eight.

3. Resistors :
   It's a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses

Hardware Requirement:

1. 1× Arduino
2. 1× Breadboard
3. 1× Seven Segment
4. 7× Resistors
5. Jump Wires

Hardware Configuration:

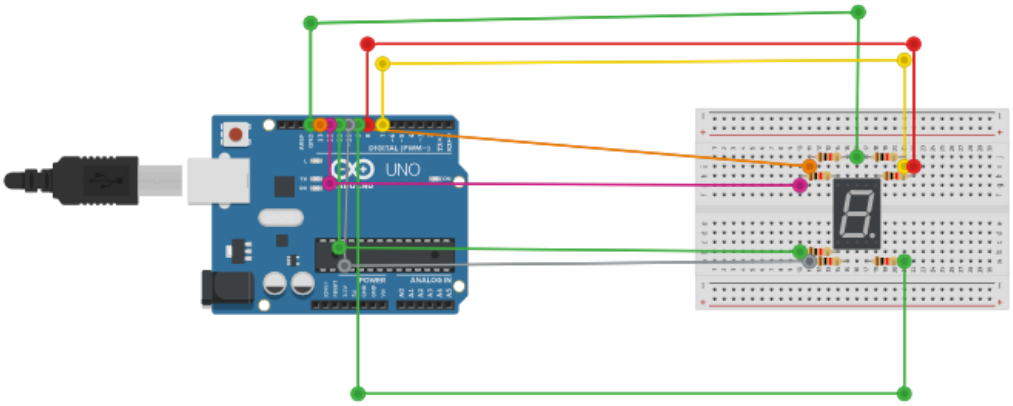


Figure 2: Using seven-segment display with Arduino UNO to display numbers from 0 to 9 after specific intervals of time.

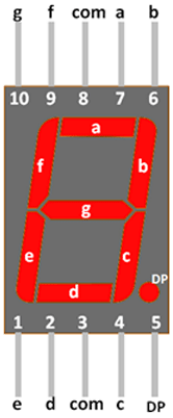


Figure 3: Seven-segment display.

Code:

```
int f = 13;
int g = 12;
int e = 11;
int d = 10;
int c = 9;
int b = 8;
int a = 7;
int de = 1000;
int count = 0x00;
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(a, 1);
  digitalWrite(b, 1);
  digitalWrite(c, 1);
  digitalWrite(d, 1);
  digitalWrite(e, 1);
  digitalWrite(f, 1);
  digitalWrite(g, 0);
  delay(de); //0

  digitalWrite(a, 0);
  digitalWrite(b, 1);
  digitalWrite(c, 1);
  digitalWrite(d, 0);
  digitalWrite(e, 0);
  digitalWrite(f, 0);
  digitalWrite(g, 0);
  delay(de); //1
  digitalWrite(a, 1);
```

```
  digitalWrite(b, 1);
  digitalWrite(c, 0);
  digitalWrite(d, 1);
  digitalWrite(e, 1);
  digitalWrite(f, 0);
  digitalWrite(g, 1);
  delay(de); //2
  digitalWrite(a, 1);
  digitalWrite(b, 1);
  digitalWrite(c, 1);
  digitalWrite(d, 1);
  digitalWrite(e, 0);
  digitalWrite(f, 0);
  digitalWrite(g, 1);
  delay(de); //3
  digitalWrite(a, 0);
  digitalWrite(b, 1);
  digitalWrite(c, 1);

  digitalWrite(d, 0);
  digitalWrite(e, 0);
  digitalWrite(f, 1);
  digitalWrite(g, 1);
  delay(de); //4
  digitalWrite(a, 1);
  digitalWrite(b, 0);
  digitalWrite(c, 1);
  digitalWrite(d, 1);
  digitalWrite(e, 0);
  digitalWrite(f, 1);
  digitalWrite(g, 1);
  delay(de); //5
  digitalWrite(a, 1);
  digitalWrite(b, 0);
  digitalWrite(c, 1);
  digitalWrite(d, 1);
  digitalWrite(e, 1);
  digitalWrite(f, 1);
  digitalWrite(g, 1);
  delay(de); //6
  digitalWrite(a, 1);
```

```
  digitalWrite(b, 1);
  digitalWrite(c, 1);
  digitalWrite(d, 0);
  digitalWrite(e, 0);
  digitalWrite(f, 0);

  digitalWrite(g, 0);
  delay(de); //7
  digitalWrite(a, 1);
  digitalWrite(b, 1);
  digitalWrite(c, 1);
  digitalWrite(d, 1);
  digitalWrite(e, 1);
  digitalWrite(f, 1);
  digitalWrite(g, 1);
  delay(de); //8
  digitalWrite(a, 1);
  digitalWrite(b, 1);
  digitalWrite(c, 1);
  digitalWrite(d, 0);
  digitalWrite(e, 0);
  digitalWrite(f, 1);
  digitalWrite(g, 1);
  delay(de); //9
}
```

Output :

The seven-segment will display the numbers starting from 0 to 9 after every 1000 ms as the delay de is declared as 1000.

Practical No: 3

Aim: Configuring a digital object counter device using 7-segment display with Arduino UNO and IR proximity sensor.

Description:

1. Breadboard :
   It is a way of constructing electronics without having to use a soldering iron. Components are pushed into the sockets on the breadboard and then extra 'jumper' wires are used to make connections.

2. Seven-segment :
   The seven-segment display has seven LEDs arranged in the shape of number eight.

3. Infrared Sensor :
   Infrared (IR) communication is a widely used and easy to implement wireless technology that has many useful applications. The most prominent examples in day to day life are TV/video remote controls, motion sensors, and infrared thermometers.

Hardware Requirement:

1. 1× Arduino
2. 1× Breadboard
3. 1× Seven Segment
4. 7× Resistors
5. 1× IR (Infrared sensor)
6. Jump Wires

Hardware Configuration:

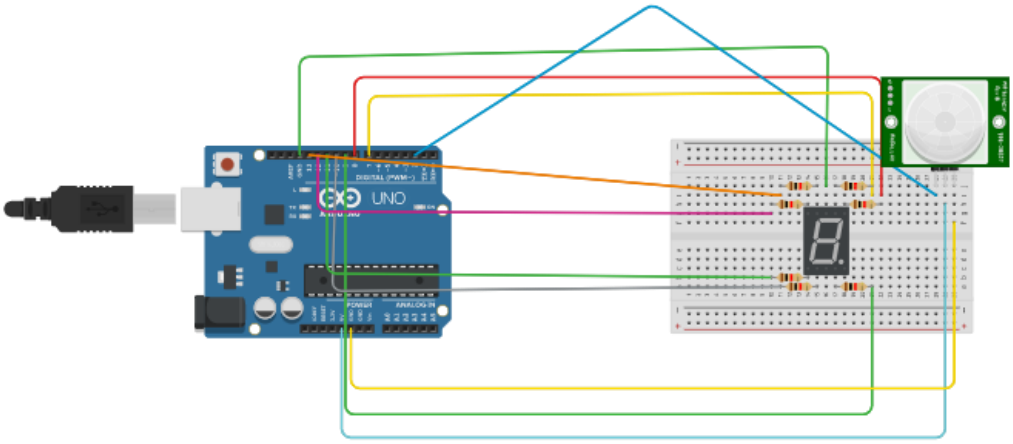


Figure 4: Configuring a digital object counter device using 7-segment display with Arduino UNO and IR proximity sensor.

Code:

```
int f = 13;
int g = 12;
int e = 11;
int d = 10;
int c = 9;
int b = 8;
int a = 7;
int buttonpin = 5; //no of push button pin
int buttonstate = 0; //variable for reading pushbttoun status
int p = 0;
int de = 1000;
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
```

```
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(buttonpin, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  buttonstate = digitalRead(buttonpin);
  if (buttonstate == HIGH) {
    p++;

    //delay(100);
  }
  if (p == 0) {
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 1);
    digitalWrite(e, 1);
    digitalWrite(f, 1);
    digitalWrite(g, 0);
    delay(de); //0
  }
  if (p == 1) {
    digitalWrite(a, 0);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 0);
    digitalWrite(e, 0);
    digitalWrite(f, 0);
    digitalWrite(g, 0);
    delay(de); //1
  }
  if (p == 2) {
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 0);
    digitalWrite(d, 1);

```

```
    digitalWrite(e, 1);
    digitalWrite(f, 0);
    digitalWrite(g, 1);
    delay(de); //2
  }
  if (p == 3) {
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 1);
    digitalWrite(e, 0);
    digitalWrite(f, 0);
    digitalWrite(g, 1);
    delay(de); //3
  }
  if (p == 4) {
    digitalWrite(a, 0);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 0);
    digitalWrite(e, 0);
    digitalWrite(f, 1);
    digitalWrite(g, 1);
    delay(de); //4
  }
  if (p == 5) {
    digitalWrite(a, 1);

    digitalWrite(b, 0);
    digitalWrite(c, 1);
    digitalWrite(d, 1);
    digitalWrite(e, 0);
    digitalWrite(f, 1);
    digitalWrite(g, 1);
    delay(de); //5
  }
  if (p == 6) {
    digitalWrite(a, 0);
    digitalWrite(b, 0);
    digitalWrite(c, 1);
    digitalWrite(d, 1);
```

```
    digitalWrite(e, 1);
    digitalWrite(f, 1);
    digitalWrite(g, 1);
    delay(de); //6
  }
  if (p == 7) {
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 0);
    digitalWrite(e, 0);
    digitalWrite(f, 0);
    digitalWrite(g, 0);
    delay(de); //7
  }

  if (p == 8) {
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 1);
    digitalWrite(e, 1);
    digitalWrite(f, 1);
    digitalWrite(g, 1);
    delay(de); //8
  }
  if (p == 9) {
    digitalWrite(a, 1);
    digitalWrite(b, 1);
    digitalWrite(c, 1);
    digitalWrite(d, 0);
    digitalWrite(e, 0);
    digitalWrite(f, 1);
    digitalWrite(g, 1);
    delay(de); //9
  }
}
```

Output :

Initially 7-segment will display 0 when the power is on. When some obstacle will be in front of IR, it will pass signal to arduino UNO. Code will then increment the counter and display it on 7 segment display. For the next obstacle, next number will be displayed and so on up to 9.

Practical No: 4

Aim: Print message on LCD display with Arduino UNO.

Description:

1. LCD :
   A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizer's. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in colour or monochrome. It is 16*2 LCD display. That is it has 16 columns and 2 rows.

Hardware Requirement:

1. 1× Arduino
2. 1× Breadboard
3. 1× LCD
4. 2× Resistors
5. Jump Wires

Hardware Configuration:

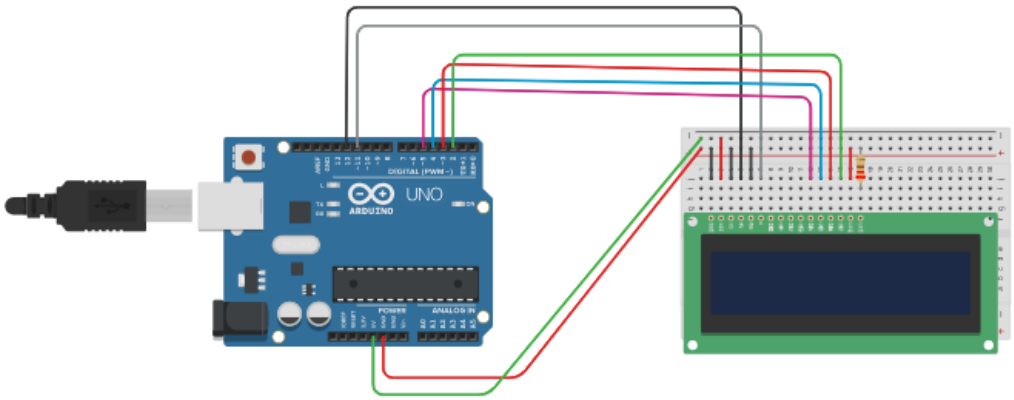


Figure 5: Print message on LCD display with Arduino UNO.

Code:

```
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12,
  en = 11,
  d4 = 5,
  d5 = 4,
  d6 = 3,
  d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
```

```
}
void loop() {
  lcd.setCursor(13, 0);
  lcd.print("OK");
  lcd.setCursor(5, 1);
  for (int thisChar = 0; thisChar < 10; thisChar++) {
    lcd.print(thisChar);
    delay(500);
  }
}
```

Output :

LCD screen display the message “Hello world”.

Practical No: 5

Aim: : Use 4 × 4 keypad to give the input in Arduino UNO serial monitor.

Description:

1. Keypad :

   The buttons on a keypad are arranged in rows and columns. A 3 × 4 keypad has 4 rows and 3 columns, and a 4 × 4 keypad has 4 rows and 4 columns.

   Keypad 4 × 4 is used for loading numerics into the microcontroller. It consists of 16 buttons arranged in a form of an array containing four lines and four columns. It is connected to the development system by regular IDC 10 female connector plugged in some development system's port.

Hardware Requirement:

1. 1× Arduino
2. 1× Keypad
3. Jump Wires

Hardware Configuration:

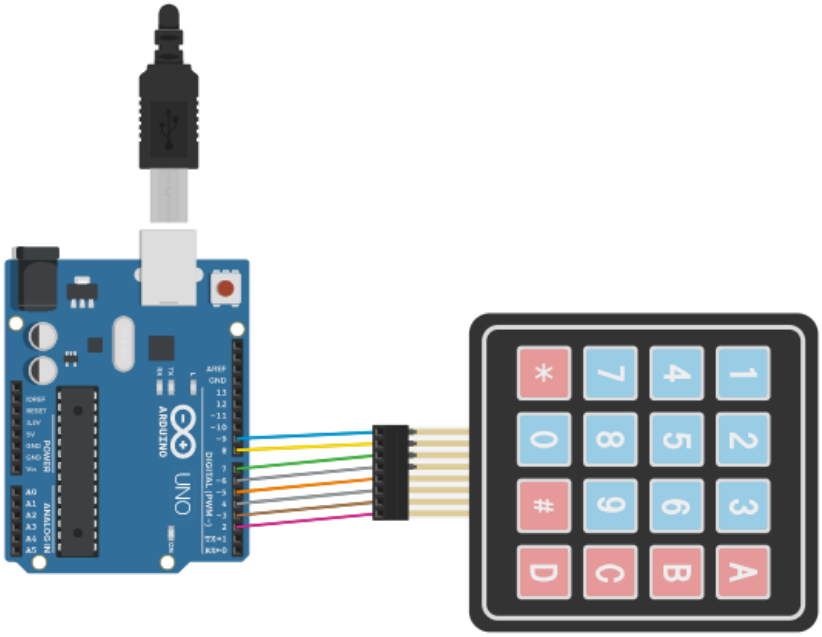


Figure 6: Use 4 × 4 keypad to give the input in Arduino UNO serial monitor

Code:

```
#include <Key.h>
#include  <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS]={
  {"1","2","3","A"},
  {"4","5","6","B"},
  {"7","8","9","C"},
  {"*","0","#","D"}
};

byte colPins[ROWS]={5,4,3,2}; //Connect to the row pinouts of keypad
byte rowPins[COLS]={9,8,7,6};//Connect to the row pinouts of keypad

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
  char key = keypad.getKey();
  if (key) {
    Serial.println(key);
  }
}
```

Output :

After you upload the code, open the serial monitor. When you press a key, the value will be printed out on serial monitor.

Practical No: 6

Aim:Interfacing of buzzer with arduino UNO.

Description:

1. Buzzer:
   A **buzzer** or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of **buzzers** and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

Hardware Requirement:

1. 1× Arduino
2. 1× Buzzer
3. Jump Wires

Hardware Configuration:

⇒ Buzzer (+ve) connect with Arduino (Pin 11)
⇒ Buzzer (-ve) connect with Arduino (GND)

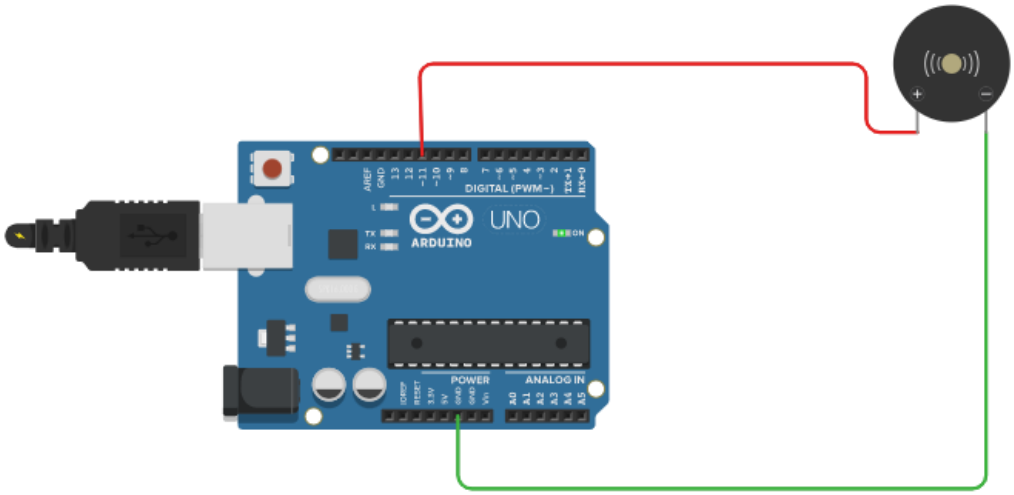


Figure 7: Interfacing of buzzer with arduino UNO.

Code 1:

```
int buzzer = 11;
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
  int i = 0;
  do {
    i++;
    tone(buzzer, 450);
    delay(200);
    noTone(buzzer);
    delay(200);
  } while (i < 3);
  delay(3000);
}
```

Code 2:

```
int buzzer = 11;

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
  tone(buzzer, 450);
  delay(500);
  noTone(buzzer);
  delay(500);
}
```

Output :

With these two different codes we will get different beep sounds. With various code manipulations we can have different beep variations.

Practical No: 7

Aim: Interfacing of ultrasonic sensor with arduino UNO.

Description:

1. Ultrasonic:
   Ultrasonic distance sensor determines the distance to an object by measuring the time taken by the sound to reflect back from that object. A typical ultrasonic distance sensor consists of two membranes. One membrane produces sound, another catches reflected echo. Basically they are speaker and micro-phone.

Hardware Requirement:

1. 1× Arduino
2. 1× Buzzer
3. Ultrasonic sensor
4. Breadboard
5. LED
6. Jump Wires

Hardware Configuration:

⇒ Ultrasonic (GND) connect with Arduino (GND).
⇒ Ultrasonic (Trig) connect with Arduino (Pin 9).
⇒ Ultrasonic (Echo) connect with Arduino (Pin 10).
⇒ Ultrasonic (VCC) connect with Arduino (+5v).
⇒ LED (-ve) connect with Arduino (GND).
⇒ LED (+ve) connect with Arduino (Pin 7).

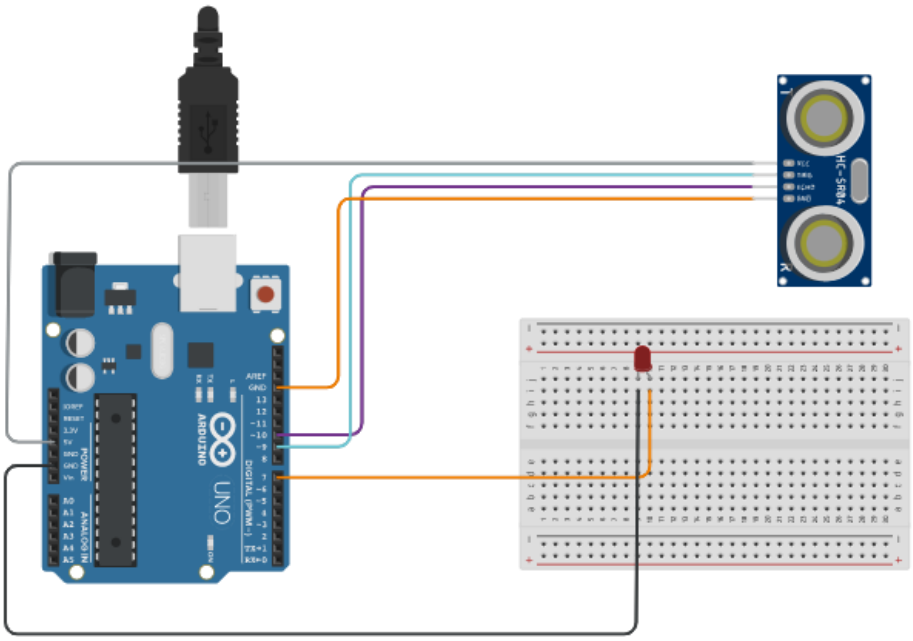


Figure 8: Interfacing of ultrasonic sensor with arduino UNO.

Code:

```
int trigpin = 9;
int echopin = 10;
int led = 7;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(led, OUTPUT);
  pinMode(trigpin, OUTPUT);
  pinMode(echopin, INPUT);
}
void loop() {
  long duration, distance;
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(1000);
  digitalWrite(trigpin, LOW);
  duration = pulseIn(echopin, HIGH);
  distance = (duration / 2) / 29.1;
  Serial.print(distance);
```

```
  Serial.println("CM");
  delay(10);
  if ((distance <= 10)) {
    digitalWrite(led, HIGH);
  } else if (distance > 10) {
    digitalWrite(led, LOW);
  }
}
```

Output :

LED glows if the obstacle is detected by ultrasonic sensor at a particular distance

Practical No: 8

Aim: Interfacing of Servo Motor with arduino UNO.

Description:

1. Servo Motor:
   A servo motor is an electrical device which can push or rotate an object with great precision. If you want to rotate and object at some specific angles or distance, then you use servo motor.

Hardware Requirement:

1. 1× Arduino
2. Servo Motor
3. Jump Wires

Hardware Configuration:

⇒ Servo Motor (Brown) connect with Arduino (GND).
⇒ Servo Motor (Red) connect with Arduino (+5v).
⇒ Servo Motor (Orange) connect with Arduino (Pin 9).

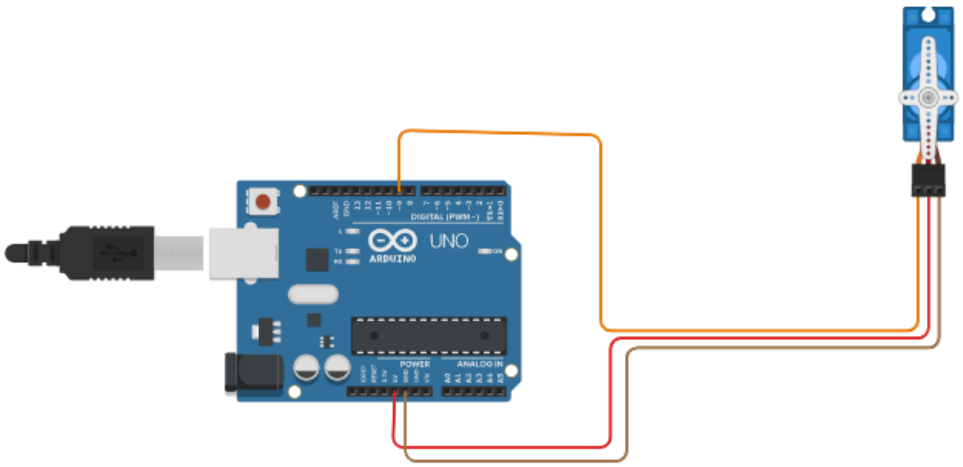


Figure 9: Interfacing of Servo Motor with arduino UNO.

Code:

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards
int pos = 0; // variable to store the servo position
void setup() {
  myservo.attach(9); // attaches the servo on pin 9 to the servo object }

}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos); // tell servo to go to position in variable "pos"
    delay(15); // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos); // tell servo to go to position in variable "pos"
    delay(15); // waits 15ms for the servo to reach the position
  }
}
```

Output :

When the program is loaded in arduino UNO, Servo motor starts rotating.

Practical No: 9

Aim: Interfacing of DHT11 with arduino UNO to read temperature and humidity which is then printed on serial monitor.

Description:

1. **DHT11**: DHT11 is allow cost digital sensor for sensing temperature and humidity.This can be easily interfaced with any microcontroller like arduino, raspberry Pi etc to measure humidity and temperature instantaneously. This sensor is used various applications such as measuring humidity and temperature values in heating, ventilation and AC systems. Offices, cars, green houses use this sensor for measuring humidity values and safety measure. This can be used for smart gardening.

Hardware Requirement:

1. 1 × *Arduino*
2. DHT11 sensor
3. 1 × *Breadboard*
4. Resistor
5. Jump Wires

Hardware Configuration:

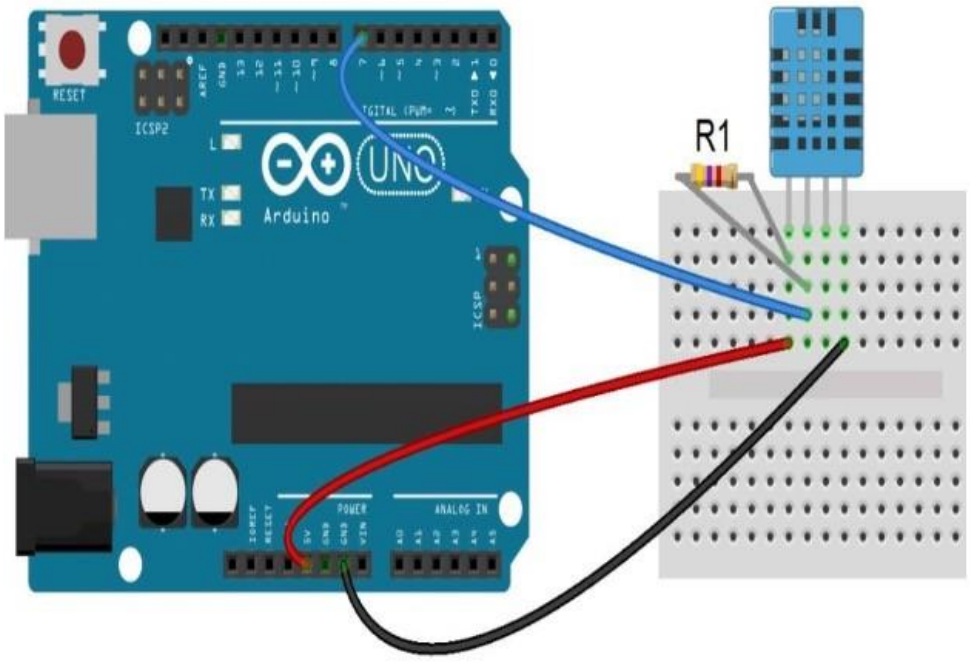


Figure 10: Interfacing of DHT11 with arduino UNO to read temperature and humidity which is then printed on serial monitor.

Code:

Prerequisite for code"

- Download the DHTLib from https://www.circuitbasics.com/wp-content/uploads/2015/10/DHTLib.zip
- Open Ardiuno and include the DHTLib.zip as In arduino IDE ⇒ Sketch ⇒ Include Library ⇒ Manage Libraries (Library manager will open) ⇒ Search for DH11 version 1.0.6 ⇒ Install “Simple dh"

```
#include <dht.h>

dht DHT;
#define DHT11_PIN 7 void setup() {
  Serial.begin(9600);
}
void loop() {
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
```

```
12    Serial.println(DHT.humidity);
13    delay(1000);
14  }
15  }
```

Output :

When the program is loaded, arduino will get the input from DHT11 and the values of temperature and humidity are displayed on serial monitor.

Practical No: 10

Aim: Interfacing of LED with NodeMCU and controlling it remotely with Blynk application on mobile.

Description:

1. **NodeMCU**: NodeMCU is low cost open source IOT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC. Arduino UNO does not have inbuilt Wi-Fi module. It provides access to the GPIO. It has 10 digital pins and only 1 analog pin. It can also be programmed directly using arduino IDE. It consumes ten times of power than arduino UNO.
2. **Blynk App**: Blynk is a platform with IOS and android apps to control Arduino, Raspberry Pi and so on. It's a digital dashboard where we can build a graphic interface for our project by simply dragging and dropping widgets. (Blynk is one of the app to control data remotely. There are other options to upload IOT data on cloud or handling IOT devices remotely through cloud such as ThingSpeak)

Hardware Requirement:

1. NodeMCU
2. Breadboard
3. LED
4. Resistor
5. Jump Wires

Hardware Configuration:

⇒ Led +ve(big leg) connect with D5 port
⇒ Led –ve(small leg) connect with GND

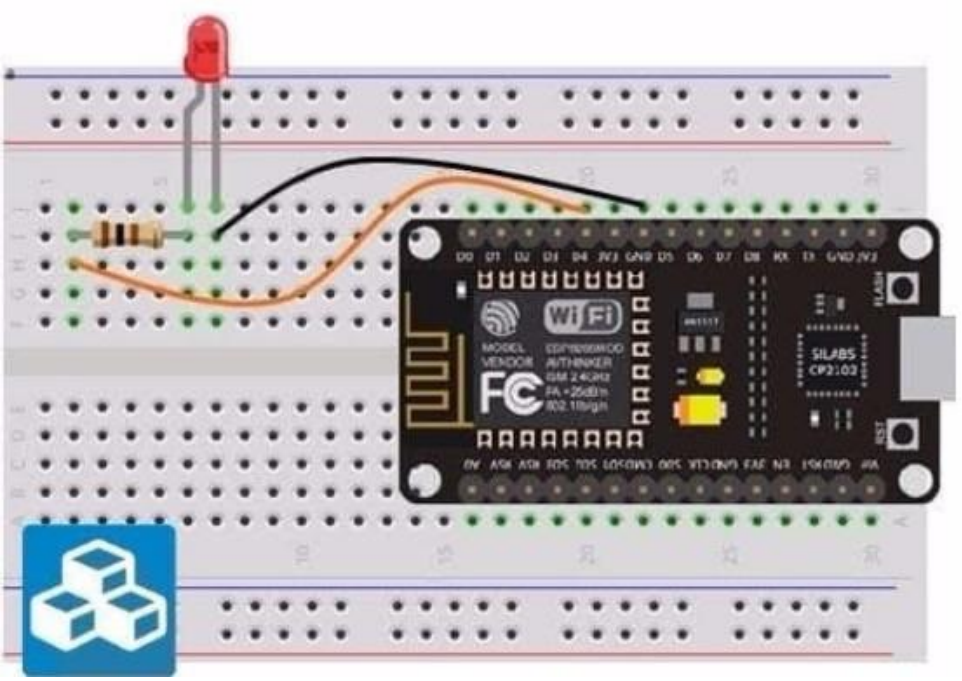
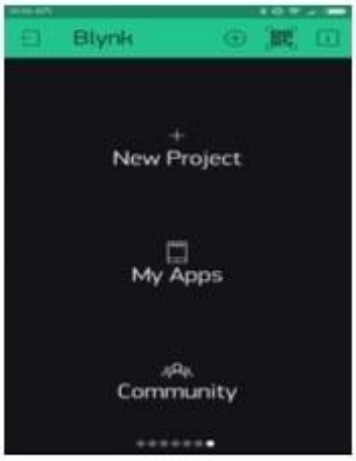


Figure 11: Interfacing of LED with NodeMCU and controlling it remotely with Blynk application on mobile.

Blynk Configuration:

To control LED connected to NodeMCU remotely we need to install Blynk app in our mobile as follows:

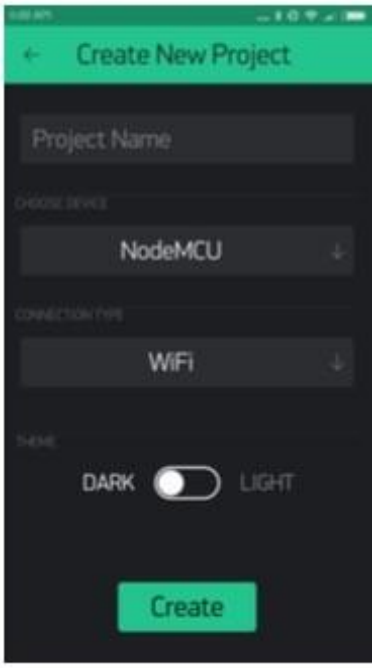
1. Download Blynk app from play store



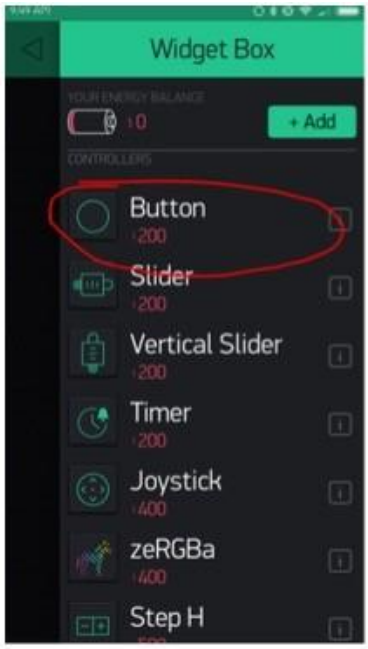
2. After downloading the app, create an account and log in. Gmail ID and password can be used

3. Click the “Create New Project” in the app to create a new Blynk app. Give any project name and select hardware as NodeMCU. Select connection type as Wi-Fi.

4. Click on Continue and then Click on Create.
5. AuthToken will be send to your registered email id
6. Add Widgets To The Project and select the Button



7. Now Configure that Button by click on it



(a) Give name to that button widget

(b) In Output area make the Pin as Digital and select any one Port number between

(c) Mode will be move to Switch side

Uploading Firmware:

1. Download following zip file:

   https : //github.com/blynkkk/blynk – library/releases/download/v0.6.1/Blynk_Release_v0.6.1.zip

2. Extract this file ⇒ we can see 2 folder tools and libraries
3. The content of tools will be copied and pasted on `C:Files (x86) Arduino`
4. The content of libraries will be copied and pasted on `C : Program Files (x86) Arduino libraries`

Arduino IDE Setup:

Open arduino IDE and perform following configuration –

1. File ⇒ Preferences ⇒ In Additional Boards Manager text box enter:

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

   ⇒ OK
2. Tools ⇒ Boards ⇒ Board Manager ⇒ Search for esp8266 by ESP8266 community 2.6.3 ⇒ Install
3. Tool ⇒ Board ⇒ Select NodeMCU
4. Tools ⇒ Select COM port for communication

Code:

Prerequisite for code

:

- Download the DHTLib from https://www.circuitbasics.com/wp-content/uploads/2015/10/DHTLib.zip
- Open Ardiuno and include the DHTLib.zip as In arduino IDE ⇒ *Sketch* ⇒ *IncludeLibrary* ⇒ *ManageLibraries*(*Libraryma SearchforDH*11*version*1.0.6⇒ *Install*"*Simpledh*"

```
#define BLYNK_PRINT Serial#include <ESP8266WiFi.h>

#include  <BlynkSimpleEsp8266_SSL.h>
 // You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon)  Auth Tokens  Copy all char auth[] = "YourAuthToken";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
```

```
char pass[] = "YourPassword";
void setup() {
  // Debug console Serial.begin(9600); Blynk.begin(auth, ssid, pass);
}
void loop() {
  Blynk.run();
}
```

Output :

Click the button from Blynk app to switch ON and OFF the LED. We can test from remotely operating.