ne of the Student: Jay [illegible]	Roll No. E7BA25	Date: 30/7/25

Experiment No. - 2

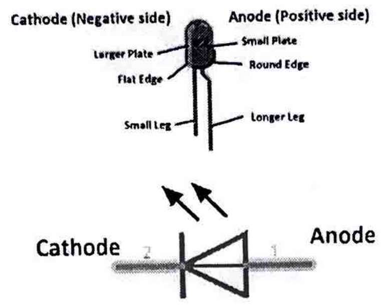
Title: Arduino based Simple program digital read/write using LED and Switch.

Hardware Requirements: Arduino Board, LED, Push Button

Software Requirements: Arduino IDE

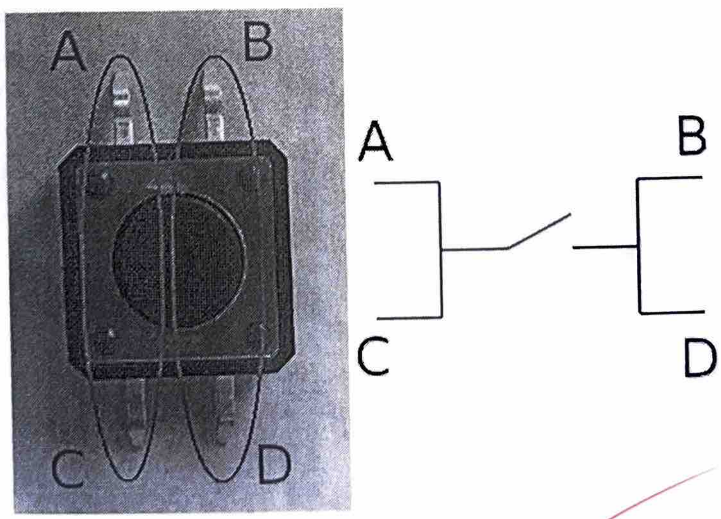
Theory:

LED:

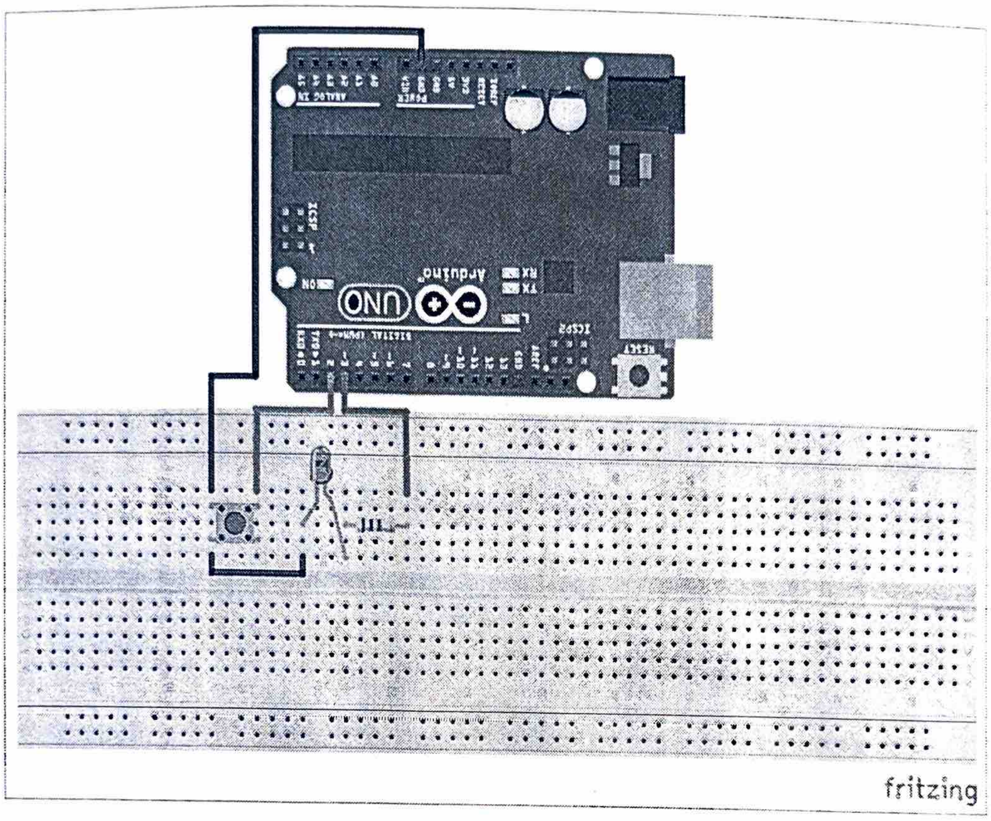


A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the 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.



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

Conclusion:

In this expriment we have studied Ardino based simple program digital read/write using LED and Switch.

Name of the Student:	Roll No.	Date:
Jay More	12[illegible]3A45	6/8/25

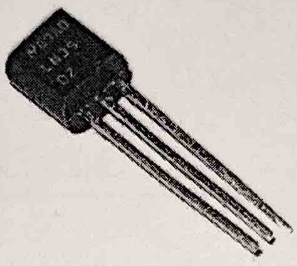
Experiment No. – 3(a)

Title: Interfacing of LM35 sensor using Arduino

Hardware Requirements: Arduino Board, LM 35 Temp. Sensor

Software Requirements: Arduino IDE

Theory:



LM35 is a temperature measuring device having an analog output voltage proportional to the temperature.

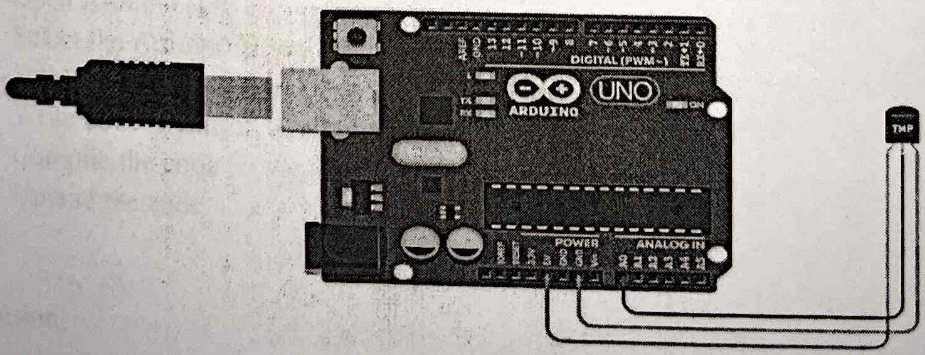
It provides output voltage in Centigrade (Celsius). It does not require any external calibration circuitry.

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



CODE:

```
// LM35 is connected to this PIN

#define sensorPin A0
void setup()
{
 // Init serial at 9600 baud
 Serial.begin(9600);
}

void loop()
{
 //Read Raw ADC Data
 int adcData = analogRead(sensorPin);

 // Convert that ADC Data into voltage
 float voltage = adcData * (5.0 / 1024.0);

 // Convert the voltage into temperature
 float temperature = voltage * 100;

 // Print the temperature data
 Serial.print("Temperature: ");
 Serial.print(temperature);
 Serial.println("*C");
 delay(800); // wait a second between readings
}
```

Name of the Student: Jay J-Mom	Roll No. T2T13Ans	Date: 13/8/25

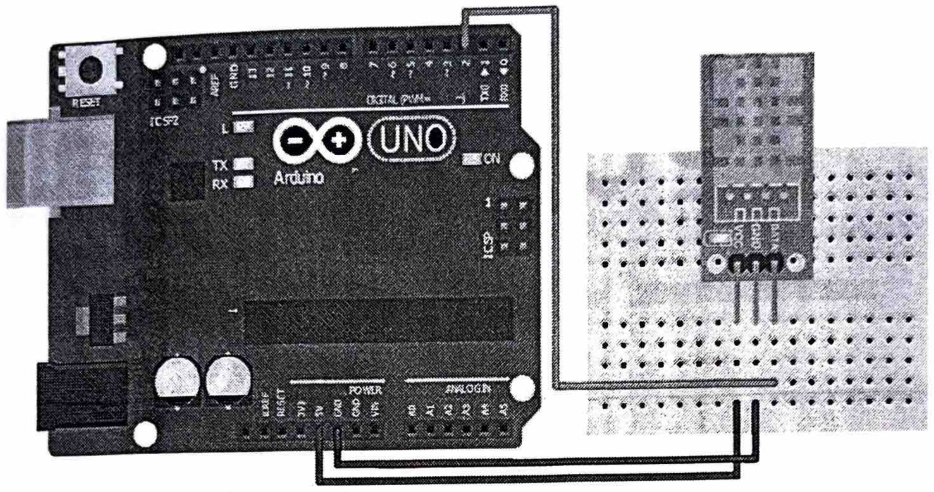
Experiment No. – 3(b)

Title: Interfacing of DHT11 for Temperature and Humidity using Arduino Uno

Hardware Requirements: Arduino Board, DHT11 temperature & humidity sensor.

Software Requirements: Arduino IDE

Theory:



The **DHT11** is a commonly used **Temperature and humidity sensor.** The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%.

DHT11 Specifications

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

Code:

```
#include "DHT.h"
#define DHTPIN 7     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
 Serial.begin(9600);
 Serial.println(F("DHTxx test!"));

 dht.begin();
}

void loop() {
 delay(2000);
 float h = dht.readHumidity();
 float t = dht.readTemperature();
 float f = dht.readTemperature(true);
 if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println(F("Failed to read from DHT sensor!"));
  return;
 }
 float hif = dht.computeHeatIndex(f, h);
 float hic = dht.computeHeatIndex(t, h, false);

 Serial.print(F(" Humidity: "));
 Serial.print(h);
 Serial.print(F("%  Temperature: "));
 Serial.print(t);
 Serial.print(F("C "));
 Serial.print(f);
 Serial.print(F("F  Heat index: "));
 Serial.print(hic);
 Serial.print(F("C "));
 Serial.print(hif);
 Serial.println(F("F"));
}
```

Name of the Student: Jay More	Roll No. ETB AL3	Date: 3/04/25

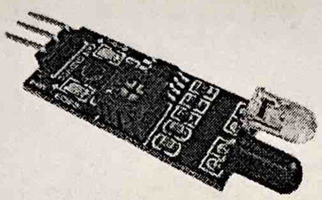
Experiment No. – 5(a)

Title: Interfacing of IR sensor for object detection using Arduino

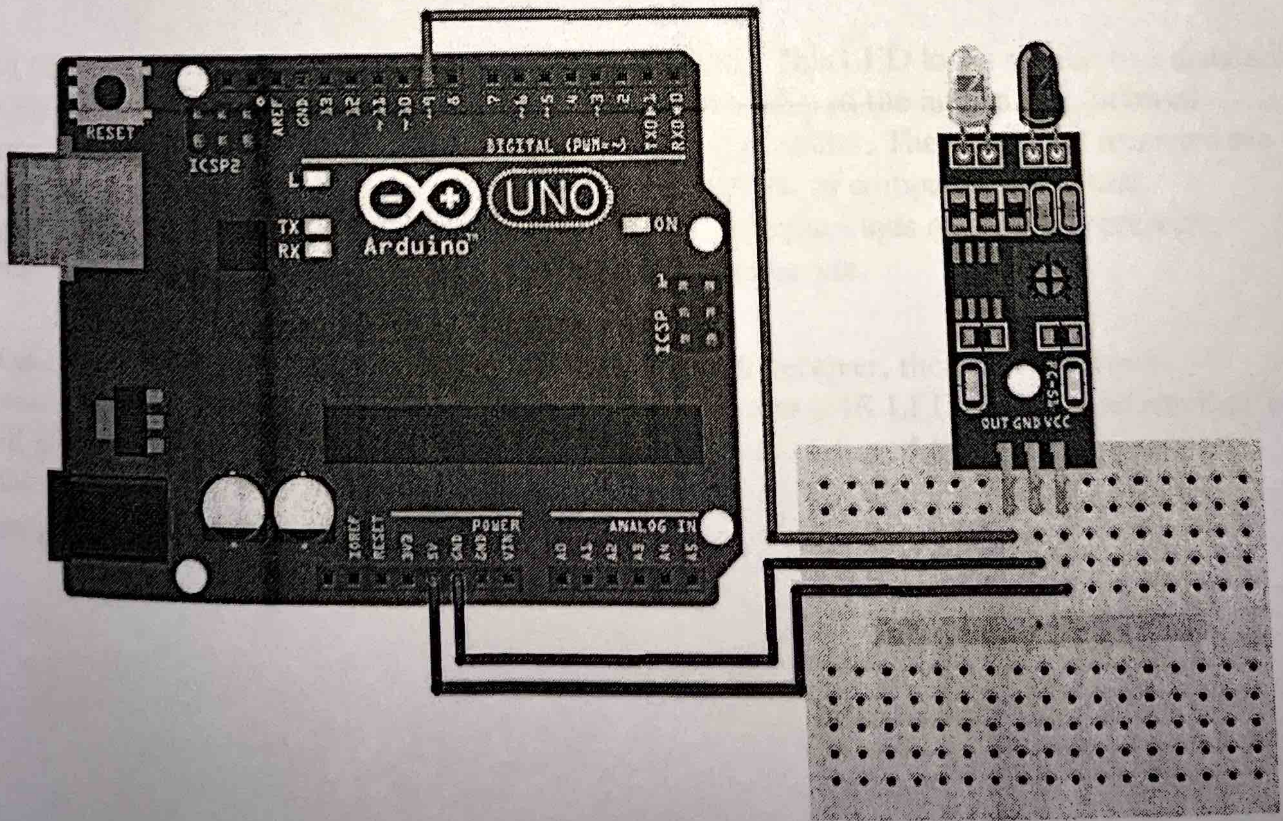
Hardware Requirements: Arduino Board, IR Sensor.

Software Requirements: Arduino IDE

Theory:



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.



Code:

```
// Arduino IR Sensor Code

int IRSensor = 9; // connect ir sensor module to Arduino pin 9
int LED = 13; // conect LED to Arduino pin 13

void setup()
{
Serial.begin(115200); // Init Serila at 115200 Baud
Serial.println("Serial Working"); // Test to check if serial is working or not
pinMode(IRSensor, INPUT); // IR Sensor pin INPUT
pinMode(LED, OUTPUT); // LED Pin Output
}

void loop()
{
int sensorStatus = digitalRead(IRSensor); // Set the GPIO as Input
if (sensorStatus == 1) // Check if the pin high or not
{
// if the pin is high turn off the onboard Led
digitalWrite(LED, LOW); // LED LOW
Serial.println("Motion Ended!"); // print Motion Detected! on the serial monitor window
}
else
{
//else turn on the onboard LED
digitalWrite(LED, HIGH); // LED High
Serial.println("Motion Detected!"); // print Motion Ended! on the serial monitor window
}
}
```

Name of the Student: Jay More	Roll No. IETB[illegible]	Date: 3/09/25

Experiment No. – 5(b)

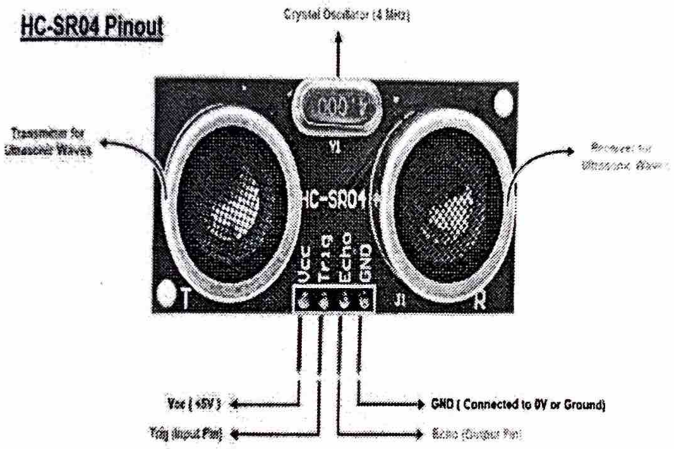
Title: Interfacing of Ultrasonic Sensor using Arduino Uno

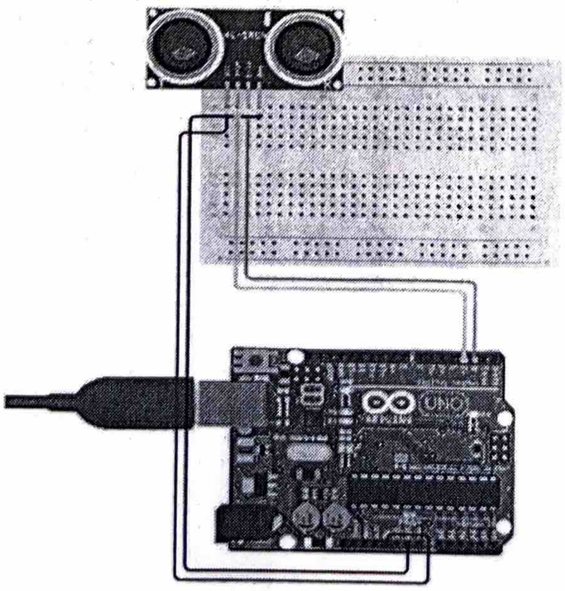
Hardware Requirements: Arduino Board, Ultrasonic Sensor

Software Requirements: Arduino IDE

Theory:

Ultrasonic Sensor:


HC-SR04 Pinout
Crystal Oscillator (4 MHz)
Transmitter for Ultrasonic Waves
Receiver for Ultrasonic Waves
HC-SR04
Y1
Vcc
Trig
Echo
GND
T
J1
R
Vcc (+5V)
Trig (Input Pin)
GND (Connected to 0V or Ground)
Echo (Output Pin)



Code:

```
//#include <HCSR04.h>

const int TriggerPin = 8; //Trig pin
const int Echopin = 9; //Echo pin
long Duration = 0;

void setup()
{
// Trigger is set to output
pinMode(TriggerPin, OUTPUT);

// Echo is set as input
pinMode(Echopin,INPUT);
Serial.begin(9600); // Serial Output
}

void loop() {
digitalWrite(TriggerPin, LOW);
delayMicroseconds(2);

// Trigger pin to HIGH
digitalWrite(TriggerPin, HIGH);
delayMicroseconds(10); // 10us high

// Trigger pin to HIGH
digitalWrite(TriggerPin, LOW);
Duration = pulseIn (Echopin,HIGH);

//Waits for the echo pin to get high
long Distance_mm = Distance(Duration);
// Use function to calculate the distance

Serial.print("Distance = "); // Output to serial
Serial.print(Distance_mm);
Serial.println("mm");
delay(1000); // Wait to do next measurement
}

long Distance (long time)
{
long DistanceCalc; // Calculation variable
DistanceCalc = ((time /2.9) / 2); // Actual calculation in mm
//DistanceCalc = time / 74 / 2; // Actual calculation in inches
return DistanceCalc; // return calculated value
}
```

Name of the Student: Jay Moon	Roll No. 1ETB1AhS	Date: 1/10/25

Experiment No. – 8

Title: Interfacing Arduino to Bluetooth Module

Hardware Requirements: Arduino Board, HC-06 Bluetooth Module, LED

Software Requirements: Arduino IDE

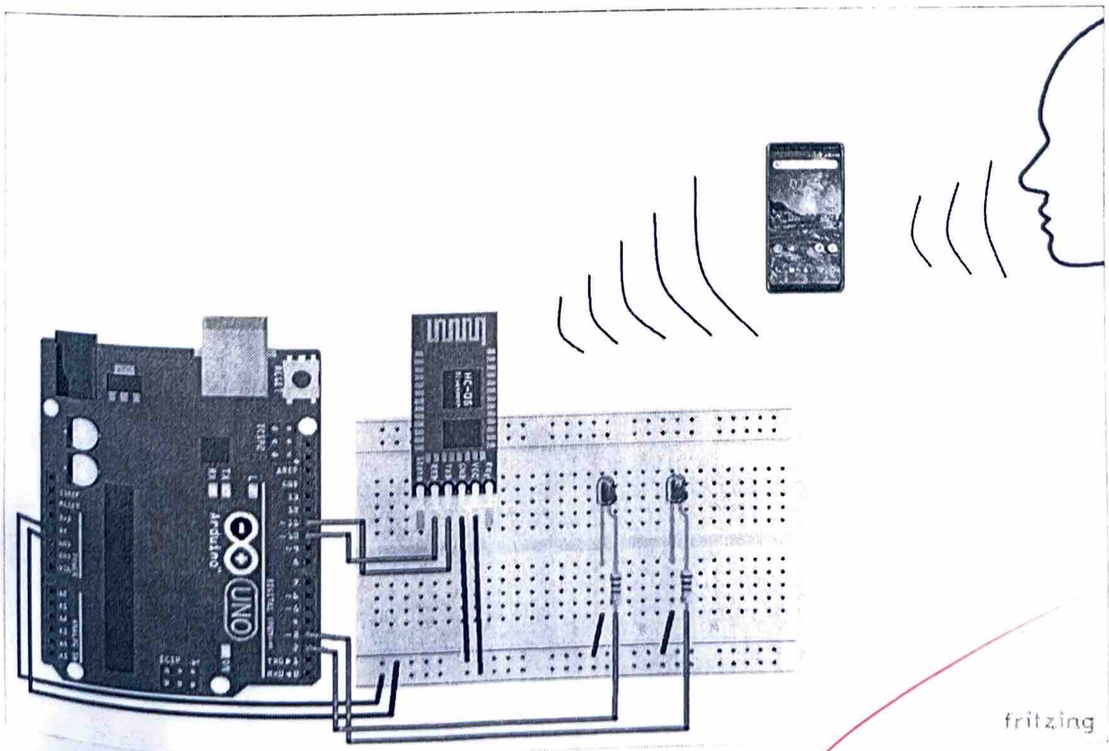
Theory:

Bluetooth can operate in the following two modes:

1. Command Mode
2. Operating Mode

In Command Mode we will be able to configure the Bluetooth properties like the name of the Bluetooth signal, its password, the operating baud rate etc. The Operating Mode is the one in which we will be able to send and receive data between the PIC Microcontroller and the Bluetooth module. Hence in this tutorial we will be toying only with the Operating Mode. The Command mode will be left to the default settings. The Device name will be HC-05 (I am using HC-06) and the password will be 0000 or 1234 and most importantly the default baud rate for all Bluetooth modules will be 9600.

The module works on 5V supply and the signal pins operate on 3.3V. hence a 3.3V regulator is present in the module itself. Hence we need not worry about it. Out of the six pins only four will be used in the Operating mode.



CODE:

```
#include <SoftwareSerial.h>

String value;
int TxD = 11;
int RxD = 10;
int servoposition;
SoftwareSerial bluetooth(TxD, RxD);


void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  Serial.begin(9600);       // start serial communication at 9600bps
  bluetooth.begin(9600);
}


void loop() {
Serial.println(value);
 if (bluetooth.available())
  {
   value = bluetooth.readString();
   if (value == "turn on all LED"){
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
   }
   if (value == "turn off all LED"){
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
   }
   if (value == "turn on LED 1"){
    digitalWrite(2, HIGH);
   }
   if (value == "turn on LED 2"){
    digitalWrite(3, HIGH);
   }
   if (value == "turn off LED 1"){
   digitalWrite(2, LOW);
   }
   if (value == "turn off LED 2"){
    digitalWrite(3, LOW);
   }
 }
}
```