**A Project Report**
On

# Employing an Arduino to control Bluetooth-enable LED's

Submitted to

**KISCIT UNIVERSITY, KAHUTA**

**KISCIT**
**UNIVERSITY**

In partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY In SOFTWARE ENGINEERING**

Submitted by

**Naila Shehzadi (222101008)**
**Iqra Matloob (222101004)**
**Laiba Pervaiz (222101002)**

Under the guidance of

**SIR TAJUMMAL HUSSAIN**

DEPARTMENT OF SOFTWARE ENGINEERING
KISCIT UNIVERSITY DR. A. Q. KHAN INSTITUTE OF
COMPUTER SCIENCE INFORMATION TECHNOLOGY

## Employing an Arduino to control Bluetooth-enabled LED's

## Apparatus:

**Arduino board:**

This is the microcontroller that will be used to control the LEDs. Examples include the Arduino UNO or Arduino Mega.

**Bluetooth module:**

This is the wireless communication module that will establish the link between the Arduino and the device with Bluetooth capability. Examples include the HC-05 or HC-06.

**LEDs:**

These are the light-emitting diodes that will be controlled by the Arduino. They can be of any color.

**Resistors:**

These are used to limit the current flowing through the LEDs and protect them from damage. The value of the resistor will depend on the specific LEDs being used.

**Jumper wires:**

These are used to connect the components together.

A device with Bluetooth capability: This is used to send commands to the Arduino through the Bluetooth module. Examples include a smartphone, tablet, or computer.

**A USB cable:**

This is used to connect the Arduino to a computer for programming and power supply.

**Breadboard:**

A breadboard or PCB board to assemble the circuit.

## Controlling LEDs using Bluetooth on an Arduino involves the following steps:

- Connecting a Bluetooth module, such as a HC-05 or HC-06, to the Arduino board. The module should be connected to the RX and TX pins on the Arduino.
- Writing an Arduino sketch that initializes the Bluetooth module and establishes a serial communication link between the Arduino and the Bluetooth module.
- Programming the Arduino to receive commands from a smartphone or other device with Bluetooth capability and use those commands to control the LEDs. This can be done by using Arduino's digital Write () function to turn the LEDs on or off based on the commands received.
- Pairing the smartphone or other device with the Bluetooth module and using a simple app or program to send commands to the Arduino.
- Upload the code to the Arduino board.
- Enjoy controlling the LEDs using Bluetooth.
- It is important to note that the exact code will depend on the type of Bluetooth module used and the specific LEDs being controlled. There are many tutorials available online that provide more detailed instructions and sample code for controlling LEDs using Bluetooth on an Arduino.

## Code for controlling an LED using Bluetooth on an Arduino:

```
#include <SoftwareSerial.h>
int led = 13; // LED is connected to digital pin 13
SoftwareSerial BTSerial(2, 3); // RX, TX
void setup() {
  pinMode(led, OUTPUT); // set LED pin as an output
  BTSerial.begin(9600); // set Bluetooth communication baud rate
  BTSerial.println("Bluetooth initialized");
}
void loop() {
  if (BTSerial.available()) { // check if there is data coming from Bluetooth
    char data = BTSerial.read(); // read the data
    if (data == '1') { // if data is '1'
      digitalWrite(led, HIGH); // turn the LED on
    } else if (data == '0') { // if data is '0'
      digitalWrite(led, LOW); // turn the LED off
    }
  }
}
```

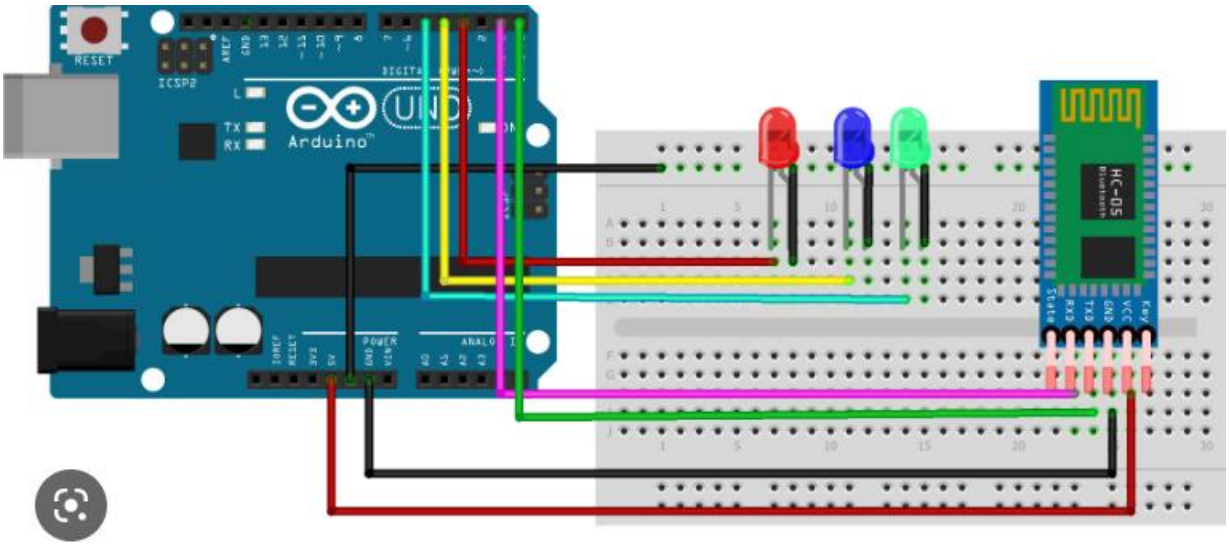
This code uses the Software Serial library to establish a serial communication link between the Arduino and a Bluetooth module. The LED is connected to digital pin 13, and the Bluetooth module's RX and TX pins are connected to pins 2 and 3 on the Arduino, respectively.

In the setup () function, the LED pin is set as an output and the Bluetooth communication baud rate is set to 9600. In the loop () function, the code checks if there is data coming in from the Bluetooth module. If there is data, it reads the data and compares it to the characters '1' and '0'. If the data is '1', the LED is turned on using the digital Write() function, and if the data is '0', the LED is turned off.

It is important to note that this is just a simple example, and the code can be modified to include more complex functionality such as dimming, color changing or multiple LED control.

Also, it is important to make sure that the baud rate of the Arduino and the baud rate of the Bluetooth module are the same, otherwise the communication between the two will not be established.

**CIRCUIT:**




**Explanation:**

In this circuit, the LED is connected to digital pin 13 on the Arduino and a current limiting resistor (R1) is connected in series with the LED. The positive leg of the LED is connected to the digital pin 13 and the negative leg of the LED is connected to the one end of the resistor R1. The other end of the resistor R1 is connected to the GND pin of the Arduino.

The Bluetooth module (HC-05 or HC-06) is connected to the Arduino's RX and TX pins. The VCC of the module is connected to the 5V of the Arduino and the GND of the module is connected to the GND of the Arduino.

It is important to note that the values of the resistor will depend on the specific LED being used and the desired current. It is important to check

the specifications of the LED and use the appropriate value of the resistor, to prevent damage to the LED.

When the circuit is powered on, the Arduino will communicate with the Bluetooth module, and will receive commands to turn the LED on or off.

You can also connect multiple LEDs with the same circuit, just connect the positive leg of the LEDs to the digital pins of the arduino and the negative legs to the GND through the resistors

## CONCLUSION:

The final result of controlling LEDs using Bluetooth on an Arduino is the ability to remotely turn the LEDs on or off, adjust their brightness, or change their color using a device with Bluetooth capability. This can be done by sending commands to the Arduino through a Bluetooth module, which then uses those commands to control the LEDs.

The final result also depends on the specific code and circuit you have used, it could be a simple on-off control or a more complex control system that include dimming, color changing, strobe effect or any other effects that you have implemented using your code.

Additionally, the final result can also include the ability to monitor the status of the LEDs or receive feedback from the Arduino. This can be useful for creating interactive projects or for monitoring the status of a system remotely.

Overall, controlling LEDs using Bluetooth on an Arduino allows for greater flexibility and convenience in controlling lighting systems, and it can be used for a wide range of applications, from home automation to art installations.