Advanced Arduino Lab - Sensors and Bluetooth

Objective

The objective of this lab is to teach students how to utilize the Arduino IDE Software platform to write logic commands that respond to an input from a sensor. This code will be used to control inputs and outputs of an Arduino Uno microcontroller. This lab will also cover how to send and receive text with a bluetooth module. This lab builds on the content of the "Arduino Programming and Building Lab" and assumes the students have a basic understanding of how to use Arduino. Please refer back to the "Arduino Programming and Building Lab" when needed.

Downloads

The Arduino IDE can be downloaded from here: https://www.arduino.cc/en/Main/Software

The library for the Ultrasonic Range Finder:
NewPing.h (https://playground.arduino.cc/Code/NewPing#Download)
The library for the Bluetooth module:
Adafruit Bluefruit LE Master library
(https://github.com/adafruit/Adafruit_BluefruitLE_nRF51)
The Adafruit Bluefruit LE Connect app (free): Google Play Store or Apple's App store.

Apparatus and Equipment Overview

The equipment that will be used in this lab includes:

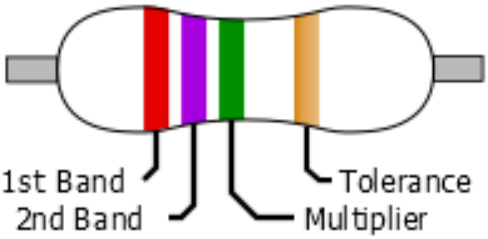
- 1 × Arduino Uno R3 microcontroller
- 1 × USB 2.0 type A plug to USB 2.0 type B plug cable
- 1 × Lab or personal computer (running Windows, Macintosh or Linux OS)
- 1 × Arduino IDE software
- 1 × Photoresistor
- 1 × Button
- 1 × 5mm LED
- 1 × 330Ω resistor
- 2 × 10kΩ resistor

- 1 × Adafruit Bluefruit LE module
- 1 x mobile phone with LE bluetooth
- 16 × Lengths of wire

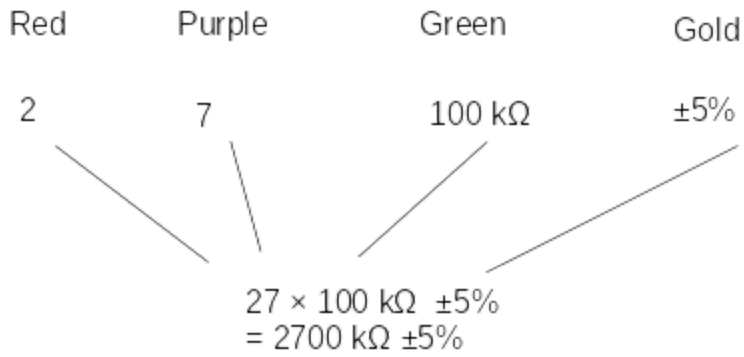
Background

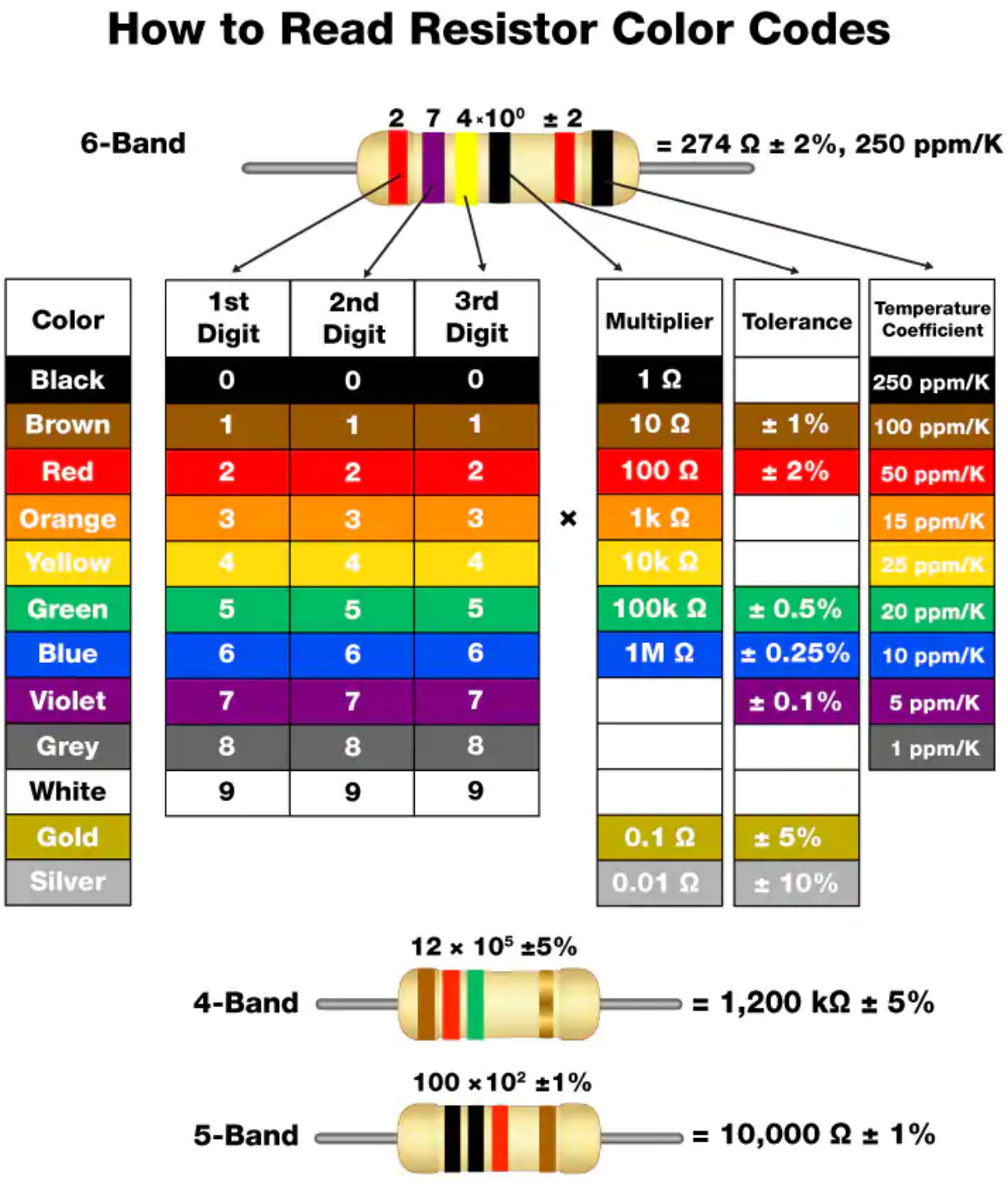
Resistor

A resistor is an electrical component which creates electrical impedance, or resistance to current flow. The amount of resistance a resistor provides can be read from the bands of colour on the resistor, which are read left to right. For four bands resistors, the first and second bands represent digits, while the third band represents a multiplier to multiply the digits of the first and second band by. The fourth band is the tolerance, it represents by how much the resistor may deviate from the value indicated by the bands. The value of the resistor can also be determined using the ohmic function of a multimeter. The table and image below shows how to read the colours.



For example, this resistor has a value of 2700 kΩ and a tolerance of ± 5%.





Resistors are often used in series with components to reduce the amount of current flowing through a circuit, often to protect components rated for lower current amounts.

Pull-up and pull-down resistors are used to bias an input on the Arduino to be either HIGH or LOW respectively. This needs to be done as the resting level of the input isn't necessarily 0. This is especially useful when working with sensors that have an analog output.

Photoresistor

A photoresistor is a resistor that changes its resistance to current in response to the amount of light it is exposed to. As the level of light the resistor is exposed to increases, its resistance decreases, conversely, as the level of light the resistor is exposed to decreases, its resistance increases.

Button

A button is an electrical component that allows electrical current to pass when the button is pressed, otherwise it does not permit electrical current to pass.

Bluetooth module

Bluetooth networks (commonly referred to as piconets) use a master/slave model to control when and where devices can send data. Every single Bluetooth device has a unique 48-bit address, commonly abbreviated BD_ADDR. This will usually be presented in the form of a 12-digit hexadecimal value. The most-significant half (24 bits) of the address is an organization unique identifier (OUI), which identifies the manufacturer. The lower 24-bits are the more unique part of the address. Bluetooth profiles are additional protocols that build upon the basic Bluetooth standard to more clearly define what kind of data a Bluetooth module is transmitting. While Bluetooth specifications define how the technology works, profiles define how it's used.
The profile(s) a Bluetooth device supports determine(s) what application it's geared towards. A hands-free Bluetooth headset, for example, would use headset profile (HSP), while a Nintendo Wii Controller would implement the human interface device (HID) profile. For two Bluetooth devices to be compatible, they must support the same profiles.

Bluetooth modules can be used to send text characters, bytes, color information, etc to a bluetooth receiver like a mobile phone or laptop. It can also receive this information from another bluetooth device (like a phone or laptop). They can come in many forms with different ways to program them and different functionalities. Examples of such modules are the Adafruit Bluefruit LE module or the HC-05 Bluetooth module. You can use a pre-made app on a phone or you can make a custom app with MIT App Inventor, Android Studio or BuildFire.

Other sensors

Ultrasonic Range Finder

An ultrasonic range finder is a component that measures distance using sound. The component emits a sound at a high frequency, above the range of human hearing, and then listens for the echo of the sound wave. The speed of sound in air is dependent on the atmospheric conditions, such as temperature, humidity and air pressure, as a result the speed of sound varies from place to place and time to time. Despite the speed of sound through air changing, the ultrasonic range finder can still determine distances fairly accurately. The ultrasonic range finder has a limit to the distance it can measure, from about 3 cm to 400 cm, as well as how often it can measure the distance, approximately every 29 ms.

Passive Infrared Motion Sensor (PIR)

A passive infrared motion sensor is a component that detects changes in the amount of infrared light it receives. When it detects a change in the amount of infrared light it receives, it brings the output pin to HIGH, when there is no change, the output pin is LOW. PIRs are often used in home security systems.

Pre-Lab Preparation

Before arriving to the lab, students should review the lab manual and familiarize themselves with the lab setup and procedure. Furthermore, it is recommended that the students review the "Arduino Programming and Building Lab" as this lab assumes the students have a solid understanding of the aforementioned lab's content. It is recommended that students use their own computers for this lab, however lab computers are available. If using a personal computer it is the responsibility of the students to download and install the Arduino IDE as well as all the required libraries, all of which can be found in the downloads section. Furthermore it is encouraged that students review and practice the Arduino syntax beforehand. The Arduino reference can be found here:
https://www.arduino.cc/reference/en/

Pre-Lab Questions

What does each band, from left to right on a resistor represent?

1st, 2nd, 3rd, multiplier, tolerance, temperature coeff

Why might a resistor be placed is series with an LED or other electrical component?

regulate current flow and to protect electronics from high current

What is the purpose of a pull-down resistor?

Used to bias LOW input in arduino

What model do bluetooth modules use?

master/slave model

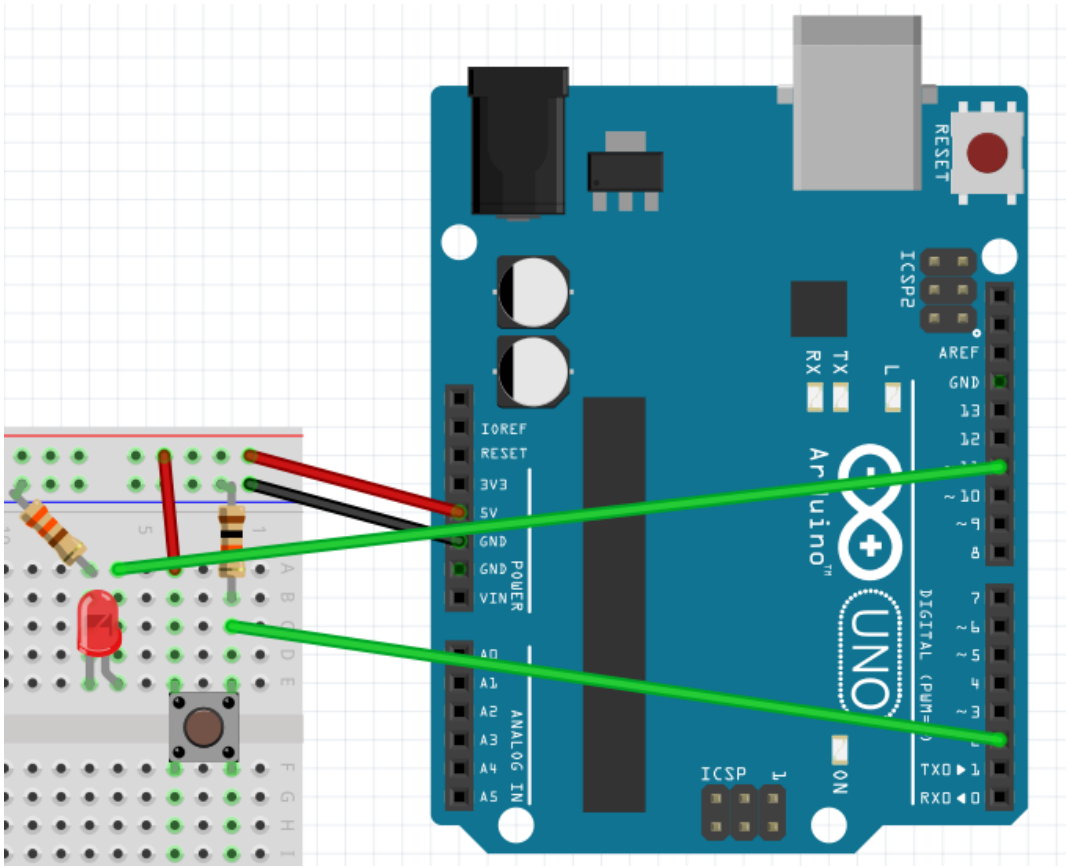
What is the main difference between ultrasonic and PIR sensors?

Ultrasound use high frequency sound whereas PIR uses infrared light

Part A - Photo Resistor, Button & LED Program

This part of the lab will cover hooking up a photoresistor, button and LED to the Arduino as well as how to create a program to control the LED using the photoresistor and button. The LED will act as an output, while to photo resistor will determine how bright the LED is while the button is pressed.

1. Connect the button, LED and photoresistor as shown in the wiring diagram below.



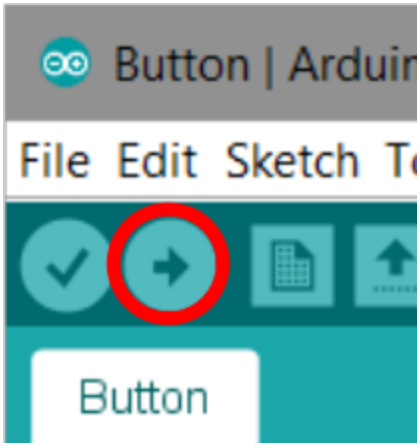
*Note the colours of the resistors, we have a 10kΩ resistor from the output of the button to ground, this is a pulldown resistor. It is used to bias the input pin to LOW.

2. Launch the Arduino IDE software. Once started select Tools>Board>Arduino/GenuinoUno and select the port the Arduino is connected to at Tools>Port. Open File>Examples>02.Digital>Button. This will open a new sketch that looks like the one below.

Button

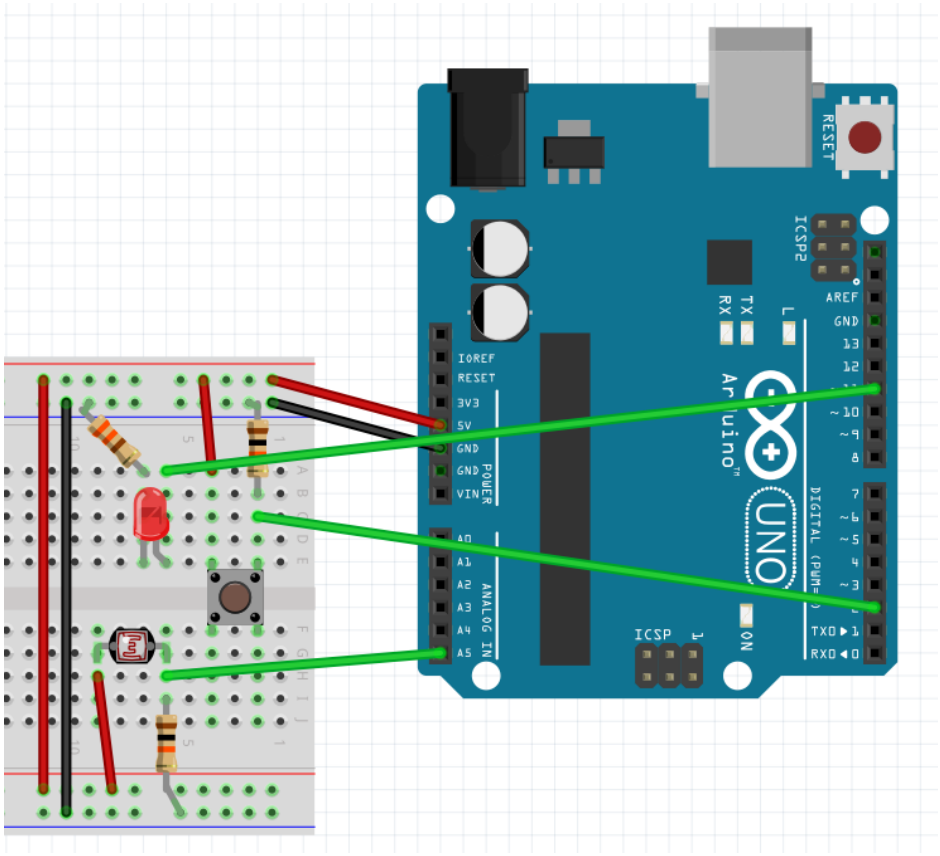
```
  created 2005
  by DojoDave <http://www.0j0.org>
  modified 30 Aug 2011
  by Tom Igoe

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Button
*/

// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

3. Now upload the sketch to the board by clicking the button highlighted in the image below. This will try to compile the program and upload it to the board. If there are any errors in compilation or uploading the Arduino IDE will display the error message at the bottom of the IDE in the console area.



4. Once the upload is complete, the program will automatically run on the microcontroller. In this case, the LED will turn on whenever the button is pressed.

5. Now modify your circuit to look like the wiring diagram below.



*Note that we change the output pin from 13 to 11, this is because we need an analog output. A pulldown resistor was placed on the output of the photoresistor.

6. Now add or modify the following lines of code to the program;
    - Modify “const int ledPin = 13” to the new pin of the LED, pin 11.
    - On the next line add “const int sensorPin = A5;”
    - On the line after “buttonState = 0;” add “int sensorValue;”
    - In setup add “Serial.begin(9600);”
    - At the beginning of the loop add “sensorValue = analogRead(sensorPin);” and immediately after add “Serial.println(sensorValue);”
    - In the if statement, when the buttonState is High, change the digitalWrite to “analogWrite(ledPin, sensorValue);”

    The program should now look like the image below.

```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Button
*/

// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  11;      // the number of the LED pin
const int sensorPin = A5;

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
int sensorValue;

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    analogWrite(ledPin, sensorValue);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

7. Upload and test out your new program. What happens when you press the button now? Try changing the amount of light the photoresistor is exposed to, what happens?

---

8. Think of a way to change the behavior of the LED in response to the input from the photoresistor and implement it!

Part B - Receiving text via Bluetooth

In this part of the lab you will create a program that will receive text from your phone to act as another button in your system. Do not erase your previous code.

1. Making sure the correct Arduino libraries are installed, navigate to File>Examples>Adafruit BluefruitLE nRF51 and open the bleuart_cmdmode example sketch. This will open a sketch with two available tabs at the top; bleuart_cmdmode and Bluefruit_Config.h.
2. Change Bluefruit_Config.h tab to utilize a software UART Connection. This is a type of serial connection (similar to the communications used in a USB device) used to communicate between the BluetoothLE module on the car and the Arduino Uno board. The sections of code required to setup a software UART connection are the common settings, the software UART settings, and the common UART settings. All of the other sections can be deleted. The pins used for the connection must also be changed to reflect which pins the module is connected to on the Arduino board. Use the photo below to set the correct pins and ensure the Bluefruit_Config.h file is correctly set up.

bleuart_cmdmode | BluefruitConfig.h §

```
// COMMON SETTINGS
// -------------------------------------------------------------------------------------------
// These settings are used in both SW UART, HW UART and SPI mode
// -------------------------------------------------------------------------------------------
#define BUFSIZE                         128   // Size of the read buffer for incoming data
#define VERBOSE_MODE                    true  // If set to 'true' enables debug output


// SOFTWARE UART SETTINGS
// -------------------------------------------------------------------------------------------
// The following macros declare the pins that will be used for 'SW' serial.
// You should use this option if you are connecting the UART Friend to an UNO
// -------------------------------------------------------------------------------------------
#define BLUEFRUIT_SWUART_RXD_PIN        16    // Required for software serial!
#define BLUEFRUIT_SWUART_TXD_PIN        17    // Required for software serial!
#define BLUEFRUIT_UART_CTS_PIN          18    // Required for software serial!
#define BLUEFRUIT_UART_RTS_PIN          15    // Optional, set to -1 if unused


// HARDWARE UART SETTINGS
// -------------------------------------------------------------------------------------------
// The following macros declare the HW serial port you are using. Uncomment
// this line if you are connecting the BLE to Leonardo/Micro or Flora
// -------------------------------------------------------------------------------------------
#ifdef Serial1    // this makes it not complain on compilation if there's no Serial1
  #define BLUEFRUIT_HWSERIAL_NAME      Serial1
#endif


// SHARED UART SETTINGS
// -------------------------------------------------------------------------------------------
// The following sets the optional Mode pin, its recommended but not required
// -------------------------------------------------------------------------------------------
#define BLUEFRUIT_UART_MODE_PIN         19    // Set to -1 if unused
```

3. Now alter the bleuart_cmdmode tab. The **software** UART setting at the top will have to be uncommented (delete the "/*" characters before the lines and "*/" after the lines), and the **hardware** UART lines must be removed (or commented out by adding "//" before the line).

```
// Create the bluefruit object, either software serial...uncomment these lines

SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN, BLUEFRUIT_SWUART_RXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);


/* ...or hardware serial, which does not need the RTS/CTS pins. Uncomment this line */
// Adafruit_BluefruitLE_UART ble(Serial1, BLUEFRUIT_UART_MODE_PIN);

/* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
//Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

/* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
//Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
//                             BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
//                             BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

4. Before the setup section, include all the variables from the first example to control the LED, button and photoresistor. See the section of code below as a reference.
    a. Note that the sensor pin has changed to 10.

```
/**************************************************************************/
/*!
    @brief  Sets up the HW an the BLE module (this function is called
            automatically on startup)
*/
/**************************************************************************/
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  11;      // the number of the LED pin
const int sensorPin = 10;

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
int sensorValue;

void setup(void)
{
```

5. Alter the setup section of the sketch. Remove the while statement and the delay statement that occur before the Serial.begin(115200) statement. Insert statements to set the LED and button states immediately after the Serial.begin statement (use the pinMode command).

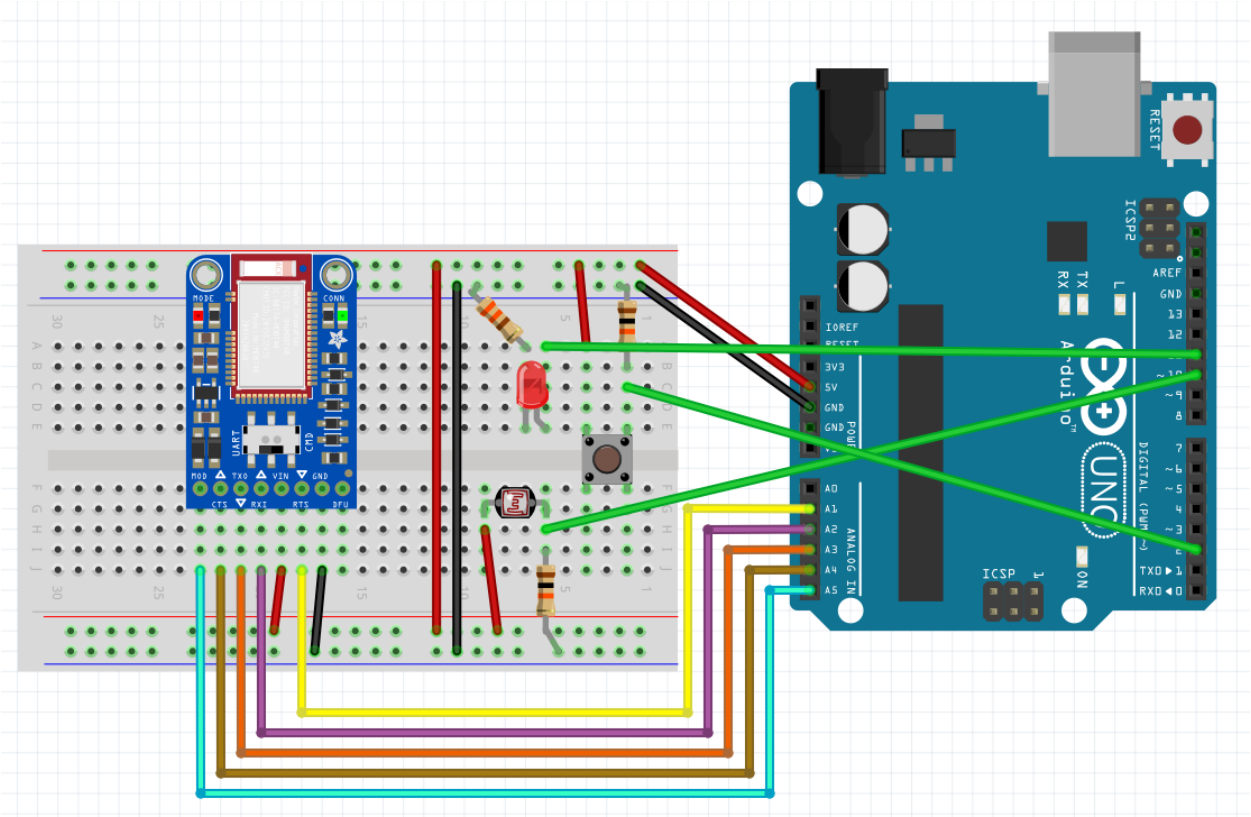
```
void setup(void)
{
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  Serial.println(F("Adafruit Bluefruit Command Mode Example"));
  Serial.println(F("---------------------------------------"));

  /* Initialise the module */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?"));
  }
  Serial.println( F("OK!") );
```

6. In the loop, after the statement “ble.readline()” delete the following if statement that contains “strcmp”. After the statement “Serial.print(F(“[Recv] ”))” add the statement “String received = String(ble.buffer);”. Now you can add statements to control the LED within the loop section of the sketch.
7. Immediately after the statement “String received = String(ble.buffer);” (near the end of the loop section) add if and if else structure statements to control LED state based on the data received via Bluetooth as well as the state of the button (taken from your previous code). See the sketch section below as a reference.

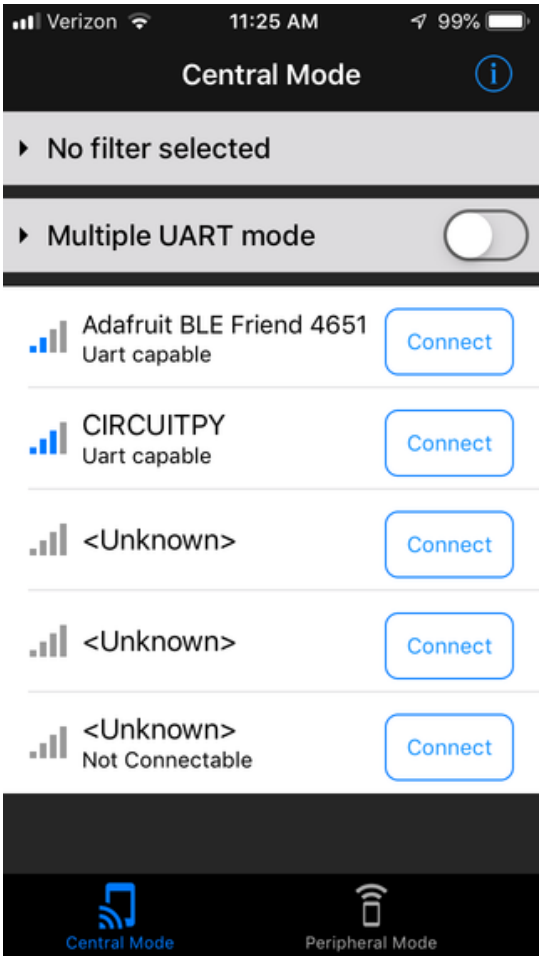
```
  // Check for incoming characters from Bluefruit
  ble.println("AT+BLEUARTRX");
  ble.readline();

  // Some data was found, its in the buffer
  Serial.print(F("[Recv] ")); Serial.println(ble.buffer);
  String received = String(ble.buffer);
  buttonState = digitalRead(buttonPin);
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);

  if (buttonState == HIGH || received == "y") {
    // turn LED on:
    analogWrite(ledPin, sensorValue);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }

  ble.waitForOK();
}
```

8. Finally upload the sketch to the Arduino.

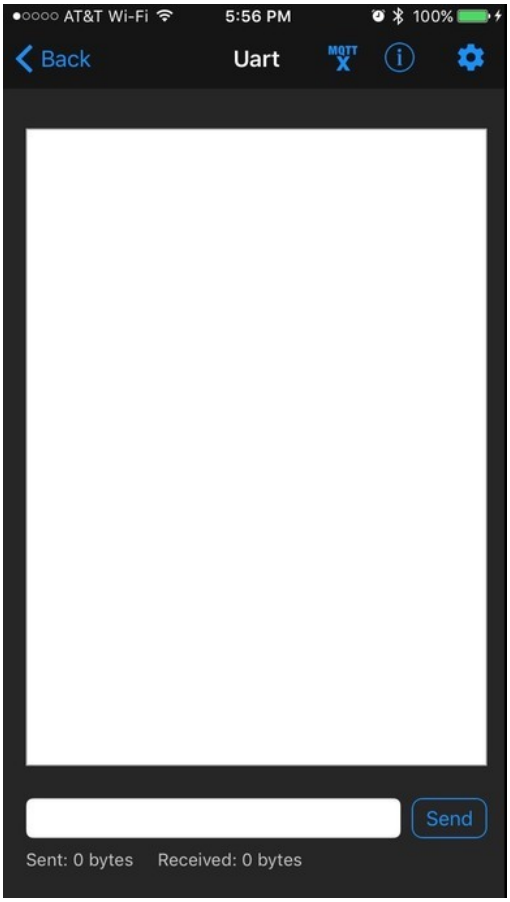
9. Add the bluetooth module to your circuit. Make sure that the switch on the module is in CMD position. See diagram below.



10. Enable bluetooth on your phone and open the Adafruit Bluefruit LE Connect app. It should scan for devices automatically.



11. Click ‘Connect’ for your Adafruit device (strongest bars). A blue light on the bluetooth module should turn on. Then choose the Uart option.
12. In the text field at the bottom, type y and watch what happens to your LED.



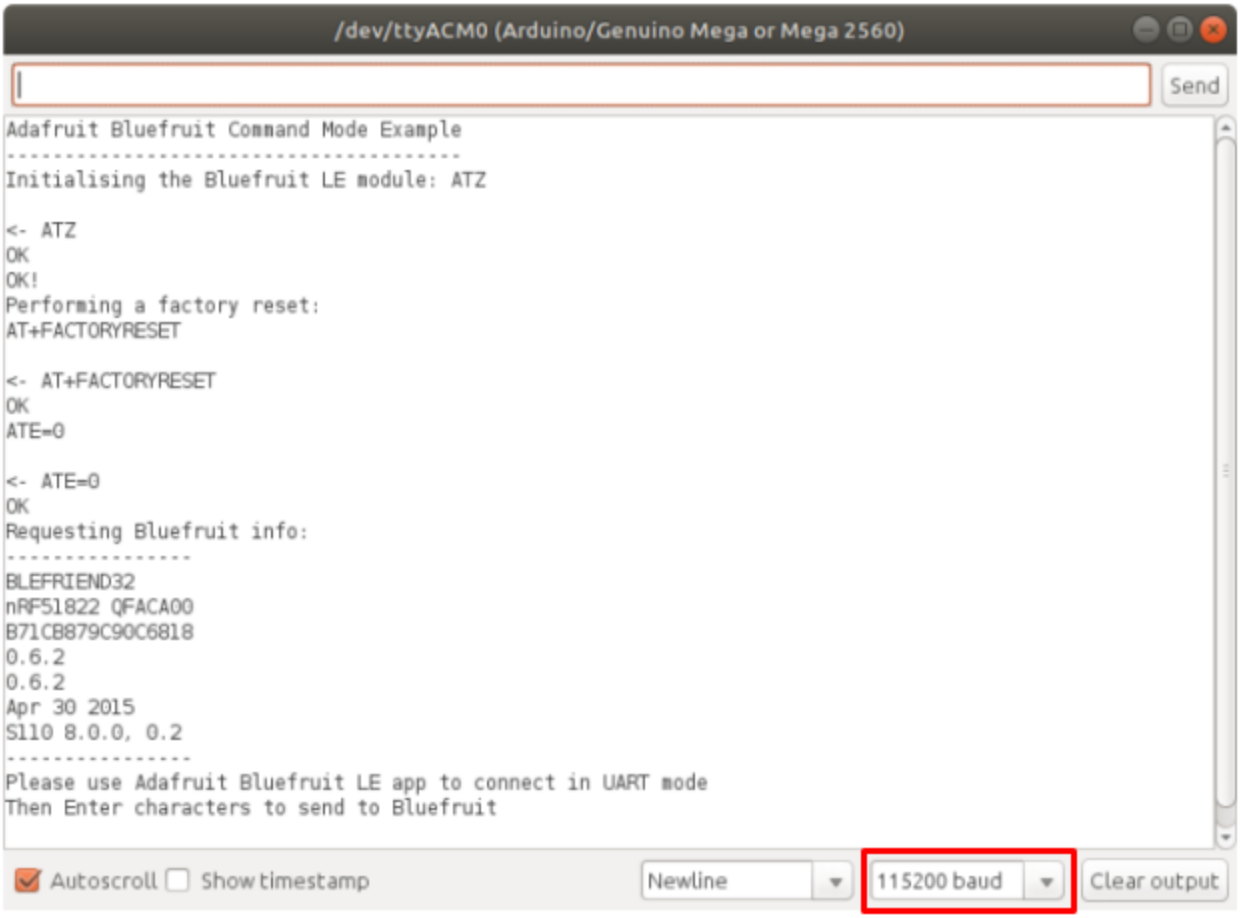
Part C - Sending text via Bluetooth

In this part of the lab you will create a program that will send the value from the photoresistor to your phone. You can send information to your phone in 2 ways. Do not erase your previous code.

Manual

The first method of sending information to your phone via Arduino is manually.

1. Open the serial monitor in Arduino (Tools>Serial Monitor) and change the baud rate to 115200.



2. In the text field at the top type some characters and click send. You should see this information in the Uart screen on your phone.

With code

The second method of sending information to your phone via Arduino is with code to do it automatically without manual input.

1. In the loop, we will add instructions so that you are alerted on your phone when the light level is low. See the sketch section below as a reference.

```
  if (buttonState == HIGH || received == "y") {
    // turn LED on:
    analogWrite(ledPin, sensorValue);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }

  if (sensorValue < 500) {
    char message[40];
    char value[12];
    itoa(sensorValue, value, 10); //convert int to char
    strcpy(message,"The light level is low! It is ");
    strcat(message,value);
    // Send characters to Bluefruit
    Serial.print("[Send] ");
    Serial.println(message);

    ble.print("AT+BLEUARTTX=");
    ble.println(message);

    // check response stastus
    if (! ble.waitForOK() ) {
      Serial.println(F("Failed to send?"));
    }
  }

  ble.waitForOK();
}
```

2. Open the serial monitor, change the light level of the photosensor and wait for a message on your phone. You might need to change the limit value of 500 depending on the behavior you want and the ambient light in the room.

Resources

In this lab you have only seen text being sent and received but there are many more things that you can do with a bluetooth module. Here are a few links that are useful for this type of bluetooth module:
https://learn.adafruit.com/bluefruit-le-connect/ios-setup
https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend/introduction