Morse Code **Course: CSCI 3331**
Lab 1 **Section: 01**

Ahmed Muntasir Hossain

Description of the Lab

The purpose of the lab was to design and implement an encoding device using an Arduino. The device would encode my first name "Muntasir" in Morse code. Morse code is a sequence of dots and dashes, and to represent these two distinct signals, two different colors of LEDs were used. Each letter of my first name would be encoded as a sequence of those two different colors and outputted to the viewer. A button is connected to the Arduino and it will have to be pressed to begin sending the signals. To continue sending the signals one after another it would have to be pressed down. Once the button is released, the device will stop sending the signal and will resume only if the button is pressed again. The signal will resume from the next letter before the button was released. Once the device has outputted my entire name in morse code, it will stop and exit the program. To restart the program, the reset button on the Arduino would have to be pressed.

Power

The breadboard has two columns marked + (positive) and – (negative). One wire is connected to each column and it can be placed in any hole on that column. This is used to supply power from the Arduino to the breadboard. The positive and negative wires connected to the breadboard are connected to the 5V power pin and GND power pin on the Arduino, respectively.

Output

The output of this device is the RGB LED. It is an electrical component consisting of three LEDs Red, Blue, and Green. In this lab, the green LED was assigned to be dash (-) , and the red LED was assigned to be dot (.). The blue LED was unused, however any combination of two colors can be used as the signal. The RGB LED has 4 pins "R" (red), "G" (green), "B" (blue), and "–" (negative) which are connected parallelly on the breadboard (different rows). One wire is connected to each pin serially (same row). The R, G, and B wires are connected to any three digital i/o pins (pin number 2 – 13) on the Arduino. The negative wire is connected to any hole (socket) on the negative line on the breadboard. This completes the wiring connection of the RGB LED. The R, G, and B pin markings on the LED may be incorrect and therefore to configure it correctly, the device should be tested. This should be done by setting each individual pin (LED color) to HIGH and observing its color. The color determined from the test can be used to know the correct configuration of the pins.

Input

The input of this device is the Push Button. The button has three pins "S", "-" (negative) and the middle pin which are connected parallelly on the breadboard. One wire is connected to each pin serially. The middle pin wire is connected to the positive line, the negative pin wire is connected to the negative line, and the "S" wire is connected to any digital i/o pin (pin number 2 – 13) on the Arduino. This completes the wiring of the push button. The button may not be configured in the conventional manner and it may be so that the "switch" is closed when the button is not pressed, and the "switch" is open when the button is pressed. This can be determined by setting the output of a particular LED to the input being received from the button (button pressed or not pressed) and observing if the light turns on or off when the button is pressed*. Additionally, there may be an error in the markings of the pins. In my push button the markings of S and the negative pin were swapped. To determine if there is an error, the not (!) operator should be used when testing the LED. When testing with the not operator, if the light does the exact opposite of its actions when the not operator was not used in the code*, then the configuration is correct, otherwise there is an error in the markings and the wires should be swapped.

Parts of Arduino

I learned about digital i/o pins. They are responsible for obtaining an input or producing an output in electrical components. The direction of the pins can be set in the code and then we can control the output of a component by digitally writing to it or accept the input from a component by digitally reading from it.

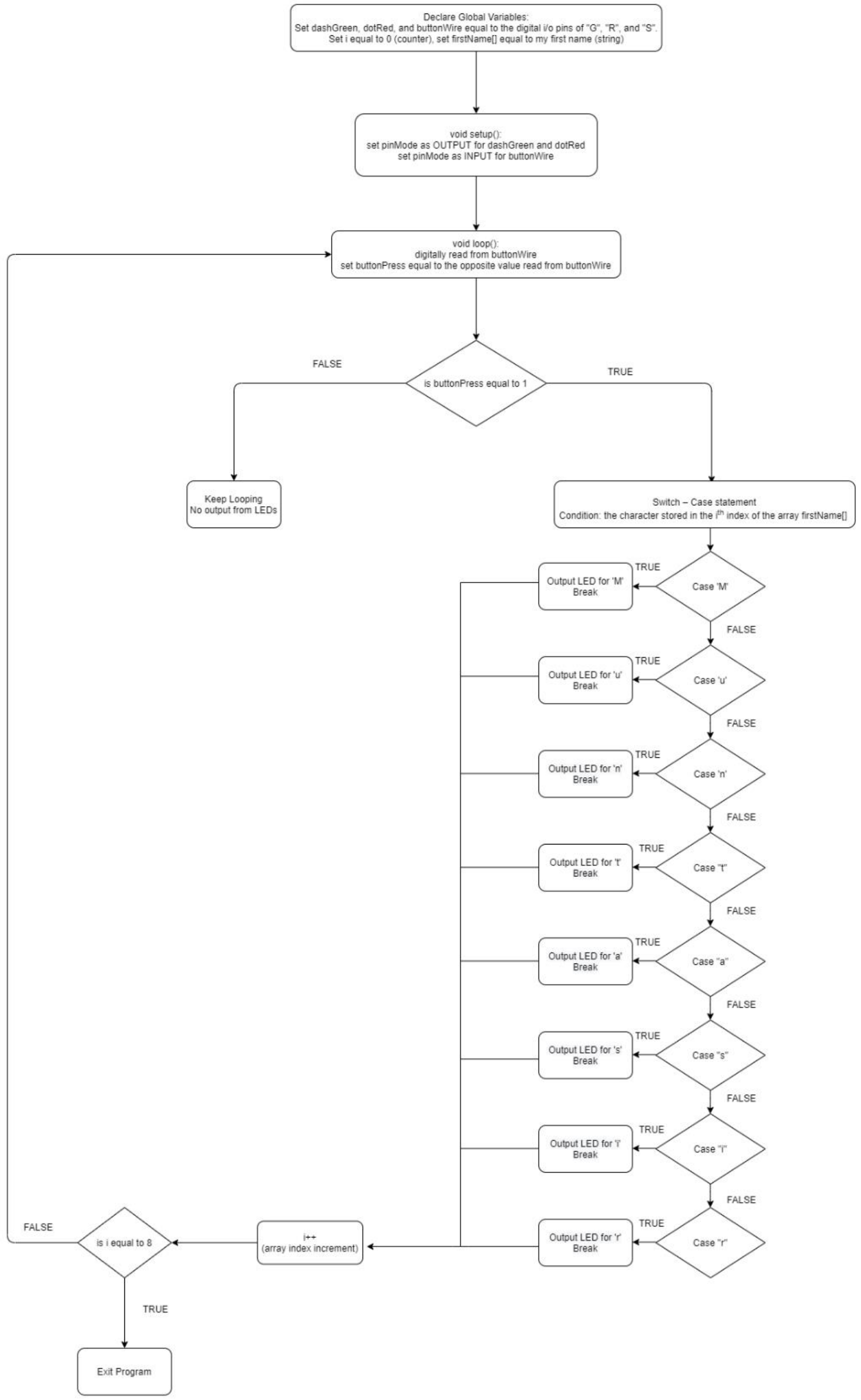
I also learned about power pins which are 5V and GND. The 5V is the positive pin and GND is the negative pin. They supply power to the breadboard from the Arduino.

Additionally, I learned about the reset button on the Arduino which resets the program and starts it from the beginning.

Morse Code **Course: CSCI 3331**
Lab 1 **Section: 01**

Ahmed Muntasir Hossain

Flow Chart of the Program Logic


Declare Global Variables:
Set dashGreen, dotRed, and buttonWire equal to the digital i/o pins of "G", "R", and "S".
Set i equal to 0 (counter), set firstName[] equal to my first name (string)
void setup():
set pinMode as OUTPUT for dashGreen and dotRed
set pinMode as INPUT for buttonWire
void loop():
digitally read from buttonWire
set buttonPress equal to the opposite value read from buttonWire
FALSE
is buttonPress equal to 1
TRUE
Keep Looping
No output from LEDs
Switch – Case statement
Condition: the character stored in the i^{th} index of the array firstName[]
TRUE
Output LED for 'M'
Break
Case 'M'
FALSE
TRUE
Output LED for 'u'
Break
Case 'u'
FALSE
TRUE
Output LED for 'n'
Break
Case 'n'
FALSE
TRUE
Output LED for 't'
Break
Case "t"
FALSE
TRUE
Output LED for 'a'
Break
Case "a"
FALSE
TRUE
Output LED for 's'
Break
Case "s"
FALSE
TRUE
Output LED for 'i'
Break
Case "i"
FALSE
TRUE
Output LED for 'r'
Break
Case "r"
i++
(array index increment)
FALSE
is i equal to 8
TRUE
Exit Program

Morse Code **Course: CSCI 3331**
Lab 1 **Section: 01**

Ahmed Muntasir Hossain

Code

```
const int dotRed = 5;                 //digital i/o pin number for "R" on the RGB LED

const int dashGreen = 4;              //digital i/o pin number for "G" on the RGB LED

const int buttonWire = 8;             //digital i/o pin number for "S" on the push button


int i = 0;                            //array index counter

const char firstName[] = "Muntasir";  //string storing first name



void setup() {


  pinMode(dotRed, OUTPUT);            //set pinMode of dotRed to OUTPUT

  pinMode(dashGreen, OUTPUT);         //set pinMode of dashGreen to OUTPUT


  pinMode(buttonWire, INPUT);         //set pinMode of buttonWire to INPUT


}


void loop() {


  int buttonPress = !digitalRead(buttonWire);   //set buttonPress equal to the not of the value read from buttonWire


  if (buttonPress)                    //true if the button is pressed

  {

    switch (firstName[i])             //switch-case condition: the character stored in the ith index of the array

    {

      case 'M':                       //morse code signal for 'M'

        digitalWrite(dashGreen, HIGH);        //Green LED is HIGH for 200 ms

        delay(200);

        digitalWrite(dashGreen, LOW);         //Green LED is LOW for 200 ms - this causes the light to flash on and
then off
```

```
    delay(200);
    digitalWrite(dashGreen, HIGH);
    delay(200);
    digitalWrite(dashGreen, LOW);
    delay(1000);                    //Waits 1000 ms before the next letter is outputted
    break;

  //same principle as above (case 'M')
  case 'u':                         //morse code signal for 'u'
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(200);
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(200);
    digitalWrite(dashGreen, HIGH);
    delay(200);
    digitalWrite(dashGreen, LOW);
    delay(1000);
    break;

  case 'n':                         //morse code signal for 'n'
    digitalWrite(dashGreen, HIGH);
    delay(200);
    digitalWrite(dashGreen, LOW);
    delay(200);
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(800);
    break;

  case 't':                 //morse code signal for 't'
    digitalWrite(dashGreen, HIGH);
    delay(200);
    digitalWrite(dashGreen, LOW);
    delay(1000);
    break;

  case 'a':                 //morse code signal for 'a'
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(200);
    digitalWrite(dashGreen, HIGH);
    delay(200);
    digitalWrite(dashGreen, LOW);
    delay(1000);
    break;

  case 's':                 //morse code signal for 's'
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(200);
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(200);
    digitalWrite(dotRed, HIGH);
```

```
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(1000);
    break;

   case 'i':                          //morse code signal for 'i'
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(200);
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(1000);
    break;

   case 'r':                          //morse code signal for 'r'
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(200);
    digitalWrite(dashGreen, HIGH);
    delay(200);
    digitalWrite(dashGreen, LOW);
    delay(200);
    digitalWrite(dotRed, HIGH);
    delay(200);
    digitalWrite(dotRed, LOW);
    delay(1000);
    break;
  }
```

```
    i++;                                          //increment counter


    if (i == 8)                                   //exit from the program if the end of the string is reached
      exit(0);
  }
}
```