Serial Communication **Course: CSCI 3331**
Lab 2 **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. An input message will be sent to the device to be encoded and outputted as Morse code, using the Serial Monitor. Morse code is a sequence of dots and dashes, and to represent these two distinct signals, two different colors of LEDs would be used. Each letter of the message would be encoded as a sequence of those two different colors and outputted to the viewer.

The user can set the color of the two signals, dots and dash using the key character ‘d’ (lowercase d) and ‘D’ (uppercase D), respectively, at the beginning of their input. The color of the dot and dash can be set by inputting ‘r’ for red, ‘b’ for blue, and ‘g’ for green after the key character. Any other letter that is inputted would result in an error message. However, for the convenience of the user and for exception handling, any letter after ‘dr’ or ‘dg’ or ‘db’ or ‘Dr’ or ‘Dg’ or ‘Db’ will be ignored, and the color of dot or dash will be set with respect to the second letter after ‘d’ or ‘D’. To clarify, the color of dot and dash would have to be set separately by sending two separate inputs with the appropriate key characters. The default color for dot and dash is red and green, respectively.

The key character ‘m’ has been preset for the user to send a message that would be encoded by the device and outputted as Morse code. The user would have to enter ‘m’ at the beginning of their input for the device to understand that the user is sending a message, in contrast to the user setting the color for dot and dash (‘d’ and ‘D’). The message would follow after the key character and can be a minimum length of 0 characters and a maximum length of 28 characters excluding the key character ‘m’. This is because the array storing the message can hold a maximum of 30 characters including the null terminator. The message should not contain any special or numeric characters. The only acceptable characters are from the modern English alphabet. If the user sends no message after the key character, then the device will print to the Serial Monitor stating that no message has been sent. Additionally, if the user inputs a special or numeric character (not an English alphabet), then an error message will be printed to the Serial Monitor stating that it is an invalid character.

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 the entire message in morse code, it will reset to the initial letter of the last message and begin outputting it. The program will continue outputting the same message in a loop until the message is changed. To restart the program, the reset button on the Arduino would have to be pressed.

The color of dot or dash can be changed in the middle of a message. The device will continue outputting the same message with a different color for the signals. This change will occur from the next letter of the message after the button is released or if the button is being continuously pressed down, then the change will occur from the next letter after the input for the color change is received.

Serial Communication **Course: CSCI 3331**
Lab 2 **Section: 01**

Ahmed Muntasir Hossain

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. 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 Universal Asynchronous Receiver-Transmitted (UART) port on the Arduino. The port is referred to as COM3 on my device. The name of the port may vary depending on the Arduino. The port is connected to a PC using a USB 2.0 cable (type A/B). Using the Serial Monitor, a message can be sent to the Arduino to be processed. Using the same port, a message can be transmitted to the PC to be displayed on the Serial Monitor.

Another 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.

Serial Communication **Course: CSCI 3331**
Lab 2 **Section: 01**

Ahmed Muntasir Hossain

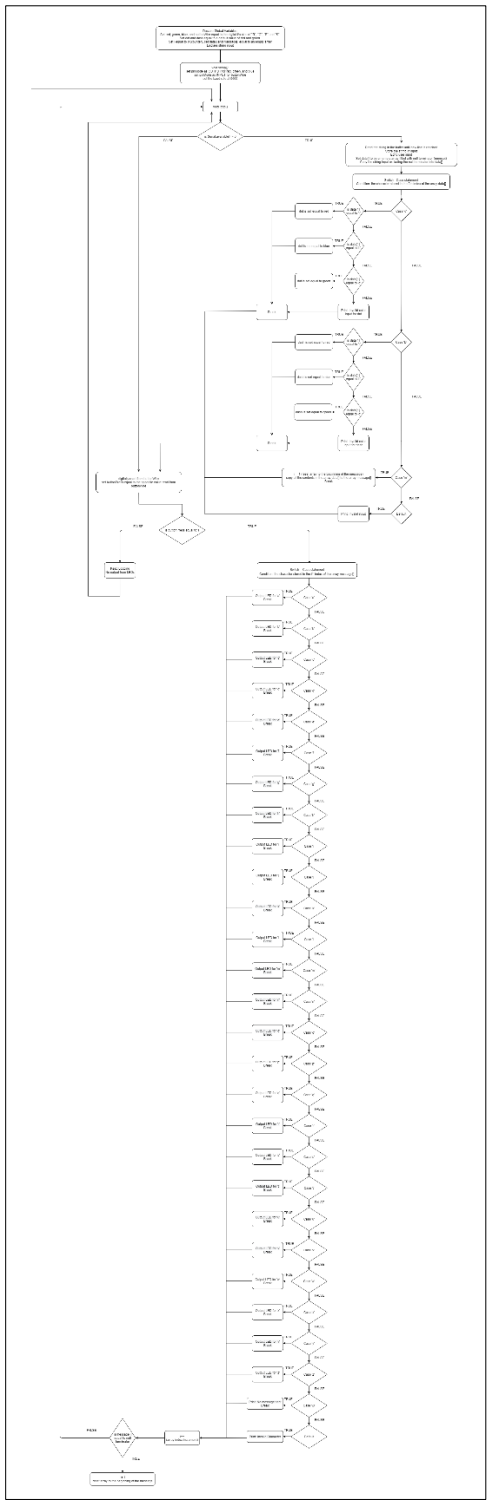
Parts of Arduino

I learned about UART port which is used for asynchronous serial communication. In asynchronous communication, there is a separate independent clock in each CPU. Information is transmitted from one CPU and received by the other CPU, and vice versa. A clock is not shared between the CPUs so there may be a delay in the receival of the bits, however, this can be countered by approximating the value of the bit to the bit that occupies the greatest percentage of time. This reduces the error and ensures that the information received is correct.

Serial Communication **Course: CSCI 3331**
Lab 2 **Section: 01**

Ahmed Muntasir Hossain

Flow Chart of the Program Logic



Please note: Quality degraded when inserting image. Flowchart is available as a separate image in the submission folder.

Serial Communication **Course: CSCI 3331**
Lab 2 **Section: 01**

Ahmed Muntasir Hossain

Code

```
const int red = 5;                  //digital i/o pin number for "R" on the RGB LED
const int green = 4;                 //digital i/o pin number for "G" on the RGB LED
const int blue = 6;                  //digital i/o pin number for "B" on the RGB LED
const int buttonWire = 8;            //digital i/o pin number for "S" on the push button

int dot = red;                      //default value of dot is set to red
int dash = green;                    //default value of dash is set to green

int i = 0;                                  //array index counter
char data[30] = {'\0'};     //{0} or {}              //array storing data (key characters) from user
char message[30] = {'\0'};  //{0} or {}               //array storing message from user
String input;                               //stores user input

void setup() {

  pinMode(red, OUTPUT);               //set pinMode of red to OUTPUT
  pinMode(green, OUTPUT);              //set pinMode of green to OUTPUT
  pinMode(blue, OUTPUT);               //set pinMode of blue to OUTPUT
  pinMode(buttonWire, INPUT);          //set pinMode of buttonWire to INPUT

  Serial.begin(9600);                 //set baud rate to 9600
}

void loop() {

  if (Serial.available() > 0)                       //if the number of bytes available in the buffer is greater than 0
  {
    input = Serial.readStringUntil('\n');                 //read the buffer until the program reaches new line and store it in input
    Serial.println(input);                          //print the user input

    memset(data, 0, 30);                            //set data to be an empty array filled with null terminators
    strncpy(data, input.c_str(), strlen(input.c_str()));          //copy the string in input excluding the null terminator into data (null terminator exists from memset)
    //Serial.println(data);                         //developer use: prints the user input. It should output the same string as input
```

Serial Communication **Course: CSCI 3331**
Lab 2 **Section: 01**

Ahmed Muntasir Hossain

```
  switch (data[0])                                  //switch-case condition: the character stored in the 0th index of the array
  {
   case 'd':                                       //sets the color of dot
    if (data[1] == 'r')                            //if the second element of the array is 'r', then dot is red
      dot = red;
    else if (data[1] == 'g')                       //if the second element of the array is 'g', then dot is green
      dot = green;
    else if (data[1] == 'b')                       //if the second element of the array is 'b', then dot is blue
      dot = blue;
    else
      Serial.println("Invalid color input for dot");          //if the second element of the array does not match any one of these characters, then print error message
    break;

   case 'D':                                       //sets the color of dash
    if (data[1] == 'r')                            //if the second element of the array is 'r', then dash is red
      dash = red;
    else if (data[1] == 'g')                       //if the second element of the array is 'g', then dash is green
      dash = green;
    else if (data[1] == 'b')                       //if the second element of the array is 'b', then dash is blue
      dash = blue;
    else
      Serial.println("Invalid color input for dash");          //if the second element of the array does not match any one of these characters, then print error message
    break;

   case 'm':                                       //takes  the message that the user sent and stores it in the message[] array
    i = 1;                                         //resets the counter to the beginning of the array
    strncpy(message, data, sizeof(data));                  //copies all the contents in data[] into message[]
    //Serial.println(message);                     //developer use: prints user input. It should output the same string as input and data
    break;

   default:                                        //prints "Invalid input" if the user does not input the appropriate key character {d, D, m}
    Serial.println("Invalid input");
  }
 }
```

Serial Communication **Course: CSCI 3331**
Lab 2 **Section: 01**

Ahmed Muntasir Hossain

```
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 (message[i])                    //switch-case condition: the character stored in the ith index of the array
 {
  case 'a':                         //morse code signal for 'a'
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(1000);
    break;

  case 'b':                         //morse code signal for 'b'
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(200);
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(1000);
    break;
```

Ahmed Muntasir Hossain

```
case 'c':                       //morse code signal for 'c'
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(1000);
  break;

case 'd':                       //morse code signal for 'd'
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(1000);
  break;

case 'e':                       //morse code signal for 'e'
  digitalWrite(dot, HIGH);
```

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

 case 'f':                    //morse code signal for 'f'
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(1000);
  break;

 case 'g':                    //morse code signal for 'g'
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
```

```
  delay(1000);
  break;

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

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

 case 'j':                    //morse code signal for 'j'
  digitalWrite(dot, HIGH);
  delay(200);
```

```
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(1000);
    break;

  case 'k':                       //morse code signal for 'k'
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(200);
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(1000);
    break;

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

```
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(1000);
  break;

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

 case 'n':                    //morse code signal for 'n'
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(1000);
  break;
```

Ahmed Muntasir Hossain

```
case 'o':                       //morse code signal for 'o'
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(1000);
  break;

case 'p':                       //morse code signal for 'p'
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(1000);
  break;

case 'q':                        //morse code signal for 'q'
  digitalWrite(dash, HIGH);
```

```
    delay(200);
    digitalWrite(dash, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(200);
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(1000);
    break;

  case 'r':                         //morse code signal for 'r'
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(200);
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(1000);
    break;

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

```
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(1000);
  break;

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

 case 'u':                           //morse code signal for 'u'
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(1000);
  break;

 case 'v':                           //morse code signal for 'v'
  digitalWrite(dot, HIGH);
  delay(200);
```

```
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(1000);
    break;

  case 'w':                         //morse code signal for 'w'
    digitalWrite(dot, HIGH);
    delay(200);
    digitalWrite(dot, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(200);
    digitalWrite(dash, HIGH);
    delay(200);
    digitalWrite(dash, LOW);
    delay(1000);
    break;

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

```
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(1000);
  break;

 case 'y':                          //morse code signal for 'y'
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dot, HIGH);
  delay(200);
  digitalWrite(dot, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(200);
  digitalWrite(dash, HIGH);
  delay(200);
  digitalWrite(dash, LOW);
  delay(1000);
  break;

 case 'z':                          //morse code signal for 'z'
  digitalWrite(dash, HIGH);
  delay(200);
```

```
      digitalWrite(dash, LOW);
      delay(200);
      digitalWrite(dash, HIGH);
      delay(200);
      digitalWrite(dash, LOW);
      delay(200);
      digitalWrite(dot, HIGH);
      delay(200);
      digitalWrite(dot, LOW);
      delay(200);
      digitalWrite(dot, HIGH);
      delay(200);
      digitalWrite(dot, LOW);
      delay(1000);
      break;

    case '\0':                          //print "No message sent" when the element in the 1st index is a null character
      Serial.println("No message sent");
      break;

    default:                            //print "Invalid character" when the ith character in the array is not in the English alphabet or a null
character
      Serial.println("Invalid character");
  }
  i++;                                  //increment counter

  if (message[i] == '\0')               //if the next character in the array is a null terminator
  {
    i = 1;                              //reset array to the beginning of the array since the end of the message has been reached

    /*delay(500);                       //developer use: white light is outputted when end of the message is reached
      digitalWrite(red, HIGH);
      digitalWrite(blue, HIGH);
      digitalWrite(green, HIGH);
      delay(500);
      digitalWrite(red, LOW);
```

Ahmed Muntasir Hossain

```
      digitalWrite(blue, LOW);
      digitalWrite(green, LOW);
      delay(1000);*/
    }
  }
}
```