EEPROM **Course: CSCI 3331**
Lab 3 **Section: 01**

Ahmed Muntasir Hossain

Description of the Lab

The purpose of the lab was to include an additional feature (EEPROM) in the existing design of the Serial Communication Morse Code Device from Lab 2. The current color of the signals (dot and dash) and the message inputted by the user would be saved in non-volatile memory EEPROM. This would prevent the signal and message details from being erased when the Arduino is disconnected from the power supply or reset, therefore, allowing the device to continue to function without inputting the data again.

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.

Parts of Arduino

I learned about the non-volatile memory present on an Arduino known as electrically erasable programmable read-only memory, EEPROM. It can store a maximum of 512 bytes, and it can only store characters in each memory address as each of these addresses holds a byte of memory. The memory addresses are from 0 to 511. The data stored in the memory is not lost when power is disconnected, however, there is a limited number of writes to the memory as it gets destroyed after a certain number of uses. Lastly, EEPROM is an older version of flash memory.

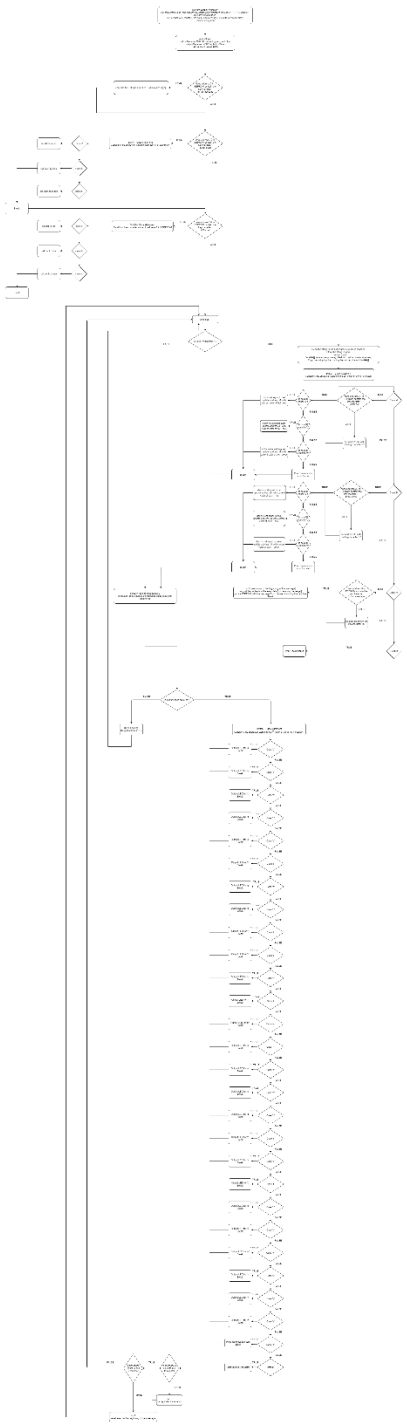
In the void setup() method, the variables are loaded (read) from the EEPROM with the values previously stored in specific memory addresses assigned for those variables. This ensures that the program is able to reuse values after the device has been disconnected.

In the void loop() method, the memory is updated (written to) when new values are inputted in those variables. Conditional statements are used to ensure that updates to EEPROM only occur when there is a change in those variables. Specific addresses in the memory are assigned by the programmer for those variables. The values in these addresses are then read from in the void setup() method.

EEPROM **Course: CSCI 3331**
Lab 3 **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.

EEPROM **Course: CSCI 3331**
Lab 3 **Section: 01**

Ahmed Muntasir Hossain

Code

```
#include <EEPROM.h>

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;                            //dot
int dash;                           //dash

int i = 1;                                  //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

 if (EEPROM.read(0) == 'm')         //if address 0 in EEPROM contains the keycharacter for message 'm'
 {
  for (int i = 0; i < 30; i++)      //first 30 bytes (address 0 - 29) in EEPROM have been assigned for the message inputted by the user
  {
    message[i] = EEPROM.read(i);    //copy the data in EEPROM into message[].
  }
 }

 if (EEPROM.read(30) == 'D')        //if address 30 in EEPROM contains the keycharacter for dash ('D')
 {
```

Ahmed Muntasir Hossain

```
  switch (EEPROM.read(31))          //set the color of dash depending on the color saved in address 31 of EEPROM
  {
   case 'r':
     dash = red;
     break;
   case 'b':
     dash = blue;
     break;
   case 'g':
     dash = green;
     break;
  }
 }

 if (EEPROM.read(32) == 'd')          //if address 32 in EEPROM contains the keycharacter for dot ('d')
 {
  switch (EEPROM.read(33))           //set the color of dot depending on the color saved in address 33 of EEPROM
  {
   case 'r':
     dot = red;
     break;
   case 'b':
     dot = blue;
     break;
   case 'g':
     dot = green;
     break;
  }
 }
}

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
```

Ahmed Muntasir Hossain

```
  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


  switch (data[0])                                //switch-case condition: the character stored in the 0th index of the array
  {
   case 'd':                                      //sets the color of dot
    if (EEPROM.read(32) != 'd')                   //address 32 in EEPROM has been assigned with the keycharacter 'd' for dot
    {
     EEPROM.write(32, 'd');                       //set address 32 with the keycharacter 'd' if it does not already contain the keycharacter
    }


    if (data[1] == 'r')                           //if the second element of the array is 'r', then dot is red
    {
     dot = red;
     EEPROM.write(33, 'r');                       //update address 33 with current color of dot - red
    }
    else if (data[1] == 'g')                      //if the second element of the array is 'g', then dot is green
    {
     dot = green;
     EEPROM.write(33, 'g');                       //update address 33 with current color of dot - green
    }
    else if (data[1] == 'b')                      //if the second element of the array is 'b', then dot is blue
    {
     dot = blue;
     EEPROM.write(33, 'b');                       //update address 33 with current color of 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 (EEPROM.read(30) != 'D')                   //address 30 in EEPROM has been assigned with the keycharacter 'D' for dash
```

Ahmed Muntasir Hossain

```
    {
      EEPROM.write(30, 'D');                          //set address 30 with the keycharacter 'D' if it does not already contain the keycharacter
    }

    if (data[1] == 'r')                               //if the second element of the array is 'r', then dash is red
    {
      dash = red;
      EEPROM.write(31, 'r');                          //update address 31 with current color of dash - red
    }
    else if (data[1] == 'g')                          //if the second element of the array is 'g', then dash is green
    {
      dash = green;
      EEPROM.write(31, 'g');                          //update address 31 with current color of dash - green
    }
    else if (data[1] == 'b')                          //if the second element of the array is 'b', then dash is blue
    {
      dash = blue;
      EEPROM.write(31, 'b');                          //update address 31 with current color of 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

    if (EEPROM.read(0) != 'm')                        //address 0 in EEPROM has been assigned with keycharacter 'm' for message
    {
      EEPROM.write(0, 'm');                           //set address 0 with the keycharacter 'm' if it does not already contain the keycharacter
    }

    for (int i = 1; i < strlen(input.c_str()) + 1; i++)          //update EEPROM with the message from the user including the new line
    {
```

Ahmed Muntasir Hossain

```
      EEPROM.write(i, message[i]);
    }
    break;

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

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;

  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;

  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");
  }

  if (message[i] != '\0')
  {
```

Ahmed Muntasir Hossain

```
    i++;                          //increment counter
   }

   if (message[i] == '\0')              //if the character in message[i] 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);
      digitalWrite(blue, LOW);
      digitalWrite(green, LOW);
      delay(1000);*/
   }
  }
}
```