34338 - Telecommunication Programming Projects with Arduino

Daniel Brasholt s214675

E22

Contents

Exercise 1: Morse

Date: Tuesday 03.01.23

SOS using For Loops

```
// Change LED & the length of a time unit
const byte ledPin = LED_BUILTIN;
const int delayTime = 300;

void setup() {
  // initialize digital pin ledPin as an output.
  pinMode(ledPin, OUTPUT);
}

// Loop running SOS
void loop() {

  // Write S (three dots)
  for (int i = 0; i < 3; i++) {
    digitalWrite(ledPin, LOW);
    delay(delayTime);
    digitalWrite(ledPin, HIGH);
    delay(delayTime);
  }
  // Write additional 2 time units of delay (for a total of 3)
  delay(delayTime * 2);

  // Same as above, except ON for 3 time units
  for (int i = 0; i < 3; i++) {
    digitalWrite(ledPin, LOW);
    delay(delayTime * 3);
    digitalWrite(ledPin, HIGH);
    delay(delayTime);
  }
  delay(delayTime * 2);

  for (int i = 0; i < 3; i++) {
    digitalWrite(ledPin, LOW);
    delay(delayTime);
    digitalWrite(ledPin, HIGH);
    delay(delayTime);
  }
  // Space between words
  delay(delayTime * 6);
}
```

Since only the built-in LED is used, no diagram is provided.

In the exercise guides and hints, `HIGH` is used to turn ON the LED. In my case, however, it has the opposite effect. Therefore, they are all flipped.

Name

```
// Change LED & the length of a time unit
const byte ledPin = LED_BUILTIN;
const int delayTime = 300;

// Function to reduce the amount of repetition
void writeLED(char dotOrDash) {
  if (dotOrDash == '.') {
    digitalWrite(ledPin, LOW);
    delay(delayTime);
    digitalWrite(ledPin, HIGH);
    delay(delayTime);
  } else if (dotOrDash == '-') {
    digitalWrite(ledPin, LOW);
    delay(delayTime * 3);
    digitalWrite(ledPin, HIGH);
    delay(delayTime);
  }
}

void setup() {
  // initialize digital pin ledPin as an output.
  pinMode(ledPin, OUTPUT);
}

// Loop running DANIEL
void loop() {
  // D
  writeLED('-');
  writeLED('.');
  writeLED('.');
  delay(delayTime * 2); // Space between letters

  // A
  writeLED('.');
  writeLED('-');
  delay(delayTime * 2);

  // N
  writeLED('-');
  writeLED('.');
  delay(delayTime * 2);

  // I
  writeLED('.');
  writeLED('.');
  delay(delayTime * 2);

  // E
```

```
  writeLED('.');
  delay(delayTime * 2);

  // L
  writeLED('.');
  writeLED('-');
  writeLED('.');
  writeLED('.');
  delay(delayTime * 6); // Space between words
}
```

Once again, no diagram is provided since only the built-in LED is used.

Questions

1.a

The numbers 0-9 are 10 different values. Therefore, a total of 4 bits would be needed ($2^4 = 16$).

If only 4 bits are used, some of these values would clash with some of the letters using 4 characters/bits - 12 letters use 4 bits to represent them, which would not leave room for 10 numbers.

1.b

This is exactly what the above code does; it uses the built-in LED instead of one of the pins and an “external” LED.

1.c

Since a is doubled each time the loop is run, and the loop is run 5 times, the value of a after the loop is complete would be $a = 2^5 = 32$.

Exercise 2: More LEDs

Date: Tuesday 03.01.23

Traffic light

```
// Change LED & the length of a time unit
const byte red = D1;
const byte yellow = D2;
const byte green = D5;
const int delayTime = 2000;

void redLED(bool on) {
  if (on) {
    digitalWrite(red, HIGH);
```

```
  } else {
    digitalWrite(red, LOW);
  }
}

void yellowLED(bool on) {
  if (on) {
    digitalWrite(yellow, HIGH);
  } else {
    digitalWrite(yellow, LOW);
  }
}

void greenLED(bool on) {
  if (on) {
    digitalWrite(green, HIGH);
  } else {
    digitalWrite(green, LOW);
  }
}


void setup() {
  // Initialize digital pins as an output.
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
  Serial.begin(115200);
}

// Loop running traffic light
void loop() {
  redLED(true);
  yellowLED(false);
  greenLED(false);
  Serial.print("STOP!");
  delay(delayTime);

  redLED(true);
  yellowLED(true);
  greenLED(false);
  Serial.print("GET READY");
  delay(delayTime);

  redLED(false);
  yellowLED(false);
  greenLED(true);
  Serial.print("GO");
  delay(delayTime);
```

```
  redLED(false);
  yellowLED(true);
  greenLED(false);
  Serial.print("BRAKE");
  delay(delayTime);
}
```

A function has been written for each LED that can either turn it off or on (false or true). Then, the different stages of the traffic light are cycled through.

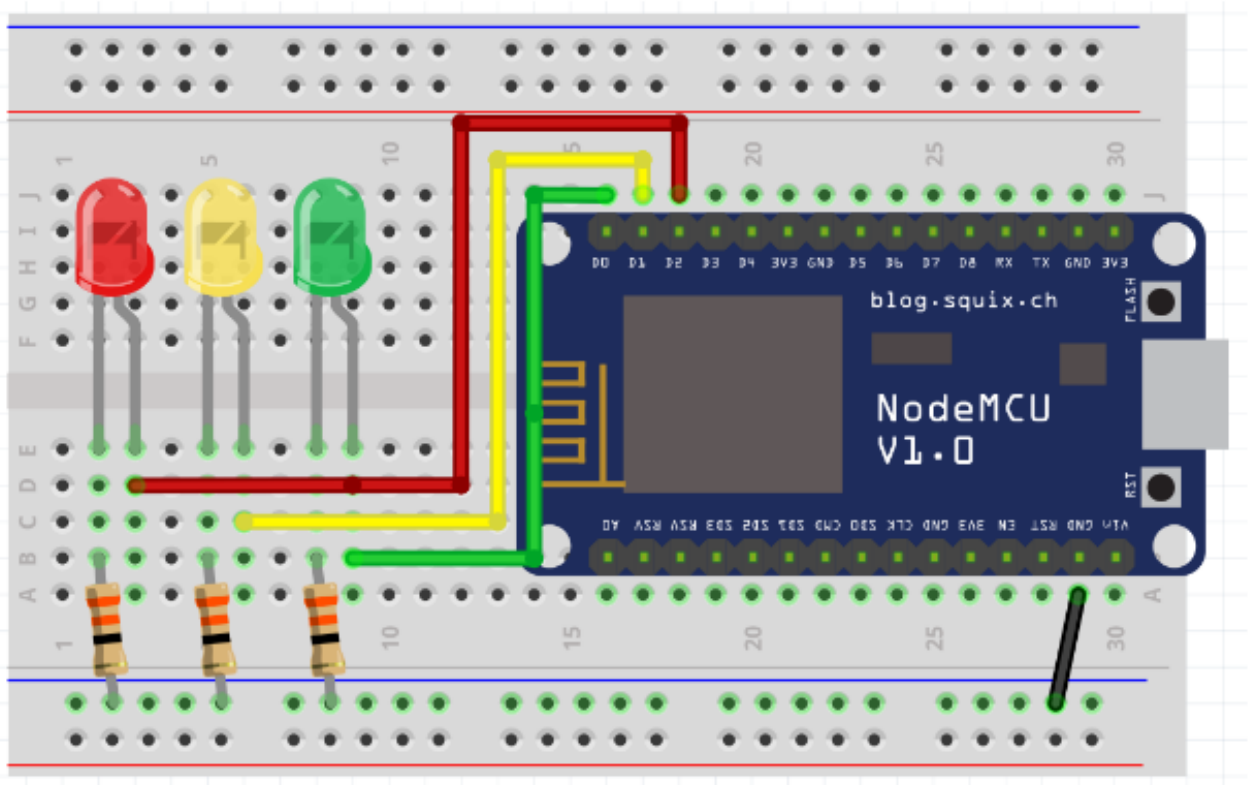

Figure 1: Schematic

I unfortunately forgot to take a picture of the setup before I tore it down.

Binary counter

```
// Change LED & the length of a time unit
const byte red = D1;
const byte yellow = D2;
const byte green = D5;
int count = 0;

void redLED(bool on) {
  if (on) {
    digitalWrite(red, HIGH);
  } else {
    digitalWrite(red, LOW);
```

```
  }
}

void yellowLED(bool on) {
  if (on) {
    digitalWrite(yellow, HIGH);
  } else {
    digitalWrite(yellow, LOW);
  }
}

void greenLED(bool on) {
  if (on) {
    digitalWrite(green, HIGH);
  } else {
    digitalWrite(green, LOW);
  }
}


void setup() {
  // initialize digital pins as outputs.
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
  Serial.begin(115200);
}

// Loop running counter
void loop() {
  count++;
  Serial.print(count);
  if (count % 2 == 1) {
    greenLED(true);
  } else {greenLED(false);}
  if (count % 4 > 1) {
    yellowLED(true);
  }else {yellowLED(false);}
  if (count % 8 > 3) {
    redLED(true);
  } else {redLED(false);}
  if (count == 16) {
    count = 0;
  }
  delay(500);
}
```

If count%2 is 1, the number is uneven and the least significant LED should turn on. If the next to last (yellow) LED should be on, count%4 should yield either 2 or 3, which is greater than 1. If the red LED should be on, count%8 should yield

greater than 3.

If the count has reached 16, it is reset to 0.

The schematic for this counter is exactly the same as shown above.

Questions

2.a

% in C is the mod operator. It gives the remainder of a division. 42%5 is therefore 2, since 40 is divisible by 5 which leaves a remainder of 2.

2.b

As explained below the code for the binary counter.

2.c

There was no specification sheet for the LEDs used, however assuming a forward voltage of 2.1V and the resistors, which were 220 ohms, the LEDs will have a forward current of about 10 mA.

Exercise 3: Digital Input

Date: Tuesday 03.01.23

Non-latching

```
const byte led = D1;
const byte button = D2;
int buttonInput;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(button, INPUT_PULLUP);

}

void loop() {
  // The input is NOT-ed to obtain the desired functionality
  buttonInput = !digitalRead(button);
  digitalWrite(led, buttonInput);

}
```

This code turns on the LED when the button is being held down. The LED is turned off when the button is let go.

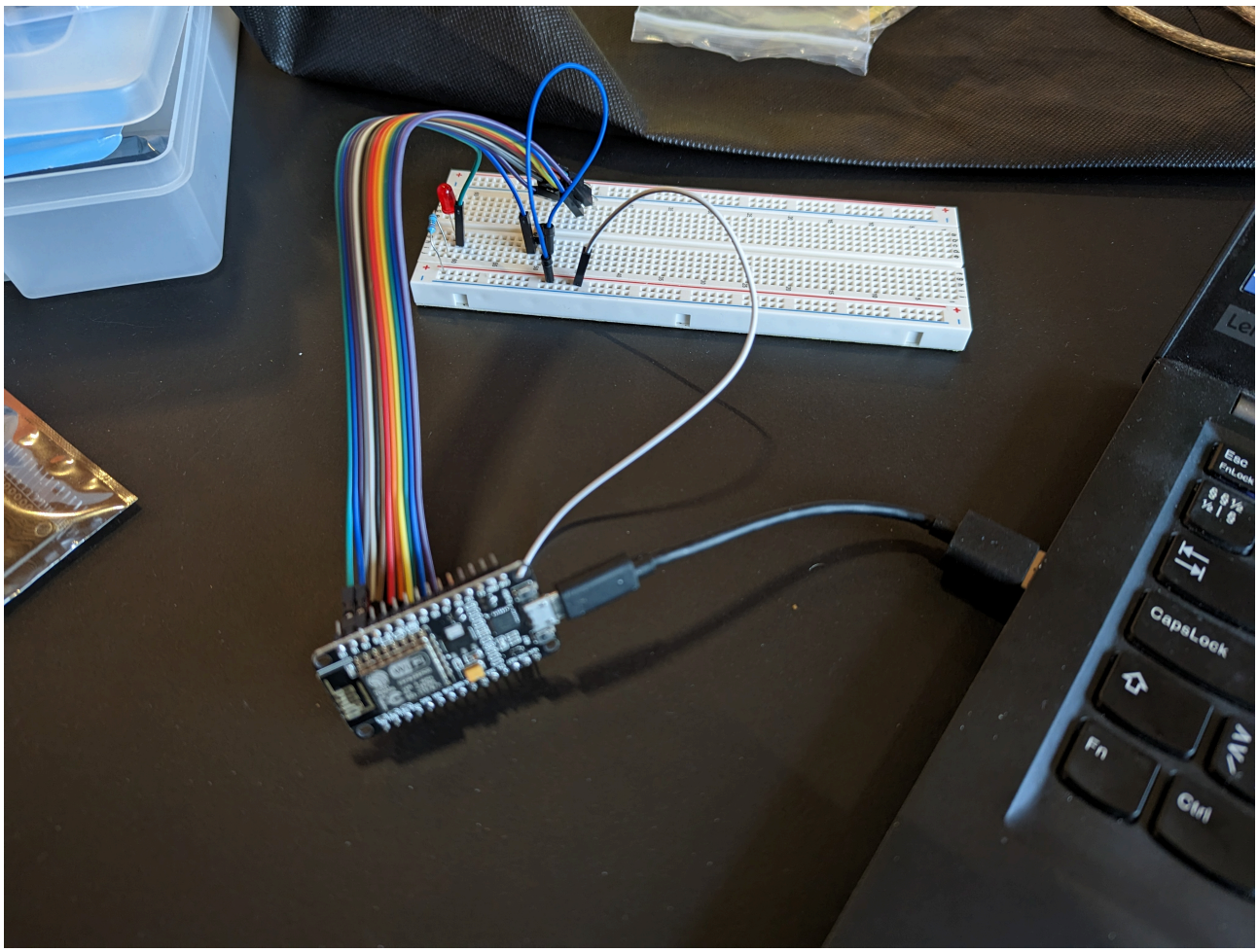

Figure 2: Schematic

Questions

3.a

The PULLUP part connects the button using a PULLUP-resistor. This makes sure that the button does not return random values when not being pressed. If this is not included, the LED does not behave as expected.

3.b

Since this part is run in the `LOOP` function, the code constantly checks (as often as possible) whether the button is pushed or not. It would then be possible to press and unpress the buttons thousands of times a second, which may be overkill.

3.c

The `!` is the NOT operator. If this is not used in the code, pressing the button will turn OFF the LED while leaving the button unpressed turns ON the LED.

Exercise 4: Fritzing

Date: Tuesday 03.01.23

Drawing

Following image shows a schematic of the circuit used to complete exercise 3:

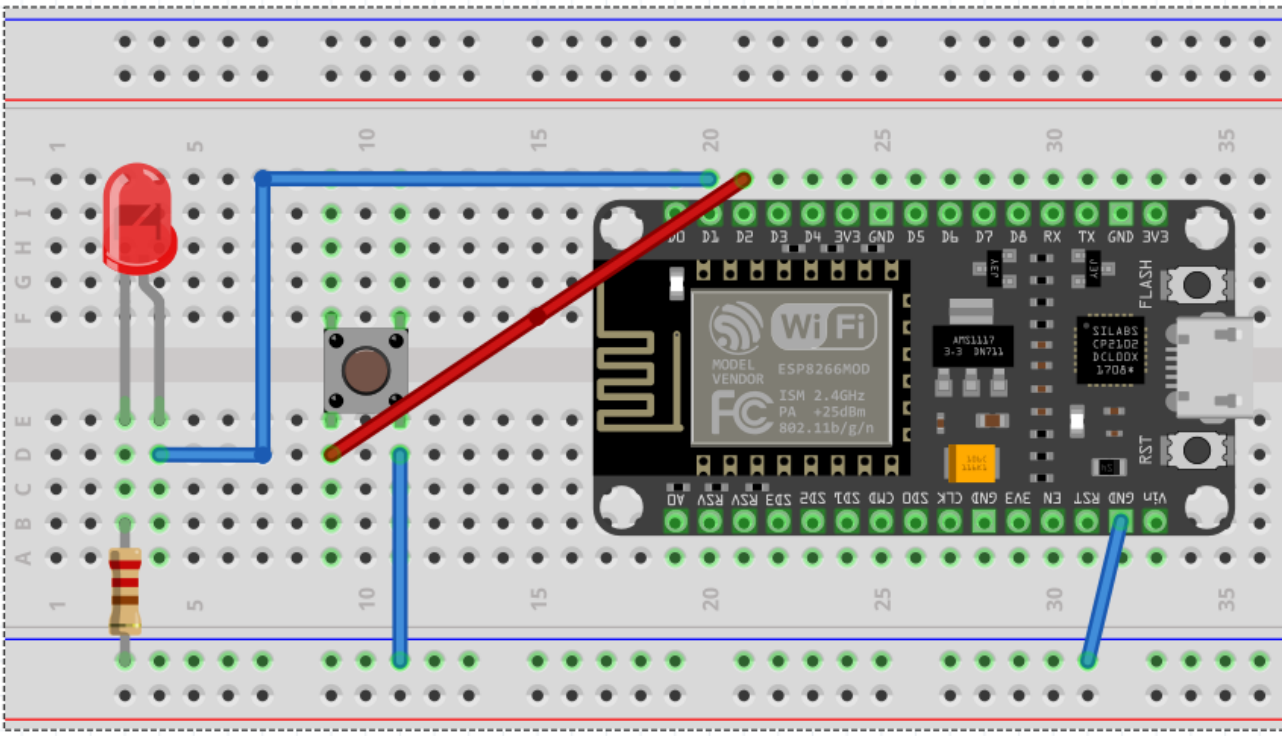

Figure 3: Exercise 3 schematic

Questions

4.a

Without any documentation, perhaps. However, a schematic like this or a picture of the circuit would be a great help.

Exercise 5: Serial Monitor

Date: Wednesday 04.01.23

Code

```
int incomingByte = 0;

void setup() {
  Serial.begin(115200);
```

```
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    Serial.print("I received: ");
    Serial.print(incomingByte, DEC);
  }
}
```

Questions

5.a

Because the `Serial.print()` call converts the incoming byte to a decimal value, a G becomes 71 and A becomes 65. This is because the character is treated as an ASCII-value where A corresponds to 65.

5.b

If the monitor sends a line ending, it will actually send two values instead of just one; one for the character (e.g. A) and one for the line ending - 10 for “New Line” and 13 for “Carriage Return”.

5.c

`(char)` will type-cast the incoming value to the character type. Thus, when printed, it will show the actual character instead of the corresponding ASCII-value. Sending a new line will now also actually send a new line to the console.

Exercise 6: Read from Serial Monitor

Date: Wednesday 04.01.23

Code

```
byte led1 = D1;
byte led2 = D2;
byte led3 = D4;
byte led4 = D7;
byte led5 = D8;
int incomingByte;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
```

```
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  Serial.begin(115200);

}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    switch ((char)incomingByte) {
      case 'a':
        digitalWrite(led1, HIGH);
        break;
      case 'b':
        digitalWrite(led2, HIGH);
        break;
      case 'c':
        digitalWrite(led3, HIGH);
        break;
      case 'd':
        digitalWrite(led4, HIGH);
        break;
      case 'e':
        digitalWrite(led5, HIGH);
        break;
      default:
        digitalWrite(led1, LOW);
        digitalWrite(led2, LOW);
        digitalWrite(led3, LOW);
        digitalWrite(led4, LOW);
        digitalWrite(led5, LOW);
    }
  }

}
```

The schematic for this circuit is the same as shown in the exercise, except that some of the pins are switched; this code uses D1, D2, D4, D7, and D8.

Questions

6.a

The `char` is a data type in C which stores the value of a single character. This is represented in memory has one byte, 8 bits, which means that a total of 256 different chars are possible.

6.b

First, `mychar` is the ASCII-value of ‘4’. Then, `val` receives the value of `'4'-'0'`, which corresponds to 4. `4+'A'-1` then yields the number 68, which corresponds to the char `D`.

Exercise 7: Reading an ASCII-encoded string

Date: Wednesday 04.01.23

Code

```
byte Rled = D1;
byte Gled = D2;
byte Bled = D4;
int R,G,B;

void setup() {
  pinMode(Rled, OUTPUT);
  pinMode(Gled, OUTPUT);
  pinMode(Bled, OUTPUT);
  Serial.begin(115200);

}

void loop() {
  if (Serial.available() > 0) {
    R = Serial.parseInt();
    G = Serial.parseInt();
    B = Serial.parseInt();
    Serial.printf("R: %d, G: %d, B: %d", R, G, B);
    analogWrite(Rled, R);
    analogWrite(Gled, G);
    analogWrite(Bled, B);
  }
}
```

This code will control the colour of the LED and log the RGB values to the serial output. The schematic used is the same as shown in the exercise, with only some pins swapped since the RGB LED had a different pinout.

Questions

7.a

An RGB value is a triplet of numbers ranging from 0 to 255 - e.g. (255,47,0). The first number represents the amount of red, the second the amount of blue, and the third the amount of green present in the colour. White will then be (255,255,255), black (0,0,0), and purple (255,0,255).

7.b

`Serial.parseInt()` will look for the next valid integer in the incoming serial. It will, by default, skip all non-integers in the incoming data, which means that 3 consecutive calls to `Serial.parseInt()` will recognize the 3 numbers in “255,0,47” and ignore the “,”.

Exercise 8: Analog Input

Date: Wednesday 04.01.23

Code

```
byte Rled = D1;
byte Gled = D2;
byte Bled = D4;
byte potentiometer = A0;
int R,G,B, sensorValue, mappedValue;

void setup()
{
  pinMode(Rled, OUTPUT);
  pinMode(Gled, OUTPUT);
  pinMode(Bled, OUTPUT);
  Serial.begin(115200);
}
void loop()
{
  sensorValue = analogRead(potentiometer);
  mappedValue = map(sensorValue, 0, 1024, 0, 1280);
  Serial.print(sensorValue);

  if (mappedValue < 512) {
    R = 255;
    G = mappedValue / 2;
    B = 0;
  }
  else if (mappedValue < 768) {
    R = 768 - mappedValue;
    G = 255;
    B = 0;
  }
  else if (mappedValue < 1024) {
    R = 0;
    G = 1024 - mappedValue;
    B = mappedValue - 768;
  }
  else {
    R = mappedValue - 1024;
```

```
    G = 0;
    B = 255;
  }

  analogWrite(Rled, R);
  analogWrite(Gled, G);
  analogWrite(Bled, B);
}
```

The code turns the LED violet when the potentiometer is at full voltage, and red when it is at $0V$ (this is opposite of the given exercise, however I flipped it in my head, and I did not want to rewrite the logic).

Schematic

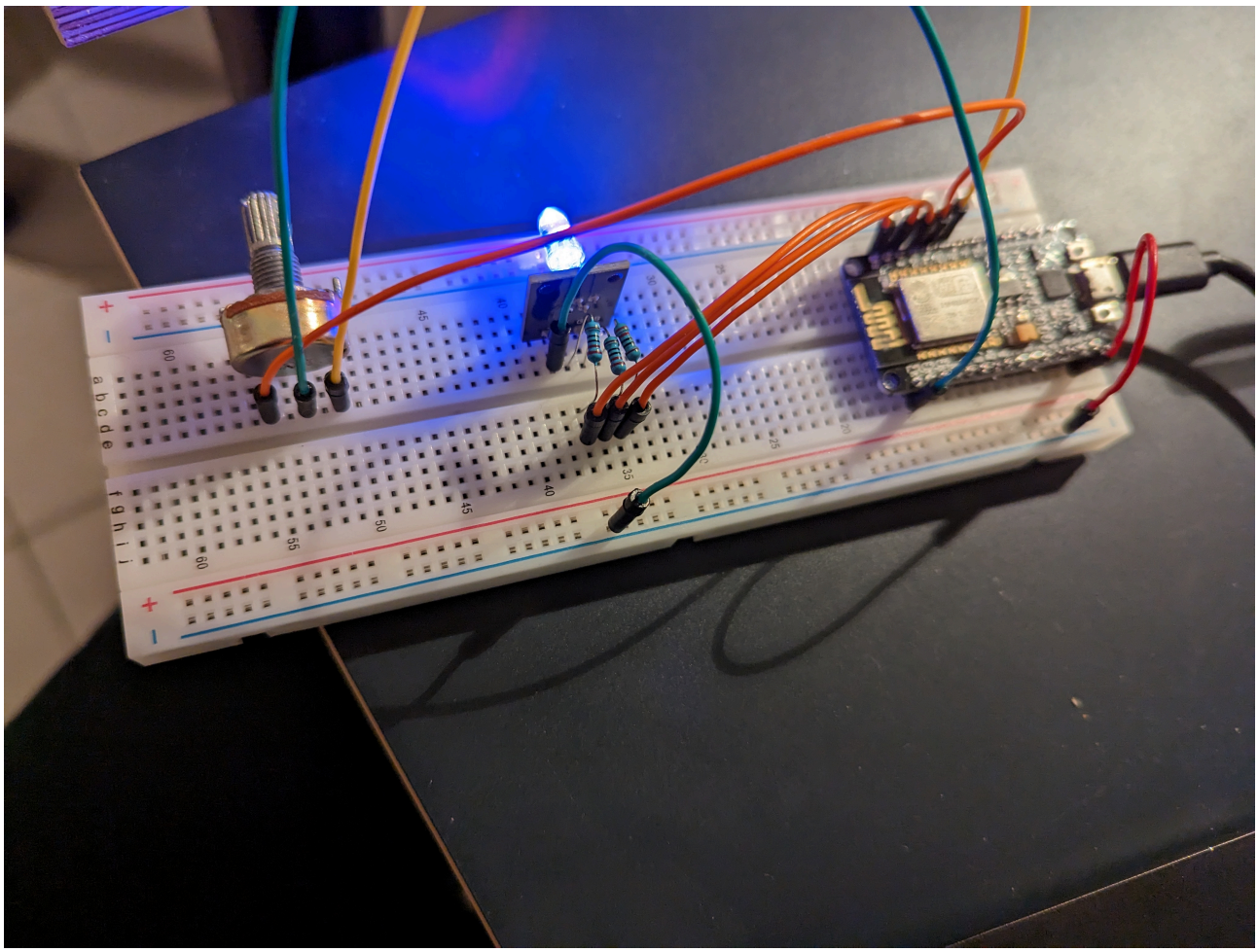

Figure 4: Circuit

Questions

8.a

The analog value is changed to a digital signal with 10 bits of precision, which gives a number between 0 and 1024.

8.b

Since the potentiometer is connected to 3.3V, this is the maximum that can be read on A0. This is then translated to 1024 by the MCU.

Exercise 9: Temperature Sensor

Code

```
const byte ANALOG_PIN = A0;
int vin;
float voltage, temperature;

void setup() {
  Serial.begin(115200);

}

void loop() {
  vin = analogRead(ANALOG_PIN);
  // Convert incoming signal to voltage
  voltage = vin * (5000 / 1024);
  // 250mV at 25C, so divide by 10
  temperature = voltage / 10;
  Serial.print(temperature);
  Serial.println("°");
}
```

The schematic used is excactly as shown in the exercise, except that the voltage is supplied from Vin instead of 3.3V.

Questions

9.a

Firstly, the conversion is to milivolts, not volts. Secondly, it assumes a 5V supply, which is not correct when using the TMP36.

9.b

On the first loop, `c` is `0 + 0`, so the char `0`. Then, it is the char `2`, then `4`, and finally `6`. This is then printed with a degree symbol.

9.c

`Serial.write()` simply writes binary to Serial. `Serial.print()` sends data in a human-readable format, optionally with an End-of-Line character.

Exercise 10: Make diodes light up depending on temperature

Code

```
const byte ANALOG_PIN = A0;
const byte GPIN = D1;
const byte YPIN = D2;
const byte RPIN = D4;
int vin;
float voltage, temperature;

void setup() {
  Serial.begin(115200);
  pinMode(GPIN, OUTPUT);
  pinMode(YPIN, OUTPUT);
  pinMode(RPIN, OUTPUT);

}

void loop() {
  vin = analogRead(ANALOG_PIN);
  // Convert incoming signal to voltage
  voltage = vin * (5000 / 1024);
  // 250mV at 25C, so divide by 10
  temperature = voltage / 10;
  Serial.print(temperature);
  Serial.println("°");
  // Turn on correct LED depending on temperature
  if (temperature < 28) {
    digitalWrite(GPIN, HIGH);
    digitalWrite(YPIN, LOW);
    digitalWrite(RPIN, LOW);
  }
  else if (temperature < 30) {
    digitalWrite(GPIN, LOW);
    digitalWrite(YPIN, HIGH);
    digitalWrite(RPIN, LOW);
  }
  else {
    digitalWrite(GPIN, LOW);
    digitalWrite(YPIN, LOW);
    digitalWrite(RPIN, HIGH);
  }
}
```

Note: obviously 28 and 30 degrees are way too big, however the temperature sensor noted about 28 degrees as a living room temperature instead of 22. This should not change the logic of the code.

Schematic

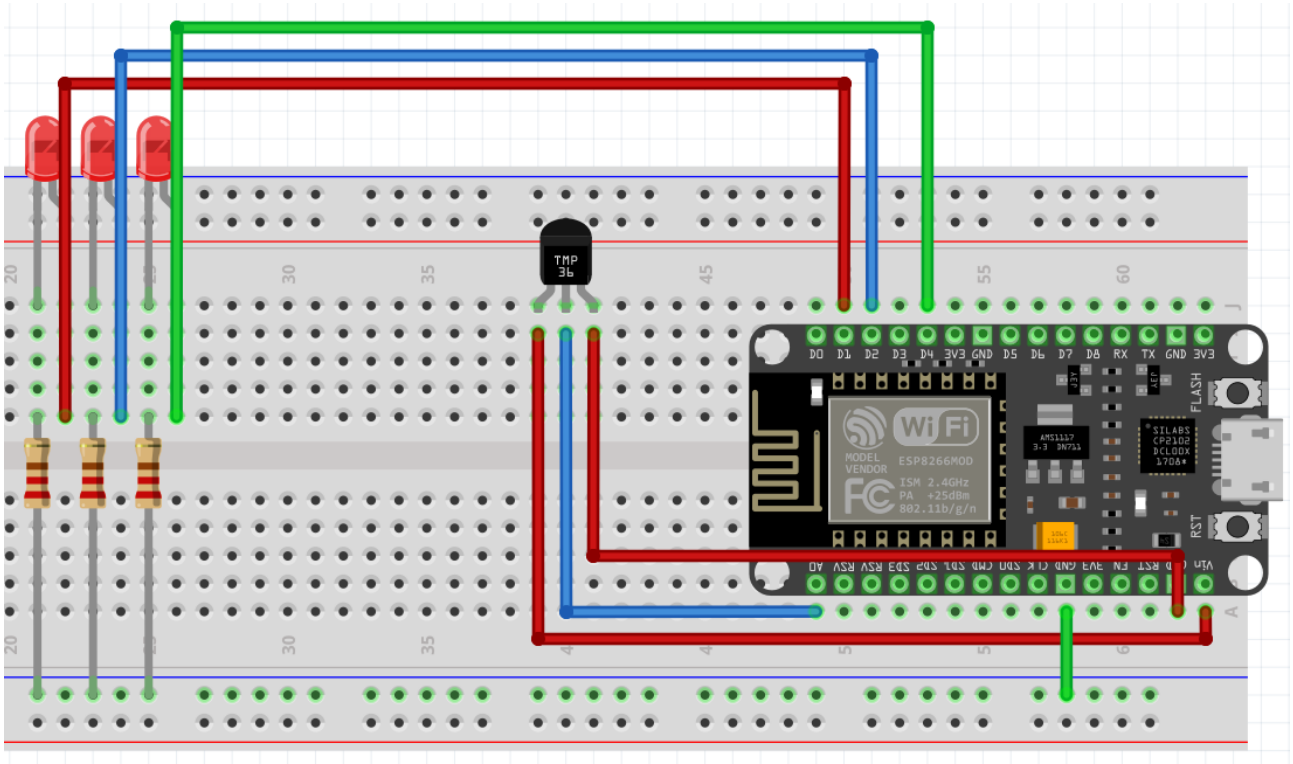

Figure 5: Schematic

Exercise 11: Output Temperature to LCD Display

Code

```
##include <LiquidCrystal_I2C.h>

int vin;
float temperature;

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
  // Initialize the LCD
  lcd.init();
  lcd.backlight();
}

void loop()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  vin = analogRead(A0);
```

```
  temperature = vin * 500 / 1024;
  // Weird errors were displayed if it was not first converted to string
  String tempString = "";
  tempString.concat(temperature);
  lcd.print(tempString);
  // Print additional !!! if temp is too high
  if (temperature > 30) {
    lcd.print("!!!");
  }
  // Do not update constantly
  delay(500);
}
```

The schematic is just a combination of the figure shown and the schematic from exercise 9.

Questions

11.a

I2C is a packet-switched serial communication bus often used in short-distance communication between some integrated circuit and a micro processor.

11.b

Power can be saved by not constantly writing to the LCD and only writing when a change is detected. It is also often not necessary to show updated constantly, so it is possible to only update every x seconds.

Exercise 14: Webserver

Date: Monday 09.01.23

Code

```
##include <ESP8266WiFi.h>
##include <WiFiClient.h>
##include <ESP8266WiFiMulti.h>   // Include the Wi-Fi-Multi library
##include <ESP8266WebServer.h>   // Include the WebServer library
##include <ESP8266mDNS.h>        // Include the mDNS library

ESP8266WiFiMulti wifiMulti;
// Create an instance of the server
ESP8266WebServer server(80);

// Can be changed to LED_BUILTIN
const int led = D2;
```

```
void handleRoot();
void handleLED();
void handleNotFound();

void setup() {
   Serial.begin(115200);
  delay(10);

  pinMode(led, OUTPUT);
  digitalWrite(led,1);

  // Connect to WiFi network
  Serial.println();
  wifiMulti.addAP("P7", "dtuiot!!");  // add Wi-Fi networks you want to connect
  wifiMulti.addAP("<ssid2>", "<password>");

  Serial.println();
  Serial.print("Connecting ...");
  //WiFi.begin(ssid, password);

  while (wifiMulti.run() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected to ");
  Serial.println(WiFi.SSID());
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin("iot")) {              // Start the mDNS responder for esp8266
    Serial.println("mDNS responder started");
  } else {
    Serial.println("Error setting up MDNS responder!");
  }



  server.on("/", HTTP_GET, handleRoot);
  server.on("/LED", HTTP_POST, handleLED);
  server.onNotFound(handleNotFound);

  // Start the server
  server.begin();
  Serial.println("Server started");
}

void loop() {
  // Check if a client has connected
  server.handleClient();
```

```
}

void handleRoot() {                          // When URI / is requested, send a
  server.send(200, "text/html", "<html><title>Internet of Things - Demonstratio
      </head><body><h1>Velkommen til denne WebServer</h1> \
      <p>Internet of Things (IoT) er \"tingenes Internet\" - dagligdags ting ko
      <p>Her kommunikerer du med en webserver på en lille microcontroller af ty
      <p>Klik på nedenstående knap for at tænde eller slukke LED på port D2</p>
      <form action=\"/LED\" method=\"POST\" ><input type=\"submit\" value=\"Ski
      <p>Med en Arduino ESP8266 kan du lave et have a sjove projekter</p> \
      <p>Vil du vide mere: Kig på hjemmesiden for uddannelsen : <a href=\"www.d
      </body></html>");
}

void handleLED() {                           // If a POST request is made to URI
  digitalWrite(led,!digitalRead(led));       // Change the state of the LED
  server.sendHeader("Location","/");         // Add a header to respond with a n
  server.send(303);                          // Send it back to the browser with
}

void handleNotFound(){
  server.send(404, "text/plain", "404: Not found"); // Send HTTP status 404 (No
}
```

Questions

14.a

A HTTP GET request is a request for information. A POST-request instead requests a change in information at the server.

14.b

A server is a computer that you can connect to for some purpose, e.g. a webserver that serves a web page or a file server that serves files.

Exercise 15: ThingSpeak

Code

```
##include <ESP8266WiFi.h>
##include <ThingSpeak.h>
##include <DHT.h>

##define DHTTYPE DHT11
##define DHTPIN D2

const char* ssid = "P7";
```

```
const char* pass = "dtuiot!!";

WiFiClient client;
DHT dht(DHTPIN, DHTTYPE);

unsigned long channelID = 2003095; //your TS channal
const char * APIKey = "UZJDYB9WDGCTCL3T"; //your TS API
const char* server = "api.thingspeak.com";
const int postDelay = 20 * 1000; //post data every 20 seconds

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, pass);
}

float data; //measured data
float dataTemp, dataHum;

void loop() {
  dataTemp = dht.readTemperature();
  dataHum = dht.readHumidity();
  Serial.println(dataTemp);
  Serial.println(dataHum);
  ThingSpeak.begin(client);
  client.connect(server, 80); //connect(URL, Port)
  ThingSpeak.setField(2, dataTemp); //set data on the X graph
  ThingSpeak.setField(3, dataHum);
  ThingSpeak.writeFields(channelID, APIKey);//post everything to TS
  client.stop();
  delay(postDelay); //wait and then post again
}
```

This will read data from a DHT11 temperature and humidity sensor.

Graph

This graph has been created on ThingSpeak:

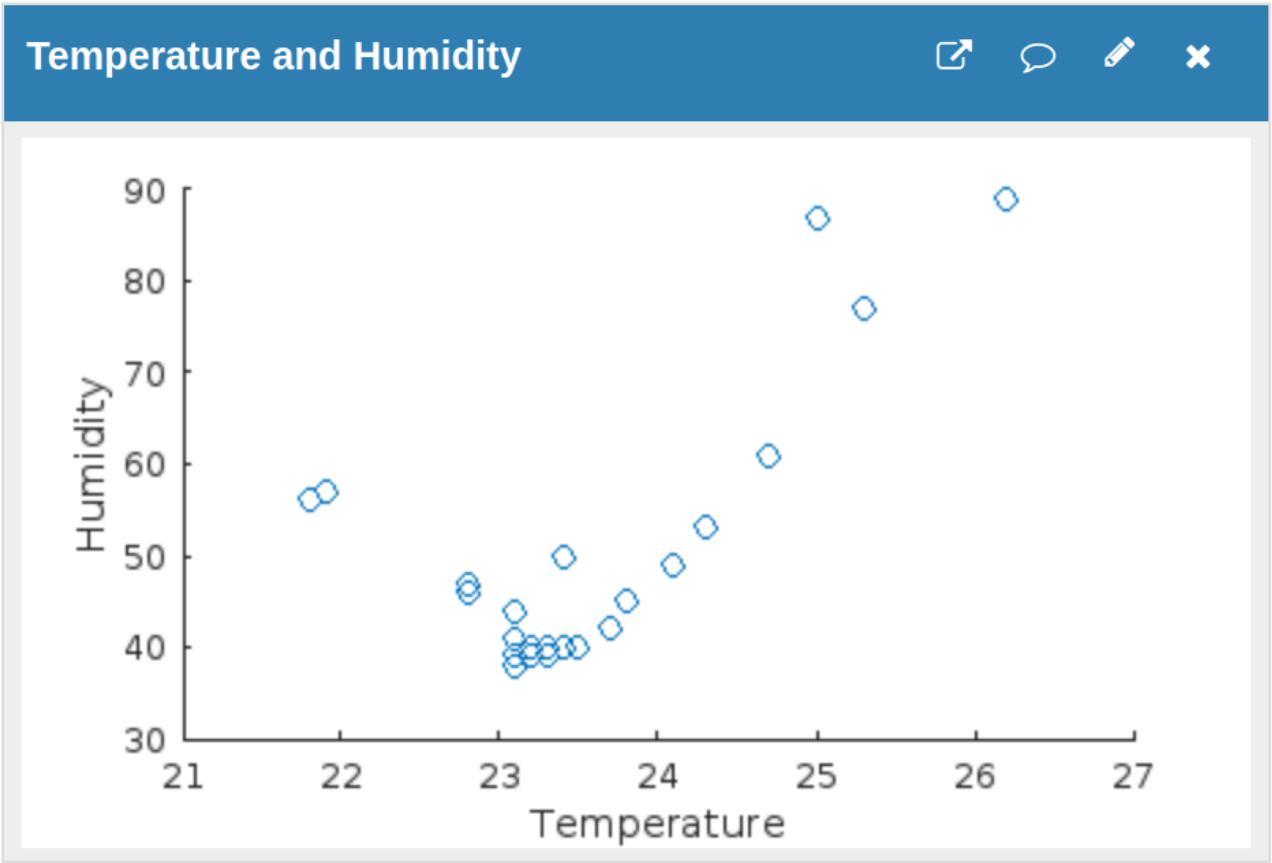
Temperature and Humidity
90
80
70
60
50
40
30
Humidity
21
22
23
24
25
26
27
Temperature

Figure 6: