

# Telecommunication Programming Projects with Arduino - Exercises

s214718, s224027

January 2025

[Link](#) [Link](#) to our github repository!!.

## Exercise 5

### 5a

Defaults the `incomingByte` variable. The `setup()` starts serial communication with baud rate of 9600. The `loop()` has an if-statement that checks how many bytes (characters) is available for reading from the serial port, if this is more than zero (data is available), we read the incoming serial data with `Serial.read()` and store it in `incomingByte`.

To output the data the program prints out the message "I received: " to the serial monitor, along with the ASCII decimal value of the ASCII character.

### 5b-5d



```
Exercise5.ino
1  int incomingByte = 0;
2
3  void setup() {
4    Serial.begin(115200);
5  }
6
7  void loop() {
8    if (Serial.available() > 0) {
9      incomingByte = Serial.read();
10     Serial.print("I received: ");
11     Serial.println(incomingByte, DEC);
12   }
13 }
14
```

Output Serial Monitor x

Message (Enter to send message to 'Arduino Uno' on 'COM3')

I received: 71

Figure 1: Screenshot of output from the given Code

We don't the character G we wrote, and instead get the ASCII decimal value because that is what the code prints.

### 5e

Then the line ending character is seen as an additional input, and has its own ASCII value. (newline) prints out 10.

## 5f

Program converts the incomingByte back to its character representation, and prints the character itself instead of its ASCII value.

## Exercise 6

### 6a

The `char` type is a type capable of representing a character symbol. A char usually requires 8 bits of memory, thus capable of representing a total of  $2^8 = 256$  different characters.

### 6b

The `char` variable `mychar` is initiated to the character '4' which has the index 52, that is, it is the 52nd representable character of the `char` type. Subtracting a character from a `char` variable is equivalent to subtracting the index of the character from the index of the `char` variable. Thus subtracting '0' with index 48 from '4' with index 52 yields the character 'EOT' with index 4. Similarly, if we further add the character 'A' with index 65 and subtract 1 we get the character 'D' with index  $4 + 65 - 1 = 68$ . Hence `mychar='D'`.

### 6c

[See the video in videos.zip]

## Exercise 7

### 7a

RGB value is used to represent the intensity of red, green and blue. The interval 0-255 is because each color is stored in 8 bits.

### 7b

Looks for the next valid integer in the incoming serial input. The function terminates if it times out, or when a non-digit is read.

### 7c-7d

The two figures show the setup of the program and the circuit. The first figure shows a screenshot of the program code used to read and parse the serial input. The second figure shows the output of the RGB LED, which glows purple when the values "200,100,40" are entered.

```

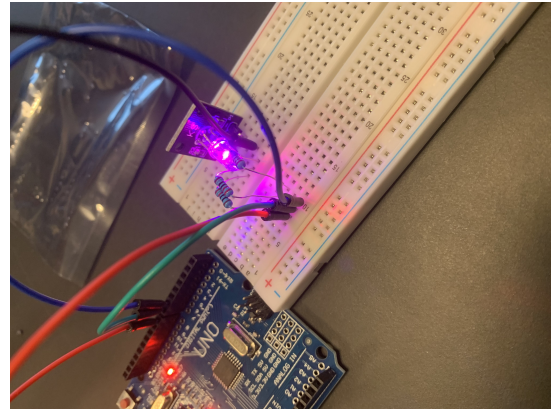
Exercise7.ino
20
21 if (Serial.available() > 0) {
22   // Read and parse the three integers from the input string
23   int redValue = Serial.parseInt(); // Parse the first integer
24   int greenValue = Serial.parseInt(); // Parse the second integer
25   int blueValue = Serial.parseInt(); // Parse the third integer
26
27   // Constrain values to 0-255 to avoid invalid input
28   redValue = constrain(redValue, 0, 255);
29   greenValue = constrain(greenValue, 0, 255);
30   blueValue = constrain(blueValue, 0, 255);
31
32   // Write the values to the RGB LED
33   analogWrite(redPin, redValue);
34   analogWrite(greenPin, greenValue);
35   analogWrite(bluePin, blueValue);
36
37   // Debugging output
38   Serial.print("Red: ");
39   Serial.print(redValue);
40   Serial.print(", Green: ");
41   Serial.print(greenValue);
42   Serial.print(", Blue: ");
43   Serial.println(blueValue);
44 }

```

Output Serial Monitor x

200,100,40

Red: 200, Green: 100, Blue: 40



(a) Screenshot of program

(b) Picture of RBG LED output

Figure 2: Side by side images: Program output and Circuit output. The input values were 200,100,40, as given in the exercise.

## Exercise 8

### 8a-8b

An ADC is used by A0 to read the analog values, and this is represented with a 10-bit resolution, meaning it can take values between 0 (0V) and 1023 (5V or 3.3V depending on the reference, in our case it's 5V ADC). If the input voltage exceeds this, the reading will be capped at 1023 (the maximum digital value).

### 8c-8d

```

Exercise8.ino
24
25 // Read the potentiometer value (0 to 1023)
26 int portValue = analogRead(potPin);
27
28 // Convert the potentiometer value to a voltage (0V to 5V)
29 float voltage = (portValue / 1023.0) * 5.0;
30
31 // Print the voltage to the serial monitor with 3 decimals
32 Serial.print("Voltage: ");
33 Serial.println(voltage, 3); // Print the voltage with 1 decimal place
34
35 // Map the potentiometer value to a range from 0 to 255
36 int ledIntensity = map(voltage, 0, 5, 20, 255);
37
38 // map() function will convert this value into a new integer, but scaled to fit between 0
39
40 // Fade the RGB LED color from purple (red + blue) to red
41 if (voltage == 0) {
42   // Purple: Both red and blue are fully on
43   analogWrite(redPin, 255);
44   analogWrite(greenPin, 0);
45   analogWrite(bluePin, 255);
46 } else if (voltage > 3.2) {
47   // Red: Only red is fully on
48   analogWrite(redPin, 255);
49   analogWrite(greenPin, 0);
50   analogWrite(bluePin, 0);
51 }
52
53 // Fade the RGB LED color from purple (red + blue) to red
54 if (voltage == 0) {
55   // Purple: Both red and blue are fully on
56   analogWrite(redPin, 255);
57   analogWrite(greenPin, 0);
58   analogWrite(bluePin, 255);
59 } else if (voltage > 3.2) {
60   // Red: Only red is fully on
61   analogWrite(redPin, 255);
62   analogWrite(greenPin, 0);
63   analogWrite(bluePin, 0);
64 }
65
66 // Fade the RGB LED color from purple (red + blue) to red
67 if (voltage == 0) {
68   // Purple: Both red and blue are fully on
69   analogWrite(redPin, 255);
70   analogWrite(greenPin, 0);
71   analogWrite(bluePin, 255);
72 } else if (voltage > 3.2) {
73   // Red: Only red is fully on
74   analogWrite(redPin, 255);
75   analogWrite(greenPin, 0);
76   analogWrite(bluePin, 0);
77 }
78
79 // Fade the RGB LED color from purple (red + blue) to red
80 if (voltage == 0) {
81   // Purple: Both red and blue are fully on
82   analogWrite(redPin, 255);
83   analogWrite(greenPin, 0);
84   analogWrite(bluePin, 255);
85 } else if (voltage > 3.2) {
86   // Red: Only red is fully on
87   analogWrite(redPin, 255);
88   analogWrite(greenPin, 0);
89   analogWrite(bluePin, 0);
90 }
91
92 // Fade the RGB LED color from purple (red + blue) to red
93 if (voltage == 0) {
94   // Purple: Both red and blue are fully on
95   analogWrite(redPin, 255);
96   analogWrite(greenPin, 0);
97   analogWrite(bluePin, 255);
98 } else if (voltage > 3.2) {
99   // Red: Only red is fully on
100  analogWrite(redPin, 255);
101  analogWrite(greenPin, 0);
102  analogWrite(bluePin, 0);
103 }

```

Output Serial Monitor x

Message (Enter to send message to 'Arduino Uno' on 'COM3')

Voltage: 0.000  
Voltage: 0.000  
Voltage: 0.000  
Voltage: 0.000

```

Exercise8.ino
24
25 // Read the potentiometer value (0 to 1023)
26 int portValue = analogRead(potPin);
27
28 // Convert the potentiometer value to a voltage (0V to 5V)
29 float voltage = (portValue / 1023.0) * 5.0;
30
31 // Print the voltage to the serial monitor with 3 decimals
32 Serial.print("Voltage: ");
33 Serial.println(voltage, 3); // Print the voltage with 1 decimal place
34
35 // Map the potentiometer value to a range from 0 to 255
36 int ledIntensity = map(voltage, 0, 5, 20, 255);
37
38 // map() function will convert this value into a new integer, but scaled to fit between 0
39
40 // Fade the RGB LED color from purple (red + blue) to red
41 if (voltage == 0) {
42   // Purple: Both red and blue are fully on
43   analogWrite(redPin, 255);
44   analogWrite(greenPin, 0);
45   analogWrite(bluePin, 255);
46 } else if (voltage > 3.2) {
47   // Red: Only red is fully on
48   analogWrite(redPin, 255);
49   analogWrite(greenPin, 0);
50   analogWrite(bluePin, 0);
51 }
52
53 // Fade the RGB LED color from purple (red + blue) to red
54 if (voltage == 0) {
55   // Purple: Both red and blue are fully on
56   analogWrite(redPin, 255);
57   analogWrite(greenPin, 0);
58   analogWrite(bluePin, 255);
59 } else if (voltage > 3.2) {
60   // Red: Only red is fully on
61   analogWrite(redPin, 255);
62   analogWrite(greenPin, 0);
63   analogWrite(bluePin, 0);
64 }
65
66 // Fade the RGB LED color from purple (red + blue) to red
67 if (voltage == 0) {
68   // Purple: Both red and blue are fully on
69   analogWrite(redPin, 255);
70   analogWrite(greenPin, 0);
71   analogWrite(bluePin, 255);
72 } else if (voltage > 3.2) {
73   // Red: Only red is fully on
74   analogWrite(redPin, 255);
75   analogWrite(greenPin, 0);
76   analogWrite(bluePin, 0);
77 }
78
79 // Fade the RGB LED color from purple (red + blue) to red
80 if (voltage == 0) {
81   // Purple: Both red and blue are fully on
82   analogWrite(redPin, 255);
83   analogWrite(greenPin, 0);
84   analogWrite(bluePin, 255);
85 } else if (voltage > 3.2) {
86   // Red: Only red is fully on
87   analogWrite(redPin, 255);
88   analogWrite(greenPin, 0);
89   analogWrite(bluePin, 0);
90 }
91
92 // Fade the RGB LED color from purple (red + blue) to red
93 if (voltage == 0) {
94   // Purple: Both red and blue are fully on
95   analogWrite(redPin, 255);
96   analogWrite(greenPin, 0);
97   analogWrite(bluePin, 255);
98 } else if (voltage > 3.2) {
99   // Red: Only red is fully on
100  analogWrite(redPin, 255);
101  analogWrite(greenPin, 0);
102  analogWrite(bluePin, 0);
103 }

```

Output Serial Monitor x

Message (Enter to send message to 'Arduino Uno' on 'COM3')

Voltage: 3.226  
Voltage: 3.226  
Voltage: 3.226  
Voltage: 3.226  
Voltage: 3.226

Figure 3: The voltage readings from the potentiometer, first by not moving it and the second by completely turning it around.

The voltage doesn't quite reach 3.3V despite having it completely turned, and could be because the reference for the ADC isn't set at 3.3V, which could lead to some mapping issues.

### 8e

Given a light intensity  $i \in \{0, \dots, 255\}$ , We can fade the RGB LED from purple (R:  $i$ , G: 0, B:  $i$ ) to red (R:  $i$ , G: 0, B: 0) by shifting the blue value from  $i$  to 0. This can be achieved with a function  $f: [0, 3.3] \rightarrow \{0, \dots, i\}$  defined as

$$f(v) = \lceil i(1 - v/3.3) \rceil,$$

```

ex8.ino
2  const uint8_t PIN_RED = 9;
3  const uint8_t PIN_GREEN = 5;
4  const uint8_t PIN_BLUE = 6;
5
6  int intensity = 30;
7
8  void setup() {
9      // put your setup code here, to run once:
10
11     Serial.begin(115200);
12
13     pinMode(A0, INPUT);
14     pinMode(PIN_RED, OUTPUT);
15     pinMode(PIN_GREEN, OUTPUT);
16     pinMode(PIN_BLUE, OUTPUT);
17     analogWrite(PIN_RED, intensity);
18     analogWrite(PIN_GREEN, 0);
19 }
20
21 int blueValue(float volt) {
22     return (int)(intensity*(1.0-volt/3.3));
23 }
24
25
26 void loop() {
27     // put your main code here, to run repeatedly:
28
29     // We read the value from the A0 pin
30     int A0Value = analogRead(A0);
31
32     // The total range of 3.3V is represented as integer values in the range 0,...,1023 (10 bit)
33     // Therefore to convert the input value to volts we have to multiply by 3.3/1023
34     // The maximum value read at A0 is 671
35
36     float voltage = (float)A0Value * (3.3 / 671.0);
37
38     Serial.print("Voltage: ");
39     Serial.print(voltage, 3);
40     Serial.println("V");
41
42
43     analogWrite(PIN_BLUE, blueValue(voltage));
44
45 }

```

Figure 4: Code used to accomplish fade.

where  $v$  denotes the voltage. Hence  $f$  determines the value of the blue channel. In practice, instead of rounding up, we cast the result as an integer value, effectively rounding to the nearest integer. The code can be seen in 4

[See the video in videos.zip]