Telecommunication Programming Projects with Arduino - Exercises

s214718, s224027

January 2025

LinkLink 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 avaliable for reading from the serial port, if this is more than zero (data is avaliable), 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

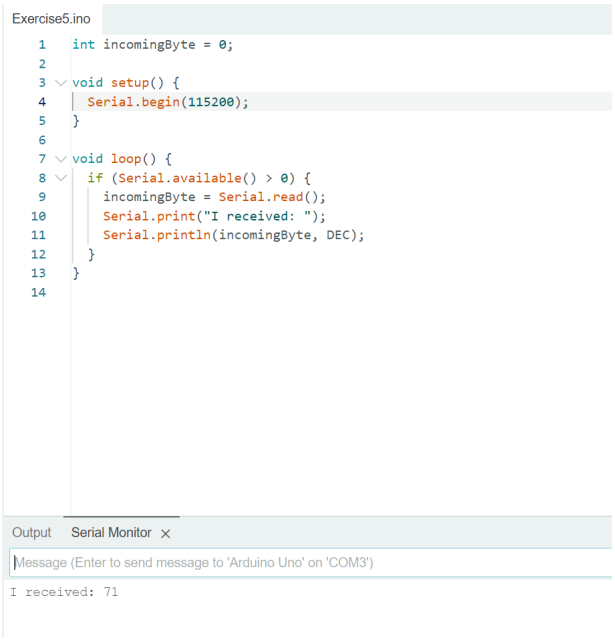


Figure 1: Screenshot of output from the given Code

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 reach color is stored in 8 bits.

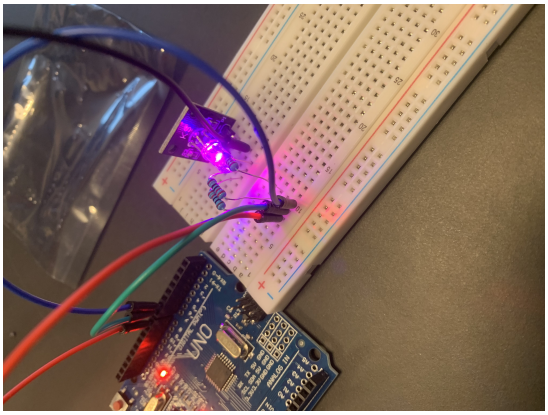
7b

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.

(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

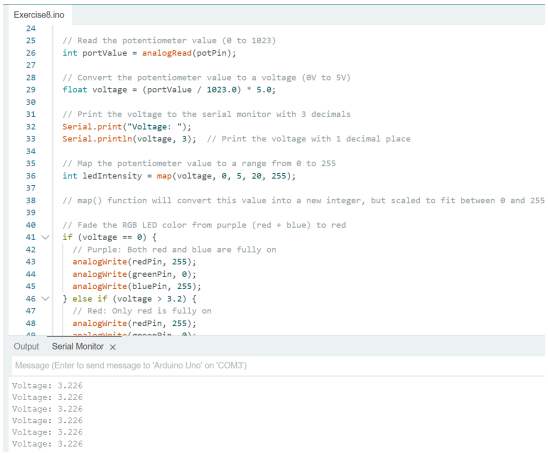


Figure 3: The voltage readings from the potentiometer, first by not moving it and the second by completely turning it around.

The voltage doesn't quire reach 3.3V despite having it compeltely 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 $i \in \{0, \ldots, 255\}$, 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 $f : [0, 3.3] \to \{0, \ldots, i\}$ defined as

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

ex8.ino

```
const uint8_t PIN_RED = 9;
const uint8_t PIN_GREEN = 5;
const uint8_t PIN_BLUE = 6;

int intensity = 30;

void setup() {
  // put your setup code here, to run once:

  Serial.begin(115200);

  pinMode(A0, INPUT);
  pinMode(PIN_RED, OUTPUT);
  pinMode(PIN_GREEN, OUTPUT);
  pinMode(PIN_BLUE, OUTPUT);
  analogWrite(PIN_RED, intensity);
  analogWrite(PIN_GREEN, 0);
}

int blueValue(float volt) {
  return (int)(intensity*(1.0-volt/3.3));
}


void loop() {
  // put your main code here, to run repeatedly:

  // We read the value from the A0 pin
  int A0Value = analogRead(A0);

  // The total range of 3.3V is represented as integer values in the range 0,...,1023 (10 bit)
  // Therefore to convert the input value to volts we have to multiply by 3.3/1023
  // The maximum value read at A0 is 671

  float voltage = (float)A0Value * (3.3 / 671.0);

  Serial.print("Voltage: ");
  Serial.print(voltage, 3);
  Serial.println("V");


  analogWrite(PIN_BLUE, blueValue(voltage));

}
```

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]