Exercise 5-8

Jonas Pedersen, Søren Graae

2025.01.07

Exercise 5

The code polls for data on the serial port (USART) and then prints the received byte in decimal. Say it receives 'A', it then prints "I received: 65".

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

int incomingByte = 0;

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);
  }
}
```

As stated it's configured to print decimal. If you send a line ending as well, that's 2 bytes in total now, it prints 10; the ASCII code for Line Feed. `(char)incomingByte` will print the ASCII symbol, as the function println is designed to do that with arguments of type `char`.

Exercise 6

A `char` is a synonym for either `uint8_t` or `int8_t` (compiler dependent), so it's 8 bits.

```
char mychar = '4';     // DEC: 52
int val = mychar-'0'; // DEC: 52 - 48 = 4
mychar = (char)(val+'A'-1); // DEC: 4 + 65 - 1 = 68
```

As seen above, the resulting value is 68.

```
#define aPin 6  // LED for character 'a' is connected to
    D6.
#define bPin 7  // LED for character 'b' is connected to
    D7.
#define cPin 8  // LED for character 'c' is connected to
    D8.
#define dPin 9  // LED for character 'd' is connected to
    D9.
#define ePin 10 // LED for character 'e' is connected to
    D10.

uint8_t rxbyte;

void setup() {
  // put your setup code here, to run once:
  pinMode(aPin, OUTPUT);
  pinMode(bPin, OUTPUT);
  pinMode(cPin, OUTPUT);
  pinMode(dPin, OUTPUT);
  pinMode(ePin, OUTPUT);

  digitalWrite(aPin, LOW);
  digitalWrite(bPin, LOW);
  digitalWrite(cPin, LOW);
  digitalWrite(dPin, LOW);
  digitalWrite(ePin, LOW);

  Serial.begin(9600);
}

void updateLEDS() {
  switch (rxbyte) {
    case 'A': case 'a':
        digitalWrite(aPin, HIGH);
        digitalWrite(bPin, LOW);
        digitalWrite(cPin, LOW);
```

```
32          digitalWrite(dPin, LOW);
33          digitalWrite(ePin, LOW);
34          break;
35
36      case 'B': case 'b':
37          digitalWrite(aPin, LOW);
38          digitalWrite(bPin, HIGH);
39          digitalWrite(cPin, LOW);
40          digitalWrite(dPin, LOW);
41          digitalWrite(ePin, LOW);
42          break;
43
44      case 'C': case 'c':
45          digitalWrite(aPin, LOW);
46          digitalWrite(bPin, LOW);
47          digitalWrite(cPin, HIGH);
48          digitalWrite(dPin, LOW);
49          digitalWrite(ePin, LOW);
50          break;
51
52      case 'D': case 'd':
53          digitalWrite(aPin, LOW);
54          digitalWrite(bPin, LOW);
55          digitalWrite(cPin, LOW);
56          digitalWrite(dPin, HIGH);
57          digitalWrite(ePin, LOW);
58          break;
59
60      case 'E': case 'e':
61          digitalWrite(aPin, LOW);
62          digitalWrite(bPin, LOW);
63          digitalWrite(cPin, LOW);
64          digitalWrite(dPin, LOW);
65          digitalWrite(ePin, HIGH);
66          break;
67
68      default:
69          digitalWrite(aPin, LOW);
70          digitalWrite(bPin, LOW);
71          digitalWrite(cPin, LOW);
72          digitalWrite(dPin, LOW);
73          digitalWrite(ePin, LOW);
74          break;
```

```
  }
}

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available() > 0) {
    rxbyte = Serial.read();
    updateLEDS();
  }
}
```

Exercise 7

An RGB value is an 8-bit unsigned integer. Thus it's between 0 and 255. This also means we can't use `digitalWrite()`. `Serial.parseInt()` polls for an integer on the Serial receive line. Anything else is dumped. It will eventually timeout (and pass 0) if no int is received.

```
#define redPin 9    // Pin used must be analog '~'
#define greenPin 10 // Pin used must be analog '~'
#define bluePin 11  // Pin used must be analog '~'

uint8_t red = 0;
uint8_t green = 0;
uint8_t blue = 0;

void updateRGB() {
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  updateRGB();

  // put your setup code here, to run once:
  Serial.begin(9600);
}
```

```
25
26 void loop() {
27   // put your main code here, to run repeatedly:
28   if (Serial.available() > 0) {
29     red = Serial.parseInt();
30     green = Serial.parseInt();
31     blue = Serial.parseInt();
32
33     updateRGB();
34   }
35 }
```

The above doesn't *fade* the LEDs as such, but sets the values. We aren't sure what is meant by fade.

Exercise 8

The ATmega328P has a 10-bit ADC, so the analog value that is read from `A0` is 10-bit; 0 - 1023. No, the values are not reasonable. This is because the default voltage reference for the ADC is 5V. This can be fixed by either connecting the potentiometer to 5V, not 3V3, or connecting 3V3 to the AREF pin and configuring the ADC to use an external voltage reference.

```
1  #define potPin 0
2  #define bluePin 9
3  // Connect the pin for red to VDD
4  // Connect the pin for green to GND
5
6  uint16_t blue = 0;
7  float voltage;
8
9  void updateRGB() {
10   analogWrite(bluePin, blue);
11 }
12
13 void setup() {
14   // put your setup code here, to run once:
15   pinMode(bluePin, OUTPUT);
16
17   analogReference(EXTERNAL);
18   updateRGB();
19
20   Serial.begin(9600);
```

```
}

void loop() {
  // put your main code here, to run repeatedly:
  blue = analogRead(potPin);
  blue = map(blue, 0, 1023, 0, 255);
  Serial.print("Analog: ");
  Serial.print(blue);
  Serial.print(", ");
  voltage = ((float)blue/255.0f) * 3.3f;
  Serial.print("Voltage: ");
  Serial.println(voltage, 3);

  updateRGB();
}
```