Postlab 01 Report Group 3_2

Q1)

In setup():
The GPIO pins that are connected to the are set as OUTPUT pins by

- pinMode(pin_number, OUTPUT)

loop():

- Two cascaded for loops iterate through the 8 led pins.
- All pins are being set on LOW by digitalWrite(led_pins[i], LOW) except for a single LED that is set to HIGH. The position of the HIGH LED is increased by 1 for every outer for-loop iteration.
- First iteration:
  [HIGH LOW LOW LOW LOW LOW LOW LOW]
- Second iteration:
  [LOW HIGH LOW LOW LOW LOW LOW LOW]
- Third iteration:
  [LOW LOW HIGH LOW LOW LOW LOW LOW]
- And so on..
- Once the last LED (23) is set to HIGH, the void loop() of the microprocessor is started again and the for-loop iterations start again from [HIGH LOW LOW LOW LOW LOW LOW LOW].

-> we get a single flashing LED that runs through the LED array from one side to the other again and again.

Q2)

Varying the resistors changes the brightness of the LEDs according to the formula: U=R*I

Q3)

Part A:
Defines the baud rate (data rate in bits per second) for the serial connection.
Part B:
If there is a Serial input available, the following will be evaluated:

1. Serial.readBytes((char*)&incoming), 1);
   Reads the first character of the available char input, saves it within the buffer and removes it from the input stream. The character is then saved within the incoming variable which is of type int => we save the ASCII integer representation of the input character.

2. dl = incoming >> 4;
   The time delay variable dl is set as the incoming integer value that is bit-right-shifted by 4, ultimately making it roughly 16 times lower than the ASCII integer representation of the input.

3. mlt = incoming % (1<<4);
   (1<<4) can also be expressed as 0b0001'0000 = 16.
   Hence we make a modulo operation on the ASCII integer representation of the input by 16.

Q4)

We are typecasting an integer address to a character address. This is used to define our buffer where we read in the Bytes. Because we want to read in a character, we typecast the incoming integer address to a character address.
The part char* stores the address of a char. In case multiple characters are stored under the variable, the pointer char* points to the starting character of that C-string (namely the dereferenced value in variable incoming).
Example:

```
char s[] = "test";
char* p = s;
```

p holds now the address of the first value in s[], namely "t".

Q5)

dl = incoming >>4;
mlt = incoming % (1<<4);

The characters that are sent are interpreted in ASCII format and stored under the variable "incoming". The operations for dl and mlt are performed on the corresponding dezimal values of the ASCII signs.
Here an example:

- "~" is sent via the serial connection.
- ASCII value of "~" is 126
- The binary value of 126 is 01111110.
- For dl this number is right shifted by 4 bits. We arrive at 01111110 -> 00000111 with a value of 7.
- For mlt the 126%(1<<4) is computed which is equal to 126%(10000) = 126%16 which is 14.
- Therefore for an input of "~" over the serial input, variable dl is set to 7 and variable mlt is set to 14.

In part C:

- a delay of length **dl*mlt** is used after a specific LED configuration has been set.
- a delay of length **dl*mlt*3** is set after all LED configurations have been played through.

Q6)

The serial connection expects an 8 bit input.

Q7)

Range dl
dl = incoming >> 4;
dl is bit-right-shifted by 4 bits, hence if incoming can be a 8-bit integer (from char) 0bXXXX'XXXX, dl will be 0b0000'XXXX. Therefore, dl can have a value between 0 and 15.
=> dl - [0, 15]

Range mlt
mlt = incoming % (1<<4);
(1<<4) can also be expressed as 0b0001'0000 = 16.
=> mlt = incoming % 16.
The modulo by 16 operation allows for values between 0 and 15.
=> mlt - [0, 15]

Q8)

In order to get dl = 3 = 0b0000'0011 we need incoming to be of the form 0b0011'XXXX => some number between 48 = 0b0011'0000 and 63 = 0b0011'1111.
In order to get mlt = 13, we need incoming to be a number 13 + x * 16 (for all x) -> [13, 29, 45, 61, 77, …].
The only number between 48 and 63 which is a of the form 13 + x * 16 is 61. The ASCII character representing 61 is the equal sign: =.
Hence, we can achieve dl = 3 and mlt = 13, by sending the character "=" (ASCII: 61 and binary: 0011'1101). We can see that a right shift of 4 results in (0000'0011) which is 3 and as explained in Q3: 61 % 16 = 13

Q9)

We can increase the range of the timing delays by changing the computational relation to the input (1) or by increasing the input all together (2).

(1) We could use the incoming variable and map it to a broader range:
    E.g. dl = incoming >> 2;
    E.g. mlt = incoming % (1<<8);
(2) Increase the serial communication allowing for larger values than 8 bits (e.g. sending 16 bit values)
    -> Serial.readBytes(((char*)&incoming),**2**);

Q10)

See code

Q11)

See code

Q12)

See code

Q13)

See code