Laboratory 1: Arduino

1 Basic working with Arduino IO, PWM and Serial

```
Give a short overview about the IO of Arduino, what pins are and what kind
of pins are there and how to use them.
```

- Pins A0 to A5 are **analog** inputs with a resolution of 10 bits. These pins are connected to an ADC (Analog-Digital-Converter) whose output is fed into the microcontroller itself. The value of the individual pins can be read using e.g. `analogRead(A0)`. This means you can read values between 0 and 1023 according toe the voltage which is applied to said pins. You read 0 when 0V are applied to a pin and 1023 when 5V are supplied.
- Pins 1 to 13 are **digital** In- and Outputs. Use `digitalRead(4)` when using the pin as an input and `digitalWrite(4)` when using it as an output. Pins 0 and 1 are intended to use for serial communication (0 -> **RX**, 1 -> **TX**). All other pins are regular but pins 3/5/6/9/10/11 are additonally capable of generating **PWM** signals.
- Pin *AREF* is a pin to supply the reference voltage for the analog inputs. Pins *SDA* & *SCL* are part of the I²C interface.

  You have to remember to configure the pin mode using the function pinMode(pin, mode) e.g. `pinMode(A0, INPUT)`

1.1 First Task to be done

```
Make an LED Blink and Fade. What is the different between the two ways to
realize it? Describe it
in few words and give an example. You can use the tutorial as base. Don’t
forget to explain the
difference concepts and ideas behind Blink and Fade. Give also a short use
case for each of this
concepts.
```

- If you want the LED to fade you need a PWM Signal which can be generated through the function `analogWrite(pin, value)`. The PWM duty cycle has a resolution of 8 bits so you can set PWM values from 0 to 255.
- If you want to make a LED blink you simply need a ON (5V) or OFF (0V) which is a digital signal that can be activated through the function `digitalWrite(pin, value)`.

  The example below shows how the function `digitalWrite()` works for a LED which should glow for one second:

```
int led = 7;

void setup() {
    pinMode(led, OUTPUT);    // Set Mode for our Pin to "Output".
    digitalWrite(led, HIGH); // Set the level on our Pin to High (LED
is on).
    delay(1000);             // Wait for 1000ms (= 1s).
    digitalWrite(led, LOW);  // Set the level on our Pin to Low (LED
is off).
}
```

We can see that the function is kept very simple. The example below shows how we can generate PWM signals with different percentages for a digital PWM Pin.

```
int led = 9; // Digital PWM Pin

void setup() {
    int value = 42;
    analogWrite(led, value); // Set Value on our Pin to 42.
    value = 255;
    analogWrite(led, value); // Set Value on our Pin to 255;
    value = 0;
    analogWrite(led, value); // Set Value on our Pin to 0;
}
```

1.2 Second Task to be done

```
Make a Buzzer Program. Describe it in few words and give an example how it
works. What is
needed to make it Buzz. Implement your own melody and describe how to
create a melody or how
to implement a song. Give also a short use case for each of this concepts.
```

- With the function `tone(pin, frequency, duration)` which we will pass the pin to which the buzzer is connected. We also pass a frequency and the duration of the note in milliseconds.

  For example:

```
void setup() {
    tone(8, random(100,2000), random(100, 2000));
}
```

The example above sets a frequency between 100 and 2000 Hz on Pin 8 with a duration between 100ms and 2000ms.

1.3 Third Task to be done

```
Expand the Blink Task with a button or a timer and a Buzzer. Create a
Program that uses the knowledge of the last exercises. Describe it in few
words how the program works. Describe the idea behind the program.
```

- A button is connected to pin 7 and the buzzer to pin 8. The button is pulled down by a 100K resistor to ground. Everytime the button is clicked (pressed and released) the frequency of the buzzer increases and the LED toggles its state. To detect button clicks we need to detect a high flank on pin 7. We do this by storing the last button state in a bool. By comparing the previous state with the current we can detect a rising flank (0 => 1).

1.4 Fourth Task to be done

```
Read and understand what Serial communication is. What other kind of
communication exist for Arduino? What are their advantages and
disadvantages? Describe at least two additional communication possibilities
additionally to Serial communication.
```

- There are also I²C and SPI. Comparing to Serial I²C is synchronous (it has a clock pin) and so you do not have to agree on a baudrate. The disadvantage of I²C is its limited speed which ranges from 0,1 Mbit/s to 3,4 Mbit/s. The ultra fast mode reaches 5,0 Mbit/s but then it is only unidirectional. Another disadvantage is that I²C is like Serial only half-duplex. SPI on the other hand needs more cables but is full-duplex and you can also chain multiple SPI capable devices together. You need:
  - SCLK (Serial Clock): Master emits this for synchronisation
  - MOSI (Master Output, Slave Input): Data output from **master**
  - MISO (Master Input, Slave Output): Data output from **slave** Optionally (when chaining multiple devices):
  - SS (Slave Select): Selects the slave

1.5 Fifth Task to be done

```
Combine task 1.1 to 1.4. Be creative. Describe, imagine a useful
application with Arduino. Be creative. Create a program that uses LED,
Buzzer, timer and serial communication. Describe what is your idea and how
do you realize it.
```

- Our program is a digital note controller. It should read the notes to play from the serial console. We realized it using `Serial.readLine()` which will receive a series of notes and delays. The song might look like:

  `<frequency>:<delay>|<frequency>:<delay>|...`

```
<frequency>:<delay>|<frequency>:<delay>|...
```

```
500:50|200:20|1000:1000|400:50
```

We have to split this string by the character '|' and split again by ':'. Then the frequency/note can be played with `tone(...)` and the delay can be executed with `delay(...)`. For the time that a note is played the LED should light up too. The LED should light up more intense when higher notes are played. This can be realized using the `map(...)` function which scales up input values to a desired output range. The input range is the frequency range e.g. 20 - 10000 (Hz) and the output range is 0 - 255 (% PWM). The LED is then lit by calling `analogWrite(...)` using the map()'ed value. With the button from Task 1.3 we are able to repeat the most recently played melody.

2 LED Matrix

2.1 First Task to be done

```
Given the Hardware (Arduino + LED Matrix 16X8 LED or Display) implement a
counter and a small application. Read the documentation of the module and
apply. Explain how the matrix or display works and what is necessary to get
started. Also describe witch parameters are important.
```

- The LED Matrix is controlled through I²C and stores the lit pixels in shift registers. To get started you need to initalize the I²C interface for the matrix using `matrix.begin(0x70)`. It is important to use the correct I²C address 0x70 (hex). After that you can start using draw functions. The given LED Matrix can be controlled with the functions `matrix.drawPixel(...)`, `matrix.clear(...)` and `matrix.drawBitmap(..)`.

  It is also very important to call `matrix.writeDisplay()` after drawing things.

2.2 Second Task to be done

```
Count with the LED from 0 to 128. Does the number fit in the Matrix? What
can you do to make the numbers fit If you are using the display does how to
make the numbers beiger and readable?  As Tipp use the function
`matrix.drawPixel(X, Y, COLOR)`. How does this function work? What are the
parameters. How do you make the content fit better?
```

- With the parameters *X* and *Y* we can set the coordinates of the pixel on the given board. With the *COLOR* parameter we can set a value to pixel. In our case we want to activate the LED so we pass the constant `LED_ON` as the third parameter to the function. To see something we need to call the function `matrix.writeDisplay()`.

2.3 Third Task to be done

```
Draw easy Bitmap on the LED Matrix or Display. How does this function work?
What else is necessary? How is the Bitmap be stored?
As Tipp use the function “matrix.drawBitmap(0, 8, om_bmp, 8, 8,
HT16K33_BLINK_CMD);”
```

- The function `matrix.drawBitmap(x, y, bitmap[], width, height, color)` can render an array onto the board. We can say on which position we want to start, which height and width our Bitmap have and which color we want. In Addition to see something we can put an array as a parameter to this function which will be rendered to our board. To see something on the board we need to call the function `matrix.writeDisplay()` after that.

2.4 Fourth Task to be done

```
Print some Text in the LED Matrix. How does this work? What options are
available? How does the example behave with longer text? How does it work?
What happens whit the memory?
As Tipp use “matrix.print("Hello");” and “matrix.setCursor(x,0);”
```

- With the function `matrix.setCursor(x, 0)` we can set the pixel position where to render the text. For example `matrix.setCursor(0, 0)` sets the pixel position on top left and `matrix.setCursor(15, 0)` set it on top right on our board. With the function `matrix.print("Hello")` we can render the given String (e.g. "Hello") onto our board.

2.5 Fifth Task to be done

```
Give a short overview and explain the most important functions and how does
the Board work.
```

- The most important functions are `pinMode(...)` which configures a pin either as an output or an input. The functions `analogRead(...)` and `digitalRead(...)` are very important for sensing things and `digitalWrite(...)` and `analogWrite(...)` are important for driving actuators.

  Besides the important IO pins the board has an USB port which allows the microcontroller to be flashed using the Arduino IDE. There is also an ISP Header for flashing the Arduino with an **I**n **S**ystem **P**rogrammer. This allows the Arduino to be flashed even when the USB Port/Flasher is broken.

3 LED Matrix as Terminal Output

```
Combine the knowledge of the two previous Tasks and write a LED-Banner that
plots the data that is send the Arduino via Terminal. The data should be
typed in a serial console and be displayed on the arduino.
Describe how your program works and what is your concept. Do some sort of
planning before you start programing.
```

1. Concept:

   First of all we have considered how we can get the data from the serial interface and what happens if there is no data to read. The next thing was how we can print the read data onto the LED Matrix board. So we knew that we need to prepare the board every time we get new data from the serial interface. To read the data from the serial interface we will use the `Serial`-library.

2. Evaluation:

   To hack the serial interface we used the functions from the `Serial` library. First of all we checked if something is available on the interface with the function `Serial.available()`. If the return value is greater than 0 then we start reading the data with the function `Serial.readString()` which will be return the data in form of a String. After we read the String from the serial interface we prepare our LED matrix board for it. We set the text size to the minimal text size with the function `matrix.setTextSize(1)`, setup the text color with `matrix.setTextColor(LED_ON)`. In addition we set the start pixel at 0, 0 with the function `matrix.setCursor(0, 0)` and rotate the text 90° with the function `matrix.setRotation(1)`, so that the text will show vertically on the board. The last thing we need to do is to clear the board with `matrix.clear()`. After the preparation we can write the read String with the function `matrix.print()`.

4 Wire layout for temperature measurement (Arduino)

```
Wire layout for temperature measurement
In the first part of the laboratory we will connect the digital thermometer
(DS18B20) and the Arduino Uno, in order to get data from the digital sensor
and to measure the temperature.
You need the following:
    - 1 or more digital thermometer (DS18B20)
    - 1 Arduino Uno or similar board.
    - 1 resistor (4,7K Ω)
    - Cables (Vcc = red, GND = black)
    - 1 Breadboard (grey element in the figure)
```

4.1 Implementation

```
The digital thermometer uses a one-wire protocol for communication.
Fortunately the Arduino IDE already provides an implementation for the
protocol (one-wire library). For this exercise we will use the already
existing library. Just download it
(http://playground.arduino.cc/Learning/OneWire) and integrated it into the
Arduino IDE.
```

- First of all we include the downloaded `OneWire`-library. After them we can create an OneWire object for our *DS18B20 Temperature chip* with `OneWire ds(pin)`. In our loop function we setup a `byte addr[8]` and a `byte data[12]` variable. The next thing we need to do is to search the address of

our Temperature chip which will be used to start the conversation with the chip. To get the adress we use the function `ds.search(addr)`. This function returns wheter True or False. If an adress was found it will be written into our variable `addr`. After them we reset our Pin, select our found address and start the conversation with `ds.write(0x44, 1)`. After them we can read the values with `ds.read()` into our variable `data`.

4.2 Temperature reading

```
Using the Arduino implement get the data from sensor and plotted at the
Serial console.
```

- In Addition to 4.1 we need to convert the read data from the Temperature chip into a readable value for us. In our example from hexadecimal representation into a String. The first thing we need to check is if the returned temperature is negative. If it is true we need to calculate a two's complement of the signed value. After them we need to mulitply by 6.25 or (100 * 0.0625), because our sensor has 0.0625 degree precision. After we got the result we seperate off the whole and the fractional portions. We do this because, when the value is shorter then 10 we need to print an extra 0 for the right value. Further more at the beginning of our convertion we saved us the sign bit. With this information we can check if it is negativ or positiv and can print a - at the beginning of our value or not.

4.3 LED scale

```
Expand the circuit with LED Matix and Print the Temperature.
```

- The last exercise with the temperature sensor follows the tasks 4.1 and 4.2. After converting the read hex temperature value into a String we can also print them onto our LED Matrix board. So we need to setup our board. We wrote a `for`-loop for our `matrix.setCursor(x, 0)` function. We use the `for`-loop to make our text glide across the board, because in case our text is to big we cut the end of the value. In this `for`-loop we setup the board. The function `matrix.setTextSize(1)` will set the font size to the minimal value. Then we forbid the text to run off right edge with the function `matrix.setTextWrap(false)`. Then we rotate the text with `matrix.setRotation(1)`, cleared the board with `matrix.clear()` and set the start pixel to 0,0 with the function `matrix.setCursor(0, 0)`. After them we can start to write the value on the board with the function `matrix.print(value)`. The printing on the board is the same principle like the printing on the Serial output. We check the sign bit and print a - if the value is minus and then we print the *whole* value. The next we do is to check if the *fractional* value is shorter than 10 and if it is true we print an extra 0 and the we print the *frational* value. So after this we got the temperature on our LED Matrix board.