WORK PACKAGE 4: TINKERCAD

EXERCISE 1: INTERRUPTS ON TIMER

Create a system based on Arduino Uno or Arduino Yun, which measures the temperature and lids the LEDs. You should use 5 LEDs for the temperature measurement. The LEDs should be turned on depending on the temperature – e.g. for 0-10 degrees Celsius, 1 LED is turned on; for 11-20, 2 LEDs are turned on, and so on.

You should define for which temperature range the LEDs should be turned on, these should be provided as variables.

You should check for the temperature periodically, using interrupts. The period should be defined in the code.

Your task is to:

- Create a board
    - with the right number of LEDs
    - with the right wiring to prevent damage to components
    - use different colors for the LEDs for the different temperature intervals
- Write the code
    - Using interrupts
    - Using the definitions

Your solution should include:

- Screenshot/picture of the board
- Source code (in C)

EXERCISE 2: ANALOG TIMER

Create a system based on Arduino Uno or Arduino Yun which implements an analog timer (seconds only). You should use a motor module to move (Micro Servo in TinkerCad).

At the same time as the motor moves, the program should send the time to the serial port.

You should use the interrupts on timer for this exercise.

Optional: you can use a second servo to measure minutes, just like in an analog timer with minutes and seconds.

EXERCISE 3: ADDRESSABLE LEDS

Create a system based on Arduino Uno or Arduino Yun, which periodically measures one of the following:

- Temperature
- Sound (using sonic sensor), or
- Light intensity

Instead of using LEDs like in Exercise 1, you should use NeoPixel ring (addressable LEDs) to display the value based on the sensor. For example, the temperature based on the temperature sensor.

Since the NeoPixel ring is a ring, you should indicate that you reached the limit of LEDs by adding one more, standard red LED. The red LED should be ON after all NeoPixel ring's LEDs are ON.

Optional: you can add speaker (Pizo) to make a sound in addition to turning ON the red LED.

EXERCISE 4: SORTING LISTS

Note: all sub tasks in this exercise (searching and sorting) should be implemented and tested in the same program.

a) Write a function which, given an integer **n**, an array of integers and the size of the array, determines if **n** is in the array. If n is in the array, the function returns the index for the first position of n (in case of several) otherwise it returns -1.

For testing this function, write a main program that tests the function with the help of an array initiated as below:

```
int test[] = {1,2,34,5,67,3,23,12,13,10};
```

The function declaration should be as follows:

```
int search_number(int number, int tab[], int size);
```

b) There are a lot of ways to sort an array. For example, we can use the bubble sort algorithm. It is not the fastest, but it is easy to understand and implement. So, write a function to sort an array of integers. The function should use the following algorithm:

- Find the minimum value in the list.
- Swap the minimum with the first in list.
- Repeat this but exclude the previous minimum on top of the list and search only in the rest of the list.

Use the following function declaration:

```
void sort (int number, int tab []);
```

Test the function by using in same program as a), and the same initiated array as above for the sub-task a). When you are done with sorting, print out the sorted array.