WORK PACKAGE 3: TINKERCAD AND C

EXERCISE 1: SET-UP

Create a simple system in TinkerCad, which will use one LED to blink periodically, e.g. once a second. Add one more LED and a button. The second LED should be on when the button is pressed, and off, when the button is not pressed.

EXERCISE 2: TEMPERATURE AND LIGHT METER

Create a system based on Arduino Uno or Arduino Yun, which measures the light intensity and temperature at the same time. The systems should monitor the dependency between these two measurements and warn about the deviations.

Normal dependencies, all other variables are treated as deviating values:

Temperature	Light intensity
< -12 °C	0 %
-12 °C - 0 °C	1% - 20%
0 °C - 20 °C	21% - 60%
>= 21 °C	61% - 100%

The system should use periodically read the temperature and light intensity (periodicity, in seconds, should be provided as a variable in the program).

The system should use three LEDs to indicate the normal dependency (GREEN), deviation when the temperature is higher than it should be, given the Light intensity (RED), and the deviation when the temperature is lower than it should be (YELLOW or BLUE).

Your solution should include:

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

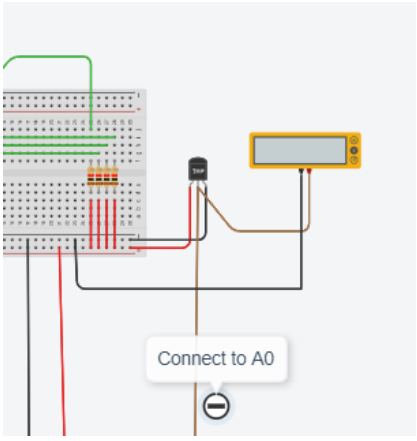
You can use DHT.h library for this exercise.

EXERCISE 3: TEMPERATURE METER V 2.0

In this exercise you should add the display of the temperature to your temperature meter and use analog conversion of the temperature meter.

The main idea is that the main-loop in the program, with short delay (500ms), checks the temperature by reading the input voltage on AD – input A0 - calculates the corresponding temperature and print it on the serial monitor.

AD converter with 10 bits resolution. You can use the Arduino IDE:s function (analogRead()) to read A0. Calculate the temperature using this formula:



Temperature (°C) = (V_{OUT} – 500) / 10

You should add Multimeter to be able to check the printed temperature for a certain in-voltage**.**

The main program should periodically check the temperature, write it to the multimeter and to the serial port (to display in TinkerCad's serial output).

Examination : Demonstrate for a TA and write the examination code in the program head. Copy the program to the same document as exercise nr 5.1.

IMPORTANT! You are not allowed to use <DHT.h> library for this exercise.

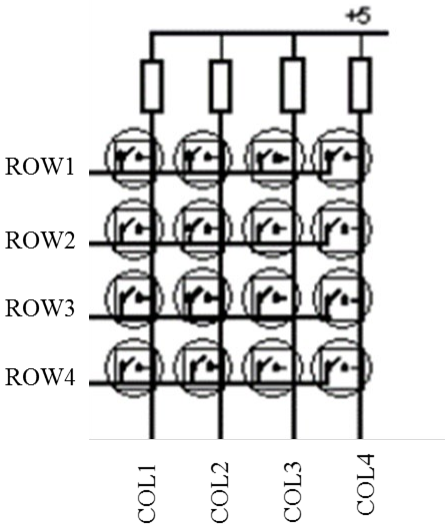
EXERCISE 3: KEYBOARD SCANNING

In TinkerCad you can connect such a simple keyboard (Figure) to Arduino Uno and then design a program that reds a pressed key number.

The principle for the keyboard and reading a key.

The keyboard is built up by four row connectors and four column connectors. At the 16 crosses between raw and column there is normally no electrical connection. At each cross there is a key and if a key is pressed there will be an electrical connection between the column and raw. (See figure below)

The rows (row 1 to row 4) is possible to connect to anything via the four left bits in the flat connector. On the four right bits the four columns are possible to connect to anything.



The keyboard can be connected to any of Arduino´s IO-ports. One possibility is to connect all 8 row/columns to Port D and configure the port D as 4 bit out/four bit in.

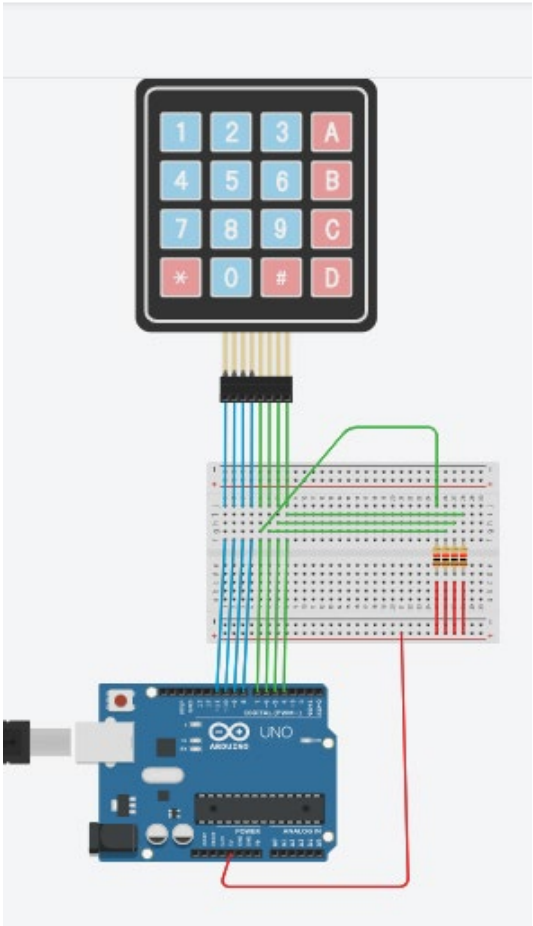
If we chose this way of connecting the keyboard it will be a bit more tricky than necessary to develop the program for reading the keyboard. This is depending of the possibility to use the same physical IO-port as IN or OUT on bit level. We will also lose the possibility to use the serial monitor for printing out some control text.

Instead we chose to connect the keyboard by connecting the rows to Arduino Port B (bit 0 – 3) and the Columns to Port D (bit 0 – 3) . By this we can use Port B as outport and Port D as inport.

To ensure the level (electric) of the columns, when no key is pressed, you normally connect them to 5V via a resistor (so called "pull up resistor"). You can see this in the figure above.

Your task:

In Tinkercad you should design a new circuit as described above and shown in the figure to right. We will later expand the circuit so it´s good to design it as the figure shows.



You should then, also in Tinkercad, develop a program reading the keyboard system above. The program should print out the key number in the serial monitor if a key is pressed. If no key is pressed there should be nothing printed out.

The program structure should be as below. You should also use the predefined macros as in earlier exercises.

You can use the function `Serial.println()` for printing on the monitor.

- configure the IO ports
- in an infinity loop
  Call a function checking if any key is pressed and if it should return it´s value.
  If a key was pressed print out the key value (0-9 , A-F) on the serial monitor.
  Delay for one second.

Demonstrate for TA and hand in the solution as for exercise 4_4 .

IMPORTANT! You are not allowed to use <keypad.h> library in this exercise!

EXERCISE 4: KEYBOARD SCANNING V 2.0

Use keypad.h library to make the same system as in Exercise 3.

EXERCISE 5: POINTERS AND ARRAYS

Write a program that creates an array of integers, `array[MAX]`, and then fill it with MAX number of random integers from 1 to 99. Let then the program prints out the following:

The value of the address of the *array* (pointer) is: xxxxxxxxxx
First integer in the array is (array[0]): xxxxxxxxxx
The last integer in the array is: xxxxxxxxxx
The size of an integer (number of bytes) is: xxxxxxxxx
The size of the whole array in bytes is: xxxxxxxxx

The program shall then, by use of a pointer, print out each integer value and then print the value multiplied by two.

EXERCISE 6: ARRAYS AND FILES

Write a program that reads in a string with a maximum of 20 characters from the keyboard and stores the string in a local string variable.
Copy the string to another string by using:
a) The library function strcpy(..)
b) Your own function ***void copyString(…)*** not using any library function.
Main program ends by printing out the copied string in the console.

The program should be able to both read in from keyboard or from a text file 'myfile.txt' containing one string of characters. You create this file with notepad or any other text editor. The reading from the text file should be done by redirect the readings from command line when program execution starts as follows: *Exerc_2_1 < myfile.txt*

Where Exerc_2_1 is the filename of the compiled program. You **shall not** use standard file managing by opening the file and the read from it.