**SZÉCHENYI ISTVÁN EGYETEM**
AUDI HUNGÁRI JÁRMŰMÉRNÖKI
KAR

# HELP FOR THE PROJECT WORK

## Measurement practice I.

## FOR VEHICLE ENGINEER STUDENTS



Version: 1.0

Széchényi István University
Department of Power Electronics and Drives

# 1. Introduction

In this document you will find some useful information and examples to help you with the compulsory project work.

The project work is about to build a simple resistor measurement system with the help of an Arduino UNO board and a liquid crystal display (LCD). The measurement method is based on the voltage division law. The system applies a known voltage (5 V) to a series resistance network that consists of a resistor with known value and the one which's value we want to determine. The Arduino board measures the voltage of the unknown resistor and determines the value of the resistor based on the voltage division law, then it displays this value on the LCD. In this file you will find examples on how to measure the voltage with the Arduino board using its analog ports and how to control the LCD. Based on these examples you should be able to do the project work, if this is not enough feel free to contact us in email or at the laboratory classes.

The examples are implemented in "Tinkercad" environment. Tinkercad is a free and easy to use online application for 3D modelling, (basic) circuit simulation, coding and microcontroller-based application simulations. Before building the actual system in the laboratory you will need to implement it in a Tinkercad simulation environment.

To write and upload code on an Arduino board, you will need to use the Arduino IDE which you can download from the official website. The Arduino IDE (Integrated Development Environment) is a software application that serves as the primary platform for programming and developing projects on the Arduino platform. Using this link you can find some basic information and a starter guide for the IDE.



## 1.1 Using Tinkercad

In order to use Tinkercad you need to create a free account, then after logging in you can create a new circuit design as you can see it in figure 1.

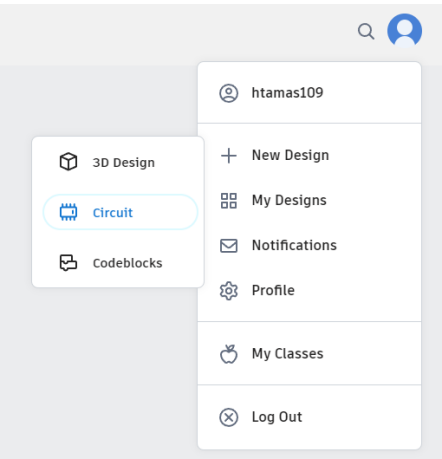


***Fig.1.*** Creating new design

In a “Circuit design” you can place various electric components, sensors, actuators, microcontrollers, breadboards etc. from the “components” menu (at the right). You can connect the components using wires by clicking on the pins of the components. The editor displays realistic images of each component, which makes it easy to use as you can see it in figure 2.

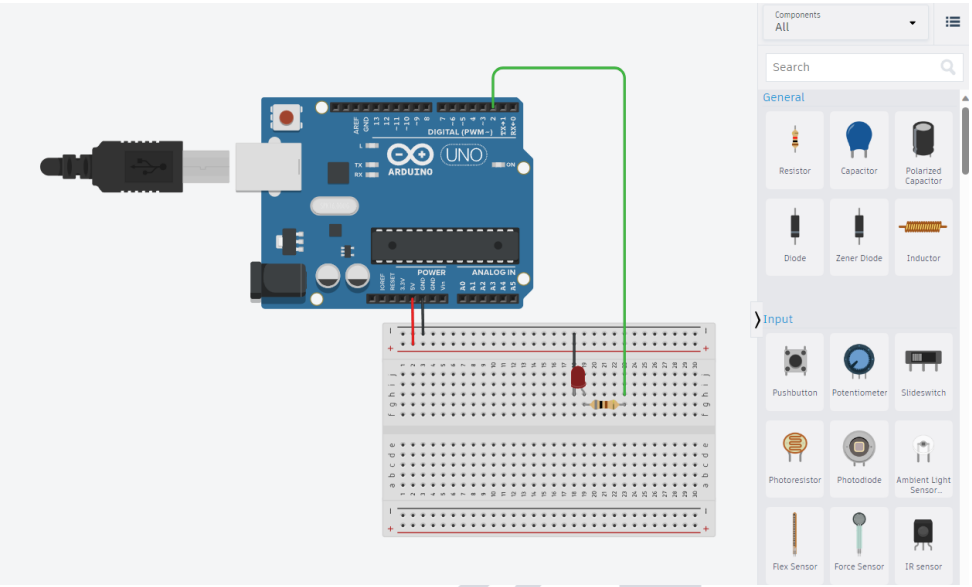


***Fig.2.*** The user interface of Tinkercad

If your circuit contains a programmable device e.g., a microcontroller, you can write a code (by clicking on the “code” button on the top right) and upload it on the device. You can write your code using blocks or text, but you must use the text format as you will need to implement your written code on a real Arduino board using the Arduino IDE. By clicking the “Start Simulation” button the code gets compiled and uploaded to the selected microcontroller automatically and the simulation will start.

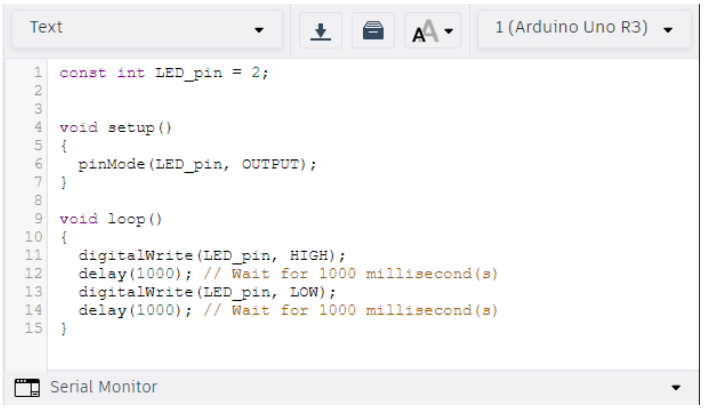


***Fig.3.*** The code editor window

## 3. Using the analog pins on the Arduino

To measure analog signals microcontrollers, use analog to digital converters (ADC). The Arduino Uno has a 10-bit ADC, that can measure voltages between 0 and 5 V. This means it will give a digital value in the range of 0 – 1023 (2^10)**.** This is called as a resolution which

indicates the number of discrete values it can produce over the range of analog values. The Arduino Uno's ADC has 6 independent channel which means it can handle 6 analog input pins. If you want to learn more about ADC's click on the link. It is important to mention that in most cases microcontrollers can only measure voltage on their analog pins.

In order to access the converted digital value of an analog pin you can use the "analogRead()" function which returns the digital value of a specified analog input. You can see how to use it in the example project provided below.

To demonstrate how to use the analog inputs of an Arduino Uno I created a Tinkercad project, which measures the voltage of a potentiometer and sends the read value to the serial monitor.

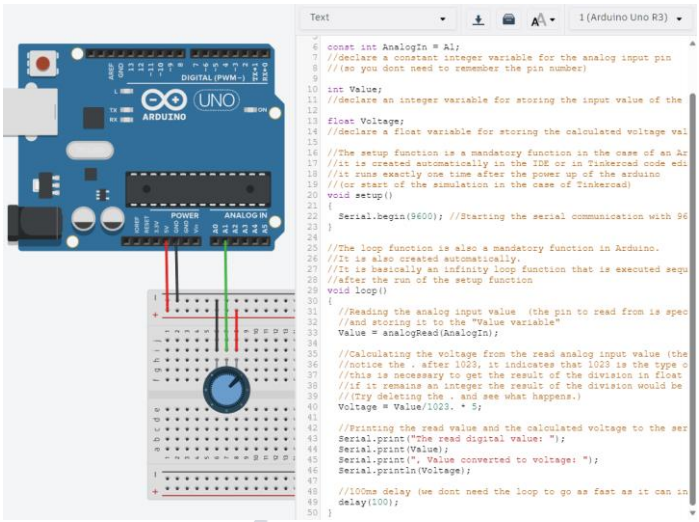


***Fig.4.*** Digital multimeter



Serial communication is a standardized communication protocol that is used in microcontrollers to communicate with the PC or with other devices. If you want to know more about serial communication click on the link. You can use the Arduino environment's built-in serial monitor (in Tinkercad too) to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin(). If you want to learn more about Arduino's serial communication click the link.
The most important functions to use the serial monitor for the project work is "Serial.begin()" and "Serial.print()", "Serial.println()". The "Serial.begin()" function starts the serial communication between the board and the PC with the baud rate specified as the input of the function (for this project 9600 is fine). The "Serial.print()" and "Serial.println()" functions are used to print information to the serial monitor, they work the same way, except the second one prints a new line character to the end. You can see how to use them in the example. You can also send data from your PC to the Arduino using the serial monitor, but that is not needed for the project work.
(In Tinkercad you can open the serial monitor by clicking the "Serial Monitor" button at the bottom of the code window.)

You can access the Tinkercad design using this URL: Using Arduino's analogRead example. To open the design, you should click on the "tinker this" button. I provided comments to each line of the code as an explanation. It is strongly encouraged to play around in all the provided projects, try to understand the code, and write it on your own, also try other components etc.

## 4. Displaying data on an LCD with Arduino

LCDs or liquid crystal displays are using liquid crystals to either block or allow the backlight to pass through, creating characters and symbols on the screen. The specific LCD you will need to use can display characters in 2 rows with 16 columns in each row, so it is a 2x16 display. Each character is displayed using a 5x7 pixel matrix.

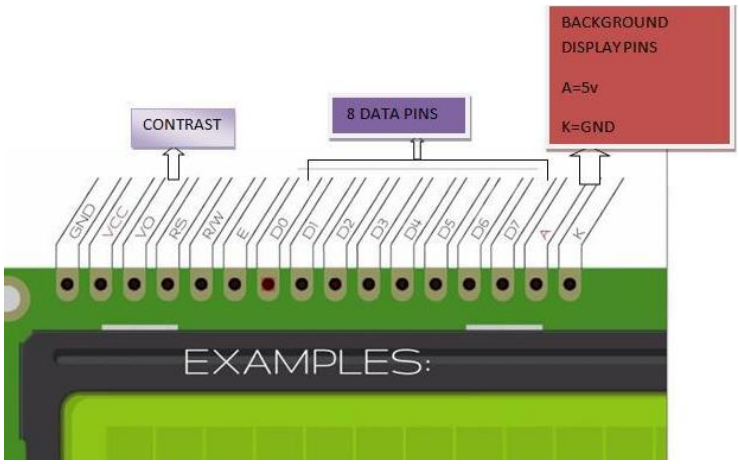


***Fig.5.*** *The used LCD.*

The LCD has two registers, the command, and the data register. The command register stores commands for the display. Commands are predefined tasks for the display like clearing the display, moving the cursor etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed. We can access the command and data register of the LCD with the 8 data pins (D0-D7) of the LCD. To choose which of the two registers we want to access we can use the RS (register select) pin of the LCD. We can read or write these two registers as well, which can be decided with the Read/Write (R/W) pin. There is also an enable pin (E) which is used to enable the LCD to perform the required task. You can also turn on or off the backlight (A,K pins) and adjust the contrast of the display with a variable resistance using the Vo pin. (If you want to know more about controlling a 2x16 LCD click on the link.)

As you can see in order to control an LCD, we need to control multiple digital inputs of the display at the same time, that requires multiple digital output of the microcontroller which is not very convenient. To overcome this problem there is an I2C communication capable GPIO (General Purpose In-/Output) expander connected to the LCD called PCF8574. This module performs the control of the parallel digital inputs of the LCD based on the commands it gets from the microcontroller on the I2C bus.

I2C is a half-duplex, synchronous serial communication protocol that uses only two lines for data communication called Serial Clock (SCL) and Serial Data (SDA). Serial Clock is used for

timing of the signals on the Serial Data line which is used for the data transfer. An I2C device can operate in Master or Slave mode, the Master device creates the clock signal on SCL and controls the communication. There can be more than one Master devices connected to the same I2C bus. Every slave device has an 8-bit Slave Adress, which is used by the master devices to address the specific slave device. The data is transmitted in the form of 9-bit packets. The communication sequence starts with a Start Condition, which is followed by the Slave Address, and then an Acknowledge bit. If you want to learn more about I2C communication click on the link.

To communicate with the LCD using I2C, you will need to use predefined libraries. A library is a collection of pre-written code that provides specific functionality and can be reused in various programs or projects. You can find bunches of libraries on the internet (for example on Github) for different specific applications. An Arduino library is typically composed of two main files, the header file (.h), and the source file (.cpp). The header file is like a user manual or blueprint for the library. It provides the set of functions, classes, and variables that can be used in your Arduino code. It acts as an interface for your Arduino code to interact with the library, listing what functions are available and what parameters they expect. The source file is where the actual code for the library is written. It contains the implementation of the functions and classes declared in the header file.

To use a library, you need to include the header file at the beginning of your code like: #include<Wire.h> and after that you are ready to use the functions and classed defined in the library. In Tinkercad you can only use the supported libraries that are already added to the platform. You can check the supported libraries by clicking on the “Libraries” button in the code editor window as you can see on figure 6. If you want to use a not supported library, you will need to manually copy the content of the source and the header file.

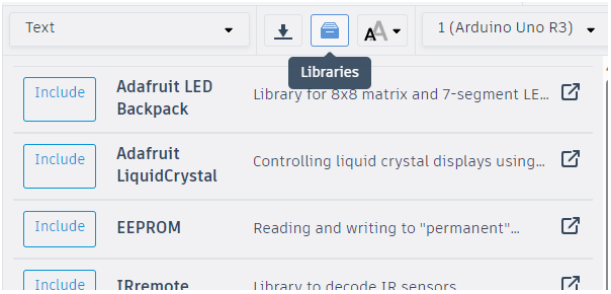


***Fig.6.*** *The supported libraries in Tinkercad.*

In the case of the Arduino IDE there are several libraries that you can use by default, but you can also add new libraries. To add new libraries and check the already added libraries you will need to use the library manager. On figure 7. you can see how to access the library manager.

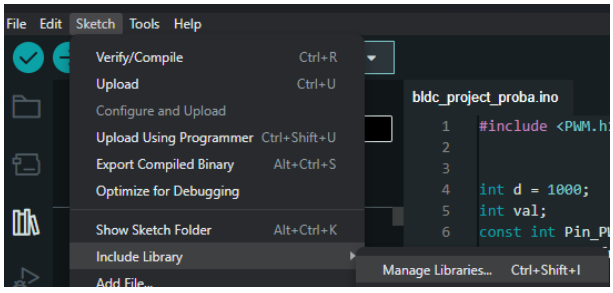


***Fig.7.*** *Opening the library manager in Arduino IDE.*

To controll the LCD you will need to use the “Wire” and the “LiquidCrystal_I2C” libraries. The Wire library is used to communicate with I2C devices, and the LiquidCrystal_I2C library is for using I2C capable LCDs.
To demonstrate how to use an I2C capable LCD with the Wire and LiquidCrystal_I2C libraries I created a Tinkercad project that you can access using the link. In Tinkercad you can set the address of the LCD, as you can see in figure 8, for the provided project the address is set 32. It is important to note that in the case of the real LCD you will need to use for the project this address is 39. In the case of the Arduino IDE, you will need to give this address as a hexadecimal number, so you will need to write it like this, “0x27”. The “0x” prefix denotes that the number provided is hexadecimal. In the case of Tinkercad you can write the address in simple decimal form.

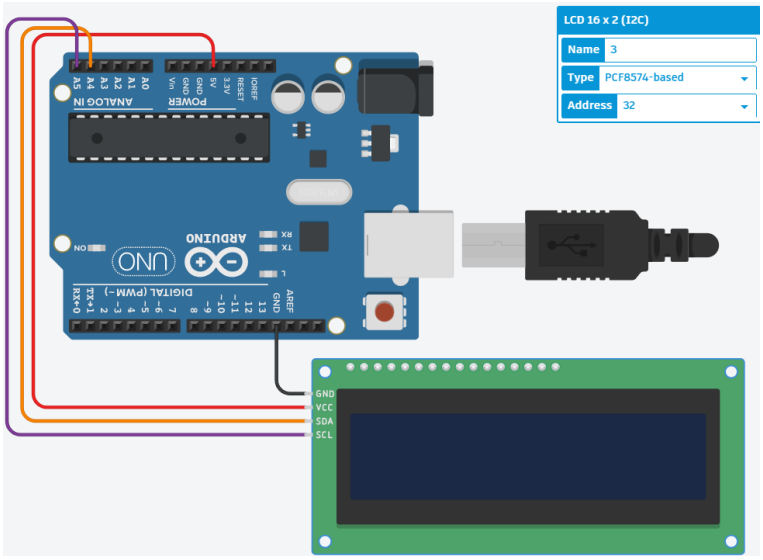


***Fig.8.*** *Using LCD in Tinkercad.*