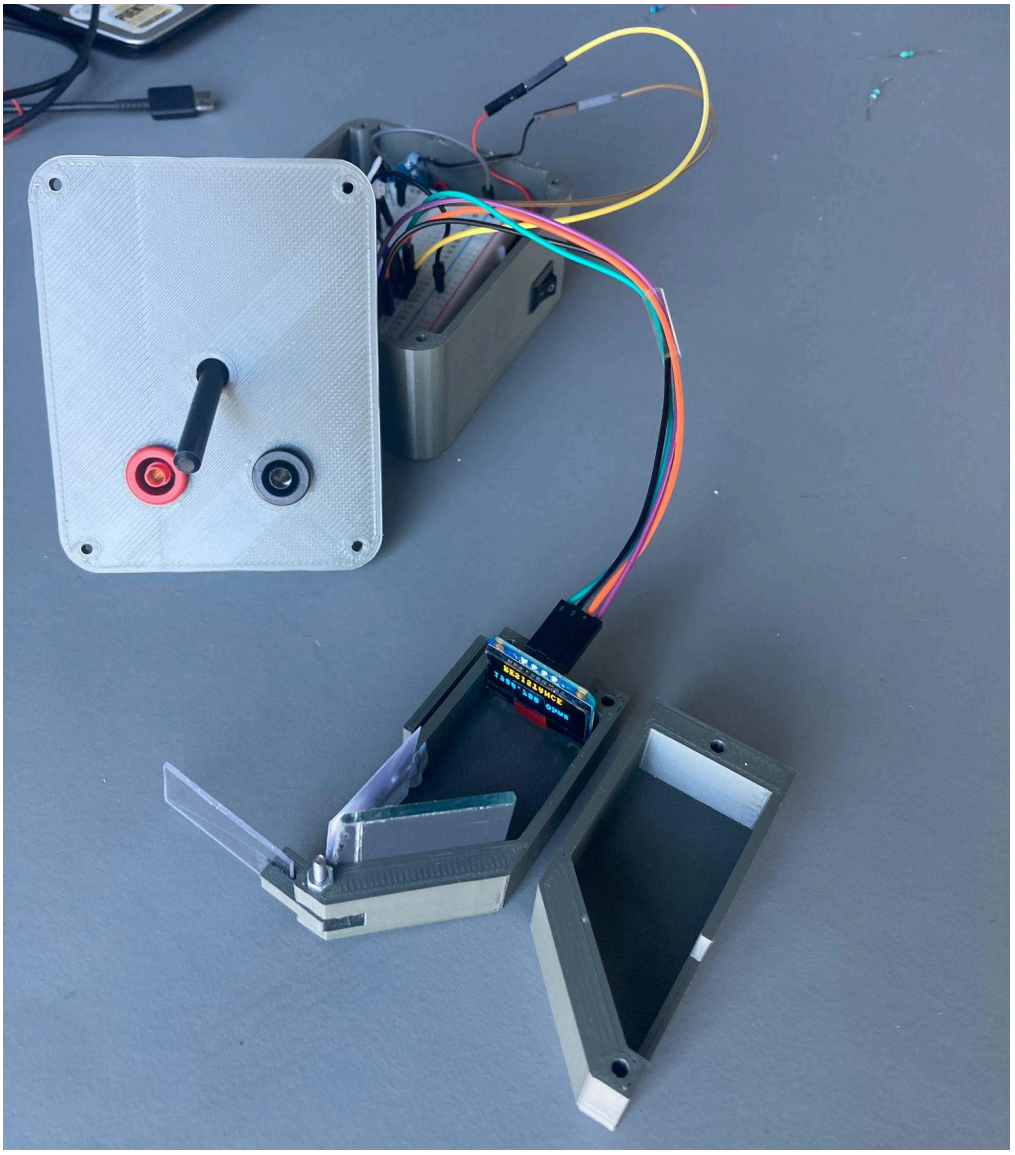
MECHATRONICS

SEMESTER PROJECT 2



GROUP 7

Joan Redón, Gonzalo Robles, Tomás Ramos, María Ling Valverde & Max Micó

Due date: June 3rd 2024

1. Introduction

For our Second Semester Project in Mechatronics, we were tasked with designing and manufacturing a multimeter that incorporates mechanical features and a Nextion display screen. With the challenge we faced, we were given a budget of 2,000 DKK to purchase all necessary components, except for the elements already offered by the university and always at our disposal.

The main functionality of the multimeter was that it needed to be able to measure at least two out of three specified passive components: resistors, capacitors, and inductors, these being directly related to three different electrical magnitudes: resistance (Ω), capacitance(F), and inductance(H) respectively. In addition to these first, basic requirements, we were also given the task of identifying a potential customer demographic and coming up with other supplementary specifications to meet the specific needs and preferences of our target customer. With this approach, we ensured that our final product would not only fulfill the technical requirements we specified but also comply with real-world problems and customer expectations.

2. Problem formulation

In the past, it has always been a challenge to measure aspects even before technology became widespread. Determining the capacity, resistance and other electrical attributes of materials and components has been crucial for assessments. The precision in measurements plays a role in advancing engineering fields. The establishment and improvement of this fundamental capability have led to the sophisticated technological environment of today. Improving measurement methods has propelled the progress of technology by boosting efficiency and reliability in applications.

Accurately measuring resistance, voltage, and current is essential for troubleshooting engineering issues for students working on related projects. Real-world measurements of these properties yield realistic and dependable results compared to theoretical calculations that often idealize outcomes beyond reach. Practical measurements consider real-life variables and imperfections that theoretical formulas cannot encompass fully, providing an understanding of components and systems, under scrutiny. This hands-on approach does not enhance data accuracy. It also enriches the learning process and problem-solving skills for both aspiring engineers and seasoned professionals.

The primary objective of this project Is to create more than a conventional multimeter. We want to create a Multimeter attached to a set of smart glasses, the multimeter would be attached to the belt of the user's pants and connected to the glasses display screen. We also must include mechanical capabilities through a DC motor and our idea is to implement a dynamo motor and the user should be able to recharge the battery by hand.

The first encounter with the objectives is the consideration of the different components to manufacture the multimeter that can fit into a 2000 DKK budget. By the end, it should be capable of correctly measuring two simultaneous units, displaying the information in the lens attached to the smart glasses, and recharging the battery by hand.

2.1 Requirements

- ➔ Must be able to measure two passive components (Resistor, Capacitor & Inductor including voltage).
- ➔ Must incorporate a DC motor in the design.
- ➔ Must use a Nextion Display screen to show the results.
- ➔ Find potential customer groups.
- ➔ It should not cost more than 2000 DKK in total.
- ➔ Should incorporate new software communication technologies (Github, Microsoft Teams, and Google Drive).
- ➔ The Multimeter should have a compact and functional design.

2.2 Risk assessment

It is of vital importance to remember that these risk assessments, defined as the systematic process of identifying, analysing, and prioritising potential risks that may affect a project, are constructed using the available information and may require modifications depending on the specific details of the project. To provide proactive risk mitigation throughout the project's lifecycle, effective risk management involves continuous monitoring, the application of appropriate mitigation measures, and the regular evaluation and updating of the risk assessment (Figure 2.1.1).

-	SCALE OF SEVERITY					
		NEGLIGIBLE	MINOR	MODERATE	MAJOR	FATAL
	RARE	Use Arduino libraries in C				
	UNLIKELY		3D printing error		Wrong planning of first component orders	
	POSSIBLE		Order unnecessary components			Break of some devices
	LIKELY		Get burnt with silicone gun	Delay in delivery	Excess of voltage	
	ALMOST CERTAIN		Problems with the code			

(Figure 2.1.1)

2.3 Time plan

This project commenced at the beginning of the semester in February and it is anticipated to be completed by June. In the interim, tasks have been distributed among team members to ensure timely completion. Each member selects tasks based on their expertise and challenges themselves with tasks in areas where they have less experience.

Figure 2.1.2 depicts a time plan as a schedule that details the timeline for project tasks and ensures that all activities are completed within the set deadlines. This plan is essential for effectively managing time and keeping the project on track.

	TASK	PROGRESS	STARTED	FINISHED
GUIDANCE	Set the general idea	100%	07/02/24	11/02/24
	Prepare the initial shopping list	100%	05/03/24	30/04/24
	Problem formulation	100%	11/02/24	05/03/24
DESIGN	First design of prototypes	100%	05/03/24	02/04/24
	Overall functional designs	100%	02/04/24	20/05/24
	Create a full Assembly	100%	20/04/24	23/05/24
	Prepare files for 3D printing	100%	23/05/24	23/05/24
ELECTRONICS	Research about necessary components	100%	05/03/24	30/04/24
	Find datasheets / Libraries	100%	05/03/24	30/05/24
	Capacitance circuit	100%	02/04/24	07/05/24
	Resistance circuit	100%	19/04/24	02/05/24
	Rotary switch system	100%	19/5/24	01/06/24
	Complete design integration	100%	25/05/24	01/06/24
MECHANICAL	3D Printing of prototypes	100%	23/05/24	24/05/24
	Measuring and cutting glass	100%	20/04/24	23/04/24
	Final 3d print design	100%	24/05/24	28/05/24

CODE	Set a main plan for the code	100%	05/03/24	17/03/24
	Create code for the screen	100%	19/05/24	01/06/24
	Create code to measure resistance	100%	23/04/24	05/05/24
	Create code for the rotary switch system	60%	28/05/24	01/06/24
	Create code to measure capacitance	100%	06/05/24	15/05/24
	Fusion codes	100%	01/06/24	02/06/24
	Testing	100%	25/05/24	02/06/24
	Troubleshooting	100%	14/05/24	01/06/24

2.4 Budget

Fitting all the pieces and components we needed into the budget took a lot of work, as sometimes we had to order things we did not end up using. The limit was set at 2000 DKK, and we ordered most of the components through RS Components and Elextra DK.

In the following table (Figure 2.3.1) there is every component we ordered, the quantity, and the total price.

Even though we ordered more components than we needed, we managed to save more than 300 DKK of the initial budget. Some of the remaining pieces we did not use were shared with classmates from other groups or saved for future semester projects or other works.

Part Name	Quantity (u)	Price	Total
ADS1115 Sensor	1	187,09 kr.	1.666,49 kr.
Buttons	2 (10u)	82,18 kr.	
PLA 3D printer filament	1	150,60 kr.	
LM324N Comparator	1 (50u)	164,50 kr.	
LM339N Comparator	1	13,28 kr.	
Battery	1	212,99 kr.	
Controller	1	137,16 kr.	
Midas MDOB 128064 WV-YBI	1	90,70 kr.	
Sliding switch	1 (5u)	24,33 kr.	
Connectors male	1	164,84 kr.	
Connectors female black	1	39,19 kr.	
Connectors female red	1	47,30 kr.	
Charging module for battery	1	15,95 kr.	
Clear Plastic Sheet	1	241,32 kr.	
Rotary Switch, 4 Position, 3 Pst	1	47,88 kr.	
3 Position 4PST Rotary Switch	1	47,18 kr.	

Figure (2.3.1)

3. Design

3.1 Brainstorm

Before the concept of smart glasses was born, we thought of making a multimeter as a phone case that would go over the user's mobile device to allow them to use their phone as a multimeter and view the measurement on their screen (Figure 3.1.1). However, we quickly realized that the structure dimensions factor was a major issue. If we were to carry on with the phone case we would not have been able to make it thin enough for it to be fittable on a phone case.

Another important aspect that made us back out of this idea was how to display the multimeter data on the phone screen, which would result in an unnecessary complication.

An alternative concept was to have the multimeter fastened to the user's forearm with a big screen, freeing up their hands from holding the device (Figure 3.1.2). We thought this would be our final design, but then we came up with a new idea, a smart glasses multimeter.

The final concept came up as an improvement over the previous idea, with the multimeter attached to the user's belt and the information shown on the smart glasses [1] (Figure 3.1.3), we would free the user's hands and also give movement of freedom of the arms so they could continue working on their project having a clear view of the component being measured and the multimeter results at the same time, so our potential customer demographic is engineers, electricians, or even students as it is an improvement of the basic multimeter that could be used by most people.

(Figure 3.1.1)

(Figure 3.1.2)

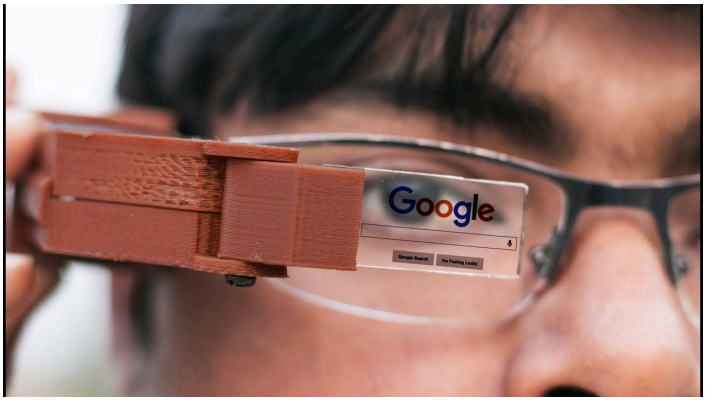
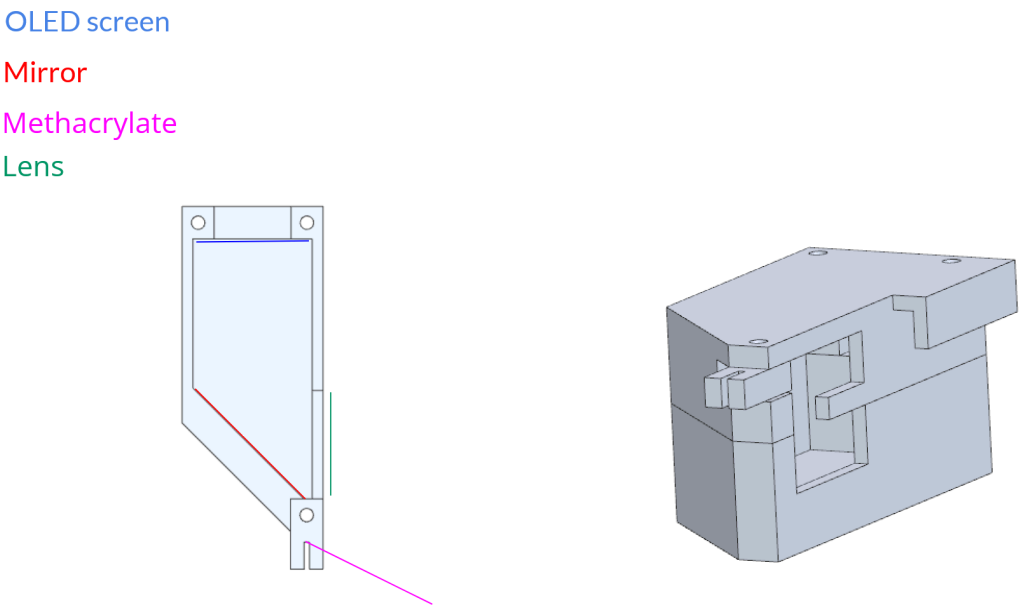


Figure (3.1.3)

3.2. NX Designs

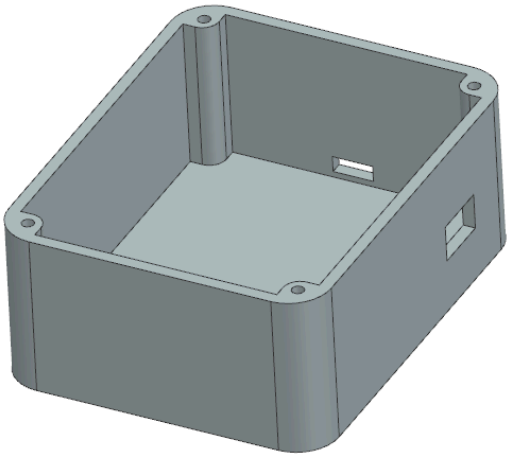
We began researching and designing the glasses with the concept already in mind. Using NX we created the prototype, which included space for the screen, a 45-degree angled mirror, the lens, and the movable methacrylate part where the screen's information would be displayed.

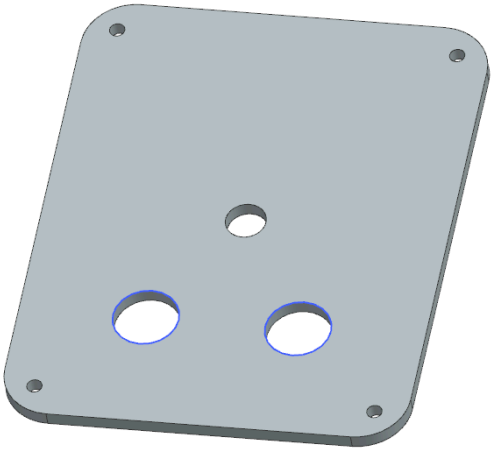
In the beginning, we had really good expectations in the prototype but ended up having some issues that needed to be taken care of. To address these problems, we made small adjustments to the design, such as creating new holes for the cables to pass through and arranging differently the holes to close them. We also reinforced the attachment of the glass box to the glasses to prevent them from falling. Eventually, we ended up with what could be our final design.



We implemented a basic design for the box that could fit the battery, the rotary switch, the Arduino, and all the necessary circuits. It has the holes needed for the on-off battery switch and the charging USB-C port for the battery. Also, we decided to use a screw-based mechanism to connect the main box to the top part of the

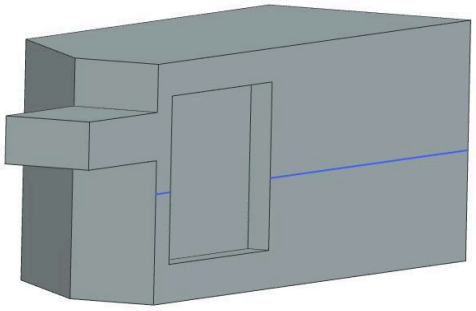
case. We will be using M3 screws to make it easy to dismount when needed. This box will connect the battery and Arduino inside of it with the screen on the glasses through some cable extensions.

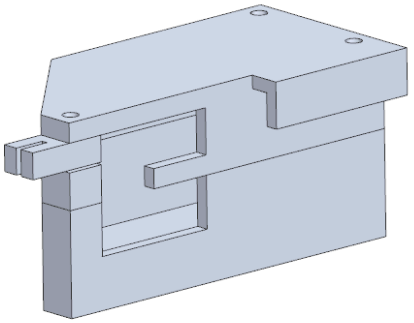




3.3 Morphology chart

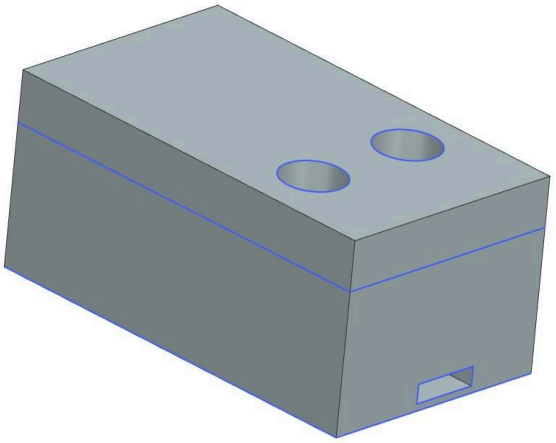
Our first design was a general sketch of what the smart glasses support box had to look like, but of course, it had many issues; the support for the methacrylate was too big, the angle for the mirror was incorrect, and the box was all in one piece so we had to split it in two so that we could install all the components inside and then close it.

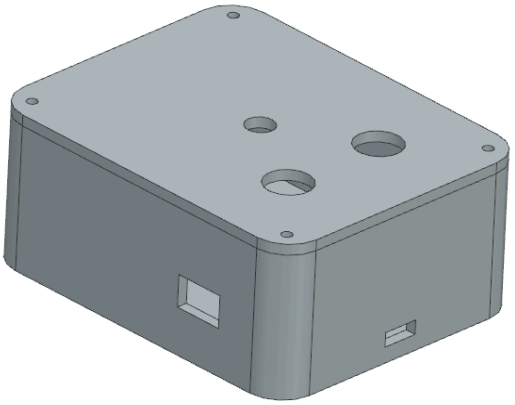




The second design did not have a hole in the back, so there was no way for the cables to get out of the box and connect to the multimeter, so we had to make some arrangements and print another prototype.

For the box design, we also had to design some versions of it, as we added extra utilities afterward like the ON/OFF switch and the hole for the cables, but overall it was an easy design, we also got to work the screw mounted mechanism at the first try so it did not struggle a lot in the making of the box.



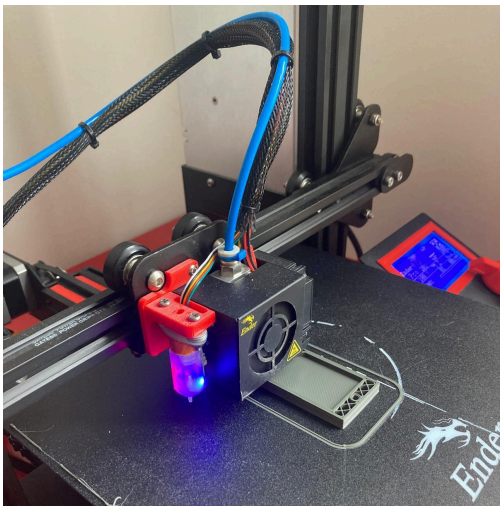


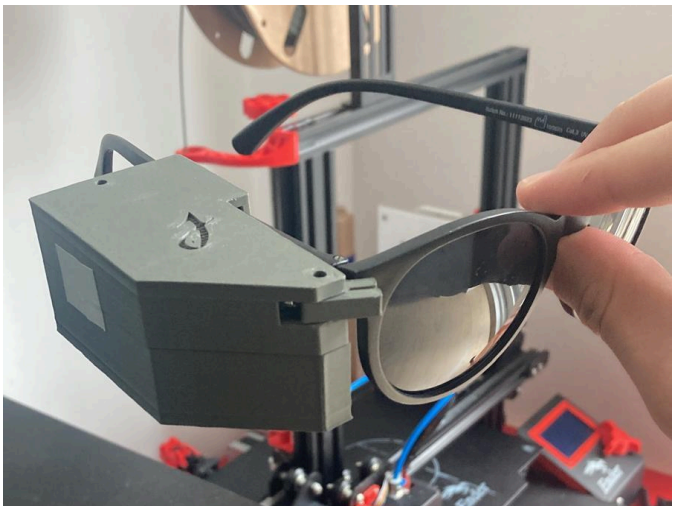
For us, it was key to have a 3D printer of our own to be able to make multiple trials and errors to find the most accurate solution, as quickly as possible, that is why we spent part of the budget on 3D filament. Although there are printers in the workshop, their availability is considerably scarce and many of them are out of work by some time in the semester.

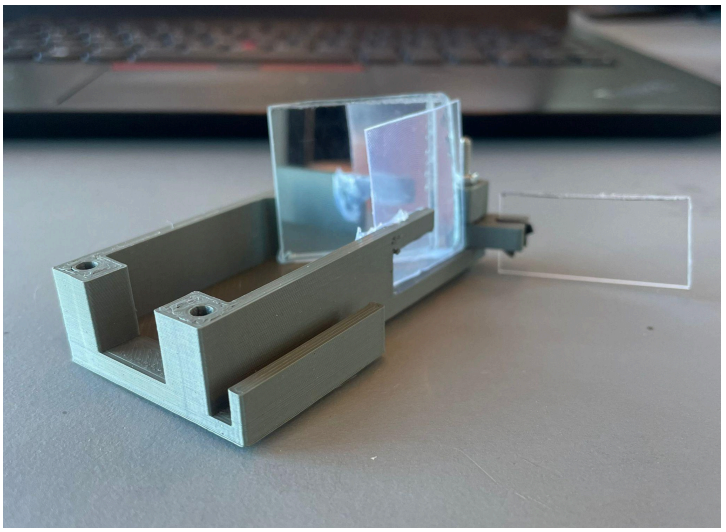
4. Prototyping

4.1 Mechanical

For the mechanical design part we needed two main parts to be designed, the glass holder for the screen and the necessary parts to display it on the methacrylate, and the box and belt attachment to store all the electronics components and battery to make the multimeter work.

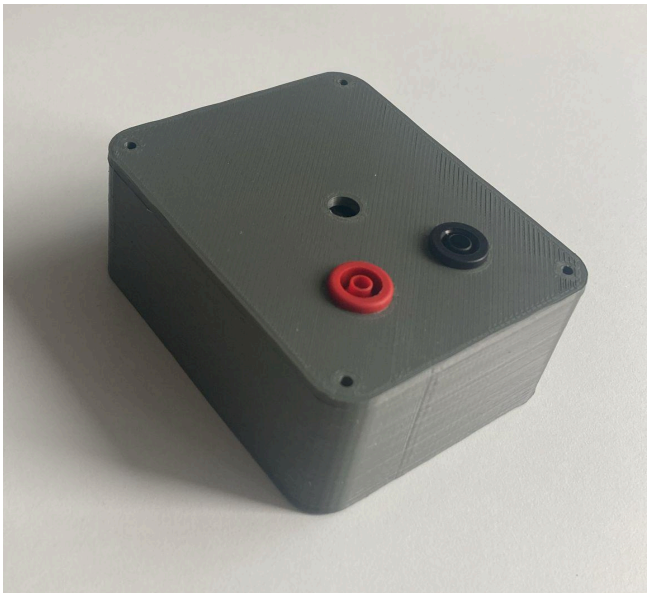






We also had to measure and cut the mirror, the fresnel lens, and the methacrylate to properly fit the smart glasses design.

For the following part, the box, we implemented a basic design that could fit the battery, the rotary switch, the Arduino, and all the necessary components of the circuit. The easiest approach was a very simple and compact design, then we came up with this.



4.2 Electrical

4.2.1.Capacitance Meter

Capacitance is the potential of an issue or circuit to accumulate and save electricity in the shape of an electrical fee [5]. This function is very essential in various electronic applications. In our multimeter design, we measure capacitance by using the usage of the basic conduct of capacitors, especially via the idea called the time steady. The time constant (τ) is an essential p arameter that defines the reaction time of a capacitor in a circuit, in specific the time it takes for the voltage on the capacitor to attain about 63.2% of its highest price all through charging. This feature

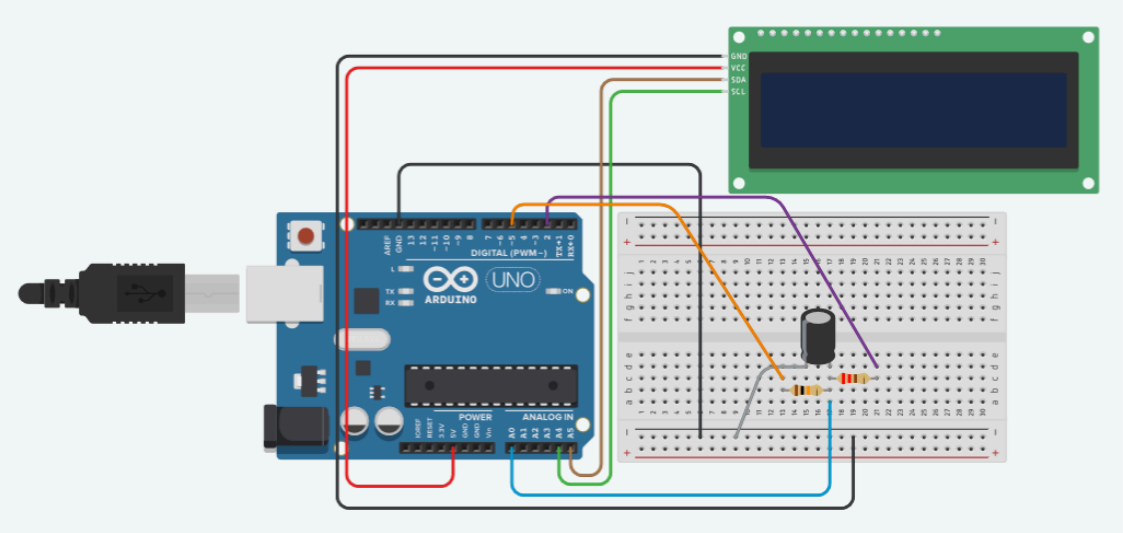
is ruled via the relationship τ = RC, in which R is the resistance in the circuit and C is the capacitance. We will use this formulation to get the capacitance size later.

In sensible phrases, capacitors with large capacitance values have longer time constants due to the fact they can keep an extra charge, resulting in a slower voltage upward thrust in the course of the charging procedure. This way, smaller capacitors charge quicker, displaying shorter time constants. Our multimeter uses this principle with the aid of measuring the time constant to decide the capacitance, the use of timers in C. By applying a known voltage to the capacitor (5V) and monitoring the time it takes for the voltage to reach the 63.2% threshold, through ADC voltage reading function, with this records our device can calculate the capacitance precisely. This method ensures unique measurements, accommodating a huge range of capacitance values, in our case from 1 µF to 3.9 F, and is important for imparting dependable information displayed at the clever glasses interface.

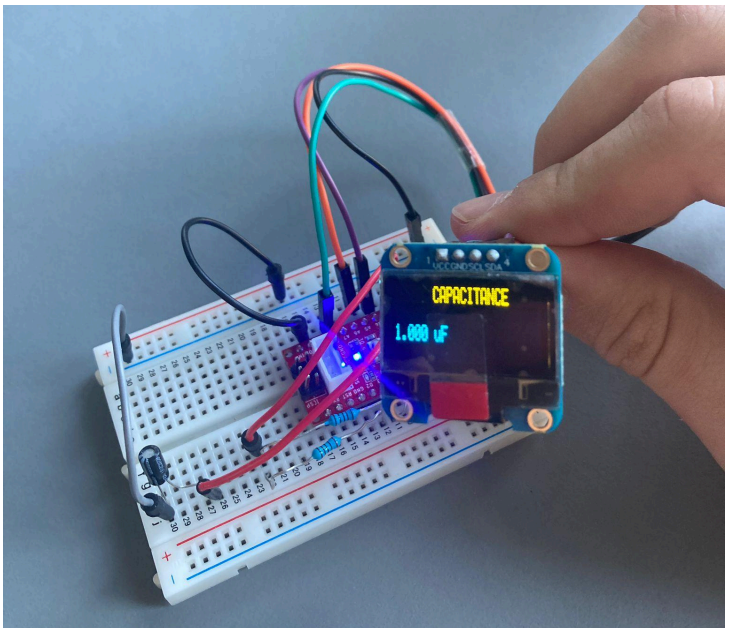
REQUIRED COMPONENTS

- Arduino Nano
- 220 Ω resistor
- 10 kΩ resistor
- I2C Display

ELECTRONIC CIRCUIT



(First concept of Capacitance made with Tinkercad)



(Individual circuit for CAPACITANCE fully working; measuring1uF capacitor)

4.2.2. Resistance Meter

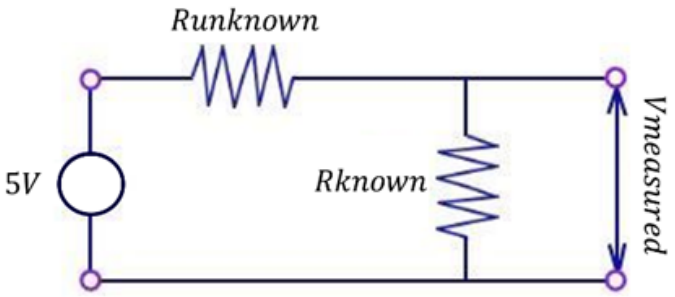
After previous investigation, it was discovered that resistance was only a simple circuit that needed a couple of resistors in series, one to measure and one for reference, in a simple voltage divider. The unknown resistor would also be receiving a voltage of 5V from the Arduino. With all this information, the calculation would be done by measuring the voltage in the middle point between the two resistors and following the formula:

$$Runknown = Rknown \cdot \left(\frac{5V}{Vmeasured} - 1\right)$$

(Fig. 4.2.2.1.)

In the formula, being the two Rs the resistors and $Vmeasured$ the voltage received in between the resistors. In addition, the $5V$ represents the v3oltage we are sending through our pins [7].

A voltage divider with all the operators and constants from before would therefore look like this:

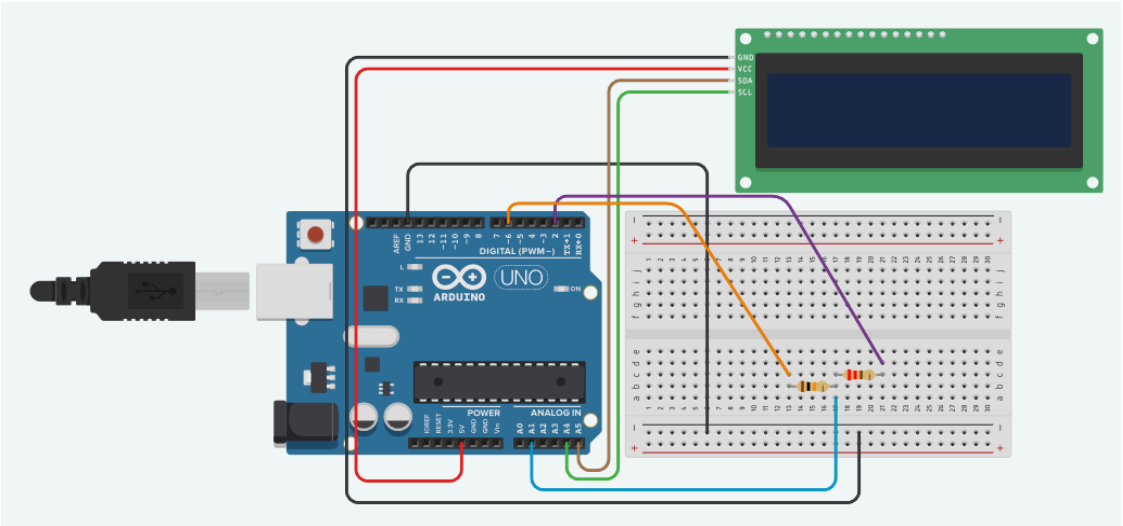


In the circuit for the resistance meter, we will use pin 1 (PIN C1) to measure the voltage between the resistors while outputting 5V $Runknown$ from the digital PIN D2 and $Rknown$ to PIN D6 without any voltage.

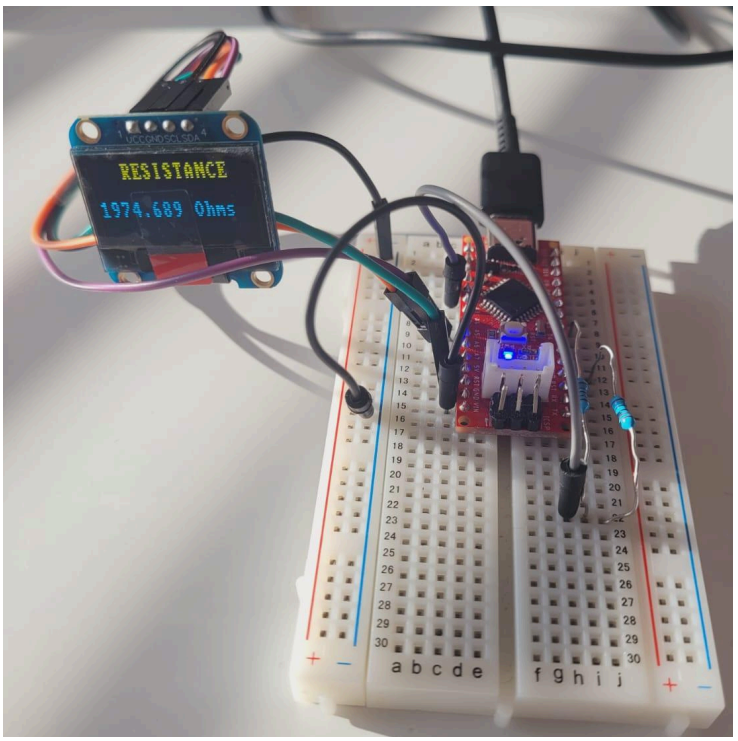
REQUIRED COMPONENTS

- Arduino Nano
- 10 kΩ resistor
- I2C Display

ELECTRONIC CIRCUIT



(First concept of Resistance made with Tinkercad.)

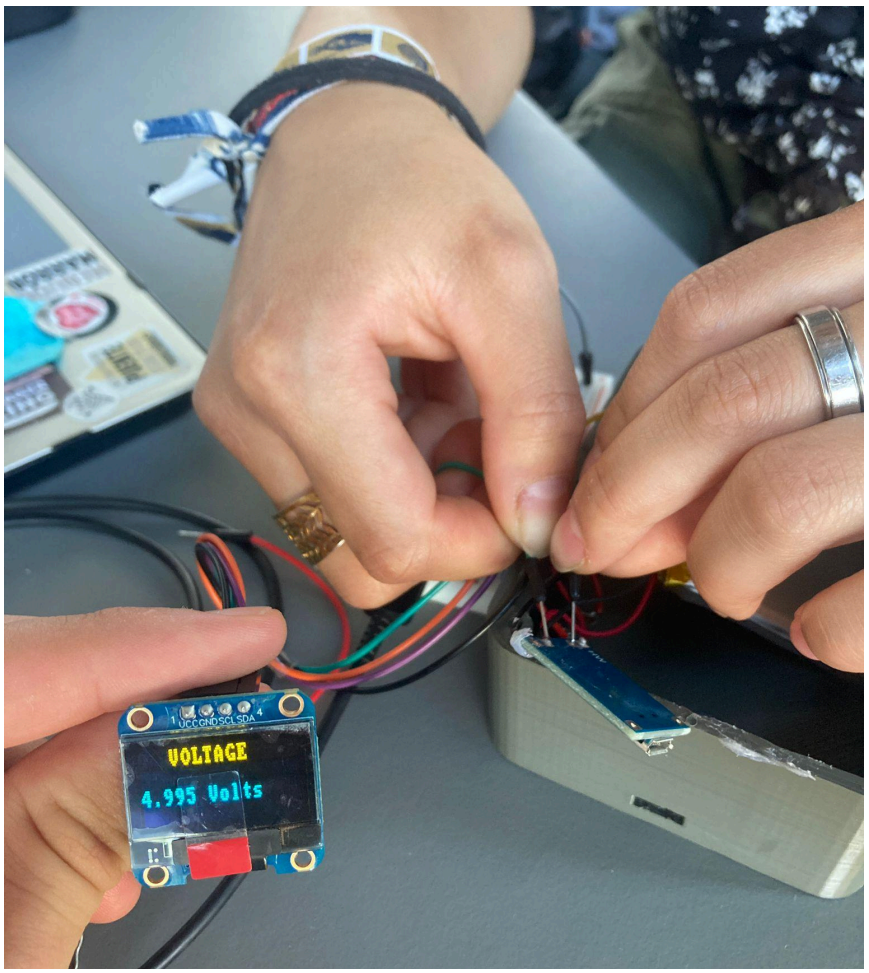


(Individual circuit for RESISTANCE fully working; measuring 2kΩ resistor.)

4.2.3. Voltage Meter

The circuit in the voltage meter is no different from the one used in the resistance meter from the previous section, as it is essentially measuring voltage. The voltage divider [2] consists of two resistors of which one is unknown. Then its value can be calculated by measuring the voltage in between, but if there were no resistor, the circuit would only measure voltage and make calculations for a non-existent resistor.

Having clarified the relation of the resistance meter to the voltage meter, then the issue of differentiating what is being measured should be dealt with in the software of the multimeter.



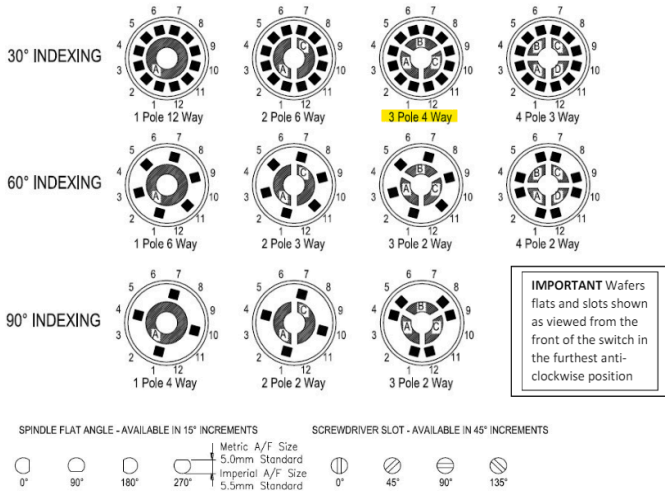
(Testing the voltage meter on our battery.)

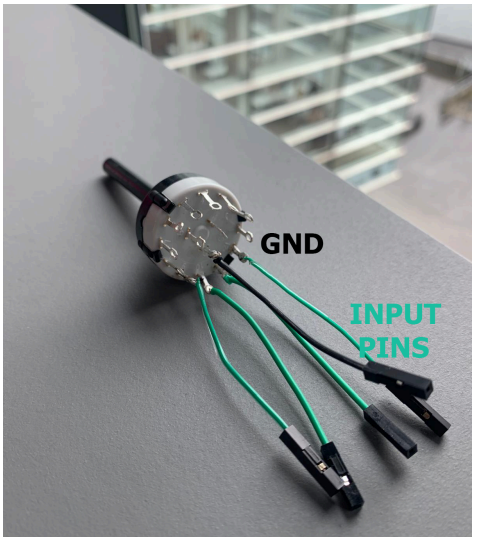
4.2.4. Rotary Switch

Continuing with the rotary switch, this type of switch will allow us to switch between the modes of our multimeter by turning a knob as if it were a potentiometer.

This specific switch has 3 inputs and 12 outputs. For installing the switch we have to connect one of the center pins to the ground (GND) and the output pins depending on how many modes we are going to develop in our multimeter, it was also required to solder 1KΩ resistors to the output pins, these resistors were going to work as Pull-Up Resistors. At last, we soldered female jumpers to the pins for easier plug-in capabilities between the rotary switch and the Arduino.

At the time of soldering, we decided to connect 4 outputs, the first one would be dedicated to the power-off state, the second position to the Capacitance Meter, a third position to the Resistance Meter, and the last position for measuring voltages. This particular switch has a modular capacity that allows us to limit the number of outputs to use by moving a washer depending on a certain angle. As we were only going to use 4 out of the 12 outputs from the rotary switch we had to set the rotary switch on a 3 pole - 4-way configuration setting the washer at a 90-degree angle, which allows us to have 1 input dedicated for every 4 outputs and 4 positions while





turning the knob.

Deciding to use the rotary switch brought us new knowledge, going from the SDU Shield that was almost plug-and-play to a device that we had never used previously and that would give us some trouble.

4.2.5. Battery switch

To make our project work while unplugged from the laptop we needed a battery, we decided to use a LiPo battery with a capacity of 2000mAh and a 3.7v, which will provide the necessary energy to power the Arduino and the display. For

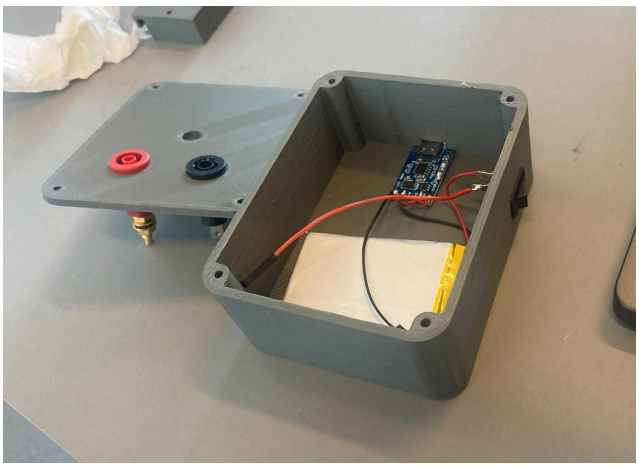
charging the battery and transferring 5 constant volts to the Arduino, we had to use a TP4056 charging module that charges the battery at 1A by plugging a micro USB.

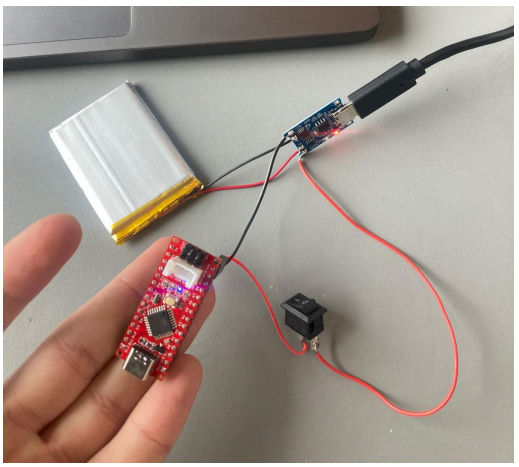
To turn on and off the Arduino and the screen, we added a switch between the charging module and the Arduino, located on the side of the box, now we had easy access to turn our on and off our project when desired.

When we first connected the battery to the charging module we had some problems with the overheating on the charging module, the battery was not overheating at all, so we thought that the problem might have been on the charging module. To check it we first attempted to charge the battery with a battery charger, isolated from the charging module.

Once the battery was charged and checked to be working fine, we could confirm that the problem was within the charging module. We then replaced it with a new one.

Now that we had a working power supply, we were capable of delivering the 5V needed to power our project.





4.3 Code

In the first place, the different properties had separate circuits, therefore we started by dividing the software tasks into three: resistance, inductance, and capacitance. Meaning to code every task individually and then put them together with the potentiometer to control the state to measure.

We then proceeded to do some research about what would be needed for each electrical magnitude. The fact that we found many other projects that were able to measure all three properties with an Arduino UNO or even an Arduino Nano helped us decide we would be working with the Seeeduino Nano. We had previous experience with this microcontroller, and it would make it more practical to work with through Visual Studio Code, also part of our previous project in the first semester. The choice of our microcontroller helped us with the pace of our progress in the codes of the multimeter, as we were expected to implement what we were taught in the course.

Once decided what should be needed in each circuit, the coding in C language was divided into investigating and applying concepts from the Embedded course. After having individual circuits and programs that turn out to be fully functional, we would proceed to put everything together and print it on the designated OLED screen (SSD1306). In terms of practicality, the use of a rotary switch to make them work simultaneously when it stops at a certain position was a supplementary component that also needed software control.

4.3.1. Capacitance Code

An Arduino can measure capacitance because the time a capacitor takes to charge is directly related to its capacitance by the next equation:

C = TC / R

where:

- TC is the time constant of the capacitor (in seconds).
- R is the resistance of the circuit (in Ohms).
- C is the capacitance of the capacitor (in Farads).

For measuring capacitance with our DIY multimeter, we are going to make use of the Arduino, to charge a capacitor through a known resistor. The time it takes for the capacitor to charge up to a certain level of voltage is measured and used to calculate the capacitance. This process investigates the essential relationship between the time it takes the capacitor to charge and its capacitance.[6]

Here's a step-by-step guide through the process:

1. **Charging the Capacitor**: The first step is to charge the capacitor, by connecting it to one of all the digital pins of the Arduino framework through a resistor, which in our setup of the circuit is a 1010KΩ resistor connected to a D5 pin. Later on, it is an important part of the process to know the exact value of the resistor as it directly impacts the accuracy at the time of measuring the capacitor.

   ```
   PORTD |= (1 << chargePin); //chargePin HIGH
   ```

2. **Measuring the Voltage**: Using the Analog-to-Digital Converter (ADC) of the Arduino, we can measure the voltage across the capacitor. The Arduino starts charging the capacitor, and simultaneously, we begin counting the time in microseconds. The voltage is measured using the analog input A0.

   ```
   uint32_t startTime = millis();
   ```

3. **Voltage Threshold**: The Arduino ADC has a 10-bit resolution, which means it can represent voltages from 0 to 5 volts in 1024 discrete steps. Thus, 0 volts is represented as 0, and 5 volts is represented as 1024. The target is to

measure the time it takes for the capacitor to reach 63.2% of the full charge. For a 5-volt system, 63.2% corresponds to an ADC value of approximately 648.

```
while (readADC(analogPin) < 648); //wait until capacitor reaches value
```

4. **Stopping the Charge**: When the analog read value reaches 648 (63.2% of full charge), we stop charging the capacitor and also stop the time counter, and store the time it costs to charge it. This measured time represents the time constant TC.

```
elapsedTime = millis() - startTime;
```

5. **Calculating Capacitance**: With the known resistance (R) and the measured time constant (TC), the capacitance (C) can be calculated using the following formula:
   C = TC / R
   The capacitance calculated value is later processed and therefore ready to be displayed.

```
microFarads = ((float)elapsedTime / resistorValue) * 1000.0;
```

6. **Range Limitations**: One of the main challenges of this setup is the range of capacitance values that can be accurately measured. Due to the limitations of the Arduino and the simplicity of the circuit, it is impractical to measure capacitors ranging from 1 pF to 100 F with a single setup. Therefore, we limit our multimeter to measure capacitance values between 0.1 µF and 3.9 F. This range is chosen to balance precision and practicality.

7. **Displaying the Results**: The final capacitance value is to be displayed on the I2C display, which is connected to the Seeeduino. In this precise project, we are using the SSD1306. This display provides a user-friendly interface capable of displaying the measured capacitance. This calculated capacitance is continuously updated in a loop to reflect the latest reading, ensuring that the user has accurate and current information as long as the multimeter is asked to measure capacitance.

8. **Preparing to measure again**, once we measure the current capacitor, we need to make some adjustments to be ready to measure again, we need to make sure that the capacitor is fully uncharged to measure again.

```
PORTD &= ~(1 << chargePin); //chargePin LOW

DDRD |= (1 << dischargePin); //dischargePin OUTPUT

PORTD &= ~(1 << dischargePin); //dischargePin LOW


 while (readADC(analogPin) > 0);//wait until capacitor
discharges


DDRD &= ~(1 << dischargePin); //dischargePin INPUT
```

4.3.2. Resistance Code

Taking all the previous information into account, the coding would proceed to comply with all of the contents of the formula. First and foremost, we would need to give $5V$ to the unknown resistor [8]. To control this as desired, together with the other codes, we would need to connect it to a digital pin (PIN D2) and set it to high to have

an output of the desired $5V$. Another digital pin (PIN D6) would be connected to the opposite end of the other resistor, and as we did in the previous step, connect it to a digital pin to control its state as well. To do as said, the DDRD and PORTD must be set as presented in the following snippet:

```
DDRD = 0b01000100; //set D2 AND D6 as OUTPUT
PORTD = 0b00000100; //set D2 as HIGH; D6 as LOW
```

This being made, to measure the voltage in the middle of the series resistors we would need to use an analog pin, in our case PIN A1. At first, the information being received would be an analog signal to be transformed into digital. In this step an ADC function, whose code snippet is shown below (Fig. 4.3.2..1), would take action and make it possible for our Seeeduino to read it and process it correctly. Previous to calling the function it would be necessary to set the ADC registers ADMUX (for voltage reference and analog PIN A1) and ADCSRA (to enable ADC). Once it is correctly set up, to call the function it would be necessary to select the voltage reference in ADMUX, set the prescaler to 128, and turn on the module in ADCSRA.

```
//ADC function

 uint16_t adc_read(uint8_t adc_channel){

  ADMUX &= 0xf0; // clear previously used channel, but keep internal
reference

  ADMUX |= adc_channel; // set the desired channel


  //start a conversion

  ADCSRA |= (1<<ADSC);

  //wait for conversion to complete
```

```
  while ( (ADCSRA & (1<<ADSC)) );

  //return to the calling function as a 16 bit unsigned int
  return ADC;
}
```

(Fig. 4.3.2.1.)

Once the digital signal ($digital$ in the formula in Fig. 4.3.2.2. further down) is obtained it would be necessary to convert it to volts to work with a voltage in the main formula. The formula to follow in the process of conversion would show as:

$$Voltage\ read = \left(\frac{digital \cdot 5V}{1024}\right) V \qquad \text{(Fig. 4.3.2.2.)}$$

Having all the data needed, now it was only a matter of introducing the data into the known function shown previously (Fig. 4.2.2.1.). With the help of a buffer, we calculated the value of $Runknown$. After some trials, we realised we needed a value for tolerance which we subtracted 30 from the original value of the formula to have greater accuracy in the results. On the other hand, we would be sacrificing accuracy in resistances under the value of 65Ω, for which we designated a fixed value of 10Ω, this being the most common value to measure below the value.

The reference resistance (previously mentioned as $Rknown$) we chose to work with was 10kΩ, which allows us to measure resistances as high as 10kΩ. These two resistances we work with have measurements of kΩ, therefore it makes sense to work with smaller variables and have them multiplied by 1,000 to have the right units in the outcome as well as to have the smaller resistances expressed without any

decimals. The range having its maximum at 10kΩ, resistors out of range would have an indicator as such.

All this prior information applied in the coding would result in the looped calculation of $Runknown$ (R2 in Fig. 4.3.2.3.) before deciding what should be printed out, which depends on whether the resistor is inside the accepted values or not (65Ω-10kΩ), having in mind that, as mentioned before, 65Ω or lower leads to a fixed 10Ω due to impossible accuracy at such levels.

```
analogValue = adc_read(ADC_PIN); // Read analog Voltage

//convert to Volts
buffer = analogValue * Vin;
Vout = (buffer)/ 1024.00;

buffer = (Vin/Vout) -1;
R2 = (Rref * buffer*1000) - 30; //*1000 because we express it in ohms
// -30 due to tolerances
```

(Fig. 4.3.2.3.)

When displaying the value of the resistor, there was a slight problem with the measurement units (Ohms). The LCD libraries would not allow printing text along with the variable, therefore we created a function (Fig. 4.3.2.4.) that would make the

```
//function used to print variables in the OLED with units, not only
the number
void DisplayFloatResistance(float resistance2) {

  char buffer[20]; //buffer to hold the formatted string
  sprintf(buffer, "%.3f Ohms", resistance2); // to format str of the
R2 value

  SSD1306_DrawString (buffer); //print R2 value in OLED
}
```

float value of the resistance into a string to be a char array and then printed, which works along with the functions of the SSD1306. Working with this new function would permit printing the final value along with Ohms. (Fig. 4.3.2.4.)

4.3.3. Voltage Code

After taking care of the resistance research and coding, it turns out that the same circuit that measures resistance and its corresponding code already measures voltage. Together with the voltage divider designed before, the calculations $Rknown$ were based on formulas requiring the measuring of a voltage ($Vmeasured$ in Fig. 4.2.2.2.) through an analog pin (PIN A1). Having coded the resistance meter, the same circuit and code would be capable of measuring volts.

Therefore, the only difference should be the magnitude to print as the output in the display, either the voltage directly measured or the resistance calculated with it. In this section, to print voltage we would simply be printing the value of the voltage (Vout in Fig. 4.3.2.3.). As it happened in the resistance, to allow printing the voltage value along with its measuring units (volts), another function was designed but with the corresponding units:

```
void DisplayFloatVoltage(float voltage) {

  char buffer[20]; //buffer to hold the formatted string
  sprintf(buffer, "%.3f Volts", voltage);

  SSD1306_DrawString (buffer);
}
```

4.3.4. Rotary Switch Code

Programming the rotary switch [4] shouldn't have been difficult, but inconveniences of how C# language interprets the code and some troubles with the PINCs happened. So one of the rotary switches is connected to the digital pin A2 to the Arduino and another is connected to A3, one dedicated for selecting the Capacitance Meter and the other one for selecting the Resistance Meter. So the PINC line sets the behavior of the connected circuit, for each of the output pins from the switch as an input signal, at the same time PORTC line sets up the initial condition of the switch that should be NOT PRESSED, with a pull-up resistor, the input pin will read a high state when the button is not pressed (1). In other words, a small amount of current is flowing between VCC and the input pin (not to ground), thus the input pin reads close to VCC. When the button is pressed, it connects the input pin directly to the ground (0). At the time of including it into the code we didn’t get it to work, maybe some incompatibilities between the digital PORTS (C) or some misinterpretations by the C code, but when trying to write the conditions with the different measurements the screen entered in a loop switching between them like there were on a loop. Otherwise working with simpler codes like only displaying some text on the display, the rotary switch works perfectly.

4.3.5. Oled Display Code

The majority of the students decided to go for a Nextion Display, an option that had been already used in the First Semester Project. The dimensions of the Nextion Display won’t fit our requirements, we needed something very small, capable of fitting on a pair of glasses. So we chose an OLED Display called SSD1306 [3], of 0,96 inches and 128x64 bichromatic pixels (Blue and Yellow) with an I2C interface, compatible with the Arduino board. This display can work by 3,3V or by 5V, the SDA pin goes to the A4 pin, SCL to A5, and the last GND goes to any ground from the board. When programming there were a lot of libraries, for example,

the ADAFRUIT_SSD1306 library, which was only working in ARDUINO IDE or C++ language, or u8g2 and u8x8 library, both compatible with C and C++ language, but we were not able to get them to work, so finally we got to get the display to show text with a library called ssd1306, that only had two more dependencies in the libraries font and twi (Two Wire Interface).

When coping with the issue of flipping the screen display to read it from the mirror reflection to the methacrylate, the first approach was to flip the words to text and directly try to print the mirrored words, but the library would not support the characters.

The previous investigation demonstrated that the flipping of the display would need more work in the code, but a solution in the physical structure of the glass model would be to place two mirrors instead of only one to double-reflect the image and obtain the correct orientation of the measuring. We considered working on the design of the glasses but did not give up on the code.

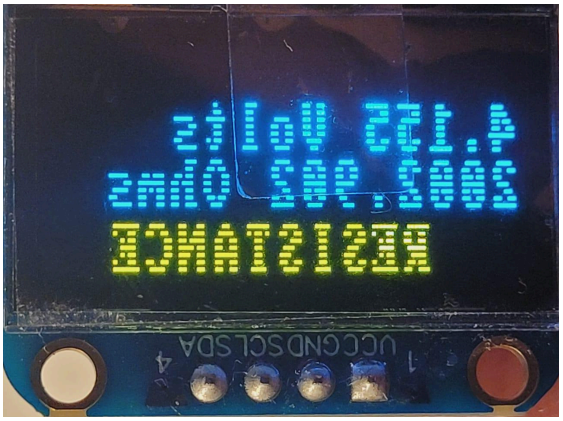
The next solution we tried to find was changing the libraries themselves to tr

y and flip the output by changing the COM Output Scan Direction to vertically flip whatever it was projecting in the first place (Fig. 4.3.5.1.). This information was found in the section 10.1.14 Set COM Output Scan Direction (C0h/C8h) of the datasheet. After many tries of library manipulation, the font.h library was then adapted to change the first font to a thicker one of 8x8 instead of the previous 5x8. As well as by manipulating the ssd1306.h we finally managed to get the display to print the text mirrored in the Y-axis.

```
#define SSD1306_COM_SCAN_DIR      0xC8      // originally 0xC0
#define SSD1306_COM_SCAN_DIR_OP   0xC0      //originally 0xC8
```

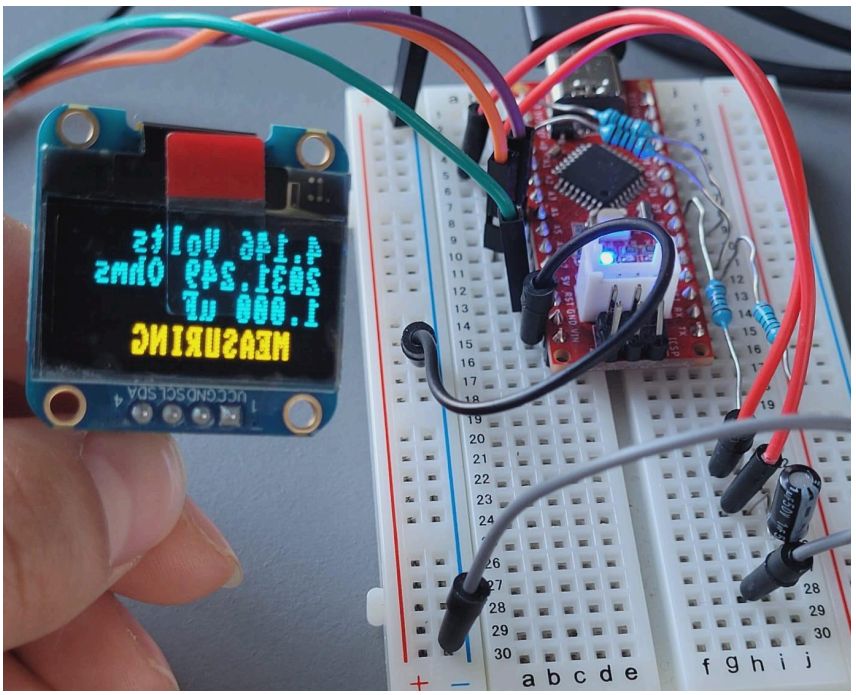
(Fig. 4.3.5.1.)

To test the new changes made to the coding worked, we proceeded to print the results from a resistor measurement and check that the mirroring would work as it should:



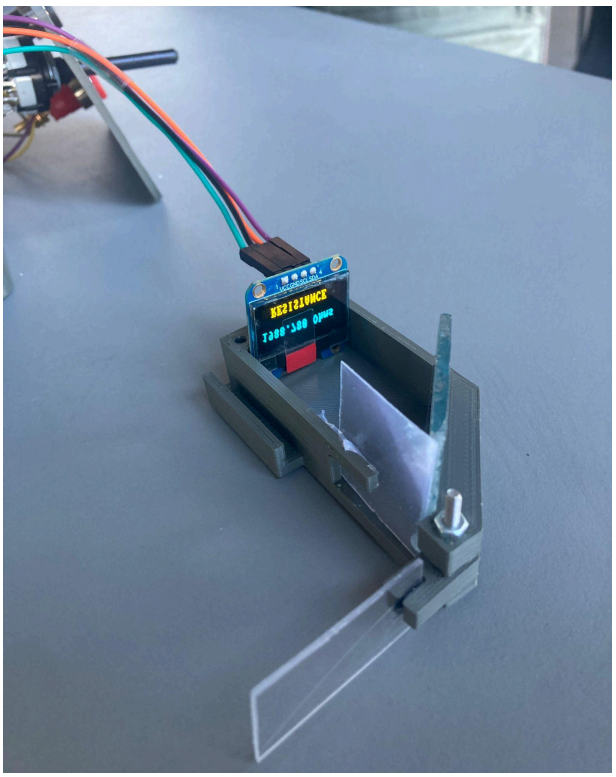
5. Test, Setup and Results

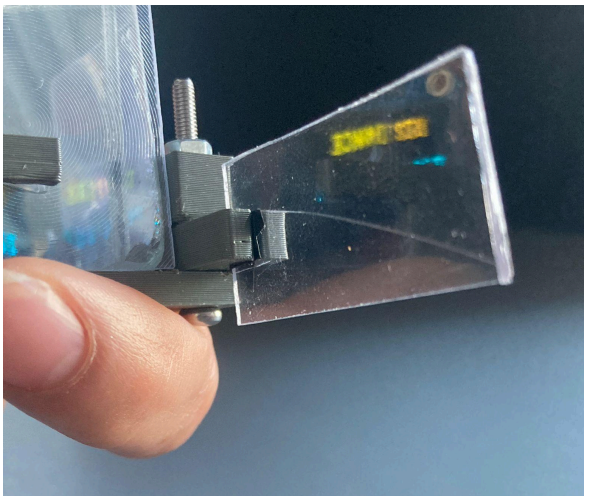
In the final product of our multimeter with incorporated projecting glasses, we tested the capability of the circuit to measure all magnitudes in the same circuit without contradictions. To do that we measured: voltage, resistance, and capacitance.



(Measuring: 2kΩ resistor, 1μF capacitor, and voltage through the resistor)

Not only do we have to correctly measure several passive components, but send the result of the measuring to the OLED in the glasses. This display would in the end be connected to the multimeter box through some cables and project the image into the methacrylate as appears:



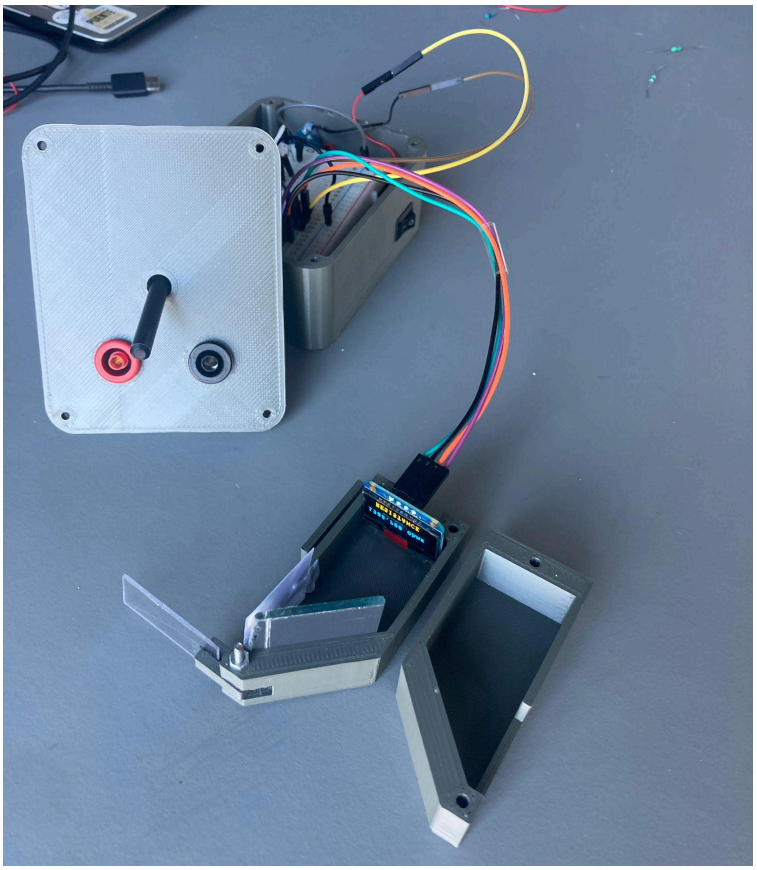


(the displayed information looks better in real life, it was difficult to capture it on camera)

About one of the requirements we were given, our initial concept for incorporating a DC motor involved attaching the motor that would typically be used to power up the lights on a bicycle, which we planned to use to manually recharge the battery of our multimeter. However, due to time restrictions and diverse issues during the project execution, we were unable to implement the previously planned mechanical capabilities.

The process of designing smart glasses proved to be more challenging than we would have thought in the first instance. With every new problem we encountered, the importance of prioritizing the functionality of the smart glasses rose above the additional feature of using the DC motor for battery charging. We also prioritized the functionality of the crucial capabilities of the multimeter. As a result, we redirected our focus towards the primary aspects of the project proposed to us, to ensure that the primary aspects of our project were completed and met our quality standards.

By the end of the project period and the due date for the report, we had a functional circuit to measure ohms, farads, and volts, along with its proper encapsulating box and wire leads for an unchallenging maneuver of measuring. We also had a set of glasses that projected measured values into a screen supported by the legs of a designated pair of glasses.



Objectives to improve before SPRO-2 exam

1. Polish the glasses holder so the information displays more sharply on the methacrylate. We will do this by trying other angles on the mirror so it can reflect the information more clearly than before.

2. Solder everything together, right now we are using a breadboard to connect all the circuits, as the code for the rotary switch that combines everything is not 100% working and we do not want to mess it up soldering if we have to change some positions later. We already have the board where we will solder it, but we will wait until the code is fully finished.

3. Make the rotary switch work along with the reading in the analog pins. At the moment the magnitude measuring and the rotary switch work correctly in separate testing but when put together there is a confronting preset of the ports.

4. Make it attachable to a belt, as we mentioned in the brainstorming we wanted to make the box easy to carry around mounted on the user's belt. We didn't have the time to craft a functional design, but we think it's an easy improvement we can develop to improve the functionality.

6. Conclusion

In conclusion, for this Second Semester Mechatronics Project, we were tasked with designing and manufacturing a multimeter that incorporates mechanical features along with a Nextion display screen.

As group 7 we managed to create a multimeter integrated into smart glasses. This has been a challenge in the development of the mechanical integration and the programming of the device.

Initially, the design of the glasses had a few defects, especially in the projection and visualization of the measurements on the OLED screen coupled to the glasses. After several attempts and searching for solutions, we managed to adapt the code and solve the problems of the visualization of the measurements reflected in the Y axis.

The team faced time constraints and problems during the project which made it impossible to implement the mechanical functionality of the DC motor to manually recharge the multimeter battery. Instead, we sought to prioritize the main functionality of the multimeter so that it could correctly measure ohms, pharaohs, and volts, effectively presenting the results in smart glasses.

Overall, we managed to meet the essential requirements: We created a multimeter integrated into smart glasses capable of measuring two passive components (Resistance and Capacitance) using a Nextion Display. Despite not having integrated all the planned features, the results obtained demonstrate an innovative and functional solution that can be used as a basis for future developments in this area. We look forward to finishing and fixing our objectives so they are ready for the SPRO-2 exam on the 24th of June.

7. References

Smart glasses [1] We inspired our smart glasses design from this project which helped us get a reference for the angle of reflection needed to properly display all the characters in the center of the methacrylate.

Voltage divider [2] Basis for the measuring of the resistance and as a consequence of the voltage too.

SSD1306 Technical data [3] Information for correct manipulation of the display. Mostly used 10.1.14 Set COM Output Scan Direction (C0h/C8h) for the mirroring manipulation.

CK Rotary switch Technical data [4] We used this technical data sheet to get all the information needed to set up the rotary switch on our code.

What is capacitance? [5] Simple explanation of what is capacitance and how capacitors work.

Capacitance on Arduino [6] Example of measuring capacitance with Arduino (.ino), we used this information to sketch our C code.

Seeeduino Nano V.3 Eagle files and pinout/schematic [7] It is required to locate the different pins and to make digital simulations in Eagle.

Atmega328 datasheet [8] This datasheet goes deeper into all the different uses and functions of the motherboard.

APPENDIX

- Initial problem formulation, we decided to change our problem formulation because the first one we did was very general and we did not have as much information and knowledge about the project topic yet. We also did not think of our final idea yet.

- Full code on GitHub, we wanted to incorporate GitHub in our project to be able to code in the newest version of our code whenever we wanted, but as we divided the code parts we ended up not using it that much, finally we uploaded the final version of the code there to be available for everyone.