Principles of Measurement & Instrumentation I

Laboratory

PHYS417

Laboratory Report

Nuray BAŞARAN

2140689

Experiment 2-Arduino-Based Measurement Devices

Experiment Date: 16.10.2019

Submission Date: 23.10.2019

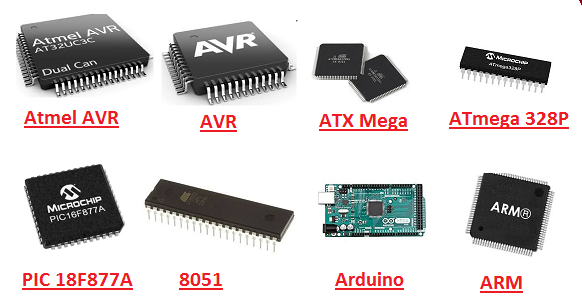
**0.1 OBJECTIVES**

The main purpose of this experiment is to teach us what is Arduino Uno, how it works and why we are using it. When we are learning our main purpose, we learned about the basic principle of the PWM(Pulse Width Modulation), and also the difference between microprocessors, microcomputers and microcontrollers. When we were using Arduino UNO, we learned functions in a software of Arduino and we controlled analog signals by using digital sources.

**0.2 INTRODUCTION**

**Microcontrollers**

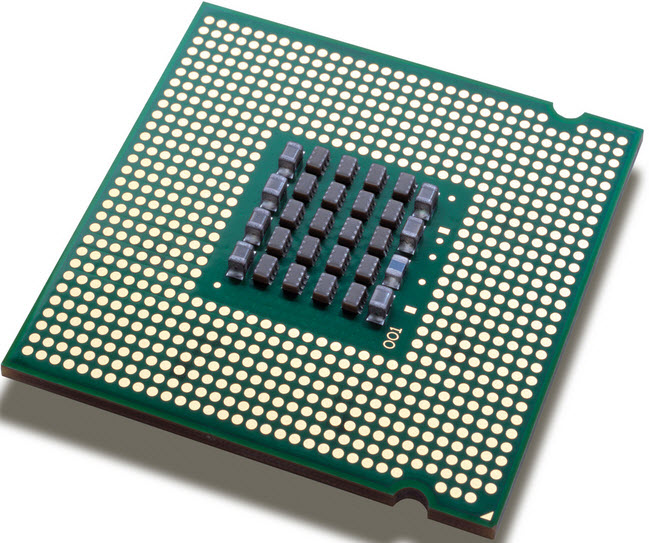
Microcontroller is a micro device which controls the other devices and machines so we call it “micro” and “controller”. A microcontroller contains its own processor and memory like a computer however it is used to solve small and specific tasks. Arduino is an example of microcontroller for this experiment.



*Figure1. Examples of microcontrollers*

**Microprocessor**

I am writing these sentences because of microprocessors. Basically, they take a lot of instruction such as addition, multiplication, division and other logical operations and perform them by using ALU (Arithmetic and Logical Unit). Register array acts like fast access memory for processing data and control unit(CU) controls the flowing of instructions and data on the system.



*Figure2.Example of microprocessor*

**Microcomputer**

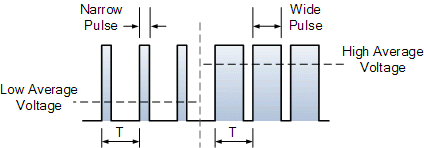
Microcomputers are designed for specific tasks. They are small, cheaper and handy. They include minimum of microprocessor, program memory, data memory and input/outputs.Also they might have timers, ADC etc…



*Figure3.Examples of microcomputers*

**Basic Principle of PWM(Pulse Width Modulation)**

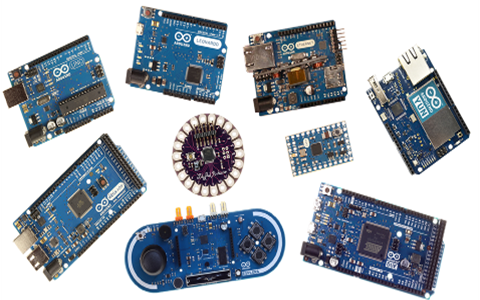
Analog signal has infinite resolution in magnitude and time so we can control it directly. This method provides us to control analog signal by using digital sources. By changing duty cyles, we can change width modulation so we can take more or less voltage.



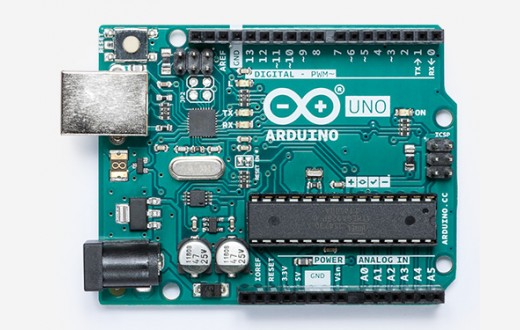
*Figure4.Pulse Width Modulation*

**Arduino**

They are open hardware boards to design devices. They are very easy to use because they can read inputs, light sensor, messages and they can turn them into output. On the board, microcontroller can take instructions which are sended by Arduino programming. We used Arduino Uno on this experiment.It has digital/analog inputs and outputs, pins, microcontroller and other circuits.

****

*Figure5.Examples of Arduino products*



*Figure6.Example of Arduino UNO*

**0.3 EQUIPMENT**

Electric board

Arduino UNO

Resistors

LED

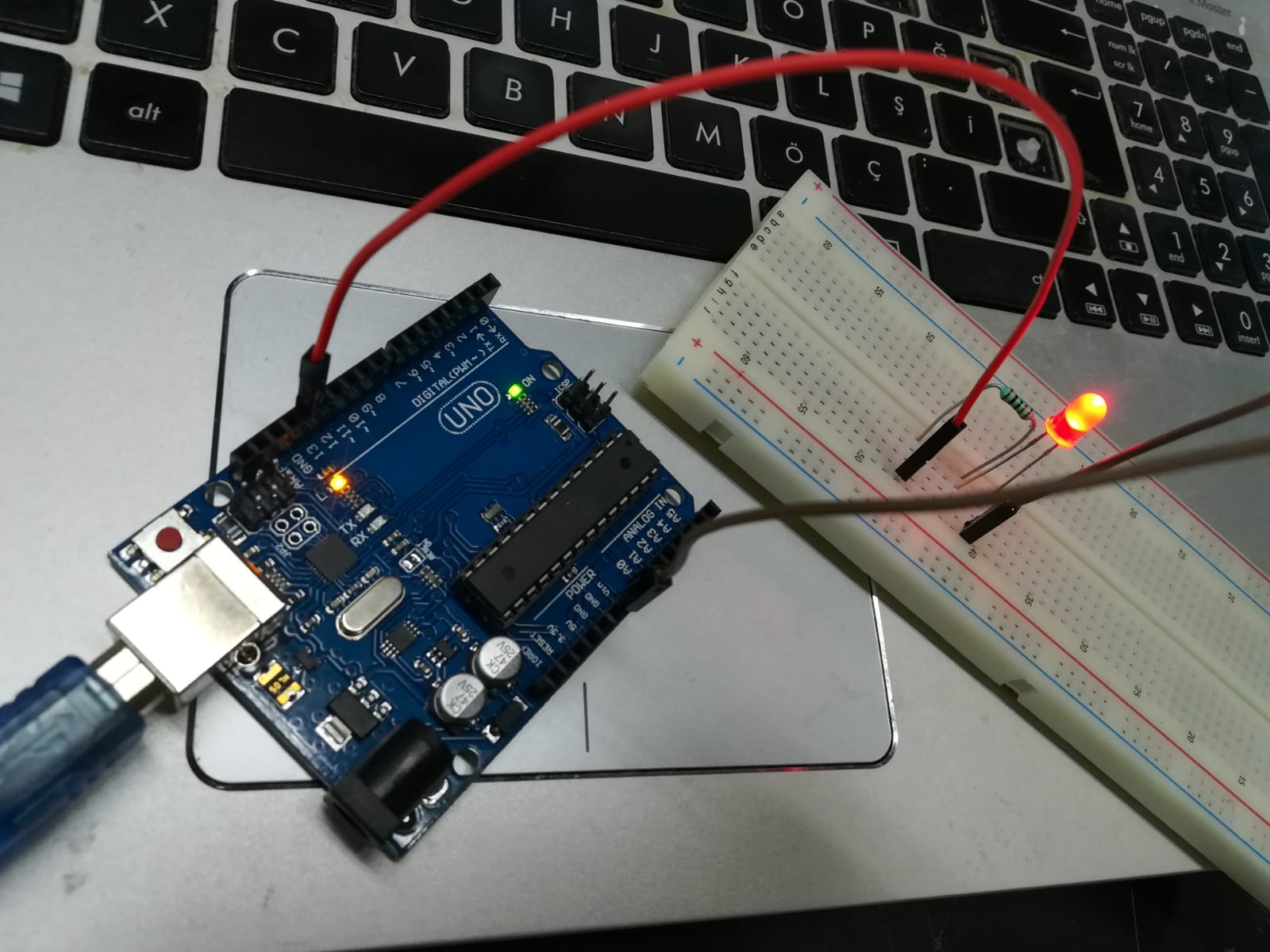
Push Button

Potentiometer

**04.PROCEDURE &0.5 CALCULATION**

1-I wrote a program on Arduino to blink an LED for one second. I used a current-limiting resistor when I designed the circuit because I wanted to reduce current so I can control the amount of current through the LED and I can protect LED from burning.

I connected the output of the resistor to Arduino pin 13 and then the long leg of LED to the resistor and the I grounded the short leg of LED on the Arduino UNO.



*Figure7. Designed Circuit for Step 1*

**Program to Blink LED**

void setup() {

// It set up Arduino to run. We make basic settings such as determining //which pin will be used for what or determining the transmission

// speed of data.

//These code is runned for one time when the program starts running.

pinMode(LED\_BUILTIN, OUTPUT);

// It set the specified pin as a INPUT or OUTPUT.

}

void loop() {

// It makes our operations to run again and again.

digitalWrite(LED\_BUILTIN, HIGH);

// It used to determine state of the digital pins.

//They can be OUTPUT and write or send data to our device.

//Also they can be INPUT and take or read data from components.

// Arduino UNO has a LED and resistor which are connected

// each other on a pin and LED\_BUILTIN takes this pin's number.

// This number is 13 in my Arduino.

//I already realized there was a light on Ardunio blinks with the LED at the same time.

// HIGH means 5V.

 delay(100);

// It pauses the program for specified time.

 digitalWrite(LED\_BUILTIN, LOW);

// LOW means 0V.

 delay(100);

// It pauses the program for specified time.

}

2- I used a 1k ohm potentiometer in this part. I connected one terminal to 5V and other one to ground and then I connected the output of the potentiometer to Arduino pin A0. I read the output by using some functions on Arduino.



*Figure8. Designed Circuit for Step 2*

**Program to Read Output of Potentiometer**

void setup() {

 //It set up Arduino to run.

**Serial**.begin(9600);

 // It adjusts rate of data in bits per second for transmission.

}

void loop() {

 //It makes our operations to run again and again.

 int sensorValue = analogRead(A0);

 //It reads A0 as an analog input. "int" means integer. "sensorvalue" is a thing we determined.

**Serial**.println(sensorValue);

 // It prints the values on serial port screen, so we can see it.

 delay(1);

 // It pauses the program for specified time.

}



*Figure9.Results on Serial Port Screen*

I see values 0 to 1019. It should be 1023 but because our potentiometer has some error inside, I see it 1019. The resolution of my Arduino analog to digital converter is 1023.

**Program to Read Results in Scale 0 to 5**

void setup() {

 //It set up Arduino to run.

**Serial**.begin(9600);

 // It adjusts rate of data in bits per second for transmission.

}

void loop() {

 //It makes our operations to run again and again.

 int sensorValue = analogRead(A0);

 //It reads A0 as an analog input. "int" means integer. "sensorvalue" is a thing we determined.

 int newvalue = map(sensorValue,0,1023,0,5);

 //I determined a newvalue and I used map function to adjust numbers to a new range.

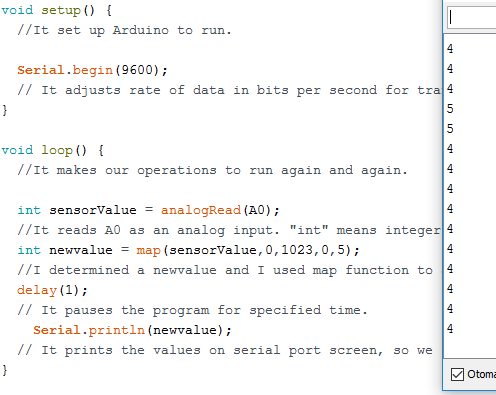
 delay(1);

 // It pauses the program for specified time.

**Serial**.println(newvalue);

 // It prints the values on serial port screen, so we can see it.

}



*Figure10.Results on Serial Port Screen*

I see the values 0 to 5 because I adjusted it on the program.

**Program to Read Values in Float Type**

void setup() {

 //It set up Arduino to run.

**Serial**.begin(9600);

 // It adjusts rate of data in bits per second for transmission.

}

void loop() {

 //It makes our operations to run again and again.

 int sensorValue = analogRead(A0);

 //It reads A0 as an analog input. "int" means integer. "sensorvalue" is a thing we determined.

 float newvalue2 = sensorValue \* (5.0 / 1023.0);

 // I determined a newvalue2 and I made it range 0 to 5.0 so I took float results.

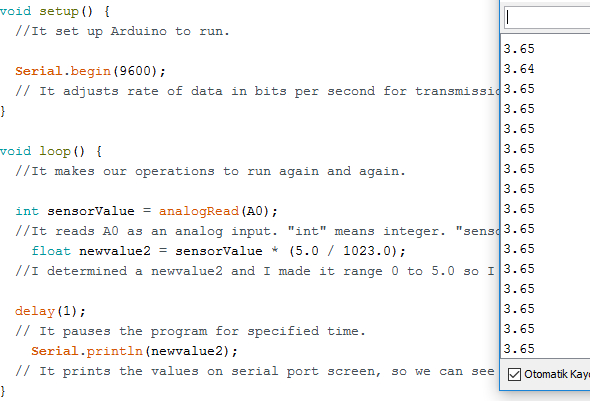
 delay(1);

 // It pauses the program for specified time.

**Serial**.println(newvalue2);

 // It prints the values on serial port screen, so we can see it.

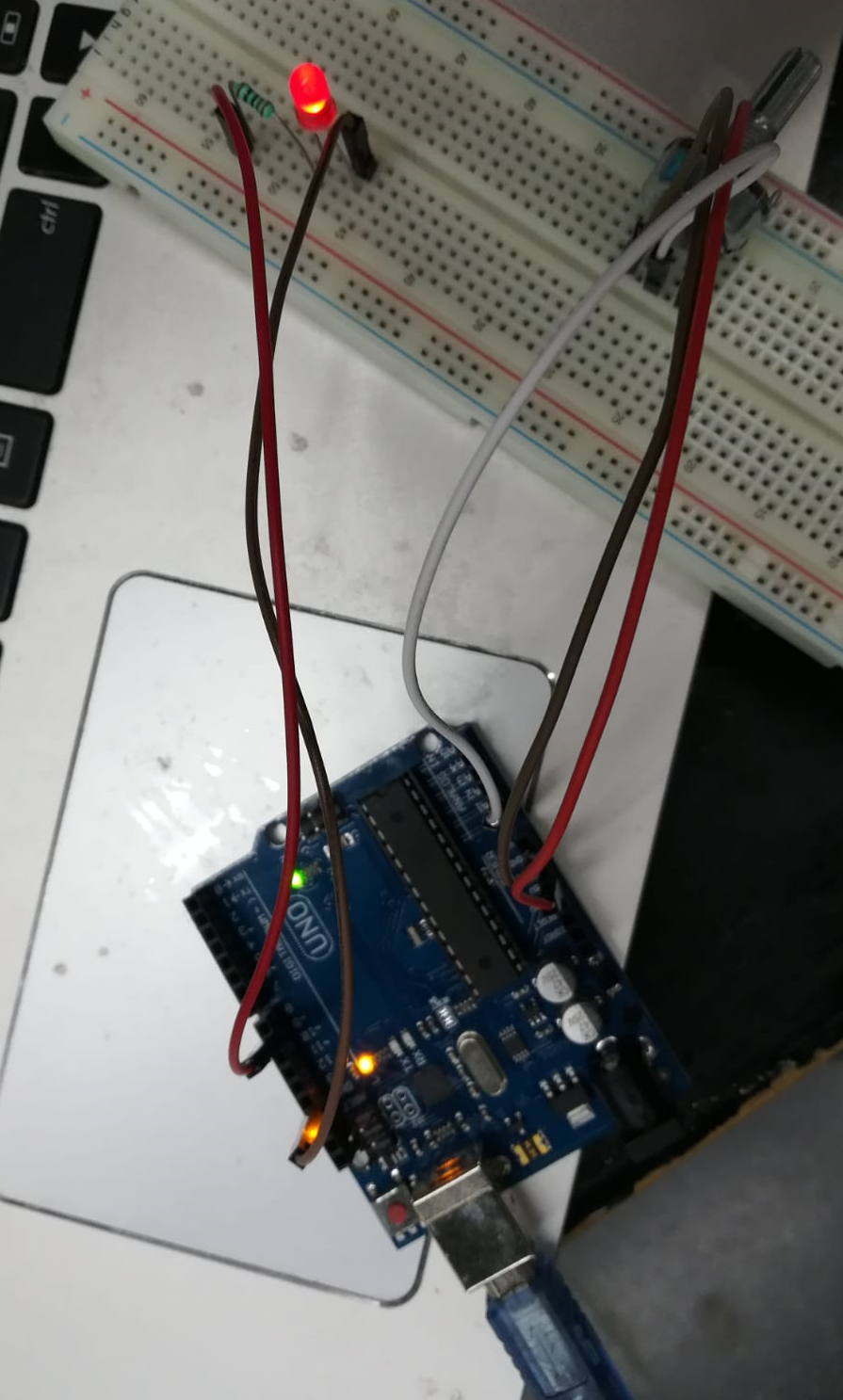
}

****

*Figure11.Results on Serial Port Screen*

I determined the range between 0 to 5.0 by doing divison then I defined the “newvalue2” as a float. Hence I took float results.

3- In this part, I combined the potentiometer and LED by using the same circuits as in step 1 and 2. I controlled the brightness of the LED by changing the resistance of the potentiometer by turning it.



*Figure12.Designed Circuit for Step 3*

**Program to Control the Brighness of an LED**

const int analogPin = A0;

//It makes behavior of the analogpin as a constant.

const int ledPin = 9;

void setup() {

 //It set up Arduino to run.

 pinMode(ledPin,OUTPUT);

 // It set the specified pin as a INPUT or OUTPUT.

**Serial**.begin(9600);

 // It adjusts rate of data in bits per second for transmission.

}

void loop() {

 // It makes our operations to run again and again.

 int analogValue = analogRead(analogPin);

 // It reads analogPin as an analog input. "int" means integer. "analogValue" is a thing we determined.

 int brightness = map(analogValue, 0, 1023, 0 ,255);

 // I determined the range of brightness 0 to 255 by using map function.

// 0-255 corresponds to%100 duty cycle.

 analogWrite(ledPin,brightness);

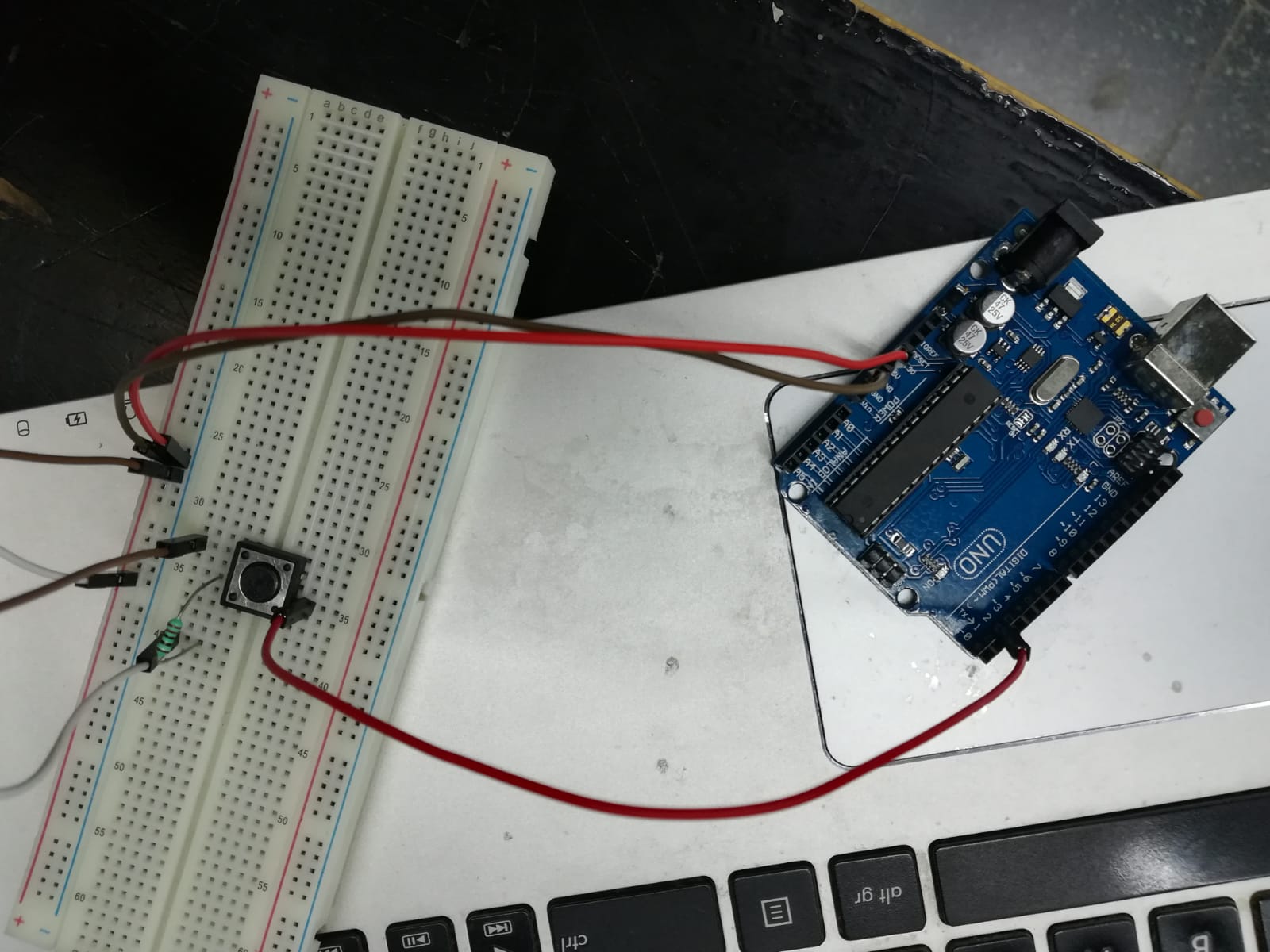
 //It writes analog value to ledPin.It is based on the PWM technique which is used for getting analog results from digital sources.

 delay(1);

 // It pauses the program for specified time.

}

4- I used a button and a resistor in this part. I made a relation pin 2 to button, button to resistor, resistor to ground and 5V to button. I designed a system to count numbers of clicks by using Arduino interrupt service routine. When I clicked to button, I saw that the value increased on the serial port screen.



*Figure13. Designed Circuit for Step 4*

**Program to Count the Number of Clicks on a Push Button**

int count = 0;

// It starts to count from 0.

const byte interruptPin = 2;

//When we click to button, we interrupt the program. To interrupt the program, we defined the pin 2 as an interruptPin.

void setup() {

   //It set up Arduino to run.

 pinMode(interruptPin, INPUT\_PULLUP);

 //If we write only "INPUT", pin listens whatever voltage is connected to pin.

 //If we write INPUT\_PULLUP,the pin is connected to internalpullup resistor. Hence we can read better readings.

 attachInterrupt(digitalPinToInterrupt(interruptPin), flag, RISING);

 // It determines a interrupt number. When we interrupt, the voltage comes from digital pin.

 // "Flag" let the program know there happens a interrupt.

 // There will be interrupt when signal goes from LOW to HIGH.

**Serial**.begin(9600);

   // It adjusts rate of data in bits per second for transmission.

}

void loop() {

   //It makes our operations to run again and again.

**Serial**.println(count);

   // It prints the values on serial port screen, so we can see it.

}

void flag() {

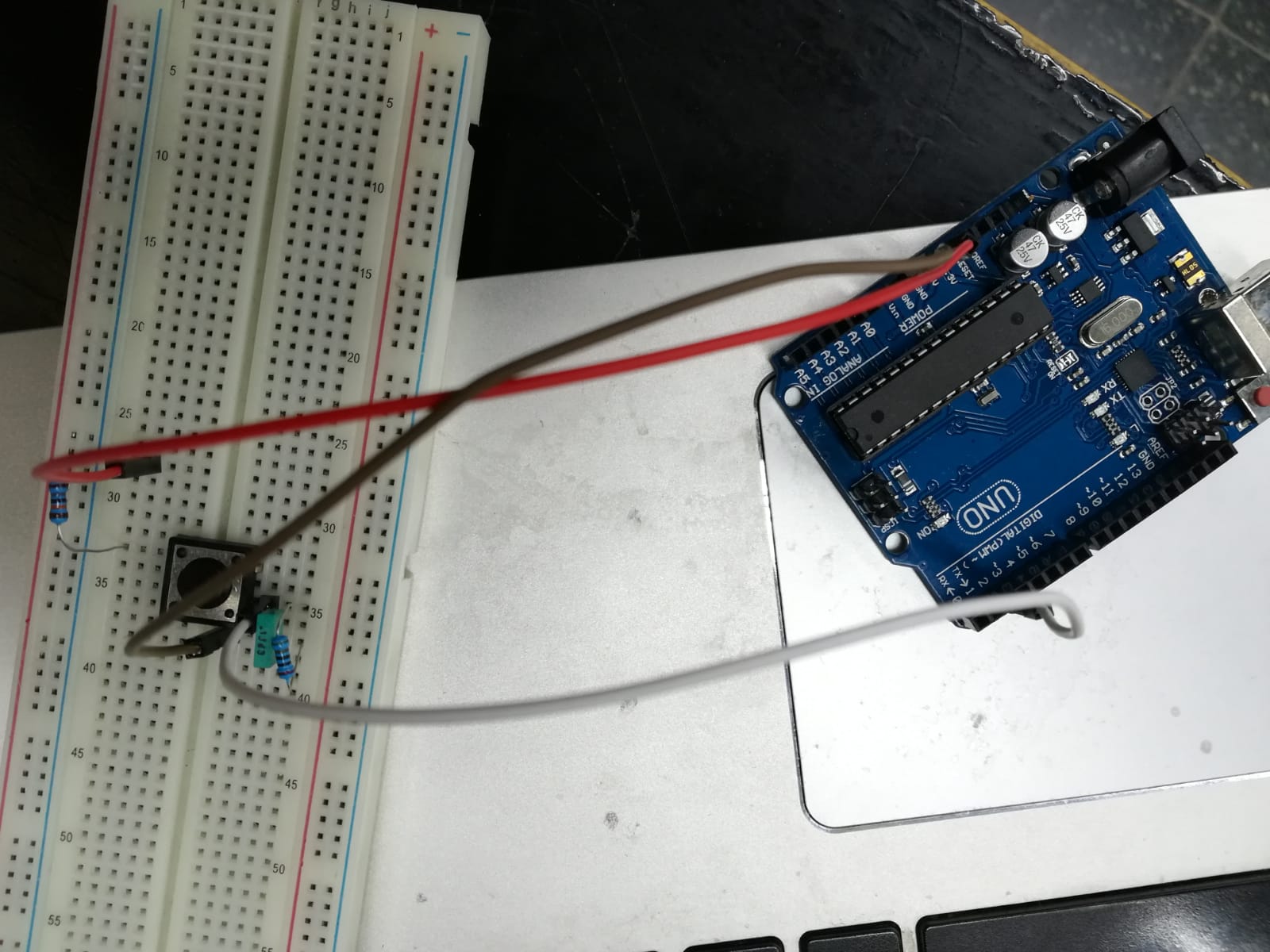
 // It provides to use flag.

 count++;

 // It increases when the flag increases

}

5- This circuit actually helped us to count the number of clicks better. We used 100nF capacitor, 120 and 2k ohm resistors and arduino. In step 4, when I pressed to button, the value on the screen sometimes increased 2 more. To get better results, we used capacitor in this step.



*Figure14. Designed Circuit for Step 5*

I used the same program which I used in step 4 on this step. My results are better than step 4. It increased one for each click because I used capacitor. But why?

When I press the button, two metal connect each other and there happens transmission. If the metals are not perfectly flat or aligned, there happens more than one contact in very small time. In the other words, there happens multiple triggering. We could solve this problem on software or hardware. We chose to solve it on hardware so we connectted additionally a capacitor and resistor to step 4 circuit. It is called RC Debouncing. Capacitor is used for filtering the instant changes in this step. When I started the program, the capacitor charges and discharges through the resistor so voltage rises slowly. Hence, there will be no unnecessary noises and so values. If RC is too short, we can’t prevent noises but if RC is too long, there will be no output for fast switch actuations.

**0.6 DISCUSSION**

Because I was using Arduino the first time in my life, I thought that it was going to be really hard and complicated. When I saw the functions on preliminary work, I thought that I had to write all codes by myself. I couldn’t think where to put these functions and how I could make connections between them. It happened because I met with Arduino very late and I didn’t search on Internet about codes. However, I think that I was no too bad, considering I used it for the first time. I lost my mind in the last part, because I used 200k ohm. It was a new experience for me.

**0.7 REFERENCES**

Arduino. (2019, October 15). Retrieved from https://en.wikipedia.org/wiki/Arduino.

Dahl, Ø. N. (2017, December 19). Current Limiting Resistor. Retrieved from https://www.build-electronic-circuits.com/current-limiting-resistor/.

Language Reference. (n.d.). Retrieved from https://www.arduino.cc/reference/en/.

Microcomputer System. (n.d.). Retrieved from https://www.sciencedirect.com/topics/engineering/microcomputer-system.

Williams, E., Williams, E., Somun, Williams, E., Mike, Nova, … Doug. (2019, October 20). Embed With Elliot: Debounce Your Noisy Buttons, Part I. Retrieved from https://hackaday.com/2015/12/09/embed-with-elliot-debounce-your-noisy-buttons-part-i/.

A Guide to Debouncing - Part 2, or, How to Debounce a Contact in Two Easy Pages, by Jack Ganssle. (n.d.). Retrieved from http://www.ganssle.com/debouncing-pt2.htm.