UNIVERSITY “POLITEHNICA” OF BUCHAREST
The Faculty of Electronics, Telecommunications and Information Technology

Determination of the voltage-illuminance plot

Coordinating teachers:
Prof.Dr.Ing Corneliu Burileanu
Assistant Crisitan Devan
Assistant Ana-Antonia Neacşu

Students:
Vlad Sterian
Cosmin-Iulian Dobrin

2020-2021

Content

I. Introduction

1. Project description

This project is meant to represent a system which tracks the position of the sun. It uses two servomotors powered by an 6V external voltage supply, a 3V photovoltaic cell that converts the light energy into a proportional voltage, four photoresistors through which the position of the photovoltaic cell can be adjusted. The voltage generated by the solar cell is displayed on an 16x2 LCD, along with the illuminance, measured in lux. All the components are controlled by an Intel Galileo second generation microcontroller.

2. Motivation

Solar panels are very useful and in demand nowadays, as more and more people are trying to switch to sources of regenerable energy, and they do just that. People are trying to move away from the power plants they have at home, by installing solar panels. Another field where having solar panels is useful is agriculture, where these panels are used to power irrigation systems.

The part where this project comes in handy is to maximize the amount of electrical energy converted by the panel, by automatically finding the best position to place the panels. Furthermore, our project represents a theoretical example of a much more powerful system, which can be used in the examples given previously.

Finally, our team is very excited about this project as it has high applicability in the lives of many people, and also, it will have a big impact in the future society. Another aspect that we find interesting is that such a system can save us money in the future.

II. Hardware resources

For this project the following components have been used:

- Intel Galileo second generation board
- 4 photoresistors
- 2 servomotors
- Breadboard
- Solar cell
- LCD
- 4 Resistors (1kΩ)
- 1 Resistor (5kΩ)
- 4 Batteries (1.5V, AA type)

1. Intel Galileo Board[1]

The Intel® Galileo Gen2 is a board based on the Intel® Quark™ SoC X1000, a 32-bit Intel® Pentium® processor-class system on a chip (SoC), operating at speeds up to 400MHz.

The board has built-in Ethernet with support for Power Over Ethernet(PoE), a USB 2.0 Host Port, micrso-SD slot, PCI Express mini-card slot, 20 digital input/output pins (of which 6 can be used as PWM outputs with 8/12-bits of resolution and 6 as analog inputs with 12 bits of resolution), a micro USB connection, an ICSP header, a JTAG header, and 2 reset buttons.

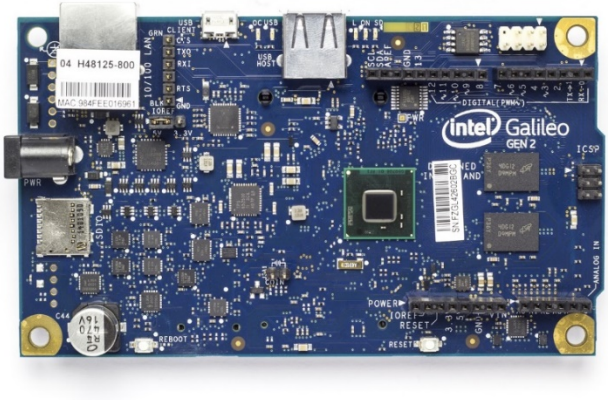


Figure II.1 – Intel Galileo Front Side

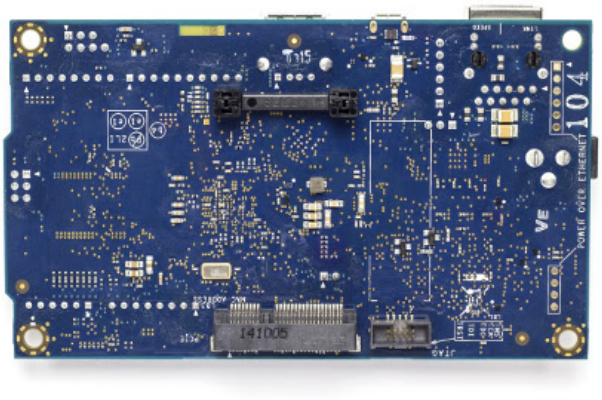


Figure II.2 – Intel Galileo Back Side

The Intel® Galileo Gen2 supports shields that operate at either 3.3v or 5v. The board is designed to be hardware and software pin-compatible with Arduino shields designed for the Uno R3. Digital pins 0 to 13 (and the adjacent AREF and GND pins), Analog inputs 0 to 5, the power header, ICSP header, and the UART port pins (0 and 1), are all in the same locations as on the Arduino Uno R3.

Power:

The Intel® Galileo Gen2 can be powered only via an external power supply. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. The board can operate on an external supply of 7 to 15 volts. The provided power supply provides 12v.

The power pins are as follows:

- VIN: The input voltage to the Intel® board. You can access the voltage supplied via the power jack through this pin.
- 5V: This pin outputs a regulated 5V from the regulator on the board.
- 3.3V: A 3.3 volt supply generated by the on-board regulator. This regulator also provides the power supply to the Quark microcontroller.
- GND: Ground pins.
- IOREF: This pin on the Arduino board provides the voltage reference with which the microcontroller operates. This can be 3.3V or 5V based on the IOREF jumper position.
- 12V power-over-Ethernet (PoE) capable

The input and output pins are as follows:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data.
- Digital I/O: Digital pins 0 through 13 and Analog pins A0 through A5 can be used as a digital input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They can operate at 3.3 or 5 volts.
- Each pin can provide (source) or receive (sink) a current of 16mA @ 5v, or a current of 8mA @ 3.3V
- PWM: Pins 3,5,6,9,10, and 11
    - Provide 8/12bit PWM output with the analogWrite() function. The resolution of the PWM can be changed with the analogWriteResolution() function.
- SPI: SPI header (ICSP header on other Arduino boards)
    - These pins support SPI communication using the SPI library.
- Analog Inputs: pins A0 through A5
    - Each analog input provide 10/12 bits of resolution. The resolution can be changed with the analogReadResolution() function.
- SDA and SCL: Support TWI communication using the Wire library.

2. Photoresistors – 5528 type [2]

Photoresistors, also known as light dependent resistors (LDR), are devices that indicate the presence or absence of light, or to measure the light intensity. In the dark, their resistance is very high. When the LDR sensor is exposed to light, the resistance is very low. LDRs are nonlinear devices.

Technical features of the used photoresistors: [3]

-maximum voltage: 150V;
-environmental temperature: -30~+70°C;
-spectrum peak value: 540 nm;
-light resistance: 10-20kΩ;
-dark resistance: 1MΩ.

3. Servomotors – SG90 type

Servomotors are high torque motors which are commonly used in robotics and several other applications due to the fact that it's easy to control their rotation. Servo motors have a geared output shaft which can be electrically controlled to turn one (1) degree at a time. [4]

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow, orange or white and should be connected to a digital pin on the Arduino board. Note that servos draw considerable power, so if it is needed

to drive more than one or two, it is probably needed to power them from a separate supply (i.e. not the 5V pin on the Arduino). It is necessary to connect the grounds of the Arduino and external power supply together. [5]

A servo motor is controlled by sending a series of pulses through the signal line. The frequency of the control signal should be 50Hz or a pulse should occur every 20ms. The width of pulse determines angular position of the servo and these type of servos can usually rotate 180 degrees (they have a physical limits of travel). [6]

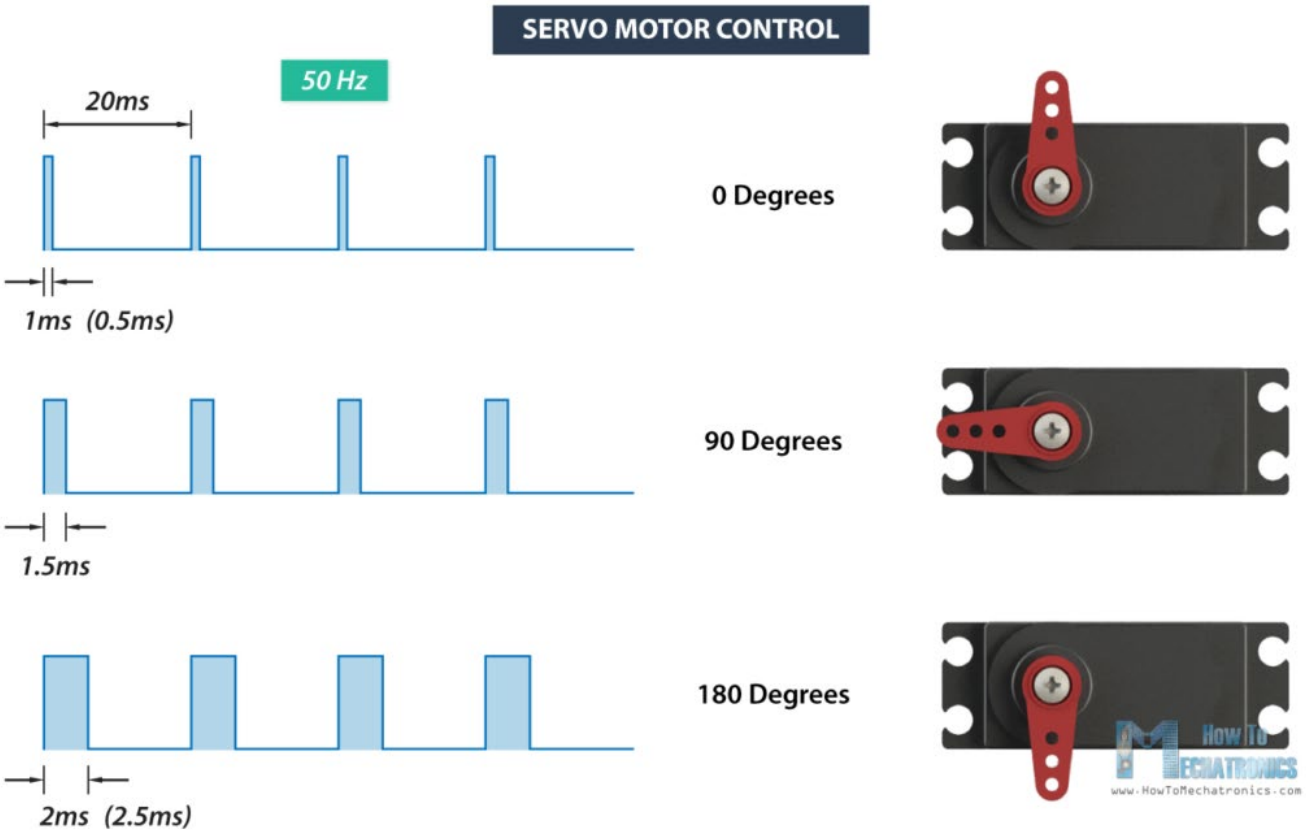


Figure II. 3 Servomotor control [6]

The servomotors used in this project are of SG90 type and they present the following characteristics: [7]

- Connector type: JR
- Modulation: analogic
- Motor type: 3 poles
- Working voltage: 4.8 V
- Dimensions: 23.0 x 12.2 x 29.0 mm
- Speed: 0.1 s / 60°
- Temperature interval: 0°C - 55° C
- Torque: 1.8 kg / cm
- Weight: 9g

4. Solar cell

Solar cell, also called photovoltaic cell, represents any device that directly converts the energy of light into electrical energy through the photovoltaic effect.[8] The photovoltaic effect is a process in which two dissimilar materials in close contact produce an electrical voltage when struck by light or other radiant energy.[9] Unlike batteries or fuel cells, solar cells do not utilize chemical reactions or require fuel to produce electric power, and, unlike electric generators, they do not have any moving parts.[8]

Figure II. 4 - Solar cell[10]

The characteristics of the solar cell used in this project are:[10]

- Output voltage: 3V
- Output current intensity: 100 mA
- Connectors for the cables: Father and Mother
- Dimensions: 55 x 55 x 5 mm

5. LCD

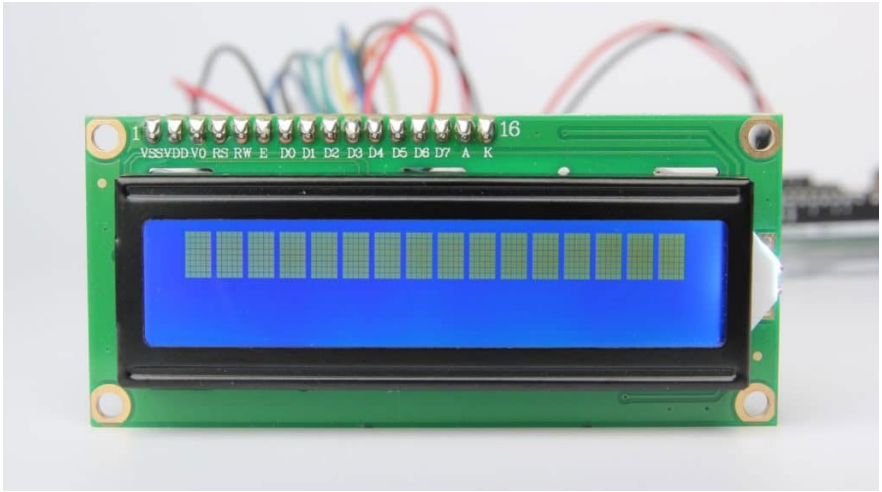


Figure II. 5 – LCD [11]

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A Read/Write (R/W) pin that selects reading mode or writing mode
- An Enable pin that enables writing to the registers
- 8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a display constrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.[12]

III. Software resources

The software resources used for this project are the following:

- Arduino IDE (version 1.8.15)
- Servo.h library
- LiquidCrystal.h library

1. Arduino IDE (version 1.8.15)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.[13]

2. The servo.h library

Allows Arduino/Genuino boards to control a variety of servo motors.
This library can control a great number of servos. It makes careful use of timers: the library can control 12 servos using only 1 timer. On the Arduino Due you can control up to 60 servos.

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite() (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins.[14]

The methods that were used in the project are the following:
- attach()
- write()
- read()

Servo – attach()

Attaches the Servo variable to a pin.

Syntax: servo.attach(pin)

Parameters: servo(a variable of type Servo), pin(the number of the pin that the servo is attached to). [15]

Servo – write()

Writes a value to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft (in degrees), moving the shaft to that orientation.

Syntax: servo.write(angle)

Parameters: servo, angle(the value to write to the servo, from 0 to 180).[16]

Servo – read()

Reads the current angle of the servo.

Syntax: servo.read()

Parameters: servo. [17]

3. The LiquidCrystal.h library

This library allows an Arduino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4- or 8-bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).[18]

The methods that were used in the project are the following:
- LiquidCrystal()
- begin()
- clear()
- setCursor()
- print()

LiquidCrystal - LiquidCrystal()

Creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. If the former, omit the pin numbers for d0 to d3 and leave those lines unconnected. The RW pin can be tied to ground instead of connected to a pin on the Arduino; if so, omit it from this function's parameters.

Syntax: LiquidCrystal(rs, enable, d4, d5, d6, d7)

Parameters:

rs: the number of the Arduino pin that is connected to the RS pin on the LCD

enable: the number of the Arduino pin that is connected to the enable pin on the LCD

d4, d5, d6, d7: the numbers of the Arduino pins that are connected to the corresponding data pins on the LCD.[19]

LiquidCrystal - begin()

Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display. begin() needs to be called before any other LCD library commands.

Syntax: lcd.begin(cols,rows)

Parameters:

lcd: a variable of type LiquidCrystal

cols: the number of columns that the display has

rows: the number of rows that the display has [20]

LiquidCrystal – clear()

Clears the LCD screen and positions the cursor in the upper-left corner.

Syntax: lcd.clear()

Parameters: lcd [21]

LiquidCrystal – setCursor()

Position the LCD cursor; It sets the location at which subsequent text written to the LCD will be displayed.

Syntax: lcd.setCursor(col, row)

Parameters:

lcd: a variable of type LiquidCrystal

col: the column at which to position the cursor (with 0 being the first column)

row: the row at which to position the cursor (with 0 being the first row) [22]

LiquidCrystal – print()

Prints text to the LCD.

Syntax: lcd.print(data)

Parameters: lcd, data(the data to print) [23]

IV. Hardware Implementation

1. Electric Diagram

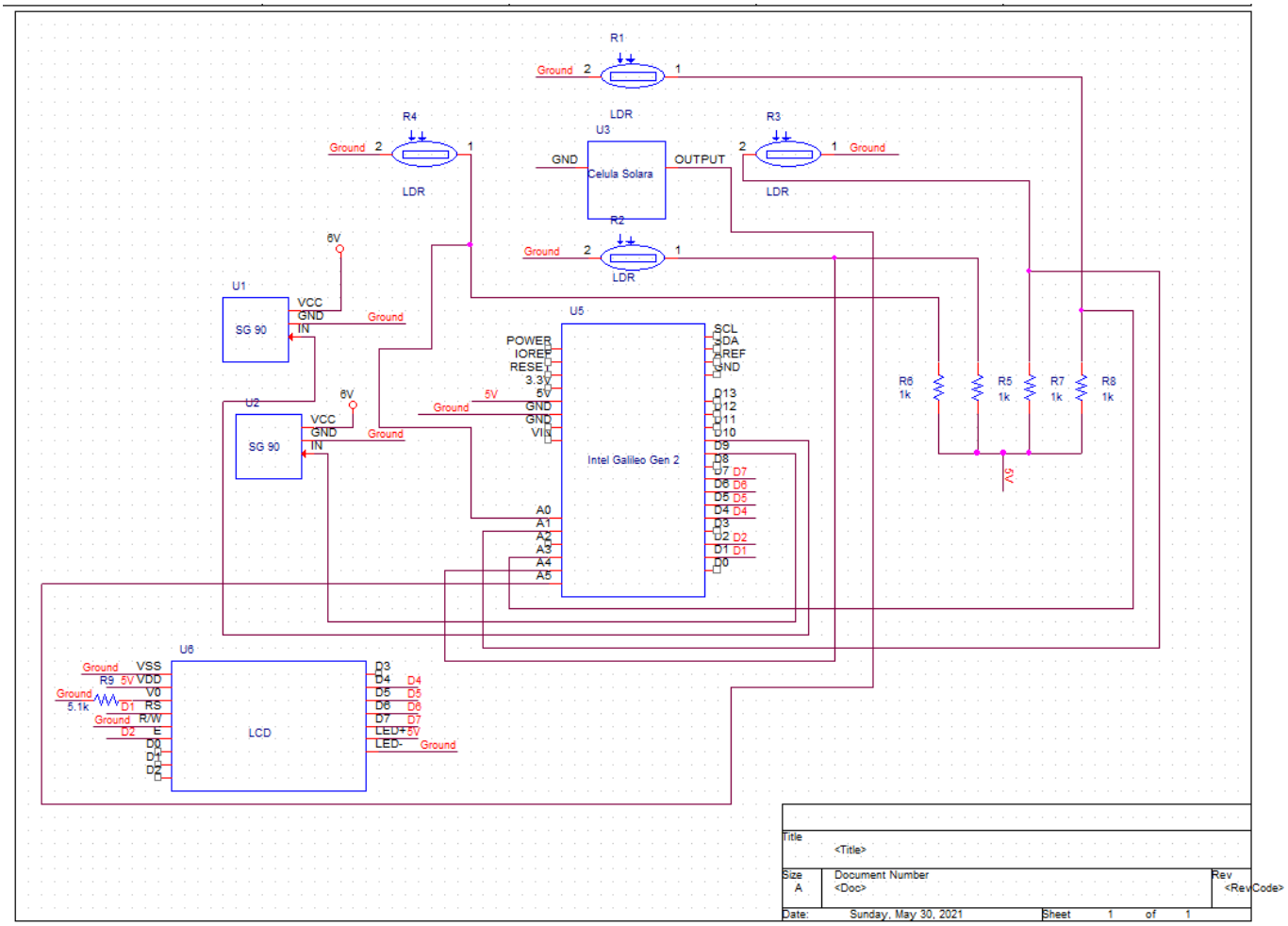


Figure IV.1 – The Electrical diagram

The figure presented above shows the whole electrical diagram for our project. The values for the resistors R5, R6, R7 and R8 was chosen 1 kΩ in order to limit the current the goes in the microcontroller so that we don't damage the board. Even in the most unfavorable cases the current going in will be less than 5mA. The value for the resistor R9 was chosen to be 5.1 kΩ after testing the contrast on the LCD with various resistors between 1 and 10 kΩ as we didn't have a potentiometer to set the contrast and finally we decided that the value of 5.1 kΩ suited us best. The 6V voltage supply used to move the servomotors is represented by 4 1.5V batteries and they power up both servomotors.

2. Working principles

In order to make a solar tracker we need to move the solar cell in the direction of the sun, (in our case for the prototype designed, we move the photovoltaic cell in the direction of an external light source, such as a flashlight). Either way the principle is still the same, by using 4 resistive divider made of one fixed resistor and the LDR, we can determine the voltage drop on each LDR by connecting them to the analogic inputs of the microcontroller. With the voltage drop on each LDR known the machine compares the difference between the voltage on the LDR placed on the left of the solar cell and the one placed on the right, then the process is repeated for the top and

bottom LDRs. Based on these differences the microcontroller moves the servomotors accordingly. The voltage drop on the resistor and LDR is 5V and is supplied by the microcontroller.

The servomotors are connected to the digital pins 9 and 10 on the microcontroller as they are PWM pins. The servos need PWM signals to be moved, this is the reason we chose those pins.

We also have a LCD connected to the board, we decided to use the 4bit mode of the LCD because all the commands we use can be done in the 4bit mode and it saves us wires. On the LCD the the voltage output of the solar cell is shown (in mV) and also the the illuminance on the solar cell (in lux). We can get the voltage output by connecting one wire from the solar cell to one analogic input and the other one to the ground, as for the illuminance we can compute it using the LDRs resistance.

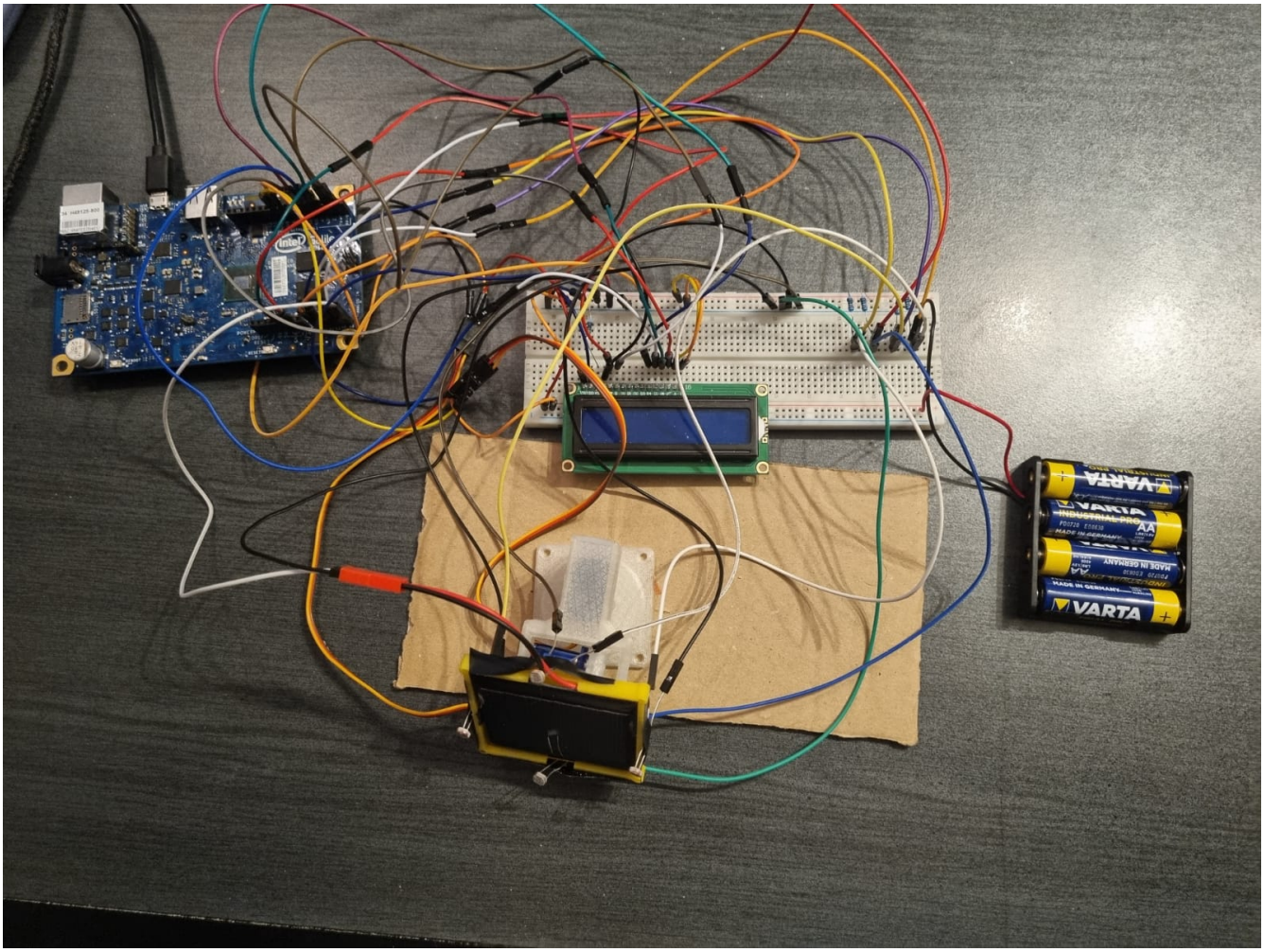


Figure IV.2 – The hardware implementation of the project

V. Software Implementation

1. Logic diagram

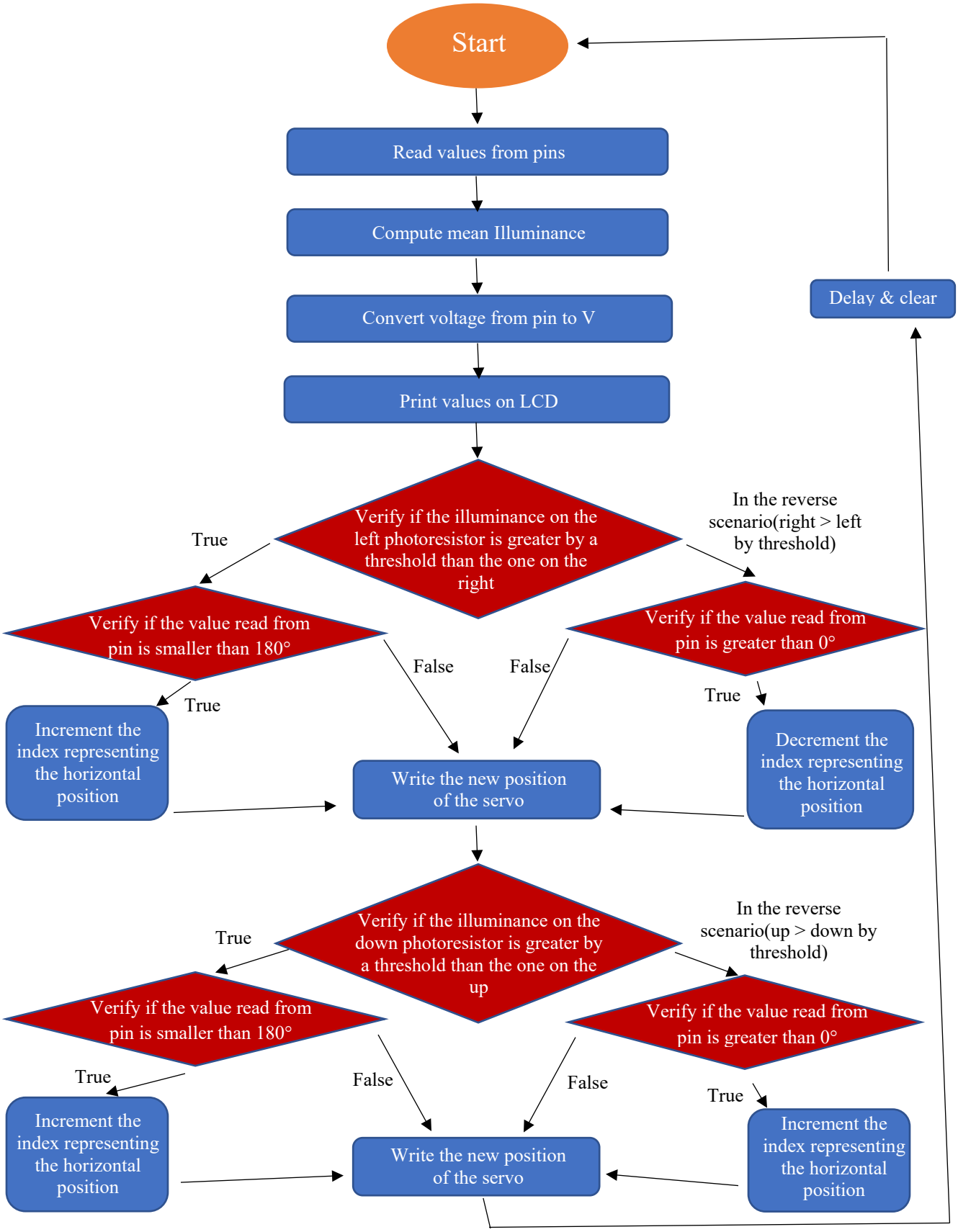


Figure IV.1 – Logic diagram

The logic diagram presented above showcases the workflow of the project. As it can be seen, the first step is to read the values from the pins linked to the photoresistors and the solar cell. This will further offer the possibility to compute the mean of the illuminance directed on the solar cell and also, to convert the values to their actual measuring units, so that they can be properly displayed on the LCD. After the values are displayed on the LCD, the difference of the values on the pins linked to the photoresistors placed on the horizontal axis will be compared to a threshold. If the difference is greater or equal with the threshold, the servomotor that is responsible for the movement on the horizontal axis will move one degree in the direction of the photoresistor linked to the pin with the greater value. This change in position will be made only if the servomotor is not already rotated to the maximum(180° or 0°). Analogously, the same algorithm is implemented to move the servomotor responsible for the change of position on the vertical axis. Afterwards, a small delay will take place, so that the movement in one direction is not too fast, and then the algorithm will repeat.

2. Source code and details

The first part of the code is fairly straight forward. The necessary libraries are included, global variables are declared and initialized, and the instances of the classes Servo and LiquidCrystal are created. The variable servoThreshold has the role of preventing the solar cell from moving uncontrollably for any small change of illuminance. The constants: rs, en, d4, d5, d6, d7, are used as parameters for the LiquidCrystal constructor, the instance of the LiquidCrystal being connected to the pins represented by those constants. The index i and the index j, are linked to the position of one of the servomotors each. They are initialized with 180° as the servomotor responsible for the movement of the solar cell on the vertical axis couldn't reach a high enough rotation, so by setting it at 90° the solar cell would have started at a quite low position. The servomotor on the horizontal axis starts as well from 180°, but this time the value was simply chosen for standardization, so both servomotors can start from the same index value. As for the variables up, down, left and right, they represent the values measured from the pins corresponding to each specific photoresistor. The illuminance variable is meant to hold the mean value of the illuminance directed on the solar cell, while the voltage variable is meant to hold the value of the voltage generated by the solar cell.

```
#include <Servo.h>
#include <LiquidCrystal.h>

const int servoThreshold = 50;

// LCD instantiation
const int rs = 1 , en = 2, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5 , d6, d7);

//Servo instantiation
Servo servo1;
Servo servo2;

int i = 180;
int j = 180;
int up = 0;
int down = 0;
int left = 0;
int right = 0;
int illuminance = 0;
int voltage = 0;
```

The „void setup()" function covers the setup of the objects instatiated previously. Each servomotor is attached to the corresponding pin on the microcontroller and then they are moved to the corresponding starting position

dictated by the indexes i and j. For the LCD, the „begin(16,2)” method is called to set the number of rows and columns that will be used when printing on the LCD, and the method „clear()” will be used to make sure that the display starts from a clean state.

```
void setup() {
  // Setup Servo
   servo1.attach(10);
   servo2.attach(9);
   servo1.write(i);
   servo2.write(j);

   // Setup LCD
   lcd.begin(16,2);
   lcd.clear();
}
```

The most important part of the software implementation stands in the „void loop()” method. The workflow of this method is also presented in the logic diagram and described in the description of the logic diagram. To consolidate that description, it is worth mentioning that the values read from the pins linked to the photoresistors are integer values between 0 and 1023, which are correlated with the voltge measured on the voltage divider between the photoresistor and a 1kΩ resistor. When displaying the mean illuminance on the LCD, the final number must be converted into a value measured with the lux measuring unit. For this reason, after the values from the pins are stored into the variables corresponding to each photoresistor, by using the „analogRead()” method, the mean of those values must be computed and then served as a parameter to a function that does the conversion from that value to the value of the illuminance in the lux measuring unit. The value returned by that function will be assigned to the illuminance variable. The voltage generated by the solar cell is also read as a value from 0 to 1023, so the conversion from that integer number to a value measured in mV was done by multiplying the integer read from the pin with the result of the division of the 5V voltage of the microcontroller to the maximum vlaue that can be read from the pin(1023). The resulting value is rounded using the „round()” function and then assigned to the voltage variable.

```
// Read the values from pins
  left = analogRead(A0);
  right = analogRead(A1);
  up = analogRead(A4);
  down = analogRead(A3);
  illuminance = round((luxConversion((up + down + left + right) / 4)));
  voltage = round(analogRead(A5)*4.8875855); // 1 unit = 4.8875855mV
```

Having the values of the illuminance and the generated voltage in the appropriate measuring units, these values will now be printed on the LCD, using the „print()” method after the cursor was set to the desired position using the „setCursor()” method.

```
// Display voltage and intensity on LCD
  lcd.setCursor(0,0);
  lcd.print("V: " + String(voltage) + " mV");
  lcd.setCursor(0,1);
  lcd.print("I: " + String(illuminance) + " lux");
```

After displaying the values, the servomotors will move the solar cell to the appropriate position. This is done by computing the difference between the variables corresponding to the photoresistors placed on the horizontal axis of the solar cell, and then comparing that value with the threshold. If the difference is greater or equal to the

threshold, the servomotor will rotate one degree by incrementing the index i, and then writing the value of the index to the pin linked to that servomotor. The incrementation is done only if the value at which the servomotor is currently rotated is less than 180°. If the difference between the two pin values corresponding to the photoresistors is smaller than the threshold, then the index i is decremented, and then its value is written to the pin linked to that servomotor. Once again, the decrementation is done only if the value at which the servomotor is currently rotated is greater than 0°. The same algorithm is repeated for the values corresponding to the photoresistors placed on the vertical axis of the solar cell.

```
// Move the servomotors
if ((left - right) > servoThreshold) {
    if (servo1.read() < 180) {
      i++;
    }
    servo1.write(i);
  } else if ((right - left) > servoThreshold) {
    if (servo1.read() > 0) {
      i--;
    }
    servo1.write(i);
  }

  if ((down - up) > servoThreshold) {
    if (servo2.read() < 180) {
      j++;
    }
    servo2.write(j);
  } else if ((up - down) > servoThreshold) {
    if (servo2.read() > 0) {
      j--;
    }
    servo2.write(j);
  }
```

At the end of the loop, a small delay was placed so that the movement in one direction is not too fast, and also the LCD is cleared.

```
delay(250);
lcd.clear();
```

Finally the software implementation presents a conversion function, that is meant to convert the value between 0 and 1023 to a value that corresponds to the lux measuring unit. This is done by converting the pin value to a voltage, and then using that voltage to compute the resistance of the LDR, which will be further used to compute the illuminance value. The formula for the illuminance was chosen after a series of experiments and searches on the internet, this being the closest one to reality(or to the illuminance value determined by the sensor of a smartphone).

```
float luxConversion(int pinValue){
  if ((pinValue != 0)) {
    const int res = 1;

    float Vout = pinValue * 4.8875855;          // Conversion analog to voltage
    float RLDR = res * (Vout/(5000-Vout));      // Conversion voltage to resistance
    float lux = 500/RLDR;                       // Conversion resitance to lumen
    return lux;
```

```
  }

  return pinValue;
}
```

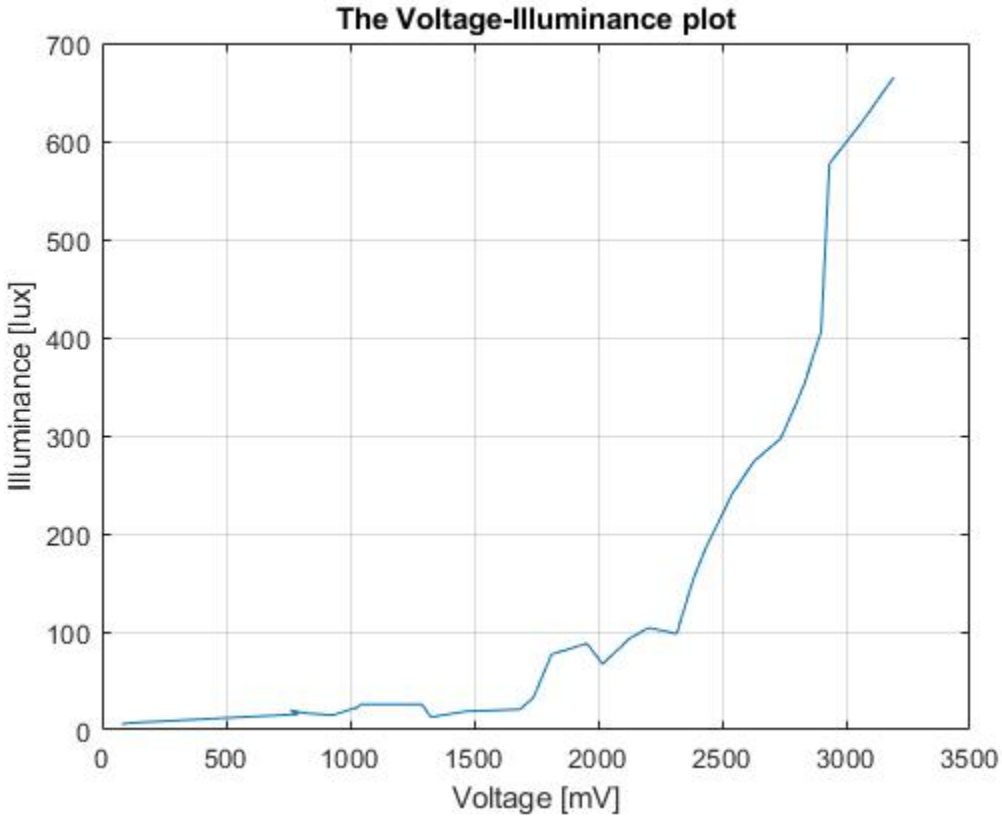
VI. Conclusions

This project was extremely interesting and challenging to implement but in the end we managed to implement the solar tracker to the best of our ability and we are proud of the work we have done. One thing that sparked our interest was the fact that this project has many real life applications.

The first problem we had to face was designing an infrastructure that can sustain the two servomotors. Fortunately we found on the internet three 3D printed parts[24] that incorporated the motors perfectly and we decided to use them because we liked the design a lot.

The second problem we faced was the calculus of the illuminance level since we didn't have enough resources to determine exactly the relationship between the illuminance and resistance of our LDRs. In order to overcome this situation we used the datasheet and created our own formula to get the illuminance from the resistance while also respecting the datasheet.

All in all our project is not perfect and can be improved quite a bit by using stepper motors and placing the LDRs in other ways. One such way of placing the LDRs is by putting all of them on another part specially designed that has 2 walls seppartating the resistors. By using this design the solar tracker is more accurate as whenever the shadow of the wall falls on a LDR the solar tracker will move to eliminate the shadow. Even if we don't have the best way of implementing this project we thoroughly enjoyed working on this project as it helped us develop our skills as future engenieers.



VII. Bibliography:

[1] https://www.arduino.cc/en/ArduinoCertified/IntelGalileoGen2

[2] https://eepower.com/resistor-guide/resistor-types/photo-resistor/#

[3] https://www.kth.se/social/files/54ef17dbf27654753f437c56/GL5537.pdf

[4] https://www.electronics-lab.com/project/using-sg90-servo-motor-arduino/

[5] https://www.arduino.cc/reference/en/libraries/servo/

[6] https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

[7]https://www.conexelectronic.ro/ro/senzori-si-module-pentru-platforme-de-dezvoltare/15482-MINISERVOMOTOR-SG90-9G.html?search_query=servomotor+sg90&results=2

[8] https://www.britannica.com/technology/solar-cell

[9] https://www.britannica.com/science/photovoltaic-effect

[10]https://www.conexelectronic.ro/ro/panouri-solare/20828-PANOU-SOLAR-0-3W-3V-100MA-CONECTOR-JST.html

[11 https://www.makerguides.com/character-lcd-arduino-tutorial/

[12] https://www.arduino.cc/en/Tutorial/LibraryExamples/HelloWorld

[13] https://www.arduino.cc/en/Guide/Environment

[14] https://www.arduino.cc/reference/en/libraries/servo/

[15] https://www.arduino.cc/reference/en/libraries/servo/attach/

[16] https://www.arduino.cc/reference/en/libraries/servo/write/

[17] https://www.arduino.cc/reference/en/libraries/servo/read/

[18] https://www.arduino.cc/en/Reference/LiquidCrystal

[19] https://www.arduino.cc/en/Reference/LiquidCrystalConstructor

[20] https://www.arduino.cc/en/Reference/LiquidCrystalBegin

[21] https://www.arduino.cc/en/Reference/LiquidCrystalClear

[22] https://www.arduino.cc/en/Reference/LiquidCrystalSetCursor

[23] https://www.arduino.cc/en/Reference/LiquidCrystalPrint

[24] https://www.thingiverse.com/thing:708819