INTERNATIONAL BURCH UNIVERSITY
FACULTY OF ENGINEERING AND NATURAL SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

MICROCONTROLLER-BASED SMART SOLAR TRACKING SYSTEM

UNDERGRADUATE PROJECT

VEDAD MUŠOVIĆ

Supervisor
Assoc. Prof. Dr. Dejan Jokić

SARAJEVO
September, 2024

MICROCONTROLLER-BASED SMART SOLAR TRACKING SYSTEM

VEDAD MUŠOVIĆ

Report Submitted in Fulfillment of Requirement for the
Degree of Bachelor of Science in Electrical and Electronics Engineering

INTERNATIONAL BURCH UNIVERSITY
2023/2024

APPROVAL PAGE

Student name and surname: Vedad Mušović

Faculty: Faculty of Engineering, Natural and Medical Sciences

Department: Department of Electrical and Electronics Engineering

Project Title: Microcontroller-Based Smart Solar Tracking System

Date of Defense: September th, 2024

I certify that this final work satisfies all the requirements as an Undergraduate Project for the Bachelor degree in Electrical and Electronics Engineering.

……
Assist. Prof. Dr. Mehrija Hasičić
Head of Department

This is to certify that I have read this final work and that in my opinion it is fully adequate, in scope and quality, as an Undergraduate Project for the Bachelor degree in Electrical and Electronics Engineering.

……
Assoc. Prof. Dr. Dejan Jokić
Supervisor

Examining Committee Members

	Title / Name and Surname	Affiliation	Signature
1.	…	Chairman	
2.	Assoc. Prof. Dr. Dejan Jokić	Mentor	
3.	…	Member	

It is approved that this final work has been written in compliance with the formatting rules laid down by the Department of Electrical and Electronics Engineering.

……

Head of Committee

MICROCONTROLLER-BASED SMART SOLAR TRACKING SYSTEM

ABSTRACT

Traditional solar panels are known not to maximize energy capture. They are stationary, unable to track movements of the sun throughout the day. This results in significant energy losses, since the solar panel is not aligned with direct sunlight. Lack of alignment reduces power output and does not maximize available solar energy. A smart solar tracking system is dynamic and it addresses this issue by automatically adjusting its position, following the path of the light, and ensuring optimal exposure for improved energy generation. Using the Arduino Uno microcontroller, alongside different electronic components, a smart and dynamic solar tracking system is built. Servo motors regulate and adjust the position of the solar panel across two axes. Light dependent resistors behave as sensors for detecting light intensity. Regular resistors control the amount of current flowing through a circuit. The code, required to run and compile the system, is written in the compiler. The goal of this project is to improve energy capture by continuously adjusting the panels to follow the light, ensuring significantly better efficiency than traditional fixed position systems. Future developments and scaling of the created prototype can lead to further improvements, revolutionizing renewable energy systems and future solar energy generation.

Keywords: Solar Tracking System, Solar Energy Optimization, Microcontroller, Arduino Uno, Arduino IDE, Embedded Systems

ACKNOWLEDGEMENTS

First of all, I would like to thank all my family members, especially my parents and my sister. They have provided me with constant and unconditional support throughout my entire education and life. Their love always gave me an additional motivation to work hard and become the best version of myself.

I would like to express a strong gratitude and thank all the professors and assistants who provided me with the best possible education throughout the past three years. Their individual perspectives and teaching ensure that every student becomes a great engineer. Professors Jasna Hivziefendić, Mehrija Hasičić, Lejla Vuić, Jasmin Kevrić and Mirza Šarić presented constant dedication, commitment, as well as encouragement. The support and guidance from each professor has been crucial in shaping both academic and professional journeys of every student.

Special thanks has to go to my supervisor, Professor Dejan Jokić, who guided me through the work on my Undergraduate Project every step of the way. During this process, I obtained great practical and theoretical knowledge with advices, support and patience from my mentor. It was truly a great privilige to have him both as professor through the past three years, and later on, as a mentor as well.

Finally, I want to extend my thanks towards my childhood friends, Tarik and Aldin, as well as all the colleagues I met during the past three years at the University. All of us grew together, and I can now say that many of them have become very good friends of mine. Special thanks goes to Asja Resić, Senija Sejfić, Faris Krličević and Tarik Muhović. I truly look forward to maintaining the bonds I made with these people and seeing where our journeys take us in the future.

DECLARATION

I hereby declare that this Undergraduate Project titled **"Microcontroller-Based Smart Solar Tracking System"** is based on my original work except quotations and citations which have been fully acknowledged. I also declare that this thesis has not been previously or concurrently submittetd for the award of any degree, at International Burch University, any other University or Institution.

…….....

Vedad Mušović

September th, 2024

TABLE OF CONTENTS

LIST OF TABLES

TABLE OF FIGURES

LIST OF EQUATIONS

LIST OF ABBREVATIONS

°C	Degrees Celcius
3D	Three-Dimensional
LDR	Light Dependent Resistor
PWM	Pulse Width Modulation
kΩ	Kiloohm
USB	Universal Serial Bus
mA	Miliamperes
V	Volts
mW	Megawatts
PCB	Printed Circuit Board
SD	Secure Digital
MHz	Megahertz
KB	Kilobyte
SRAM	Static Random-Access Memory
IDE	Integrated Development Environment
ADC	Analog to Digital Converter

CHAPTER 1

INTRODUCTION

Dynamic solar panel installations, using single axis or dual axis tracking systems, offer significant advantages over stationary solar panels. By continuously adjusting their orientation to follow the path of the light, dynamic panels can capture a much greater amount of solar radiation throughout the day, leading to higher energy production. Studies show that single-axis tracking panels can boost power output by up to 66% compared to fixed panels, with improvements in efficiency ranging from 30% to 74% depending on weather conditions. [1] This increased efficiency translates to a more effective use of available solar energy, which can shorten the payback period for the initial investment and make solar energy more economically viable. Stationary solar panels, fixed in a single position, struggle to maximize energy capture due to their inability to adapt to the sun's changing position. This fixed position means that during parts of the day when the sun is at an angle relative to the panel, energy production can be significantly lowered. As a result, stationary panels are significantly less efficient than their dynamic counterparts, particularly when the sun is low in the sky or during cloudy conditions. While fixed panels are generally simpler and less expensive to install and maintain, they miss out on the significant gains in energy production. Therefore, while stationary solar panels are considered as a cost-effective solution, the improved performance and efficiency of dynamic solar panels make them a more attractive option for maximizing energy output and achieving long term savings. The type of solar tracker used depends on the weather in the region, as well as the latitude. In regions closer to Earth's equator, the sun is higher in the sky, requiring less tracking.

Higher latitudes need more precise adjustments for optimal efficiency. Solar trackers come at a higher price compared to the fixed solar panels. However, power generated using solar trackers is significantly higher compared to fixed solar panels. [2] To implement a dynamic solar tracking system, a variety of microcontrollers can be utilized to control the solar panel's orientation based on light intensity.

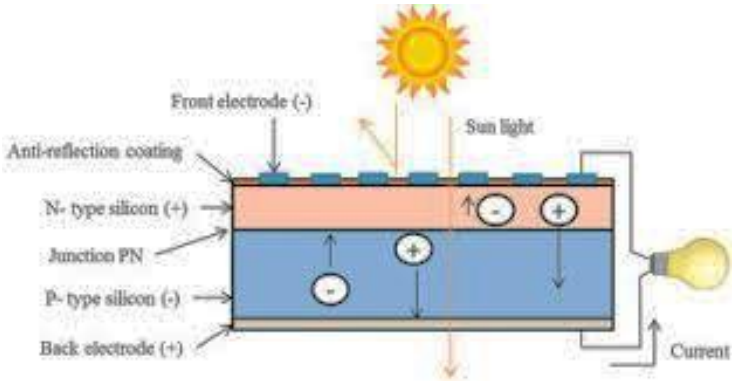

FIGURE 1.1. The Working Principle of a Photovoltaic Cell [3]

As shown in Figure 1.1, when a photon is absorbed at the depletion layer, which is the boundary between N-type and P-type silicon in a solar panel, it generates an electron-hole pair. The electrons are drawn toward the N-type silicon side, creating a potential voltage. This process increases electrical energy, which can then be stored in a battery. Figure 1.1 illustrates how Photovoltaic cells operate and the process by which solar radiation is converted into electrical energy. It is important to note that solar panels operate the best at a temperature of around 25 Degrees Celcius (°C). As expected, the rise in temperature will lead to an increase in current. [3] Readings from can be interfaced to obtain the necessary signals for system optimization. The solar tracker system design involves constructing a tracking frame with actuator and sensors to rotate the solar panel towards the direction of maximum light intensity. The system processes the light intensity data from the sensors and sends control signals to linear actuators to tilt the solar panel accordingly. The sensor will send data in the form of analog signals. Later on, the electronic components are soldered together, after proper components are selected. The tracker's physical construction's dimensions and weight are carefully designed to ensure stability and proper alignment of the solar panel.

CHAPTER 2

RESEARCH

The objective of this project is to design an efficient solar tracking system that provides optimal alignment according to the changing light conditions. To achieve this, the system will use Three-Dimensional (3D) printing technology for the physical structure, allowing for the panel and the motors to be mounted properly. The electrical circuit design will incorporate various components, including servo motors for movements. Light Depedent Resistors (LDRs) are implemented for light detection. A key focus will be put on integrating the electronic components with Arduino Uno microcontroller, which will process the sensor data and control the servo motors through Pulse Width Modulation (PWM) signals. Additionally, a 'voltage divider' circuit is formed. The resistors, with a value of 10 Kiloohms (kΩ), will be implemented to convert LDR readings into usable voltage levels for the microconroller. The circuit can be formed on both the breadboard and the Printed Circuit Board (PCB). Energy capture will be drastically improved by continuously adjusting the orientation in response to the varying light conditions.

2.1. Microcontroller-Based Solar Tracking System

A dual axis solar tracking system adjusts the position of solar panels along two axes: horizontal (azimuth) and vertical (elevation). This movement enables that the panels remain oriented towards the strongest source of light throughout the day and across all four seasons. On the other side of the spectrum, fixed solar panels cannot achieve that.

The term macro tracking refers to the overall ability to maintain the solar panel's orientation relative to the Sun path. This involves understanding the position of the sun based on time of day and geographical location. The term micro tracking involves fine adjustments to the panel's position to account for short term variations in sunlight due to obstacles, weather conditions, or panel alignment errors. This project focuses on micro tracking to fine tune the solar panel's position in real time, implementing sensors and servo motors to adjust the angle of the panel for optimal light exposure. A prototype of the project, the microcontroller-based smart solar tracking system model, is presented in Figure 2.1. The model can be separated into two major segments: the 3D printed physical structure, and the electrical circuit – comprised of the circuit on the breadboard, sensors, and the Arduino Uno microcontroller.

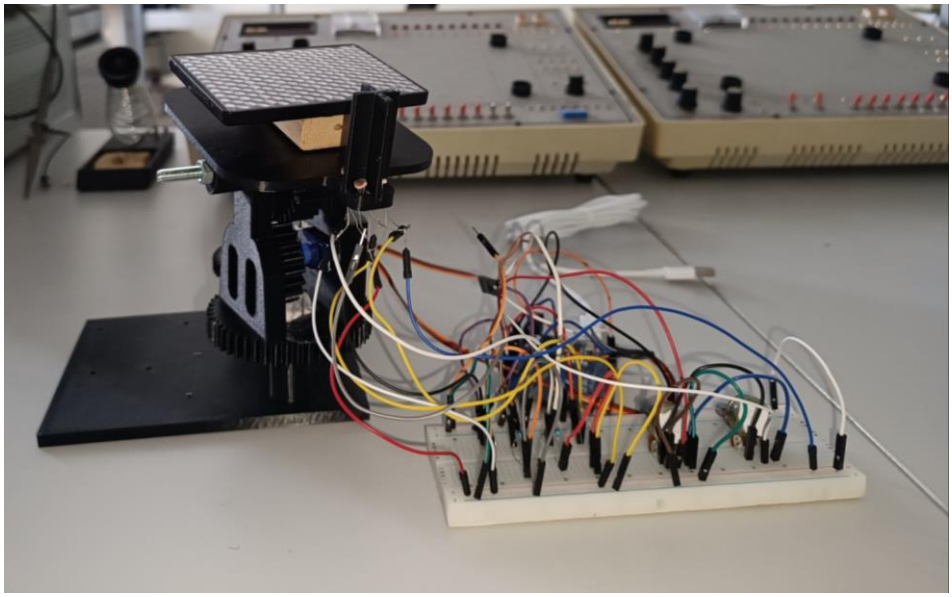

FIGURE 2.1. The Prototype of the Solar Tracking System

2.2. Power and Consumption

The performance of a solar panel can be considerably increased by using a dynamic axis solar tracking system. By adjusting the angle of the panel throughout the day,

sunlight exposure is optimized and more electricity is generated than with a stationary panel, which only receives optimal sunlight at specific times. [4] However, the added components such as motors, sensors, and microcontrollers increase power consumption for operation, although this additional usage is usually offset by the increased energy production. The Arduino itself consumes relatively little power. For example, an Arduino Uno typically uses around 50 Miliamperes (mA) to 60 mA when running at 5 Volts (V). This power draw can increase slightly depending on additional peripherals or sensors, used in the system. Power consumption of Arduino Uno is 250 Miliwatts (mW), when powered at 5V. This consumption is fairly low, but it adds up over time, especially if the system is solar-powered or battery-operated. The power consumption of any microcontroller varies depending on the specific components connected to it and operating conditions of the circuit. Power consumption of the circuit should be measured and tracked to ensure that it operates within the capabilities of the power supply. [5] LDRs, commonly referred to as 'Photoresistors', are used with a voltage divider circuit to provide input to the Arduino's analog pins. Servo motor power consumption depends on their current state. Naturally, if they hold their current position they consume less power, compared to when they move to a certain position. At the beggining stages of developing a dynamic solar tracking system, it is advised to start with servo motors of lower power, in order to reduce power consumption. Additionally, it is crucial to calibrate the sensors together with servo motors. This will eliminate unnecessary movements of the servo motor, and reduce power consumption.

2.3. Cost and Reliability

Dynamic solar tracking systems with a microcontroller are by nature more complex and therefore more susceptible to mechanical and electronic failures, especially in certain extreme weather conditions where moving parts, that is, the servo motors may wear out over time. Software issues can also lead to tracking malfunctions, further impacting system reliability. In contrast, stationary solar panels have no moving parts, making them require less maintenance. However, many applications desire to capture the maximum light energy. Before implementing the dynamic solar tracking system, an analysis of initial investments and the long-term financial benefits needs to be

performed. The added complexity of a dynamic solar tracking system increases the initial investment. Components include the microcontrollers, motors, and sensors. Installation is more complex as well due to the additional wiring and mechanical assembly required. Additionally, long term maintenance costs may rise as moving parts wear out and need to be repaired or replaced. While microcontroller-based solar tracking systems come with higher initial costs and more maintenance requirements, their long-term financial benefits can be substantial. Power generated using solar trackers is significantly higher compared to fixed solar panels. [2] The key advantage of dynamic solar panel system is ability to maximize energy output by continuously adjusting the angle of the panel. This ensures the panels are always optimally aligned to absorb the maximum amount of light, depending on location and weather conditions. Additionally, if the system is grid tied, the extra energy generated can be sold back to the grid. Over time, this can make dynamic solar tracking systems more profitable than traditional stationary alternatives. Government help, tax credits, and other financial incentives often make these systems more affordable at developing stages. The continuous increase in electricity prices presents more reasons for investing in development of solar tracking systems.

2.4. Cost Estimate for the Project Prototype

Conducting a cost estimate for a dynamic smart solar tracking system is crucial to determine its financial viability in different applications. The system includes various electronic components. The initial investment is typically higher than that of the stationary panels, so it is essential to ensure that the potential energy savings justify the expenditure. Additionally, long term maintenance costs, such as replacing mechanical parts which wear out or addressing software malfunctions, should be taken into account. A thorough cost analysis helps with future savings and return on investment. Cost estimate for each component of the dual axis solar tracking system, created in this project, is shown in Table 2.1. Moreover, the table does not include the additional resources needed for the circuit and the system. These include a computer, Universal Series Bus (USB) cable, 3D Printer, soldering equipment and circuit wiring.

COMPONENT	PRICE
Arduino Uno Microcontroller	15 €
Breadboard / Printed Circuit Board (PCB)	6 € / 3 €
Light Dependent Resistor (LDR) x 4	0.55 € x 4 = 2.20 €
10 Kiloohm (kΩ) Resistor x 4	0.05 € x 4 = 0.20 €
10 Kiloohm (kΩ) Potentiometer x 2	2.50 € x 2 = 5 €
Servo Motor (SG90 Micro Servo 9G) x 2	4.50 € x 2 = 9 €
Black 3D Printer Filament	20 €

TABLE 2.1. Cost Estimate for the Project Prototype

When summing up the cost of components from Table 2.1, the total price of the prototype comes up to 57.40 €. This value represents an approximation, since the prices of different components are different from country to country.

2.5. Scalability

Dynamic solar panel systems are scalable. However, adding more panels increases the complexity of tracking systems and energy consumption for controlling movement across multiple axes. Each additional panel requires its own set of sensors and actuators, which complicates the installation in arrays over larger areas. Additionally, different electronic components, such as motor drivers, sensors, better charge controllers and battery storages can potentially be added. Microcontrollers with higher processing powers and faster operating speeds are a serious future option as well. Devoloping a user-friendly application to allow for remote monitoring is also an option in the near future. In summary, scaling and expanding the dynamic solar tracking system requires careful planning to handle the increased complexity of tracking the movement of any potentially new panels to be added to the initial system prototype.

CHAPTER 3

3D PRINTED MODELS

The prototype of the solar tracking system involves a physical construction. 3D printed models of physical components were chosen as parts for the physical structure. They are made to suit all circuit components. An Original Prusa 3D Printer was used to 3D print all the components. The 3D printer is controlled through the PrusaSlicer software.

3.1. 3D Models in PrusaSlicer Software

The first step is to create the necessary 3D models in the software. The created files have a .stl file extension. Inside of PrusaSlicer, the Tracker Base, is shown in Figure 3.1. It has holes for servo motor and for the gear and panel base.

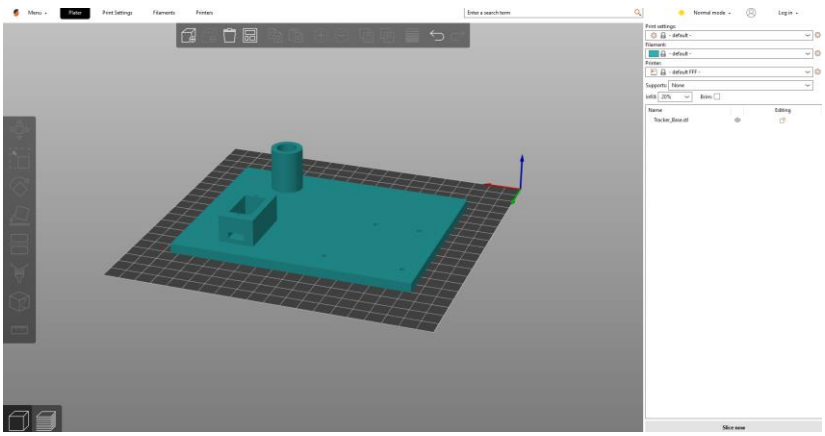

FIGURE 3.1. The 'Base' for the Entire Solar Tracking System in PrusaSlicer

3D models of other components in PrusaSlicer are shown in Figure 3.2. They include: Base for the Gear and Panel Mout (Figure 3.2) is used for panel and vertical gear mount. It has 3D printed holes for shaft and the servo motor. Base Gear Shaft (Figure 3.2) enables the base for panel and vertical gear to connect to the tracker base. This shaft fits into circular hole of tracker base, depicted in Figure 3.1.

Vertical Servo Gear (Figure 3.2) allows the servo motor to fit into a hole on the side of base for panel and gear mount. Vertical servo gear is glued to the servo motor. 3D dimension ensure that this shaft contacts the panel bracket just enough for it to be moved. Horizontal Servo Gear (Figure 3.2) is used for left and right movement. Servo motor fits into a square hole, seen in Figure 3.1. This gear is glued to the servo motor. 3D model dimensions ensure that this gear will contact the panel and gear base enough for it to be moved, when servo motor movement gets initiated.

Panel Bracket (Figure 3.2) provides a bracket where the solar panel is placed. Holes are made on the back side, where the panel gear is located, for the shaft to connect it to the panel base. Additionally, it has four miniature holes on the front. They are made for the LDR's to go through them and effectively capture sunlight. LDR Divider (Figure 3.2) separates each of the four LDRs in order to remove potential outside disturbances. It ensures that each LDR's reading is independent.

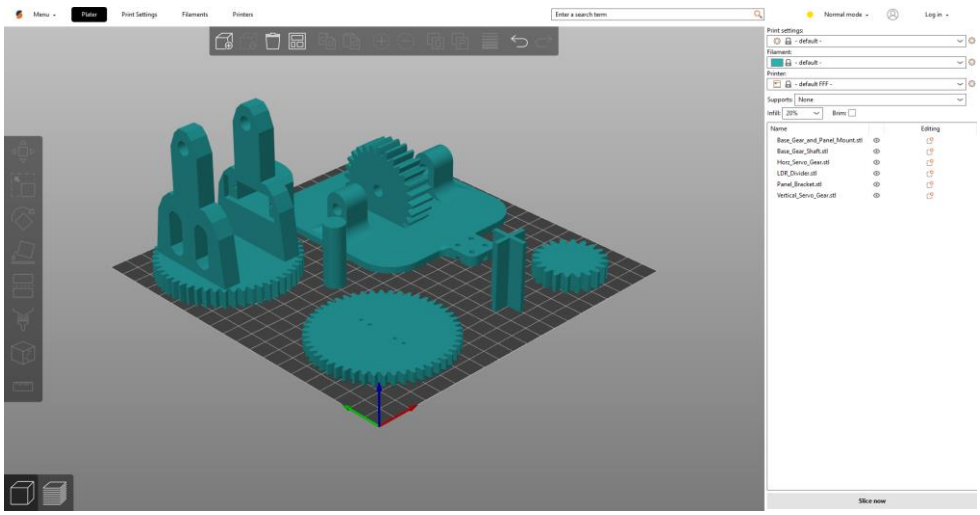

FIGURE 3.2. The 3D Models for the Solar Tracking System in PrusaSlicer

Figure 3.1 and Figure 3.2 show the creation of each 3D component, that is, the .stl file. Inside of the created.stil file, each 3D Model needs to have its infill defined. Infill refers to the internal structure of a 3D printed object. It can vary in density and pattern, affecting the object strength, weight, and material usage. Higher infill density makes the object stronger but uses more material and takes significantly longer to print. Components requiring higher strength, such as mechanical parts or structural supports, benefit from higher infill values. Smaller components have a lower percentage infill. The type of material used, the infill pattern and the density have a remarkable effect on characteristics of a 3D printed structure. [6]

After setting up the .stl file, it is saved as .3mf file. Finally, the file needs to be exported to Prusa G-code Viewer. This is performed for every single 3D printed component's file. When inside of Prusa G-code Viewer, the information about the printer itself is obtained. This is observed in Figure 3.3. This gives the general information about the infill, the filament and the estimated time it will take to 3D print the desired model.

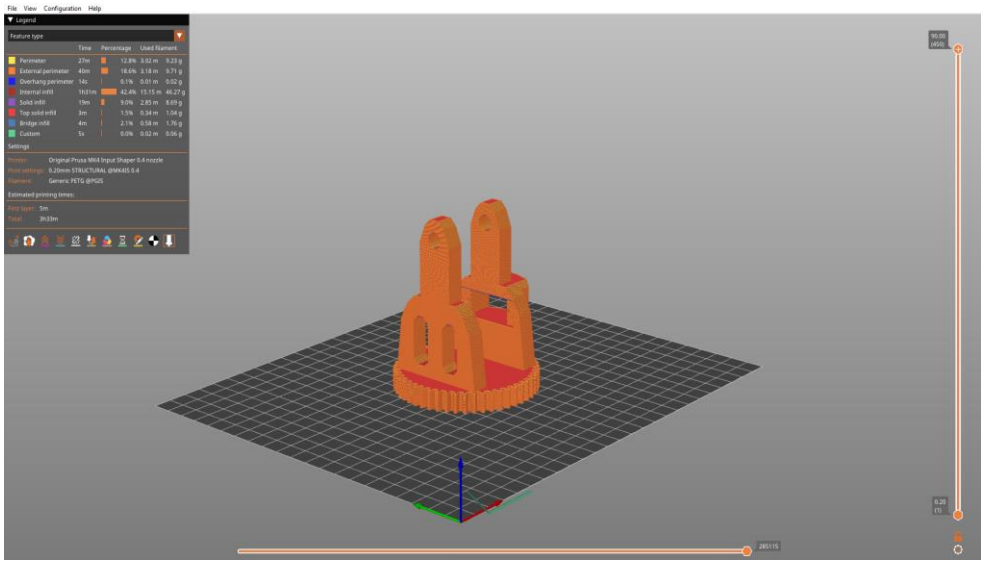

FIGURE 3.3. A 3D Component's Model in Prusa G-code Viewer

The full software process is completed. Finally, the Prusa G-code can be transferred to 3D Printer's SD (Secure Digital) card. This card is then inserted into the Original Prusa 3D Printer and the 3D printing process can officially begin.

3.2. Physical 3D Printing

The color of the filament (used to 3D print the necessary models) is black, because it often hides imperfections and layers lines better than lighter colors, resulting in a smoother appearance. Additionally, black filament provides a consistent and uniform color, making it ideal for prototypes and final products where aesthetics are important. A plastic filament, resistive to heat is used for 3D printing, since the temperatures of and around a running 3D printer reach high values. Measurements conducted during the process of 3D printing show the value of 39 °C (Degrees Celcius). The recorded value is depicted in Figure 3.4, shown below.

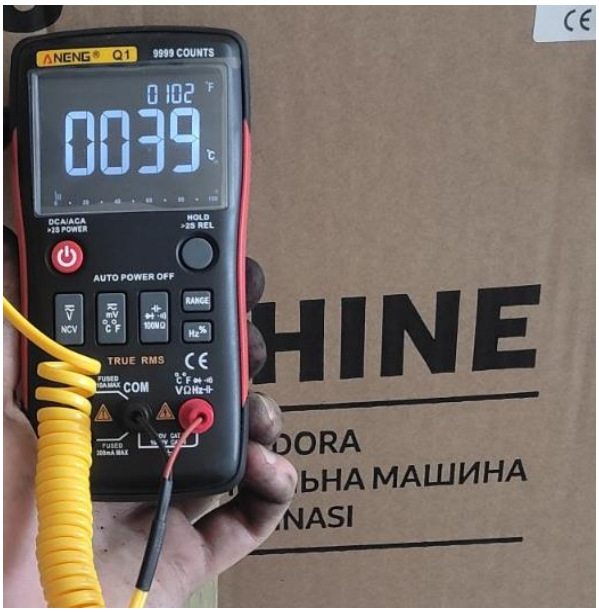

FIGURE 3.4. The Measured Temperature of 3D Printer under Operation

As seen in Figure 3.4, the 3D Printer is isolated during its operation. The reason for this is that it is very sensitive even to the smallest outside interactions, which can disturb its functioning. Because of that, the 3D Printer is enclosed within a closed space. After the printing of the 3D models is completed, they are left on the print area surface to cool down. Since the process is performed under high temperatures, the 3D printed components adhere to the print area and should be removed slowly and carefully to avoid damage. The 3D printed components are shown in Figure 3.5 below.

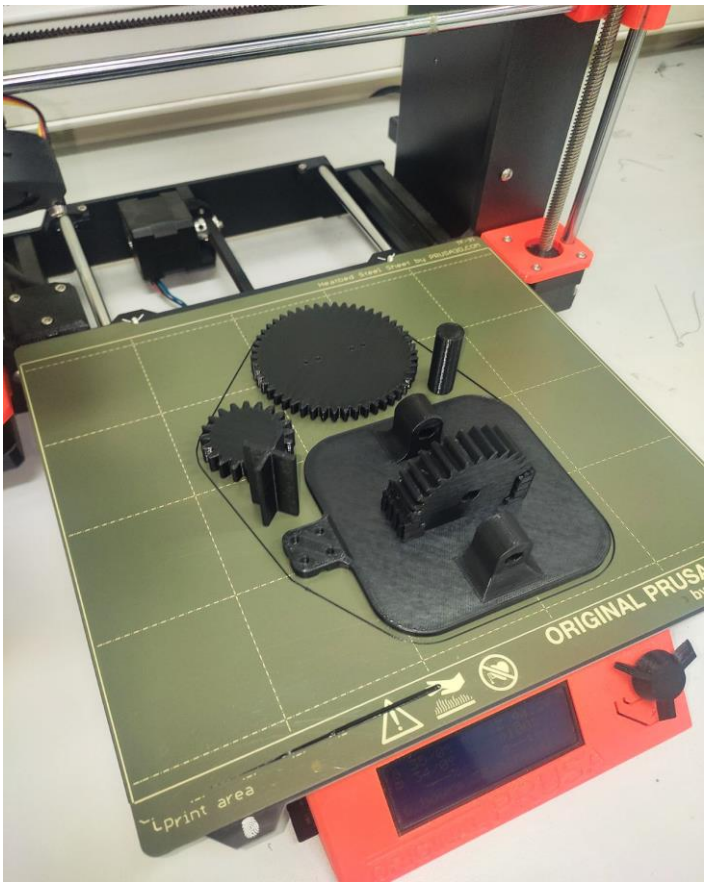

FIGURE 3.5. The 3D Printed Components, positioned on the Print Area

After removing the components from the print area, any support structures or excess material is cleaned up. Dust and particles are removed from the 3D models. Although the 3D Printer is very precise, there will always be small imperfections at certain parts of the 3D models. Potentially rough surfaces are smoothed out using an object like sandpaper. Finally, after completing the additional actions, the 3D printed models are ready to later be used as a physical structure of the dual axis solar tracking system.

CHAPTER 4

CIRCUIT COMPONENTS

This section discusses the components used in the solar tracking system. The components include: Arduino Uno microcontroller, servo motors, LDR's, regular resistors, solar panel and the breadboard. Each component has its distinct purpose.

4.1. Arduino Uno

Arduino Uno (Figure 4.1) has processing capabilities with clock speeds up to 16 Megahertz (MHz), making it suitable for a wide range of applications. Arduino Uno is a microcontroller board consisting of 14 digital input/output pins. Out of them, 6 can be used as PWM outputs and 6 as analog inputs. Additionally, Arduino Uno includes: a USB connection, a power jack, a serial programming header and a reset button. [7]

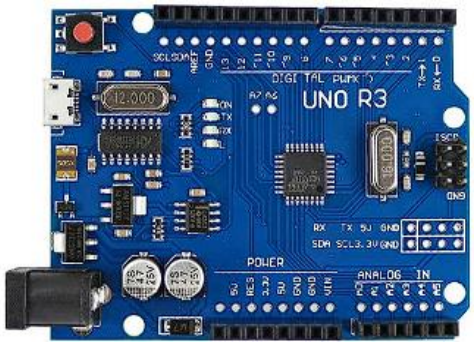

FIGURE 4.1. Arduino Uno [7]

Arduino Uno is made by Arduino.cc, a successful Italian Hardware and Software Company. The Arduino Uno microcontroller, shown in Figure 4.1, is based on the ATmega328P microcontroller. It has a total of 32 Kilobytes (KB) of flash memory, as well as an additional 2 KB of Static Random-Access Memory (SRAM).

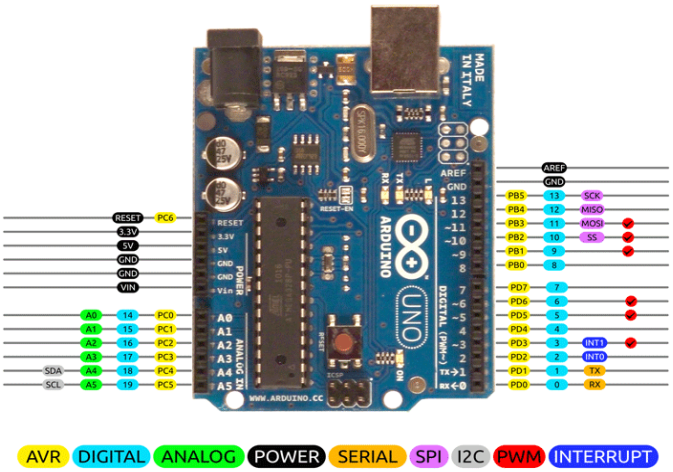

FIGURE 4.2. Arduino Uno Pinout Diagram [8]

Arduino Uno Pinout, depicted in Figure 4.2, shows all pins and their functionalities. These pins can be configured as input or as output. Firstly, there are the power pins. ‘3V3’ pin provides 3.3 V power output. ‘GND’ pins are used for ground connections. ‘5V’ pin provides the 5 V power output for certain applications. Arduino Uno has 5 V as its operating voltage. On Arduino Uno, digital pins range from 0 to 13. They are used to read or send ‘HIGH’ or ‘LOW’ signals (0 or 1). ‘LOW’ signal is signal close to the operating voltage of 5 V, while ‘HIGH’ signal is the signal close to the ground voltage, that is, 0 V. Digital pins detect the states of digital sensors. In this project, they are used to facilitate the movement of servo motors, specifically PWM digital pins 10 and 11. Analog pins, measure a range of voltages and are typically used to read sensor data, in this project, from LDRs. Additionally, sensor data from the two potentiometers is also read by the analog pins. One potentiometer is used for the purpose of adjusting the light threshold for which servo motors movements are triggered. The other potentiometer creates a delay between the individual positional adjustments, analogous to reducing the velocity of the two servo motors.

4.2. Servo Motors - SG90 Micro Servo 9G

Servo motor, a closed loop control system shown in Figure 4.3, consist of a small DC motor, gears for torque multiplication, and a feedback system, usually in the form of a potentiometer or an encoder. SG90 Micro Servo 9G motor has a total of three pins. Digital pin is used so Arduino can send a control signal to control the servo motor's angle. The other two pins are made for a fixed power supply and ground.

FIGURE 4.3. SG90 Micro Servo 9G [9]

The servo motor adjusts its position in order to move to the desired angle. When it comes to the feedback mechanism, the servo motor continuously receives feedback from its internal position sensor. This feedback allows Arduino Uno to monitor the actual position of the servo motor and make necessary adjustments until the solar panel is correctly oriented towards the sun and reaches desired angle. This feedback loop mechanism is present in closed loop systems. This is a very known concept of linear control, where it is possible to fine tune and adjust the output to a desired value. Servo motor feedback loop mechanism is shown in Figure 4.4.

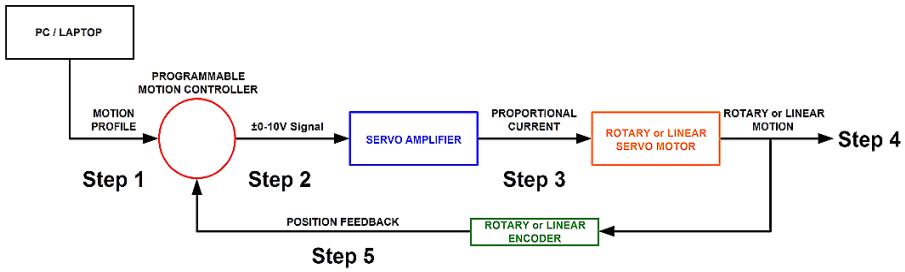

FIGURE 4.4. Closed Loop Mechanism of Servo Motor [10]

As depicted in Figure 4.4, the process begins with creating a program or motion profile in a compiler and then uploading it to the motion controller. This program includes parameters like speed, acceleration, and the desired position. Next, the motion controller, using the parameters from the program, sends a reference signal of up to 10 volts to the servo amplifier. The servo amplifier then uses this reference signal to supply the necessary current, enabling the motor to generate the required force to move to the desired position. After this, the motor moves to the specified position at the programmed speed and acceleration. Finally, the motor's position is fed back to the controller, to ensure the target position is achieved and maintained.

Servo motor can rotate to approximately 180 degrees (90 in each direction). It comes with three wire connections: power (red), ground (brown), and control signal (orange). The control wire sends PWM signal to the servo motor to control its position. The typical duty cycle ranges from 5% to 10% and this shape is shown in Figure 4.5. This means that the control signal should have a pulse width between 5% and 10% of the total cycle time. The standard cycle time for servo motors is 20 milliseconds. [11]

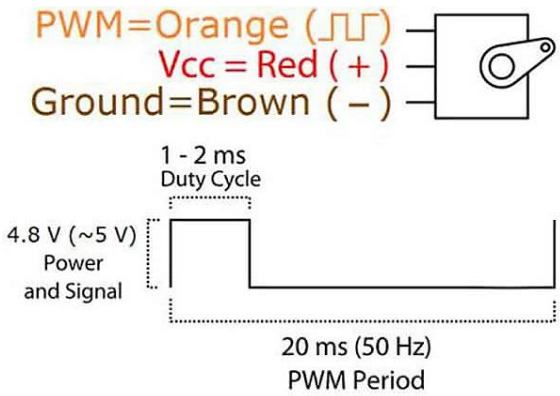

FIGURE 4.5. Servo Motor Duty Cycle [11]

The typical pulse width for the neutral position (at 90 degrees) is around 1.5 miliseconds, which corresponds to a duty cycle of 7.5%. A full clockwise rotation (at 0 degrees) has a pulse width of around 1 miliseconds and corresponds to a duty cycle of 5%. A full counterclockwise rotation (at 180 degrees) has a pulse width of around 2 miliseconds and corresponds to a duty cycle of 10%.

4.3. Light Dependent Resistors – LDRs

Photoresistors, also known as LDRs, are components sensitive to light. Their resistance decreases when the light intensity increases. This is observed in Figure 4.6.

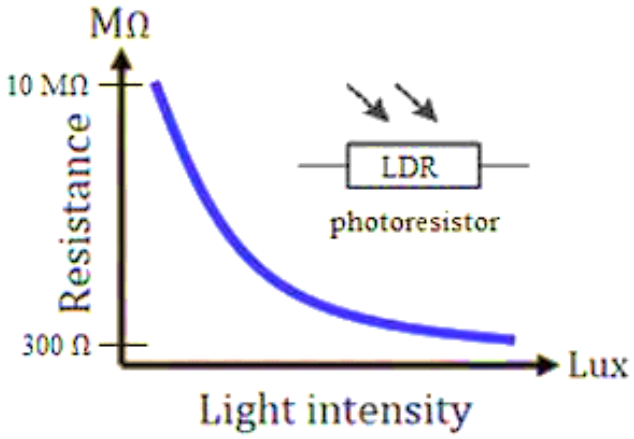

FIGURE 4.6. LDR's Resistance as a Function of Light Intensity [12]

In order to illustrate the functionality of an LDR, the latter can be used for voltage control by light intensity, as indicated by the circuit shown in Figure 4.7. The output voltage varies as a function of the resistance of the LDR what is proven in Figure 4.6.

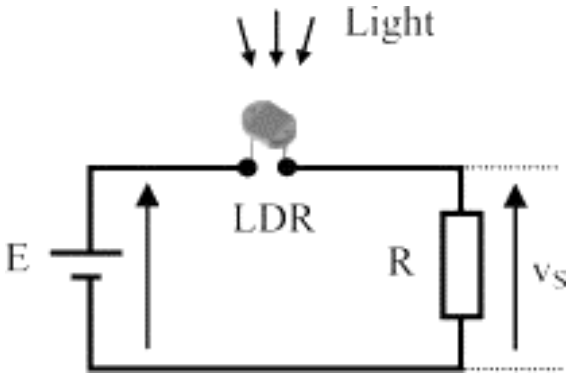

FIGURE 4.7. How the Output Voltage varies with LDR's Resistance [13]

The circuit, shown in Figure 4.7, is commonly referred to as a 'Voltage Divider' circuit. The circuit properties are shown in Equation 4.1. In this equation, R is the resistance

of the regular resistor, with R_{LDR} being the resistance of the LDR. When it comes to the voltages in the equation, E is the input voltage, while V_S is the output voltage.

$$V_s = \frac{R}{R + R_{LDR}} E$$

EQUATION 4.1. Output Voltage Formula in a 'Voltage Divider' Circuit

LDR is composed of photo conducting material. When the light hits this material, the material absorbs the radiation. The electrons will move from the valance band of the semiconductor to the conduction band. The more electrons in the conduction band of the resistor, the lower the resistance of the resistor. [13] The LDRs act as sensors to detect the intensity of the light. By measuring the resistance of the LDRs, the amount of sunlight falling on each sensor can be determined.

4.4. Resistors

The primary function of a resistor is to control the amount of current flowing through a circuit and to create a specific voltage drop. In this project, the resistors of 10 kΩ (kiloohms) are used. This type of resistor is shown in Figure 4.8.

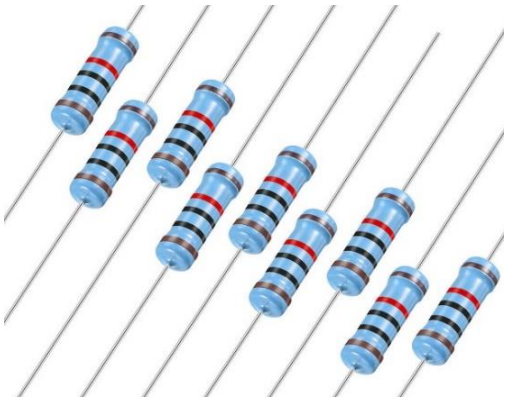

FIGURE 4.8. 10 kΩ Resistors [14]

The resistors create a voltage divider circuit with the LDRs. Because of this, the resistance variations of the LDRs can be converted into voltage variations. The voltage variations can then be read by the Arduino Uno's analog pins. The resistors in solar tracker circuits involving LDRs and Arduino Uno should primarily create the right voltage divider to allow the microcontroller to read the light intensity accurately. Arduino Uno analog pins read voltages between 0 and 5 V. The resistance of an LDR can vary widely depending on light conditions. The resistor paired with each LDR forms a voltage divider, producing a voltage that Arduino can read. The goal is to select a resistor that, together with the LDR, creates a voltage range that covers most of the 0-5 V range for the expected light conditions.

A flashlight test was performed in order to determine the exact resistance to use for the four resistors. LDR is put through different light intensity conditions. At total dark conditions for the light scenarios of interest, the resistance of the LDR is around 150 kΩ. When conditions are slighly less dark, at a moderate light level, the resistance of the LDR is between 20 kΩ and 35 kΩ. The second scenarios of testing included shining a flashlight at the LDR and measuring its resistance. When light is shined as close as possible, the resistance reaches its lowest levels. These levels are around 300 Ω.

When the LDR is exposed to bright light conditions, its resistance drops significantly. For example, with a 330 Ω resistor, the voltage change for a given light intensity change is more pronounced. This results in more precise readings in bright conditions. However, the voltage divider circuit will have very small voltage drop from 5 V for higher resistance in the dark environment. When the flashlight is put further from the LDR, the resistance is between 5 kΩ and 15 kΩ. These levels are most optimal and realistic as well in real-life applications. Because of this, the selected resistance value should fall in this range. A resistance value of 10 kΩ is chosen for each of the four resistors. It provides a wider voltage range for varying light conditions. This helps in maintaining readable values across both bright and dark light scenarios. This resistance value is still low enough to have good resolution in bright conditions.

4.5. Potentiometers

Potentiometers are variable resistors with three terminals. They can be adjusted manually to change their resistance. A potentiometer used in this project is shown in Figure 4.9. Terminal 1 represents the positive (supply) terminal, while Terminal 2 represents ground. The resistance between these two terminals does not change and it represents the total rated resistance of the potentiometer. The unique feature of potentiometers is the 'Wiper', which moves along a resistive track. A knob/slider on top of the potentiometer can be moved in order to modify the resistance value. If adjusting the wiper, the resistance between Terminal 1 and Terminal 2 changes, while the resistance between Terminal 2 and Terminal 3 adjusts inversely.

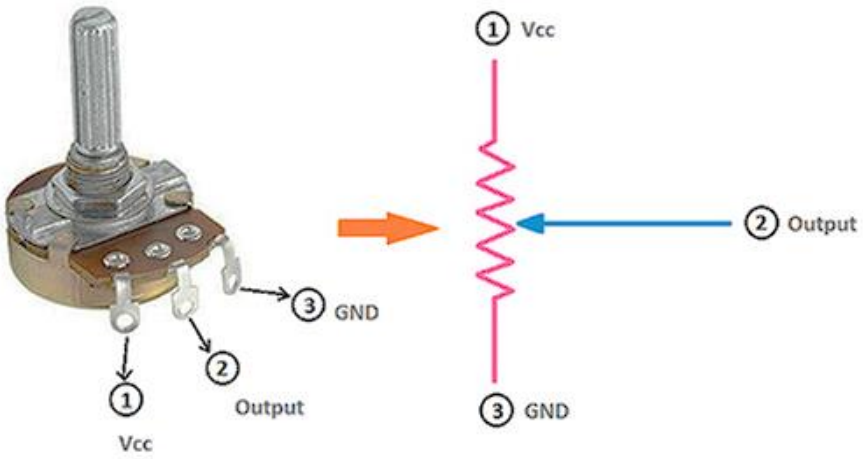

FIGURE 4.9. A 10 kΩ Potentiometer used in this Project [15]

The principle behind wiper's functioning is shown in Figure 4.10. Essentially, as the resistance between Terminal 1 and Terminal 2 increases, the resistance between Terminal 2 and Terminal 3 decreases by the same amount, and vice versa. Using two measuring devices, the resistances across both Terminals 1-2 and Terminals 2-3 can be measured directly. As observed in Figure 4.9, this project uses two 10 kΩ potentiometers, for the two different and specific purposes, in the electrical circuit.

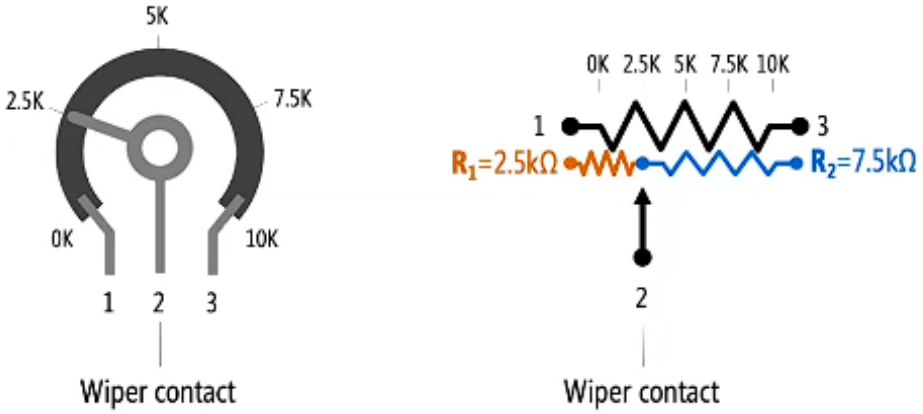

FIGURE 4.10. The Principle behind Potentiometer's Variable Resistance [16]

In this dynamic solar tracking system, two potentiometers are used, for two different purposes. Terminal 1 of each potentiometer is connected to the power supply rail (5V), Terminal 3 is connected to ground, while Terminal 2 (wiper) of each potentiometer is connected to its individual Analog pin on Arduino Uno. One of the potentiometers is used for tolerance adjustment. This potentiometer sets the tolerance level for the difference in light intensity between the sensors. It determines how much variation in light between the LDRs can be tolerated before the servos adjust their positions. In the code, written in Arduino Integrated Development Environment (IDE), a section 'analogRead(5) / 4' reads potentiometer's value connected to analog pin A5, which is divided by 4 to scale it down. This scaled value is used as the tolerance threshold for the servo motor's movement. The condition is written to check if the difference in light intensity exceeds the tolerance. If it does, the vertical servo is adjusted.

The second potentiometer sets the delay between consecutive adjustments of the servos. It controls how quickly the servos move to correct their positions. A higher value results in a longer delay, making the system less responsive but potentially reducing jitter, while a lower value results in more frequent adjustments. The code 'analogRead(4) / 18' reads from the potentiometer on analog pin A4 is divided by 18 to scale it down. The value it is divided by depends on the potentiometers value. In this case, 18 is suitable for 10 kΩ potentiometer. This value is used to create a delay in the loop() function. The created delay affects how often the servos are updated based on the new readings of light intensity values.

4.6. Mini Solar Panel

A solar panel has a primary purpose to convert sunlight into electrical power, which can be used or stored for various applications. The solar panel is mounted on a support structure which allows it to move across two axes. Figure 4.11 shows the actual solar panel used in this project. The goal is to position the solar panel optimally.

FIGURE 4.11. The Mini Solar Panel used in the Project

The servo motors, controlled by Arduino Uno, is responsible for adjusting the position of the solar cell based on the readings from the LDRs. This adjustment ensures that the panel is constantly oriented towards the sun, optimizing energy capture. Solar tracking algorithm function includes that the Arduino reads the light intensity from the LDRs. After this, it calculates the difference in illumination on either side, and uses this information to adjust the position of the solar panel with servo motors. This dynamic tracking enhances the efficiency of the solar cell.

4.7. Breadboard

A breadboard is a tool used for building and testing electrical circuits. It facilitates the organization of the circuit, allows for easy modifications during prototyping, and serves as a platform for building the dynamic solar tracking system. Breadboards have

a grid of holes where electronic components and wires can be easily inserted and connected. The rows and columns inside the breadboard are internally connected in a specific pattern, making it easy to link the components together. A typical breadboard, which is used in this project as well, is presented in Figure 4.12. Power rails run along the top and bottom sides for connecting power and ground, reducing the time it takes to create the electrical circuit. In Figure 4.12, the power rail is noted by '+' sign, and it is of red color. Ground rail is noted by '-' and the color blue.

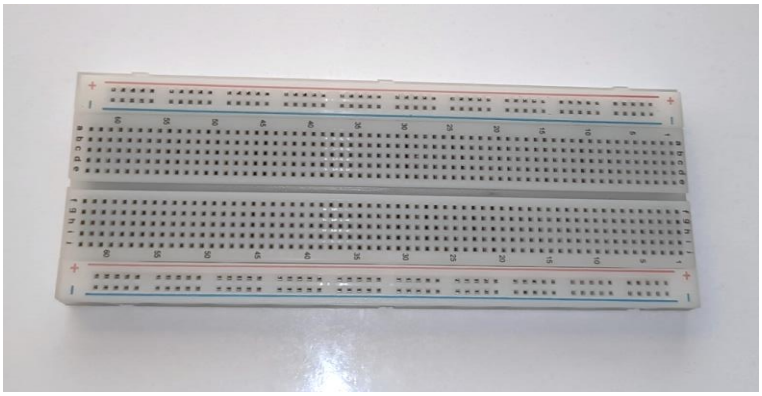

FIGURE 4.12. Breadboard used in the Project

Power and ground rails mean that the holes along their lines are connected internally. From Figure 4.12, it is seen that these lines go horizontally. Finally, in the center of the circuit there is a small indent, which separates the two sections (bottom and down). In these sections, the holes are connect internally in a vertical vay. This presents one more major advantage of using a breadboard to make circuit prototypes. Breadboards are made of plastic and are reusable, which makes them excellent for prototyping and experimenting with different setups, as well as making quick changes to the circuit.

4.8. Alternative - Printed Circuit Board - PCB

An alternative to using a breadboard is the Printed Circuit Board – PCB. Although a bit more complex, it is an excellent solution to scale the prototype in the near future. A single side copper PCB can be used to make the circuit more reliable. It makes the

entire design more compact. This type of PCB has copper traces on only one side of the board. Electronic components are mounted on the side opposite the copper traces. Single side copper PCB have many advantages. They are commonly used in electric circuits due to easy manufacturing and cost efficiency. There are two common PCB technologies. In through hole technology, component leads are inserted in holes surrounded by conductive pads. In surface mount technology, the component is placed on the PCB so that the pins line up with the conductive pads on the surfaces of the PCB. Solder paste, previously applied to the pads, holds the components in place temporarily. In both through hole and surface mount, the components are then soldered. Once cooled and solidified, the solder holds components in place permanently and electrically connects them.

Soldering process involves using the soldering iron and making the necessary connections on the copper side of the PCB. Figure 4.13 shows the copper side of the PCB, where soldering with a soldering iron is performed. With solder, sides of each component can firstly be connected to the PCB. As observed in Figure 4.13, the solder creates electrical connections between the components it contacts. This means that the connections can be facilitated in two ways: when wires, and when the solder contacts the two locations. Twe two rails created by solder effectively behave the same way as ground and power rails on a breadboard, except, with the PCB, there is practically no concern about circuit components accidentally disconnecting from the circuit

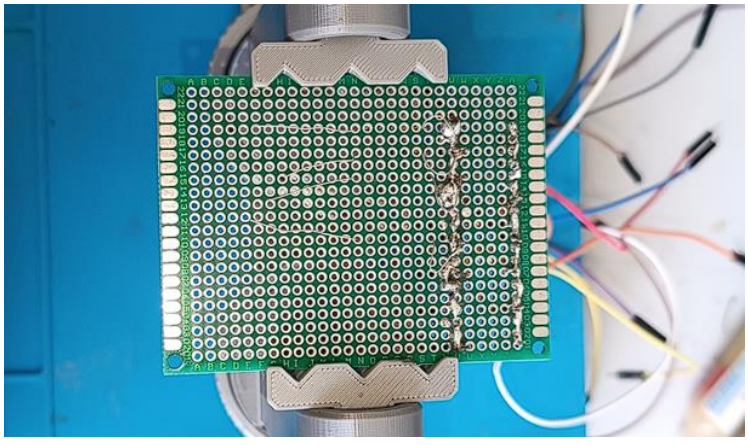

FIGURE 4.13. Single Side Copper PCB used in the Project

CHAPTER 5

ELECTRICAL CIRCUIT CONNECTION

With the components of the physical structure 3D printed, and the proper electrical circuit components selected, the electric circuit can be connected. Creating the electrical circuit of dual axis solar tracking system involves soldering with soldering iron. The base of the circuit is 'housed' on a breadboard, with the rest of the components from Chapter 4 completely or partially connected to it. Figure 5.1 presents a schematic of the electrical circuit for the dynamic solar tracking system, created in Tinkercad. The unique ability of Tinkercad is to create 3D digital designs of circuits online, in order to actually visualize a project, before testing it in practice. Tinkercad offers a versatile environment for both designing and simulating the electronic circuits used in this project, allowing to build and visualize the entire setup.

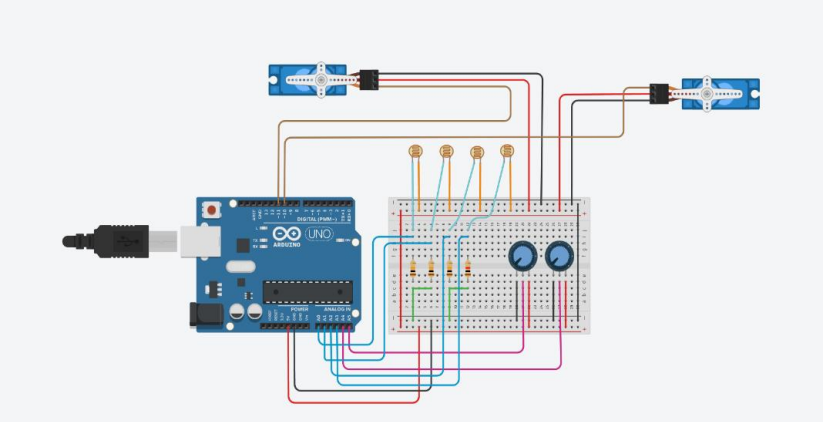

FIGURE 5.1. A Schematic of the Project's Electrical Circuit

5.1. Setting up the Breadboard

Firstly, on the breadboard, the power rails for 5 V are identified. As already discussed, this is the voltage value at which the Arduino microcontrollers operate. This power supply will power the circuit, as well as the two servo motors. When it is required to connect a component to this rail, a terminal or a wire is simply put into the rail. The same concept applies to the second rail. The only difference is the fact this rail is made for ground. It will be used for grounding the two servo motors, as well as the overall electric circuit. Before testing circuit's operation, it is necessary to inspect these connections, in order to ensure safety. On the Arduino Uno, '5V' pin is connected to the power rail of the Arduino Uno with a jumper wire. Similarly, 'GND' pin is connected to the ground rail of the Arduino Uno the exact same way.

5.2. Connecting Two Servo Motors

As discussed in Chapter 4.2, SG90 Micro Servo 9G motor has three pins. The first pin is connected to the 5 V power supply rail on the breadboard. The second pin is connected to the ground rail on the breadboard. The third pin is for the control signal. The control wire sends a PWM signal to the servo to control its position. This will adjust the angle of the servo motor and is performed with extremely fast times. The control pin of the servo motor for horizontal movement is connected to the the Digital Pin 10 of Arduino Uno. The control pin of the servo motor for vertical movement is connected to Digital Pin 11. Both of the pins are PWM pins. This means that the pins are perfectly suitable for the control signals, which have very short duty cycles.

5.3. Connecting the LDRs and Regular Resistors

The next components of the circuit are the LDRs. They are put through four miniature holes on the front of 3D printed Panel Bracket. Additionally, they are separated from each other by the LDR divider, in order to effectively capture sunlight. They behave as light detection sensors, and later on form a voltage divider circuit with the resistors, converting changes in light intensity into varying voltage levels. This data is then used

to adjust position of the solar panels, ensuring they are always oriented towards brightest light source for optimal energy geneation. In the electrical circuit, after putting each of the four LDRs on the panel mount, they can be connected to necessary components. One leg of the each LDR is connected to the power rail of the breadboard. The other leg of each LDR is soldered to a wire and connected with one end of the resistor. Additionaly, an analog pin of Arduino Uno is connected to this node with a jumper wire. The following analog pins are used for the individual LDRs in the circuit: Bottom Right LDR – A3, Bottom Left LDR – A1, Top Right LDR – A2, Top Left LDR – A0. While one leg of the resistor goes into a node with one side of LDR and an analog pin, the other one is connected to the ground rail on the breadboard. The chosen resistors are 10 kΩ each. In section 4.4, it was discussed how the tests were conducted in order to determine which resistor value is the most optimal to use for the circuit. In the voltage divider circuit, if small resistor values (below 1 kΩ) are used, the voltage drop is noticeable for extremely bright light conditions (low LDR values).

As the conditions become darker, the change in voltage would become less and less apparent. This is why, after conducting the flashlight test in Section 4.4, 10 kΩ resistor values are chosen as ones that suit the operating conditions of this solar tracker the best. It helps in maintaining readable sensor values across different lighting scenarios, including both bright and dark conditions. This resistance value is still low enough to have good resolution and precision in bright light conditions. Resistance will ensure a power efficient operation since the circuit will draw less current compared to if lower resistance values were used. A 10 kΩ resistor provides a wider range of voltages for varying light conditions. This helps in maintaining readable values from the sensors across different lighting scenarios, including both bright and dark conditions.

5.4. Connecting the Potentiometers

One terminal of both potentiometers in the circuit is connected to the power rail on the breadboard, with the other one being connected to the ground rail on the breadboard. Finally, the wipers of the individual potentiometers are connected to Analog pins of Arduino Uno. The wiper of a potentiometer is connected to an analog pin because it

provides a variable voltage based on the position of the wiper along the resistive track. When turning the potentiometer, the wiper moves between the two end terminals, creating a range of voltages between 0V and the supply voltage. The analog pin reads this changing voltage as a continuous value, which can then be used for the applications of controlling servo motor speed, or setting sensor thresholds. Analog pins can interpret the range of values, unlike digital pins which can only detect high or low states. Both of the potentiometers are of 10 kΩ value. The potentiometer for controlling the light threshold is connected to Analog pin A5, while the potentiometer for controlling the delay time of the servo motors is connected to Analog pin A4.

5.5. The Final Circuit

The final circuit, shown in Figure 5.2 and Figure 5.3, is formed after complementing all of the previously mentioned steps and ensuring their proper execution. The two figures represent the different view points of the electrical circuit, with its wiring, electronic components, Arduino Uno microcontroller, as well the 3D printed model.

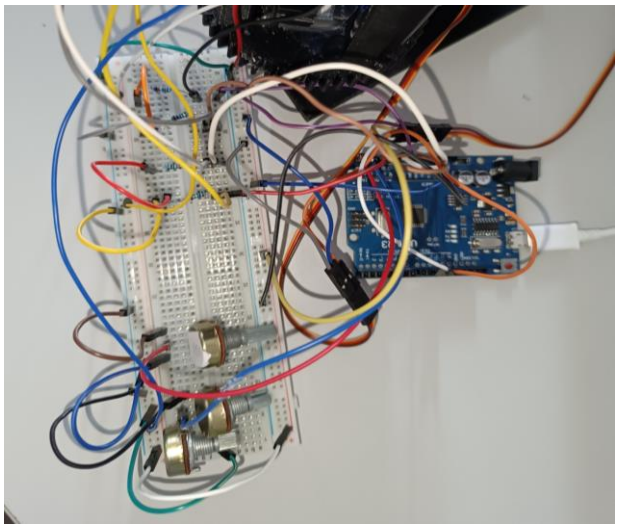

FIGURE 5.2. The Circuit with Electronic Components and Arduino Uno

Additionally, Table 5.1 provides an overview of all pins used on the Arduino Uno microcontroller and the specific component which is attached to that specific pin.

Component	Arduino Uno Pin
Top Left LDR	Analog Pin A0
Bottom Left LDR	Analog Pin A1
Top Right LDR	Analog Pin A2
Bottom Right LDR	Analog Pin A3
Potentiometer – Delay	Analog Pin A4
Potentiometer - Threshold	Analog Pin A5
Servo Motor - Horizontal	Digital Pin 10
Servo Motor - Vertical	Digital Pin 11
Power / Ground Rail	GND / 5V

TABLE 5.1. Pin Connections, made to the Arduino Uno Microcontroller

Figure 5.3 represents how the LDRs, mounted on the 3D printed model, are connected to the rest of the electrical circuit. After creating both the electrical circuit and the 3D printed model, the code, required to run the system, can be written in Arduino IDE.

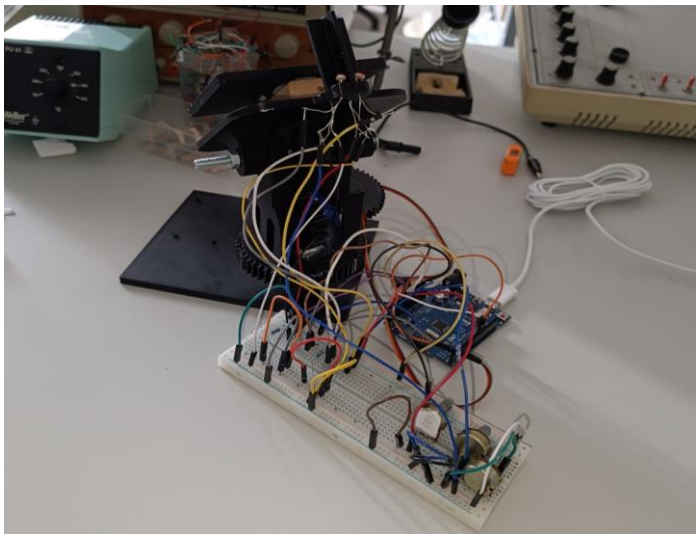

FIGURE 5.3. The Circuit with Electronic Components and Arduino Uno

CHAPTER 6

THE CODE

Writing the code, required for the system based on Arduino Uno microcontroller to operate, required many considerations to be taken into account. The code can naturally be very complex, however, the goal was to simplify it when possible. This chapter presents the code from Arduino IDE, as well as explains every single section of it. Figure 6.1 and Figure 6.2 include the code, written in Arduino IDE for this project.

DiplomskiKod.ino

```
#include <Servo.h>

// instances of the servo class from Servo library
Servo horizServo; // servo for horizontal movement
int horizAngle = 90; // setting the initial position
Servo vertServo; // servo for vertical movement
int vertAngle = 90; // setting the initial position

int ldrTopLeft = 0;  // top left LDR - analog pin
int ldrTopRight = 1; // top right LDR - analog pin
int ldrBottomLeft = 2; // bottom left LDR - analog pin
int ldrBottomRight = 3; // bottom right LDR - analog pin

void setup()
{
  Serial.begin(9600); // baud rate
  horizServo.attach(10); // digital pin for horizontal servo motor - using Servo library
  vertServo.attach(11); // digital pin for vertical servo motor - using Servo library
}

void loop()
{
  int lightTopLeft = analogRead(ldrTopLeft);     // sensor value
  int lightTopRight = analogRead(ldrTopRight);   // sensor value
  int lightBottomLeft = analogRead(ldrBottomLeft);  // sensor value
  int lightBottomRight = analogRead(ldrBottomRight); // sensor value
  int delayTime = analogRead(4) / 18;   // potentiometer for delay time - servo motor control
  int tolerance = analogRead(5) / 4;    // potentiometer for light intensity tolerance

  // determining the difference between light in opposing directions
  int avgTop = (lightTopLeft + lightTopRight) / 2;   // average value of the top two sensors
  int avgBottom = (lightBottomLeft + lightBottomRight) / 2;  // average value of the bottom two sensors
  int avgLeft = (lightTopLeft + lightBottomLeft) / 2;  // average value of the two left sensors
  int avgRight = (lightTopRight + lightBottomRight) / 2; // average value of the two right sensors
  int diffVertical = avgTop - avgBottom;   // measuring vertical differences (betweeen top and bottom)
  int diffHorizontal = avgLeft - avgRight; // measuring horizontal differences (between left and right)

  if (-1 * tolerance > diffVertical || diffVertical > tolerance) // if the vertical light difference exceeds tolerance
  {
```

FIGURE 6.1. The Code in Arduino IDE - Part 1

DiplomskiKod.ino

```
int diffVertical = avgTop - avgBottom;   // measuring vertical differences (between top and bottom)
int diffHorizontal = avgLeft - avgRight; // measuring horizontal differences (between left and right)

if (-1 * tolerance > diffVertical || diffVertical > tolerance) // if the vertical light difference exceeds tolerance
{
  if (avgTop > avgBottom) // if top sensors detect more light than bottom sensors
  {
    vertAngle = ++vertAngle; // increment angle for the vertical servo motor
    if (vertAngle > 180)
    {
      vertAngle = 180; // boundary angle
    }
  }
  else if (avgTop < avgBottom) // if bottom sensors detect more light than top sensors
  {
    vertAngle = --vertAngle; // decrement angle for the vertical servo motor
    if (vertAngle < 0)
    {
      vertAngle = 0; // boundary angle
    }
  }
  vertServo.write(vertAngle); // update vertical servo angle - using Servo library
}

if (-1 * tolerance > diffHorizontal || diffHorizontal > tolerance) // if the horizontal light difference exceeds tolerance
{
  if (avgLeft > avgRight)  // if sensors on the left detect more light than sensors on the right
  {
    horizAngle = --horizAngle; // decrement horizontal angle
    if (horizAngle < 0)
    {
      horizAngle = 0; // boundary angle
    }
  }
  else if (avgLeft < avgRight) // if sensors on the right detect more light than sensors on the left
  {
    horizAngle = ++horizAngle; // increment horizontal angle
    if (horizAngle > 180)
    {
      horizAngle = 180; // boundary angle
    }
  }
  horizServo.write(horizAngle); // update horizontal servo angle - using Servo library
}

delay(delayTime); // delay for smooth operation, enabled with the use of the potentiometer
}

```

FIGURE 6.2. The Code in Arduino IDE - Part 2

6.1. Code Analysis

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

The code begins with incorporating the 'Servo' library. This library is included to control the servo motors in order to move the solar panel. It is compatible with majority of Arduino development boards. Servos come with built-in gears and a shaft that can be precisely controlled for movement. Standard servos can position the shaft at specific angles, typically between 0 and 180 degrees, allowing for a controlled motion. Servo library enables the use of up to 12 motors on the majority of Arduino boards. It enables

servo motors object creation, and enables methods, such as: attach(), write(), writeMicroseconds(), read(), attached(), as well as detach().

```
Servo horizServo; // servo for horizontal movement
int horizAngle = 90; // setting the initial position
Servo vertServo; // servo for vertical movement
int vertAngle = 90; // setting the initial position
```

The code to define the two Servo objects is written, one for horizontal servo motor movement (horizServo), and one for vertical servo motor movement (vertServo). The initial positions are set, for both servo motors, to 90 degrees from the default reference point of the servo motors, which is usually the middle of their range. This means that if the servo motor was previously positioned at a different angle, it will move to 90 the position of degrees. The value of 90 degrees typically represents the center position for standard servos, where their range spans from 0 to 180 degrees.

```
int ldrTopLeft = 0;  // top left LDR - analog pin
int ldrTopRight = 1; // top right LDR - analog pin
int ldrBottomLeft = 2; // bottom left LDR - analog pin
int ldrBottomRight = 3; // bottom right LDR - analog pin
```

The integers are defined as analog pins (0, 1, 2 and 3) for the four LDRs that sense and detect light intensity in different directions: Top Left, Top Right, Bottom Left, and Bottom Right. The LDRs create a voltage divider circuit with the regular resistors. Because of this, the resistance variations of the LDRs can be converted into voltage variations. The voltage variations can then be read by the Arduino Uno's analog pins.

```
void setup()
{
  Serial.begin(9600); // baud rate
   horizServo.attach(10); // digital pin for horizontal
servo motor - using Servo library
  vertServo.attach(11); // digital pin for vertical servo
motor - using Servo library
}
```

The setup() function in Arduino code runs once when the program starts. Firstly, serial communication is enabled between the microcontroller and computer at baud rate of 9600, that is, 9600 bits per second. The horizontal servo motor is attached to digital pin 10. This configures the pin to control the servo's position. Similarly, the vertical servo motor is attached to digital pin 11 for controlling vertical movement. Both of the servo motors are attached by using the attach() method from Servo library. After this part of the code, the main function, which is the loop(), can be written.

```
void loop() {
```

The main logic of the system is executed continuously in the loop function. It processes the readings of the sensors and controls the servo motors based on the light detected.

```
int lightTopLeft = analogRead(ldrTopLeft);      // sensor
value
int lightTopRight = analogRead(ldrTopRight);    // sensor
value
int lightBottomLeft = analogRead(ldrBottomLeft);  // sensor
value
int  lightBottomRight  =  analogRead(ldrBottomRight);  //
sensor value
int delayTime = analogRead(4) / 18;   // potentiometer for
delay time - servo motor control
int tolerance = analogRead(5) / 4;    // potentiometer for
light intensity tolerance
```

Analog readings, measuring the light intensity, are taken from each LDR. Each LDR returns a value between 0 (dark) and 1023 (bright). The analogRead() function returns a value between 0 and 1023. This range corresponds to the 10-bit resolution of Arduino's Analog to Digital Converter (ADC). Potentiometers, connected to analog pins 4 and 5, control the delay between movements and the tolerance level for light differences. The delay helps make the servo motors movement smooth, and it eliminates any sudden movements. Dividing by 18 scales the value to create a reasonable delay in milliseconds. Similar scaling is done for the tolerance value. The tolerance determines the light threshold for which the servo motors should move.

```
int avgTop = (lightTopLeft + lightTopRight) / 2;    // average value of the top two sensors
int avgBottom = (lightBottomLeft + lightBottomRight) / 2;  // average value of the bottom two sensors
int avgLeft = (lightTopLeft + lightBottomLeft) / 2;  // average value of the two left sensors
int avgRight = (lightTopRight + lightBottomRight) / 2; // average value of the two right sensors
```

The average light intensity is calculated for the top, bottom, left, and right LDR pairs. This guides the system about the relative light strength in different directions. Light sensor values from the opposite pairs of LDRs are averaged.

```
int diffVertical = avgTop - avgBottom;    // measuring vertical differences (betweeen top and bottom)
int diffHorizontal = avgLeft - avgRight; // measuring horizontal differences (between left and right)
```

The difference between average light values for top and bottom sensors (diffVertical) is used to determine vertical servo motor's movement. The same applies for left and right sensors, where (diffHorizontal) is determined for the horizontal servo movement.

```
if (-1 * tolerance > diffVertical || diffVertical > tolerance) // if vertical difference exceeds tolerance
  {
    if (avgTop > avgBottom) // if top sensors detect more light than bottom sensors
    {
      vertAngle = ++vertAngle; // increment angle for the vertical servo motor
      if (vertAngle > 180)
      {
        vertAngle = 180; // boundary angle
      }
    }
   else if (avgTop < avgBottom) // if bottom sensors detect more light than top sensors
    {
      vertAngle = --vertAngle; // decrement angle for the vertical servo motor
      if (vertAngle < 0)
      {
```

```
        vertAngle = 0; // boundary angle
      }
    }
     vertServo.write(vertAngle); // update vertical servo
angle - using Servo library
  }
```

In this section, for the vertical movement, the difference in measured light intensity between the two top and the two bottom sensors, is compared to the light intensity tolerance. The segment (-1 * tolerance) ensures that the difference value (diffVertical) is compared against both positive and negative tolerance values. For example, if the tolerance is 50, then the difference between top and bottom light sensors must be greater than 50 or less than -50 to trigger a servo adjustment. If the top LDRs detect more light than the bottom, the servo angle is incremented (the panel moves up). The boundary condition ensures the angle does not exceed 180 degrees. If the bottom LDRs detect more light, the servo angle is decremented (the panel moves down), with the angle restricted to a minimum of zero (0) degrees. Finally, the calculated vertical angle is written to the servo, moving it.

```
if (-1 * tolerance > diffHorizontal || diffHorizontal >
tolerance) // if horizontal difference exceeds tolerance
  {
     if (avgLeft > avgRight)  // if sensors on the left
detect more light than sensors on the right
    {
       horizAngle = --horizAngle; // decrement horizontal
angle
      if (horizAngle < 0)
      {
        horizAngle = 0; // boundary angle
      }
    }
    else if (avgLeft < avgRight) // if sensors on the right
detect more light than sensors on the left
    {
       horizAngle = ++horizAngle; // increment horizontal
angle
      if (horizAngle > 180)
      {
        horizAngle = 180; // boundary angle
      }
    }
```

```
        horizServo.write(horizAngle);  //  update  horizontal
servo angle - using Servo library
  }
```

In this section, firstly, the difference between the light intensities on the left and right LDRs, is stored in (diffHorizontal). It determines if horizontal adjustment is necessary. The code checks if this difference exceeds the tolerance, prompting an adjustment if needed. If more light is detected on the left, the horizontal angle is decremented (the panel moves left). The condition ensures the angle does not go below 0 degrees, the servo motor's minimum angle value. If more light is detected on the right, the angle is incremented (the panel moves right), restricted to a maximum angle of 180 degrees. Finally, the horizontal angle of the servo motor is updated, physically adjusting the position of the system. Servo library enables the use of write() method in the program.

```
delay(delayTime); // delay for smooth operation, enabled
with the use of the potentiometer
}
```

The main loop() function ends with the line of code for delay, previously mentioned for being able to be modified with the use of potentiometer. The delay is applied to mainly slow down the loop and make the movements smoother, allowing for gradual adjustments of the two servo motors. Adjusting the potentiometer effectively changes the interval between the updates of the servo motors positions, allowing for smoother or quicker movements of the microcontroller-based smart solar tracking system. The Appendix chapter contains the entire code, written for this project in Arduino IDE.

CHAPTER 7

DISCUSSION - RESULTS

After constructing the 3D printed model, connecting the entire electrical circuit, and writing the code for the program, this project's prototype is completed. Now, the solar tracking system can begin to operate. A smart solar tracking system, based on a microcontroller, operates by continuously adjusting the angle of the servo motors to align with the strongest light source in its environment. This is analogous to a dynamic solar tracking system, which should position the solar panel towards the sun in the sky throughout the day and across seasons. Unlike stationary panels that are fixed in a single direction, the dual axis tracker moves on two axes: the horizontal (azimuth) and vertical (elevation). This movement allows the system to track the sun's trajectory from sunrise to sunset, as well as account for seasonal changes in the angle of the sun, optimizing energy absorption. In the prototype, the LDRs provide input to a microcontroller, in this case Arduino Uno, which processes the data and calculates the optimal positioning of the system. Based on this information, the microcontroller sends signals to the motors or servos that adjust the tilt (elevation) and rotation (azimuth).

For the testing of this system, a regular flashlight from a mobile phone can be used to test if the system operates the desired way. This flashlight is sufficient for covering small surfaces, such as the LDRs. It is ideal to use a smaller light source, when doing the testings of the prototype, and not yet include outside lights. Additionally, when testing the prototype, it was ensured that the amount of the light from outside is as minimized as possible. Firstly, all of the existing lights in the laboratory room at the university were turned off. After that, the curtains were shut, so the amount of light coming from outside is minimized. Furthermore, the 3D printed model contains an

LDR divider, mentioned in Section 3.1. LDR divider is mounted on the Panel Bracket, the 3D model where the panel is, later on, mounted. Careful creation of the 3D model ensures that this piece of the physical construction properly fits into its desired space. The desired space is a carefully calibrated, as well as deep enough indented in the Panel Bracket. In there, a small amount of glue can be placed in order for LDR divider to be stable. The purpose of LDR divider is to ensure that each LDR reading is independent. When placed at its location, it separates each LDR from the other ones, removing potential disturbances. A close visual of LDR divider can be observed in Figure 7.1, which itself represents the testing of the smart solar tracking system prototype, by shining a flashlight at a specific censor of the solar tracking system.

FIGURE 7.1. A View of the LDR Divider, shown while testing the Prototype

As observed from Figure 7.1, each of the LDRs is separated by the LDR divider, as defined in the code. As observed, the LDR in the bottom left position, shown in Figure 7.1, is responsible for the movements in the bottom and left directions. It makes the bottom light intensity average with bottom right LDR, and, left light intensity average with top left LDR. Basically, the bottom left LDR, naturally, detects the most amount

of light compared to others. This LDR will have a lower resistance and therefore allow more current to flow, which leads to a higher voltage drop across the regular resistor connected to this LDR in a voltage divider circuit. Arduino interprets this higher voltage as an indication that the light is the most intense in the bottom left direction. Other LDRs detect lower light intensities and have higher resistance, leading to lower voltage values across their regular resistors. Based on this data, the system adjusts the servo motors to rotate towards that direction, optimizing alignment with the light source. In general, LDR divider ensures the separation between the sensors, preventing potential interferences between them, making the system more accurate. Another example of system performance is given in Figure 7.2, where the servo motor is able to rise, since the flashlight was placed to shine the light from high above the sensors.

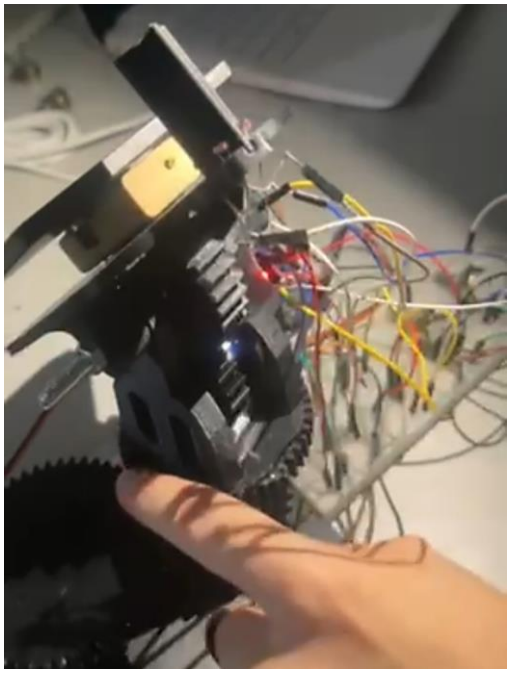

FIGURE 7.2. System Orientation towards the Top

In Figure 7.2, the servo motor for vertical movement is moved up, since it was detected by the LDRs that there is a strong amount of light in that direction. The servo motor approximately reaches its maximum vertical position in this example. In the code, it is already written that this position cannot be exceeded. The horizontal position of the system depicted by Figure 7.1, is practically the same as the one in Figure 7.2. The

horizontal position of the light source remained unchanged, while the vertical one was elevated, by adjusting the servo motor angle towards the top, approaching 180 degrees.

FIGURE 7.3. System Orientation towards the Top, Horizontal Adjustment

In Figure 7.3, a horizontal movement is performed, in order to orientate towards the strongest light source. This position is placed much more towards the right, and, thus, the servo motor completes a movement towards that position. After, the current horizontal position of the servo is maintained. This means that the light is shined almost equally placed between the LDRs on the left and the LDRs on the right side, meaning that the horizontal servo motor maintains approximately the same position, only making slight positional adjustments in that position.

The movement in either vertical or horizontal direction will appear when there is a clearly stronger source of light coming from a certain direction. Finally, a test performed when the strongest light source comes from the bottom direction, is shown in Figure 7.4 and Figure 7.5. The position shown in these two figures is close to the maximum bottom position of the vertical servo motor. Similarly to the maximum top limit for the vertical servo motor, in order to protect the circuit's wiring and prevent potential malfunctions, the maximum bottom limit is written in the code.

FIGURE 7.4. System Orientation towards the Bottom

The difference between Figure 7.4 and Figure 7.5 is in the horizontal orientation of the solar tracking system, since it is observed that vertical orientation stays approximately the same - placed towards the bottom. The flashlight shifts towards the right side in Figure 7.5, leading to the movement of the horizontal servo motor in that direction. The LDRs accurately detect the change in direction of the strongest light source.

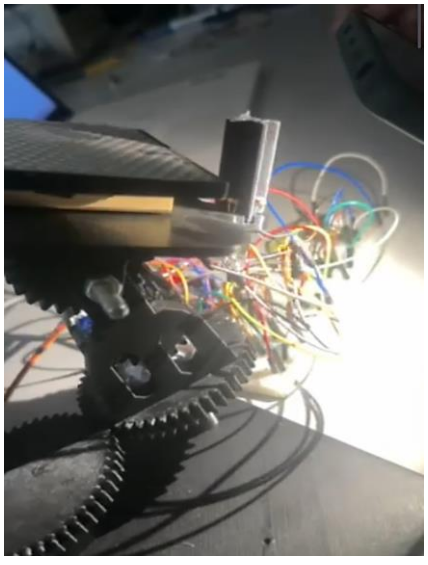

FIGURE 7.5. System Orientation towards the Bottom, Horizontal Adjustment

CHAPTER 8

CONCLUSION

The main goal of this project was to demonstrate the effective application of a microcontroller in developing an efficient solar tracking system. The project aims to optimize solar energy capture by ensuring the solar tracking system consistently aligns with the source of light. With optimization, upgrades and future scaling, the prototype will be able to significantly contribute to sustainable energy solutions and provides a variety of possibilites to revolutionize solar technology solutions. It is important to acknowledge limitations, including reliance on sensor technology and potential difficulties if scaling for multiple solar panels. However, by addressing and mitigating these limitations, the current development can lead to further improvements, drastically improving the efficiency of modern solar tracking systems.

Through integration of sensors, in the form of light dependent resistors, and servo motors, the system performs real time adjustments. The 3D printed components, modeled and assembled, provide a stable physical structure that houses the circuit with the electronic components. The circuit provides strong and reliable connections, making the system perform consistently. One of the significant advantages of Arduino is its ability to be interfaced with various components. The use of the Arduino IDE allowed for coding and debugging processes. Defining the specific constraints for servo motors and LDRs, along with writing the setup and loop functions, forms the foundation of the project code. The code reads real time data about the light intensity from the LDRs, processes it, and moves the servo motors accordingly.

The final code implementation ensures a smooth and efficient operation by controlling the servo motors based on the input received from sensors. It also allows for customization, enabling users to adjust parameters such as delay time and light tolerance, making the system adaptable to specific applications and conditions. The results obtained from testing the final prototype indicate a realistic possibility for a strong and significant improvements in solar energy capture, compared to traditional static solar panel systems. The dual axis tracking system consistently adjusts to the optimal angles, maximizing energy absorption throughout the day, regardless of the weather conditions. The performance validation proves the potential for practical applications of the system in residential and commercial solar energy installations.

This project successfully achieved its goals by combining the advanced microcontroller capabilities with practical engineering solutions. The microcontroller-based smart solar tracking system provides an innovative application of both electronics and embedded systems in the field of renewable energy.

REFERENCES

[1] Mahendran, M., Ong, H.L., Lee, G.C., & Thanikaikumaran, K. (2013). An experimental comparison study between single-axis tracking and fixed photovoltaic solar panel efficiency and power output: Case study in East Coast Malaysia. *CORE*. Retrieved September 7, 2024 from https://core.ac.uk/download/pdf/159179631.pdf

[2] Nureni, Y. (2022). A comparative study of using fixed solar and solar tracker panels in photovoltaic energy generation system. *International Journal of Engineering Technology Sciences & Entrepreneurship (IJETSE)*. Retrieved September 9, 2024 from https://www.researchgate.net/publication/359391673_a_comparative_study_of_using_fixed_solar_and_solar_tracker_panels_in_photovoltaic_energy_generation_system

[3] Sampaio, P. G. V., & González, M. O. A. (2017). Photovoltaic solar energy: Conceptual framework. *Renewable and Sustainable Energy Reviews*, *74*, 590–601. Retrieved September 9, 2024 from https://www.sciencedirect.com/science/article/abs/pii/S1364032117303076?via%3Dihub

[4] Krismadinata, K., Asnil, A., Husnaini, I., Lapisa, R., Maulana, R., Yuhendri, M., Astrid, E., Logamani, P. (2023). Photovoltaic Energy Harvesting with Static and Dynamic Solar Modules Employing IoT-Enabled Performance Monitoring. *TEM Journal*, 12(3), 1354–1363. Retrieved September 7 2024 from https://www.temjournal.com/content/123/TEMJournalAugust2023_1354_1363.pdf

[5] Barua, S., & Rahman, M. (2023). Analysis of power consumption of different microcontroller. *North South University*. Retrieved September 3, 2024 from https://www.researchgate.net/publication/366812314_Analysis_of_Power_Consumption_of_Different_Microcontroller

[6] Ma, Q., Rejab, R., Kumar, A.P., Fu, H., Kumar, N.M., Tang, J. (2020). Effect of infill pattern, density and material type of 3D printed cubic structure under quasi-static loading. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, pp. 1–19. Retrieved August 1, 2024 from https://doi.org/10.1177/0954406220971667

[7] Deshmukh, G., Gawade, V., Godse, V., Londhe, N., & Gawari, D. (2018). Smart Navigational Shoes for Bikers/Cyclists. *International Journal of Computer Applications*, 180, 6–10. Retrieved September 9, 2024 from https://doi.org/10.5120/ijca2018917062

[8] Components 101 (2021). *Arduino Uno*. Retrieved September 10, 2024 from https://components101.com/microcontrollers/arduino-uno

[9] Di Pasquo, G. (2021). SG90 Servo Characterization. *Zenodo (CERN European Organization for Nuclear Research)*. Retrieved August 3, 2024 from https://www.researchgate.net/publication/353754375_SG90_Servo_Characterization

[10] H2W Technologies (2015). *Closed Loop Servo Motor Operating Instructions*. Retrieved August 4, 2024 from https://www.h2wtech.com/page/closed-loop-servo-motor-operating-instructions

[11] Pcheung Teaching (2019). *SERVO MOTOR SG90 DATA SHEET*. Retrieved August 3, 2024, from http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

[12] Mustafa, K. R., Mustafa, R. M., & Ramadani, R. M. (2023). Measuring the Voltage, Current and Resistance of the LDR Sensor through the Arduino UNO. *Asian Journal of Research in Computer Science*, 16(4), 211-222. Retrieved August 5, 2024 from https://www.researchgate.net/publication/375038565_Measuring_the_Voltage_Current_and_Resistance_of_the_LDR_Sensor_through_the_Arduino_UNO

[13] Haraoubia, B. (2018). Nonlinear Two-terminal Devices. In B. Haraoubia (Ed.), *Nonlinear Electronics 1* (pp. 1-81). Algiers. Elsevier.

[14] YIC-Electronics (2024). *What is a 10k Resistor? 10k Ohm Resistor Color Code*. Retrieved August 5, 2024, from https://www.yic-electronics.com/blog/What-is-a-10k-Resistor-10k-Ohm-Resistor-Color-Code.html

[15] Schweber, B. (2021). The Fundamentals of Digital Potentiometers and How to Use Them. *DigiKey*. Retrieved September 6, 2024 from https://www.digikey.com/en/articles/the-fundamentals-of-digital-potentiometers

[16] Makeability Lab (2020). *Lesson 4: Potentiometers*. Retrieved September 6, 2024 from https://makeabilitylab.github.io/physcomp/arduino/potentiometers.html

APPENDICES

A code, required for the microcontroller-based smart solar tracking system to operate, is written for Arduino Uno Microcontroller, in Arduino IDE compiler. This appendix contains the full code written for this project. The code is presented below, with brief explanations for it written in the comments next to the individual lines of code.

```
#include <Servo.h>

// instances of the servo class from Servo library
Servo horizServo; // servo for horizontal movement
int horizAngle = 90; // setting the initial position
Servo vertServo; // servo for vertical movement
int vertAngle = 90; // setting the initial position

int ldrTopLeft = 0;  // top left LDR - analog pin
int ldrTopRight = 1; // top right LDR - analog pin
int ldrBottomLeft = 2; // bottom left LDR - analog pin
int ldrBottomRight = 3; // bottom right LDR - analog pin

void setup()
{
  Serial.begin(9600); // baud rate
   horizServo.attach(10);  //  digital  pin  for  horizontal
servo motor - using Servo library
  vertServo.attach(11); // digital pin for vertical servo
motor - using Servo library
}

void loop()
{
  int lightTopLeft = analogRead(ldrTopLeft);     // sensor
value
  int lightTopRight = analogRead(ldrTopRight);   // sensor
value
```

```
   int lightBottomLeft = analogRead(ldrBottomLeft);   // sensor value
   int lightBottomRight = analogRead(ldrBottomRight); // sensor value
  int delayTime = analogRead(4) / 18;    // potentiometer for delay time - servo motor control
  int tolerance = analogRead(5) / 4;     // potentiometer for light intensity tolerance

  // determining the difference between light in opposing directions
  int avgTop = (lightTopLeft + lightTopRight) / 2;    // average value of the top two sensors
  int avgBottom = (lightBottomLeft + lightBottomRight) / 2;  // average value of the bottom two sensors
  int avgLeft = (lightTopLeft + lightBottomLeft) / 2;  // average value of the two left sensors
  int avgRight = (lightTopRight + lightBottomRight) / 2; // average value of the two right sensors
  int diffVertical = avgTop - avgBottom;    // measuring vertical differences (betweeen top and bottom)
  int diffHorizontal = avgLeft - avgRight; // measuring horizontal differences (between left and right)

   if (-1 * tolerance > diffVertical || diffVertical > tolerance) // if the vertical light difference exceeds tolerance
  {
    if (avgTop > avgBottom) // if top sensors detect more light than bottom sensors
    {
      vertAngle = ++vertAngle; // increment angle for the vertical servo motor
      if (vertAngle > 180)
      {
        vertAngle = 180; // boundary angle
      }
    }
   else if (avgTop < avgBottom) // if bottom sensors detect more light than top sensors
    {
      vertAngle = --vertAngle; // decrement angle for the vertical servo motor
      if (vertAngle < 0)
      {
        vertAngle = 0; // boundary angle
      }
```

```
    }
     vertServo.write(vertAngle); // update vertical servo
angle - using Servo library
  }

  if (-1 * tolerance > diffHorizontal || diffHorizontal >
tolerance) // if the horizontal light difference exceeds
tolerance
  {
     if (avgLeft > avgRight)  // if sensors on the left
detect more light than sensors on the right
    {
       horizAngle = --horizAngle; // decrement horizontal
angle
      if (horizAngle < 0)
      {
        horizAngle = 0; // boundary angle
      }
    }
    else if (avgLeft < avgRight) // if sensors on the right
detect more light than sensors on the left
    {
       horizAngle = ++horizAngle; // increment horizontal
angle
      if (horizAngle > 180)
      {
        horizAngle = 180; // boundary angle
      }
    }
      horizServo.write(horizAngle);  //  update  horizontal
servo angle - using Servo library
  }

  delay(delayTime); // delay for smooth operation, enabled
with the use of the potentiometer
}
```