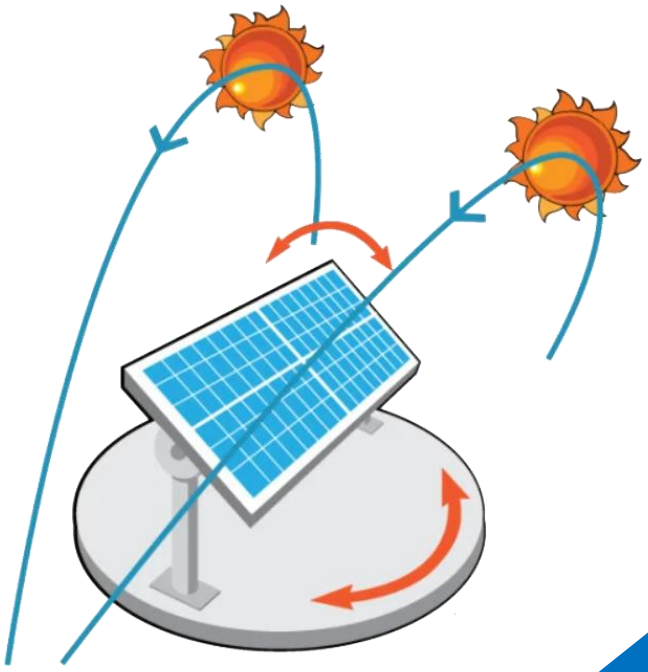
Mansoura University – Faculty of Engineering
Mechatronics Engineering Program

TRACKING THE SUN

An Innovative **Dual Axis Solar Tracker** With Enhanced Weather Monitoring



January 28th, 2023

UNDER THE SUPERVISION OF:

Eng. Mohamed Zaki
Eng. Ahmed Beshbeshy
Eng. Mohamed ElMadawy

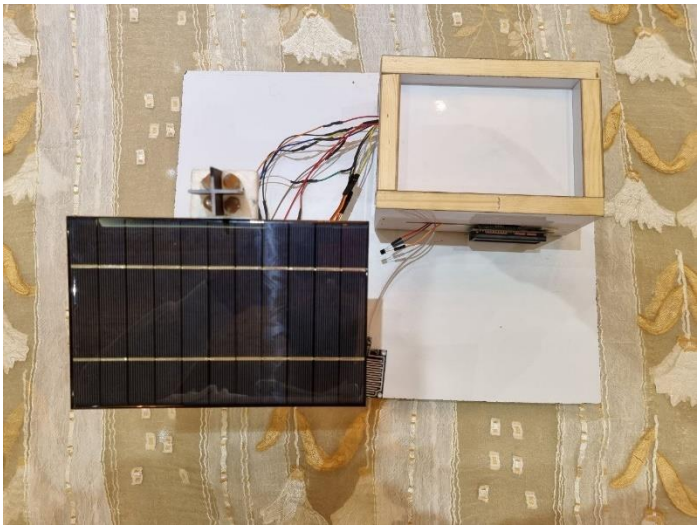
Dual Axis Solar Tracker

Project Technical Report
Internal Training

TEAM MEMBERS

Software & Testing	800167243	هاني علي إبراهيم العيسوي
Software & Simulation	800161918	محمد محمود محمد السعيد زهران
PCB Design	800161979	محمد محمود محمد علي أبو سراري
PCB Fabrication	800161798	ريم سامح الغرباوي محمد
Mechanical Design	803084389	سلمى حمدي رمضان المرسي
Research & Documentation	800162016	ناهد ممدوح عبد الرحمن عبد الله
Assembly	800167244	محمد علاء محمد خليل القوقا

ABSTRACT



The report **"Tracking The Sun: An Innovative Dual Axis Solar Tracker With Enhanced Weather Monitoring"** presents an in-depth analysis of the design, development, and implementation of a dual-axis solar tracker. The project aimed to improve the efficiency and performance of solar panels by utilizing a dual-axis tracking system to follow the movement of the sun throughout the day.

The report begins by providing an **overview of energy resources**, with a focus on the differences between non-renewable and renewable energy sources. It then delves into the concept of solar tracking, explaining the benefits and limitations of various solar tracking systems.

The next chapter of the report covers the **brainstorming, researching, and budget estimation process**. The team began by brainstorming different ideas for the project, including the selection of components and the overall design. This was followed by extensive research on the latest technologies and industry trends related to dual-axis solar trackers. Finally, the team estimated the budget for the project and kept track of the actual expenses incurred.

One of the key aspects of the report is the detailed analysis of the **mechanical components** and their interactions. The team examined the different components, such as the solar panel, the motor, and the servos, and how they were assembled together. This helped them understand

how the mechanical components interacted with each other and how the servos were placed in relation to the other components.

The chapter on **hardware components and PCB fabrication** in the book delves into the importance of selecting the right components for the project. It covers the various sensors used, including the Light Dependent Resistance (LDR) sensor, the Temperature Sensor (LM35), and the Rain Sensor. The section on the Rain Sensor provides an in-depth analysis of its specifications and the reasons for its inclusion in the project. Additionally, the chapter also covers the Servo Motor specifications and its important role in the overall functioning of the dual-axis solar tracker. The chapter provides valuable insights for readers interested in understanding the various components used in the design and development of a dual-axis solar tracker.

The report also covers the analysis of LDR sensor placement and orientation. The placement and orientation of the LDR sensors are critical factors in ensuring accurate tracking and measurements of light intensity so that the servos can make precise movements. The team took a different approach in the placement and orientation of the Light Dependent Resistor (LDR) sensors. Instead of placing them at the corners of the solar tracker, they created a custom 3D printed piece in the shape of a cross that was placed at the top center of the solar panel.

The report also covers the **programming, code, and simulation** aspects of the project. The team used Arduino IDE and Visual Studio Code as their working environment and developed the code for the dual-axis solar tracker. The report also includes a detailed explanation of the general dual-axis tracking logic and how it was implemented in the code.

Finally, the report concludes by **summarizing** the key findings and results of the project. The team was able to successfully design and develop a dual-axis solar tracker that improved the efficiency and performance of solar panels. The report also highlights the enhanced weather monitoring capabilities that were integrated into the project. Overall, the report provides valuable insights for anyone interested in the design and development of dual-axis solar trackers.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER (1)
INTRODUCTION

1.1 ENERGY RESOURCES

Energy is a vital component of our daily lives, powering everything from our homes to our cars to our factories. It is essential for modern society to function smoothly and efficiently. However, not all energy sources are created equal. Overall, energy resources are essential for powering our daily lives, maintaining our economy, and creating a sustainable future. Investing in renewable energy sources and energy efficiency can help us to meet our energy needs while also protecting the environment and public health. In this Section, we will delve into the differences between renewable and non-renewable energy sources, examining their pros and cons and exploring the potential for each to shape our energy future. We will explore the science behind these energy sources.

1.1.1 NON-RENEWABLE ENERGY

Non-renewable energy sources, also known as fossil fuels, are energy sources that are formed over millions of years from the remains of plants and animals. These sources are finite and will eventually run out, hence the name "non-renewable." The most common non-renewable energy sources are coal, oil, and natural gas.

Figure 1 – Non-Renewable Energy Resource (Coil)

Non-renewable energy sources have been the primary source of energy for the world for centuries, providing the power needed for industrialization, urbanization and economic growth. However, their finite nature, combined with the negative impacts of their extraction, transportation, and use on the environment and public health, has led to a growing interest in renewable energy sources.

Figure 2 – Non-Renewable Energy Resource (Oil)

One of the main disadvantages of non-renewable energy sources is that they are finite and will eventually run out. As the demand for energy continues to increase, the extraction of these resources becomes increasingly difficult and expensive. Additionally, the extraction and transportation of these resources can lead to environmental damage, such as habitat destruction, air and water pollution, and greenhouse gas emissions.

Another major disadvantage of non-renewable energy sources is their negative impact on public health. The burning of fossil fuels releases pollutants into the air, which can lead to respiratory and cardiovascular diseases. The extraction and transportation of these resources can also lead to spills and other accidents, which can have serious health consequences for workers and local communities.

Figure 3 – Pollution Resulted from Oil Refiners

Despite the limitations and negative impacts of non-renewable energy sources, they will likely continue to play a role in our energy mix for the foreseeable future. However, as the world moves towards a more sustainable future, it is important to invest in renewable energy sources and energy efficiency measures to reduce our dependence on non-renewable energy sources and mitigate their negative impacts.

1.1.2 RENEWABLE ENERGY

Renewable energy is energy that is generated from natural resources, such as sunlight, wind, rain, and geothermal heat, which are replenished naturally. Unlike non-renewable energy sources, such as coal, oil, and natural gas, renewable energy sources produce significantly less pollution and greenhouse gas emissions, making them a cleaner and more environmentally friendly option. There are several types of renewable energy, including solar, wind, hydro, geothermal, and biomass. Solar energy is generated by harnessing the power of the sun through the use of solar panels. Wind energy is generated by harnessing the power of the wind through the use of wind turbines.

Figure 4 – Wind Turbines

Hydro energy is generated by harnessing the power of moving water through the use of hydroelectric power plants. Geothermal energy is generated by harnessing the heat of the earth through the use of geothermal power plants. Biomass energy is generated by burning organic matter, such as wood or waste, to produce heat or electricity.

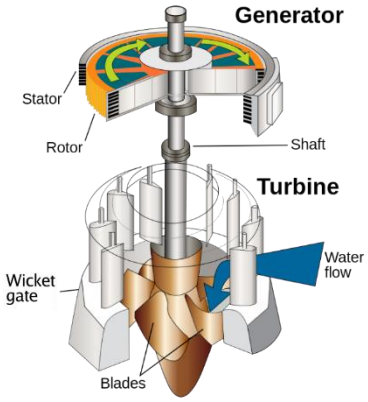


Figure 5 – Construction of Hydroelectric Turbine

The advantages of renewable energy over non-renewable energy include sustainability, cost-effectiveness, resilience, job creation, and the ability to generate power locally. Renewable energy is sustainable because it is replenished naturally and will never run out. The cost of renewable energy is decreasing, making it more cost-effective in the long term than non-renewable energy. Renewable energy sources are distributed and decentralized, meaning they are less vulnerable to disruption from natural disasters or other events.

Renewable energy sources require more labor per unit of energy generated than non-renewable energy sources, which creates more jobs. Renewable energy sources can be used to generate power locally, which reduces the need for expensive transmission and distribution infrastructure.

Solar energy is the fastest-growing renewable energy source in the world and it is considered as one of the most promising renewable energy sources. Solar energy is a clean, sustainable, and cost-effective way to generate electricity, and it can be used in a variety of applications, including residential, commercial, and industrial.

Figure 6 – Solar Energy Plant in Egypt

Solar energy has the potential to provide a significant portion of the world's energy needs and it could be a key component in the transition to a low-carbon economy.

1.2 SOLAR TRACKING

Solar tracking is a technology that allows solar panels to move with the sun throughout the day, in order to optimize the amount of energy they generate. There are three main types of solar trackers: stationary, single axis, and dual axis.

Figure 7 – Stationary Solar Panel Used to Generate Electricity for a House

Stationary solar panels are fixed in one position, usually facing south in the northern hemisphere, and are not able to move with the sun. These types of solar panels are the simplest and least expensive to install, but they also generate the least amount of energy because the panels are not always facing the sun directly.

Single axis solar trackers are able to move in one direction, usually from east to west, in order to follow the sun as it moves across the sky. This allows the panels to generate more energy than stationary panels, but they are still not able to adjust to the sun's elevation in the sky. Single axis solar trackers are more expensive to install than stationary panels, but they generate more energy, making them a more cost-effective option in the long term.

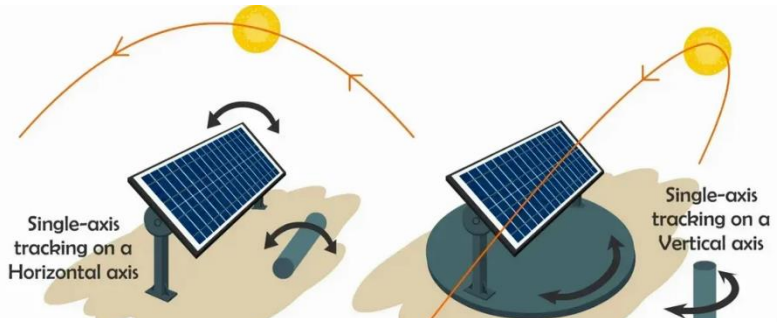


Figure 8 – Tow Different Types of Single Axis Trackers

Dual axis solar trackers are the most advanced type of solar tracker, and are able to move in two directions, following the sun as it moves across the sky and adjusting to its elevation. This allows them to generate the most energy of all three types of solar trackers. Dual axis solar trackers are the most expensive to install, but they generate the most energy and are the most cost-effective option in the long term.

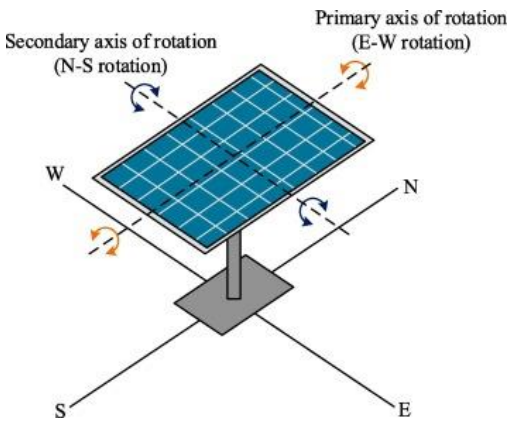


Figure 9 – Dual Axis Tracker

In summary, the choice between stationary, single axis, and dual axis solar trackers depends on the specific needs of the project and budget available. Stationary solar panels are the most cost-effective option, but they generate the least amount of energy. Single axis solar trackers generate more energy than stationary panels, but they are not able to adjust to the sun's elevation. Dual axis solar trackers generate the most energy and are the most cost-effective option in the long term, but they are also the most expensive to install.

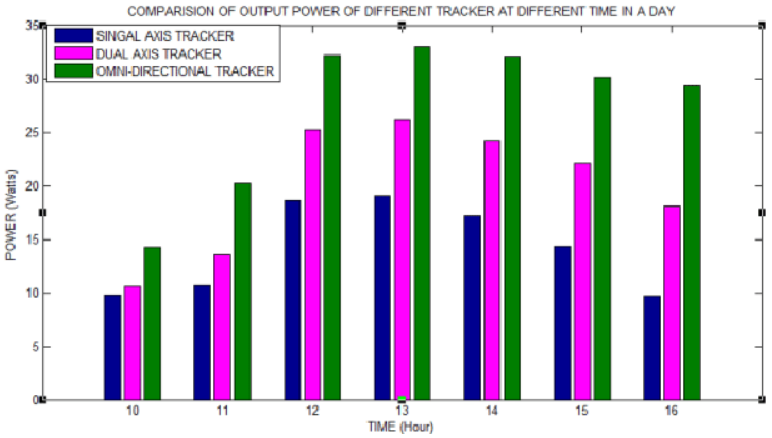


Figure 11 - Output Power of Different Trackers Throughout a Day

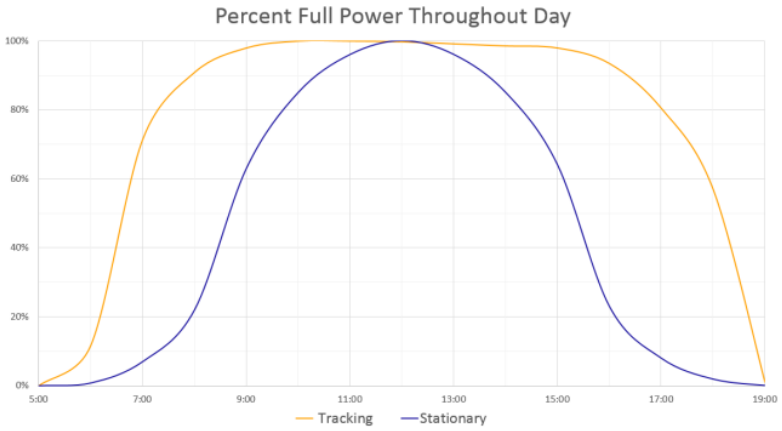


Figure 10 - Maximum Power Using Stationary Solar Panels vs Solar Trackers

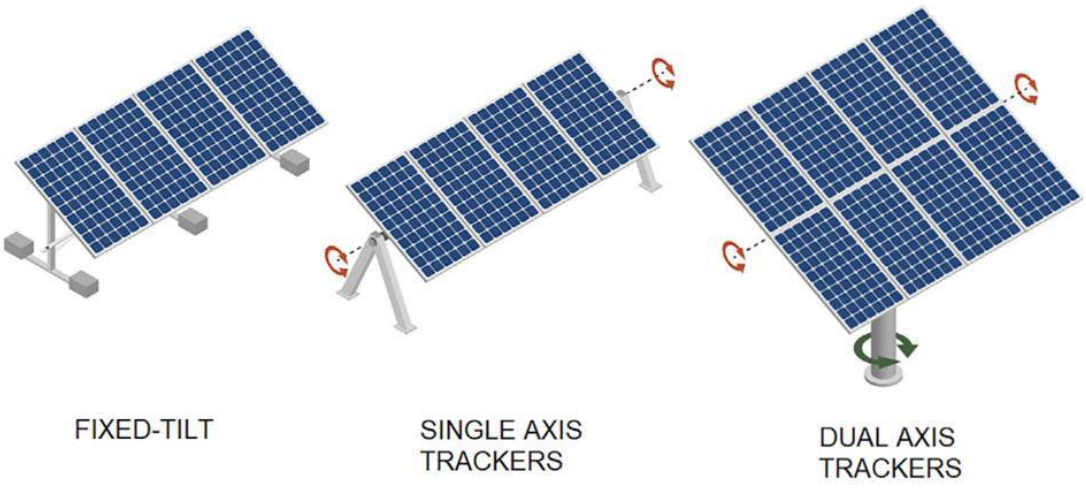


Figure 12 - Comparison Between Stationary, Single Axis and Dual Axis Solar Trackers

1.3 PROJECT TIMELINE

1. Brainstorming
2. Researching
3. Budget Estimation
4. Mechanical Design & Printing
5. Hardware Components & PCB Fabrication
6. Programming, Code & Simulation
7. Testing & Assembling
8. Documentation

CHAPTER (2)
BRAINSTORMING, RESEARCHING AND BUDGET ESTIMATION

2.1 BRAINSTORMING

Brainstorming is an important phase in any project, including a dual axis solar tracker project. During the brainstorming phase, team members come together to generate new ideas and think creatively about how to approach the project. It's a method of generating a large number of ideas in a short period of time.

The brainstorming process can be conducted in a variety of ways, but it typically involves bringing together a group of people who have relevant expertise or experience related to the project. This group then works together to generate as many ideas as possible, without judging or evaluating them at this stage. The goal is to come up with as many new and creative ideas as possible, even if they seem impractical or unrealistic.

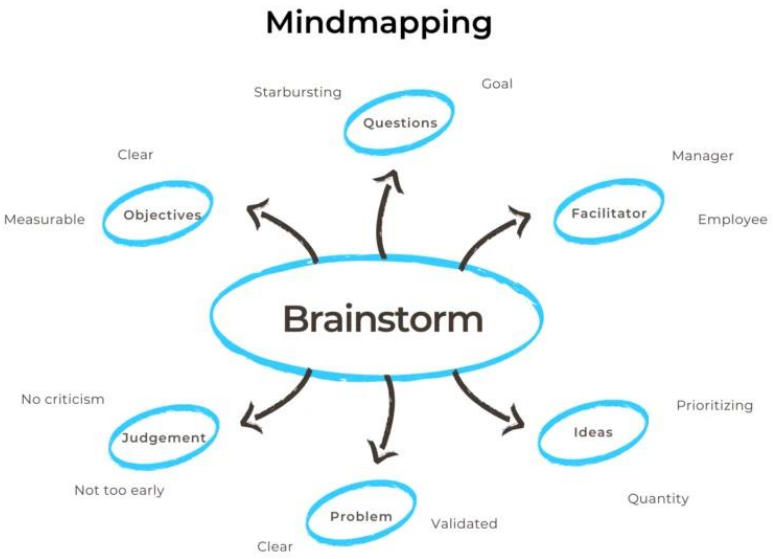


Figure 13 - Brainstorming Parts

To facilitate the brainstorming process, it's important to create an open and inclusive environment where everyone feels comfortable sharing their ideas. It's important to make it clear that there are no bad ideas and that everyone's input is valued. It's also important to establish ground rules for the brainstorming session, such as no criticism or negativity, and encouraging the free flow of ideas.

Once all ideas have been generated, the next step is to evaluate them and select the most promising ones to pursue further. This can be done through a process of voting, discussion, or other methods. It's important to keep in mind that the goal of brainstorming is to generate new and creative ideas, not to find the perfect solution right away.

2.2 RESEARCHING

The researching stage in a dual axis solar tracker project is a crucial step in the development process, as it sets the foundation for the success of the project. This stage involves identifying the most suitable body design for the project, selecting the best components to use, and gathering code resources and general information about solar energy and solar tracking projects.

Figure 14 - Research Phase

When it comes to design, it is important to consider factors such as the size of the solar panel, the location of the tracker, and the expected performance of the system. One important factor to consider is the type of solar tracking system, such as single axis and dual axis systems. Dual axis systems are more complex but generally more efficient, as they can track the sun's movement in both the horizontal and vertical planes. It is important to consider the trade-off between complexity and efficiency when choosing a design.

The selection of components is also an important part of the researching stage. It is important to choose high-quality components that are reliable and efficient. This can include studying different types of motors, sensors, and control systems that are used in solar tracking systems. It is also important to consider the cost and availability of the components when making a selection.

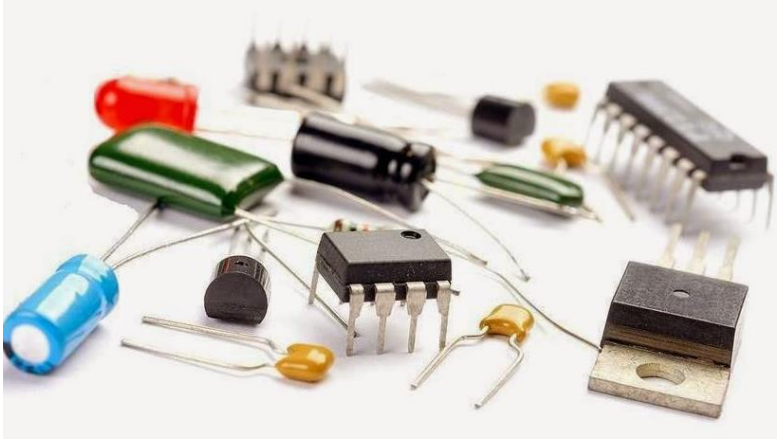


Figure 15 - Selection of Electronic Components

Gathering code resources and general information about solar energy and solar tracking projects is also an important part of the researching stage. This can include researching open-source libraries and programming languages that can be used to control the solar tracker. It can also include studying the basics of solar energy, including how solar panels work and the different types of solar panel technologies that are available.

The control system is also a crucial aspect of a solar tracker's performance. It is responsible for controlling the movement of the solar panels and ensuring that they are oriented towards the sun. Engineers and designers can improve the control system by using microcontroller such as Arduino or AVR Microcontroller. These microcontrollers can be programmed and configured to control the movement of the solar panels and to ensure that they are oriented towards the sun. These microcontrollers are widely used in the industry and have many resources and libraries available for use in the control system.

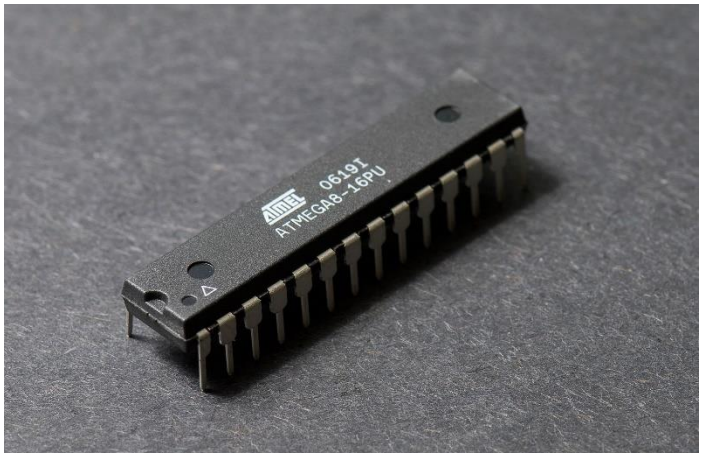


Figure 16 - Atmega328 Chip Used in Arduino

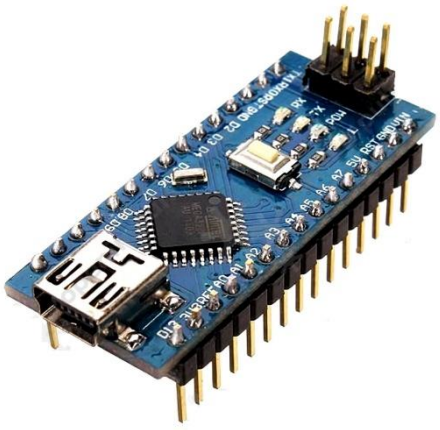


Figure 17 - Microcontroller Used in Project (Arduino Nano)

Improving solar trackers requires a thorough understanding of the design, components, and control system of the tracker. By carefully considering these factors, designers and engineers can create solar trackers that are more efficient, cost-effective, and durable, ultimately leading to a better performance of the solar tracker.

Overall, the researching stage is a vital step in ensuring that the dual axis solar tracker project is successful. It will help to identify the most efficient and cost-effective design, components, and code resources, which will ultimately lead to a better performance of the solar tracker.

2.3 BUDGET

2.3.1 ESTIMATED BUDGET

LDR	4	£32.00	Link	T-Block	2	£5.00	Link
100k ohm Resistors	4	£1.00	Link	Solar Panel	1	£210.00	Link
Mini Servo Motors	2	£200.00	Link	PCB + Fabrication	1	£50.00	Link
Arduino Nano	1	£170.00	Link	Raindrop Sensor	1	£30.00	Link
L7805 Regulator	1	£3.50	Link	LCD (option)	1	£6.00	Link
L7809 Regulator	1	£3.50	Link	Toggle Switch	1	£6.00	Link
Capacitors	4	£4.00	Link	Buzzer	1	£5.00	Link
Battery Holder	1	£20.00	Link				
3.7 Battery	3	£45.00	Link				
Total To Pay		£791.00					

Figure 18 – Estimated Budget for the Project

2.3.2 ACTUAL BUDGET

Hardware	Price	Hardware	Price	Mechanical	Price
Rain_drop_sens	£30.00	PCB-Single Layer	£20.00	3D Printed	£35.00
LCD	£45.00	Arduino nano	£170.00	3D Printed	£40.00
LM35	£60.00	Solar Panel	£230.00	Grind	£100.00
Toggle switch	£6.00	Servo Motors	£200.00	Reports Print	£200.00
7805-7809	£7.50	PCB Fabrication	£50.00		
Fuse-Holder	£5.00	batteries	£45.00		
Buzzer	£5.00	battery holder	£20.00		
Potentiometer	£1.50	LDR	£35.00		
Resistors	£1.00	jmb	£10.00		
Connectors	£17.00	Capacitors	£2.00		
T-Block	£15.00				
Total Paid :		£1,350.00			

Figure 19 – Actual Budget for the Project

CHAPTER (3)
MECHANICAL DESIGN AND PRINTING

3.1 SELECTION OF SUITABLE MECHANISM

The mechanical design phase is an important step in the development of a dual axis solar tracking system. During this phase, engineers and designers must consider a variety of factors to create a design that is both efficient and reliable.

One of the key considerations during the mechanical design phase is the size and weight of the solar panels. This will determine the size and strength of the mechanical components needed to support the panels and move them as needed to track the sun. Additionally, the location of the solar tracker must also be taken into account, as certain environments may require specific design considerations such as wind resistance or durability in harsh weather conditions.

In order to track the movement of the sun in two directions, various positions of Light Dependent Resistors (LDRs) have been utilized. One such configuration is depicted in *Figure 20A*, which features a solar sensing device consisting of a four-quadrant LDR sensor and a cylindrical shade. This design utilizes the principle of shadowing, where the PV panel's orientation is determined by the intensity of light received by the LDRs. If the panel is not perpendicular to the sunlight, the shadow of the cylinder will fall on one or two of the LDRs, resulting in a differential of light intensity.

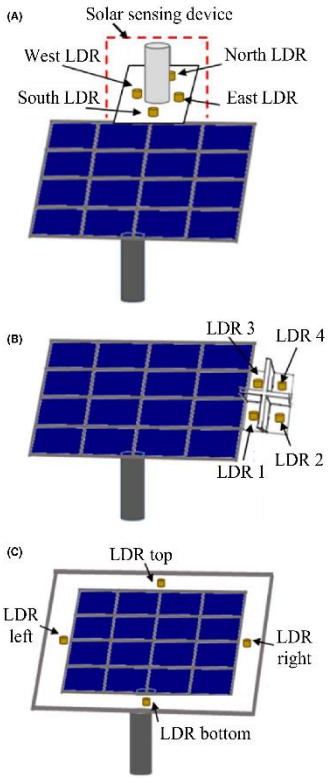


Figure 20 – Different Positions of LDRs

Another configuration, as shown in *Figure 20B*, includes a four-quadrant LDR sensor that is separated by two barriers. In this configuration, the optimal orientation of the PV panel is achieved when the intensity of light received by LDR1 and LDR2 is equal to that received by LDR3 and LDR4. Both of these sensing devices can be positioned at the top, left, or right of the PV panel. It is important to note that when using either of these devices, the dimensions of the barrier (length) and the distance between the LDR sensors and the barrier should be carefully considered to ensure accurate measurements.

In Figure 20C, the LDRs are positioned on the center of each side of the PV panel. This configuration allows for precise tracking of the sun's movement in two directions and is widely used in solar tracking systems.

3.2 ACQUIRING EXACT DIMENSIONS

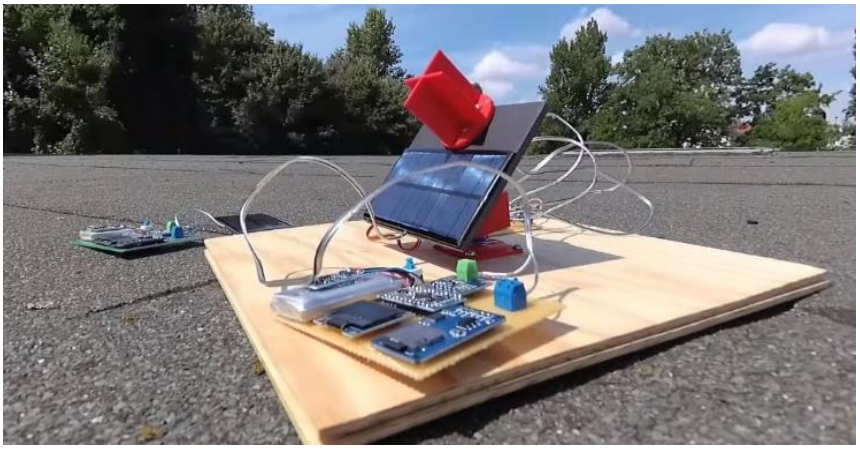


Figure 21 – LDRs Position Applied in the Project

The positioning of LDRs as described in the previous paragraph plays a crucial role in the mechanical design of a dual-axis solar tracker project. The information gathered by the LDR sensors is used to accurately orient the PV panel towards the sun, allowing for maximum solar energy capture. The design and dimensions of the barriers, as well as the distance between the LDR sensors and the barriers, must be carefully considered to ensure optimal performance of the solar tracker. The configuration of LDRs in the center of each side of the PV panel, as shown in Figure 20C, is often preferred as it allows for precise tracking and efficient energy capture. Overall, the proper placement and design of LDRs is a key aspect of the mechanical design of a dual-axis solar tracker project.

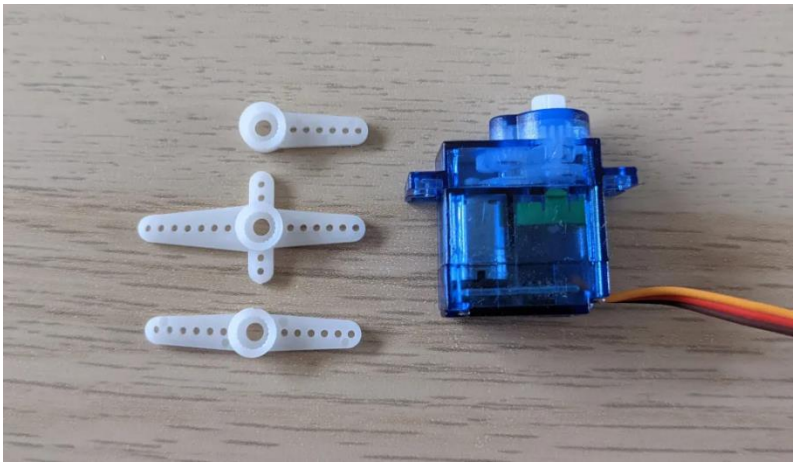


Figure 22 - Servo Motor with Different Size Horns

One of the significant challenges encountered during the design and development process of the dual-axis solar tracker project was determining the precise dimensions of the servo motors and their horns. This was crucial in ensuring that the movement of the solar panel was as smooth and precise as possible. The dimensions of the servos and their horns play an essential role in the overall performance of the solar tracker, as they

affect the accuracy and efficiency of the solar panel's movement. The task of finding these dimensions required careful research and experimentation to ensure that the final design met the project's requirements and specifications. Additionally, the alignment and calibration of the servos and horns are also important to ensure that the panel is always pointing in the right direction for maximum sunlight capture. Overall, the determination of the servos and horns dimensions was a critical aspect of the project that required a significant amount of attention and effort to achieve optimal results.

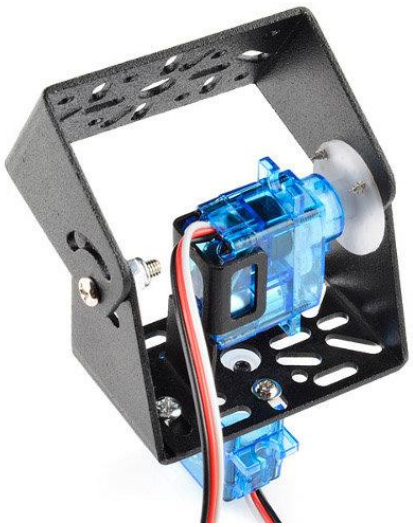


Figure 23 – Servo Motors Fixed in Pan Tilt Bracket

3.3 SOLIDWORKS PARTS & 3D PRINTING

In summary, the mechanical design phase is a crucial aspect in the development of a dual axis solar tracking system. Engineers and designers must pay attention to a wide range of factors to create a design that is both efficient and reliable. Factors to consider include the size and weight of the solar panels, the location of the solar tracker and the environmental conditions, the type of drive system used to move the solar panels, and the control system that will be used to position the solar panels in relation to the sun. By considering these factors and making appropriate design choices, the final design will be able to achieve optimal performance and efficiency.

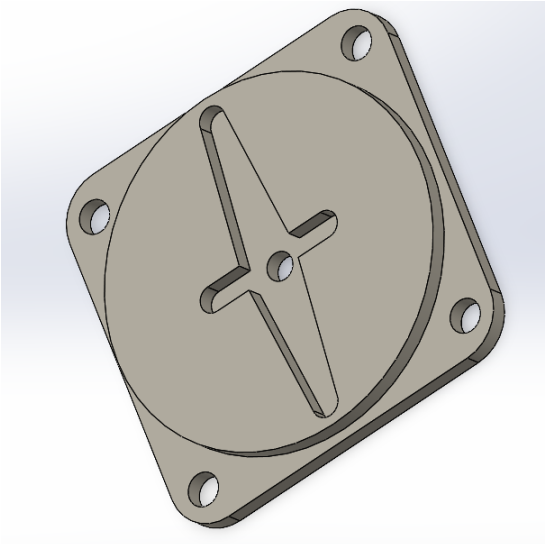


Figure 24 – Mechanical Design (Part 1)

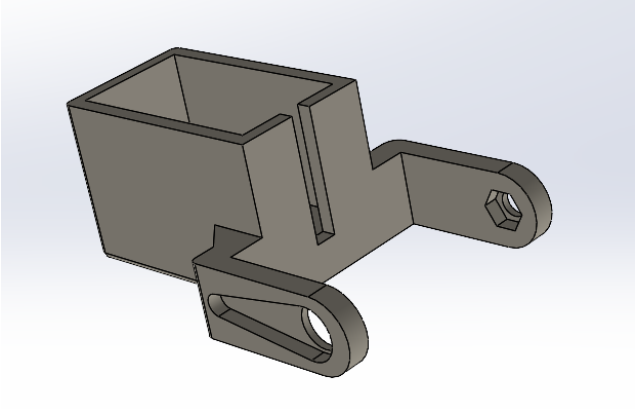


Figure 25 – Mechanical Design (Part 2)

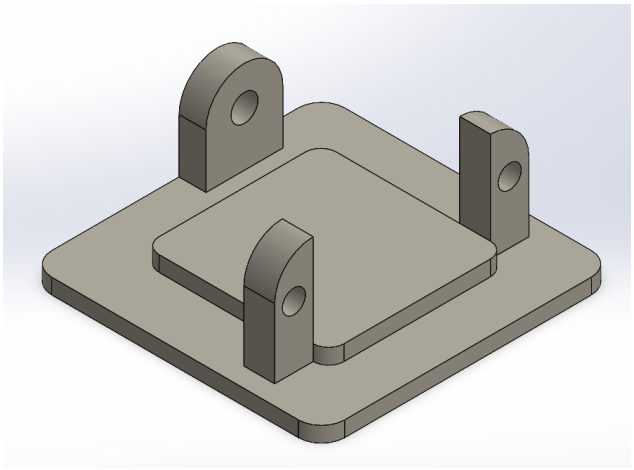


Figure 26 – Mechanical Design (Part 4)

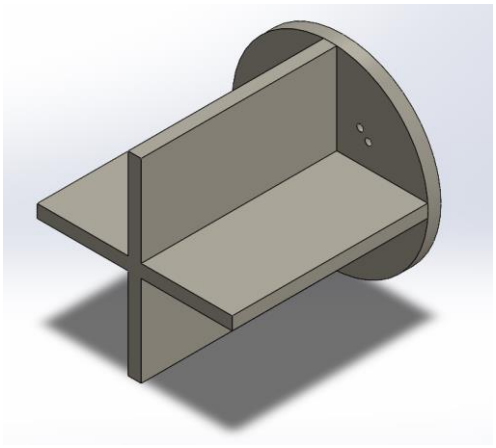


Figure 27 – Mechanical Design (Part 3)

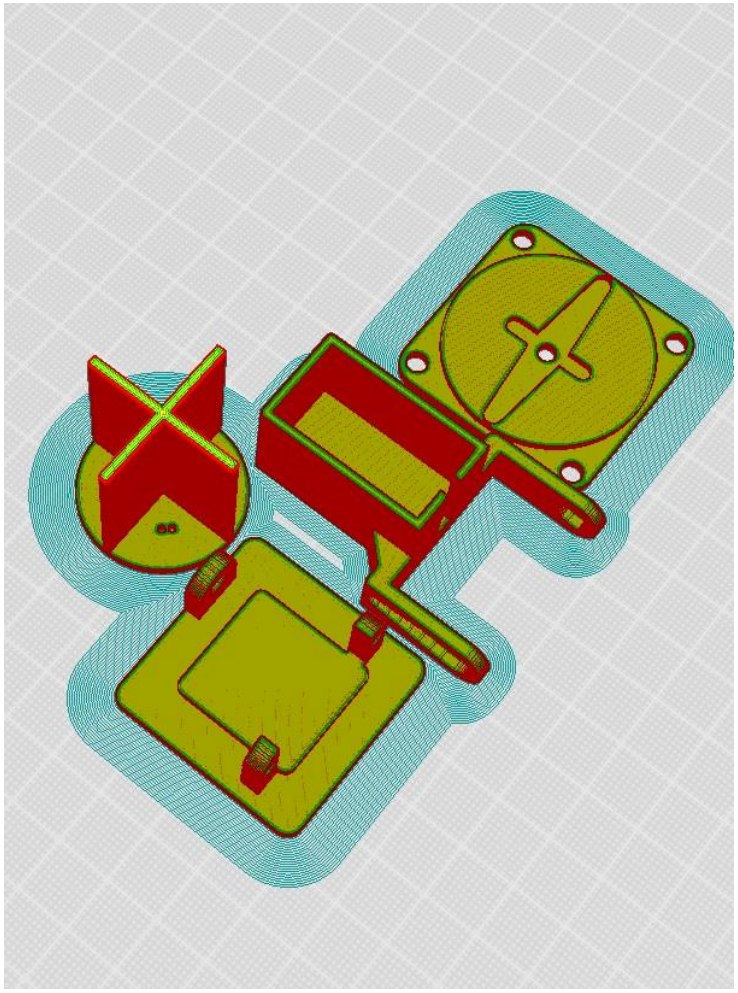


Figure 28 – Mechanical Parts Before Printing

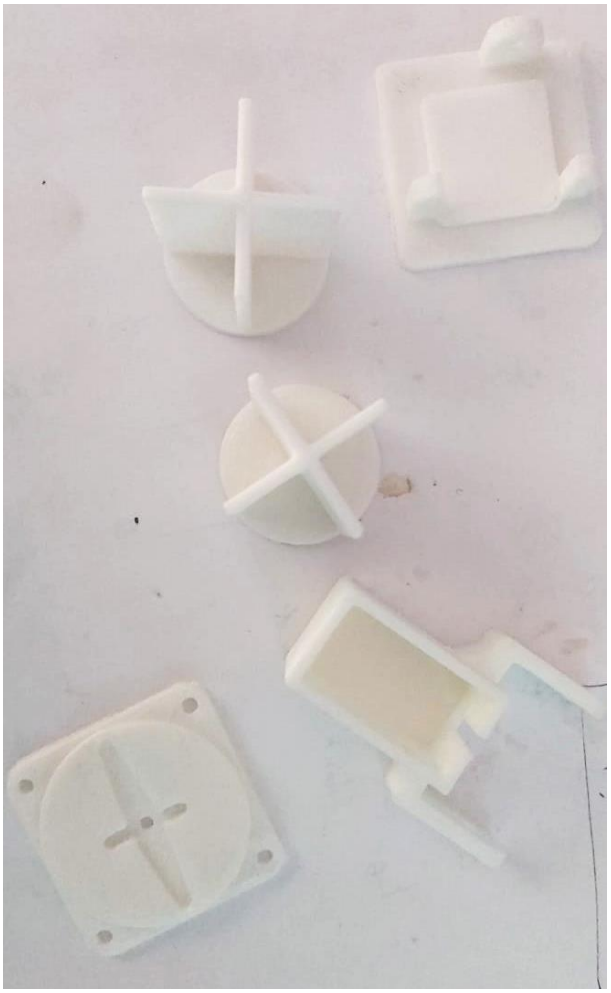


Figure 29 – Mechanical Parts After Printing

CHAPTER (4)
HARDWARE COMPONENTS & PCB FABRICATION

4.1 IMPORTANCE OF COMPONENTS SELECTION

The hardware selection in a dual axis solar tracker project is crucial to its overall performance and efficiency. It includes components such as motors, sensors, gears and transmission, control system and structural components. These components work together to move the solar panels and track the sun's movement, ensuring maximum energy production. The selection of these components is a balance between cost, performance and reliability. Careful consideration must be given to the design requirements and environmental conditions to ensure the project operates efficiently and reliably over its lifetime.

4.2 SENSORS

4.2.1 LIGHT DEPENDENT RESISTANCE (LDR)

4.2.1.1 LDR SPECIFICATIONS

The Light Dependent Resistor (LDR) is a critical component in the design and functionality of our dual axis solar tracker project. LDRs, also known as photo resistors, photocells or photoconductors, are electronic devices that exhibit a change in resistance in response to changes in light intensity. These devices are based on the principle of photoconductivity, which is the inverse relationship between light and resistance. As the light intensity on the LDR increases, the resistance of the LDR decreases. By measuring the resistance of the LDRs, we are able to detect the direction of the light source and adjust the position of the solar panel accordingly to maximize energy production.

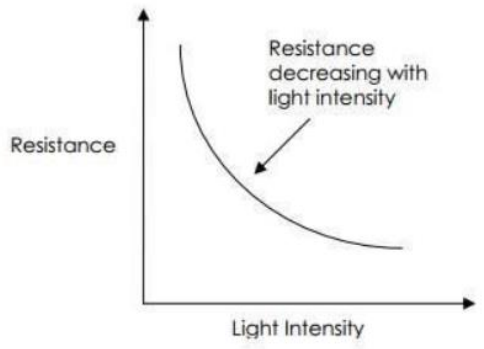


Figure 30 - Typical LDR Resistance vs Light Intensity Graph

In our project, we have chosen to use 10mm LDRs for their suitable size and compatibility with our mechanical components. This size is convenient for our project since it can fit well with the mechanical parts size. Additionally, we have employed the use of four LDRs,

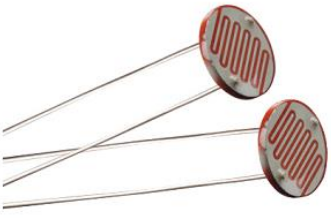


Figure 31 – 10 mm LDRs

one for each direction, to ensure comprehensive coverage and precise tracking of the sun's movement. This will help us to detect the direction of the light from all angles and move the solar panel towards the light to make the most of the energy produced by the sun. Using multiple LDRs will increase the accuracy and reliability of the system.

4.2.1.2 LDR CONNECTIONS & CONTROL

LDR have 2 pins, using voltage divider circuit we connect 1 pin of each LDR to the power (5v), another pin is connected to fixed resistance 100KΩ or 10KΩ and another pin of the resistance is connected to Ground part of the circuit.

LDRs is connected to Arduino using the connections between LDRs and fixed resistors, we connect LDRs to analog pins of the Arduino can read the value of the light of each LDR and deal with it using servo motors and other parts of our project.

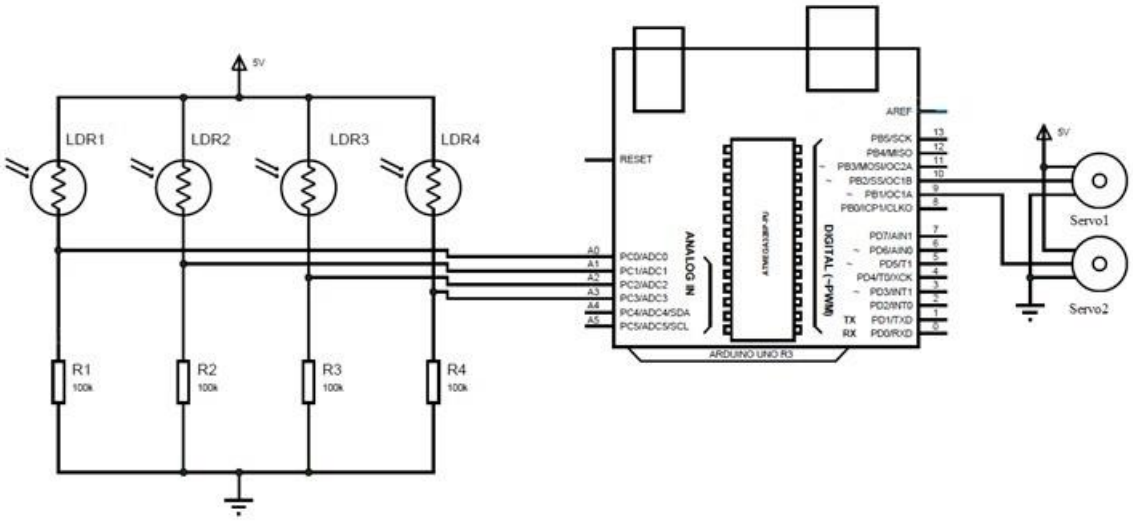


Figure 32 – LDRs Connections with Arduino

4.2.2 TEMPERATURE SENSOR (LM35)

In the solar system it's important to know the weather state so it was important to us to use temperature sensor to measure the temperature, we choose LM35 temperature sensor because it is easy to use and due to the resolution of this sensor.

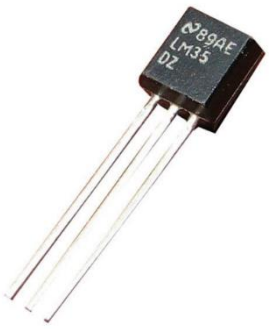


Figure 33 – Temperature Sensor (LM35)

The sensor consists of 3 pins which we need to connect it to can deal with, the first pin is the power pin of the sensor which needs

to be connected to 5v DC, the Second pin of the sensor is the signal pin which we use it to connect to the Arduino, we connect this pin to an analog pin of the micro controller, the last pin of the sensor is Ground pin which we connect it to the ground of our circuit.

The principle of working of this sensor depends on the temperature, which is be measured by it, the output of the sensor is 10mV/˚C which we can indicate with it the temperature of the environment which the sensor is put in, if the output volt is 350mV that means the temperature is 35˚C.

4.2.3 RAIN SENSOR

4.2.3.1 RAIN SENSOR SPECIFICATIONS

Raindrop sensor is basically a board on which nickel is coated in the form of lines. Rain sensor module allows to measure moisture via analog output pins, and it provides a digital output when a certain threshold is exceeded, it can also provide an analog output depending on the values selected in the code.

Figure 34 – Rain Module

The module includes a PCB that "collects" the rain drops. As raindrops are collected on the circuit board, they create paths of parallel resistance. The sensor is a resistive dipole that shows less resistance when wet and more resistance when dry. When there is no rain drop on board it increases the resistance, so we get high voltage. When rain drop present, it reduces the resistance because water is a conductor of electricity and presence of water connects nickel lines in parallel so reduces resistance and reduces voltage drop across it.

Figure 35 – Rain Module Pin Configuration (Part 1)

The sensor consists of two parts the first part has 4 pins, first pin is power pin(5v), the second and the third pin is signal pins which should be connected to the Arduino we choose to connect the sensor to an analog pin, the fourth pin is the ground pin which is be connected to the ground of the circuit, it has also 2 pins to connect the second part.

Figure 36 - *Rain Module Pin Configuration (Part 2)*

The second part of the sensor is the part which detect the water and the rain, it is connected to the first part with 2 pins.

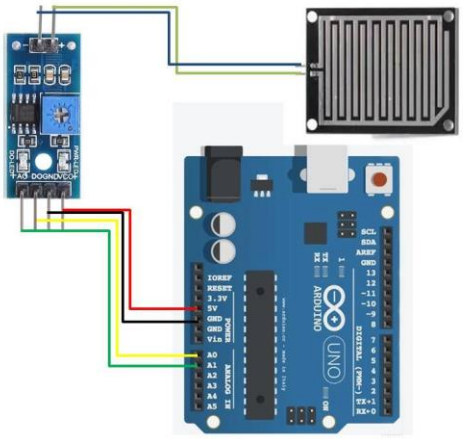


Figure 37 – Rain Module Connected to Arduino

4.2.3.2 WHY USING RAIN SENSOR?

In the usual solar tracker projects, the raindrop sensor isn't used, but in our model of the DAST (Dual-Axis Solar Tracker), we intend to use it for many good reasons, one most important reason of all is to help clean the solar panel in the times of rain from dirt and dust by adjusting the panel tilting angle based on the presence of rain, making it more inclined to let the rain wash away the dirt easier.

Figure 38 – Dusty Solar Tracker

Solar panels can still operate in the rain, but their power output depends on cloud coverage. Heavy rain clouds will most likely hinder energy production, but rainfall provides a safe and easy way to clean solar panels. Rainfall can rinse solar panel surfaces, preventing layers of dirt and debris from forming and blocking future sunlight. Solar power system may store less energy when it rains but it will clean the panel surface and ease the maintenance task. These losses resulting from snow, dirt, dust and other particles covering the solar panel are called soiling losses.

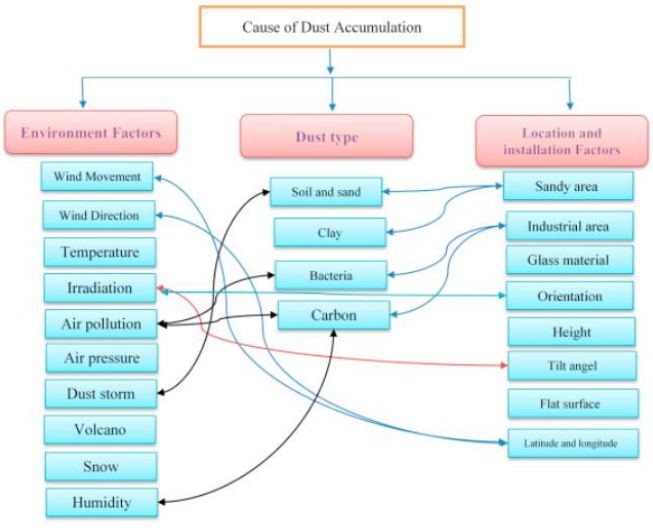


Figure 39 - Causes of Dust Accumulation

Dust is a thin layer that covers the surface of the solar array, and the typical dust particles are less than 10 µm in diameter but this depends on the location and its environment. Dust is generated from many sources such as pollution by wind, pedestrian volcanic eruptions, and vehicular movements among many others. The accumulated dust over time aggravates the soiling effect. In fact, the amount of accumulated dust on the

surface of the PV module affects the overall energy delivered from the PV module on a daily, monthly, seasonal and annual basis.

In Figure 40, the pattern of dust distribution in different parts of the world is assessed and it was found that the Middle East and North Africa have the worst dust accumulation zones in the world, so the concept of using rain to clean the surface of solar panels must be considered as one of our priorities.

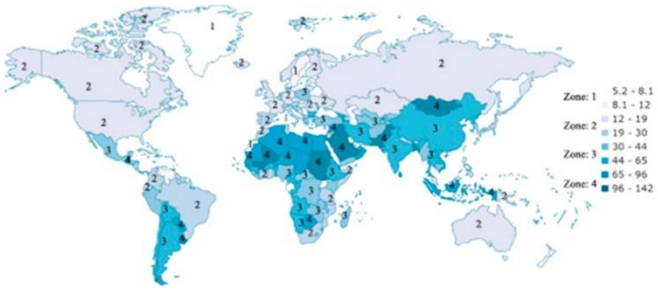


Figure 40 - Dust Distribution over the World

4.3 MOTORS

4.3.1 SERVO MOTOR

4.3.1.1 SERVO MOTOR SPECIFICATIONS

In our project, we utilized Servo motors to move the solar panel towards the light source. Servo motors are a type of actuator that is commonly used in robotics and automation applications. They are capable of precise control of angular position, velocity, and acceleration. Initially, we considered using Mini Servo Motors for this task, however, due to the weight of the solar panel, we ultimately decided to use Metal Gears Servo Motors (MG90S).

Figure 41 – MG90S Servo Motor

The MG90S motors have dimensions of 22mm x 12mm x 29mm, making them suitable for our project. Additionally, they have a torque rating of 1.8kg/cm when connected to a 4.8v power source and 2.2kg/cm when connected to 6v or higher. This increased torque capability allows for easy movement of the solar panel. Furthermore, the MG90S motors are micro servo motors which makes them a perfect fit for our project in terms of size and weight.

4.3.1.2 SERVO MOTOR CONTROL

Servo motors typically consist of three pins, including a power pin, a signal pin, and a ground pin. In our project, the power pin of the servo motors was connected to the 5v power source of the circuit, which was derived from a 12v battery source. The signal pin was connected to the Arduino, allowing for the transmission of control signals to the motor. The ground pin was connected to the ground of the circuit.

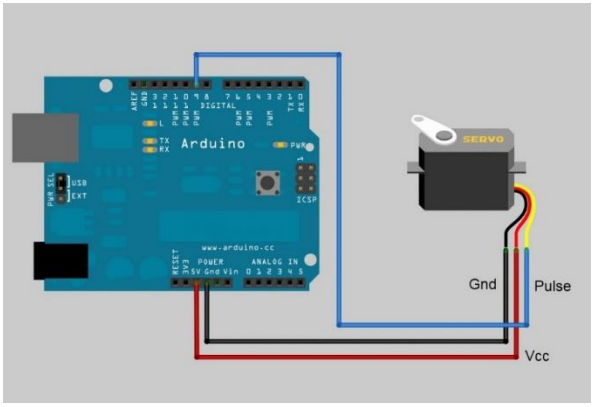


Figure 42 – Servo Motor Connected to Arduino

To control the servo motors, the signal pin is connected to a PWM (pulse-width modulation) pin on the Arduino. PWM pins on the Arduino allow for precise control of the servo motors' angular position, velocity, and acceleration. However, it's important to note that not all pins on the Arduino are PWM pins, so it's necessary to be mindful of the connections when working with servo motors and the Arduino to ensure proper functionality.

4.4 POWER

4.4.1 BATTERIES

The power source for our circuit is a 12v DC battery system, which is comprised of three lithium batteries connected in series, each with a voltage rating of 3.7v. These batteries were chosen for their stability, consistent current and voltage output. The power from the batteries is directed through a switch, which serves as an on/off mechanism for the circuit and project as a whole. The power then passes through a fuse, which is an electronic component designed to protect the circuit from damage caused by high current.

Figure 43 – 12v Batteries (Each of 3.7v)

Figure 44 – Battery Holder

4.4.2 REGULATORS

The power that passes through the fuse is then regulated using a L7809 voltage regulator, which reduces the voltage from 12v to 9v, in order to match the input voltage requirement of the Arduino. Another track of the power is further regulated using a L7805 voltage regulator, which reduces the voltage from 9v to 5v, in order to match the input voltage requirements of the LCD, sensors, and motors used in the project. The tracks through which the power passes on the PCB are designed to be 1.8mm wide, allowing for efficient current flow and minimizing the risk of problems related to high current. It's important to note that the current increases when the servo motors are in use, as they require power to move the solar panel.

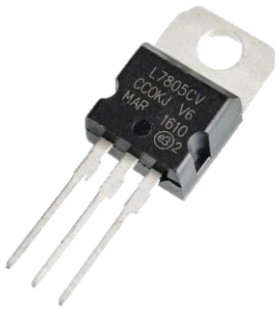


Figure 45 – L7805 Voltage Regulator

4.5 MICROCONTROLLER

The microcontroller is a crucial component in our project, serving as the central processing unit that controls and coordinates all the other subsystems. We have selected the Arduino Nano as the microcontroller for our project due to its suitability and cost-effectiveness.

The Arduino Nano has a total of 14 digital pins and 8 analog pins, which are designated as D0 to D13 and A0 to A7 respectively. These pins can be configured as either input or output, depending on the requirements of the project. The Arduino Nano is a suitable microcontroller for our project as it has a sufficient number of pins for our needs.

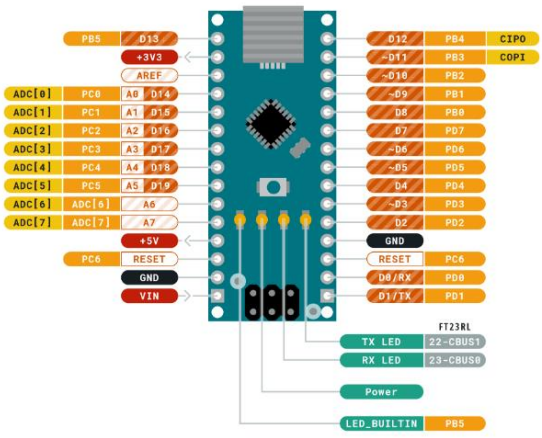


Figure 46 – Arduino Nano Pin Configuration

It's important to note that the Arduino Nano requires a 9v power input voltage. Therefore, we have used a voltage regulator to reduce the voltage from 12v, which is the output of our

battery system, to 9v to ensure that the microchip of the Arduino is protected from any damage that could be caused by an excessive voltage input.

4.6 PRINTED CIRCUIT BOARD (PCB)

4.6.1 PCB LAYOUT DESIGN

In this project, we decided to use a Printed Circuit Board (PCB) for the electronics circuit of the project. The PCB is a board that contains the electronic components and the copper tracks that connect these components together. The design of the PCB required specialized software, and we chose to use the Eagle program for PCB design.

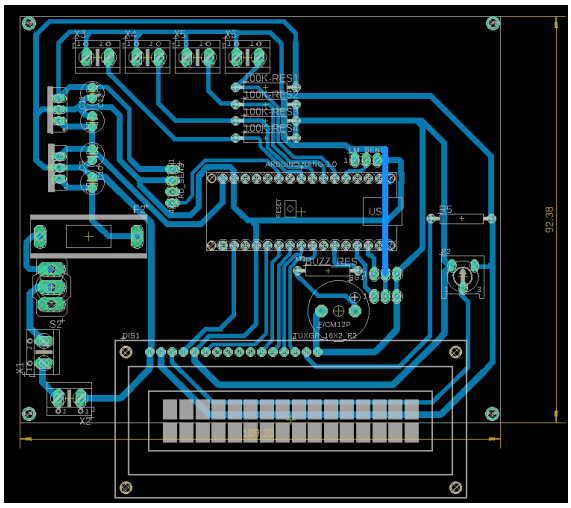


Figure 47 – PCB Layout

Designing a PCB is a multi-step process that involves two main phases: schematic design and board design. The schematic design phase is where we choose the components and make the connections between them using the software. This phase is crucial as it involves understanding the circuit and how the components interact with one another.

Once the schematic design is complete, the board design phase begins. This phase involves choosing the appropriate footprint for the real components, designing the holes for those components, and designing the copper tracks with the appropriate width. It's important to have a good understanding of electronics circuit design and PCB design, as well as proficiency in the chosen software, to complete this phase.

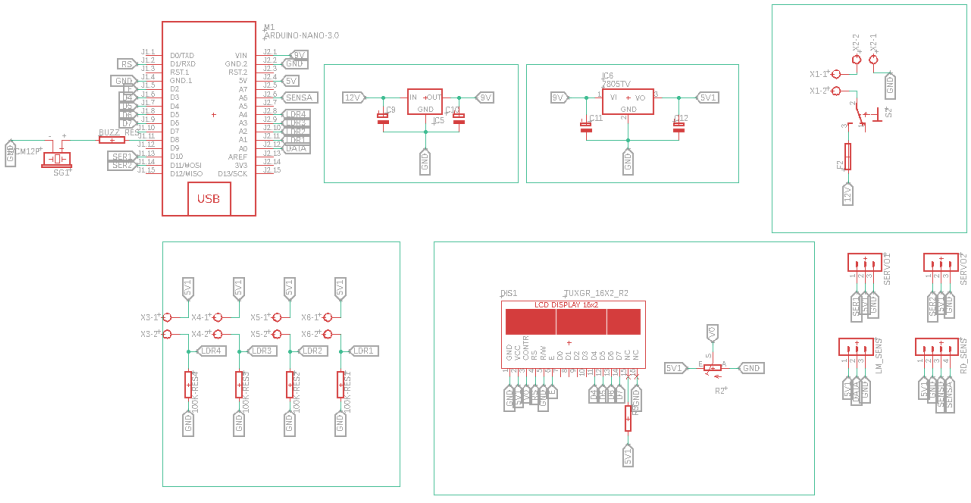


Figure 48 – Schematic Design

Once the board design is complete, we exported the

design to a PDF format, which we then used to print the design onto a PCB fiber board. This allowed us to fabricate the circuit and complete the project.

4.6.2 PCB FABRICATION

The fabrication of a printed circuit board (PCB) involves several steps, beginning with the printing of the design on glossy paper, which is then used to transfer the design onto a fiber-copper board. The first step of the fabrication process is to align the printed paper with the fiber-copper board and heat it using a hot iron for approximately 15 minutes. This process is known as "photo-resist transfer," and it transfers the ink tracks from the paper onto the board.

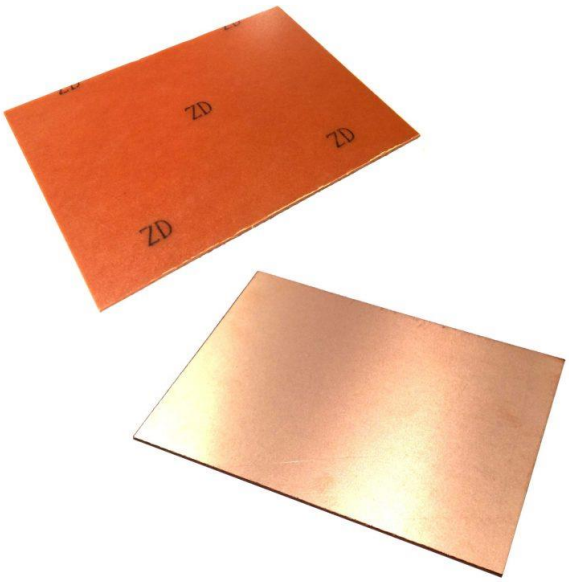


Figure 49 – Copper Board

Next, the board is placed in an acid bath for around 5 minutes, which etches away the unneeded copper and leaves behind the desired circuit traces. The second step of the fabrication process is drilling the holes that will be used to mount the electronic components. This is done using a stable drill and suitable bits for the holes.

After drilling, the components are placed in their designated positions and soldered onto the board using a soldering iron and tin. The next step is to test the circuit using an Avo meter to ensure that there are no shorts between power and ground, and that all the components are functioning properly.

Finally, the circuit is powered up and connected to the necessary wires, completing the hardware aspect of the PCB fabrication process. It is essential to review and test the circuit thoroughly before using it to prevent any damage or malfunction.

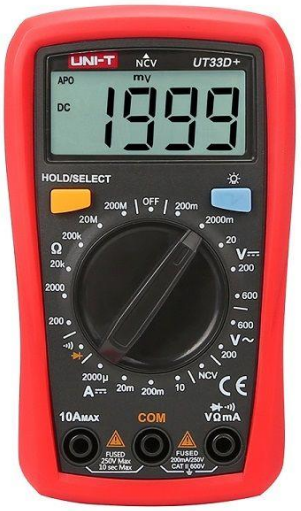


Figure 50 – AVO Meter

4.6.3 PCB FABRICATION PROCESS

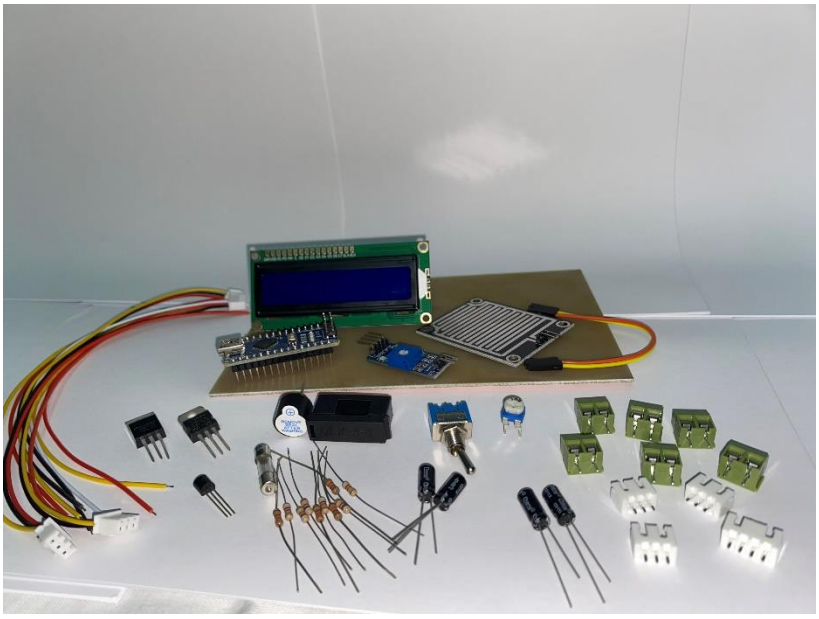


Figure 52 – Components Purchased

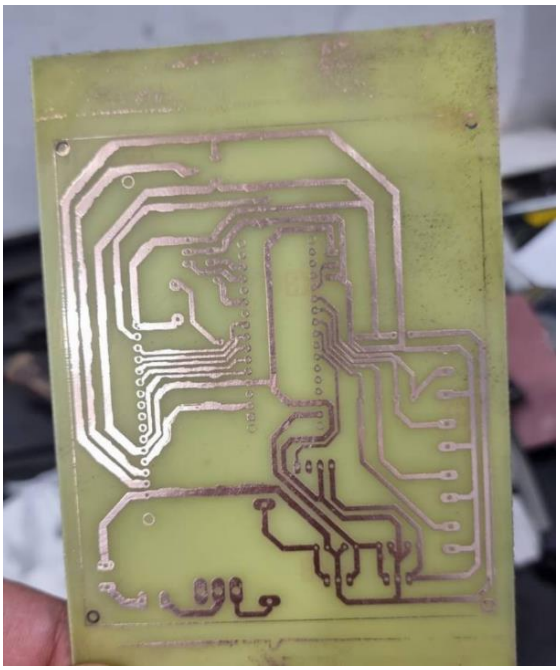


Figure 51 – Printed Layout

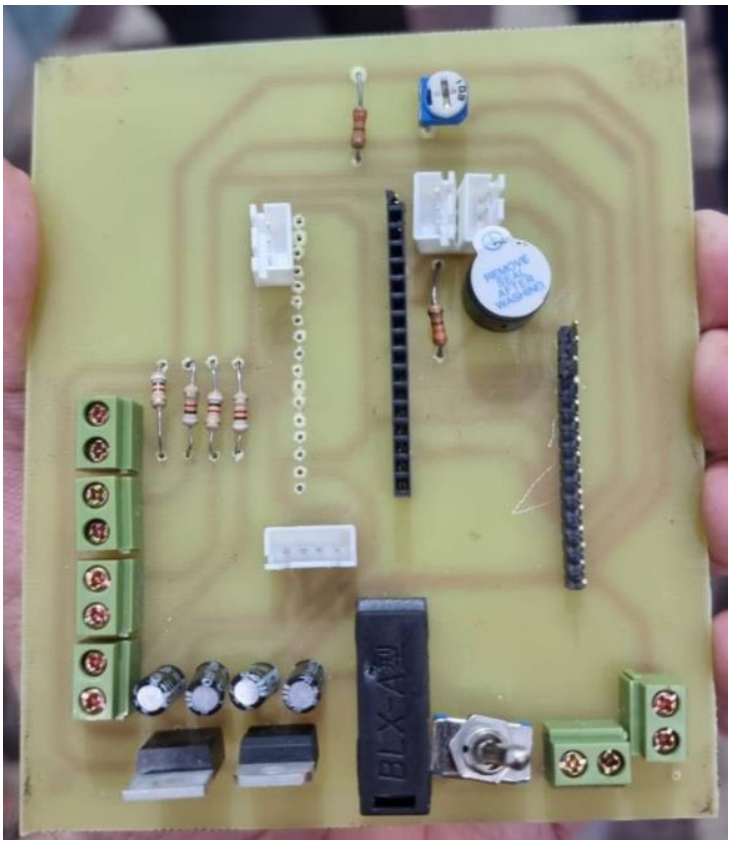


Figure 53 – PCB with Some Components Installed

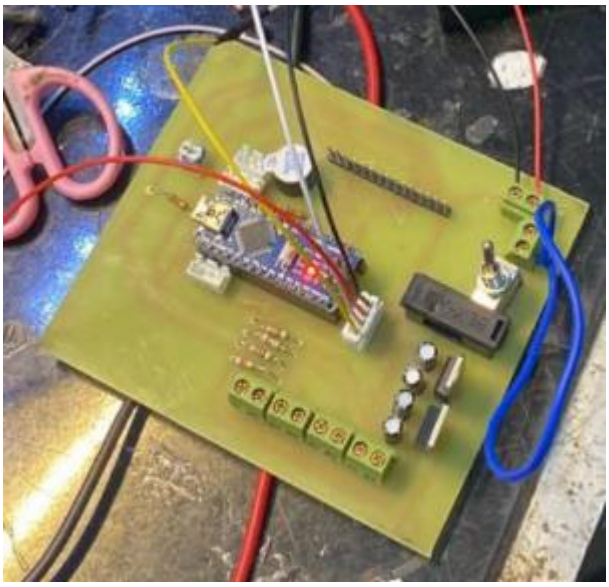


Figure 54 – Final PCB

CHAPTER (5)
PROGRAMMING ,CODE, AND SIMULATION

5.1 WORKING ENVIRONMENT

5.1.1 ARDUINO IDE

The Arduino IDE (Integrated Development Environment) is a software application that allows users to write, upload and debug code for the Arduino platform. It is an open-source tool that supports a wide range of microcontroller boards, including the popular Arduino Uno and Arduino Mega. The IDE provides a simple and user-friendly interface for writing code in the Arduino programming language, which is based on C/C++. It also includes a variety of built-in libraries and example sketches to help users get started with their projects.

Figure 55 – Arduino IDE Interface

However, the Arduino IDE will not be our main platform for developing our code for multiple reasons that we are going to discuss, nonetheless it is crucial in our development process as we need to have it installed while we work on other editors and IDEs.

The Arduino IDE presents a number of benefits, including its user-friendly interface and accessibility for individuals with varying skill levels. Additionally, it boasts a robust community and a wide array of resources, including tutorials and examples. However, in this discussion, we will delve into the reasons for not utilizing the Arduino IDE in our project.

5.1.1.1 WHY THE ARDUINO IDE MAY NOT BE THE BEST OPTION:

- Limited debugging capabilities: While the IDE provides basic debugging features such as serial monitor and breakpoints, it may not be as powerful as other IDEs for more advanced debugging tasks.
- Limited code organization: The Arduino IDE does not have advanced features for organizing and structuring code, which can make it difficult to maintain large and complex projects.
- No built-in version control: The Arduino IDE does not have built-in support for version control systems such as Git, which can make it difficult to collaborate on projects or manage code revisions.
- Limited code highlighting: The Arduino IDE does not offer advanced code highlighting or syntax checking, which can make it more difficult to read and write code.
- Limited code completion: The Arduino IDE does not have advanced code completion or autocomplete features, which can make it more difficult to write code quickly and efficiently.
- Limited project management: The Arduino IDE does not have advanced project management features such as task tracking or bug reporting, which can make it more difficult to manage large and complex projects.
- Limited community support: While the Arduino platform has a large community of users and developers, the Arduino IDE may not have as much community support as other IDEs.
- Limited customizability: The Arduino IDE may not be as customizable as other IDEs, which can make it more difficult to tailor the development environment to your specific needs.
- Limited testing capabilities: The Arduino IDE does not have advanced testing capabilities such as unit testing or integration testing, which can make it more difficult to ensure that code is working correctly.

- Limited code analysis: The Arduino IDE does not have advanced code analysis features such as static code analysis or code metrics, which can make it more difficult to identify and fix issues in your code.
- Limited code generation: The Arduino IDE does not have advanced code generation capabilities which can make it more difficult to automate repetitive tasks.
- Limited code navigation: The Arduino IDE does not have advanced code navigation capabilities which can make it more difficult to navigate through large and complex code bases.

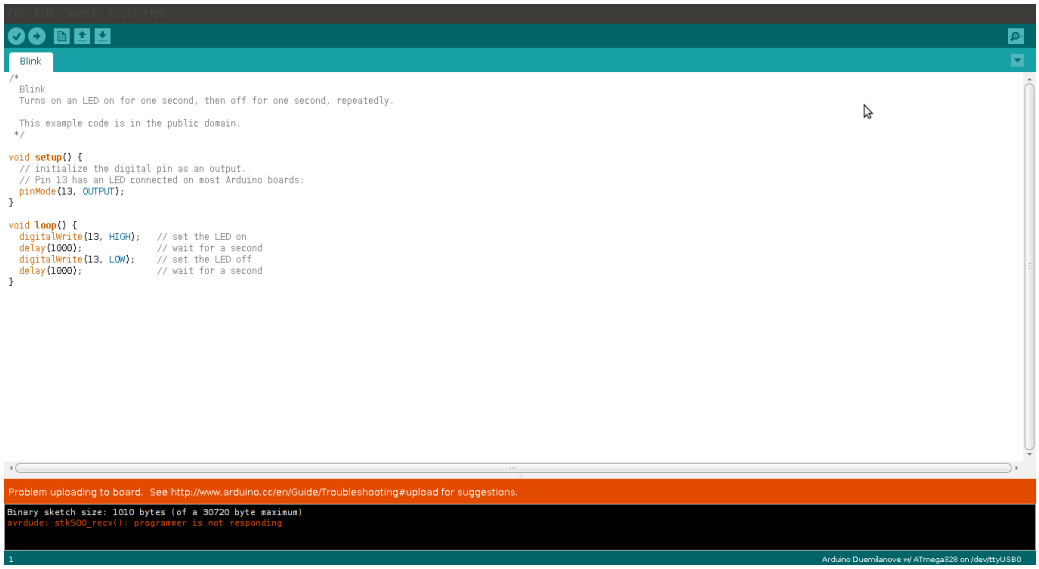


Figure 56 – Limited Interface of Arduino IDE

One of the key factors that influenced our decision to not use the Arduino IDE was the lack of a built-in **dark mode feature**.

5.1.1.2 CODING IN THE DARK: PROGRAMMER PREFERENCE REVEALED

The absence of this feature can make prolonged use of the Arduino IDE difficult on the eyes, especially for those who work in dimly lit environments or for extended periods of time. Additionally, a dark mode feature can also reduce strain on the eyes and improve the overall user experience. The lack of

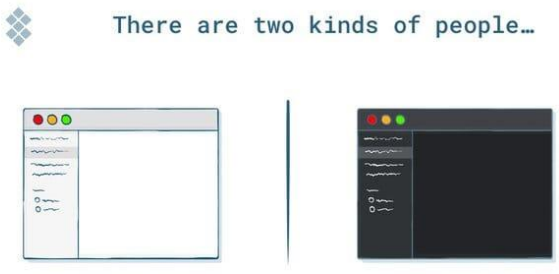


Figure 57 – Light Mode & Dark Mode IDEs

this feature was a significant concern for our team working on the **Dual Axis Solar Tracker Project** and ultimately led to our decision to explore alternative development environments.

A recent poll conducted among programmers revealed that 63% of respondents preferred using dark mode for coding, while 37% preferred using light mode. This suggests that the majority of programmers find dark mode to be more comfortable and easier on the eyes when working for prolonged periods of time. The poll was distributed to a diverse group of programmers from various backgrounds and experience levels. This poll indicates that there is a clear preference among programmers for dark mode, but it also highlights the importance of personal preference and individual work style in determining the best coding environment.

Figure 58 – Programmers Preferring Light Mode vs Dark Mode

5.1.2 VISUAL STUDIO CODE

Microsoft Visual Studio Code (Vs Code) is a widely used code editor that has several advantageous features. One of its main benefits is the flexibility it offers in terms of customization, enabling developers to adjust the environment according to their requirements. Furthermore, it features a built-in dark mode, which is known to reduce strain on the eyes during prolonged coding sessions. Additionally, Vs Code offers support for a variety of programming languages and has an active community that provides ample resources such as tutorials and examples. It also offers a vast selection of plugins and extensions to enhance its functionality. And finally, it is an open-source software that is available on multiple platforms, making it easily accessible to a wide

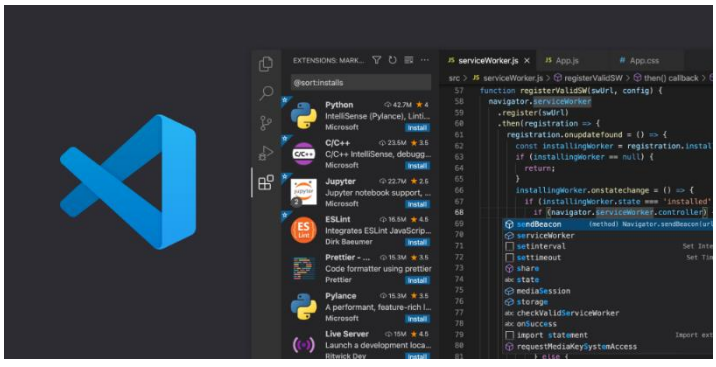


Figure 59 – VS Code Program

range of users. All these features make Microsoft Visual Studio Code a great choice for programmers seeking a flexible and customizable code editor.

VS Code has almost all of our needs and requirements that we are going to need in a large coding project like ours as you can see in the figure below.

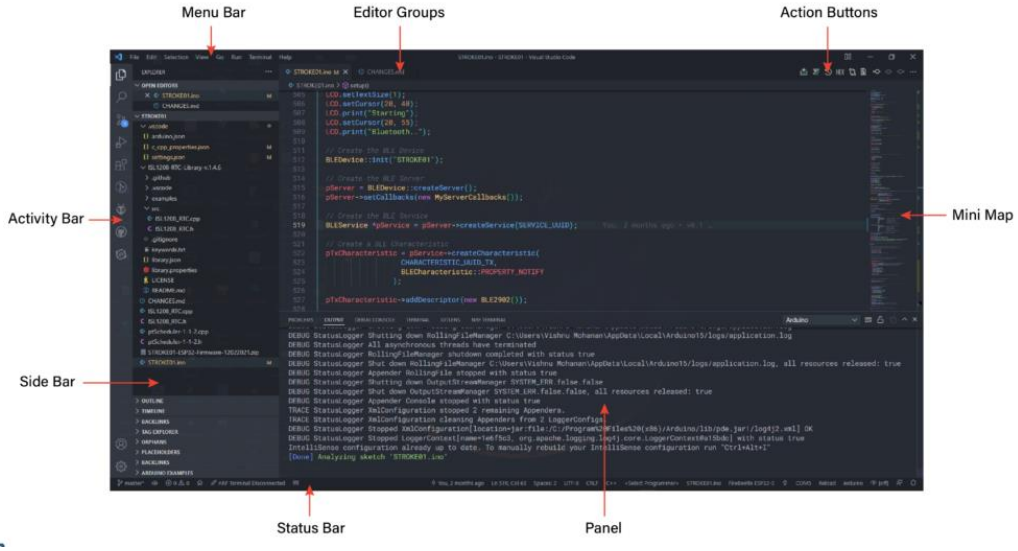


Figure 60 – VS Code Interface

5.1.2.1 USING VSCODE TO WRITE CODE FOR ARDUINO

Installing the Arduino extension on Microsoft Visual Studio Code (Vs Code) is a simple process that can greatly enhance the functionality of the code editor for programming Arduino boards. The extension allows for an integrated development environment (IDE) within Vs Code, providing a seamless workflow for editing, compiling, and uploading code to Arduino boards.

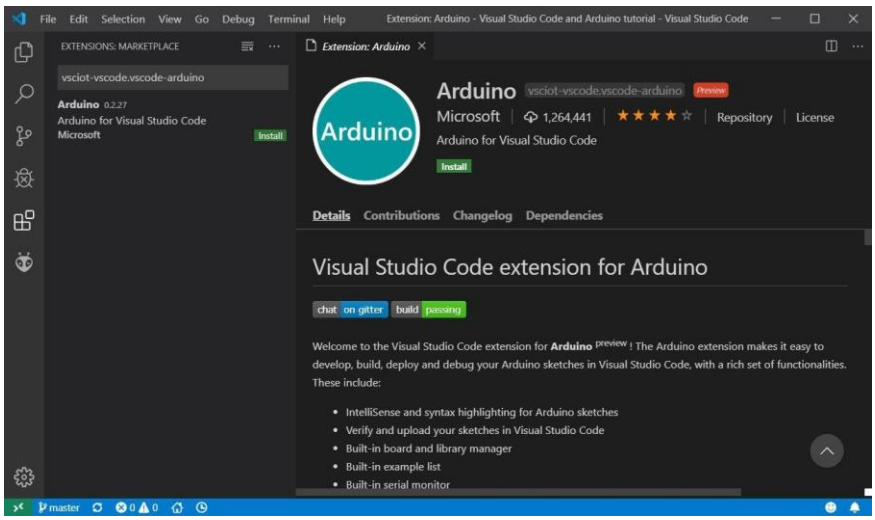


Figure 61 – VS Code Arduino Extension

The Arduino extension allows for the creation of new sketches, or projects, and includes a built-in code editor with syntax highlighting and autocomplete for Arduino-specific

commands. The extension also includes a built-in serial monitor for monitoring serial communication with the Arduino board, and a built-in library manager for managing and installing libraries.

One of the key features of the Arduino extension is its support for a wide range of boards, including the most popular Arduino boards such as the Arduino Uno, Mega, and Nano, as well as boards from other manufacturers like Adafruit and SparkFun. This makes the extension a versatile tool for programming a wide range of devices.

In conclusion, the Arduino extension for Microsoft Visual Studio Code is a powerful and versatile tool for programming Arduino boards. Its wide range of features, including the built-in code editor, serial monitor, library manager, and support for a wide range of boards, make it a valuable addition to the toolkit of any Arduino developer. The installation process is straightforward, and the extension can be easily enabled once installed. This extension would be a valuable addition for any developers using Vs code for their Arduino projects.

5.2 THE PROCESS OF CONCEPTUALIZING THE PROJECT LOGIC

5.2.1 GETTING FAMILIAR WITH THE MECHNICAL COMPONENTS AND THEIR INTERACTIONS

Getting familiar with the mechanical components and their interactions is a crucial step in the process of writing the project code for the dual axis solar tracker. The mechanical design and assembly of the solar tracker determine how the servos are placed and how they need to be calibrated. Additionally, understanding the mechanical interactions and dependencies between the various components is essential for developing accurate and reliable code.

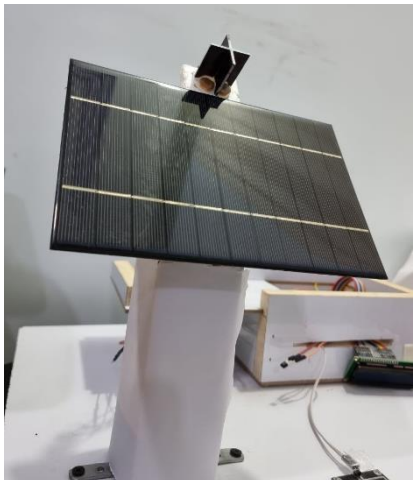


Figure 62 – Shot from The Project

To begin, I studied the mechanical design and assembly of the solar tracker. I examined the different components, such as the solar panel, the motor, and the servos, and how they were assembled together. This helped me understand how the mechanical components interacted with each other and how the servos were placed in relation to the other components.

Next, I analyzed the servo placement and calibration. Servos are responsible for moving the solar panel to track the sun and it's essential to understand how they are placed and calibrated to ensure accurate solar panel alignment. I examined the servo's position, orientation, and movement range, and determined how they needed to be calibrated to ensure optimal performance.

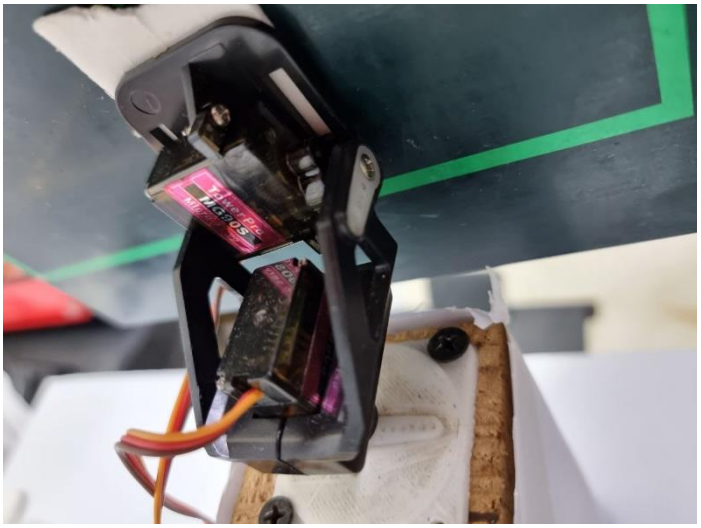


Figure 63 – Two Servos Placements

5.2.2 DETERMINING LDR SENSORS' PLACEMENT AND CONFIGURATION

The placement and orientation of the LDR sensors are critical factors in ensuring accurate tracking and measurements of light intensity so that the servos can make precise movements. It is important to consider factors such as the angle of incidence of sunlight, the sensor's field of view, and the surrounding environment when determining the placement and orientation of the sensors.

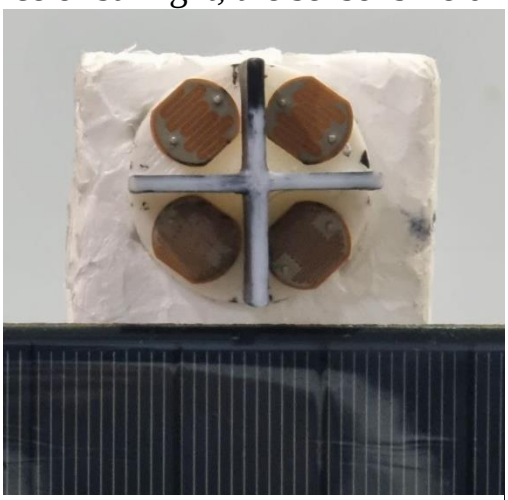


Figure 64 – Cross-Shaped 3D Printed Piece

In our dual axis solar tracker project, we took a different approach in the placement and orientation of the Light Dependent Resistor (LDR) sensors. Instead of placing them at the corners of the solar tracker, we created a **custom 3D printed piece** in the shape of a cross that was placed at the

top center of the solar panel. This cross-shaped piece housed all four LDR sensors, allowing for a more centralized and efficient measurement of sunlight.

We also made sure that the LDRs are placed in a way to avoid any light leakage from the sides. We did this by **painting the sides of the 3D printed cross-shaped piece in black**, which helped to absorb any stray light that might have otherwise affected the sensor readings.

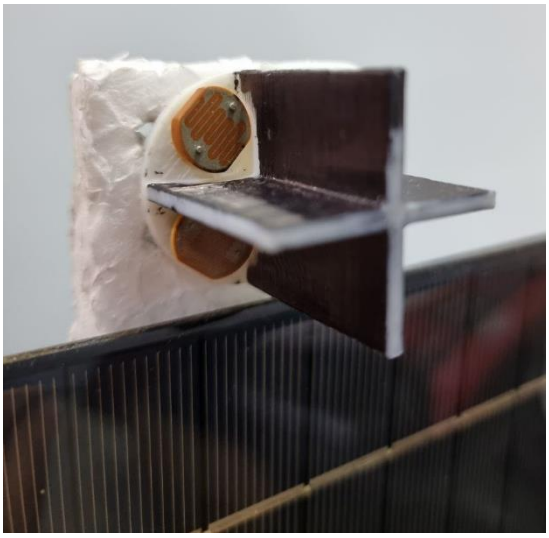


Figure 65 – Black Sides of Cross-Shaped Piece

We also marked the cross-shaped 3D printed piece from the top so that we can accurately identify which side will be facing up.

5.2.3 GENERAL DUAL AXIS TRACKING LOGIC

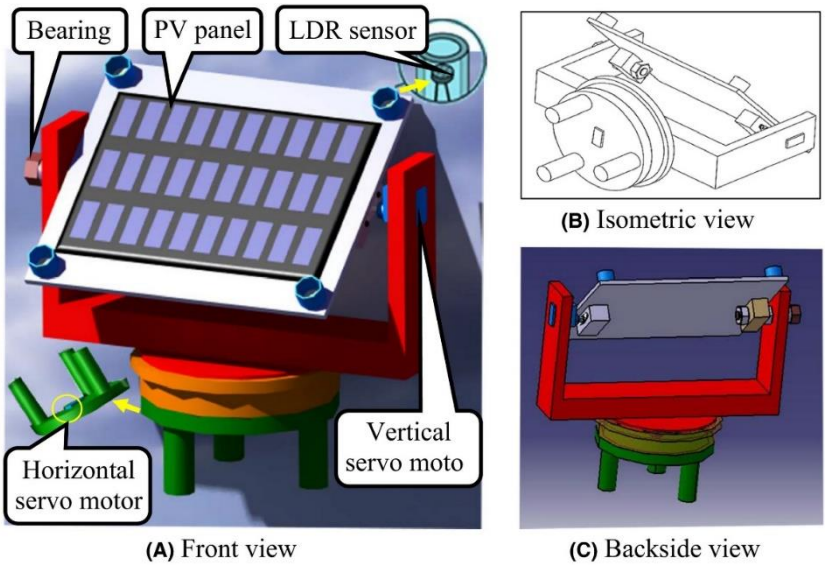


Figure 66 – General Dual Axis Configuration

The general dual axis tracking logic is a crucial aspect of our innovative dual axis solar tracker project. This logic involves a series of steps that work together to ensure accurate and efficient

tracking of the sun's movement. The following is an overview of the general dual axis tracking logic:

1- The first step in the general dual axis tracking logic is to measure the values of the 4 LDR (light dependent resistor) sensors. These sensors are strategically placed and calibrated to accurately measure the light intensity of the sun and determine the direction of the sun's movement.

2- Next, we compare each opposite pair of sensor readings to determine where the value of the light intensity is greater. This is done to ensure that the solar panel is aligned in the direction of the sun and is receiving the maximum amount of sunlight.

3- By comparing the **Top Left** sensor to the **Top Right** sensor, we are able to determine whether the horizontal servo needs to move to the left or the right. The same process is applied to the **Bottom Left** and **Bottom Right** sensors. This allows for precise horizontal tracking of the sun's movement.

4- Lastly, by comparing the **Top Left** sensor to the **Bottom Left** sensor, we are able to determine whether the vertical servo needs to move up or down. The same process is applied to the **Top Right** and **Bottom Right** sensors. This allows for precise vertical tracking of the sun's movement.

Figure 67 – Range of Movement of V/H Servos

The following pictures explains previous points.

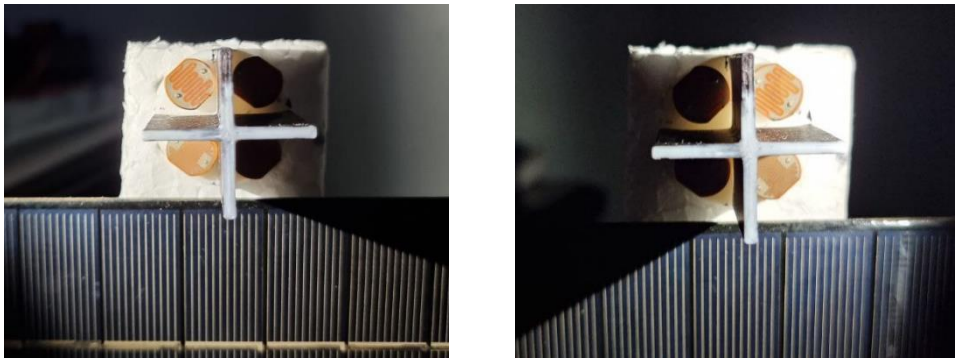


Figure 68 – Point Number 3 Explanation

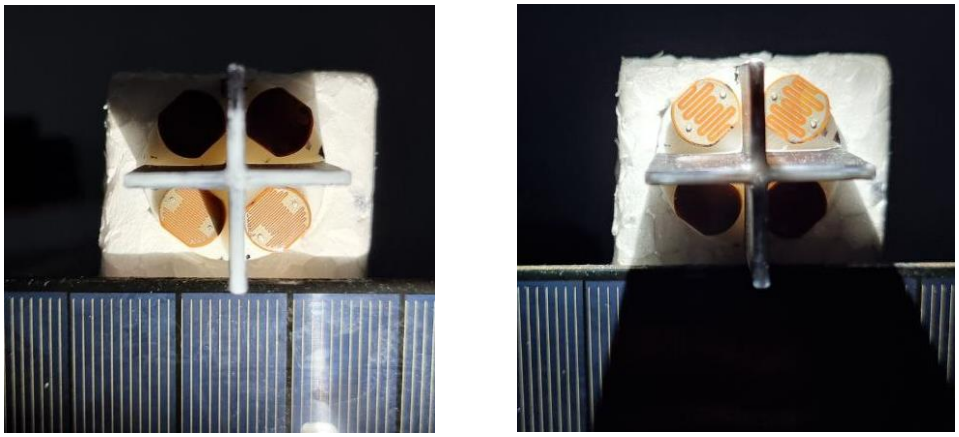


Figure 69 – Point Number 4 Explanation

By following these steps, our dual axis solar tracker is able to accurately track the sun's movement throughout the day and ensure that the solar panel is receiving the maximum amount of sunlight. Additionally, our innovative design also includes enhanced weather monitoring capabilities, which further improves the efficiency and performance of the solar tracker.

5.3 THE PROJECT CODE

```
/*******************************************************************************
 * Project: Dual Axis Solar Tracker
 *
 * Written By: Hany Elesawy
 *
 * Version: v6.0
 *
 * Version History:
 * v1.0 Added Quad LDR                                  - 2022.12.14
 * v1.1 Added Dual Axis Control                         - 2022.12.15
 * v1.2 Fixed a bug in rotation direction               - 2023.01.14
 *
 * v2.0 changed the tracking algorithm (removed average)    - 2023.01.15
 *      surprisingly this improved tracking drastically
 *
 * v3.0 Added RainDrop Sensor Support                   - 2023.01.19
 * V3.1 Enabled a simple rain buzzer alarm (with delay) - 2023.01.19
 * v4.0 Sharp tracker angle on heavy rain (for cleaning)    - 2023.01.19
 *
 * v5.0 Added LCD Logic + Integration                   - 2023.01.26
 * v6.0 Removed most delays in favour of using Millis()     - 2023.01.27
 * v6.2 Added tracking_response_time                    - 2023.01.27
 *******************************************************************************/
```

```
/*-------------------------------------------------------------------------*/
/*------------------ Project Specific Behaviour Description ---------------*/
/*-------------------------------------------------------------------------*/
/*

    VERTICAL SERVO  : is the servo responsible for equalizing "avgLdrTop" and "avgLdrBot" readings
    HORIZONTAL SERVO: is the servo responsible for equalizing "avgLdrLeft" and "avgLdrRight" readings

    When the tracker is flat (looking up)            ---> Vertical Servo is with angle = 0
    When the tracker is looking down (to the ground) ---> Vertical Servo is with angle = 180

    When the tracker is looking to the left          ---> Horizontal Servo is with angle = 0
    When the tracker is looking to the Right         ---> Horizontal Servo is with angle = 180


    Ideally, we don't want the solar tracker looking down as there is nothing to be done there.
    If we don't limit it, sometimes it will take the easy path to equalize the avgLdrTop, avgLdrBot readings.
    When it looks down. the reading will be equalized.


*/
```

```
/*-------------------------------------------------------------------------------*/
/*------------------------------ Header Files -----------------------------------*/
/*-------------------------------------------------------------------------------*/

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

/*-------------------------------------------------------------------------------*/
/*------------------------------ Global -----------------------------------------*/
/*-------------------------------------------------------------------------------*/

Servo servo_horizontal;
Servo servo_vertical;

//Parameters:    (rs, enable, d4, d5, d6, d7)
LiquidCrystal lcd(0 , 2     , 3 , 4 , 5 , 6 );

/*----- Calibration and Angle Tweaks -----*/

int servoV_angle       = 0;              // Starting Vertical Angle (Flat)
int servoV_UpperLimit  = 105;            // Looking Down Limit
int servoV_LowerLimit  = 0;              // Looking Up Limit
int servoV_SharpAngle  = 80;            // Some Sharp Angle to wash solar panels by rain

int servoH_angle       = 0;              // Starting Horizontal Angle
int servoH_UpperLimit  = 180;            // Looking to the Right Limit
int servoH_LowerLimit  = 0;              // Looking to the Left Limit
int servoH_SharpAngle  = 115;            // Some Sharp Angle to wash solar panels by rain

/*------ Dual Axis Response Time ------*/
int tracking_response_time = 30;         // How fast should the whole project React to light changes.

/*--------------- Flags ---------------*/
bool isItRaining = false;
bool isItRainingHard = false;
```

```

/*----- Time Intervals of Events (ms) -----*/
#define interval_LCDPrint    200               // Time interval to print on LCD
#define interval_BuzzerBeep  500               // Time to buzzer High


unsigned long currentTime               =0;

unsigned long previousTime_LCDPrint     =0;     // When was the last time LCD printed something?
unsigned long previousTime_BuzzerHigh   =0;     // When was the last time Buzzer High?
unsigned long previousTime_BuzzerLow    =0;     // When was the last time Buzzer low?

/*---- Other Variables ----*/
float temperatureC_LM35;
float temperatureC = 19.68;

/*-------------------------------------------------------------------------------*/
/*------------------------------ Setup ------------------------------------------*/
/*-------------------------------------------------------------------------------*/
void setup()
{
    lcd.   begin(16,2);
    //Serial.begin(9600);

    /*----- Pin Configuration ----------------*/

    #define servTop 11
    #define servBot 10

    #define ldrTopL A1
    #define ldrTopR A2
    #define ldrBotL A3
    #define ldrBotR A4

    #define LM35     A0
    #define rainDrop A6
```

```
115
116     #define LM35     A0
117     #define rainDrop A6
118
119     #define buzzer 8
120
121
122
123     /*----- Pin Modes (Input / Output) ----------------*/
124
125     pinMode(ldrTopL , INPUT);
126     pinMode(ldrTopR , INPUT);
127     pinMode(ldrBotL , INPUT);
128     pinMode(ldrBotR , INPUT);
129
130     pinMode(rainDrop, INPUT);
131     pinMode(LM35    , INPUT);
132
133     pinMode(buzzer  , OUTPUT);
134
135     servo_horizontal.attach(servBot);
136     servo_vertical.attach(servTop);
137
138
139     /* --- Default Orientation ---*/
140     servo_horizontal.write(servoH_angle);
141     servo_vertical.write(servoV_angle);
142 }
143
144 void loop()
145 {
146
147     delay(tracking_response_time);
148
149
150     currentTime = millis();
```

```
151
152     /*------------- Analog Readings -------------*/
153
154     int ADC_ldrTopL  = analogRead(ldrTopL);
155     int ADC_ldrTopR  = analogRead(ldrTopR);
156     int ADC_ldrBotL  = analogRead(ldrBotL);
157     int ADC_ldrBotR  = analogRead(ldrBotR);
158
159     int ADC_LM35 = analogRead(LM35);
160     int ADC_RainDrop   = analogRead(rainDrop);
161
162     servoH_angle = servo_horizontal.read();
163     servoV_angle = servo_vertical.read();
164
165     float voltage =  ADC_LM35 * (5.0/1024.0);
166     temperatureC_LM35 = voltage * 100;
167     /*-----------------------------------------*/
```

```
168
169     if( RainStatus() == true)
170     {
171         AlarmBeepBeep();
172 
173         servoAllSharpAngle();
174     }
175     else
176     {
177         /*--- Vertical Servo Positioning ---*/
178         if (ADC_ldrTopL > ADC_ldrBotL)
179         {
180             servoRotateUp();
181         }
182         else if (ADC_ldrBotL > ADC_ldrTopL)
183         {
184             servoRotateDown();
185         }
186 
187         if (ADC_ldrTopR > ADC_ldrBotR)
188         {
189             servoRotateUp();
190         }
191         else if (ADC_ldrBotR > ADC_ldrTopR)
192         {
193             servoRotateDown();
194         }
195 
196         /*--- Horizontal Servo Positioning ---*/
197         if (ADC_ldrTopL > ADC_ldrTopR)
198         {
199             servoRotateLeft();
200         }
201         else if (ADC_ldrTopR > ADC_ldrTopL)
202         {
203             servoRotateRight();
204         }
205 
```

```
203             servoRotateRight();
204         }
205 
206         if (ADC_ldrBotL > ADC_ldrBotR)
207         {
208             servoRotateLeft();
209         }
210         else if (ADC_ldrBotR > ADC_ldrBotL)
211         {
212             servoRotateRight();
213         }
214 
215     }
216 }
217 
```

```
218
219 /*-------------------------------------------------------------------------------*/
220 /*----------------------- User Defined Functions --------------------------------*/
221 /*-------------------------------------------------------------------------------*/
222
223 bool RainStatus()
224 {
225     int ADC_RainDrop = analogRead(rainDrop);
226
227     if(ADC_RainDrop < 200)
228     {
229         if(currentTime - previousTime_LCDPrint > interval_LCDPrint )
230         {
231             lcd.clear();
232             lcd.setCursor(0,0);
233             lcd.print("Heavy Rain !!");
234
235             lcd.setCursor(0,1);
236             lcd.print("Temp: ");
237             lcd.print(temperatureC);
238             lcd.print("C");
239
240             //Serial.println("Heavy Rain !!");
241
242             previousTime_LCDPrint = currentTime;
243         }
244
245         isItRaining = true;
246         isItRainingHard = true;
247     }
248     else if(ADC_RainDrop < 300)
249     {
250         if(currentTime - previousTime_LCDPrint > interval_LCDPrint )
251         {
252             lcd.clear();
253             lcd.setCursor(0,0);
254             lcd.print("Moderate Rain !!");
255
256             lcd.setCursor(0,1);
257             lcd.print("Temp: ");
258             lcd.print(temperatureC);
259             lcd.print("C");
260
261             //Serial.println("Moderate Rain !!");
```

```
261             //Serial.println("Moderate Rain !!");
262
263             previousTime_LCDPrint = currentTime;
264         }
265
266         isItRaining = true;
267         isItRainingHard = true;
268     }
269     else if(ADC_RainDrop < 600)
270     {
271         if(currentTime - previousTime_LCDPrint > interval_LCDPrint )
272         {
273             lcd.clear();
274             lcd.setCursor(0,0);
275             lcd.print("Light Rain !!");
276
277             lcd.setCursor(0,1);
278             lcd.print("Temp: ");
279             lcd.print(temperatureC);
280             lcd.print("C");
281
282             //Serial.println("Light Rain !!");
283
284             previousTime_LCDPrint = currentTime;
285         }
286
287         isItRaining = true;
288         isItRainingHard = false;
289     }
290     else
291     {
292         if(currentTime - previousTime_LCDPrint > interval_LCDPrint )
293         {
294             lcd.clear();
295             lcd.setCursor(0,0);
296             lcd.print("Weather is Clear");
297
298             lcd.setCursor(0,1);
299             lcd.print("Temp: ");
300             lcd.print(temperatureC);
301             lcd.print("C");
302
303             //Serial.println("Weather is Clear");
```

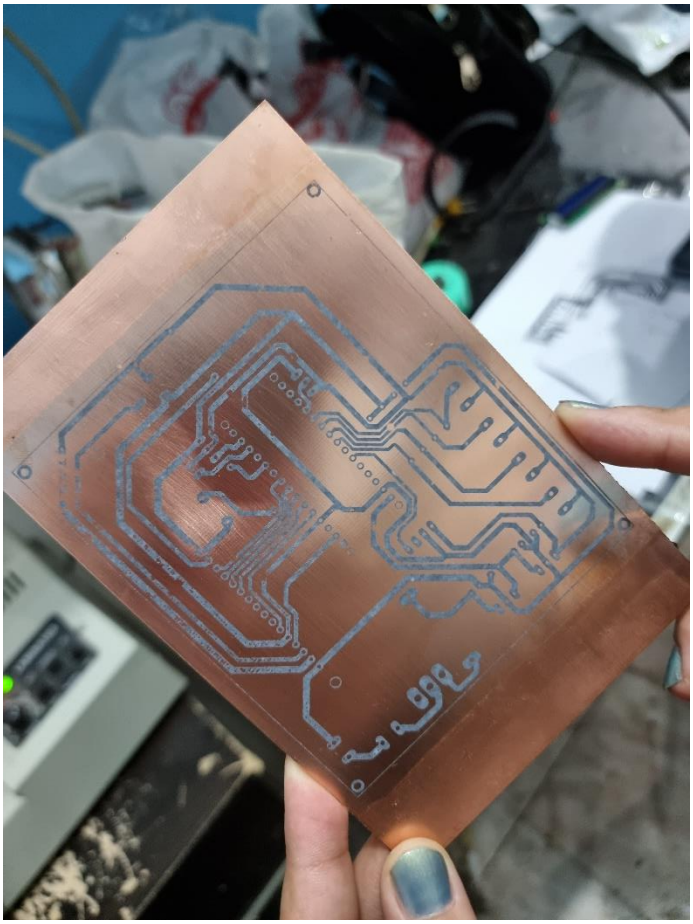
```
303             //Serial.println("Weather is Clear");
304
305             previousTime_LCDPrint = currentTime;
306         }
307
308         isItRaining = false;
309         isItRainingHard = false;
310     }
311
312     return isItRainingHard;
313
314 }
315
316 void AlarmBeepBeep()
317 {
318     for(int i=0; i<3; i++)
319     {
320         digitalWrite(buzzer, HIGH);
321         delay (500);
322         digitalWrite(buzzer, LOW);
323         delay (500);
324     }
325
326 }
327 void servoAllSharpAngle()
328 {
329     while(servoV_angle > servoV_SharpAngle)
330     {
331         servo_vertical.write(--servoV_angle);
332         delay(10);
333     }
334     while(servoV_angle < servoV_SharpAngle)
335     {
336         servo_vertical.write(++servoV_angle);
337         delay(10);
338     }
339
340     while(servoH_angle > servoH_SharpAngle)
341     {
342         servo_horizontal.write(--servoH_angle);
343         delay(10);
344     }
345     while(servoH_angle < servoH_SharpAngle)
```

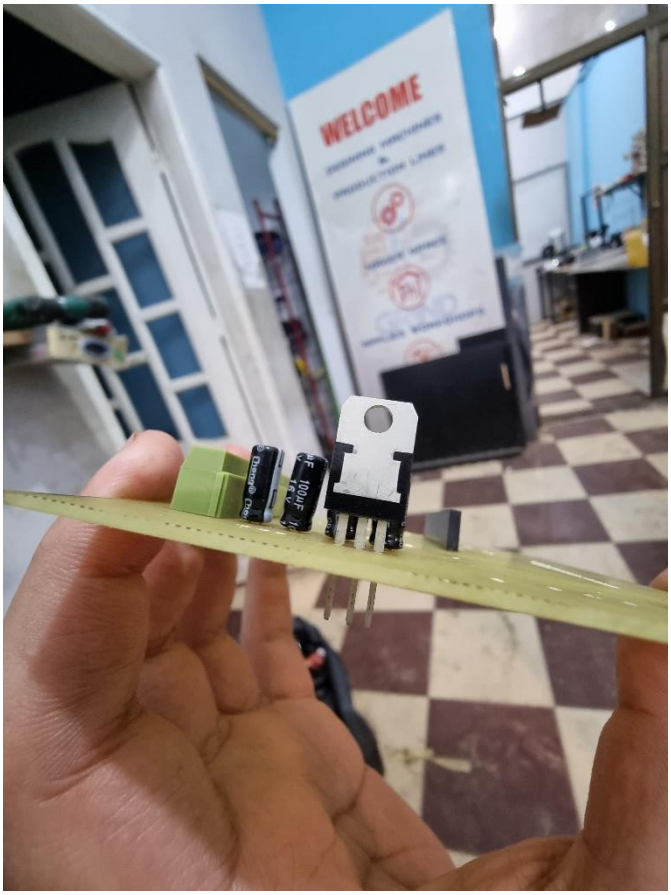
```
345         while(servoH_angle < servoH_SharpAngle)
346         {
347             servo_horizontal.write(++servoH_angle);
348             delay(10);
349 
350         }
351 
352     }
353 
354     void servoRotateUp()
355     {
356         //Serial.println("servoRotateUp");
357 
358         if (servoV_angle < servoV_LowerLimit)
359             servoV_angle = servoV_LowerLimit;
360 
361         servo_vertical.write(--servoV_angle);
362     }
363 
364     void servoRotateDown()
365     {
366         //Serial.println("servoRotateDown");
367 
368         if (servoV_angle > servoV_UpperLimit)
369             servoV_angle = servoV_UpperLimit;
370 
371         servo_vertical.write(++servoV_angle);
372     }
373 
374     void servoLockVertical()
375     {
376         //Serial.println("servoLockVertical");
377 
378         servo_vertical.write(servoV_angle);
379     }
380 
```

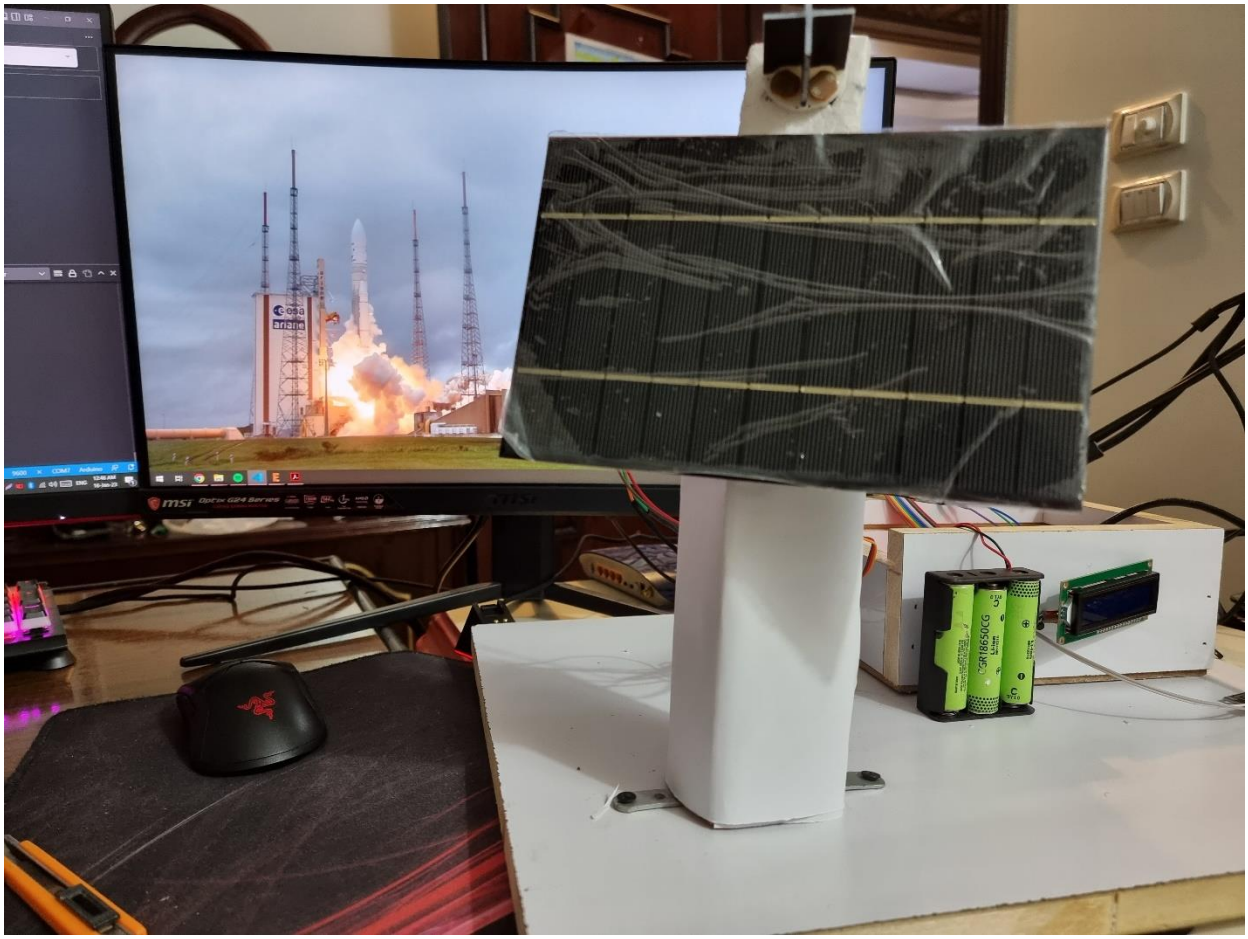
```
380
381   void servoRotateLeft()
382   {
383       //Serial.println("servoRotateLeft");
384
385       if (servoH_angle < servoH_LowerLimit)
386           servoH_angle = servoH_LowerLimit;
387
388       servo_horizontal.write(--servoH_angle);
389   }
390   void servoRotateRight()
391   {
392       //Serial.println("servoRotateRight");
393       if (servoH_angle > servoH_UpperLimit)
394           servoH_angle = servoH_UpperLimit;
395
396       servo_horizontal.write(++servoH_angle);
397   }
398   void servoLockHorizontal()
399   {
400       //Serial.println("servoLockHorizontal");
401       servo_horizontal.write(servoH_angle);
402   }
403
```

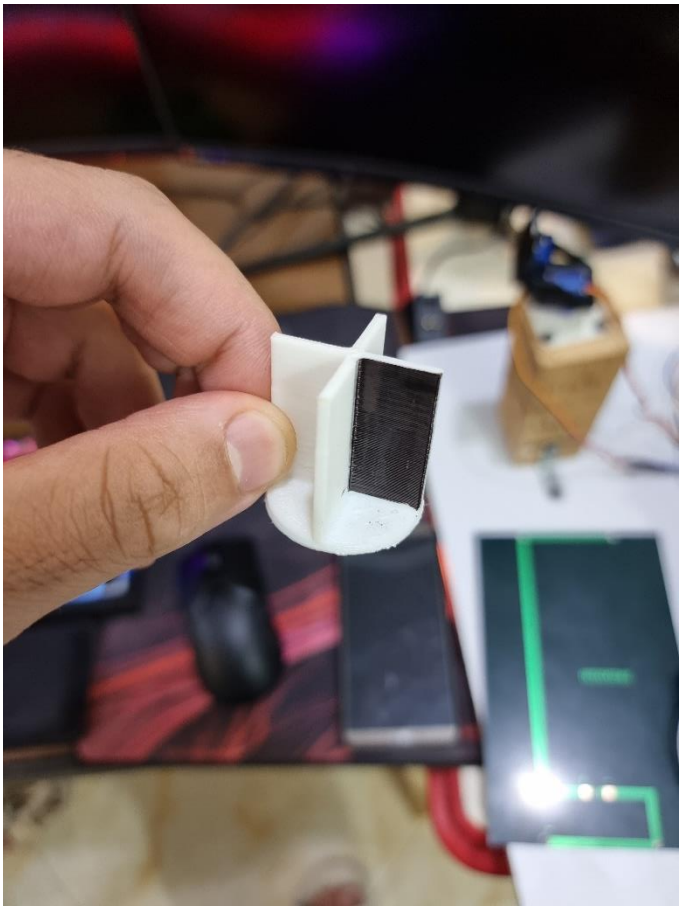
CHAPTER (6)
PROJECT PROGRESS AND DEVELOPMENT: A PHOTO JOURNAL

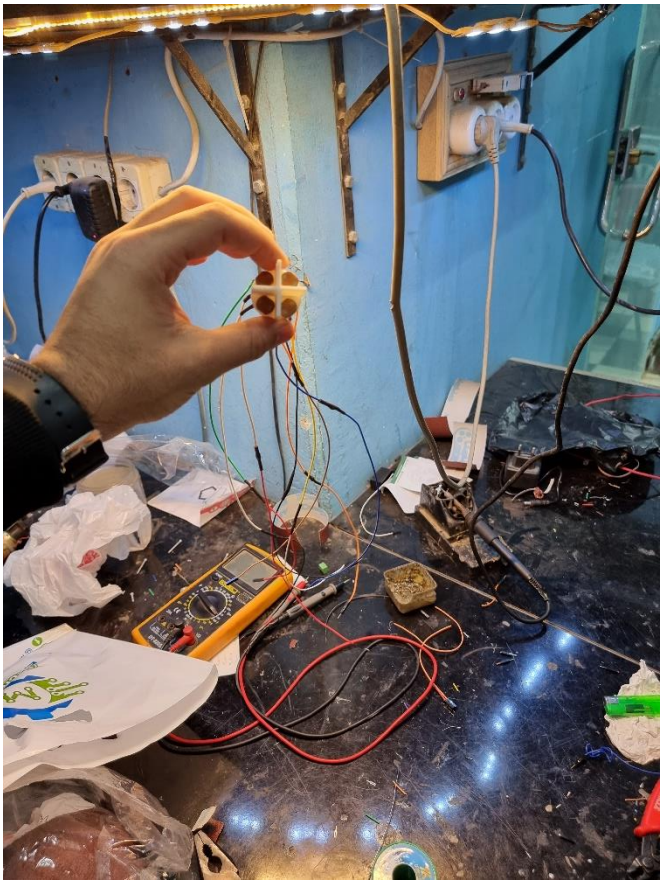

LAMINATOR

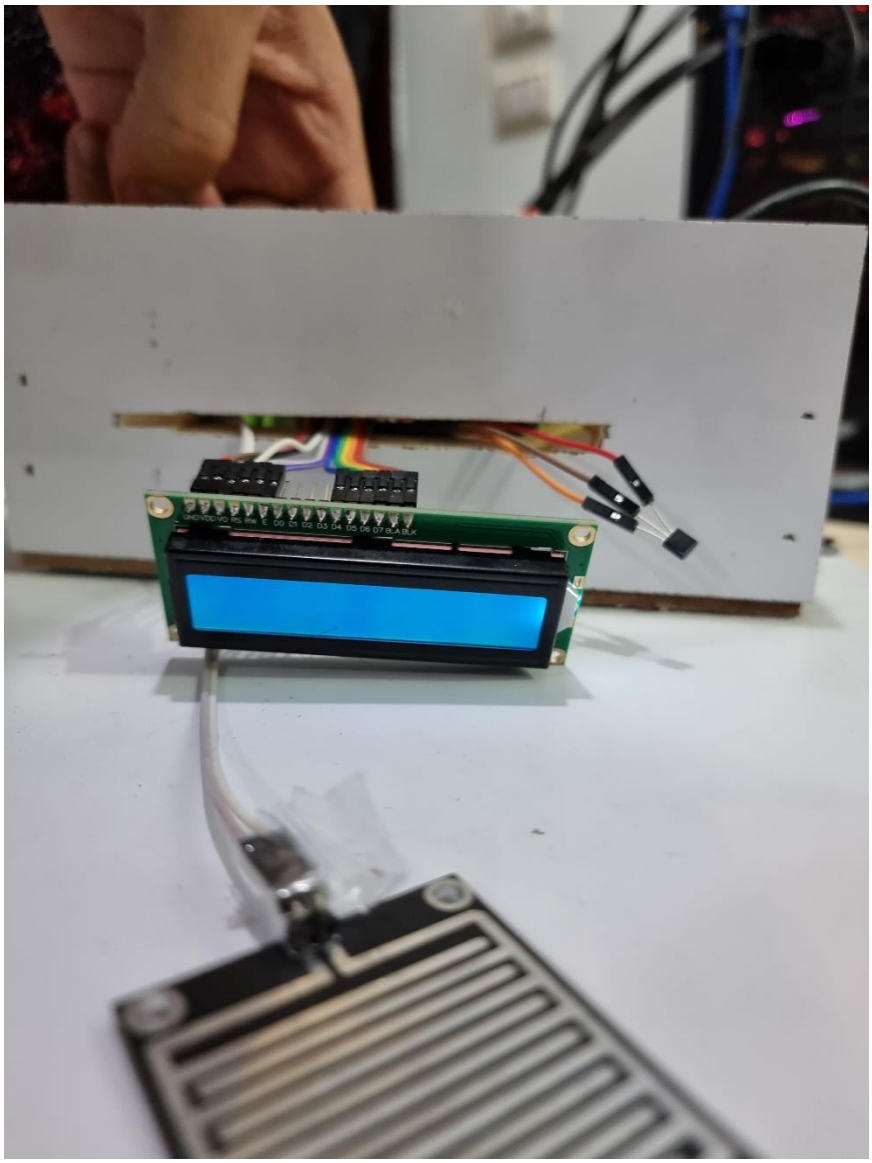


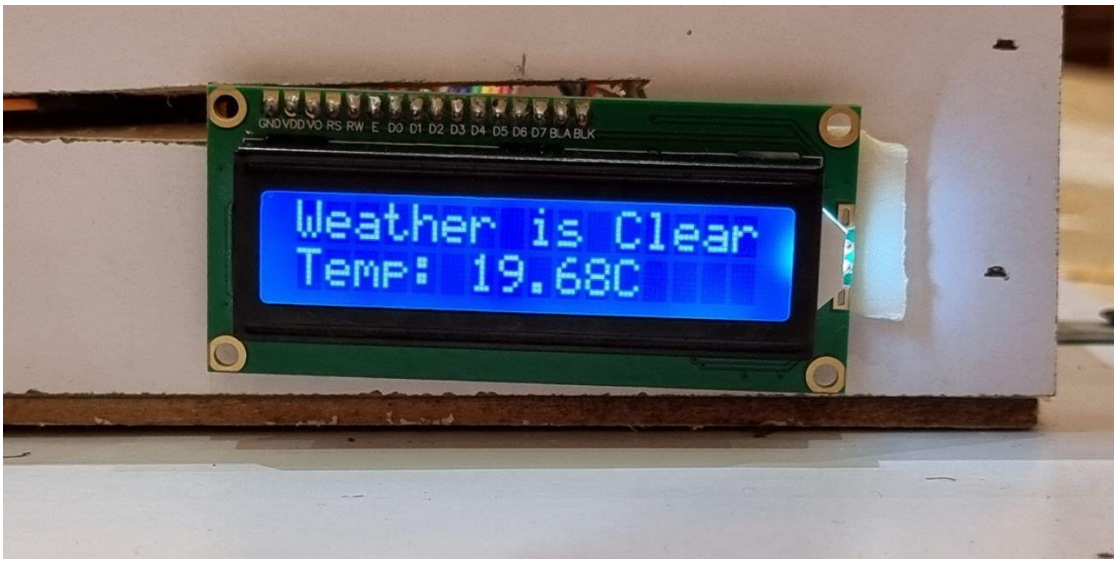

WELCOME

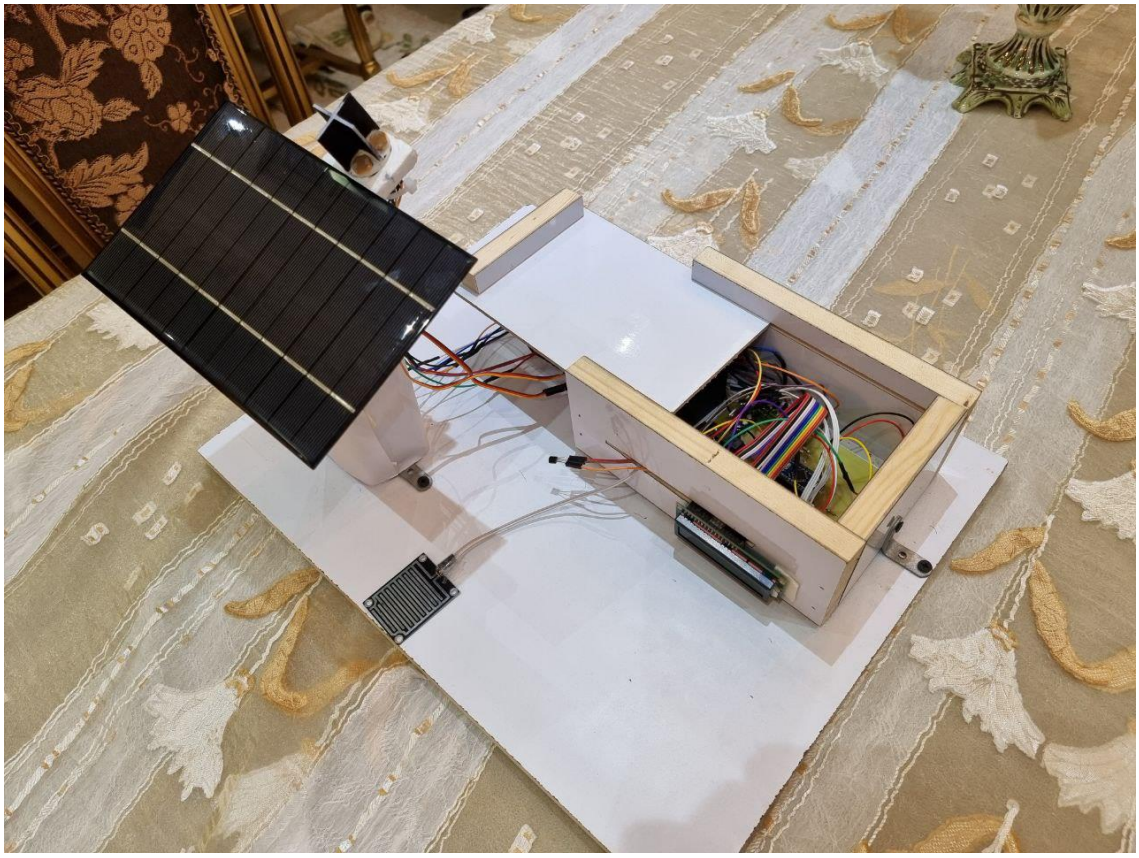

esa
ariane
msi

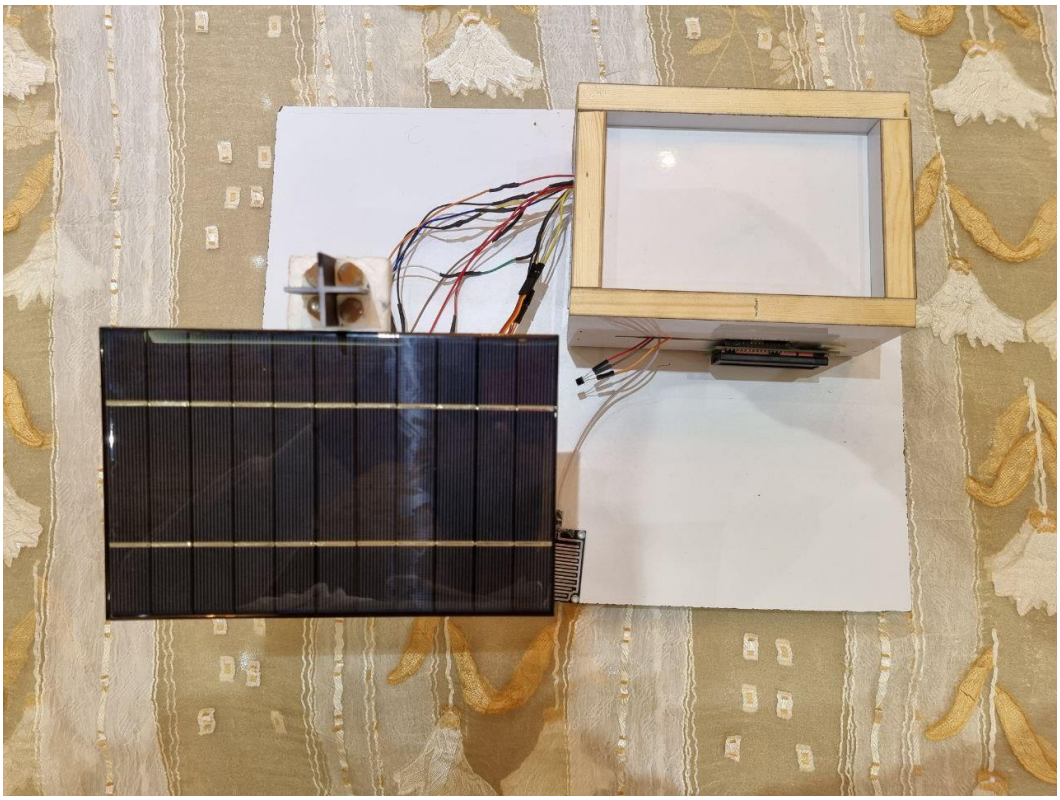







GND VDD VO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 BLA BLK
Weather is Clear
Temp: 19.68C





CONCLUSION

In conclusion, our dual axis solar tracker project was a successful endeavor that achieved its goals of accurately tracking and maximizing the amount of sunlight received by a solar panel. Throughout the project, we employed various techniques and technologies such as brainstorming, researching, budget estimation, mechanical design and printing, hardware component selection, and programming and simulation.

Starting with the introduction chapter, we discussed the importance of energy resources and the need for renewable energy sources. We also provided an overview of solar tracking and the project timeline.

In the second chapter, we delved into the process of brainstorming, researching, and budget estimation. We emphasized the importance of these initial steps in guiding the direction of the project and ensuring that it was feasible within the given budget constraints.

The third chapter focused on the mechanical design and printing of the solar tracker. We discussed the selection of a suitable mechanism and the importance of acquiring exact dimensions. We also highlighted the use of Solidworks and 3D printing in the design and fabrication process.

The fourth chapter discussed the selection and importance of hardware components, specifically sensors. We provided an in-depth analysis of the Light Dependent Resistance (LDR), Temperature Sensor (LM35), and Rain Sensor and their roles in the solar tracker.

In the fifth chapter, we discussed the programming and simulation aspect of the project. We highlighted the use of the Arduino IDE and Visual Studio Code as the working environment for writing the code. We also discussed the process of conceptualizing the project logic, getting familiar with the mechanical components and their interactions, determining LDR sensors' placement and configuration, and the general dual axis tracking logic.

The project logic is a crucial aspect of our innovative dual axis solar tracker project. The general dual axis tracking logic involves a series of steps that work together to ensure accurate

and efficient tracking of the sun's movement. The first step is to measure the values of the 4 LDR (light dependent resistor) sensors. These sensors are strategically placed and calibrated to accurately measure the light intensity of the sun and determine the direction of the sun's movement. Next, we compare each opposite pair of sensor readings to determine where the value of the light intensity is greater. This is done to ensure that the solar panel is aligned in the direction of the sun and is receiving the maximum amount of sunlight. By comparing the Top Left sensor to the Top Right sensor, we are able to determine whether the horizontal servo needs to move to the left or the right. The same process is applied to the Bottom Left and Bottom Right sensors, allowing for precise horizontal tracking of the sun's movement. Lastly, by comparing the Top Left sensor to the Bottom Left sensor, we are able to determine whether the vertical servo needs to move up or down. The same process is applied to the Top Right and Bottom Right sensors, allowing for precise vertical tracking of the sun's movement.

Overall, this project represents a significant step forward in the field of solar tracking and renewable energy. The dual axis solar tracker we developed is a cost-effective and efficient solution that can be used to increase the efficiency and effectiveness of solar panels. The techniques and technologies used in this project have the potential to be applied to other projects in the field of renewable energy and beyond.

In future, we may work on improving the design of our solar tracker by incorporating the use of Artificial Intelligence algorithms and Machine learning techniques which will enhance the performance of the solar tracker and make it more efficient. Also, we can integrate the solar tracker with a battery backup system which will provide a continuous power supply even on cloudy days.

In conclusion, we are proud of the successful completion of our dual-axis solar tracker project, and we hope it will inspire future innovations in the field of renewable energy. We believe.

REFERENCES

- Xianzhang Feng (2009**),** "Research and design of a smart home system based on raindrop sensor" | IEEEXPLORE.IEEE.ORG
- Daniel A Pritchard (1983), "Sun Tracking by Peak Power Positioning for Photovoltaic Concentrator Arrays" | ieeexplore.ieee.org
- John D. Garrison (2002) **,** "A program for calculation of solar energy collection by fixed and tracking collectors", | www.sci.sdsu.edu
- Dr. Dhanabal Rengasamy (2013) | "Comparison of efficiencies of solar tracker systems with static panel single-axis tracking system and dual-axis tracking system with fixed mount" | www.researchgate.net
- Jing-Min Wang (2013), "Design and Implementation of a Sun Tracker with a Dual-Axis Single Motor for an Optical Sensor-Based Photovoltaic System**"** | www.mdpi.com
- Manuel Ramos (2017) , "Characterization of LM 35 Sensor for Temperature Sensing of Concrete" | www.semanticscholar.org
- Ravi Kishore Kodali (2017) , **"**IoT based weather station**"** | ieeexplore.ieee.org
- Hanif Ali Sohag (2015) , "An accurate and efficient solar tracking system using image processing and LDR sensor" | ieeexplore.ieee.org
- Agus Kurniawan (2021) "IoT Projects with Arduino Nano 33 BLE Sense" | https://link.springer.com/
- Marek Szindler (2013**)** "Electrical properties mono-and polycrystalline silicon solar cells" | www.researchgate.net
- N.Othman (2014**),**" Performance analysis of dual-axis solar tracking system" | ieeexplore.ieee.org