

DE-42 (MTS)

ABDULLAH,

JAMSHED,

MARIAM,

USMAN

DESIGN AND DEVELOPMENT OF DUAL AXIS SOLAR TRACKING SYSTEM



**COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING NATIONAL
UNIVERSITY OF SCIENCES AND TECHNOLOGY
RAWALPINDI
2024**

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING



DE-42 MTS

PROJECT REPORT

DESIGN AND DEVELOPMENT OF DUAL AXIS SOLAR TRACKING SYSTEM

Submitted to the Department of Mechatronics Engineering
in partial fulfilment of the requirements
for the degree of

Bachelor of Engineering

in

Mechatronics

2024

Sponsoring DS:

Dr. Muhammad Osama Ali (Supervisor)

Dr. Tayyab Zafar (Co-Supervisor)

Submitted By:

Sardar Abdullah Khan Durrani

Jamshed Karamat

Mariam Afsar Khan

Muhammad Usman

ACKNOWLEDGMENTS

We would like to thank Allah Almighty for His blessings in enabling us to perform this extensive research for our thesis on the design and development of dual axis solar tracking system. In addition, we humbly thank Dr. Muhammad Osama, our supervisor, whose insightful counsel enabled us to overcome numerous obstacles. We also acknowledge the invaluable guidance that our co-supervisor, Dr. Tayyab Zafar, has given us. Lastly, a big thank you to our parents, friends, and colleagues for their support.

ABSTRACT

The sun is an incredibly powerful source of renewable energy. Harnessing this renewable energy source is crucial to reducing our dependence on fossil fuels and mitigating the effects of climate change. Solar panel systems are a renewable energy resource that provide sustainable and cost-effective power, thus tackling the energy crisis and environment problems.

Currently most solar panel systems are fixed. In comparison to them the Dual Axis Solar Tracking Systems are a promising technology to increase the efficiency of solar panels. A dual axis solar tracker is a device that follows the sun's movement in both the horizontal and vertical planes. This allows solar panels to remain perpendicular to the sun's rays throughout the day, maximizing energy output.

The aim of this project is the development of a dual axis solar tracker system by designing, manufacturing, assembling the mechanical structure and then programming it with the help of sensors and microcontroller to track the movement of the sun along with an actuating system to move the mechanism, hence making it capable of increasing the overall efficiency of the system.

Raspberry Pi 5 and Arduino UNO have been used as microcontrollers. A mechanical structure capable of performing dual axis motion has been developed. This is moved through an actuating system consisting of DC motors and a drive card that allows the motors to function as servos. The active tracking is done through LDR sensors and implementation of a control mechanism. Voltage and current sensors are used to acquire data regarding the power generation of solar panel. Finally, the power generated by static and dual axis solar tracking systems is calculated. Results regarding efficiency of the system are analyzed.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	i
ABSTRACT.....	ii
TABLE OF CONTENTS.....	iii
LIST OF FIGURES	vii
LIST OF TABLES.....	viii
Chapter 1 - INTRODUCTION	1
1.1. Motivation:	1
1.2. Problem Statement and Existing Solutions:	1
1.3. Novelty and Contribution:	1
Chapter 2 - BACKGROUND AND LITERATURE REVIEW	3
2.1. Working of Solar Panels:.....	3
2.2. Movement of Sun:	3
2.3. Need for Solar Tracking Systems:.....	3
2.4. Solar Tracking Systems and Their Classification:	4
2.4.1. Classification Based on Nature of Tracking:	4
2.4.2. Classification Based on Freedom of Motion:	5
2.5. Importance of Sensors in Solar Tracking Systems:.....	7
2.6. Sensors Used in Active Solar Tracking Systems	7
2.6.1. Light Dependant Resistors (LDRs):	7
2.6.2. Light to Frequency Convertor (LTF):.....	9
2.7. Sensors Used in Chronological Solar Tracking Systems:	10
2.7.1. Real Time Clock:	10
2.7.2. Position Sensor:	11
2.7.3. GPS Sensor:	11
2.7.4. Solid State Magnetic Compass Sensor:	11

2.7.5.	Dual Axis Inclinometer:.....	11
2.8.	Types of Motors and Actuators Used in Solar Tracking Systems:	11
2.8.1.	DC Motors:	12
2.8.2.	Stepper Motors:.....	12
2.8.3.	Servo Motors:	12
2.8.4.	Other Actuators:.....	13
2.9.	Types of Controllers and Algorithms Used in Solar Tracking Systems:	13
2.9.1.	Microcontroller Based Controllers:	13
2.9.2.	Control Algorithms:.....	14
2.9.3.	GPS Based Controllers:	14
2.9.4.	IoT Enabled Controllers:	15
2.10.	Experimental Results of Various Dual Axis Solar Tracking Systems:	15
Chapter 3 - METHODOLOGY		17
3.1.	Overview:	17
3.2.	Mechanical Structure:.....	17
3.2.1.	CAD Design:.....	17
3.2.2.	Material Survey:	20
3.2.3.	Structure Analysis:.....	20
3.2.4.	Structure Manufacturing and Assembly:	22
3.2.5.	Issues, Changes and Final Structure:	23
3.3.	Actuating System:	24
3.3.1.	Servo System:	24
3.3.2.	Components of the Servo System:.....	25
3.3.3.	PCB Design for Motor Driver Circuit:	27
3.3.4.	Incorporation of Limit Switching in Actuating System:	28
3.3.5.	Control Logic for Actuators:.....	29
3.3.6.	Code for Servo Driver:	29

3.4.	Microcontroller:.....	30
3.5.	Sensors:.....	31
3.5.1.	ADS1115 Module with Sensors:	31
3.5.2.	LDRs:.....	31
3.5.3.	Current and Voltage Sensors:	33
3.6.	Data Logging Module:	35
3.7.	Implementation of the Control Mechanism:.....	36
3.7.1.	Small-Scale Model:.....	37
3.7.2.	Initial Codes for LDRs, ADS1115 and Servos for Arduino:.....	38
3.7.3.	Arduino Code for Active Pan Axis Tracking:	38
3.7.4.	Arduino Code for Active Dual Axis Tracking:	39
3.7.5.	Coding in Raspberry Pi:.....	40
3.7.6.	Code for Servo Control in Pi:	40
3.7.7.	Code for Active Dual Axis Tracking in Raspberry Pi:.....	41
3.7.8.	Code for Data Acquisition and Logging:.....	42
3.7.9.	Final Code:.....	43
3.8.	Solar Panel:.....	46
3.9.	Electronics:	46
Chapter 4 - EXPERIMENTAL RESULTS AND ANALYSIS		47
4.1.	Overview:	47
4.2.	Data Collection and Experimental Setup:	47
4.3.	Data Acquired from Static Solar Panel System:.....	47
4.4.	Data Acquired from Dual Axis Solar Panel System:	48
4.5.	Power Consumed by Dual Axis Solar Tracking System:.....	48
4.6.	Analysis of Results:.....	49
4.6.1.	Comparison of Total Energy Production:.....	49
4.6.2.	Efficiency of Dual Axis Solar Tracking System:	50

4.6.3. Comparison of Net Energy Available:.....	50
Chapter 5 - CONCLUSION.....	52
5.1. Overview:	52
5.2. Summary of Achievements:	52
5.3. Future Recommendations:.....	53
5.4. Final Thoughts:.....	53
REFERENCES.....	56
ANNEXES.....	56
Annex A:.....	58
Annex B:	60
Annex C:	64
Annex D:.....	68
Annex E:.....	69
Annex F:.....	73
Annex G:	75
Annex H:	79
Annex I:.....	84

LIST OF FIGURES

Figure 1. CAD model of the mechanical structure.....	18
Figure 2. Spur Gears	19
Figure 3. Axial bearing	19
Figure 4. Connection between static and dynamic portion.....	20
Figure 5. Analysis of factor of safety	21
Figure 6. Analysis of factor of von Mises stresses.	21
Figure 7. Analysis of strain.....	22
Figure 8. Final Structure.....	24
Figure 9. Servo motors.....	26
Figure 10. Block Diagram of actuating system	27
Figure 11. PCB design.....	28
Figure 12. 3D PCB layout.....	28
Figure 13. LDR connections.....	32
Figure 14. ADS1115 connections LDR module.....	33
Figure 15. Current sensor connections	34
Figure 16. Voltage sensor connections	34
Figure 17. ADS1115 connections current and voltage sensors	35
Figure 18. Connections of data logging module.....	36
Figure 19. Data logging module	36
Figure 20. Small-scale dual axis tracker module.....	37
Figure 21. Flowchart of code.....	45
Figure 22. Solar panel specifications	46
Figure 23. Comparison of energy production by both system	49
Figure 24. Comparison of final energy available by both system	51

LIST OF TABLES

Table 1. Data Acquired by Static Solar Panel	47
Table 2. Data Acquired by Dual Axis Tracker Panel	48
Table 3. Percent Gain in Energy Production	49

Chapter 1 - INTRODUCTION

1.1. Motivation:

The application of solar energy has become a crucial means of tackling the critical issues of energy security and environmental sustainability. It is more important than ever to switch to renewable energy sources as the world struggles with the negative effects of climate change and the depletion of finite reserves of fossil fuel. Solar energy is one of the most promising of these renewable energy sources, providing a clean, plentiful, and endless source of energy. This motivation is driven by the understanding of the critical role that solar energy plays in the global energy system. Using solar energy to generate electricity promotes resilience and energy independence while also helping to cut greenhouse gas emissions. However, the static orientation of standard fixed solar panels limits their performance by preventing them from capturing the best sunlight all day long. This inefficiency highlights the need for novel solutions that can improve solar energy systems' efficiency.

1.2. Problem Statement and Existing Solutions:

The incapacity of typical solar panel systems to adjust to the sun's dynamic movement across the sky is one of their main problems. Although single-axis solar tracking systems provide some relief by either tilting or rotating the solar panels in accordance with the azimuthal movement of the sun, they are unable to take solar elevation angle fluctuations into consideration. As a result, these systems continue to produce less energy than they should, especially as the seasons change. By offering both azimuthal and elevation tracking capabilities, current dual-axis solar tracking systems overcome this drawback and maximize solar panel efficiency. These systems are mostly used in large-scale solar projects; however, they are frequently costly and complex. For this reason, they are not widely used, especially in Pakistan, particularly in small-scale residential and commercial contexts where simplicity and affordability are crucial.

1.3. Novelty and Contribution:

This project suggests a novel method for the design and development of a dual-axis solar tracking system suited for small-scale applications in light of the difficulties mentioned above. This project's main innovation is that it prioritizes accessibility,

affordability, and simplicity without sacrificing functionality. The mechanical structure is designed and built using inexpensive and easily accessible materials from the market. The sensors used for tracking and current, voltage measurements are also readily available and economic. The actuating system uses a self designed and fabricated servo driver making it affordable without compromising the performance of the system.

Chapter 2 -BACKGROUND AND LITERATURE REVIEW

2.1. Working of Solar Panels:

Solar panels are made up of numerous solar cells also called photovoltaic cells organized in specific configuration. Solar energy is converted into electrical energy using solar panels by the photovoltaic effect. These cells are made up of semiconductor material commonly made of silicon. Sunlight is composed of photons, when it strikes the surface of these cells the electrons in cells energized, causing them to move within the material. Electric current is produced by the electron's motion, which is then obtained and directed through an electrical circuit. This electron movement generates electricity known as direct current (DC). To make this energy usable for home, businesses etc. we convert direct current (DC) current to alternating current (AC) current by the help of inverters as AC is the standard form of electricity used in most applications [1].

2.2. Movement of Sun:

The movement of the sun both daily and seasonally, is a fundamental aspect of the nature. On daily basis sun rises in the east and sets in the west due to the Earth's rotation on its axis. The angle and height of the sun in the sky change throughout the day, inducing factors such as shadow length and the intensity of sunlight. Additionally, the observer's latitude also affects the sun's daily course at higher latitudes, there are more changes in solar altitude. On a seasonal scale, the sun's movement is categorized by changes in its declination, which is the apparent latitude where the sun is directly above at noon. The seasons changes because of this variation. During summer, the sun reaches its highest declination, resultant in lengthier days and shorter nights. Contrarywise, in winter, the sun's lower declination leads to shorter days and longer nights [2]. The days in which time of day is equal to time of night occurs around March 21st and September 23rd. The study of the sun's movement enhances our knowledge in contributes to harnessing of solar energy.

2.3. Need for Solar Tracking Systems:

The need for solar tracking systems stems from the quest for optimizing the efficiency and output of solar energy systems. Traditional fixed solar panels are effective but

have limitations in harnessing sunlight to its complete potential. Solar tracking systems address this limitation by orienting solar panels dynamically to follow the sun's path across the sky. This dynamic orientation ensures that solar panels are consistently facing the sun, maximizing the absorption of sunlight throughout the day. Solar tracking systems come in various types, including single-axis and dual-axis trackers. Single-axis trackers follow the sun's movement either in the east-west or north-south direction [3], while dual-axis trackers combine both movements for even greater precision in all directions. The implementation of solar tracking technology is especially helpful in regions with high solar incidence as it enhances energy output.

2.4. Solar Tracking Systems and Their Classification:

A solar tracking system is a device that maximizes the amount of sunshine by aligning a solar panel with the sun's movement. As the intensity of sunshine drops, it adjusts its position automatically. The solar tracker is engineered to ensure that there is always a 90-degree angle between the sun and the solar panel. When compared to fixed modules, the generation of power can be enhanced by approximately 40% by using solar trackers. This tracking device has a 180-degree rotational range. As a result, a solar tracker is far better than a fixed module [4].

Solar Tracking systems can be classified into two types.

2.4.1. Classification Based on Nature of Tracking:

This is further divided into passive and active solar tracking systems:

A. Passive Solar Tracking Systems:

This tracking mechanism is based on the theory of thermal expansion of materials or a compressed gas fluid with a low boiling point that is propelled in one direction or another. On either side of the solar panel, there is usually a shape memory alloy or a chlorofluorocarbon (CFC). The two sides are in balance when the panel is oriented perpendicular to the sun. When the sun moves, heat from one side of the panel causes that side to expand while the other side contracts, rotating the panel. A passive system has the potential to increase efficiency by 23% [5]. These systems are cheaper in comparison to active systems but are not commercially popular [6].

B. Active Solar Tracking Systems:

These systems make use gear trains and motors to direct the tracker in response to the controller's instruction based on the direction of the sun. Throughout the day, the sun's location is tracked. Depending on how it is made, a tracker in the dark will either stop or go to sleep. Light-sensitive sensors, such as LDRs, are used, their voltage output is fed into a microcontroller, which uses actuators to change the solar panel's position [7].

2.4.2. Classification Based on Freedom of Motion:

This is further divided into single axis and dual axis solar tracking systems:

A. Single Axis Solar Tracking Systems:

Single axis trackers have one degree of freedom that acts as an axis of rotation. The axis of rotation of single axis trackers is typically aligned along a true North meridian. It is possible to align them in any cardinal direction with advanced tracking algorithms.

The axis of rotation that single axis tracker consists of are:

- **Horizontal Single Axis Tracker:** It is the most prevalent kind of single-axis tracker design and works better in lower latitudes. Its rotational axis is horizontal with respect to the earth. They have a fixed axis that rotates from east to west and is parallel to the ground, making them incredibly flexible.
- **Vertical Single Axis Tracker:** The axis of rotation of these trackers is vertical with respect to the ground. During the day, they typically rotate from east to west. At higher latitudes, these trackers perform better than horizontal axis trackers. The face of the module in vertical single-axis trackers is usually oriented at an angle to the rotational axis. It passes over a rotationally symmetric cone that resembles a module path around the rotational axis.
- **Polar Aligned Single Axis Tracker:** These trackers are well known standard technique in ascending a telescope reinforce shape. The tilted

single axis is aligned to the axis of rotation at polar star. Therefore, it is called a polar aligned single axis tracker.

- **Tilted Single Axis Tracker:** The trackers' ability to rotate along both horizontal and vertical axes is typically what makes them famous. Tilt angles of this system are frequently restricted for lowering the elevated end height and wind profile. Assuming an orientation parallel to the rotational axis, they usually face the module. It passes over a rotationally symmetric cone that resembles a module track around the rotational axis. By tracking the sun throughout the day, these trackers provide the best efficiency and solar tracking capabilities available.

B. Dual Axis Solar Tracking Systems:

Dual axis tracker has two degrees of freedom. It tracks the movement of the sun from East to West through the day, and from East to North or South through the season. The movement from East to West also known as Zenithal Angle and the other from East to North or South that happens through the year also called Azimuthal Angle. As this tracker move along the sun's direction vertically and horizontally, they help to achieve maximum solar energy.

- **Tip-Tilt Dual Axis Tracker:** The primary axis of a tip-tilt dual axis tracker is oriented horizontally with respect to the ground. This tracker uses an ascended panel array mounted atop a long pole. A movement from east to west is achieved by rotating the array around a pole's peak. One end post of the main axis of rotation can be divided among trackers to lower the installation cost. They are incredibly flexible, and in order to properly orient the trackers in relation to one another, their axis of rotation must be parallel to one another.
- **Azimuth-Altitude Dual Axis Tracker:** These trackers have their primary axis which is set vertically with reference to the ground. The elevation axis also known as secondary axis is ordinary just as the primary axis. Its function is quite same to tip-tilt system, but they vary in the array rotation for everyday tracking. They use a large ring mounted on the ground with the array mounted on a series of rollers

instead of rotating the array around the top of the pole. The main advantage of this tracker is that it allows to support more large arrays. However, compare to TTDAT, it may decrease the system density especially considering inter-tracker shading when the system is placed close together than the diameter of the ring. They are largely used in different research on tracking system based on their references [4].

2.5. Importance of Sensors in Solar Tracking Systems:

The solar tracking mechanism is an electromechanical system that ensures solar radiation is always perpendicular to the surface of the solar cells, maximizing the amount of energy that may be harvested. Sensors play a crucial role in this tracking of the sun. Both the Active and Chronological Solar tracking systems require Sensors. Data from the sensors is used by the microcontroller to control the motors and hence align the tracking system perpendicular to the sun's rays.

2.6. Sensors Used in Active Solar Tracking Systems

Active Solar Tracking Systems, also called the closed loop tracking systems, rely on photosensors in order to track the sun. The most popular photosensor used for this purpose is the Light Dependent Resistor (LDR). Some solar tracking systems have also used it along with Light to Frequency Convertor (LTF) sensor [8].

2.6.1. Light Dependant Resistors (LDRs):

The basis for the operation of light-dependent resistors (LDRs) is photoconductivity. LDRs are frequently employed in circuits that need to detect the presence or amount of light. Their photosensitivity is precisely why they were designed [9]. As the name suggests, their resistance varies with light intensity. The resistance of LDRs decreases as the light intensity increases.

LDRs are the most common photosensor used in solar tracking systems. The difference in the use of these sensors in tracking systems is in the way they are placed, and the comparison techniques used to gain valuable readings from them.

A. Different Placement of LDRs in Solar Tracking Systems:

LDRs have been placed in various ways in solar tracking systems.

- **Square Formation:** One popular way of LDR placement is at each corner of the solar panel structure. We can name this placement as the square formation. In this placement we have a top left, top right, bottom left, and bottom right LDR. Hence, value at the top of the panel structure is dependent on the data we get from 2 LDRs. Similarly, right, left, and bottom values are also dependent on 2 LDRs. The LDRs in the square formation in some tracking systems have been placed either on the corners of the complete panel structure [10] or have been mounted in this formation at the top of the panel with the help of a cross shaped piece. The LDRs were then positioned within each of the cross's four corners, giving the LDRs a square formation [11]. This sort of placement is also aided by an opaque plate, so that some LDRs are shadowed and the one that is illuminated is the one with most sunlight. For the case of cross-piece, the goal of it is also to cast a shadow on two or more LDRs if the cross is not pointing perpendicularly towards the sun [12].
- **Cross Formation:** Another type of placement of LDRs used in solar tracking systems is at the top, bottom, left and right of the panel structure. We can call this placement the cross formation. In this placement, the values at the top, bottom, left and right of the solar panel are dependent on one LDR only [13].

B. Different Comparison Techniques for LDR Values:

In [10] the LDRs are placed in square formation at each corner of the solar panel. Each LDR is connected to a resistor in such a way that they act as a voltage divider and the output is provided to analog input pins of microcontroller. The Comparison technique is then carried out through the microcontroller. After getting analog values of each LDR i.e., top left, top right, bottom left and bottom right, the averages are calculated such that average of two top LDRs give Top Avg. Similarly, two bottom LDRs give Bottom Avg, two right LDRs give Right Avg and two left LDRs give Left Avg. The averages are then compared. If there is a variation in top and bottom averages, then the vertical motor is tilted accordingly.

Simultaneously, the left and right averages are compared. Their difference is accommodated by rotating the horizontal motor.

The solar tracking system in [11] and [12] uses the same comparison technique as [10]. The only difference in [11] is that a variable resistor is attached to the microcontroller. The values of the variable resistor are compared with the readings from the four Light Dependent Resistors (LDRs). The system's sensitivity can be changed by adjusting the variable resistors' set point. After averages are calculated, Difference Vertical (DV) and Difference Horizontal (DH) are calculated through $(AVT-AVB)/2$ and $(AVR-AVL)/2$. DV and DH should be greater than the setpoint in order to start the alignment process.

In [13] 4 LDRs are placed in the cross formation. A 5th LDR is used to sense the nighttime. The readings from all these LDRs are passed to a light comparison unit. Hence, the comparison technique is carried out by this unit rather than the microcontroller. Diodes and comparator circuits make up this light comparison unit. To compare the voltage level between two LDRs, a comparator circuit is utilized. Here, the voltage levels of the north-south and east-west sides, or the horizontal and vertical axes, are compared using two dual comparator ICs (LM1458). After comparing their voltages in the respective axis +Vcc and -Vcc are sent to the diodes from two individual outputs of LM1458. The final comparator sends the output to a diode after comparing the voltage level coming from the night-detecting sensor with a preset reference voltage. Since the microcontroller only operates in the range of 0 to +5 volts, diodes are utilized in this situation to ignore the reverse voltage drop.

2.6.2. Light to Frequency Converter (LTF):

Light to Frequency Converter (LTF) sensors have been used along with LDRs in solar tracking systems because the frequency is exactly proportional to irradiation, hence the use of light-to-frequency converters is an extremely sensitive and accurate procedure. LDRs are voltage based while LTFs are directly related to power, especially current. Additionally, in [8], the LTF helps synchronize the PILOT and PANEL so that no additional switches are required.

This design of a solar tracking system is based on two primary components: PANEL, a revolving cell system, and PILOT, a tracking component that continuously measures maximum irradiation. Three components make up the tracking system: a light-to-frequency converter sensor (LTF), and two light-dependent resistors (LDR). The LTF ascertains the location of maximal irradiation, whereas the two LDRs are used to ascertain the direction of rotation. The three sensors are fixed to a holder in the shape of a large W, with the LTF fixed in the center and the LDRs positioned in the troughs on either side. The system is initially facing east, awaiting the rising of the sun. At that point, the PILOT begins following the sun. Two light LDRs placed on the PILOT are used for this. The voltages of the two LDRs are continually compared, and the PILOT turns in the direction of the LDR with more incident light than the other by a predetermined value that the user has already specified. The two light-to-frequency converters (LTF), one located on the panel and one on the pilot, take over after the direction has been set. a procedure for comparing the induced frequency caused by the PILOT LTF with the PANEL LTF starts. If the difference exceeds the pre-set, PANEL follows PILOT until the two frequencies equalize, indicating alignment. The PANEL then stops and remains in its present position while the PILOT continues monitoring the sun freely and the comparison procedure resumes.

2.7. Sensors Used in Chronological Solar Tracking Systems:

Chronological Tracking Systems require mathematical equations and already existing sun paths to program a microcontroller in order to track the sun. This programming requires information such as current position, position of magnetic north, inclination, and real time and date etc. All this information is provided through different sensors.

In [14] the sensors used for Chronological tracking are:

2.7.1. Real Time Clock:

A real time clock is a type of clock that records the current time. The microcontroller tracks the sun's annual motion and the nighttime in order to place the solar panel at its starting position. It does this by using the month and hour numbers obtained from the RTC device.

2.7.2. Position Sensor:

Position sensors track the annual motion of the sun. Here, the position sensor is a variable resistor coupled to another resistor. Thus, the position sensor output varies along with the variable resistor's resistance. This circuit's output is delivered to the controller, and the sun's different latitude angle during yearly motion is represented by different voltages in the position sensor circuit's output.

In [15] the sensors used are:

2.7.3. GPS Sensor:

Controller has internal real time measuring system capable of providing correct date and time even if the system gets disconnected from the power. After powering up controller, current global position of the system is provided by the satellite using the GPS according to the current date and time.

2.7.4. Solid State Magnetic Compass Sensor:

The position of a magnetic north is provided to the controller by this sensor.

2.7.5. Dual Axis Inclinometer:

By using this sensor, the controller is capable of determining if the system needs possible adjusting because of the soil inclination.

All these sensors are connected to the local controller who is responsible to calculate correct position of the Sun on the horizon. Based on the values of all these sensors, the controller can calculate the location of the Sun according to the current time and date. After the calculation of the initial sun position, controller starts to orientate solar panels according to the code of chronological tracking.

2.8. Types of Motors and Actuators Used in Solar Tracking Systems:

In solar tracking systems, various types of motors can be used to control the movement of solar panels along both horizontal and vertical axes. Each type of actuator has its own advantages and disadvantages. Some common types of actuators used in solar tracking systems are:

2.8.1. DC Motors:

The most widely used motors in the world are direct current, or Dc motors. Some of the major advantages of Dc motors are that they are simple and easy to control, cost-effective and can be easily interfaced with microcontrollers. In comparison to other motor types, it is less expensive and uses less energy just while it is moving [16]. These types of motors are often used with gear reduction modules to increase the torque in applications where low speed, but high torque is required.

2.8.2. Stepper Motors:

Stepper motors can also be employed in solar tracking systems for their precise angular positioning capabilities. These motors enable accurate adjustments of solar panels along horizontal and vertical axes, ensuring they follow the sun's path effectively. Stepper motors operate in discrete steps, allowing incremental movement and precise alignment without continuous feedback. Their reliability and ease of control make them a popular choice, ensuring efficient solar tracking and maximizing energy harvesting efficiency [17]. However, stepper motors are relatively expensive compared to DC motors and require complex control electronics.

2.8.3. Servo Motors:

Servo motors are indispensable components in solar tracking systems, especially in the context of optimizing the efficiency of solar panels. These motors excel in providing accurate and controlled motion, allowing solar panels to precisely follow the sun's path. Equipped with feedback systems, servos continuously monitor the position of the panels and make real-time adjustments, ensuring they remain aligned with the sun's position throughout the day. Servo motors offer high torque at low speeds, enabling smooth and precise movement necessary for both horizontal and vertical adjustments in solar tracking systems. Their ability to maintain a specific position without drifting, along with their rapid response to changing conditions, makes them an ideal choice for applications where precision and reliability are paramount. In solar tracking,

servos play a vital role in maximizing energy capture by ensuring optimal alignment, leading to increased overall energy efficiency of the solar power system.

2.8.4. Other Actuators:

Other types of actuators used in solar tracking are linear, hydraulic and pneumatic actuators. Linear actuators offer straightforward linear motion and are ideal for specific tracking designs, allowing solar panels to adjust their tilt accurately. However, Linear actuators often have a restricted range of motion, making them unsuitable for solar tracking systems that require extensive movement. Achieving rotational movement with linear actuators requires additional mechanisms, potentially increasing the complexity of the tracking system design. Hydraulic actuators, on the other hand, are renowned for their high torque and power output, making them suitable for heavy-duty applications in larger solar tracking systems but Hydraulic systems require regular maintenance due to the possibility of fluid leaks, which can be detrimental to the environment and the system's functionality. Pneumatic actuators, operating on compressed air, offer a simple and efficient solution for certain industrial solar tracking applications, delivering reliable linear or rotational motion. However, Pneumatic actuators generally have lower power density compared to hydraulic or electric systems, which might limit their application in heavy-duty solar tracking setups.

2.9. Types of Controllers and Algorithms Used in Solar Tracking Systems:

Controllers used in solar tracking systems play a crucial role in ensuring that solar panels accurately follow the sun's movement, optimizing energy capture. Several types of controllers and control algorithms are utilized in these systems:

2.9.1. Microcontroller Based Controllers:

Following microcontrollers are usually used:

A. Arduino:

Arduino microcontrollers are popular in DIY and small-to-medium scale solar tracking projects. They are versatile, affordable, and can interface with various sensors and actuators [18].

B. Raspberry Pi:

Raspberry Pi boards offer more computational power than Arduino and can run complex control algorithms. They are suitable for larger or more sophisticated solar tracking systems.

C. PIC Microcontrollers:

Programmable Integrated Circuits (PICs) are commonly used in industrial-grade solar tracking applications due to their reliability and efficiency.

2.9.2. Control Algorithms:

Different types of control algorithms are:

A. Proportional-Integral-Derivative (PID) Control:

PID controllers are widely used for their ability to provide precise and stable control. They adjust the position of solar panels based on the error signal [19].

B. Fuzzy Logic Control:

Fuzzy logic controllers can handle imprecise input data and are effective in dealing with uncertainties. They are used in situations where the system parameters are not well-defined.

C. Machine Learning Algorithms:

Machine learning techniques, such as neural networks, can optimize solar tracking by learning patterns from data, adapting to changing environmental conditions, and enhancing accuracy over time.

2.9.3. GPS Based Controllers:

GPS modules provide real-time information about the sun's position based on the system's geographical location and time. This data is used by the controller

to calculate the solar azimuth and elevation angles, ensuring precise solar tracking. These types of controllers are often used in passive solar-tracking systems.

2.9.4. IoT Enabled Controllers:

Controllers integrated with the Internet of Things (IoT) technology can transmit data to cloud platforms. Remote monitoring and control are possible, allowing users to access real-time tracking data and adjust remotely. This is also useful in doing data analysis for comparing the efficiency of these systems compared to a fixed system.

2.10. Experimental Results of Various Dual Axis Solar Tracking Systems:

In [10], the experiment was conducted by calculating and comparing the power output (through current and voltage) of fixed solar panel system with the dual axis solar tracking system for a day from 9AM to 5PM. After that the normalized enhancement was calculated by the formula:

$$\text{Normalized Enhanced } \eta \% = [(Dual Axis (Watt) - Fixed Array(Watt)) / Maximum Power(Watt)] \times 100$$

Similarly, in [13], current, voltage and then power is plotted in a table from 8am to 5pm, for a day, for both static and dual axis tracking panel. After this, the power gained by tracking panel is calculated. The result shows that throughout the day, the static panel gives a total power of 40.798 watts. In the meantime, the tracking panel gives 52.948 watts of power altogether during the day. Consequently, 52.78% is the average power gain of a tracking panel over a stationary panel.

In [11] for a duration of one day, the dual-axis solar system is evaluated hourly by monitoring its no-load voltage and contrasting it with the static tracker. The plot of no-load voltage against time is displayed in a figure which clearly shows that a dual-axis solar tracker generated more voltage than a static tracker at any time.

The system in [14] is a hybrid system so the results show the power output of static, hybrid, and continuous (active) tracker systems. 45.21 watts, 56.69 watts, and 58.24 watts, respectively, are the total power of the static panel, hybrid tracking system, and

continuous tracking system for the duration of the day. Thus, the hybrid tracking system's average power increase over the static panel is 25.62%. Likewise, the continuous tracking system has an average power gain of 28.10% over the static panel and 4.19% over the hybrid tracking system. However, the motors in the two solar tracking systems use different amounts of power. Thus, a hybrid tracking system saves 44.44% of the power compared to a continuous tracking method. Hence, it can be concluded that the hybrid dual-axis solar tracking system's power gain is nearly identical to that of a continuous dual-axis solar tracking system, while the hybrid tracker's system operation saves 44.44% of the energy compared to a continuous tracking system.

In [15] a comparison is made between stationary and chronological dual axis solar tracking system. This comparison is shown to be made over a period of 12 months. From the results, largest energy differences are seen in the months of May, June, July, and August. It is concluded that chronologically implemented dual axis system provides 30-40% more energy than fixed system.

In this case, [12] readings are taken under the light of the torch which is kept at different angles with respect to the static panel. The static solar panel is positioned 45 degrees from the horizontal. The power output differences between the moving and static panels are plotted in a table. From the table it is seen that the overall power generated by the moving panel is more than the total power produced by the static panel. These findings demonstrate that the goal of utilizing the sun tracking system to optimize the solar panel's energy production has been accomplished.

Chapter 3 -METHODOLOGY

3.1. Overview:

What we aim to achieve is a dual axis solar tracking system. To get to this final result we need a mechanical structure, actuating system, sensors, and a control mechanism.

First step of the project is the mechanical structure; so, the design and development of dual axis mechanical structure is done. This includes the cad model, material survey, structure analysis and then manufacturing & assembly of structure.

In order to drive the mechanism, we require actuators, so we have fabricated a servo drive for the movement of the mechanism.

Sensors are used to track the sun hence optimal placement of sensors and their calibration to get valid data plays a critical role in the tracking of sun. Sensors are also used for data acquisition.

The microcontroller is then programmed to control the complete mechanism. The data from the sensors is used to drive the motors via the controller so the panels are aligned with the sun.

Data analysis is then done, and the final results are presented to show the efficiency after testing and comparing the fixed and dual axis systems.

3.2. Mechanical Structure:

The first step of the project was to design a mechanical structure that would be capable of performing two axis motion; namely pan and tilt, bear the necessary loads, and be suitable for small scale applications hence be accessible, affordable, and simple. After the design, material survey and structure analysis were done.

3.2.1. CAD Design:

We first came up with a CAD design of the structure. We used SolidWorks software for this purpose.

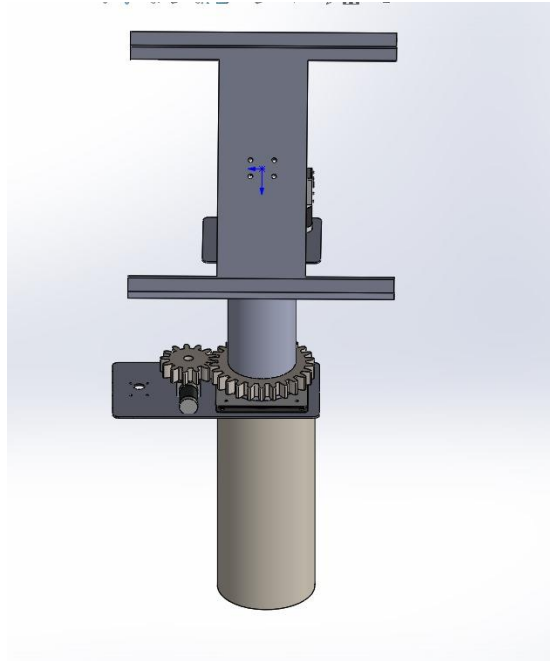


Figure 1. CAD model of the mechanical structure

As seen from the CAD model, the mechanical structure consists of two portions.

A. Static Portion:

The bottom portion is static. It consists of two square plates and a cylinder in between. The bottom square plate serves as a base and the cylinder is hollow. The cylinder is used to pass wires from top to bottom. The second square plate is at the top of the cylinder and attaches with a bearing to connect to the dynamic part. It also has a motor mount on it.

B. Dynamic Portion:

The upper portion moves. It again consists of a cylinder between two plates.

- **Pan Movement:** The next plate, attached with the cylinder has holes to screw the bearing with the plate. This is responsible for the pan rotation. Two spur gears are used in the pan axis rotation mechanism. The cylinder gives the structure further height.

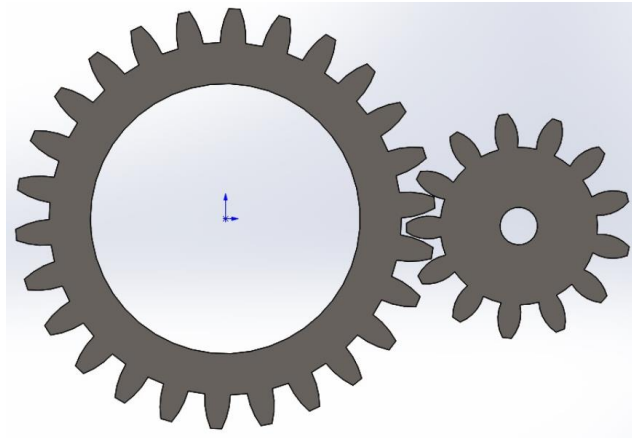


Figure 2. Spur Gears

- **Tilt Movement:** We then have a tilt mechanism mounted on the top of the structure. The tilt portion consists of a U-shaped mechanism. The mechanism is static from one end while the other moving end hosts the solar panel. This mechanism is restricted from moving freely by the self-locking motor.

The lower static and upper dynamic parts have a bearing connection between them. An axial bearing is used for this purpose.

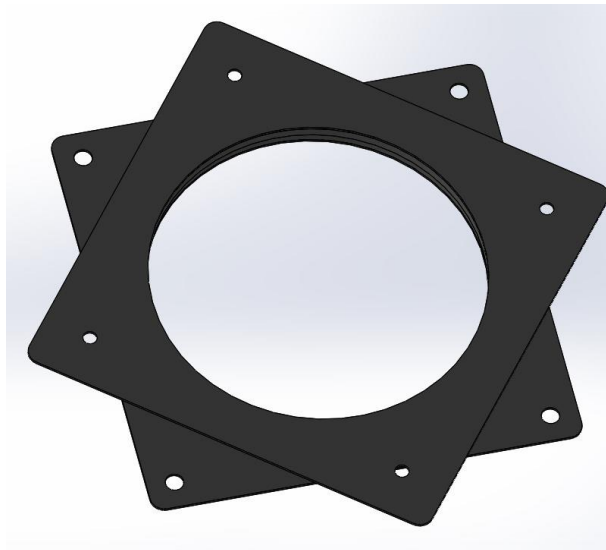


Figure 3. Axial bearing

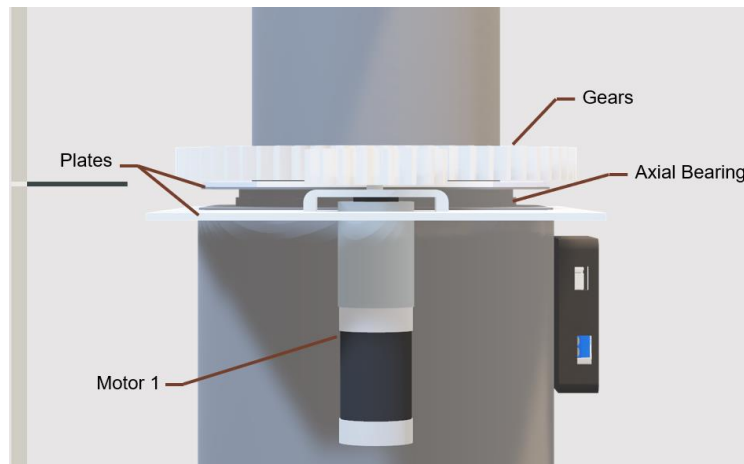


Figure 4. Connection between static and dynamic portion.

3.2.2. Material Survey:

After initial research, we decided to make a structure of mild steel and aluminium. We decided the square plates will be of mild steel and the cylinders of aluminium. However, after material survey and market research, we learned that aluminium was much more expensive than mild steel, and aluminium to mild steel welding is quite difficult. The rates of aluminium were 2300Rs/kg while that of mild steel were 230Rs/kg. After these findings, we concluded that a better decision would be to make a structure of mild steel completely. Mild steel is less costly, easily available, and arc welding; to weld mild steel with mild steel, is a relatively much easier process than the welding of aluminium with mild steel. The only drawback is that a complete mild steel structure is relatively heavier than an aluminium and mild steel structure. This concern was however satisfied once the structure analysis was completed.

3.2.3. Structure Analysis:

After the design of the mechanical structure and material survey, we did a structure analysis on our mechanical structure design using the Fusio360 software. From our findings we concluded that our structure was capable of bearing all the necessary loads and strains. Our concern about the structure being heavier than aluminium and mild steel structure was also satisfied here because being heavier not only did not affect the functionality of the structure and mechanism but also increased the factor of safety of the structure. Some structure analysis results are displayed below:

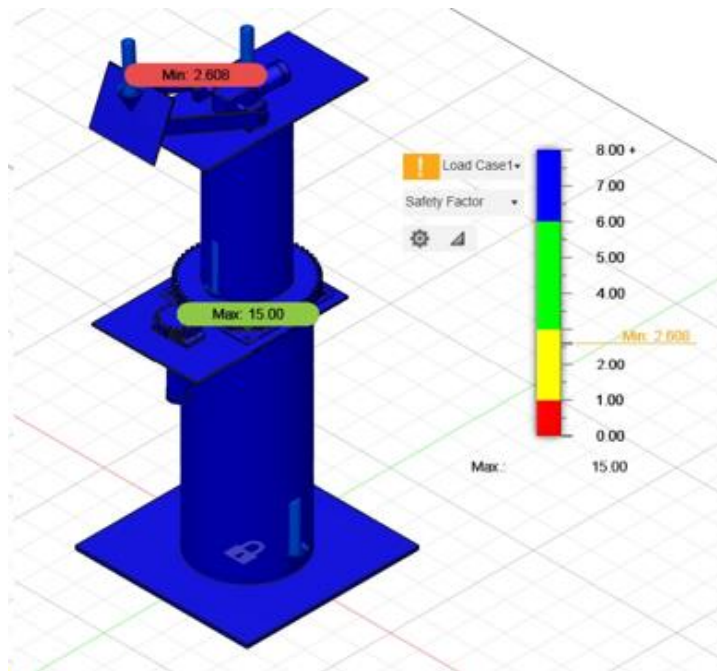


Figure 5. Analysis of factor of safety

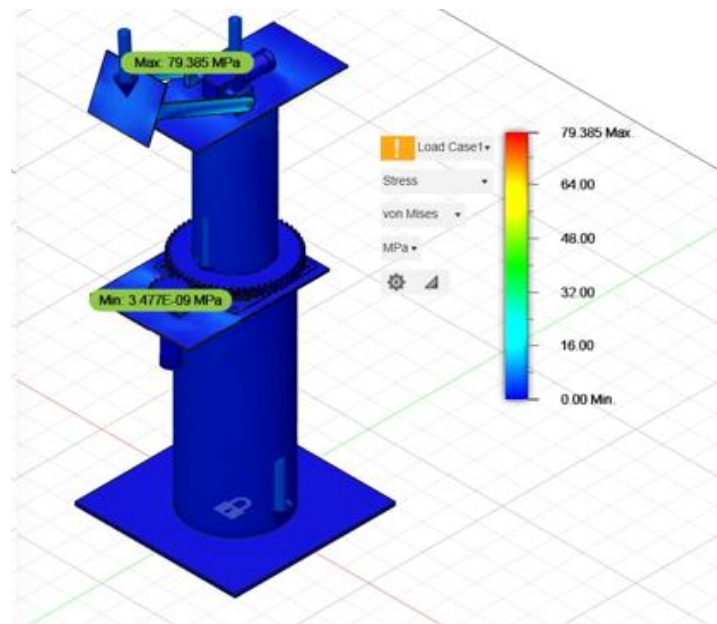


Figure 6. Analysis of factor of von Mises stresses

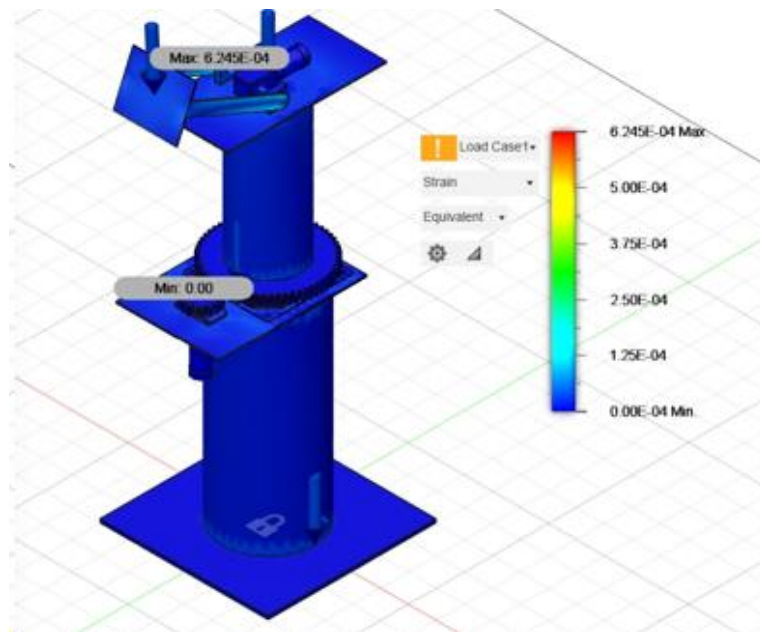


Figure 7. Analysis of strain

3.2.4. Structure Manufacturing and Assembly:

The basic design and parts have been explained in section 3.2.1. Consulting that explanation, further explanation of manufacturing and assembly of each part will be done in this section.

- 1) The next cylinder was cut according to dimensions, from a mild steel cylinder pipe that we purchased. This cutting was done through an arc cutting process. This cylinder is welded between two plates.
- 2) The second plate was laser cut. Holes were drilled in this plate through laser cutting as well. The holes in this plate were made to screw motor in as well as the axial bearing.
- 3) After this plate the assembly has an axial bearing. This bearing is press fit between two plates.
- 4) After the axial bearing, we have another plate. This plate was also laser cut and has holes to screw the bearing.
- 5) Above this we have a cylinder welded to the previous plate. This cylinder was cut in the same way as the previous cylinder.
- 6) Above the cylinder we have another welded plate that was laser cut. It

has holes for motor placement.

- 7) On the previous plate we have two small plates welded vertically with the previous plate. These two plates have holes each for the bearings of the tilt mechanism.
- 8) We then have the U-joint tilt mechanism. It was also laser cut and welded.
- 9) Finally, we have a solar panel mount which was laser cut and welded as well.
- 10) Each motor is mounted on the shaft with the help of set screws.
- 11) The pan portion has a spur gear mechanism connected with the first motor. These gears are placed by sliding them on the upper rotating cylinder.
- 12) The tilt mechanism is controlled directly by the motor attached to the U-mechanism.

3.2.5. Issues, Changes and Final Structure:

We made two major changes to our mechanical structure. The first change is the cylinder design instead of the square design. Our CAD model initially had a square structure between the plates instead of the cylinder. We changed this because cylindrical mild steel pipes were readily available in the market.

The other change was related to the issues we were facing in our tilt mechanism. The rotating torque of our motor is 100kg cm due to which at a moment arm of 10cm we could only bear 10 kg load while the weight of solar panel along with the panel mount is heavier than this weight. To solve this problem, we made a chain and sprocket mechanism with a U-shaped mount. This chain and sprocket mechanism was changed because it was not giving consistent results. We shifted to a simple mechanism in which the motor is connected directly to the shaft welded with the U-shaped mechanism. This was capable of bearing our loads efficiently.



Figure 8. Final Structure

3.3. Actuating System:

We need an actuating system to move the mechanism. We have used an actuating system based on DC motors converted into servos through the use of servo driver circuit and feedback mechanism.

3.3.1. Servo System:

A servo system is a closed-loop control system designed to accurately control the position, speed, or torque of a mechanical system, typically a motor. Servo systems are widely used in various applications where precise and dynamic motion control is needed, such as robotics, CNC machines, industrial automation, and aerospace systems. The closed-loop nature of servo systems allows for accurate and responsive control, making them essential in many modern industries.

In the context of dual-axis solar tracking systems, servos play a crucial role in maximizing the efficiency of solar panels by continuously orienting them towards the sun. Servos provide precise control over the orientation of solar panels in both azimuth (horizontal) and elevation (vertical) axes. This precision ensures that the panels are always positioned optimally to capture maximum sunlight throughout the day, maximizing energy generation. Servos offer fast

and dynamic response times, allowing them to quickly adapt to changes in solar position caused by factors such as cloud cover or changes in atmospheric conditions. This responsiveness helps maintain maximum energy output even in varying weather conditions.

3.3.2. Components of the Servo System:

Following are the main components of a Servo System:

A. Servo Motor:

The actuator responsible for providing the mechanical power and motion. Since we had to move a huge weight of the solar panel and its mount, we needed motors that could handle that kind of torque. We settled on using high torque geared motors (5840-31ZY worm gear DC motor). These motors utilize a gearbox to reduce the rpm while increasing the torque. We used 12V 11 RPM and 12V 7 RPM for the pan and tilt axis respectively. Both are capable of providing 100 kg cm of torque which is adequate for our application. These motors also use a self-locking worm gear system which means that if no power is applied, the shaft stays in its position even if any external force is applied. This is especially beneficial for our dual axis tracking system because we don't want the motors to be drawing power all the time. Motors are turned on periodically to change the orientation of the panel towards the sun and then turned off. This uses very less power since the motors are off most of the time. Self-locking feature is also helpful during windy weather conditions.



Figure 9. Servo motors

B. Feedback Device:

This component provides information about the motor's actual position, speed, or other relevant parameters back to the controller. Common types of feedback devices include encoders, resolvers, or potentiometers.

We are using an AS5600 magnetic encoder. This is an IC module that measures magnetic field. It provides a serial output and has 10 bit resolution for maximizing precision. A magnet is placed on the shaft of the motor and this encoder IC is placed at some distance to the magnet, The encoder continuously monitors the magnetic field and relays this information to the controller. This information can be used to accurately determine the position and speed of the motor shaft.

C. Controller:

The brain of the servo system. It processes input signals (commands) and compares them to the feedback from the motor. Based on the error between the desired and actual state, the controller generates control signals to adjust the motor's behaviour.

We are using atmega328p as the controller for this purpose. A PI controller is implemented in the atmega328p which brings the motor to the desired position quickly and accurately. The derivative controller is not used as the motor is moving at relatively low speeds.

For slow-moving motors, a PI controller is usually sufficient and beneficial because it provides a balance between simplicity and performance. It can

effectively eliminate steady-state error and is generally easier to tune compared to a PID controller, which includes the derivative component that may not be necessary for slow-moving systems.

D. H-Bridge:

An H-bridge was also used to control the dc motors. It was implemented using two BTS7960 half bridge ICs and a buffer in between to isolate the microcontroller.

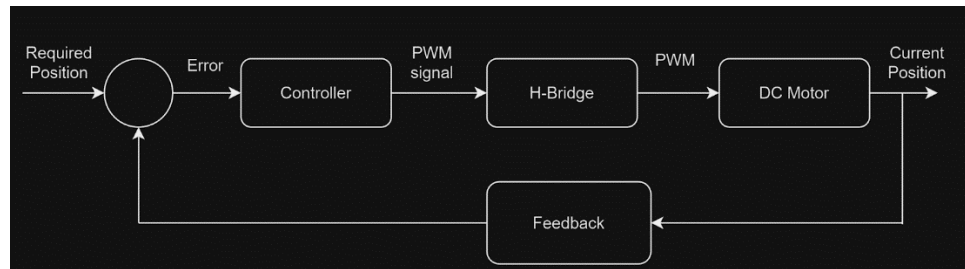


Figure 10. Block Diagram of actuating system

3.3.3. PCB Design for Motor Driver Circuit:

A PCB was designed keeping in view our requirements. The controller, power delivery system, H-bridge and the buffer ICs were part of the design. All of these components were fitted on a 60mm by 60mm 4-layer PCB board. Utilizing one layer as a ground plane for noise reductions, and one for VCC. The other two layers were used for routing. A JST connector was also placed to the encoder connections. The PCBs were then ordered from JLC PCB. All the components were soldered by hand. The PCB was then tested through various stages namely burning bootloader, uploading code, H-bridge testing, encoder testing. All stages were verified before the PCB was used in the project.

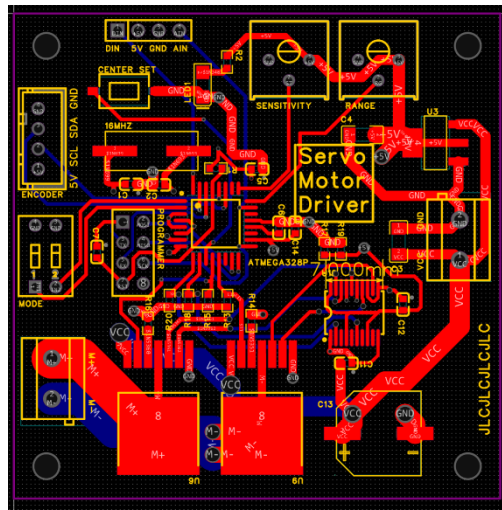


Figure 11. PCB design

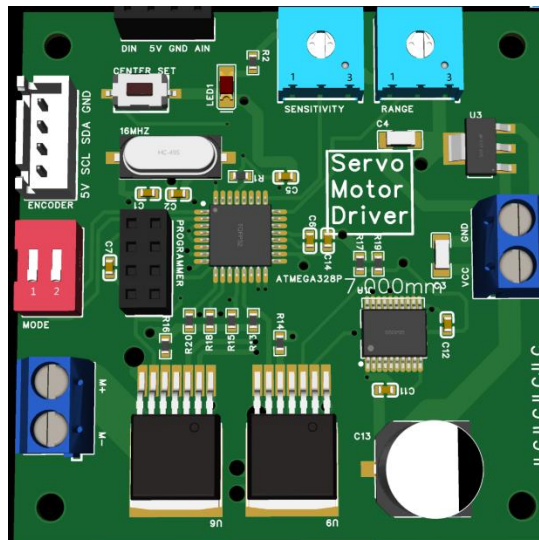


Figure 12. 3D PCB layout

3.3.4. Incorporation of Limit Switching in Actuating System:

Each servo also has two limit switches for the two extreme positions. The limit switches help in the calibration of both servos. The magnetic field of the magnet can drift over time, so the calibration is essential to provide accurate control over the direction of the motors. At startup, both motors move to their extreme positions, the encoder value at extreme positions is stored and mapped to a servo signal. A servo signal can be then used to control the position of the motor. The motor can now be moved anywhere in its range.

3.3.5. Control Logic for Actuators:

The controller continuously monitors the value of the encoder. At startup, the motor is moved to one extreme position, the extreme position is detected when a limit switch is pressed. The value of encoder at that point is stored. The motor is then moved in the other direction until the limit switch in that direction has been pressed indicating that the limit in that direction has been reached. The value of encoder at this point is also stored. This gives us an encoder range, which our motors can access and move anywhere between this range. The incoming servo signal from the master controller is then mapped to this range. A servo signal uses duty cycle to control the orientation of a servo. A duty cycle of 1ms usually means 0 degrees and a pulse width of 2ms means 180 degrees. The master controller can now control the orientation of the servos by varying the duty cycle of the servo signal. This is analogous to `Servo.write(Angle)` in Arduino. No movement is done if the input servo signal is out of this range.

3.3.6. Code for Servo Driver:

The code that we burned in the servo driver circuit is given in Annex A. In this code, after importing the necessary libraries, we define the pins where the related components are connected. We then defined a function “calibration” for calibrating the motors. In setup function, we set up the pins defined and calibrate the motors by calling the calibrate function. We then declare and define variables to be used in the PI control loop. In the loop function, we check the pulse width of the input signal, map it to our range of encoder values to the position in terms of encoder value, then implement some necessary checks to avoid this value exceeding our range of motor motion. We then move the motors in the direction required until we are at the encoder value mapped by the input signal. These steps are repeated.

As mentioned previously, the motors we are using act as servos due to the magnetic encoders mounted in front of the motors. We were facing an issue regarding this for our tilt motor due to the limitation of our mechanical structure. It was not possible to mount the encoder of the tilt motor in a way that it gave accurate readings. To solve this issue, we burned a different code in the motor

driver circuit of the tilt motor. The code is given in Annex B. This code works in such a way that after importing the necessary libraries, we define the pins where the related components are connected. In setup, we set the pins as either input or output accordingly. After this, we move the motor in both extremes till limit switches for calibration. We then stop the motor in the midpoint and await input signal. In the loop function, we continuously check for input pulse width. If the pulse width indicates an angle of greater than 90, we move the motor in the upward direction. If the pulse width indicates an angle of lower than 90, we move the motor in the downward direction. The speed of the motor is related to how much the input angle differs from 90. If the difference is large, the motor will move with high speed. If the input angle is 90, no action will be performed, and the motor will stay still.

3.4. Microcontroller:

We are using three microcontrollers in our project. One is the Atmega328P. The other two are Raspberry Pi 5 and Arduino UNO. We are using Atmega328P in the servo driver circuit used in the actuating mechanism of the structure. Raspberry Pi 5 is being used for data logging while active tracking is being done through Arduino UNO. Initially, we wanted to use Raspberry Pi 5 as our primary controller. We decided to work on Raspberry Pi 5 for a new experience and to learn and acquire practice and expertise on a controller we have not used before. However, keeping efficiency and power consumption in mind, which is the main aim of our project, we decided to achieve active tracking via Arduino UNO as it consumes much less power than Pi. Nevertheless, we have developed active tracking codes for both these controllers for a small-scale model of dual axis tracking. The codes for Arduino were refined for the actual model while the codes for Raspberry Pi 5 can assist in the future prospects of this project. Chronological tracking along with active tracking can make a hybrid dual axis solar tracking system. Raspberry Pi has much more computational power than Arduino microcontrollers for the algorithms of chronological tracking. Hence our project can be further modified, without any change in its structure, and actuating system while only adding and refining to its initial control algorithm. All these codes are provided in annexures.

3.5. Sensors:

Our project is the active dual axis solar tracking system. For this reason, it requires sensors that actively track the sun. Depending on the values of these sensors, the mechanism is moved so that it faces the sun. Along with the sensors used in active tracking, we also require sensors that will measure voltage and current generated so we can log this data as part of our project and make necessary calculations regarding the power generated by the solar panel system. The sensors used for active tracking are LDRs. The sensors for current and voltage measurement include ASC712 current sensing module and a self-fabricated voltage divider circuit.

3.5.1. ADS1115 Module with Sensors:

As mentioned previously, codes for the small-scale model were developed for both the controllers. Pi works on digital signals instead of analogue signals. Our sensors give analogue values. For this reason, we have used an ADC module i.e. ADS1115 with our sensors so they can be used effectively by both the controllers. ADS1115 is a 16-bit analogue to digital convertor with 4 channels and communicates through i2c. Because the module can run on supply voltages ranging from 2V to 5.5V, it can be used with any popular 3.3V and 5V microcontroller or processor, including the Arduino and Raspberry Pi. Up to four of these modules can be connected to the same I2C bus by configuring the module's I2C address to one of the four available addresses. Therefore, you may effectively increase the number of analogue inputs on an Arduino or other microcontroller by up to sixteen [20]. We are using a total of 2 of these modules. One is used with LDRs while the other is used with voltage and current sensors.

3.5.2. LDRs:

The basis for the operation of light-dependent resistors (LDRs) is photoconductivity. LDRs are frequently employed in circuits that need to detect the presence or amount of light. Their photosensitivity is precisely why they were designed [9]. As the name suggests, their resistance varies with light intensity. The resistance of LDRs decreases as the light intensity increases.

A. Placement of LDRs:

We had to decide whether to place our LDRs in a cross or square formation. We decided on the square placement as each top, bottom, left, and right values relating light intensity is then dependent on 2 LDRs each instead of 1. The LDRs are mounted on top of the panel in this formation with the aid of a cross-shaped piece as done in [11]. This sort of placement with the cross-shaped piece is also aided by an opaque plate, so that some LDRs are shadowed and the one that is illuminated is the one with most sunlight, hence giving better readings. For the case of cross-piece, the goal of it is also to cast a shadow on two or more LDRs if the cross is not pointing perpendicularly towards the sun [12].

B. LDRs and ADS1115 Module:

As mentioned before, one of the ADS1115 module is used with LDRs. ADS1115 has 4 channels, and we are using 4 LDRs. Hence our data from LDRs is easily converted by this module and utilized by the microcontroller.

C. Connections and Circuit Diagram:

- The resistance of LDRs varies with light intensity. Due to this, the voltage across the LDRs is dependent on the light intensity as well. For this reason, a voltage divider circuit is used. An LDR is connected in series with a resistor. One end of the LDR is connected with an external supply. The end of LDR common with one of the resistor legs, is connected to one channel of the ADC. The other leg of the resistor is connected to the -ve of the power supply and ground is made common with the microcontroller.

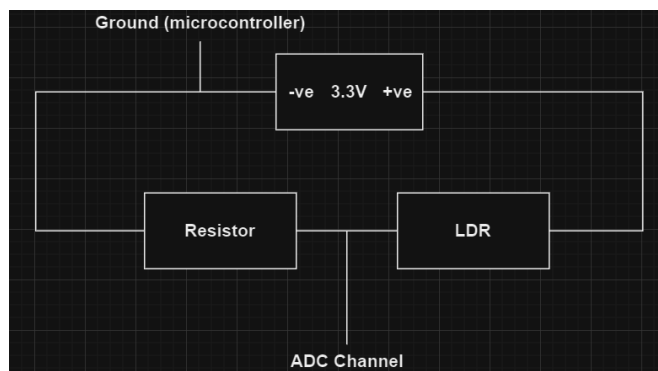


Figure 13. LDR connections

- ADS1115 has 10 pins. 4 are channel pins, 1 is ALRT, 1 is ADDR, 1 is SDA, 1 is SCL. Then we have GND and Vdd. Vdd pin and GND pin is connected to 3.3V and GND pin of microcontroller respectively. SCL and SDA of this module and microcontroller are connected. For LDRs connection with this module the ADDR pin is either free or connected to Vdd. This assigns the address of the module. If this module ADDR pin is free then the module with current and voltage sensors will have ADDR connected to Vdd and vice versa. The four channels are then connected with the LDRs such that Top Right (TR) is connected at A0, Bottom Right (BR) is connected at A1, Top Left (TL) is connected at A2, and Bottom Left (BL) is connected at A3.

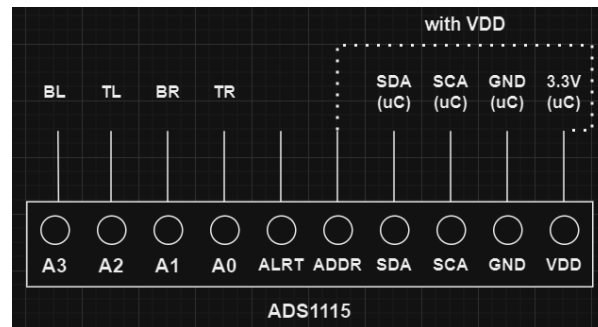


Figure 14. ADS1115 connections LDR module

3.5.3. Current and Voltage Sensors:

We have used current and voltage sensors with a data logging module so that we can get the values of current, and voltage generated by the solar panel and hence draw conclusions and results regarding the power generated by the static panel and dual axis tracker.

A. Current Sensor:

We have used ASC712 30A range current sensing module. It is a hall effect sensor. Current flows through the onboard hall sensor circuit in its IC. The hall effect sensor uses its ability to generate a magnetic field to detect incoming current. The voltage produced by the hall effect sensor, upon detection, is proportionate to its magnetic field and is subsequently utilized for measuring the current [21].

B. Voltage Sensor:

We have made our own voltage sensor. We have used the basic concepts of voltage divider for this purpose. Our voltage sensor is a voltage divider circuit which measures voltage across the resistor to compute the source voltage.

C. Connection and Circuit Diagram:

- External source, the current of which has to be measured is connected to the ports on one side of the sensor. The other side of the sensor has VCC, OUT and GND pins. VCC is connected to any external supply, the GND is made common with -ve of this supply and GND of microcontroller. The OUT pin is connected to A1 channel of the second ADS1115 module.

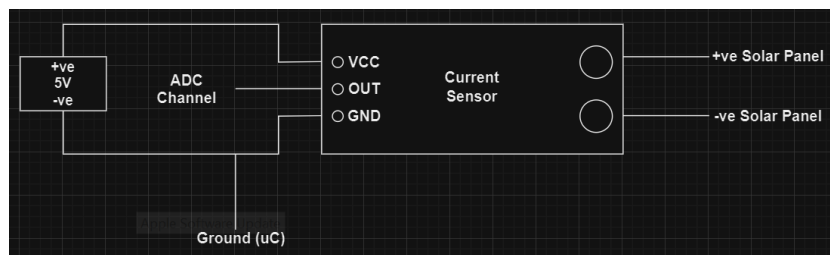


Figure 15. Current sensor connections

- Voltage divider circuit for voltage measurement is shown in the following figure:

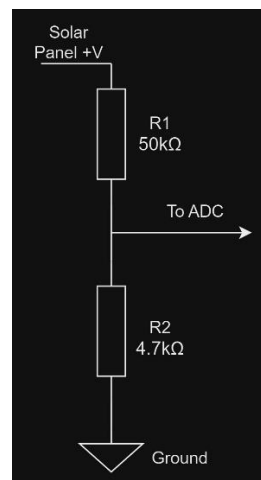


Figure 16. Voltage sensor connections

- Connections of the ADS1115 module for these sensors is similar to the ones explained in the case of LDRs.

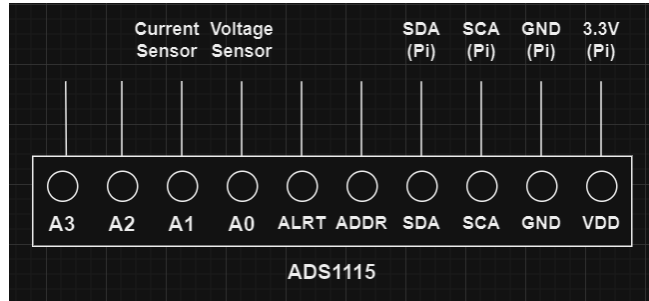


Figure 17. ADS1115 connections current and voltage sensors

3.6. Data Logging Module:

For the demonstration of results in this project, it is necessary to calculate the power generated with a static solar panel and power generated with a solar panel placed on a dual axis solar tracking system. For this purpose, we require a circuit module that calculates short circuit current and open circuit voltage of the solar panel. We designed and programmed a circuit that calculates the open circuit voltage of the solar panel. After this the circuit is closed through a MOSFET acting as a switch. This closing is controlled by Raspberry Pi. Once the circuit is closed the current is measured and then the circuit is opened again.

The electrical circuit consists of two branches parallel to each other and parallel to the solar panel. One branch has the voltage sensor attached to it. The other branch has the current sensor attached to it in series with a MOSFET and optocoupler. The gate of the MOSFET is attached to a GPIO pin of Raspberry Pi. When this pin is powered, a voltage acts on the gate, closing the circuit. Optocoupler is used to control the MOSFET while isolating the Raspberry Pi.

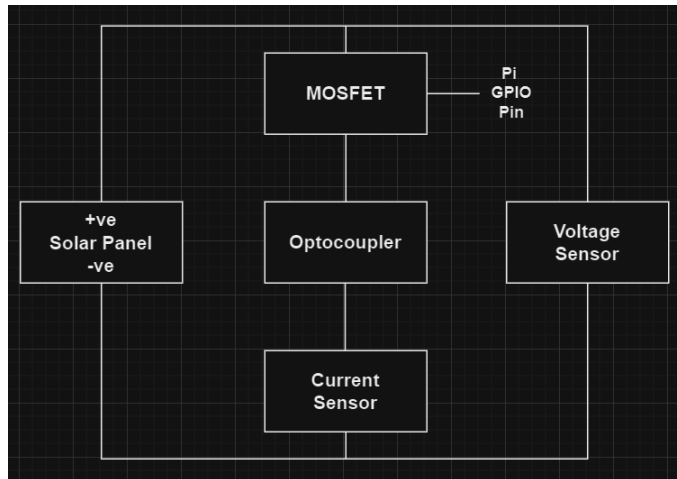


Figure 18. Connections of data logging module

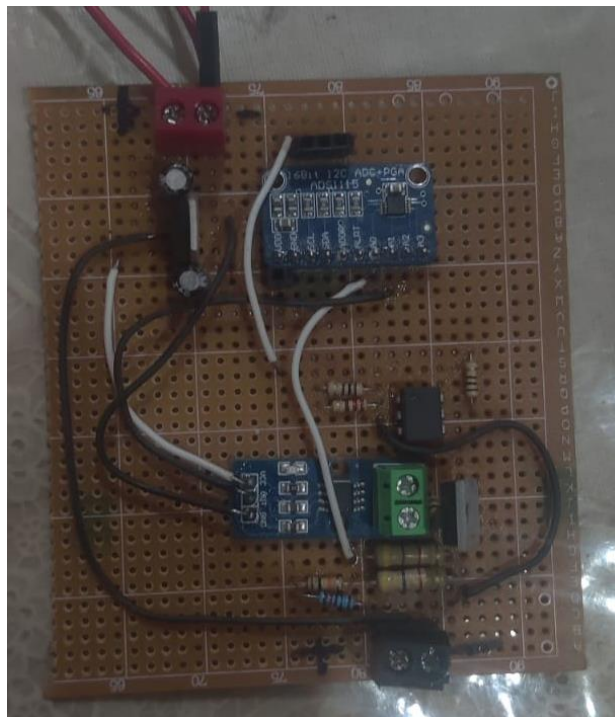


Figure 19. Data logging module

The code for acquisition of data using this data logging circuit is explained ahead in the section of control algorithm.

3.7. Implementation of the Control Mechanism:

The final code used to control the mechanism was developed through a series of steps. We initially started writing a code on Arduino. The reason for this was to test the basic logics of our code, test the sensors, and work on a controller we are more familiar with

in order to debug errors and identify issues easily. These initial codes were written for a small-scale model. We also developed the codes for dual axis active tracking for this model in Raspberry Pi 5.

3.7.1. Small-Scale Model:

We made a small-scale model of a dual axis tracker to test the codes being written. This model was made from cardboard. It had a cardboard base with cross-shaped cardboard plates glued to the base. The LDRs were connected in square formation between these plates. We also made a temporary breadboard circuit to connect the LDRs with Arduino through ADS1115. A 9g servo motor was attached to the base of the model at the bottom for pan movement. Another 9g servo motor was attached to one side for tilt motion. The servo motors were also connected with Arduino. We tested all the codes of Arduino and later, of Raspberry Pi, on this small-scale model before testing on the actual motors of our mechanism and the complete structure. We did this so that the process of debugging would be easier. Once you are certain that the code is fine, you can solve the problems that arise with a different and better outlook.

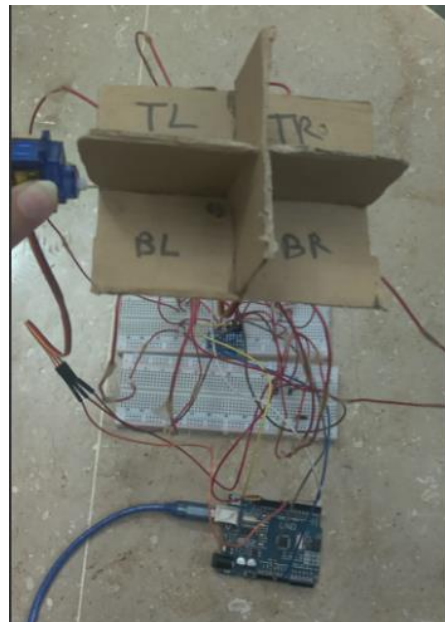


Figure 20. Small-scale dual axis tracker module

3.7.2. Initial Codes for LDRs, ADS1115 and Servos for Arduino:

We wrote the first pieces of code to get the values of LDRs through the ADS1115. For this, we had to get familiar with the ADS1115 Adafruit library. The code that shows how we got these values of LDRs is attached in Annex C. First, we create an ADC object. Next, we assign names to each channel. The value from each channel is taken through readADC_SingleEnded() function. Once values from each channel were taken; Top Right (TR), Bottom Right (BR), Top Left (TL), and Bottom Left (BL) LDRs, Top, Bottom, Left, Right were calculated through averages of LDRs present at each location. For example, $Top = (TR+TL)/2$ and so on. These values were then displayed on the serial monitor. We then used these values to identify the difference between Top and Bottom and Left and Right LDR values in order to move the motors in appropriate direction to minimize these differences hence align with the highest light intensity.

We also wrote and checked basic codes using the servo libraries to test our 9g servo motors. We identified the points of 0 and 180 degree for each servo as we have to either increment or decrement the angles to get to our desired position while active tracking.

3.7.3. Arduino Code for Active Pan Axis Tracking:

After acquiring the LDR values, we started working on a code that actively tracks the light intensity in the pan axis direction. The code has been attached in annex D. In this code we created a function that reads the LDR values from each channel of the ADC, calculates the Top, Bottom, Left, Right values as explained before. It finally calculates the difference between Right and Left LDR values. In the main loop we first call this function and check if a difference of more than 500 exists between these values. This threshold of difference can be changed according to our needs. If such a difference exists it means light intensity is greater towards the right side. We introduce a while loop which functions as long as the difference is greater than 500. In this loop we change the pan servo angle in such a way that our module pans towards the right side. We keep checking the difference value continuously by calling the function and

then changing the angle until the difference condition is met. Similarly, we check if a difference of less than -500 exists between these values. This threshold of difference can be changed according to our needs. If such a difference exists it means light intensity is greater towards the left side. We introduce a while loop which functions as long as the difference is less than -500. In this loop we change the pan servo angle in such a way that our module pans towards the left side. We keep checking the difference value continuously by calling the function and then changing the angle until the difference condition is met.

We tested this code on our small-scale module and made it track a flashlight. It worked effectively so we proceeded to incorporate tilt axis in it as well.

3.7.4. Arduino Code for Active Dual Axis Tracking:

After being satisfied with the code for pan axis tracking, we introduced similar functions for tilt axis tracking. The complete code is attached in Annex E. We created another function to check the difference between Top and Bottom LDR values. In this function we read the LDR values from each channel of the ADC, calculate the Top, Bottom, Left, Right values as explained before. Finally, we calculate the difference between Top and Bottom LDR values. In the main loop, after the pan tracking code, we call this function and check if a difference of more than 500 exists between these values. If such a difference exists it means light intensity is greater towards the top. We introduce a while loop which functions as long as the difference is greater than 500. In this loop we change the tilt servo angle in such a way that our module tilts upwards. We keep checking the difference value continuously by calling the function and then changing the angle until the difference condition is met. Similarly, we check if a difference of less than -500 exists between these values. If such a difference exists it means light intensity is greater towards the bottom. We introduce a while loop which functions as long as the difference is less than -500. In this loop we change the tilt servo angle in such a way that our module tilts down. We keep checking the difference value continuously by calling the function and then changing the angle until the difference condition is met.

We tested the complete code on our small-scale model and made it track a flashlight. It worked efficiently. We were satisfied with the results of our code and knew that the logics were sound. The code was now ready to be converted according to Raspberry Pi.

3.7.5. Coding in Raspberry Pi:

Once satisfied with the results of Arduino code, we moved towards Raspberry Pi. The basic logic of the code was the same, but it had to be converted according to the libraries available in Raspberry Pi and the coding language of Pi i.e. Python. The initial process of coding was similar to that of Arduino. We learned how to get LDR values from ADS1115 using the Adafruit library of Raspberry Pi. We then experimented on the code for servo control.

3.7.6. Code for Servo Control in Pi:

The major problem we faced while coding in Raspberry Pi was related to servo control. The libraries of Arduino allow you to input an angle and it goes to that position on its own. You can also read the current angle and make necessary increments or decrements. The servo control in Raspberry Pi is not as straight forward. We first tried using the angular servo library. The problem with this was that it would not allow us to increment small angles to adjust our servo. The least possible increment we were getting through it was 15 degrees. This was not acceptable as we require small degree increment for more efficient tracking.

To solve this problem, we created our own function to convert duty cycle to angle and vice versa. For this purpose, we used the RPi.GPIO library to access the GPIO pins of Raspberry Pi. We know that maximum angle (180) corresponds to maximum duty cycle and minimum angle (0) corresponds to minimum duty cycle. Hence, we used the concept of interpolation to make an equation [22]:

$$(\text{duty cycle} - \text{minimum duty cycle}) / (\text{maximum duty cycle} - \text{minimum duty cycle}) = (\text{angle} - 0) / (180 - 0)$$

$$\text{duty cycle} = ((\text{angle} / 180) * (\text{maximum duty cycle} - \text{minimum duty cycle})) + \text{minimum duty cycle}$$

Similarly,

$$\text{angle} = ((\text{duty cycle} - \text{minimum duty cycle}) / (\text{maximum duty cycle} - \text{minimum duty cycle})) * 180$$

These equations are used in the functions to calculate duty cycle from angle and angle from duty cycle. For servo control we need to know what the current angle of the motor is. To find this we first use an in-built function of i.e. `_dc`. This function gives the current duty cycle. We then call our function of duty cycle to angle to get the current angle. The angle is changed according to our needs. This new angle is converted to the required duty cycle using our function of angle to duty cycle. We then use another in-built function of `ChangeDutyCycle` to input the new duty cycle to get our required angle.

This code of servo control is attached in Annex F. We tested this code on the servos of our small-scale model. We also tested this code, along with the servo driver circuit, on the actual motors of our mechanism to test the servo control.

3.7.7. Code for Active Dual Axis Tracking in Raspberry Pi:

Once the problem of servo control was fixed, we incorporated the code of active tracking with servo control code. In the active tracking code, the servo movement is dependent on the LDR values. This code was similar to the code written for Arduino. We had already done the basic LDR data acquisition in Pi, we had written a code for servo control, the only thing left was to change the functions of Right and Left, and Top and Bottom difference calculations. We wrote these functions and constructed the code attached in Annex G. As mentioned before, this code is similar to the arduino code with an additional two functions related to servo control. Hence, we have 4 functions. Two functions calculate differences while the other two are the duty cycle to angle and vice versa functions. We have a 'while TRUE' function that is a main loop. In this function we first check the left and right LDR difference values. If a difference of more than 500 exists between these values, we introduce a while loop which functions as long as the difference is greater than 500. In this loop we change the pan servo angle in such a way that our module pans towards the right side. We keep checking the difference value continuously by calling the function and

then changing the angle until the difference condition is met. This change of angle is the portion of the code that differs from the Arduino code. This is because this angle change is done through the functions of duty cycle to angle and angle to duty. These angle changes are explained in the previous heading. Similarly, we check if a difference of less than -500 exists between these right and left values. If such a difference exists it means light intensity is greater towards the left side. We introduce a while loop which functions as long as the difference is less than -500. In this loop we change the pan servo angle in such a way that our module pans towards the left side. We keep checking the difference value continuously by calling the function and then changing the angle until the difference condition is met.

After this we check the difference between top and bottom LDR values. If top is greater the mechanism tilts up. If top is less, the mechanism tilts down. The code works in the similar way as explained above.

We tested this code on our small-scale module and made it track a flashlight. This code also worked effectively.

3.7.8. Code for Data Acquisition and Logging:

As mentioned in the data logging module heading, we had to measure a short circuit current and an open circuit voltage. For this reason, we had to make our circuit short for a while. This was controlled by the Raspberry Pi. In addition to this, we had to gain the data from the sensors and save the data. We wrote a code for data logging attached in Annex H. In this code we first defined the channels of the ADS1115. We got voltage from channel 0 and current from channel 1. We first measure voltage simply by reading the values from the ADS1115 module. We then generate a high output at the Raspberry Pi pin connected to the gate of the MOSFET in order to close the circuit. We then calculate the current and after that open the circuit by sending low to the previous pin. Since the current sensor outputs a voltage corresponding to a current, we change the voltage reading to current accordingly. The data is written in a local file and sent to google script as well for us to access real time data logging.

We initially faced an issue in our data logging module. The module would not output correct current values. This was solved by increasing the time of short circuit so that the sensor can read data while the circuit is short. Closing the circuit for a lesser time was resulting in an error since the circuit would open before taking the current reading.

This data logging code was testing on the solar panel itself. We acquired the results of power generation by the static solar panel and dual axis tracker using this code.

We wrote a command on Pi's crontab to run this program as soon as the Pi powers on. Hence, you do not need a display attached to run your command. We did this by typing "sudo crontab -e" on the terminal, then "@reboot python3 [path of program] &". After this we save and exit and our program is ready to run at startup.

3.7.9. Final Code:

As mentioned before, initially we wanted to achieve active tracking through Raspberry Pi. Considering the power consumed by the controllers and our aim to increase efficiency as much as possible, we finally decided to accomplish active dual axis solar tracking through the use of Arduino UNO. For this purpose, the Arduino codes for the small-scale model were refined to work on our actual system. They were also refined according to the solar tracking instead of flashlight testing. We had to change threshold values accordingly for this. The final code of our dual axis solar tracker is attached in Annex I.

As mentioned in the actuating portion of the thesis, the motors we are using act as servos due to the magnetic encoders mounted in front of the motors. We were facing an issue regarding this for our tilt motor due to the limitation of our mechanical structure. It was not possible to mount the encoder of the tilt motor in a way that it gave accurate readings. To solve this issue, we burned a different code in the motor driver circuit of the tilt motor. That code works in such a way that it stops when the given input angle is 90. It moves upwards when it is greater than 90 and moves downwards when it is less than 90. The speed of moving up and down depends on how much more or less the input angle is from

90 respectively. The servo driver code for the pan motor, however, works as any other servo. For this reason, we had to adjust the tilt portion of our Arduino code, so it works according to the code in the tilt servo driver.

The final code attached in Annex I resembles the code of active dual axis tracking in small-scale model attached in Annex E. We made some changes regarding the thresholds and tilt movement. The functions of calculating differences are slightly changed in the sense that we defined global variables to be used.

We also added watchdog timer and EEPROM library in this code. the reason for this was to reset our Arduino in case it hangs. The current angle is stored in EEPROM so even if the Arduino hangs and resets, it stays at the angle it was at before resetting. Due to this we do not lose any tracking progress and the mechanism continues to track from where it left in case of hanging and resetting.

The logic of the tilt portion of the code is similar to the pan motion. The only difference is in incorporating the motion of the motor in accordance with the servo driver code. The upward motion is achieved by giving an input angle of 150 instead of angle increments used in pan tracking. This upward motion is followed by a delay and rechecking of LDR values. Hence, the motor moves upwards for a small amount of time, achieving little angle increments, and then checks the LDR values to identify if it should repeat this or stop. Same logic is used to move downwards but with an input angle of 40.

The loop for tracking repeats after every 15 minutes so the mechanism adjusts according to the sunlight after every 15 minutes.

The flowchart explaining our final code is given below:

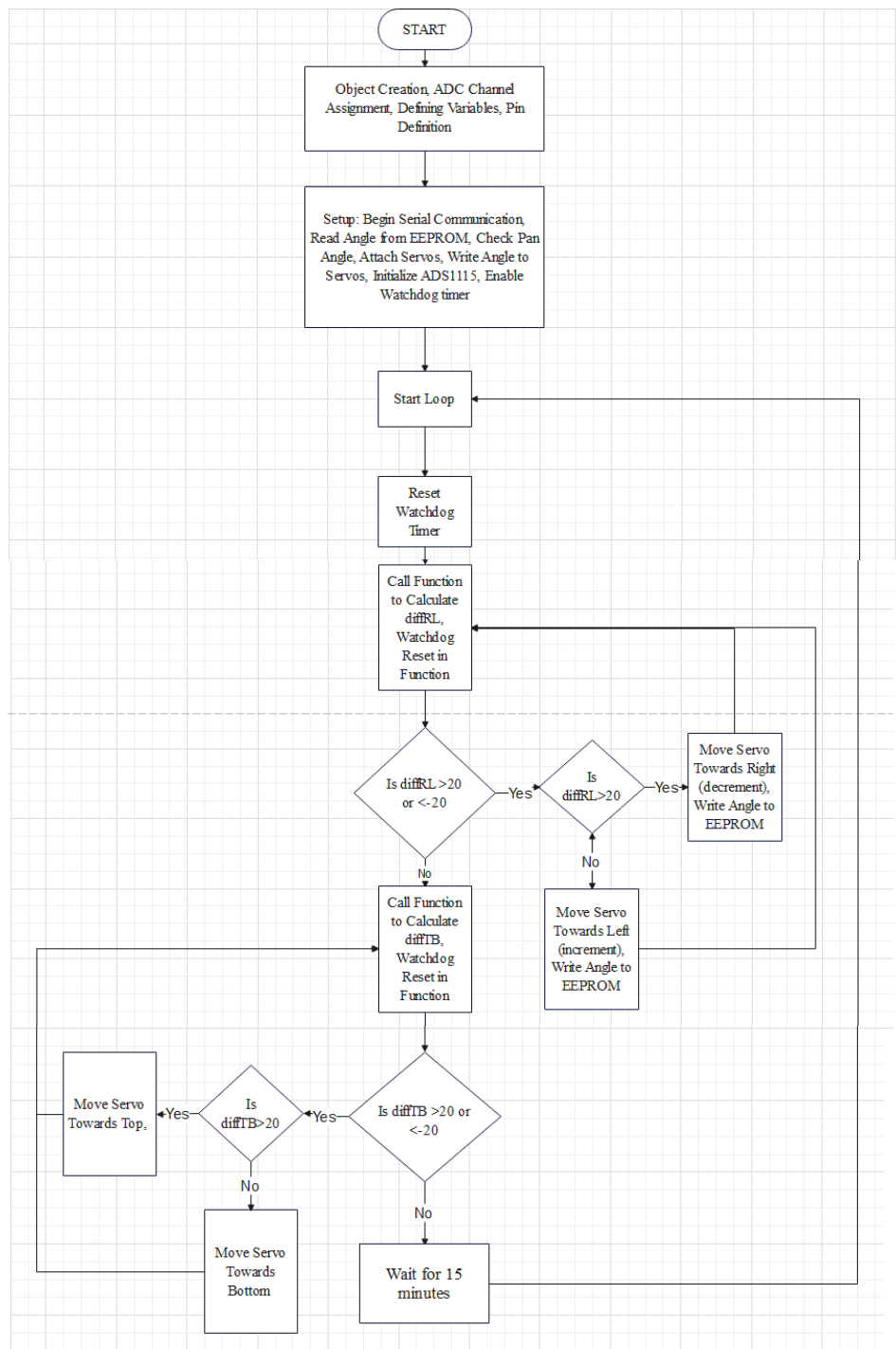
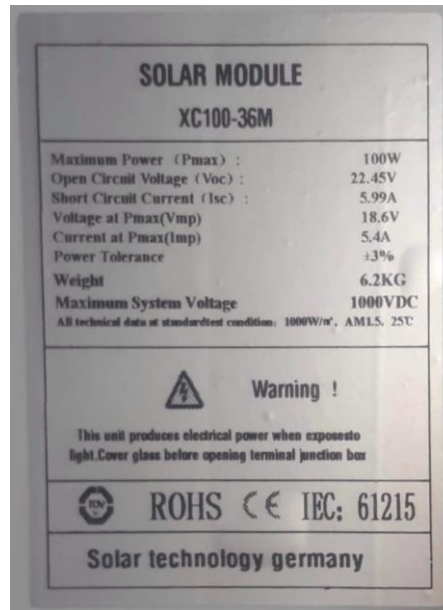


Figure 21. Flowchart of code

3.8. Solar Panel:

We are using a 4 by 2.5ft solar panel. The weight of the panel is 6.2 kg. The panel is rated for a maximum power of 100W. Its open circuit voltage is 22.45V and short circuit current is 5.99A.





SOLAR MODULE	
XC100-36M	
Maximum Power (Pmax) :	100W
Open Circuit Voltage (Voc) :	22.45V
Short Circuit Current (Isc) :	5.99A
Voltage at Pmax(Vmp)	18.6V
Current at Pmax(Imp)	5.4A
Power Tolerance	±3%
Weight	6.2KG
Maximum System Voltage	1000VDC
All technical data at standard test condition: 1000W/m ² , AM1.5, 25°C	
 Warning !	
This unit produces electrical power when exposed to light. Cover glass before opening terminal junction box	
 ROHS  IEC: 61215	
Solar technology germany	

Figure 22. Solar panel specifications

3.9. Electronics:

Basic electronics and circuit diagrams have been explained where needed previously. This section covers over all power supply to the components. We have a 7.2AH 12V lead acid battery. This is used as an external supply to all components. We have two 5A buck convertors. One is used to power Arduino at 5V the other is used to power ADS module at 3.3V. Motors are connected to battery via the servo driver.

Chapter 4 -EXPERIMENTAL RESULTS AND ANALYSIS

4.1. Overview:

Chapter 4 presents the experimental results and analysis of power generated by static solar panel system and dual axis solar tracking system. We acquired voltage and current generated by each system throughout the day. Power is calculated through this data. We analyse which system generates more power. Along with this, we also consider how much power is consumed by each component of our dual axis solar tracking mechanism. These calculations are then used to draw the results and conclusions about the efficiency of our system.

4.2. Data Collection and Experimental Setup:

The system is powered on by turning on the battery and then turning on the Arduino. Data logging circuit is connected to Raspberry Pi 5. Solar panels are connected to this circuit and Pi is powered on. This starts our system. The motors first calibrate and then start tracking. The datalogging also starts.

4.3. Data Acquired from Static Solar Panel System:

Data of the static solar panel was logged from 12pm to 8pm. Voltage and Current were calculated after every minute. From that we calculated power generated at each minute. We calculated average power for each minute and multiplied it with 60 to get an approximate of energy produced within that minute. We then converted this into watt-hour. Hence, the total estimated energy produced for each hour is shown in the table below:

TABLE 1. DATA ACQUIRED BY STATIC SOLAR PANEL

Time	Total Energy (Ws)
12pm to 1pm	11.17017
1pm to 2pm	73.16799
2pm to 3pm	72.46134
3pm to 4pm	57.2074
4pm to 5pm	43.89767
5pm to 6pm	16.96094
6pm to 7pm	4.053667
7pm to 8pm	0.346667
	$\Sigma = 279.27$

This shows that the estimated total energy produced from 12pm to 8pm by a static panel is 279.27Wh.

4.4. Data Acquired from Dual Axis Solar Panel System:

Data of the dual axis solar tracking panel was logged from 12pm to 8pm. Voltage and Current were calculated after every minute. From that we calculated power generated at each minute. We calculated average power for each minute and multiplied it with 60 to get an approximate of energy produced within that minute. We then converted this into watt-hour. Hence, the total estimated energy produced for each hour is shown in the table below:

TABLE 2. DATA ACQUIRED BY DUAL AXIS TRACKER PANEL

Time	Total Energy (Ws)
12pm to 1pm	13.68831
1pm to 2pm	82.76008
2pm to 3pm	81.60767
3pm to 4pm	78.56311
4pm to 5pm	72.24367
5pm to 6pm	56.66819
6pm to 7pm	25.28995
7pm to 8pm	0.924611
	$\Sigma = 411.75$

This shows that the estimated total energy produced from 12pm to 8pm by a dual axis solar tracking panel is 411.75Wh.

4.5. Power Consumed by Dual Axis Solar Tracking System:

The tracking system was powered by a fully charged battery. After tracking for the day, the battery was charged again. The charger showed that it had to provide 1550mAH to charge battery to maximum. This shows that our complete system consumed 1550mAH. This is equivalent to 18.6Wh since our battery is of 12V. Hence, the total energy consumed by our dual axis solar tracking system from 12pm to 8pm is 18.6Wh.

4.6. Analysis of Results:

The results are analysed in multiple ways. We have compared the energy produced by both the systems. After that we have calculated the efficiency of our dual axis solar tracking system. Considering, our system consumes power in actuation, we have then compared the net energies available of both systems.

4.6.1. Comparison of Total Energy Production:

Data regarding total estimated energy produced by each panel is given table 3. The percent gain is calculated by the following formula:

$$\% \text{ Gain} = \left(\frac{\text{final} - \text{initial}}{\text{initial}} \right) * 100$$

In this formula final is the energy produced by dual axis tracking panel and initial is the energy produced by static panel.

TABLE 3. PERCENT GAIN IN ENERGY PRODUCTION

Time	Total Energy Static (Wh)	Total Energy Dual Axis (Wh)	Percent Gain per Hour
12pm to 1pm	11.17017	13.68831	22.5%
1pm to 2pm	73.16799	82.76008	13.12%
2pm to 3pm	72.46134	81.60767	12.6%
3pm to 4pm	57.2074	78.56311	37.33%
4pm to 5pm	43.89767	72.24367	64.57%
5pm to 6pm	16.96094	56.66819	234.1%
6pm to 7pm	4.053667	25.28995	524%
7pm to 8pm	0.346667	0.924611	167%
	$\Sigma = 279.27$	$\Sigma = 411.75$	47.44%

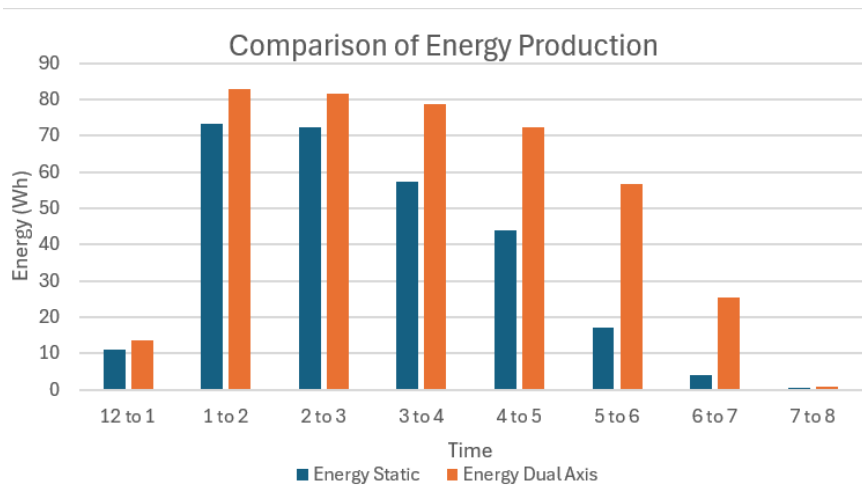


Figure 23. Comparison of energy production by both system

This table shows gain in the total energy produced for each hour by dual axis tracking panel over static panel. Since the energy produced by the static panel, from 12pm to 8pm, is 279.27Wh and that produced by dual axis tracking panel for the same amount of time is 411.75Wh; the total gain in the energy produced by dual axis tracking system over static system is 47.44%. From the graphs we can see that the major difference in energy production occurs during the non-peak hours. This is because dual axis tracking panels can align according to the sun's position and maximize the energy output.

4.6.2. Efficiency of Dual Axis Solar Tracking System:

Efficiency of our dual axis solar tracking system can be calculated through energy consumption and energy generation. The energy consumed by the system from 12pm to 8pm is 18.6Wh while the energy produced by our solar panel is 411.75Wh. Hence, efficiency of our system is calculated through the formula:

$$\text{Efficiency} = (\text{net energy output} \div \text{energy generated}) \times 100$$

Net energy output is gained by subtracting 18.6 from 411.75 i.e. 393.15Wh. Using this in the efficiency formula we get the result that our system is 95.48% efficient.

4.6.3. Comparison of Net Energy Available:

In addition to the previous results, we see that the net energy available by our dual axis solar tracking system is 393.15Wh after we subtract the energy consumed by the actuating system. This energy is still 40.78% more than the total energy generated by a static panel.

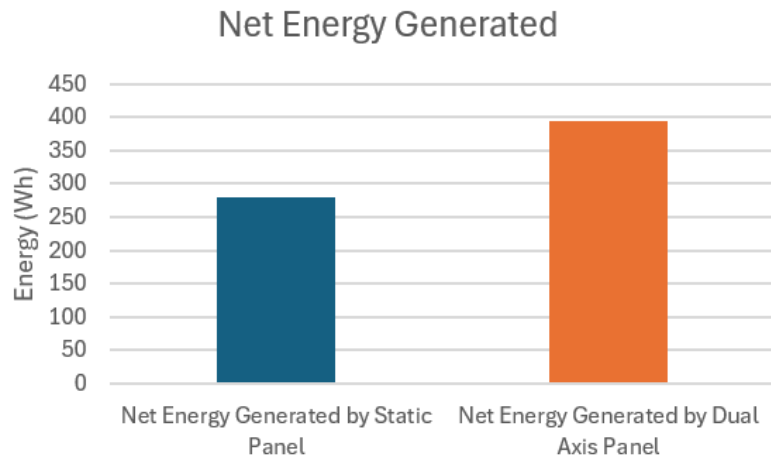


Figure 24. Comparison of final energy available by both system

Chapter 5 -CONCLUSION

5.1. Overview:

In order to maximize the efficiency of solar energy collection, we have successfully designed and built a dual-axis solar tracking system in this thesis. The project required the integration of hardware, the implementation of control mechanism, and a thorough investigation of various tracking systems. The main accomplishments are outlined in this conclusion, along with some suggestions for future work to improve the system even further.

5.2. Summary of Achievements:

A. Design and Implementation:

- Created a sturdy, dual-axis solar tracking system that can precisely track the sun during daytime.
- Employed an array of sensors, microcontrollers, and actuators to accomplish exact solar panel movement and positioning.

B. Efficiency Improvement:

- Compared the performance of the static solar panels and the dual axis tracking system.
- Showed a notable rise in energy capture, and the tracking system indicated an estimated 40.78% gain in energy production.
- Showed that our system is 95.48% efficient in regards to the energy produced and consumed by our system

C. Control Mechanism:

- Implemented a reliable control mechanism that modifies the panel's orientation in response to LDR data based on the sun's position in real time.

D. Hardware Integration:

- Successfully combined key hardware elements, such as motors, light sensors, voltage and current sensors, and a microcontroller, to guarantee smooth operation and communication.
- Through thorough testing and quality checks, the system's durability and stability were guaranteed.

E. Cost-Effectiveness:

- Optimized the choice of components and materials to get a cost-effective solution without sacrificing performance.

5.3. Future Recommendations:

A. Optimization of Control Algorithm:

- Investigate how to include cutting-edge machine learning methods to forecast solar patterns and improve tracking precision.
- Examine adaptive algorithms that can dynamically adapt to weather variations and seasonal fluctuations.
- Incorporate chronological tracking algorithms to the current module in order to make a hybrid dual axis solar tracking system.

B. Integration with Smart Grids:

- Provide the solar tracking system the ability to communicate with smart grid technologies to enhance energy storage and distribution.
- Incorporate functionalities that provide remote supervision and management through IoT (Internet of Things) technologies.

C. Enhanced Durability and Maintenance:

- Investigate materials and designs that improve the durability of the system in harsh environmental conditions.
- Create a thorough maintenance plan with instructions to guarantee efficiency and reliability over the long run.

5.4. Final Thoughts:

The dual-axis solar tracking system that was created for this project is a noteworthy development in the field of solar energy technology. This system can make a significant contribution to renewable energy programs by improving the reliability and efficiency of solar energy collection. Further optimization and development of this system's capabilities can be achieved by future research and development activities, guided by the recommendations offered. This will promote more widespread adoption and integration of solar energy into the global energy landscape.

REFERENCES

- [1] Mohd Rizwan Sirajuddin Shaikh, Santosh B. Waghmare, Suvarna Shankar Labade, Pooja Vittal Fuke, Anil Tekale, "A Review Paper on Electricity Generation from," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 2017.
- [2] Murat Kacira, Mehmet Simsek, Yunus Babur, Sedat Demirkol, "Determining optimum tilt angles and orientations of photovoltaic panels in Sanliurfa, Turkey," *Renewable Energy*, vol. 29, no. 8, pp. 1265-1275, 2004.
- [3] Hussain Mohammad Fahad, Aporajita Islam, Mohaimenul Islam, Md. Fahim Hasan, Wasima Fariha Brishty, Md. Mosaddequr Rahman, "Comparative Analysis of Dual and Single Axis Solar Tracking System Considering Cloud Cover," in *2019 International Conference on Energy and Power Engineering (ICEPE)*, Dhaka, Bangladesh, 03 June 2019.
- [4] B. B. Priti Debbarma, "A Review on Solar Tracking System and Their Classification," *IJRAR*, p. 1, 2019.
- [5] M. C. Adrian C., "Azimuth-Altitude Dual Axis Solar Tracker," WORCESTER, 2010.
- [6] H. K. A. J. A. M. H. A. K. S. A. Mousazadeh, "A review of principle and sun-tracking methods for maximizing solar systems output," *Renewable and Sustainable Energy Reviews*, vol. XIII, pp. 1800-1818, 2009.
- [7] V. M. A. Antonio L. Luque, *Concentrator Photovoltaic*, 2007.
- [8] M. Ghassoul, "A dual solar tracking system based on a light to frequency converter using a microcontroller," *Fuel Communications*, vol. 6, March, 2021.
- [9] "What is Light Dependent Resistor : Circuit & Its Working," [Online]. Available: <https://www.elprocus.com/ldr-light-dependent-resistor-circuit-and-working/>.
- [10] Vikash Kumara, Sanjeev Kumar Raghuwanshi, "Design and Development of Dual Axis Solar Panel Tracking System for," in *International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM-2019)*, Jaipur, India, February, 2019.
- [11] Arunachalam Sundaram, Hassan Zuhair Al Garni, "A Smart Garden System with a Dual-Axis Solar Tracker," *International Journal of Advanced Science and Technology*, vol. 29, no. 8, pp. 1390-1397, 2020.
- [12] Anish Sarla, Sai Charan Reddy Dandu, "Sun Tracking System," Karlskrona, Sweden, June 2022.
- [13] Sanzidur Rahman, Rashid Ahammed Ferdous, Mohammad Abdul Mannan, Mahir Asif Mohammed, "Design & Implementation of a Dual Axis Solar Tracking

- System,” *American Academic & Scholarly Research Journal* , vol. 5, no. 1, Jan, 2013.
- [14] Rashid Ahammed Ferdaus, Mahir Asif Mohammed, Sanzidur Rahman, Sayedus Salehin, Mohammad Abdul Mannan, “Energy Efficient Hybrid Dual Axis Solar Tracking System,” *Journal of Renewable Energy*, 2014.
- [15] Miloš Jovanović, Dr Zeljko V Despotovic, Đorđe Urukalo, “THE CHRONOLOGICAL SYSTEM FOR SOLAR TRACKING IMPLEMENTED ON MOBILE SOLAR GENERATOR,” in *The Fifth International Conference on Renewable Electrical Power Sources-ICRES*, Belgrade, Serbia, Oct, 2017.
- [16] S. M. R. M. T. J. P. Muthukumar a, “Energy efficient dual axis solar tracking system using IOT,” *Measurement: Sensors*, p. 7, 2023.
- [17] M. D. S. D. a. V. B. Ashish Patil, “Design and prototyping of dual axis solar tracking system for performance enhancement of solar photo-voltaic power plant,” in *E3S Web Conf*, Pune, 2020.
- [18] K. I. H. E. A. ASNIL, “DESIGN AND PERFORMANCE OF DUAL AXIS SOLAR TRACKER BASED ON LIGHT SENSORS TO MAXIMIZE THE PHOTOVOLTAIC ENERGY OUTPUT,” *Journal of Theoretical and Applied Information Technology* , vol. 100, p. 5, 2022.
- [19] C. D. Á. F. L. N. D. G. G. J. R. Joel J. Ontiveros, “Evaluation and Design of Power Controller of Two-Axis Solar Tracking by PID and FL for a Photovoltaic Module,” *International Journal of Photoenergy*, 2020.
- [20] “ADDICORE,” [Online]. Available: <https://www.addicore.com/products/ads1115-16-bit-adc-4-channel>.
- [21] Shawn, “seeedstudio,” 2020. [Online]. Available: <https://www.seeedstudio.com/blog/2020/02/15/acs712-current-sensor-features-how-it-works-arduino-guide/>.
- [22] L. Miller, “learn robotics,” 25 April 2024. [Online]. Available: <https://www.learnrobotics.org/blog/raspberry-pi-servo-motor/>.

ANNEXES

Annex A:

```
//Importing necessary libraries
#include "AS5600.h"
#include "Wire.h"
AS5600 as5600;

//Pin assignments:
#define LPWM 6
#define RPWM 5
#define counterClockWise 12
#define clockWise 11
#define DIN 2

//Vairables to store extreme positions of the motor
int clockWiseEncoder = 0;
int counterClockWiseEncoder = 0;

//Function that calibrates the motors
void callibration() {
    //Rotate clockwise to reach clockwise switch
    while(digitalRead(clockWise) == HIGH) {
        digitalWrite(RPWM, LOW);
        analogWrite(LPWM, 150);
        Serial.println(as5600.getCumulativePosition());
    }
    digitalWrite(RPWM, LOW);
    digitalWrite(LPWM, LOW);
    delay(3333);
    clockWiseEncoder = as5600.getCumulativePosition();
    Serial.print("Clockwise: ");
```

```

Serial.println(clockWiseEncoder);
delay(2000);
//Rotate anticlockwise to reach anticlockwise switch
while(digitalRead(counterClockWise) == HIGH) {
    digitalWrite(LPWM, LOW);
    analogWrite(RPWM, 150);
    Serial.println(as5600.getCumulativePosition());
}
digitalWrite(RPWM, LOW);
digitalWrite(LPWM, LOW);
delay(3333);
counterClockWiseEncoder = as5600.getCumulativePosition();
Serial.print("CounterClockwise: ");
Serial.println(counterClockWiseEncoder);
delay(2000);
}

//Setup all the pins and do the calibration
void setup() {
    pinMode(RPWM, OUTPUT); // Set RPWM pin as output
    pinMode(LPWM, OUTPUT); // Set LPWM pin as output
    pinMode(DIN, INPUT);
    pinMode(clockWise, INPUT_PULLUP);
    pinMode(counterClockWise, INPUT_PULLUP);
    TCCR0B = TCCR0B & B11111000 | B00000001;
    Wire.begin();
    Serial.begin(9600);
    Serial.println("Callibration");
    as5600.begin(4);
    as5600.setDirection(AS5600_CLOCK_WISE);
    callibration();
}

```

```

//PID constant
float Ki = 0.01;
float Kp = 1;

unsigned int accError = 0;
int currentAngle = 0;

//Move servo to the position designated by input signal
void loop() {
    int pulseWidth = pulseIn(DIN, HIGH); // Read pulse width of servo
    signal
    if(pulseWidth >=300 && pulseWidth <=1500) {
        int targetAngle = map(pulseWidth, 300, 1500,
        counterClockWiseEncoder+200, clockWiseEncoder-200);
        Serial.print("Target Angle: ");
        Serial.println(targetAngle);
        if (targetAngle < counterClockWiseEncoder+100) targetAngle =
        counterClockWiseEncoder+100;
        if (targetAngle > clockWiseEncoder-100) targetAngle =
        clockWiseEncoder-100;
        currentAngle = as5600.getCumulativePosition();
        int error = targetAngle - currentAngle;
        Serial.print("Current Angle: ");
        Serial.println(currentAngle);
        //Only perform correction if error is more then 100
        if(abs(error) > 100) {
            int speed = map(abs(error), 120, 6000, 100, 230);
            speed = Kp * speed + Ki * accError;
            speed = constrain(speed, 120, 230);
            //Move in the counter clockwise direction
            if(targetAngle < currentAngle) {
                digitalWrite(LPWM, LOW);
                analogWrite(RPWM, speed);
                delay(100);
            }
        }
    }
}

```

```
    }  
else if (targetAngle > currentAngle) {  
    digitalWrite(RPWM, LOW);  
    analogWrite(LPWM, speed);  
    delay(100);  
}  
accError += abs(error);  
}  
else{  
    digitalWrite(RPWM, LOW);  
    digitalWrite(LPWM, LOW);  
  
}}  
else{  
    digitalWrite(RPWM, LOW);  
    digitalWrite(LPWM, LOW);  
}  
delay(100);  
}
```

Annex B:

```
//Import necessary Libraries
#include "AS5600.h"
#include "Wire.h"
AS5600 as5600;

//Pin assignments
#define LPWM 6
#define RPWM 5
#define downLimit 11
#define upLimit 12
#define DIN 2

//Setup the pins
void setup() {
    pinMode(RPWM, OUTPUT); // Set RPWM pin as output
    pinMode(LPWM, OUTPUT); // Set LPWM pin as output
    pinMode(DIN, INPUT);
    pinMode(downLimit, INPUT_PULLUP);
    pinMode(upLimit, INPUT_PULLUP);
    Wire.begin();
    Serial.begin(9600);
    Serial.println("Calibration");

    //Calibration
    //Move down till limit switch
    digitalWrite(LPWM, LOW);
    analogWrite(RPWM, 100);
    while(digitalRead(downLimit) == HIGH);
    digitalWrite(RPWM, LOW);
    digitalWrite(LPWM, LOW);
```

```

delay(3000);

//Move up till limit switch
digitalWrite(RPWM, LOW);
analogWrite(LPWM, 215);
while(digitalRead(upLimit) == HIGH);
digitalWrite(RPWM, LOW);
digitalWrite(LPWM, LOW);
delay(2000);

//Move Down till centre
digitalWrite(LPWM, LOW);
analogWrite(RPWM, 100);
delay(3000);
digitalWrite(RPWM, LOW);
digitalWrite(LPWM, LOW);
}

void loop() {
    int pulseWidth = pulseIn(DIN, HIGH); // Read pulse width of servo
    signal

    Serial.println(pulseWidth);
    //If the pulse width is sort of in the middle, do nothing, else
    if(pulseWidth > 500 && pulseWidth < 2400) {
        if(pulseWidth < 1350 || pulseWidth > 1500) {
            //Need to move Down
            if(pulseWidth < 1350) {
                if(digitalRead(downLimit) == LOW) {
                    digitalWrite(LPWM, LOW);
                    digitalWrite(RPWM, LOW);
                }
            }
            else {

```

```

        int motorSpeed = (1350 - pulseWidth)/4;    //Max :500 so
divide by 3.4
        if(motorSpeed > 140) motorSpeed = 140;
        Serial.print("Going down with speed: ");
        Serial.println(motorSpeed);
        //Move Downward
        digitalWrite(LPWM, LOW);
        analogWrite(RPWM, motorSpeed);
        delay(100);
    }
}
else if(pulseWidth > 1500) {
    if(digitalRead(upLimit) == LOW) {
        digitalWrite(LPWM, LOW);
        digitalWrite(RPWM, LOW);
    }
    else {
        //Move Upward
        int motorSpeedR = (pulseWidth - 1500)/4;    //Max :500 so
divide by 3.4
        if(motorSpeedR > 210) motorSpeedR = 210;
        Serial.print("Going up with speed: ");
        Serial.println(motorSpeedR);
        digitalWrite(RPWM, LOW);
        analogWrite(LPWM, motorSpeedR);
        delay(100);
    }
    //MoveUpward
}
}
else {
    Serial.println("Input:90");
    digitalWrite(RPWM, LOW);

```



```
        digitalWrite(LPWM, LOW);
    }}
else {
    Serial.println("Out of Range");
    digitalWrite(RPWM, LOW);
    digitalWrite(LPWM, LOW);
}
}
```

Annex C:

```
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#define ADS1115_ADDRESS 0x49

//Create an ADC object
Adafruit_ADS1115 ads;

//LDR connected to channels on ADS1115
const int TR = 0;
const int BR = 1;
const int TL = 2;
const int BL = 3;

//Setting up the ADS1115 module
void setup(void)
{
  ads.begin(ADS1115_ADDRESS);
  Serial.begin(9600);
  Serial.println("Hello!");
  // Initialize the ADC
  if (!ads.begin(ADS1115_ADDRESS))
  {
    Serial.println("Failed to initialize ADS.");
    while (1);
  }

  // Set the gain (PGA) for better accuracy. You can adjust this based on
  your requirements.

  ads.setGain(GAIN_TWOTHIRDS);

  Serial.println("Ready.");
}
```

```

void loop(void)
{
    // Read the raw ADC value from LDR
    uint16_t adcTR;
    uint16_t adcBR;
    uint16_t adcTL;
    uint16_t adcBL;
    adcTR = ads.readADC_SingleEnded(TR);
    adcBR = ads.readADC_SingleEnded(BR);
    adcTL = ads.readADC_SingleEnded(TL);
    adcBL = ads.readADC_SingleEnded(BL);

    //Calculate further values
    uint16_t TOP;
    uint16_t BOTTOM;
    uint16_t LEFT;
    uint16_t RIGHT;

    TOP=(adcTR+adcTL)/2;
    BOTTOM=(adcBR+adcBL)/2;
    LEFT=(adcTL+adcBL)/2;
    RIGHT=(adcTR+adcBR)/2;

    // Print results
    Serial.print("LDR Value TR: "); Serial.print(adcTR);
    Serial.print("\tLDR Value BR: "); Serial.print(adcBR);
    Serial.print("\tLDR Value TL: "); Serial.print(adcTL);
    Serial.print("\tLDR Value BL: "); Serial.print(adcBL);
    Serial.print("\tLDR Value Top: "); Serial.print(TOP);
    Serial.print("\tLDR Value Bottom: "); Serial.print(BOTTOM);
    Serial.print("\tLDR Value Left: "); Serial.print(LEFT);
    Serial.print("\tLDR Value right: "); Serial.print(RIGHT);

    delay(1000);}

```

Annex D:

```
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <Servo.h>
#define ADS1115_ADDRESS 0x49

//Object creation
Adafruit_ADS1115 ads;
Servo panservo;

// LDR connected to channels on ADS1115
const int TR = 0;
const int BR = 1;
const int TL = 2;
const int BL = 3;

//Servo pin definition
const int panServoPin = 10;

//Setting up the ADS1115 module and servo pin
void setup(void)
{
  panservo.attach(panServoPin);
  ads.begin(ADS1115_ADDRESS);
  Serial.begin(9600);
  Serial.println("Hello!");
  ads.begin(ADS1115_ADDRESS);
  if (!ads.begin(ADS1115_ADDRESS))
  {
    Serial.println("Failed to initialize ADS.");
    while (1);
  }
}
```

```

    ads.setGain(GAIN_TWOTHIRDS);
    Serial.println("Ready.");
}

//Function to calculate difference between right and left LDR values
void calculateLDRValuesAndDifferencesRL(uint16_t &adcTR, uint16_t
&adcBR, uint16_t &adcTL, uint16_t &adcBL, int16_t &diffRL)
{
    // Read the raw ADC value from LDR
    uint16_t TOP, BOTTOM, LEFT, RIGHT;

    adcTR = ads.readADC_SingleEnded(TR);
    adcBR = ads.readADC_SingleEnded(BR);
    adcTL = ads.readADC_SingleEnded(TL);
    adcBL = ads.readADC_SingleEnded(BL);

    //calculate further values
    TOP = (adcTR + adcTL) / 2;
    BOTTOM = (adcBR + adcBL) / 2;
    LEFT = (adcTL + adcBL) / 2;
    RIGHT = (adcTR + adcBR) / 2;

    //calculate difference
    diffRL = RIGHT - LEFT;
}

void loop(void)
{
    uint16_t adcTR, adcBR, adcTL, adcBL;
    int16_t diffTB, diffRL;

    //pan tracking

```

```

    calculateLDRValuesAndDifferencesRL(adcTR,   adcBR,   adcTL,   adcBL,
diffRL);
    if (diffRL>500) {
        while(diffRL>500){
            // Moves towards right
            panservo.write(panservo.read() - 1);
            delay(100);
            calculateLDRValuesAndDifferencesRL(adcTR,   adcBR,   adcTL,   adcBL,
diffRL);
        }
    }
    else if(diffRL<-500){
        while(diffRL<-500){
            // Moves towards left
            panservo.write(panservo.read() + 1);
            delay(100);
            calculateLDRValuesAndDifferencesRL(adcTR,   adcBR,   adcTL,   adcBL,
diffRL);
        }
    }
    delay(100);
}

```

Annex E:

```
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <Servo.h>
#define ADS1115_ADDRESS 0x48

//Object creation
Adafruit_ADS1115 ads;
Servo panservo;
Servo tiltservo;

// LDR connected to channels on ADS1115
const int TR = 0;
const int BR = 1;
const int TL = 2;
const int BL = 3;

//Servo pin definition
const int panServoPin = 10;
const int tiltServoPin = 9;

//Setting up the ADS1115 module and servo pins
void setup(void) {
  panservo.attach(panServoPin);
  tiltservo.attach(tiltServoPin);
  ads.begin(ADS1115_ADDRESS);
  Serial.begin(9600);
  Serial.println("Hello!");
  ads.begin(ADS1115_ADDRESS);
  if (!ads.begin(ADS1115_ADDRESS)) {
    Serial.println("Failed to initialize ADS.");
    while (1);
  }
}
```

```

    }

    ads.setGain(GAIN_TWOTHIRDS);

    Serial.println("Ready.");
}

//Function to calculate difference between right and left LDR values
void calculateLDRValuesAndDifferencesRL(uint16_t &adcTR, uint16_t
&adcBR, uint16_t &adcTL, uint16_t &adcBL, int16_t &diffRL)
{

    // Read the raw ADC value from LDR
    uint16_t TOP, BOTTOM, LEFT, RIGHT;

    adcTR = ads.readADC_SingleEnded(TR);
    adcBR = ads.readADC_SingleEnded(BR);
    adcTL = ads.readADC_SingleEnded(TL);
    adcBL = ads.readADC_SingleEnded(BL);

    //calculate further values
    TOP = (adcTR + adcTL) / 2;
    BOTTOM = (adcBR + adcBL) / 2;
    LEFT = (adcTL + adcBL) / 2;
    RIGHT = (adcTR + adcBR) / 2;

    //calculate difference
    diffRL = RIGHT - LEFT;
}

//Function to calculate difference between top and bottom LDR values
void calculateLDRValuesAndDifferencesTB(uint16_t &adcTR, uint16_t
&adcBR, uint16_t &adcTL, uint16_t &adcBL, int16_t &diffTB)
{

    // Read the raw ADC value from LDR
    uint16_t TOP, BOTTOM, LEFT, RIGHT;

```



```

adcTR = ads.readADC_SingleEnded(TR);
adcBR = ads.readADC_SingleEnded(BR);
adcTL = ads.readADC_SingleEnded(TL);
adcBL = ads.readADC_SingleEnded(BL);

//calculate further values
TOP = (adcTR + adcTL) / 2;
BOTTOM = (adcBR + adcBL) / 2;
LEFT = (adcTL + adcBL) / 2;
RIGHT = (adcTR + adcBR) / 2;

//calculate difference
diffTB = TOP - BOTTOM;
}

void loop(void)
{
  uint16_t adcTR, adcBR, adcTL, adcBL;
  int16_t diffTB, diffRL;

  //pan tracking
  calculateLDRValuesAndDifferencesRL(adcTR,   adcBR,   adcTL,   adcBL,
diffRL);
  if (diffRL>500) {
    while(diffRL>500){
      // Moves towards right
      panservo.write(panservo.read() - 1);
      delay(100);
      calculateLDRValuesAndDifferencesRL(adcTR,   adcBR,   adcTL,   adcBL,
diffRL);
    }
  }
}

```

```

else if(diffRL<-500){
    while(diffRL<-500){
        // Moves towards left
        panservo.write(panservo.read() + 1);
        delay(100);
        calculateLDRValuesAndDifferencesRL(adcTR,  adcBR,  adcTL,  adcBL,
diffRL);
    }
}

//tilt tracking
calculateLDRValuesAndDifferencesTB(adcTR,  adcBR,  adcTL,  adcBL,
diffTB);
if (diffTB>500) {
    while(diffTB>500){
        // tilts up
        tiltservo.write(tiltservo.read() - 1);
        delay(100);
        calculateLDRValuesAndDifferencesTB(adcTR,  adcBR,  adcTL,  adcBL,
diffTB);
    }
}
else if(diffTB<-500){
    while(diffTB<-500){
        // Tilt down
        tiltservo.write(tiltservo.read() + 1);
        delay(100);
        calculateLDRValuesAndDifferencesTB(adcTR,  adcBR,  adcTL,  adcBL,
diffTB);
    }
}
delay(100);
}

```

Annex F:

```
import RPi.GPIO as GPIO
import time

# Set the hardware PWM pin number for the servo
servo_pin = 12 # Assuming you're using GPIO pin 12 for hardware PWM

# Set the PWM frequency (Hz) and duty cycle ranges for the servo
frequency = 50 # 50 Hz is typical for servo motors
duty_cycle_min = 2.5 # Minimum duty cycle for 0 degrees
duty_cycle_max = 12.5 # Maximum duty cycle for 180 degrees

# Initialize GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(servo_pin, GPIO.OUT)

# Create PWM object
pwm = GPIO.PWM(servo_pin, frequency)

# Function to convert angle to duty cycle
def angle_to_duty_cycle(angle):
    duty_cycle = ((angle / 180) * (duty_cycle_max - duty_cycle_min))
    + duty_cycle_min
    return duty_cycle

# Function to convert duty cycle to angle
def duty_cycle_to_angle(duty_cycle):
    angle = ((duty_cycle - duty_cycle_min) / (duty_cycle_max -
duty_cycle_min)) * 180
    return angle

try:
    pwm.start(0) # Start PWM with 0% duty cycle (servo in neutral
```

```

position)

    current_angle=0

    # Read current duty cycle and convert to angle
    #   current_duty_cycle = 0
    while True:

        current_duty_cycle=angle_to_duty_cycle(current_angle)
        pwm.ChangeDutyCycle(current_duty_cycle)
        new_duty_cycle=pwm._dc
        new_angle=duty_cycle_to_angle(new_duty_cycle)
        current_angle=new_angle+1
        if current_angle > 180:
            current_angle = 0 # Reset to 0 degrees if it exceeds 180
        print("Current Angle:", current_angle, "New Angle:",
new_angle)

        print("Current dc:", current_duty_cycle, "New dc:",
new_duty_cycle)

        time.sleep(0.1) # Adjust the delay as needed for
responsiveness

except KeyboardInterrupt:

    pass

# Clean up GPIO
pwm.stop()
GPIO.cleanup()

```

Annex G:

```
import RPi.GPIO as GPIO
from time import sleep
import board
import time
import busio

import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn

#ADS Constants
i2c=busio.I2C(board.SCL, board.SDA)
GAIN = 2/3

#initialize adc
adc = ADS.ADS1115(i2c, gain=GAIN)

#servo pins
pan_servo_pin = 12
tilt_servo_pin = 13

# Set the PWM frequency (Hz) and duty cycle ranges for the servo
frequency = 50 # 50 Hz is typical for servo motors
duty_cycle_min = 2.5 # Minimum duty cycle for 0 degrees
duty_cycle_max = 12.5 # Maximum duty cycle for 180 degrees

# Initialize GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(pan_servo_pin, GPIO.OUT)
GPIO.setup(tilt_servo_pin, GPIO.OUT)

# Create PWM object
pwm1 = GPIO.PWM(pan_servo_pin, frequency)
pwm2 = GPIO.PWM(tilt_servo_pin, frequency)

#adc channels
TR=AnalogIn(adc,ADS.P0)
BR=AnalogIn(adc,ADS.P1)
TL=AnalogIn(adc,ADS.P2)
```

```

BL=AnalogIn(adc,ADS.P3)

# Function to convert angle to duty cycle
def angle_to_duty_cycle(angle):
    duty_cycle = ((angle / 180) * (duty_cycle_max - duty_cycle_min)) +
duty_cycle_min
    return duty_cycle

# Function to convert duty cycle to angle
def duty_cycle_to_angle(duty_cycle):
    angle = ((duty_cycle - duty_cycle_min) / (duty_cycle_max -
duty_cycle_min)) * 180
    return angle

#Function to get right left difference
def calculate_ldr_values_and_differences_rl():
    adcTR=TR.value
    adcBR=BR.value
    adcTL=TL.value
    adcBL=BL.value
    TOP=(adcTR+adcTL)/2
    BOTTOM=(adcBR+adcBL)/2
    LEFT=(adcTL+adcBL)/2
    RIGHT=(adcTR+adcBR)/2
    return RIGHT-LEFT

#Function to get top bottom difference
def calculate_ldr_values_and_differences_tb():
    adcTR=TR.value
    adcBR=BR.value
    adcTL=TL.value
    adcBL=BL.value
    TOP=(adcTR+adcTL)/2
    BOTTOM=(adcBR+adcBL)/2
    LEFT=(adcTL+adcBL)/2

```

```

RIGHT=(adcTR+adcBR)/2

return TOP-BOTTOM

pwm1.start(0)
pwm2.start(0)
while True:
    diff_rl=calculate_ldr_values_and_differences_rl()
    if diff_rl>500:
        while diff_rl>500:
            current_duty_cycle=pwm1._dc
            current_angle = duty_cycle_to_angle(current_duty_cycle)
            new_angle = current_angle - 1 # Decrement by 1 degree
            if new_angle < 0:
                new_angle = 0 # Ensure angle does not go below minimum
            new_duty_cycle=angle_to_duty_cycle(new_angle)
            pwm1.ChangeDutyCycle(new_duty_cycle)
            sleep(0.1)
            diff_rl=calculate_ldr_values_and_differences_rl()
    elif diff_rl<-500:
        while diff_rl<-500:
            current_duty_cycle=pwm1._dc
            current_angle = duty_cycle_to_angle(current_duty_cycle)
            new_angle = current_angle + 1 # Increment by 1 degree
            if new_angle > 180:
                new_angle = 180 # Ensure angle does not exceed maximum
            new_duty_cycle=angle_to_duty_cycle(new_angle)
            pwm1.ChangeDutyCycle(new_duty_cycle)
            sleep(0.1)
            diff_rl=calculate_ldr_values_and_differences_rl()

    diff_tb=calculate_ldr_values_and_differences_tb()
    if diff_tb>500:
        while diff_tb>500:

```

```

current_duty_cycle=pwm2._dc
current_angle = duty_cycle_to_angle(current_duty_cycle)
new_angle = current_angle - 1 # Decrement by 1 degree
if new_angle < 0:
    new_angle = 0 # Ensure angle does not go below minimum
new_duty_cycle=angle_to_duty_cycle(new_angle)
pwm2.ChangeDutyCycle(new_duty_cycle)
sleep(0.1)
diff_tb=calculate_ldr_values_and_differences_tb()
elif diff_tb<-500:
    while diff_tb<-500:
        current_duty_cycle=pwm2._dc
        current_angle = duty_cycle_to_angle(current_duty_cycle)
        new_angle = current_angle + 1 # Increment by 1 degree
        if new_angle > 180:
            new_angle = 180 # Ensure angle does not exceed maximum
        new_duty_cycle=angle_to_duty_cycle(new_angle)
        pwm2.ChangeDutyCycle(new_duty_cycle)
        sleep(0.1)
        diff_tb=calculate_ldr_values_and_differences_tb()
sleep(0.5)

```


Annex H:

```
import RPi.GPIO as GPIO
import gpiod
from time import sleep
import board
import time
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
import requests

# using datetime module
import datetime;

#Used for turning mosfet ON/OFF
currentOn = 17

#ADS Contants
i2c=busio.I2C(board.SCL, board.SDA)
GAIN = 2/3

#initialize adc
adc = ADS.ADS1115(i2c, gain=GAIN)

#Server Endpoint
url = 'https://script.google.com/macros/s/AKfycbyp3-uEarm00061J9V5-
vKooteaAuZLV9KRWTFhCIHa_FY0VJ8qg3czwnYyw0NQJJvD/exec'

GPIO.setmode(GPIO.BCM)
GPIO.setup(currentOn, GPIO.OUT)
GPIO.output(currentOn, GPIO.LOW)

voltage = AnalogIn(adc, ADS.P0)
```

```

current = AnalogIn(adc, ADS.P1)
done = False

while not done:
    #Measure voltage
    try:
        voltage = AnalogIn(adc, ADS.P0)
        print("Voltage: ", voltage.voltage*9.5)

        #Measure Current
        #led_line.set_value(1)
        GPIO.output(currentOn, GPIO.HIGH)
        sleep(5)
        current = AnalogIn(adc, ADS.P1)
        currentVoltage = round((current.voltage), 3)
        print("Current Voltage: ", currentVoltage)

        GPIO.output(currentOn, GPIO.LOW)
        difference = currentVoltage - 2.501
        print("Difference: ", difference)

        currentCalc = abs(difference/0.073)
        print("Current : ", currentCalc)
        # ct stores current time
        with open('/home/mariam/Desktop/picode/13thmaydata.csv', 'a')
as f: # Open file for writing
            ct = datetime.datetime.now()
            print("timestamp:-", ct)
            txt1 = "Voltage = {}".format(round(voltage.voltage*9.5,3))
            txt2 = "Current = {}".format(round(currentCalc, 3))
            txt3 = "Time = {}\n".format(ct)
            print(txt1 + txt2 + txt3)

```

```

        f.write("{}".format(txt1 + txt2 + txt3))
        f.close()

        dataObj = {'data1': voltage.voltage * 9.5, 'data2':
currentCalc}

        requests.post(url, json = dataObj)
        print("Data sent")
    except:
        print("Error While collecting or sending data")
        with open('/home/mariam/Desktop/picode/13thmaydata.csv', 'a')
as f: # Open file for writing
            f.write("Top Line was not sent")
            f.close()

    sleep(50)

```

Annex I:

```
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <Servo.h>
#include <EEPROM.h>
#include <avr/wdt.h>

#define ADS1115_ADDRESS 0x49

// Object creation
Adafruit_ADS1115 ads;
Servo panservo;
Servo tiltservo;

// LDR connected to channels on ADS1115
const int TR = 0;
const int BR = 1;
const int TL = 2;
const int BL = 3;

// Variables defined
uint16_t TOP, BOTTOM, LEFT, RIGHT;
uint16_t adcTR, adcBR, adcTL, adcBL;
int16_t diffTB, diffRL;

// Initial angles given to servos
int panAngle = 90;
int tiltAngle = 90;

// Servo pin definition
```

```

const int panServoPin = 9;
const int tiltServoPin = 10;

// Setting up the ADS1115 module and servo pins
void setup(void) {
  Serial.begin(9600);
  Serial.println("Starting setup...");

  // Read the last angles from EEPROM
  panAngle = EEPROM.read(0);

  // Default angle if the EEPROM value is invalid
  if (panAngle < 0 || panAngle > 180)
  {
    panAngle = 90;
  }

  panservo.attach(panServoPin);
  panservo.write(panAngle);
  tiltservo.attach(tiltServoPin);
  tiltservo.write(tiltAngle);

  Wire.begin();
  Wire.setClock(100000);
  if (!ads.begin(ADS1115_ADDRESS))
  {
    Serial.println("Failed to initialize ADS.");
    while (1);
  }
  ads.setGain(GAIN_TWOTHIRDS);
  Serial.println("Setup completed.");

  // Initialize watchdog timer

```

```

    wdt_enable(WDTO_8S); // Set watchdog timer to 8 seconds
    delay(5000);
}

// Function to calculate difference between right and left LDR values
int calculateLDRValuesAndDifferencesRL() {
    adcTR = ads.readADC_SingleEnded(TR);
    adcBR = ads.readADC_SingleEnded(BR);
    adcTL = ads.readADC_SingleEnded(TL);
    adcBL = ads.readADC_SingleEnded(BL);

    // Calculate further values
    TOP = (adcTR + adcTL) / 2;
    BOTTOM = (adcBR + adcBL) / 2;
    LEFT = (adcTL + adcBL) / 2;
    RIGHT = (adcTR + adcBR) / 2;

    // Calculate difference
    diffRL = RIGHT - LEFT;
    Serial.print("RL: "); Serial.println(diffRL);
    return diffRL;
}

// Function to calculate difference between top and bottom LDR values
int calculateLDRValuesAndDifferencesTB() {
    adcTR = ads.readADC_SingleEnded(TR);
    adcBR = ads.readADC_SingleEnded(BR);
    adcTL = ads.readADC_SingleEnded(TL);
    adcBL = ads.readADC_SingleEnded(BL);

    // Calculate further values
    TOP = (adcTR + adcTL) / 2;
    BOTTOM = (adcBR + adcBL) / 2;

```

```

LEFT = (adcTL + adcBL) / 2;
RIGHT = (adcTR + adcBR) / 2;

// Calculate difference
diffTB = TOP - BOTTOM;
Serial.print("TB: "); Serial.println(diffTB);
return diffTB;
}

void loop(void) {
    wdt_reset(); // Reset watchdog timer at the start of each loop
iteration

    // Pan tracking
    diffRL = calculateLDRValuesAndDifferencesRL();
    if (diffRL > 20) {
        while (diffRL > 20 && panAngle > 0) {
            panAngle--;
            panservo.write(panAngle);
            EEPROM.write(0, panAngle); // Store the current pan angle in
EEPROM
            delay(100);
            diffRL = calculateLDRValuesAndDifferencesRL();
            wdt_reset(); // Reset watchdog timer within the loop
        }
    } else if (diffRL < -20) {
        while (diffRL < -20 && panAngle < 180) {
            panAngle++;
            panservo.write(panAngle);
            EEPROM.write(0, panAngle); // Store the current pan angle in
EEPROM
            delay(100);
            diffRL = calculateLDRValuesAndDifferencesRL();
            wdt_reset(); // Reset watchdog timer within the loop

```

```

    }
}
delay(2000);
// Tilt tracking
diffTB=calculateLDRValuesAndDifferencesTB();
if (diffTB>20) {
    while(diffTB>20){
        // tilts up
        tiltservo.write(150);
        delay(100);
        diffTB=calculateLDRValuesAndDifferencesTB();
    }
    tiltservo.write(90);
}
else if(diffTB<-20){
    while(diffTB<-20){
        // Tilt down
        tiltservo.write(40);
        delay(100);
        diffTB=calculateLDRValuesAndDifferencesTB();
    }
    tiltservo.write(90);
}
delay(900000);
}

```