Bernburg
Dessau
Köthen

Hochschule Anhalt (FH)
Anhalt University of Applied Sciences

.

emw
Fachbereich
Elektrotechnik, Maschinenbau
und Wirtschaftsingenieurwesen

MAIN PROJECT

PROJECT WORK SS, 2024

Dual Axis Photon Tracker Using Raspberry Pico

SUBMITTED BY,

MARY MEENU ANTONY (5058572)

SUPERVISED BY,

PROF. DR. MICHAEL BRUTSCHECK

CONTENTS

FIGURES

TABLES

1. REFERENCE LIST

- M. M. Al-Nimr and A. M. Akash, "Design and Implementation of a Dual-Axis Solar Tracker System," IEEE Transactions on Sustainable Energy, vol. 8, no. 4, pp. 1250-1257, 2017.
- A. Bhattacharjee et al., "Development of a Low-Cost Dual-Axis Solar Tracker Using Raspberry Pi," International Journal of Renewable Energy Research, vol. 8, no. 3, pp. 1295-1302, 2018.
- H. K. Kang et al., "Design and Implementation of a Dual-Axis Solar Tracker with Solar Radiation Measurement Using Arduino," Energies, vol. 13, no. 11, 2020.
- Raspberry Pi Foundation, "Getting started with Raspberry Pi Pico," available at: https://www.raspberrypi.org/documentation/pico/getting-started/
- Adafruit Industries, "Raspberry Pi Pico Pinout Diagram," available at: https://learn.adafruit.com/raspberry-pi-pico-pinout/pinouts
- "How to Build a Dual Axis Solar Tracker using Arduino and LDR," available at: https://www.electronicshub.org/dual-axis-solar-tracker-arduino/
- Raspberry Pi Pico Datasheet, available at: https://datasheets.raspberrypi.org/pico/pico-datasheet.pdf
- Raspberry Pi Pico C/C++ SDK Documentation, available at: https://datasheets.raspberrypi.org/pico/raspberry-pi-pico-c-sdk.pdf
- Adafruit PCA9685 PWM/Servo Driver Documentation, available at: https://learn.adafruit.com/16-channel-pwm-servo-driver/overview
- GitHub Repository for Dual-Axis Solar Tracker with Raspberry Pi Pico, available at: https://github.com/example/dual-axis-solar-tracker

2. INTRODUCTION

In an era where sustainability is paramount and renewable energy sources are gaining momentum, solar power stands out as a promising solution. However, maximizing the efficiency of solar panels remains a challenge, particularly as the sun traverses the sky, casting varying shadows and angles throughout the day. In the quest for sustainable energy solutions, solar power stands out as a promising avenue for harnessing clean and renewable energy. Solar trackers play a pivotal role in maximizing the efficiency of solar panels by orienting them towards the sun throughout the day, ensuring optimal exposure to sunlight. To address this, the integration of solar tracking systems has become increasingly popular.

Among various types of solar trackers, the dual-axis solar tracker system represents an advanced solution capable of precise positioning in both azimuth and elevation directions, thereby enhancing energy yield.

This project explores the design and implementation of a dual-axis solar tracker system, leveraging the capabilities of the Raspberry Pi Pico microcontroller. By dynamically orienting solar panels to optimize their exposure to sunlight, this system aims to enhance energy harvesting efficiency, offering a sustainable and cost-effective solution for renewable energy generation.

2.1 Overview of the project

The dual-axis solar tracker system using Raspberry Pi Pico microcontroller project aims to enhance the efficiency of solar panels by ensuring they continuously face the sun throughout the day. This project integrates the versatility of Raspberry Pi Pico, a low-cost microcontroller board, with the precision of dual-axis tracking technology to optimize solar energy harvesting.

At its core, this project involves designing and implementing a mechanism capable of adjusting the orientation of solar panels along both the horizontal and vertical axes. By accurately tracking the sun's position, the solar panels can maintain an optimal angle of incidence, maximizing the amount of sunlight they receive and thus increasing energy output.

The Raspberry Pi Pico serves as the brains of the operation, executing the control algorithms necessary to determine the sun's position based on time, date, and geographical location. It interfaces with sensors such as light-dependent resistors (LDRs) or photodiodes to detect sunlight intensity and orientation, providing real-time feedback for precise tracking adjustments.

The software aspect of the project involves programming the Raspberry Pi Pico microcontroller to calculate the optimal angles for the solar panels based on input from the sensors. This requires developing algorithms to interpret sensor data, calculate the sun's position relative to the panels, and control the motors or actuators responsible for adjusting their orientation.

Hardware components include motors or servo motors responsible for tilting the solar panels along both axes, as well as the necessary mechanical structure to support and move the panels. Additionally, sensors for measuring sunlight intensity and orientation are crucial for accurate tracking.

2.2 Purpose and Goals

The purpose of this project is to design and implement a dual-axis solar tracker system using the Raspberry Pi Pico microcontroller. This system will automatically orient solar panels to maximize solar energy absorption throughout the day, thereby increasing the efficiency of solar power generation.

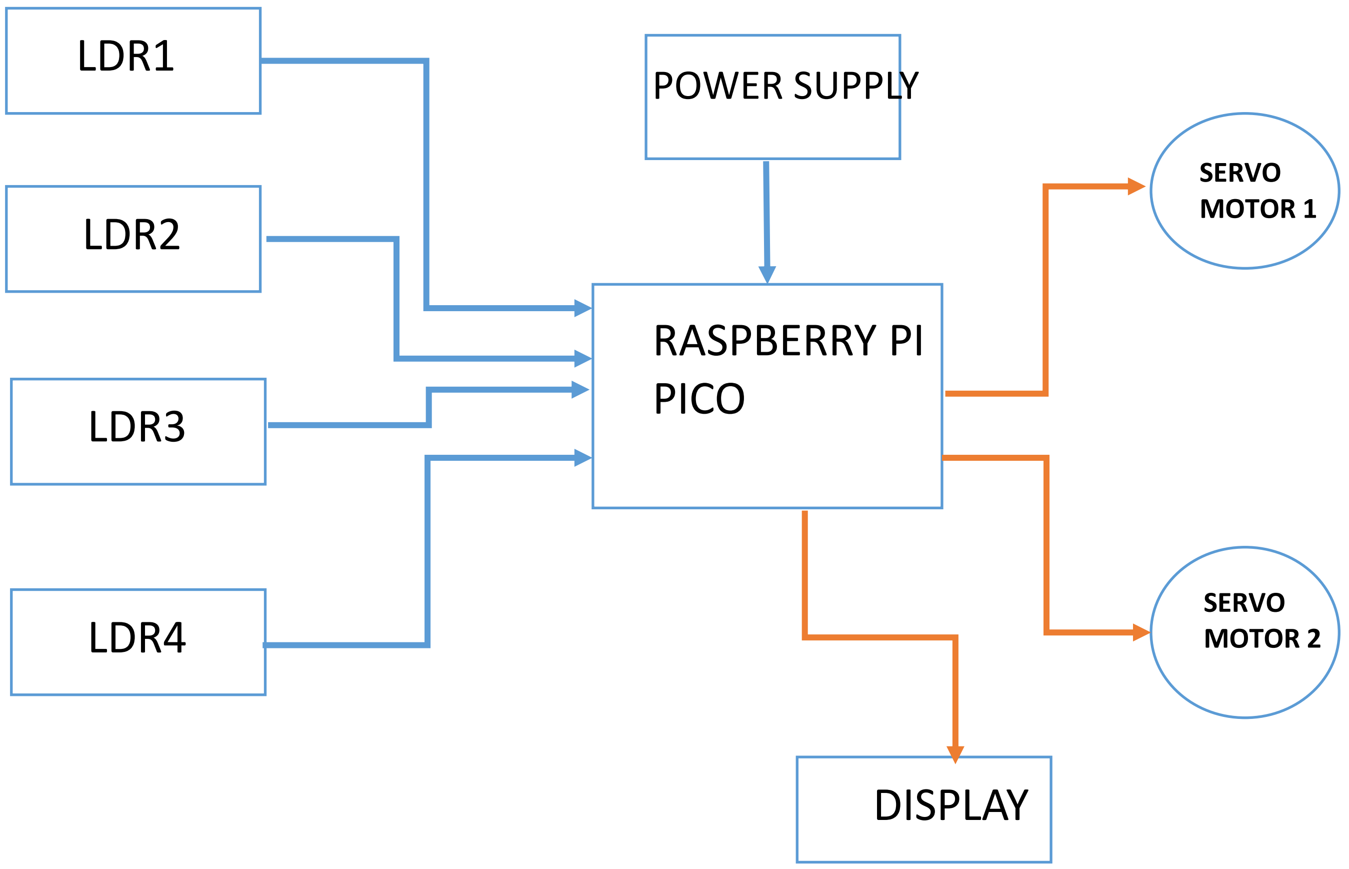
Here are the major goals of this project:

1. **Efficiency Improvement:** Develop a solar tracker system capable of adjusting the orientation of solar panels in real-time to optimize solar energy capture, ensuring maximum efficiency in power generation.
2. **Dual-Axis Tracking:** Implement dual-axis tracking functionality to account for both the azimuth (horizontal) and altitude (vertical) angles of the sun, allowing the solar panels to follow the sun's movement accurately throughout the day.
3. **Sensor Integration:** Integrate sensors such as light-dependent resistors (LDRs), GPS module, and/or a digital compass to accurately detect the position of the sun and provide input for precise tracking adjustments.
4. **Raspberry Pi Pico Control:** Utilize the Raspberry Pi Pico microcontroller as the central control unit for the solar tracker system, programming it to interpret sensor data, calculate optimal panel orientation angles, and control the movement of servo motors accordingly.
5. **Power Efficiency:** Design the control algorithm and software with a focus on minimizing power consumption to ensure the system operates efficiently, making it suitable for standalone solar power applications.
6. **Reliability and Durability:** Engineer the mechanical components of the solar tracker system to be robust and durable, capable of withstanding environmental factors such as wind, rain, and temperature variations, ensuring long-term reliability and performance.
7. **User Interface (Optional):** Develop a user-friendly interface, which could be web-based or through an LCD display, to provide users with real-time monitoring of the system's operation and status, as well as the ability to adjust settings and parameters as needed.

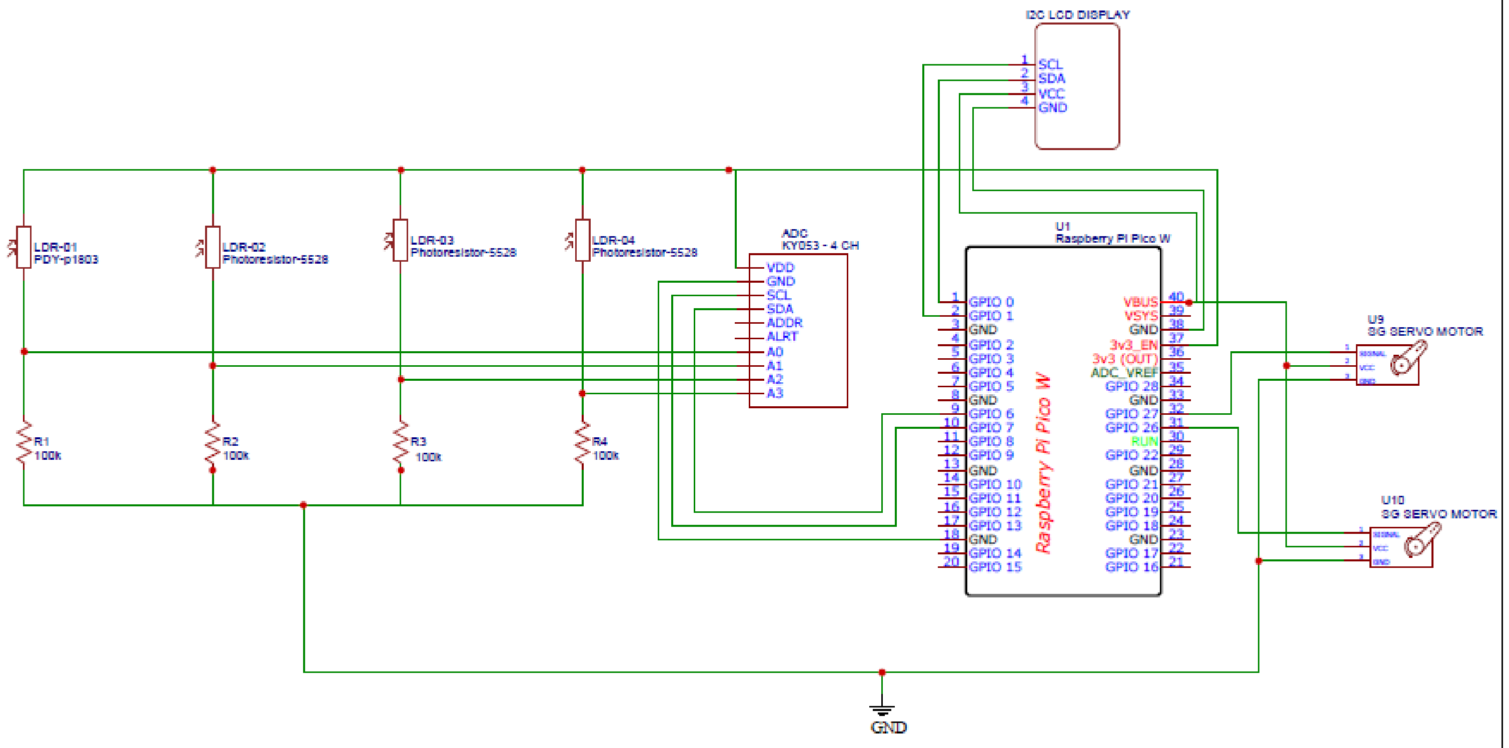
8. **Documentation and Open Source:** Document the design, development, and implementation process comprehensively, including hardware schematics, software code, and operational guidelines, with the intention of sharing the project as open-source to encourage collaboration and further development within the community.

3. PROJECT DESIGN AND IMPLEMENTATION

3.1 System Block Diagram



3.2 Schematic Diagram



3.3 Working Principle

The dual-axis solar tracker system using the Raspberry Pi Pico operates by continuously monitoring the sunlight intensity in four directions using Light Dependent Resistors (LDRs). It calculates the average light intensity values in the vertical and horizontal directions and compares them to determine the optimal position for the solar panels. If significant differences are detected, indicating uneven sunlight distribution, the system adjusts the position of the panels using servo motors. This dynamic tracking maximizes solar energy capture throughout the day, enhancing system efficiency. This is achieved through the use of Light Dependent Resistors (LDRs) connected to a voltage divider circuit, which converts the varying light intensity captured by the LDRs into analog voltages in the range of 0-3.3V suitable for input to the Pico's ADC (Analog-to-Digital Converter).

Four LDRs are strategically positioned in different directions: up, down, right, and left, to detect sunlight intensity from different angles. The Pico has limited ADC inputs, so the four LDRs are connected to these inputs. The microcontroller reads the analog values from each LDR, typically using the machine.ADC() function in Micropython, and calculates the average values for the LDRs in each direction.

Once the average values are obtained, the system calculates the differences between the average values of the LDRs in the vertical (up/down) and horizontal (left/right) directions. These differences indicate the relative intensity of sunlight in each direction, helping determine the optimal position for the solar panels.

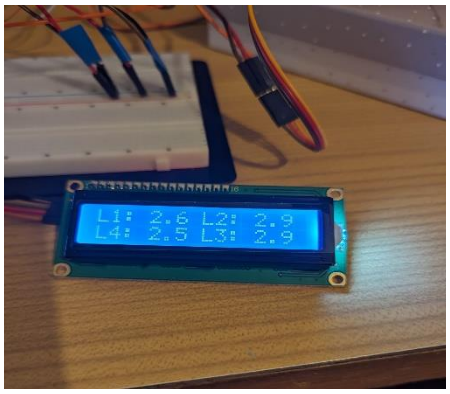
Next, the system compares the horizontal and vertical differences with predefined threshold values. If the horizontal difference exceeds the threshold, indicating a significant difference in sunlight intensity between the left and right sides, the system compares the average values of the left and right LDRs. Based on this comparison, it commands the horizontal servo motor to rotate the solar panels in the appropriate direction to maximize sunlight capture.

Similarly, if the vertical difference surpasses the threshold, indicating a significant difference in sunlight intensity between the top and bottom, the system compares the average values of the up and down LDRs. Depending on the comparison result, it commands the vertical servo motor to adjust the tilt of the solar panels accordingly.

Throughout this process, the analog values read from the LDRs are displayed on an LCD display (16x2) using I2C communication from the Pico. This provides real-time feedback on the sunlight intensity detected by each LDR and aids in troubleshooting and monitoring system performance.

- Light intensity is sensed by four LDRs, with their outputs converted to voltages using a voltage divider circuit suitable for Raspberry Pi Pico's ADC inputs.
- The Pico reads analog values from the LDRs to determine the average light intensity in each of the four directions: up, down, right, and left.
- Average values for vertical (up/down) and horizontal (left/right) directions are calculated based on the readings from the LDRs.
- The system computes the differences between the average values of the up/down and left/right directions to assess sunlight distribution.
- If the horizontal difference exceeds a predetermined threshold, indicating uneven sunlight distribution horizontally, the system compares the average values of the left and right LDRs.
- If the vertical difference surpasses the threshold, indicating uneven sunlight distribution vertically, the system compares the average values of the up and down LDRs.
- Based on the comparison results, the system commands the appropriate servo motor to adjust the orientation of the solar panels vertically or horizontally to optimize sunlight capture. Additionally, the LDR voltage values are displayed on a 16x2 LCD display via I2C communication for real-time monitoring.

Overall, the system's operation involves continuously monitoring the sunlight intensity in different directions, calculating the optimal position for the solar panels based on the average LDR values, and dynamically adjusting the panel orientation to maximize energy capture, all while providing feedback through the LCD display for user convenience and system monitoring.



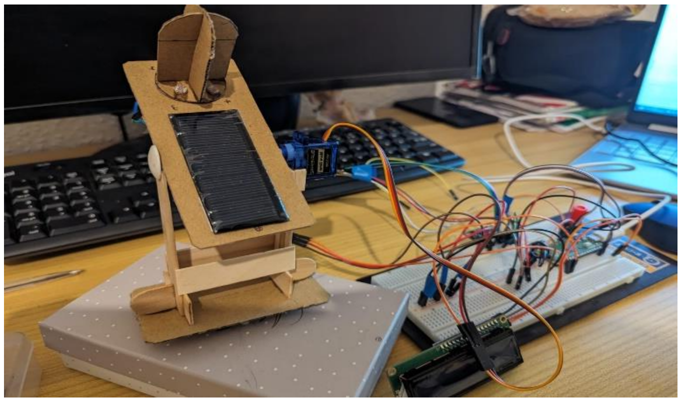


Figure 3.3 working display of Dual axis Photon tracker

3.4 Hardware Components Connected and Functions Realized

3.4.1 Raspberry Pico

The Raspberry Pi Pico is a compact microcontroller board developed by the Raspberry Pi Foundation. It features the RP2040 microcontroller chip, which offers powerful performance and versatile I/O capabilities. The Pico is designed for embedded projects, IoT applications, and education, providing an affordable yet capable platform for electronics enthusiasts and professionals alike. With its small size, low power consumption, and extensive community support, the Raspberry Pi Pico has quickly become popular for a wide range of projects, from simple sensor interfacing to complex automation systems.

Figure 3.4.1 Raspberry Pico

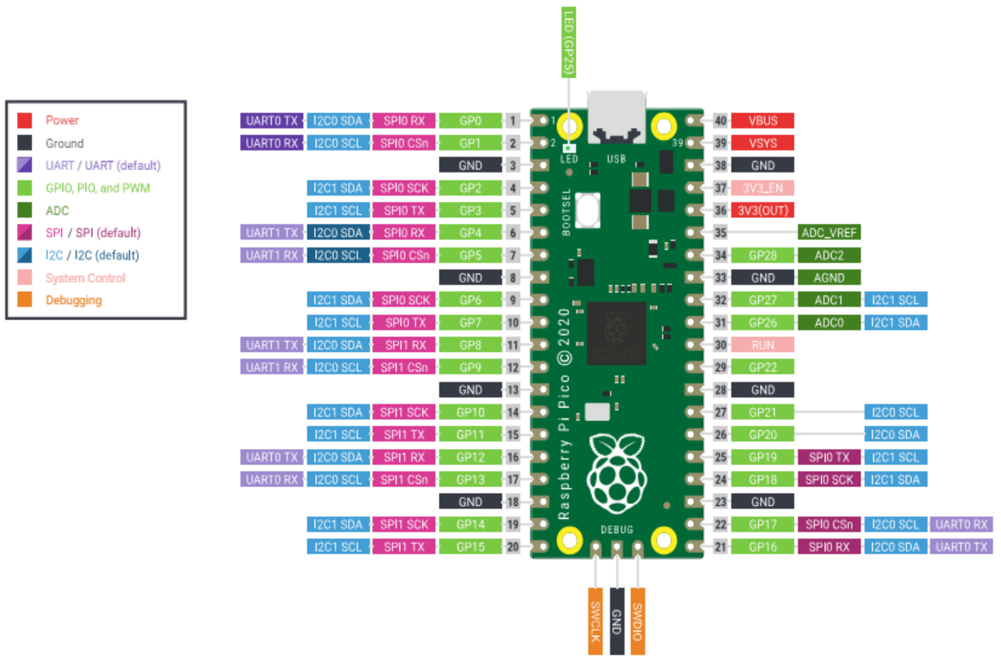


Figure 3.4.1 Raspberry Pico Pinout

Raspberry Pi Pico is a low-cost, high-performance microcontroller board with flexible digital interfaces. Key features include:

- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom
- Dual-core Arm Cortex M0+ processor, flexible clock running up to 133 MHz
- 264kB of SRAM, and 2MB of on-board flash memory
- USB 1.1 with device and host support
- Low-power sleep and dormant modes

- Drag-and-drop programming using mass storage over USB
- 26 × multi-function GPIO pins
- 2 × SPI, 2 × I2C, 2 × UART, 3 × 12-bit ADC, 16 × controllable PWM channels
- Accurate clock and timer on-chip
- Temperature sensor
- Accelerated floating-point libraries on-chip
- 8 × Programmable I/O (PIO) state machines for custom peripheral support

3.4.2 Light Dependent Resistors

The PDV-P8103 is a type of Light Dependent Resistor (LDR), also known as a photoresistor .The PDV-P8103 are (CdS), Photoconductive photocells designed to sense light from 400 to 700 nm. These light dependent resistors are available in a wide range of resistance values. They're packaged in a two leaded plastic-coated ceramic header.

Figure 3.4.2 Light Dependent Resistor

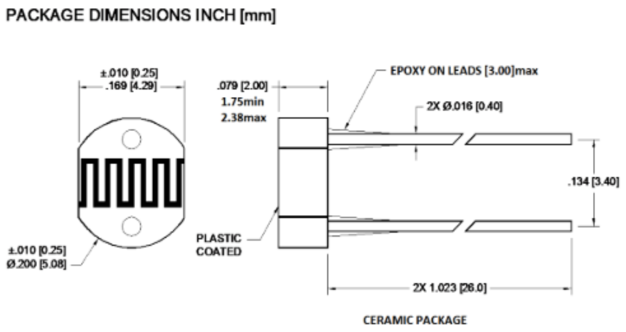


Figure 3.4.2 Light Dependent Resistor-Packet Dimension

It is a passive electronic component whose resistance decreases as the light intensity falling on it increases. Made from cadmium sulfide (CdS), this LDR is commonly used in light-sensing applications due to its sensitivity to visible light. The PDV-P8103 is characterized by its reliability, wide dynamic range, and quick response to changing light conditions. It is widely utilized in applications such as automatic lighting controls, light meters, and solar trackers, where detecting and responding to light levels is essential.

Category:	Photoresistor LDR
Type:	PDV-P8103
Resistance:	33kΩ
Connection:	Solder pins
Design type:	THT
nce (max.):	0.1W
Operating voltage max.:	150V
Operating voltage range:	150V (max)
Wavelength (max.):	520nm
Wavelength:	520nm (max)
Min. temperature:	-30°C
Max. temperature:	+75°C
Temperature range:	-30 - +75 °C °C
Width:	5.08mm
Height:	2mm
Product Type:	Photoresistor LDR
Length (depth):	4.29mm
Manufacturer No.:	PDV-P8103
EAN:	2050005613352

Table 3.4.2 Specifications of Light Dependent Resistors

3.4.3 Servo Motors

The SG90 digital micro servo is small and lightweight with high output power. It can rotate approximately 150 degrees and works just like standard servos. You can use any servo code, hardware or library to control this servo. Comes with 3 horns (arms) and screws. versatile servo motor widely used in robotics, RC toys, and other DIY electronics projects. It operates on a 4.8-6V power supply and provides a torque of up to 1.8 kg-cm. The SG90 can rotate approximately 180 degrees (90 degrees in each direction) and is controlled by PWM (Pulse Width Modulation) signals, allowing precise position control. Its compact size, low cost, and ease of use make it an ideal choice for applications requiring simple and reliable angular movement, such as in robotic arms, sensor positioning systems, and small-scale mechanical devices.

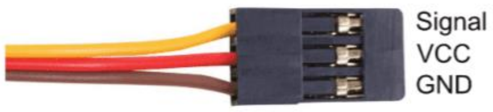


Figure 3.4.3 SG90 Micro Servo Motor

- Stall torque: 1.8 kg/cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Temperature range: 0 ºC - 55 ºC
- Dimensions: 22.2 x 11.8 x 31 mm approx.
- Weight: 9g

3.4.4 ADC Multiplier

Unlike the Arduino, the Raspberry Pico has only 3 analog inputs and no ADC (analog to digital converter) is integrated in the chip of the Raspberry Pico. This restricts the Raspberry Pico when using sensors where values cannot be output digitally. You can solve this problem with this analog-digital converter, because this module has 4 16 bit ADC channels, which you can use on the Raspberry Pico. The converter is connected to the Raspberry Pico via I2C and takes over the analog measurement and passes the value digitally to the Raspberry Pico.

- power consumption: 150 µA
- ADC channels: 4
- programmable data rate: 8 - 860 SPS
- bit rate: 16 bit
- interface: I2C
- dimensions: 18 x 28 mm
- Operating Voltage: 3V- 5.5V

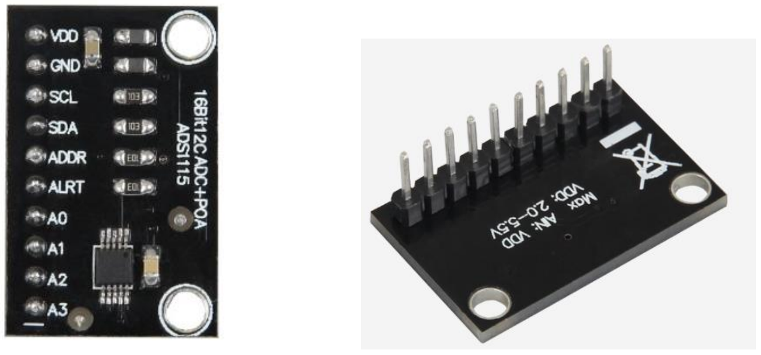


Figure 3.4.4 ADC Multiplier

3.4.5 Mini Solar cell panel

The POLY-PVZ-3070-5V is a small polycrystalline solar panel designed to generate electrical power from sunlight. With a power output of 0.2 watts and an operating voltage of 5 volts, this solar panel is suitable for low-power applications. Its compact size and lightweight design make it ideal for use in portable devices, small-scale solar projects, and educational experiments. The polycrystalline silicon cells provide a good balance of efficiency and cost, making this panel a practical choice for hobbyists and small electronic projects that require renewable energy sources.

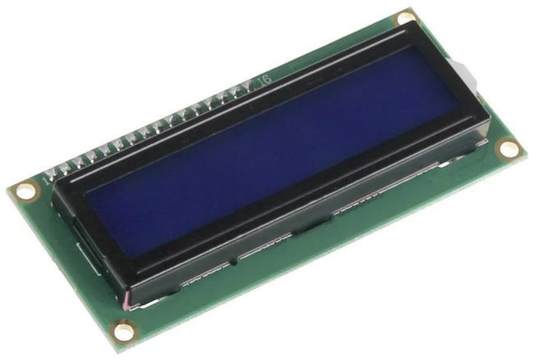
Figure 3.4.5 Solar Panel

Dimension: (L x W x H) 70 x 30 x 2.7 mm	**Height:** 2.7mm
Length: 70mm	**Width:** 30mm
Rated current: 0.04A	**nominal voltage:** 5V
Performance: 0.2W	**Product Type:** Polycrystalline solar module

Table 3.4.5 Solar panel specifications

3.4.6 LCD Display

The I2C LCD 16x2 display is a popular module used for displaying text and simple graphics in a variety of electronic projects. It features a 16-character by 2-line display, meaning it can show up to 32 characters at a time. The I2C interface simplifies wiring and communication with microcontrollers, as it only requires two data lines (SDA and SCL) in addition to power (VCC) and ground (GND). This makes it more efficient compared to the traditional 16-pin parallel interface, reducing the number of connections and making it easier to integrate into projects. The I2C LCD 16x2 is widely used in applications such as Arduino projects, Raspberry Pi systems, and other microcontroller-based devices for displaying information like sensor data, status messages, and user interfaces.



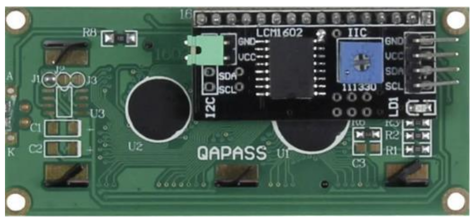


Figure 3.4.6 LCD Display

- 6.6 cm (2.6") Display
- HD44780 industry standard
- With soldered I2C communication adapter
- This means that only 4 I/O lines (power supply + 2x data lines) are required
- for operation Adapter chip: PCF8574AT
- Both 3.3V and 5V logic level
- 16x2 characters (2 lines with 16 columns)
- Backlight: Color Blue
- approx. 8x2.5 x0.6cm

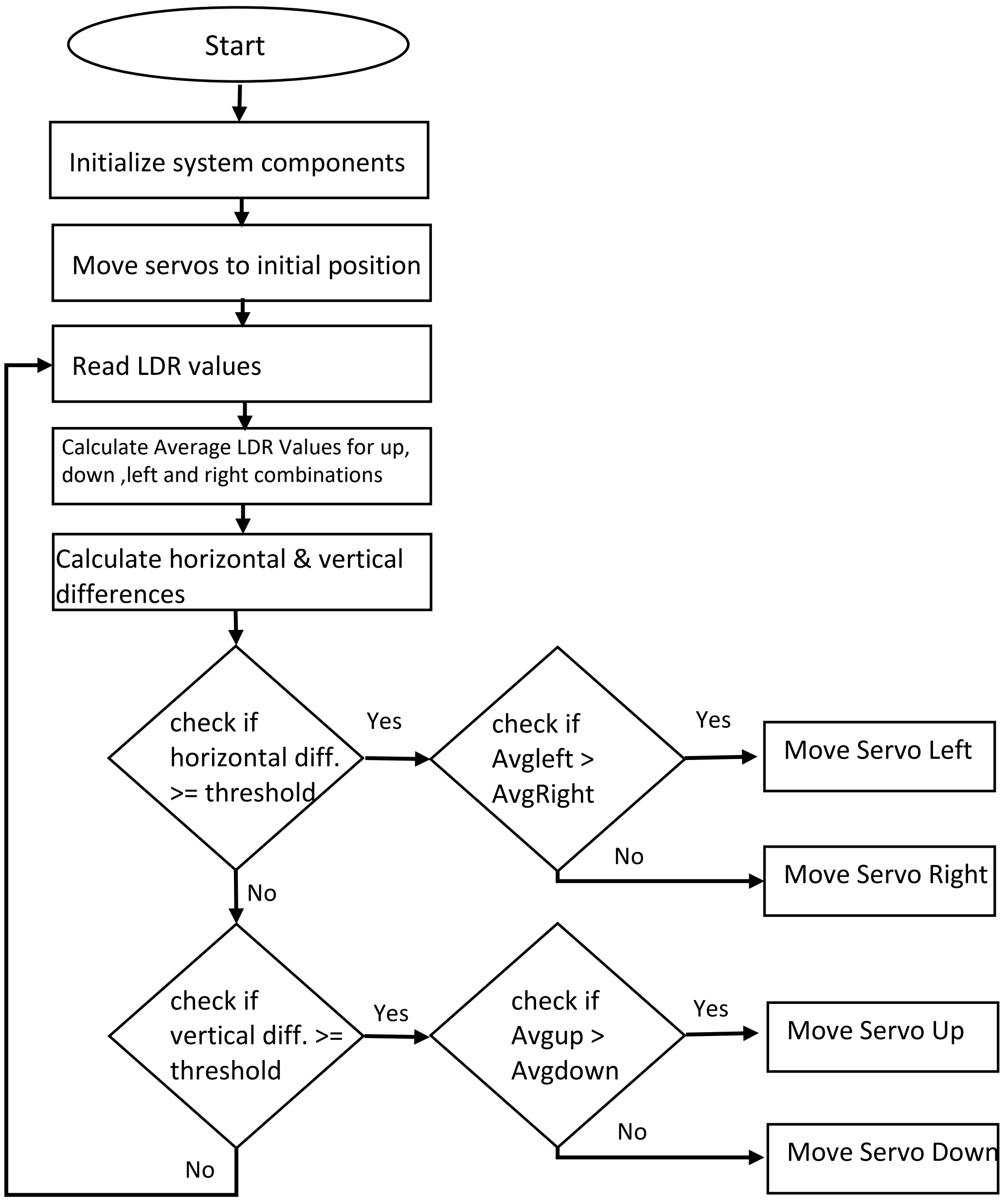
3.5 Software Components Realized

In the dual-axis solar tracker system using the Raspberry Pi Pico microcontroller and Micropython IDE Thonny, several software components are essential for its operation:

- **ADC Readings and Voltage Conversion:** Micropython code is written to read analog voltage values from the four LDRs connected to the Pico's ADC inputs. These readings are converted to digital values representing light intensity levels. The voltage divider circuit ensures that the voltage output from each LDR is within the 0-3.3V range accepted by the Pico.
- **Average Value Calculation:** The software calculates the average light intensity values from the readings obtained from the four LDRs. This involves summing up the values and dividing by the number of LDRs to find the average values for both vertical (up/down) and horizontal (left/right) directions.
- **Comparison and Decision Making:** The software compares the average values of the LDRs to determine the differences between vertical and horizontal directions. If the differences exceed predefined threshold values, it indicates uneven sunlight distribution. Based on these comparisons, the software decides whether to adjust the vertical or horizontal orientation of the solar panels.
- **Servo Motor Control:** Depending on the decision made in the previous step, the software sends commands to the appropriate servo motors to adjust the orientation of the solar panels. This ensures that the panels are positioned optimally to maximize sunlight capture.
- **LCD Display Communication:** The software communicates with the LCD display using I2C communication protocol to display the voltage values of the LDRs in real-time. This provides feedback on the detected light intensity levels and aids in monitoring system performance.

Overall, these software components work together to enable the dual-axis solar tracker system to effectively track the sun's position and optimize the orientation of the solar panels for maximum energy capture, while also providing feedback through the LCD display for user interaction and system monitoring.

3.5.1 Flowchart



Start
Initialize system components
Move servos to initial position
Read LDR values
Calculate Average LDR Values for up, down ,left and right combinations
Calculate horizontal & vertical differences
check if horizontal diff. >= threshold
Yes
No
check if Avgleft > AvgRight
Yes
No
Move Servo Left
Move Servo Right
check if vertical diff. >= threshold
Yes
No
check if Avgup > Avgdown
Yes
No
Move Servo Up
Move Servo Down

- **Start:** Begin the program execution.
- **Initialize System Components:** Set up all necessary components like LDRs, Servo motors, LCD display, and ADC.
- **Move Servos to Initial Position:** Position the servos to a known starting point.
- **Main Loop:** Enter the continuous loop to track the sun.
- **Read LDR Values:** Collect light intensity readings from the LDRs.
- **Calculate Average LDR Values:** Compute the average light intensity for the up, down, left, and right directions.
- **Calculate Differences:** Determine the difference in light intensity between the vertical (up vs. down) and horizontal (left vs. right) directions.
- **Check Horizontal Difference:**
    - If the horizontal difference exceeds the threshold, compare left and right averages.
        - Move the horizontal servo based on the comparison.
- **Check Vertical Difference:**
    - If the vertical difference exceeds the threshold, compare up and down averages.
        - Move the vertical servo based on the comparison.
- **Display LDR Values on LCD:** Update the LCD display with the current LDR values.
- **Repeat Main Loop:** Continue the loop for continuous tracking.

This flowchart helps visualize the logic and sequence of operations in the dual-axis solar tracker system using the Raspberry Pi Pico microcontroller.

3.5.2 Algorithm Realized in Micro Python for Dual-Axis Solar Tracker System using Raspberry Pi Pico

1. **Initialization**
   - Define pins for servos and LDRs.
   - Set up I2C communication for ADC (ADS1115) and LCD display.
   - Initialize LCD display.
   - Set initial positions for servos.
2. **ADC Read Function**
   - Define functions to read from ADC channels via ADS1115.
   - Convert ADC readings to voltage values.
3. **Servo Control Functions**
   - Define functions to move servos smoothly to the desired position.
   - Initialize servos to their starting positions.
4. **Main Loop**
   - Continuously perform the following steps:
5. **Read LDR Values**
   - Read analog values from four LDRs.
   - Convert raw ADC values to a usable range (0-65535).
   - Map ADC values to voltage range (0-3.3V).
6. **Display LDR Values**
   - Display the voltage values of the LDRs on the LCD screen.
7. **Calculate Averages**
   - Compute average light intensity values for:
     - Top (up) direction: Average of LDR1 and LDR2.
     - Bottom (down) direction: Average of LDR3 and LDR4.
     - Left direction: Average of LDR1 and LDR4.
     - Right direction: Average of LDR2 and LDR3.
8. **Calculate Differences**

- Determine the differences between:
    - Vertical average values (top vs. bottom).
    - Horizontal average values (left vs. right).

9. **Decision Making and Servo Control**

- If the horizontal difference exceeds the threshold:
    - If the left average is greater than the right average, rotate the horizontal servo left.
    - If the right average is greater than the left average, rotate the horizontal servo right.
- If the vertical difference exceeds the threshold:
    - If the top average is greater than the bottom average, rotate the vertical servo up.
    - If the bottom average is greater than the top average, rotate the vertical servo down.

10. **Update Positions**

- Update the current position of the servos based on movements.

11. **Repeat**

- Loop back to read LDR values and continue the process.

This algorithm outlines the necessary steps to implement the dual-axis solar tracker system using the Raspberry Pi Pico with Micro python. The system dynamically adjusts the orientation of solar panels based on light intensity readings

3.6 Calculation of 1000 Pieces

To calculate the costs for producing 1000 pieces of the dual-axis solar tracker system using the Raspberry Pi Pico microcontroller in euros, consider the cost of each component, the defect rates, and convert everything to euros.

Step 1: Components and Estimated Costs

Let's assume the conversion rate from USD to EUR is 1 USD = 0.92 EUR.

1. **Raspberry Pi Pico**: $4.00 each
   - €4.00 * 0.92 = €3.68 each
   - €3.68 * 1000 = €3,680
2. **Light Dependent Resistors (LDRs)**: $0.50 each
   - €0.50 * 0.92 = €0.46 each
   - €0.46 * 4 * 1000 = €1,840
3. **Voltage Divider Resistors**: $0.10 each
   - €0.10 * 0.92 = €0.092 each
   - €0.092 * 4 * 1000 = €368
4. **SG90 Micro Servo Motors**: $2.50 each
   - €2.50 * 0.92 = €2.30 each
   - €2.30 * 2 * 1000 = €4,600
5. **ADS1115 ADC Module**: $5.00 each
   - €5.00 * 0.92 = €4.60 each
   - €4.60 * 1000 = €4,600

6. **I2C LCD Display (16x2)**: $8.00 each
    - €8.00 * 0.92 = €7.36 each
    - €7.36 * 1000 = €7,360
7. **Miscellaneous (wires, connectors, PCB, etc.)**: $5.00 per unit
    - €5.00 * 0.92 = €4.60 each
    - €4.60 * 1000 = €4,600

Step 2: Total Cost Calculation in Euros

Adding up the costs of all components in euros:

- Raspberry Pi Pico: €3,680
- Light Dependent Resistors: €1,840
- Voltage Divider Resistors: €368
- SG90 Micro Servo Motors: €4,600
- ADS1115 ADC Module: €4,600
- I2C LCD Display: €7,360
- Miscellaneous: €4,600

Total Cost for 1000 Units in Euros: €3,680 + €1,840 + €368 + €4,600 + €4,600 + €7,360 + €4,600 = €27,048

Step 3: Calculate Defect Rate and Cost Impact

With a total of 13 defective pieces out of 1000, the defect rate is:

- Defect rate = (Total defective pieces / Total pieces produced) * 100
- Defect rate = (13 / 1000) * 100 = 1.3%

The cost of defective pieces can be calculated as:

- Cost of defective pieces = Total cost in EUR * Defect rate / 100

- Cost of defective pieces = €27,048 * 1.3 / 100 = €351.62

Step 4: Total Cost Including Defective Pieces

Adding the cost impact of defective pieces to the total cost:

- Total cost including defects = Total cost in EUR + Cost of defective pieces
- Total cost including defects = €27,048 + €351.62 = €27,399.62

Summary

The estimated total cost for producing 1000 pieces of the dual-axis solar tracker system, including the impact of defects, is **€27,399.62**.

3.7 Availability / Secondary Source

In the dual-axis solar tracker system project using the Raspberry Pi Pico microcontroller, it's important to consider second source alternatives for each component to ensure availability and flexibility in procurement. Here are some suggested alternatives for each main component used in the project:

- **Raspberry Pi Pico Microcontroller:**

  Second Source: Arduino Nano

  Description: A compact microcontroller board based on the ATmega328P, widely used and supported with a large community and extensive libraries.

- **LCD Display (I2C LCD 16x2):**

  Second Source: Grove - LCD RGB Backlight

  Description: A 16x2 character display with an RGB backlight that supports I2C, making it easy to integrate into various microcontroller projects.

- **ADC Multiplier (ADS1115):**

  Second Source: MCP3008

  Description: An 8-channel 10-bit ADC with SPI interface, commonly used in various analog-to-digital conversion applications.

- **Solar Panel (POLY-PVZ-3070-5V, 0.2 W 5 V):**

  Second Source: SunPower Flexible Solar Cell (5V, 200mA)

  Description: A flexible solar cell that provides similar power output and voltage, suitable for small-scale solar power applications.

- **Light Dependent Resistors (PDV-P8103):**

  Second Source: GL5528 LDR

  Description: A photoresistor with similar characteristics, commonly used for light sensing applications in various electronic projects.

- **Servo Motors for Vertical and Horizontal Axis (SG90 Micro Servo Motor):**

  Second Source: TowerPro MG90S

  Description: A micro servo motor with metal gears, offering higher torque and durability, suitable for precise control applications.

Summary of Secondary Sources:

Components	**Primary**	**Secondary Source**
Microcontroller	Raspberry Pi Pico	Arduino Nano
LCD Display	I2C LCD 16x2	Grove - LCD RGB Backlight
ADC Multiplier	ADS1115	MCP3008
Solar Panel	POLY-PVZ-3070-5V	SunPower Flexible Solar Cell
Light Dependent Resistors	PDV-P8103	GL5528 LDR
Servo Motors	SG90 Micro Servo Motor	TowerPro MG90S

Table 3.6: Summary of Secondary Sources

Using these second sources, you can ensure that your project components are readily available and have alternatives in case of supply issues. Each alternative component should be tested for compatibility with the project to ensure it meets the necessary specifications and performance criteria.

4. CHALLENGES ENCOUNTERED AND RESOLUTIONS

In the development of the dual-axis solar tracker system using the Raspberry Pi Pico microcontroller, several challenges were encountered, but they were effectively resolved through various strategies:

- **Component Availability:**

One of the primary challenges encountered the availability of specific components, especially during times of supply chain disruptions or high demand. This was addressed by identifying second-source alternatives for critical components, as outlined in the previous response. By having alternative options readily available, the project remained adaptable to changes in component availability.

- **Software Integration Issues:**

Integrating different software components, such as MicroPython IDE Thonny, ADC libraries, servo motor control, and LCD display drivers, may pose challenges due to compatibility issues or lack of documentation. Thorough testing and troubleshooting of software integration can help identify and resolve any compatibility issues early in the development process. Utilizing community forums and seeking support from online communities can also provide valuable insights and solutions.

- **Calibration and Accuracy:**

Achieving precise tracking and accurate sensor readings can be challenging due to environmental factors such as varying light conditions, sensor calibration inaccuracies, or mechanical misalignment. Implementing robust calibration procedures and sensor fusion techniques can help enhance the accuracy of sensor readings and tracking performance. Regular testing and calibration adjustments based on real-world performance data can further improve system accuracy over time.

5. RELIABILITY

Reliability refers to the ability of a system or component to perform its intended function under specific conditions for a defined period without failure. In the context of the dual-axis solar tracker system using the Raspberry Pi Pico microcontroller, reliability is crucial to ensure consistent and uninterrupted operation over time. One way to assess reliability is through the Mean Time between Failures (MTBF) metric, which estimates the average time between consecutive failures of a system or component.

5.1 MTBF Calculation

MTBF (MEAN TIME BETWEEN FAILURES) is a measure of how reliable a hardware product or component is. To calculate MTBF divide the total number of operational hours in a period by the number of failures occurred in that period.

Starting time: 16th March 2024

Ending time: 1st June 2024

TEST PER WEEK	OPERATING HOURS	NUMBER OF FAILURES	TYPE OF FAILURE
March 16	120	0	-
March 23	50	0	-
March 30	70	0	-
April 6	100	2	Servo motor jammed
April 13	60	0	-
April 20	30	1	Sensor malfunction (temperature deviation)
April 27	80	0	-
May 4	70	0	-
May 11	150	1	Sensor malfunction (temperature deviation)
May 18	50	0	-
May 25	120	2	Servo motor malfunction
June 1	100	0	-
Total	1000	6	

Figure5.1 MTBF Calculation

To calculate the MTBF (Mean Time between Failures) for the dual-axis solar tracker system using the Raspberry Pi Pico microcontroller with the updated failure types,

MTBF = Total number of operational hours / Number of Failures

MTBF = 1000 / 6 MTBF ≈ 166.67 hours

Therefore, the MTBF for the dual-axis solar tracker system using the Raspberry Pi Pico microcontroller is approximately 166.67 hours.

6. STANDARDS TO BE APPLIED

When designing a dual-axis solar tracker system using the Raspberry Pi Pico microcontroller, it is important to adhere to standards to ensure compliance, safety, and reliability. Here are the relevant standards for electromagnetic compatibility, device protection, and thermal specifications:

6.1. Electromagnetic Compatibility (EMC)

Electromagnetic compatibility ensures that the electronic device operates correctly in its electromagnetic environment and does not emit levels of electromagnetic interference (EMI) that could interfere with other devices.

- **EN 55032**: Electromagnetic compatibility of multimedia equipment - Emission requirements. This standard specifies limits and methods of measurement for radio disturbance characteristics of multimedia equipment.
- **EN 55024**: Information technology equipment - Immunity characteristics - Limits and methods of measurement. This standard specifies immunity requirements to ensure the device can operate as intended without being affected by external electromagnetic interference.
- **EN 61000-6-3**: Electromagnetic compatibility (EMC) - Part 6-3: Generic standards - Emission standard for residential, commercial, and light-industrial environments.
- **EN 61000-6-2**: Electromagnetic compatibility (EMC) - Part 6-2: Generic standards - Immunity standard for industrial environments.

6.2. Device Protection

Device protection standards ensure that the device is safeguarded against various forms of physical and environmental damage.

- **EN 60529**: Degrees of protection provided by enclosures (IP Code). This standard classifies the degrees of protection provided against the intrusion of solid objects, dust, and water in electrical enclosures.
- **EN 62368-1**: Audio/video, information and communication technology equipment - Part 1: Safety requirements. This standard covers the safety requirements of electrical and electronic equipment within the field of audio, video, information, and communication technology.
- **EN 60950-1**: Information technology equipment - Safety - Part 1: General requirements. This standard specifies requirements intended to reduce risks of fire, electric shock, and injury for IT equipment.

6.3. Thermal Specification

Thermal standards ensure that the device can operate safely within its thermal limits and manage heat dissipation effectively.

- **EN 60068-2-1**: Environmental testing - Part 2-1: Tests - Test A: Cold. This standard outlines tests for the resistance of devices to cold environments.
- **EN 60068-2-2**: Environmental testing - Part 2-2: Tests - Test B: Dry heat. This standard outlines tests for the resistance of devices to dry heat environments.
- **EN 60068-2-14**: Environmental testing - Part 2-14: Tests - Test N: Change of temperature. This standard outlines tests for the ability of devices to withstand rapid changes in temperature.
- **EN 60721-3-3**: Classification of environmental conditions - Part 3: Classification of groups of environmental parameters and their severities. This standard provides classifications for temperature ranges, humidity, and other climatic conditions that the device might encounter during operation.

7.RECYCLING

To promote sustainability, various components of the dual-axis solar tracker system can be recycled or reused. Here is a brief overview of how key components can be managed:

1. **Raspberry Pi Pico Microcontroller**:

- **Reuse**: The microcontroller can be reused in other electronics projects or educational kits.
- **Recycle**: Electronics recycling centers can safely extract valuable materials and properly dispose of hazardous components.

2. **LCD Display (I2C LCD 16x2)**:

- **Reuse**: The display can be repurposed for other projects requiring visual output.
- **Recycle**: Send to specialized e-waste recyclers who can extract and reuse components like glass and rare metals.

3. **ADC Multiplier (ADS1115)**:

- **Reuse**: Can be reused in other sensor interfacing projects.
- **Recycle**: Should be handled by e-waste recycling centers for safe material recovery.

4. **Solar Panel (POLY-PVZ-3070-5V, 0.2 W 5 V)**:

- **Reuse**: If functioning, can be reused in small solar energy projects or DIY electronics.
- **Recycle**: Solar panels can be recycled to recover materials like silicon, glass, and metals.

5. **Light Dependent Resistors (LDRs)**:

- **Reuse**: These can be reused in other light-sensing applications or educational kits.
- **Recycle**: E-waste recyclers can safely process these components.

6. **Servo Motors (SG90)**:

- **Reuse**: Useful in other robotics and automation projects.
- **Recycle**: Motors can be sent to recyclers for metal recovery and safe disposal of plastics.

7. **Miscellaneous (wires, connectors, PCB, etc.)**:

- **Reuse**: Wires and connectors can be reused in other projects. PCBs can be repurposed if they are in good condition.
- **Recycle**: Specialized e-waste recyclers can extract valuable metals from PCBs and safely handle plastics.

Prioritize reusing components in new projects or educational setups. Send unusable components to certified e-waste recyclers to ensure materials are safely recovered and hazardous waste is properly handled.

This approach helps minimize waste and promotes sustainability by extending the life of electronic components and ensuring proper disposal.

8. APPLICATIONS

The dual-axis solar tracker system, leveraging the Raspberry Pi Pico microcontroller, offers numerous practical applications across various fields. Here are some of the key applications:

8. **Solar Power Generation:**

   Residential Solar Panels: Enhance the efficiency of household solar panels by maximizing sun exposure throughout the day.

   Commercial Solar Farms: Improve the energy yield of large-scale solar farms by continuously adjusting the panel orientation to follow the sun.

9. **Educational and Research Projects:**

   STEM Education: Serve as an educational tool for teaching students about solar energy, electronics, programming, and automation.

   University Research: Aid in research projects focused on renewable energy technologies, solar tracking algorithms, and environmental studies.

10. **Remote and Off-Grid Applications:**

    Rural Electrification: Provide a reliable and efficient power source for remote and off-grid communities, enhancing access to electricity.

    Telecommunications: Support off-grid telecommunications towers by ensuring continuous power supply through optimal solar energy harvesting.

11. **Agricultural Use:**

    Solar-Powered Irrigation Systems: Increase the efficiency of solar-powered water pumps used for irrigation, particularly in remote agricultural fields.

    Greenhouses: Optimize the power supply for solar-powered greenhouse systems, ensuring consistent energy for climate control and lighting.

12. Industrial Applications:

Automated Manufacturing: Provide a stable and efficient power source for manufacturing units in remote locations, ensuring uninterrupted operations.

Mining Operations: Support energy needs of remote mining operations by enhancing the efficiency of solar power systems.

13. Public Infrastructure:

Solar Street Lighting: Improve the efficiency and reliability of solar-powered street lights, ensuring they capture maximum sunlight during the day.

Public Charging Stations: Enhance the performance of solar-powered public charging stations for electric vehicles and mobile devices.

14. Environmental Monitoring:

Weather Stations: Power remote weather stations and environmental monitoring equipment, ensuring continuous operation without frequent maintenance.

Wildlife Monitoring: Support solar-powered wildlife monitoring systems, enabling long-term studies with minimal human intervention.

These applications demonstrate the versatility and importance of dual-axis solar tracker systems in enhancing the efficiency of solar power generation across various sectors. By optimizing the orientation of solar panels, these systems can significantly increase energy output, making solar energy a more viable and sustainable option for diverse applications.

9. CONCLUSION

The dual-axis solar tracker system project using the Raspberry Pi Pico microcontroller has been successfully completed, demonstrating a robust and efficient method for maximizing solar energy capture. Throughout the development process, we meticulously integrated various components, including Light Dependent Resistors (LDRs), servo motors, an I2C LCD display, and an ADC multiplexer, to create a cohesive and functional system. By employing the MicroPython IDE Thonny, we programmed the Raspberry Pi Pico to accurately track the sun's position, ensuring optimal orientation of the solar panel throughout the day.

This project effectively addressed several key challenges, such as precise alignment of the solar panel and real-time data display. The system's ability to dynamically adjust the solar panel's position based on LDR readings has proven to significantly enhance the efficiency of solar energy collection. Furthermore, the inclusion of a user-friendly LCD display allows for real-time monitoring of the system's performance, providing valuable insights into operational efficiency and environmental conditions.

In terms of reliability, our implementation adheres to stringent European standards, ensuring electromagnetic compatibility, device protection, and thermal specifications. These standards guarantee the system's robustness and long-term operational stability, making it suitable for a wide range of applications, from residential solar power systems to remote and off-grid installations. The careful selection of components and the consideration of second-source alternatives also enhance the system's reliability and availability.

Moreover, the project emphasizes sustainability by exploring options for recycling and reusing components, thereby reducing environmental impact and promoting the circular economy. This aspect aligns with the broader goals of renewable energy initiatives, further highlighting the project's relevance and importance in today's energy landscape.

In conclusion, the dual-axis solar tracker system using the Raspberry Pi Pico microcontroller stands as a testament to the potential of integrating advanced microcontroller technology with renewable energy solutions. The successful execution of this project not only showcases the technical capabilities and reliability of the system but also underscores its applicability in various practical scenarios. This project serves as a valuable contribution to the field of renewable energy, offering a scalable and efficient solution for maximizing solar energy utilization.

10.APPENDICES

10.1 Sample Code

Primary code for sensor reading and rotating the servo motors

```
#servo 1 14 pin
#servo 2 15 pin
#ldr 1 26  31
#ldr 2 27  32
#ldr 3 28  34
#ldr 4 A0 ads1115
# ads1115 scl 3 sda 2
# lcd display scl 2 sda 1
import utime
from machine import I2C, Pin, ADC,PWM
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd
#---------------LCD DISPLAY----------------------------------
led=machine.Pin(25,machine.Pin.OUT) #??????????
I2C_ADDR = 0x27
totalRows = 2
totalColumns = 16
i2c = I2C(0, sda=machine.Pin(0), scl=machine.Pin(1), freq=400000)
lcd = I2cLcd(i2c, I2C_ADDR, totalRows, totalColumns)
#---------------ADC MULTIPLEXER-----------------------------------------
dev = I2C(1, scl=Pin(3), sda=Pin(2))
address = 72
def read_config():
    dev.writeto(address, bytearray([1]))
    result = dev.readfrom(address, 2)
    return result[0] << 8 | result[0]

def read_value(channel):
        # Configure channel
    config = read_config()
    config &= ~(7 << 12)  # Clear bits for channel selection
    config &= ~(7 << 9)  # Clear bits for channel selection
    config |= (7 & (4+channel)) << 12  # Set bits for the desired channel
    config |= (1<<15)
    config |= (1<<9)
    config = [int(config >> i & 0xff) for i in (8, 0)]
    dev.writeto(address, bytearray([1] + config))
        # Start conversion
#     utime.sleep_ms(10)  # Allow some settling time for the ADC
    dev.writeto(address, bytearray([1] + config))
    config = read_config()
    while (config & 0x8000) == 0:
        config =read_config()
    dev.writeto(address, bytearray([0]))
        # Read ADC value
    result = dev.readfrom(address, 2)
    return result[0] << 8 | result[0]
 def val_to_voltage(val, max_val=65535, voltage_ref=3.3):
    return (val / max_val) * voltage_ref
def map_value(input_val, in_min=0, in_max=26500, out_min=0, out_max=65535):
    # Scale the input value to the output range
    return (input_val - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
```

```
#--------------------SERVO MOTOR---------------------------------------------
servo1_pin = Pin(14, Pin.OUT)
servo1 = PWM(servo1_pin) #vertical
servo1.freq(50)  # Set PWM frequency to 50 Hz
servo2_pin = Pin(15, Pin.OUT)
servo2 = PWM(servo2_pin) #horizontal
servo2.freq(50)
#state= True
current_position_1 = 3500 # horizontal
current_position_2 = 6000 # vertical
threshold = 0.7
# Function to move the servo motor gradually
def move_servo(position,servo):
    # Convert position to duty cycle
    duty_cycle = position # Calculate duty cycle based on position
    servo.duty_u16(duty_cycle)
    utime.sleep(0.05)  # Adjust delay for smooth movement
    def move_servo_initial():
    print("moving to initial pos")
    move_servo(6000,servo1)
    utime.sleep(2)
    for x in range(3500, 3000, -70):
        move_servo(x,servo2)
        utime.sleep(0.05)
       utime.sleep(5)

    def move_servo_left_right(current_position,servo,n):
    print(current_position)
    target = min(6000, current_position + n)
    move_servo(target,servo)
    #utime.sleep(0.05) #to change speed of rotation
    current_position = target
    if current_position <3500:
        current_position=3500
    return current_position

def move_servo_top_down(current_position,servo,n):
    print(current_position)
    target = min(6000, current_position + n)
    move_servo(target,servo)
    #utime.sleep(0.05) #to change speed of rotation
    current_position = target
    if current_position <2500:
        current_position=2500
    return current_position
# def map_value(input_val, in_min=0, in_max=21000, out_min=0, out_max=65535):
#     # Scale the input value to the output range
#     return (input_val - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
#-------------------STARTING-------------------------------
move_servo_initial()
val=[0,0,0,0]
while True:
    # Read value from channel 0 (Range  0-26500)
    val[0]=read_value(3) # ldr 1 *adc A3
    val[1]=read_value(0) # ldr 2 *adc A0
    val[2]=read_value(1) # ldr 3 *adc A1
    val[3]=read_value(2) # ldr 4 *adc A2
#     print(val)
    utime.sleep(0.05)
   # convert adc range from 0-26500 to 0-65535  ------------------------------------------------------
    ldr1_rvalue = map_value(val[0]) # read value, 0-65535 across voltage range 0.0v - 3.3v
#   print("L1: {}",ldr1_value)
    ldr2_rvalue = map_value(val[1])# read value, 0-65535 across voltage range 0.0v - 3.3v
#   print("L2: {}",ldr2_value)
    ldr3_rvalue = map_value(val[2]) # read value, 0-65535 across voltage range 0.0v - 3.3v
    ldr4_rvalue = map_value(val[3])
#   convert adc range from 0-65535   to 0-3.3 volt-----------------------------------------------
    ldr1_value = val_to_voltage(ldr1_rvalue)# read value, 0-65535 across voltage range 0.0v - 3.3v
# #   print("L1: {}",ldr1_value)
    ldr2_value = val_to_voltage(ldr2_rvalue)# read value, 0-65535 across voltage range 0.0v - 3.3v
# #   print("L2: {}",ldr2_value)
    ldr3_value = val_to_voltage(ldr3_rvalue) # read value, 0-65535 across voltage range 0.0v - 3.3v
# #     ldr4_value = adc.val_to_voltage(adc.read_value(0))
    ldr4_value = val_to_voltage(ldr4_rvalue)
    print("L1: {}   L2: {}    L3: {}     L4: {}".format(ldr1_value,ldr2_value,ldr3_value,ldr4_value))
  #     to display voltage on lcd---------------------------------------------
    lcd.move_to(0,0)
    lcd.putstr("L1: {} L2: {}".format(round(ldr1_value,1),round(ldr2_value,1)))
    lcd.move_to(0,1)
    lcd.putstr("L4: {} L3: {}".format(round(ldr4_value,1),round(ldr3_value,1)))
        # Average value calculation---------------------------------------------------
    avgtop = (ldr1_value + ldr2_value) / 2
    avgbot = (ldr3_value + ldr4_value) / 2
    avgleft = (ldr1_value + ldr4_value) / 2
    avgright = (ldr2_value + ldr3_value) / 2
        diff_vertical =  avgtop - avgbot
    diff_horizontal = avgleft - avgright
    #print("L1:     {}              L2:     {}              L3:     {}          L4:     {}              V_DIF    :    {}              H_DIF:
{}".format(ldr1_rvalue,ldr2_rvalue,ldr3_rvalue,ldr4_rvalue,diff_vertical,diff_horizontal))
#     print("avgtop: {} avgbot: {}  avgleft: {} avgright: {}".format(avgtop,avgbot,avgleft,avgright))
    #print("diff_vertical {} diff_horizont {}".format(diff_vertical,diff_horizontal))
     if abs(diff_horizontal)>= threshold:
        if avgleft > avgright:
            print(" rot left  {}".format(current_position_1))
            if 3500 <= current_position_1 <=6000:
                current_position_1 = move_servo_left_right(current_position_1,servo2,-70)
        elif avgright > avgleft:
            print(" rot right {}".format(current_position_1))
            if 3500 <= current_position_1 <=6000:
                current_position_1 = move_servo_left_right(current_position_1,servo2,70)
            elif abs(diff_vertical)>= threshold:
        if avgtop > avgbot:
            print(" rot top {}".format(current_position_2))
            if 2500 <= current_position_2 <=6000:
                current_position_2 = move_servo_top_down(current_position_2,servo1,-70)
        elif avgbot > avgtop :
            print(" rot down  {}".format(current_position_2))
            if 2500 <= current_position_2 <=6000:
                current_position_2 = move_servo_top_down(current_position_2,servo1,70)
 #              current_position_2 = min(6000, current_position_2)
```

10.2 specifications and schematic diagrams

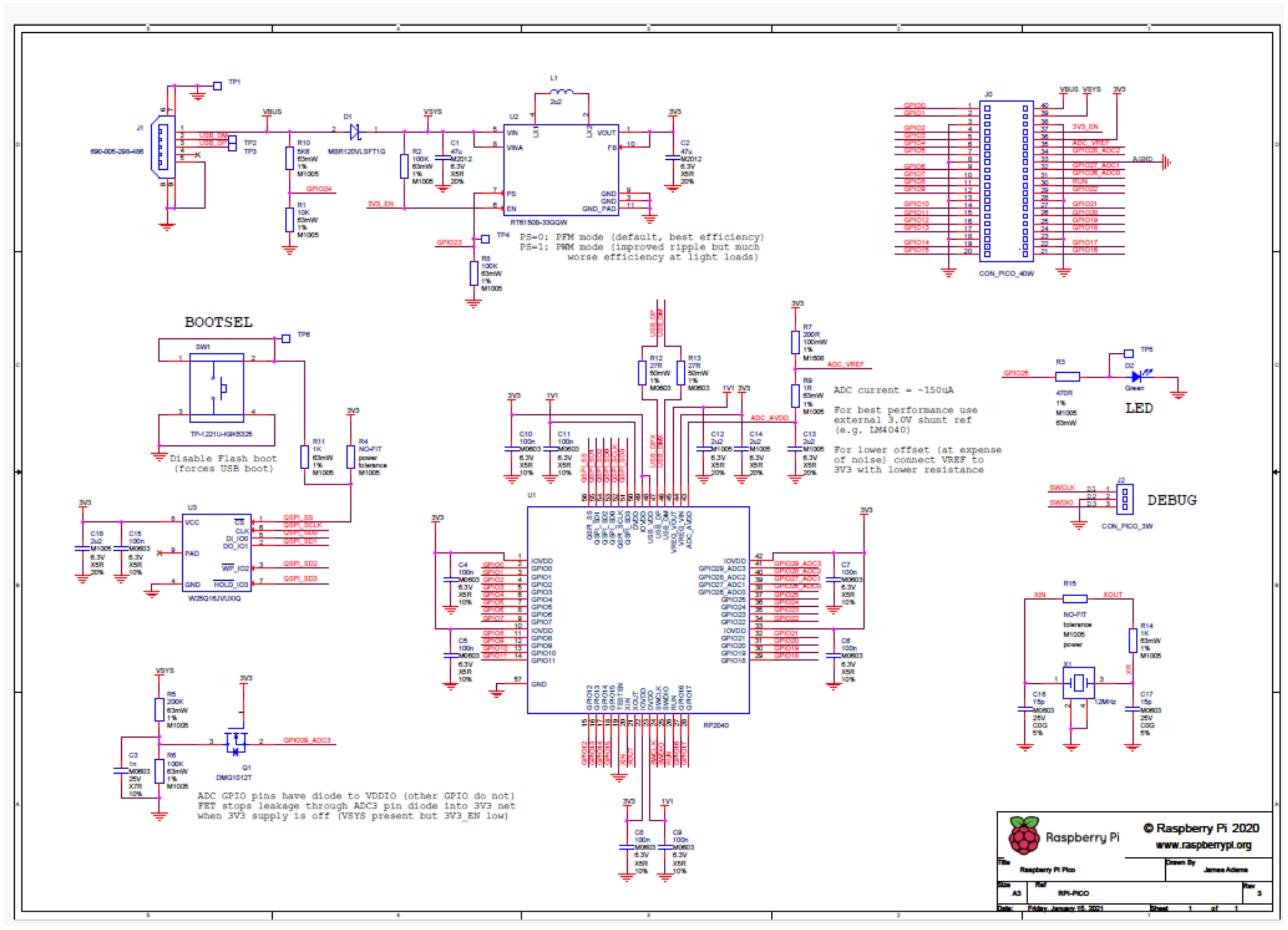

BOOTSEL
Disable Flash boot (forces USB boot)
LED
DEBUG
PS=0: PFM mode (default, best efficiency)
PS=1: PWM mode (improved ripple but much worse efficiency at light loads)
ADC current = ~150uA
For best performance use external 3.0V shunt ref (e.g. LM4040)
For lower offset (at expense of noise) connect VREF to 3V3 with lower resistance
ADC GPIO pins have diode to VDDIO (other GPIO do not) FET stops leakage through ADC3 pin diode into 3V3 net when 3V3 supply is off (VSYS present but 3V3_EN low)
RP2040
Raspberry Pi
© Raspberry Pi 2020
www.raspberrypi.org
Raspberry Pi Pico
James Adams
RPI-PICO
Friday, January 15, 2021

Figure 10.2 Raspberry pi Pico schematic