Table of Contents

1. Introduction

1.1 Purpose

The purpose of this documentation is to guide you through building a smart solar system using ESP32 as the microcontroller. The system will use a light sensor to track the sun's direction and adjust the servo motor to face the sun. Additionally, AWS web services such as Dynamo DB will be utilized to store solar data for monitoring and analysis.

The implementation process involves hardware setup, connecting the ESP32 to the light sensor and servo motors, and establishing a reliable power supply using lithium batteries and a solar cell. In terms of software setup, users need to install the Arduino IDE, configure an AWS account, and set up AWS IoT Core and DynamoDB to handle data storage. The programming aspect involves writing code to read light sensor data, calculate the sun's position, and control the servo motors accordingly. The system's operation encompasses solar tracking algorithms, data storage in AWS, and power management techniques to optimize energy usage.

1.2 Overview

The smart solar system is a sophisticated setup comprising essential components such as the ESP32 microcontroller, a light sensor, two servo motors, a solar cell, and two lithium batteries. The core functionality revolves around the ESP32, which acts as the central control unit. It reads data from the light sensor, enabling it to gauge the sun's position in the sky accurately. Based on this information, the ESP32 intelligently adjusts the angles of the two servo motors. These motors are responsible for orienting the solar panel to face the sun optimally throughout the day, ensuring maximum solar energy absorption. As a result, the system enhances the overall efficiency of solar power generation.

To ensure continuous operation and energy autonomy, the solar cell comes into play. It generates power from sunlight and utilizes it to charge the two lithium batteries. These batteries serve as the primary power source for the ESP32 and other components of the system, facilitating independent and sustainable operation. Furthermore, to enable seamless data management and advanced analytics, the ESP32 establishes communication with Amazon Web Services (AWS) IoT Core. The system transmits solar data, including light intensity readings and servo motor angles, to AWS DynamoDB—a powerful cloud-based database service. This integration empowers users to remotely monitor and analyze the solar system's performance, making data-driven decisions to optimize energy generation and usage. The AWS cloud infrastructure ensures data security, scalability, and accessibility from anywhere in the world, making the smart solar system a robust and cutting-edge renewable energy solution.

1.3 System Components

The components required for building the smart solar system are as follows:

1. ESP32 Development Board
2. Light Sensor (e.g., LDR or Photodiode)
3. Two Servo Motors
4. Solar Cell (3 to 7V)
5. Two 3.3V Lithium Batteries
6. Breadboard and Jumper Wires
7. Power Supply (e.g., USB Cable or Battery Pack)

ESP32 Development Board:vThe ESP32 is a powerful and versatile microcontroller that forms the heart of the smart solar system. It features built-in Wi-Fi and Bluetooth capabilities, making it ideal for wireless communication and integration with AWS IoT Core. The ESP32's processing capabilities and ample GPIO pins enable it to efficiently read data from the light sensor and control the servo motors for precise solar tracking. Its ease of programming and compatibility with Arduino IDE make it accessible for both beginners and experienced developers.

Light Sensor (e.g., LDR or Photodiode): The light sensor is a crucial component in the system, responsible for detecting and measuring light intensity. Light Dependent Resistors (LDRs) or photodiodes are commonly used as light sensors in solar tracking applications. As the sun's position changes throughout the day, the intensity of light falling on the sensor varies. The ESP32 utilizes this data to calculate the sun's direction accurately, enabling the servo motors to adjust the solar panel accordingly.

Two Servo Motors: The two servo motors are pivotal in the solar tracking mechanism. These motors are responsible for rotating the solar panel to align with the sun's position. The servo motors' precise control and ability to rotate to specific angles ensure the solar panel maximizes sun exposure, resulting in increased energy production. By adjusting the angles of the servo motors based on the light sensor's readings, the system can efficiently track the sun throughout the day.

Solar Cell (3 to 7V): The solar cell is the primary energy source for the smart solar system. It converts sunlight into electrical energy, generating power that charges the lithium batteries. The voltage range of the solar cell should match the requirements of the system and the lithium batteries for efficient energy harvesting. The use of a high-quality solar cell ensures optimal power generation, enabling the system to function independently, especially in remote and off-grid installations.

Two 3.3V Lithium Batteries: The lithium batteries act as energy storage units, enabling the system to operate during periods of low sunlight or at night. These batteries offer high energy density, longer lifespan, and lightweight properties, making them suitable for portable solar applications. The ESP32 and other components draw power from the lithium batteries, ensuring continuous

operation and power autonomy.

Power Supply (e.g., USB Cable or Battery Pack):The power supply is essential for providing initial power to the ESP32 during development and testing. It can be connected via a USB cable or an external battery pack. While testing, the power supply ensures that the system operates correctly. However, in practical applications, the solar cell and lithium batteries take over as the primary power sources, making the system self-sustaining and environmentally friendly.

1.4 Prerequisites

1. Basic knowledge of electronics and programming.
2. An AWS account for using AWS web services.
3. Arduino IDE or compatible software for programming the ESP32.

2. Hardware Setup

2.1 ESP32 Connection Diagram

Connect the ESP32 to the breadboard and power it using the 3.3V lithium batteries or a USB cable. Refer to the ESP32 datasheet or your development board's pinout diagram to make the connections.

2.2 Servo Motor Connection

Connect the two servo motors to the ESP32 using jumper wires. You will need two GPIO pins to control the direction of each servo motor.

2.3 Light Sensor Connection

Connect the light sensor (e.g., LDR) to the ESP32 analog input pin. The analog input pin will read the values from the light sensor and calculate the sun's position based on the intensity of light.

2.4 Power Supply

Connect the solar cell to the lithium batteries to charge them during daylight. Ensure the solar cell voltage range is compatible with the lithium battery voltage range.

3. Software Setup

3.1 ESP32 Programming Environment

Install the Arduino IDE or compatible software and the ESP32 board manager. Choose the appropriate ESP32 board from the board manager and install the necessary libraries.

3.2 AWS Account Setup

Create an AWS account if you don't have one. Log in to the AWS Management Console to access the AWS services needed for this project.

3.3 AWS Configuration

Create an AWS Amplify Project

- Go to the AWS Amplify Console and log in to your AWS account.
- Click on "Get Started" to create a new Amplify project.
- Choose the option to connect your app repository, and follow the instructions to connect to your code repository where the ESP32 code is stored.

Set Up AWS Amplify Backend Environment

- Once your app repository is connected, Amplify will automatically detect your codebase and present an option to set up a backend environment.
- Choose the appropriate backend environment (e.g., "Dev" for development) and let Amplify create the backend resources for you.

Set Up Authentication

- In the Amplify Console, navigate to the "Authentication" section and configure user authentication for your smart solar system app. Choose the desired authentication method, such as Amazon Cognito, to manage user authentication securely.

Set Up AWS DynamoDB

- Navigate to the "Data" section in the Amplify Console and choose "Create new API."
- Define the data model for your solar data, including attributes like timestamp, light intensity, and servo motor angles.
- Amplify will automatically create an AWS DynamoDB table based on your data model.

Accessing AWS Services from ESP32

- To allow your ESP32 to communicate with the AWS services, you'll need to use AWS Amplify libraries for IoT and DynamoDB in your ESP32 code. These libraries provide APIs for securely sending data to DynamoDB and managing authentication.

6Configure AWS Amplify on ESP32

- Set up your ESP32 development environment with the necessary libraries and dependencies for communicating with AWS Amplify.
- Use the generated AWS Amplify configuration from the backend environment to configure the authentication and DynamoDB access on the ESP32.

- Sending Solar Data to DynamoDB
- In your ESP32 code, implement the logic to read data from the light sensor and calculate the sun's direction.
- Use the AWS Amplify IoT library to securely send this solar data to the AWS DynamoDB table.

3.5 Programming the ESP32

Write the Arduino code to read data from the light sensor, calculate the sun's direction, and control the servo motors accordingly. Use the AWS IoT SDK to connect to AWS IoT Core and send the solar data to DynamoDB.

4. Operation

4.1 Solar Tracking Algorithm

The ESP32 should read the light sensor values and convert them into an angle representing the sun's direction. The solar tracking algorithm could involve comparing the light intensity from different directions to find the sun's position and adjust the servo motors accordingly.

4.2 Data Storage and Retrieval

The ESP32 will send the solar data, such as timestamp, light intensity, and servo motor angles, to AWS IoT Core. AWS IoT Core will route the data to DynamoDB for storage. You can then use AWS services like AWS Lambda or AWS Glue to analyze and visualize the solar data.

4.3 Power Management

Implement power management in your code to optimize power usage. For example, you can put the ESP32 into sleep mode during the night or when the battery voltage is low to conserve energy.

5. Conclusion

This system can efficiently track the sun's direction, optimize power usage, and store solar data for remote monitoring and analytics.

6. References

ESP32 Documentation: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
- AWS IoT Core Developer Guide: https://docs.aws.amazon.com/iot/latest/developerguide/
- AWS DynamoDB Developer Guide: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/