IOT based Sun Tracking Solar Panel

1st EN20417284 Ranasinghe E.P
Dept.of Electrical and Electronic Engineering
SLIIT
Malabe, Sri Lanka
en20417284@my.sliit.lk

2nd EN20404482 Mallawaarachi H.R
Dept.of Electrical and Electronic Engineering
SLIIT
Malabe, Sri Lanka
en20404482@my.sliit.lk

3rd EN20401580 Thilakshan D.M.H
Dept.of Electrical and Electronic Engineering
SLIIT
Malabe, Sri Lanka
en20401580@my.sliit.lk

I. Abstarct

Solar energy is a renewable resource that will never run out and can be collected in many different ways specially using solar panels. The use of solar energy has steadily grown over the last couple of decades, with more and more people realizing the tremendous value of utilizing solar panels due to the high demand and scarcity of the fossil fuels. Although the solar panels are abundantly used in present day, the power generation efficiency has been a greater challenge due to sun's movement and stillness of the solar panel. Therefore, as an engineering solution for this challenge, an IOT based sun tracking solar panel has been introduced in this report.

II. Introduction

The annual global energy consumption is getting higher from every year, and it is estimated to 580 million terajoules. Usage of non-renewable resource is getting higher with the increasing global demand for the energy. Since it is limited source, the world tries to meet the rising energy demand by depending on renewable energy resources. Sunlight is a one of the major renewable resource available and it can directly use by capturing the suns energy. When sun light directly falls on the solar panel, it will absorb the energy from the sun light and produce electrical charges. For that solar panels to have maximum efficiency, they should always place as they directly facing to the sunlight falling direction. Therefore, solar tracking devices are created to change the direction of solar panels with the change of sunlight falling direction. Solar trackers can adjust the angle of solar panel and it keeps the panel perpendicular to the sun as maximum amount of sunlight falls on it. When more sunlight strikes the solar panel, less light is reflected, and more energy can be absorbed by the panel. This process will give more solar energy which can be converted into power.

In this assignment, the designed sun taking solar panel is a handheld, portable and user-friendly device which use a 3.7V lithium iron rechargeable battery to power up the microcontroller and other components. ESP32 microcontroller was used since it consists of two cores and FreeRTOS library was used for multi-threading. The angle of the solar panel is changed by using a MG90s Metal gear servo motor according the two LDR sensor inputs. The program was built to change the servo motor rotating angle according to the ADC values which are given by LDR analog inputs. Separate LCD display with I2C module has been used to display the amount of current generated from the solar panel, and the tilted angle of the solar panel. Therefore, for the design, a voltage sensor and 1kΩ resistor have been used to measure the amount of current that generated using ohms law. Finally, the device will be connected to a webserver and the generated current and tilted angle will be shown there as well.

III. Methodology

A. Detailed Requirement Analysis

As mentioned earlier, a sun tracking solar panel has to be implemented as Real Time Operating Systems Lab 04. The following detail requirement analysis was carried out to identify and document key features of the device as well as state of its feasibility according to the proposed design.

1) Functional Requirements: According to the given device specifications, the following functions which are expected from the designing product were identified.

- **Tilting the solar panel attached to the device according to the position of the sun -** In order to obtain the maximum power generation efficiency, the device should tilt the solar panel until the solar rays falls perpendicular to the solar panel.
- **Amount of generating current should display –** Due to falling sunlight, the solar panel generates a current and it is required to measure using a sensor or any other appropriate method and display in a screen (LCD).

- **Tilted angle of the solar panel should display –** The rotating angle of the solar panel should be measured, and the tilted angle should be displayed in the same screen.
- **The components of the device should power up using rechargeable battery**
- **The device should be portable – The** device should be handheld and movable from place to place without much effort.
- **Set of push buttons to change the parameter showing in the display –** Upon pressing the push button, the displaying parameter should change on the display.

*2) **Technical Requirements**:* When achieving the key requirements of the project which are mentioned in the functional requirements section, there are number of technical issues that can happen due to poor planning of the project. Hence in this section, possible technical issues that can arise are considered.

- To tilt the solar panel according to the position of the sun and the falling sun light, a rotary mechanism has to be used. When selecting a rotary actuator for this purpose, the torque of the rotary actuator should be considered.
- As the device required to be portable and handheld, the hardware structure of the sun tracker should be made using a light weight yet strong material.
- Rechargeable and considerably higher in capacity battery setup can avoid power related issues of the device with proper voltage regulation.
- By using serial communication methods like I2C, the excess use of pins of the microcontroller can be avoided as well as can obtain a better performance.

*3) **Translational Requirements:*** For the successful implementation of the device, planning the steps of the project at the beginning is crucial. Therefore, the following steps of the device implementation were considered.

❖ Deciding the main components to be used in the project

- **ESP32 Microcontroller –** This microcontroller has 2 cores and using FreeRTOS library multi-threading can be done. In addition to that, the inbuilt Wi-Fi module can be used to connect the microcontroller to a web server as well.
- **MG90S Metal Gear Micro Servo Motor –** This is a micro servo motor with metal gears inside. This small and lightweight servo motor has a high output power which makes this servo motor vary useful for considerably higher torque applications such as this.
- **16×2 LCD display –** As this is a cost-effective display in which the following required key specifications are available, it was selected to implement the device.
    - Includes two rows which can produce 16 characters in a row
    - Can display alphanumeric characters
    - Can operate in two modes like 4-bit and 8-bit
    - 4.7 V – 5.3 V operating voltage
- **Solar panel –** This is a small-scale solar panel with 15.5 cm × 8.5 cm of dimensions, and it can produce maximum of 350 mA which is enough for this project.
- **Light Dependent Resistors (LDR) –** Two LDRs were intended to use to determine the angle of ration of the solar panel according to sunlight intensity on LDRs.
- **I2C Module –** This module was chosen to be used for the serial communication between ESP32 and 16×2 LCD display. Not only this module increases the efficiency, but also saves many pins of the microcontroller.
- **DC voltage sensor –** Using this sensor and 1kΩ resistor, the generated current can be calculated using ohms law.
- **Two rechargeable Li-ion batteries** – According the key requirements, two 5000 mAh 3.7 V Li-ion batteries were selected to use to provide enough power to the sun tracking device.

❖ Budget Calculation

Table 1 Budget Calculation

Component	Unit price (LKR)	Quantity	Amount (LKR)
ESP32 Microcontroller	2180	1	2180
MG90S Metal Gear Micro Servo Motor	750	1	750
Solar panel	600	1	600
16×2 LCD display	700	1	700
I²C Module	365	1	365
DC voltage sensor	145	1	145
Two rechargeable 5000 mAh 3.7 V Li-ion batteries	375	2	750
Light Dependent Resistor (LDR)	15	2	30
Total Amount			5520

❖Hardware Structure Implementation

The following figure is a rough sketch of the support structure which holds the solar panel. The base of the structure is made from albesia wood which makes it strong and lightweight. Then, the support pillar is made from plywood which is another lightweight material. On top the support pillar, the servo motor is

attached in a small socket made from the same wood and it can be rotated to adjust the position of the servo motor.

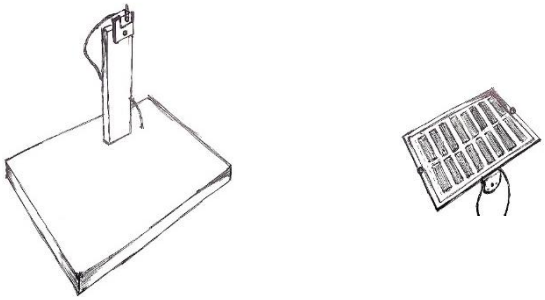

Figure 1 Initial Structure

Thereafter, the solar panel was attached to a rectangular piece of plywood which is a little broader than the size of the solar panel. Then, two LDRs are attached at the two corners of the solar panel.

❖ Software Part implementation and testing

Here in this stage, the cording, testing, and debugging was planned as follows.

Stage 1:

- Taking LDR analog inputs
- Current reading testing using DC voltage sensor & 1kΩ resistor.
- Servo motor testing
- I^2C display cording and testing

Stage 2:

- LDR analog input reading and determining the angle of rotation of the servo motor
- Displaying the current reading and calculated angle of rotation in the LCD

Stage 3:

- Mounting everything on the implemented hardware support and carry out final tests.

*4) **Operational Requirements:** To ensure the proper function of the project,*

- Since a small solar panel is using in this project, the device is required to place in a proper environment where ample amount of sunlight is present.

As the device is not designed to recharge the batteries using the solar power generated, it is required to keep an eye on the charge of the device's batteries.

B. Components used

❖ ESP 32

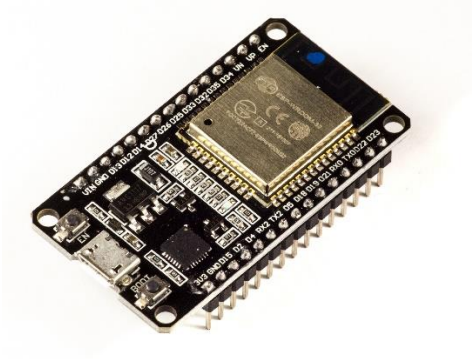

Figure 2 ESP32

A low-cost, low-power microcontroller with integrated Wi-Fi and dual-mode Bluetooth. It comes in both single core and dual core modules. This microcontroller has many features. Among those, few of the significant features are mentioned below.

- Processors:
  - CPU: Xtensa dual-core 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
  - Ultra-low power (ULP) co-processor
- Memory: 320 KiB RAM, 448 KiB ROM
- Wireless connectivity:
  - Wi-Fi: 802.11 b/g/n
- Peripheral interfaces:
  - 34 × programmable GPIOs
  - 12-bit SAR ADC up to 18 channels (Can be set to 10-bit as well)

❖ DC voltage Sensor

Figure 3 DC Voltage Sensor

As shown in the following figure, the voltage sensor is basically a Voltage Divider consisting of two resistors

with resistances of 30KΩ and 7.5KΩ (5 to 1 voltage divider). Hence the output voltage is reduced by a factor of 5 for any input voltage.

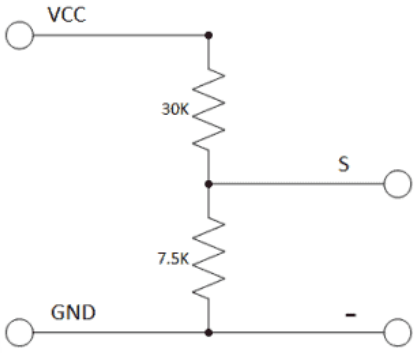

Figure 4 Internal structure of DC voltage sensor

Since ESP32 analog input pins accept voltages up to 3.3V, the input voltage should not be greater than 3.3Vx5=16.5V.

As a summery,

- Voltage input range: DC0-25 V.
- Voltage detection range: DC0.02445 V-25 V.
- Voltage analog resolution: 0.00489 V.
- Output interface: "+ "connected 5/3.3V, "-" connected GND, "s" connected Arduino AD pins.
- DC input interface: red terminal positive with VCC, negative with GND.

❖ I^2C module connected LCD (Liquide Crystal Display)

Figure 5 I2C connected module connected LCD

LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability, and programmer friendly. **16×2 LCD** is named because it has 16 Columns and 2 Rows. There are a lot of combinations available like, 8×1, 8×2, 10×2, 16×1, etc. but the most used one is the 16×2 LCD. So, it will have (16×2=32) 32 characters in total.

❖ Solar panel

Figure 6 Solar Panel

The solar panel used is a high quality 18V 2.5W polycrystalline stored energy power solar panel system solar cells charger with dimensions of 19.4 x 12 x 0.3cm. This has a high conversion rate, high efficiency output and excellent low light effect. Some of the key features of the solar panel is as follows,

- 2.5 watts of power
- 18 V of voltage
- Made from Polycrystalline Silicon
- Has 19.4 cm x 12 cm x 0.3 cm of dimensions
- 70 g in weight

❖ Rechargeable Li-ion batteries

Figure 7 Rechargeable Li -ion batteries

The chosen rechargeable li ion battery has the following key features.

- 3.7 V of voltage
- 5000 mAh capacity

❖ LDR (Light Dependent Resistor)

An LDR is a component with a (variable) resistance that varies depending on the amount of light it receives. They can be used in light sensing circuits as a result of this. To detect the intensity of the sun light this LDR is most suitable component.

Figure 8 A typical LDR

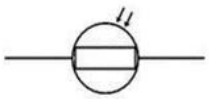

Figure 9 LDR Circuit Symbol

Table 2 LDR Parameters

PARAMETER	VALUE
Max. power dissipation at 30^{0}C	250 mW
Max. current	75 mA
Max. voltage peak	100 V
Typ. resistance at 10 lux	9 kΩ
Typ. resistance at 1000 lux	400 Ω
Dark Capacitance	3.5 pF

Following shows the relation of resistance with the light intensity hitting on the LDR,

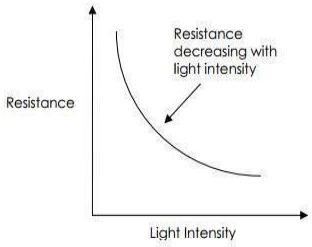

Figure 10 Typical LDR resistance VS light intensity graph

Above figure clearly indicates that when light intensity increases the resistance will decreases.

❖ MG90S Metal Gear Servo Motor

Figure 11 MG90S Metal Gear Servo Motor

This servo is small is size, lightweight and has high output power. This servo has metal gears for strength and durability. This servo can rotate approximately 180 degrees and can be able to control like a normal standard servo.

Specifications

- Weight: 13.4 g
- Dimension: 22.5 x 12 x 35.5 mm approx.
- Stall torque: 1.8 kgf·cm (4.8V), 2.2 kgf·cm (6 V)
- Operating speed: 0.1 s/60 degree (4.8 V), 0.08 s/60 degree (6 V)
- Operating voltage: 4.8 V - 6.0 V
- Dead band width: 5 µs

❖ Designed support structure

Figure 12 Designed Hardware structure

C. Method of tracking the sun using LDR sensor input

❖ Concept: -

When sun light comes to the LDR depending on the light intensity the LDR resistance will vary. In the low light intensity, the LDR resistance will be high and in higher intensities the LDR resistance will be lower. The intensity is measures from the unit called *lux.* In low intensities such as 10lux range the LDR resistance will increase up to about 9 kΩ s. In higher intensities such as 1000lux range the LDR resistance will decrease up to about 400 Ω s. when connecting an LDR, needed to connect another resistor in series with the LDR.

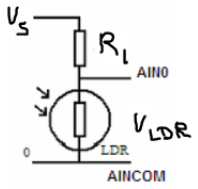

Figure 13 Connection of the LDR to M.controller

Then the LDR voltage can be calculated as follows,

$$V_{LDR} = \frac{V_{supply}}{R_{LDR} + R_1} \times R_{LDR}$$

$$V_{LDR} = \frac{V_{supply}}{((1 + R_1/R_{LDR}))}$$

Then the ADC value always vary between in 0 – 1023 when we use 10bit ADC in ESP32. By using 2 LDR input values the sun tracking was done. Connection of 2 LDR as follows,

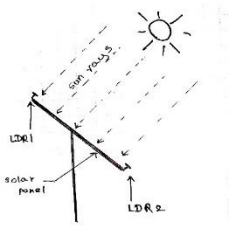

Figure 14 placement of two LDRs

When the moment that sun was positioned parallel to the solar panel and the sunrays hitting on the LDRs are exactly vertical, the two input values of the LDRs should be same. By using this condition can be able to rotate the solar panel until the two LDR input values are equal to track the sun.

❖ Method and calculations to get tilt angle of the solar panel

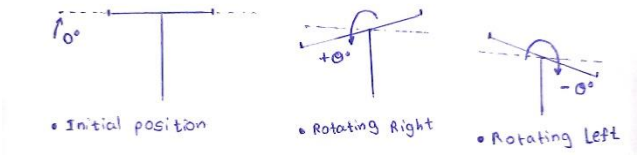

Figure 15 angle position of the panel

In the initial state the panel is on 0º But, actual servo is in 45º position. Since ESP32's operating voltage is 3.3V and PWM signal also varies from 3.3V – 0V. But operating voltage of the servo motor is 5V. Because of that 3.3V PWM signal will not be able to control a servo motor according to the correct plan. The results will be deviate from the expected results. If the program is written to rotate servo to 180º But, the servo will rotate to 90º. Because of this problem 3.3V PWM can be able to rotate 0º - 90º angle only. When the sun or the light source will change its position, the servo will rotate according to sun tracking mechanism. Then the angle will change. According to the above problem the real angle can be represent follows,

$$\textbf{\textit{Real angle}} = 45 - (\theta/2)$$

❖ Coding

The objective is to get 2 ADC values from 2 LDRs and track the Sun or the Light source using those two values. When it comes to programming, this task is separated in to two tasks. One is to get two ADC values from 2 LDRs and the other is to rotate panel according to the movement of the Sun or the Light source.

- Reading the ADC values from 2 LDRs

This task will run on Core1 on the ESP32 with a separate thread. Following is the flow chart of that task.

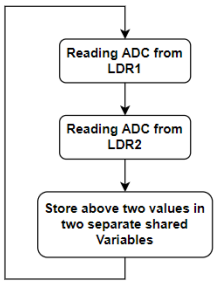

Figure 16 Task flow chart

The following task is implemented on a separate function as follows,

```
void TaskADC(void *parameters) {

  while(1){             //starting an infinity loop
    analogReadResolution(10);
    LDRValue1 = analogRead(LDRpin1); // read the value from the LDR
    LDRValue2 = analogRead(LDRpin2); // read the value from the LDR
  }
}
```

And the task was created on Core1 as follows,

```
xTaskCreatePinnedToCore(TaskADC,    //function
                        "Task ADC Core1", //name of the task
                        1024,         //stack size(bytes)
                        NULL,         //parameter to pass to function
                        1,            //giving lower priority
                        &task_ADC,       //task handle
                        1);           //Core ID
```

- Tracking of the sun using 2 ADC values.

This task will run on Core1 on the ESP32 with a separate thread. Following is the flow chart of that task

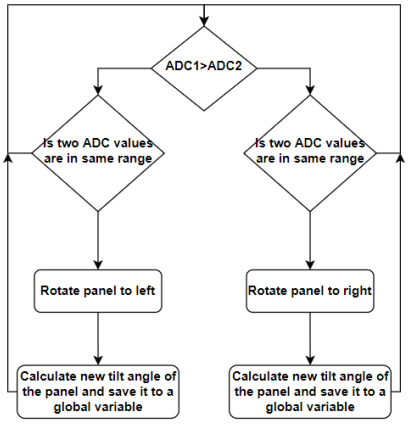

Figure 17 Task flow chart

The following task is implemented on program as follows,

```
if(LDRValue1 < LDRValue2){    //rotate left
    vTaskDelay(pdMS_TO_TICKS(10));
    while(( ((LDRValue2-50) < LDRValue1) & (LDRValue1 < (LDRValue2+50)) ) == 0){
        if(rotate_angle > 10){
          rotate_angle = rotate_angle - 3;
          myservo.write(rotate_angle);
          real_angle = 45 - (rotate_angle/2);
        }
        vTaskDelay(pdMS_TO_TICKS(10));
    }
}
```

And the task was created on Core1 as follows,

```
xTaskCreatePinnedToCore(TaskROTATE,    //function
                        "Task Rotate Core1", //name of the task
                        6000,         //stack size(bytes)
                        NULL,         //parameter to pass to function
                        1,            //giving lower priority
                        &task_Rotate,       //task handle
                        1);           //Core ID
```

D. Method of measuring the generated current

❖ Concept

As mentioned earlier, a DC voltage sensor and a 1kΩ resistor was used to measure the current. Using ohms law (V = IR), the generated current at that moment can be calculated. The following figure shows the connections of DC voltage sensor and the 1kΩ resistor to the solar panel.

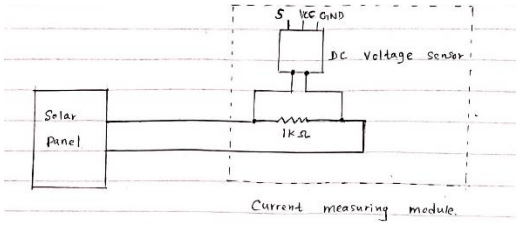

Figure 18 Voltage sensor connections

Thereafter, the voltage across 1kΩ resistor can be calculated as follows,

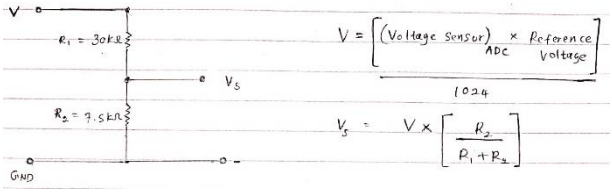

Figure 19 Voltage Calculation

In the next step, using ohms law, the current generated by the solar panel can be calculated.

Generated current = V_s / 1kΩ

❖ Current measuring cording

First, the following global variables were created.

```
int voltage_sensor = 35;
float adc_voltage = 0.0;
float Voltage = 0.0;
float Current = 0.0;
float R1 = 30000.0;
float R2 = 7500.0;
float Reference_Voltage = 3.3;
```

Then, the task handler was created and set to NULL as follows,

```
static TaskHandle_t task_CURRENT_MEASURING = NULL;
```

Thereafter, a separate task was created pinned to core 1 of the ESP32 as follows,

```
xTaskCreatePinnedToCore(TaskCURRENT_MEASURING,
                        "Task CURRENT_MEASURING Core1",
                        2500,
                        NULL,
                        1,
                        &task_CURRENT_MEASURING,
                        1);
```

Here, TaskCURRENT_MEASURING is a pointer to the function which measure the current. "TaskCURRENT_MEASURING Core1" is the name of the task, stack depth is 2500 bytes, no parameters and therefore NULL, priority is set to 1, task_CURRENT_MEASURING is the task handler variable and final argument which is Core ID is set to 1 (core 1 of ESP32 is used).

After creating the task, the following function was created to measure the generated current.

```
void TaskCURRENT_MEASURING(void *parameter) {

  while (1) {
    analogReadResolution(10);
    float voltage_sensor_adc_value = analogRead(voltage_sensor);
    adc_voltage  = (voltage_sensor_adc_value * Reference_Voltage) / 1024.0;
    Voltage = adc_voltage / (R2/(R1+R2));
    Current = (Voltage / 1000)*1000;
  }
}
```

Using this function, the generated current is measured repeatedly.

E. I2C display programming

Here, LiquidCrystal_I2C.h library was used to make the configuration easier. First, SCL pin and the SDA pin of the I2C LCD were connected to the GPIO22 and GPIO21 pins of the ESP32 respectively.

Thereafter, a separate task handler was created as set to NULL. It is as follows,

```
static TaskHandle_t task_DISPLAY = NULL;
```

Then, using FreeRTOS xTaskCreatePinnedToCore() function, another separate task was created to drive the LCD. It is as follows,

```
xTaskCreatePinnedToCore(TaskDISPLAY,
                        "Task DISPLAY Core1",
                        2500,
                        NULL,
                        1,
                        &task_DISPLAY,
                        1);
```

Here, TaskDISPLAY is a pointer to the function which drives the LCD display. “Task DISPLAY Core1” is the name of the task, stack depth is 2500 bytes, no parameters and therefore NULL, priority is set to 1, task_DISPLAY is the task handler variable and final argument which is Core ID is set to 1 (core 1 of ESP32 is used).

After creating the task, the following function was created to drive the I2C LCD display.

```
void TaskDISPLAY(void *parameter){

  while(1){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Current = ");
    lcd.print(Current);
    lcd.print("mA");
    lcd.setCursor(0, 1);
    lcd.print("Angle = ");
    lcd.print(real_angle);

    vTaskDelay(pdMS_TO_TICKS(500));
    lcd.clear();
  }
}
```

F. IOT part initialization

❖ HTML initialization

EPS32 webserver accessed by the Ip address of the esp-32 by a client in the same network. In creating the Solar Tracker server via HTML, in the body part of the HTML file two paragraphs are made as shown in the code below to display the current and the rotation of the system.

```
<body>
  <h2>ESP32 Solar Tracker Server</h2>
  <p>
    <span class="labels">Rotation</span>
    <span id="rotation">%ROTATION%</span>
    <sup class="units">&deg;</sup>
  </p>
  <p>
    <span class="labels">Current</span>
    <span id="humidity">%CURRENT%</span>
    <sup class="units">A</sup>
  </p>
</body>
```

Figure 20 HTML Initialization

In the script of the HTML file the code is written to initialize a set Interval () function which is initialized to run every 10 seconds, to obtain the latest Rotation and the Current of the system.

Once the new updated variables are received the relevant variables are replaced in the HTML webserver to the new values.

```
<script>
setInterval(function ( ) {                /*runs every 10 sencods*/
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("rotation").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/rotaion", true);         //gets the latest ROTATION
  xhttp.send();
}, 10000 ) ;
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("current").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/current", true);
  xhttp.send();
}, 10000 ) ;
</script>
```

Figure 21 HTML Script

❖ Processor Function

A function known as processor (), is used to replace the placeholders in our HTML text with the actual rotation and current values in the GUI webserver interface. Which is later used in the webserver to update the values in Realtime

```
// Replaces placeholder with new values
String processor(const String & var) {
  if (var == "ROTATION") {
    return String(real_angle);
  }
  else if (var == "CURRENT") {
    return String(Current);
  }
  return String();
}
```

Figure 22 Processor Function

❖ Connecting to the WIFI

Then WiFi object is created to connect to the internet with using WIFI.h library

```
#include "WiFi.h"
```

The network credentials of the network to connect is defined

```
//Network Credentials
const char* ssid = "EN20417284";
const char* password = "12345678";
```

Using RTOS concepts the WiFi of the esp32 is connected.

```
xTaskCreatePinnedToCore(TaskConnectToNet,
                        "IoT",
                        6000,
                        NULL,
                        1,
                        NULL,
                        1);
```

The function is initialized such that once the Wi-Fi is connected it will print the Local IP address in the serial monitor.

❖ Asynchronous webserver

When the URL is requested by a client, we send the webpage stored in index_html text in the program memory of the ESP32 by using the following code.

```
const char index_html[] PROGMEM = R"===(
```

Then processor () function is also passed with the HTML request to update the relevant values obtained by the variables (Rotation and Current).

Additional, to those two HTTP handlers are used to update the Rotation and the Current reading, it should be sent via plain text using char so, c_str() function is used to convert the String variables to char.

```
xTaskCreatePinnedToCore(TaskUpdateValues,
                        "Task Update values",
                        6000,
                        NULL,
                        1,
                        NULL,
                        0);

void TaskUpdateValues(void *parameters) {

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send_P(200, "text/html", index_html, processor);
  });
  server.on("/rotation", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send_P(200, "text/plain", String(real_angle).c_str());
  });
  server.on("/current", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send_P(200, "text/plain", String(Current).c_str());
  });
}
```

Figure 23 Update value function

The to start the server we utilize the AsyncWebServer object declared before using the HTTP port 80 and, in the Setup, (), the server. begin () function is used to initialize webserver.

```
AsyncWebServer server(80);
server.begin();
```

IV. Results and Discussion

❖ Results

In the figures below are shown the implementation of the project and the results obtained.

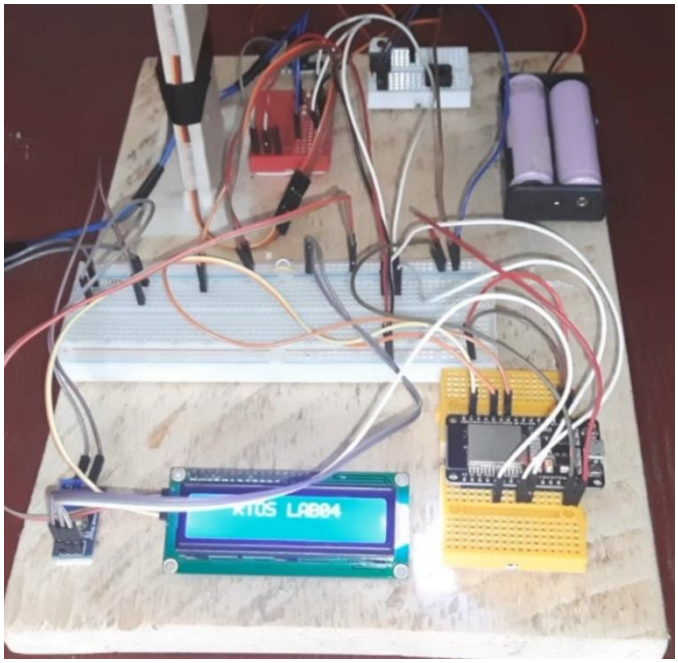

Figure 24 IOT based sun tracker physical implementation

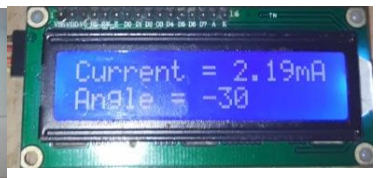

Figure 25 Outputs obtained according to the direction of the sun

ESP32 Solar Tracker Server

Rotation %ROTATION% °

Current %CURRENT% A

Figure 26 Initialized web server

Once the Realtime data is received via the sensors the Web server also shows the relevant parameters as shown below.

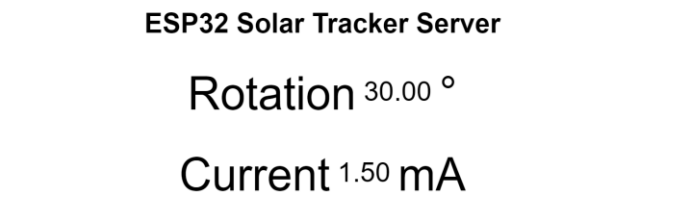

Figure 27 HTML webserver with updated values

While the ESP32 is connecting to the Wi-Fi it displays the following message in the Serial monitor

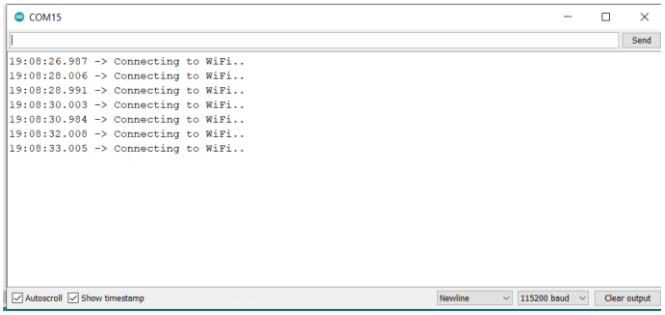

Figure 28 Serial monitor while ESP32 connecting to Wi Fi

Once the ESP32 is connected to the internet it prints the IP address of the local host which could be accessed by aby client in the system with the same network as shown below.

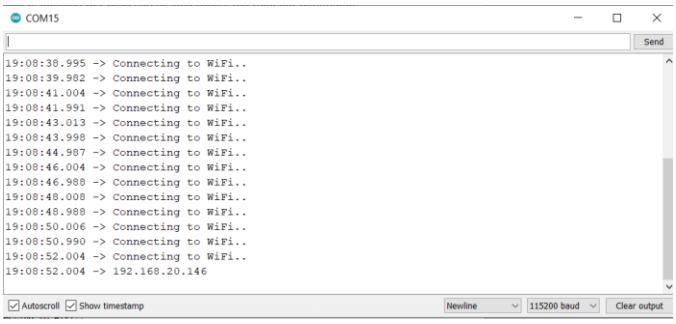

Figure 29 Serial monitor displaying the IP address when connected to Wi Fi

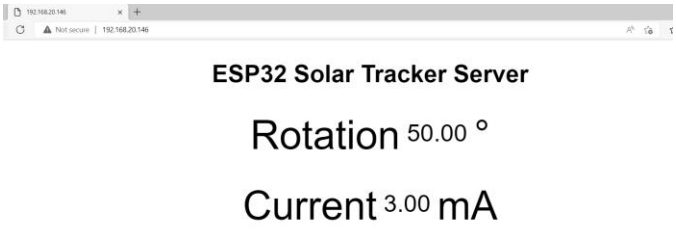

Figure 30 Webserver in accessed via the IP address of the ESP32

❖ Discussion

In this designing of Sun Tracking Solar Panel, there are some several unique features in beside of other Sun Tracking Solar Panels. Mainly the designing of hardware structure is unique for this design. The hardware structure was designed as it is handheld and user friendly. Therefore, the dimensions of the overall structure and assembled components are important. Two LDR sensors and current sensors were placed as those components can easily connected to the Microcontroller in the controller unit. In this design, 3.7V two lithium iron rechargeable batteries are used to power the system. LCD screen displays the amount of current generated from the solar panel, and the tilted angle of the solar panel. After observing the results, the expected results were obtained. Correct angle and the currents were displayed on the LCD screen. And when testing the motion of the solar tracker, the panel was moved according to the movement of the light source as expected. The accuracy of that task is much greater than expected because the reading of the two LDRs are in a separate task. In the IOT section of the project the HTML page was initialized, the page was only accessible when only other client devices are connected to the same network as the ESP32, when the data in the global variables updates the asynchronous webserver checks the data and updates it. other means of connecting the solar tracker to the internet were checked using an android app and using IOT cloud, in which only one user device could access the data, so it was much more desired to host a web server using HTML which could be accessed by many clients at the same time. As improvements to the webserver the User login could be introduced to improve the privacy of the data and the user's space and hosting the webpage as a website could be also utilized to implement the project as a IoT based server.

In this assignment, a IOT based sun tracking solar panel was implemented using ESP32 microcontroller. The purpose of this device is to obtain maximum power generation efficiency by tilting the solar panel according to the position of the sun so that sunlight falls perpendicular to the solar panel thereby increasing the efficiency of the power generation.

V. References

- [1]"ESP32 DHT11/DHT22 Web Server using Arduino IDE | Random Nerd Tutorials", Random Nerd Tutorials, 2022. [Online]. Available: https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-web-server-arduino-ide/. [Accessed: 09- Jul- 2022].
- [2]"GitHub - HirushaRadeeshan/RTOS-Lab4-IOT-Based-Sun-Tracking-Solar-Panel", GitHub, 2022. [Online]. Available: https://github.com/HirushaRadeeshan/RTOS-Lab4-IOT-Based-Sun-Tracking-Solar-Panel/tree/main#rtos-lab4-iot-based-sun-tracking-solar-panel. [Accessed: 10- Jul- 2022].
- [3] J. Yoshitake, "solar tracker | technology", Encyclopedia Britannica, 2022. [Online]. Available: https://www.britannica.com/technology/solar-tracker. [Accessed: 04- Jul- 2022].
- [4] "Sun Tracking Solar Panel", Electronic Hub, 2018. [Online]. Available: https://www.electronicshub.org/sun-tracking-solar-panel/. [Accessed: 03- Jul- 2022].
- [5] "Interfacing 0-25V DC Voltage Sensor with Arduino", How To Electronics, 2022. [Online]. Available: https://how2electronics.com/interfacing-0-25v-dc-voltage-sensor-with-arduino/. [Accessed: 02- Jul- 2022].