UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE INGENIERÍA ELECTRÓNICA

DESARROLLO DE UN SEGUIDOR SOLAR PARA CALIBRACIÓN DE SENSORES DE RADIACIÓN EN LA EMPRESA PÚBLICA INAMHI ORIENTADO A IOT

Trabajo de titulación previo a la obtención del
Título de Ingeniero Electrónico

AUTORES: Ronnie Smith Pulupa Mallitasig
Eduardo Alejandro Seraquive Lopez
TUTORA: Carmen Johanna Celi Sánchez

Quito-Ecuador
2023

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Ronnie Smith Pulupa Mallitasig con documento de identificación N° 1724002587 y Eduardo Alejandro Seraquive Lopez con documento de identificación N° 1750775403; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 07 de marzo del año 2023

Atentamente,

Ronnie Smith Pulupa Mallitasig
1724002587

Eduardo Alejandro Seraquive Lopez
1750775403

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Ronnie Smith Pulupa Mallitasig con documento de identificación N° 1724002587 y Eduardo Alejandro Seraquive Lopez con documento de identificación N° 1750775403, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación : "Desarrollo de un seguidor solar para calibración de sensores de radiación en la empresa pública INAMHI orientado a IOT", el cual ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 07 de marzo del año 2023

Atentamente,

Ronnie Smith Pulupa Mallitasig
1724002587

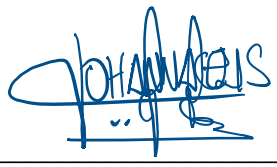
Eduardo Alejandro Seraquive Lopez
1750775403

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Carmen Johanna Celi Sánchez con documento de identificación N° 1717437808 docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UN SEGUIDOR SOLAR PARA CALIBRACIÓN DE SENSORES DE RADIACIÓN EN LA EMPRESA PÚBLICA INAMHI ORIENTADO A IOT, realizado por Ronnie Smith Pulupa Mallitasig con documento de identificación N° 1724002587 y por Eduardo Alejandro Seraquive Lopez con documento de identificación N° 1750775403, obteniendo como resultado final el trabajo de titulación bajo la opción de proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 07 de marzo del año 2023

Atentamente,



Ing. Carmen Johanna Celi Sánchez MSc
1717437808

DEDICATORIA

Esta tesis está dedicada a:

A Dios quien con su guía y protección me ha permitido llegar a cumplir un sueño más.

A mi amado padre Leonardo que ha sido el pilar más importante en mi vida quien con su enseñanza, paciencia, motivación y amor me ha acompañado a lo largo de este camino para lograr culminar una meta más en mi vida.

A mis hermanos por su cariño y por estar conmigo en cada momento.

Eduardo Alejandro Seraquive Lopez

Este trabajo va dedicado principalmente a mis padres Jenny y Daniel, quienes en el transcurso de este largo camino me han apoyado incondicionalmente y me han enseñado que con dedicación y esfuerzo se pueden lograr grandes cosas, gracias por su amor y constancia para formarme como un hombre de bien.

A mi hermana, por siempre alentarme a no rendirme y celebrar como propios mis logros y objetivos cumplidos.

A mi tío Guillermo, quien es como mi segundo padre, que siempre tuvo unas palabras de aliento y su apoyo fue fundamental para la consecución de este proyecto.

Ronnie Smith Pulupa Mallitasig

AGRADECIMIENTO

Quiero expresar mi gratitud a mi padre por su esfuerzo incondicional para brindarme una educación de calidad y a mis hermanos por estar siempre presentes.
Mi más profundo agradecimiento a mis amigos que me han acompañado en los momentos difíciles y por extender su mano cuando los he necesitado.
Finalmente quiero agradecer a la Ing. Carmen Johanna Celi, principal colaboradora durante todo este proceso, quien con su conocimiento y dirección permitió el desarrollo de este proyecto.

Eduardo Alejandro Seraquive Lopez

A cada uno de los docentes de la Universidad Politécnica Salesiana que colaboraron con nuestra formación ética y profesional.
Agradezco a los buenos amigos, que fueron un gran apoyo en esta vida universitaria.
Además agradezco a la Ing. Johanna Celi quien nos motivó para poder culminar este proyecto.

Ronnie Smith Pulupa Mallitasig

ÍNDICE DE CONTENIDO

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN ii

CESIÓN DE DERECHOS DE AUTOR iii

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN iv

DEDICATORIA v

AGRADECIMIENTO vi

ÍNDICE DE CONTENIDO vii

ÍNDICE DE FIGURAS x

ÍNDICE DE TABLAS xii

RESUMEN xiii

ABSTRACT xiv

Introducción xv

CAPÍTULO 1 1

ANTECEDENTES 1

1.1 Planteamiento del Problema 1

1.2 Justificación del proyecto 1

1.3 Objetivos 2

1.3.1 Objetivo General 2

1.3.2 Objetivos Específicos 2

CAPÍTULO 2 3

MARCO TEÓRICO 3

1.4 Introducción 3

1.5 Radiación Solar 3

1.6 Piranómetro 3

1.7 Datalogger 4

1.8 Campbell CR1000 4

1.9 Ecuaciones Solares 5

1.9.1 Meridiano de hora estándar local (LSTM) 5
1.9.2 Ecuación del Tiempo (EoT) 5
1.9.3 Factor De Corrección De Tiempo (TC) 5
1.9.4 Tiempo Solar Local (LST) 5
1.9.5 Ángulo horario (HRA) 6
1.9.6 Ángulo de declinación 6
1.10 Ángulos de seguimiento solar 6
1.10.1 Ángulo de elevación 6
1.10.2 Ángulo de azimut 7
1.11 Lógica difusa 7
1.12 Componentes: 7
1.12.1 Raspberry Pi 4 7
1.12.2 Arduino Mega 7
1.12.3 GPS u-blox M8 8
1.12.4 RTC Ds3231 9
1.12.5 Pantalla TFT ILI9341 9
1.12.6 Sensor acelerómetro Mma7455l 3 10
1.12.7 Brújula Electrónica HMC5883l 10
1.12.8 Servomotor INJORA 70 kg, 180° 11
1.12.9 Servomotor digital FEETECH 35 kg, 360° 11
1.13 Protocolo I2C 12
1.14 Protocolo SPI 12
1.15 Protocolo serial 12
1.16 Node Red 13
CAPÍTULO 3 14
DISEÑO E IMPLEMENTACION 14
1.17 Diseño del seguidor solar 14

1.17.1 Elección del software CAD 14
1.17.2 Especificaciones del diseño 14
1.18 Esquema general seguidor solar 18
1.19 Algoritmo seguidor solar 19
1.19.1 IDE Arduino 20
1.19.2 Etapa inicial 21
1.19.3 Etapa de ecuaciones solares 22
1.19.4 Etapa de control 24
1.19.4.1 Control FUZZY 26
1.19.5 Visualización de datos 29
1.20 Datos IoT 30
1.20.1 Instalación IDE Arduino en Raspberry 30
1.20.2 Instalación de NODE-RED 30
1.20.3 Programación Node-Red 31
1.20.4 Comunicación con el datalogger y recopilación de datos en la nube 32
1.21 Montaje 40
CAPÍTULO 4 42
PRUEBAS Y RESULTADOS 42
1.22 Procesamiento matemático 42
1.23 Control fuzzy 46
1.24 Comparación de datos entre la base fija y el seguidor solar 47
1.25 Visualización de los datos a través de IoT 49
Conclusiones 50
Recomendaciones 52
Referencia 53
Anexos 57

ÍNDICE DE FIGURAS

Figura 2.2.1 sensor de radiación piranómetro 4
figura 2.2.2 datalogger campbell cr1000 5
figura 2.3 arduino mega 2650 8
figura 2.4 módulo gps neo 8m 8
figura 2.5 módulo rtc ds 3231 9
figura 2.6 pantalla tft ili 9341 9
figura 2.7 módulo acelerómetro mma 74551 10
figura 2.8 brújula electrónica 10
figura 2.9 servomotor 70 kg, 180° 11
figura 2.10 servomotor 35 kg, 360° 11
figura 3.1 software solidworks 14
figura 3.2 pieza 1 base seguidor solar 15
figura 3.3 pieza 2 en forma de u (móvil) 15
figura 3.4 pieza 3 plancha para los sensores (móvil) 16
figura 3.5 estructura seguidor solar 16
figura 3.6 chumacera para el movimiento de la pieza 2 17
figura 3.7 chumacera lado izquierdo y derecho para el movimiento de la pieza 3 17
figura 3.8 estructura seguidor solar 18
figura 3.9 esquema general seguidor solar 19
figura 3.10 máquina de estados del algoritmo 20
figura 3.11 ide de arduino 20
figura 3.12 diagrama de flujo etapa inicial 22
figura 3.13 diagrama de flujo ecuaciones y ángulos solares 23
figura 3.14 diagrama de flujo etapa de control 25
figura 3.15 estructura del sistema difuso 26
figura 3.16 fuzzy logic designer de matlab 26
figura 3.17 funciones de membresía del error de azimut y elevación 27
figura 3.18 funciones de membresía de los servomotores 28
figura 3.19 reglas lingüísticas 28
figura 3.20 interfaz gráfica convertidor de código. 29
figura 3.21 visualización de información en la pantalla ili 9341 29
figura 3.22 instalación arduino en raspberry 30
figura 3.23 instalación y actualización node red en raspberry 31

figura 3.24 cadena de información en ide de arduino 31
figura 3.25 esquema gráfico de la programación en node-red 32
figura 3.26 software loggernet 33
figura 3.27 ventana para iniciar la configuración del datalogger 33
figura 3.28 ventana para seleccionar el modelo de datalogger 34
figura 3.29 ventana para seleccionar el tipo de conexión 34
figura 3.30 ventana para seleccionar el puerto de comunicación 35
figura 3.31 selección de los parámetros del datalogger 35
figura 3.32 menú de herramientas software loggernet 36
figura 3.33 selección del tipo de sensor que se va a conectar al datalogger 36
figura 3.34 configuración de parámetros del sensor 37
figura 3.35 configuración de fecha u hora del datalogger 37
figura 3.36 selección de las tablas de datos para descargar 38
figura 3.37 configuración de la hora para la recolección automática de los datos 38
figura 3.38 archivo de datos en la nube 39
figura 3.39 opción para exportar los datos a una hoja de cálculo 39
figura 3.40 visualización de los datos en una hoja de cálculo desde la nube 40
figura 3.41 diseño de pcb en proteus 40
figura 3.42 montaje de elementos 41
figura 3.43 a. Diseño gabinete b. Gabinete con panel de control 41
figura 4.1 calculadora solar 42
figura 4.2 código para ecuaciones solares en matlab 43
figura 4.3 código para ecuaciones solares en arduino 43
figura 4.4 comparación de datos 47
figura 4.5 comparación de datas entre el seguidor solar y la base fija 48
figura 4.6 visualización de datos en el celular 49

ÍNDICE DE TABLAS

Tabla 4.1 comparación resultados ecuaciones solares 44
tabla 4.2 comparativa código arduino y programa en matlab 46
tabla 4.3 fragmento de los datos recolectados por la base fija y el seguidor solar..... 48

RESUMEN

El presente trabajo propuso el desarrollo de un seguidor solar de dos grados de libertad para una captación más exacta de los datos de radiación a diferencia del método actual que se la realiza en una base fija, el proyecto se basa principalmente en seguir la posición exacta del sol mediante Global Positioning System (GPS) y enviar los datos adquiridos de los sensores de radiación a la nube.

Para la implementación se usó el microcontrolador Arduino Mega donde se encuentra toda la programación, que mediante inteligencia artificial comprendiendo los conceptos básicos de lógica difusa se encarga de controlar los procesos para la movilidad de la estructura respecto al sol.

Mediante la aplicación de plataformas de almacenamiento como Google Drive se logró enviar los datos a la nube para el análisis de tendencias, desde cualquier punto con conexión a internet.

Por medio de la aplicación de los conceptos de Internet of things (IoT), se desarrolló una interfaz mediante el uso Node-Red y la tarjeta de desarrollo raspberry pi 4 para el monitoreo remoto del estado de la estructura, donde se puede visualizar la posición precisa en la que se encentra el seguidor solar.

Palabras claves: seguidor solar , ecuaciones solares, azimut, elevación, sensores de posición, servomotores , iot.

ABSTRACT

This work proposed the development of a solar tracker with two degrees of freedom for a more accurate collection of radiation data unlike the current method that is performed on a fixed base, the project is based primarily on tracking the exact position of the sun by Global Positioning System (GPS) and send the data acquired from the radiation sensors to the cloud.

For the implementation we used the Arduino Mega microcontroller where all the programming is located, which through artificial intelligence understanding the basic concepts of fuzzy logic is responsible for controlling the processes for the mobility of the structure with respect to the sun.

Through the application of storage platforms such as Google Drive it was possible to send the data to the cloud for trend analysis, from any point with internet connection.

Through the application of Internet of things (IoT) concepts, an interface was developed using Node-Red and the Raspberry Pi 4 development board for remote monitoring of the structure's status, where the precise position of the solar tracker can be visualized.

Keywords: solar tracker, solar equations, azimuth, elevation, position sensors, servo motors, iot.

INTRODUCCIÓN

Una estación meteorológica es el sito en el cual se realizan mediciones y observaciones precisas de los distintos rangos de valores meteorológicos utilizando los dispositivos adecuados.

Este proyecto está basado en mejorar el método de calibración de uno de estos instrumentos que son los sensores de radiación, de lo cual se encarga el Instituto Nacional de Meteorología e Hidrología (INAMHI) y que lo ha venido realizando basada en colocar los sensores en una base fija y comparar los valores que estos entregan con un patrón para posteriormente calibrarlos.

En vista de que la posición del sol es variable y la base donde se colocan los sensores es fija estos no pueden captar toda la radiación que emite el sol, lo cual impide calibrar de manera exacta los sensores.

En base a esta necesidad se implementa un seguidor solar de dos grados de libertad integrado por sensores, tarjetas electrónicas, software y otros componentes electrónicos que permita seguir la posición exacta del sol por medio de GPS

CAPÍTULO 1
ANTECEDENTES

1.1 Planteamiento del Problema

En la actualidad la calibración de los sensores llevada a cabo en el laboratorio de metrología INHAMI se lo realiza de forma estática la cual según (Carpio, 2021) esta manera de recolectar los datos de radiación es ineficiente porque se tiene menos cantidad de radiación captada por los sensores y esto se refleja con una reducción aproximada del 22%

Debido a que las longitudes de onda que ingresan a la superficie terrestre son variables por la existencia de nubes u otros factores que se involucran con la radiación normal directa (DNI), radiación horizontal difusa (DHI) y la radiación horizontal global (GHI) el promedio de dispersión de datos podría variar y esto conlleva a que la regresión lineal de estos datos sea errónea.

Además, podría aumentar el sesgo de los elementos a calibrar lo que conllevaría que la incertidumbre de medición de los sensores de radiación supere a la tolerancia permitida del 5% con respecto al patrón y no se pueda emitir las respectivas certificaciones para su uso.

Para la captación de los datos generados por los sensores es necesario que un técnico especializado este presente en el lugar y tenga con él los implementos necesarios para la comunicación al datalogger que son una computadora y los cables de comunicación, esto provoca una incomodidad ya que los equipos en ciertas ocasiones se descargan y en el sitio no hay la facilidad para cargarlos esto ha ocasionado un retraso en el trabajo.

1.2 Justificación del proyecto

La calibración de los sensores de radiación consiste en colocar un patrón sobre una estructura de preferencia móvil y los datos que entreguen los sensores a calibrar deben ser lo más cercanos a los datos del patrón, por el contrario, si son valores alejados se encuentra una ecuación mediante una regresión lineal la cual permitirá que los valores de los sensores a calibrar cumplan con la tolerancia permitida.

(Carpio, 2021) menciona que si existe un aumento de la cantidad de radiación transformada a voltaje al implementar un mecanismo móvil de dos ejes mejorando la captación de las longitudes de onda en un 22.31%.

Según (Pelayo López et al., 2018) concluye que se produce el doble de potencia eléctrica en un panel con seguimiento solar ya que aprovecha el 19% de la energía disponible, mientras en un panel estático aprovecha solamente el 11.5%, obteniendo para este caso un aumento de la eficiencia en un 7.5%.

El proyecto consiste en desarrollar un sistema de seguimiento solar de dos ejes controlado por GPS, algoritmos de trayectoria, sensores de posición y además contará con una arquitectura de comunicación para la visualización, análisis y corrección de datos desde cualquier lugar donde se disponga de un ordenador y conexión a internet para ser aprovechado por el INAMHI.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un seguidor solar a través de un sistema de posicionamiento global (GPS) y sensores de posición para la calibración de sensores de radiación en la empresa pública INAMHI orientado a IoT.

1.3.2 Objetivos Específicos

- Establecer las ecuaciones que describen la trayectoria del sol para implementarlas en el hardware y software de control mediante el análisis matemático del estado del arte.

- Diseñar un seguidor solar con dos grados de libertad para el movimiento en elevación y azimut a través de un software CAD.

- Implementar en la estructura de 2 ejes un algoritmo para el seguimiento de la trayectoria del sol mediante hardware y software especializado.

- Establecer una comunicación entre un datalogger y cloud para el análisis de tendencias mediante una arquitectura de comunicación.

- Comparar los resultados de niveles de radiación entre el sistema fijo y móvil para su validación, mediante pruebas experimentales.

CAPÍTULO 2
MARCO TEÓRICO

1.4 Introducción

En este capítulo se describen los conceptos básicos de las temáticas que comprende este proyecto para su desarrollo, como descripción y unidades de medida de la radiación solar, elementos que serán usados en el desarrollo del software y hardware como sensores, actuadores, dataloggers y protocolos de comunicación.

También se detalla aspectos de inteligencia artificial como la lógica difusa y la aplicación de IoT para el monitoreo remoto de datos.

1.5 Radiación Solar

Es la energía que emana el sol a través de radiación electromagnética que incide sobre la atmósfera. Se mide de forma horizontal, por medio de el sensor de radiación o piranómetro, que se coloca en un lugar sin sombra. Su unidad de medida es vatios por metro cuadrado (W/m^2).(Roldán, 2012)

La radiación solar que se mide en una estación meteorológica es dada en unidades de potencia y está en watts por metro cuadrado (W/m^2). Suponiendo que los datos se recojan cada 10 minutos se trata de la potencia medida en 10 minutos y en el caso de que se requiera convertir la radiación global solar en unidades de energía se debe multiplicar cada valor de potencia en W/m^2 por 600 segundos (segundos en 10 minutos) y el resultado se obtendrá en julios por metro cuadrado (J/m^2).(Roldán, 2011)

1.6 Piranómetro

El piranómetro es un instrumento que permite medir la radiación solar que incide sobre la Tierra, es decir mide la densidad de flujo de radiación solar (W/m^2) en un campo de 180 grados, este tipo de sensores son usados en estaciones meteorológicas.(Calventus et al., 2006)

El principio físico en el que se fundamenta es un transductor sobre el cual cae la radiación a través de un domo de vidrio, para el correcto funcionamiento de este sensor se debe considerar ciertos aspectos al momento de la instalación. Debe ser colocado sobre una superficie plana, tener 180° de visibilidad y recibir la luz solar dependiendo

de las diferentes posiciones del sol. Cuando el sol se encuentra en el zenit, es cuando la medida es más exacta.(Calventus et al., 2006)

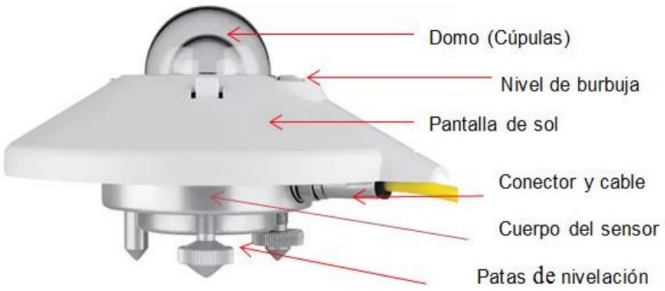


Figura 2.0.1 Sensor de radiación Piranómetro

Fuente: ((HydroMet, 2023))

1.7 Datalogger

El Datalogger es un dispositivo electrónico que permite monitorear y registrar datos en tiempo real a través de sensores pertenecientes al mismo o que se encuentran conectados exteriormente. La mayoría se componen por microcontroladores y se comunican con un computador a través de un software específico para programar la recolección de datos.(Camargo et al., 2019)

Una de las principales ventajas de los registradores de datos es que permiten realizar la recopilación automática de los datos las 24 horas del día, por esta razón una de sus principales aplicaciones es en estaciones meteorológicas.(Camargo et al., 2019)

1.8 Campbell CR1000

El CR1000 es el registrador de datos más utilizado. Puede ser usado en una vasta gama de funciones de medición y control. Es sumamente resistente en caso de ser instalado en condiciones extremas y muy confiable para entornos remotos, además consta de un módulo de medición y control y un panel de cableado. El bajo consumo de energía permite que el CR1000 funcione durante períodos prolongados con una batería recargada con un panel solar, lo que elimina la necesidad de alimentación de CA. (Campbell, 2020)

Figura 0.2.2 Datalogger Campbell CR1000

Fuente:(Campbell, 2020)

1.9 Ecuaciones Solares

1.9.1 Meridiano de hora estándar local (LSTM)

Meridian Standard Time es el meridiano de referencia para una zona horaria específica.(Honsberg & Bowden, 2019)

$$\mathrm{LSTM} = 15^{\circ} * \Delta \mathrm{T}_{\mathrm{GMT}} \qquad (1)$$

$$\mathrm{GMT}_{\mathrm{Ecuador}} = -5$$

1.9.2 Ecuación del Tiempo (EoT)

Esta ecuación corrige la excentricidad(desviación) de la órbita terrestre e inclinación de su eje, medidas en minutos..(Honsberg & Bowden, 2019)

$$EoT = 9.87\sin(2A) - 7.53\cos(A) - 1.5\sin(A) \qquad (2)$$

Donde:

$$\mathrm{A} = \frac{360}{365}(\mathrm{d})(-81{,}0) \qquad (3)$$

1.9.3 Factor De Corrección De Tiempo (TC)

El factor de corrección representa el cambio en la hora solar local en una zona en específico causado por cambios en la longitud en dicha zona y se mide en minutos.(Honsberg & Bowden, 2019)

$$\mathrm{TC} = 4(\mathrm{longitude} - \mathrm{LSTM}) + \mathrm{EoT} \qquad (4)$$

1.9.4 Tiempo Solar Local (LST)

La La hora solar local se puede encontrar ajustando LT usando las dos correcciones anteriores.(Honsberg & Bowden, 2019)

$$\mathrm{LST} = \mathrm{LT} + \frac{\mathrm{TC}}{60} \qquad (5)$$

1.9.5 Ángulo horario (HRA)

El ángulo horario se define como el número de grados en que el sol se mueve a lo largo del día, al mediodía solar este es de 0°. Por la mañana el ángulo horario es negativo, mientras que por la tarde se convierte en positivo.(Honsberg & Bowden, 2019)

$$\text{HRA} = 15^{\circ}(\text{LST} - 12) \qquad (6)$$

1.9.6 Ángulo de declinación

El ángulo de declinación, denotado por δ, varía estacionalmente debido a la inclinación de la Tierra sobre su eje de rotación y a la rotación de la Tierra alrededor del sol, este se mantiene en 23,45 ° tanto negativa o positivamente. Sólo en los equinoccios de primavera y otoño el ángulo de declinación es igual a 0 °.(Honsberg & Bowden, 2019)

$$\delta = -23.45^{\circ} * \text{Cos}(\frac{360}{365} * (\text{d} + 10)) \qquad (7)$$

La declinación es cero en los equinoccios (22 de marzo y 22 de septiembre), positivos durante el verano del hemisferio norte y negativo durante el invierno del hemisferio norte. La declinación alcanza un máximo de 23,45 ° en 22 de junio (solsticio de verano en el hemisferio norte) y un mínimo de -23,45 ° el 22 de diciembre (solsticio de invierno en el hemisferio norte).(Honsberg & Bowden, 2019)

1.10 Ángulos de seguimiento solar

1.10.1 Ángulo de elevación

El ángulo de elevación (usado indistintamente como ángulo de altitud) es la altura angular del sol en el cielo medido desde la horizontal.

Una fórmula para el ángulo de elevación al mediodía solar puede determinarse de acuerdo con la fórmula para lugares

en el hemisferio norte:

$$\alpha = 90 - \varphi + \delta \qquad (8)$$

Y en el hemisferio sur:

$$\alpha = 90 + \varphi - \delta \qquad (9)$$

Donde: φ es la latitud del lugar de interés.(Honsberg & Bowden, 2019)

Mediante la siguiente ecuación se puede encontrar este ángulo:

$$\propto = sin^{-1}[\sin(\varphi)\sin(\delta) + \cos(\varphi)\cos(\delta)\cos(HRA)] \qquad (10)$$

$$\text{Donde: HRA} = 15^{\circ}(\text{LST} - 12) \qquad (11)$$

1.10.2 Ángulo de azimut

El azimut es la dirección de la brújula de donde proviene la luz del sol. En los equinoccios, el sol sale directamente desde el este y se pone directamente al oeste, independientemente de la latitud, con lo que el ángulo azimut es de 90 ° al amanecer y al atardecer 270 °(Honsberg & Bowden, 2019).

$$\text{azimut} = \cos^{-1}[\frac{\sin\delta\cos\varphi - \cos\delta\sin\varphi\cos(\text{HRA})}{\cos\alpha}] \quad (12)$$

Donde α es la elevación.

1.11 **Lógica difusa**

La lógica difusa es otra opción a la lógica clásica, que se usa como herramienta tanto para el control de sistemas industriales complejos, como para otros sistemas expertos en los que no exista información exacta. Mediante un conjunto de reglas lingüísticas de control se tomarán las decisiones con la que ha de funcionar el sistema. Un ejemplo de una regla sería: SI el sol cambia de posición ENTONCES los servomotores se posicionan para seguir su dirección (Ross, 2016)

1.12 **Componentes:**

1.12.1 **Raspberry Pi 4**

La Raspberry es una computadora en una placa única, su tamaño es compacto con prestaciones de hardware suficientes para el procesamiento de datos requerido. Es cuarenta veces más rápido que una placa Arduino en comparación con la velocidad de su reloj. Este dispositivo tiene 128x1000 veces más memoria de acceso aleatorio (RAM) que Arduino. Puede ejecutar todo el sistema operativo Linux y Windows en arquitectura Advanced RISC Machine (ARM) 64 por sí solo.(Sidqi et al., 2018).

1.12.2 **Arduino Mega**

Para ese proyecto se decidió optar por el microcontrolador Arduino Mega 2560, este dispositivo proporciona una solución flexible en todos los aspectos, como *general purpose input/output* (GPIO), memoria, analog to digital converter (ADC), modo de comunicación, consumo de energía ideal para este sistema a implementar. Esta placa se basa en el microcontrolador ATmega2560 de 8 bits. Tiene un oscilador de cristal integrado de 19 MHz. Tiene 54 GPIO (entrada-salida de propósito general), 16 pines son pines de entrada analógica y el resto son pines digitales. Tiene 4

receptores/transmisores asíncronos universales (UART), un puerto USB, encabezado de programación en serie en circuito (ICSP) y un pulsador para reiniciar integrado. La instalación principal tiene un módulo WI-FI a bordo. Funciona con una fuente de alimentación de 5V.(Ashok et al., 2021)

Figura 0.3 Arduino Mega 2650

Fuente: (Wikimedia Commons)

1.12.3 GPS u-blox M8

El módulo GY-NEO-8M es un módulo GPS avanzado basado en uBlox m8N que admite el protocolo de comunicación UART con antena activa. Este módulo tiene una batería recargable, puede conectar directamente a una computadora mediante un convertidor de USB a TTL. NEO-8M puede recibir información y luego calcular la posición geográfica con muy alta precisión y alta velocidad. Además de admitir BeiDou, Galileo, GLONASS, GPS/QZSS, el módulo tiene una memoria interna para guardar configuraciones. NEO-8M es compatible con Arduino.

La disposición del sensor es la siguiente:

- GND: Tierra
- TX: envío de datos a través del protocolo UART
- RX: recibir datos a través del protocolo UART
- VCC: fuente de alimentación 5V

Este módulo se muestra en la imagen a continuación. (u-blox, 2020)

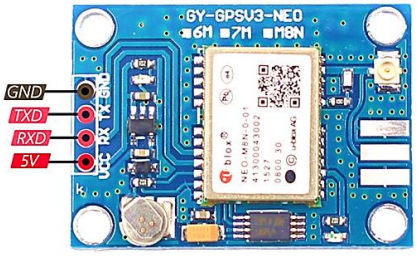


Figura 0.4 Módulo GPS Neo 8m

Fuente: (Electro Peak)

1.12.4 RTC Ds3231

RTC (Real Time Clock) es un módulo que se encarga de controlar el tiempo real. Al disponer un bajo consumo energético se alimenta mediante una batería externa para no perder la sincronización. Utilizar un RTC es más fiable que los relojes integrados en los microcontroladores de Arduino. Se comunican mediante el Protocolo I2C. La memoria EEPROM AT24C32 permite almacenar 32Kbits (4K Bytes) de datos de manera permanente (Naylamp Mechatronics, 2021).

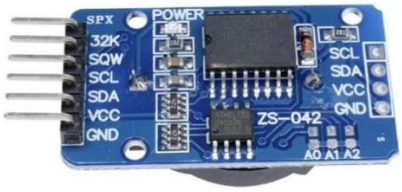


Figura 0.5 Módulo RTC Ds 3231

Fuente: (GM Electronic)

1.12.5 Pantalla TFT ILI9341

El módulo LCD posee una resolución de 240 x 320 en sus 2,4 in. Esta maneja la mayoría de las tareas de visualización, muchos contenidos, gráficos, datos de sensores, manual de usuario, etc. La biblioteca ILI9341 de Adafruit funciona perfectamente con esta pantalla. El voltaje de funcionamiento de la pantalla es de 3,3V, por lo que puede conectar los pines de la pantalla directamente a la MCU de 3,3V. Si necesita que funcione en la MCU de 5V, asegúrese de usar un traductor de nivel lógico.

Tipo: panel táctil Tamaño: 2,4" SPI Serial Área de visualización: 36,72 (ancho) x 48,96 (alto) mm Elemento del controlador: matriz activa a-Si TFT Disposición de píxeles: franja vertical RGB Controlador IC: ILI9341 Luz de fondo: LED blanco Dirección de visualización: 6 en punto, compatible con la interfaz 5110 (Smart Prototyping, 2021).

Figura 0.6 Pantalla TFT ILI 9341

Fuente:(Bento, 2018)

1.12.6 Sensor acelerómetro Mma7455l 3.

Modulo acelerómetro que dispone 3 ejes, se comunica mediante el protocolo I2C, posee un filtro pasa bajos con compensación de temperatura por lo que consume poca energía, se lo alimenta con 5v que puede ser suministrado por la fuente propia de la placa Arduino o por una fuente externa. La sensibilidad se puede seleccionar entre 3 rangos disponibles como son 2g, 4g u 8g. Además, el offset de 0g y la sensibilidad vienen preconfigurados de fábrica y no requieren componentes externos.(Movil Tronics, 2022)

Figura 0.7 Módulo acelerómetro Mma 74551

Fuente: (MGSYSTEM)

1.12.7 Brújula Electrónica HMC5883l

Módulo de 3 ejes, que mediante la dirección del campo magnético terrestre permite obtener la ubicación respecto al norte magnético. Su comunicación se la realiza mediante el protocolo I2C, y es necesario que ningún campo magnético externo interfiera en este módulo para que sus datos sean correctos. El HMC5883L posee un voltaje de operación bajo (3V), por lo tanto, la placa tiene incluida toda la circuitería necesaria para trabajar de manera segura con 5V, esto permite que se use tanto con microcontroladores de 3V o 5V.(Electroniclab, 2022).

Figura 0.8 Brújula electrónica

Fuente: (Electrónica Aplicada, EPA)

1.12.8 Servomotor INJORA 70 kg, 180°

Servo estándar digital de alto rendimiento, bajo nivel de ruido. El par máximo de este RC Servo puede alcanzar hasta 70 kg/cm. (8,4V). Engranajes metálicos de alta precisión con anodizado duro, fuerte adaptabilidad Rotación de grado bien controlada, viene con brazos de metal de 15T (INJORA, 2021).

Figura 0.9 Servomotor 70 kg, 180°

Fuente (INJORA)

1.12.9 Servomotor digital FEETECH 35 kg, 360°

FT6335M 35kg 360° es un servomotor de tamaño pequeño, ligero, gran par, alta velocidad y menos ruido. Servo RC, ampliamente utilizado en juguetes, coches de carreras de control remoto, aviones, barcos, micro robots, compatible con Arduino, Raspberry Pi El mecanismo de dirección es adecuado para vehículos de orugas a escala 1/8, 1/10, tanques, vehículos todoterreno, robots de control remoto y barcos. Este servo digital tiene una disipación de calor rápida, alta sensibilidad y larga vida útil (FEETECH, 2022).

Figura 0.10 Servomotor 35 kg, 360°

Fuente (FEETECH)

1.13 Protocolo I2C

I2C (Circuito Inter-Integrado) es un protocolo de comunicación serial síncrona. I2C tiene una interfaz de dos hilos y admite comunicación semidúplex, desarrollado por Philips Semiconductor en 1982. I2C tiene una arquitectura maestro-esclavo. I2C admite velocidades de datos de 100 kbps (modo estándar), 400 kbps (modo rápido) y un máximo de hasta 3,4 Mbps (modo de alta velocidad). Cada dispositivo I2C tiene los siguientes dos pines:

- SCL Serial Clock.
- SDA Serial Data.

El maestro es responsable de generar el reloj (típicamente en el rango de kHz) en la línea SCL. La trama I2C normalmente consta de 20 bits (bit de inicio + dirección de 7 bits + bit R/W + bit ACK + datos de 8 bits + bit ACK + bit de parada) (Kumari & Gayathri, 2017).

1.14 Protocolo SPI

Es un protocolo de comunicación serial síncrona. SPI tiene una interfaz de cuatro hilos y admite comunicación dúplex completa, desarrollado por Motorola a mediados de la década de 1980. SPI tiene una arquitectura maestro-esclavo. Tiene un solo maestro y múltiples dispositivos esclavos.

El maestro genera reloj (SCLK), generalmente en el rango de MHz. Idealmente, la línea de selección de esclavos (SS) es alta. El maestro baja la línea SS del dispositivo esclavo con el que desea comunicarse. Esta señal es el comienzo de la comunicación. Con cada pulso de reloj en el SCLK, los contenidos de datos almacenados en los registros de MOSI y MISO se desplazan y se transfieren entre sí. Después de que se envían todos los bits de datos (generalmente 8 bits), el maestro levanta la línea SS del esclavo para indicar el final de la comunicación. En este punto, el Maestro tiene el contenido inicial almacenado en el Esclavo y el Esclavo tiene el contenido inicial almacenado en el Maestro. Por lo tanto, es una comunicación full dúplex (Trivedi et al., 2018).

1.15 Protocolo serial

Es un protocolo de transferencia de datos en serie, es utilizado de dos maneras: síncrona cuando el receptor y el emisor usan la misma señal de reloj y asíncrona

cuando el emisor debe proporcionar una señal de sincronización previo al envío de cualquier dato.

La industria creó estándares de comunicación para este protocolo de comunicación: RS-232, RS-422 y RS-485. La principal diferencia entre estos estándares afecta a la parte física de la comunicación, es decir, al modo en el que se transmiten las señales eléctricas y el uso de los pines de los conectores, configurando de esta manera a la distancia máxima que se puede conseguir con cada uno de ellos (Dominique Breton & Daniel Charlet, 2003).

1.16 Node Red

Node.RED es una herramienta que sirve para realizar la comunicación entre dispositivos de hardware y servicios en línea especialmente lo relacionado con el internet de las cosas. Mediante un editor de flujo basado en funciones de JavaScript permite la programación para dicha comunicación. Desde la versión 0.14, los nodos *Message Queuing Telemetry Transport* (MQTT) pueden realizar conexiones TLS correctamente configuradas.(Chanthakit & Rattanapoka, 2018). Node-Red facilita las conexiones simplemente dibujando cables o enlaces y agregándoles parámetros. Los diagramas de flujo se explican por sí mismos. Cada nodo tiene entradas que son msg.payload escritas en javascript. Al hacer doble clic en un nodo, se abre un espacio para la inclusión del código (Rajalakshmi & Shahnasser, 2017).

CAPÍTULO 3

DISEÑO E IMPLEMENTACION

En este capítulo se explica el desarrollo tanto de hardware y de software que se emplea para la consecución del proyecto que su principal objetivo es la construcción de un seguidor solar de 2 ejes el cual va a permitir al laboratorio de metrología de la empresa pública INAMHI calibrar sus sensores de radiación solar.

Se precisa los procedimientos realizados como la elaboración de los planos con las medidas adecuadas para el correcto funcionamiento, la programación para los procesos del seguidor solar, la obtención de los datos a la nube y la aplicación de IoT mediante el monitoreo remoto desde un celular.

1.17 Diseño del seguidor solar

1.17.1 Elección del software CAD

El diseño estructural del seguidor solar se lo realizó en el software **SOLIDWORKS,** el cual permite el modelado mecánico de piezas y ensamblajes tanto en 2D como en 3D, para este proyecto se realizó el modelado de 3 piezas.

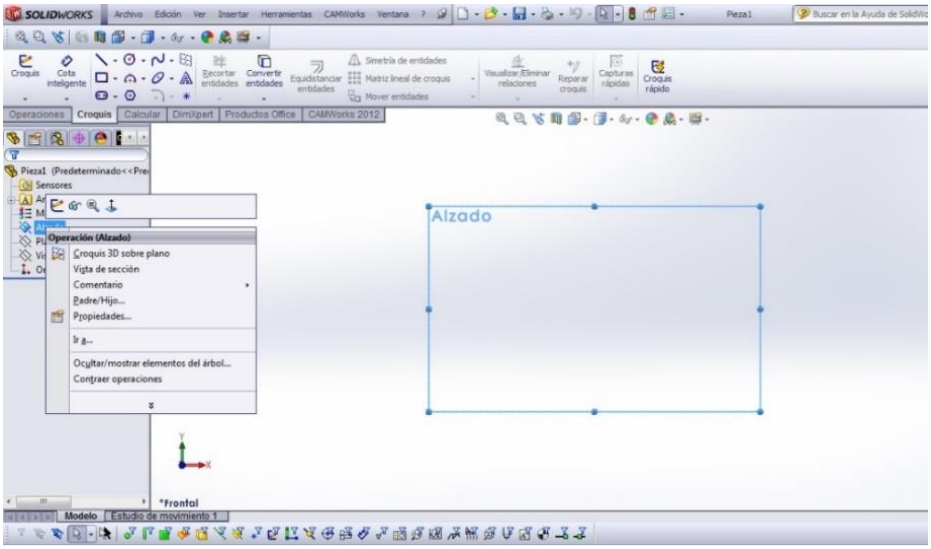


Figura 0.1 Software Solidworks

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.17.2 Especificaciones del diseño

El primer modelado mecánico es la base, que consta de 4 patas de 120 cm de altura desde el suelo, cada pata se realizó mediante un tubo cuadrado de 4 cm de ancho y 2 cm de espesor.

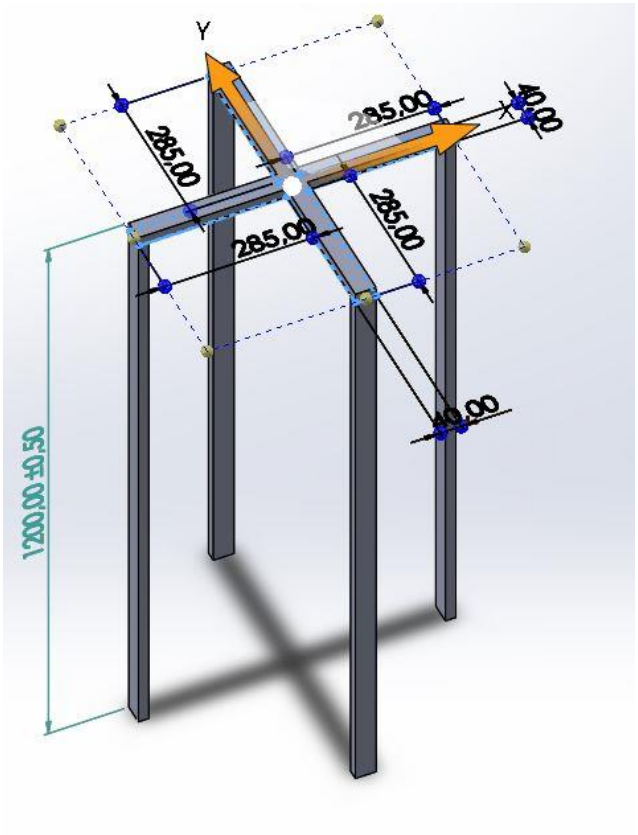


Figura 0.2 Pieza 1 base seguidor solar

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

En el segundo modelado mecánico se usó una plancha de tol de 137 cm de largo por 8.6 cm de ancho y 5 mm de espesor, posteriormente se realizó dos dobleces a 38 cm para así obtener una pieza en forma de (U).

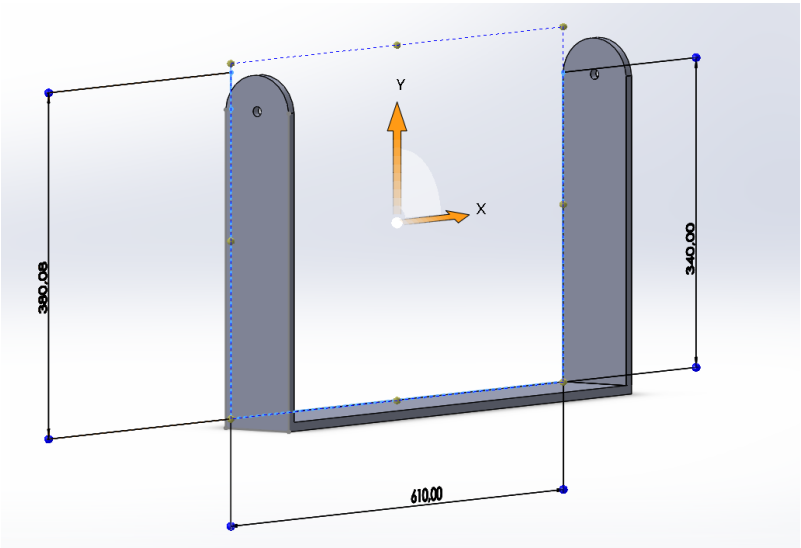


Figura 0.3 Pieza 2 en forma de U (móvil)

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Finalmente se realizó el diseño de la última pieza la cual es una plancha cuadrada de 56 cm de largo y 40 cm de ancho, con una base pequeña en el centro de 5 cm de largo y 5 cm de ancho para colocar el acelerómetro.

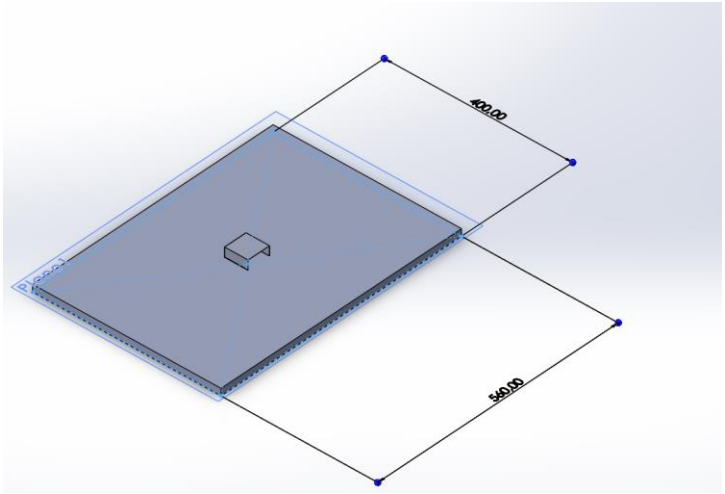


Figura 0.4 Pieza 3 plancha para los sensores (móvil)

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Mediante la opción ensamblaje de SOLIDWORKS se puede unir todos lo modelados mecánicos antes realizados para adquirir una perspectiva correcta del diseño del proyecto.

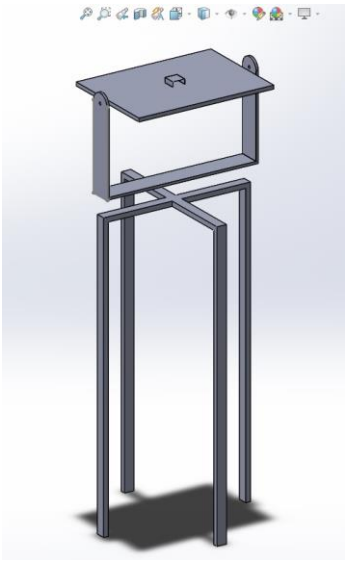


Figura 0.5 Estructura seguidor solar

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

La altura final de la estructura desde el suelo a la plancha donde se colocarán los sensores para calibración es de 170cm, esto debido a que por norma la persona que va a colocar los sensores a calibrar no debe estar encorvada.

Las chumaceras UCFL-208 se encargan del movimiento de la pieza dos y la pieza tres, la chumacera que se encarga de la movilidad de la segunda pieza se encuentra ubicada en la base de 4 patas donde mediante un eje y un servomotor de 35kg se encarga de rotar dicha pieza.

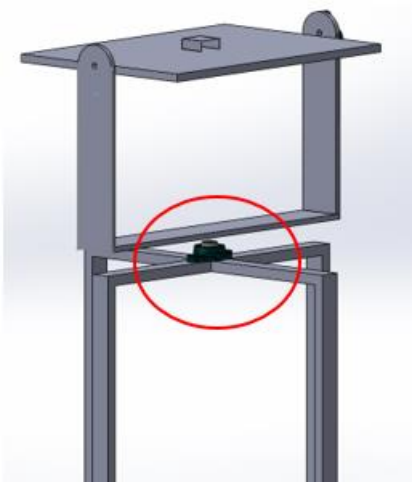


Figura 0.6 Chumacera para el movimiento de la pieza 2

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

La movilidad de la plancha donde se colocarán los sensores que se van a calibrar que es la tercera pieza que se diseñó, se colocó 2 chumaceras a los extremos de la pieza 2 que se conectan mediante un eje y también a un servomotor de 70 kg.

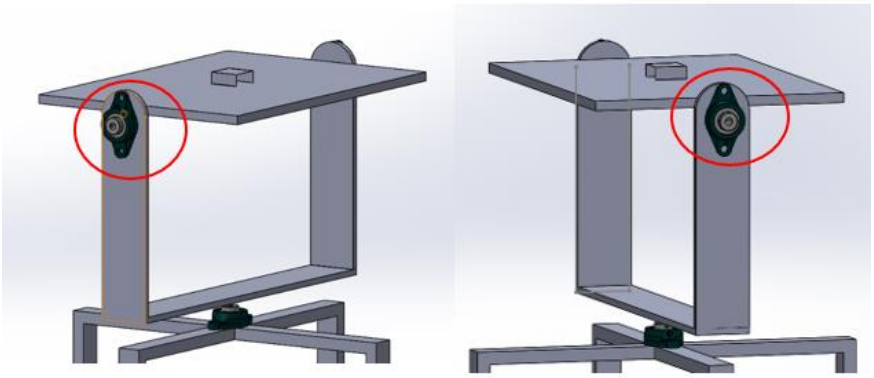


Figura 0.7 Chumacera lado izquierdo y derecho para el movimiento de la pieza 3

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Figura 0.8 Estructura Seguidor Solar

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.18 Esquema general seguidor solar

El diagrama de bloques que se presenta en la Figura 3.9, muestra la interacción de los elementos que componen el seguidor de línea, donde los bloques de color verde representan los sensores tales como RTC y GPS que proporcionan la fecha, hora y posicionamiento (latitud y longitud) respectivamente, además el magnetómetro y acelerómetro que proveerán la posición en que se encuentra el seguidor para compararlos con los ángulos de las ecuaciones, las flechas de color verde representan los datos enviados de los módulos hacia el microcontrolador, este realizará el procesamiento de las ecuaciones y el algoritmo de control fuzzy para los servomotores de azimut y elevación. Por último, la flecha de color rojo muestra la información que irá del microcontrolador hacia la pantalla LCD.

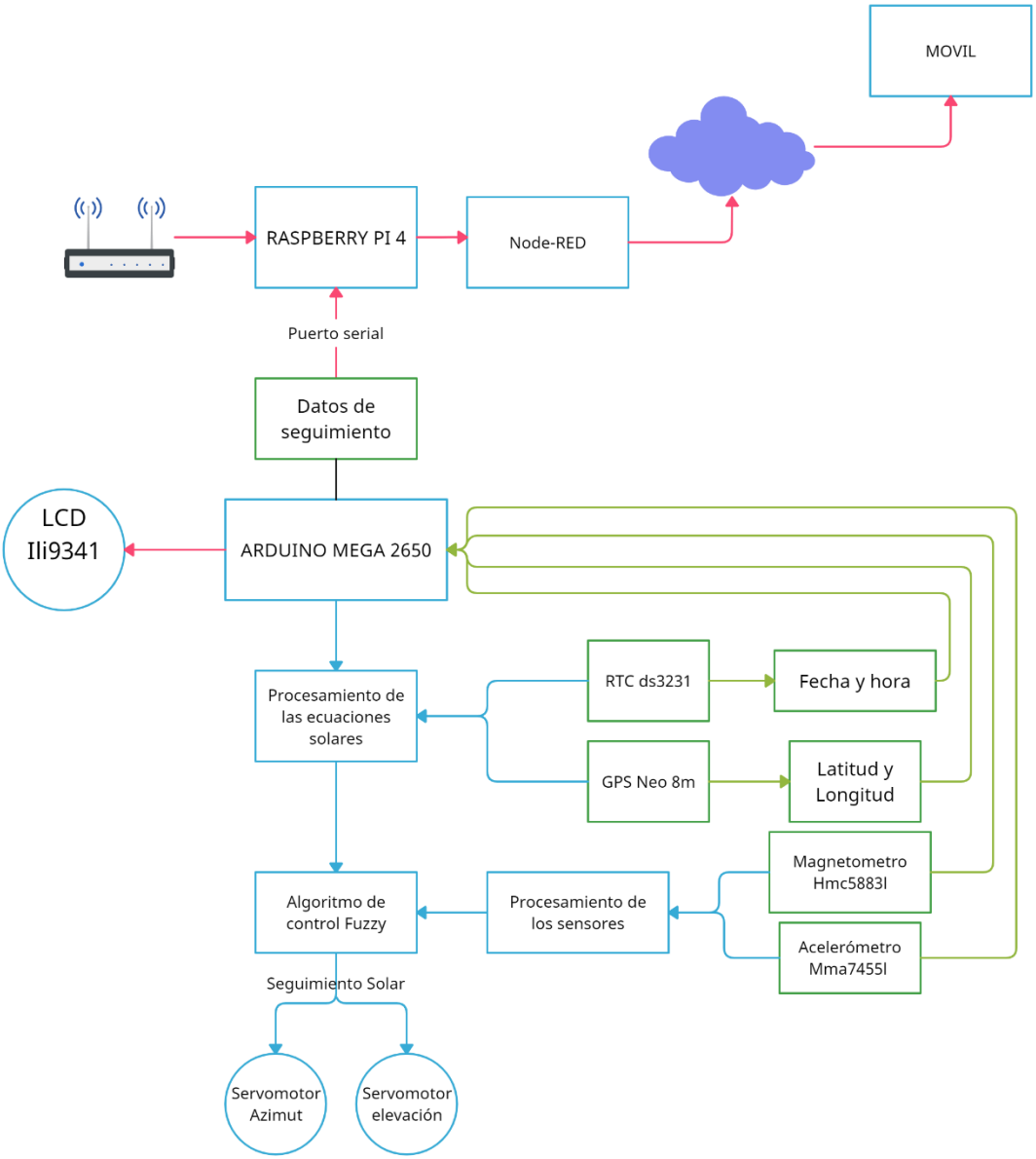


Figura 0.9 Esquema general seguidor solar

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.19 Algoritmo seguidor solar

La programación para este proyecto en primer lugar se basó en una máquina de estados, que su estructura se presenta en la Figura 3.10, donde cada estado será evaluado periódicamente mientras se cumplen las condiciones necesarias para saltar al siguiente estado y así hasta completar su ciclo, en el siguiente apartado se describe la programación, mientras que en el Anexo 2 el código completo ejecutado en el entorno de programación.

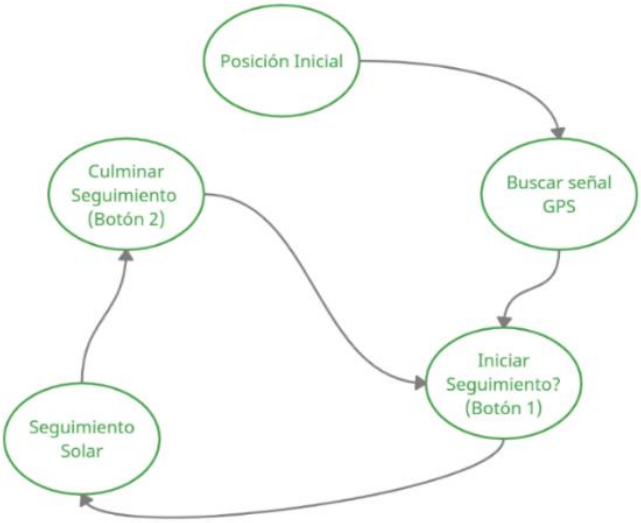


Figura 0.10 Máquina de estados del algoritmo

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.19.1 IDE Arduino

Este proyecto se realizó en el microcontrolador ARDUINO Mega 2650, su codificación se efectuó en la IDE de Arduino, el cual es un editor de código que dispone de las bibliotecas necesarias para los sensores, pantalla, GPS y servomotores. Así también este microcontrolador permite los protocolos con los que se comunican los diferentes elementos que la componen como son el protocolo I2C para el acelerómetro y magnetómetro, protocolo Serial para el GPS y envío de datos yprotocolo SPI para la pantalla que presentará la información del seguidor.

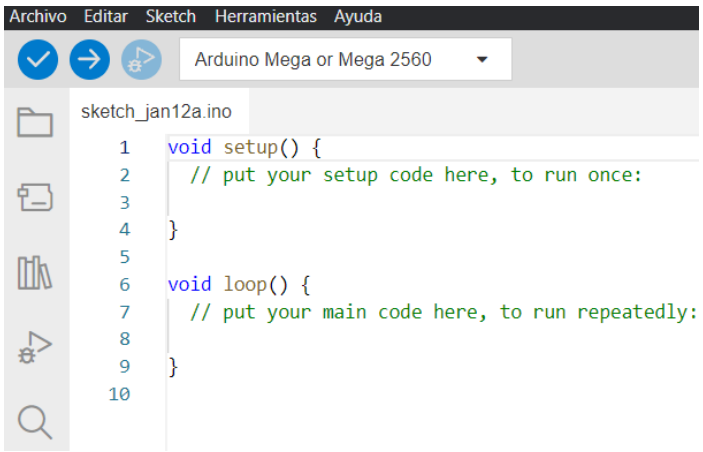


Figura 0.11 IDE de Arduino

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.19.2 Etapa inicial

La posición inicial se estableció para que los sensores de radiación sean colocados para su calibración, además para que la estructura sea posicionada hacia el norte geográfico.

El módulo GPS depende de que los satélites envíen la posición global adecuada, por esto se programó que el algoritmo continue siempre y cuando reciba tanto la latitud y longitud correcta, además del posicionamiento este módulo también proporciona el dato de fecha y hora, pero en el meridiano de Greenwich, por consiguiente, este actualizará al módulo RTC teniendo en cuenta el uso horario donde se lo usará, en este caso fue el GMT -5.

Una vez estos datos sean interpretados y almacenados en la memoria del microcontrolador, este puede continuar con su proceso.

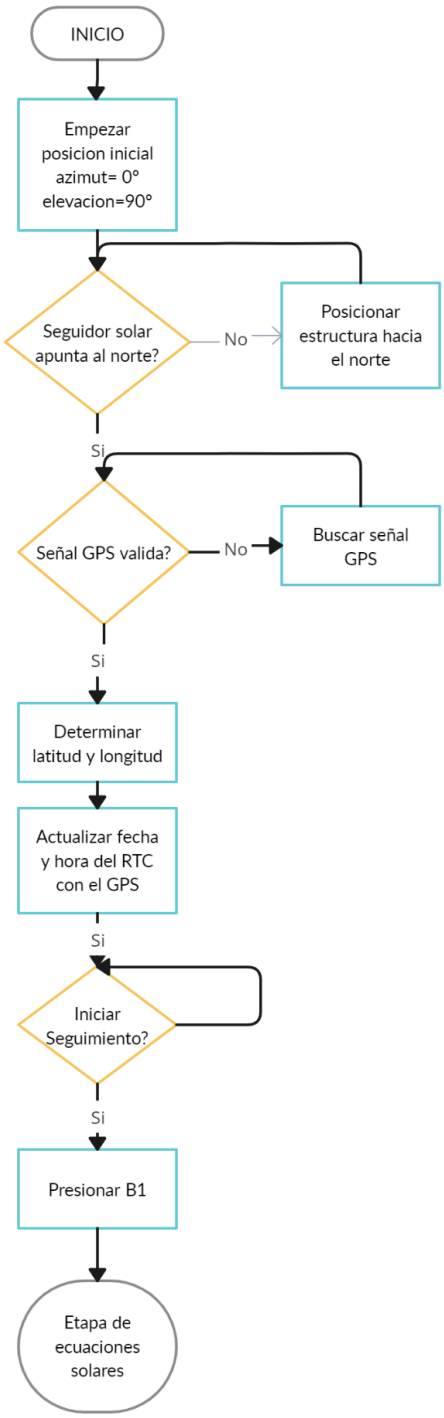


Figura 0.12 Diagrama de flujo Etapa inicial

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.19.3 Etapa de ecuaciones solares

En la Figura 3.13 se expone el proceso por el cual se determinan los ángulos solares que describen la trayectoria del sol. Este comienza cuando el botón de inicio es accionado, se realizó la lectura de posicionamiento y datos horarios, por medio de este

último se realizó el cálculo de la hora standard local que se obtiene de la Ecuación 1. Posteriormente para la Ecuación 2 y Ecuación 7 es necesario conocer el día del año que transcurre, para esto mediante la Ecuación 13 se determinó dicho dato, donde d corresponde al número entre 1 a 365, la variable bmes corresponde al mes del año que transcurre, dm corresponde al día del mes y finalmente el ajuste por si el año sea bisiesto.

$$d = (bmes * 30) + dm \pm ajuste \qquad (13)$$

Para hallar el tiempo solar local que se define como la hora en que el sol se encuentra en su punto más alto, primero se calculó el factor de corrección (Ecuación 4), que indica la variación del tiempo solar local y la longitud que se obtiene del módulo GPS. La declinación terrestre varía a lo largo del año debido a la inclinación de la tierra sobre su eje, por esto se consideró mediante la Ecuación 7

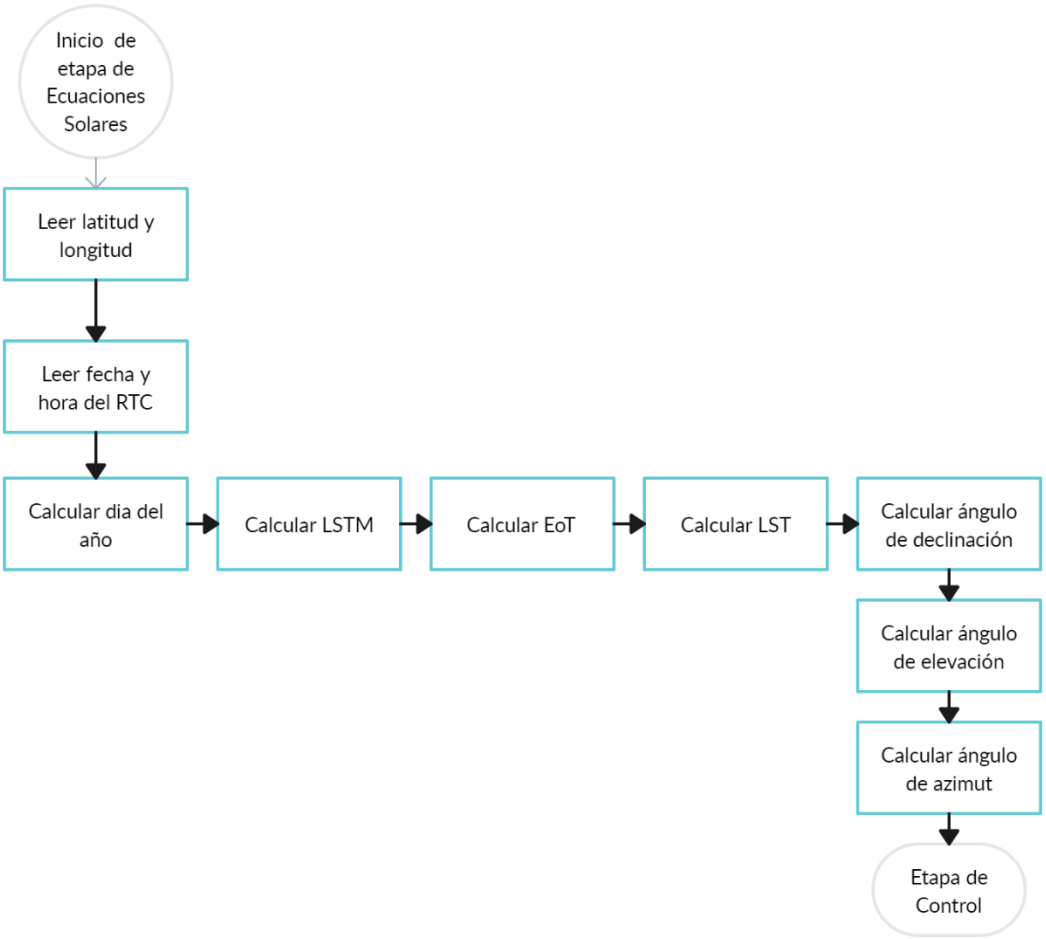


Figura 0.13 Diagrama de Flujo Ecuaciones y ángulos solares

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Se calculó los ángulos que describen su trayectoria, en primer lugar la altura angular en que se encuentra el sol o ángulo de elevación descrita en la Ecuación 10, la cual es medida desde el horizonte, esta depende principalmente de la latitud, el día del año, la declinación y el ángulo horario descrito. Y para el ángulo de azimut que se lo describe en la Ecuación 12 y se lo mide desde el norte geográfico, de la misma manera que la elevación depende de la latitud y además del mismo ángulo de elevación.

1.19.4 Etapa de control

En esta etapa donde se controlan los servomotores encargados del movimiento del sistema, en primer lugar, se obtuvo los valores del magnetómetro encargado del ángulo azimut de la estructura y los valores del acelerómetro, que refleja el ángulo de elevación de igual manera de la estructura.

Mediante la comparación de los ángulos obtenidos mediante la resolución de las ecuaciones solares y los ángulos adquiridos por los sensores de posición antes descritos, se calculó el error existente entre estos para que esta información forme parte de las variables de entrada para el controlador.

Se colocó un botón para la finalización del seguimiento solar, el cual regresa a la etapa inicial del seguimiento y por ende posición inicial de la estructura.

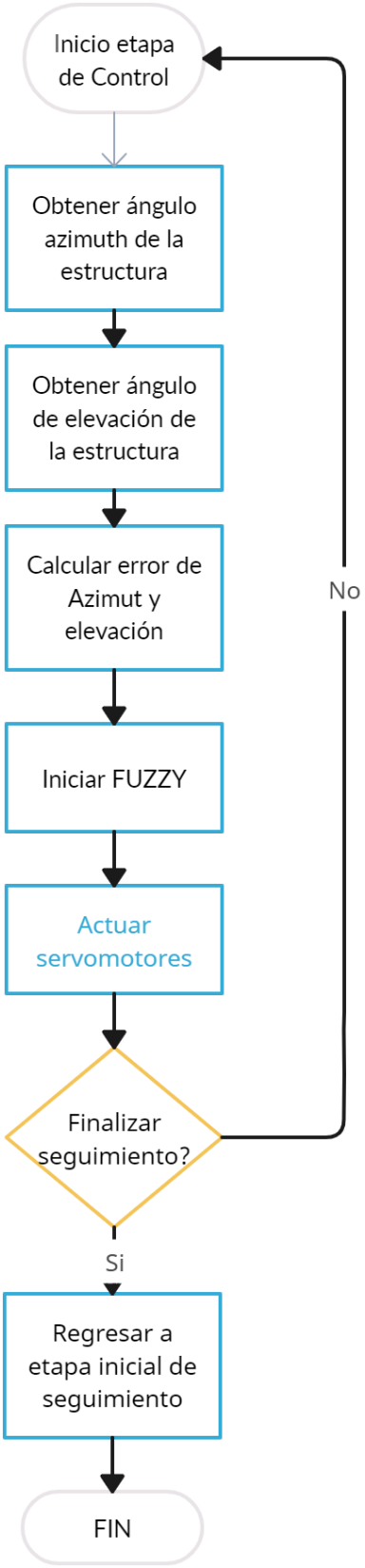


Figura 0.14 Diagrama de flujo etapa de control

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.19.4.1 Control FUZZY

En el funcionamiento de los servomotores se diseñó un control difuso, el que se determinó como un sistema mandami, de múltiples entradas y múltiples salidas (MIMO) donde sus entradas son los errores calculados de azimut y elevación, mientras que sus salidas los dos servomotores de estos ángulos. Las reglas lingüísticas de control que ejecutarán las decisiones de la base matemática del sistema forman parte de la base de conocimientos, donde se convertirán los valores reales en valores difusos, para luego relacionar dichos conjuntos de entrada y salida para representar las reglas que definirán el sistema, y finalmente adecuar estos valores difusos para el proceso de control.

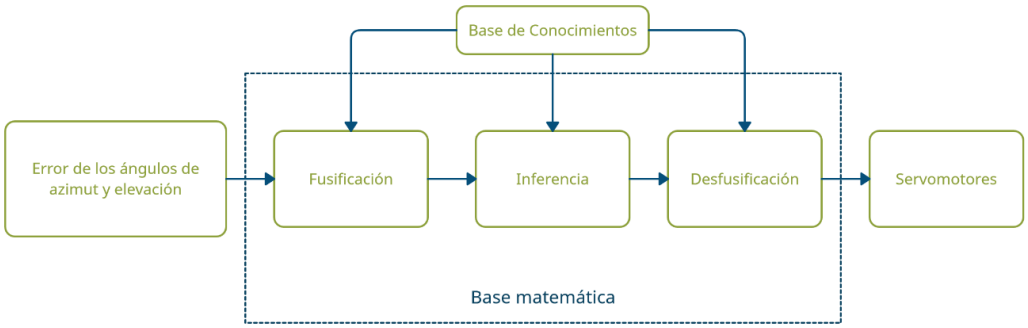


Figura 0.15 Estructura del sistema difuso

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

En la base matemática se usó el toolbox “fuzzy logic designer” de Matlab que permite crear y editar sistemas de inferencia difusos, esta aplicación se la ejecuta mediante el command Windows.

En la Figura 3.16 se presenta de color amarillo las entradas del sistema difuso, mientras el color celeste las salidas del sistema.

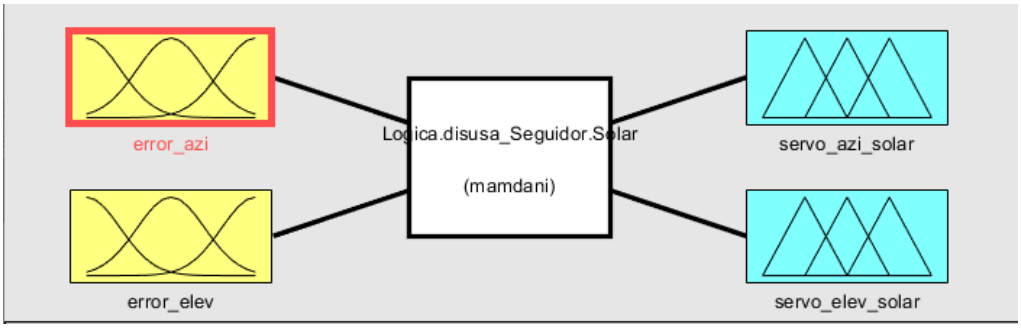


Figura 0.16 Fuzzy logic designer de Matlab

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

En la Figura 3.17a se describe el error de azimut con un universo de discurso de entre $-360^{\circ}\ a\ 360^{\circ}$, mientras en la Figura 3.17b el error de elevación con un universo de discurso de $-90^{\circ}\ a\ 90^{\circ}$. Las variables lingüísticas se las interpretó por si es un error medio o es un error grande, ya sea positivo o negativo. En las dos entradas se definió el punto centro con un error de ± 3 grados, esto para evitar que los servomotores oscilen demasiado.

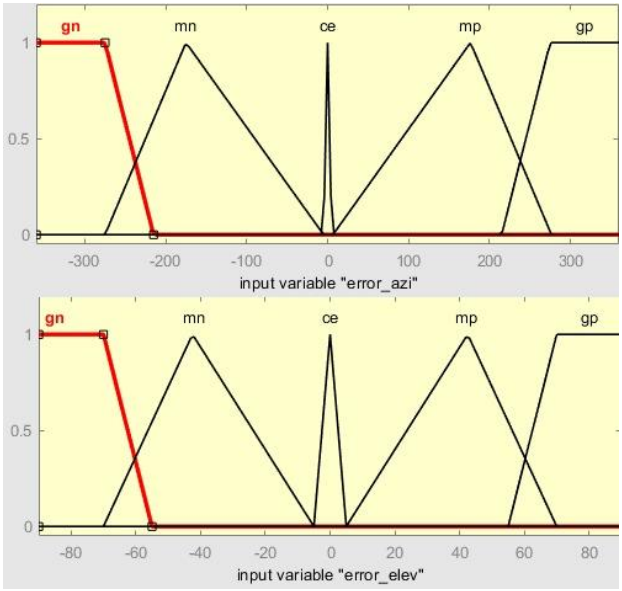


Figura 0.17 Funciones de membresía del error de azimut y elevación

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Las dos salidas del sistema representadas en la Figura 3.18 se determinó con el universo de discurso entre $-2\ a\ 2$, esto debido a que el movimiento que realizarán no debe ser muy rápido ni brusco, más bien preciso. Sus conjuntos difusos se definieron uno por si el error es positivo, el segundo por su error es negativo, y finalmente el conjunto central por si no existiese error.

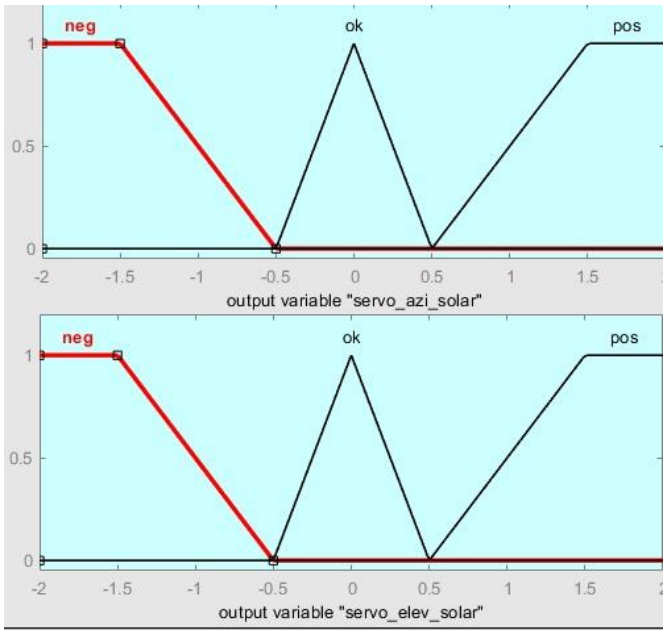


Figura 0.18 Funciones de membresía de los servomotores

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Finalmente se definió las reglas que definirán la forma en que actuará el seguidor solar en base a los conjuntos difusos de las entradas y salidas.

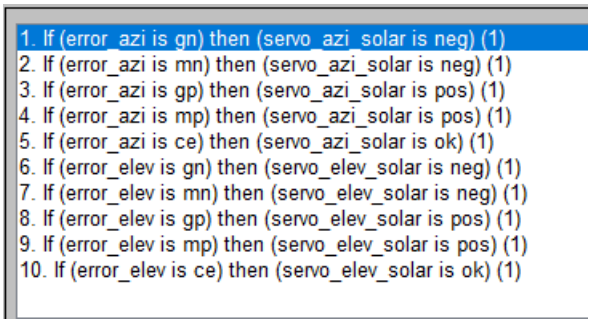


Figura 0.19 Reglas lingüísticas

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

La conversión del sistema de control fuzzy creado en Matlab con las variables de entradas, salidas y reglas de inferencia al lenguaje de programación Arduino c el cual es compatible con la programación de todo el proyecto se usó la página web "http://www.makeproto.com/projects/fuzzy/matlab_arduino_FIST/index.php" que permite dicha conversión, en la Figura 3.20 se presenta la interfaz gráfica de la página que se usó para este paso.

Figura 0.20 Interfaz gráfica convertidor de código.

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.19.5 Visualización de datos

Mediante la pantalla Ili9341 comunicada por protocolo SPI al Arduino se programó para que despliegue información referente al seguidor solar como:

-Fecha y hora determinada por el RTC.

-Los ángulos en que el sol se ubica, los que están calculadas mediante las ecuaciones solares.

-Los ángulos en que se encuentra la estructura, determinados por el acelerómetro y magnetómetro.

-Modo de operación, donde indica si la estructura se encuentra ubicada correctamente, cuando la señal del GPS es válida, el momento en que está listo para realizar el seguimiento y el instante en que está finalizando el proceso.

Figura 0.21 Visualización de información en la pantalla ILI 9341

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.20 Datos IoT

El envío de los datos tales como el modo de funcionamiento, los ángulos de la estructura y los ángulos calculados, desde el microcontrolador hacia la red y observar en el celular se usó la raspberry pi4 con su sistema operativo raspberry PI OS de 32 bits y se comunicó a esta mediante el cable de comunicación serial.

1.20.1 Instalación IDE Arduino en Raspberry

La instalación de este entorno de programación se la realizó para comprobar el puerto en que se conecta el Arduino, así como para confirmar los datos enviados mediante puerto serial.

La versión de la IDE de Arduino que se usó en este sistema operativo fue la versión 1.8.19, ya que esta cuenta con la opción de descarga para **Linux ARM de 32 bits** y esta se lo encuentra en el sitio oficial de la aplicación y es de descarga gratuita.

Para su instalación fue necesario un conjunto de comandos ejecutados por terminal, donde se buscó el paquete, se lo descomprimió y finalmente se lo instaló para su uso, tal y como se presenta en la Figura 3.22.

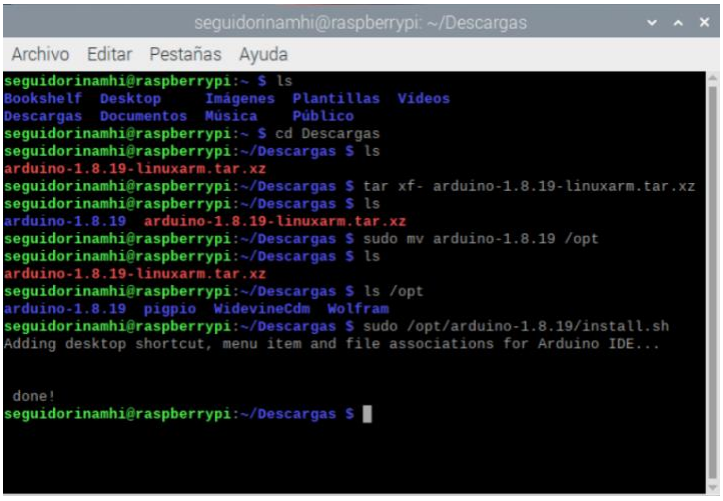


Figura 0.22 Instalación Arduino en Raspberry

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.20.2 Instalación de NODE-RED

De la misma manera que la aplicación anterior, la instalación y actualización de esta se la realizó por medio de terminal, en primer lugar mediante el comando **sudo apt-get upgrade** se actualizaron los paquetes existentes en la Raspberry, para luego por el comando **bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-**

installers/master/deb/update-nodejs-and-nodered) se lo instaló, en adición a esto algo muy importante para que este inicie automáticamente una vez iniciado el sistema operativo y así pueda leer automáticamente los datos, se usó el comando **sudo systemctl enable nodered.service.**

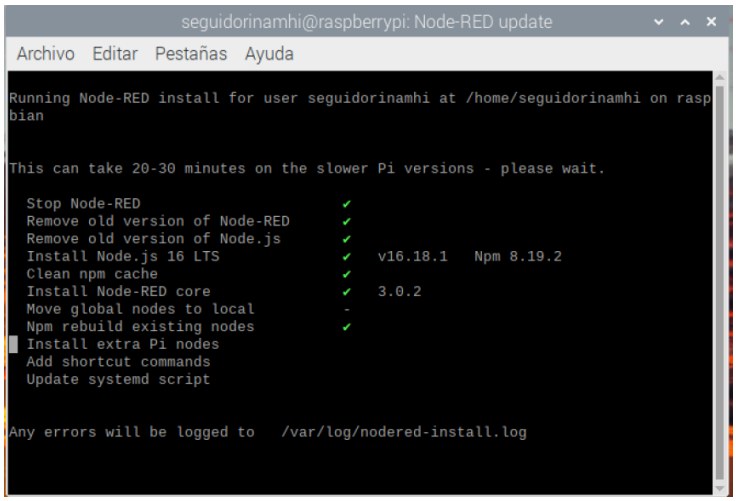


Figura 0.23 Instalación y actualización Node Red en Raspberry

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.20.3 Programación Node-Red

Los datos se enviaron por USB, para esto en las configuraciones del dispositivo se activó la interfaz Serial Port y Serial Console. La información se la compartió mediante una cadena por el puerto serial como se muestra en la Figura 3.24, para que esta sea leída en Node-Red.

```
void cadena(){
  String CadenaT;
  String modstr;
  String azsolstr;
  String azeststr;
  String elevsolstr;
  String eleveststr;

   modstr=flag1;
   if(flagstop==1)modstr="4";
   azsolstr=azsol;
   azeststr=azest;
   elevsolstr=elevsol;
 eleveststr=elevest;

CadenaT=modstr+","+azsolstr+","+azeststr+","+elevsolstr+","+elevesstr;
  Serial.println(CadenaT);
}
```

Figura 0.24 Cadena de información en Ide de Arduino

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Una vez abierta la aplicación en el navegador, se procedió descargar la extensión serialport para recibir los datos y la extensión remote para visualizar los datos en el móvil.

En la Figura 3.25 se presenta la programación visual que se efectuó en el aplicativo. Para la configuración del primer bloque se definió el puerto en que el Arduino se encontraba conectado y del cual se leyó la información.

El segundo bloque es de función donde mediante lenguaje de programación JavaScript los datos comprimidos en la cadena se presenten individualmente para su comprensión.

El siguiente bloque corresponde al dashboard, el cual fue el encargado de la publicación de la información por separado. Y finalmente el acceso remoto el cual proporcionó un código QR que mediante la aplicación REMOTE RED descargado en el celular permitió la visualización de esta mediante el móvil.

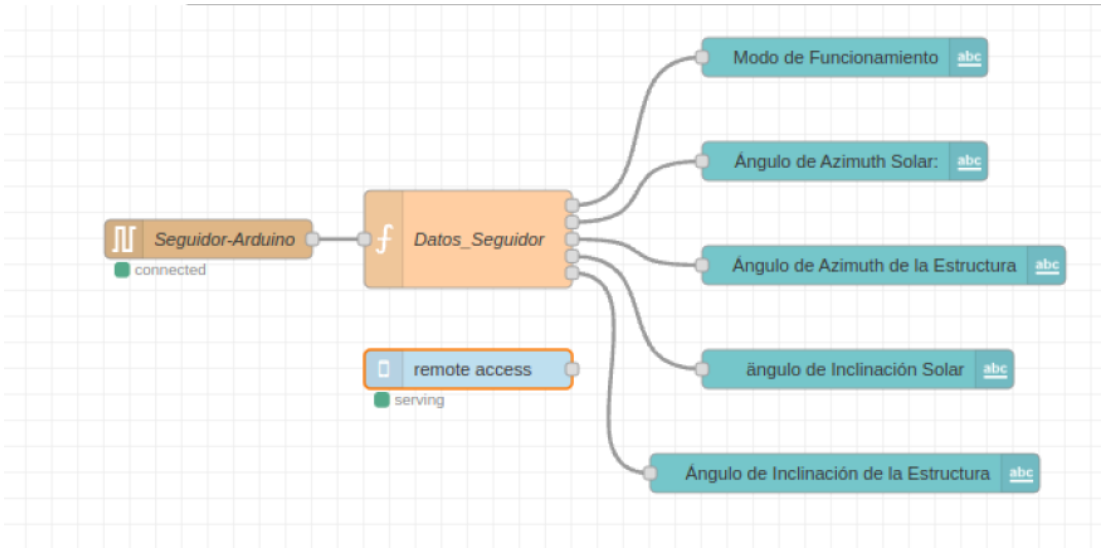


Figura 0.25 esquema gráfico de la programación en Node-RED

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.20.4 Comunicación con el datalogger y recopilación de datos en la nube

Mediante el datalogger Campbell CR1000 se obtuvo los datos de los sensores de radiación que se encuentren conectados, para visualizar estos datos es necesario establecer una serie de parámetros en el software LoggerNet y una comunicación serial RS232 a USB entre la PC y el datalogger.

Figura 0.26 Software LoggerNet

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Dentro del menú de opciones que ofrece el software LoggerNet se escoge el parámetro de configuración donde se abrirá una nueva ventana.

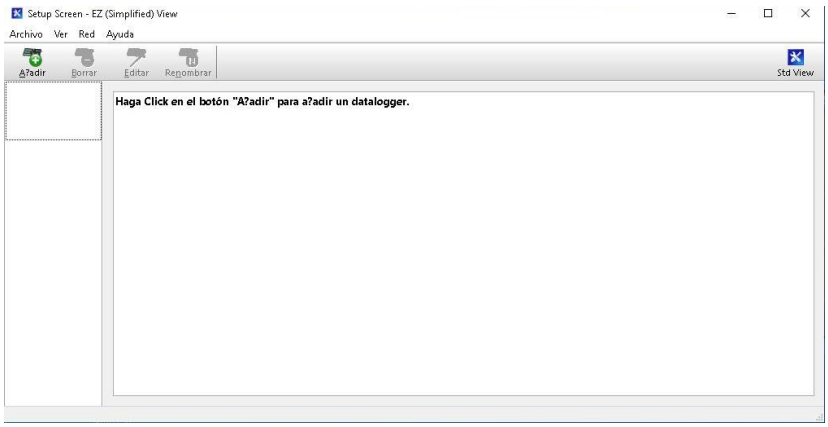


Figura 0.27 Ventana para iniciar la configuración del datalogger

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

En la parte superior de esta nueva ventana al hacer clic en la opción añadir se despliega otra ventana para iniciar la configuración del datalogger, donde se selecciona el modelo de datalogger CR1000.

Figura 0.28 Ventana para seleccionar el modelo de datalogger

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Una vez que se escogió el modelo de datalogger, se selecciona la opción de conexión directa.

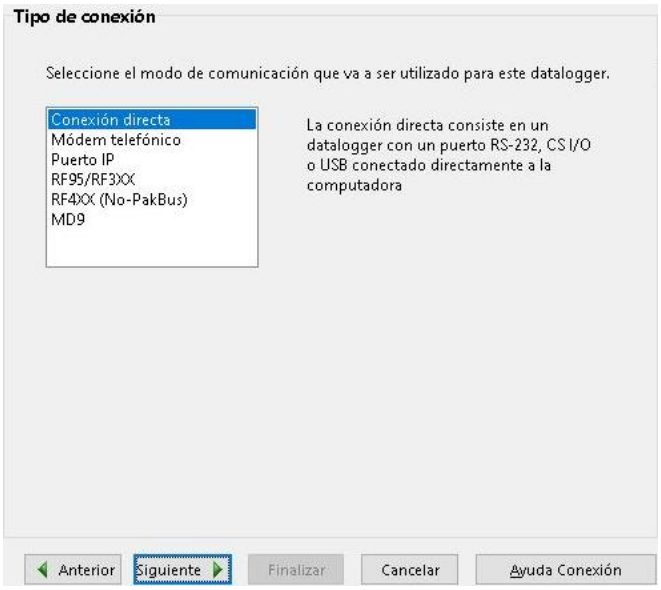


Figura 0.29 Ventana para seleccionar el tipo de conexión

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Se selecciona el puerto de comunicación que en este caso es el (COM4), si la PC no reconoce el puerto se debe verificar la instalación de los drivers.

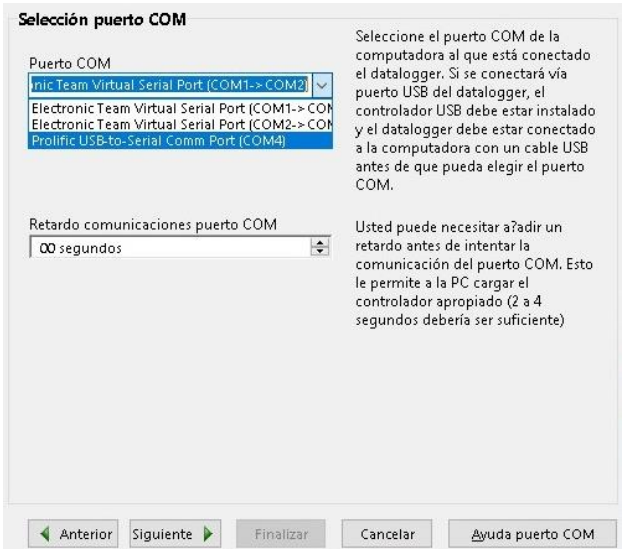


Figura 0.30 Ventana para seleccionar el puerto de comunicación

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Se configuró los parámetros del datalogger como la velocidad de transmisión que en este caso es 9600 bps, la dirección Pakbus que se colocó 8 y los tiempos de respuesta.

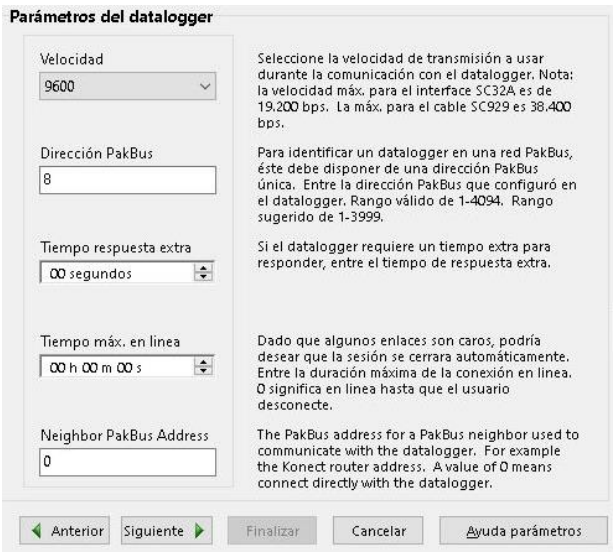


Figura 0.31 Selección de los parámetros del datalogger

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Para crear el programa que permita al datalogger leer los datos de los sensores que según el tipo en este caso de radiación están enviando, se debe escoger la opción Short Cut que está en el menú principal del programa.

Figura 0.32 Menú de herramientas software LoggerNet

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Se despliega una nueva ventana, se selecciona la opción de crear un nuevo proyecto y se escoge el modelo de datalogger que se va a implementar, en este caso CR1000.

En la siguiente ventana se escoge el tipo de sensor que se va a utilizar para ello se escoge la opción de Meteorological-Solar Radiation, se despliega una lista con varios modelos de sensores, con los datos de placa del sensor se compara y selecciona el modelo correcto que en este caso es (CMP6).

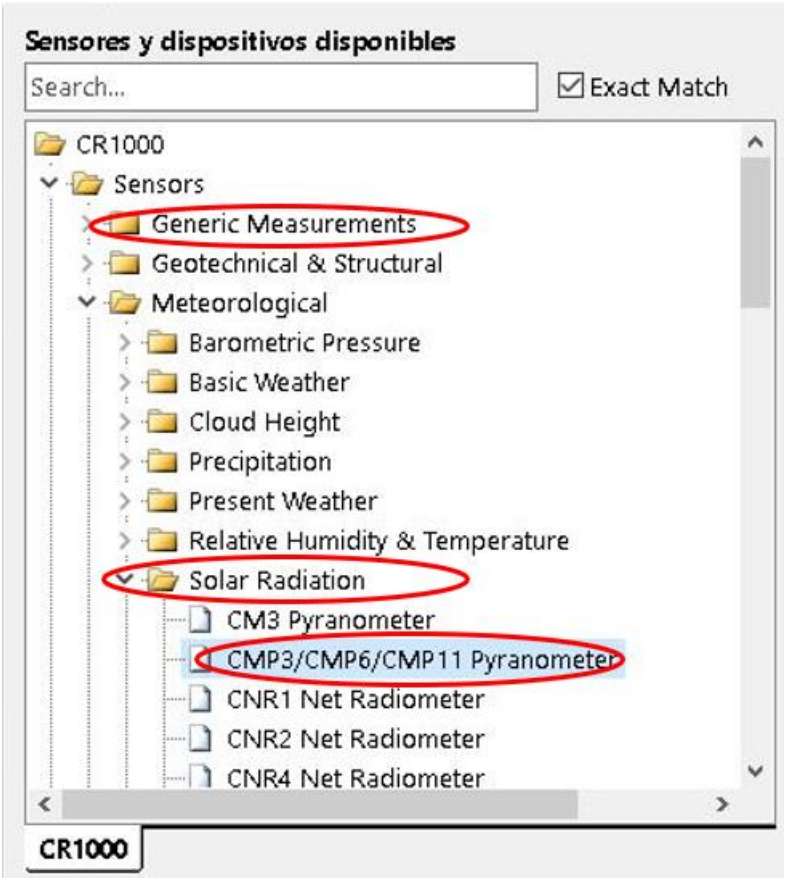


Figura 0.33 Selección del tipo de sensor que se va a conectar al datalogger

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Se configuró la sensibilidad del sensor que es de 17.27x10^-6 V/Wm^2 y las unidades de medida, también se define que parámetros se va a obtener como el valor máximo, el valor mínimo, el promedio, esto dependiendo de la necesidad del usuario.

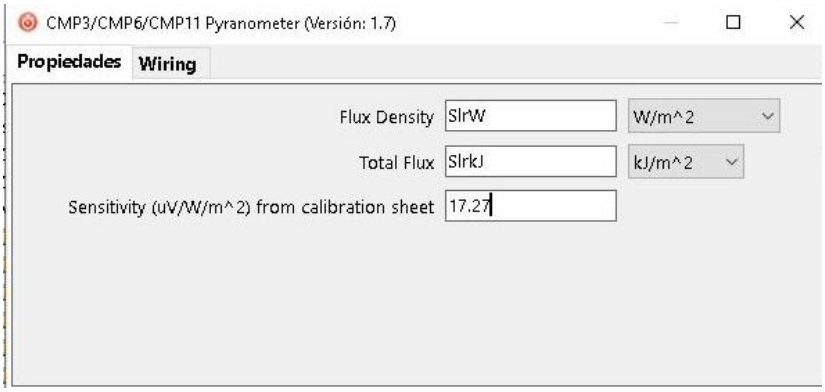


Figura 0.34 Configuración de parámetros del sensor

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Una vez creado el programa se regresa a la ventana de configuración inicial donde se ajusta la hora y fecha del reloj del datalogger con la zona horaria correspondiente.

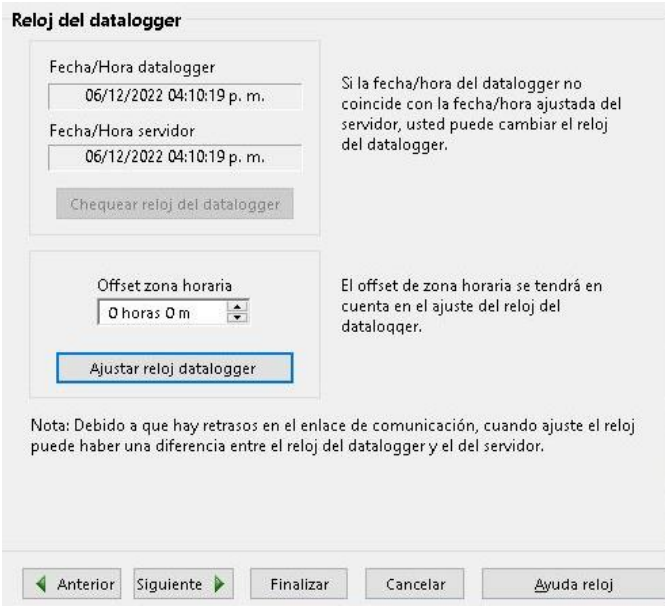


Figura 0.35 Configuración de fecha u hora del datalogger

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Se selecciona las tablas donde se encuentran los datos que envían los sensores y que se desea guardar en la nube en este caso se guardara únicamente la tabla Public.

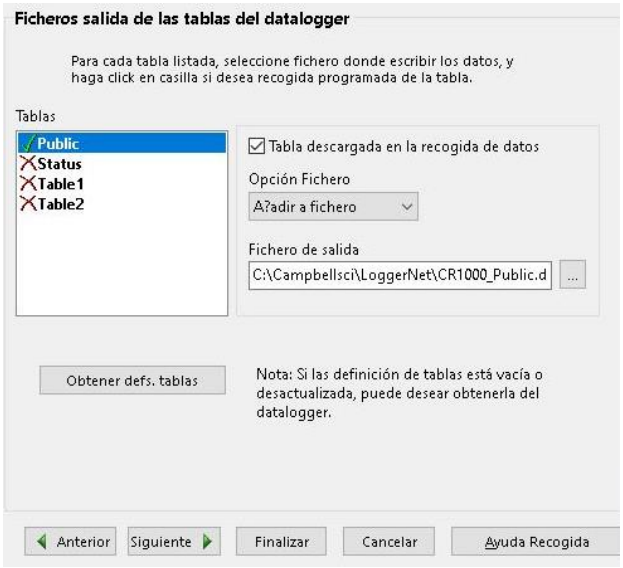


Figura 0.36 Selección de las tablas de datos para descargar

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Se activó la casilla de recolección programada para determinar la hora a la que el datalogger empezara a enviar los datos a la nube y cada que intervalo de tiempo.
En este caso la recolección de los datos a la nube inicia a las 8 am y cada 1 minuto envía un nuevo dato.

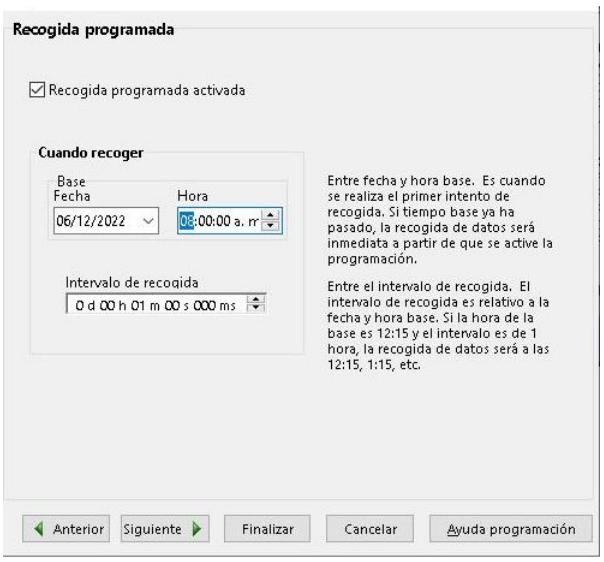


Figura 0.37 Configuración de la hora para la recolección automática de los datos

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

La verificación los datos que se están obteniendo se puede realizar desde cualquier dispositivo remotamente únicamente con el usuario y contraseña de Google Drive.

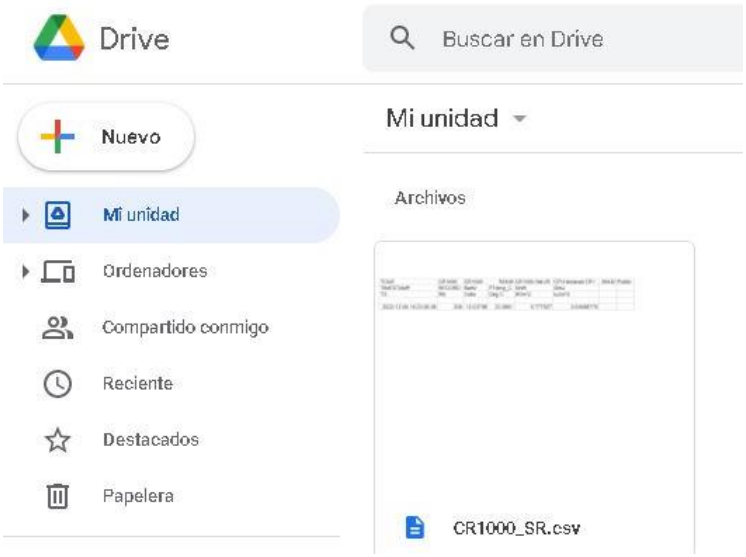


Figura 0.38 Archivo de datos en la nube

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

El archivo se sube automáticamente a la nube y se puede observar los datos en tiempo real haciendo doble clic en el archivo.

Si se desea obtener los datos en un documento de Excel se escoge la opción Abrir con Hojas de cálculo de Google.

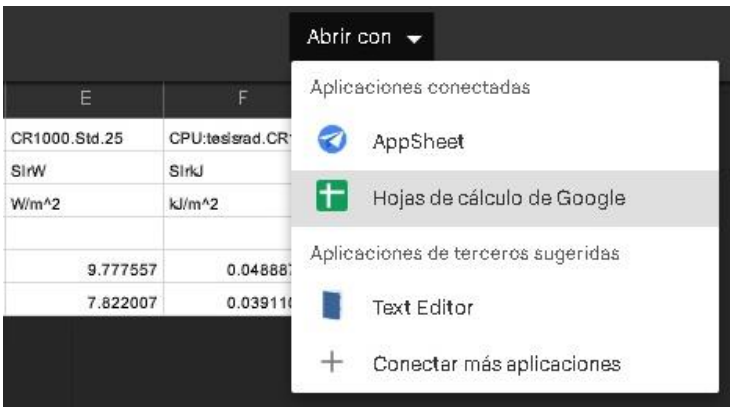
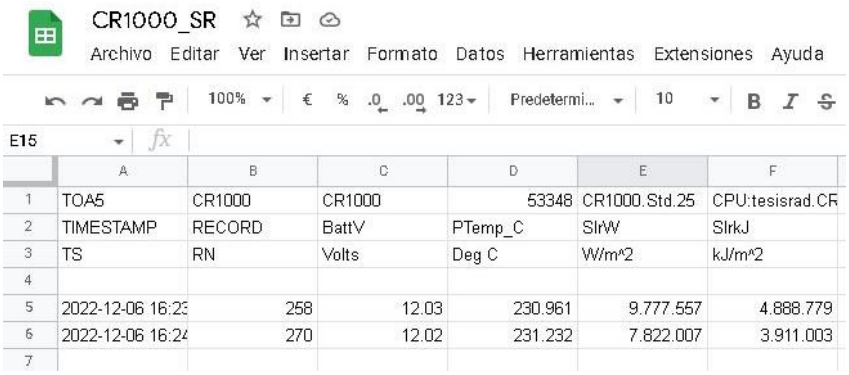


Figura 0.39 Opción para exportar los datos a una hoja de cálculo

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Se crea automáticamente una hoja de Excel con los datos que se va obteniendo, posteriormente se puede descargar esta hoja de cálculo para el análisis correspondiente.



CR1000_SR

Archivo Editar Ver Insertar Formato Datos Herramientas Extensiones Ayuda

E15

	A	B	C	D	E	F
1	TOA5	CR1000	CR1000	53348	CR1000.Std.25	CPU:tesisrad.CR
2	TIMESTAMP	RECORD	BattV	PTemp_C	SlrW	SlrkJ
3	TS	RN	Volts	Deg C	W/m^2	kJ/m^2
4						
5	2022-12-06 16:23	258	12.03	230.961	9.777.557	4.888.779
6	2022-12-06 16:24	270	12.02	231.232	7.822.007	3.911.003
7						

Figura 0.40 Visualización de los datos en una hoja de cálculo desde la nube

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.21 Montaje

Mediante Proteus, aplicación dedicada para la ejecución de proyectos electrónicos en todas sus etapas, en este caso se usó para la realización de la placa donde se colocó los elementos que componen este proyecto, se descargó el empaquetado del Arduino mega para realizar sus conexiones, los únicos componentes que van directamente a la placa son el GPS y el RTC, mientras que todos los demás componentes que van a la estructura y en el panel de control.

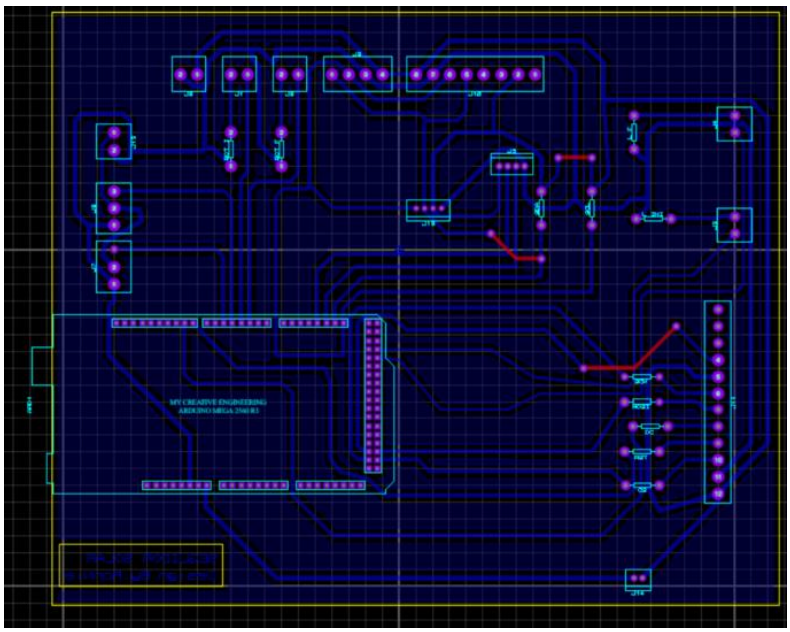


Figura 0.41 Diseño de PCB en Proteus

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

En su fabricación se usó una baquelita de fibra de vidrio con medida de 20cm x 15cm, se le sometió al método del planchado y acido férrico, para finalmente distribuir los componentes como se representa en la figura 3.42.

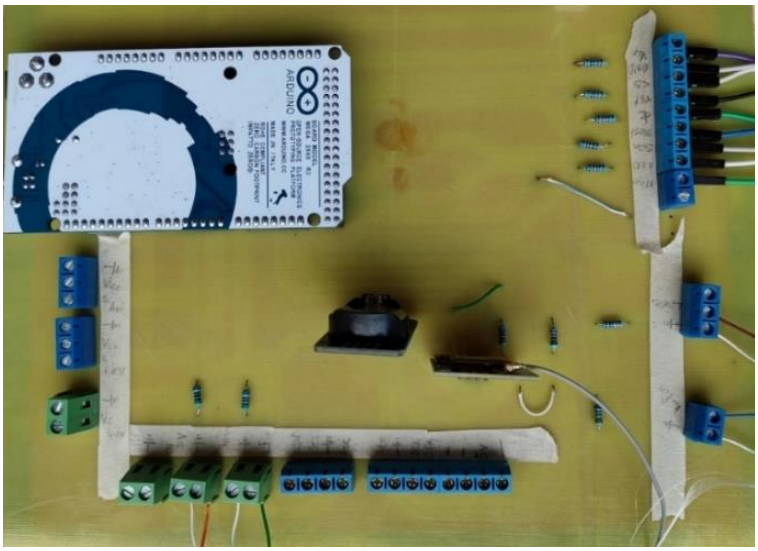


Figura 0.42 Montaje de Elementos

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

En la Figura 3.43a se presenta el diseño en el programa Solidworks del gabinete tipo nema de 40x40 donde se colocó el panel de control Figura 3.43b, en el cual se instaló un switch para prender y apagar el sistema, el botón de color blanco (B1) que inicia el seguimiento luego de que los datos de posicionamiento, fecha y hora se hayan establecido, un botón de color amarillo, que da por finalizado el seguimiento y finalmente la pantalla LCD donde se visualizarán los datos de funcionamiento del proyecto.

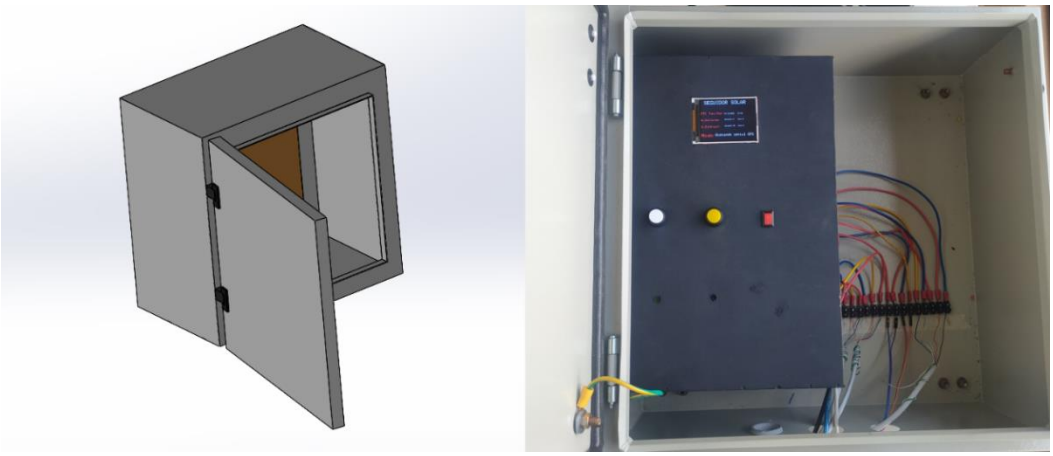


Figura 0.43 a. diseño gabinete b. gabinete con panel de control

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

CAPÍTULO 4
PRUEBAS Y RESULTADOS

La validación del funcionamiento de este proyecto se basó en pruebas para comprobar que el procesamiento de ecuaciones, seguimiento solar, datos IoT y envío de datos a la nube funcionen correctamente.

1.22 Procesamiento matemático

El primer punto para la verificación del funcionamiento de este proyecto, se centró en la capacidad que el microcontrolador Arduino Mega2650 dispone para el procesamiento matemático de las ecuaciones solares.

En primer lugar, se usó el procesamiento de una página web en el cual mediante el ingreso de datos de ubicación, fecha, hora y zona horaria proporciona los valores correspondientes a los ángulos de azimut y elevación, en la Figura 4.1 se presenta la interfaz gráfica de la página web.

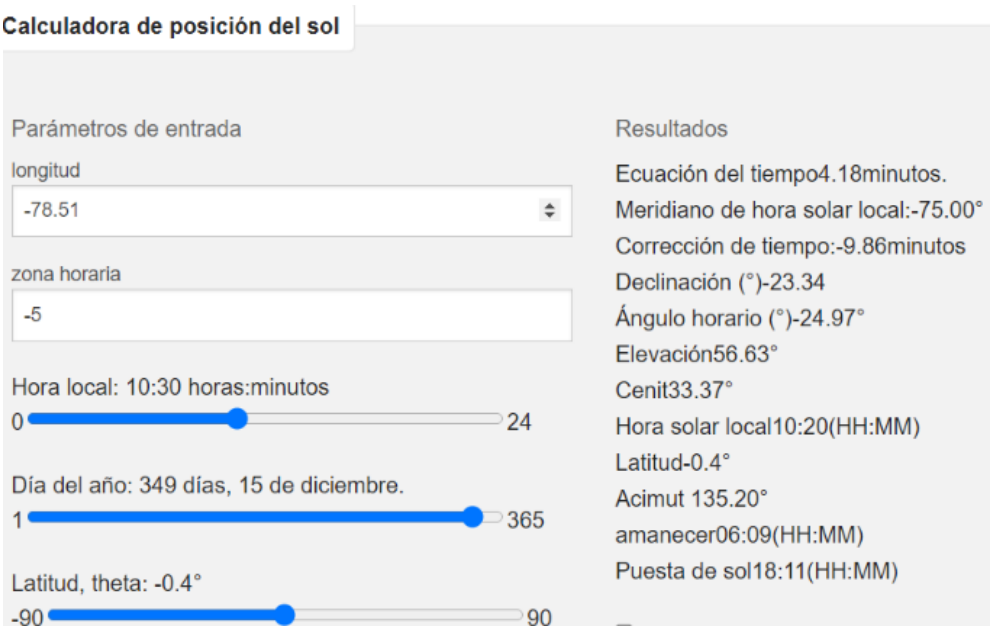


Figura 0.1 Calculadora Solar

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

El segundo método que se usó para su validación fue mediante un script realizado en Matlab dado que este software dedicado al procesamiento de datos que se lo usa mediante una computadora personal, en la Figura 4.2 se presenta una parte del código realizado, mientras que en el Anexo 4 se muestra el código completo.

```
%ecuaciones solares
LSTM=(15*gtmoff)*d2r;
B=(360/365)*(diayear-81)*d2r;
EoT=(9.87*sin(2*B))-(7.53*cos(B))-(1.5*sin(B));
TC=(4*(r2d)*(lonrad-LSTM))+EoT;
fac=minuto/60;
LT=hora+fac;
LST=LT+(TC/60);
HRA=(15*(LST-12))*d2r;
declinacion=asin(sin(23.45*d2r)*sin(d2r*(0.9863)*(diayear-81)));
decligrad=declinacion*r2d;

%angulos de seguiimiento
elevacion=asin(sin(declinacion)*sin(latrad)+cos(declinacion)*cos(latrad)*cos(HRA));
azimuth=acos(((sin(declinacion)*cos(latrad))-(cos(HRA)*cos(declinacion)*sin(latrad)))
```

Figura 0.2 Código para ecuaciones solares en Matlab

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

El tercer método fue realizar un código en la IDE de Arduino, que al igual que el código de Matlab ingresando los datos requeridos procesa las ecuaciones mediante los recursos que ofrece el microcontrolador Arduino Mega. En la Figura 4.3 se presenta parte del código usado y en el Anexo 5 se presenta el código completo.

```
//Ecuaciones Solares
double LSTM = (15*GMT)*d2r;
double B = (0.9863)*(diayear - 81) * d2r;
double EoT = (9.87 * sin(2*B)) - (7.53 * cos(B)) - (1.5 * sin(B));
double TC = (4 * (r2d * (lonrad - LSTM))) + EoT;
double LT = hora + (minuto/60);
double LST = LT + (TC/60);
double HRA = (15 * (LST-12))* d2r;
double declinacion = asin(sin(0.4092)*sin(d2r*(0.9863)*(diayear - 81)));
//Angulos de Seguimiento
double elevacion = asin(sin(declinacion)*sin(latrad) +
cos(declinacion)*cos(latrad)*cos(HRA));
double azimuth = acos((sin(declinacion)*cos(latrad) -
(cos(HRA)*cos(declinacion)*sin(latrad)))/ cos (elevacion));
```

Figura 0.3 Código para ecuaciones solares en Arduino

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

En la Tabla 4.1 se presenta la comparativa de resultados para los tres métodos usados para la obtención de las ecuaciones solares. Se realizó pruebas en diferentes fechas del año, así como distintas horas a lo largo del día, para determinar la exactitud del cálculo en el microcontrolador Arduino Mega.

Tabla 0.1 Comparación resultados ecuaciones solares

FECHA Y HORA	MATLAB	ARDUINO	CALCULADORA SOLAR
3 de enero del 2022 8:30	3 -4.5894 30.1263 116.4329	Dias del anio: 3.00 EoT: -4.69 Elevacion: 30.28 Azimut: 116.44	Día del año: 3 Ecuación del tiempo-4.59 Elevación30.12° Acimut 116.47°
15 de febrero del 2022 11:20	46 -14.5711 68.7580 127.5163	Dias del anio: 46.00 EoT: -14.57 Elevacion: 68.83 Azimut: 127.53	Día del año: 46 días Ecuación del tiempo-14.57 Elevación68.61° Acimut 128.09°
28 de marzo del 2022 15:00	87 -5.6201 49.8409 274.0039	Dias del anio: 87.00 EoT: -5.63 Elevacion: 49.95 Azimut: 274.00	Día del año: 87 Ecuación del tiempo-5.62 Elevación49.83° Acimut 274.11°
6 de abril del 2022 13:00	96 -2.7898 77.6052 299.8181	Dias del anio: 96.00 EoT: -2.80 Elevacion: 77.64 Azimut: 299.80	Día del año: 96 Ecuación del tiempo-2.79 Elevación77.53° Acimut 300.44°
14 de mayo del 2022 10:40	134 3.7596 61.0208 311.2331	Dias del anio: 134.00 EoT: 3.76 Elevacion: 61.11 Azimut: 48.75	Día del año: 134 Ecuación del tiempo3.76 Elevación60,90° Acimut 48.43°
25 de junio del 2022 17:10	176 -2.2819 15.0619 294.3722	Dias del anio: 176.00 EoT: -2.22 Elevacion: 15.24 Azimut: 294.40	Día del año: 176 Ecuación del tiempo-2.28 Elevación15.06° Acimut 294.37°

3 de julio del 2022 8:40	184 -3.8415 32.2122 297.6495	Dias del anio: 184.00 EoT: -3.78 Elevacion: 32.38 Azimut: 62.27	Día del año: 184 Ecuación del tiempo-3.84 Elevación32.21° Acimut 62.32°
20 de agosto del 2022 14:30	232 -3.0500 54.8468 291.3538	Dias del anio: 232.00 EoT: -3.16 Elevacion: 54.91 Azimut: 291.67	Día del año: 232 Ecuación del tiempo-3.05 Elevación54.77° Acimut 291.77°
30 de septiembre del 2022 9:10	273 10.8447 46.5978 95.0920	Dias del anio: 273.00 EoT: 10.65 Elevacion: 46.71 Azimut: 94.77	Día del año: 273 Ecuación del tiempo10.84 Elevación46.59° Acimut 95.24°
7 de octubre del 2022 16:50	280 12.9588 17.6939 96.6258	Dias del anio: 280.00 EoT: 12.79 Elevacion: 17.95 Azimut: 263.61	Día del año: 280 Ecuación del tiempo12.96 Elevación17.69° Acimut 263.20°
18 de noviembre del 2022 11:20	321 14.4432 68.5486 153.6755	Dias del anio: 322.00 EoT: 14.38 Elevacion: 68.51 Azimut: 153.81	Día del año: 322 Ecuación del tiempo14.21 Elevación68.18° Acimut 154.05°
31 de diciembre del 2022 16:00	365 -3.2563 31.3917 242.8861	Dias del anio: 359.00 EoT: -0.10 Elevacion: 30.75 Azimut: 242.67	Día del año: 365 Ecuación del tiempo-3.26 Elevación31.38° Acimut 242.86°

Mediante el cálculo del error porcentual para los valores obtenidos de los ángulos que describen la trayectoria del sol, se obtuvo un error promedio del 0,65% para el ángulo de elevación y de de 0,2% para el ángulo de azimut.

El error más alto en el ángulo de elevación fue de 2,01% a las16:00 del mes de diciembre, mientras que el error más bajo fue de 0,14% a las 13:00 del mes de abril.

Para el ángulo de azimut el error más bajo fue de 0,01% a las 15:00 del mes de marzo y el más alto de 0,66% a las 10:40 del mes de mayo.

1.23 Control fuzzy

La comprobación de la conversión se realizó comparando los valores obtenidos en las reglas de Matlab y las obtenidas mediante el código en la IDE de Arduino como se presenta en la Tabla 4.2.

Con esto se corroboró que los resultados proporcionados por el código convertido son los mismo que los resultados obtenidos en el toolbox fuzzy de Matlab.

Tabla 0.2 Comparativa Código Arduino y programa en Matlab

Valores de error		Arduino	Fuzzy de MATLAB
Azimut	Elevación		
-300	-60	Error Azimut -1.47 Error Elevacion -1.35	servo_azi_solar = -1.47 servo_elev_solar = -1.35
-90	-20	Error Azimut -0.80 Error Elevacion -0.57	servo_azi_solar = -0.804 servo_elev_solar = -0.568
180	30	Error Azimut 1.46 Error Elevacion 1.16	servo_azi_solar = 1.46 servo_elev_solar = 1.16
150	20	Error Azimut 1.43 Error Elevacion 0.57	servo_azi_solar = 1.43 servo_elev_solar = 0.568
78	-68	Error Azimut 0.21 Error Elevacion -1.45	servo_azi_solar = 0.212 servo_elev_solar = -1.45

Para los datos obtenidos en la tabla 4.2 se obtuvo un error porcentual promedio de 0,288% para el ángulo de azimut, mientras que para el ángulo de elevación un 0,14% de error.

1.24 Comparación de datos entre la base fija y el seguidor solar

Para determinar la eficiencia del seguidor solar respecto a una base fija de calibración, se colocó un sensor de radiación en cada estructura durante un día desde las 10:05 am hasta 16:32 am.

Figura 0.4 Comparación de datos

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

Los datos muestran una diferencia entre la base fija y el seguidor solar, para lo cual se calculó la diferencia porcentual que existe entre estos valores mediante la fórmula:

$$\left|\frac{Primer\ valor - Segundo\ valor}{(Primer\ valor + Segundo\ valor)/2}\right| \times 100\% \qquad (14)$$

Se recolecto más de 700 datos de radiación durante todo el día y se determinó que el seguidor solar presenta una eficiencia del 13,96% respecto a la base fija, este valor se determinó al calcular el promedio de la diferencia porcentual de todos los valores recolectados.

Tabla 0.3 Fragmento de los datos recolectados por la base fija y el seguidor solar

TIMESTAMP TS	Sensor Fijo W/m^2	Sensor Móvil W/m^2	Diferencia %
2022-12-20 10:05:48.41	580.1263	750.6048	25.62178
2022-12-20 10:06:00.71	594.6294	772.8659	26.06759
2022-12-20 10:06:30.64	591.7275	768.504	25.99212
2022-12-20 10:07:00.67	585.9263	758.6933	25.69753
2022-12-20 10:07:30.65	583.0245	753.2414	25.47650
2022-12-20 10:08:00.7	574.3226	742.3406	25.52179
2022-12-20 10:08:30.67	585.9242	756.5104	25.41445
2022-12-20 10:09:00.66	583.0236	753.2402	25.47650
2022-12-20 10:09:30.67	568.5198	729.2576	24.77124
2022-12-20 10:10:00.7	554.0237	707.4651	24.32703
2022-12-20 10:10:30.66	548.2224	703.1048	24.75490
2022-12-20 10:11:00.7	591.73	757.6064	24.58637
2022-12-20 10:11:30.66	626.5377	801.2097	24.46819
2022-12-20 10:12:00.67	629.4365	805.5677	24.54783
2022-12-20 10:12:30.65	632.3372	807.7478	24.36115
2022-12-20 10:13:00.7	626.5345	814.2864	26.06180
2022-12-20 10:13:30.65	623.6339	809.9261	25.99015

En la imagen a continuación se puede observar que el piranómetro colocado en el seguidor solar tiene una mejor captación de la radiación ya que los datos del mismo presentan valores mayores y el único momento en donde los valores son similares es al medio día, ya que tanto el seguidor solar como la base fija se encuentran en la misma posición.

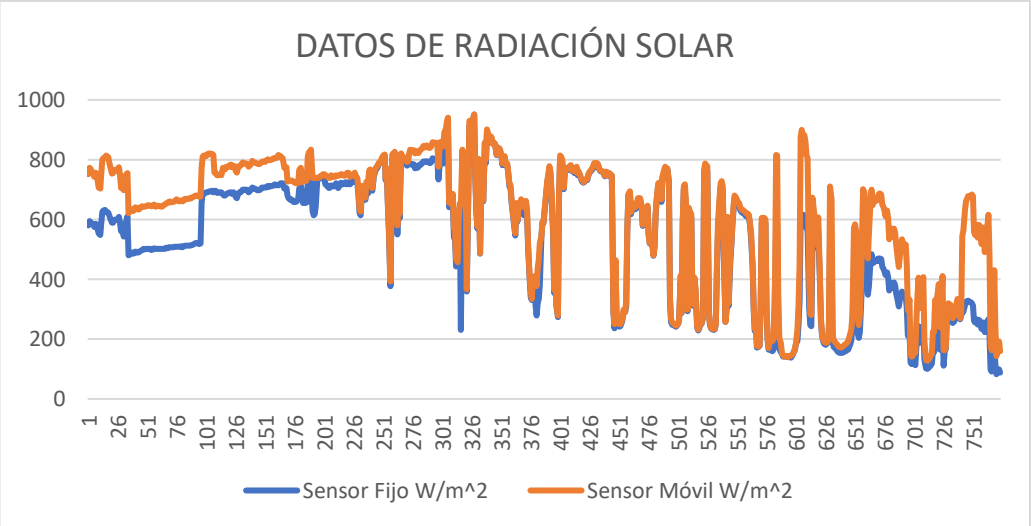


Figura 0.5 Comparación de datas entre el seguidor solar y la base fija

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

1.25 Visualización de los datos a través de IoT

Mediante la aplicación "Remote RED" funcional para Android e IOS y la programación realizada en NODE RED, se logró la representación de los datos en cualquier dispositivo móvil y por ende monitorear el funcionamiento del seguidor solar como se presenta a distintas horas en las Figuras 4.6.

Figura 0.6 Visualización de datos en el celular

Figura elaborada por: Alejandro Seraquive y Ronnie Pulupa

La conexión a internet para la raspberry se lo realizó mediante ethernet, gracias a esto la subida de datos al programa Node-RED es inmediata y así también su visualización.

CONCLUSIONES

A través del establecimiento de las ecuaciones solares en la estructura de dos ejes implementada en la empresa pública INAMHI, se consiguió una correcta descripción de la trayectoria del sol, la misma que permitió mejorar la toma de datos de sensores de radiación a comparación del sistema estático que existe actualmente.

El software de modelado Solidworks permitió diseñar el seguidor solar, basando su diseño en las dimensiones de cada una de sus piezas que forman parte del mecanismo, que fue realizado en plancha de tol de 5mm, el seguidor solar con todos los sensores y actuadores posee un peso de 20kg y sus dimensiones son de 60cm de largo y ancho, alto 170cm, además los sensores de movimiento y servomotores poseen protección contra el agua, mientras que los demás componentes se aseguran en el panel de control hermético para asegurar su vida útil.

Mediante la comparación de los ángulos calculados por las ecuaciones solares y los ángulos proporcionados por los sensores de posición colocados en cada eje, se encontró el error de posición para luego con la implementación de un sistema fuzzy MIMO, la cual traducida al lenguaje de programación de Arduino encuentra la posición correcta a la que debe llegar la estructura, manteniendo así los sensores de radiación siempre perpendicular a la trayectoria del sol, este sistema realizado en Matlab al compararlo con la programación de Arduino se obtuvo un error porcentual promedio de 0,288% para el ángulo de azimut, mientras que de 0,14 para el de elevación.

Por medio de Node-RED y su aplicativo móvil Remote Red, se consiguió la visualización de los modos de funcionamiento, tales como la posición inicial, búsqueda de la señal GPS, inicio del seguimiento y finalización del proceso, así también como los ángulos calculados y los ángulos proporcionados por los sensores de posición, además debido a la conexión a internet por Ethernet los datos visualizados en el móvil son de manera inmediata al menos en segundos, porque no se logra identificar el cambio en el último dígito del reloj.

Mediante la aplicación de Google Drive que dispone un almacenamiento de 15Gb, se estableció la comunicación entre el datalogger y cloud para el envío y recepción de

datos, lo cual permite monitorear y analizar los datos recolectados por los sensores desde cualquier dispositivo remotamente, lo cual no permitía la base existente en la empresa pública INAMHI ya que en esta un personal capacitado debía acercarse a la estación meteorológica para adquirir los datos.

Al comparar los datos recolectados por los piranómetros colocados tanto en la base fija como en el seguidor solar se procedió a calcular la diferencia porcentual que existe entre ambos dispositivos y se determinó una eficiencia superior del 13.96 % del seguidor solar respecto a la base fija existente en la empresa pública INAMHI, haciendo que este proyecto sea idóneo para la toma de datos por los próximos años gracias a su eficiencia y durabilidad de los materiales empleados en la construcción.

RECOMENDACIONES

Se recomienda colocar el módulo magnetómetro de tal manera que los campos magnéticos de la estructura o cualquier material metálico no lo afecten, ya que esto conllevaría obtener valores erróneos y por ende el seguimiento no sería el adecuado.

Al momento de encender este proyecto se debe revisar que la conexión a internet por parte de la Raspberry pi 4 sea estable, para que el envío y visualización de datos se pueda ejecutar de manera adecuada.

Para garantizar una mayor confiabilidad en la captación de los datos es necesario colocar el seguidor solar en un lugar que se encuentre libre de objetos cercanos que puedan generar algún tipo de sombra.

Se recomienda utilizar un ordenar que disponga de Windows 10 o una versión anterior, ya que en las nuevas versiones de Windows 11 se presenta problemas con los drivers del puerto serial RS232 y no se puede establecer una correcta comunicación entre el datalogger y la PC.

Para el análisis de tendencias de los datos obtenidos por los piranómetros se recomienda que el tiempo de muestreo de cada dato sea de 30 segundos, esto permitirá una mejor calibración de los sensores.

Si bien las protecciones de los servomotores y sensores de posición ayudan a prevenir que estos se mojen, no se recomienda mantener la estructura por mucho tiempo a la intemperie, por otro lado, el panel de control que contiene los demás componentes, al estar herméticamente sellado no presentarían mayor inconveniente con el agua.

REFERENCIA

Ashok, P., Jadkar, S., & Pathak, A. (2021). Weather Station for Solar PV Power Plant Using Arduino Mega. *2021 International Conference on Computer Communication and Informatics (ICCCI)*, 1–6. https://doi.org/10.1109/ICCCI50826.2021.9402478

Bento, A. (2018). IoT: Nextion X TFT ILI9341, Results of an experimental and comparative survey. *International Research Journal of Engineering, IT & Scientific Research*.

Calventus, Y., Carreras, R., Casals, M., Colomer, P., Costa, M., Jaén, A., Montserrat, S., Oliva, A., Quera, M., & Roca, X. (2006). *Tecnología energética y medio ambiente I, Volumen1* (UPC,2006, Vol. 1).

Camargo, J., Perdomo, C., & Bermúdez, B. (2019). *Proyectos prácticos con PSoC5LP* (Ediciones de la U, Vol. 1).

Campbell. (2020, January 6). *CR1000 Measurement and Control Datalogger*. https://www.campbellsci.es/cr1000

Carpio, J. (2021). *COMPARACIÓN DE MODELOS DE SEGUIMIENTO SOLAR Y ANÁLISIS ECONÓMICO-ELÉCTRICO DE UN SISTEMA SOLAR FIJO* [Carrera de Electrónica y Telecomunicaciones]. Universidad de Cuenca.

Chanthakit, S., & Rattanapoka, C. (2018). MQTT Based Air Quality Monitoring System using Node MCU and Node-RED. *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, 1–5. https://doi.org/10.1109/ICT-ISPC.2018.8523891

Dominique Breton, & Daniel Charlet. (2003, January 2). the Serial Protocol for the Experiment Control System of LHCb. *Serial Protocol for the Experiment Control System of LHCb*, 2–5.

Electroniclab. (2022, January 2). *HMC5883L – GY-271 – Magnetómetro de 3 ejes – Brújula Digital*. HMC5883L – GY-271. https://electronilab.co/tienda/hmc5883l-gy-273-magnetometro-de-3-ejes-brujula-digital/

FEETECH. (2022, January 2). *7.4V 35kg.cm degree magnetic coding servo*. Magnetic Coding Servo. https://feetechrc.com/74v-35-kgcm-digital-360-degree-magnetic-coding-aluminum-medium-shell-steel-gear-iron-core-steering-gear.html

Honsberg, C. B., & Bowden, S. G. (2019). *Photovoltaics Education Website*. The Sun's Position. www.pveducation.org

HydroMet. (2023). *Piranómetros para la medición precisa de la radiación solar*. https://www.otthydromet.com/en/products/pyranometers?origin=dropdown&c1=products&c2=solar-instruments&c3=pyranometers&clickedon=pyranometers

INJORA. (2021, January 1). *INJORA INJS035 70 KG Large Torque Waterproof Metal Gear Digital Servo for RC Model*. INJORA INJS035 70 KG. https://www.injora.com/products/25kg-35kg-large-torque-metal-gear-digital-servo-for-rc-model

Kumari, R. S. S., & Gayathri, C. (2017). Interfacing of MEMS motion sensor with FPGA using I2C protocol. *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 1–5. https://doi.org/10.1109/ICIIECS.2017.8275932

Movil Tronics. (2022, January 1). *Acelerómetro MMA7455 3 Ejes Digital*. MMA7455 3 Ejes Digital. https://moviltronics.com/tienda/acelerometro-mma7455/

Naylamp Mechatronics. (2021, January 1). *Modulo RTC DS3231*. Modulo RTC DS3231 + EEPROM AT24C32 (I2C) . https://naylampmechatronics.com/sensores/107-modulo-rtc-ds3231-eeprom-at24c32-i2c.html

Pelayo López, J. A., Luna Soto, A., Bernabe Ramos, F., & Guzmán Flores, B. (2018). Comparativa de la eficiencia entre un sistema fotovoltaico con seguimiento solar y un sistema fotovoltaico fijo / Comparison between a photovoltaic solar tracker efficiency and a fixed photovoltaic system. *CIBA Revista Iberoamericana de Las Ciencias Biológicas y Agropecuarias*, *7*(13), 105–129. https://doi.org/10.23913/ciba.v7i13.76

Rajalakshmi, A., & Shahnasser, H. (2017). Internet of Things using Node-Red and alexa. *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*, 1–4. https://doi.org/10.1109/ISCIT.2017.8261194

Roldán, J. (2011). *Determinación del potencial solar* (Paraninfo, Ed.; Primera, Vol. 1).

Roldán, J. (2012). *Determinación del potencial solar* (Paraninfo, Ed.; Segunda, Vol. 1).

Ross, Timothy. J. (2016). *Fuzzy Logic with Engineering Applications* (Wiley, Ed.; 4Th ed.).

Sidqi, R., Rio Rynaldo, B., Hadi Suroso, S., & Firmansyah, R. (2018). Arduino Based Weather Monitoring Telemetry System Using NRF24L01+. *IOP Conference*

Series: Materials Science and Engineering, *336*, 012024. https://doi.org/10.1088/1757-899X/336/1/012024

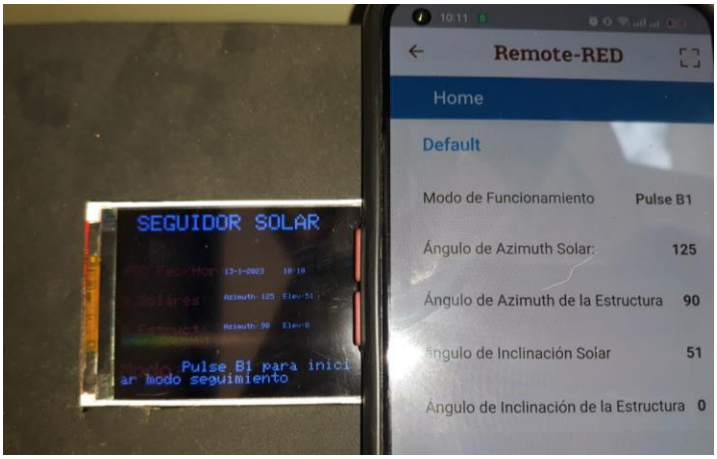
Smart Prototyping. (2021, January 1). *LCD TFT de 2,4 pulgadas (ILI9341, SPI, 240x320)*. ILI9341. https://www.smart-prototyping.com/2_4-inch-TFT-LCD-Display-Module-ILI9341-SPI-240-320

Trivedi, D., Khade, A., Jain, K., & Jadhav, R. (2018). SPI to I2C Protocol Conversion Using Verilog. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1–4. https://doi.org/10.1109/ICCUBEA.2018.8697415

u-blox. (2020). *u-blox M8 concurrent GNSS modules*. https://content.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_UBX-15031086.pdf

ANEXOS

Anexo 1 Pruebas de seguimiento



Anexo 2 Codigo de programación del Seguidor solar

```
#include <Servo.h>
#include <RTClib.h>
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
#include <TinyGPSPlus.h>
#include <Wire.h>
#include <RTClib.h>
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
#define pantalla_RST 8
#define pantalla_DC 9
#define pantalla_CS 53
#define pantalla_MOSI 51
#define pantalla_MISO 50
#define pantalla_CLK 52
Adafruit_ILI9341  tft  =  Adafruit_ILI9341(pantalla_CS,  pantalla_DC,  pantalla_MOSI,
pantalla_CLK, pantalla_RST, pantalla_MISO);
  int i=0;  String mod="", modant="";
int h=24, hant=24, m=60, mant=60;
int d=31, dant=31, me=12, meant=12, a=2022, aant=2022;
int azsol=0, azsolant=0, elevsol=0, elevsolant=0;
int azest=0,azestant=0, elevest=0, elevestant=0;
int pinInterr=2;
int pinInterr2=3;
int flag1=0, flagstop=0;
#include "fis_header.h"///////////////Fuzzy////////////////////////////
const int fus_gcentrada = 2;
const int fus_gcsalida = 2;
const int fus_gcreglas = 10;
FIS_TYPE g_fisInput[fus_gcentrada];
FIS_TYPE g_fisOutput[fus_gcsalida];
  int errorAZ, errorElev;
  float correAZ, correElev;
int degree=0; ////////////////////SERVO////////////////////////////
```

```
String words;
Servo servo1;
Servo servo2;
MPU6050sensor(0x69);
int ax, ay, az; ////////////////////////////ACELEROMETRO/////////////////////
static const uint32_t GPSBaud = 9600; ////////////////////GPS/////////////////////////
TinyGPSPlus gps;
#defineHMC5883L_ADDR0x1E//////////////////////////MAGNETOMETRO/////////////////
bool haveHMC5883L = false;
bool detectHMC5883L (){
  Wire.beginTransmission(HMC5883L ADDR);
  Wire.write(10);
  Wire.endTransmission();
  Wire.requestFrom(HMC5883L_ADDR, 3);
  if(3 == Wire.available()) {
    char a = Wire.read();
    char b = Wire.read();
    char c = Wire.read();
    if(a == 'H' && b == '4' && c == '3') return true; } return false;}
RTC_DS3231 rtc;
float angulo_xA; ////////////////////Todas las variables/////////////////
float angulo_yA;
int posactAZ=0;
int posactELEV=0;
float artan,grado;
float division=0.000;
String cadena1,cadena2,cadena3;
int GMTrtc=-5;
int rtcwrHora,rtcwMin,rtcwrSeg,rtcwrDia,rtcwrMes,rtcwrAnio;
int sec, minu, ho, daymonth,mes,year;
int di;
float flat, flon;
double GMT=-5.0;
double LSTM,Be,EcT,CorrTie,Hloc,HLS,AngH,decli;
double latiR, longR;
double elev,azi,elevG,aziG;
double decimal2radian= 0.0174; //Grados a radianes
double radian2decimal=57.2957; //Radianes a grados
void setup(){
  Serial.begin(9600);
  Serial1.begin(GPSBaud);  //
  tft.begin();
  tft.begin();
  tft.setRotation(3);   ////////////////////pantalla pincipal
  tft.fillScreen(ILI9341_BLACK);
  tft.setCursor(20, 0);
  tft.setTextColor(ILI9341_WHITE);  tft.setTextSize(3);
  tft.println("SEGUIDOR SOLAR");
  tft.setTextColor(ILI9341_RED);
  tft.setTextSize(2);
  tft.setCursor(0, 70);
  tft.println("RTC Fec/Hor: ");
  tft.setCursor(0, 110);
  tft.println("A.Solares: ");
  tft.setCursor(0, 150);
  tft.println("A.Estruct: ");
  tft.setTextSize(3);
  tft.setCursor(0, 200);
  tft.println("Modo: ");
  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(1);
  tft.setCursor(140, 75); ////////////////////////FECHA
  tft.print(d);
  tft.print("-");
  tft.print(me);
  tft.print("-");
  tft.print(a);
  tft.setCursor(220, 75); ////////////////////////HORA
  tft.print(h);
  tft.print(":");
  tft.print(m);
tft.setTextSize(1);   ////////////////////////ANGULOS SOLARES
  tft.setCursor(140, 110);
  tft.print("Azimuth:");
  tft.setCursor(190, 110);
  tft.print(azsol);
  tft.setCursor(220, 110);
  tft.print("Elev:");
  tft.setCursor(250, 110);
  tft.print(elevsol); ////////////////////////ANGULOS ESTRUCTURA
  tft.setTextSize(1);
  tft.setCursor(140, 150);
  tft.print("Azimuth:");
```

```
  tft.setCursor(190, 150);
  tft.print(azest);
  tft.setCursor(220, 150);
  tft.print("Elev:");
  tft.setCursor(250, 150);
  tft.print(elevest);
  Wire.begin();
  sensor.initialize();
  TWBR = 78;  // 25 kHz
  TWSR |= _BV (TWPS0);  // change prescaler
  if (! rtc.begin()) {        // si falla la inicializacion del modulo
 while (1); }        // bucle infinito que detiene ejecucion del programa
pinMode(pinInterr, OUTPUT); /////////interrupciones ////////////////////////////
attachInterrupt(digitalPinToInterrupt(pinInterr), sumflag, RISING);
pinMode(pinInterr2, OUTPUT);
attachInterrupt(digitalPinToInterrupt(pinInterr2), flagstp, RISING);
  servo1.attach(4,500,2500);
  servo1.write(90);
  posicionInicialElevacion();
  servo2.attach(5,500,2500);
  servo2.write(180);
  mod="Posicion Inicial";
  actualizarpantall();
  delay(2000);
 while(flag1==0){   //// Posicionamiento  0° hacia el norte
 cadena(); posicionamiento();actualizarpantall();
  delay(1000);}
  while(flag1==1&&flagstop==0){ cadena();  //Busqueda de GPS
  smartDelay(1000); GPS();
  mod="Buscando senial GPS";
  actualizarpantall();}RTCwrite();
  delay(500);
actualizarpantall(); delay(500);  }
void cadena(){
  String CadenaT;
  String modstr;
  String azsolstr;
  String azeststr;
  String elevsolstr;
  String eleveststr;
   modstr=flag1;
   if(flagstop==1)modstr="4";
   azsolstr=azsol;
   azeststr=azest;
   elevsolstr=elevsol;
 eleveststr=elevest;
CadenaT=modstr+","+azsolstr+","+azeststr+","+elevsolstr+","+eleveststr;
  Serial.println(CadenaT);}
void loop(){
  RTC(); dayyear();  Ecuaciones();   angulos(); magnetometro(); ACELEROMETRO();
actualizarpantall();
delay(1000);
  while(flag1==2&&flagstop==0){
    cadena();
  mod="Pulse B1 para iniciar modo seguimiento";
  delay(2000);
  actualizarpantall();}
mod="Seguimiento";
actualizarpantall();
//////////////////// SEGUIMIENTO///////////////
while(flag1==3&&flagstop==0){
  actualizarpantall();
  cadena();
  mod="Seguimiento";
  RTC();dayyear(); Ecuaciones();   angulos();  magnetometro();  ACELEROMETRO();
errorAZ=aziG-grado;
errorElev=elevG-(90-angulo_yA);
while(errorAZ>5||errorElev>5){    ////////////////////////////Fuzzy/////////////////
magnetometro();ACELEROMETRO();
errorAZ=aziG-grado;
errorElev=elevG-(90-angulo_yA);Fuzzy();
  posactAZ=servo2.read();
  posactAZ=posactAZ-correAZ;
  /// SACAR EL ERROR Y MANDAR AL FUZZY
  posactELEV=servo1.read();
  posactELEV=posactELEV+correElev;
  servo1.write(posactELEV);   //pin 4 elev
//posactAZ=180-posactAZ;    ///sentido de giro cambiado
servo2.write(posactAZ); //pin 5 azi
actualizarpantall();
delay(100);}
actualizarpantall();
  delay(60000);}
```

```
while(flag1==3&&flagstop==1){cadena(); ////////////////////PARADA/////////////
mod="Finalizando seguimiento";
actualizarpantall();
delay(1500);
posicionInicialElevacion(); ////////////////////volver a posicion standard
posicionInicialAzimuth();
delay(1000);
flag1=2; flagstop=0;}}
void actualizarpantall(){
if(h!=hant || m!=mant)actHora();
  if(d!=dant || me!=meant|| a!=aant)actFecha();
  if(azsol!=azsolant || elevsol!=elevsolant)actaglsol();
  if(azest!=azestant || elevest!=elevestant)actaglest();
  if(mod!=modant)modos();}
void Fuzzy(){//////////////errorAZ, errorElev;
  // Read Input: error/azhi
    g_fisInput[0] = errorAZ;
    // Read Input: error_inclinacion
    g_fisInput[1] = errorElev;
    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;
    fis_evaluate();
    correAZ=g_fisOutput[0];
    correElev=g_fisOutput[1];}
void sumflag(){
  flag1=flag1+1;
  delay(5000);
  if (flag1>3)flag1=3;}
void flagstp(){  flagstop=flagstop+1;
if (flag1!=3)flagstop=0;
  delay(5000);
  if (flagstop>1)flagstop=1;}
void posicionamiento(){ magnetometro();
  if(grado>92 || grado<88 ){
    mod="Posicione el seguidor hacia el norte";
    azest=90-grado;
    actualizarpantall(); }
  else{  mod="Seguidor posicionado";
    actualizarpantall();
    delay(1500);
    flag1=1;}}
void posicionInicialElevacion(){
  int deact=servo1.read();
        while(servo1.read()!=90)  {
          if(servo1.read()>90){
          servo1.write(deact);
          deact=deact-1;
          delay(100); }else{
          servo1.write(deact);
          deact=deact+1;
          delay(100); } }}
void posicionInicialAzimuth(){
  int deact=servo2.read();
        while(servo2.read()!=180) {
          if(servo2.read()>180){
          servo2.write(deact);
          deact=deact-1;
          delay(100);}else{
          servo2.write(deact);
          deact=deact+1;
          delay(100); }  } }
static void smartDelay(unsigned long ms){
  unsigned long start = millis();
  do  { while (Serial1.available())
      gps.encode(Serial1.read()); } while (millis() - start < ms); }
void dayyear(){
  int monthprev=mes-1;
  di=((monthprev)*30)+(daymonth);
  if(monthprev==1)di=di+1;
  if(monthprev==2)di=di-1;
  else if(monthprev==5||monthprev==6)di=di+1;
  else if(monthprev==7)di=di+2;
  else if(monthprev==8||monthprev==9)di=di+3;
  else if(monthprev==10||monthprev==11)di=di+4;
 if (year%4==0){}
 else if (year%100==0);
 else if (year%400==0) di=di+1;}
void Ecuaciones(){
    LSTM=(15*GMT)*decimal2radian;
    Be=(0.9863)*(di-81)*decimal2radian;
    EcT=(9.87*sin(2*Be))-(7.53*cos(Be))-(1.5*sin(Be));
    CorrTie=(4*(radian2decimal*(longR-LSTM)))+EcT;
    double fac=minu/60.0;
```

```
    Hloc=ho+fac;
    HLS=Hloc+(CorrTie/60.0);
    AngH=(15*(HLS-12))*decimal2radian;
    decli=asin(sin(23.45*decimal2radian)*sin(decimal2radian*(0.9863)*(di-81)));
    double decliG=decli*radian2decimal;
    double HRAgrad= AngH*radian2decimal;}
void angulos(){
  elev=asin(sin(decli)*sin(latiR)+cos(decli)*cos(latiR)*cos(AngH));
 azi=acos((sin(decli)*cos(latiR)-(cos(AngH)*cos(decli)*sin(latiR)))/cos(elev));
  elevG=elev*radian2decimal;
  aziG=azi*radian2decimal;
  if(ho>=12){
    aziG=360-aziG;  }
    elevsol=elevG;
    azsol=aziG;}
void GPS(){  if (gps.location.isValid()){
    flat=gps.location.lat();
    flon=gps.location.lng();
    latiR=flat*decimal2radian;
  longR=flon*decimal2radian;
  flag1=2;}
  else{   flag1=1;}
  if (gps.date.isValid())  {  rtcwrMes=gps.date.month();   rtcwrDia=gps.date.day();
   rtcwrAnio=gps.date.year();}
  if (gps.time.isValid()){
    if (gps.time.hour() < 10); rtcwrHora=gps.time.hour();
    if (gps.time.minute() );
    rtcwMin=gps.time.minute();
    if (gps.time.second() < 10);
    rtcwrSeg=gps.time.second(); }}
void RTCwrite(){
for(int i=0; i<(abs(GMTrtc)); i++){
  if(GMTrtc>0){
    rtcwrHora=rtcwrHora+1;
    if((rtcwrHora)>=23){
      rtcwrHora=0;
      rtcwrDia=rtcwrDia+1;      }}
  if(GMTrtc<0){
    rtcwrHora=rtcwrHora-1;
    if((rtcwrHora)<0){
      rtcwrHora=23;
      rtcwrDia=rtcwrDia-1;}}}
rtc.adjust(DateTime(rtcwrAnio,rtcwrMes,rtcwrDia,rtcwrHora,rtcwMin,rtcwrSeg));
delay(500);}
void RTC(){
DateTime fecha = rtc.now();     // funcion que devuelve fecha y horario en formato
 daymonth=(fecha.day());
 d=daymonth;
 mes=fecha.month();
 me=mes;
 year=fecha.year();
 a=year;
 ho=fecha.hour();
 h=ho;
 minu=fecha.minute();
 m=minu;
 sec=fecha.second();}
void magnetometro(){
  bool detect = detectHMC5883L(); if(!haveHMC5883L) {
    if(detect) {
      haveHMC5883L = true;
      Wire.beginTransmission(HMC5883L_ADDR); //open communication with HMC5883
      Wire.write(0x02); //select mode register
      Wire.write(0x00); //continuous measurement mode
      Wire.endTransmission(); }
    else{   return;}}else{
    if(!detect) { haveHMC5883L = false; return;}}
  int x,y,z; //triple axis data
  float declinacion=-4.13;    //-4.19;
  Wire.beginTransmission(HMC5883L_ADDR);
  Wire.write(0x03);
  Wire.endTransmission();
  Wire.requestFrom(HMC5883L_ADDR, 6);
  if(6<=Wire.available()){
    x = Wire.read()<<8; //X msb
    x |= Wire.read(); //X lsb
    z = Wire.read()<<8; //Z msb
    z |= Wire.read(); //Z lsb
    y = Wire.read()<<8; //Y msb
    y |= Wire.read(); //Y lsb }
  float x2,y2;
  x2=x;
  y2=y;
```

```
  division=(y2/x2);
  artan=atan(division);
  grado=artan*(180/3.14);
  if((x<0)&&(y>0))grado=180+grado;
  if((x<0)&&(y<0))grado=180+grado;
  if((x>0)&&(y<0))grado=360+grado;
grado=grado+declinacion;
  if(grado<0)grado=grado+360;
  if(grado>360)(grado=(360-grado)*-1);
  azest=grado;}
void ACELEROMETRO() {
  sensor.getAcceleration(&ax, &ay, &az);
  angulo_xA=atan(ax/sqrt(pow(ay,2) + pow(az,2)))*(180.0/3.14);
  angulo_yA=atan(ay/sqrt(pow(ax,2) + pow(az,2)))*(180.0/3.14);
  elevest=90-angulo_yA;
  delay(10);}
////////////////////////////////////actualizacion lcd//////////////////////////////
void modos(){
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.setCursor(85, 200);
  tft.println(modant);
  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(2);
  tft.setCursor(85, 200);
  tft.println(mod);
  modant=mod;}
void actaglest(){
  tft.setTextSize(1);
  tft.setTextColor(ILI9341_BLACK);
  tft.setCursor(190, 150);
  tft.print(azestant);
  tft.setCursor(250, 150);
  tft.print(elevestant);
  tft.setTextColor(ILI9341_WHITE);
  tft.setCursor(190, 150);
  tft.print(azest);
  tft.setCursor(250, 150);
  tft.print(elevest);
  azestant=azest;
  elevestant=elevest;}
void actaglsol(){
  tft.setTextSize(1);
   tft.setTextColor(ILI9341_BLACK);
  tft.setCursor(190, 110);
  tft.print(azsolant);
  tft.setCursor(250, 110);
  tft.print(elevsolant);
   tft.setTextColor(ILI9341_WHITE);
  tft.setCursor(190, 110);
  tft.print(azsol);
  tft.setCursor(250, 110);
  tft.print(elevsol);
  azsolant=azsol;
  elevsolant=elevsol;}
  void actHora(){   //(220, 75); ///actualizacion de ho
  tft.setTextSize(1);
  tft.setCursor(220,75);
  tft.setTextColor(ILI9341_BLACK);
  tft.print(hant);
  tft.print(":");
  tft.print(mant);
  tft.setCursor(220,75);
  tft.setTextColor(ILI9341_WHITE);
  tft.print(h);
  tft.print(":");
  tft.print(m);
  delay(1000);
  hant=h;
  mant=m; }
void actFecha(){
  tft.setTextSize(1);
  tft.setCursor(140, 75);
  tft.setTextColor(ILI9341_BLACK);
  tft.print(dant);
  tft.print("-");
  tft.print(meant);
  tft.print("-");
  tft.print(aant);
  tft.setCursor(140, 75);
  tft.setTextColor(ILI9341_WHITE);
  tft.print(d);
  tft.print("-");
```

```
  tft.print(me);
  tft.print("-");
  tft.print(a);
  dant=d;
  meant=me;
  aant=a;}
////////////////////////////////////Fuzzy/////////////////////////////////
FIS_TYPE fus_trapmf(FIS_TYPE x, FIS_TYPE* p){
    FIS_TYPE a = p[0], b = p[1], c = p[2], d = p[3];
FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0 : ((c != d) ? ((d - x) / (d - c)) : 0)));
FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0 : ((a != b) ? ((x - a) / (b - a)) : 0)));
    return (FIS_TYPE) minimo(t1, t2);}
FIS_TYPE fus_trimf(FIS_TYPE x, FIS_TYPE* p){
    FIS_TYPE a = p[0], b = p[1], c = p[2];
    FIS_TYPE t1 = (x - a) / (b - a);
    FIS_TYPE t2 = (c - x) / (c - b);
    if ((a == b) && (b == c)) return (FIS_TYPE) (x == a);
    if (a == b) return (FIS_TYPE) (t2*(b <= x)*(x <= c));
    if (b == c) return (FIS_TYPE) (t1*(a <= x)*(x <= b));
    t1 = minimo(t1, t2);
    return (FIS_TYPE) maximo(t1, 0);}
FIS_TYPE fus_minimo(FIS_TYPE a, FIS_TYPE b){
    return minimo(a, b);}
FIS_TYPE fus_maximo(FIS_TYPE a, FIS_TYPE b){
    return maximo(a, b);}
FIS_TYPE fus_array_operation(FIS_TYPE *array, int size, _FIS_ARR_OP pfnOp){
    int i;
    FIS_TYPE ret = 0;
    if (size == 0) return ret;
    if (size == 1) return array[0];
    ret = array[0];
    for (i = 1; i < size; i++){
        ret = (*pfnOp)(ret, array[i]); }
    return ret;}
_FIS_MF fus_gMF[] ={
    fus_trapmf, fus_trimf};
int fus_gIMFCount[] = { 5, 5 };
int fus_gOMFCount[] = { 3, 3 };
FIS_TYPE fus_gMFI0Coeff1[] = { -360, -360, -275, -215 };
FIS_TYPE fus_gMFI0Coeff2[] = { -275, -175, -5 };
FIS_TYPE fus_gMFI0Coeff3[] = { 215, 275, 360, 360 };
FIS_TYPE fus_gMFI0Coeff4[] = { 6.522198731501, 176.522198731501, 276.522198731501 };
FIS_TYPE fus_gMFI0Coeff5[] = { -5, 0, 5 };
FIS_TYPE* fus_gMFI0Coeff[] = { fus_gMFI0Coeff1, fus_gMFI0Coeff2, fus_gMFI0Coeff3,
fus_gMFI0Coeff4, fus_gMFI0Coeff5 };
FIS_TYPE fus_gMFI1Coeff1[] = { -90, -90, -70, -55 };
FIS_TYPE fus_gMFI1Coeff2[] = { -70, -42.5, -5 };
FIS_TYPE fus_gMFI1Coeff3[] = { -5, 0, 5 };
FIS_TYPE fus_gMFI1Coeff4[] = { 5, 42.5, 70 };
FIS_TYPE fus_gMFI1Coeff5[] = { 55, 70, 90, 90 };
FIS_TYPE* fus_gMFI1Coeff[] = { fus_gMFI1Coeff1, fus_gMFI1Coeff2, fus_gMFI1Coeff3,
fus_gMFI1Coeff4, fus_gMFI1Coeff5 };
FIS_TYPE** fus_gMFICoeff[] = { fus_gMFI0Coeff, fus_gMFI1Coeff };
FIS_TYPE fus_gMFO0Coeff1[] = { -2, -2, -1.5, -0.5 };
FIS_TYPE fus_gMFO0Coeff2[] = { -0.5, 0, 0.5 };
FIS_TYPE fus_gMFO0Coeff3[] = { 0.5, 1.5, 2, 2 };
FIS_TYPE* fus_gMFO0Coeff[] = { fus_gMFO0Coeff1, fus_gMFO0Coeff2, fus_gMFO0Coeff3 };
FIS_TYPE fus_gMFO1Coeff1[] = { -2, -2, -1.5, -0.5 };
FIS_TYPE fus_gMFO1Coeff2[] = { -0.5, 0, 0.5 };
FIS_TYPE fus_gMFO1Coeff3[] = { 0.5, 1.5, 2, 2 };
FIS_TYPE* fus_gMFO1Coeff[] = { fus_gMFO1Coeff1, fus_gMFO1Coeff2, fus_gMFO1Coeff3 };
FIS_TYPE** fus_gMFOCoeff[] = { fus_gMFO0Coeff, fus_gMFO1Coeff };
// Input membership function set
int fus_gMFI0[] = { 0, 1, 0, 1, 1 };
int fus_gMFI1[] = { 0, 1, 1, 1, 0 };
int* fus_gMFI[] = { fus_gMFI0, fus_gMFI1};
// Output membership function set
int fus_gMFO0[] = { 0, 1, 0 };
int fus_gMFO1[] = { 0, 1, 0 };
int* fus_gMFO[] = { fus_gMFO0, fus_gMFO1};
// Rule Weights
FIS_TYPE fus_gRWeight[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
// Rule Type
int fus_gRType[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
// Rule Inputs
int fus_gRI0[] = { 1, 0 };
int fus_gRI1[] = { 2, 0 };
int fus_gRI2[] = { 3, 0 };
int fus_gRI3[] = { 4, 0 };
int fus_gRI4[] = { 5, 0 };
int fus_gRI5[] = { 0, 1 };
int fus_gRI6[] = { 0, 2 };
int fus_gRI7[] = { 0, 5 };
```

```
int fus_gRI8[] = { 0, 4 };
int fus_gRI9[] = { 0, 3 };
int* fus_gRI[] = { fus_gRI0, fus_gRI1, fus_gRI2, fus_gRI3, fus_gRI4, fus_gRI5, fus_gRI6,
fus_gRI7, fus_gRI8, fus_gRI9 };
// Rule Outputs
int fus_gRO0[] = { 1, 0 };
int fus_gRO1[] = { 1, 0 };
int fus_gRO2[] = { 3, 0 };
int fus_gRO3[] = { 3, 0 };
int fus_gRO4[] = { 2, 0 };
int fus gRO5[] = { 0, 1 };
int fus_gRO6[] = { 0, 1 };
int fus_gRO7[] = { 0, 3 };
int fus_gRO8[] = { 0, 3 };
int fus_gRO9[] = { 0, 2 };
int* fus_gRO[] = { fus_gRO0, fus_gRO1, fus_gRO2, fus_gRO3, fus_gRO4, fus_gRO5, fus_gRO6,
fus_gRO7, fus_gRO8, fus_gRO9 };
FIS_TYPE fus_gIMin[] = { -360, -90 };
FIS_TYPE fus_gIMax[] = { 360, 90 };
FIS_TYPE fus_gOMin[] = { -2, -2 };
FIS_TYPE fus_gOMax[] = { 2, 2 };
FIS_TYPE fus_MF_out(FIS_TYPE** fuzzyRuleSet, FIS_TYPE x, int o){
    FIS_TYPE mfOut;
    int r;
    for (r = 0; r < fus_gcreglas; ++r){
        int index = fus_gRO[r][o];
        if (index > 0){index = index - 1;
            mfOut = (fus_gMF[fus_gMFO[o][index]])(x, fus_gMFOCoeff[o][index]);}
        else if (index < 0){index = -index - 1;
            mfOut = 1 - (fus_gMF[fus_gMFO[o][index]])(x, fus_gMFOCoeff[o][index]);}
        else{ mfOut = 0;}
        fuzzyRuleSet[0][r] = fus_minimo(mfOut, fuzzyRuleSet[1][r]);}
    return fus_array_operation(fuzzyRuleSet[0], fus_gcreglas, fus_maximo);}
FIS_TYPE fus_defuzz_centroid(FIS_TYPE** fuzzyRuleSet, int o){
    FIS_TYPE step = (fus_gOMax[o] - fus_gOMin[o]) / (FIS_RESOLUSION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;
    for (i = 0; i < FIS RESOLUSION; ++i){
        dist = fus_gOMin[o] + (step * i);
        slice = step * fus_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;}
    return ((area == 0) ? ((fus_gOMax[o] + fus_gOMin[o]) / 2) : (momentum / area));}
void fus_evaluate(){
    FIS_TYPE fuzzyInput0[] = { 0, 0, 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0, 0, 0 };
    FIS_TYPE* fuzzyInput[fus_gcentrada] = { fuzzyInput0, fuzzyInput1, };
    FIS_TYPE fuzzyOutput0[] = { 0, 0, 0 };
    FIS_TYPE fuzzyOutput1[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyOutput[fus_gcsalida] = { fuzzyOutput0, fuzzyOutput1, };
    FIS_TYPE fuzzyRules[fus_gcreglas] = { 0 };
    FIS_TYPE fuzzyFires[fus_gcreglas] = { 0 };
    FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules, fuzzyFires };
    FIS_TYPE sW = 0;
    int i, j, r, o;
    for (i = 0; i < fus_gcentrada; ++i){
        for (j = 0; j < fus_gIMFCount[i]; ++j){
            fuzzyInput[i][j] =
                (fus_gMF[fus_gMFI[i][j]])(g_fusInput[i], fus_gMFICoeff[i][j]);}}
    int index = 0;
    for (r = 0; r < fus_gcreglas; ++r){
        if (fus_gRType[r] == 1){
            fuzzyFires[r] = FIS_MAX;
            for (i = 0; i < fus_gcentrada; ++i){
                index = fus_gRI[r][i];
                if (index > 0)
 fuzzyFires[r] = fus_minimo(fuzzyFires[r], fuzzyInput[i][index - 1]);  else if (index
< 0 fuzzyFires[r] = fus_minimo(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);     else
                    fuzzyFires[r] = fus_minimo(fuzzyFires[r], 1);}}else{
            fuzzyFires[r] = FIS_MIN;
            for (i = 0; i < fus_gcentrada; ++i){
                index = fus_gRI[r][i];
                if (index > 0)
                      fuzzyFires[r] = fus_maximo(fuzzyFires[r], fuzzyInput[i][index -
1]);
                else if (index < 0)
      fuzzyFires[r] = fus_maximo(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
                else
                    fuzzyFires[r] = fus_maximo(fuzzyFires[r], 0);            }}
        fuzzyFires[r] = fus_gRWeight[r] * fuzzyFires[r];
        sW += fuzzyFires[r];}
```

```
    if (sW == 0){
        for (o = 0; o < fus_gcsalida; ++o){
            g_fusOutput[o] = ((fus_gOMax[o] + fus_gOMin[o]) / 2);}} else{
        for (o = 0; o < fus_gcsalida; ++o){
            g_fusOutput[o] = fus_defuzz_centroid(fuzzyRuleSet, o);}}}
```

Anexo 4

```
decimal2radian=0.0174533;
radian2decimal=57.2958;
%ingreso de información
latitud= -0.3;
longitud= -78.5132273;
ho='Que hora quiere calcular(formato 24 horas)';
hour= input(ho);
mi='minutos';
minute= input(mi);
day='dia mes ';
daymonth= input(day);
month='mes ';
monthanio= input(month);
anio='que año es ';
anio= input(anio);
GTM='zona en la que se ubica';
gtmoff=input(GTM);
LatR=latitud*decimal2radian;
longiR=longitud*decimal2radian;
monthprev=monthanio-1;
dayanio=((monthprev)*30)+daymonth;
if monthprev==1
dayanio=dayanio+1;
end
if monthprev==2
dayanio=dayanio-1;
elseif monthprev==5
dayanio=dayanio+1;
elseif monthprev==6
dayanio=dayanio+1;
elseif monthprev==7
dayanio=dayanio+2;
elseif (monthprev==8)
dayanio=dayanio+3;
elseif (monthprev==9)
dayanio=dayanio+3;
elseif (monthprev==10)
dayanio=dayanio+3;
elseif (monthprev==11)
dayanio=dayanio+4;end
if mod(anio,4)==0
elseif mod(anio,100)==0
elseif mod(anio,400)==0
dayanio=dayanio+1;end
%ecuaciones solares
LSTM=(15*gtmoff)*decimal2radian;

118
BE=(0,986301)*(dayanio-81)*decimal2radian;
EcTi=(9.87*sin(2*BE))-(7.53*cos(BE))-(1.5*sin(BE));
CORRTIE=(4*(radian2decimal)*(longiR-LSTM))+EcTi;
Factor=minute/60;
HLOC=hour+Factor;
HSL=HLOC+(CORRTIE/60);
AngHo=(15*(HSL-12))*decimal2radian;
decli=asin(sin(23.45*decimal2radian)*sin(decimal2radian*(0.9863)*(dayanio-81)));
decliG=decli*radian2decimal;
elev=asin(sin(decli)*sin(LatR)+cos(decli)*cos(LatR)*cos(AngHo));
azi=acos(((sin(decli)*cos(LatR))-
(cos(AngHo)*cos(decli)*sin(LatR)))/cos(elev));
elevG=elev*radian2decimal;
aziG=azi*radian2decimal;
aziG=360-aziG;
S=12-(radian2decimal*(1/15*(acos(-tan(LatR)*tan(decli))))-CORRTIE/60);
SS=12+(radian2decimal*(1/15*(acos(-tan(LatR)*tan(decli))))-CORRTIE/60);
AngHo=AngHo*radian2decimal;
disp(dayanio)
disp(EcTi)
CORRTIE
HLOC
HSL
decliG
disp(elevG)
```

```
disp(aziG)
```

Anexo 5

```
double decimal2radian = 0,01745329;
double radian2decimal = 57.295779;
double lat = -0.3;
double long = -78.5132273;
double hour = 16.0;
double min = 0.0;
double daymonth = 25.0;
double me = 12.00;
double GMT = -5.00;
void setup()
{Serial.begin(9600);
double latiR = lat * decimal2radian;
double longR = long * decimal2radian;
//Ec
int monthprev=me-1;
double dayanio = ((monthprev)* 30) + (daymonth);
if(monthprev==1)dayanio=dayanio+1;
if(monthprev==2)dayanio=dayanio-1;
else if(monthprev==5||monthprev==6)dayanio=dayanio+1;
else if(monthprev==7)dayanio=dayanio+2;
else if(monthprev==8||monthprev==9)dayanio=dayanio+3;
else if(monthprev==10||monthprev==11)dayanio=dayanio+4;
double LSTM = (15*GMT)*decimal2radian;
double Be = (0.9863)*(dayanio - 81) * decimal2radian;
double EcT = (9.87 * sin(2*Be)) - (7.53 * cos(Be)) - (1.5 * sin(Be));
double CorrTie = (4 * (radian2decimal * (longR - LSTM))) + EcT;
double Hloc = hour + (min/60);
double HLS = Hloc + (CorrTie/60);
double AngH = (15 * (HLS-12))* decimal2radian;
double decli = asin(sin(0.4092)*sin(decimal2radian*(0.9863)*(dayanio - 81)));
double elev = asin(sin(decli)*sin(latiR) +
cos(decli)*cos(latiR)*cos(AngH));
double azi = acos((sin(decli)*cos(latiR) -
(cos(AngH)*cos(decli)*sin(latiR)))/ cos (elev));
//Angulos en grados
double elevG = elev*radian2decimal;
double aziG = azi*radian2decimal;
if (hour <=12)
{ aziG = aziG;}else
{ aziG = 360 - aziG;}
//Calculo de la duración del día
double scero = (-tan(latiR)*tan(decli));
double suno = (0.0666 * acos(scero)) * radian2decimal;
double sdos = suno-(CorrTie/60);
double aman = 12 - sdos; double anoch = 12 + sdos;
delay(1000);
Serial.print("Dias del anio: ");Serial.println(dayanio);
Serial.print("EcT: ");Serial.println(EcT);
//Serial.print("CorrTie: ");Serial.println(CorrTie);
```



```
//Serial.print("Declinacion: ");Serial.println(decli*radian2decimal);
//Serial.print("AngH: ");Serial.println(AngH*radian2decimal);
Serial.print("Elevacion: ");Serial.println(elevG);
Serial.print("Azimut: ");Serial.println(aziG);
//Serial.print("Amanecer: ");Serial.println(aman);
//Serial.print("Anochece: ");Serial.println(anoch);}
void loop(){}
```