Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Software Embebido

Adaptación de algoritmo de posicionamiento solar embebido con aplicación en energía solar

Tesis

Que como parte de los requisitos para obtener el grado de
Maestro en Software Embebido

Presenta:

Ing. Antonio Trejo Morales

Dirigido por:

Dr. Guillermo Ronquillo Lomelí

Santiago de Querétaro, Qro., México. Agosto, 2017

Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Software Embebido

Adaptación de algoritmo de posicionamiento solar embebido con aplicación en energía solar

Opción de titulación
Tesis

Que como parte de los requisitos para obtener el grado de
Maestro en Software Embebido

Presenta:
Ing. Antonio Trejo Morales

Dirigido por:
Dr. Guillermo Ronquillo Lomeli

Dr. Guillemo Ronquillo Lomeli Presidente	Firma
Dr. Leonardo Barriga Rodríguez Secretario	Firma
MSI. Gabriela Xicoténcatl Ramírez Vocal	Firma
MSI. Fausto Abraham Jacques García Suplente	Firma
MSI. José Alfredo Acuña García Suplente	Firma

M.I.S.D. Juan Salvador Hernández Valerio
Director de la Facultad

Dra. Ma. Guadalupe Flavia Loarca Piña
Directora de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Agosto, 2017

RESUMEN

Este trabajo presenta la adaptación de un algoritmo de posicionamiento solar embebido en una tarjeta Raspberry Pi que determina el cálculo de la posición solar a partir de las coordenadas geográficas y tiempo universal, para el posicionamiento de un mecanismo de seguidor solar en dos ejes, con aplicación en energía solar. Para esta investigación se seleccionó un seguidor solar con dos ejes de rotación: Azimutal y Elevación. Se seleccionó y adaptó el algoritmo de posición solar (SPA) astronómico, el cuál calcula los ángulos solares Azimutal y Elevación en un momento dado para una ubicación específica. Aplica únicamente para un intervalo de años que va desde el -2000 hasta 6000, con una incertidumbre de +/- 0.0003 grados. Se seleccionaron y configuraron los componentes específicos, tales como: un GPS, un RTC, un LCD, entre otros, requeridos por el sistema de software acorde a la aplicación. Posteriormente se realizó la configuración de las interfaces de comunicación entre la tarjeta Raspberry Pi y los componentes. También se realizó la programación del sistema de software que contiene el algoritmo SPA. Finalmente se ejecutaron pruebas de funcionamiento del sistema de software, con lo que se obtuvo un panorama general del rendimiento del algoritmo independientemente de las condiciones climáticas y época del año, demostrando que resulta eficiente en el cálculo de la posición del Sol con el SPA embebido en una tarjeta Raspberry Pi, permitiendo realizar el seguimiento de la trayectoria solar en cualquier lugar geográfico, de manera versátil y que puede ser útil para aplicaciones de investigación.

(**Palabras Clave:** seguidor solar, SPA, energía solar, Raspberry Pi, software embebido)

SUMMARY

This study presents the adaptation of a solar position algorithm embedded in a Raspberry Pi card which determines the calculation of the solar position using geographic coordinates and universal time for the positioning of a solar tracking mechanism with two axes with application in solar energy. Selected for this study was a solar tracking with two rotation axes: azimuth and elevation. An astronomical solar position algorithm (SPA) was selected and adapted which calculates the azimuth and elevation solar angles at a certain moment for a specific location. It is applied only for an interval of years that goes from -2000 to 6000 with uncertainties of +/- 0.0003 degrees. Specific components were selected and configured, such as; a GPS, a RTC and a LCD, among others, required by the software system in accordance with the application. After, a configuration of the communication interfaces between the Raspberry Pi card and the components was carried out. Also carried out was the programming of the software system that contains the SPA. Finally, functioning tests of the software system were done with which a general panorama of the yield of the algorithm was obtained, independent of climatic conditions and the time of year. This demonstrates that calculating the position of the sun with the SPA embedded in a Raspberry Pi card is efficient, thus making possible the following of the solar trajectory in any geographic location in a versatile manner and one that can be useful for research applications.

(**Key words:** solar tracking, SPA, solar energy, Raspberry Pi, embedded software)

AGRADECIMIENTOS

A mis padres, porque gracias a ellos hoy puedo ver alcanzada mi meta, porque siempre estuvieron impulsándome en los momentos más difíciles. Gracias por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida.

A mis hermanos Ángel, Armando y Alejandro, les agradezco por su apoyo moral y motivación, los quiero.

A mi esposa Herendida Navarro Cortina por tenerme paciencia en épocas malas, por compartir alegrías y nostalgias a lo largo de estos dos años, por acompañarme en mis desvelos y sobre todo en mis logros. Le agradezco por su apoyo, motivación y cariño, porque me ha sabido entender. Gracias por estar a mi lado.

A mis profesores por guiarme, por brindarme dedicación, esmero y sobre todo llenarme de valiosos conocimientos.

A mi asesor Guillermo Ronquillo Lomelí por brindarme la oportunidad de realizar este proyecto, por su confianza y consejos durante este proceso. También agradezco a mis amigos Irving Ramírez y Rodolfo Coria por dedicarme tiempo, brindándome un poco de sus conocimientos y experiencias para poder realizar un buen trabajo.

Agradezco a todas las personas del Centro de Ingeniería y Desarrollo Industrial (CIDESI), que pusieron un granito de arena para poder culminar este trabajo. Mil palabras no bastarían para agradecerles su apoyo, su comprensión y sus consejos en los momentos difíciles.

Agradezco el apoyo económico recibido por parte del Centro de Ingeniería y Desarrollo Industrial (CIDESI), de igual forma, agradezco al Centro Mexicano de Innovación en Energía Solar (CeMIE-Sol), en el marco de la Convocatoria 2013-02, del FONDO SECTORIAL CONACYT - SENER - SUSTENTABILIDAD ENERGÉTICA, dentro del Proyecto Estratégico: P13 “Laboratorios de pruebas para baja y media temperatura, laboratorio para el diseño e integración de sistemas termo solares asistido por computadora”, por medio del cual fue posible desarrollar el presente trabajo de Tesis.

A todos, gracias.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS iv

ÍNDICE DE TABLAS vi

1. INTRODUCCIÓN 1

1.1. Antecedentes 1

1.2. Alcance del estudio 2

1.2.1. Objetivo general 2

1.2.2. Objetivos específicos 2

1.2.3. Preguntas de investigación 2

1.2.4. Justificación 3

1.2.5. Formulación de hipótesis 3

1.2.6. Variable dependiente 3

1.2.7. Variable independiente 4

1.2.8. Variables circunstanciales 4

1.2.9. Alcances y Limitaciones 4

1.2.10. Organización 5

2. REVISIÓN DE LITERATURA 6

2.1. Algoritmos de cálculo de la posición del Sol 6

2.2. Sistemas embebidos de seguimiento solar 10

3. METODOLOGÍA 16

3.1. Energías alternativas 16

3.2. Energía solar 17

3.3. Radiación solar 18

3.4. Movimiento aparente del Sol respecto al planeta Tierra 19

3.4.1. Husos horarios 21
3.4.2. La trayectoria solar 22
3.5. Seguidores solares 24
3.5.1. Tipos de seguimiento del Sol 25
3.5.2. Métodos de seguimiento 25
3.6. Algoritmo de seguimiento solar seleccionado 26
3.7. Tarjeta de evaluación para el sistema de control 27
3.7.1. Raspberry Pi 28
3.7.2. Protocolos de comunicación 29
3.7.3. Características de las tarjetas de evaluación 32
3.8. Sistema embebido seleccionado 34
3.9. Prototipo seleccionado para seguimiento solar 34

4. DESARROLLO E IMPLEMENTACIÓN DEL ALGORITMO 36

4.1. Metodología técnica 36
4.1.1. Análisis funcional del sistema 36
4.1.2. Diseño de la arquitectura del sistema 42
4.1.3. Especificación de componentes 43
4.1.4. Configuración de las comunicaciones 45
4.1.5. Programación del sistema embebido 55
4.1.6. Pruebas y puesta en marcha del sistema 60

5. RESULTADOS 67

5.1. Discusión 76
5.2. Aplicaciones y uso del proyecto 76

6. CONCLUSIONES 77

6.1. Trabajos futuros 78

REFERENCIAS **79**

A. APÉNDICES **85**

A.1. Código fuente del sistema embebido 85

A.2. Archivo de cabecera SPA (spa.h) 99

A.3. Código fuente de funciones SPA (spa.c) 103

A.4. Resultados de la prueba del software 125

ÍNDICE DE FIGURAS

3.1. Energías alternativas. 16
3.2. Autoconsumo con energía solar fotovoltaica. 17
3.3. Radiación solar global o total. 19
3.4. Movimiento aparente del Sol. 20
3.5. Husos Horarios. 21
3.6. Trayectoria aparente del Sol. 22
3.7. Modelo geométrico de la trayectoria solar. 23
3.8. Seguidor Solar. 24
3.9. Tarjeta Raspberry Pi 2 - Modelo B. 29
3.10. Protocolo I2C. 30
3.11. Protocolo RS-232. 31
3.12. Sistema embebido seleccionado. 34
3.13. Prototipo seleccionado para seguimiento solar. 35

4.1. Diagrama de secuencia. 37
4.2. Diagrama a bloques. 38
4.3. Arquitectura general del sistema. 42
4.4. Diagrama de comunicación Raspberry Pi, LCD y RTC. 44
4.5. Diagrama de comunicación Raspberry Pi con GPS. 44
4.6. Diagrama de comunicación Raspberry Pi con controlador de seguidor solar. . 45
4.7. Edición del archivo modules. 46
4.8. Edición del archivo raspi-blacklist.conf. 47
4.9. Dispositivos conectados en el bus I2C. 47
4.10. Edición del archivo inittab. 49
4.11. Visualización de tiempo con RTC. 51

4.12. Adaptación de algoritmo de posicionamiento solar embebido. 54
4.13. Validación de coordenadas GPS del sitio de pruebas. 61
4.14. Validación de los módulos RTC y LCD. 62
4.15. Validación del módulo controlador de los dos ejes del seguidor solar. 63
4.16. Prueba de integración y ejecución del sistema de software. 64
4.17. Vista general de la prueba del sistema completo. 65
4.18. Ángulos Azimutales arrojados en las pruebas del sistema de software. 66
4.19. Ángulos de Elevación arrojados en las pruebas del sistema de software. . . . 66

5.1. Sistema de referencia del sensor Solar MEMS ISS-T60. 67
5.2. Prueba de exactitud de los ángulos arrojados por el sistema de software. . . . 70
5.3. Tendencia de los ángulos dados por el sensor Solar MEMS ISS-T60. 70
5.4. Resultados de ángulos Azimutales dados por el algoritmo SPA. 73
5.5. Tendencia del error en el ángulo Azimutal dado por el algoritmo SPA. 73
5.6. Resultados de ángulos de Elevación dados por el algoritmo SPA. 74
5.7. Tendencia del error en el ángulo de Elevación dado por el algoritmo SPA. . . 74

ÍNDICE DE TABLAS

2.1. Parámetros solares calculados por los diferentes algoritmos. 9

3.1. Características de las tarjetas de evaluación. 33

4.1. Lista de componentes. 40

5.1. Resultados de ángulos de SolarTracking con respecto a Solar MEMS ISS-T60. 68

5.2. Comparación de ángulos de posición solar. 71

A.1. Resultados de los ángulos de posición solar calculados por el algoritmo SPA. 125

1. INTRODUCCIÓN

Los sistemas de seguimiento solar tienen un papel importante en el desarrollo de aplicaciones de energía solar, especialmente para los sistemas de alta concentración solar que convierten directamente la energía solar en energía térmica o eléctrica (Luque-Heredia et al., 2007). En las últimas dos décadas, varios tipos de mecanismos de seguimiento solar se han propuesto para mejorar el rendimiento y aprovechamiento de la energía solar en colectores solares. Aunque el grado de precisión que se requiere depende de las características específicas del sistema de concentración solar que se analiza, en general, cuanto mayor es la concentración, se necesitará más alta precisión del sistema de seguimiento (Blanco-Muriel et al., 2001). Para conseguir una buena precisión de seguimiento, los sistemas de seguimiento solar normalmente emplean sensores para la realimentación de la señal de error en el sistema de control, con el fin de recibir de forma continua la máxima radiación solar en el receptor.

1.1. Antecedentes

Los dos tipos comunes de sensores utilizados para este fin son sensores de lazo cerrado y sensores de lazo abierto. Los diversos problemas ambientales que tiene actualmente el ecosistema, son la contaminación, la escasez de recursos no renovables, la gran demanda energética, el calentamiento climático, o simplemente aquellos lugares que no poseen suministro eléctrico, ya sea, por razones geográficas o por elevados costos de instalación, situación que motiva a miles de personas y entidades empresariales a la búsqueda de alternativas energéticas limpias y eficientes, tales como, la energía solar. Para captar la radiación solar, en la mayoría se utilizan paneles solares fijos, provocando que el desempeño en cuanto a eficiencia se vean afectados por diversos factores: principalmente la orientación del sol que varía según el día del año por el ángulo que presente el panel con respecto al Sol, este ángulo es causado por el movimiento de rotación de la tierra y que a lo largo del año, éste ángulo también es afectado por el movimiento de traslación. También influye la hora, las condiciones atmosféricas y la Latitud del lugar de posición del panel solar. Por estos cambios en la orientación solar en paneles fijos, la energía solar no se aprovecha al máximo, por lo tanto, se debe controlar el valor de estos ángulos para mejorar la captación de radiación solar.

1.2. Alcance del estudio

1.2.1. Objetivo general

Adaptar algoritmo de posicionamiento solar embebido que determina el cálculo de la posición solar a partir de las coordenadas geográficas y tiempo universal, para el posicionamiento de un mecanismo de seguidor solar en dos ejes, con aplicación en energía solar, que permita realizar el seguimiento de la trayectoria solar en cualquier lugar geográfico, de manera que tenga la suficiente versatilidad para aplicaciones de investigación.

1.2.2. Objetivos específicos

1. Realizar el análisis del estado del arte en sistemas de seguimiento solar.
2. Seleccionar el hardware adecuado para implementar el software embebido referente al algoritmo de posicionamiento solar.
3. Diseñar y construir un prototipo electrónico funcional que satisfaga los requerimientos para la ejecución del algoritmo de posicionamiento solar, como apoyo para la implementación del software creado.
4. Seleccionar un algoritmo para el cálculo y seguimiento de la trayectoria solar, teniendo en cuenta los algoritmos de seguimiento solar existentes.
5. Programar el sistema de software embebido que contenga el algoritmo escrito en el lenguaje de programación C, que sea capaz de calcular la posición solar.
6. Evaluar el desempeño del sistema embebido de seguimiento solar.

1.2.3. Preguntas de investigación

1. ¿Cuál será el lenguaje de programación más adecuado para desarrollar el software embebido?
2. ¿Cómo funciona un seguidor solar? ¿Cuáles son las partes que lo componen?

3. ¿Qué tipo de tarjetas controladoras existen actualmente?

4. ¿Cómo es el movimiento aparente del Sol respecto a la tierra?

5. ¿Cuáles son los tipos de seguimiento solar existentes en la actualidad?

6. ¿Qué métodos de seguimiento hay en la actualidad?

1.2.4. Justificación

Con el propósito de obtener un máximo rendimiento de captación de radiación solar en sistemas solares; los algoritmos de seguimiento solar son usados en las industrias para generar energía térmica y eléctrica. Por un lado, se utiliza el seguimiento solar en colectores solares de canal parabólico (un eje) y disco parabólico (dos ejes), para producir energía térmica solar a diferentes temperaturas. Por otro lado, el seguimiento solar es usado en celdas fotovoltaicas de dos ejes para producir energía eléctrica. Algunos sistemas de seguimiento solar usan sensores para seguir la trayectoria del Sol, sin embargo, estos fallan bajo condiciones atmosféricas adversas, y como consecuencia disminuye la eficiencia de los sistemas solares. Si se quiere mejorar la eficiencia de captación solar se deben controlar los valores de los ángulos de la posición del Sol, mediante un algoritmo de posicionamiento solar embebido, que permita conocer la posición de Sol en tiempo real, para ser usado en sistemas de seguimiento solar cuando las señales de los sensores no estén disponibles.

1.2.5. Formulación de hipótesis

El seguimiento de la trayectoria solar en cualquier lugar geográfico se puede determinar en tiempo real en un sistema embebido mediante el uso de las coordenadas geográficas y tiempo universal, para el posicionamiento de un mecanismo de seguidor solar en dos ejes.

1.2.6. Variable dependiente

1. El tiempo universal y la posición geográfica determinan la posición solar.

2. A mayor perpendicularidad a los rayos solares mayor es la radiación solar captada.

3. Con un dispositivo de seguimiento del sol mayor es el rendimiento de los paneles solares.

1.2.7. Variable independiente

1. Las condiciones climatológicas y errores de los mecanismos de seguimiento causan desviaciones en el posicionamiento solar.

2. La mala orientación de los paneles solares provoca baja captación solar y menor eficiencia.

1.2.8. Variables circunstanciales

1. Las condiciones atmosféricas son variables circunstanciales por los problemas ecológicos que han dañado nuestro ecosistema, provocando que existan diversos fenómenos que impidan la visibilidad y detección de los rayos solares, trayendo como consecuencia que el seguidor no pueda realizar el seguimiento.

1.2.9. Alcances y Limitaciones

Alcances

El sistema de software embebido será capaz de calcular continuamente la trayectoria del sol, a su vez proporcionar dos ángulos para posicionar y apuntar al Sol el plano horizontal superior de un mecanismo de seguidor solar, esto con el objetivo de captar la mayor cantidad de energía solar, y así, poder ser aprovechada por aplicaciones que requieren de esta energía, por ejemplo, celdas solares, colectores solares, entre otros.

El sistema de software embebido en una tarjeta podrá desplegar información referente a la trayectoria (ángulos de ubicación del Sol) mediante un LCD que será incluido en el prototipo electrónico. Será completamente automático a lo largo del día, intercalando un tiempo programado para el reposicionamiento a un estado inicial por la noche.

Limitaciones

El resultado del algoritmo de posicionamiento solar que arrojará el sistema embebido, no se evaluará sobre un sistema de posicionamiento real, por los altos costos de estos sistemas, siendo así, evaluado en un mecanismo prototipo ya existente y que no está considerado dentro de este trabajo, para simular el seguimiento de la trayectoria solar.

El análisis de los resultados se realizará evaluando los errores de la posición calculada respecto a la posición medida, la validación del sistema en aplicaciones reales no es parte del proyecto, ya que el tiempo no permitirá un análisis de esta naturaleza.

1.2.10. Organización

Para la adaptación de un algoritmo de posicionamiento solar embebido en una tarjeta Raspberry Pi que controla el posicionamiento de un seguidor solar, fue necesario definir los antecedentes, posteriormente los alcances de estudio presentado los datos generales del proyecto, tales como: los objetivos, preguntas de investigación, justificación, hipótesis, entre otros. En el capítulo 2 varios algoritmos para el cálculo de la posición del Sol fueron analizados y se determinaron los más adecuados para la aplicación a partir de una revisión de la literatura, estos métodos fueron estudiados y desarrollados en el capítulo 3. En el capítulo 4 se aplica una metodología técnica para determinar la construcción del software y el diseño electrónico del prototipo de seguidor solar. Posteriormente se exponen en el capítulo 5 los resultados esperados, una breve discusión, las aplicaciones y uso del proyecto. Finalmente en el capítulo 6 se presentan las conclusiones y los posibles trabajos futuros.

2. REVISIÓN DE LITERATURA

2.1. Algoritmos de cálculo de la posición del Sol

Hoy en día existe una gran cantidad de artículos referentes al cálculo de la posición del Sol, en base a ello, podemos decir que se clasifican en dos categorías:

1. La primer categoría se basa en formulas relativamente simples y algoritmos que, dado un día del año, estima la posición del Sol, con parámetros como, la declinación solar o la ecuación del tiempo (Cooper, 1969; Lamm, 1981; Spencer, 1971; Swift, 1976).

2. La segunda categoría se compone de algoritmos más complejos (Michalsky, 1988; Pitman and Vant-Hull, 1978; Walraven, 1978), que dada la ubicación exacta de observación, calcula la posición del Sol en eclíptica (longitud eclíptica, posición oblicua), celeste (declinación, ascensión) o coordenadas horizontales locales (distancia del Zenith y Azimut solar).

Por lo general, un sistema de rastreo solar tiene que conocer las coordenadas horizontales locales del Sol en la ubicación del sistema (especificado por su Longitud y Latitud geográfica) en cualquier instante dado del tiempo (especificado por la fecha y el tiempo universal). Desde este punto de vista, ninguno de los algoritmos relatados es completo. La una o la otra, su entrada es más especializada (es decir, los valores de la entrada al algoritmo no son la Longitud y Latitud geográfica, fecha y tiempo universal) o en su salida el algoritmo no proporciona las coordenadas del Sol horizontales locales, o ambos.

En el grupo de algoritmos y fórmulas usadas para calcular las coordenadas celestes del Sol del número del día del año, los más importantes y el más extensamente citado en la literatura solar son las fórmulas de Spencer (Spencer, 1971). En el artículo de Spencer, así como en Iqbal (Iqbal, 1983), se dice que estas fórmulas tienen un error máximo de 3 minutos del arco para la declinación y aproximadamente 35 segundos para la ecuación del tiempo. Sin embargo, no parece que esta reclamación es correcta. Propio Spencer indica en otra parte en su artículo que el mero uso del número del día del año como un número entero introduce un error en la declinación solar que puede variar aproximadamente 0.5 grados en los equinoccios y menos de 1 minuto del arco en los solsticios. Incluso considerando que el día del año

varía continuamente con el tiempo, cualquier análisis de la declinación solar proporcionada por la fórmula de Spencer muestra claramente que el error máximum puede ser tan grande como 0.28 grados. Michalsky (Michalsky, 1988) ha mencionado ya esto. Otros autores como Cooper (Cooper, 1969) han proporcionado ecuaciones para calcular la declinación solar que son más simples que aquellos de Spencer, pero menos exactos también.

Pitman y Vant-Hull (Pitman and Vant-Hull, 1978) fueron de los primeros en indicar que los algoritmos de alta precisión fueron necesarios para rastrear sistemas o comparar las predicciones de modelos irradiantes directos con lecturas de pirómetros. Basado en las ecuaciones del calendario astronómico y el calendario astronómico americano y Almanaque Náutico de 1961, presentó fórmulas para el cálculo de las coordenadas del Sol eclípticas, su declinación y la ecuación del tiempo. También hablan de muchos de los fenómenos que afectan la posición del Sol y estiman los errores resultando cuando estos fenómenos son desatendidos. Aunque no proporcionen fórmulas para corregir para la refracción, ponen límites los errores asociados con la ausencia de tal corrección. Se estima que éstos son aproximadamente 5 minutos del arco para más de 10 grados de elevación y aproximadamente 4 minutos del arco para más de 15 grados. Afirman que su programa, llamado *SUNLOC* basado en sus fórmulas, en su modo más exacto, puede calcular las coordenadas horizontales verdaderas del Sol (es decir, las coordenadas del Sol horizontales desatendiendo los efectos de la refracción atmosférica) con un error máximo a la orden de 41 segundos del arco. Lamentablemente, no proporcionan el código de programación SUNLOC en su artículo.

En el mismo año, 1978, Walraven presentó un algoritmo completo para calcular el horizonte local de las coordenadas del Sol (Walraven, 1978). Este algoritmo, al igual que el algoritmo de Pitman y Vant-Hull, también se basa en las ecuaciones de las efemérides Astronómicas. En su documento, Walraven afirma que se puede usar el algoritmo para estimar la posición del Sol con una precisión de 36 segundos de arco. La publicación del artículo de Walraven fue seguido por un Erratum (Walraven, 1979) y varias notas técnicas y cartas del editor (Archer, 1980; Ilyas, 1983, 1984; Muir, 1983; Pascoe, 1984; Spencer, 1989; Wilkinson, 1981, 1983). La mayoría de ellas se refiere a la aplicación del algoritmo desarrollado en FORTRAN presentado en el documento original. Curiosamente, un error significativo asociado con la implementación del algoritmo de Walraven parece haber permanecido oculto. Tiene que ver con el cálculo de una de las entradas, el número de día. Este número es el número de

día del año comenzando con 1 para el primero de enero, excepto en los años bisiestos cuando 1 debería ser restado del número del día si es antes del primero de Marzo. Sin embargo, el algoritmo no funciona correctamente durante varios días antes del primero de marzo en los años bisiestos si la mencionada sustracción se lleva a cabo. Entre los errores que se han informado, el más importante es el hecho de que el algoritmo no funciona adecuadamente para el Hemisferio Sur (Spencer, 1989). Para facilitar el uso del algoritmo de Walraven, Archer (Archer, 1980) propone una subrutina para calcular el número de día, en el año, mes y día como entrada. Esta subrutina es válida para el período 1901-2099 y no incluye la sustracción. El artículo de Archer es el primero de los casos más examinados ya que ofrece una ecuación para calcular la corrección de la refracción atmosférica en la determinación de la elevación.

En el año 1981, Lamm presentó una expresión para calcular la ecuación de tiempo, alegando un error absoluto máximo de 3.6 segundos de tiempo (Lamm, 1981). Otros autores como Holanda y Mayer en el año de 1988 han implementado un programa para calcular la posición del Sol mediante la ecuación propuesta por Lamm y compara sus resultados con los obtenidos mediante el algoritmo de Walraven. De esta comparación se concluye que los errores de unos 5 grados en la determinación de la elevación solar podría ser el resultado de la utilización de la expresión de Lamm.

Diez años después de la publicación del algoritmo de Walraven, Michalsky (Michalsky, 1988) ha publicado otro algoritmo basado también en ecuaciones simplificadas de almanac astronómico. A pesar de que el algoritmo utiliza menos operaciones para calcular la posición del Sol y ser más preciso que el de Walraven, tiene la misma desventaja, en su forma original, no funciona adecuadamente para el hemisferio sur. A diferencia de Walraven, Michalsky incluye una relación para estimar el efecto de la refracción en la aparente elevación del Sol. Este modelo proporciona la refracción aparente de elevación del Sol en función de su elevación verdadera.

Kambezidis y Papanikolaou en el año de 1990 sostienen la necesidad de adoptar este modelo en lugar del propuesto por Archer en 1980, ya que el modelo de Archer causa problemas en la estimación de la hora de amanecer y atardecer (Kambezidis and Papanikolaou, 1990). Lo que parece implícitamente abordar es el hecho de que el modelo de Archer no es un modelo definido para un intervalo corto de ángulos de elevación real negativo en

el intervalo [22,7268 y 22,6788]. Esta gama de valores, es hasta ahora el mínimo para que cualquier modelo de refracción funcione, y además no excluir el uso del modelo de refracción de Archer para cualquier aplicación práctica.

Otros países han propuesto modelos más sofisticados de refracción exclusivos que toman como entrada las variables meteorológicas, tales como la temperatura y la presión, además de la verdadera elevación del Sol (Camblor-Ordiz, 1995; Meeus, 1998; Duffett-Smith, 1992). En la siguiente Tabla 2.1, se muestran las variables calculadas por cada uno de los algoritmos analizados.

Tabla 2.1: Parámetros solares calculados por los diferentes algoritmos.

Algoritmo	**Declinación**	**Ascensión recta**	**Ecuación del tiempo**	**Azimutal**	**Elevación**
Cooper, 1969	X	-	X	-	-
Spencer, 1971	X	-	X	-	-
Swift, 1976	X	X	-	X	X
Pitman y Vant-Hull, 1978	X	-	X	-	-
Walraven, 1978	X	X	-	X	X
Lamm, 1981	-	-	X	-	-
Michalsky, 1988	X	X	X	X	X

Otra posibilidad para seguir el Sol es utilizar la tabla basada en el almanac astronómico o la aplicación de la computadora interactiva multi-anual almanaque (mica) disponible desde el Observatorio Naval de Estados Unidos. Sin embargo, tal sistema sería difícil de manejar por los microcontroladores utilizados en sistemas de seguimiento.

2.2. Sistemas embebidos de seguimiento solar

Como resultado de innumerables investigaciones y a su vez experimentación y pruebas, se han derivado diversos desarrollos para abordar el problema de poder incrementar el aprovechamiento de la radiación solar. Convirtiendo a los sistemas de seguimiento solar en sistemas completamente híbridos debido a que se componen de diversos mecanismos en conjunto con peculiares métodos de programación y control.

En 2014, fue presentado por Rizvi et al un método para calcular la posición del Sol que podría ser útil en el seguimiento del Sol sin sensores. El algoritmo propuesto es una compilación de diferentes ecuaciones que calculan la posición solar, arrojando los ángulos de Azimutal y Elevación en coordenadas horizontales (Rizvi et al., 2014). Por eso, este algoritmo puede ser utilizado fácilmente para seguimiento en sistemas de baja concentración que según (Blanco-Muriel et al., 2001) de muy alta precisión no se requiere. La exactitud del algoritmo ha sido probada y verificada con el algoritmo de posición del Sol NREL y la diferencia de medias fue encontrada para ser 0.22 grados en el ángulo Azimutal y 0.4 grados en el ángulo de Elevación. El algoritmo toma hora local, fecha y lugar como parámetros de entrada y calcula la posición del Sol en el sistema de coordenadas horizontal. El algoritmo fue ***implementado en una computadora*** con el propósito de trasmitir la posición de seguimiento a través de un bus de red de área de control (CAN por sus siglas en inglés) al controlador del seguidor.

En ese mismo año, Sidek et al aborda un sistema de seguimiento solar a dos ejes portátil basado en la ecuación astronómica, el sistema de seguimiento utiliza el módulo GPS y un sensor brújula digital para determinar la ubicación y la información de partida del sistema respectivamente (Sidek et al., 2014). Por otra parte, el sistema de seguimiento está basado en un ***microcontrolador PIC18f4680, que tiene embebido un controlador PID*** para elevar el panel fotovoltaico colocándole la señal de realimentación del encoder absoluto. Además, cuenta con dos microcontroladores PIC18f4431 que controlan los ejes Azimutal y Elevación del seguidor solar. De igual manera, el sistema de posicionamiento PID embebido mejora el seguimiento de precisión en la localización de los ángulos de Elevación y Azimutal hasta ± 0.2 grados de ángulo.

En 2013, Haizai et al presenta una a aplicación de un sistema de seguimiento solar, usando un servo sistema ***dSPACE DS1104 R&D como tarjeta controladora*** del seguimiento,

está diseñado para mejorar la eficiencia de captación solar. dSPACE sigue al Sol con un método que combina el seguimiento de la trayectoria del Sol y seguimiento electróptico. Adicionalmente, con el fin de reducir la cantidad de cálculos, se propone un T-S fuzzy system basado en el seguimiento de la trayectoria de Sol (Peng et al., 2013). Cuando el día esta soleado, el sistema elige el seguimiento electróptico para obtener alta sensibilidad, de lo contrario, cuando el día esta lluvioso o nublado, el T-S fuzzy system es elegido para el seguimiento de la trayectoria del Sol y así hacer el sistema más confiable.

Un año anterior, es decir en el 2012, Engin et al desarrollo un algoritmo de control que mejora el rendimiento y la fiabilidad del seguidor solar de dos ejes. Para lograr este objetivo, su estudio se concentra en la optimización de la ***tarjeta microcontroladora LM3S811***, unidad de hardware y el software, la cual deduce la posición correcta del Sol a través de un GPS (Engin and Engin, 2012). La estrategia de control que propone, consiste en una combinación entre; una estrategia de seguimiento mediante un ciclo abierto y una estrategia con un ciclo cerrado. La tarjeta microcontroladora LM3S811 calcula correctamente los ángulos de Elevación y Azimutal de la posición del Sol. La utilización de un motor DC y la activación de un actuador lineal de DC reducen el error en la localización de los ángulos de Azimutal y Elevación, siendo de 0.1°.

En primer lugar, un sensor de bucle cerrado, tal como una cámara CCD (dispositivo de carga acoplada) o foto-detector, se utiliza para detectar la posición de la imagen solar sobre el receptor y una señal de retroalimentación se envía al controlador si la imagen solar se mueve lejos del receptor. Los sistemas de seguimiento solar que emplean sensores en lazo cerrado se conocen como seguidores solares de circuito cerrado. En los últimos 20 años, el enfoque de seguimiento de circuito cerrado ha sido utilizado tradicionalmente en la actividad de seguimiento del Sol (Kalogirou, 1996; Berenguel et al., 2004; Lee et al., 2006; Arbab et al., 2009). Por ejemplo, Kribus et al diseñó un controlador de circuito cerrado para heliostatos, que mejoraron el error de la imagen solar hasta el 0,1 mrad, con la ayuda de cuatro cámaras CCD instalados en el objetivo (Kribus et al., 2004). Sin embargo, este método es bastante caro y complicado, ya que requiere cuatro cámaras CCD y cuatro radiómetros para ser colocado sobre el objetivo. Posteriormente, las imágenes captadas por las cámaras solares CCD deben ser analizadas por un procesador para generar la retroalimentación para el controlador de corrección y corregir los errores de seguimiento.

En 2007, Rubio et al fabricó y evaluó una nueva estrategia de control para un seguidor solar fotovoltaico (PV) que opera en dos modos de seguimiento, el modo de seguimiento normal y modo de búsqueda. El modo rastreo normal combina un modo de seguimiento de lazo abierto que se basa en modelos de seguimiento solar y un modo de seguimiento en lazo cerrado que corresponde al controlador electro-óptico para obtener un error de seguimiento solar menor que el valor límite especificado y suficiente radiación solar para producir energía eléctrica. El modo de búsqueda se iniciará cuando el error de seguimiento solar es grande o no se produce energía eléctrica. El seguidor solar se moverá de acuerdo con un patrón en espiral cuadrada en el plano de Elevación y Azimutal para detectar la posición del Sol hasta que el error de seguimiento sea lo suficientemente pequeño (Rubio et al., 2007).

En 2006, fue presentado por Luque-Heredia et al un sistema de monitoreo de error de seguimiento solar que utiliza un sensor optoelectrónico monolítico. De acuerdo con los resultados de este estudio, el sistema de monitoreo alcanzó una precisión del seguimiento de 0.1°. Sin embargo, el criterio es que este sistema de seguimiento necesita tener todos los días con cielo despejado para operar, debido a que la incidencia de la luz tiene que estar por encima de un cierto umbral para garantizar que se cumpla una resolución mínima requerida (Luque-Heredia et al., 2006). Ese mismo año, Aiuchi et al desarrolló un helióstatos con un sistema de control con montura ecuatorial y un foto-sensor con circuito cerrado. Los resultados experimentales mostraron que el error del dispositivo de seguimiento del helióstato se estimó en 2 mrad durante buen clima (Aiuchi et al., 2006). No obstante, este método de seguimiento sólo se puede utilizar con seguidores solares con una configuración de montaje ecuatorial que no es una estructura mecánica del seguidor común, es complicado debido a que el centro de gravedad para el colector solar está lejos del pedestal. Además, Chen et al presentó estudios de sensores solares digitales y analógicos basado en escala óptica graduada y medición con el principio de compensación óptica no lineal respectivamente. Los sensores de Sol digital y analógico propuestos tienen una precisión de 0.02° y 0.2° correspondiente al campo de visión de ± 64° y ± 62°, respectivamente (Chen et al., 2006; Chen and Feng, 2007). La principal desventaja de estos sensores es que el campo de visión, que está en el intervalo de aproximadamente ± 64° para ambos sentidos de Elevación y Azimutal, es bastante pequeña en comparación para el rango dinámico de movimiento del seguidor solar en la práctica, que es de aproximadamente ± 70° y ± 140° para la Elevación y Azimutal respectivamente.

Aunque sistemas de seguimiento solar en lazo cerrado pueden producir seguimiento con mucha mejor precisión, este tipo de sistema perderá su señal de retroalimentación cuando el sensor está sombreado o cuando el Sol está bloqueado por las nubes. Como método alternativo para superar la limitación de seguidores solares en circuito cerrado, se introdujeron seguidores solares de lazo abierto mediante el uso de sensores de lazo abierto que no requieren ninguna imagen solar como retroalimentación. El sensor de lazo abierto se asegurará de que el colector solar se coloca en ángulos pre-calculados, que se obtienen a partir de una fórmula o algoritmo especial de acuerdo con fecha, hora e información geográfica.

En 2004, Abdallah et al diseño un sistema de seguimiento solar de dos ejes, que es operado por un sistema de control en lazo abierto. Se utilizó un controlador lógico programable (PLC) para calcular el vector solar, para controlar el seguidor solar y para que siguiera la trayectoria del Sol (Abdallah and Nijmeh, 2004). Además, Shanmugam et al presentó un programa de computadora escrito en Visual Basic que es capaz de determinar la posición del Sol y por lo tanto conducir un concentrador de plato parabólico (PDS) a lo largo del eje Este-Oeste o el eje Norte-Sur para recibir la máxima radiación solar (Shanmugam and Christraj, 2005).

En general, ambos enfoques de seguimiento solar mencionados anteriormente tienen ventajas y desventajas, por lo que se han desarrollado algunos sistemas de seguimiento solar híbridos incluyendo sensores tanto de lazo abierto y como de lazo cerrado. A principios del siglo XXI, Nuwayhid et al adoptó ambos esquemas de seguimiento en lazo abierto y lazo cerrado en un concentrador parabólico unido a un sistema de rastreo polar:

1. En el esquema de circuito abierto, un equipo actúa como controlador para el cálculo de dos ángulos de rotación, es decir, ángulo de declinación solar y ángulo de hora, para conducir el concentrador a lo largo de la declinación y ejes polares.

2. En el esquema de circuito cerrado, nueve foto resistores (LDR) están colocados en una matriz de una forma circular ïris"para facilitar seguimiento solar con un alto grado de precisión (Nuwayhid et al., 2001).

En ese mismo año, Luque-Heredia et al propuso un nuevo algoritmo PI basado en seguimiento solar híbrido para un sistema de concentración fotovoltaica. En su diseño, el

sistema puede actuar tanto en el modo de lazo abierto y lazo cerrado. Un modelo matemático que implica una función del tiempo y coordenadas geográficas, así como un conjunto de perturbaciones que proporcionan una pre alimentación en lazo abierto para la estimación de la posición del Sol. Para determinar la posición del Sol con alta precisión, se introdujo un circuito de retroalimentación de acuerdo a la rutina de corrección del error que se deriva de la estimación del error de las ecuaciones de Sol que son causados por las perturbaciones externas en la etapa actual en base a su historial de trayectoria (Luque-Heredia et al., 2004).

Es un hecho que el requisito de precisión en el seguimiento es muy dependiente de la formulación y aplicación del seguidor solar. En este caso, cuanto mayor sea la distancia entre el concentrador solar y el receptor, será requerida mayor precisión del seguimiento esto es porque la imagen solar se vuelve más sensible al movimiento del concentrador solar. Como resultado, un helióstato o seguidor solar fuera del eje normalmente requiere una precisión muy superior en comparación con los seguidores solares en un eje, debido a que la distancia entre el helióstato y el objetivo es normalmente más larga, especialmente para una configuración del sistema de receptor central.

En este contexto, una precisión de rastreo en el intervalo de unos pocos miliradianes (mrad) es suficiente para el control del seguidor solar sobre ejes para mantener un buen desempeño cuando la luz solar altamente concentrada está implicada (Chong et al., 2010). A pesar de tener muchos métodos de seguimiento solar sobre ejes, los diseños disponibles para lograr una buena precisión de seguimiento, pocos mrad, son complicados y caros. Vale la pena señalar que los sistemas seguidores solares convencionales sobre ejes suelen adoptar dos configuraciones comunes, que son la Elevación y Azimutal, y el movimiento de inclinación (rastreo polar), limitado por las fórmulas matemáticas básicas disponibles del sistema de seguimiento solar. *Para el sistema de seguimiento de Elevación y Azimutal, los ejes de seguimiento solar deben estar estrictamente alineados con el cenit y al norte verdadero. El movimiento de inclinación en un sistema de seguimiento, los ejes de seguimiento solar deben estar estrictamente alineados con el valor de Latitud y al norte verdadero.*

La principal causa de los errores de seguimiento solar es la forma en que se hace la alineación antes mencionada y cualquier defecto en la instalación o fabricación dará lugar a una baja precisión de seguimiento. De acuerdo a lo anterior para el sistema de seguimiento de Elevación y Azimutal, una desalineación de eje Azimutal con relación al eje cenit de

0.4° puede causar error de seguimiento que va desde 6.45 hasta 6.52 mrad (Chong and Wong, 2009). En la práctica, la mayoría de las plantas de energía solar en todo el mundo utilizan una gran superficie de colectores solares para ahorrar en costos de fabricación, lo que ha ocasionado indirectamente, que el trabajo de alineación de los ejes de seguimiento solar sea mucho más difícil. En este caso, la alineación de los ejes de seguimiento implica una gran cantidad de trabajo pesado, trabajos mecánicos y civiles, debido a la exigencia de los ejes gruesos necesarios para soportar el movimiento de grandes colectores solares, que normalmente tiene un área total de recolección en el intervalo de varias decenas de metros cuadrados. En condiciones difíciles, una alineación muy precisa es realmente un gran desafío para el fabricante, ya que un ligero desajuste se traducirá en importantes errores de seguimiento solar. Para superar este problema, se propone usar una fórmula general para seguimiento solar en ejes para permitir que el seguidor solar en cualquiera de los dos ejes dé seguimiento orientado arbitrariamente (Chong and Wong, 2009).

La integración de la fórmula general en el algoritmo de seguimiento solar se da con el objetivo de poder seguir al Sol con precisión y de manera rentable, incluso si hay alguna desalineación en la configuración del movimiento Azimutal ideal o del movimiento de inclinación. En nuevos sistema de seguimiento, cualquier desalineación o defecto puede rectificarse sin necesidad de ninguna modificación intensiva o drástica de trabajo a cualquiera de los componentes de hardware o software del sistema de seguimiento. En otras palabras, aunque las alineaciones de los ejes de Elevación y Azimutal, no se realicen correctamente durante la instalación, los nuevos algoritmos de seguimiento solar tienen la capacidad de alineación, únicamente cambiando los valores de *phi* (ϕ - tiempo), *lambda* (λ - Azimutal) y *zeta* (ζ - Elevación) en el programa de seguimiento.

La ventaja de los nuevos algoritmos de seguimiento es que pueden simplificar el trabajo de fabricación e instalación de colectores solares con mayor tolerancia en términos de alineación de seguimiento de ejes. Esta estrategia ha permitido grandes ahorros en términos de costo, tiempo y esfuerzo al omitir soluciones más complicadas propuestas por otros investigadores, tales como la adición de un controlador por retroalimentación en lazo cerrado o de una estructura mecánica flexible y compleja para corregir el error de seguimiento solar (Chen et al., 2001).

3. METODOLOGÍA

El objetivo de esta investigación desde una perspectiva fundamental es el desarrollo de un sistema de software que implementa y adapta un algoritmo de posicionamiento solar embebido. Hoy en día existen varios algoritmos competentes que calculan la posición del Sol, teniendo diferentes niveles de precisión, con errores máximos en la posición solar que abarcan desde 0.19 a 0.0027 grados, cubriendo así una amplia gama de aplicaciones posibles. Los algoritmos están optimizados para reducir en lo posible su costo en cuanto a software y hardware de control de movimiento. El rango de tiempo de vigencia de estos algoritmos es variable, algunos ya caducaron en el año 2010, otros vencen en el año 2050 e inclusive uno de ellos vence hasta el año 6000, esta vigencia permite a los usuarios emplear con éxito estos algoritmos, incluso en proyectos a largo plazo.

3.1. Energías alternativas

Se les consideran energías alternativas, limpias o renovables, a toda la energía que se obtiene de fuentes naturales inagotables por tener la capacidad de regenerarse por medio de métodos naturales (García-Lara, 2010), ver Figura 3.1.

Figura 3.1: Energías alternativas.
(Tomada de:
`http://www.esysa.com/slider/images/energias_limpias1.jpg`)

Las fuentes de energías renovables se dividen en dos categorías: las limpias y las contaminantes. Entre las limpias se encuentra la energía solar, como una de las fuentes generadoras de energías renovables, también encontramos a la energía eólica, la energía hidráulica y aquellas causadas por fenómenos geofísicos como la energía mareomotriz y la energía geotérmica (García-Lara, 2010). Y dentro de las contaminantes son el producto que se obtiene a partir de la biomasa, cuando se utiliza como combustible, donde durante la combustión emite dióxido de carbono y gases de efecto invernadero.

3.2. Energía solar

La energía solar juega un papel muy importante para la humanidad desde épocas inmemoriales y hoy en día se le considera como la fuente proveedora de energía más poderosa, e inagotable con la que cuenta el planeta Tierra. La energía solar se obtiene a partir de la captación y aprovechamiento de la radiación del Sol y es aprovechada principalmente para dos propósitos; para producir energía térmica o para generar energía fotovoltaica.

El aprovechamiento de este tipo de energía limpia e inagotable consiste en utilizar la radiación en calor para el calentamiento de agua, secado de granos o incluso para producir vapor. La energía solar es comúnmente utilizada en el método fotovoltaico, que por medio de paneles solares fabricados de semiconductores generan electricidad una vez que los rayos solares incurran en la superficie de estos captadores (García-Lara, 2010), ver Figura 3.2.

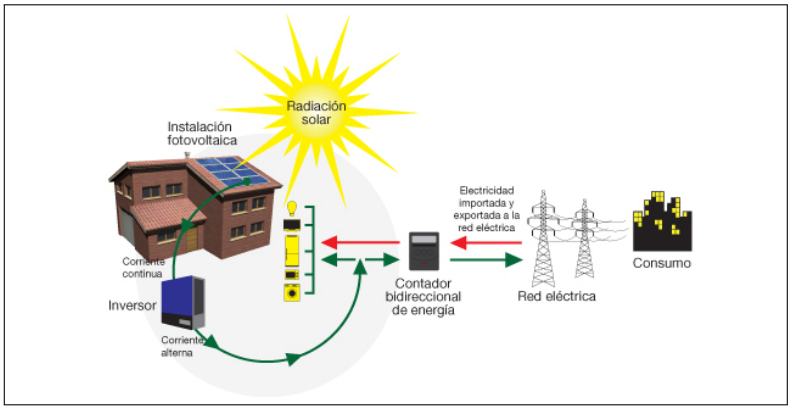


Figura 3.2: Autoconsumo con energía solar fotovoltaica.
(Tomada de: `http://sud.es/public/assets/Pagina_153/Esquema-Autoconsumo_cas.jpg`)

3.3. Radiación solar

El Sol es la estrella central del sistema solar, se encuentra a una temperatura aproximadamente 5,785 °K (grados Kelvin). Se le conoce como radiación solar al grupo de radiaciones electromagnéticas emitidas por el Sol . Dicha estrella cuenta en su interior con una serie de reacciones de fusión nuclear, estas a su vez producen una pérdida de masa que posteriormente se convierten en energía (Cengel, 2003). Esta energía es emitida en todas direcciones al exterior del Sol y llega parte de ella al planeta Tierra mediante la radiación solar, para ser aprovechada principalmente para una gran variedad de aplicaciones térmicas, por medio de colectores, paneles o hornos solares, el aprovechamiento que realiza cada uno de ellos depende en gran parte de la radiación solar que puedan captar, esto es debido a que cuando la radiación solar atraviesa la atmósfera del planeta Tierra sufre algunas transformaciones, ver Figura 3.3, de ahí que se manejan los siguientes tipos de radiación:

1. **Radiación solar directa:** esta radiación se recibe directamente del Sol, sin sufrir ningún tipo de dispersión cuando atraviesa la atmósfera del planeta Tierra.

2. **Radiación solar difusa:** al entrar en contacto con la atmósfera sufre cambios, siendo reflejada por las nubes que va en todas direcciones, como consecuencia de las reflexiones múltiples, no sólo de las nubes sino de las partículas de polvo atmosférico, montañas, árboles, edificios, entre otros. Este tipo de radiación se caracteriza por no producir sombra respecto a los objetos opacos interpuestos. Las superficies horizontales son las que más radiación difusa reciben.

3. **Radiación solar reflejada:** es el tipo de radiación que es emitida de parte de todos los objetos que se encuentran en la superficie del planeta Tierra. Las superficies verticales son las que más radiación reflejada reciben.

4. **Radiación solar global o total:** es la suma de las radiaciones directa y difusa. Mientras que la radiación total, es la suma de todas las radiaciones, directa, difusa y reflejada (Cengel, 2003).

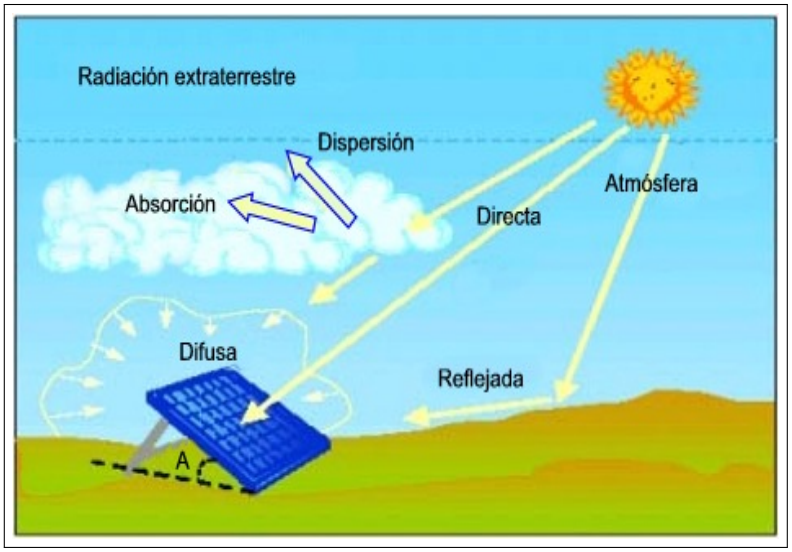


Figura 3.3: Radiación solar global o total.
(Tomada de: `http://www.pce-iberica.es/medidor-detalles-tecnicos/images/dosimetro-radiacion-esquema-radiacion-solar.jpg`)

Por lo tanto, la radiación que realmente nos importa es la directa, debido a que es la que mayor potencial tiene. También debemos tener en cuenta ciertos factores: En primer lugar, *el factor climático* es importante, en un día nublado, tendremos una radiación difusa; en cambio, si es soleado, la radiación recibida será directa. El segundo factor, *es la inclinación* de la superficie que recibe la radiación. Y, el tercer factor, es *la presencia o ausencia de superficies reflectantes*, las superficies claras son las que más reflejan la radiación solar, por este motivo, las casas se pintan de blanco.

3.4. Movimiento aparente del Sol respecto al planeta Tierra

El planeta Tierra se mueve al rededor del Sol con dos tipos de movimientos; el movimiento de rotación y el movimiento de traslación:

1. **El movimiento de rotación:** es el movimiento que el planeta Tierra realiza girando sobre su propio eje, a lo largo de un eje imaginario denominado eje terrestre en dirección oeste a este, en un periodo de tiempo aproximado de 24 horas, 0 minutos y 57.33 segundos, conocido como día sidéreo (Beltrán-Adán, 2007).

2. **Por otro lado el movimiento de traslación:** es el tiempo que tarda el planeta Tierra en completar el recorrido a través de su órbita alrededor del Sol y este se lleva a cabo en 365.2422 días (Beltrán-Adán, 2007).

Pero para la vista del ojo humano tal pareciera que el Sol es el que se mueve al rededor del planeta Tierra, en dirección del Este al Oeste. A este movimiento es lo que se le llama movimiento aparente del Sol (Jiménez-Vicente, 2015), y varia de estación a estación, es decir en otoño el movimiento del Sol es diferente al movimiento del Sol en invierno, en primavera es otra trayectoria diferente y en verano otra totalmente diferente a las tres anteriores estaciones del año, ver Figura 3.4, y más aún cuando el lugar de observación está a una Latitud alta.

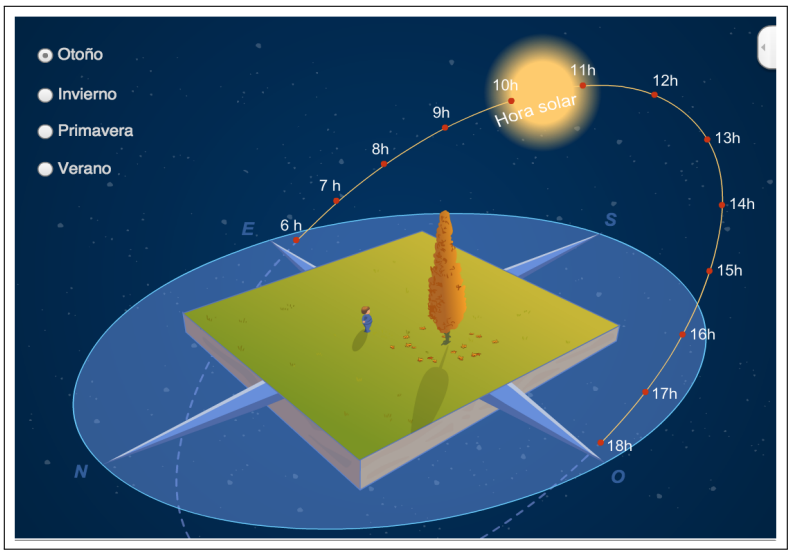


Figura 3.4: Movimiento aparente del Sol.
(Tomada de: `https://www.edumedia-sciences.com/es/fl/565`)

Para ubicar un punto en la superficie terrestre con precisión nos guiamos en base a las coordenadas geográficas, Latitud para Norte y Sur, y Longitud para Este y Oeste, establecidas a partir de una red geográfica de líneas imaginarias llamadas meridianos y paralelos. En resumen, las coordenadas geográficas son los ángulos laterales formados por los meridianos y paralelos del planeta Tierra.

3.4.1. Husos horarios

Las zonas horarias también son conocidas como husos horarios y son cada una de las veinticuatro áreas en que se divide el planeta Tierra, ver Figura 3.5, están centrados en meridianos de una longitud múltiplo de 15°, dando como resultado divisiones de veinticuatro franjas imaginarias de una hora, denominadas husos horarios (International, 2014). Por lo tanto, cada 15° de longitud hay una hora de diferencia, una más hacia el Este y una menos hacia el Oeste.

Figura 3.5: Husos Horarios.
(Tomada de: `http://www.worldtimezone.com/`)

Todos los husos horarios parten del denominado Tiempo Universal Coordinado (UTC), que está centrado sobre el meridiano de Greenwich en Londres (0°). Al pasar de un huso horario a otro en dirección Este hay que sumar una hora, por otro lado, al pasar de Este a Oeste hay que restar una hora. Es importante resaltar, que todos los lugares del planeta Tierra que están en el mismo meridiano tienen la misma hora solar, ya que todos los puntos que atraviesa tienen al Sol en la vertical a medio día. Existen países que poseen su propia zona horaria, por lo que no siguen el patrón que marca el Tiempo Universal Coordinado (International, 2014); si son las 12:00 UTC, estos países tendrán el siguiente horario local: Afganistán: 16:30 (UTC +4:30 horas), Australia: territorio del Norte 21:30 (UTC +9:30 horas)., India: 17:30 (UTC +5:30 horas), Irán: 15:30 (UTC +3:30 horas), Entre otros.

3.4.2. La trayectoria solar

Cada día el Sol sigue una trayectoria circular alcanzando su punto más alto al mediodía, de la misma manera para poder ubicar la trayectoria solar primero hay que conocer nuestra ubicación en el planeta Tierra. Para localizar un punto en el planeta Tierra se utiliza el sistema de coordenadas geografías expresando en ángulos la latitud y longitud propios del lugar (Beltrán-Adán, 2007). El ángulo de latitud nos indica que tan retirados estamos del Ecuador y el ángulo de longitud proporciona la posición de acuerdo al meridiano de Greenwich, ver Figura 3.6 y Figura 3.7.

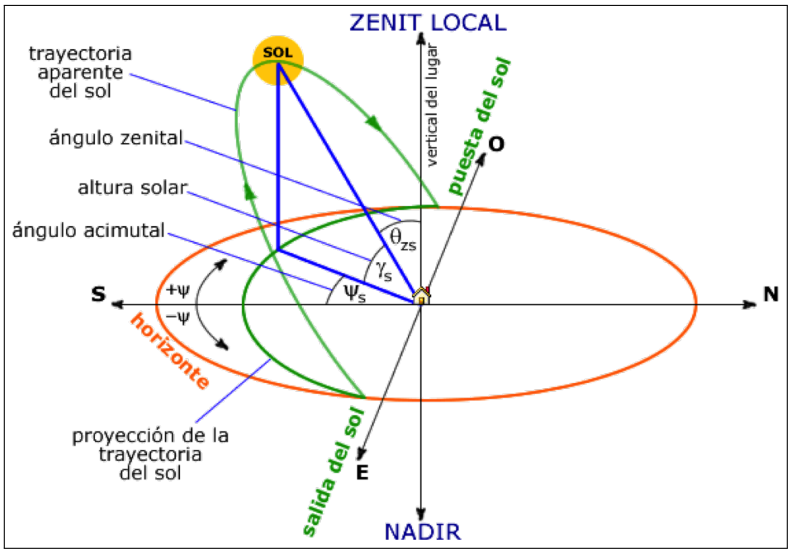


Figura 3.6: Trayectoria aparente del Sol.
(Tomada de: `http://www.ujaen.es/investiga/solar/07cursosolar/home_main_frame/02_radiacion/01_basico/images/posi_sol.gif`)

El valor del ángulo de altitud y azimut está en función de la hora, día del año y latitud del lugar que puede ser calculado por métodos numéricos (Peng et al., 2013). En primer lugar se determina el ángulo de inclinación del planeta Tierra (δ) debido al movimiento de traslación:

$$\delta = 23{,}45^{o} * \sin\left(\frac{360^{o} * (284 + N)}{365}\right) \tag{3.1}$$

Dónde: N es el día consecutivo del año que se desea conocer, N=1 representa el 01 de enero y N=365 representa el 31 de diciembre. El ángulo horario (ω) es el ángulo comprendido entre el meridiano local y la hora de estudio, es decir, es el ángulo que se forma entre el medio día solar y la hora de referencia, que se obtiene por:

$$\omega = (h+E)*15+\psi-300^{o} \tag{3.2}$$

Dónde: h representa el tiempo local, E representa el desajuste, ψ representa la longitud local.

Con los siguientes datos se obtiene el ángulo de la altura del Sol (θ):

$$\sin(\theta) = \sin(\phi)*\sin(\delta)+\cos(\phi)*\cos(\delta)*\cos(\omega) \tag{3.3}$$

Dónde el (ϕ) representa ángulo de la latitud. Y el ángulo de azimut del Sol (α) se calcula con la siguiente ecuación:

$$\sin(\alpha) = \frac{\cos(\delta)*\sin(\omega)}{\cos(\theta)} \tag{3.4}$$

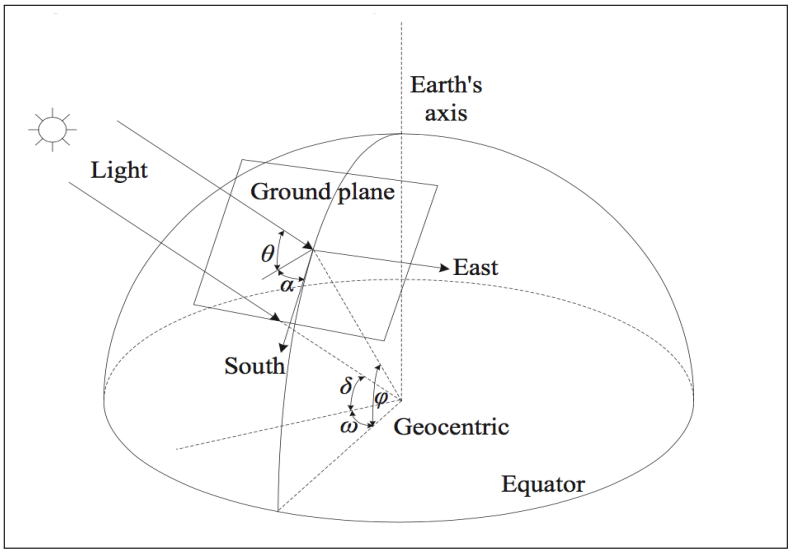


Figura 3.7: Modelo geométrico de la trayectoria solar.
(Tomada de: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6561859&isnumber=6560882`)

Es necesario aplicar una función inversa cuando el ángulo azimutal es mayor a 90°, es decir, cuando la posición del Sol rebasa el eje o plano vertical Este-Oeste, bajo el siguiente razonamiento:

$$\cos(\omega) < \frac{\tan(\delta)}{\tan(\phi)} \Rightarrow as = \sin^{-1}\left[\frac{\cos(\delta) * \sin(\omega)}{\cos(\theta)}\right] \tag{3.5}$$

$$\cos(\omega) < \frac{\tan(\delta)}{\tan(\phi)} \Rightarrow as = 180^{o} - \sin^{-1}\left[\frac{\cos(\delta) * \sin(\omega)}{\cos(\theta)}\right]$$

3.5. Seguidores solares

Un seguidor solar es la tecnología más adecuada principalmente para mejorar la producción de electricidad de una instalación fotovoltaica. Consiste de un sistema mecatrónico conformado básicamente por una parte fija y una móvil, con la finalidad de orientar paneles solares, colectores solares, entre otros, para aumentar la captación de radiación solar, cuenta con una superficie horizontal que debe permanecer aproximadamente perpendicular a los rayos del Sol durante el día (Jing-Min and Chia-Liang, 2013), ver Figura 3.8.

Figura 3.8: Seguidor Solar.
(Tomada de: `http://www.999automation.com/blog/wp-content/uploads/2013/03/tracking-the-sun.jpg`)

Existen métodos comunes que pueden utilizarse para implementar un sistema de seguimiento solar, estos son clasificados: según el tipo de movimiento que realicen (seguidor de uno o dos ejes) y según el algoritmo de seguimiento (según la luminosidad o con programación astronómica).

3.5.1. Tipos de seguimiento del Sol

Los sistemas de seguimiento solar se dividen en sistemas de un eje y dos ejes. Los sistemas de seguimiento solar de un eje únicamente pueden seguir la luz del Sol girando alrededor de un eje horizontal fijo, mientras que los sistemas de seguimiento solar de doble eje pueden girar sobre dos ejes: X, Y, acción que asegura que la luz del Sol llegue siempre verticalmente al plano horizontal situado en la parte superior de la estructura del seguidor solar (Jing-Min and Chia-Liang, 2013). Por otro lado, los tipos de seguimiento que pueden llevarse a cabo son:

1. **Seguimiento Ecuatorial:** consiste en el seguimiento de dos ejes paralelos a los del planeta Tierra (Norte-Sur / Este-Oeste), donde el movimiento en Norte-Sur proporciona el movimiento diurno y el movimiento Este-Oeste proporciona el ángulo que se toma con respecto al eje polar.

2. **Seguimiento Acimutal:** es uno de los más empleados, sobre todo cuando se trata de estructuras pesadas. Uno de los ejes es vertical y proporciona el seguimiento en acimut, el otro eje es horizontal y proporciona el seguimiento de altura. El seguimiento se realiza en dos ejes.

3.5.2. Métodos de seguimiento

El seguimiento se puede realizar por distintos métodos, por ejemplo:

1. **Seguimiento por sensores:** es el método más común que permite la detección del Sol por medio de sensores que detectan la presencia de radiación solar, los más comunes son los fotorresistivos que varían el valor resistivo dependiendo de la cantidad de luz que capten.

2. **Seguimiento por reloj solar:** este tipo de seguimiento está sujeto al tiempo que comprende un día (24 horas), presentando un movimiento efectivo de 12 horas, y no considera cambios climáticos repentinos.

3. **Seguimiento por coordenadas calculadas:** este tipo de seguimiento sigue la trayectoria del Sol entre cada posición mediante el cálculo de sus ***coordenadas astronómicas***, no precisa de la presencia de radiación, los sistemas coordenados son inmunes a los días nublados y otro tipo de circunstancia que puede producir errores; como por ejemplo: los destellos y tormentas.

3.6. Algoritmo de seguimiento solar seleccionado

Hoy en día existen numerosos algoritmos desarrollados para el seguimiento del Sol con un intercambio entre la precisión y complejidad, siendo exactos exactos dentro de 1° de error y suficientes para la mayoría de las aplicaciones de seguimiento solar, a continuación se listan tres de ellos principalmente:

1. **Algoritmo PSA:** Plataforma Solar de Almería (PSA), este algoritmo utiliza el tiempo universal (UT) para remover la incertidumbre causada por los tiempos locales de las diferentes zonas. Este algoritmo simplificado es exacto a 0.5 minutes (aproximadamente 0.008 grados) de arco para el año 1999-2015 (Blanco-Muriel et al., 2001). La ubicación se ingresa con latitud y longitud con los minutos y segundos convertidas a fracciones de un grado. El ángulo del azimuth se mide desde el norte verdadero del Norte no magnético y el ángulo zenith se mide respecto a la vertical. El ángulo de elevación se mide respecto a la horizontal.

2. **Algoritmo SPA:** Algoritmo de Posición Solar (SPA). Este algoritmo calcula los ángulos solares Zenith y Azimuth en un momento dado para una ubicación específica. Aplica únicamente para un intervalo de años que va desde el -2000 hasta 6000, con una incertidumbre de +/- 0.0003 grados, basados en fecha, hora, y posición GPS en el planeta Tierra (Reda and Andreas, 2008a; Michalsky, 1988; Meeus, 1998). Este algoritmo está clasificado como un algoritmo astronómico debido al alto grado de precisión.

3. **Algoritmo SOLPOS:** Algoritmo para Posición Solar e Intensidad. Este algoritmo calcula la posición solar aparente y la intensidad (energía solar máxima teórica) basada en la fecha, hora y lugar en la tierra. Algoritmo válido a partir del año 1950 a 2050, con una incertidumbre de +/- 0.01 grados (Laboratory, 2000).

De los algoritmos mencionados anteriormente en el capítulo 2, únicamente tres de ellos: Swift (Swift, 1976), Walraven (Walraven, 1978) y Michalsky (Michalsky, 1988), están completos en el sentido de que proporcionan el cálculo de las coordenadas horizontales del Sol, en otras palabras, proporcionan los ángulos de Azimutal y Elevación. Para establecer cuál es el mejor, el autor Blanco-Muriel (Blanco-Muriel et al., 2001) et al presenta una comparación con los valores correspondientes proporcionados por los algoritmos.

Por lo tanto, se elige el algoritmo SPA (Reda and Andreas, 2008a) del Laboratorio Nacional de Energías Renovables (NREL) por la característica de que funciona sin retroalimentación de la posición del Sol, además de la utilización de fórmulas y cálculos astronómicos, ver Apéndice A.2 y ver Apéndice A.3.

Aunque el algoritmo de SPA puede considerarse suficiente para la mayoría de las aplicaciones de posicionamiento solar, es necesario señalar que para que pueda operar en un sistema de software embebido, así como su facilidad de uso y eficiencia informática se le realizará una adaptación para ingresar al algoritmo automáticamente las coordenadas geográficas; latitud (Norte y Sur) y longitud (Este y Oeste), tiempo universal en formato de fecha/hora y ser ejecutado de forma embebido.

3.7. Tarjeta de evaluación para el sistema de control

Una de las grandezas de los sistemas embebidos es su hardware y software que están específicamente diseñados y optimizados para resolver un problema concreto eficientemente. El concepto de embebido, también se le conoce como: empotrado. La característica principal que diferencia a los Sistemas Embebidos de los demás sistemas electrónicos es que, por estar insertados dentro del dispositivo que controlan, están sujetos en mayor medida a cumplir requisitos de tamaño, fiabilidad, consumo y costo (Jacques-García et al., 2014).

Si bien hasta hace unos años, implicaba un alto costo económico implementar un sistema de control para cualquier prototipo, de tal forma que fuera capaz de realizar diversas

tareas programadas, en la actualidad la reproducción de microcontroladores, microprocesadores y plataformas de desarrollo, simplifican el trabajo y reducen costos de gran manera. Hoy en día encontramos en el mercado una infinidad de tarjetas, como la Raspberry Pi, Cubieboard, BeagleBone Black, Banana Pi, Arduino, Intel Galileo, entre otras, a las cuales se les puede embeber un sistema de software para la ejecución de una tarea específica.

3.7.1. Raspberry Pi

La Raspberry Pi es una tarjeta electrónica con un tamaño aproximado a una tarjeta de crédito, esta tarjeta de pequeñas dimensiones que cuenta con componentes similares a una computadora, fue desarrollada por un grupo de la Universidad de Cambridge en el Reino Unido con la finalidad de promover el desarrollo en las ciencias de la computación dirigida a los niños, esta tarjeta es de bajo costo y es una de las más utilizadas para el desarrollo de proyectos por estudiantes, así como expertos en programación quienes las utilizan para desarrollos más sofisticados. Esta tarjeta fue una de las primeras en desarrollar este concepto de computadora de bolsillo. Con esta tarjeta prácticamente se tiene una computadora del tamaño de una tarjeta de crédito, en la que podemos realizar las mismas actividades que en una computadora de escritorio, sólo que con recursos más limitados (Raspberry-Pi, 2015).

Existe una infinidad de componentes externos que son diseñados específicamente para esta tarjeta, tales como, sensores, cámaras, paneles LCD touch, entre otros, que permiten expandir las capacidades de esta tarjeta para realizar un proyecto más completo. El diseño de la tarjeta cuenta con un System-on-a-chip Broadcom BCM2835, un procesador central (CPU) quad-core ARM Cortex-A7 a 900MHz. Actualmente se encuentra en el mercado la tarjeta Raspberry Pi 2 modelo B, ahora con menor consumo de energía, cuenta con cuarenta pines GPIO que incluye los puertos: UART, I2C, SPI4, USB, ver Figura 3.9.

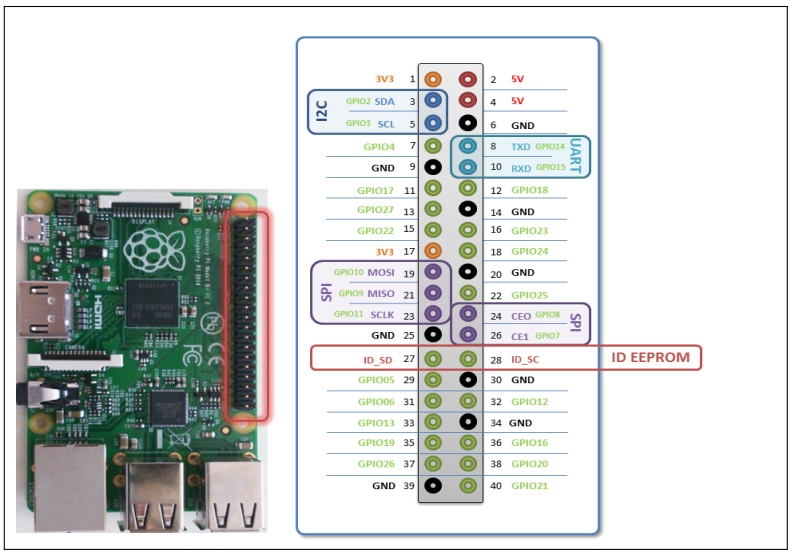


Figura 3.9: Tarjeta Raspberry Pi 2 - Modelo B.
(Tomada de:
`https://raspilab.files.wordpress.com/2014/08/gpiosb.png`)

Las características principales de esta tarjeta son las siguientes:

1. Procesador quad-core ARM Cortex-A7 a 900MHz.
2. Consumo energético: 900mA, 3.5W con una fuente de alimentación micro USB de 5V.
3. Conector MIPI CSI que permite instalar un módulo de cámara.
4. Salidas de vídeo; conector HDMI y conector de 3.5mm como salida de audio.
5. Almacenamiento con tarjeta micro SD y periféricos GPIO, SPI, I2C y UART.
6. Sus dimensiones son 85.60 x 56.50 mm.
7. Los sistemas operativos oficiales soportados por la tarjeta son basados en GNU/Linux: Debian, Feldora, Arch Linux, Slacvware Linux, RISC OS2.

3.7.2. Protocolos de comunicación

Un protocolo de comunicación es un conjunto de reglas que se rigen bajo estándares que permiten que dos o más elementos de un sistema de comunicación hablen entre ellos, para transmitir y/o compartir información a través de un medio físico e inclusive inalámbrico.

Protocolo I2C

I2C, por sus siglas en inglés (Inter- Integrated Circuits), es un protocolo de transferencia de datos en serie, diseñado por Phillips a principios de los 80's, su principal característica es la de facilitar la comunicación entre memorias, microcontroladores y otros dispositivos, permitiendo la transferencia de información entre diferentes dispositivos conectados en paralelo, ver Figura 3.10. El protocolo I2C es muy popular y ampliamente utilizado, con características, tales como: capacidad de control, menos conexión de alambre (Zhengwei, 2010). En este protocolo existen dos tipos de dispositivos, el maestro y los esclavos, el dispositivo maestro es el único que puede iniciar una comunicación, a su vez, los dispositivos esclavos únicamente pueden escuchar lo que envía el dispositivo maestro.

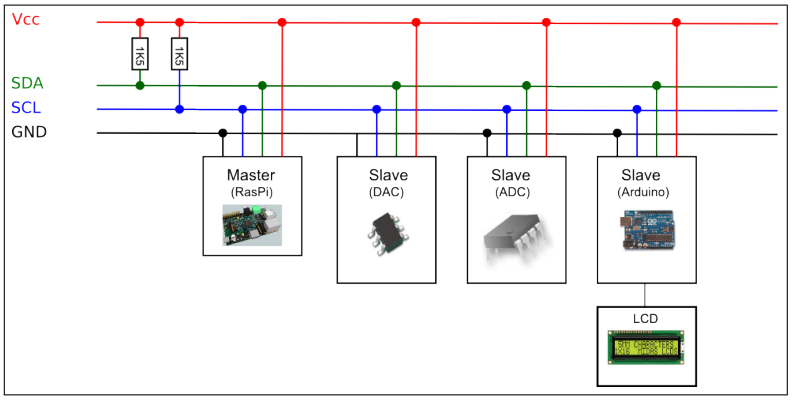


Figura 3.10: Protocolo I2C.
(Tomada de:
`http://quick2wire.com/wp-content/uploads/2012/05/image00.png`)

Cada dispositivo conectado al bus tiene un número de dirección que es asignado por software, conservando permanentemente una relación maestro/esclavo entre el microcontrolador y los dispositivos conectados. Las características principales del bus I2C son las siguientes:

1. Velocidad estándar de 100 Kbit/s y 400 Kbit/s.

2. Configuración maestro/esclavo.

3. Dos líneas: la de datos (SDA) y la de reloj (SCL).

Protocolo RS-232

RS-232 es un protocolo de comunicación serial desarrollado por la Asociación de la Industria Electrónica en el año de 1962. Este protocolo es uno de los más utilizados, ya que se puede implementar en una gran variedad de soluciones, tanto en productos industriales y comerciales, ya sea para realizar una conexión con una impresora, algún módem o instrumentación industrial.

El protocolo RS-232 tiene como desventaja que únicamente se puede realizar la comunicación punto a punto, ver Figura 3.11, esto significa que sólo podemos tener la comunicación directa entre dos dispositivos; se utiliza en entornos de bajo ruido y solamente en aplicaciones de corta distancia, comúnmente 15 metros (Group, 2002).

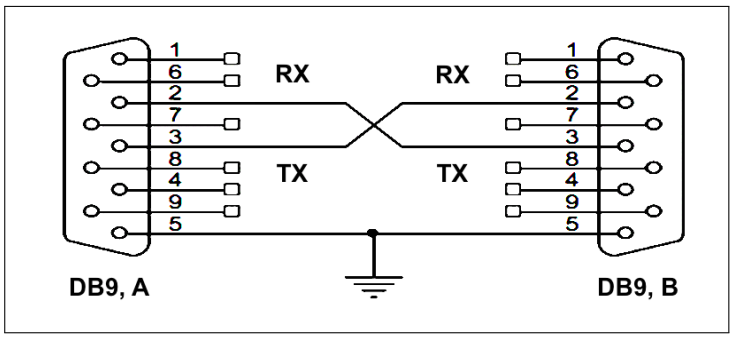


Figura 3.11: Protocolo RS-232.
(Tomada de:
`http://www.coscom.org/Images/schnittstellenbelegung_rs_232.gif`)

Las características más importantes de la comunicación serial son:

1. La velocidad de transmisión, los bits de datos, los bits de parada y la paridad.
2. Distancia máxima de 12 metros (a una velocidad de 10 Mbit/s).
3. Rango de bus de -7V a +12V.

3.7.3. Características de las tarjetas de evaluación

Por las capacidades y características, la tarjeta Raspberry Pi es muy utilizada en proyectos por estudiantes y profesionales, actualmente se pueden encontrar desde pequeños proyectos desarrollados con esta tecnología, como un centro multimedia, hasta grandes inventos, como centros de seguridad, la imaginación es el límite para poder desarrollar proyectos con esta tecnología.

Ahora están cada vez más de moda un sin fin de placas y/o tarjetas, debido a ello en la Tabla 3.1, se describen las mejores características de la tarjeta Raspberry Pi modelo B+, comparándola con otros modelos de la misma gama de cualidades existentes en el mercado, cada una tiene sus fortalezas y debilidades, donde una plataforma es mejor que otra para una determinada aplicación, que dependiendo los requerimientos del proyecto o problema a resolver son utilizadas.

Tabla 3.1: Características de las tarjetas de evaluación.

Propiedad	*BananaPi*	*RaspberryPi*	*CubieBoard*	*Olimex*	*Arduino*	*BeagleBone*
Tamaño	8.56cmx5.60cm	8.56cmx5.65cm	10cmx6cm	8.4cmx6.0cm	7.5cmx5.3cm	8.64cmx5.33cm
Procesador	ARM Cortex-A7 Dual-Core	ARM Cortex-A7 Quad-Core	Cortex A8	Cortex A8	ATMega 328	ARM Cortex-A8
Reloj	1GHZ	900MHZ	1GHZ	1GHZ	16MHZ	700MGZ
Memoria RAM	1G DDR3	1GB	512MB/1GB	512MB	2KB	256MB
Memoria Flash	64GB en un slot SD, 2T en un disco SATA 2.5	Hasta 32 GB vía micro SD	4GB	4GB	31KB	4GB(microSD)
Memoria Eeprom	–	–	–	2KB	1KB	–
Alimentación	5V	5V	5V	5V	7-12V	5V
Potencia mínima	N/A	900mA(3.5W)	N/A	N/A	42mA(3w)	170mA(0.85W)
Digitales GPIO	26	8	96	68/74	14	66
Entradas Analóg.	N/A	N/A	N/A	N/A	6 10-bit	7 12-bit
PWM	1	–	–	–	6	8
Puertos i2C	1	1	1	3	2	2
Puertos SPI	1	1	1	1	1	1
Puertos UART	1	1	1	2	1	5
S. O.	Android,Firefox OS, Linux etc.	Debian GNU/Linux Fedora, Arch Linux	N/A	N/A	Arduino Tool	Python, Scratch, Squeak, Cloud9/Linux
Puertos Ethernet	10/100/1000	10/100	10/100	10/100	N/A	10/100
Puertos USB	2 USB 2.0, 1 micro USB	4 USB 2.0	2 USB 2.0, 1 micro USB 2.0	2 USB 2.0	N/A	N/A
Salida de Vídeo	CVBS y HDMI, RGB/LVDS	HDMI	HDMI	HDMI	N/A	N/A
Salida de Audio	3.5mm jack	3.5mm jack	N/A	N/A	N/A	Analógica

3.8. Sistema embebido seleccionado

Para la implementación del software y algoritmo SPA, se opta por la tarjeta Raspberry Pi 2 en su versión B, ver Figura 3.12, ya que es una tarjeta de desarrollo que proporciona versatilidad para múltiples aplicaciones, cuenta con un sistema operativo Linux que pone a nuestra disposición una colección de herramientas de programación, así como librerías que nos permiten interactuar con el hardware a través de los diferentes buses de comunicación que esta tarjeta posee.

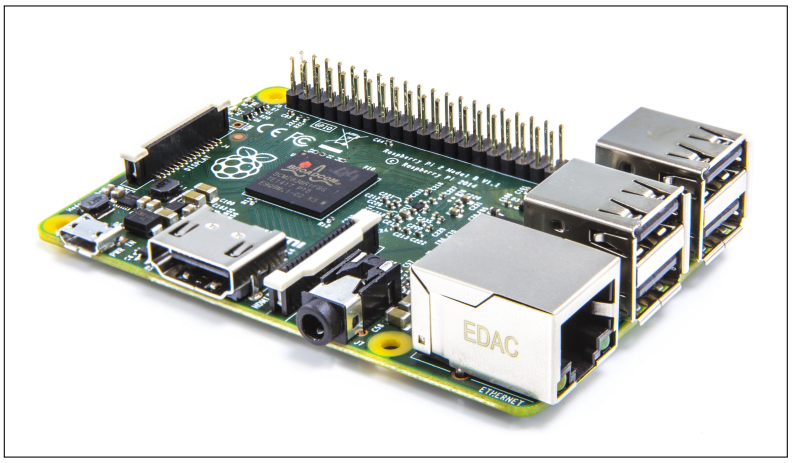


Figura 3.12: Sistema embebido seleccionado.
(Tomada de:
`https://www.adafruit.com/images/1200x900/1914-01.jpg`)

3.9. Prototipo seleccionado para seguimiento solar

El prototipo para seguidor solar no forma parte de esta investigación, pero para fines de demostración, se elige uno existente en el mercado con el propósito de evaluar la posición de los ángulos Azimutal y Elevación, que el algoritmo SPA arroja como resultado, ver Figura 3.13. La selección del prototipo para seguidor solar se realiza considerando los requerimientos del algoritmo SPA y que los componentes se puedan adquirir fácilmente, para evitar la fabricación de piezas mecánicas de difícil manufactura.

A continuación se enumeran los requisitos principales que debe cubrir el prototipo:

1. Que contenga dos ejes: Azimutal y Elevación.
2. Que contenga un controlador para los dos ejes.
3. Que contenga una interfaz de comunicación I2C o RS-232.
4. Que el tiempo de entrega no exceda las 8 semanas.

Figura 3.13: Prototipo seleccionado para seguimiento solar.
(Fuente propia)

El prototipo se compone de dos estaciones de rotación de precisión series URS100BCC que proporcionan precisión de 360° de movimiento continuo y que incluyen un motor DC que dispone de un codificador de resolución ultra-alta, 8000 cts/rev, montado en un tornillo sin fin. Y un controlador ESP301 integrado con dos ejes de movimiento, que cuenta con una interfaz USB a RS-232 estándar y funciones del panel frontal extendidas, adicionalmente, implementa comandos de movimiento y algoritmos de control PID compatibles con las estaciones de rotación URS.

4. DESARROLLO E IMPLEMENTACIÓN DEL ALGORITMO

A partir de lo expuesto en los capítulos anteriores referente a la revisión de literatura y metodología, en los cuales se describieron los tipos de seguidores solares, tipos de algoritmos de seguimiento solar, y tipos de tarjetas evaluadoras, en este capítulo se exponen los detalles sobre el desarrollo del proyecto de investigación con el propósito de demostrar la hipótesis, cumplir con los objetivos y dar una respuesta concreta al problema que se planteó. Para ello se comienza con la implementación de un sistema de software embebido en una tarjeta, la cual contendrá el algoritmo de posicionamiento solar para el control y orientación de un prototipo de seguimiento solar.

4.1. Metodología técnica

En años anteriores diseñar una tarjeta electrónica era casi imposible para una persona cualquiera, ya que el alto costo de software, de hardware, y la exigencia de conocimientos significaban verdaderas limitaciones, pues hoy en día ya no lo es, y además, aprovechando la ventaja de los nuevos algoritmos de seguimiento de que simplifican el trabajo de fabricación e instalación, permitiendo grandes ahorros en términos de costo, tiempo y esfuerzo al omitir las soluciones más complicadas, tales como la adición de un controlador por retroalimentación en lazo cerrado o de una estructura mecánica flexible y compleja con infinidad de sensores para corregir el error de seguimiento solar. Es por ello que para la resolución del presente proyecto de investigación se propone lo siguiente:

4.1.1. Análisis funcional del sistema

El procedimiento que se propone se expone en la Figura 4.1, consta básicamente de los siguientes elementos u objetos: módulo GPS, módulo RTC, tarjeta Raspberry Pi, prototipo Seguidor Solar, y módulo LCD.

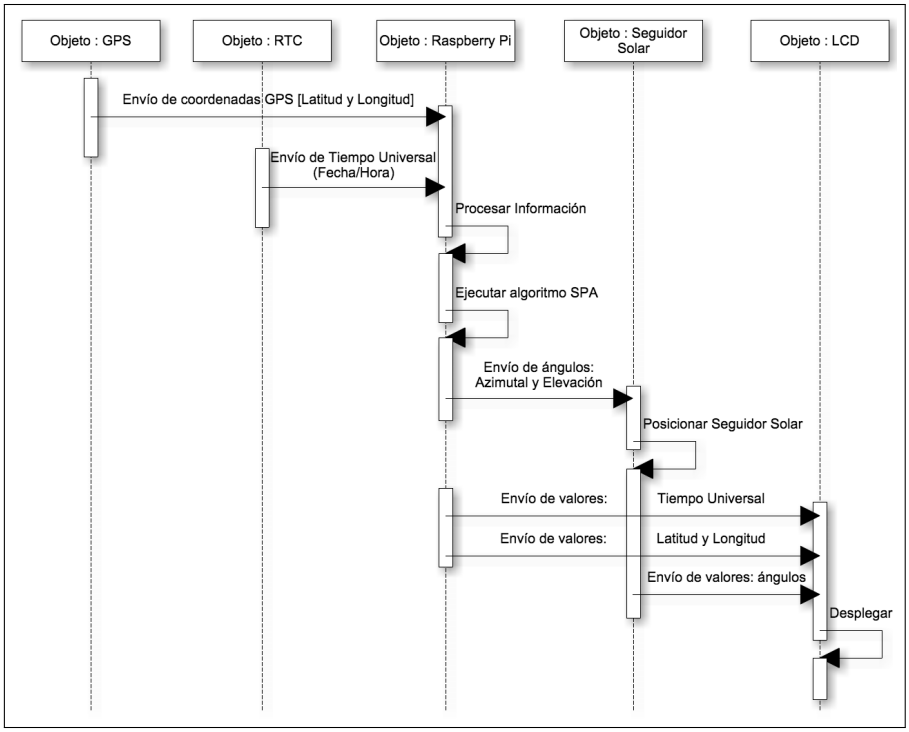


Figura 4.1: Diagrama de secuencia.

Descripción del diagrama:

1. Primeramente, un módulo GPS estará trasmitiendo a la tarjeta Raspberry Pi su posición GPS, tales como: Latitud, Longitud y Altitud.

2. De igual manera, se tendrá conectado un RTC (Reloj de Tiempo Real) a la tarjeta Raspberry Pi, el cual estará trasmitiendo la hora de reloj.

3. Una vez establecida la comunicación, la tarjeta Raspberry Pi recibe los datos, los procesa y los envía al algoritmo que calcula los ángulos de posición del Sol (Azimutal y Elevación).

4. Posteriormente la Raspberry Pi se encarga de enviar estos ángulos a un controlador de dos motores para posicionar el prototipo de seguidor solar.

5. Finalmente la Raspberry Pi desplegará en un LCD información como: fecha/hora, coordenadas GPS, ángulos de posición Azimutal y Elevación.

Diagrama a bloques

En el siguiente diagrama a bloques, ver Figura 4.2, se especifica más a detalle la comunicación que existe entre los distintos módulos, así como los protocolos que permiten la comunicación entre ellos:

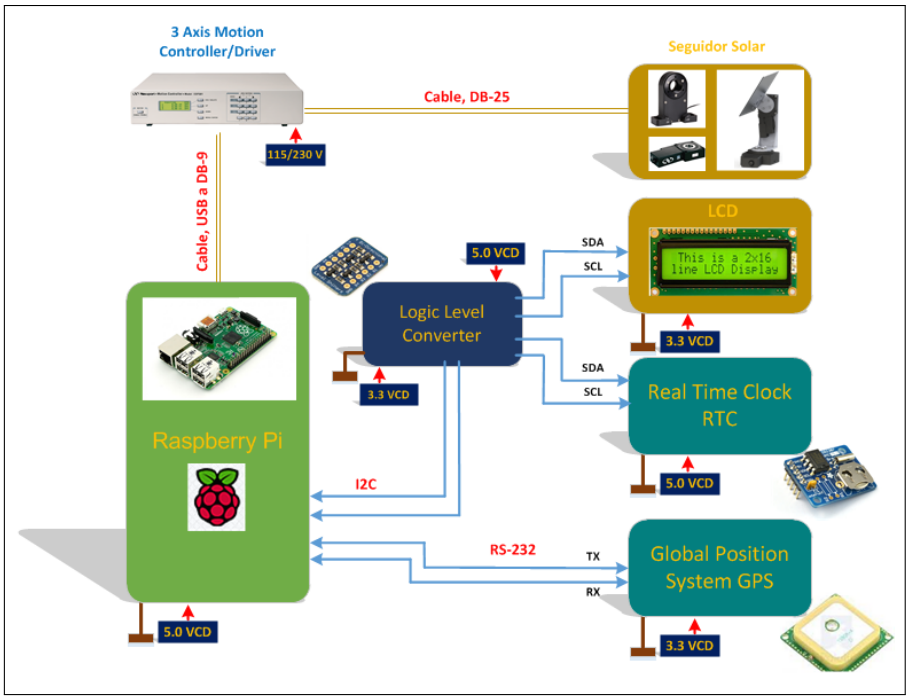


Figura 4.2: Diagrama a bloques.

Descripción:

1. Se puede observar que el GPS se comunica con la tarjeta Raspberry Pi a través de un bus de comunicación serial RS-232.

2. Para comunicar el LCD con la tarjeta Raspberry Pi se utiliza el bus de comunicación I2C, donde: la Raspberry Pi es el dispositivo Maestro y el LCD el dispositivo esclavo.

3. De igual manera y dadas las cualidades que proporciona el protocolo I2C, de poder tener más de un dispositivo como esclavo, se conecta una tarjeta RTC.

4. Se agregan dos elementos más, un controlador de motores para tres ejes, y dos mesas rotatorias motorizadas. Para comunicar la tarjeta Raspberry Pi con el controlador se utiliza el bus de comunicación RS-232, el cual se encarga de posicionar las dos mesas rotatorias motorizadas.

Requerimientos del sistema electrónico

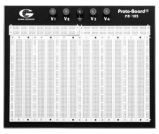
A continuación se listan los requerimientos considerados para el desarrollo del sistema electrónico:

1. Una Tarjeta Raspberry Pi B+ con accesorios.
2. Un dispositivo LCD 16x2 caracteres.
3. Un módulo GPS de 66 canales w/10 Hz.
4. Una tarjeta para Reloj de Tiempo Real.
5. Dos mesas giratorias motorizadas de precisión con un controlador para dos ejes.
6. Un cable Adaptador USB a RS-232.
7. Dos fuentes de alimentación de 5 Volts.
8. Estructura de prototipo de seguidor solar (placa para el eje Azimutal, placa para el eje de Elevación, tornillería para ensamble de seguidor solar).
9. Un circuito adaptador de voltajes bidireccional de 4-canales de 5 Volts a 3.3 Volts.
10. Placa de circuito impreso para montaje de tarjeta de reloj de tiempo real, adaptador de voltajes y módulo GPS.
11. La selección y pruebas del prototipo mecatrónico del seguidor solar se realiza en el laboratorio de Mecatrónica del Centro de Ingeniería y Desarrollo Industrial (CIDESI).

Selección de componentes

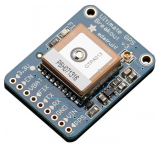
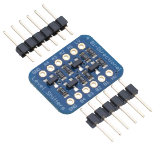
Para implementar el sistema electrónico, primeramente se tienen que seleccionar todos los diferentes componentes que lo conforman. Por lo tanto, a continuación se listan en la Tabla 4.1 cada componente seleccionado según el grado de eficiencia que le brinda al sistema de software y que a su vez permitirá posicionar el prototipo de seguidor solar a través del algoritmo de posicionamiento solar.

Tabla 4.1: Lista de componentes.

Componente	Descripción	Distribuidor	Precio
	Raspberry Pi 2 - Model B - ARMv7 with 1G RAM	Adafruit.	44.95 USD
	Adafruit Perma-Proto Half-sized Breadboard PCB - Single	Adafruit.	4.90 USD
	Cables para protoBoard se usan para la conexión y alimentación de los dispositivos como el LCD, GPS, RTC.	Adafruit	2.00 USD
	Cargador de 5v. se utiliza para alimentar la protoboard y los demás componentes a utilizar.	Adafruit	3.00 USD
	LCD Shield Kit w/ 16x2 Character Display - Only 2 pins used! - BLUE AND WHITE.	Adafruit	19.95 USD
	DS1307 Real Time Clock breakout board kit.	Adafruit	9.00 USD

Continúa en la siguiente página...

Tabla 4.1: ...Continuación de la página anterior.

Componente	Descripción	Distribuidor	Precio
	Motorized Rotation Stage.	Newport	3,405.60 USD
	ESP301 Motor Controller/Driver, 3 axis, USB.	Newport	3,961.10 USD
	Ultimate GPS Breakout - 66 channel w/10 Hz updates - Version 3.	Adafruit	39.95 USD
	4-channel I2C-safe Bi-directional Logic Level Converter.	Adafruit	3.95 USD

4.1.2. Diseño de la arquitectura del sistema

Mediante el uso de plataformas de desarrollo de software embebido, se pretende realizar el prototipo electrónico de seguidor solar para dos ejes de movimiento, que consta de un sistema de software embebido, un sistema de electrónica de control, y un sistema mecatrónico, componentes con los cuales se podrá orientar al Sol para obtener una mayor y eficiente captación de radiación, ver Figura 4.3.

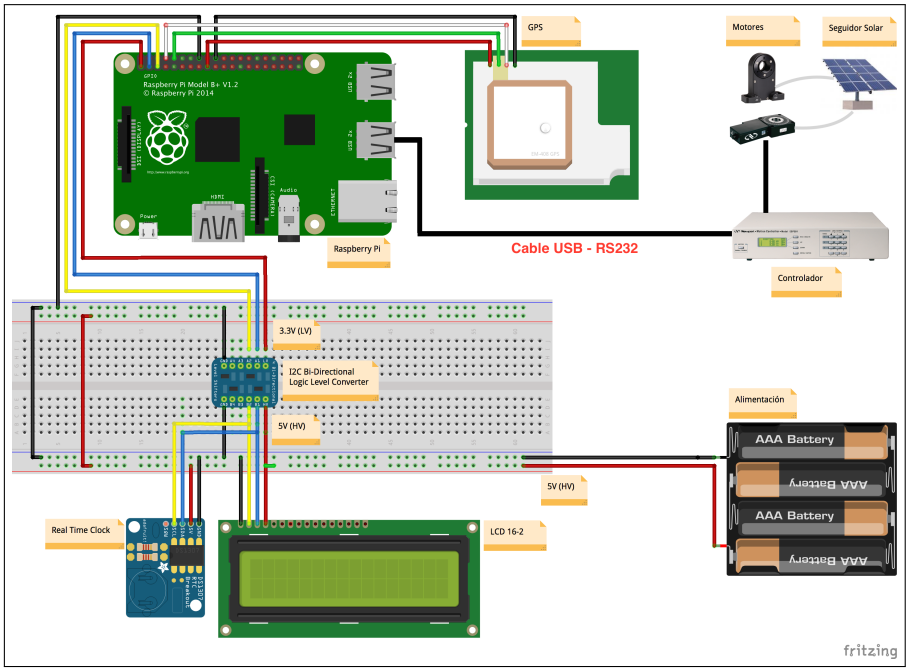


Figura 4.3: Arquitectura general del sistema.

Arquitectura de hardware

El software se encuentra embebido en la memoria micro SD de la tarjeta Rapberry Pi, que cuenta con un procesador de arquitectura ARM bajo un sistema operativo basado en el núcleo de Linux llamado Raspbian, una distribución derivada de Debian que provee la herramienta de compilación gcc para código fuente escrito con instrucciones de lenguaje de programación C, mismo que nos permite manipular los puertos de propósito general de entradas y salidas (GPIO) de la tarjeta.

Los puertos GPIO de la tarjeta Raspberry Pi se componen de cuarenta pines distri-

buidos en dos columnas de veinte pines cada una. Cada pin tiene si propósito específico, el mismo que puede ser visualizado en la Figura 3.9.

Arquitectura de software

La plataforma para desarrollar la programación del sistema de software e implementación del algoritmo, es el lenguaje C, mediante la herramienta de entorno de desarrollo Geany versión 1.24.1, debido a que es un entorno de desarrollo integrado pequeño, ligero y ampliamente compatible con distribuciones Linux.

4.1.3. Especificación de componentes

Como ya se mencionó anteriormente en el análisis funcional del sistema, los componentes para el diseño electrónico inicia con una tarjeta Raspberry Pi que tiene comunicación con un Reloj de Tiempo Real (RTC) que proporciona a la tarjeta los datos confiables de fecha/hora, y un display LCD con el fin de visualizar los resultados del algoritmo SPA, estos tres componentes se relacionan entre sí, a través del protocolo I2C, definiendo a la Raspberry Pi como el controlador maestro, mientras que el RTC y LCD definidos como esclavos.

En los pines 3 y 5 de la tarjeta Raspberry Pi, existe un puerto de comunicaciones SDA/SCL I2C, donde el pin 3 es utilizado para la trasmisión y recepción de datos, y el pin 5 para el control de reloj. Este puerto I2C es configurado mediante la modificación del archivo ***modules***.

A continuación se muestra el diagrama de las conexiones para la comunicación entre la tarjeta Raspberry Pi, un display LCD y el RTC, ver Figura 4.4.

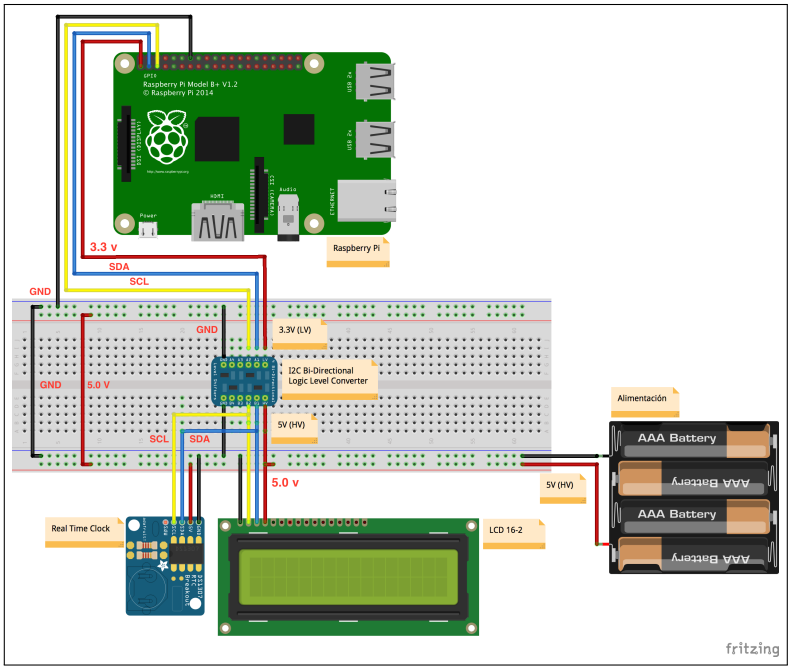


Figura 4.4: Diagrama de comunicación Raspberry Pi, LCD y RTC.

De igual manera, se tiene presente un módulo GPS, el objetivo de este módulo es proveer a la tarjeta Raspberry Pi los valores de Latitud, Longitud y Altitud a través del puerto de comunicaciones RS-232, ver Figura 4.5. En los pines 8 y 10 de la tarjeta Raspberry Pi, existe un TX/RX asíncrono UART, el mismo que consiste en un puerto de comunicaciones serial, donde el pin 8 es utilizado para la trasmisión de datos, y el pin 10 para la recepción de datos. Este puerto serial es configurado mediante la modificación del archivo ***cmdline.txt***.

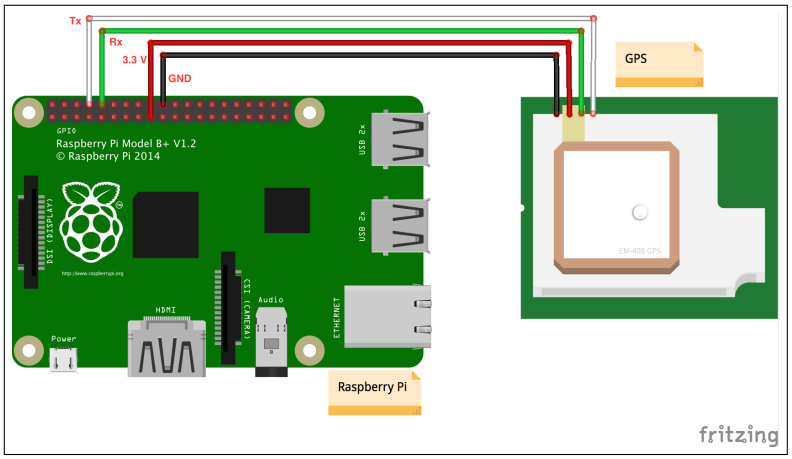


Figura 4.5: Diagrama de comunicación Raspberry Pi con GPS.

Finalmente, la tarjeta Raspberry Pi se comunica con un módulo Controlador, este módulo es el encargado de accionar lo motores para posicionar el prototipo de seguidor solar. Primeramente la Raspberry Pi envía los ángulos de Azimutal y Elevación a través de comandos por medio del puerto RS-232 hacia el Controlador, posteriormente el Controlador es el encargado de interpretar los comandos para accionar los motores, ver Figura 4.6.

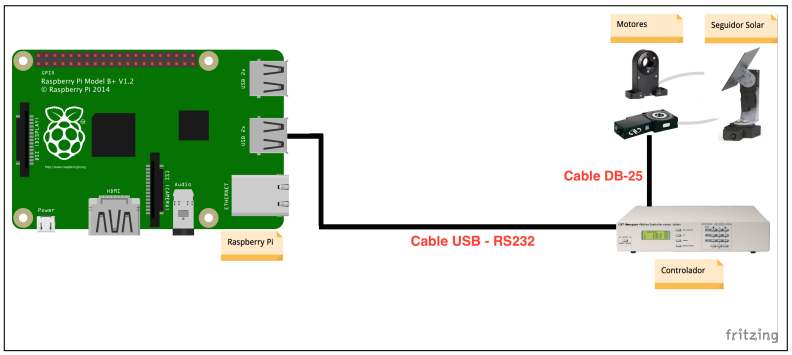


Figura 4.6: Diagrama de comunicación Raspberry Pi con controlador de seguidor solar.

4.1.4. Configuración de las comunicaciones

Terminadas las conexiones, el siguiente paso es lograr la comunicación con los diferente módulos del sistema electrónico, para ello, previamente se tienen que configurar y habilitar los puertos de comunicación I2C y RS-232 en la tarjeta Raspberry Pi. A continuación se describen los pasos que se requieren para realizar ésta tarea en la tarjeta.

Configuración del protocolo I2C

Partimos del supuesto de que se tiene instalado como sistema operativo la distribución de Linux Raspbian en la tarjeta Raspberry Pi. Para ello se requiere abrir la aplicación LXTerminal, la cual permitirá ejecutar los siguientes comandos.

1. Primeramente se procede a abrir y editar el archivo ***modules***. En la terminal escribir la siguiente instrucción:

```
1 pi@raspberrypi~$ sudo nano /etc/modules
```

2. Agregar las siguientes líneas de código al archivo ***modules***, ver Figura 4.7:

```
1 i2c-bcm2708
2 i2c-dev
```

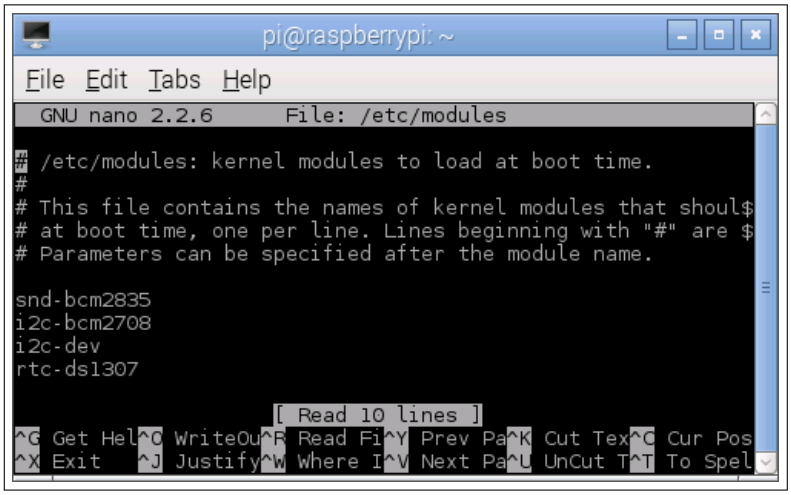


Figura 4.7: Edición del archivo *modules.*

A continuación guardar los cambios presionando las teclas Ctrl + X, seleccionar la opción Guardar, y salir del editor.

3. El bus I2C permite conectar en serie varios dispositivos. Para conocer las direcciones del LCD y el RTC, se requiere instalar la utilidad ***i2c-tools*** con el siguiente comando:

```
1 pi@raspberrypi~$ sudo apt-get install i2c-tools
```

4. De igual manera se procede a editar el archivo ***raspi-blacklist.conf***. En la terminal escribir la siguiente instrucción:

```
1 pi@raspberrypi~$ sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

5. A continuación editar y comentar el archivo, ver Figura 4.8, agregando al inicio de cada línea un (#) en las dos líneas siguientes:

```
1 #blacklist spi-bmc2708
2 #blacklist i2c-bcm2708
```

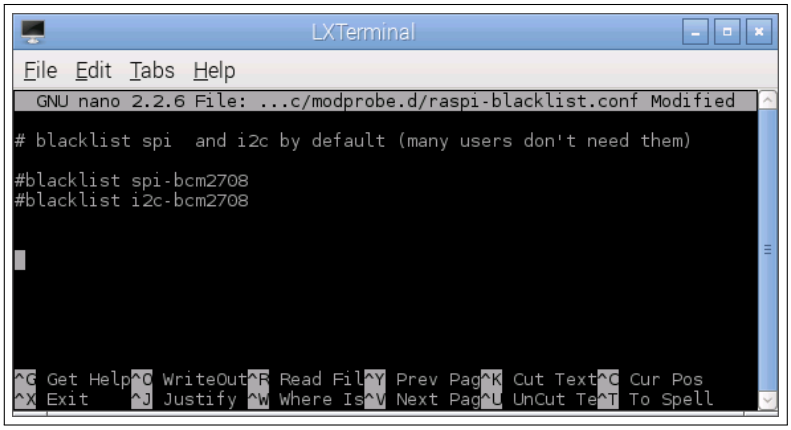


Figura 4.8: Edición del archivo *raspi-blacklist.conf*.

A continuación guardar los cambios presionando las teclas Ctrl + X, seleccionar la opción Guardar, y salir del editor.

Una vez completado los pasos anteriores, escribir el siguiente comando en la terminal para ver las direcciones de los dispositivos LCD y RTC conectados en el bus I2C, ver Figura 4.9:

```
1 pi@raspberrypi~$ sudo i2cdetect -y 1
```

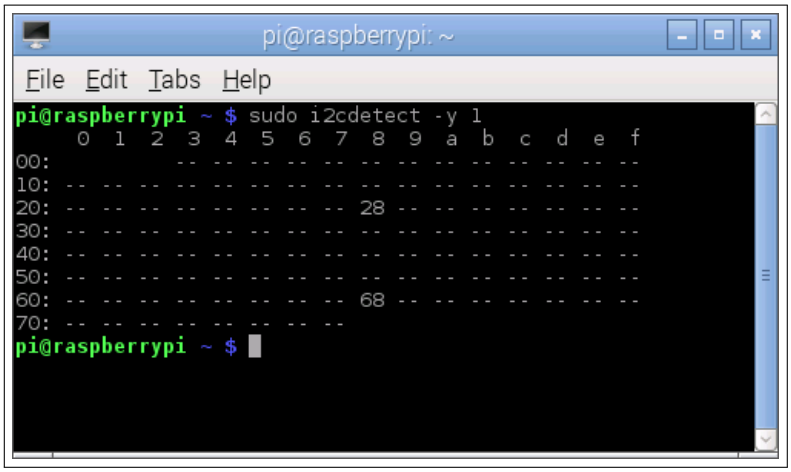


Figura 4.9: Dispositivos conectados en el bus I2C.

Configuración del protocolo RS-232

Para utilizar el puerto RS-232 en la tarjeta Raspberry Pi es necesario realizar los siguientes cambios, debido a que por configuración de fábrica, el puerto serie está configurado para entradas y salidas de consola. De igual manera se requiere abrir la aplicación LXTerminal, la cual permitirá ejecutar los siguientes comandos.

1. Primeramente se procede a abrir el archivo ***cmdline***. En la terminal escribir la siguiente instrucción:

```
pi@raspberrypi~$ sudo nano /boot/cmdline.txt
```

2. Se edita el archivo para deshabilitar la información de booteo que se envía por el puerto, para ello identificar la siguiente línea:

```
dwc_otg.lpm_enable = 0 console = ttyama0, 115200 kgdboc = ttyama0, 115200 console = tty1 root = /dev/↩
      mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

   Identificar el siguiente texto:

```
console = ttyama0, 115200 kgdboc = ttyama0, 115200
```

   Eliminarlo, de tal manera que quede de la siguiente forma:

```
dwc_otg.lpm_enable = 0 console = tty1 root = /dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

   Con esto se detendrá la inicialización de envío de información de arranque. A continuación guardar los cambios presionando las teclas Ctrl + X, seleccionar la opción Guardar, y salir del editor.

3. El siguiente paso es editar el archivo ***inittab*** que se encuentra en la ruta "etc/inittab" para deshabilitar el logg de consola en el puerto serial, abrir el archivo con la siguiente instrucción:

```
pi@raspberrypi~$ sudo nano /etc/inittab
```

4. Buscar la siguiente línea:

```
1 T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Comentarla agregando el símbolo de # al inicio de la línea, ver Figura 4.10:

```
1 #T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

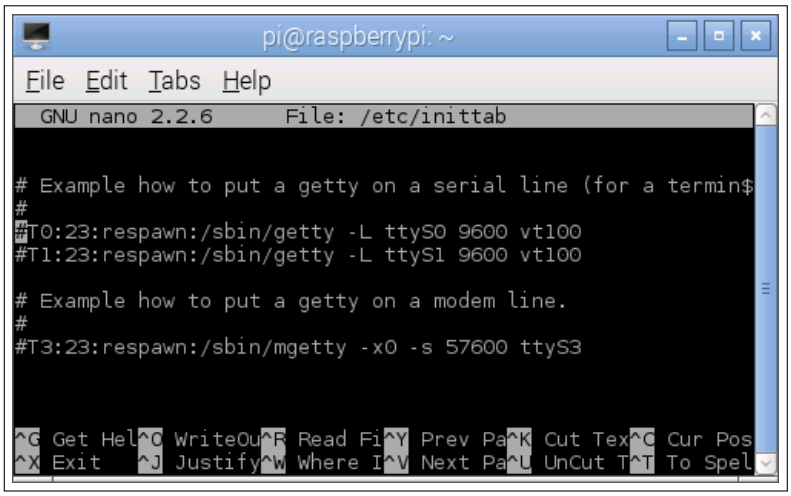


Figura 4.10: Edición del archivo *inittab*.

A continuación guardar los cambios presionando las teclas Ctrl + X, seleccionar la opción Guardar, y salir del editor.

5. Finalmente reiniciar la Raspberry Pi con el comando:

```
1 pi@raspberrypi~$ sudo reboot
```

Con estos pasos, los puertos I2C y RS232 quedan configurados y disponibles en la tarjeta Raspberry Pi para lograr la comunicación con los distintos módulos a través del sistema de software embebido.

Instalación de librerías

En el lenguaje de programación C, al igual que en otros lenguajes, es necesario utilizar librerías que contienen comandos especiales que periten desarrollar software de una manera más sencilla. Para el caso del manejo de los puertos RS232 e I2C de la tarjeta Raspberry Pi es necesario instalar las librerías WiringPi y wiringSerial. WiringPi es una librería de estilo Arduino escrita en C, desarrollada por Gordon Henderson, esta librería facilita el acceso a los cuarenta pines de propósito general de la tarjeta Raspberry Pi, la librería tiene soporte para los puertos UART/RS-232, SPI, I2C, PWM. El proceso de instalación de la librería se realiza siguiendo los pasos que a continuación se describen, esto lo hacemos abriendo la aplicación LXTerminal, preferentemente estar logeado como super usuario (root):

1. Primeramente instalar GIT:

```
1 pi@raspberrypi~$ sudo bash
2 root@raspberrypi~# apt-get install git-core
```

2. Actualizar la tarjeta Raspberry Pi:

```
1 root@raspberrypi~# apt-get update
```

3. Descargar la librería wiringPi :

```
1 root@raspberrypi~# git clone git://git.drogon.net/wiringPi
```

4. Acceder a la carpeta descargada:

```
1 root@raspberrypi~# cd wiringPi
2 root@raspberrypi~# git pull origin
```

5. Acceder a la carpeta interna:

```
1 root@raspberrypi~# cd wiringPi
```

6. Instalar:

```
1 root@raspberrypi~# ./build
```

Configuración de RTC

Ahora procedemos a configurar el módulo de RTC, también es necesario tener abierta la aplicación LXTerminal, preferentemente estar logeado como super usuario (root):

1. Primeramente abrir una terminal en modo root y escribir las siguientes instrucciones:

```
1 pi@raspberrypi~$ sudo bash
2 root@raspberrypi~# sudo modprobe rtc-ds1307
```

2. Una vez completadas las instrucciones anteriores, ejecutamos la siguiente instrucción:

```
1 root@raspberrypi~# echo ds1307 0x68 > /sys/class/i2c-adapter/i2c- 0/new_device
```

3. Ahora ya tenemos configurado el módulo, para verificar la hora del RTC, hacer la petición con la siguiente instrucción:

```
1 root@raspberrypi~# sudo hwclock -r
```

   El resultado de ejecutar ésta instrucción se muestra en la Figura 4.11.

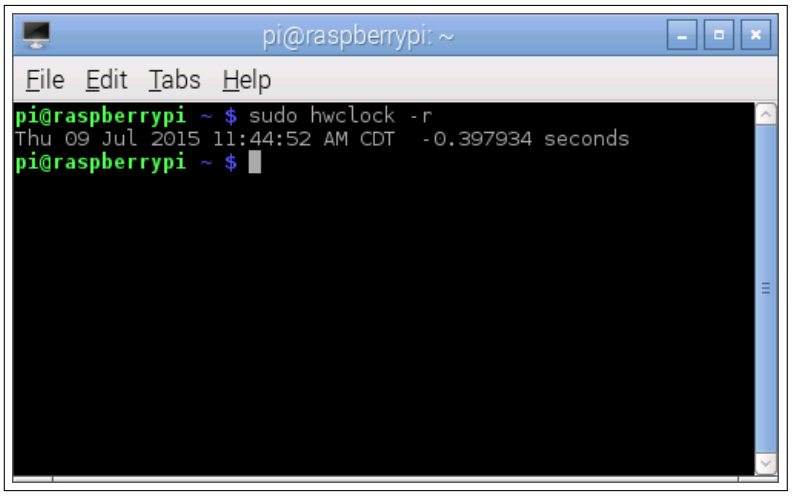


Figura 4.11: Visualización de tiempo con RTC.

4. Si es la primera vez que se utiliza el módulo necesitamos configurar el RTC, para ello se tiene que ingresarle la fecha y hora actual. Primeramente se requiere conectar la

tarjeta Raspberry Pi a Internet para que le actualice automáticamente la fecha y hora actual, posteriormente utilizar el siguiente comando para configurar la hora en el RTC:

```
root@raspberrypi~# sudo hwclock -w
```

Para corroborar que el módulo se ha actualizado con la fecha y hora, es necesario usar el comando anterior para visualizar en la terminal la nueva hora:

```
root@raspberrypi~# sudo hwclock -r
```

5. Si deseamos agregar el RTC a la lista de módulos del núcleo de la tarjeta /etc/modules/", que se carga cada vez que se reinicia la tarjeta Raspberry Pi, necesitamos modificar el archivo de configuración ***modules***, para ello abrimos el archivo con la siguiente instrucción:

```
root@raspberrypi~# sudo nano /etc/modules
```

6. Una vez abierto el archivo ingresar la siguiente línea al final del documento:

```
rtc-ds1307
```

A continuación guardar los cambios presionando las teclas Ctrl + X, seleccionar la opción Guardar, y salir del editor.

7. Ahora debemos editar el archivo ***rc.local*** para agregar el módulo en la lista de procesos a de arranque, para ello editamos el archivo con la siguiente instrucción:

```
root@raspberrypi~# sudo nano /etc/rc.local
```

8. Agregar las siguientes dos líneas antes de la línea de “exit0”:

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c- 1/new_device
sudo hwclock -s
```

A continuación guardar los cambios presionando las teclas Ctrl + X, seleccionar la opción Guardar, y salir del editor.

Con esto se ha configurado el módulo RTC para que con cada reinicio de la tarjeta Raspberry Pi se cargue automáticamente.

Diagrama de flujo del sistema de software

Finalmente y una vez completado la configuración de los puertos en la tarjeta Raspberry Pi, se procede a definir el procedimiento que se seguirá para la programación del sistema de software embebido, ver Figura 4.12. El programa desarrollado se compone de tres módulos internos:

1. Un módulo de funciones que se encargan de la adquisición de datos y procesamiento de los mismos.

2. Un módulo adicional de funciones que corresponde a la ejecución del algoritmo SPA que calcula los ángulos de posicionamiento.

3. Y finalmente otro módulo de funciones que se encargan del despliegue de datos en un LCD y envío de comandos para el posicionamiento del prototipo de seguidor solar.

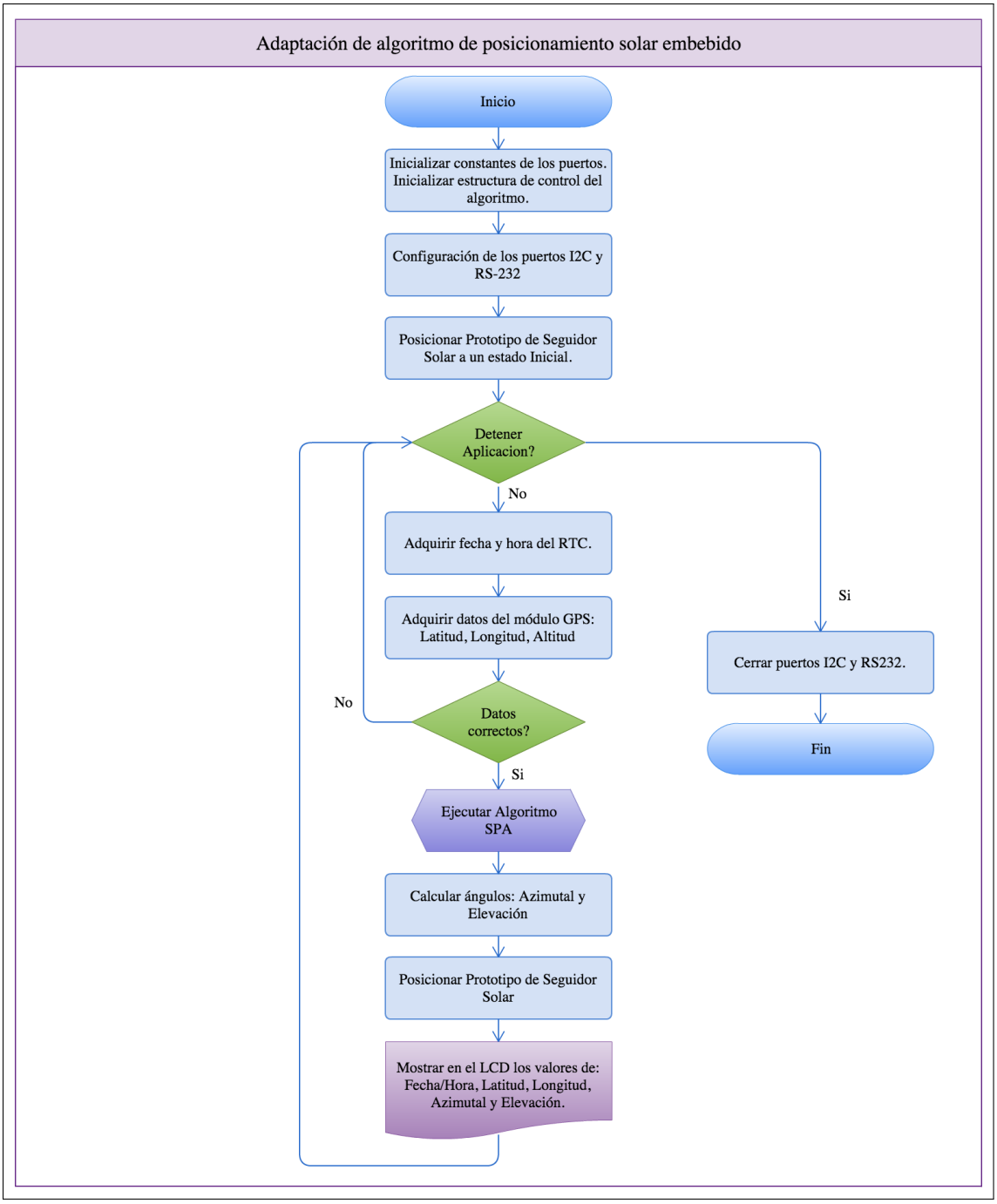


Figura 4.12: Adaptación de algoritmo de posicionamiento solar embebido.

4.1.5. Programación del sistema embebido

Generalmente un sistema de seguimiento solar requiere conocer las coordenadas geográficas, especificado la Latitud y Longitud del punto donde se encuentra instalado, además requiere de un instante de tiempo, dado en fecha y hora universal.

El sistema de software propuesto se basa en el requerimiento de controlar el prototipo de seguidor sin el uso de sensores externos, haciendo uso del algoritmo de posicionamiento solar SPA para calcular la posición del Sol por medio de las ecuaciones antes expuestas. Es importante resaltar que a este algoritmo se le realizaran algunas modificaciones, tales como:

1. Redefinir la estructura de variables locales del algoritmo SPA a variables globales, con el objetivo de tener acceso inmediato a ellas desde cualquier otro archivo *.c.

2. Agregar función que permite asignar los valores de fecha y hora universal proporcionada por el RTC a las variables del algoritmo SPA.

3. Agregar función que permite asignar los valores de Latitud, Longitud y Altitud proporcionados por el GPS a las variables del algoritmo SPA.

4. Modificar la función $spa\_calculate(\&spa)$ del algoritmo para insertar líneas de código que arman los comandos de posicionamiento del prototipo de seguidor solar, que inmediatamente después de calcular los ángulos de ubicación del Sol, se le enviaran al módulo controlador mediante el puerto de comunicaciones RS-232 los comandos de posicionamiento.

Implementación de código

Con las librerías <unistd.h>, <wiringPi>, <wiringSerial.h>y "spa.h"que es un archivo de cabecera perteneciente al algoritmo SPA, se crea el programa en lenguaje C con el nombre de ***SolarTracking.c***. El programa utiliza la función open() para abrir los puertos, setup() para configurar los puertos, read() y write() para leer y escribir, setPos() para posicionar el prototipo de seguidor solar, entre otras.

A continuación se describe paso a paso las funciones del código C, para dar una idea más concreta del funcionamiento del sistema de software embebido.

Descripción del código "SolarTracking"

Los cálculos presentados en este documento se basan en el algoritmo SPA del Laboratorio Nacional de Energías Renovables (NREL) que está clasificado como un algoritmo astronómico, debido al alto grado de precisión.

- Primero se definen las librerías requeridas y necesarias para la compilación y ejecución del programa, de igual manera, se incluye el archivo de cabecera "spa.h"que hace referencia al algoritmo spa, ver Código A.16 listado en el Apéndice A.1.

- De igual manera se definen las variables y se inicializan para el manejo y control del GPS, LCD y RTC, puertos I2C, RS-232 y algoritmo SPA, ver Código A.2 listado en el Apéndice A.1.

    1. Para comunicarse con el módulo GPS se debe configurar el puerto RS-232 con una velocidad de transmisión de ***9600 baudios*** y el nombre del puerto de comunicaciones serial que le corresponde es el ***/dev/ttyAMA0***.
    2. Para comunicarse con el modulo LCD se le debe configurar la dirección ***0x28*** y para el RTC la dirección ***0x68***, ambos con el puerto de comunicaciones I2C ***/dev/i2c-1***.
    3. Para comunicarse con el controlador del prototipo de seguidor solar, se debe configurar el puerto RS-232 conectado al puerto USB de la tarjeta Raspberry Pi, con una velocidad de transmisión de ***19200 baudios*** y el nombre del puerto de comunicaciones serial que le corresponde es el ***/dev/ttyUSB0***.

- A continuación definir la estructura de datos "***spa_data spa***", que contiene las variables utilizadas por el algoritmo SPA, para fines ilustrativos sólo se muestran las variables más importantes, para mayor detalle de esta estructura, consultar el archivo de cabecera en el Apéndice A.2.

- También se define la función ***WriteToMotors (int fd, char *CMD)***, para enviar los comandos al controlador que realiza el movimiento de los ejes Azimutal y Elevación del prototipo de seguidor solar, ver Código A.3 listado en el Apéndice A.1.

- A continuación se define la función ***WriteToLCD(int fd, char *MSJ)***, para escribir un mensaje de texto en el display LCD a través del puerto I2C, ver Código A.4 listado en el Apéndice A.1. Esta función recibe como parámetros dos elementos: la referencia del puerto I2C y el mensaje de texto.

- También se define la función ***CleanToLCD (int fd)***, para construir y enviar un comando a través del puerto I2C que tiene como objetivo limpiar la información visualizada en el display LCD, ver Código A.5 listado en el Apéndice A.1. Esta función recibe como parámetro la referencia del puerto I2C.

- Se define otra función llamada ***SetCursor(int fd, int column, int row)***, para construir y enviar un comando a través del puerto I2C que tiene como objetivo posicionar el cursor en una coordenada (x,y) en el display LCD, ver Código A.6 listado en el Apéndice A.1. Esta función recibe como parámetro la referencia del puerto I2C.

- La función ***CurrentTime(void)***, ver Código A.7 listado en el Apéndice A.1, obtiene el tiempo universal, proporciona la fecha y hora exacta, posteriormente asigna estos valores a la estructura del algoritmo SPA, actualizando las variables de spa.year, spa.month, spa.day, spa.hour, spa.minute y spa.second, además, envía estos valores a la función WriteToLCD() para el despliegue de estos valores en el display LCD.

- De igual manera se define la función ***splitBuffer(char *tramaGPG)***, esta función recibe como parámetro de entrada una trama de Bytes, la cual para analizarla y la dividirla en fragmentos con base a un token, en este caso es una coma (,), cada fragmento representa una valor que el módulo GPS transmite, ver Código A.9 listado en el Apéndice A.1. Es importante señalar que el módulo GPS-MTK3339 soporta 6 sentencias en formato NMEA-0183 relacionadas con la información que entrega el GPS, y son las siguientes:

  1. **$ GPGGA (default ON)**.
  2. $ GPVTG (default OFF).
  3. $ GPRMC (default ON).

4. $ GPGSA (default ON).
5. $ GPGSV (default ON, 0.2Hz).
6. $ GPGLL (default OFF).

El protocolo NMEA - National Marine Electronics Association, es una especificación combinada eléctrica y de datos entre aparatos electrónicos marinos, generalmente receptores GPS. Al momento de encender el módulo GPS, este dispositivo automáticamente envía el primer paquete de Bytes en la cual se podrá apreciar la siguiente trama de datos, ver Código A.8 listado en el Apéndice A.1.

- La función ***ReadGPS(void)*** obtiene la trama de Bytes de datos del puerto de comunicaciones serial RS-232, ver Código A.10 listado en el Apéndice A.1. Una vez obtenida la trama de Bytes de datos, se la envía a la función ***splitBuffer(char *tramaGPG)*** encargada de descifrar el contenido.

- El programa requiere ejecutar dos funciones a la vez, para ello se crean dos hilos de ejecución (thread): el primer hilo ***PI_THREAD (executeSPA)*** se encarga de obtener la fecha y hora actual, leer las coordenadas GPS, y ejecutar la función ***executeSolarTracking()*** que hace referencia al algoritmo SPA y posiciona el seguidor solar, ver Código A.11 listado en el Apéndice A.1. El segundo hilo ***PI_THREAD (displayData)*** se encarga de mostrar al usuario la fecha, la hora, las coordenadas GPS, y los ángulos de posición del Sol, mediante un LCD, ver Código A.12 listado en el Apéndice A.1.

- A continuación se define la función principal del sistema de software embebido ***main (int argc, char *argv[])***, ver Código A.13 listado en el Apéndice A.1, su propósito es iniciar la ejecución del sistema. Por sintaxis y requerimiento del lenguaje de programación C, esta función está preparada para recibir argumentos, tales como: el número de argumentos y los valores de cada argumento, pero es importante aclarar que esta función no utiliza estos argumentos, por lo tanto se hace caso omiso de ello, y no infiere en el funcionamiento del sistema de software. Aquí principalmente se configuran y abren los puertos I2C, RS232, también se crean y ejecutan los dos hilos: executeSPA y displayData. Como ya se mencionó anteriormente, la configuración básica de cada puerto son las siguientes:

1. Para los puertos seriales RS-232 son:

   Nombre del puerto = /dev/ttyAMA0 y Baud rate = 9600.

   Nombre del puerto = /dev/ttyUSB0 y Baud rate = 19200.

2. Para el puerto I2C:

   Nombre del puerto = /dev/i2c-1, la dirección para el LCD = 0x28.

   Y la dirección para el RTC = 0x68.

- Finalmente se define la función ***executeSolarTracking(void)***, ver Código A.14 listado en el Apéndice A.1, con el propósito de preparar y configurar los valores de los parámetros de entrada que requiere el algoritmo de posición solar (SPA), dentro de esta función se realiza la llamada a la función ***spa_algorithm(dDate, dTime, dGPS, timezone)*** para obtener los valores de los ángulos de posición, posteriormente son transmitidos mediante comandos al controlador del prototipo de seguidor solar.

Inclusión del algoritmo SPA

Como ya se mencionó anteriormente en el capítulo 3 referente a la metodología, el algoritmo de posición solar (SPA) determina la posición del sol a partir del cálculo de los ángulos de Elevación y Azimutal en el período comprendido entre el año -2000 hasta el año 6000, con una incertidumbre de +/- 0.0003°.

A continuación, el algoritmo SPA (Reda and Andreas, 2008b) se presenta de forma simplificada, introduciendo algunos cambios para adaptarlo y embeberlo en el código fuente del desarrollo del presente sistema de software, para ello es necesario incluir el archivo de encabezado ***spa.h***, ver Apéndice A.2, al inicio del código del programa principal, además se requiere realizar el llenado de los parámetros de entrada al algoritmo, para posteriormente ejecutar la función de cálculo de SPA: ***char *spa_algorithm((int dDate[3], int dTime[3], double dGPS[3], int timezone))***.

- La implementación del algoritmo de posición solar (SPA) se ejecuta a partir de la función ***spa_algorithm()***, ver Código A.17 listado en el Apéndice A.3, esta función determina la posición del sol en un momento dado para una ubicación específica. Como parámetros de entrada al algoritmo son: la fecha, la hora, la posición GPS y la zona

horaria. La hora para un instante de tiempo, considerando la fecha: en año, mes y día, y el tiempo universal en: horas, minutos y segundos. El lugar está dado por la posición GPS con datos de Latitud, Longitud y Altitud en grados, donde la Latitud se considera positiva hacia el Norte y la Longitud positiva hacia el Este y la zona horaria para el centro del país de México le corresponde -6. Debido a la complejidad del algoritmo el resto del código fuente escrito en lenguaje C se ha incluido en el Apéndice A.3.

4.1.6. Pruebas y puesta en marcha del sistema

Las pruebas de funcionamiento del sistema de software se dividen en las pruebas individuales a cada componente, verificando la comunicación entre la tarjeta Raspberry Pi con los módulos de RTC, LCD, GPS, Controlador del prototipo de seguidor solar, además de las pruebas del Algoritmo de Posición Solar SPA.

Pruebas de sistema por módulos

La primera prueba consiste en verificar que la información trasmitida por el módulo GPS corresponda aproximadamente a las coordenadas GPS reales del sitio de prueba, por los que se recurre a corroborar con las coordenadas GPS que Google Maps proporciona, ver Figura 4.13:

Para llevar a cabo la prueba de adquisición de puntos geográficos GPS utilizando el módulo GPS se realizó el siguiente procedimiento:

1. Verificar las alimentaciones tanto de la tarjeta Raspberry Pi con 5 volts, como el módulo GPS con 3.3 volts.

2. Para la prueba del módulo1 GPS se requiere que la antena se encuentre en un lugar donde reciba las señales entregadas por los satélites.

3. Una vez que se verificaron las conexiones, se procede a ejecutar el programa que monitorea la recepción de datos del módulo GPS.

4. El software captura de datos entregados por el módulo GPS, verifica si el dato recibido corresponde a una sentencia GPGGA para procesar la información, visualiza en panta-

lla los valores de Latitud, Longitud y Altitud, además envía estos valores al algoritmo SPA.

(a) Coordenadas GPS de la ubicación de CIDESI calculadas por Google Maps.

(b) Coordenadas GPS de la ubicación de CIDESI calculadas por el Módulo GPS.

Figura 4.13: Validación de coordenadas GPS del sitio de pruebas.

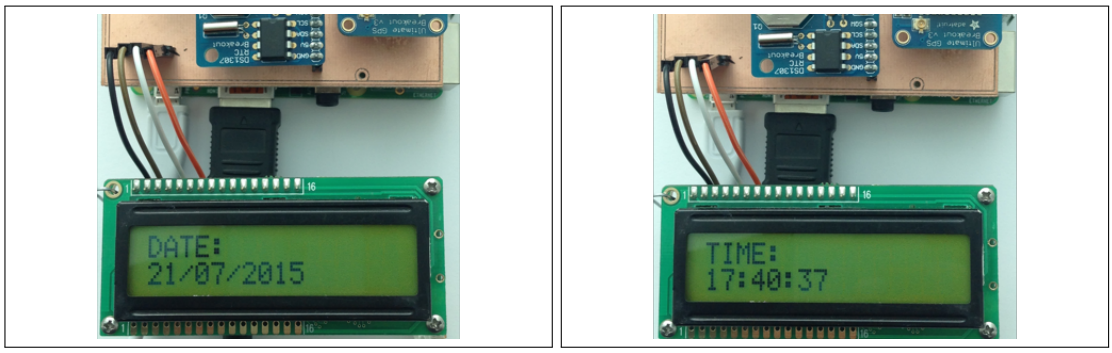
Al finalizar la prueba de verificación de coordenadas GPS proporcionadas por el módulo, se observa que los valores de Latitud y Longitud tienen una diferencia de +/- 0.249749 grados con respecto al valor de Latitud proporcionado por Google Maps, y para el valor de Longitud existe una diferencia de +/- 0.161697 grados. Este error en los puntos GPS puede llegar a inferir en el cálculo de la posición del Sol, por lo tanto, se recomienda que para aplicaciones futuras sea reemplazado el módulo GPS por otro que proporcione los puntos GPS más exactos.

La siguiente prueba es validar el funcionamiento de los módulos RTC y LCD, para ello se realiza el siguiente procedimiento:

1. Verificar las alimentaciones tanto del módulo RTC, como el módulo LCD con 3.3 volts y 5.0 volts respectivamente.

2. Una vez que se verificaron las conexiones, se procede a ejecutar el programa de software que obtiene la fecha y hora del RTC, al obtener estos valores se valida la prueba del funcionamiento del módulo.

3. De igual manera, para validar el funcionamiento del LCD, el software envía los valores de fecha y hora al módulo LCD, los valores son visualizados en el display LCD, ver Figura 4.14, además el programa envía estos valores al algoritmo SPA.



(a) Fecha obtenida del RTC y mostrada en el display LCD.

(b) Hora obtenida del RTC y mostrada en el display LCD.

Figura 4.14: Validación de los módulos RTC y LCD.

Otra de las pruebas es validar el funcionamiento del módulo controlador de los dos ejes del prototipo de seguidor solar, para ello se realiza el siguiente procedimiento:

1. Verificar las alimentaciones tanto del módulo controlador, como de los dos motores con 110.0 volts.

2. Una vez que se verificaron las conexiones, se procede a ejecutar el programa de software que envía comandos de posición al controlador.

3. Para validar el funcionamiento del controlador, el software envía los siguientes valores de prueba referente a los ángulos de posición: para el eje Azimutal se envía el ángulo de posición con valor de 323.73 º y para el eje de Elevación se envía el ángulo de posición con valor de 62.8254 º, respectivamente, ver Figura 4.15. Posteriormente estos ángulos serán reemplazos por los ángulos calculados por el algoritmo SPA.

(a) Valores mostrados en el display del controlador.

(b) Vista general de la posición del seguidor.

Figura 4.15: Validación del módulo controlador de los dos ejes del seguidor solar.

Finalmente se realiza la prueba de ejecución del Algoritmo de Posición Solar (SPA), es importante resaltar que esta prueba requiere de los módulos RTC, LCD, GPS y Controlador funcionando en conjunto, por los que da lugar a una integración de todos los módulos antes mencionados, ya que el algoritmo es dependiente de estos, debido a que requiere de parámetros de entrada que cada módulo gestiona, tales como: la fecha y hora, los valores de posición geográfica GPS, y como salida, este algoritmo proporciona los ángulos de posición Azimutal y Elevación, ver Figura 4.16, dichos ángulos son enviados al controlador para posicionar los dos ejes del prototipo de seguidor solar. Para ello se realiza el siguiente procedimiento:

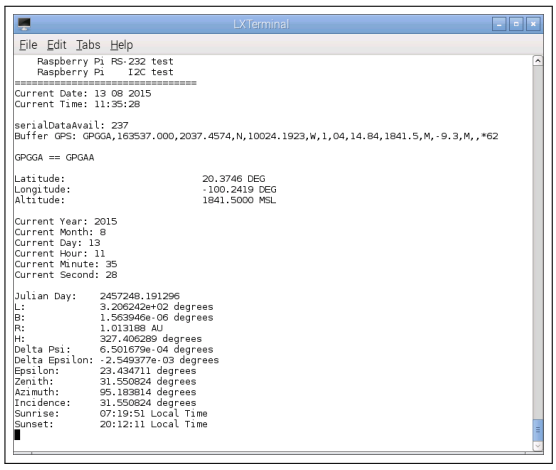
1. Primeramente, el módulo GPS envía al software ***SolarTracking.c*** embebido en la tarjeta Raspberry Pi, su posición GPS, tales como: Latitud, Longitud y Altitud.

2. De igual manera, el módulo RTC (Reloj de Tiempo Real) provee a la tarjeta Raspberry Pi la hora de reloj.

3. A continuación el software a través de la colección de funciones que se programaron, recibe los datos, los procesa y los envía al algoritmo que calcula los ángulos de posición del Sol (Azimutal y Elevación).

4. Posteriormente el software se encarga de enviar estos ángulos al controlador de los dos ejes mediante motores para posicionar el prototipo de seguidor solar.

5. Finalmente el mismo programa de software muestra en un display LCD la información de: fecha/hora, coordenadas GPS, ángulos de posición Azimutal y Elevación.



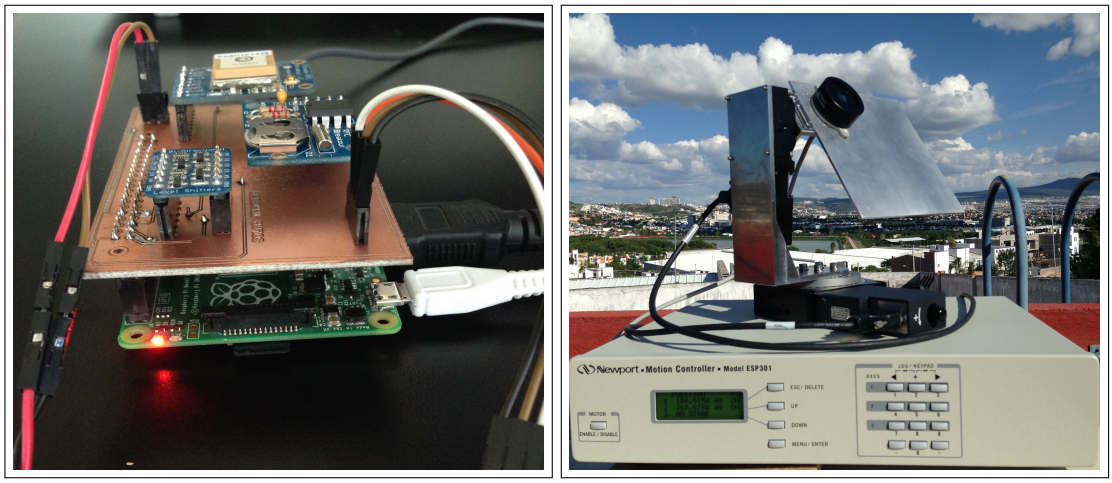
(a) Datos arrojados por el algoritmo SPA.

(b) Vista general de los módulos que interactúan con el sistema de software.

Figura 4.16: Prueba de integración y ejecución del sistema de software.

Pruebas del sistema completo

El sistema de software embebido en la tarjeta Raspberry Pi fue sometido a un periodo de trabajo de 24 horas diarias por 15 días, almacenando los valores de fecha/hora, ángulo Azimutal y ángulo de Elevación con un intervalo de tiempo de 5 minutos por muestra, para garantizar que el software no sature la memoria de la tarjeta, la prueba se realizó sin contratiempos y al final se obtuvieron los datos satisfactoriamente, ver Figura 4.17.



(a) Tarjeta Raspberry Pi y módulos **(b)** Prototipo seguidor solar

Figura 4.17: Vista general de la prueba del sistema completo.

En las pruebas fue necesario realizar ajustes por software a los ángulos de Azimutal y Elevación, finalmente se obtienen los siguientes valores, ver Tabla A.1 listada en el Apéndice A.4.

De igual manera, durante las pruebas hubo nubosidad, en ocasiones se presentó lluvia, debido a estas condiciones los módulos NO presentaron comportamientos variables a lo largo del día. En la Figura 4.18 y la Figura 4.19 se muestra el rendimiento general del sistema, se observa que aun con las condiciones meteorológicas adversas durante las pruebas, el sistema produjo sin ningún problema los ángulos de la trayectoria del Sol.

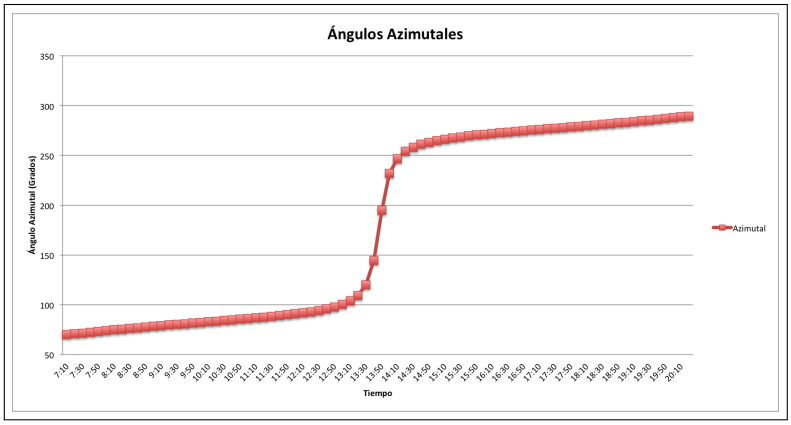


Figura 4.18: Ángulos Azimutales arrojados en las pruebas del sistema de software.

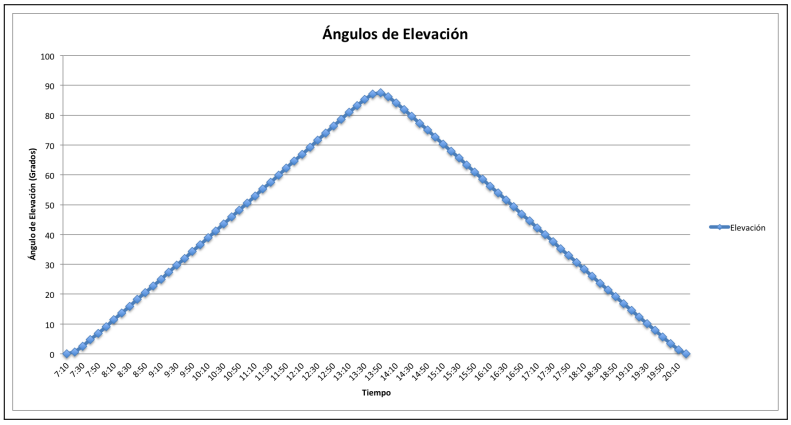
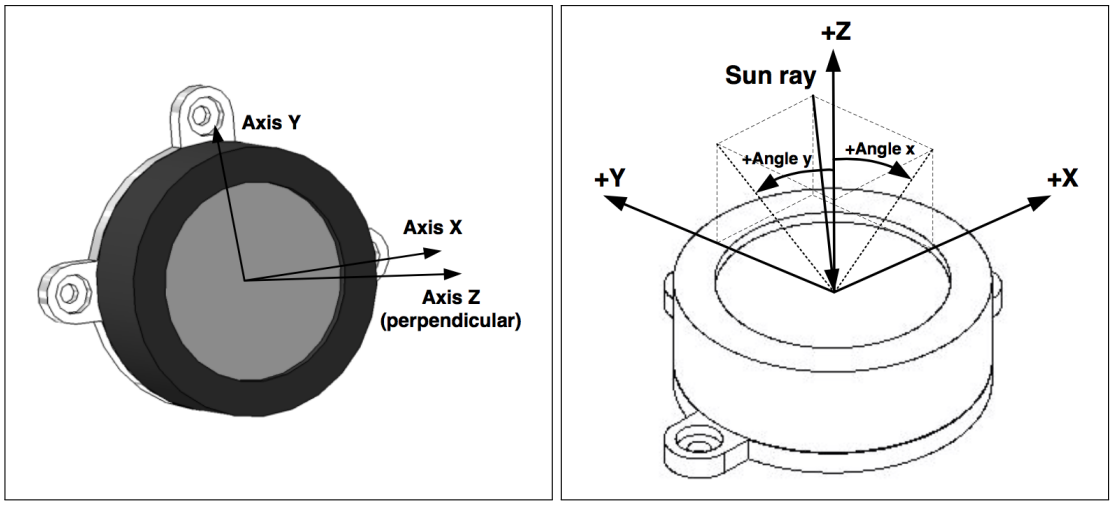


Figura 4.19: Ángulos de Elevación arrojados en las pruebas del sistema de software.

5. RESULTADOS

Finalmente se realiza la prueba de validación y funcionamiento del software y Algoritmo de Posición Solar (SPA), para dicha validación fue necesario la utilización de un instrumento óptico de diagnóstico que mide con precisión la posición del Sol, a fin de evaluar los ángulos de posición Azimutal y Elevación que calcula el algoritmo. Así mismo se comparan estos ángulos con sistemas de seguimiento disponibles en la red de Internet (Van der Staay, 2015; LRPS, 2011; Andreas and Reda, 2011).

El instrumento utilizado fue el sensor de Sol de la serie de modelos Solar MEMS ISS-T60 cuyo funcionamiento es proporcionar mediciones ópticas precisas de la incidencia solar por medio de vectores incidentes de rayos de luz cuando entran por su ventana de visión, ver Figura 5.1. Utiliza un sustrato de alta precisión tipo circuito integrado de detección para medir los ángulos incidentes de los rayos del Sol (S.L., 2015).



(a) Sistema de referencia. **(b)** Referencia para los ángulos medidos.

Figura 5.1: Sistema de referencia del sensor Solar MEMS ISS-T60.

Estas características hacen un dispositivo adecuado para sistemas de navegación en vehículos no tripulados, sistemas de seguimiento solar, por ejemplo en control de altitud del avión, control de helióstatos, control de altitud utilizando fuentes de luz, así como para medir los niveles de radiación solar de posicionamiento de alta precisión.

El ISS -TX hace un procesamiento de filtrado interno para las mediciones de ángulos con las siguientes características: filtro electrónico de tercer orden de Butterworth con frecuencia de muestreo de 50 Hz y frecuencia de corte de 0.4 Hz .

El ángulo X y el ángulo Y especifican la posición angular del rayo incidente del Sol dentro del campo de visión del sensor ISS -T60, ambos ángulos se proporcionan en grados. La precisión del sensor aumenta cuando se aproxima a cero grados (perpendicular). Las características con las que opera el sensor son: campo de vista de 120º, exactitud menor al 10 % (3 σ), precisión menor a 0.06º de sensibilidad con un ángulo de resolución de 0.01º.

Por lo tanto, el primer experimento consistió en probar la exactitud de los ángulos arrojados por el sistema de software desarrollado para seguimiento del Sol embebido en la tarjeta Raspberry Pi (SolarTracking) con respecto a las mediciones del sensor Solar MEMS ISS-T60, los resultados obtenidos se muestran en la Tabla 5.1, en la cual se observa que los ángulos X y Y son aproximados a un ángulo de cero grados, estos resultados demuestran que el sistema de software para seguimiento solar opera de manera eficiente y confiable con una precisión de +/- 1.0632 grados para el ángulo X, y para el ángulo Y de +/- 0.1247 grados:

Tabla 5.1: Resultados de ángulos de SolarTracking con respecto a Solar MEMS ISS-T60.

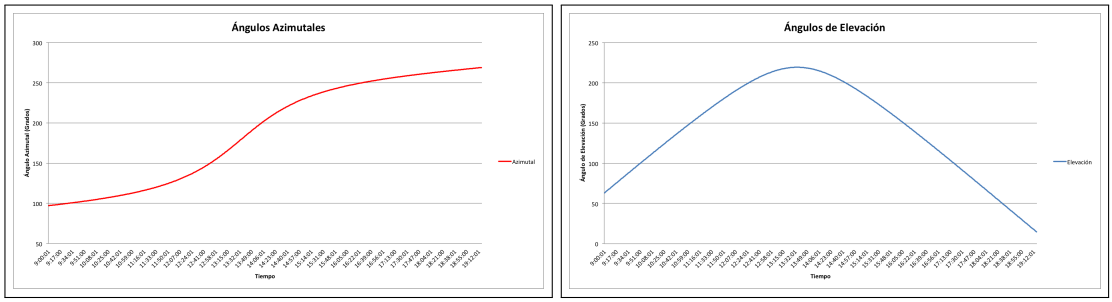
Fecha/Hora	**SolarTracking**		**Solar MEMS ISS-T60**	
21/09/2015	Azimutal (grados)	Elevación (grados)	Ángulo X (grados)	Ángulo Y (grados)
9:00:01	97.1472	63.2488	0.6600	0.0400
9:17:00	98.8734	75.5515	0.9600	-0.0200
9:34:01	100.7086	87.8247	1.1400	0.0600
9:51:00	102.6741	100.0009	1.5000	-0.3200
10:08:01	104.8128	112.1053	1.8500	-0.2800
10:25:00	107.1599	124.0629	1.8100	-0.3700
10:42:01	109.7800	135.8852	1.6800	-0.1900
10:59:00	112.7341	147.4798	1.8200	-0.0700
11:16:01	116.1261	158.8301	1.7200	-0.0270
11:33:00	120.0632	169.8070	1.9600	-0.1300
11:50:01	124.7174	180.3361	2.0500	-1.3500
12:07:00	130.2686	190.2117	1.4900	-0.0520
12:24:01	136.9763	199.2421	1.0000	-0.1500

Continúa en la siguiente página...

Tabla 5.1: ...Continuación de la página anterior.

Fecha/Hora	SolarTracking		Solar MEMS ISS-T60	
12:41:00	145.0606	207.0771	1.2500	0.7700
12:58:01	154.7242	213.3396	1.5400	0.6600
13:15:00	165.8629	217.5405	1.5700	0.5900
13:32:01	178.0437	219.2806	1.5300	0.6200
13:49:00	190.3505	218.3506	1.8700	0.6100
14:06:01	201.8945	214.8579	1.6800	0.9200
14:23:00	212.0186	209.1833	1.6200	1.0500
14:40:01	220.5849	201.7701	1.5200	1.0100
14:57:00	227.6782	193.0782	1.5600	1.6000
15:14:01	233.5626	183.4180	1.5300	1.4000
15:31:00	238.4601	173.0810	0.9600	1.3100
15:48:01	242.6015	162.2095	0.9700	1.1600
16:05:00	246.1392	150.9760	0.9100	0.9100
16:22:01	249.2179	139.4299	0.8800	0.5400
16:39:00	251.9254	127.6845	0.9000	0.1800
16:56:01	254.3497	115.7441	0.4500	-0.2000
17:13:00	256.5406	103.6937	0.8200	-0.2100
17:30:01	258.5539	91.5157	0.5800	0.1600
17:47:00	260.4190	79.2817	0.2000	-0.2900
18:04:01	262.1734	66.9636	-0.1900	0.5000
18:21:00	263.8356	54.6283	-0.5300	0.4100
18:38:01	265.4332	42.2473	-1.6500	0.2700
18:55:00	266.9786	29.8976	-2.0600	1.1900
19:12:01	268.4943	17.5957	-2.3600	1.3200

La Figura 5.2 se muestran las tendencias de los ángulos Azimutal y Elevación del sistema de software SolarTracking durante las pruebas de validación de exactitud en el seguimiento de la trayectoria del Sol.



(a) Tendencia del ángulo Azimutal. **(b)** Tendencia del ángulo de Elevación.

Figura 5.2: Prueba de exactitud de los ángulos arrojados por el sistema de software.

De igual manera en la Figura 5.3 se muestran las tendencias de los ángulos X, Y calculados por el sensor Solar MEMS ISS-T60 durante las pruebas de validación de exactitud en el seguimiento de la trayectoria del Sol.

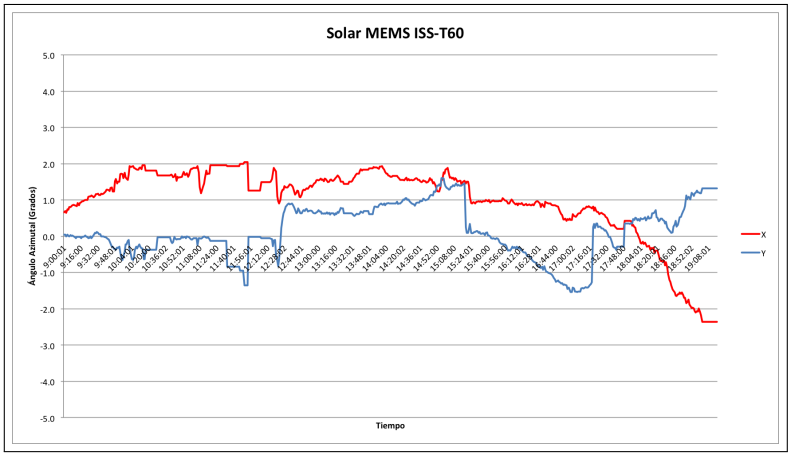


Figura 5.3: Tendencia de los ángulos dados por el sensor Solar MEMS ISS-T60.

Otro de los experimentos es comprar los resultados de los ángulos calculados por el sistema de software SolarTracking con respecto a los ángulos de Azimutal y Elevación arrojados por sistemas de seguimiento solar disponibles en la red de Internet. A continuación se muestra un resumen de los resultados obtenidos en las pruebas, ver Tabla 5.2.

Esta prueba se realizó el 20 de Agosto de 2015, se puede observar que a las 13:45:00 horas el algoritmo llega el ángulo de Elevación a su punto más alto siendo de 87.23 º, por lo que se atribuye a la altura máxima del Sol, posterior a esa hora, el ángulo de Elevación decrementa.

Tabla 5.2: Comparación de ángulos de posición solar.

Fecha/Hora	**SolarTracking**		**Dom calculadora**		**Solar Calculator**		**MIDC SPA Calc.**	
20/08/2015	Azimutal (grados)	Elevación (grados)	Azimutal (grados)	Elevación (grados)	Azimutal (grados)	Elevación (grados)	Azimutal (grados)	Elevación (grados)
7:15	75.7969	-2.3457	75.6988	-1.8289	75.7000	-1.4600	75.7000	-2.5026
7:30	77.1155	1.4213	77.0350	1.2799	77.0400	1.2700	77.0362	1.2732
7:45	78.3886	4.6774	78.3254	4.5155	78.3300	4.5100	78.3266	4.5119
8:00	79.6248	8.0651	79.5788	7.8932	79.5800	7.8900	79.5800	7.8906
8:15	80.8324	11.4987	80.8033	11.3192	80.8000	11.3100	80.8046	11.3172
8:30	82.0193	14.9566	82.0073	14.7705	82.0100	14.7700	82.0086	14.7687
8:45	83.1949	18.4343	83.1988	18.2384	83.2000	18.2400	83.2001	18.2368
9:00	84.3662	21.9241	84.3861	21.7185	84.3900	21.7200	84.3874	21.7171
9:15	85.5407	25.4195	85.5781	25.2079	85.5800	25.2100	85.5795	25.2066
9:30	86.7291	28.9218	86.7846	28.7044	86.7900	28.7000	86.7860	28.7032
9:45	87.9436	32.4329	88.0164	32.2061	88.0200	32.2000	88.0179	32.2050
10:00	89.1942	35.9438	89.2863	35.7111	89.2900	35.7100	89.2878	35.7100
10:15	90.4971	39.4561	90.6095	39.2176	90.6100	39.2200	90.6111	39.2166
10:30	91.8715	42.9683	92.0049	42.7236	92.0100	42.7200	92.0066	42.7226
10:45	93.3406	46.4773	93.4964	46.2267	93.5000	46.2300	93.4982	46.2258
11:00	94.9358	49.9816	95.1152	49.7242	95.1200	49.7200	95.1172	49.7233
11:15	96.7009	53.4803	96.9034	53.2125	96.9100	53.2100	96.9055	53.2116
11:30	98.7025	56.9279	98.9193	56.6869	98.9200	56.6900	98.9216	56.6860
11:45	100.9936	60.4284	101.2464	60.1406	101.2500	60.1400	101.2489	60.1397
12:00	103.7350	63.8643	104.0091	63.5638	104.0100	63.5600	104.0118	63.5628
12:15	107.1101	67.2554	107.3995	66.9411	107.4000	66.9400	107.4025	66.9402
12:30	111.4352	70.5731	111.7283	70.2474	111.7300	70.2500	111.7316	70.2464
12:45	117.2682	73.7800	117.5204	73.4384	117.5200	73.4400	117.5241	73.4373
13:00	125.5670	76.7836	125.6913	76.4307	125.7000	76.4300	125.6951	76.4295
13:15	138.0220	79.4146	137.8003	79.0585	137.8000	79.0600	137.8038	79.0572
13:30	156.8764	81.3259	155.9069	80.9955	155.9100	80.9900	155.9085	80.9941

Continúa en la siguiente página...

Tabla 5.2: ...Continuación de la página anterior.

Fecha/Hora	SolarTracking		Dom calculadora		Solar Calculator		MIDC SPA Calc.	
13:45	181.9405	81.9834	179.9956	81.7395	179.9900	81.7400	179.9934	81.7381
14:00	205.8397	80.8890	204.0686	80.9897	204.0600	80.9900	204.0632	80.9885
14:15	223.7206	78.9382	222.1481	79.0488	222.1400	79.0500	222.1420	79.0478
14:30	235.6203	76.3026	234.2370	76.4187	234.2300	76.4200	234.2315	76.4180
14:45	243.6084	73.2852	242.3953	73.4249	242.3900	73.4200	242.3905	73.4244
15:00	249.1798	70.0804	248.1795	70.2328	248.1800	70.2300	248.1753	70.2324
15:15	253.3332	66.7494	252.5029	66.9255	252.5000	66.9200	252.4993	66.9251
15:30	256.5962	63.3490	255.8894	63.5472	255.8900	63.5500	255.8862	63.5468
15:45	259.2602	59.9068	258.6489	60.1229	258.6500	60.1200	258.6460	60.1226
16:00	261.5088	56.4365	260.9733	56.6681	260.9700	56.6700	260.9707	56.6677
16:15	263.4586	52.9513	262.9867	53.1925	262.9800	53.1900	262.9843	53.1921
16:30	265.1918	49.4531	264.7725	49.7028	264.7700	49.7000	264.7703	49.7024
16:45	266.7657	45.9428	266.3890	46.2038	266.3900	46.2000	266.3869	46.2034
17:00	268.2169	42.4312	267.8781	42.6990	267.8800	42.7000	267.8761	42.6986
17:15	269.5784	38.9134	269.2710	39.1912	269.2700	39.1900	269.2691	39.1908
17:30	270.8714	35.3954	270.5916	35.6827	270.5900	35.6800	270.5898	35.6823
17:45	272.1139	31.8792	271.8587	32.1756	271.8600	32.1700	271.8570	32.1750
18:00	273.3200	28.3699	273.0875	28.6716	273.0900	28.6700	273.0859	28.6710
18:15	274.5024	24.8661	274.2908	25.1725	274.2900	25.1700	274.2892	25.1718
18:30	275.6723	21.3691	275.4794	21.6803	275.4800	21.6800	275.4778	21.6796
18:45	276.8401	17.8773	276.6630	18.1973	276.6600	18.1900	276.6615	18.1963
19:00	278.0116	14.4027	277.8505	14.7262	277.8500	14.7200	277.8491	14.7250
19:15	279.1968	10.9445	279.0503	11.2715	279.0500	11.2700	279.0488	11.2701
19:30	280.4035	7.5124	280.2703	7.8420	280.2700	7.8400	280.2689	7.8401
19:45	281.6411	4.1286	281.5187	4.4612	281.5200	4.4600	281.5173	4.4581
20:00	282.9165	0.8899	282.8039	1.2257	282.8000	1.2200	282.8024	1.2294
20:15	284.2368	-2.8490	284.1344	-2.5105	284.1300	-2.5100	284.1330	-2.5172

De manera general, el sistema de software SolarTracking produjo un error promedio de 0.27770 y 0.00658 grados en los ángulos de azimutal y elevación con respecto a los algoritmos de Dom calculadora (Van der Staay, 2015), Solar Calculator (LRPS, 2011) y MIDC SPA Calculator (Andreas and Reda, 2011), es importante señalar que este margen de error puede atribuirse a la resolución de decimales en los parámetros de entrada para cada

sistema de seguimiento, tales como los puntos GPS de Latitud y Longitud, adicional a ello, también se debe a la resolución de decimales en los resultados de los ángulos calculados por los sistemas de seguimiento.

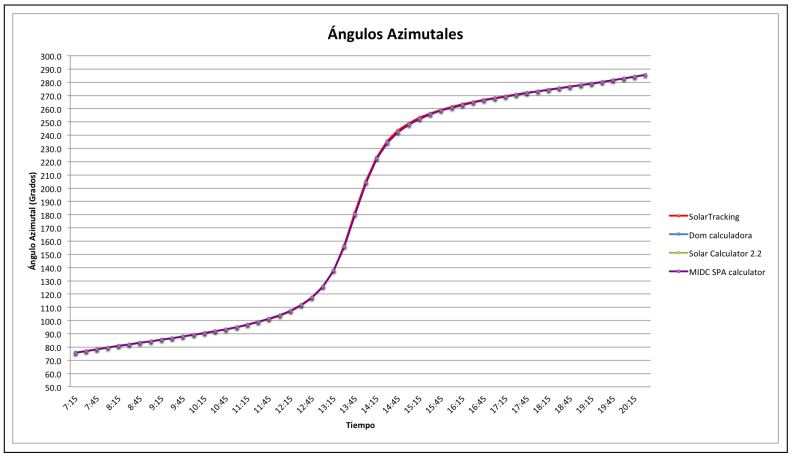


Figura 5.4: Resultados de ángulos Azimutales dados por el algoritmo SPA.

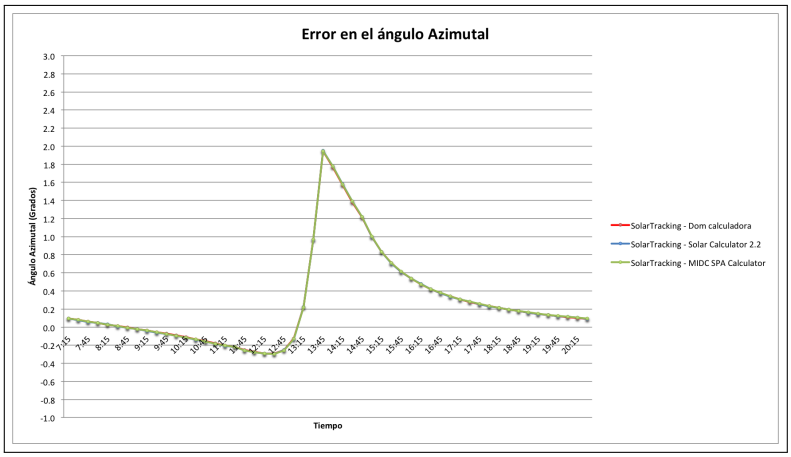


Figura 5.5: Tendencia del error en el ángulo Azimutal dado por el algoritmo SPA.

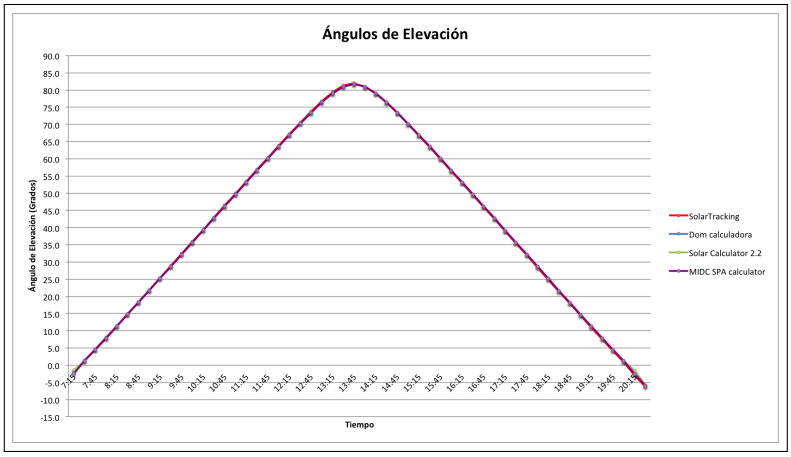


Figura 5.6: Resultados de ángulos de Elevación dados por el algoritmo SPA.

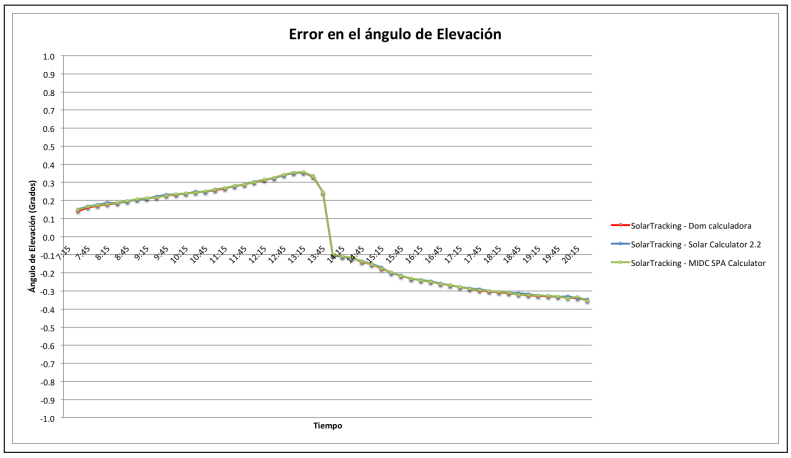


Figura 5.7: Tendencia del error en el ángulo de Elevación dado por el algoritmo SPA.

Los resultados mostrados en la Figura 5.4, Figura 5.5, Figura 5.6 y la Figura 5.7 muestran la tendencia de los ángulos de posición del Sol referente a los resultados obtenidos en la pruebas. Estas gráficas dan una idea del funcionamiento y efectividad de los algoritmos astronómicos operando independientemente de las condiciones climáticas que se presentan durante largos periodos en el año, para el cálculo de la posición del sol, siendo una buena opción para ser considerados en proyectos donde se requiera de seguimiento solar.

Al concluir el presente proyecto de investigación, y una vez finalizado y montado el diseño electrónico del prototipo, los resultados son los siguientes:

El sistema de software principal que contiene el algoritmo de posicionamiento solar embebido en la tarjeta Raspberry Pi, calcula el movimiento que debe realizar el seguidor en base a lo siguiente:

1. El software obtiene las coordenadas geográficas a través de un dispositivo GPS a partir de un sub programa previamente programado y que forma parte del sistema de software principal.

2. El sistema de software obtiene el tiempo universal en formato fecha/hora que en conjunto con las coordenadas geográficas son realimentados al algoritmo de cálculo de posición solar.

3. El software ejecuta el algoritmo, calcula y retorna la posición solar, proporcionando los ángulos de Azimutal y Elevación.

4. El sistema de software embebido despliega la información referente a la trayectoria (ángulo de ubicación del sol) mediante un LCD que es incluido en el prototipo electrónico.

5. El sistema de software embebido envía los comandos que incluyen los ángulos de Azimutal y Elevación para activar y accionar los motores que ajustan la orientación e inclinación del seguidor.

Se realizó la validación de los ángulos Azimutal y Elevación generados por el sistema de software embebido con respecto a la posición real del Sol con sistemas de seguimiento solar disponibles en la red de Internet, de igual manera se validó con un sensor Solar que proporciona los ángulos de error referente a los rayos del Sol.

Se cumplió la hipótesis planteada al inicio de este proyecto, obteniendo un sistema de software que implementa un algoritmo de seguimiento solar corriendo en una tarjeta Raspberry Pi.

El seguidor solar se orientó al Este-Oeste mediante un instrumento de orientación, es decir con ayuda de una brújula se posicionó el prototipo para una configuración inicial.

El sistema de seguimiento es completamente automático a lo largo del día, intercalando un tiempo programado para el reposicionamiento.

5.1. Discusión

En este trabajo de investigación se seleccionó y se probó un algoritmo de posicionamiento solar basado en el modelo de la trayectoria del Sol; a diferencia de otros trabajos que utilizan algoritmos de posicionamiento solar basados en sensores.

La ventaja del algoritmo de posicionamiento solar embebido es que el prototipo seguidor solar seguirá la trayectoria del Sol aun cuando no esté visible por las condiciones atmosféricas y climatológicas.

En la actualidad la utilización de energías renovables para transformar la energía eléctrica es técnicamente viable y económicamente rentable, de hecho una de las líneas de investigación de CIDESI es incursionar en esta área, orientando sus investigaciones a esta promisoria fuente de generación.

5.2. Aplicaciones y uso del proyecto

Los resultados de este trabajo de investigación permitirán utilizar el algoritmo de posicionamiento solar embebido en un tarjeta Raspberry Pi para nuevos sistemas de seguimiento solar, como seguidores de concentración de energía fotovoltaica, estaciones meteorológicas, seguidores con paneles solares, seguidores de dish-stirling, etc., cualquier sistema que requiera de un controlador de seguimiento de Sol, permitiendo reducir los costos en el sistema de software de seguidores solares.

El prototipo es capaz de seguir el Sol en su trayectoria diaria, apuntando el plano horizontal del seguidor directamente al Sol con el objetivo de captar la mayor cantidad de energía, y así poder ser aprovechada por aplicaciones que requieren de esta energía solar, como celdas solares, colectores solares, y hornos solares.

6. CONCLUSIONES

En el presente trabajo de investigación se implementó, adaptó y probó un algoritmo de posicionamiento solar con un prototipo funcional de seguidor solar para demostrar la orientación al Sol.

Se analizaron los diferentes tipos de seguidores en el mercado obteniendo características de cada uno para su análisis, se investigó la trayectoria aparente del Sol, además de que se investigó la trayectoria de seguimiento de cada seguidor de acuerdo a su número de ejes de rotación para obtener una idea del algoritmo de control que se debe elegir.

El entendimiento de las ecuaciones para obtener la trayectoria aparente del Sol y la selección del algoritmo de control para seguidores solares, específicamente en el algoritmo SPA, permitieron desarrollar un sistema de software embebido que adapta el algoritmo SPA para un prototipo de seguidor solar controlado por cálculos astronómicos que arroja el algoritmo, con una resolución que se encuentra dentro de las especificadas en los algoritmos de posicionamiento solar.

Las pruebas realizadas al sistema permitieron obtener un panorama general del funcionamiento del algoritmo para el cálculo de la trayectoria del Sol independientemente de las condiciones climatológicas. Su uso y aplicación del sistema de software puede entonces implementarse de manera más objetiva para garantizar el funcionamiento del sistema de seguimiento sin problemas en una tarjeta Raspberry Pi, a pesar de las limitaciones con las que cuenta esta tarjeta, tales como: memoria RAM y procesador.

Se analizaron los ángulos Azimutal y Elevación que se obtienen en la ejecución del sistema de software embebido, con lo que se puede concluir:

Se cumplió la hipótesis planteada al inicio de este proyecto, adaptando el algoritmo de posición solar SPA para calcular los ángulos de posición solar con un margen de error no mayor a 0.1 grados para posicionar un mecanismo de seguidor solar.

El método de seguimiento para los cálculos astronómicos presenta una buena resolución que puede incluso disminuirse acortando los tiempos de ajuste. En las pruebas el error fue menor a 0.5º en el ángulo Azimutal y menor a 0.01º en el ángulo de Elevación.

Al no utilizar sensores solares, el seguidor solar no es afectado por las condiciones climatológicas y atmosféricas.

La ventaja de utilizar este algoritmo es que fácilmente puede ser programado en una tarjeta Raspberry Pi, trayendo como resultado un sistema de seguimiento que puede ser fabricado a un costo más barato.

6.1. Trabajos futuros

Es factible continuar la línea de investigación aportando ahora el control automático de corrección de error en la orientación y nivelación de seguidores solares o desarrollar tecnología propia para el resto de los componentes del sistema, por ejemplo: un módulo para controlar los dos ejes del seguidor. Esta información puede servir como referencia para desarrollar sistemas controladores de posicionamiento solar.

Considerar en trabajos futuros, aplicando en sistemas de seguidores reales los siguientes puntos:

1. Integrar sensores de posición en los mecanismos de movimiento que retroalimente al algoritmo con la posición angular real del seguidor, permitiendo realizar las correcciones necesarias para solucionar posibles desajustes debidos a los mecanismos o agentes externos.

2. Integrar limitadores o finales de carrera que impidan el desplazamiento angular más allá de determinados límites de seguridad previamente programados.

3. Por último, que el software decida colocar al seguidor en posición de seguridad (inclinación horizontal) a partir de una señal de alarma o un posible reset.

4. Se recomienda realizar pruebas de funcionamiento del sistema bajo distintas condiciones y época del año y validar que efectivamente resulta más eficiente la captación de rayos solares mediante un seguidor solar.

REFERENCIAS

Abdallah, S. and Nijmeh, S. (2004). Two axes sun tracking system with plc control. *Energy Conversion Management*, 45:1931–1939.

Aiuchi, K., Yoshida, K., Onozaki, M., Katayama, Y., Nakamura, M., and Nakamura, K. (2006). Sensor-controlled heliostat with an equatorial mount. *Solar Energy*, 80:1089–1097.

Andreas, A. and Reda, I. (2011). Midc spa calculator. *Alliance for Sustainable Energy*. Accesado el 13 de Agosto de 2015, `http://www.nrel.gov/midc/solpos/spa.html`.

Arbab, H., Jazi, B., and Rezagholizadeh (2009). M. a computer tracking system of solar dish with two-axis degree freedoms based on picture processing of bar shadow. renew. *Renew. Energy*, 34:1114–1118 n.

Archer, C. B. (1980). Comments on calculating the position of the sun. *Solar Energy 25*, 25:91.

Beltrán-Adán, J. (2007). Prototipo fotovoltaico con seguimiento del sol para procesos electroquímicos. Master's thesis, cenidet. Accesado el 29 de Abril de 2015, `http://www.cenidet.edu.mx/subaca/web-mktro/submenus/investigacion/tesis/40%20Jose%20Beltran%20Adan.pdf`.

Berenguel, M., Rubio, F., Valverde, A., Lara, P., Arahal, M., Camacho, E., and M., L. (2004). An artificial vision-based control system for automatic heliostat positioning offset correction in a central receiver solar power plant. *Solar Energy*, 76:563–575.

Blanco-Muriel, M., Alarcon-Padilla, D., Lopez-Moratalla, T., and Lara-Coira, M. (2001). Computing the solar vector. *Solar Energy*, 70:431–441. Accesado el 24 de Febrero de 2015, `http://pvcdrom.pveducation.org/SUNLIGHT/sunPSA.HTM`.

Camblor-Ordiz, A. (1995). Teoria de astronomia. pages 107–113.

Cengel, Y. A. (2003). *Heat transfer - fundamentals of thermal radiation*, page 561. McGraw-Hill, 2da. edition.

Chen, F. and Feng, J. (2007). Analogue sun sensor based on the optical nonlinear compensation measuring principle. *Meas. Sci. Technol*, 18:2111–2115.

Chen, F., Feng, J., and Hong, Z. (2006). Digital sun sensor based on the optical vernier measuring principle. *Measurement Science and Technology*, 17:2494–2498.

Chen, Y., Chong, K., Bligh, T., Chen, L., Yunus, J., Kannan, K., Lim, B., Lim, C., Alias, M., Bidin, N., Aliman, O., Salehan, S., Rezan, S., Tam, C., and Tan, K. (2001). Non-imaging, focusing heliostat. *Solar Energy*, 71:155–164.

Chong, K. and Wong, C. (2009). General formula for on-axis sun-tracking system and its application in improving tracking accuracy of solar collector. *Solar Energy*, 83:298–305.

Chong, K., Wong, C., Siaw, F., and Yew, T. (2010). Optical characterization of non-imaging planar concentrator for the application in concentrator photovoltaic system. *Solar Energy Eng*, page 9. accepted for publication.

Cooper, P. I. (1969). The absortion of radiation in solar stills. *Solar Energy*, 12:333–346.

Duffett-Smith, P. (1992). Practical astronomy with your calculator. pages 64–65.

Engin, M. and Engin, D. (2012). Sun tracking control strategy for improved reliability and performance. *Education and Research Conference (EDERC), 2012 5th European DSP*, pages 85–89.

García-Lara, C. M. (2010). Energías renovables. *Revista informativa de ingeniería ambiental*. Accesado el 25 de Febrero de 2014, `http://www.unicach.edu.mx/_/ambiental/descargar/Gaceta5/Energias%20Renovables.pdf`.

Group, V. I. T. S. (2002). Rs232 data standard & protocol. page 2. Accesado el 30 de Marzo de 2015, `http://www.vicon-security.com/Collateral/Documents/English-US/support/RS232DataStandardProtocol-.pdf`.

Ilyas, M. (1983). Solar position programs: refraction, sidereal time and sunset / sunrise. *Solar Energy 31*, pages 437–438.

Ilyas, M. (1984). Reply to comments by dr. d. j. b. pascoe. *Solar Energy 34*, page 206.

International, T. (2014). Husos horarios. Accesado el 29 de Abril de 2015, http://www.horamundial.com/husos.php.

Iqbal, M. (1983). An introduction to solar radiation. *Academ-ic Press*, pages 23–25.

Jacques-García, F. A., Trejo-Morales, A., Guerrero-Cruz, V. H., Rico-Hernández, R. A., Hernández-Valerio, J. S., Olmos-Trejo, C. A., Xicoténcatl-Ramírez, G., and Chávez-Morales, U. (2014). Método criptográfico de sustitución simétrica para el análisis del cómputo matricial en sistemas embebidos. *SABES Revista electrónica de divulgación de la investigación*, 7.

Jiménez-Vicente, J. (2015). Breve historia del calendario. *eduMedia*. Accesado el 29 de Abril de 2015, https://www.edumedia-sciences.com/es/a565-movimiento-aparente-del-sol.

Jing-Min, W. and Chia-Liang, L. (2013). Design and implementation of a sun tracker with a dual-axis single motor for an optical sensor-based photovoltaic system. *sensors*. Accesado el 30 de Enero de 2014.

Kalogirou, S. A. (1996). Design and construction of a one-axis sun-tracking system. *Solar Energy*, 57:465–469.

Kambezidis, H. D. and Papanikolaou, N. S. (1990). Solar position and atmospheric refraction. *Solar Energy*, 44:143.

Kribus, A., Vishnevetsky, I., Yogev, A., and Rubinov (2004). T. closed loop control of heliostats. *Energy*, 29:905–913.

Laboratory, N. R. E. (2000). Solar position and intensity. *Renewable Resource Data Center*. Accesado el 25 de Febrero de 2015, http://rredc.nrel.gov/solar/codesandalgorithms/solpos/.

Lamm, L. O. (1981). A new analytic expression for the equation of time. *Solar Energy*, 26:465.

Lee, C., Yeh, H., Chen, M., Sue, X., and Tzeng, Y. (2006). Hcpv sun tracking study at iner.in proceedings of the ieee 4th world conference on photovoltaic energy conversion.

LRPS, I. S. (2011). Solar calculator 2.2. *Iesmith*. Accesado el 13 de Agosto de 2015, http://www.iesmith.net/tools/solarcalc.html.

Luque-Heredia, I., Cervantes, R., and Goulven, Q. (2006). A sun tracking error monitor for photovoltaic concentrators. *IEEE Explore*, 1:706–709.

Luque-Heredia, I., Gordillo, F., and Rodriguez, F. (2004). A pi based hybrid sun tracking algorithm for photovoltaic concentration. *Solar Energy*, page 10.

Luque-Heredia, I., Moreno, J., Magalhaes, P., Cervantes, R., Quemere, G., and Laurent, O. (2007). Inspira's cpv sun tracking. In *Concentrator Photovoltaics*, volume 130 of *Springer Series in Optical Sciences*, chapter 11, pages 221–251. Springer Berlin Heidelberg, Germany.

Meeus, J. (1998). *Astronomical Algorithms*. Willmann-Bell Inc., 2da. edition.

Michalsky, J. J. (1988). The astronomical almanac's algorithm for approximate solar position (1950-2050). *Solar Energy*, 40-3:227–235.

Muir, L. R. (1983). Comments on the effect of atmospheric refraction on the solar azimuth. *Solar Energy*, 30:295.

Nuwayhid, R., Mrad, F., and Abu-Said, R. (2001). The realization of a simple solar tracking concentrator for the university research applications. *Renew. Energy*, 24:207–222.

Pascoe, D. J. B. (1984). Comments on solar position programs: refraction, sidereal time and sunset / sunrise. *Solar Energy*, 34:205–206.

Peng, H., Du, T., and Gu, W. (2013). Application design of a sun-tracking system. *IEEE Xplore*, pages 5094–5098.

Pitman, C. L. and Vant-Hull, L. L. (1978). Errors in locating the sun and their effect on solar intensity predictions. *Solar Energy*, pages 701–706.

Raspberry-Pi, F. (2015). What is a raspberry pi? Accesado el 30 de Enero de 2015, http://www.raspberrypi.org/help/what-is-a-raspberry-pi/.

Reda, I. and Andreas, A. (2008a). Solar position algorithm for solar radiation applications. *Solar Energy*, 76:577–589. Accesado el 25 de Febrero de 2015, http://www.nrel.gov/midc/spa/#register.

Reda, I. and Andreas, A. (2008b). Solar position algorithm for solar radiation applications. *National Renewable Energy*, 01:1–56. Accesado el 09 de Marzo de 2015, http://www.nrel.gov/docs/fy08osti/34302.pdf.

Rizvi, A., Addoweesh, K., El-Leathy, A., and Al-Ansary, H. (2014). Sun position algorithm for sun tracking applications. *Industrial Electronics Society, IECON 2014 - 40th Annual Conference of the IEEE*, pages 5595–5598.

Rubio, F., Ortega, M., Gordillo, F., and Lopez-Martinez, M. (2007). Application of new control strategy for sun tracking. *Energy Conver. Manage*, 48:2174–2184.

Shanmugam, S. and Christraj, W. (2005). The tracking of the sun for solar paraboloidal dish concentrators. *Solar Energy Eng*, 127:156–160.

Sidek, M., Hasan, W., Ab Kadir, M., Shafie, S., Radzi, M., Ahmad, S., and Marhaban, M. (2014). Gps based portable dual-axis solar tracking system using astronomical equation. *Power and Energy (PECon), 2014 IEEE International Conference on*, pages 245–249.

S.L., S. M. T. (2015). Renewable energy catalogue 2015 - sensors for solar tracking. *Solar MEMS*. Accesado el 13 de Agosto de 2015, http://www.solar-mems.com/smt_pdf/renewable_energy_catalogue.pdf.

Spencer, J. W. (1971). Fourier series representation of the position of the sun. *Search*, 2:5.

Spencer, J. W. (1989). Comments on the astronomical almanac's algorithm for approximate solar position (1950-2050). *Solar Energy 42*, page 353.

Swift, L. W. (1976). Algorithm for solar radiation on mountain slopes. *Water Resources Research*, 12:108–112.

Van der Staay, M. (2015). Dom calculadora de posición - azimut y zenith. *solartopo*. Accesado el 13 de Agosto de 2015, http://www.solartopo.com/orbita-solar.htm.

Walraven, R. (1978). Calculating the position of the sun. *Solar Energy*, 20:393–397.

Walraven, R. (1979). Erratum. *Solar Energy*, 22:195.

Wilkinson, B. J. (1981). An improved fortran program for the rapid calculation of the solar position. *Solar Energy 27*, pages 67–68.

Wilkinson, B. J. (1983). The effect of atmospheric refraction on the solar azimuth. *Solar Energy*, 30:295.

Zheng-wei, H. (2010). I2c protocol design for reusability. *Information Processing (ISIP), 2010 Third International Symposium on*, 10:83–86.

A. APÉNDICES

A.1. Código fuente del sistema embebido

Código A.1: Código fuente de librerías en lenguaje *C*

```
1 // Librerias exclusivas para Raspberry Pi
2 #include <wiringPi.h>
3 #include <wiringSerial.h>
4 #include <linux/i2c-dev.h>
5
6 // Incluye el archivo de cabecera spa.h
7 #include "spa.h"
```

Código A.2: Código fuente de variables e inicialización en lenguaje *C*

```
1 // Variable de paquete de datos del GPS
2 char *tramaGPS[15];
3
4 // Variables para Tiempo Universal
5 unsigned  long utcTime, utcHour;
6 unsigned  long utcMinutes;
7 unsigned  long utcSeconds;
8
9 // Variables para Tiempo Est\'{a}ndar del Este
10 unsigned  long estTime, estHour;
11
12 // Variables para GPS
13 float   latitude;
14 int     latDegrees;
15 float latMinutes;
16
17 float  longitude;
18 int  longDegrees;
19 float longMinutes;
20
21 float  altitude;
22 int  altDegrees;
23 float altMinutes;
24
25 int flagGPS = 1;
26 char old_buffer[100] = {0};
27
28 // Variables para i2c LCD
29 int fd_LCD;
30 int  address = 0x28;
31 char *fileName = "/dev/i2c-1";
32
33 // Variables para i2c RTC
34 int fd_RTC;
```

```
35 int addressRTC = 0x68;
36
37 // Variables para RS-232
38 int fd_RS232;
39 int baudRate = 9600;//38400;
40 char *portName = "/dev/ttyAMA0";
41
42 // Variables for RS-232 USB
43 int fd_RS232USB;
44 int baudRateUSB = 19200;
45 char *portNameUSB = "/dev/ttyUSB0";
46
47 // Variable para controlar errores
48 int flagPort = 0;
49
50 // Variables para trama GPS
51 char start[1];
52 char stop[1];
53 char current_buffer[100] = {0};
54 char buffer[100] = {0};
55 int dataAvailable, currentCharacter;
56
57 char cVal[16] = {0};
58 char dest[16] = {0};
59
60 // Variables para Archivos
61 FILE *filePointer;
62
63 // Variables para Timer
64 time_t startTime, endTime;
65 float timeElapsed = 0.0;
66
67 time_t startTime2, endTime2;
68 float timeElapsed2 = 0.0;
69
70 // Variables para \'{a}ngulos
71 char cAzimuth[15]={0};
72 char cElevation[15]={0};
73
74 // Variables para env\'{i}o de par\'{a}metros
75 int year = 0;
76 int month = 0;
77 int day = 0;
78 int hour = 0;
79 int minute = 0;
80 int second = 0;
81 int myTimeZone = 0;
82
83 // Funci\'{o}n prototipo
84 int executeSolarTracking ();
```

Código A.3: Código fuente de la función WriteToMotors() en lenguaje *C*

```c
int WriteToMotors (int fd, char *CMD)
{
 int n=0;
 n= strlen(CMD);
 if ((write(fd, CMD, n)) != n) {
  printf("Error writing to RS-232 to USB\n");
  return 0;
 }

 return 0;
}
```

Código A.4: Código fuente de la función WriteToLCD() en lenguaje *C*

```c
int WriteToLCD (int fd, char *MSJ)
{
 int n=0;
 n= strlen(MSJ);
 if ((write(fd, MSJ, n)) != n) {
  printf("Error writing to i2c slave\n");
  exit(1);
 }

 return 0;
}
```

Código A.5: Código fuente de la función CleanToLCD() en lenguaje *C*

```c
int CleanToLCD (int fd)
{
 char buf[3];
 buf[0] = 0xFE;
 buf[1] = 0x58;
 WriteToLCD(fd, buf);

 return 0;
}
```

Código A.6: Código fuente de la función SetCursor() en lenguaje *C*

```c
int SetCursor(int fd, int column, int row)
{
 char buf[10];
 buf[0] = 0xFE;
 buf[1] = 0x47;
 buf[2] = column;
 buf[3] = row;
 if ((write(fd, buf, 4)) != 4) {
  printf("Error writing to i2c slave\n");
  exit(1);
 }

 return 0;
}
```

Código A.7: Código fuente de la función CurrentTime (void) en lenguaje *C*

```c
int CurrentTime (void)
{
 time_t t;
 time_t fecha;
 struct tm *temp;
 struct tm fecha_tm;

 int myZone=0;
 struct tm *tm;
 char tm_zone[50];

 // Obtener fecha (time_t)
 if ((fecha = time(NULL)) == (time_t) -1)
   return 0;

 tm=localtime(&t);
 strftime(tm_zone,50,"%z",tm);
 sscanf(tm_zone, "%d", &myZone);
 myTimeZone = myZone/100;

 // Obtener fecha (struct tm)
 temp = localtime(&fecha);
 memcpy(&fecha_tm, temp, sizeof fecha_tm);

 year  = fecha_tm.tm_year + 1900;
 month  = fecha_tm.tm_mon+1;
 day  = fecha_tm.tm_mday;

 hour  = fecha_tm. tm_hour;
 minute  = fecha_tm.tm_min;
 second  = fecha_tm.tm_sec;
 return 0;
}
```

Código A.8: Ejemplo de la trama de Bytes que transmite el módulo GPS

```
$GPGAA,hhmmss.ss,ddmm.mmmm,n,dddmm.mmmm,e,q,ss,y.y,a.a,z,g.g,z,t.t,iii,*CC

Donde:
      GPGAA        : GPS fixed data identifier
      hhmmss.ss    : Coordinated Universal Time (UTC), also known as GMT
      ddmm.mmmm,n  : Latitude in degrees, minutes and cardinal sign
      dddmm.mmmm,e : Longitude in degrees, minutes and cardinal sign
      q            : Quality of fix.  1 = there is a fix
      ss           : Number of satellites being used
      y.y          : Horizontal dilution of precision [HDOP]
      a.a,M        : GPS antenna altitude in meters
      g.g,M        : geoidal separation in meters
      t.t          : Age of the defferential correction data
      iiii         : Deferential station's ID
      *CC          : checksum for the sentence
```

Código A.9: Código fuente de la función splitBuffer(char *tramaGPG) en lenguaje *C*

```
int splitBuffer(char *tramaGPG)
{
 int i=0;
 char *pch;
 flagGPS = 0;
 pch = strtok(tramaGPG,",");

 // Limpiar Arreglo
 tramaGPS[0]  = 0; // Message ID - GPGAA
 tramaGPS[1]  = 0; // UTC Time
 tramaGPS[2]  = 0; // Latitude
 tramaGPS[3]  = 0; // N/S Indicator
 tramaGPS[4]  = 0; // Longitude
 tramaGPS[5]  = 0; // E/W Indicator
 tramaGPS[6]  = 0; // Position Fix indicator
 tramaGPS[7]  = 0; // Satellities Used
 tramaGPS[8]  = 0; // HDOP
 tramaGPS[9]  = 0; // MSL Altitude
 tramaGPS[10] = 0; // Units
 tramaGPS[11] = 0; // Age of Diff. Corr.
 tramaGPS[12] = 0; // Diff. Rf. Station ID
 tramaGPS[13] = 0; // Checksum

 printf("%s == GPGGA \n",pch);
 printf("%d\n\n",strcmp(pch, "GPGGA"));

 if(strcmp(pch, "GPGGA")== 0)
 {
  while(pch != NULL)
  {
   pch = strtok(NULL,",");
```

```
32    tramaGPS[i] = pch;
33
34    switch (i+1)
35    {
36     case 0: // Message ID - GPGAA
37       break;
38
39     case 1: // UTC Time: hhmmss.ss
40       //sscanf(tramaGPS[i], "%ld", &utcTime);
41       break;
42
43     case 2: // Get Latitude: ddmm.mmmm
44       // Convertir latitude de String a Numero
45       sscanf(tramaGPS[i], "%f", &latitude);
46       printf("Lat.: \t\t\t %f\n", latitude);
47       if(latitude == 0.0) {
48        flagGPS = 1;
49        latitude = 2062.4273879797343; //Valores obtenidos de Google Maps.
50       }
51       latitude = latitude/100;
52       printf("Latitude: \t\t\t %3.4f DEG\n", latitude);
53       break;
54
55     case 3: // N/S Indicator:
56       //printf("N/S Indicator: \t\t\t %s \n",tramaGPS[i]);
57       break;
58
59     case 4: // Get Longitude: dddmm.mmmm
60       sscanf(tramaGPS[i], "%f", &longitude);
61       if(longitude == 0.0) {
62        flagGPS = 1;
63        longitude     = 10040.366381406784;
64       }
65       longitude = (longitude/100) * -1;
66       printf("Longitude: \t\t\t %3.4f DEG\n", longitude);
67       break;
68
69     case 5: // E/W Indicator:
70       //printf("E/W Indicator: \t\t\t %s \n",tramaGPS[i]);
71       break;
72
73     case 6: // Get Position Fix indicator
74       //printf("Get Position Fix indicator: \t %s \n",tramaGPS[i]);
75       break;
76
77     case 7: // Satellities Used
78       //printf("Satellities Used: \t\t %s \n",tramaGPS[i]);
79       break;
80
81     case 8: // HDOP
82       //printf("HDOP: \t\t\t\t %s \n\n",tramaGPS[i]);
83       break;
84
85     case 9: // Get MSL Altitude
86       sscanf(tramaGPS[i], "%f", &altitude);
```

```
87        if(altitude == 0) {
88         flagGPS = 1;
89         altitude = 1829;
90        }
91        printf("Altitude: \t\t\t %3.4f MSL\n\n", altitude);
92        break;
93
94      case 10: // Units
95        //printf("Units: \t\t\t\t %s \n",tramaGPS[i]);
96        break;
97
98      case 11: //  Age of Diff. Corr
99        //printf("Age of Diff. Corr: \t\t %s \n",tramaGPS[i]);
100       break;
101
102     case 12: //  Diff. Rf. Station ID
103       //printf("Diff. Rf. Station ID: \t\t %s \n",tramaGPS[i]);
104       break;
105
106     case 13: //  Checksum
107       //printf("Checksum: \t\t\t %s \n",tramaGPS[i]);
108       break;
109    }
110    i=i+1;
111   }
112  }
113  else {
114   flagGPS = 1;
115  }
116  return 0;
117 }
```

Código A.10: Código fuente de la función ReadGPS(void) en lenguaje *C*

```c
int ReadGPS(void)
{
 int i = 0 , j = 0;

 // Limpiar el Buffer Serial
 serialFlush(fd_RS232);

 // Inicializar variables
 start[0] = 0x24;
 stop[0]= 0x24;

 delay(1000);
 dataAvailable = serialDataAvail(fd_RS232)                          ;
 printf("\nserialDataAvail: %d\n", dataAvailable);

 for (i=0;i<dataAvailable;i++)
 {
  currentCharacter = serialGetchar(fd_RS232);
  if ( currentCharacter == start[0])
  {
   for(;;)
   {
    currentCharacter = serialGetchar(fd_RS232);
    if (currentCharacter == stop[0])
    {
     current_buffer[j] = '\0';
     break;
    }
    else
    {
     current_buffer[j] = currentCharacter;
     j++;
    }
   }
   if (strcmp(current_buffer,old_buffer) != 0)
   {
    strcpy(old_buffer, current_buffer);
   }
   printf("Buffer GPS: %s \n", old_buffer);
   serialFlush(fd_RS232);

   break;
  }
 }
 serialFlush(fd_RS232);
 if (dataAvailable > 100)
  splitBuffer(old_buffer);

 return 0;
}
```

Código A.11: Código fuente de la función PI_THREAD(executeSPA) en lenguaje *C*

```
PI_THREAD (executeSPA)
{
    for(;;)
    {
        startTime = time(NULL);

        // Obtener Fecha y Hora actual
        CurrentTime ();
        // Obtener datos GPS
        if (flagGPS == 1)
            ReadGPS ();

        // Ejecutar algoritmo SPA y posicionar seguidor solar
        executeSolarTracking ();

        endTime = time(NULL);

        if(endTime != startTime)
            timeElapsed = difftime(endTime, startTime);

        printf("%f seconds \n\n\n", timeElapsed);

        if (flagGPS == 0)
        {
            // Pausa de 2000ms
            delay(3000-(((int) timeElapsed)*1000));
        }else
        {
            // Pausa de 10000ms
            delay(11000-(((int) timeElapsed)*1000));
        }
    }
    return 0;
}
```

Código A.12: Código fuente de la función PI_THREAD(displayData) en lenguaje *C*

```
PI_THREAD (displayData)
{
    char strTIME[20] = {0};
    char str[15];

    for(;;)
    {
        startTime2 = time(NULL);

        CleanToLCD(fd_LCD);
        SetCursor(fd_LCD, 1, 1);
        WriteToLCD(fd_LCD, "DATE:");
        SetCursor(fd_LCD, 1, 2);
        sprintf(strTIME,"%02d/%02d/%d",day,month,year);
        WriteToLCD(fd_LCD, strTIME);

        delay(2000);

        CleanToLCD(fd_LCD);
        SetCursor(fd_LCD, 1, 1);
        WriteToLCD(fd_LCD, "TIME:");
        SetCursor(fd_LCD, 1, 2);
        sprintf(strTIME,"%02d:%02d:%0d",hour,minute,second);
        WriteToLCD(fd_LCD, strTIME);

        delay(1000);

        CleanToLCD(fd_LCD);
        SetCursor(fd_LCD, 1, 1);
        WriteToLCD(fd_LCD, "Latitude:");
        SetCursor(fd_LCD, 1, 2);
        sprintf(str,"%f",latitude);
        WriteToLCD(fd_LCD, str);

        delay(1000);

        CleanToLCD(fd_LCD);
        SetCursor(fd_LCD, 1, 1);
        WriteToLCD(fd_LCD, "Longitude:");
        SetCursor(fd_LCD, 1, 2);
        sprintf(str,"%f",longitude);
        WriteToLCD(fd_LCD, str);

        delay(1000);

        CleanToLCD(fd_LCD);
        SetCursor(fd_LCD, 1, 1);

        WriteToLCD(fd_LCD, "Azimuth:");
        SetCursor(fd_LCD, 9, 1);
        strcpy(str, "");
        strncat(str, cAzimuth, 8);
        WriteToLCD(fd_LCD, str);
```

```
54
55          SetCursor(fd_LCD, 1, 2);
56          WriteToLCD(fd_LCD, "Elevat.:");
57          SetCursor(fd_LCD, 9, 2);
58          strcpy(str, "");
59          strncat(str, cElevation, 8);
60          WriteToLCD(fd_LCD, str);
61
62          endTime2 = time(NULL);
63          if(endTime2 != startTime2)
64              timeElapsed2 = difftime(endTime2, startTime2);
65
66          // Pausa de 60seconds
67          delay(30000-(((int) timeElapsed2)*1000));
68      }
69      return 0;
70  }
```

Código A.13: Código fuente de la función main() en lenguaje *C*

```
1  int main (int argc, char *argv[])
2  {
3      // Abrir puerto I2C para leer y escribir al LCD
4      if ((fd_LCD = open(fileName, O_RDWR)) < 0) {
5          printf("Failed to open i2c port for LCD\n");
6          exit(1);
7      }
8
9      // Asignar los parametros de configuracion al dispositivo
10     if (ioctl(fd_LCD, I2C_SLAVE, address) < 0) {
11         printf("Unable to get bus access to talk to slave\n");
12         exit(1);
13     }
14
15     // Configurar libreria wiringPi para RS-232 con GPS
16     if (wiringPiSetup < 0) {
17         printf("Failed to wiringPiSetup \n");
18         exit(1);
19     }
20
21     // Abrir puerto RS232 para leer y escribir
22     if ((fd_RS232 = serialOpen(portName,baudRate)) < 0) {
23         printf("Failed to open RS-232 port\n");
24         exit(1);
25     }
26
27     // Abrir puerto USB para leer y escribir
28     if ((fd_RS232USB = serialOpen(portNameUSB,baudRateUSB)) < 0) {
29         printf("Failed to open RS-232 USB port\n");
30         flagPort=1;
31         exit(1);
32     }
33
```

```
34      // Habilitar Motor 1 y Motor 2
35      if(flagPort != 1)
36      {
37          WriteToMotors (fd_RS232USB, "1MO\r");
38          WriteToMotors (fd_RS232USB, "2MO\r");
39
40          // Posicionar a cero el Motor 1 y Motor 2
41          WriteToMotors (fd_RS232USB, ("1OR0\r"));
42          WriteToMotors (fd_RS232USB, ("2OR0\r"));
43      }
44
45      filePointer = fopen("trayectoria.txt", "a+");
46      fprintf(filePointer, "Date-Time \t\t\t latitude\t longitude\t azimuth\t zenith\n");
47      fclose(filePointer);
48
49      // Crear hilos
50      int x=piThreadCreate(executeSPA);
51      if(x!=0)
52      {
53          //printf("El hilo no puedo iniciar...\n");
54      }
55
56      int y=piThreadCreate(displayData);
57      if(y!=0)
58      {
59          //printf("El hilo no puedo iniciar...\n");
60      }
61
62      // Iniciar ciclo infinito
63      for(;;)
64      {
65          delay(1000);
66      }
67
68      return 0;
69  }
```

Código A.14: Código fuente de la función executeSolarTracking(void) en lenguaje *C*

```
int executeSolarTracking ()
{
    int i = 0;
    char *pch;
    char cmd[80]={0};

    int timezone = -5;
    int dDate[3] = {0};
    int dTime[3] = {0};
    double dGPS[3] = {0};

    char *getAngles= {0};

    timezone = -5;        //myTimeZone;

    dDate[0] = day;
    dDate[1] = month;
    dDate[2] = year;

    dTime[0] = hour;
    dTime[1] = minute;
    dTime[2] = second;

    dGPS[0] = latitude;
    dGPS[1] = longitude;
    dGPS[2] = altitude;

    // Obtener angulos de la funcion spa_algorithm
    getAngles =  spa_algorithm (dDate, dTime, dGPS, timezone);

    // Buscar el token ","
    pch = strtok(getAngles,",");

    while(pch != NULL)
    {
        switch (i)
        {
            case 0: // Azimuth angle
                    printf("Azimuth angle:      %s\n",pch);
                    strcpy (cAzimuth, pch);
                    break;

            case 1: // Elevation angle
                    printf("Elevation angle:      %s\n",pch);
                    strcpy (cElevation, pch);
                    break;
        }
        i++;
        pch = strtok(NULL,",");
    }

    // Construir comando para posicionar el seguidor
    strcpy (cmd,"1PA");
```

```
54      strcat (cmd,cElevation);
55      strcat (cmd,"\r");
56
57      // Escribir el comando y posicionar a xxx grados
58      if(flagPort != 1)
59      {
60          WriteToMotors (fd_RS232USB, cmd);
61      }
62      delay(100);
63
64      strcpy (cmd,"2PA");
65      strcat (cmd,cAzimuth);
66      strcat (cmd,"\r");
67
68      // Escribir el comando y posicionar a xxx grados
69      if(flagPort != 1)
70      {
71          WriteToMotors (fd_RS232USB, cmd);
72      }
73      delay(100);
74
75      return 0;
76  }
```

A.2. Archivo de cabecera SPA (spa.h)

Código A.15: Código fuente del archivo de cabecera SPA (spa.h)

```
1  ////////////////////////////////////////////////////////////////////////
2  //                     HEADER FILE for SPA.C                          //
3  //                                                                    //
4  //   Solar Position Algorithm (SPA) for Solar Radiation Application   //
5  //                                                                    //
6  //   Filename: spa.h                                                  //
7  //                                                                    //
8  //   Measurement & Instrumentation Team Solar Radiation Research      //
9  //   Laboratory. National Renewable Energy Laboratory. 1617 Cole Blvd,//
10 //    Golden, CO 80401                                                //
11 ////////////////////////////////////////////////////////////////////////
12
13 ////////////////////////////////////////////////////////////////////////
14 //                                                                    //
15 // Usage:                                                             //
16 //                                                                    //
17 //   1) In calling program, include this header file,                 //
18 //      by adding this line to the top of file:                       //
19 //           #include "spa.h"                                        //
20 //                                                                    //
21 //   2) In calling program, declare the SPA structure:                //
22 //           spa_data spa;                                            //
23 //                                                                    //
24 //   3) Enter the required input values into SPA structure            //
25 //      (input values listed in comments below)                       //
26 //                                                                    //
27 //   4) Call the SPA calculate function and pass the SPA structure    //
28 //      (prototype is declared at the end of this header file):       //
29 //           spa_calculate(&spa);                                     //
30 //                                                                    //
31 //   Selected output values (listed in comments below) will be        //
32 //   computed and returned in the passed SPA structure.  Output       //
33 //   will based on function code selected from enumeration below.     //
34 //                                                                    //
35 //   Note: A non-zero return code from spa_calculate() indicates that //
36 //         one of the input values did not pass simple bounds tests.  //
37 //         The valid input ranges and return error codes are also     //
38 //         listed below.                                              //
39 //                                                                    //
40 ////////////////////////////////////////////////////////////////////////
41
42 #ifndef __solar_position_algorithm_header
43 #define __solar_position_algorithm_header
44
45 #include <stdio.h>
46 #include <stdlib.h>
47 #include <time.h>
48 #include <math.h>
```

```c
#include <string.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <time.h>
#include <dirent.h>
#include <sys/param.h>

// Enumeration for function codes to select desired final outputs from SPA
enum {
    SPA_ZA,           //calculate zenith and azimuth
    SPA_ZA_INC,       //calculate zenith, azimuth, and incidence
    SPA_ZA_RTS,       //calculate zenith, azimuth, and sun rise/transit/set values
    SPA_ALL,          //calculate all SPA output values
};

typedef struct
{
    //----------------------INPUT VALUES-----------------------
    int year;            // 4-digit year,      valid range: -2000 to 6000, error code: 1
    int month;           // 2-digit month,         valid range: 1 to  12,  error code: 2
    int day;             // 2-digit day,           valid range: 1 to  31,  error code: 3
    int hour;            // Observer local hour,   valid range: 0 to  24,  error code: 4
    int minute;          // Observer local minute, valid range: 0 to  59,  error code: 5
    int second;          // Observer local second, valid range: 0 to <60,  error code: 6

    double delta_ut1;    // Fractional second difference between UTC and UT which is used
                         // to adjust UTC for earth's irregular rotation rate and is derived
                         // from observation only and is reported in this bulletin:
                         // http://maia.usno.navy.mil/ser7/ser7.dat,
                         // where delta_ut1 = DUT1
                         // valid range: -1 to 1 second (exclusive), error code 17

    double delta_t;      // Difference between earth rotation time and terrestrial time
                         // It is derived from observation only and is reported in this
                         // bulletin: http://maia.usno.navy.mil/ser7/ser7.dat,
                         // where delta_t = 32.184 + (TAI-UTC) - DUT1
                         // valid range: -8000 to 8000 seconds, error code: 7

    double timezone;     // Observer time zone (negative west of Greenwich)
                         // valid range: -18   to   18 hours,   error code: 8

    double longitude;    // Observer longitude (negative west of Greenwich)
                         // valid range: -180  to  180 degrees, error code: 9

    double latitude;     // Observer latitude (negative south of equator)
                         // valid range: -90   to   90 degrees, error code: 10

    double elevation;    // Observer elevation [meters]
                         // valid range: -6500000 or higher meters,    error code: 11

    double pressure;     // Annual average local pressure [millibars]
                         // valid range:    0 to 5000 millibars,       error code: 12
```

```
104
105     double temperature;  // Annual average local temperature [degrees Celsius]
106                          // valid range: -273 to 6000 degrees Celsius, error code; 13
107
108     double slope;        // Surface slope (measured from the horizontal plane)
109                          // valid range: -360 to 360 degrees, error code: 14
110
111     double azm_rotation; // Surface azimuth rotation (measured from south to projection of
112                          //     surface normal on horizontal plane, negative east)
113                          // valid range: -360 to 360 degrees, error code: 15
114
115     double atmos_refract;// Atmospheric refraction at sunrise and sunset (0.5667 deg is typical)
116                          // valid range: -5   to   5 degrees, error code: 16
117
118     int function;        // Switch to choose functions for desired output (from enumeration)
119
120     //-----------------Intermediate OUTPUT VALUES--------------------
121     double jd;           //Julian day
122     double jc;           //Julian century
123
124     double jde;          //Julian ephemeris day
125     double jce;          //Julian ephemeris century
126     double jme;          //Julian ephemeris millennium
127
128     double l;            //earth heliocentric longitude [degrees]
129     double b;            //earth heliocentric latitude [degrees]
130     double r;            //earth radius vector [Astronomical Units, AU]
131
132     double theta;        //geocentric longitude [degrees]
133     double beta;         //geocentric latitude [degrees]
134
135     double x0;           //mean elongation (moon-sun) [degrees]
136     double x1;           //mean anomaly (sun) [degrees]
137     double x2;           //mean anomaly (moon) [degrees]
138     double x3;           //argument latitude (moon) [degrees]
139     double x4;           //ascending longitude (moon) [degrees]
140
141     double del_psi;      //nutation longitude [degrees]
142     double del_epsilon; //nutation obliquity [degrees]
143     double epsilon0;     //ecliptic mean obliquity [arc seconds]
144     double epsilon;      //ecliptic true obliquity  [degrees]
145
146     double del_tau;      //aberration correction [degrees]
147     double lamda;        //apparent sun longitude [degrees]
148     double nu0;          //Greenwich mean sidereal time [degrees]
149     double nu;           //Greenwich sidereal time [degrees]
150
151     double alpha;        //geocentric sun right ascension [degrees]
152     double delta;        //geocentric sun declination [degrees]
153
154     double h;            //observer hour angle [degrees]
155     double xi;           //sun equatorial horizontal parallax [degrees]
156     double del_alpha;    //sun right ascension parallax [degrees]
157     double delta_prime; //topocentric sun declination [degrees]
158     double alpha_prime; //topocentric sun right ascension [degrees]
```

```
159     double h_prime;     //topocentric local hour angle [degrees]
160
161     double e0;          //topocentric elevation angle (uncorrected) [degrees]
162     double del_e;       //atmospheric refraction correction [degrees]
163     double e;           //topocentric elevation angle (corrected) [degrees]
164
165     double eot;         //equation of time [minutes]
166     double srha;        //sunrise hour angle [degrees]
167     double ssha;        //sunset hour angle [degrees]
168     double sta;         //sun transit altitude [degrees]
169
170     //---------------------Final OUTPUT VALUES-----------------------
171     double zenith;       //topocentric zenith angle [degrees]
172     double azimuth_astro;//topocentric azimuth angle (westward from south) [for astronomers]
173     double azimuth;      //topocentric azimuth angle (eastward from north) [for navigators and solar radiation]
174     double incidence;    //surface incidence angle [degrees]
175
176     double suntransit;   //local sun transit time (or solar noon) [fractional hour]
177     double sunrise;      //local sunrise time (+/- 30 seconds) [fractional hour]
178     double sunset;       //local sunset time (+/- 30 seconds) [fractional hour]
179
180 } spa_data;
181
182 //-------------- Utility functions for other applications (such as NREL's SAMPA) --------------
183 double deg2rad(double degrees);
184 double rad2deg(double radians);
185 double limit_degrees(double degrees);
186 double third_order_polynomial(double a, double b, double c, double d, double x);
187 double geocentric_right_ascension(double lamda, double epsilon, double beta);
188 double geocentric_declination(double beta, double epsilon, double lamda);
189 double observer_hour_angle(double nu, double longitude, double alpha_deg);
190 void   right_ascension_parallax_and_topocentric_dec(double latitude, double elevation,
191           double xi, double h, double delta, double *delta_alpha, double *delta_prime);
192 double topocentric_right_ascension(double alpha_deg, double delta_alpha);
193 double topocentric_local_hour_angle(double h, double delta_alpha);
194 double topocentric_elevation_angle(double latitude, double delta_prime, double h_prime);
195 double atmospheric_refraction_correction(double pressure, double temperature,
196                                          double atmos_refract, double e0);
197 double topocentric_elevation_angle_corrected(double e0, double delta_e);
198 double topocentric_zenith_angle(double e);
199 double topocentric_azimuth_angle_astro(double h_prime, double latitude, double delta_prime);
200 double topocentric_azimuth_angle(double azimuth_astro);
201
202 //Calculate SPA output values (in structure) based on input values passed in structure
203 int spa_calculate(spa_data *spa);
204 char *spa_algorithm (int dDate[3], int dTime[3], double dGPS[3], int timezone);
205 #endif
```

A.3. Código fuente de funciones SPA (spa.c)

Código A.16: Código fuente de variables y constantes en lenguaje *C*

```
1  ////////////////////////////////////////////////////////////////////////
2  //                                                                    //
3  //   Solar Position Algorithm (SPA) for Solar Radiation Application   //
4  //                                                                    //
5  //   Filename: spa.c                                                  //
6  //                                                                    //
7  //   Measurement & Instrumentation Team Solar Radiation Research      //
8  //   Laboratory. National Renewable Energy Laboratory. 1617 Cole Blvd,//
9  //    Golden, CO 80401                                                //
10 ////////////////////////////////////////////////////////////////////////
11
12 ////////////////////////////////////////////////////////////////////////
13 //   See the spa.h header file for usage                              //
14 //                                                                    //
15 //   This code is based on the NREL technical report "Solar Position  //
16 //   Algorithm" for Solar Radiation.                                  //
17 //   Application" by I. Reda & A. Andreas                             //
18 ////////////////////////////////////////////////////////////////////////
19
20 // Incluye el archivo de cabecera spa.h
21 #include "spa.h"
22
23 char cAngle[15] = {0};
24 char anglesSPA[80] = {0};
25
26 // Declaracion de la estructura SPA
27 spa_data spa;
28
29 #define PI         3.14159265358979323846264338327950288419 71
30 #define L_COUNT 6
31 #define B_COUNT 2
32 #define R_COUNT 5
33 #define Y_COUNT 63
34 #define SUN_RADIUS 0.26667
35 #define L_MAX_SUBCOUNT 64
36 #define B_MAX_SUBCOUNT 5
37 #define R_MAX_SUBCOUNT 40
38 #define TERM_Y_COUNT TERM_X_COUNT
39 enum {TERM_A, TERM_B, TERM_C, TERM_COUNT};
40 enum {TERM_X0, TERM_X1, TERM_X2, TERM_X3, TERM_X4, TERM_X_COUNT};
41 enum {TERM_PSI_A, TERM_PSI_B, TERM_EPS_C, TERM_EPS_D, TERM_PE_COUNT};
42 enum {JD_MINUS, JD_ZERO, JD_PLUS, JD_COUNT};
43 enum {SUN_TRANSIT, SUN_RISE, SUN_SET, SUN_COUNT};
44
45 const int l_subcount[L_COUNT] = {64,34,20,7,3,1};
46 const int b_subcount[B_COUNT] = {5,2};
47 const int r_subcount[R_COUNT] = {40,10,6,2,1};
```

Código A.17: Código fuente de la función spa_algorithm(parametros de entrada) en lenguaje *C*

```
char *spa_algorithm (int dDate[3], int dTime[3], double dGPS[3], int timezone)
{
    int result;

 ////////////////////PARAMETROS DE ENTRADA/////////////////////

    spa.day   = (double) dDate[0];
    spa.month  = (double) dDate[1];
    spa.year  = (double) dDate[2];

    spa.hour  = (double) dTime[0];
    spa.minute  = (double) dTime[1];
    spa.second   = (double) dTime[2];

    spa.timezone = (double) timezone;

    spa.latitude = dGPS[0];
    spa.longitude = dGPS[1];
    spa.elevation = dGPS[2];

    spa.delta_ut1     = 0;
    spa.delta_t       = 67;
    spa.pressure      = 1015;
    spa.temperature   = 16;
    spa.slope         = 0;
    spa.azm_rotation  = -10;
    spa.atmos_refract = 0.5667;
    spa.function      = SPA_ALL;

    // Obtener angulos: Azimutal y elevacion
    result = spa_calculate(&spa);

    if (result == 0)  // Validar errores del SPA
    {
        spa.zenith = (90-spa.zenith)*1;

        sprintf(cAngle, "%f", spa.azimuth);
        strcpy(anglesSPA, cAngle);

        sprintf(cAngle, "%f", spa.zenith);
        strcat(anglesSPA, ", ");
        strcat(anglesSPA, cAngle);
    }else
        printf("SPA Error Code: %d\n", result);

    return (char *) anglesSPA;
}
```

Código A.18: Código fuente de la función spa_calculate(spa_data *spa) de SPA en lenguaje *C*

```c
int spa_calculate(spa_data *spa)
{
    int result;

    result = validate_inputs(spa);

    if (result == 0)
    {
        spa->jd = julian_day (spa->year,   spa->month,  spa->day,       spa->hour,
                    spa->minute, spa->second, spa->delta_ut1, spa->timezone);

        calculate_geocentric_sun_right_ascension_and_declination(spa);

        spa->h  = observer_hour_angle(spa->nu, spa->longitude, spa->alpha);
        spa->xi = sun_equatorial_horizontal_parallax(spa->r);

        right_ascension_parallax_and_topocentric_dec(spa->latitude, spa->elevation, spa->xi,
                                spa->h, spa->delta, &(spa->del_alpha), &(spa->delta_prime));

        spa->alpha_prime = topocentric_right_ascension(spa->alpha, spa->del_alpha);
        spa->h_prime     = topocentric_local_hour_angle(spa->h, spa->del_alpha);

        spa->e0      = topocentric_elevation_angle(spa->latitude, spa->delta_prime, spa->h_prime);
        spa->del_e   = atmospheric_refraction_correction(spa->pressure, spa->temperature,
                                                         spa->atmos_refract, spa->e0);
        spa->e       = topocentric_elevation_angle_corrected(spa->e0, spa->del_e);

        spa->zenith        = topocentric_zenith_angle(spa->e);
        spa->azimuth_astro = topocentric_azimuth_angle_astro(spa->h_prime, spa->latitude,
                                                                spa->delta_prime);
        spa->azimuth       = topocentric_azimuth_angle(spa->azimuth_astro);

        if ((spa->function == SPA_ZA_INC) || (spa->function == SPA_ALL))
            spa->incidence  = surface_incidence_angle(spa->zenith, spa->azimuth_astro,
                                                      spa->azm_rotation, spa->slope);

        if ((spa->function == SPA_ZA_RTS) || (spa->function == SPA_ALL))
            calculate_eot_and_sun_rise_transit_set(spa);
    }

    return result;
}
```

Código A.19: Código para la definición de las condiciones periódicas de la Tierra

```
////////////////////////////////////////////////
///  Earth Periodic Terms
////////////////////////////////////////////////
const double L_TERMS[L_COUNT][L_MAX_SUBCOUNT][TERM_COUNT]=
{
    {
        {175347046.0,0,0},
        {3341656.0,4.6692568,6283.07585},
        {34894.0,4.6261,12566.1517},
        {3497.0,2.7441,5753.3849},
        {3418.0,2.8289,3.5231},
        {3136.0,3.6277,77713.7715},
        {2676.0,4.4181,7860.4194},
        {2343.0,6.1352,3930.2097},
        {1324.0,0.7425,11506.7698},
        {1273.0,2.0371,529.691},
        {1199.0,1.1096,1577.3435},
        {990,5.233,5884.927},
        {902,2.045,26.298},
        {857,3.508,398.149},
        {780,1.179,5223.694},
        {753,2.533,5507.553},
        {505,4.583,18849.228},
        {492,4.205,775.523},
        {357,2.92,0.067},
        {317,5.849,11790.629},
        {284,1.899,796.298},
        {271,0.315,10977.079},
        {243,0.345,5486.778},
        {206,4.806,2544.314},
        {205,1.869,5573.143},
        {202,2.458,6069.777},
        {156,0.833,213.299},
        {132,3.411,2942.463},
        {126,1.083,20.775},
        {115,0.645,0.98},
        {103,0.636,4694.003},
        {102,0.976,15720.839},
        {102,4.267,7.114},
        {99,6.21,2146.17},
        {98,0.68,155.42},
        {86,5.98,161000.69},
        {85,1.3,6275.96},
        {85,3.67,71430.7},
        {80,1.81,17260.15},
        {79,3.04,12036.46},
        {75,1.76,5088.63},
        {74,3.5,3154.69},
        {74,4.68,801.82},
        {70,0.83,9437.76},
        {62,3.98,8827.39},
        {61,1.82,7084.9},
        {57,2.78,6286.6},
```

```
        {56,4.39,14143.5},
        {56,3.47,6279.55},
        {52,0.19,12139.55},
        {52,1.33,1748.02},
        {51,0.28,5856.48},
        {49,0.49,1194.45},
        {41,5.37,8429.24},
        {41,2.4,19651.05},
        {39,6.17,10447.39},
        {37,6.04,10213.29},
        {37,2.57,1059.38},
        {36,1.71,2352.87},
        {36,1.78,6812.77},
        {33,0.59,17789.85},
        {30,0.44,83996.85},
        {30,2.74,1349.87},
        {25,3.16,4690.48}
    },
    {
        {628331966747.0,0,0},
        {206059.0,2.678235,6283.07585},
        {4303.0,2.6351,12566.1517},
        {425.0,1.59,3.523},
        {119.0,5.796,26.298},
        {109.0,2.966,1577.344},
        {93,2.59,18849.23},
        {72,1.14,529.69},
        {68,1.87,398.15},
        {67,4.41,5507.55},
        {59,2.89,5223.69},
        {56,2.17,155.42},
        {45,0.4,796.3},
        {36,0.47,775.52},
        {29,2.65,7.11},
        {21,5.34,0.98},
        {19,1.85,5486.78},
        {19,4.97,213.3},
        {17,2.99,6275.96},
        {16,0.03,2544.31},
        {16,1.43,2146.17},
        {15,1.21,10977.08},
        {12,2.83,1748.02},
        {12,3.26,5088.63},
        {12,5.27,1194.45},
        {12,2.08,4694},
        {11,0.77,553.57},
        {10,1.3,6286.6},
        {10,4.24,1349.87},
        {9,2.7,242.73},
        {9,5.64,951.72},
        {8,5.3,2352.87},
        {6,2.65,9437.76},
        {6,4.67,4690.48}
    },
    {
```

```
        {52919.0,0,0},
        {8720.0,1.0721,6283.0758},
        {309.0,0.867,12566.152},
        {27,0.05,3.52},
        {16,5.19,26.3},
        {16,3.68,155.42},
        {10,0.76,18849.23},
        {9,2.06,77713.77},
        {7,0.83,775.52},
        {5,4.66,1577.34},
        {4,1.03,7.11},
        {4,3.44,5573.14},
        {3,5.14,796.3},
        {3,6.05,5507.55},
        {3,1.19,242.73},
        {3,6.12,529.69},
        {3,0.31,398.15},
        {3,2.28,553.57},
        {2,4.38,5223.69},
        {2,3.75,0.98}
    },
    {
        {289.0,5.844,6283.076},
        {35,0,0},
        {17,5.49,12566.15},
        {3,5.2,155.42},
        {1,4.72,3.52},
        {1,5.3,18849.23},
        {1,5.97,242.73}
    },
    {
        {114.0,3.142,0},
        {8,4.13,6283.08},
        {1,3.84,12566.15}
    },
    {
        {1,3.14,0}
    }
};

const double B_TERMS[B_COUNT][B_MAX_SUBCOUNT][TERM_COUNT]=
{
    {
        {280.0,3.199,84334.662},
        {102.0,5.422,5507.553},
        {80,3.88,5223.69},
        {44,3.7,2352.87},
        {32,4,1577.34}
    },
    {
        {9,3.9,5507.55},
        {6,1.73,5223.69}
    }
};

```

```
const double R_TERMS[R_COUNT][R_MAX_SUBCOUNT][TERM_COUNT]=
{
    {
        {100013989.0,0,0},
        {1670700.0,3.0984635,6283.07585},
        {13956.0,3.05525,12566.1517},
        {3084.0,5.1985,77713.7715},
        {1628.0,1.1739,5753.3849},
        {1576.0,2.8469,7860.4194},
        {925.0,5.453,11506.77},
        {542.0,4.564,3930.21},
        {472.0,3.661,5884.927},
        {346.0,0.964,5507.553},
        {329.0,5.9,5223.694},
        {307.0,0.299,5573.143},
        {243.0,4.273,11790.629},
        {212.0,5.847,1577.344},
        {186.0,5.022,10977.079},
        {175.0,3.012,18849.228},
        {110.0,5.055,5486.778},
        {98,0.89,6069.78},
        {86,5.69,15720.84},
        {86,1.27,161000.69},
        {65,0.27,17260.15},
        {63,0.92,529.69},
        {57,2.01,83996.85},
        {56,5.24,71430.7},
        {49,3.25,2544.31},
        {47,2.58,775.52},
        {45,5.54,9437.76},
        {43,6.01,6275.96},
        {39,5.36,4694},
        {38,2.39,8827.39},
        {37,0.83,19651.05},
        {37,4.9,12139.55},
        {36,1.67,12036.46},
        {35,1.84,2942.46},
        {33,0.24,7084.9},
        {32,0.18,5088.63},
        {32,1.78,398.15},
        {28,1.21,6286.6},
        {28,1.9,6279.55},
        {26,4.59,10447.39}
    },
    {
        {103019.0,1.10749,6283.07585},
        {1721.0,1.0644,12566.1517},
        {702.0,3.142,0},
        {32,1.02,18849.23},
        {31,2.84,5507.55},
        {25,1.32,5223.69},
        {18,1.42,1577.34},
        {10,5.91,10977.08},
        {9,1.42,6275.96},
        {9,0.27,5486.78}
```

```
    },
    {
        {4359.0,5.7846,6283.0758},
        {124.0,5.579,12566.152},
        {12,3.14,0},
        {9,3.63,77713.77},
        {6,1.87,5573.14},
        {3,5.47,18849.23}
    },
    {
        {145.0,4.273,6283.076},
        {7,3.92,12566.15}
    },
    {
        {4,2.56,6283.08}
    }
};
```

Código A.20: Código para definir las condiciones periódicas para la notación en longitud y oblicuidad

```
//////////////////////////////////////////////////////////////////
///  Periodic Terms for the notation in longitude and obliquity
//////////////////////////////////////////////////////////////////
const int Y_TERMS[Y_COUNT][TERM_Y_COUNT]=
{
    {0,0,0,0,1},
    {-2,0,0,2,2},
    {0,0,0,2,2},
    {0,0,0,0,2},
    {0,1,0,0,0},
    {0,0,1,0,0},
    {-2,1,0,2,2},
    {0,0,0,2,1},
    {0,0,1,2,2},
    {-2,-1,0,2,2},
    {-2,0,1,0,0},
    {-2,0,0,2,1},
    {0,0,-1,2,2},
    {2,0,0,0,0},
    {0,0,1,0,1},
    {2,0,-1,2,2},
    {0,0,-1,0,1},
    {0,0,1,2,1},
    {-2,0,2,0,0},
    {0,0,-2,2,1},
    {2,0,0,2,2},
    {0,0,2,2,2},
    {0,0,2,0,0},
    {-2,0,1,2,2},
    {0,0,0,2,0},
    {-2,0,0,2,0},
    {0,0,-1,2,1},
    {0,2,0,0,0},
```

```
    {2,0,-1,0,1},
    {-2,2,0,2,2},
    {0,1,0,0,1},
    {-2,0,1,0,1},
    {0,-1,0,0,1},
    {0,0,2,-2,0},
    {2,0,-1,2,1},
    {2,0,1,2,2},
    {0,1,0,2,2},
    {-2,1,1,0,0},
    {0,-1,0,2,2},
    {2,0,0,2,1},
    {2,0,1,0,0},
    {-2,0,2,2,2},
    {-2,0,1,2,1},
    {2,0,-2,0,1},
    {2,0,0,0,1},
    {0,-1,1,0,0},
    {-2,-1,0,2,1},
    {-2,0,0,0,1},
    {0,0,2,2,1},
    {-2,0,2,0,1},
    {-2,1,0,2,1},
    {0,0,1,-2,0},
    {-1,0,1,0,0},
    {-2,1,0,0,0},
    {1,0,0,0,0},
    {0,0,1,2,0},
    {0,0,-2,2,2},
    {-1,-1,1,0,0},
    {0,1,1,0,0},
    {0,-1,1,2,2},
    {2,-1,-1,2,2},
    {0,0,3,2,2},
    {2,-1,0,2,2},
};

const double PE_TERMS[Y_COUNT][TERM_PE_COUNT]={
    {-171996,-174.2,92025,8.9},
    {-13187,-1.6,5736,-3.1},
    {-2274,-0.2,977,-0.5},
    {2062,0.2,-895,0.5},
    {1426,-3.4,54,-0.1},
    {712,0.1,-7,0},
    {-517,1.2,224,-0.6},
    {-386,-0.4,200,0},
    {-301,0,129,-0.1},
    {217,-0.5,-95,0.3},
    {-158,0,0,0},
    {129,0.1,-70,0},
    {123,0,-53,0},
    {63,0,0,0},
    {63,0.1,-33,0},
    {-59,0,26,0},
    {-58,-0.1,32,0},
```

```
    {-51,0,27,0},
    {48,0,0,0},
    {46,0,-24,0},
    {-38,0,16,0},
    {-31,0,13,0},
    {29,0,0,0},
    {29,0,-12,0},
    {26,0,0,0},
    {-22,0,0,0},
    {21,0,-10,0},
    {17,-0.1,0,0},
    {16,0,-8,0},
    {-16,0.1,7,0},
    {-15,0,9,0},
    {-13,0,7,0},
    {-12,0,6,0},
    {11,0,0,0},
    {-10,0,5,0},
    {-8,0,3,0},
    {7,0,-3,0},
    {-7,0,0,0},
    {-7,0,3,0},
    {-7,0,3,0},
    {6,0,0,0},
    {6,0,-3,0},
    {6,0,-3,0},
    {-6,0,3,0},
    {-6,0,3,0},
    {5,0,0,0},
    {-5,0,3,0},
    {-5,0,3,0},
    {-5,0,3,0},
    {4,0,0,0},
    {4,0,0,0},
    {4,0,0,0},
    {-4,0,0,0},
    {-4,0,0,0},
    {-4,0,0,0},
    {3,0,0,0},
    {-3,0,0,0},
    {-3,0,0,0},
    {-3,0,0,0},
    {-3,0,0,0},
    {-3,0,0,0},
    {-3,0,0,0},
    {-3,0,0,0},
};
```

Código A.21: Código para la conversión de Radianes a Grados y viceversa

```
double rad2deg(double radians)
{
    return (180.0/PI)*radians;
}

double deg2rad(double degrees)
{
    return (PI/180.0)*degrees;
}

int integer(double value)
{
 return value;
}
```

Código A.22: Código para limitar los grados poniendo limites máximos y mínimos

```
double limit_degrees(double degrees)
{
    double limited;

    degrees /= 360.0;
    limited = 360.0*(degrees-floor(degrees));
    if (limited < 0) limited += 360.0;

    return limited;
}

double limit_degrees180pm(double degrees)
{
    double limited;

    degrees /= 360.0;
    limited = 360.0*(degrees-floor(degrees))
    if      (limited < -180.0) limited += 360.0;
    else if (limited >  180.0) limited -= 360.0;

    return limited;
}

double limit_degrees180(double degrees)
{
    double limited;

    degrees /= 180.0;
    limited = 180.0*(degrees-floor(degrees));
    if (limited < 0) limited += 180.0;

    return limited;
}
```

Código A.23: Código para definir limite de tiempo en minutos y conversión de días fraccionarios a hora local

```
double limit_zero2one(double value)
{
    double limited;

    limited = value - floor(value);
    if (limited < 0) limited += 1.0;

    return limited;
}

double limit_minutes(double minutes)
{
    double limited=minutes;

    if      (limited < -20.0) limited += 1440.0;
    else if (limited >  20.0) limited -= 1440.0;

    return limited;
}

double dayfrac_to_local_hr(double dayfrac, double timezone)
{
    return 24.0*limit_zero2one(dayfrac + timezone/24.0);
}

double third_order_polynomial(double a, double b, double c, double d, double x)
{
    return ((a*x + b)*x + c)*x + d;
}
```

Código A.24: Código para validar las entradas del SPA de acuerdo a ciertos parámetros

```
int validate_inputs(spa_data *spa)
{
    if ((spa->year        < -2000) || (spa->year        > 6000)) return 1;
    if ((spa->month       < 1    ) || (spa->month       > 12  )) return 2;
    if ((spa->day         < 1    ) || (spa->day         > 31  )) return 3;
    if ((spa->hour        < 0    ) || (spa->hour        > 24  )) return 4;
    if ((spa->minute      < 0    ) || (spa->minute      > 59  )) return 5;
    if ((spa->second      < 0    ) || (spa->second      >=60  )) return 6;
    if ((spa->pressure    < 0    ) || (spa->pressure    > 5000)) return 12;
    if ((spa->temperature <= -273) || (spa->temperature > 6000)) return 13;
    if ((spa->delta_ut1   <= -1  ) || (spa->delta_ut1   >= 1  )) return 17;
    if ((spa->hour        == 24  ) && (spa->minute      > 0   )) return 5;
    if ((spa->hour        == 24  ) && (spa->second      > 0   )) return 6;

    if (fabs(spa->delta_t)       > 8000    ) return 7;
    if (fabs(spa->timezone)      > 18      ) return 8;
    if (fabs(spa->longitude)     > 180     ) return 9;
    if (fabs(spa->latitude)      > 90      ) return 10;
```

```
    if (fabs(spa->atmos_refract) > 5       ) return 16;
    if (     spa->elevation      < -6500000) return 11;

    if ((spa->function == SPA_ZA_INC) || (spa->function == SPA_ALL))
    {
        if (fabs(spa->slope)         > 360) return 14;
        if (fabs(spa->azm_rotation)  > 360) return 15;
    }

    return 0;
}
```

Código A.25: Código para declarar los días Julianos dando parámetros de fecha y hora por partes

```
double julian_day (int year, int month, int day, int hour, int minute, double second, double dut1, double tz)
{
    double day_decimal, julian_day, a;

    day_decimal = day + (hour - tz + (minute + (second + dut1)/60.0)/60.0)/24.0;

    if (month < 3) {
        month += 12;
        year--;
    }

    julian_day = integer(365.25*(year+4716.0)) + integer(30.6001*(month+1)) + day_decimal - 1524.5;

    if (julian_day > 2299160.0) {
        a = integer(year/100);
        julian_day += (2 - a + integer(a/4));
    }

    return julian_day;
}
```

Código A.26: Código para definir efemérides de días Julianos en centenario y milenario

```
double julian_century(double jd)
{
    return (jd-2451545.0)/36525.0;
}

double julian_ephemeris_day(double jd, double delta_t)
{
    return jd+delta_t/86400.0;
}

double julian_ephemeris_century(double jde)
{
    return (jde - 2451545.0)/36525.0;
}

double julian_ephemeris_millennium(double jce)
{
    return (jce/10.0);
}
```

Código A.27: Código para definir valores de la Tierra: Latitud y Longitud heliocéntrica; radio de Tierra; Latitud y Altitud geocéntrica y plazo del período de la Tierra

```
double earth_periodic_term_summation(const double terms[][TERM_COUNT], int count, double jme)
{
    int i;
    double sum=0;

    for (i = 0; i < count; i++)
        sum += terms[i][TERM_A]*cos(terms[i][TERM_B]+terms[i][TERM_C]*jme);

    return sum;
}

double earth_values(double term_sum[], int count, double jme)
{
    int i;
    double sum=0;

    for (i = 0; i < count; i++)
        sum += term_sum[i]*pow(jme, i);

    sum /= 1.0e8;

    return sum;
}

double earth_heliocentric_longitude(double jme)
{
    double sum[L_COUNT];
    int i;
```

```
29
30      for (i = 0; i < L_COUNT; i++)
31          sum[i] = earth_periodic_term_summation(L_TERMS[i], l_subcount[i], jme);
32
33      return limit_degrees(rad2deg(earth_values(sum, L_COUNT, jme)));
34
35  }
36
37  double earth_heliocentric_latitude(double jme)
38  {
39      double sum[B_COUNT];
40      int i;
41
42      for (i = 0; i < B_COUNT; i++)
43          sum[i] = earth_periodic_term_summation(B_TERMS[i], b_subcount[i], jme);
44
45      return rad2deg(earth_values(sum, B_COUNT, jme));
46
47  }
48
49  double earth_radius_vector(double jme)
50  {
51      double sum[R_COUNT];
52      int i;
53
54      for (i = 0; i < R_COUNT; i++)
55          sum[i] = earth_periodic_term_summation(R_TERMS[i], r_subcount[i], jme);
56
57      return earth_values(sum, R_COUNT, jme);
58
59  }
60
61  double geocentric_longitude(double l)
62  {
63      double theta = l + 180.0;
64
65      if (theta >= 360.0) theta -= 360.0;
66
67      return theta;
68  }
69
70  double geocentric_latitude(double b)
71  {
72      return -b;
73  }
74
75  double mean_elongation_moon_sun(double jce)
76  {
77      return third_order_polynomial(1.0/189474.0, -0.0019142, 445267.11148, 297.85036, jce);
78  }
```

Código A.28: Código para definir el manejo de anomalías del Sol y otros parámetros para la luna

```
double mean_anomaly_sun(double jce)
{
    return third_order_polynomial(-1.0/300000.0, -0.0001603, 35999.05034, 357.52772, jce);
}
double mean_anomaly_moon(double jce)
{
    return third_order_polynomial(1.0/56250.0, 0.0086972, 477198.867398, 134.96298, jce);
}

double argument_latitude_moon(double jce)
{
    return third_order_polynomial(1.0/327270.0, -0.0036825, 483202.017538, 93.27191, jce);
}

double ascending_longitude_moon(double jce)
{
    return third_order_polynomial(1.0/450000.0, 0.0020708, -1934.136261, 125.04452, jce);
}

double xy_term_summation(int i, double x[TERM_X_COUNT])
{
    int j;
    double sum=0;

    for (j = 0; j < TERM_Y_COUNT; j++)
        sum += x[j]*Y_TERMS[i][j];

    return sum;
}
```

Código A.29: Código para definir el movimiento del eje de la tierra por el que se inclina más o menos sobre el plano de la eclíptica; su longitud y oblicuidad

```
void nutation_longitude_and_obliquity(double jce, double x[TERM_X_COUNT], double *del_psi,
                                                                           double *del_epsilon)
{
    int i;
    double xy_term_sum, sum_psi=0, sum_epsilon=0;

    for (i = 0; i < Y_COUNT; i++) {
        xy_term_sum  = deg2rad(xy_term_summation(i, x));
        sum_psi     += (PE_TERMS[i][TERM_PSI_A] + jce*PE_TERMS[i][TERM_PSI_B])*sin(xy_term_sum);
        sum_epsilon += (PE_TERMS[i][TERM_EPS_C] + jce*PE_TERMS[i][TERM_EPS_D])*cos(xy_term_sum);
    }

    *del_psi     = sum_psi     / 36000000.0;
    *del_epsilon = sum_epsilon / 36000000.0;
}

double ecliptic_mean_obliquity(double jme)
{
```

```
    double u = jme/10.0;

    return 84381.448 + u*(-4680.93 + u*(-1.55 + u*(1999.25 + u*(-51.38 + u*(-249.67 +
                      u*(  -39.05 + u*( 7.12 + u*(  27.87 + u*(  5.79 + u*2.45)))))))));
}

double ecliptic_true_obliquity(double delta_epsilon, double epsilon0)
{
    return delta_epsilon + epsilon0/3600.0;
}

double aberration_correction(double r)
{
    return -20.4898 / (3600.0*r);
}

double apparent_sun_longitude(double theta, double delta_psi, double delta_tau)
{
    return theta + delta_psi + delta_tau;
}

double greenwich_mean_sidereal_time (double jd, double jc)
{
    return limit_degrees(280.46061837 + 360.98564736629 * (jd - 2451545.0) +
                                      jc*jc*(0.000387933 - jc/38710000.0));
}

double greenwich_sidereal_time (double nu0, double delta_psi, double epsilon)
{
    return nu0 + delta_psi*cos(deg2rad(epsilon));
}

double geocentric_right_ascension(double lamda, double epsilon, double beta)
{
    double lamda_rad   = deg2rad(lamda);
    double epsilon_rad = deg2rad(epsilon);

    return limit_degrees(rad2deg(atan2(sin(lamda_rad)*cos(epsilon_rad) -
                                      tan(deg2rad(beta))*sin(epsilon_rad), cos(lamda_rad))));
}

double geocentric_declination(double beta, double epsilon, double lamda)
{
    double beta_rad    = deg2rad(beta);
    double epsilon_rad = deg2rad(epsilon);

    return rad2deg(asin(sin(beta_rad)*cos(epsilon_rad) +
                        cos(beta_rad)*sin(epsilon_rad)*sin(deg2rad(lamda))));
}

double observer_hour_angle(double nu, double longitude, double alpha_deg)
{
    return limit_degrees(nu + longitude - alpha_deg);
}

```

```
74 double sun_equatorial_horizontal_parallax(double r)
75 {
76     return 8.794 / (3600.0 * r);
77 }
```

Código A.30: Código para la definición de los parámetros parallax y topocentricos

```
1  void right_ascension_parallax_and_topocentric_dec(double latitude, double elevation,
2          double xi, double h, double delta, double *delta_alpha, double *delta_prime)
3  {
4      double delta_alpha_rad;
5      double lat_rad   = deg2rad(latitude);
6      double xi_rad    = deg2rad(xi);
7      double h_rad     = deg2rad(h);
8      double delta_rad = deg2rad(delta);
9      double u = atan(0.99664719 * tan(lat_rad));
10     double y = 0.99664719 * sin(u) + elevation*sin(lat_rad)/6378140.0;
11     double x =              cos(u) + elevation*cos(lat_rad)/6378140.0;
12
13     delta_alpha_rad =      atan2(                - x*sin(xi_rad) *sin(h_rad),
14                                  cos(delta_rad) - x*sin(xi_rad) *cos(h_rad));
15
16     *delta_prime = rad2deg(atan2((sin(delta_rad) - y*sin(xi_rad))*cos(delta_alpha_rad),
17                                  cos(delta_rad) - x*sin(xi_rad) *cos(h_rad)));
18
19     *delta_alpha = rad2deg(delta_alpha_rad);
20 }
21
22 double topocentric_right_ascension(double alpha_deg, double delta_alpha)
23 {
24     return alpha_deg + delta_alpha;
25 }
26
27 double topocentric_local_hour_angle(double h, double delta_alpha)
28 {
29     return h - delta_alpha;
30 }
31
32 double topocentric_elevation_angle(double latitude, double delta_prime, double h_prime)
33 {
34     double lat_rad         = deg2rad(latitude);
35     double delta_prime_rad = deg2rad(delta_prime);
36
37     return rad2deg(asin(sin(lat_rad)*sin(delta_prime_rad) +
38                         cos(lat_rad)*cos(delta_prime_rad) * cos(deg2rad(h_prime))));
39 }
40
41 double atmospheric_refraction_correction(double pressure, double temperature,
42                                          double atmos_refract, double e0)
43 {
44     double del_e = 0;
45
46     if (e0 >= -1*(SUN_RADIUS + atmos_refract))
```

```
        del_e = (pressure / 1010.0) * (283.0 / (273.0 + temperature)) *
                1.02 / (60.0 * tan(deg2rad(e0 + 10.3/(e0 + 5.11))));

    return del_e;
}

double topocentric_elevation_angle_corrected(double e0, double delta_e)
{
    return e0 + delta_e;
}

double topocentric_zenith_angle(double e)
{
    return 90.0 - e;
}

double topocentric_azimuth_angle_astro(double h_prime, double latitude, double delta_prime)
{
    double h_prime_rad = deg2rad(h_prime);
    double lat_rad     = deg2rad(latitude);

    return limit_degrees(rad2deg(atan2(sin(h_prime_rad),
                         cos(h_prime_rad)*sin(lat_rad) - tan(deg2rad(delta_prime))*cos(lat_rad))));
}

double topocentric_azimuth_angle(double azimuth_astro)
{
    return limit_degrees(azimuth_astro + 180.0);
}

double surface_incidence_angle(double zenith, double azimuth_astro, double azm_rotation,
                                                                     double slope)
{
    double zenith_rad = deg2rad(zenith);
    double slope_rad  = deg2rad(slope);

    return rad2deg(acos(cos(zenith_rad)*cos(slope_rad)  +
                        sin(slope_rad )*sin(zenith_rad) * cos(deg2rad(azimuth_astro - azm_rotation))));
}

double sun_mean_longitude(double jme)
{
    return limit_degrees(280.4664567 + jme*(360007.6982779 + jme*(0.03032028 +
                    jme*(1/49931.0   + jme*(-1/15300.0     + jme*(-1/2000000.0))))));
}

double eot(double m, double alpha, double del_psi, double epsilon)
{
    return limit_minutes(4.0*(m - 0.0057183 - alpha + del_psi*cos(deg2rad(epsilon))));
}

double approx_sun_transit_time(double alpha_zero, double longitude, double nu)
{
    return (alpha_zero - longitude - nu) / 360.0;
}
```

```
102
103 double sun_hour_angle_at_rise_set(double latitude, double delta_zero, double h0_prime)
104 {
105     double h0             = -99999;
106     double latitude_rad   = deg2rad(latitude);
107     double delta_zero_rad = deg2rad(delta_zero);
108     double argument       = (sin(deg2rad(h0_prime)) - sin(latitude_rad)*sin(delta_zero_rad)) /
109                                                       (cos(latitude_rad)*cos(delta_zero_rad));
110
111     if (fabs(argument) <= 1) h0 = limit_degrees180(rad2deg(acos(argument)));
112
113     return h0;
114 }
```

Código A.31: Código para calcular y definir la aproximación de elevación y colocación del Sol

```
1  void approx_sun_rise_and_set(double *m_rts, double h0)
2  {
3      double h0_dfrac = h0/360.0;
4
5      m_rts[SUN_RISE]    = limit_zero2one(m_rts[SUN_TRANSIT] - h0_dfrac);
6      m_rts[SUN_SET]     = limit_zero2one(m_rts[SUN_TRANSIT] + h0_dfrac);
7      m_rts[SUN_TRANSIT] = limit_zero2one(m_rts[SUN_TRANSIT]);
8  }
9
10 double rts_alpha_delta_prime(double *ad, double n)
11 {
12     double a = ad[JD_ZERO] - ad[JD_MINUS];
13     double b = ad[JD_PLUS] - ad[JD_ZERO];
14
15     if (fabs(a) >= 2.0) a = limit_zero2one(a);
16     if (fabs(b) >= 2.0) b = limit_zero2one(b);
17
18     return ad[JD_ZERO] + n * (a + b + (b-a)*n)/2.0;
19 }
20
21 double rts_sun_altitude(double latitude, double delta_prime, double h_prime)
22 {
23     double latitude_rad    = deg2rad(latitude);
24     double delta_prime_rad = deg2rad(delta_prime);
25
26     return rad2deg(asin(sin(latitude_rad)*sin(delta_prime_rad) +
27                         cos(latitude_rad)*cos(delta_prime_rad)*cos(deg2rad(h_prime))));
28 }
29
30 double sun_rise_and_set(double *m_rts,   double *h_rts,   double *delta_prime, double latitude,
31                         double *h_prime, double h0_prime, int sun)
32 {
33     return m_rts[sun] + (h_rts[sun] - h0_prime) /
34           (360.0*cos(deg2rad(delta_prime[sun]))*cos(deg2rad(latitude))*sin(deg2rad(h_prime[sun])));
35 }
```

Código A.32: Código para calcular parámetros SPA de ascensión (alfa) y la declinación (delta)

```
void calculate_geocentric_sun_right_ascension_and_declination(spa_data *spa)
{
    double x[TERM_X_COUNT];

    spa->jc = julian_century(spa->jd);

    spa->jde = julian_ephemeris_day(spa->jd, spa->delta_t);
    spa->jce = julian_ephemeris_century(spa->jde);
    spa->jme = julian_ephemeris_millennium(spa->jce);

    spa->l = earth_heliocentric_longitude(spa->jme);
    spa->b = earth_heliocentric_latitude(spa->jme);
    spa->r = earth_radius_vector(spa->jme);

    spa->theta = geocentric_longitude(spa->l);
    spa->beta  = geocentric_latitude(spa->b);

    x[TERM_X0] = spa->x0 = mean_elongation_moon_sun(spa->jce);
    x[TERM_X1] = spa->x1 = mean_anomaly_sun(spa->jce);
    x[TERM_X2] = spa->x2 = mean_anomaly_moon(spa->jce);
    x[TERM_X3] = spa->x3 = argument_latitude_moon(spa->jce);
    x[TERM_X4] = spa->x4 = ascending_longitude_moon(spa->jce);

    nutation_longitude_and_obliquity(spa->jce, x, &(spa->del_psi), &(spa->del_epsilon));

    spa->epsilon0 = ecliptic_mean_obliquity(spa->jme);
    spa->epsilon  = ecliptic_true_obliquity(spa->del_epsilon, spa->epsilon0);

    spa->del_tau   = aberration_correction(spa->r);
    spa->lamda     = apparent_sun_longitude(spa->theta, spa->del_psi, spa->del_tau);
    spa->nu0       = greenwich_mean_sidereal_time (spa->jd, spa->jc);
    spa->nu        = greenwich_sidereal_time (spa->nu0, spa->del_psi, spa->epsilon);

    spa->alpha = geocentric_right_ascension(spa->lamda, spa->epsilon, spa->beta);
    spa->delta = geocentric_declination(spa->beta, spa->epsilon, spa->lamda);
}
```

Código A.33: Código para calcular la ecuación del tiempo (EOT) y elevacion del Sol; transición y configuración del RTS

```
void calculate_eot_and_sun_rise_transit_set(spa_data *spa)
{
    spa_data sun_rts;
    double nu, m, h0, n;
    double alpha[JD_COUNT], delta[JD_COUNT];
    double m_rts[SUN_COUNT], nu_rts[SUN_COUNT], h_rts[SUN_COUNT];
    double alpha_prime[SUN_COUNT], delta_prime[SUN_COUNT], h_prime[SUN_COUNT];
    double h0_prime = -1*(SUN_RADIUS + spa->atmos_refract);
    int i;

    sun_rts  = *spa;
```

```
    m        = sun_mean_longitude(spa->jme);
    spa->eot = eot(m, spa->alpha, spa->del_psi, spa->epsilon);

    sun_rts.hour = sun_rts.minute = sun_rts.second = 0;
    sun_rts.delta_ut1 = sun_rts.timezone = 0.0;

    sun_rts.jd = julian_day (sun_rts.year,   sun_rts.month,  sun_rts.day,       sun_rts.hour,
                        sun_rts.minute, sun_rts.second, sun_rts.delta_ut1, sun_rts.timezone);

    calculate_geocentric_sun_right_ascension_and_declination(&sun_rts);
    nu = sun_rts.nu;

    sun_rts.delta_t = 0;
    sun_rts.jd--;
    for (i = 0; i < JD_COUNT; i++) {
        calculate_geocentric_sun_right_ascension_and_declination(&sun_rts);
        alpha[i] = sun_rts.alpha;
        delta[i] = sun_rts.delta;
        sun_rts.jd++;
    }

    m_rts[SUN_TRANSIT] = approx_sun_transit_time(alpha[JD_ZERO], spa->longitude, nu);
    h0 = sun_hour_angle_at_rise_set(spa->latitude, delta[JD_ZERO], h0_prime);

    if (h0 >= 0) {

        approx_sun_rise_and_set(m_rts, h0);

        for (i = 0; i < SUN_COUNT; i++) {

            nu_rts[i]      = nu + 360.985647*m_rts[i];

            n              = m_rts[i] + spa->delta_t/86400.0;
            alpha_prime[i] = rts_alpha_delta_prime(alpha, n);
            delta_prime[i] = rts_alpha_delta_prime(delta, n);

            h_prime[i]     = limit_degrees180pm(nu_rts[i] + spa->longitude - alpha_prime[i]);
            h_rts[i]       = rts_sun_altitude(spa->latitude, delta_prime[i], h_prime[i]);
        }

        spa->srha = h_prime[SUN_RISE];
        spa->ssha = h_prime[SUN_SET];
        spa->sta  = h_rts[SUN_TRANSIT];

        spa->suntransit = dayfrac_to_local_hr(m_rts[SUN_TRANSIT] - h_prime[SUN_TRANSIT] / 360.0,
                                              spa->timezone);

        spa->sunrise = dayfrac_to_local_hr(sun_rise_and_set(m_rts, h_rts, delta_prime,
                           spa->latitude, h_prime, h0_prime, SUN_RISE), spa->timezone);

        spa->sunset  = dayfrac_to_local_hr(sun_rise_and_set(m_rts, h_rts, delta_prime,
                           spa->latitude, h_prime, h0_prime, SUN_SET),  spa->timezone);

    } else spa->srha= spa->ssha= spa->sta= spa->suntransit= spa->sunrise= spa->sunset= -99999;
}
```

A.4. Resultados de la prueba del software

Tabla A.1: Resultados de los ángulos de posición solar calculados por el algoritmo SPA.

Fecha/Hora	**SolarTracking**	
01/Agosto/2015	Azimutal (grados)	Elevación (grados)
07:10	69.896104	0.000000
07:15	70.419458	0.000000
07:20	70.787628	0.588340
07:25	71.221914	1.580443
07:30	71.637808	2.592160
07:35	72.054005	3.649943
07:40	72.484102	4.780635
07:45	72.864466	5.807593
07:50	73.265943	6.916158
07:55	73.669333	8.053660
08:00	74.047668	9.140963
08:05	74.424774	10.24372
08:10	74.811927	11.395008
08:15	75.184148	12.519738
08:20	75.550597	13.643835
08:25	75.906677	14.751644
08:30	76.277932	15.922737
08:35	76.622336	17.023414
08:40	76.990123	18.213691
08:45	77.336797	19.349235
08:50	77.680175	20.486629
08:55	78.021549	21.629593
09:00	78.345349	22.724442
09:05	78.700163	23.935677
09:10	79.018937	25.033605
09:15	79.361915	26.225028
09:20	79.692880	27.383081
09:25	80.006599	28.488883
09:30	80.346878	29.695676
09:35	80.649944	30.776906

Continúa en la siguiente página...

Tabla A.1: ...Continuación de la página anterior.

Fecha/Hora	SolarTracking	
09:40	80.974351	31.940115
09:45	81.298778	33.108033
09:50	81.621210	34.273354
09:55	81.949454	35.462709
10:00	82.261294	36.595411
10:05	82.582604	37.763542
10:10	82.904002	38.932894
10:15	83.210885	40.048527
10:20	83.533388	41.219426
10:25	83.869838	42.437554
10:30	84.194717	43.610057
10:35	84.506112	44.728770
10:40	84.824897	45.867442
10:45	85.156070	47.042026
10:50	85.490098	48.216795
10:55	85.834190	49.415931
11:00	86.159888	50.537400
11:05	86.522146	51.768688
11:10	86.864002	52.914571
11:15	87.228592	54.115306
11:20	87.582138	55.257988
11:25	87.944217	56.405033
11:30	88.320637	57.571753
11:35	88.708985	58.742318
11:40	89.118900	59.944646
11:45	89.538495	61.135590
11:50	89.969105	62.314595
11:55	90.404950	63.462416
12:00	90.857726	64.606356
12:05	91.357039	65.808240
12:10	91.848720	66.932834
12:15	92.405701	68.134957
12:20	92.970854	69.277387
12:25	93.621640	70.501391
12:30	94.267642	71.623150

Continúa en la siguiente página...

Tabla A.1: ...Continuación de la página anterior.

Fecha/Hora	SolarTracking	
12:35	95.005817	72.798155
12:40	95.810863	73.960552
12:45	96.718050	75.132604
12:50	97.764317	76.325666
12:55	98.942865	77.492326
13:00	100.235337	78.589501
13:05	101.930593	79.792886
13:10	103.870208	80.911640
13:15	106.424295	82.072698
13:20	109.511087	83.138872
13:25	113.797712	84.213511
13:30	120.198226	85.286234
13:35	129.801070	86.254562
13:40	144.665714	87.039880
13:45	168.013514	87.522113
13:50	194.942691	87.491035
13:55	217.355084	86.960660
14:00	231.774478	86.123467
14:05	241.015921	85.105257
14:10	246.759240	84.075940
14:15	250.964404	82.967669
14:20	253.997909	81.865077
14:25	256.439493	80.701808
14:30	258.236925	79.627038
14:35	259.858749	78.445876
14:40	261.220354	77.259178
14:45	262.312416	76.149706
14:50	263.304328	74.998890
14:55	264.232846	73.779930
15:00	264.995255	72.663656
15:05	265.718896	71.499709
15:10	266.384186	70.331071
15:15	267.011977	69.137734
15:20	267.581666	67.975250
15:25	268.130735	66.781032

Continúa en la siguiente página...

Tabla A.1: ...Continuación de la página anterior.

Fecha/Hora	SolarTracking	
15:30	268.628700	65.633330
15:35	269.133438	64.407203
15:40	269.580007	63.270771
15:45	270.026623	62.083605
15:50	270.453873	60.904278
15:55	270.869122	59.717055
16:00	271.243913	58.612048
16:05	271.627851	57.448679
16:10	272.026913	56.207656
16:15	272.387407	55.060302
16:20	272.749971	53.882065
16:25	273.105826	52.704345
16:30	273.455789	51.526928
16:35	273.776759	50.431790
16:40	274.115128	49.263136
16:45	274.458427	48.063852
16:50	274.798523	46.864980
16:55	275.128349	45.690138
17:00	275.456336	44.515887
17:05	275.766908	43.397000
17:10	276.086777	42.239639
17:15	276.409655	41.067453
17:20	276.728304	39.907782
17:25	277.040930	38.768341
17:30	277.371369	37.563733
17:35	277.682473	36.429757
17:40	278.003373	35.262047
17:45	278.324630	34.094987
17:50	278.644429	32.936819
17:55	278.967203	31.772028
18:00	279.287969	30.619882
18:05	279.609877	29.468966
18:10	279.924728	28.349887
18:15	280.266315	27.143652
18:20	280.580378	26.042429

Continúa en la siguiente página...

Tabla A.1: ...Continuación de la página anterior.

Fecha/Hora	SolarTracking	
18:25	280.919583	24.862028
18:30	281.254590	23.706063
18:35	281.598886	22.528721
18:40	281.916728	21.452120
18:45	282.259937	20.300941
18:50	282.613167	19.128683
18:55	282.952509	18.015346
19:00	283.314048	16.843226
19:05	283.653384	15.756603
19:10	284.014396	14.615337
19:15	284.386926	13.453925
19:20	284.764081	12.295323
19:25	285.113651	11.237298
19:30	285.500152	10.085554
19:35	285.880698	8.9706450
19:40	286.278079	7.8271870
19:45	286.654926	6.7632370
19:50	287.055832	5.6544200
19:55	287.462930	4.5548000
20:00	287.884482	3.4477450
20:05	288.283632	2.4348160
20:10	288.720403	1.3771230
20:15	289.158926	0.3860260
20:20	289.578643	0.0000000