

Universidad Nacional De La Plata

FACULTAD DE INGENIERÍA

POSICIONADOR DE ANTENA

Autor:Gastón Valdez

Directores: Martín Salibe, Elias Fliger

Índice

Capítulos	Página
I Definición del proyecto	1
1. Posicionador para antena	2
1.1. Introducción	2
1.2. Instituto Argentino de Radioastronomía	2
1.3. Descripción de la Antena y posicionador	2
1.4. Metodología de trabajo	3
1.5. Requerimientos del sistema	4
2. Componentes para la construcción de posicionador	5
2.1. introducción	5
2.2. Componentes del Proyecto	5
3. Selección de hardware para implementación del sistema	7
3.1. Introducción	7
3.2. Componentes de hardware	7
3.2.1. Interfaz de red - Materiales disponibles	8
3.2.2. Interfaz de usuario	8
3.2.3. Microcontroladores disponibles	8
II Desarrollo de software y redes TCP/IP, Interfaz PC- usuario	12
4. Redes	13
4.1. Introducción	13
4.2. Redes de Area local	13
4.3. Modelo de capas	14
4.3.1. Modelo OSI	14
4.4. Modelo TCP/IP	16
4.4.1. capa 1 - Capa de Host a red	17
4.4.2. capa 2 - Interred	17
4.4.3. capa 3 - Transporte	17
4.4.4. capa 4 - Aplicación	17
4.5. OSI vs TCP/IP	17
4.6. Protocolo IP	18
4.6.1. Direcciones IP	19
4.7. Protocolo DHCP	20
4.8. Protocolo TCP	21
4.8.1. Servicio TCP	21
4.9. Sniffer de red -WireShark	22

5. Interfaz De usuario	23
5.1. introducción	23
5.2. Interfaz de Usuario	23
5.2.1. Orbitron	23
5.2.2. Stellarium	24
5.2.3. Celestia	24
5.2.4. Pass	24
5.2.5. Gpredict	25
5.3. Comparativa de interfaces y selección del software	25
5.4. Software Gpredict	26
5.4.1. Libreria Hamlib	27
5.4.2. Selección de satelites y creación del perfil	28
5.4.3. Configuración del rotador en Gpredict	28
5.5. Software Stellarium	32
5.5.1. Configuración de la red en Stellarium	32
5.5.2. Protocolo de comunicacion Stellarium	34
6. Software del Microcontrolador	36
6.1. Introducción	36
6.2. Diagrama del sistema	36
6.3. Esquema eléctrico de los componentes	36
6.4. Diagrama general del software	41
6.5. Función de autocalibración	43
6.5.1. resultados de la función de autocalibración	49
6.6. Control de la posición	50
6.6.1. Resultados de la función de autocalibración y control	52
6.7. Programación de Scheduler o planificación	54
6.7.1. Funcionamiento base de tiempo para scheduler	54
6.7.2. Clear Timer on Compare Match (CTC) Mode	55
6.7.3. Programación del Software scheduler	57
6.7.4. Resultados del software de planificación	59
6.8. Conexión del software con Gpredict y Stellarium	59
6.8.1. Conexión a la red mediante DHCP	60
6.8.2. Configuración de Gpredict - Programación de la comunicación	61
6.8.3. Programación del microcontrolador	63
6.8.4. Stellarium	64
6.8.5. Programación para conectarse con Stellarium	66
6.8.6. Resultados de la conexión con Gpredict y Stellarium	68
6.9. Unión de partes del software	70
6.9.1. Resultados	72
6.9.2. análisis del código realizado	72
6.10. Conclusiones	73
anexo capítulo	74

ÍNDICE	III
A. Códigos de programacion	76
A.1. Códigos de programación C/C++	76
 III Física de los satelites y sistemas de coordenadas astronómicas	77
7. Dinámica de los satelites elementos keplerianos	78
7.1. Introducción	78
7.2. Conceptos básicos	78
7.2.1. cinemática	78
7.3. Leyes de kepler	78
7.3.1. Problema de los dos cuerpos	78
7.3.2. Problema de n cuerpos	78
7.4. Parámetros orbitales o keplerianos	78
7.5. Determinación de las orbitas	78
7.6. Involucrando el tiempo en las ecuaciones	78
7.7. Calculos de orbitas y modelos matematicos	78
7.8. estaciones terrenas	78
 8. Sistemas de coordenadas astronómicas	79
 IV Sistema electrónico de posicionamiento - Construcción de prototipo y resultados	80
 Apéndice	81
A. Protocolos de comunicación Serie	82
A.1. Introducción	82
A.2. Protocolo SPI	82
A.3. Procotolo I2C	84
A.4. Chip ETHERNET W5100 shield	85
A.5. Display LCD con adaptador para I2C	86
 B. Programación sobre plataforma Arduino UNO - Periféricos del microcontrolador	88
B.1. PlatformIO - Configuración y flags de compilación	88
B.2. Primer Programa	88
B.3. Compilación condicional	88
B.4. Sentencias sobre el entorno arduino	88
B.5. Conversor Analógico Digital	88

Fase I

Definición del proyecto

Posicionador para antena

Resumen

Se definen los requerimientos del sistema y las necesidades del radiotelescopio. Además, se muestra una planificación del trabajo a lo largo de este texto.

1.1 Introducción

En el marco de la catedra Proyecto Final de la carrera de ingeniería electrónica, de la fac. de ingeniería, perteneciente a la Universidad Nacional De La Plata, se realiza un sistema electrónico para el posicionamiento de una antena, cuyo lugar de realización es el Instituto Argentino de Radioastronomía(IAR), en la modalidad con director. Este instituto, se dedica a la radioastronomía, que es la observación del cielo mediante ondas de radio. Dicha institución, quiere realizar la bajada de datos satelitales, medir la potencia total, vender el servicio a terceros, velar por el cumplimiento de normas internacionales, etc. Utilizando un receptor de comunicaciones adosado a una antena parabólica en desuso, se obtienen estos datos.

La antena, posee un radio de 2 metros aproximadamente, la misma tiene un sistema mecánico, que mueve la antena mediante dos motores, en dos ejes independientes entre si. En el presente texto, se aprovechan estos dos motores, y se realiza un sistema electrónico de posicionamiento automático para esta antena. Por lo expuesto en el párrafo anterior, el sistema, para realizar la bajada de datos satelitales, debe realizar el seguimiento de satélites que se encuentren dentro de los ángulos de visibilidad de la antena.

1.2 Instituto Argentino de Radioastronomía

En el Instituto Argentino de Radioastronomía(IAR), posee dos antenas parabólicas (ver figura 1.1, donde una de ellas se ve de fondo), de radio 30 mts aproximadamente, las cuales son utilizadas para observaciones astronómicas. La emisión de potencia de los satélites, puede interferir en las observaciones astronómicas, y podrían realizarse filtros adaptativos, para estos receptores. Otra posible aplicación es la venta de estos datos a privados, verificar el cumplimiento de normas de potencia emitida (esto brinda poder de policía a la institución) por satélites, y un sin fin de aplicaciones.

Para realizar esta medida de potencia, el IAR, requiere realizar el seguimiento de los satélites que se encuentren dentro de la visibilidad que posea la antena, para poder medir esta potencia total y realizar un cálculo de la potencia emitida por los mismos. A continuación se muestra la antena sobre la que se realiza el trabajo, y de fondo, una de las antenas principales de la institución.

1.3 Descripción de la Antena y posicionador

La antena tiene el sistema mecánico que se observa en la figura 1.2.

En ella, se observa, que existen dos motores, uno para cada eje, además, tiene un sistema que permite medir la posición de la antena mediante dos potenciómetros adosados al eje de cada motor. Estos ejes son independientes entre si, y su medida también.

En el presente trabajo, se va a desarrollar un sistema que sea capaz de realizar el movimiento de la antena, aprovechando el sistema de motores existente sobre la misma. El sistema, que realiza el movimiento de la antena, se conoce como "posicionador". Este sistema, recibe una posición, en



Figura 1.1: antena ubicada en el iar, actualmente en desuso



(a) Motor de primer eje de la antena



(b) Motor de segundo eje de la antena

Figura 1.2: antena ubicada en el iar, actualmente en desuso

dos coordenadas, y tiene que mover la antena hacia esas coordenadas recibidas. Estas coordenadas recibidas, son las posiciones de los satélites, los cuales se van moviendo, por ende, mientras el satélite este por encima de la antena, o su "horizonte visible", debe realizar el seguimiento de la misma, actuando sobre los motores, y acomodando la antena, a donde este el satélite en cuestión.

En la actualidad, existen diversos software para realizar el seguimiento de satélites en tiempo real, estos consultan bases de datos existentes en internet, y realizan el cálculo en base a modelos matemáticos. En este documento, se hará uso de estos software, para poder actualizar la posición a cada instante y mover los motores hacia donde corresponda. Además, este dispositivo, debe ser controlador desde una PC que esté ubicada dentro de la institución.

1.4 Metodología de trabajo

El trabajo, se va a dividir en cuatro etapas, denominadas "fase 1, fase 2, fase 3 y fase 4" respectivamente. En la primera fase, se va a definir los requerimientos del sistema (capítulo actual), luego se va a proponer una solución para cumplir estos requerimientos (cap. 2), y luego se van a seleccionar algunas piezas electrónicas para la construcción de la solución (cap 3.).

En la segunda fase, se va a desarrollar el software que debe realizar el sistema de control, tanto para el usuario, como para la computadora que controla la antena. El orden del trabajo, es primero desarrollar el software sobre la computadora, y luego buscar interfaces disponibles, para conectarnos con ese equipo mediante el uso de internet.

En la tercera fase, se va a realizar una investigación sobre los sistemas de coordenadas, como realizar las transformaciones de estas entre si, y la física de los satélites y modelos matemáticos.

En la cuarta fase, se va a desarrollar el sistema de posicionamiento de los motores, luego se

desarrolla la interfaz para conectarse a la computadora que realiza el control del sistema. Luego, una vez desarrollado estas interfaces, se prueba el sistema realizando algún seguimiento, sea a satélites, o a estrellas.

1.5 Requerimientos del sistema

De lo expuesto en las secciones anteriores, podemos obtener los requerimientos para este proyecto. Los requerimientos para este proyecto se dividen en dos tipos:

1. Requerimientos Funcionales.
2. Requerimientos de sistema.

Donde en el primero, se definen aquellas cosas relacionadas al comportamiento del dispositivo a diseñar, mientras el segundo, se refiere a como llevar a cabo la solución en si.

Requerimientos	Requisitos funcionales	Medir posición de la antena.
		Recibir coordenadas desde una PC dentro del IAR.
		Tener control sobre la posición de la antena.
		Estacionar la antena en la posición del Zenith cuando no realice seguimientos sobre satélites.
		Seguimiento de satélites, naturales y artificiales, y estrellas.
	Requisitos De Sistema	No puede utilizar ningún tipo de red inalámbrica.
		Existencia de componentes en el mercado local.
		Escalable.
		Independencia entre los componentes del sistema.
		Calibración automática de sentido de movimiento y posiciones angulares.
		Manejo desde una PC dentro de la institución.
		Bajo Costo.

Tabla 1.1: Requerimientos del sistema

Componentes para la construcción de posicionador

resumen

En este capítulo se seleccionan cuales son los componentes necesarios para satisfacer los requerimientos de la sección 1.5.

2.1 introducción

En este capítulo, se van a mostrar cuales son los componentes, que debe tener el sistema en su versión final. Estos componentes se basan en los requerimientos definidos en la sección 1.5. En el mismo, se ha dividido en dos tipos de requerimientos: Funcionales y de sistema.

2.2 Componentes del Proyecto

La interconexión de las partes dentro de la institución debe responder al siguiente diagrama del sistema general:

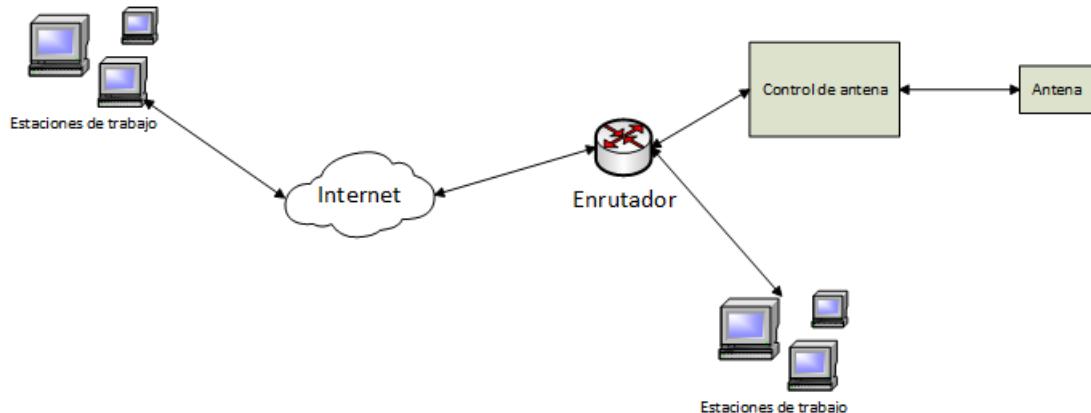


Figura 2.1: Diagrama del sistema general

Donde el bloque desarrollado en la presente tesis es el control de antena de la figura 2.1, y la configuración de software sobre las estaciones de trabajo dentro de la institución.

Por lo expuesto en la sección 1.5(requisitos de sistema), el sistema que realiza el control de posición sobre la antena debe tener los siguientes componentes:

1. Microcontrolador o computadora para realizar el movimiento de la antena, además debe responder a una PC y realizar el movimiento de la antena.
2. Interfaz de usuario con el estado del sistema(interfaz electrónica o pantalla para mostrar el estado del sistema).
3. Interfaz de red, cableada para poder recibir ordenes desde una PC dentro de la institución.
4. Mediciones angulares en ambos ejes de la antena.
5. Sistemas electrónicos para manejo de motores.

6. Software PC de seguimiento de satélites que cuente con conectividad a la red.

Estos ítems, cumplen todos los requerimientos, tanto funcionales como de sistema. La escalabilidad y la independencia se realiza mediante el microcontrolador, ya que, se puede actualizar el software sobre el mismo, logrando que el mismo sea escalable. La autocalibración, control, y el seguimiento de satélites, naturales o artificiales, y estrellas, se consigue con la combinación del microcontrolador con los sistemas electrónicos de manejo de motores. La recepción desde una PC de los datos, se realiza mediante la interfaz de red. Ambos requerimientos(funcionales y de sistema), deben realizarse mediante la programación del microcontrolador, y la construcción del sistema electrónico de manejo de los motores. Cabe destacar, que se deben realizar dos sistemas electrónicos para el manejo de los motores, ya que el mismo, tiene dos motores, independientes entre si, el cual genera movimientos independientes de la antena. El requisito de bajo costo y disponibilidad en el mercado local, se analiza en el siguiente capítulo.

Selección de hardware para implementación del sistema

resumen

Aquí, definimos algunos de los componentes de hardware seleccionados para cumplir con los requerimientos, en particular, definimos el microcontrolador, la interfaz de red, y la interfaz de usuario del sistema. En esta sección, el análisis se basa sobre componentes disponibles en el IAR, de estos componentes, seleccionamos aquellos, que cumplen el requisito de bajo costo, y luego sobre los mismos, analizamos su facilidad de programación, librerías disponibles, módulos disponibles sobre estos, y otros aspectos.

3.1 Introducción

Se definen el microcontrolador seleccionado, los criterios de selección del mismo, además, se define la interfaz de red, y la interfaz de usuario. Los componentes se han seleccionado en base a los siguientes criterios:

1. Compatibilidad entre las partes.
2. Disponibilidad de documentación para el desarrollo.
3. Cantidad de puertos entrada/salida de cada microcontrolador o computadora

Al ser una antena, que debe seguir satélites, se debe tener en cuenta, que los satélites, varían su velocidad, según el punto de la órbita en que se encuentren. Dado que esta velocidad varía, este seguimiento, debe adaptarse a estos cambios. Estos cambios son del orden de los segundos, y cualquier microprocesador actual funcionan en orden de los megahertz, con lo cual, si se realiza el control en tiempos del orden de milisegundos, el control podría realizarse sin ningún tipo de inconveniente. Por este motivo, la velocidad de la computadora, no es un factor crítico a tener en cuenta en los puntos de vista para la selección del hardware.

3.2 Componentes de hardware

La computadora, o microcontrolador, debe tener interfaces, para conectarse con el mundo exterior. El hecho es que requiere realizar la medición de dos posiciones en simultáneo, que son la declinación y la altura. La antena, posee adosado, dos potenciómetros, que cumplen la función de encoders. Por ende, al tener adosado estos dos potenciómetros, el microcontrolador, debe tener como mínimo dos canales o puertos de entrada que posean un conversor analógico-digital cada uno.

La interfaz de red, debe ser independiente del microprocesador o controlador, para cumplir con el requerimiento de escalabilidad, por ende, se debe utilizar alguna solución integrada que se conecte al microcontrolador principal, y puedan intercambiar mensajes entre ellos.

La interfaz para el estado de la antena, se usará una pantalla, la cual mostrará el estado de la misma. Para ello, se usará un display LCD de 16x2 que se encuentra disponible para su uso.

De lo expuesto en los párrafos anteriores, el microcontrolador, debe ser independiente de todo el hardware asociado. Además debe poseer una electrónica asociada a cada motor, que permita

encender o apagar cada motor de forma independiente. Además, debe ser capaz de controlar el sentido de giro de cada motor. Dado que estas maniobras las debe realizar el microcontrolador, se requieren de al menos cuatro puertos disponibles sobre el microcontrolador.

3.2.1. Interfaz de red - Materiales disponibles

La placa disponible en el IAR, es la siguiente: chipEthernetW5100. Las misma se muestra a continuación.

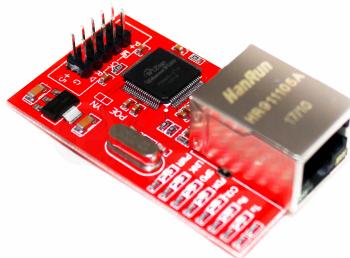


Figura 3.1: Chip Ethernet W5100

La placa ethernet W5100(ver figura 3.1) es un controlador dedicado a las redes. Al recibir un dato, este lo transmite, mediante un puerto paralelo, o por puerto SPI(serial paralell interface). Cabe destacar, que este desarrollo, solo tiene disponible el bus SPI, ya que la placa, solo viene con este protocolo. Además, esta placa, no puede compartir el bus, debido a el diseño de la misma. La misma debe modificarse, para poder compartir el Bus SPI. Uno de los requerimientos es que no debe utilizar redes inalámbricas, por este motivo, se utiliza esta interfaz de red.

3.2.2. Interfaz de usuario

Como se explico en la sección 3.2, se va a utilizar un display LCD de 16 columnas y 2 filas,que esta disponible para su uso en el Instituto Argentino de Radioastronomía.

El display LCD, es una pantalla, la cual es capaz de mostrar texto, valores numéricos,crear símbolos propios,etc.El display LCD disponible, se muestra en la figura 3.2:

La imagén, se ve que el display LCD, tiene adosado, un controlador, este controlador es un circuito integrado denominado PCF8574. Este dispositivo, es capaz, de expandir la cantidad de pines de un microcontrolador, utilizando el protocolo I2C. La ventaja de esto, es que a partir de dos pines se pueden controlar 8 puertos, y esto ahorra en cantidad de puertos de entrada y salida a la hora de elegir el microprocesador. Por ende, el microcontrolador seleccionado debe poseer un controlador o interfaz I2C

3.2.3. Microcontroladores disponibles

Los microcontroladores disponibles dentro de la institución, están embebidos dentro de placas de desarrollo, ya diseñadas, y comerciales. Por lo expuesto en las secciones anteriores(ver secciones 3.2.1,3.2.2),el microcontrolador seleccionado debe tener las siguientes características:

1. Cantidad de conversores analógico/digital: 2

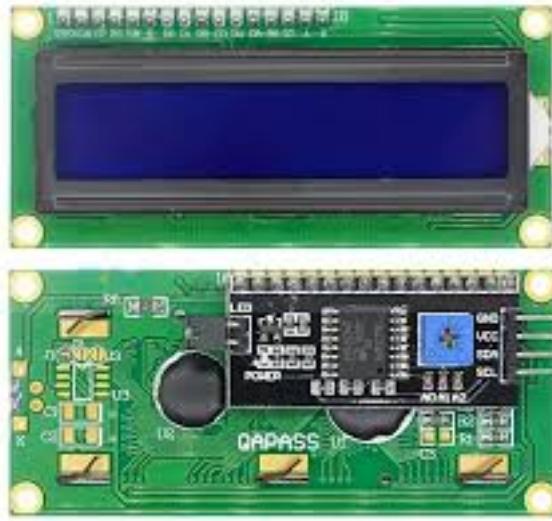


Figura 3.2: Display LCD en el IAR

2. Cantidad de pines disponibles: 4 (mínimo)
3. Puerto SPI: conexionado del chip ethernet
4. Puerto I2C: para la conexión del display

De los placas de desarrollo que hay actualmente en el IAR, se tienen las siguiente placas a analizar:

1. EDU-CIAA
2. Arduino Uno
3. STM32VL discovery

Donde, de las tres, debemos elegir aquella que tenga un menor valor económico, estudiar las soluciones de software y documentación que posean, y revisar si tienen puertos SPI e I2C para el display usado como interfaz de usuario y el chip Ethernet W5100.

3.2.3.1. EDU-CIAA



Figura 3.3: Eduu Ciaa disponible en el IAR

La placa EDU-CIAA, es una computadora de software abierto, desarrollada en argentina. Su nucleo se basa en un microprocesador cortex ARM 4. Su microcontrolador es el LPC4337. La documentación existente, está incompleta. Algunas partes, están desarrolladas y otras partes, aún están en desarrollo. Posee todas las interfaces necesarias para interactuar con los demás módulos (SPI, I2C, conversores A/D y pines disponibles), se puede realizar una depuración sobre la misma placa, sobre un protocolo denominado JTAG.

Posee un IDE, y los archivos de compilación (makefile) para cargar el código sobre esta placa. La placa se muestra

en la figura 3.3. La EDU-CIAA esta pensada en software y hardware libre. Todos los esquemáticos de circuitos, y los componentes de software, existen disponibilidad para descargarse y modificarse libremente.

Esta placa de desarrollo, esta pensada para aprender programación sobre microcontroladores, por ende, tiene todas sus interfaces integradas sobre la placa. Posee librerías disponibles para el uso de SPI e I2C. Estas librerías, no poseen documentación, por lo tanto, deben programarse a nivel de registros esta configuración, o realizar una ingeniería inversa sobre las librerías para describir su funcionamiento.

3.2.3.2. Arduino UNO

Figura 3.4: Placa Arduino Uno en el IAR



es que poseen librerías para el manejo de los puertos SPI e I2C. En particular, existen librerías para el manejo del display mediante el uso de I2C y el chip ethernet W5100 de manera nativa, es decir, sin necesidad de instalar o descargar ninguna librería adicional. Posee disponibilidad de pines, y puertos PWM. También posee conversores analógicos digitales. La documentación disponible, se encuentra en su sitio oficial, de manera ordenada. La información no oficial, es decir, en internet es abundante, y existen varios millones de proyectos basados en el ecosistema arduino dentro de uno de los repositorios más grandes de software: GitHub.

3.2.3.3. STM32VL discovery



Figura 3.5: Placa STM32VLDISCOVERY disponible en el IAR

se encuentra en idioma inglés. La placa se muestra en la figura 3.5

La placa de desarrollo, arduino UNO(ver figura 3.4) es una placa de desarrollo basada en hardware y software libre. Esta pensada para desarrollos y prototipos rápidos, y posee un entorno de desarrollo integrado, el cual viene preparado para cargar el software dentro de él. Su lenguaje de programación es C/C++. Todos los esquemáticos, y software estan disponibles en internet. Se basa en el microcontrolador AtMega328P.

Un punto a favor de esta placa de desarrollo,

esta placa de desarrollo, es desarrollada por la empresa STMicroelectronics. Según la información oficial de su página, esta placa esta pensada el aprendizaje del microcontrolador STM32F100. Este microcontrolador es el nucleo de esta placa, es un procesador con arquitectura ARM. Tiene 64 pines disponibles, varios conversores analógicos digitales. Tiene ejemplos y disponibilidad de software en forma gratuita. No posee documentacion sobre la construcción de la placa. Existe muy poca informacion por fuera de la documentación oficial. Existen algunos foros de sistemas embedidos, pero la discusión es muy escasa, y en general

3.2.3.4. Comparación de microcontroladores

Dado, que se han analizado las tres placas de desarrollo que están disponibles, se van a comparar sus prestaciones y además su disponibilidad en el mercado local. Esto se realiza, basado en las hojas de datos de cada microcontrolador. Este es necesario, ya que si el equipo se daña durante el desarrollo, o mediante su uso, puede cambiarse de forma inmediata, sin pérdida de tiempo. Esta comparación se realiza en la siguiente tabla:

Nombre de la placa	Arduino Uno	EDU-CIAA	STM32VLDISCOVERY
Microcontrolador	ATmega328P	ARM Cortex-M4F	STM32F100
Canales ADC	6 canales -10 bit	3 canales -10 bit	16 canales - 12bit
Puertos Disponibles	15	80	70
Comunicación I2C	Si	Si	Si
Comunicación SPI	Si	Si	Si
Disponibilidad	Si	No	No
Precios promedios	\$1000(disponible en todo el país)	\$5.732(único local)	\$5100(mercado limitado)

Tabla 3.1: Cuadro comparativo de placas disponibles en el Instituto Argentino de radioastronomía

De la tabla 3.1, la placa seleccionada es la denominada Arduino UNO, ya que posee soluciones de software sobre el display, y soporta el chip ethernet W5100 de manera nativa. Esto, implica que la velocidad de desarrollo es superior a la de las otras placas, ya que no se requieren conocimientos detallados sobre los protocolos.

Por último, el hardware seleccionado para realizar la segunda fase de esta tesis es el siguiente:

1. Placa de desarrollo o microcontrolador: Arduino UNO
2. Interfaz de red: Chip Ethernet W5100
3. Interfaz de usuario: Display LCD

Estos tres componentes, ante cualquier tipo de fallo, estan disponibles en el mercado local, y esto facilita el seguimiento del proyecto en caso de algún tipo de fallo en los componentes de hardware.

Fase II

**Desarrollo de software y redes
TCP/IP, Interfaz PC- usuario**

Redes

resumen

Se definen los componenentes basicos de una red: los modelos OSI y TCP/IP. Luego analizamos los protocolos TCP, y DHCP,y el protocolo IP. Luego, se propone el software WireShark para ver estos protocolos funcionando sobre una red ya implementada.

4.1 Introducción

En este capítulo analizamos las redes de computadoras,y como se conectan entre sí los dispositivos de una red. Además, se analizan dos modelos de capas principales: OSI y TCP/IP, donde cada uno consta de varias capas, donde se brinda un breve resumen de cada capa. Luego, se analizan, dos protocolos para reconocer dispositivos en una red, el protocolo DHCP,que le asigna una dirección, y el protocolo IP,que le asigna una red. Además, se introduce el concepto de sniffer, y se muestra como visualizar el contenido de los datos que estan circulando en una red. Los conceptos en que profundiza el presente capítulo, son aquellos de mayor interés para la construcción del dispositivo.

4.2 Redes de Area local

Existen tres tipos de redes: Redes de area local, de area metropolitana, y redes de área amplia. El dispositivo desarrollado en el presente trabajo, se conecta a la red de área local de la institución, por ende, es importante entender de una manera elemental el funcionamiento de las redes, sus protocolos, y sus modelos de capas.

Las redes de área local, son aquellas que interconectan distintos equipos de una institución, departamento de una empresa, etc, para que se realize un intercambio de información, a través de un medio. Las redes de área local, generalmente se dice que son redes LAN, por sus siglas en inglés(Local Area Network). En general, son redes de propiedad privada. Existe un estandar, para estas redes LAN cuando se conectan de forma inalámbrica: IEEE 802.11 (comunmente denominado WiFi), y cuando son alámbricas existe el estandar IEEE 802.7(comunmente denominado ethernet). Las redes WiFi, no se utilizan en este trabajo, por utilizar tecnicas de radiofrecuencia, que interfieren con los receptores de comunicación. En el caso de redes alámbricas, las computadoras, realizan un enlace punto a punto, a traves de un dispositivo denominado **Switch**.Un switch, posee varias entradas para conectar mas de una PC,y el trabajo de este, es transmitir mensajes entre computadoras, conociendo la dirección de destino, que viene incluida en el mensaje. Si se requieren redes más grandes, pueden interconectarse Switchs entre sí, y armar redes de mayor tamaño.

En el presente trabajo, se conecta el dispositivo a un switch, que esta ubicado en sala de control,del Instituto Argentino De Radioastronomía. Además al finalizar el trabajo, se solicitará que se le asigne una dirección fija dentro de la red institucional.

En el trabajo realizado en este presente informe, usamos un modelo conocido como arquitectura “cliente-servidor”. Esta arquitectura, se basa en que hay maquinas clientes, que solicitan un servicio, a otra estación de trabajo, denominada servidor. Los clientes, son las estaciones de trabajo del lugar, y el servidor es el dispositivo desarrollado en esta tesis, ya que es el que brinda el servicio de apuntamiento de antena. En otras palabras, las estaciones de trabajo deben conectarse al dispositivo desarrollado en el presente documento. De ahí, que se requiera el permiso correspondiente para fijar la dirección del dispositivo.

4.3 Modelo de capas

Para reducir la complejidad de las redes, se organizan en capas, donde cada capa, ofrece ciertos servicios a capas superiores, mientras les oculta detalles relacionados con la forma en que se implementan estos servicios. El uso de capas, no es un problema inherente de las redes, sino que se aplica en otros ámbitos de las ciencias de la computación. La cantidad de capas es variable, y depende del problema en cuestión. En términos de redes de computadoras, se usan dos modelos principalmente, el modelo OSI y el modelo TCP/IP. El primero consta de siete capas, el segundo consta de cinco capas.

Para entender este sistema de capas, consideramos una capa y le ponemos un numero K. Supongamos que la capa K de una PC desea comunicarse con la capa K de otra. La forma en que ambas se comunican, se denomina protocolo, que define las reglas y convenciones usadas en esta comunicación.

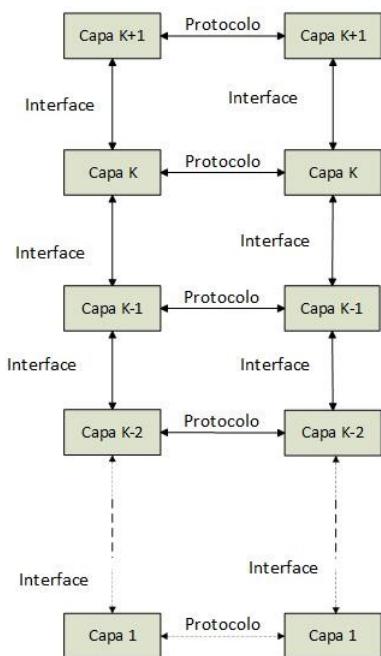


Figura 4.1: Conexion entre capas. En esta figura se observan la interfaz, protocolo y conexionado

El modelo OSI y el modelo TCP/IP. Se diferencian en la cantidad de capas de cada modelo, los nombres de cada capa, pero la funcionalidad sigue siendo la misma: intercambiar información entre dos dispositivos, usando un esquema de capas, las cuales permiten que el diseñador tenga mas facilidad a la hora de diseñar una red.

4.3.1. Modelo OSI

El modelo OSI, se basa en una propuesta desarrollada por la organización internacional de normas(ISO), en un intento de estandarización para los protocolos. En si, es un modelo general, pero sus protocolos no se utilizan masivamente. La sigla OSI proviene de "open system interconnection" (sistema de interconexión abierto). Este modelo tiene siete capas, y los principios utilizados para llegar a ellas son los siguientes:

En realidad, las capas, no se comunican directamente entre sí, sino que pasan la información (datos y control) a la capa inferior (en nuestro ejemplo, capa k-1), y así, hasta el nivel más bajo (capa 1). En el nivel más bajo, envía la información a otra maquina dentro de la red, y ella se encarga de pasar los datos a las capas superiores, en un proceso inverso al realizado para enviar la información. La comunicación entre capas se realiza a través de una entidad denominada interface. La figura 4.1 ilustra el proceso de capas descrito.

A un conjunto de capas y protocolos se conoce como Arquitectura de Red. La especificación de una arquitectura debe contener suficiente información como para permitir que un programador escriba el programa, o construya el hardware para cada capa, de manera que se cumpla correctamente el protocolo apropiado.

Debido a que existen muchas computadoras dentro de una organización, estos protocolos, necesitan de alguna manera identificar el destinatario del mensaje y quien lo envía, esto se denomina direccionamiento o nombramiento en las capas. Para realizar esto, se definen dos modelos:

1. Se debe crear una capa en donde se requiera un nivel diferente de abstracción
2. Cada capa debe realizar una función bien definida
3. La función de cada capa se debe elegir teniendo en cuenta la definición de protocolos estandarizados internacionalmente
4. Es necesario elegir los límites de las capas de modo que se minimice el flujo de información a través de las interfaces
5. La cantidad de capas debe ser suficiente como para no tener que agrupar funciones distintas en la misma capa; además, debe ser lo bastante pequeña como para que la arquitectura no se vuelva inmanejable.

Las capas, se muestran en la figura a continuación:



Figura 4.2: Capas del Modelo OSI

4.3.1.1. capa 1 - Capa física

Esta capa, se relaciona con la transmisión de bits a través de un canal de comunicación. Los aspectos de diseño tienen que ver con las interfaces mecánica, eléctrica y de temporización, así como con el medio de transmisión físico que se encuentra bajo la capa física.

4.3.1.2. capa 2 - Enlace de datos

La función principal es transformar un medio de transmisión puro, en una línea que esté libre de errores. Además, proporciona medios para activar, desactivar y mantener el enlace (es decir, la comunicación). Aquí, está definido un componente de hardware: la tarjeta o placa de red. Cada placa de red tiene un número único conocido como MAC address, y es único para cada dispositivo en el mundo.

4.3.1.3. capa 3 - Capa de red

La capa de red, realiza la transferencia de información entre sistemas finales, liberando a las capas superiores del conocimiento de los medios de transmisión y tecnologías de conmutación usadas.

Si hay demasiados paquetes en la subred al mismo tiempo, se interpondrán en el camino unos con otros y formarán cuellos de botella. El manejo de la congestión también es responsabilidad de la capa de red, en conjunto con las capas superiores que adaptan la carga que colocan en la red.

4.3.1.4. capa 4 - Transporte

En esta capa, se reciben datos de las capas superiores, y los " particiona" en unidades más pequeñas, para ser enviadas a las capa de red. Además, debe asegurarse que estos paquetes lleguen a destino. En otras palabras, parte los mensajes de las capas superiores para ser transmitidas al medio de comunicación.

4.3.1.5. capa 5 - Sesión

Esta capa, proporciona mecanismos para el control del diálogo (decidir quien debe transmitir) entre las aplicaciones de los sistemas finales. En muchos casos, estos servicios son totalmente imprescindibles, sin embargo, en algunas aplicaciones, su utilización es obligatoria.

4.3.1.6. capa 6 - Presentación

Aquí, se define el formato de los datos que van a intercambiarse en las aplicaciones., y ofrece a las aplicaciones un conjunto de servicios de transformación de datos. Ejemplos específicos, son compresión y cifrado de datos.

4.3.1.7. capa 7 - Aplicación

Esta capa, proporciona a los programas de aplicación, un medio para que accedan al entorno OSI. Aquí, se encuentran funciones de administración, y los mecanismos para la implementación de aplicaciones distribuidas. En esta capa, tenemos las aplicaciones de uso general, como la transferencia de ficheros, correo electrónico, navegadores web, etc.

4.4 Modelo TCP/IP

El modelo TCP/IP es usado para conectar computadoras entre sí a través de una red interna dentro de una organización. Este modelo se muestra en la figura 4.3, y se dará una breve explicación de cada capa:

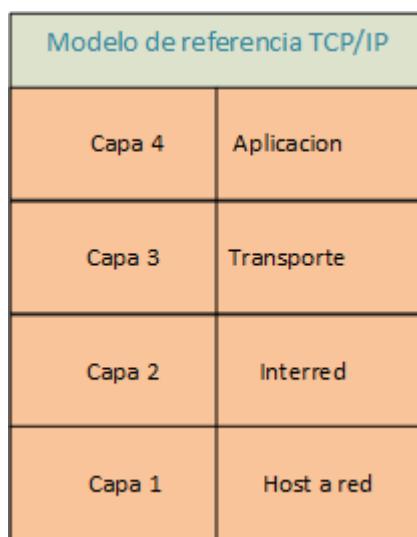


Figura 4.3: Modelo de capas TCP/IP

4.4.1. capa 1 - Capa de Host a red

La capa 1, del modelo TCP/IP, tambien se la suele nombrar capa de enlace. En este documento utilizamos como nombre de capa "host a red". De esta capa, no se puede decir mucho, excepto que puntualiza que el host se debe conectar a la red mediante el mismo protocolo para que se puedan enviar paquetes IP (véase la siguiente sección). Este protocolo no está definido y varia de una red a otra.

4.4.2. capa 2 - Interred

Su trabajo es permitir que los hosts injecten paquetes dentro de cualquier red y que viajen a su destino de manera independiente. Una analogía es el servicio de correo. Una persona, deposita una serie de cartas internacionales en un buzón, y la mayoría de ellas se entregarán en la dirección correcta del país de destino. Es probable que las cartas, viajen a través de una o más puertas de enlace de correo internacional, pero esto es transparente a los usuarios. Este trabajo, lo realiza la capa de interred, que da transparencia a los usuarios de como viajan estos paquetes dentro de una red. La capa de interred, define un paquete de formato y protocolo denominado protocolo IP (internet protocolo). El trabajo de la capa de interred es entregar paquetes IP al destinatario.

4.4.3. capa 3 - Transporte

Está diseñada para permitir que las entidades iguales en los hosts origen y destino puedan llevar a cabo una conversación. Aquí se han definido dos protocolos de transporte de extremo a extremo: el TCP (protocolo de control de transmisión), es un protocolo confiable, que permite un flujo de bytes que se origina en una máquina se entregue sin errores en cualquier otra máquina. Divide el flujo de bytes entrantes en mensajes discretos y pasa cada uno de ellos a la capa de interred. En el destino, ocurre el proceso inverso, es decir, el destino reconstruye el mensaje recibido. TCP maneja control de flujo para asegurarse que un emisor rápido no sature a un receptor lento con más mensajes de los que puede manejar. El segundo protocolo de esta capa UDP (protocolo de datagrama de usuario) es un protocolo no confiable, se usa para aplicaciones que no desean la secuenciación o control de flujo de TCP y desean proporcionar el suyo. Es decir, se usa en aplicaciones donde la entrega puntual es más importante que la precisa.

4.4.4. capa 4 - Aplicación

Contiene todos los protocolos de nivel más alto. Los primeros fueron TELNET, FTP, SMTP, etc. Con el tiempo, se han agregado otros protocolos, por ejemplo, HTTP, DNS, etc. En la siguiente imagen, se muestra la relación entre las capas del modelo TCP/IP

4.5 OSI vs TCP/IP

Los dos modelos tienen mucho en común. Ambos se basan en un modelo de capas, independientes entre sí. Es importante tener en cuenta que se están comparando los modelos de referencia, no los protocolos de comunicación entre si. En la siguiente figura se muestra la comparación de las capas:

El modelo OSI, define tres partes básicas:

1. Servicios
2. Interfaces

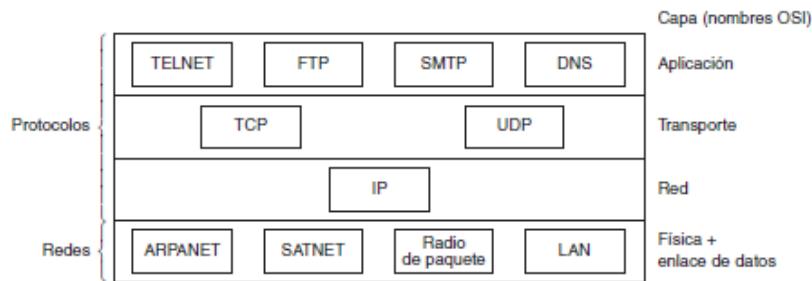


Figura 1-22. Protocolos y redes en el modelo TCP/IP inicialmente.

Figura 4.4: Esquema de interconexión del modelo TCP/IP

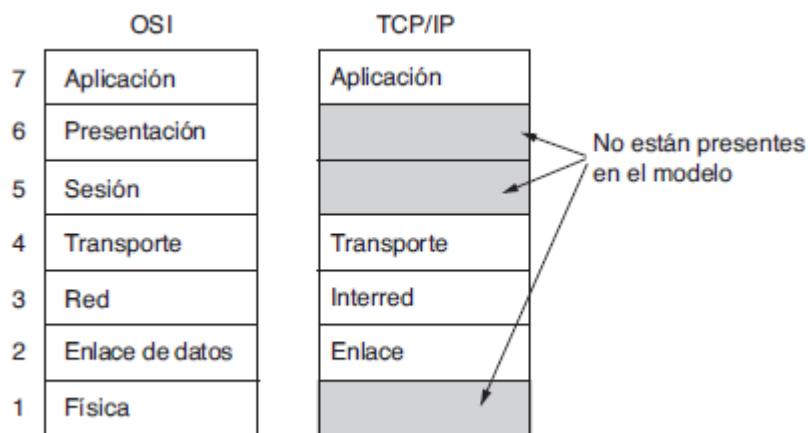


Figura 4.5: Comparativa entre modelo OSI y TCP/IP

3. Protocolos

El modelo OSI, define bien estas tres partes, y las distingue de una manera clara, cosa que no realiza el modelo TCP/IP. Sin embargo, los protocolos del modelo OSI, son difíciles de implementar, por esto se opta por los protocolos que se definen en el modelo TCP/IP.

Una diferencia importante, se observan en la figura 4.5, en el que se observa que el modelo TCP/IP tiene cuatro capas, mientras el otro tiene siete. Una ventaja, en el modelo OSI, es que el protocolo de una capa, puede cambiarse con facilidad, sin afectar las demás capas, esto no es así en el modelo TCP/IP.

Otra diferencia está en área de comunicación orientada a la conexión y no orientada a la conexión. El modelo OSI soporta ambas comunicaciones en la capa de transporte. En el modelo TCP/IP solo tiene un modo en la capa de red (no orientado a la conexión), pero soporta ambos modos en la capa de transporte, lo que da a los usuarios la oportunidad de elegir, en casos de protocolos sencillos de solicitud-respuesta.

4.6 Protocolo IP

La capa de red, se encarga de llevar todos los paquetes a destino. Podría ocurrir, que existan varios enruteadores en el medio. Esto contrasta con la capa de enlace de datos, que asegura la fiabilidad de un extremo a otro de un cable. El protocolo IP, se encuentra en el estandar RFC 1958, y es de acceso público. Este protocolo se denomina IP por las siglas "Internet Protocol".

La comunicación en internet funciona de la siguiente manera: la capa de transporte, fragmenta los datos de la capa superior, y se los transmite a la capa de red. Los enruteadores, reenvian cada paquete, a través de mas de un enruteador, hasta llegar a destino. En el destino, la capa de red entrega los datos a la capa de transporte y se los envia a la capa de aplicación. Cuando todas las piezas llegan finalmente a la máquina de destino, la capa de red las vuelve a ensamblar para formar el datagrama original. Un datagrama es la información que viaja en una red, y esta compuesta por números binarios. Si bien, esta definición es bastante burda, es suficiente para el presente trabajo. El datagrama IP, se muestra en la figura siguiente

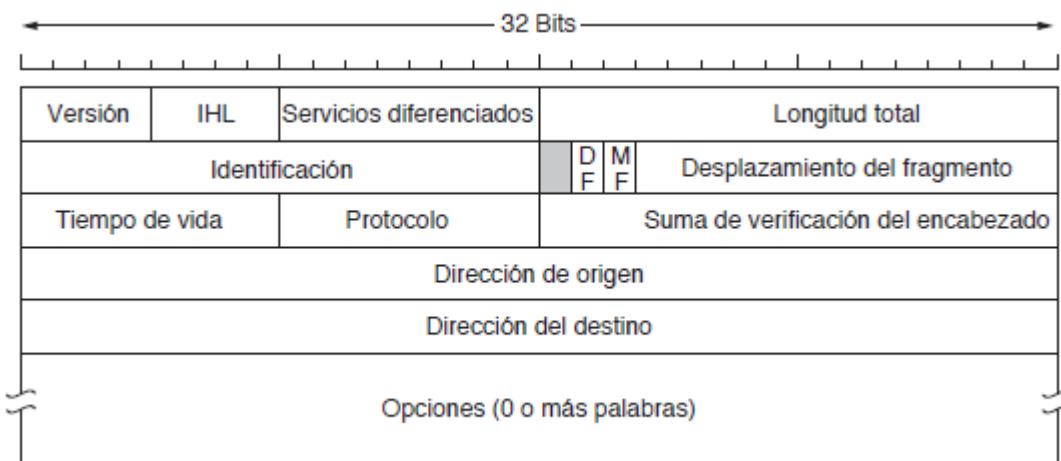


Figura 4.6: Datagrama IP

Este datagrama, se compone de dos partes: encabezado y carga útil. El encabezado, se compone de 20 bytes fija y una parte opcional de longitud variable. Los bits se transmiten de izquierda a derecha, y de arriba hacia abajo en la figura 4.6. Todos los parámetros, estan definidos en el estandar mencionado al principio de esta sección, en este trabajo, solo nos concentraremos en dos parámetros: Dirección origen y Dirección destino. Estas son las que debemos utilizar para configurar el software, y que las estaciones de trabajo puedan conectarse a nuestro dispositivo. Estas direcciones, se conocen como direcciones IP.

4.6.1. Direcciones IP

Si se observa, la figura 4.6, se observa que hay 32 bits(o 4 bytes) para indicar estas direcciones. Estas direcciones están compuestas de una porción de longitud variable para indicar la red en los bits superiores, y de una porción de host en los bits inferiores. La porción de red tiene el mismo valor para todos los hosts en una sola red, como una LAN Ethernet. Esto significa que una red corresponde a un bloque contiguo de espacio de direcciones IP. A este bloque se le llama prefijo.

Las direcciones IP se escriben en notación decimal con puntos. En este formato, cada uno de los 4 bytes se escribe en decimal, de 0 a 255. Por ejemplo, la dirección hexadecimal 80D00297 de 32 bits se escribe como 128.208.2.151. Para escribir los prefijos, se proporciona la ip menor en el bloque y tamaño del mismo. El tamaño se determina mediante el número de bits en la porción de red, los restantes bits(bits de host) pueden variar. Esto significa, que el prefijo debe ser potencia de dos. Por convención, el prefijo se escribe después de la ip, con los bits dedicados a los host. Por ejemplo, la siguiente notación 128.208.0.0/24, significa que hay 24 bits para la red, y 8 bits para el host.

Como el prefijo que indica la red, puede ser de longitud variable, entonces se debe realizar una operación and binario con un numero de unos en la parte de red, y ceros restantes. Este número con el que se realiza esta operación, se denomina **Máscara de subred**. En el ejemplo anterior: 128.208.0.0/24, la máscara de subred es en decimal: 255.255.255.0. Si realizamos la operación and bit a bit entre 255.255.255.0 y 128.208.0.0, obtenemos 128.208.0.0, lo cual indica que pertenece a la red 128.208.0.0. En este trabajo, solo requerimos de estos conceptos, para configurar el software de cada estación de trabajo, y realizar la conexión con el dispositivo desarrollado en el presente trabajo.

4.7 Protocolo DHCP

De la sección anterior, se obtiene que cada computadora, estación de trabajo, o dispositivo que se conecte a la red, debe al menos configurarse la dirección IP, y la máscara de subred, para que puedan intercambiar mensajes entre ellos, o utilizar cualquier servicio disponible en la red. Esto, puede resultar un proceso tedioso, por esto, se define en el estandar RFC 2131 un protocolo de asignación dinámicas de ips, conocido como DHCP(Protocolo de Configuración Dinámica de Host, del inglés Dynamic Host Configuration Protocol). En el estandar, están definidas la forma de comunicación para obtener estos datos, que es básicamente la dirección IP asignada de manera automática. Dentro de cada red, debe existir un servidor DHCP. Para iniciar la solicitud, la máquina, envía a través de la red, una solicitud, denominada DHCPDISCOVER. Si el servidor esta disponible, envía la dirección en un paquete denominado DHCPOFFER. Un paquete DHCP esta conformado por los bits que se muestran en la figura 4.7:

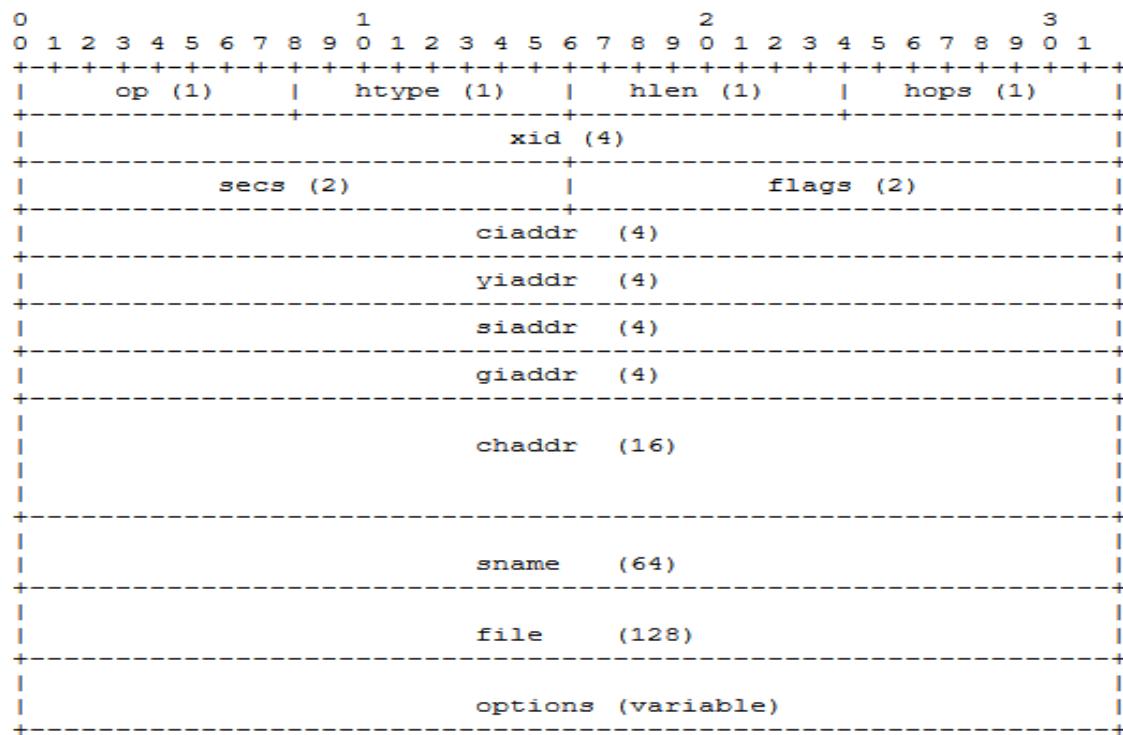


Figura 4.7: Datagrama para intercambio de mensajes del protocolo DHCP

Estos parámetros, son bits, los cuales deben enviarse a través de la red. Estos campos de bits, no serán explicados, ya que están implementados dentro de la librería W5100 que maneja el entorno de arduino UNO. Estos bits están descritos en el estandar indicado en el primer párrafo de la esta

sección, y son de acceso público.

4.8 Protocolo TCP

Este protocolo, pertenece a la capa de transporte. Se diseño específicamente para proporcionar un flujo de bytes confiable de extremo a extremo a través de una interred no confiable. Una interred difiere de una sola red debido a que sus diversas partes podrían tener diferentes topologías, anchos de banda, retardos, tamaños de paquete y otros parámetros. TCP se diseñó para adaptarse de manera dinámica a las propiedades de la interred y sobreponerse a muchos tipos de fallas.

La capa IP no ofrece ninguna garantía de que los datagramas se entregará de manera apropiada, ni tampoco una indicación sobre qué tan rápido se pueden enviar los datagramas. Corresponde a TCP enviar los datagramas con la suficiente rapidez como para hacer uso de la capacidad sin provocar una congestión; también le corresponde terminar los temporizadores y retransmitir los datagramas que no se entreguen.

4.8.1. Servicio TCP

El servicio TCP se obtiene al hacer que tanto el servidor como el receptor creen puntos terminales, llamados sockets. Cada socket tiene un número (dirección) que consiste en la dirección IP del host y un número de 16 bits que es local para ese host, llamado puerto. Un puerto, es definido por la capa de aplicación, que es el punto de acceso a la capa de aplicación. Para obtener el servicio TCP, hay que establecer de manera explícita una conexión entre un socket en una máquina y un socket en otra máquina. Existen puertos que se denominan "conocidos", y son casi un estandar, por ejemplo el puerto 80 para el protocolo HTTP. Los puertos definidos van de 0 a 1024. Los puertos de 1024. Los puertos desde el 1024 hasta el 49151 son puertos que se pueden registrar en IANA, pero las aplicaciones son las que seleccionan sus propios puertos. Todas las conexiones TCP son full dúplex y de punto a punto. Full dúplex significa que el tráfico puede ir en ambas direcciones al mismo tiempo. Punto a punto significa que cada conexión tiene exactamente dos puntos terminales. TCP no soporta la multidifusión ni la difusión. La entidad TCP emisora y receptora intercambian datos en forma de segmentos. Un segmento TCP consiste en un encabezado fijo de 20 bytes (más una parte opcional), seguido de cero o más bytes de datos. El software de TCP decide qué tan grandes deben ser los segmentos. Puede acumular datos de varias escrituras para formar un segmento, o dividir los datos de una escritura en varios segmentos. Una cabecera está formada de la siguiente manera:

Bits	0-3	4-7	8-15	16-31
0	Puerto Origen			Puerto Destino
32	Número de Secuencia			
64	Número de Acuse de Recibo (ACK)			
96	Longitud Cabecera TCP	Reservado	Flags	Ventana
128	Suma de Verificación (Checksum)			Puntero Urgente
160	Opciones + Relleno (Opcional)			
224	DATOS			

Figura 4.8: Cabecera del protocolo TCP

En el encabezado TCP, se tiene puerto origen, puerto destino. Observando, el datagrama de la figura 4.6, tenemos dos IPs: IP origen, e IP destino, y el protocolo TCP, tiene dos puertos(origen y destino), y el mismo protocolo. Este identificador de conexión se denomina 5-tupla, ya que cinco

elementos son los que participan. Por último, cabe destacar, que los puertos y los datos, los define el programador o software, en la capa de aplicación. El programador al usar el protocolo TCP, no debe realizar una verificación de los datos, ya que el protocolo mismo, se encarga de esto.

Estos elementos, se definen, ya que son necesarios para poder avanzar en los conceptos en el presente trabajo, ya que en un futuro, se deben configurar los software de los usuarios, y la programación del microcontrolador seleccionado en el capítulo anterior, basados en estos conceptos para poder realizar una integración exitosa, entre el microcontrolador y las computadoras del lugar.

4.9 Sniffer de red -WireShark

Un programa que pueda revisar los paquetes de una red que llegan a una computadora, y los pueda visualizar, se denomina sniffer. Un sniffer de red, no es más que un software, en el cual se pueden visualizar los paquetes recibidos, en la computadora que esta instalado. El software elegido para realizar esta visualización de los paquetes se llama WireShark. Se ha elegido este software, dado que es software libre, y pueden filtrarse los paquetes por puertos, y por IP. Además tiene muchas otras opciones que no serán contadas, ya que no se utilizan en el desarrollo del dispositivo. En la primera pantalla, se debe seleccionar la placa de red de la PC (podría tener mas de una), en la que se encuentra instalado, y luego puede realizar un análisis. A continuación se muestra una imagen de un análisis hecho sobre una PC.

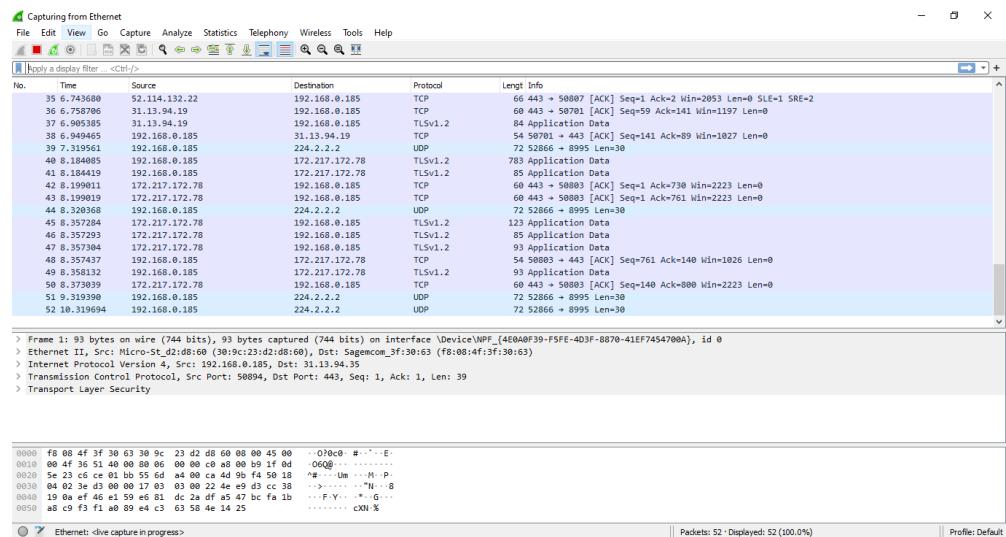


Figura 4.9: análisis de los datos dentro de una red utilizando WireShark

Como se ve en la figura, se observa IP de origen, Ip de destino, y ambos puertos, y además el paquete TCP reensamblado, y el tipo de protocolo utilizado para la comunicación.

Este software, se va a utilizar, para comprobar la conexión de los programas que realicen seguimiento de satélites. Además, se puede comprobar los protocolos implementados por los programas.

Interfaz De usuario

resumen

Se van a definir las interfaces sobre las computadoras de la institución. Estas interfaces, se basan en software existente. Para seleccionar las interfaces, se realiza una comparación de todos estos. Luego de esto, se va a mostrar como realizar la configuración de los dos software seleccionados.

5.1 introducción

En esta sección, analizamos los software que son capaces de mostrar las posiciones de los satélites en tiempos real. Luego de un análisis, se seleccionan dos de estos mismos. Los software existentes deben tener capacidad de enviar las posiciones de los satélites usando la red local de la institución. Luego del análisis de los programas, se muestran cuales deben ser los pasos a seguir para poder configurar dichos programas.

5.2 Interfaz de Usuario

La figura 2.1, se observa que hay estaciones de trabajo. En esta sección, definimos el software de seguimiento de satélites, para las estaciones de trabajo del lugar. Algunos de los software que se analizan fueron obtenidos desde la página web de la AMSAT (asociación mundial de satélites de radioaficionados), y otros desde la investigación a través de internet. Los distintos software analizados son los siguientes:

- Gpredict
- Stelarium
- Orbitron
- Celestia
- Pass

Hay que destacar, que todos estos software, requieren la ubicación del usuario (en este caso, de la antena), en sistemas de coordenadas terrestres (latitud y longitud del lugar, con su respectivo signo).

5.2.1. Orbitron

Este software permite la ubicación de satélites en tiempo real, permite simulaciones donde se pueden observar el día y la hora aproximada de llegada del satélite al plano del horizonte (ver capítulo 8) donde se encuentre la antena. Además, permite que observar la posición del sol y la luna en tiempo real.

Como contra de este software, es que no permite realizar el seguimiento de estrellas, y no puede conectarse a la red, para enviar estos datos a través de la red local. Solo tiene conexión con los puertos (puertos USB) de la PC para obtener los datos. Para poder enviar estos datos, a través de la red, se debe simular un puerto virtual, obtener esos datos, y enviarlos a través de la red local. Es decir, requiere un programa que haga de intermediario.

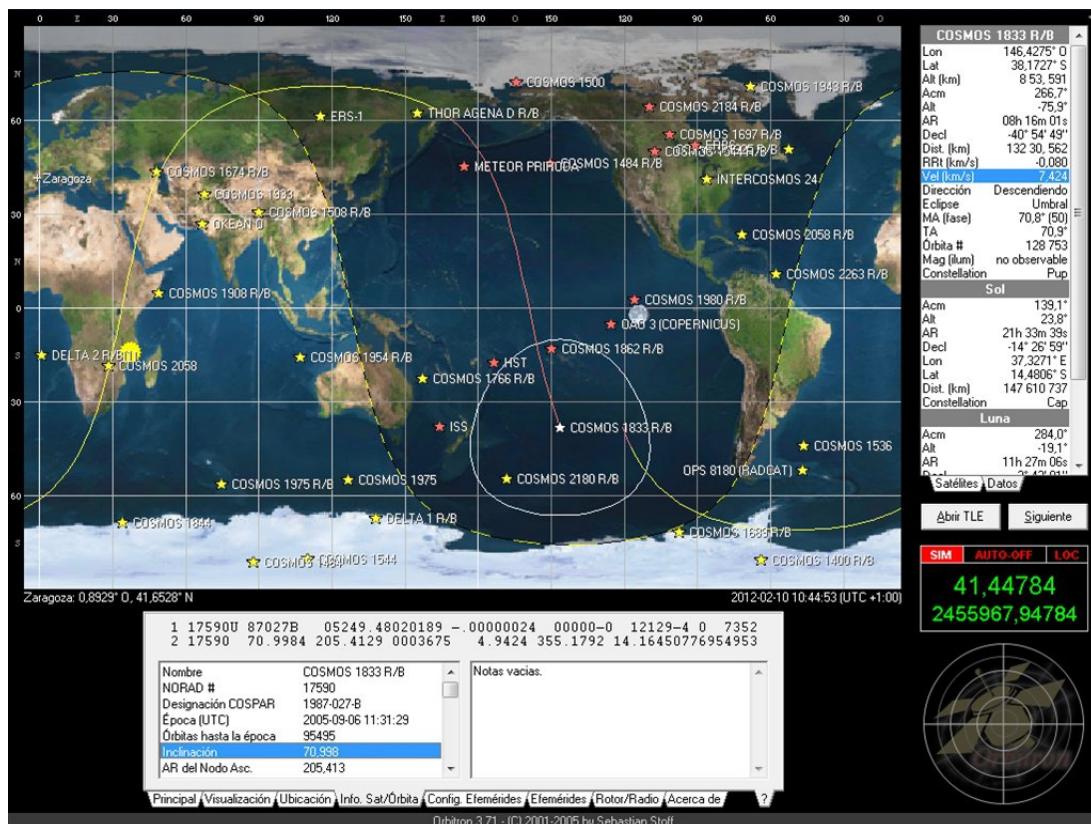


Figura 5.1: Captura de pantalla del software Orbitron

5.2.2. Stellarium

El software stellarium, posee un seguimiento en tiempo real de estrellas, y esta pensado para realizar el seguimiento usando telescopios. Tiene distintos módulos, lo que lo hace personalizable. Tiene soporte para estrellas, y satelites. Estos datos, pueden enviarse a través de la red, para que los reciba un telescopio, y realize el apuntado del telescopio. Existen plugins, de este software, que permiten enviar las coordenadas a dispositivos mediante la red. Una imagen del software puede visualizarse en la figura 5.2

5.2.3. Celestia

Celestia es un software, que permite realizar "viajes." a las estrellas. Este software tiene un fin educativo, puede realizar algunas simulaciones, y se puede viajar al espacio a donde uno desee". Este software no posee ningún tipo de comunicación con el exterior.

5.2.4. Pass

Pass no es una aplicación para PC en sí. Es una página web con todos los satélites encima de un mapa. La página es <http://amsat.org.ar/pass>. La captura de pantalla se muestra en la figura 5.4.

Este software, obtiene la geolocalización a partir de la conexión a internet. A partir, de ellas, obtiene la orientación de la antena. Debido a que obtiene las coordenadas a partir de la conexión a internet, se tiene un error en el apuntamiento de la antena, ya que estas coordenadas están



Figura 5.2: Imagén al abrir el software Stellarium

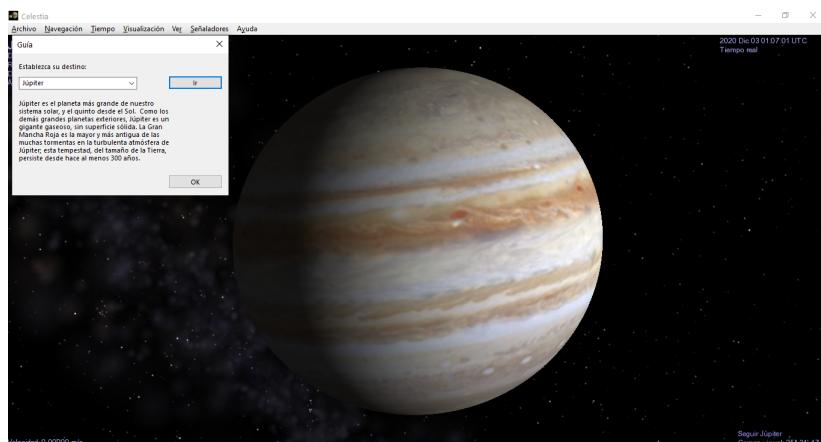


Figura 5.3: Viaje a júpiter a traves de celestia

desfasadas respecto a las reales.

5.2.5. Gpredict

El software Gpredict es un software que muestra en tiempo real las ubicaciones de los satélites dentro de un mapa, que permite, la configuración de sistemas receptores y sistemas de rotación para antenas. El mismo es libre, su código fuente está disponible en internet, como así suprograma compilado, el mismo puede usarse en Windows y Linux. Este, provee una base de datos de satélites, y se actualiza con la frecuencia que se configure. Además, soporta envío de datos mediante el uso de una red local, configurando sus parámetros. Una captura de pantalla puede verse en la figura 5.5.

5.3 Comparativa de interfaces y selección del software

Una vez, brindando las comparativas de cada software relevado, se realiza una tabla comparativa en la tabla 5.1

De la tabla, se observa que hay dos software que pueden usarse sin ningún complemento adicional: Gpredict y Stellarium. El stellarium es un software pensado para telescopios, pero conociendo

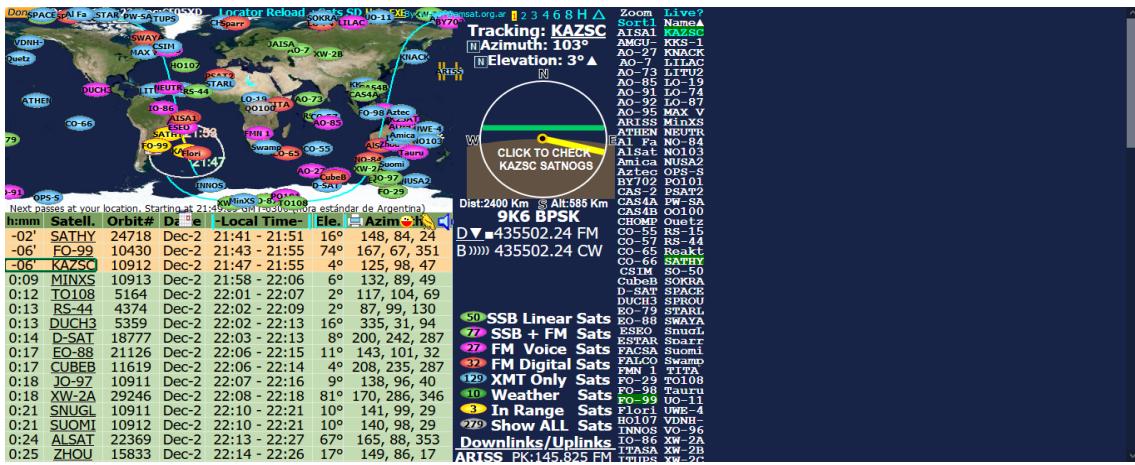
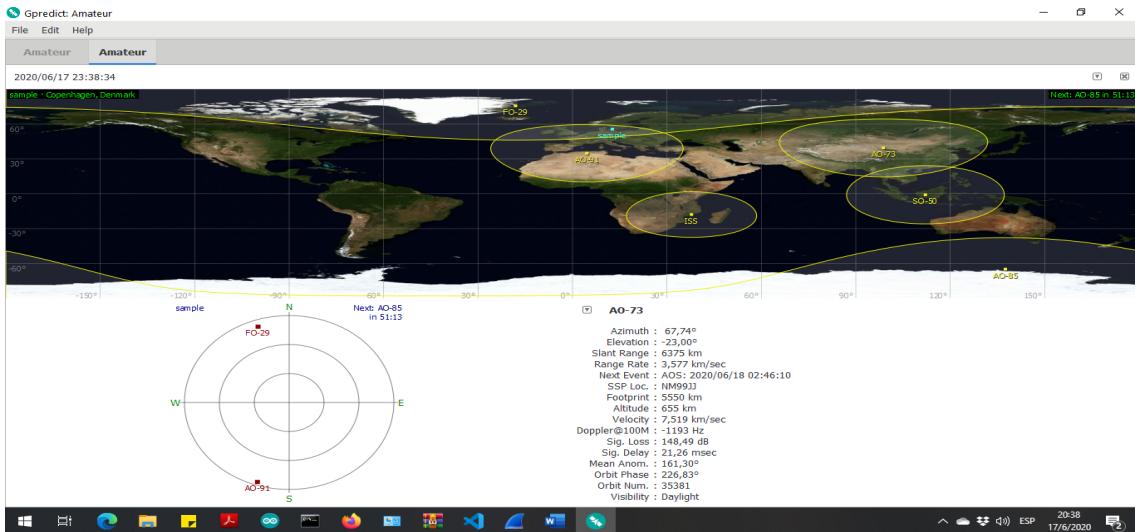
Figura 5.4: Captura de pantalla de la página <http://amsat.org.ar/pass>

Figura 5.5: Captura de pantalla del software Gpredict

su protocolo, puede obtenerse las coordenadas para apuntar una antena, pues el software no es capaz de reconocer el tipo de dispositivo a apuntar. El Gpredict, es un software especializado en seguimiento de satélites en tiempo real. En lo que resta del presente texto, se utilizará el software Gpredict y el Stellarium para realizar el apuntamiento de la antena.

5.4 Software Gpredict

El software Gpredict, se basa en el software libre. Su código fuente esta disponible para descargarse y compilarse, o puede descargarse directamente el archivo binario compilado. Tiene soporte para entornos Windows y Linux. Su lenguaje es C, y se basa en la librería GTK para el entorno gráfico. El software, posee un código que calcula las futuras posiciones de los satélites, en base a modelos matemáticos dentro de su código, estas se actualizan según se actualice su base de datos de satélites. Estas bases de datos se actualizan con frecuencia predeterminada por el usuario. Para el soporte de red, su software basa su comunicación en la librería llamada HAMLIB, disponible en varios lenguajes de programación para usarse. Esta librería, propone su propio protocolo de comunicación a través de la red. Esta librería, además tiene funciones relacionadas al ajuste de los receptores de comunicaciones mediante conexión a la red local.

Software	Orbitron	Gpredict	Celestia	Stelarium	pass
Posición Geográfica	Sí	Si	No	Si	Si
Soporte para redes	No	Si	No	Si	No
Satélites	Si	Si	No	Si	Si
Estrellas	No	No	Si	Si	No

Tabla 5.1: Comparativa entre los distintos software

5.4.1. Librería Hamlib

Hamlib es una capa de software dedicada al manejo de radios y rotadores de tipo comercial,y no comercial. Esta capa de software actúa por debajo de las interfaces gráficas, y aplicaciones de usuario. El software principal(en este caso Gpredict), llama a esta librería para comunicarse con los dispositivos de radio o rotadores.

Esta capa de software se relaciona con distintos tipos de radios, y tiene protocolos comerciales(algunos, se debe ver la lista de dispositivos soportados) ya implementados. Además, permite la incorporación de dispositivos que aun no estén implementados,ya que puede descargarse su código fuente desde internet.

Esta librería, define cuatro programas para realizar pruebas, estos programas son:

- rigctl: Manejo de radios por puerto serie
- rotctl: Manejo de rotadores por puerto serie
- rigctld: Manejo de radios mediante el protocolo TCP/IP
- rotctld: Manejo de rotadores mediante el protocolo TCP/IP

Los dos últimos de la tabla anterior, son los que tienen interés dentro de este documento. Al realizar la opción de seguimiento de satélites, internamente Gpredict, llama a "rotctld", y mediante un protocolo de comunicación documentado dentro de la documentación oficial de Hamlib, se realiza el intercambio de mensajes entre la aplicación y el rotador.

5.4.1.1. Protocolo de comunicación

El protocolo de comunicación implementado dentro del programa "rotctld" se basa en texto plano, separado por espacios en blanco, y tiene un carácter de fin de mensaje, con el carácter salto de linea, o conocido como "\n" dentro del código ASCII.

Existen dos tipos de comandos: los denominados métodos Get y métodos Set. Los primeros obtienen información del rotador, mientras los últimos, le envían información al rotador de que es lo que debe realizar. Algunos de los comandos utilizados por este programa(Gpredict en conjunto con rotctld) son los siguientes:

Métodos Get		Métodos Set	
Comando	Descripción	Comando	descripción
p	Obtener posición actual del rotador	P	Enviar posición al rotador
q	Desconectar el rotador Cierra la conexión	S	Parar el rotador

Tabla 5.2: Comandos de rotctld usados por Gpredict

Además de estos comandos, existen un comando que no encaja en la categoría de Get o Set, es el comando M, que le indica la velocidad de seguimiento de la antena, este no esta disponible en Gpredict.

Esta librería, además, posee un formato de respuesta, el cual se establece con separadores de espacio en blanco, y terminados en un salto de línea, para el caso de métodos GET. Estos métodos, piden información al rotador, y está definida cada respuesta dentro del manual de HAMLIB. Estas, pueden verse en su manual.

5.4.2. Selección de satélites y creación del perfil

El software Gpredict, permite seleccionar los satélites a realizar seguimiento mediante la elección de los mismos, y creando lo que se denominan "modulos". Cada módulo puede seleccionar los satélites a seguir, además, pueden agregarse satélites a seguir, añadiendo la base de datos manualmente.

Al iniciar el software(ver figura 5.5) este viene con un módulo predeterminado, que se denomina "Amateur". Este viene con algunos satélites precargados.

Para crear un perfil, debe dirigirse a file → new module, como muestra en la figura 5.6 Al

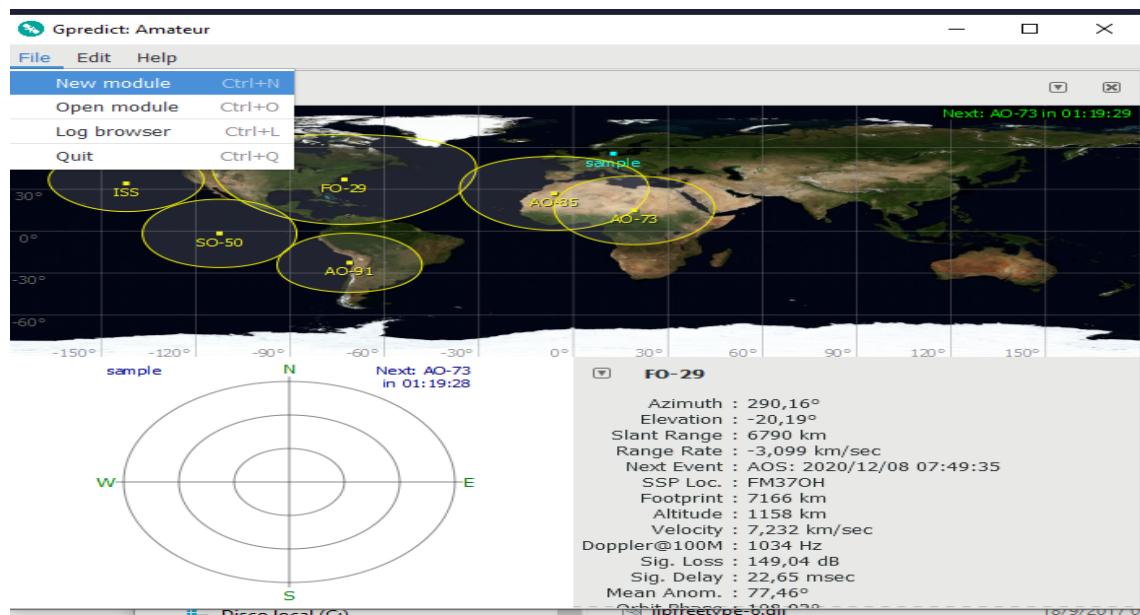


Figura 5.6: Creación de un módulo dentro de Gpredict

realizar esto, se abre la ventana, donde selecciona los satélites a seguir. Esta ventana se muestra en la figura 5.7. En esta ventana, seleccionamos los satélites, y lo nombramos a nuestro módulo. En este caso, lo denominamos arduino.

Donde dice "Ground station",se debe configurar latitud y longitud de la antena. Luego se presiona OK.Después de esto, ya se creo el módulo. Para abrirlo cuando desee, debe dirigirse a file → open module, y elegir "arduino" como módulo.

5.4.3. Configuración del rotador en Gpredict

Para configurar el rotador,en la página que se abre al iniciar el programa(ver figura 5.5), debe dirigirse a edit → preferences. Al realizar esto, se abre la siguiente ventana:

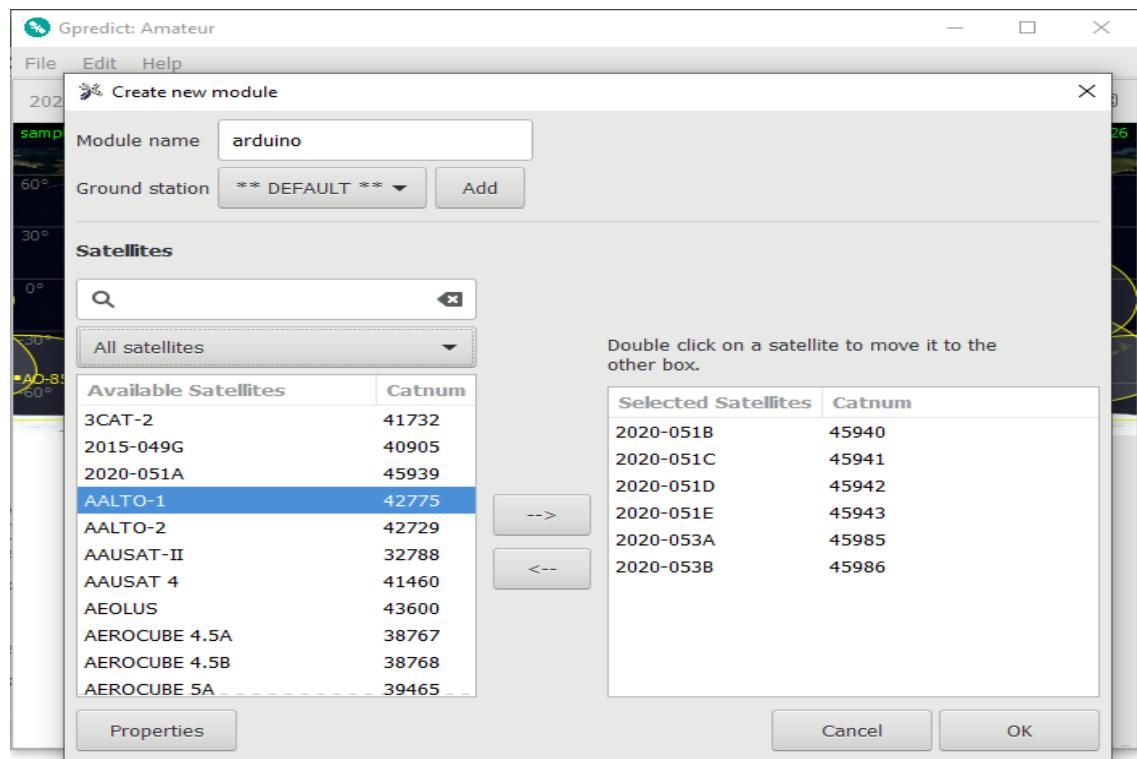


Figura 5.7: Selección de satélites para el nuevo módulo creado

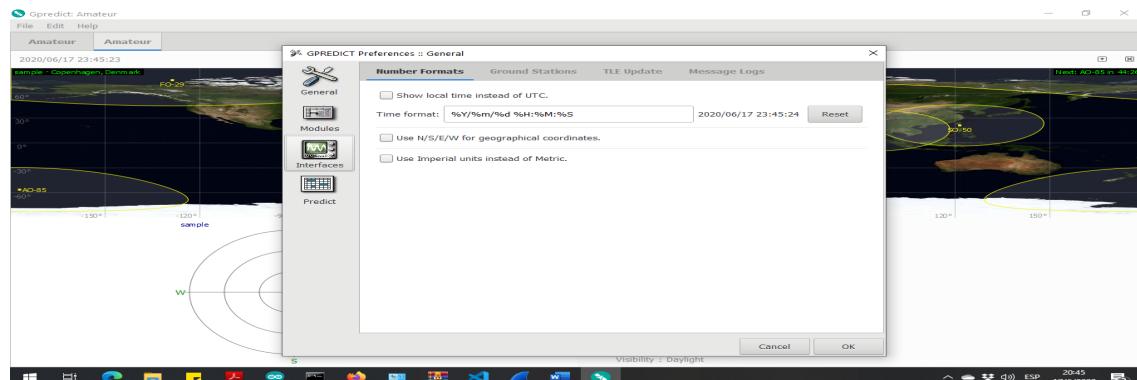


Figura 5.8: Configuración de rotador

Luego de esto, debe presionar en la parte izquierda de la figura anterior, donde dice "interfaces". Al realizar esto, debe presionar, en la pestaña "rotators". A continuación se la ventana que se abre al presionar sobre interfaces

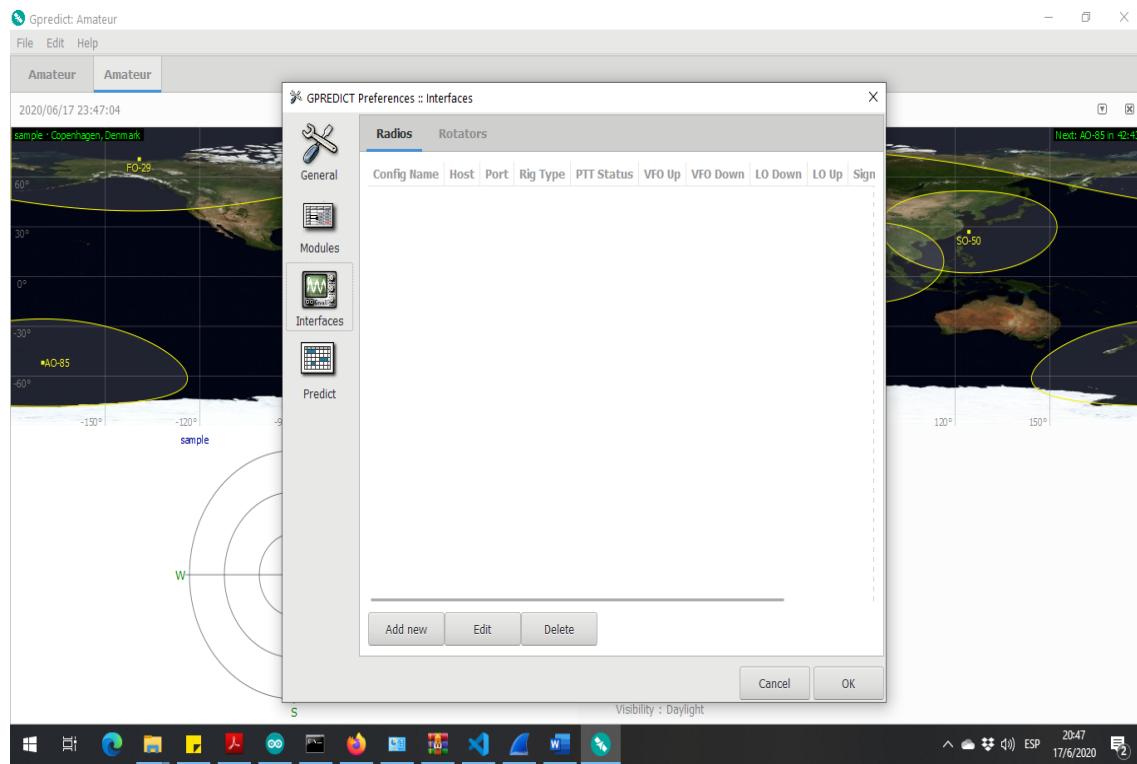


Figura 5.9: Configuración de rotador

Luego, debe presionar en la pestaña rotators, y presionar donde dice "add new", y se abre una ventana como la que se muestra en la figura 5.10:

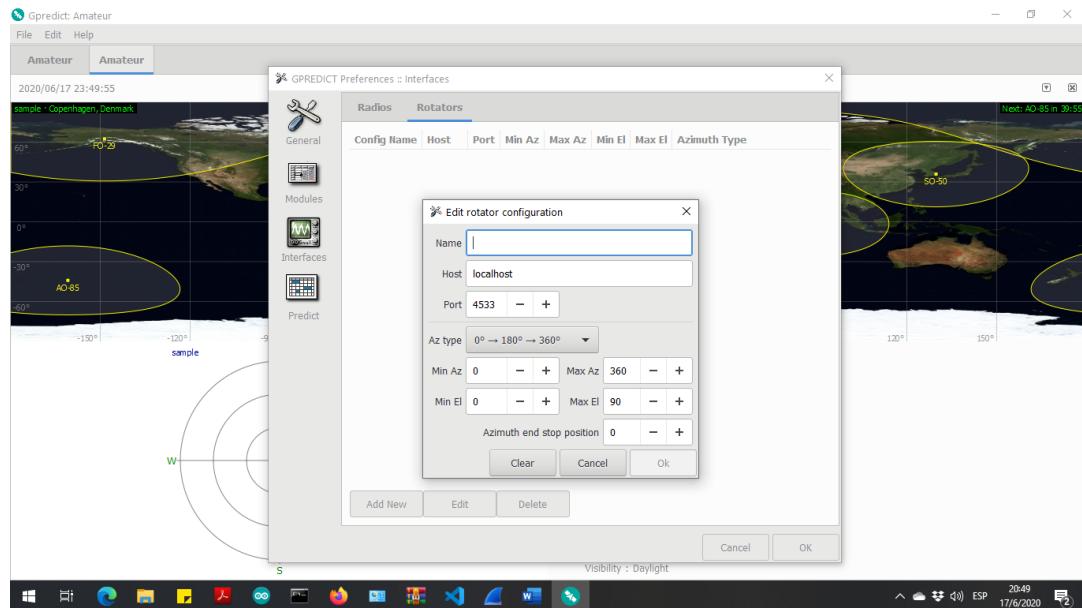


Figura 5.10: Configuración de rotador

Los parámetros a configurar son nombre del rotador(el que deseé el usuario), angulos máximos y mínimos de la antena. Estos parámetros se verán en la fase 3 del presente trabajo. Además, se observan que se debe definir el host y el puerto. El host, es la IP del dispositivo desarrollado en el presente documento, y el puerto, se define según el gusto del usuario,en nuestro caso, se va a

utilizar el puerto por default(4533).Luego de llenar los datos, debe seleccionar OK, y con esto, está configurado el rotador. Se pueden definir tantos rotadores como se deseen, se debe repetir los pasos.

Una vez creado el rotador, para visualizar el rotador desde la pantalla principal, debe dirigirse a la parte derecha de la pantalla principal, y presionar ahí. En ese lugar, aparece un menu, debe dirigirse a "antena control", como se ve en la figura 5.11

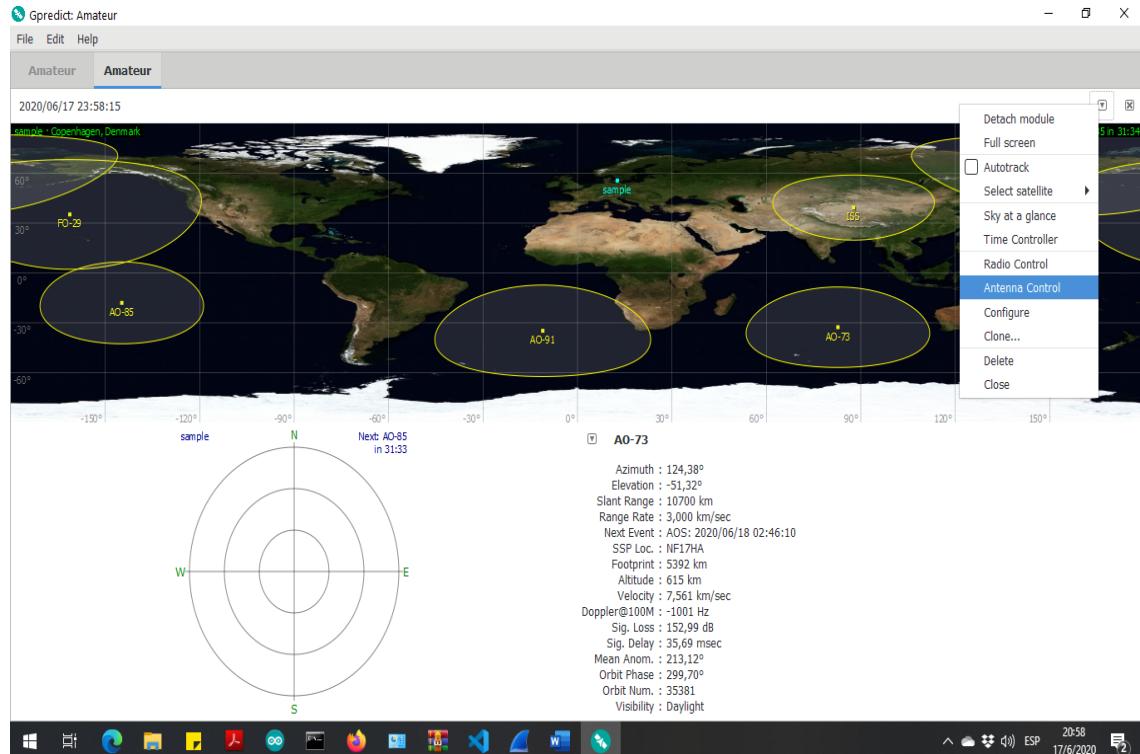


Figura 5.11: Ver el rotador o rotadores configurados

Al realizar este paso, aparece el rotador de antena, que se ve en la siguiente figura

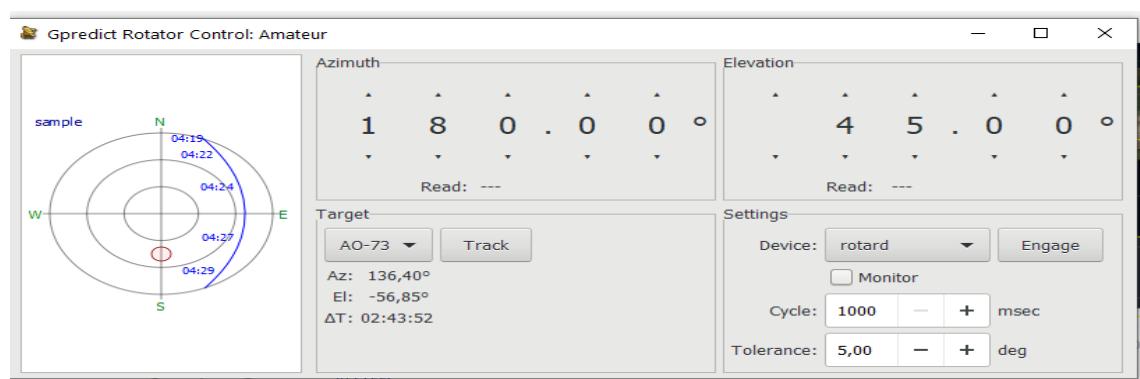


Figura 5.12: Panel de control del rotador del Gpredict

En esta figura, se observan dos botones, y dos menús, y un mapa de la antena. Estos dos botones son "track" y "engage". En el menú de track, realizamos el seguimiento automático. El menú, engage, sirve para realizar la calibración de la antena. El sistema de coordenadas empleado por este software son horizontal-azimutal, con el cero en el polo norte. Estas coordenadas son definidas

más adelante, en específico en la fase 3, donde se muestra la configuración de este software para el trabajo de este documento.

Las medidas angulares, corresponden a los angulos de la antena, estos se van a mostrar en el capítulo correspondiente a sistemas de coordenadas. Por ahora, solo basta con conocer la interfaz, luego se ajustan los detalles, para configurar el Gpredict, en base a la antena que tiene la institución, ajustando angulos y límites de la misma. Esto se realiza en la fase 3, donde se van a mostrar los sistemas de coordenadas, y las relaciones entre ellos. En el próximo capítulo, se muestra como debe realizarse el seguimiento de satélites, mediante la comunicación con el dispositivo desarrollado en este documento.

5.5 Software Stellarium

Este programa, es libre, y esta pensado en el apuntamiento de telescopios. Posee seguimiento de estrellas y satélites. Debido a que el apuntamiento de los telescopios, y/o antenas, depende de la posición del observador en la tierra, se debe configurar la posición geográfica dentro del software, para que se pueda realizar el apuntamiento de forma correcta. Para elegir la posición, dentro de el software stellarium, oprimiendo el botón F6 o moviendo el cursor del mouse hacia la izquierda, y seleccionando posición, se abre la siguiente ventana de configuración:

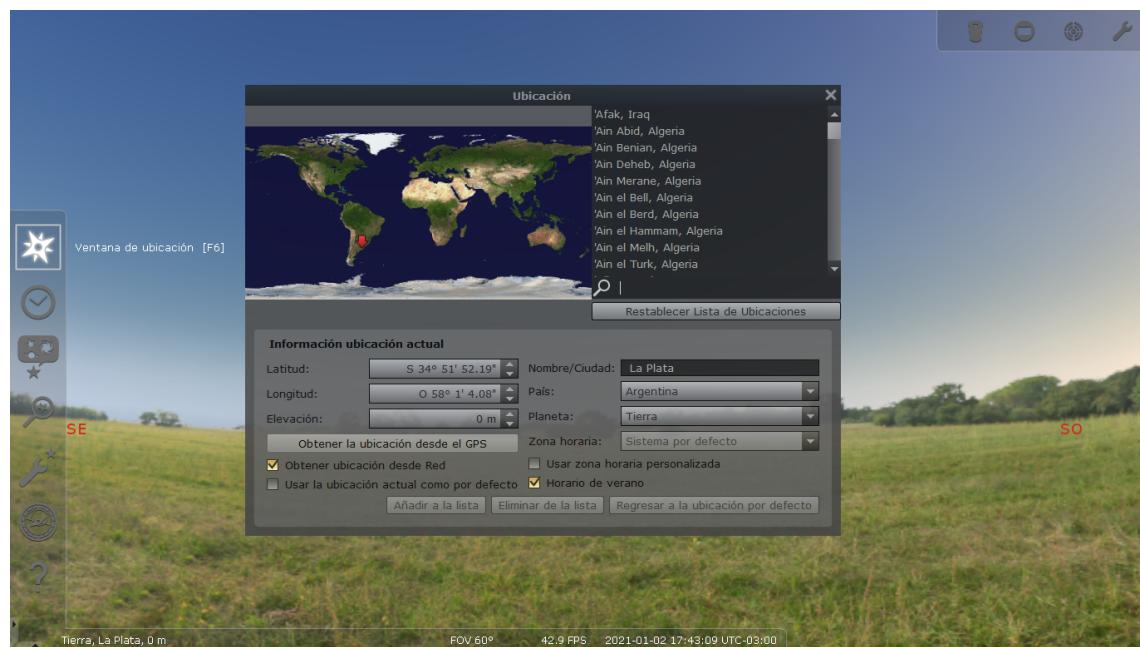


Figura 5.13: configuración de coordenadas locales dentro de stellarium

En esa ventana, debe seleccionarse, la latitud, longitud y altitud del lugar en el que se encuentra el telescopio o antena.

5.5.1. Configuración de la red en Stellarium

Para poder conectar una PC de la institución, con el dispositivo, debe configurar el puerto(socket) con el cual se intercambiaran mensajes, y la dirección IP del dispositivo receptor,que será el encargado de mover la antena o telescopio. Para configurar el telescopio, debe dirigirse a configuracion, luego presionar en la pestaña pluggins, y en la parte izquierda seleccionar "control del telescopio". A continuación se deja la imagen de este proceso:

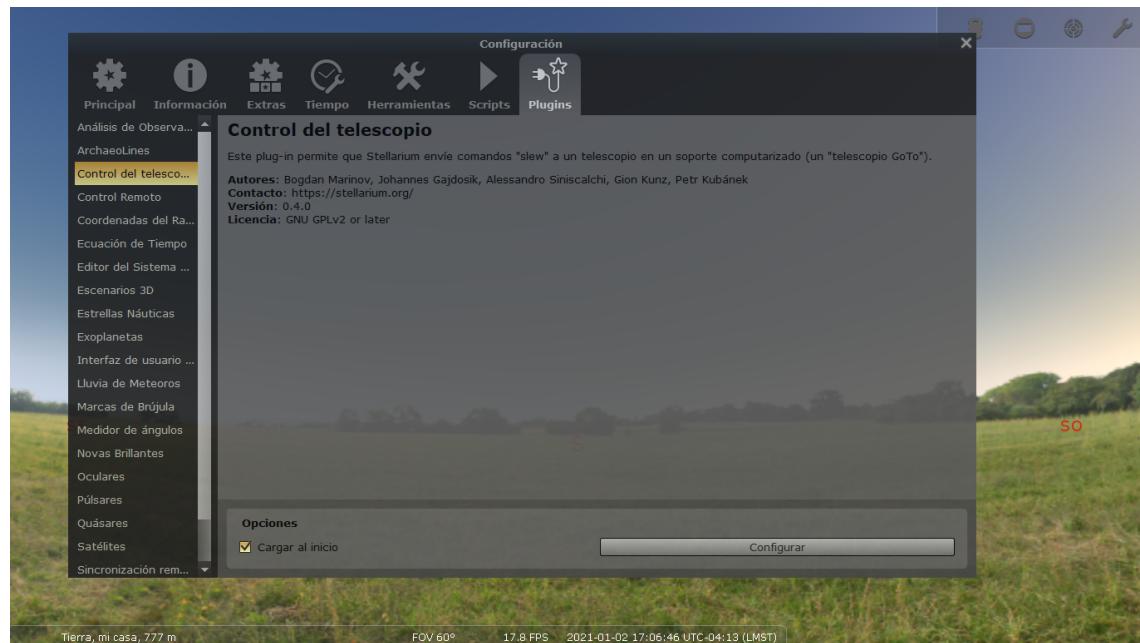


Figura 5.14: configuración de coordenadas locales dentro de stellarium

Una vez, en esta ventana, se debe seleccionar el botón “configurar”, debajo a la izquierda de esta ventana, y luego, se abre una ventana como la que se muestra a continuación.

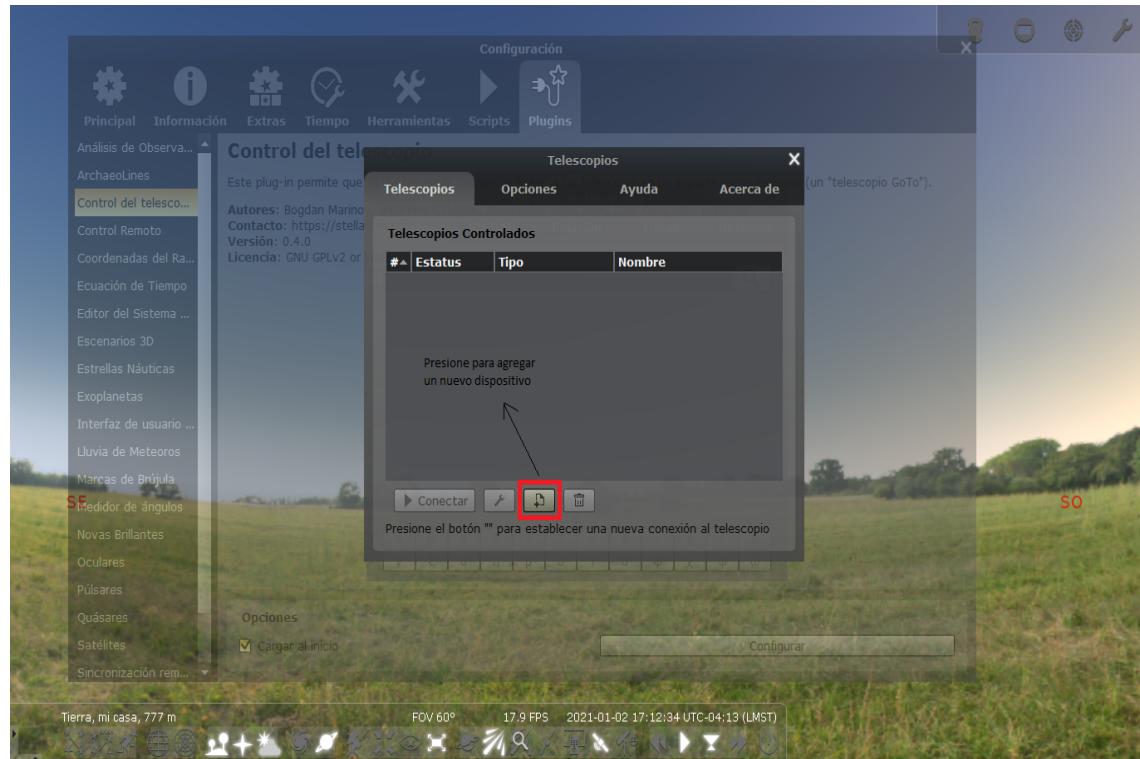


Figura 5.15: configuración de coordenadas locales dentro de stellarium

Luego, debe presionar donde muestra la imagen anterior para empezar a agregar el primer dispositivo. Se realiza un click sobre el símbolo recuadrado en rojo de la figura anterior, se abre la siguiente ventana, donde deben elegirse los parámetros para configurar el dispositivo hacia cual

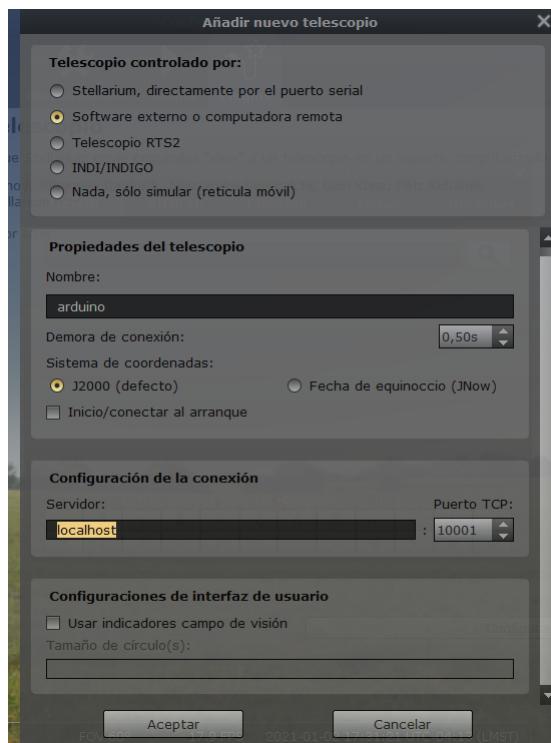


Figura 5.16: Ventana de configuración de red del software Stellarium.

enviará los datos. Donde dice "localhost", debe introducirse la IP del dispositivo que es capaz de realizar el movimiento de la antena, el nombre, es a elección del usuario, y hay dos tipos de coordenadas: J2000 y fecha de equinoccio. Elejimos J2000. Este sistema de coordenadas, refiere a que los movimientos de los astros en el cielo, poseen pequeñas variaciones, y requieren una corrección. Esto esta relacionado a los movimientos de la tierra.

Una vez configurado el dispositivo, queda por intentar entender el protocolo, que este emplea para comunicarse con los dispositivos.

5.5.2. Protocolo de comunicación Stellarium

El stellarium, no es capaz de recibir datos en este modo de configuración, pero si es capaz de enviar datos al dispositivo conectado en la red. Este emplea un dos números binarios, uno para cada eje. Según su documentación, al configurar un dispositivo, este invoca a un programa denominado "telescope server", el cual trabaja sobre TCP/IP. Este, es el que envía las coordenadas al dispositivo. Los mensajes se basan en bytes, donde estan agrupados del siguiente modo:

- LENGTH : 2 bytes - Indica la longitud total del mensaje
- TIME : 8 bytes. Tiempo UT a partir de 01/01/1970 en microsegundos. Actualmente en desuso
- RA: 4 bytes(sin signo) - ascension recta:
- DEC : 4 bytes,con signo
- status: 4bytes con signo. Si status = 0 ->OK, si status<0 hay algún error.

Cabe destacar, que internet, o las redes, son protocolos big-endian, mientras que el microcontrolador atMEGA328p(arduino UNO) es little-endian. Esto se debe solucionar reordenando los bits de llegada hacia el dispositivo. Además, debe realizar una transformación de coordenadas para poder realizar el apuntamiento de la antena.

Software del Microcontrolador

Resumen

Se describe todo el proceso realizado, para programar el software para el microcontrolador ATmega328P seleccionado, bajo el entorno arduino, y todas sus librerías. Además, se muestra como deben interconectarse los componentes, y los circuitos correspondientes para realizar las pruebas sobre cada parte del software. Para cada parte del software desarrollado, se han realizado las correspondientes pruebas, y se presentan los resultados en cada sección. Luego una vez, realizadas todas las pruebas, se procede a unir todo el software, y se realiza una prueba final, y se convalidan los resultados. La versión de software presentada en este capítulo, no es la versión final, sino que se agregan más funcionalidades en la siguiente fase(fase 3).

6.1 Introducción

En este capítulo, se aborda una parte del desarrollo de software de todo el proyecto. La parte restante, se aborda en las fases tres y cuatro respectivamente. El software debe ser capaz de responder a los requerimientos planteados en la tabla 1.1. En esta parte del trabajo, dividimos la programación en varias etapas:

1. Control de posición
2. Autocalibración
3. Scheduler o planificación
4. Conexión del dispositivo a la red. Conectar con Gpredict y stellarium

Una vez resueltos, todos los puntos anteriores, se pasa a unir todo el código desarrollado a lo largo de las presentes secciones de este capítulo. Luego, se deja comentado dentro del código aquellas partes que se desarrollen mas adelante a lo largo de este texto. La parte que resta es la transformación de coordenadas, de coordenadas ecuatoriales a coordenadas locales horizontales. El código en su versión final se muestra en el apéndice.

6.2 Diagrama del sistema

En base a los componentes seleccionados, en el capítulo 3, estos deben interconectarse entre si, mediante sus protocolos de comunicación. El sistema de control, se compone del diagrama en bloques mostrado en la figura 6.1.

Este diagrama en bloques muestra cual es el protocolo de comunicación utilizado por cada dispositivo. La conexión entre el bus SPI soportada por el microcontrolador y el chip ethernet, es el modo 0(ver apéndice A), y la conexión con el display es mediante el protocolo SPI(descripto en el apéndice A). El bloque de drivers de motores, son dos controladores de motores, y dos encoders, pero se ha puesto una única caja, ya que el sistema de control es el mismo en ambos motores.

6.3 Esquema eléctrico de los componentes

El lenguaje de programación del entorno arduino es C/C++, y se desarrolla una parte en C y otra parte en C++. El software, debe comunicarse con el display LCD(ver figura 3.2) mediante

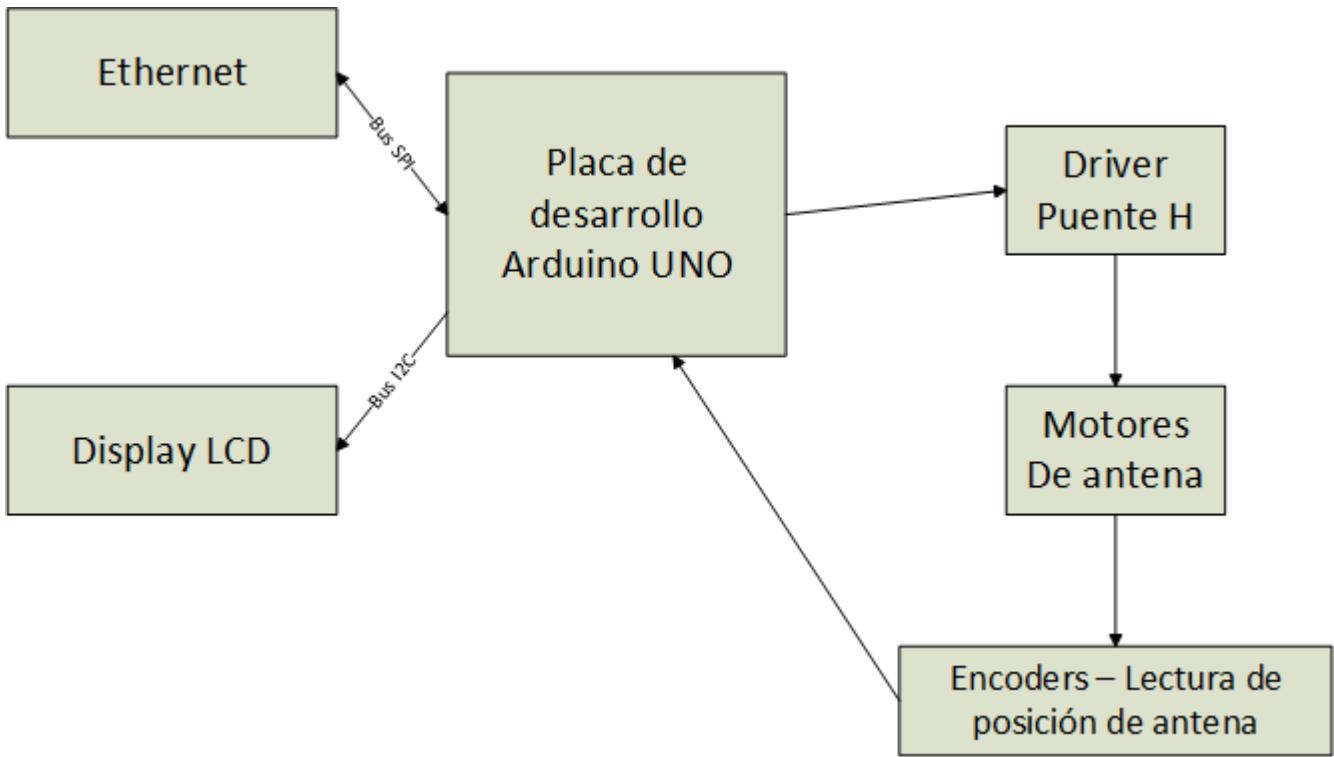


Figura 6.1: Diagrama general del sistema de control

el protocolo I2C. Este protocolo esta descripto en el apéndice A, y con el chip W5100(ver figura 3.1) mediante el protocolo SPI(ver apéndice A). Por ende, antes de empezar a realizar cualquier tipo de programación, se deben conectar los componentes entre si, para poder realizar la programación, y las pruebas. No se ha utilizado ningún simulador, ya que se disponen de los materiales y componentes, e instrumental necesario para realizar la medición sobre los elementos directamente.

Antes, de realizar cualquier conexión, se realiza un análisis de los pines disponibles en la placa de desarrollo Arduino Uno, para realizar el desarrollo sin cambiar los pines físicos a lo largo de este trabajo. En primer lugar, se consideran los pines que deben ser utilizados para conectar el chip ethernet. Este debe poseer cinco pines, de los cuales, tres son obligatorios, y no pueden cambiarse desde el software, ya que utiliza el puerto SPI físico del microcontrolador que esta en la placa, estos son los pines 11,12 y 13 respectivamente. El pin de Slave Select, y reset, pueden elegirse indistintamente usando cualquier puerto digital. La conexión del LCD, requiere de comunicación I2C, el cual utiliza dos pines de la placa de desarrollo. Estos pines son los llamados A4 y A5 dentro de la placa.Además, de estos, se requiere cuatro pines adicionales, para controlar el sentido de giro de cada motor(dos pines para cada motor). Estos, deben poseer modulación por ancho de pulso, para poder realizar un control de velocidad en proximos desarrollos de este dispositivo. En este informe, solo se realizará un control de tipo ON/OFF. Los pines disponibles que poseen modulación de ancho de pulso son los pines 9-10, 5 y 6. Luego de estos pines, se deben seleccionar dos pines adicionales, para poder medir la posición angular de la antena,de estos pines se eligen los pines A0 y A1 respectivamente. A continuación, se deja la imagen de cuales son aquellos puertos que se han seleccionado.

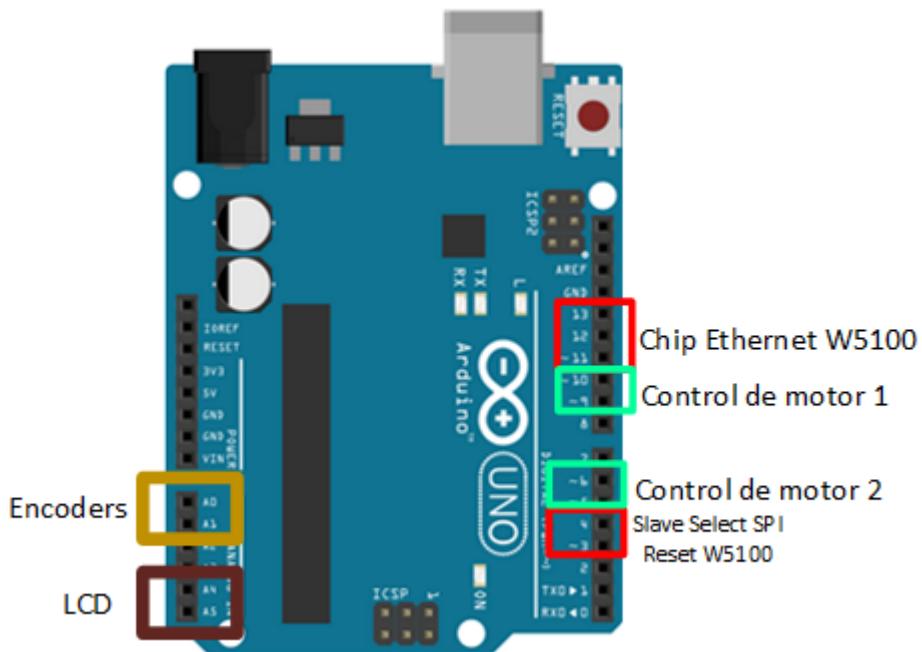


Figura 6.2: Pines seleccionados sobre la placa de desarrollo arduino UNO para realizar el prototipo

Una vez, definidos los puertos a utilizar, se deben conectar los componentes a la placa de desarrollo. Esta conexión, se realiza usando cables denominados "dupont" en una protoboard. Para saber como se deben conectar el display LCD y el ethernet Shield W5100, se deben conocer su disposición de pines,o en lenguaje de la jerga electrónica, se debe conocer el "pinout"de cada dispositivo. La explicación de cada pin disponible de cada dispositivo,se muestra en el apéndice A. Se muestra el pinout de cada dispositivo en la figura 6.3:

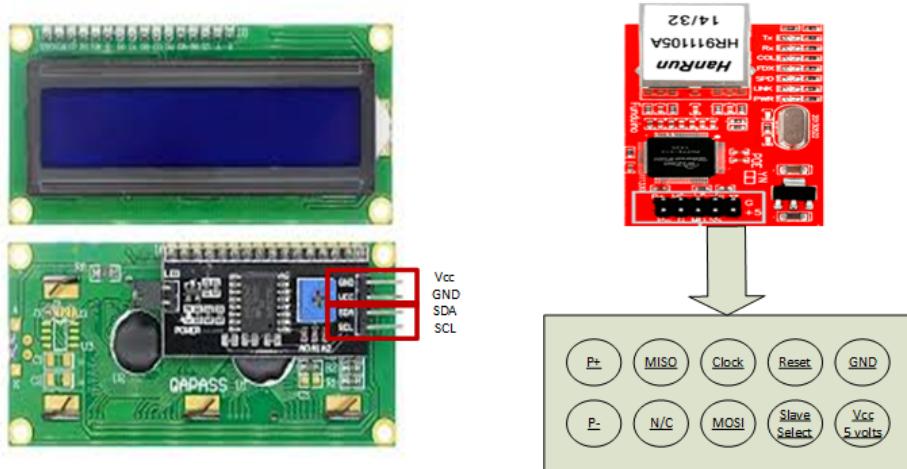


Figura 6.3: Pinout de ambos componentes, para poder realizar la conexión con la placa de desarrollo de Arduino Uno

Para realizar las conexiones, revisando el diagrama de conexiones dentro del apéndice A, se deben conocer cuales son los pines que corresponden a las señales SPI e I2C dentro de la placa de desarrollo. Observando el manual y la hoja de datos, obtenemos que los pines de la placa de desarrollo son los siguientes:

- Chip W5100
 - Pin 13 SCK
 - Pin 12 MISO
 - Pin 11 MOSI
 - Pin 4 Slave Select(SS)
 - Pin 3 Reset
- Display LCD
 - Pin A5 SCL
 - Pin A4 SDA

Por último, ya que no se dispone de la conexión al motor aún, se prueban conectando en los pines A1 y A0, dos potenciómetros, de 10Kohms cada uno, ya que cada motor tiene adosado un potenciómetro que es utilizado como encoder. Para conocer si el sentido de giro es correcto, en los pines 5,6,9 y 10, se conecta una resistencia y un led, estos led tienen por finalidad, mostrar si el motor gira en sentido correcto, pero estos no se muestran en el esquemático. El esquema de conexiones se muestra en la figura 6.5.

A continuación se deja una imagen del armado del circuito en una protoboard, en ella, se incluyen las resistencias y diodos led que no se encuentran en la imagen 6.5.

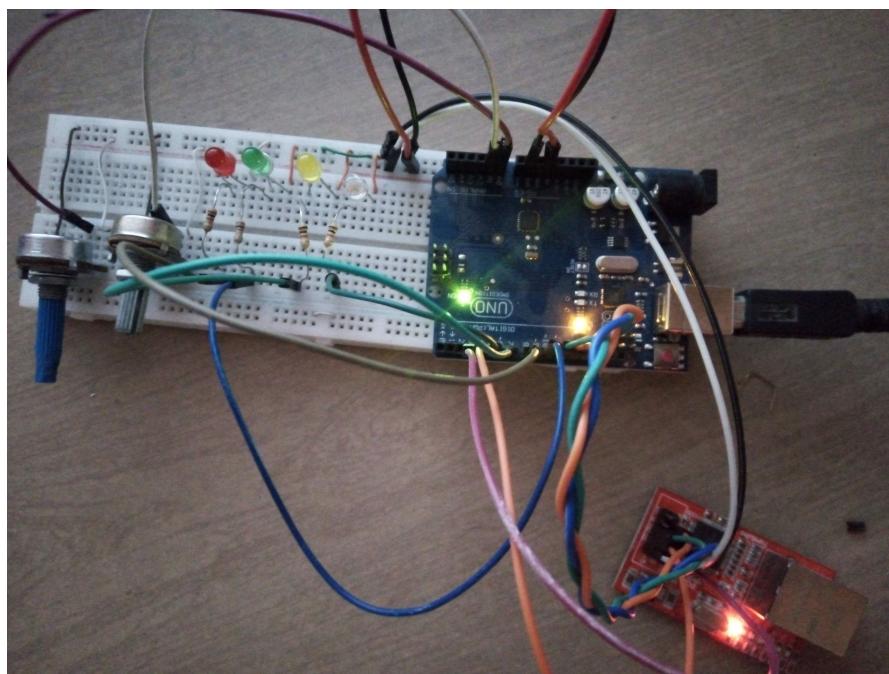


Figura 6.4: Imagen del protoboard armado para realizar las primeras pruebas con el software.

Donde se ha conectado los diodos led y los potenciómetros de la siguiente manera:

- led rojo: puerto 9
- led verde: puerto 10
- led amarillo: puerto 6

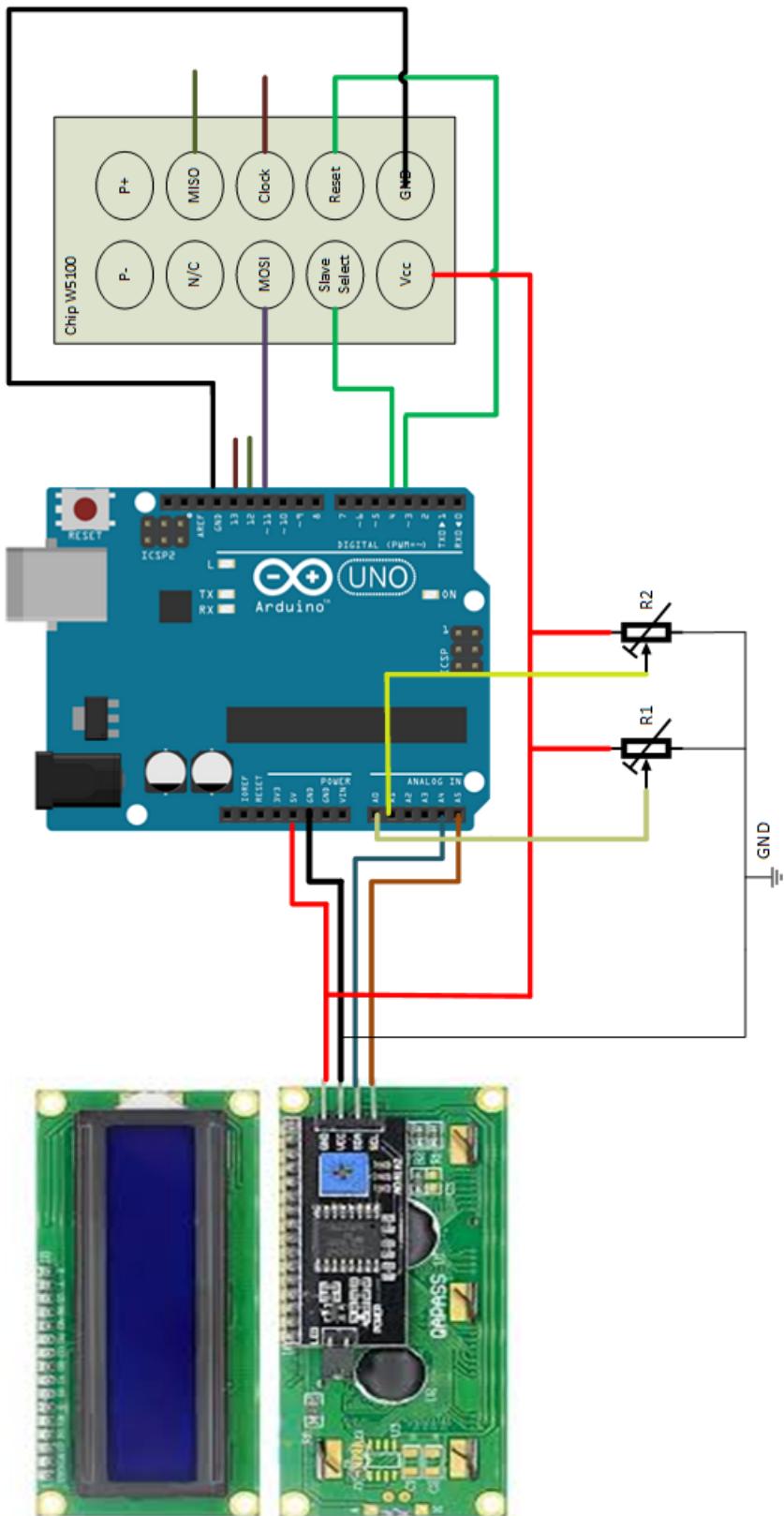


Figura 6.5: Esquema de conexiones entre la placa de desarrollo y los periféricos utilizados

- led azul(el led transparente de la imagen): puerto 5
- potenciómetro azul: puerto A0
- potenciómetro gris: puerto A1

6.4 Diagrama general del software

El software, debe cumplir los requerimientos presentados en el capítulo inicial del presente documento. En principio, debería tener la capacidad de realizar la autocalibración, al inicio de su programa. Por otro lado, en caso de que no se esté siguiendo ningún satélite o estrella, debería tener la capacidad de volver a la posición de equilibrio de la antena. Esta posición de equilibrio se denomina cenit.

Cada uno de los motores de la antena, posee adosado un potenciómetro, que gira con el motor, esto funciona como un sistema de encoders, para medir la posición angular en base a la tensión. Esta tensión, se mide sobre los pines A0 y A1 de la placa de desarrollo principal. Esta medida, debe actuar sobre los pines 5 y 6 para un eje, y sobre los pines 10 y 9 para el otro eje. Además, debe saber el sentido de giro al prender el led 5 y 6, o 10 y 9. El sentido de giro debe obtenerse de la función de autocalibración.

El sistema de control, es del tipo ON/OFF, el cual mide la posición y apaga el motor cuando llega a la posición indicada. Son dos controles independientes para cada motor. El estado de apagado, es equivalente a poner a nivel bajo los puertos 5 y 6, o 9 y 10, según de que eje se trate. Esta posición a la que debe moverse la antena, viene dada a través de la red, a partir de los programas presentados en el capítulo anterior.

Además, el software debe informar en todo momento al usuario de su estado (medida angular en ambos ejes, si esta en el cenit, debe escribir la palabra cenit), y la dirección IP asignada por la red, por medio del display LCD.

Por lo expuesto en los párrafos anteriores de la presente sección, debe realizarse un sistema temporizado, que lea los puertos analógicos A0 y A1, y realizar una acción de control en base a su valor. La acción de control es encender el/los motores en un determinado sentido de giro, y apagarlo cuando llegue a esta. Además, en caso que no esté realizando ningún seguimiento, se debe verificar que la antena se encuentre en el cenit. Esto debe realizarse, ya que podrían existir vientos, y/o condiciones climáticas adversas que puedan mover la antena de su posición de equilibrio(el cenit).

Por lo expuesto en párrafos anteriores, se desarrolla el diagrama de software en la figura 6.6. En este diagrama, se muestra el software desarrollado en el microcontrolador, y como actúan los distintos puertos del microcontrolador con el hardware. El driver de cada motor, en la figura, se realiza en la fase 4, y es un diseño de hardware. Los encoders 1 y 2 respectivamente, son potenciómetros adosados al eje de cada motor. Estos potenciómetros son de 10Kohms cada uno. Las coordenadas angulares, se denominan angulo de azimut y altura respectivamente, en astronomía de posición, y por eso se le dió ese nombre. En el capítulo 8 se tiene una descripción más detallada de este tipo de coordenadas.

En la figura se observan dos cuadros, uno denominado "POLLING", y otro denominado "ISR - Esquema de tiempos". El cuadro de POLLING indica que el puerto SPI se mira todo el tiempo, para conocer si existe uno de los programas de la PC(Gpredict, o Stellarium) quiere realizar un movimiento sobre la antena.

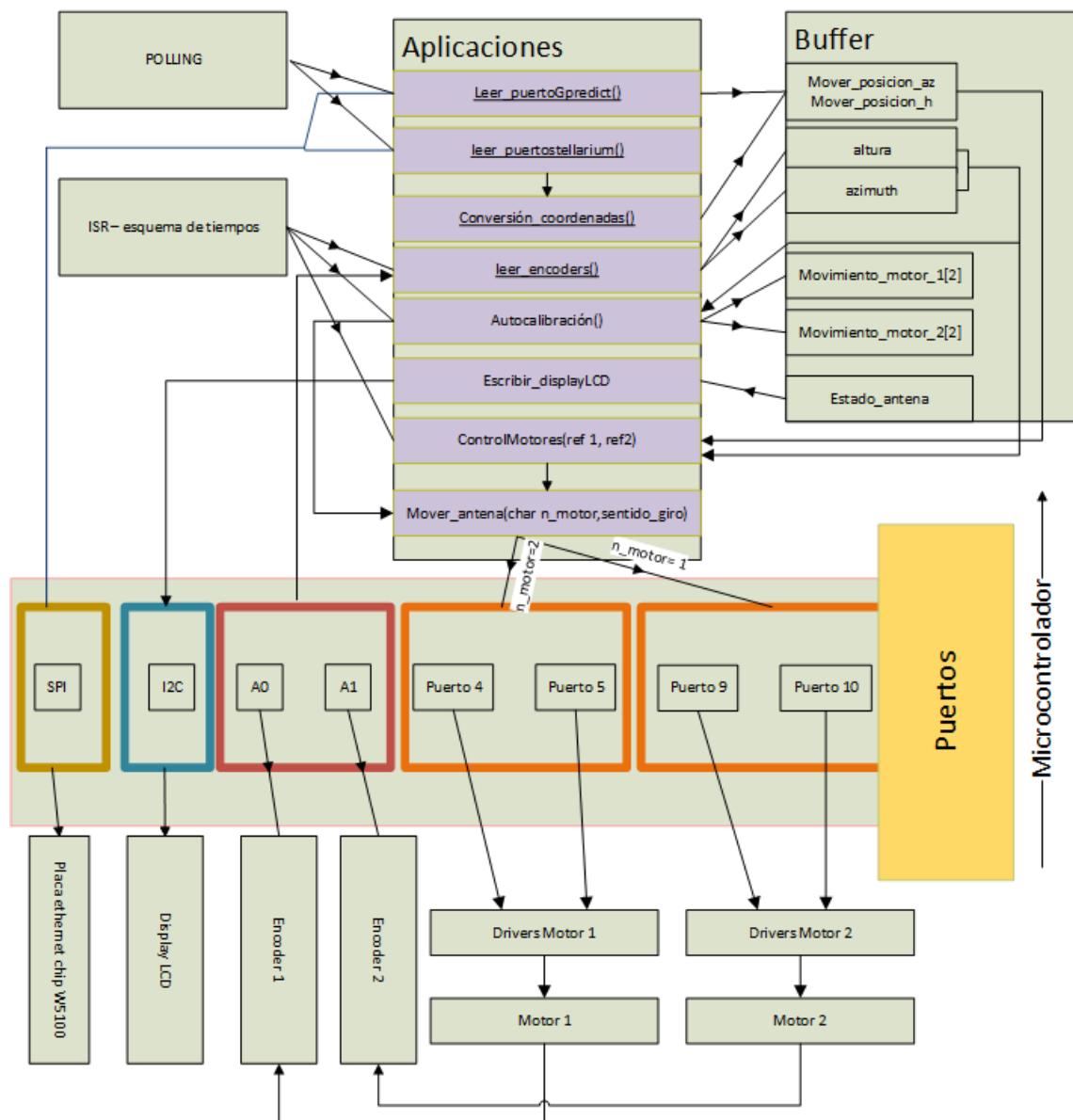


Figura 6.6: Diagrama general con las partes de software y hardware

El cuadro donde dice "Buffer", son las variables globales que interactúan con las funciones de el programa. Estas variables globales son:

- mover_posicion_az y mover_posicion_h: referencia para el control de posición, obtenida a través de los programas Gpredict o Stellarium.
- altura: Posición del ángulo de altura de la antena.
- azimuth: Posición del ángulo de azimuth de la antena
- movimiento_motor_1[2] y movimiento_motor_2[2]: en esta se guarda el sentido de giro de cada motor. Dado, que se requieren dos pines para controlar el sentido del motor, y el software debe guardar el sentido de giro, al encender un puerto u otro, estos se guardan en esta variable de tipo vector, siendo:

- movimiento_motor_1[0]: Guarda el pin correspondiente al sentido de giro de azimuth, sentido Oeste - Este
- movimiento_motor_1[1] : idem, salvo que el sentido es contrario.
- movimiento_motor_2[0]: Guarda el puerto correspondiente al sentido de giro desde el plano del suelo hasta llegar a 90° respecto a este.
- movimiento_motor_2[1] : Idem, pero guarda el puerto correspondiente al sentido contrario. (a 90° respecto del suelo, hacia el plano del suelo).

En este diagrama, se han puesto las funciones principales. Cada función, se compone de distintas variables que sirven de soporte al programa en general. Estas no se han graficado en el diagrama de la figura 6.6. Estas funciones, que sirven de soporte a cada función, se muestran a lo largo del presente capítulo.

El orden del trabajo sobre el software es primero realizar la función de autocalibración. La función de autocalibración, implica que debe llamar a la función mover_antena(n_motor,sentido_giro). En esta sección, se desarrollan ambas funciones. Una vez obtenida y realizada las pruebas sobre la función de autocalibración, se procede a realizar el sistema de control. Esto debe realizarse de esta manera, ya que si no estan calibrados los sentidos de giro del motor, el control no sabra de que forma actuar sobre ellos. Después se continua con la lectura de los programas Gpredict y Stellarium respectivamente. Luego de esto, se realiza el esquema de polling y el esquema de tiempos o planificación. En la jerga de programación, se denomina "programación por scheduler". Luego en la siguiente fase(fase 3), se desarrolla la teoria de coordenadas, y se implementa la función de transformación de coordenadas. La función que escribe en el display, se desarrolla en la parte final del presente capítulo.

6.5 Función de autocalibración

Esta función, es la primera función a desarrollar. Se supone que el lector posee conocimientos basicos de programación en C/C++, y como debe preparar el entorno para el desarrollo. El entorno elegido es visual Studio Code con el plugin de PlatformIO. En el apéndice se encuentra como instalarlo y empezar a utilizarlo.

Antes de realizar la programación, se realiza un archivo, denominado "pinout_ard_uno.h", el cual tendrá todas las definiciones de puertos, mostrada en la figura 6.2. El archivo contiene las sentencias mostradas en el código 1:

Donde las variables TIMERS_CLOCKS y DEBUG(resaltadas en la 1) son utilizados para realizar compilaciones condicionales, mientras se desarrolla. Las compilaciones condicionales se explican en el apéndice B .

Una vez definido los puertos, para realizar esta función de autocalibración, debe conocer cual es el sentido de giro, al poner en estado alto, el pin 10 y 9, o 6 y 5, y guardar este resultado. Para realizar esto, cada motor que mueve la antena, tiene adosado un potenciómetro. El giro de este potenciómetro, nos indica la medida angular. Un potenciómetro, consta de tres pines, donde uno esta conectado a la fuente de tension, otro a tierra, y el del medio se dirige al pin A0 o A1 del microcontrolador. Al girar este, cambia su resistencia entre el punto medio y tierra, cambiando al tensión, y esta tensión es la que se mide desde el microcontrolador. A continuación, se deja la imagen de un potenciómetro, y sus conexiones:

```

1  /*** PINES PARA EL CONTROL DE LOS MOTORES ***/
2
3 // motor de cenit
4 #define MOTOR_1_S1 5
5 #define MOTOR_1_S2 6
6 // motor de azimuth
7 #define MOTOR_2_S1 9
8 #define MOTOR_2_S2 10
9
10 /* PINES ETHERNET */
11 #define PINSS 4
12 #define PINRESET 3
13
14 /* PINES ENCODERS*/
15 #define PINENCODERAZ A0 //MEDIDA DE AZIMUTH
16 #define PINENCODERH A1 //MEDIDA DE ALTURA
17
18 // flags para utilizar depuracion
19 #define TIMER_CLOCKS 0 // para depurar las aplicaciones sin timers establecidos
20 #define DEBUG 1 // depuracion de aplicaciones usando puerto serie .
21

```

codigo 1: definición de los puertos del microcontrolador. El nombre del archivo es "pinout_ard_uno.h".

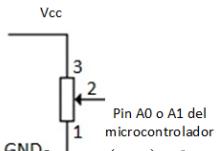


Figura 6.7: Vista de un potenciómetro, y sus respectivas conexiones de tensión, y con el microcontrolador.

Como se observa en la figura(6.7), la salida de tensión(pin 2 de la figura), con respecto a GND, cambia, cuando se gira el eje del potenciómetro. Esta tensión, es la que se mide el conversor analógico digital del microcontrolador atMEGA328p. El conversor analógico-digital se explica en el apéndice B

Para poder realizar la autocalibración, se debe definir, si la tensión en un extremo de la posición de la antena, es máxima, y en el otro extremo es mínima. Con esto, el software se dará cuenta, en que sentido se esta moviendo la antena(esta definición debe realizarse en ambos ejes de la antena). El software se dará cuenta, ya que podra revisar si la tensión de entrada, esta aumentando o disminuyendo, y en base a esto, se asignan los sentidos de giro dentro de las variables movimiento_motor_1 y movimiento_motor_2.

Los sentidos de giro de máxima y mínima tensión, se deben definir sobre los ejes de la antena. Estos ejes se mueven como muestra la siguiente figura:

La antena, solo tiene movilidad ESTE- OESTE, apuntando hacia el sur. Por este motivo, definimos que la mínima tensión esta dada en el oeste, y la máxima tension en el este,donde se define 0° en el OESTE, y 180° en el ESTE, en sentido antihorario. En sentido del eje horizontal, definimos la mínima tensión, a 90° respecto al piso, y máxima tensión cuando la antena se encuentre a 0° del plano del suelo. Utilizamos esta convención para realizar la medida sobre el ángulo de altura. Además, debe definirse a que eje del movimiento de la antena, corresponde a cada puerto. Siguiendo el código 1, se observa que se define el puerto A0 para el ángulo azimutal y el puerto A1, para el ángulo de altura. Las definiciones de tensión y puertos, se resumen en la siguiente tabla:

Por lo expuesto en los parrafos anteriores, la función de autocalibración debe realizar los siguientes pasos:

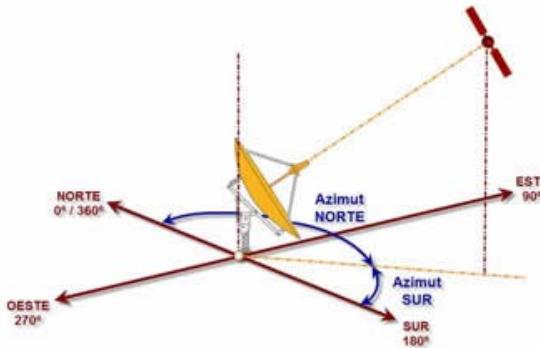


Figura 6.8: Movimientos que puede realizar la antena usando sus dos motores.

Eje	Puerto	punto de máxima tensión	punto de mínima tensión
azimuth	A0	Este(180°)	Oeste(0°)
altura	A1	paralelo al suelo	90° respecto al suelo

Tabla 6.1: Definición de sistema de coordenadas para la antena

1. Poner en alto, los pines 5 y 10 respectivamente, y los pines 6 y 9 en bajo
2. Tomar el dato de los puertos analógicos-digitales de los pines A0 y A1.
3. Guardar este dato, y compararlo con el próximo. Si es mayor, gira en un sentido u otro. Esto debe realizarse para ambos ejes.
4. Esperar que llegue al final de su recorrido la antena(ambos ejes). Luego guarda el valor final. Se da cuenta, que llega a su recorrido final, si las últimas tres muestras son identicas
5. invertir los pines del paso 1, y realizar los pasos 2 y 3 respectivamente, pero ahora, guarda el segundo valor final. Luego se comparan ambos, y el programa, puede saber en que sentido giraron los motores. Con estos valores, se calcula la resolución angular del apuntador.

Esta función, se debe realizar al iniciar el equipo.

Antes, de realizar la función de autocalibración, se ha realizado la función, que lea desde los potenciómetros la posición actual, y las guarde en la variable azimuth y altura, respectivamente. Para estas funciones, se han creado dos archivos: uno denominado "lectura_encoders.cpp" y otro denominado "lectura_encoders.p". Este último, tiene los prototipos de las funciones compartidas, que deben ser accedidas desde otra parte del código. El archivo "lectura_encoders.cpp", tiene las funciones propias de su funcionamiento, y el comportamiento de las funciones definidas en "lectura_encoders.h".

La única función definida en este archivo es `leer_encoders()` es devolver el valor leido por el conversor analógico digital, cuando se está autocalibrando. Esta función se muestra en el código 2. Si no se está autocalibrando, deberá devolver las coordenadas correspondientes. En este caso, esta parte, se va programar en la próxima sección.

Una vez realizado el proceso de lectura de los encoders por parte del microcontrolador, ahora resta la función de autocalibración. Esta se divide en dos archivos, uno denominado "control_motores.h" y "control_motores.cpp".

Esta función de autocalibración, debe ser capaz de mover los motores, por ende, se crea la función "`mover_antena(char n_motor,sentido)`", donde n_motor es el motor de azimuth si n_motor

```

1 #include "Arduino.h"
2 #include "../pinout_ard_uno.h"
3 // variables tipo buffer
4 int azimuth ;
5 int altura ;
6 void leer_antena() ;

7
8 extern enum _state_antena
9 {
10     AUTOCAL ,
11     NO_AUTOCAL,
12 } antena ;

13 void leer_encoders()
14 {
15     azimuth = analogRead(PINENCODERAZ) ;
16     altura = analogRead(PINENCODERH) ;
17     if (antena==AUTOCAL)
18     {
19         return;
20     }
21     //transformacion de coordenada--> retornar la transformación
22 }
```

codigo 2: archivo lectura_encoders.cpp.

es uno, o el motor de altura si toma el valor 2. Esta variable define el número de motor. Los sentidos, se definen como 0,1, o 2 respectivamente, siendo:

- sentido = 0 : Apagar motor
- sentido = 1 : Encender el puerto MOTOR_1_S1(ver código 1, idem para las otras variables mencionadas) en alto y MOTOR_1_S2 en estado bajo si el número de motor es 1, si el número de motor es 2,se definen en alto el puerto MOTOR_2_S1 y en bajo el puerto MOTOR_2_S2
- sentido = 2: define los puertos en forma inversa al sentido que los define sentido = 1

Una vez, se ha definido la función **mover_antena(char n_motor,char sentido)**, se pasa a realizar la función de autocalibración. El código de la función de mover_antena(char n_motor,n_antena), se observa en el apéndice x1

La función de autocalibración, utiliza las funciones leer_encoders y mover_antena, y funciones auxiliares. Estas funciones son:

- **assign_value_autocal()**
- **mover_antena(char n_motor, char sentido)**
- **function_compare_autocalibracion()**
- **assignar_sentidos_motores()**

Además, utiliza variables auxiliares. Estas se denominan:

- ult4ad[4]: guarda los ultimos cuatro valores leidos del encoder. En ult4ad[0] y ult4ad[1] guarda los ultimos dos valores del angulo de azimuth, y en ult4ad[2] y ult4ad[3] guarda los dos últimos valores del angulo de altura.
- estado_autocalibracion[2]: guarda el sentido(1,2, o 0) del motor en el estado actual. El valor de estado_autocalibracion[0] corresponde al motor de azimuth, y el otro al valor de sentido del motor de altura
- calibracion_encoders[4]: guarda los valores maximos y minimos leidos por los encoders. calibracion_encoders[0] y calibracion_encoders[1] corresponden a los valores del encoder correspondiente al ángulo de azimuth. Los otros dos, corresponden al valor del ángulo de altura.

No se mostrará todo código desarrollado para esta función de autocalibración. En su lugar, se da el diagrama de flujo del funcionamiento de esta función en la figura 6.9. La función asignar_sentidos_motores() se encarga de guardar los sentidos dentro de las variables buffer_motor_asignacion_1 y motor_asignacion_2, según los criterios de la sección anterior.

La idea del algoritmo es la siguiente:

1. LLamar a la función assign_value_autocal(). Esta lee los encoders, y pone el sentido = 1 en ambos motores.
2. Luego espera un segundo. signa los valores correspondientes dentro de las variables. Luego le asigna el sentido 1 a cada motor. Además, asigna los valores de calibracion_encoders[0] y calibracion_encoders[1] en uno.
3. LLama a la función de comparación function_compare_autocalibracion(). Esta función, es la encargada de realizar las comparaciones y cambiar el estado de calibracion_encoders[0] o calibracion_encoders[1], cuando cambia por primera vez, se cambia a 2, y la tercera vez, cambia a cero. La comparación debe realizarse cada un segundo, para que la antena, tenga el tiempo suficiente de moverse.
4. assignar_sentidos_motores(): Toma los valores de calibracion_encoders, y los compara. En base a eso, asigna los sentidos dentro de las variables asignacion_motor_1 y asignacion_motor_2 respectivamente.

La función de comparación, en primer lugar lee los valores actuales de los encoders, y luego los compara con los anteriores para saber si el valor actual es igual al anterior. Luego, borra los valores actuales, y los pasa como valores antiguos, y los compara con los que siguen. Los códigos de programación, se encuentran en el apéndice.

Luego, la función de autocalibración, viene definida de la siguiente manera:

6.5.1. resultados de la función de autocalibración

Para revisar los resultados de la función de autocalibración, se ha definido una bandera de compilación denominada DEBUG donde, se van a imprimir los resultados por puerto serie.

El prototipo armado es el de la figura 6.4, y las conexiones del potenciómetro y los diodos leds, se encuentran debajo de la imagen.

Para realizar la prueba del programa, se pone el potenciómetro en una posición inicial, y luego se gira el potenciómetro. Hay tres posiciones iniciales, que empieza en uno de los dos extremos, o que empieza en un punto medio. Si empiezan en un extremo, solo se debería girar una vez hacia el

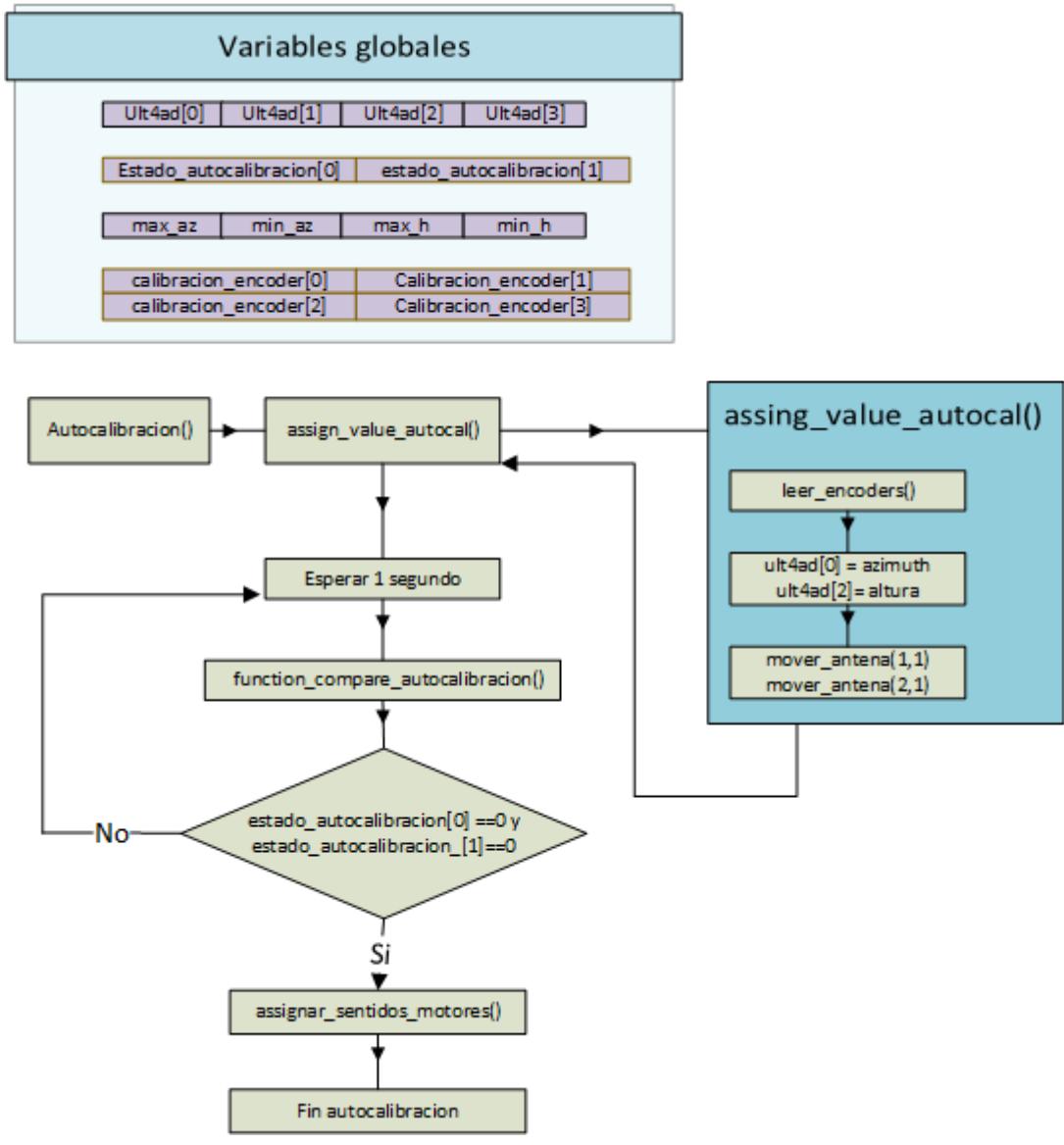


Figura 6.9: Diagrama de flujo de la función de autocalibración.

otro extremo. Si empieza en un punto medio, se debe girar dos veces, una para un extremo, y luego hacia el otro extremo. Para realizar esta prueba, dentro del código 1 se pone la bandera DEBUG en 1, y se utiliza el código mostrado en 3. Este código arroja los resultados por el puerto serie. Estos resultados, se resumen en la siguiente tabla, donde el sentido viene dado, por los potenciómetros vistos de frente.

Al analizar la tabla anterior, se observa, que la función de autocalibración, responde correctamente para ambos motores. Por ejemplo, la primer fila del motor 1, se empieza del punto medio, con el sentido siendo 1 (ver sección anterior), y se gira el potenciómetro hacia la derecha, y guarda el valor leído del conversor A/D:1023 en este caso. Cambia de sentido, con el valor de sentido 2, y se gira el potenciómetro hacia el otro extremo. En este caso, el valor leído es 1. Luego, el sentido Oeste - Este es el sentido con valor 1, ya que el primer giro, aumentó la tensión. Luego al invertir los puertos, se gira hacia el otro lado, y se observa, que la tensión disminuye. Esto indica que el sentido de giro es en el sentido este - oeste. Luego el valor guardado del sentido Oeste - Este debe

```

1 void autocalibracion()
2 {
3     assign_value_autocal() ;
4     delay(1000) ;
5     function_compare_autocalibracion() ;
6     while (estado_autocalibracion[0] !=0 || estado_autocalibracion[1] != 0)
7     {
8         delay(1000) ;
9         function_compare_autocalibracion() ;
10    }
11 //ASIGNACIÓN DE MOTORES
12 assignar_sentidos_motores() ;
13 #if DEBUG==1
14 // depuración por puerto serie .
15 Serial.print("calibracion encoders az: ");
16 Serial.print(calibracion_encoders[0]); Serial.print(" ");
17 Serial.println(calibracion_encoders[1]) ;
18 Serial.print("calibracion encoders h: ") ;
19 Serial.print(calibracion_encoders[2]); Serial.print(" ");
20 Serial.println(calibracion_encoders[3]) ;
21 Serial.print("movimiento_motor_1: ") ;
22 Serial.print(movimiento_motor_1[0],DEC) ; Serial.print(" ");
23 Serial.println(movimiento_motor_1[1],DEC) ;
24 Serial.print("movimiento_motor_2: ") ;
25 Serial.print(movimiento_motor_2[0],DEC) ;
26 Serial.print(" ");
27 Serial.print(movimiento_motor_2[1],DEC) ;
28
29 #endif
30 }
```

codigo 3: Código de la función de autocalibración. Esta definido en el archivo control_motores.cpp”

ser 1, y el sentido contrario, debe ser 2. Se analiza de la misma manera los restantes, y se observa que el comportamiento es el esperado.

Cabe destacar, que el giro de los potenciómetros se ha realizado de forma manual. Los potenciómetros que van a usarse, están adosados al eje de la antena, y estos realizan el movimiento, a medida que gira el/los motores.

6.6 Control de la posición

El control de la posición, consta en leer la posición enviada desde el software Gpredict o Stellarium, y mover la antena hacia esa posición. Además, si no recibe, ninguna posición de estos programas, el software, debe ser capaz de regresar a la posición de equilibrio, o denominada cenit. Esta posición es equivalente a 90º en posición azimutal y 90º respecto del suelo.

Recordando, que cada motor, tiene adosado un potenciómetro, que es utilizado como encoder, se procedió a medir el potenciómetro. Este potenciómetro se encuentra adosado a cada motor, y deben medirse, para conocer la variación de la tensión en función del ángulo.

Para realizar esta medición, se ha realizado un script en el lenguaje python, que se conecta al microcontrolador, y un programa sobre el microcontrolador. El microcontrolador, envía los datos leídos del potenciómetro, y el script, los guarda en un archivo de texto. Ambos programas se encuentran en el anexo del presente capítulo.

motor 1						
posición inicial	primer giro	segundo giro	calibracion_encoders		asignacion_motor_1	
			índice 0	índice 1	índice 0	índice 1
punto medio	giro derecha	giro izquierda	1023	1	1	2
punto medio	giro izquierda	giro derecha	0	1018	2	1
extremo izquierdo	giro derecha	x	0	1007	2	1
extremo derecho	giro izquierda	x	1023	1	1	2
motor 2						
posición inicial	primer giro	segundo giro	calibracion_encoders		asignacion_motor_2	
			índice 2	índice 3	índice 0	índice 1
punto medio	giro derecha	giro izquierda	898	1	2	1
punto medio	giro izquierda	giro derecha	17	1023	1	2
extremo izquierdo	giro derecha	x	0	1023	1	2
extremo derecho	giro izquierda	x	1023	1	2	1

Tabla 6.2: Resultados de la función de autocalibración.

Una vez creados ambos programas, sobre el microcontrolador, y sobre la pc(script en python), se conectó el punto medio del potenciómetro al microcontrolador, y se realizó el giro del motor con una fuente de laboratorio. Ambos motores, se girán de tal manera que la antena, tenga su recorrido completo en ambos ejes. La tensión de la fuente fue de 24 V. Estos datos, se registrarán cada 1 milisecondo. Los resultados fueron los siguientes:

Se observa en el ángulo de altura(ver figura 6.10a), que el potenciómetro responde de forma lineal en función del tiempo, mientras que el ángulo de azimuth, utiliza una serie de pulsos para medir la posición angular(ver figura 6.10b). Este último, será reemplazado por un potenciómetro comercial en la fase 4. Esté, será de tipo lineal.

De la función de autocalibración, obtenemos los valores máximos y mínimos del conversor A/D sobre el eje de altura, e idem cuando se adicione el potenciómetro en el eje de azimuth. A partir de estos puntos, obtenemos la relación entre la tensión leída y los angulos. Es decir, obtenemos una relación lineal entre la tensión del potenciómetro y el ángulo de la antena. En el microcontrolador, se usan los valores leídos del conversor analógico digital, para la programación.

Dado que el potenciómetro es lineal, puede construirse la ecuación de una recta, a partir de dos puntos. Estos dos puntos, son los valores máximos y mínimos del conversor A/D, y los ángulos máximos y mínimos de la antena. La convención utilizada para los ángulos se encuentra en la tabla 6.1.

Para el ángulo de azimuth, se tienen los puntos $p_1 = (0^\circ, \min\{calibracion_encoders[0], calibracion_encoders[1]\})$ y $p_2 = (180^\circ, \max\{calibracion_encoders[0], calibracion_encoders[1]\})$. Al tener dos puntos, se puede armar la ecuación de una recta, en términos de $\theta_{az} = f(AD_0)$, siendo θ_{az} el ángulo de azimuth. La ecuación de la recta, viene dada por:

$$\theta_{az} = \frac{\Delta\theta_{az}}{\Delta A_d} (AD_0 - \min\{calibracion_encoders[0], calibracion_encoders[1]\}) \quad (6.6.1)$$

Siendo:

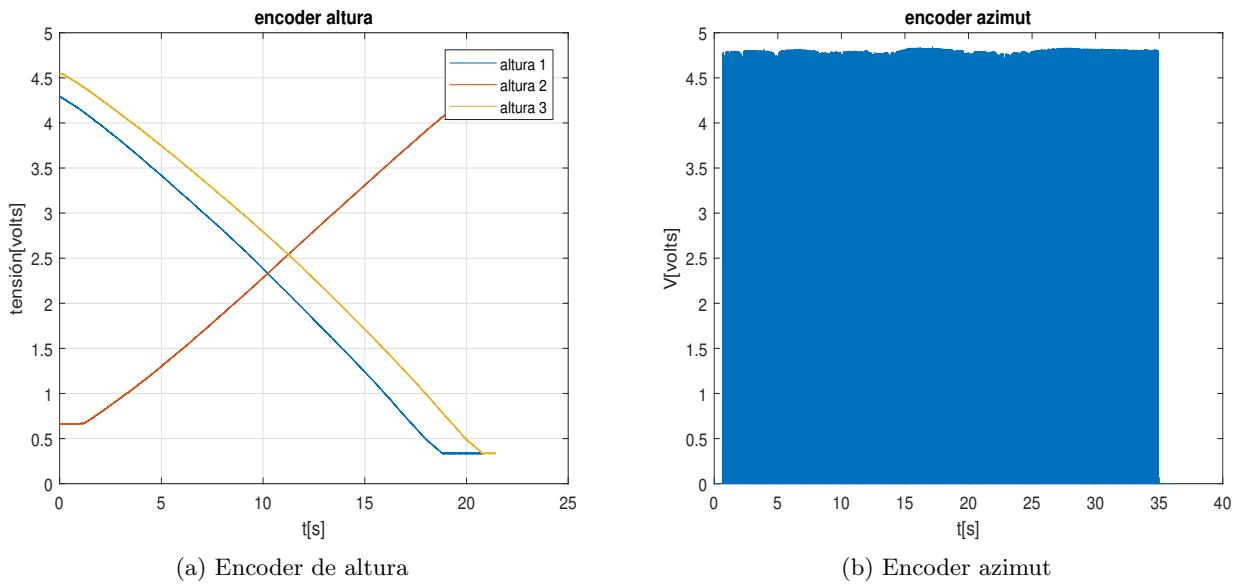


Figura 6.10: Medidas realizadas sobre los encoders de la antena en función del tiempo.

AD_0 : Valor leido por el conversor Analogico digital del puerto 0

$$\Delta\theta_{az} = 180^\circ - 0^\circ$$

$$y_1 = \max\{calibracion_encoders[0], calibracion_encoders[1]\}$$

$$y_2 = \min\{calibracion_encoders[0], calibracion_encoders[1]\}$$

$$\Delta A_d = y_2 - y_1$$

Para el eje de altura, se realiza un procedimiento similar al anterior, se obtiene la ecuación de la altura en función del valor leído de tensión. Denominando θ_h al ángulo de altura, se obtiene la siguiente ecuación

$$\theta_h = \frac{\Delta\theta_h}{\Delta A_d} (AD_1 - \max\{\text{calibracion_encoders}[0], \text{calibracion_encoders}[1]\}) \quad (6.6.2)$$

Siendo:

AD_1 : Valor leido por el conversor Analogico digital del puerto 1

$$\Delta\theta_h = 90^\circ - 0^\circ$$

$$y_1 = \min\{calibracion_encoders[0], calibracion_encoders[1]\}$$

$$y_2 = \max\{calibracion_encoders[0], calibracion_encoders[1]\}$$

$$\Delta A_d = y_2 - y_1$$

Las ecuaciones mostradas anteriormente, se implementan dentro de la función `leer_encoders`.

Además de esto, se debe calcular la resolución angular del dispositivo para cada eje. Esta viene dada por las pendientes de las rectas anteriores, multiplicadas por 1, ya que es el mínimo valor de cuenta que posee el conversor analógico digital. Las resoluciones, son entonces:

$$\begin{aligned} res_h &= \frac{\Delta\theta_h}{\Delta A_d} 1 = \frac{\Delta\theta_h}{\Delta A_d} \\ res_{az} &= \frac{\Delta\theta_h}{\Delta A_d} 1 = \frac{\Delta\theta_h}{\Delta A_d} \end{aligned} \quad (6.6.3)$$

El esquema de control se muestra en la figura 6.11.

En esta parte, se programa el recuadro de la parte control_motores(ref1,ref2)" de la figura 6.11. El control, mira una señal de error, donde la señal de error en la figura viene dada por $e[k]$. Hay dos señales de error, estas las denominados e_1 y e_2 . Estas señales de error vienen dadas por:

$$\begin{aligned} e_1[k] &= ref_1 - \text{azimut} \\ e_2[k] &= ref_2 - \text{altura} \end{aligned} \quad (6.6.4)$$

En base a esta señal de error, que es entrada para el bloque control ON/OFF", decide hacia donde debe moverse el motor. El sistema se para siempre que la señal de error sea menor que la resolución de cada eje. Este sistema en proximas versiones, se va a cambiar el bloque controlONOFF" por un controlador denominado PID.

El control ON/OFF, actua de según la siguiente tabla, donde para seleccionar el sentido, se basa en las variables movimiento_motor_1 y movimiento_motor_2, que se obtienen de la función de autocalibración.

eje	error	sentido de movimiento	valor variable sentido
Azimut	$e < res_{az}$	este → oeste	movimiento_motor_1[1]
	$e > res_{az}$	oeste → este	movimiento_motor_1[0]
	$ e < res_{az}$	motor parado	0
altura	$e < res_h$	$90^\circ \rightarrow$ plano del suelo	movimiento_motor_2[1]
	$e > res_h$	plano del suelo $\rightarrow 90^\circ$	movimiento_motor_2[0]
	$ e < res_h$	motor parado	0

El valor para que se pueda realizar el control correctamente, debe medirse el ruido sobre el sistema(más específicamente, la potencia del ruido),y luego proponer un valor mayor a él para realizar el control, en lugar de la resolución. Se va a probar, si la antena, responde con este valor de resolución, o debe reajustarse. Este procedimiento se realiza al final del presente texto,aclarando dicha situación. El código realizado para el control, se encuentra en el apéndice del presente texto.

6.6.1. Resultados de la función de autocalibración y control

La función de control, requiere de la definición de los datos brindados por la función de autocalibración. Para la función de autocalibración, en la tabla 6.3 definimos los sentidos, siendo los resultados de las variables los que se encuentran en tabla 6.2. Luego, en base a estas variables,se observan cuales puertos son los correspondientes en base a los diodos led referidos en la figura 6.4. La resolución, se definió en la sección anterior. El valor de referencia utilizado para realizar las pruebas es el denominado cenit: 90° en altura y 90° en azimut.

En la tabla 6.3, se observa que la función de control responde tal cual lo esperado, ya que se observa que al cambiar los puntos iniciales, para el mismo error, se invierte la dirección de los puertos. Esto es así, ya que la función de autocalibración, guarda el sentido de movimiento de la antena, y la función de control, los orienta en el sentido correcto.

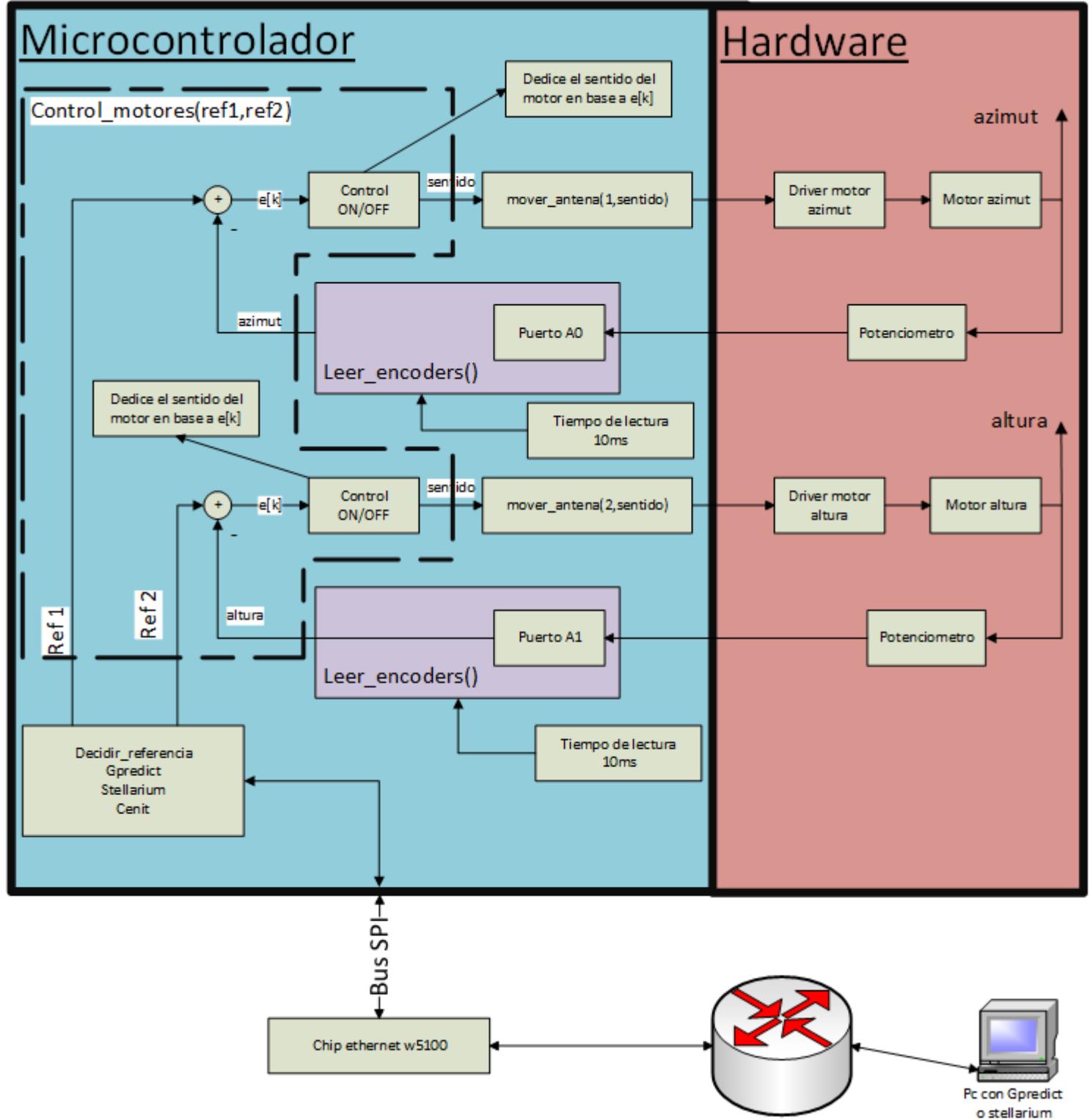


Figura 6.11: Sistema de control microcontrolado implementado en este trabajo.

Motor 1: Motor de azimut - Referencia 90°								
autocalibración			control					
Posición inicial	Primer Giro	Segundo giro	$e > res_{az}$		$e < -res_{az}$		$ e \leq res_{az}$	
			puerto on	puerto off	puerto on	puerto off	puerto off	puerto off
punto medio	giro izquierda	giro derecha	MOTOR_1_S2	MOTOR_1_S1	MOTOR_1_S1	MOTOR_1_S2	MOTOR_1_S1	MOTOR_1_S2
punto medio	giro derecha	giro izquierda	MOTOR_1_S1	MOTOR_1_S2	MOTOR_1_S2	MOTOR_1_S1	MOTOR_1_S1	MOTOR_1_S2
extremo izquierdo	giro derecha	x	MOTOR_1_S2	MOTOR_1_S1	MOTOR_1_S1	MOTOR_1_S2	MOTOR_1_S1	MOTOR_1_S2
extremo derecho	giro izquierda	x	MOTOR_1_S1	MOTOR_1_S2	MOTOR_1_S2	MOTOR_1_S1	MOTOR_1_S1	MOTOR_1_S1
Motor 2: Motor de altura - Referencia 90°								
autocalibración			control					
Posición inicial	Primer Giro	Segundo giro	$e > res_h$		$e < -res_h$ ¹		$ e \leq res_h$	
			puerto on	puerto off	puerto on	puerto off	puerto on	puerto off
punto medio	giro izquierda	giro derecha	MOTOR_2_S1	MOTOR_2_S2	x	x	MOTOR_2_S1	MOTOR_2_S2
punto medio	giro derecha	giro izquierda	MOTOR_2_S2	MOTOR_2_S1	x	x	LOW	LOW
extremo izquierdo	giro derecha	x	MOTOR_2_S1	MOTOR_2_S2	x	x	MOTOR_2_S1	MOTOR_2_S2
extremo derecho	giro izquierda	x	MOTOR_2_S2	MOTOR_2_S1	x	x	MOTOR_2_S1	MOTOR_2_S2

Los valores de los puertos en on y off,están definidas en el archivo "pinout_ard_uno.h", cuyo código se muestra en el código 1.

¹ En el caso del motor de altura, al ser la referencia el máximo valor angular, el error, no puede darse que $e < -res_h$

Tabla 6.3: Resultados de la función de control en conjunto con la función de autocalibración.

6.7 Programación de Scheduler o planificación

La programación por scheduler(o planificación por su traducción al español), es la base del sistema operativo en tiempo real(ver FreeRtos). El sistema consiste en la ejecución de tareas, que se denominan aplicaciones. Estas tareas, se ejecutan cada cierto tiempo, donde el tiempo lo define el programador. Esto puede definir tareas de mayor o menor prioridad,y permite la ejecución de tareas de forma asincrónica – sincronica. La ejecución sincronica, es la ejecución a tiempo controlado de la aplicación, mientras que la forma asincrónica, puede ejecutarse en cualquier parte del software. En este trabajo, se realiza, la programación de una base de tiempo para controlar la ejecución de tareas, y las funciones necesarias para controlar los relojes. Finalmente se probaron estos relojes, con el uso de un osciloscopio, y midiendo los tiempos.

6.7.1. Funcionamiento base de tiempo para scheduler

Este se basa en el concepto de interrupción.Las interrupciones se explican en el apendice B. Se realiza una interrupción por timer. Una interrupción por timer es una interrupción que ocurre con cierta frecuencia,definida por el programador. En el caso del Atmel ATMEGA 328P, posee tres timer, en el presente trabajo, se utiliza el timer2. El diagrama de este se muestra en la figura 6.12.

Antes de empezar, se debe definir una frecuencia de funcionamiento, para el control del timer. Esta frecuencia se selecciona del clock principal, y se hace pasar por un divisor de frecuencia, llamado preescaler. Este preescaler, solo realiza divisiones de frecuencia en potencias de dos. Este preescaler se configura de un registro llamado “TCCR2B”, que además configura otros parámetros. Una vez configurada la frecuencia de las interrupciones, contamos cuantas interrupciones ocurren, y se tiene el tiempo para cada tarea.

Este timer tiene 4 modos de funcionamiento. Los cuales son:

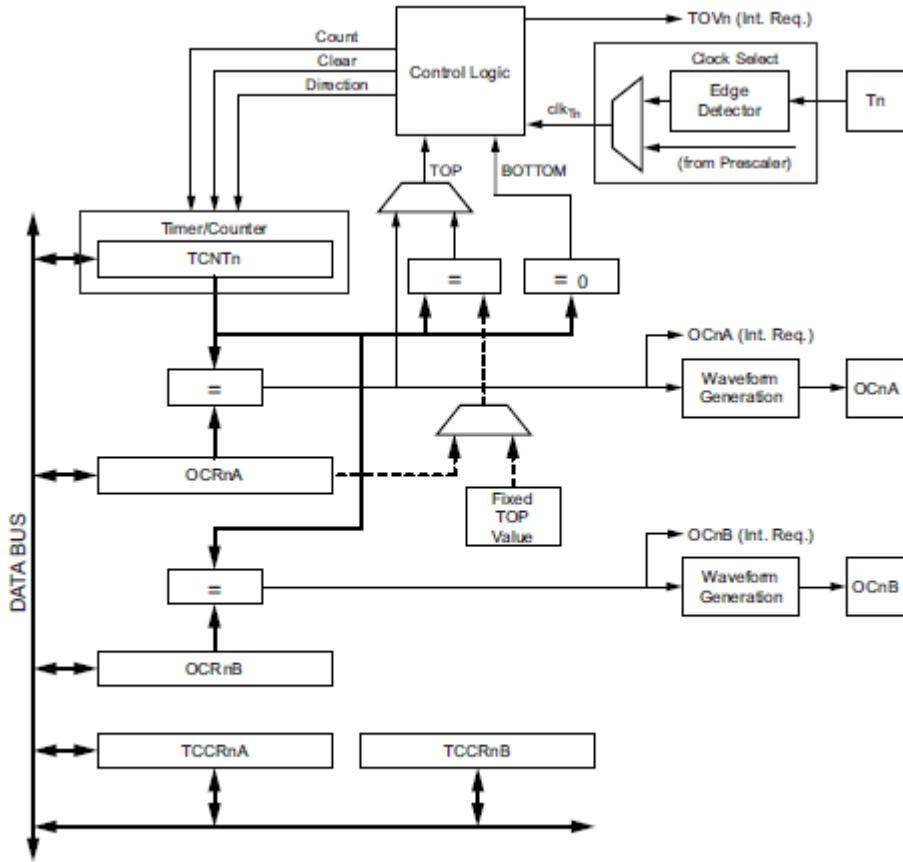


Figura 6.12: Diagrama del timer 2

- NORMAL MODE
- Clear Timer on Compare Match (CTC) Mode
- Fast PWM Mode
- Phase Correct PWM Mode

Sin entrar en los detalles de cada uno de ellos, explicados en la hoja de datos del microcontrolador Atmel Atmega328P, los últimos dos modos, se usan para realizar un PWM, y el primero, cuenta hasta una cantidad fija, definida en 255. El modo elegido para este propósito fue el CTC(explicado en la siguiente sección), que es el que mejor se adapta a los requerimientos de una base de tiempo controlada.

6.7.2. Clear Timer on Compare Match (CTC) Mode

Para configurar este modo, se deben configurar algunos registros, mostrados en la figura 6.12. En ella, se ve que los registros, tienen la terminación "nx". Esta terminación corresponde al número de timer y al canal. Así, n puede tomar el valor 0,1 o 2, y x la letra A o B. Por ejemplo el registro OCR2A, corresponde al timer 2, y al canal A del mismo.

El registro TCNT2, se incrementa de a uno, con la frecuencia de preescaler seleccionada en el registro TCCR2B. Cuando el valor de TCNT2 coincide con el valor cargado en el registro OCR2A,

se lanza una interrupción por timer. El siguiente esquema aclara lo anterior.

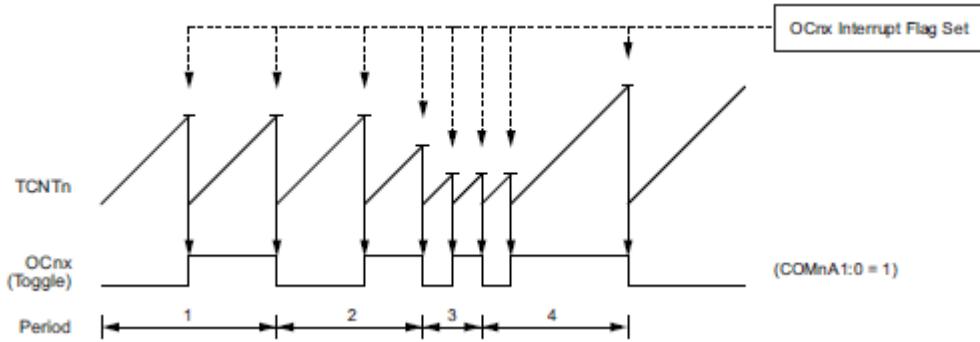


Figura 6.13: Diagrama de tiempos del modo CTC. Extraido de la hoja de datos del microcontrolador

En el se ve, que la interrupción se ejecuta cuando TCNT2 alcanza a OCR2A. Para el calculo de la frecuencia, usamos la siguiente formula, que viene dada por el fabricante del dispositivo

$$f_{OCnx} = \frac{f_{clk_I/O}}{2N(1 + OCRnx)} \quad (6.7.1)$$

donde

$$f_{OCnx} : \text{frecuencia de el modo CTC.} \quad (6.7.2)$$

$$N : \text{valor del preescaler,son algunas potencias de dos.} \quad (6.7.3)$$

$$f_{clk_I/O} : \text{frecuencia del reloj utilizado por el microcontrolador.} \quad (6.7.4)$$

Esta ecuación para calcular la frecuencia, es la frecuencia de una onda cuadrada, en el caso de la imagen 6.13, es el periodo enumerado con "1", en la imagen. Por este motivo, para conocer la frecuencia de la interrupción, se debe dividir por dos, al valor que brinda esta ecuación.

El valor de $f_{clk_I/O}$ es 16 Mhz, que es la frecuencia de reloj utilizada por la placa arduino UNO. En este caso, buscamos que el valor sea entero, o múltiplo de 10. Para realizar esto, se utilizaron varias pruebas. Se concluyeron los siguientes valores

- N = 32
- OCR2A = 49

Si reemplazamos en la ecuación 6.7.1 nos brinda un valor de 5Khz. Entonces, la interrupción ocurrira, con una frecuencia de 10Khz, o cada 100 μ s.

Para configurar el valor de N(denominado preescaler,dentro de la hoja de datos del microcontrolador), y la configuración CTC, dentro de cada timer, existen dos registros de configuración. Estos se llaman TCCR2A y TCCR2B, y Estos registros se deben configurar con los siguientes valores binarios:

- **TCCR2A = 0b00000010**
- **TCCR2B = 0b00000011**

El valor de cada bit, esta detallado en la hoja de datos del microcontrolador, y no se van a explicar en el presente documento. Al final de realizar todo este análisis, se construyó una función, que automatice esta tarea de configuración. La función tiene por nombre `Base_tiempo()`. El código se muestra a continuación.

```

1 void Base_tiempo()
2 {
3     SREG = (SREG & 0b01111111);
4     TCNT2 = 0 ;
5     TIMSK2 = TIMSK2 | 0b00000010 ;
6     TCCR2A = 0b00000010;
7     TCCR2B = 0b00000011; // 0.5 Mhz n= 32
8     OCR2A = 49;
9     SREG = (SREG & 0b01111111) | 0b10000000 ;
10 }
```

codigo 4: Función base de tiempo.

Esta función, además de configurar el modo CTC, el preescalador , y el registro OCR2A, configura las interrupciones. La configuración de las interrupciones puede obtenerse de la hoja de datos del microcontrolador.

6.7.3. Programación del Software scheduler

El sistema, se compone de una determinada cantidad de relojes, siendo esta cantidad variable por cada programador en la cantidad que se desee. En el caso de este trabajo, se hicieron 8 relojes, y se testearon. Primero deben crearse los relojes, que pueden o no estar configurados. Estos relojes, los definimos como una matriz de 8x4, donde cada fila es el numero de reloj, y cada columna es hora, minuto, segundo y milisegundo. Esta matriz se llama timer dentro del código. Luego, deben crearse las funciones para interactuar con ella, y además, debe crearse una bandera, que diga, que reloj o relojes se vencieron. Esto se logra con un vector de eventos, donde va guardando los relojes que se vencen. Se usa un vector, porque pueden vencerse mas de un reloj al mismo tiempo. Se lo denomina `FlagRepEvent[]` . Ademas, creamos un vector que nos indica que relojes están activos, y lo llamamos `timer_activo`. Todas estas variables, necesitan, comunicarse entre si, y la comunicación entre ellas se realiza usando funciones. En la figura 6.14 se muestra un diagrama del software programado.

6.7.3.1. Funciones

En primer lugar, se crean dos archivos, uno denominado "tiempo.cpp" "tiempo.h". Dentro de ambos archivos se encuentra la programación de las funciones. En el sistema, se crearon las siguientes funciones, las cuales luego se van a encapsular en una librería.

- `TimerStart(char n_reloj,char h, char m, char s , int ms)`
- `AnalizoTimer(char n_reloj, char_index)`
- `timerClose(char n_reloj)`
- `timerStop(char n_reloj)`

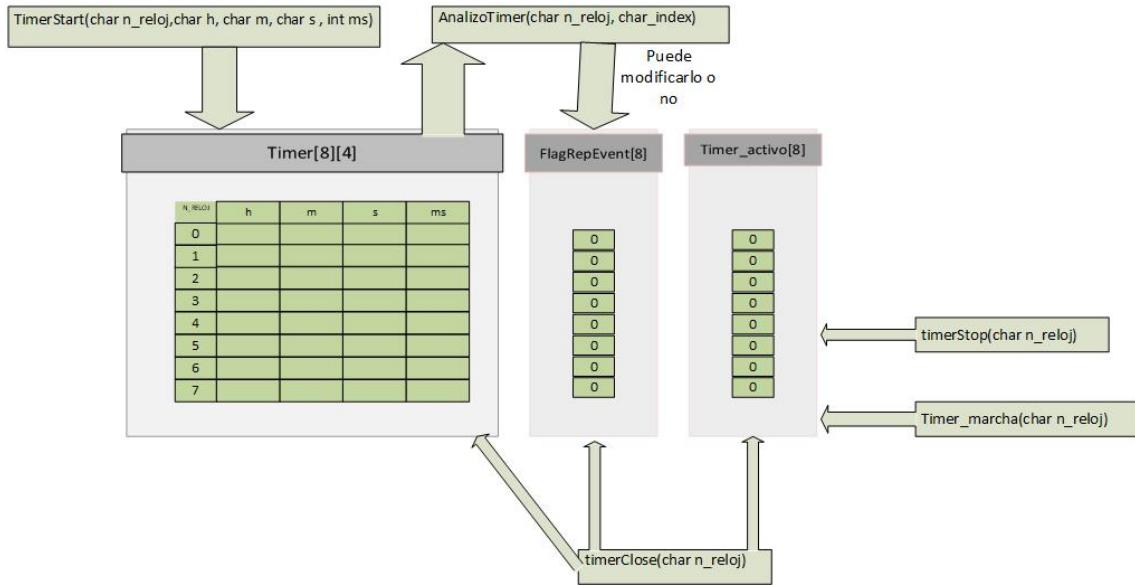


Figura 6.14: Diagrama de software para el software de scheduler

- **timer_marcha(char n_reloj)**

A continuación se da un breve resumen de lo que realiza cada una de ellas:

TimerStart: inicializa el numero de reloj, con la cantidad de tiempo que desea el programador que se ejecute la tarea. Esta función carga la variable `timer[n_reloj][]` con los valores de hora, minuto, segundo y milisegundo que desea el programador(es decir, carga la fila), y además carga el vector `timer_activo[]` con el valor de reloj.

AnalizoTimer: Esta función se ejecuta dentro de la interrupción. Su finalidad es descontar un milisegundo a los relojes que previamente se habían cargado. Esta lo que realiza, es solamente el decremento de un milisegundo en los relojes que esten activos en el flag `timer_activo`. La función decremente igual que lo haría un cronometro de bolsillo. Es decir, llega ms a cero, y mira los segundos, si los segundos no son cero, decrementa en uno el segundo, y pone en 999 la variable de ms en el numero de reloj correspondiente.

TimerClose: Esta función apaga el timer correspondiente, modificando el `timer_activo`(poniéndolo a cero) y poniendo todo el reloj a cero.

TimerStop: Deja de contar, es equivalente a parar un cronometro, para luego seguir contando. Esta función modifica `timer_activo` y lo pone en cero, pero no modifica el vector de relojes, los cuales siguen sin ser modificados. Para volver a arrancar, se debe utilizar la función `timer_marcha`, que vuelve a poner el funcionamiento el reloj que se había frenado.

6.7.3.2. Funcionamiento del software

Este software se basa en el concepto de interrupción. Se genera una interrupción cada 100us, y con una variable se cuentan 10 de ellas, y ahí ha transcurrido 1ms. Luego de 10 interrupciones, se llama a la función `AnalizoTimer()`, que descuenta en un milisegundo de los relojes que esten activos, y si luego de descontar, todo el reloj es cero,(es decir, la fila `n_reloj` es toda cero), activa el flag `FlagRepEvent` correspondiente, para ejecutar la acción correspondiente. Esta acción, es activada por el switch -case de `timerEvent()` y ejecuta la aplicación correspondiente.

El modo de utilizarlo es llamar a la función `Base_tiempo()`,dentro del inicio del programa,

luego con esto, se activan las interrupciones por timer. Acto seguido a esto, hay que definir el numero de reloj a activar con la función TimerStart, o cuando se desee llamar, para activar el reloj correspondiente. Luego, debe definirse la acción a ejecutar por ese reloj, la cual debe estar definida dentro de la función timerEvent.

Todo el código desarrollado para estas funciones, se encuentra en el anexo del presente capítulo.

6.7.4. Resultados del software de planificación

Luego de haber construido todo el software, se realizaron pruebas, prendiendo y apagando dos led, cada determinada cantidad de tiempo, y se ha utilizado un osciloscopio para ver la forma de onda.

Se configuro un reloj de 872ms, y se midio con un osciloscopio, ambos lanzados a tiempos distintos, es decir, existe un delay entre ambos. Los resultados se midieron en el osciloscopio, en una primera prueba. La segunda prueba, fue configurar dos relojes, y parar uno y relanzarlo, mientras el otro permanecia prendido. Los resultados sobre el osciloscopio se observan en las siguientes imagenes:



(a) Primera prueba realizada sobre el software de scheduler
 (b) Segunda prueba realizada sobre el software de scheduler

Figura 6.15: prueba sobre software de manejo de scheduler

Los códigos utilizados para medir esta función, no se encuentra dentro de este documento. Este, debe realizarse utilizando las funciones anteriores, y para la segunda prueba, poner un contador y parar el segundo reloj cuando llegue a un determinado valor, y seguir contando, y cuando alcance un segundo valor,volver a lanzar el mismo reloj.

Como observamos, en la primer prueba, fue generar una onda cuadrada, donde el estado de alto era de 872ms, y bajo de ese mismo tiempo. En la figura 6.15a, se observa la forma de onda en alto, y puede verse que con los cursores del osciloscopio, la medida, fue del mismo tiempo.

La segunda prueba, se observa en la imagen 6.15b, donde, se ha parado un reloj, durante un tiempo, y se ha vuelto a empezar, manteniendo una onda cuadrada sobre el otro puerto. El resultado fue satisfactorio, ya que se observa, en el osciloscopio, como se ha frenado un puerto, mientras el otro sigue funcionando,y al cabo de un tiempo, volvió a empezar.

6.8 Conexión del software con Gpredict y Stellarium

En esta sección, se muestran los componentes principales del software desarrollado para comunicarse con estos programas, y como obtener la posición hacia donde debe ir la antena. En la figura 6.11, se observa un recuadro, que dice "decidir referencia". En esta parte del desarrollo, trata, de ver la forma de decidir esa referencia. Las referencias provienen de tres fuentes posibles:

- Gpredict.
- Stellarium.
- Ninguna - Debe mantenerse en el cenit.

Para ello, dentro del programa principal, se crean dos variables: denominadas `ref1` y `ref2` respectivamente. Estas tendrán por defecto el valor del cenit (90° en ángulo de azimut y 90° en altura). Cuando existe una conexión, con alguno de estos programas, estos valores cambian automáticamente a aquellas coordenadas enviadas por algunos de los programas. Una vez finalizada la conexión con alguno de estos programas, estas vuelven a su valor inicial(posición del cenit).

Antes de empezar a realizar el software para comunicarnos con ambos programas, debe obtenerse la dirección IP, que en esta etapa, será asignada por DHCP. Luego, cuando se conecte a la red institucional, se le asignará una dirección ip fija, dada por el administrador de la red.

6.8.1. Conexión a la red mediante DHCP

En principio, el chip ethernet w5100, se conecta a la red, mediante un cable denominado `utp`. Este cable, se le adiciona un conector, denominado `RJ11`, en ambos extremos. Este cable, se debe conectar a un router o switch.

Una vez, armado el cable y conectado a la red, se debe obtener los parámetros de la red, mediante el uso de el servicio de DHCP(por el momento,en un futuro será fija la dirección IP del dispositivo). El entorno arduino, provee una librería para trabajar sobre el chip ethernet W5100. Esta librería, se denomina `.hethernet.h`, y viene por defecto en su entorno. Esta librería, se provee de varias funciones, para poder realizar la conexión con el chip W5100 y obtener la dirección IP mediante el protocolo DHCP.

Para comenzar, a utilizar esta librería, en primer lugar, se debe armar la conexión como muestra la figura 6.5. Una vez armado, se debe conectar el cable de red a la placa que posee el chip W5100. Una vez realizado, se debe realizar la petición DHCP mediante el uso de la librería.

En primera instancia, debe realizarse la configuración del puerto de slave select. Esto se realiza con la sentencia

```
Ethernet.init(PINSS)
```

Una vez definido el pin de chip select, se debe asociar una dirección al dispositivo, denominada "dirección mac". Esta dirección es un identificador que esta asociada a cada hardware que deseé conectarse a la red. En el caso del chip w5100, esta dirección, se la debe dar el programador, ya que se configura mediante software. La manera de definir una dirección es:

```
byte mac [] = {0x00, 0xCD, 0xEF, 0xEE, 0xAA, 0xBC};
```

Ahora, debe procederse a obtener la dirección IP por DHCP. Las sentencias para obtener la dirección ip se muestran en el código 5. La dirección IP,que se le asigna mediante el protocolo DHCP, se lee en el puerto serie, con este código.

Una vez, obtenidos todos los parámetros de la red, se deben configurar los programas, Gpredict, y Stellarium. La dirección IP que se obtuvo a partir de los pasos mencionados, fue la 192.168.0.150. Esta, es la que se va a utilizar para configurar los programas.

```

1 if (Ethernet.begin(mac) == 0)
2 {
3     Serial.print("obt_Ip") ;
4     Serial.print(F("Fallo DHCP"));
5 }
6 else {
7     Serial.print(Ethernet.localIP());
8 }
```

codigo 5: Obtención de la dirección IP usando el protocolo DHCP

6.8.2. Configuración de Gpredict - Programación de la comunicación

En primer lugar, debe realizarse la configuración del software Gpredict, siguiendo los pasos que se muestran en la sección 5.4.3, en la figura 5.10. Antes de realizar la configuración sobre el software, se debe recordar, que Gpredict, toma el cero en el eje azimutal, en el polo norte geográfico, y en sentido de las agujas del reloj, aumenta la cantidad de grados, hasta llegar a 360º. Por ende, la configuración de Gpredict en el eje azimutal, debe ser entre 90º (corresponde al este), y 270º (corresponde al oeste geográfico). Dicho esto, se configura el rotador como muestra la siguiente figura:

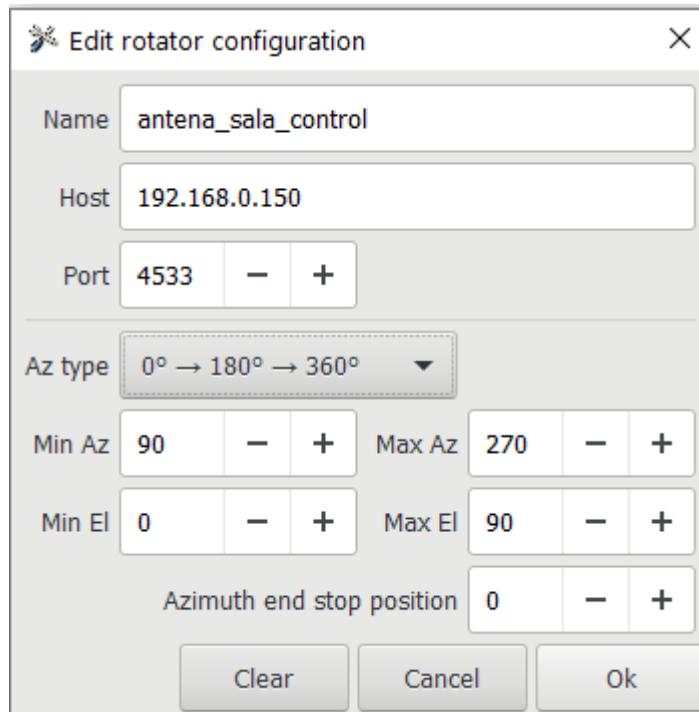


Figura 6.16: Imagen de la configuración del software Gpredict para la antena ubicada en sala de control del IAR.

6.8.2.1. Panel de control del rotador

Si se procede a abrir el rotador recién creado, se observará la siguiente imagen (ver sección 5.4.3):

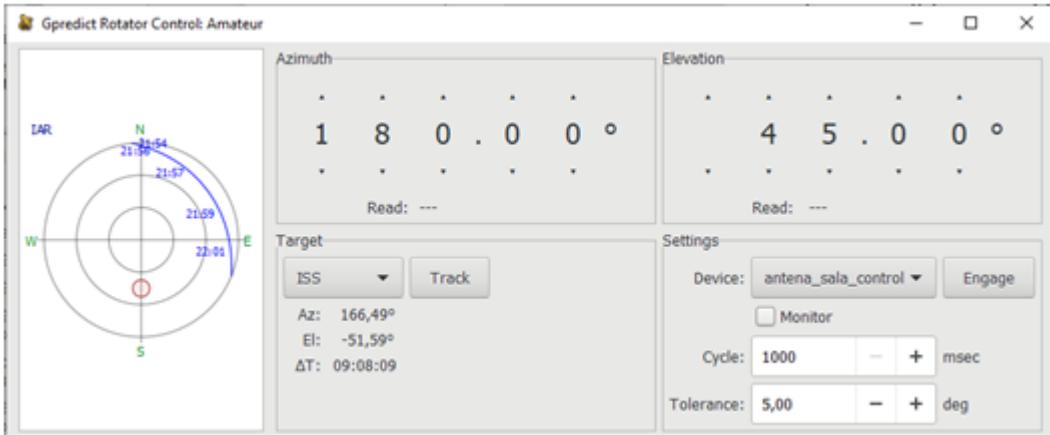


Figura 6.17: Panel de control del rotador en Gpredict

Observamos en la imagen, que se tienen los siguientes botones:

- azimuth: posee las coordenadas del rotador, que se envían mediante el protocolo TCP/IP. Si el rotador esta conectado, en la parte read, aparece la posición del rotador.
- elevación: idem que azimuth, salvo que en el eje de azimuth
- target: Se selecciona el satelite que se desea seguir. Oprimiendo el boton track, se aplica la posición de azimut y elevación del satelite a seguir, tanto a azimuth como a elevación
- settings: Se selecciona el rotador, y en tolerance, indica la tolerancia entre el valor leido y el valor enviado por el rotador. Cycle, es para uso interno del programa, y no debe modificarse. El boton engage es para conectarse al rotador mediante la red.

En la parte izquierda de la figura 5.12, se encuentra una gráfica en coordenadas polares. Los círculos concéntricos, indican el ángulo de altura, siendo el círculo exterior un ángulo de altura de 0° , y el centro de 90° . El ángulo desde el norte, en sentido de las agujas del reloj, indica el ángulo de azimuth. Este ángulo es el que esta representado en la ventana de azimuth y elevación, con un círculo rojo. Por ejemplo, en la imagen de la figura 5.12, se observa, 180° azimuth, y 45° elevación, y el círculo rojo, esta marcado apuntando al sur, y aproximadamente en la mitad del radio del centro, hasta el círculo mas grande. La imagen 6.18 aclara la explicación dada. En la figura, se observa que hay un recuadro a rayas rojas, esto indican, la visibilidad de la antena instalada en sala de control en el IAR.

De la figura, se observa, que en el ángulo de azimuth, se tiene un desfase de 270° , entre la convención de gpredict, y la convención utilizada en este trabajo. Por este motivo, a la coordenada recibida por el software Gpredict, debe restarsele esta cantidad. Si denominamos θ_{azgpr} a la coordenada recibida por Gpredict, y θ_{az} a la coordenada que debe apuntarse, según nuestra convención, se tiene la siguiente ecuación :

$$\theta_{az} = 270^\circ - \theta_{azgpr} \quad (6.8.1)$$

Por último, se observa en la sección target y settings, los botones engage y track. Su comportamiento, se describe con la siguiente tabla:

De la tabla anterior, se observa, que la comunicación mediante el protocolo TCP/IP, se debe oprimir el botón Engage. Si se oprime este botón, la coordenada enviada, será la que aparece en la

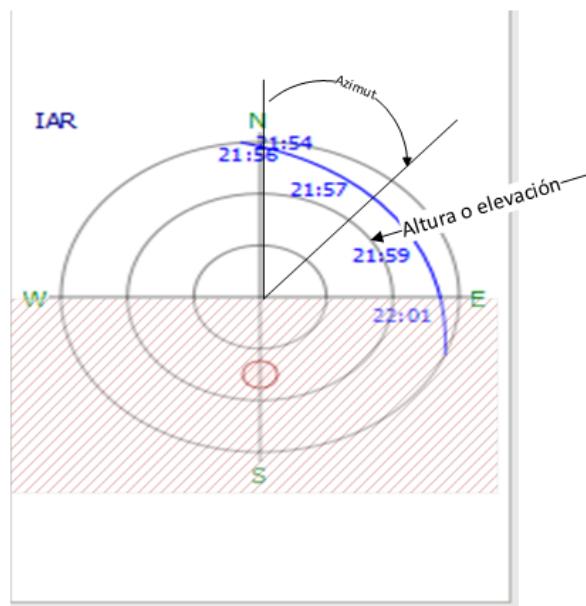


Figura 6.18: Diagrama de gráfico polar en software Gpredict.

track	engaje	Descripción.
off	off	No se envían comandos al rotador y tampoco se lee su posición
on	off	Los widget del controlador se actualizan con la posición destino pero no se envían comandos al rotador. La posición actual del rotador no se lee de nuevo. Si el satélite objetivo esta fuera del alcance, los widgets del controlador se establecerán en donde se espera que aparezca el satélite en el siguiente AOS
off	on	Los widgets del controlador no se actualizan con la posición de destino, pero los comandos son enviados al rotador y se lee la posición actual. Este modo se puede usar para controlar manualmente el rotador
on	on	Los widgets del controlador se actualizan con la posición destino y la posición se envia al rotador. La posición actual se envia desde el dispositivo rotador. Si el satélite objetivo esta fuera del rango, se intenta establecer el siguiente AOS

Tabla 6.4: comportamiento al oprimir los botones track y engage

parte del panel donde dice “azimuth y elevación”. Si se oprime track, y luego engage, la coordenada enviada por TCP/IP, será aquella en la cual, se encuentre el satélite seleccionado.

6.8.3. Programación del microcontrolador

Después de configurar el programa Gpredict, para que realice la comunicación con el dispositivo desarrollado en el presente trabajo, utilizando la librería “ethernet”, provista por el propio desarrollador de la placa de desarrollo, se debe crear un objeto, denominado servidor. La sentencia, para crear este objeto es la siguiente:

```

1 #define PORT_GPREDICT 4533
2 EthernetServer Gpredict(PORT_GPREDICT)

```

codigo 6: definición del objeto servidor dentro del entorno arduino

Donde se encuentra definido `#define PORT_GPREDICT 4533`, es el puerto utilizado para la comunicación mediante el protocolo TCP/IP.

Luego, dentro del código principal, se debe realizar una captura de estos datos para poder manipularlos. Estos datos vienen en la forma: comando Parametro1 parametro2 n o simplemente con el comando

n. La diferencia, entre uno y otro, es que el que viene sin los parámetros, requiere la respuesta desde el dispositivo, con el mismo formato. Estos comandos, son los que se muestran en la sección 5.4.1.1.

Dentro de código principal, para saber si existe algún dispositivo que quiera conectarse, se debe realizar las siguientes sentencias:

```

1 EthernetClient cliente_gpr = Gpredict.available() ;
2 if(cliente_gpr)
3 {
4 }
5

```

codigo 7: captura de paquetes recibidos mediante el software Gpredict

Dentro de las llaves iría la acción a realizar según el comando recibido. Los comandos que utiliza Gpredict, son los que se muestran en la tabla 5.2. Los métodos set, vienen de la forma comando par1 par2”, siendo par1, el ángulo azimutal, y el par2, el ángulo de altura. Los métodos get, piden información, y al recibirlos, se debe responder con un salto de linea entre parámetros. Para exemplificar: supongase que se lee el valor de ”P 155.20 38.3”. Esto significa que debe apuntar la antena al punto 155.20° en azimuth, y 38.3° en altura, pero no debe responder la petición. Si en vez de una P, llegase una ”p”, este comando, indica que debe responder su posición al programa. Supongamos que la antena, se encuentra en la posición 125.0° en azimuth, y 39° en altura, la respuesta a esta petición será ”125.0

n 39”

n”, y en la pantalla de Gpredict, aparecerán estas coordenadas. En este ejemplo, no se aplicó la corrección de las coordenadas, estan implementadas dentro del software. En el caso, que llegue un comando q(desconectarse del gpredict), se debe cambiar la referencia al cenit. En caso de el comando S, se debe para la antena, mediante la función ”mover_antena”.

Todo el código desarrollado se encuentra en el apéndice del presente capítulo.

6.8.4. Stellarium

Para el stellarium, debe configurarse el software, según la sección 5.5.1. Una vez allí, se configura el software como muestra la figura 6.19.

Una vez configurado, debe presionar control + 0.º en la barra inferior, debe buscar el icono que le diga ”mover el telescopio a las coordenadas definidas. Una vez allí, se le abrirá una ventana con la

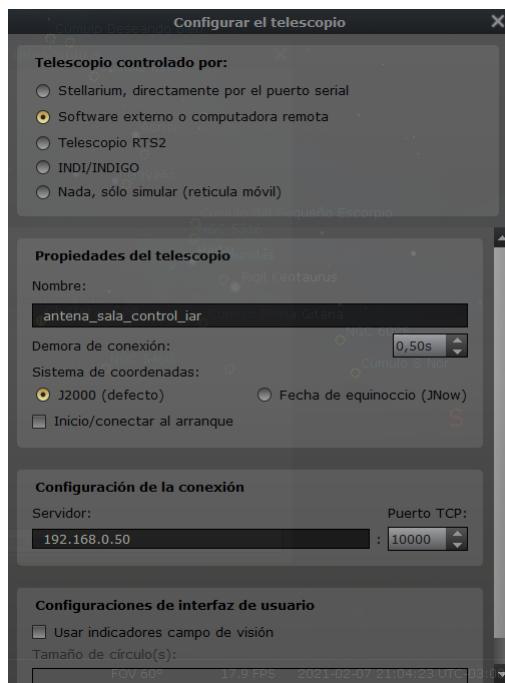


Figura 6.19: configuración del stellarium

opción de “configurar telescopio”. Debe abrir esa ventana, y le aparecerá el telescopio configurado recientemente. Puede configurar más de un telescopio si así lo desea. Aparecerá una ventana como la que se muestra en la figura 6.20.

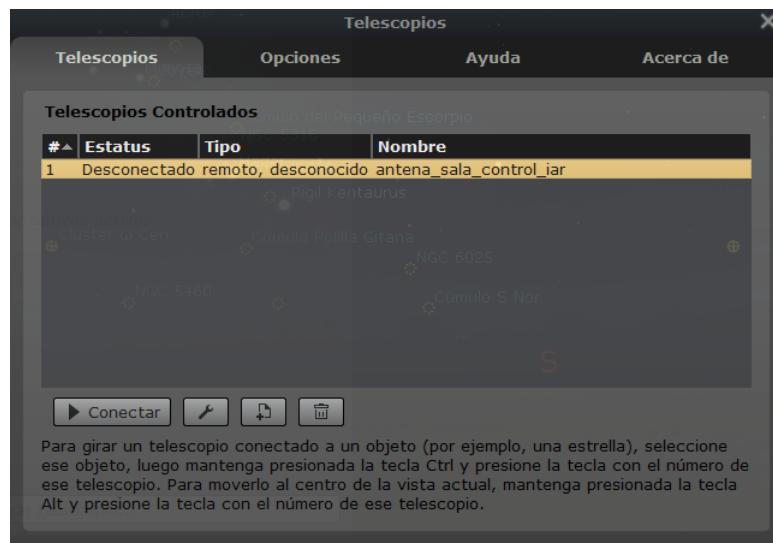


Figura 6.20: Apertura del telescopio recién creado.

Una vez, ahí dentro, presionando el botón conectar, se aprecia la siguiente ventana que aparece en la figura 6.21.

donde se observan los siguientes botones y opciones:

- Rotar: envia las coordenadas al dispositivo que mueve la antena, en el caso del presente documento, es el dispositivo desarrollado en este trabajo.
- Sync: En desuso, no posee ningun tipo de uso.

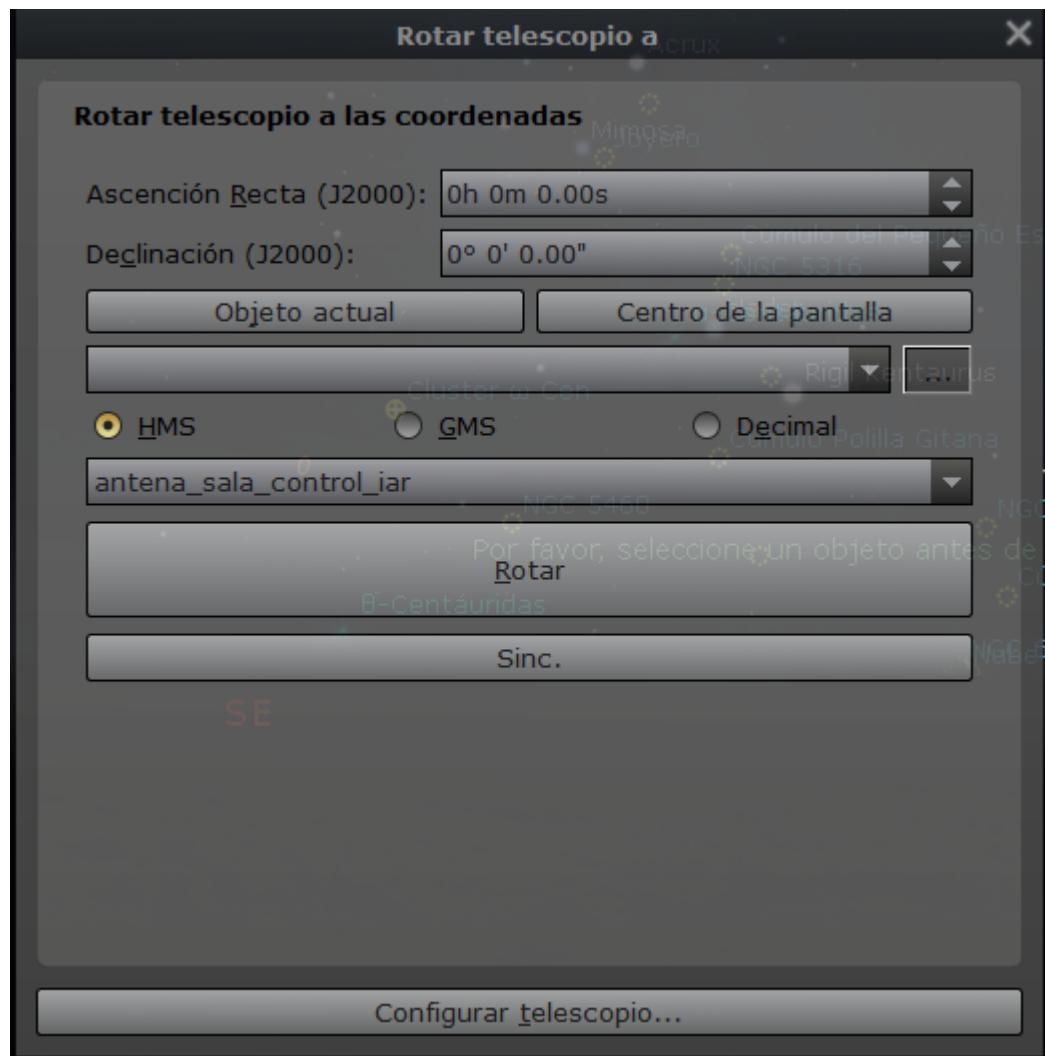


Figura 6.21: Imagen del panel de control del telescopio usando Stellarium

- Ascencion recta: tipo de coordenadas para visualizar estrellas.
- declinación: Idem que ascención recta.
- objeto actual,centro de la pantalla: Permite poner las coordenadas en a donde debe apuntar el telescopio, usando algún astro seleccionado, o el centro de la pantalla.
- HMS, GMS,DECIMAL: tipo de ángulo mostrado. HMS es utilizando el sistema de medición angular basado en horas, el segundo, es el sexagesimal, y el tercero es el decimal.

Este software, no trae límites para el control de la antena, ya que al ser un software utilizado para telescopios, supone que puede girar 360° en azimut y 90° en altura. El sistema de coordenadas utilizado se denomina ecuatorial, y debe realizarse una transformación de coordenadas para poder mover la antena. Esta transformación, se realiza en la siguiente fase del desarrollo del proyecto.

6.8.5. Programación para conectarse con Stellarium

En este caso, se debe definir un nuevo objeto servidor. La nueva definición del objeto se realiza de la siguiente manera:

```
1 #define PORT_STELLARIUM 10000
2 EthernetServer stellarium(PORT_STELLARIUM) ; // socket tcp/ip para stellarium
```

codigo 8: definición de objeto servidor para conectarse con el stellarium

En la sección [5.5.2](#), se explica como envía los datos el stellarium, es decir, la trama de datos. En ésta sección, se explica que el programa stellarium, llama a un software externo para enviar estos datos a través del protocolo TCP/IP. Este se denomina "telescope_server". La codificación que utiliza, es la siguiente, dentro de los bits, la cual según su documentación, utiliza los ángulos en un sistema horario:

- ascención recta:
 - ascención recta 0hs = 0x100000000 =
 - ascención recta 12hs = 0x80000000
 - ascención recta 24hs = 0x00
- declinación:
 - declinacion -90° = -0x400000000
 - declinacion 0° = 0x000000000
 - declinacion -90° = -0x400000000

y con estos valores, se realiza una regla de tres simple, para ambos ejes coordenados.

Dado que los bits, llegan desordenados, (es decir, llegan en orden inverso), se los debe ordenar, para luego realizar la regla de tres simple, las instrucciones para organizar estos datos son:

donde las variables dec y RA, son las coordenadas recibidas desde el software stellarium, los bits recibidos, se guardan en el vector uint8_t sunP[20].

Una vez se realiza esto, se debe definir el código para que el microcontrolador se comunique con el software stellarium. Este código, se realiza dentro del bucle principal del programa, y se le añade el código , dentro del comportamiento. El código es el siguiente:

```
1 uint8_t sunP[19] ;
2 ...
3
4 dec = 0x00000000 | (long (sunP[19])<<24) | (long (sunP[18])<<16) | (long
5   ↵ (sunP[17])<<8) | (long (sunP[16])<<0);
6 RA = 0x00000000 | (long (sunP[15])<<24) | (long (sunP[14])<<16) | (long
7   ↵ (sunP[13])<<8) | (long (sunP[12])<<0);
```

codigo 9: Reorganización de los datos recibidos desde el programa stellarium dentro del microcontrolador

```

1 EthernetClient cliente_s = stellarium.available() ;
2 long int dec ;
3 unsigned long int RA = 0 ;
4 uint8_t sunP[20] ; // vector datos recibidos
5 char state_con = 0 ;
6 if (cliente_s)
7 {
8     % aquí se implementa el código mostrado en el código 9
9 }
```

codigo 10: Parte del software que se encarga de conectarse con el software stellarium programado dentro del microcontrolador.

Luego, dentro de cada bloque if (ver código 7 y código 10), van otras líneas adicionales, las cuales no se han mostrado para no extender el documento. Estas líneas adicionales se encuentran dentro del anexo del presente capítulo, y se encargan de mantener la conexión entre los programas de la PC, y el microcontrolador.

6.8.6. Resultados de la conexión con Gpredict y Stellarium

En esta sección, se comprueba la conexión, el intercambio de mensajes entre stellarium y el microcontrolador atmega328p, y el microcontrolador y el Gpredict, utilizando como interface entre la red, el dispositivo el chip ethernet w5100.

La herramienta seleccionada para realizar este análisis es el software WireShark, que se muestra en el capítulo de redes. Cada software se analiza por separado. Cabe mencionar, que el chip utilizado, no soporta ambos software trabajando en simultáneo, debido a sus limitaciones físicas, impuestas por el fabricante. Por este motivo, se prueban ambos software por separado.

6.8.6.1. Conexión con Gpredict.

Una vez, configurado el software Gpredict, para conectarse, se debe abrir el panel de control del rotador, y se debe oprimir el botón “engage” para calibrar la antena. Si la conexión funciona, dentro de los cuadros de azimuth y elevación, se obtienen las posición de la antena (donde dice read.en la figura 5.12). Esta posición, aparece como una cruz roja en el polar plot. El punto rojo, es la posición a la que se debe dirigir la antena, la cual es la coordenada azimuth y altura, marcada con números más grandes en la imagen 5.12.

En la prueba siguiente, se le dió a la antena, una altura de 45° y ángulo de azimuth de 180°. Esta posición esta marcada con un círculo rojo dentro de la gráfica polar. La posición que se encuentra el rotador, es de 253.78° en azimuth y 47.85° en altura, esta posición se encuentra marcada con una cruz roja dentro de la gráfica polar (ver figura 6.22). A continuación, se han movido los potenciómetros manualmente, y se nota que al llegar a la posición, todos los leds del protoboard, permanecen apagados. Esto es un indicativo, que tanto el control como el software responden de manera apropiada.

Luego de esta prueba, se ha probado el seguimiento de un satélite, de forma manual (es decir, moviendo los potenciómetros), en este caso, aparece el círculo rojo, la cruz indicando la posición de la antena, y un punto rojo, el cual indica la posición del satélite en el plano polar. Durante la prueba, se ha utilizado el software wireshark, para monitorizar los comandos enviados y recibidos

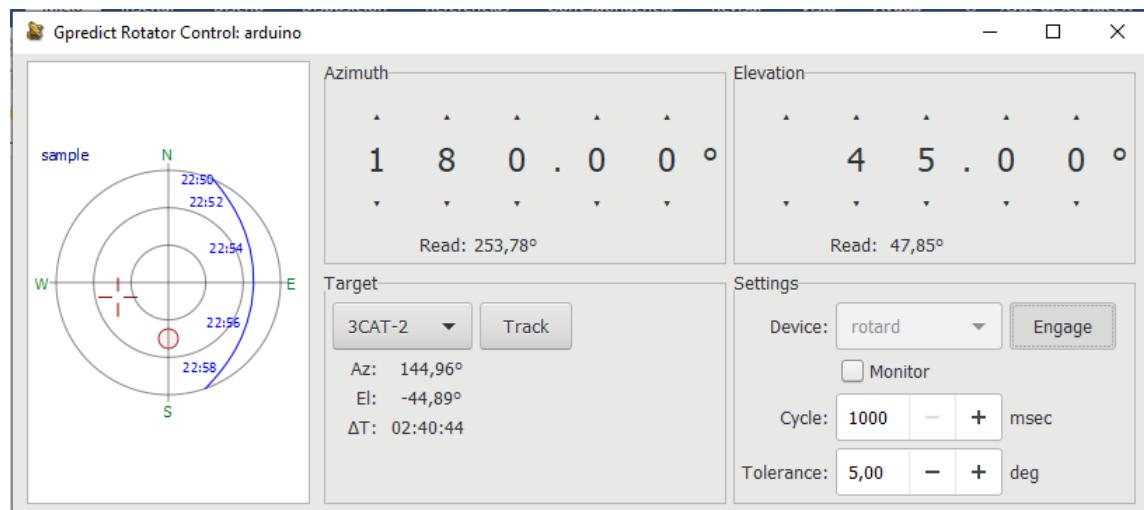


Figura 6.22: Conexión del microcontrolador al software Gpredict mediante el uso del protocolo TCP/IP

desde Gpredict. En los comandos enviados y recibidos, no se ha mostrado ningún tipo de anomalía. A continuación se dejan los resultados del panel de control y del software wireshark

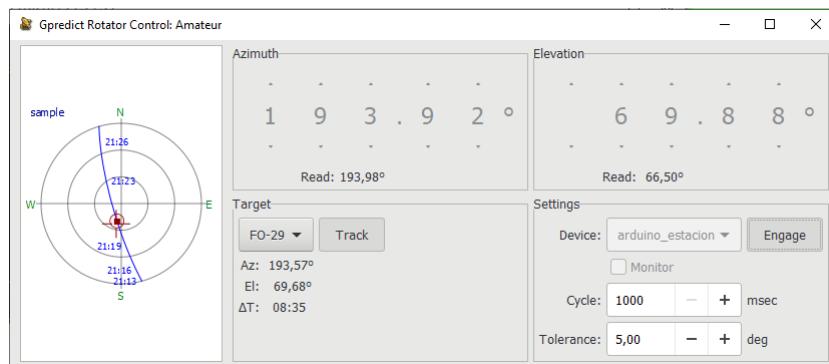


Figura 6.23: Panel de control del monitor realizando un seguimiento satelital

6.8.6.2. Conexión con Stellarium.

En el caso de stellarium, al no poseer un panel de control, como el Gpredict, se debe verificar de alguna forma, que las coordenadas recibidas, sean las correctas. En este caso, se usa el puerto serie como medio de depuración. Dentro del código, se transforman las coordenadas y se verifica que la unidad angular sea la correcta. Además, se realiza una verificación mediante el software wireshark para asegurarse que la comunicación sea fiable, y se respeten los protocolos entre ambos dispositivos(PC y microcontrolador). Para verificar si las coordenadas son las correctas, se ha oprimido el botón “centro” de la pantalla, y luego el botón “rotar”. La figura 6.25 muestra las coordenadas enviadas por el panel de control del telescopio del software stellarium, y en la figura 6.26 se observan los resultados por puerto serie.

Además, se revisa la conexión mediante el software wireshark. Los resultados del análisis se muestran a continuación:

No.	Time	Source	Destination	Protocol	Length	Info
39	9.554398	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=41 Ack=78 Win=64163 Len=0
40	9.566423	192.168.0.51	192.168.0.127	TCP	60	[TCP Window Update] 4533 + 49786 [ACK] Seq=78 Ack=41 Win=2048 Len=0
41	10.202295	192.168.0.127	192.168.0.51	TCP	69	49786 → 4533 [PSH, ACK] Seq=41 Ack=78 Win=64163 Len=15
42	10.215048	192.168.0.51	192.168.0.127	TCP	60	4533 + 49786 [PSH, ACK] Seq=78 Ack=56 Win=2033 Len=6
43	10.215048	192.168.0.51	192.168.0.127	TCP	60	[TCP Window Update] 4533 + 49786 [ACK] Seq=84 Ack=56 Win=2048 Len=0
44	10.255272	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=56 Ack=84 Win=64157 Len=0
45	10.317685	192.168.0.127	192.168.0.51	TCP	56	49786 → 4533 [PSH, ACK] Seq=56 Ack=86 Win=64157 Len=2
46	10.329530	192.168.0.51	192.168.0.127	TCP	67	4533 + 49786 [PSH, ACK] Seq=84 Ack=58 Win=2046 Len=13
47	10.371354	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=58 Ack=97 Win=64144 Len=0
48	10.388728	192.168.0.51	192.168.0.127	TCP	60	[TCP Window Update] 4533 + 49786 [ACK] Seq=97 Ack=58 Win=2048 Len=0
49	11.041172	192.168.0.127	192.168.0.51	TCP	69	49786 → 4533 [PSH, ACK] Seq=58 Ack=97 Win=64144 Len=15
50	11.051897	192.168.0.51	192.168.0.127	TCP	60	4533 + 49786 [PSH, ACK] Seq=97 Ack=73 Win=2033 Len=6
51	11.051897	192.168.0.51	192.168.0.127	TCP	60	[TCP Window Update] 4533 + 49786 [ACK] Seq=103 Ack=73 Win=2048 Len=0
52	11.103452	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=73 Ack=103 Win=64138 Len=0
53	11.156702	192.168.0.127	192.168.0.51	TCP	56	49786 → 4533 [PSH, ACK] Seq=73 Ack=103 Win=64138 Len=2
54	11.169040	192.168.0.51	192.168.0.127	TCP	67	4533 + 49786 [PSH, ACK] Seq=103 Ack=75 Win=2046 Len=13
55	11.219095	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=75 Ack=116 Win=64125 Len=0
56	11.233348	192.168.0.51	192.168.0.127	TCP	60	[TCP Window Update] 4533 + 49786 [ACK] Seq=116 Ack=75 Win=2048 Len=0
57	11.872378	192.168.0.127	192.168.0.51	TCP	69	49786 → 4533 [PSH, ACK] Seq=75 Ack=116 Win=64125 Len=15
58	11.884635	192.168.0.51	192.168.0.127	TCP	60	4533 + 49786 [PSH, ACK] Seq=116 Ack=90 Win=2033 Len=6
59	11.884635	192.168.0.51	192.168.0.127	TCP	60	[TCP Window Update] 4533 + 49786 [ACK] Seq=122 Ack=98 Win=2048 Len=0
60	11.941305	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=90 Ack=122 Win=64119 Len=0
61	11.988251	192.168.0.127	192.168.0.51	TCP	56	49786 → 4533 [PSH, ACK] Seq=90 Ack=122 Win=64119 Len=2
62	12.000087	192.168.0.51	192.168.0.127	TCP	67	4533 + 49786 [PSH, ACK] Seq=122 Ack=92 Win=2046 Len=13
63	12.041334	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=92 Ack=135 Win=64108 Len=0
64	12.052483	192.168.0.51	192.168.0.127	TCP	60	[TCP Window Update] 4533 + 49786 [ACK] Seq=135 Ack=92 Win=2048 Len=0
65	12.819689	192.168.0.127	192.168.0.51	TCP	56	49786 → 4533 [PSH, ACK] Seq=92 Ack=135 Win=64106 Len=2
66	12.864265	192.168.0.51	192.168.0.127	TCP	67	4533 + 49786 [PSH, ACK] Seq=135 Ack=94 Win=2046 Len=13
67	12.919559	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=94 Ack=148 Win=64093 Len=0
68	12.932047	192.168.0.51	192.168.0.127	TCP	60	[TCP Window Update] 4533 + 49786 [ACK] Seq=148 Ack=94 Win=2048 Len=0
69	13.688476	192.168.0.127	192.168.0.51	TCP	56	49786 → 4533 [PSH, ACK] Seq=94 Ack=148 Win=64093 Len=2
70	13.701127	192.168.0.51	192.168.0.127	TCP	67	4533 + 49786 [PSH, ACK] Seq=148 Ack=96 Win=2046 Len=13
71	13.701700	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=96 Ack=161 Win=64090 Len=0

Figura 6.24: Captura de paquetes TCP/IP utilizando el programa wireshark mediante el seguimiento de un satélite

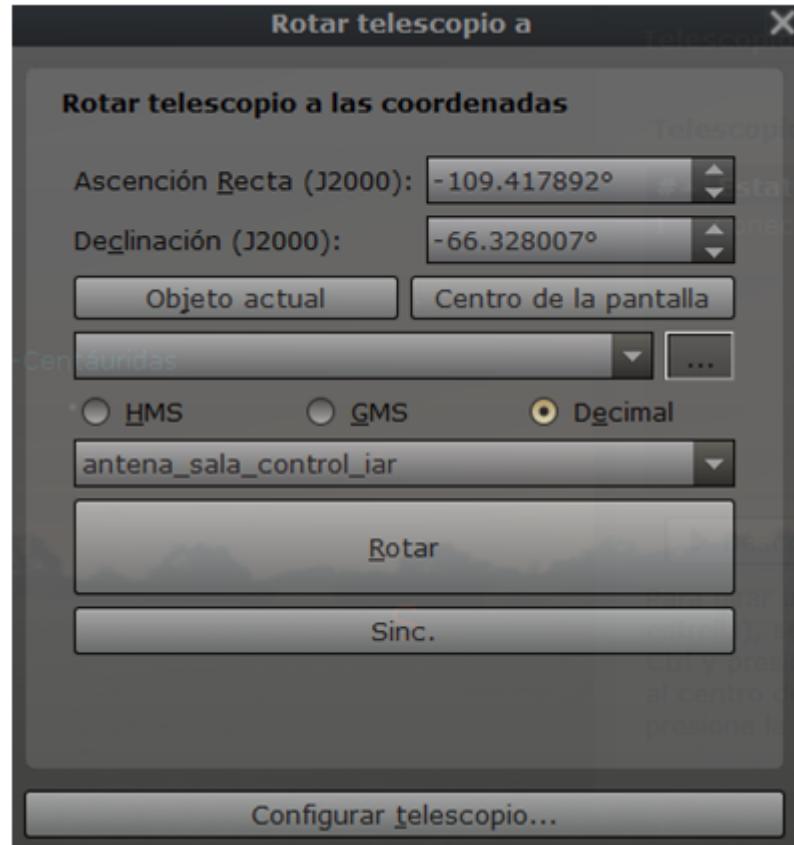


Figura 6.25: Panel de control del rotador del stellarium al momento de realizar las pruebas sobre el software

6.9 Unión de partes del software

En esta sección se procede a unir todas las partes desarrolladas en esta etapa. Esta unión, se realiza utilizando el esquema de la figura 6.1. La primera parte, debe realizar la función de

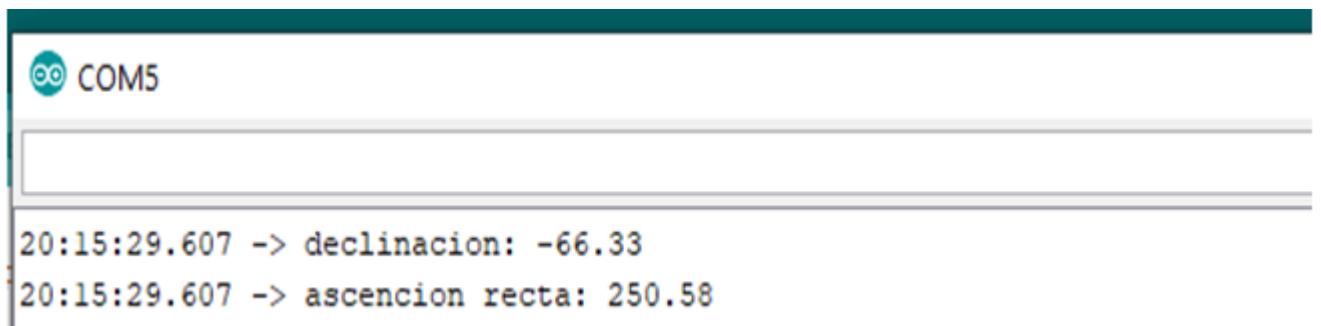


Figura 6.26: Coordenadas recibidas por el stellarium

No.	Time	Source	Destination	Protocol	Length	Info
15	17.287374	192.168.0.127	192.168.0.51	NBNS	92	Name query NBSTAT
18	17.296717	192.168.0.51	192.168.0.127	ICMP	70	Destination unreachable
21	18.789496	192.168.0.127	192.168.0.51	NBNS	92	Name query NBSTAT
22	18.799747	192.168.0.51	192.168.0.127	ICMP	70	Destination unreachable
23	20.289959	192.168.0.127	192.168.0.51	NBNS	92	Name query NBSTAT
24	20.299849	192.168.0.51	192.168.0.127	ICMP	70	Destination unreachable
28	21.830253	192.168.0.127	192.168.0.51	TCP	66	49898 → 10000 [SYN]
29	21.849818	192.168.0.51	192.168.0.127	TCP	60	10000 → 49898 [SYN]
30	21.850040	192.168.0.127	192.168.0.51	TCP	54	49898 → 10000 [ACK]
41	34.035676	192.168.0.127	192.168.0.51	TCP	74	49898 → 10000 [PSH]
42	34.070840	192.168.0.51	192.168.0.127	TCP	60	10000 → 49898 [ACK]
43	34.087505	192.168.0.51	192.168.0.127	TCP	74	10000 → 49898 [PSH]
44	34.119106	192.168.0.127	192.168.0.51	TCP	54	49898 → 10000 [FIN]
45	34.138724	192.168.0.51	192.168.0.127	TCP	60	10000 → 49898 [ACK]
46	34.138724	192.168.0.51	192.168.0.127	TCP	60	10000 → 49898 [FIN]
47	34.138852	192.168.0.127	192.168.0.51	TCP	54	49898 → 10000 [ACK]
48	34.184201	192.168.0.127	192.168.0.51	TCP	66	49899 → 10000 [SYN]
49	34.194159	192.168.0.51	192.168.0.127	TCP	60	10000 → 49899 [SYN]
50	34.194310	192.168.0.127	192.168.0.51	TCP	54	49899 → 10000 [ACK]
52	41.054255	192.168.0.127	192.168.0.51	TCP	54	49899 → 10000 [FIN]
53	41.065485	192.168.0.51	192.168.0.127	TCP	60	10000 → 49899 [ACK]
54	41.065485	192.168.0.51	192.168.0.127	TCP	60	10000 → 49899 [FIN]
55	41.065610	192.168.0.127	192.168.0.51	TCP	54	49899 → 10000 [ACK]

Figura 6.27: Coordenadas recibidas por el stellarium

autocalibración, luego debe iniciar el control de la antena, para volverla a su posición de equilibrio.

Para realizar el control de la antena, se debe llamar a la función `Leer_encoders()`, acto seguido, a la función de control, `controlMotores(ref1, ref2)`. Esto debe realizarse periódicamente, por ende, dentro de la programación del scheduler deben ponerse ambas funciones. El tiempo de control seleccionado es 10ms. Estas funciones, se encuentran dentro del archivo "timer.cpp". Para inicializar dichas funciones, se debe realizar las siguientes sentencias:

Luego de realizar esto último, se obtiene la dirección IP mediante DHCP , y se programa la conexión dentro del bucle principal del programa. A continuación se muestra la parte de inicialización, y el bucle principal del programa, donde el código obtenido hasta el momento, se muestra en el apéndice del presente capítulo

En esta parte del desarrollo, se ha utilizado el puerto serie para mostrar información. En la

```

1 #define NUMBER_CLOCK 0
2 #define HOUR_CLOCK 0
3 #define MINUTE_CLOCK 0
4 #define SEGUNDO_CLOCK 0
5 #define MILISECOND_CLOCK 10
6
7     aqui siguen otras definiciones ...
8 ...
9     puesta en marcha del control cada 10 milisegundos
10 Base_tiempo() ;
11 timerStart(NUMBER_CLOCK,HOUR_CLOCK,MINUTE_CLOCK,SEGUNDO_CLOCK,MILISECOND_CLOCK)
12
13 en el bucle principal del programa debe ir
14 timerEvent() ;

```

lenguaje	archivos	codigo	comentarios	lineas en blanco	total
C++	8	692	189	197	1078
ini	1	13	9	2	24

Tabla 6.5: Tabla resumén de las lineas de desarrollo realizadas hasta el momento

próxima sección, se utiliza el display LCD para mostrar información. Esto se realiza en el mismo reloj definido para utilizar la función leer_encoders y controlMotores(ref1,ref2) ;

6.9.1. Resultados

Para realizar la depuración del software realizado hasta este momento del desarrollo, se han realizado los siguientes pasos:

1. Se conecto Gpredict con el rotador.
2. Se movieron los potenciómetros, y se revisó que el ángulo de movimiento total de la antena sea el de la figura 6.17
3. Se conecto Stellarium con el rotador.
4. Se mueve la posición del software stellarium, y quedan registrados los datos por puerto serie.

Luego de realizados estos pasos, se comprobó que todo el software respondía de forma correcta. No se incluirá ninguna imagen, ya que las respuestas son las mismas que corresponden a cada sección del presente capítulo de este documento. En esta sección, se ha construido la mayor parte del software.

6.9.2. análisis del código realizado

En esta sección, se muestran las principales métricas relacionadas al código. En primer lugar, se cuentan las lineas de desarrollo, utilizando la herramienta de visual Studio Code denominada VSCodeCounter. Los resultados se resumen a continuación:

Luego de esto, se analiza, los resultados arrojados por el compilador, que el código ocupa 80 % de la memoria de programa, y 52.9 % de la memoria RAM.

Viendo la gran cantidad de memoria de programa que ocupa, se realizo un análisis con mayor profundidad del código, utilizando una herramienta de desarrollo conocida como ".avr-obj-dump". La explicación de esta herramienta rebasa el alcance de este documento. Solamente se dirá que es una herramienta para analizar la memoria de microcontroladores AVR, revisando variables, objetos, etc. Una de las conclusiones, que se pueden obtener utilizando esté metodo, es que las librerías de arduino, se cargan de forma completa dentro del microcontrolador. No distingue entre funciones que usa y cuales no, el compilador. Debido a esta situación, se han programado una cantidad moderada de lineas de código para obtener tanta capacidad de memoria flash. A continuación se deja una imagen de los resultados del obj dump

```

008004b5 g      0 .bss  0000000d .hidden lcd
00005bf8 g      .text  00000000 __ctype_istrue
008002da g      0 .bss  00000002 .hidden azimuth
00002fe0 g      F .text  0000002a .hidden _ZN10W5100Class11writeSnPORTEhj
0000ffa0 g      *ABS* 00000000 __DATA_REGION_LENGTH__
00005ec4 g      F .text  00000008 __divsf3
00005d38 g      .text  00000036 .hidden __epilogue_restores_
00000eae g      F .text  0000002e .hidden _ZN17LiquidCrystal_I2C4sendEhh
00001484 g      F .text  00000040 .hidden _ZN9DNSClientC2Ev.constprop.23
00800373 g      0 .bss  00000001 .hidden twi_inRepStart.lto_priv.139
0000097a g      F .text  00000030 .hidden _Z12mover_antennacc.part.0.lto_priv.142
00005c28 g      F .text  00000022 __fp_round
000002ea g      F .text  0000001c .hidden _ZN14HardwareSerial4peekEv
00800298 g      0 .bss  00000040 .hidden timer
00005fa8 g      F .text  0000000e __fixfsfi
00001da2 g      F .text  00000050 .hidden twi_transmit
00800288 g      0 .bss  00000002 .hidden antenna
00003244 g      F .text  000000f2 .hidden _ZN13EthernetClass10socketSendEhPKhj
00005c4a g      F .text  00000044 __fp_split3
00000612 g      F .text  00000092 .hidden _ZN5Print11printNumberEmh
00000250 g      F .text  00000054 .hidden _ZN5Print5writeEPKhj
00000bae g      F .text  00000206 .hidden _Z32function_compare_autocalibracionv
00003570 g      F .text  000000b8 .hidden _ZN11EthernetUDP11parsePacketEv
00000068 g      .text  00000000 __trampolines_start
000005b4 g      F .text  00000034 .hidden _ZN6StringC1ERKS_
00005ad2 g      F .text  00000048 __fp_cmp
008004c2 g      0 .bss  00000006 .hidden Gpredict
00000ad6 g      F .text  000000d8 .hidden _Z10timerEventv
00002a0e g      F .text  00000016 .hidden _ZN10W5100Class5setSSEv
00800528 g      0 .bss  00000002 errno
00006462 g      .text  00000000 __etext

```

Figura 6.28: Programa desensamblado en código assembler del microcontrolador. Se utilizó la herramienta avr-obj-dump.

6.10 Conclusiones

En este capítulo, se muestra como han sido los pasos para construir un apuntador de antena basado en tecnología arduino, y sus librerías. Las librerías, poseen la desventaja de cargarse en memoria sin distinción acerca del uso o no de la funcionalidad. Esto, es un punto en contra, pero la velocidad de desarrollo mediante el uso de estas librería, es superior si uno debe realizar los desarrollos desde cero.

Se ha logrado realizar un control de tipo ON/Off, a pesar de las limitaciones del microcontrolador, y que respondan de manera adecuada a los requerimientos del proyecto, además, posee un

manejo muy preciso de los tiempos de actuación, a pesar de ser un procesador de bajo costo en el mercado local.

Para finalizar, esta sección, se ha construido la gran parte del software en esta sección, queda la parte de transformación de coordenadas, que será analizada y aplicada en la próxima fase. La construcción de esta función, requiere de conocimientos de trigonometría esférica, ya que todas estas coordenadas utilizadas, se refieren a una esfera.

```
1 void setup()
2 {
3     // debugger --
4     Serial.begin(9600) ;
5     /*** inicializacion de puertos del motor como salida *****/
6     init_pins_motores() ;
7     autocalibracion() ;
8     // inicializacion de ethernet
9     Ethernet.init(PINSS) ;
10    byte mac [] = {0x00, 0xCD, 0xEF, 0xEE, 0xAA, 0xBC}; // dirmac
11    #if ETHERNET_IP_LOCAL
12    if (Ethernet.begin(mac,ip,DNS,gateway,mask_net) == 0)
13    {
14        Serial.print(F("Fallo DHCP"));
15    }
16    else
17    {
18        Serial.print(Ethernet.localIP());
19    }
20    #else
21    if (Ethernet.begin(mac) == 0)
22    {
23        Serial.print(F("Fallo DHCP"));
24    }
25    else
26    {
27        Serial.print(Ethernet.localIP());
28    }
29    #endif
30    // inicio scheduler -- >
31    Base_tiempo() ;
32    timerStart(NUMBER_CLOCK,HOUR_CLOCK,MINUTE_CLOCK,SEGUNDO_CLOCK,MILISECOND_CLOCK) ;
33 }
34
35 void loop()
36 {
37     timerEvent() ;
38     EthernetClient cliente_gpr = Gpredict.available() ;
39     EthernetClient cliente_s = stellarium.available() ;
40     long int dec ;
41     unsigned long int RA = 0 ;
42     uint8_t sunP[20] ; // vector datos recibidos -- sunposition
43     char state_con = 0 ;
44     if(cliente_gpr)
45     {
46
47     }
48     if(cliente_s)
49     {
50
51     }
52
53 }
```

Códigos de programacion

A.1 Códigos de programación C/C++

Fase III

**Física de los satélites y sistemas
de coordenadas astronómicas**

Dinámica de los satélites elementos keplerianos

7.1 Introducción

En este capítulo, se introducen los conceptos básicos relacionados con la física de los satélites, cálculo de las órbitas, y su solución para encontrar las posiciones de los satélites. Estas posiciones vienen dadas desde una base de datos, pero el software (gpredict y stellarium), deben calcular las posiciones en base a su última actualización. La materia que se encarga de este estudio, se denomina mecánica orbital. Esta materia, es absolutamente general, ya sea para movimientos de estrellas, satélites naturales y artificiales. Sus principios son dos leyes básicas de la naturaleza: Las tres leyes de Newton y las tres leyes de Kepler. A partir de estas dos leyes, y utilizando cálculo diferencial, puede obtenerse la solución, para las posiciones de los satélites, estrellas, etc, en función del tiempo, a partir de una condición inicial.

7.2 Conceptos básicos

En esta sección, se abordan los conceptos básicos para la deducción de las tres leyes de Kepler. Se omiten los conceptos de cálculo diferencial y vectores en el plano. Si el lector no está familiarizado con algunos de estos temas, se sugiere revisar literatura al respecto. Además, si el lector, está familiarizado con las leyes de Newton, sistemas de coordenadas, puede saltarse esta sección.

Para una compresión completa de la mecánica de los objetos celestes (ya sea satélites artificiales, naturales, y estrellas, etc), se requiere en primer lugar un sistema de coordenadas, y un sistema de referencia de tiempo. En general se elige el centro del sistema de coordenadas, como el centro del planeta Tierra, ya que la mayor parte de las observaciones se realizan desde la Tierra.

7.2.1. cinemática

Para seguir el movimiento de una partícula en el espacio euclídeo, se necesita un marco de referencia, que consta de un reloj, y un sistema de coordenadas. En mecánica no relativista, el reloj es universal para todos los sistemas de coordenadas. Dada una partícula P, que está en un instante t, en la posición $r(t)$ dada por: $r(t) = x(t)i + y(t)j + z(t)k$

7.3 Leyes de Kepler

7.3.1. Problema de los dos cuerpos

7.3.2. Problema de n cuerpos

7.4 Parámetros orbitales o keplerianos

7.5 Determinación de las órbitas

7.6 Involucrando el tiempo en las ecuaciones

7.7 Cálculos de órbitas y modelos matemáticos

7.8 Estaciones terrenas

Sistemas de coordenadas astronómicas

Fase IV

Sistema electrónico de posicionamiento - Construcción de prototipo y resultados

Apéndice

Protocolos de comunicación Serie

A.1 Introducción

En la presente sección, se describen dos protocolos utilizados con los periféricos seleccionados para el presente trabajo final. Los dos protocolos que utilizan estos dispositivos se denominan Serial Peripheral Interface o SPI, por sus siglas en inglés, y el protocolo I2C (también se lo puede encontrar con el nombre de Two Wire o TWI). Ambos protocolos definen señales de entrada y salida, y un dispositivo maestro, que es el encargado de enviar los datos a los periféricos correspondientes. En este trabajo, el dispositivo maestro, es el microcontrolador ATmega328P. Ambos protocolos son de tipo serie. Además, se revisa el pinout disponible de cada dispositivo, y se da una breve explicación sobre cada pin que poseen los periféricos(w5100 ethernet y display LCD con chip adaptador para I2C.

A.2 Protocolo SPI

El protocolo SPI, define tres señales: MOSI,MISO y SCK o CLOCK. Dado que el protocolo SPI, consta de un dispositivo maestro, puede constar de varios dispositivos esclavos. Para seleccionar el dispositivo con el que se quiere comunicar, se debe agregar un cable adicional por dispositivo, este cable adicional se denomina Slave Select, o SS. El dispositivo maestro, selecciona un dispositivo esclavo para comunicarse a través de esta línea, denominada SS. A continuación se deja una imagen de las conexiones entre un dispositivo maestro y tres esclavos:

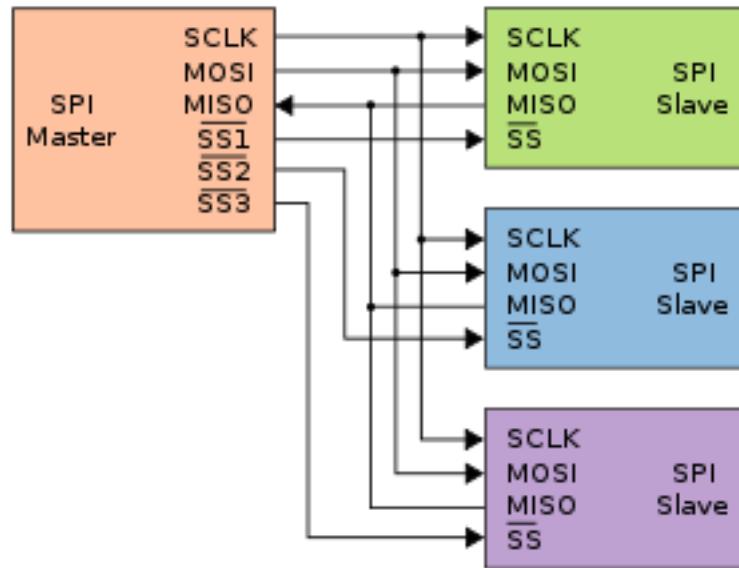


Figura A.1: Esquema de conexiones entre un dispositivo maestro y un dispositivo esclavo usando el protocolo SPI.

Las señales que utiliza el protocolo SPI son las siguientes:

- **SCK o CLOCK:** Se encarga de generar una señal de reloj. Esta señal es generada por el dispositivo maestro.

- MOSI(Master Output Slave Input): Son los datos que envia el dispositivo maestro al esclavo.
- MISO(Master Input Slave Output): Son los datos que envia el dispositivo esclavo al maestro.

El dispositivo maestro, es el encargado de seleccionar a que dispositivo se quiere comunicar, mediante la linea o cable de SS en la figura A.1. Además, el maestro, puede recibir información del dispositivo esclavo. Por tanto, la comunicación es de tipo "Full Duplex"(ambos dispositivos son capaces de transmitir y de recibir al mismo tiempo). Todos los datos salientes, se transmiten en serie, por ende, es requisito, que cada dispositivo maestro y/o esclavo, posean un registro de desplazamiento a medida que se reciben los datos.

Esta comunicación es sincrónica, la señal de CLOCK, genera una señal de reloj, de una determinada frecuencia, y esta es utilizada para la sincronía con las señales MOSI y MISO respectivamente, para que se transfieran los datos entre si. La señal de clock, solo esta activa durante la comunicación, luego se queda en estado alto o bajo, según se haya definido previamente. Debido a esto, se pueden definir cuatro modos de funcionamiento, dos basados en la polaridad del reloj, y otros dos, basados en la fase de la señal de reloj. A continuación se deja un diagrama de tiempos de las señales involucradas en el protocolo SPI.

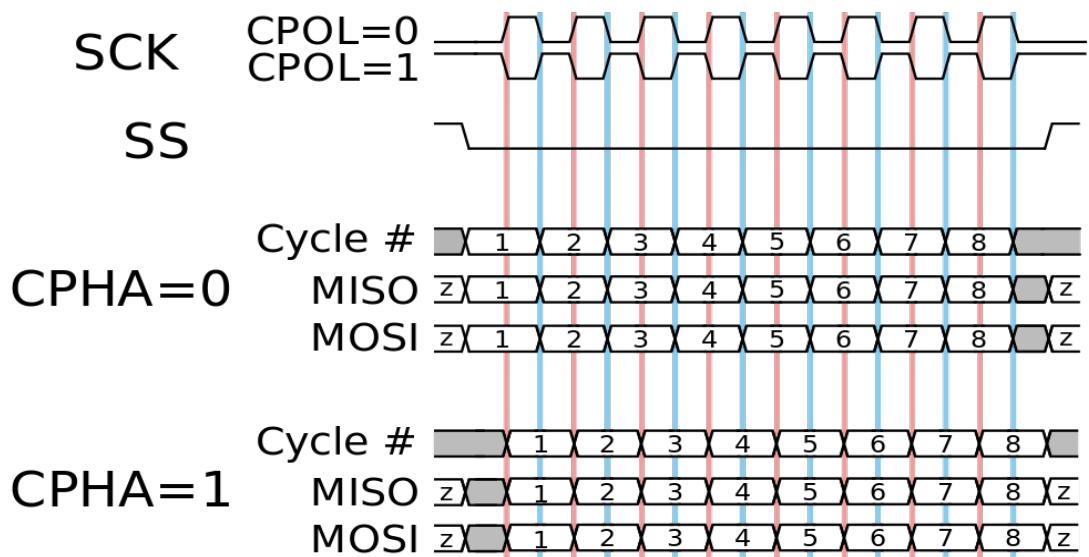


Figura A.2: Señales del protocolo SPI en función del tiempo. La letra z dentro del diagrama indica estado de alta impedancia

Como se observa, en la figura, primero se pone la señal de slave select(ss) en nivel bajo, y luego empieza la comunicación. Cuando finaliza la comunicación, vuelve a su estado alto. Las señales se denominan CPOL por las siglas de "clock polarity" y CPHA por "clock phase". De la figura, se observa que hay cuatro modos de funcionamiento:

- Modo 0: Con CPOL = 0 y CPHA = 0. Modo en el cual el estado del reloj permanece en estado lógico bajo y la información se envía en cada transición de bajo a alto, es decir alto activo.
- Modo 1: CPOL=1 y CPHA = 1. El estado del lógico del reloj es alto, y se envian datos cuando hay una transisción del estado alto al estado bajo.

- Modo 2: CPOL = 1 y CPHA = 0. El estado lógico del reloj es alto, y se envian los datos, y se envian en cada transicion de bajo a alto.(existe un desfase entre el estado logico y el envio de los datos).
- Modo 3: CPOL = 1 y CPHA = 1. Modo en el cual el estado del reloj permanece en estado lógico alto y la información se envía en cada transición de alto a bajo, es decir bajo activo.(existe un desfase entre el estado logico y el envio de los datos).

Por ultimo, se debe destacar, que no todos los dispositivos soportan todos los modos, por ende, para realizar una comunicación entre dos dispositivos, se debe verificar que ambos por los menos tengan un modo en común para que la comunicación pueda llevarse a cabo.

A.3 Procotolo I2C

El protocolo I2C, o Wire Serial, es un protocolo de comunicación serie, el cual define dos señales: SDA y SCL. La señal SDA es para transmitir los datos en formato serie(SDA proviene de las siglas serial data), y SCL, es una señal de reloj(SCL proviene de serial clock, en otras bibliografias, las puede encontrar como SCK o CLK).

Este protocolo, presenta una arquitectura del tipo maestro esclavo. El maestro es el encargado de enviar los datos, y los reciben los esclavos. Puede existir más de un maestro. El diagrama eléctrico es el siguiente:

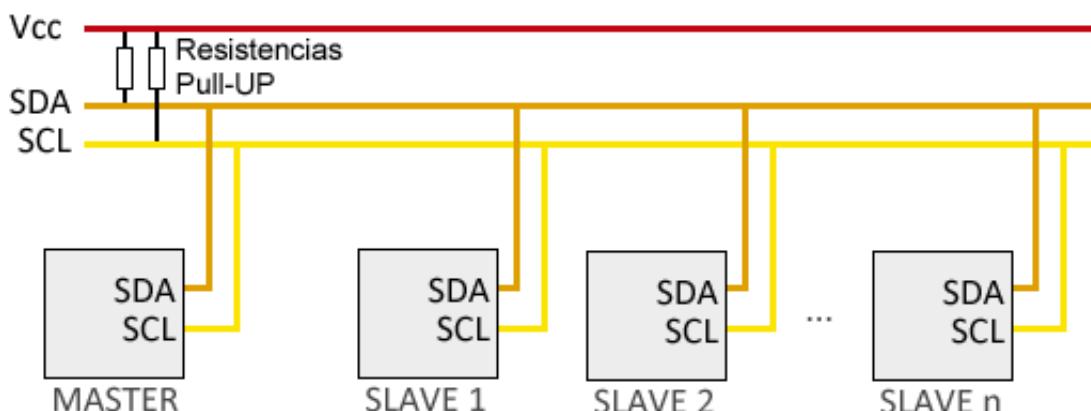


Figura A.3: diagrama de las conexiones entre una maestro y n esclavos mediante el uso de el protocolo I2C

Dado, que solo tiene dos cables, la forma de identificar un dispositivo es mediante una dirección, esta dirección, es un número binario formado por 7 bits, o 10 bits(en este último caso, son muy escasos los dispositivos), según el dispositivo. Si la dirección del esclavo, contienen solo 7 bits, esto quiere decir, que se puede conectar hasta 128 dispositivos. Esta dirección, debe ser dada por el fabricante del dispositivo o debe configurarse de alguna forma provista por el fabricante.

El maestro, se comunica con los esclavos, mediante el uso de una secuencia de inicio, y una secuencia de parada, con esto, los esclavos reconocen si se quiere comunicar con alguno de ellos. Entre mensajes, existen bits de reconocimiento que el esclavo envía al maestro, para que el maestro pueda reconocer de que forma se estan recibiendo los datos por parte de los esclavos o esclavo correspondiente.

Como se observa en la figura A.3, el estado del canal de comunicación por defecto es el estado alto. Cuando el master quiere iniciar la comunicación, pone la linea SDA en bajo, eso le indica a

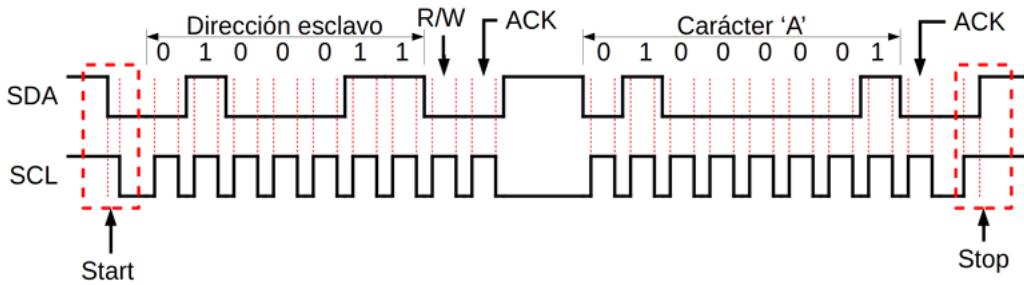


Figura A.4: Diagrama de tiempos que utiliza el protocolo I2C. En este caso, se utiliza una dirección de 7 bits

los esclavos que se va a iniciar una comunicación por parte del master. Acto seguido, se inicia la señal de SCL, la cual es una señal de reloj (ver figura A.4). Luego se envían de forma sincronizada la dirección del esclavo con el cual se quiere comunicar, y un bit extra, el cual le avisa si requiere información del dispositivo o va a enviarle información. El dispositivo que posee esa dirección, reconoce que le van a enviar datos, y envía un bit de ACK, sobre la línea SDA, poniéndola en bajo. Luego, envía otros 8 bits, y el esclavo envía un ACK, poniendo la línea SDA en bajo. Así, cada 8 bits, luego cuando la línea SCL permanece en alto, significa que la comunicación ha finalizado. Si el dispositivo no está disponible, o no existe esa dirección, la señal SDA permanece en alto, y se corta la comunicación. Por otro lado, si la comunicación es del esclavo al maestro, el maestro también debe enviarle el bit de confirmación correspondiente.

Cabe destacar, que la velocidad de comunicación viene dada por la señal de reloj. Se definen cuatro modos de velocidad para el bus I2C:

- MODO ESTANDAR
- MODO RAPIDO
- MODO HIGH SPEED
- MODO ULTRA FAST

Cada modo, define una frecuencia para su señal de reloj. Cuando se desean conectar maestros y esclavos, debe verificarse con su hoja de datos que ambos posean el mismo modo de velocidad, para cada dispositivo.

A.4 Chip ETHERNET W5100 shield

El chip ethernet W5100, ya viene en un módulo integrado, mostrado en la figura 3.1. Este módulo, posee 10 pines. Este módulo, trae un circuito integrado denominado W5100, el cual, se utiliza para proveer soluciones ethernet. Soporta los protocolos TCP/IP, y se puede configurar, para que se conecte a la red local mediante un cable ethernet.

Los puertos que tiene este módulo son los siguientes (ver figura A.5):

- pin Vcc: Pin donde se debe conectar una fuente de alimentación de 5Volts
- pin GND: Se conecta la tierra del sistema.
- pin SS: Cable de slave select para el protocolo SPI
- pin MOSI: Línea MOSI para el protocolo SPI

- pin MISO Linea MISO para el protocolo SPI
- pin SCK o CLOCK: Linea CLOCK para el protocolo SPI
- pin Reset: Resetea el dispositivo ethernet, borrando todas sus configuraciones.
- pin P+: Power of Ethernet. Cable destinado a recibir alimentación por la linea de red ethernet.Recibe potencia positiva
- pin P-: Power of Ethernet. Cable destinado a recibir alimentación por la linea de red ethernet.Recibe potencia negativa

Cabe destacar, que las lineas P+ y P-, son para poder utilizar el estandar "Power of Ethernet", el cual define lineas para la transmisión de la alimentación a traves de un cable de red.

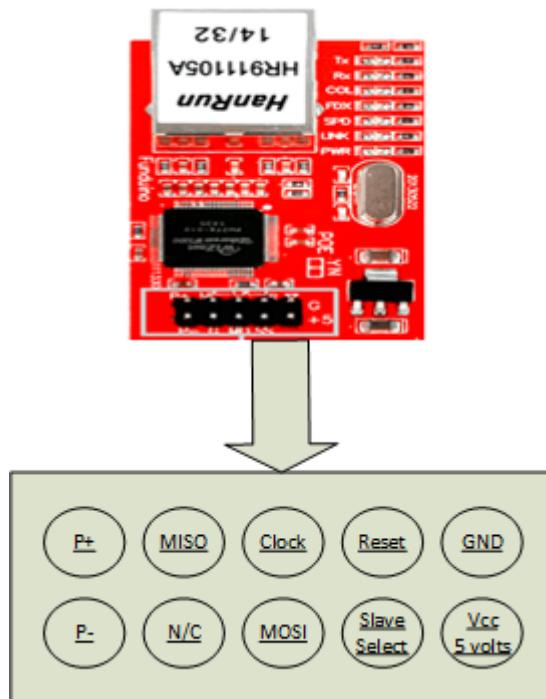


Figura A.5: Esquema de pines que posee el modulo W5100.

Si se observa la hoja de datos del dispositivo, se observan que los modos SPI soportados son los modos 0 y 3 respectivamente.

A.5 Display LCD con adaptador para I2C

En el caso del display, el display, trae integrado un adaptador para I2C. Este adaptador es el circuito integrado PCF8574. Este según su hoja de datos, se utiliza para expandir la cantidad de puertos de un microcontrolador, mediante el protocolo I2C a través de dos puertos, ya que este protocolo solo utiliza dos lineas para su comunicación.

Este dispositivo, (PCF8574) puede usarse para leer los puertos a la salida, o para escribir los puertos a su salida. En este caso particular, esta placa con este circuito integrado, esta realizada para aplicarla exclusivamente a los display LCD, y se provee la libreria para su uso con el display LCD.

Para conocer la dirección de comunicación, se debe revisar la hoja de datos, y puede darse hasta 8 direcciones distintas, en base a los puertos del circuito integrado, denominados A0,A1,A2,por el

fabricante. Estos puertos del circuito integrado PCF8574 se encuentran disponibles en la placa para ser modificados. La siguiente imagen muestra su ubicación en la placa:

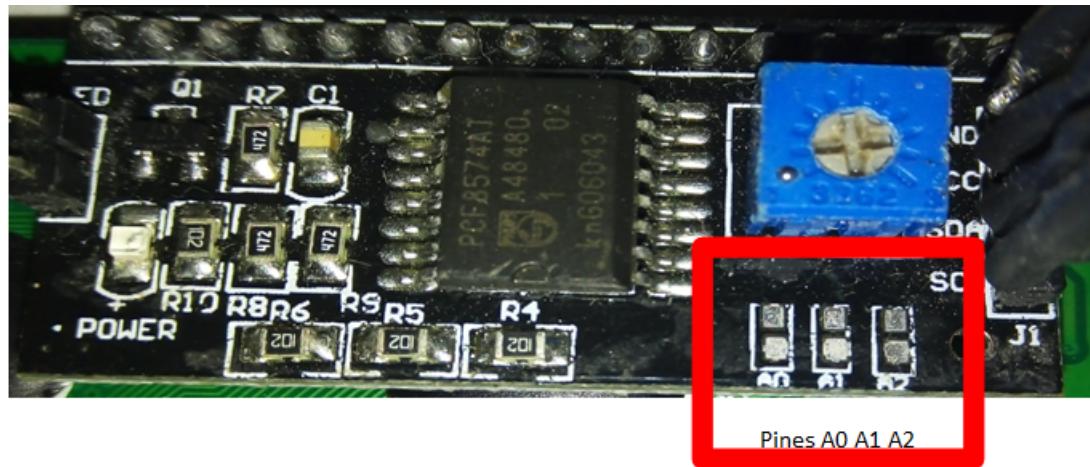


Figura A.6: Foto del display LCD de la parte de atrás, donde se observan los pines A0,A1 y A2 accesibles para poder modificarlos.

Esta placa por defecto, trae todos los pines A0,A1 y A2 en alto. Si queremos pasar algun de esos puertos a bajo, se debe estaniar en donde se encuentra el cuadrado rojo en la imagen A.6. De la hoja de datos, se explica que la dirección posee 7 bits, donde los siete bits estan dados por la forma 0100A2A1A0, si A0,A1 o A2 esta en alto, se indica un uno, caso contrario, se completa con cero. Por ejemplo si consideramos el caso de la figura, no ha sido modificado, por ende, A2 = 1 , A1 = 1 y A0 = 1. Esto nos indica que el dispositivo tiene la dirección 0100111, que es 27 en base hexadecimal.

Entonces, este procedimiento de poder cambiar la dirección del puerto I2C, permite conectar hasta 8 chips iguales, obteniendo 64 puertos de entrada y salida, manejado únicamente desde un dispositivo maestro, a través de dos líneas. Esto claramente presenta la ventaja de requerir menor cantidad de puertos, que un display LCD, y la diferencia de precios entre un display LCD, con chip I2C es infima, y no conviene optar por el display LCD sin el chip (la diferencia en pesos argentinos ronda los \$200, en esta época equivale a un 1,25 dólares aproximadamente).

Los puertos que tiene disponibles a la salida, son Vcc Y GND. En el pin de Vcc deben conectarse 5 voltios, y en GND la tierra. Los puertos SDA y SCL, son los puertos del protocolo I2C, y se deben conectar como en la figura A.3. En el caso, de conectarla a la placa Arduino Uno, como en este caso, las resistencias de pull-up, estan implementadas sobre la misma placa.

Programación sobre plataforma Arduino UNO - Periféricos del microcontrolador

B.1 PlatformIO - Configuración y flags de compilación

B.2 Primer Programa

B.3 Compilación condicional

B.4 Sentencias sobre el entorno arduino

B.5 Conversor Analógico Digital