

**Universidad Nacional de La Plata**

---



**Instituto Argentino  
de Radioastronomía**



**Facultad de ingeniería  
Departamento de electrotecnia  
Cátedra de trabajo final**  
Tesis para obtener el grado de Ingeniería electrónica

---

## **Posicionador de antena para bajada de datos satelitales**

---

Autor: Gastón Valdez<sup>1</sup>

Nº alumno 60847/5

Directores: Martín Salibe<sup>2</sup>

Elias Fliger<sup>3</sup>

**La plata, Año 2021**

---

<sup>1</sup> [gaston.cb.90@gmail.com](mailto:gaston.cb.90@gmail.com)   <sup>2</sup> [salibemartin@gmail.com](mailto:salibemartin@gmail.com)   <sup>3</sup> [elias.s.f@gmail.com](mailto:elias.s.f@gmail.com)



---

# Índice

Capítulos	Página
<b>I Definición del proyecto</b>	<b>1</b>
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	6
2.1. Introducción . . . . .	6
2.2. Componentes del Proyecto . . . . .	6
3. Selección de hardware para implementación del sistema	8
3.1. Introducción . . . . .	8
3.2. Componentes de hardware . . . . .	8
3.2.1. Interfaz de red - Materiales disponibles . . . . .	9
3.2.2. Interfaz de usuario . . . . .	9
3.2.3. Microcontroladores disponibles . . . . .	9
<b>II Desarrollo de software y redes TCP/IP, Interfaz PC- usuario</b>	<b>13</b>
4. Redes	14
4.1. Introducción . . . . .	14
4.2. Redes de Área local . . . . .	14
4.3. Modelo de capas . . . . .	15
4.3.1. Modelo OSI . . . . .	15
4.4. Modelo TCP/IP . . . . .	17
4.4.1. Capa 1 - Capa de Host a red . . . . .	18
4.4.2. Capa 2 - Interred . . . . .	18
4.4.3. Capa 3 - Transporte . . . . .	18
4.4.4. Capa 4 - Aplicación . . . . .	18
4.5. OSI vs TCP/IP . . . . .	19
4.6. Protocolo IP . . . . .	19
4.6.1. Direcciones IP . . . . .	20
4.7. Protocolo DHCP . . . . .	21
4.8. Protocolo TCP . . . . .	21
4.8.1. Servicio TCP . . . . .	22
4.9. Sniffer de red -WireShark . . . . .	23

<b>5. Interfaz De usuario</b>	<b>24</b>
5.1. Introducción . . . . .	24
5.2. Interfaz de Usuario . . . . .	24
5.2.1. Orbitron . . . . .	24
5.2.2. Stellarium . . . . .	25
5.2.3. Celestia . . . . .	26
5.2.4. Pass . . . . .	26
5.2.5. Gpredict . . . . .	27
5.3. Comparativa de interfaces y selección del software . . . . .	27
5.4. Software Gpredict . . . . .	27
5.4.1. Libreria Hamlib . . . . .	28
5.4.2. Selección de satélites y creación del perfil . . . . .	29
5.4.3. Configuración del rotador en Gpredict . . . . .	30
5.5. Software Stellarium . . . . .	33
5.5.1. Configuración de la red en Stellarium . . . . .	33
5.5.2. Protocolo de comunicación Stellarium . . . . .	35
<b>6. Software del Microcontrolador</b>	<b>37</b>
6.1. Introducción . . . . .	37
6.2. Diagrama del sistema . . . . .	37
6.3. Esquema eléctrico de los componentes . . . . .	38
6.4. Diagrama general del software . . . . .	42
6.5. Función de autocalibración . . . . .	44
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 . . . . .	54
6.7. Programación de Scheduler o planificador . . . . .	54
6.7.1. Funcionamiento base de tiempo para scheduler . . . . .	55
6.7.2. Clear Timer on Compare Match (CTC) Mode . . . . .	56
6.7.3. Programación del Software scheduler . . . . .	58
6.7.4. Resultados del software de planificación . . . . .	60
6.8. Conexión del software con Gpredict y Stellarium . . . . .	60
6.8.1. Conexión a la red mediante DHCP . . . . .	61
6.8.2. Configuración de Gpredict - Programación de la comunicación . . . . .	62
6.8.3. Programación del microcontrolador . . . . .	64
6.8.4. Stellarium . . . . .	65
6.8.5. Programación para conectarse con Stellarium . . . . .	66
6.8.6. Resultados de la conexión con Gpredict y Stellarium . . . . .	68
6.9. Integración de partes del software . . . . .	72
6.9.1. Resultados . . . . .	72
6.10. Programación del display LCD . . . . .	72
6.11. Análisis del código realizado . . . . .	73
6.12. Conclusiones . . . . .	74

---

<b>III Sistemas de coordenadas astronómicas</b>	<b>75</b>
<b>7. Sistemas de coordenadas astronómicas</b>	<b>76</b>
7.1. introducción . . . . .	76
7.2. Esfera - Elementos fundamentales . . . . .	76
<b>IV Sistema electrónico de posicionamiento - Construcción de prototipo y resultados</b>	<b>77</b>
<b>8. construcción del manejador de los motores</b>	<b>78</b>
<b>Apéndice</b>	<b>79</b>
<b>A. Protocolos de comunicación Serie</b>	<b>80</b>
A.1. Introducción . . . . .	80
A.2. Protocolo SPI . . . . .	80
A.3. Procotolo I2C . . . . .	82
A.4. Chip ETHERNET W5100 shield . . . . .	83
A.5. Display LCD con adaptador para I2C . . . . .	84
<b>B. Programación sobre plataforma Arduino UNO</b>	<b>86</b>
B.1. PlatformIO . . . . .	86
B.2. Configuración de platformIO . . . . .	87
B.3. Carga de software dentro del microcontrolador . . . . .	88
B.4. Compilación condicional . . . . .	89
B.5. Sentencias sobre el entorno arduino . . . . .	89
B.6. Conversor Analógico Digital . . . . .	90
<b>bibliografía</b>	<b>91</b>

**Fase I**

## **Definición del proyecto**

# Posicionador para antena

## Resumen

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

### 1.1 Introducción

En el marco de la cátedra Proyecto Final de la carrera de ingeniería electrónica, de la fac. de ingeniería, perteneciente a la Universidad Nacional De La Plata, se diseña 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. Para tal fin se emplea un receptor de comunicaciones adosado a una antena parabólica en desuso para obtener 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 sí. Se describen los trabajos realizados para recuperar los motores de la antena, desarrollo del sistema electrónico y software de posicionamiento y bajada de datos satelitales. 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

El Instituto Argentino de Radioastronomía(IAR),posee dos antenas parabólicas(ver figura 1.1, donde se aprecia la antena de fondo), de 30 mts de diámetro 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 (velando por el correcto uso del espacio radioeléctrico) por satélites, y un sin fin de aplicaciones.

Para realizar esta medida de potencia, el IAR, requiere 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. En la imagen 1.1 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.

La antena cuenta con dos motores, uno por cada eje de movimiento, independientes entre sí: altura y azimut. Cada eje cuenta, además, con la medición de posición mediante un potenciómetro, con el que se realimenta al sistema de posicionamiento.

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 del primer eje de la antena



(b) Motor del segundo eje de la antena

Figura 1.2: Encoders asociados a los motores de las antenas.

dos coordenadas, y tiene que mover la antena hacia las coordenadas recibidas. Estas coordenadas que recibe, 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 programas 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 alguno de estos programas, 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 divide 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 a 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 decidir qué softwares se usarán sobre la/s computadoras de la institución, y luego programar el microcontrolador o PC que realiza el control sobre la antena.

En la tercera fase, se va a realizar una investigación sobre los sistemas de coordenadas y como se realizan las transformaciones de estas entre sí.

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, o realizando algún apuntamiento de tipo manual. Estas fases, y capítulos a lo largo del texto, se resumen en la siguiente tabla:

Fase	Capítulos	Descripción de la fase
Fase 1	capítulo 1	Definición del proyecto
	capítulo 2	Definición de los componentes que requiere el proyecto
	capítulo 3	Selección de los componentes de hardware, en base a requerimientos
Fase 2	capítulo 4	Estudio de redes de computadoras
	capítulo 5	Software sobre la PC para el usuario final.
	capítulo 6	Programación del microcontrolador, y conexión con Gpredict y Stellarium
Fase 3	capítulo 7	Sistemas de coordenadas esféricos
	capítulo 8	Implementación de los sistemas de coordenadas esféricos dentro del microcontrolador.
Fase 4	capítulo 9	Desarrollo de un controlador para los motores de la antena
	capítulo 10	Resultados y conclusiones del trabajo - Futuras versiones

Tabla 1.1: Resumen del trabajo en cada fase del proyecto

## 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: se definen aquellas cosas relacionadas al comportamiento del dispositivo a diseñar
2. Requerimientos de sistema: se refiere a como llevar a cabo la solución en sí.

La siguiente tabla, contiene una lista de ambos requerimientos:

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 Cenit 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 del sentido de movimiento y posiciones angulares.
		Bajo costo.

Tabla 1.2: Requerimientos del sistema

# Componentes para la construcción de posicionador

## resumen

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

## 2.1 Introducción

En este capítulo, se muestra que componentes, 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:

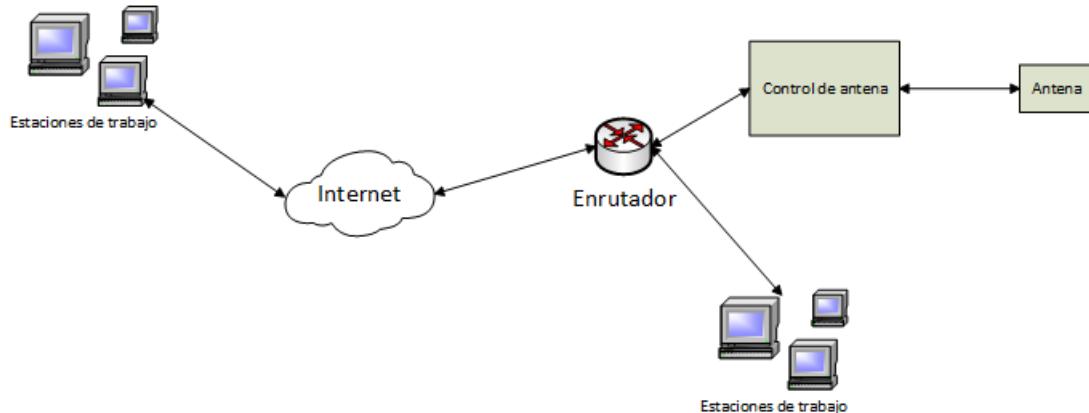


Figura 2.1: Diagrama del sistema general

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 del sistema), el sistema que debe realizar el control de la 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 comandar el movimiento de la antena.
2. Interfaz de usuario con el estado del sistema(interfaz o pantalla para mostrar el estado del sistema).
3. Interfaz de red, cableada para poder recibir órdenes 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 sí, 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

Se definen los componentes de hardware seleccionados para cumplir con los requerimientos, en particular, el microcontrolador, la interfaz de red, y la interfaz de usuario del sistema. El análisis se basa sobre componentes disponibles en el IAR, se seleccionan aquellos, que cumplen el requisito de bajo costo, y luego sobre los mismos, se analiza su facilidad de programación, librerías y módulos disponibles sobre estos, y otros aspectos.

### 3.1 Introducción

Se definen los criterios de selección del microcontrolador, además, se define la interfaz de red y la 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, o su arquitectura, 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 simultaneo, que son el ángulo de acimut 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 dos canales o puertos de entrada que posean un conversor analógico-digital cada uno, para adquirir la señal de ambos potenciómetros.

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, es 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. También, 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 (dos puertos para cada motor, y con un único puerto, se selecciona el sentido de giro).

### 3.2.1. Interfaz de red - Materiales disponibles

La placa disponible en el IAR, es la siguiente: chip Ethernet W5100. La 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 Peripheral Interface). Cabe destacar, que esta placa, solo tiene disponible el bus SPI, ya que la placa, solo viene con este protocolo. Además, esta placa, no puede compartir el bus, debido al diseño de la misma. Si desea compartir el bus, debe modificarse. 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 expuso en la sección 3.2, se va a utilizar un display LCD de 16 columnas y 2 filas, que está disponible para su uso en el Instituto Argentino de Radioastronomía.

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

La imagen, 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
2. Cantidad de pines disponibles: 4 (mínimo)

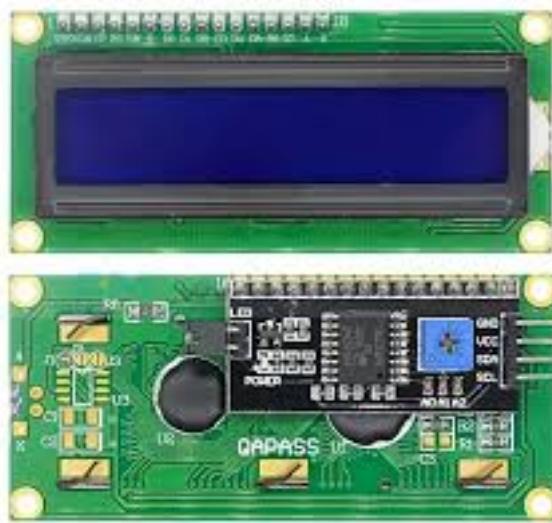


Figura 3.2: Display LCD en el IAR

3. Puerto SPI: conexionado del chip ethernet
4. Puerto I2C: para la conexión del display

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

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

Donde, de las tres, se debe 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. El estudio realizado se basa en las referencias [1]-[3]

### 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 núcleo 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 está 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, está 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

y en general 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. Está 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 están disponibles en Internet. Se basa en el microcontrolador At-Mega328P.

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 está pensada para el aprendizaje del microcontrolador STM32F100. Este microcontrolador es el núcleo 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 documentación sobre la construcción de la placa. Existe muy poca información por fuera de la documentación oficial. Existen algunos foros de sistemas embebidos, pero la discusión es muy escasa,

### 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 en el mercado local	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 falla, están disponibles en el mercado local, y esto facilita el seguimiento del proyecto en caso de algún fallo en los componentes de hardware.

## **Fase II**

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

# Redes

## resumen

Se definen los componentes básicos de una red: los modelos OSI y TCP/IP. Luego se analizan los protocolos TCP, DHCP, y el protocolo IP. Al finalizar se inspecciona el protocolo sobre una red LAN.

### 4.1 Introducción

En este capítulo se analizan 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 una. 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. También, se introduce el concepto de sniffer, y se muestra como visualizar el contenido de los datos que están circulando en una red. Los conceptos en que profundiza el presente capítulo, son aquellos de mayor interés para la construcción del dispositivo. Este capítulo se basa en un resumen de las referencias [4] y [5].

### 4.2 Redes de Área local

Existen tres tipos de redes: redes de área local, de área 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 se interconectan distintos equipos de una institución, departamento de una empresa, etc., para que se realice 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 estándar, para estas redes LAN cuando se conectan de forma inalámbrica: IEEE 802.11 (comúnmente denominado WiFi), y cuando son alámbricas existe el estándar IEEE 802.7 (comúnmente denominado ethernet). Las redes WiFi, no se utilizan en este trabajo, por utilizar técnicas de radiofrecuencia, que interfieren con los receptores de comunicaciones. En el caso de redes alámbricas, las computadoras, realizan un enlace punto a punto, a través de un dispositivo denominado **Switch**. Un switch, posee varias entradas para conectar más 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, se usa un modelo conocido como arquitectura “cliente-servidor”. En esta arquitectura, las maquinas clientes 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. 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. TCP/IP de siete capas, cinco capas respectivamente.

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.

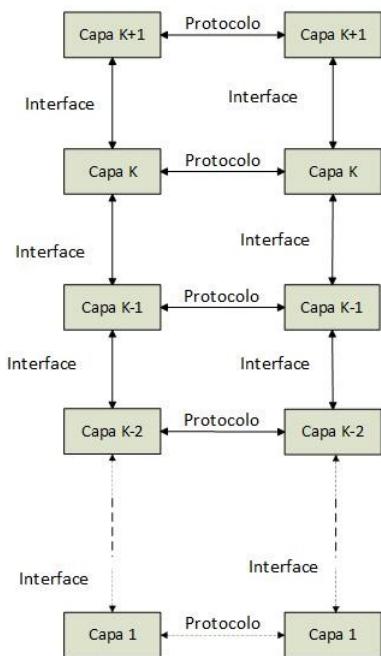


Figura 4.1: Conexión entre capas. En esta figura se observan las interfaces, protocolo y conexionado

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.

Este 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 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:

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 más facilidad a la hora de diseñar una red.

#### 4.3.1. Modelo OSI

El modelo OSI(open systems interconnection, modelo de interconexión de sistemas abiertos), se basa en una propuesta desarrollada por la organización internacional de normas (ISO), en un intento de estandarización para los protocolos. En sí, 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 cuenta con siete capas, y los principios utilizados para llegar a ellas son los siguientes([5]):

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 enviada a la 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 quién 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(Transmission Control Protocol/Internet Protocol, protocolo de control de transmisión/protocolo de Internet) es usado para conectar computadoras entre si a través de una red interna dentro de una organización. Este modelo se muestra en la siguiente figura, y se dará una breve explicación de cada capa:

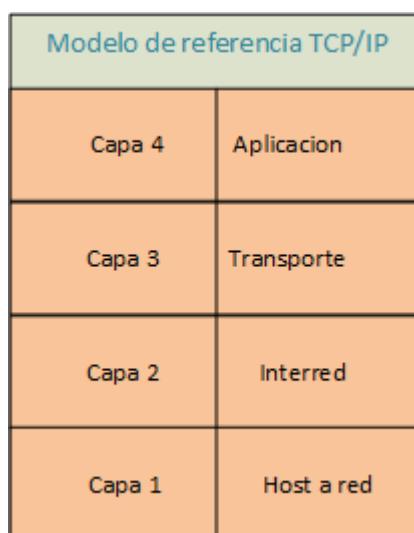


Figura 4.3: Modelo de capas TCP/IP

#### 4.4.1. Capa 1 - Capa de Host a red

La capa 1, del modelo TCP/IP, también 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 varía de una red a otra.

#### 4.4.2. Capa 2 - Interred

Su trabajo es permitir que los hosts injetan 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 (ver referencia [4]). La capa de interred, define un paquete de formato y protocolo denominado protocolo IP (Internet protocol). 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 (transmission control protocol, 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

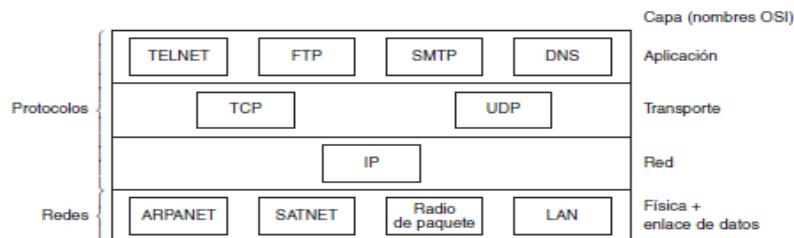


Figura 4.4: Esquema de interconexión del modelo TCP/IP

## 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. En la siguiente figura se muestra la comparación de las capas:

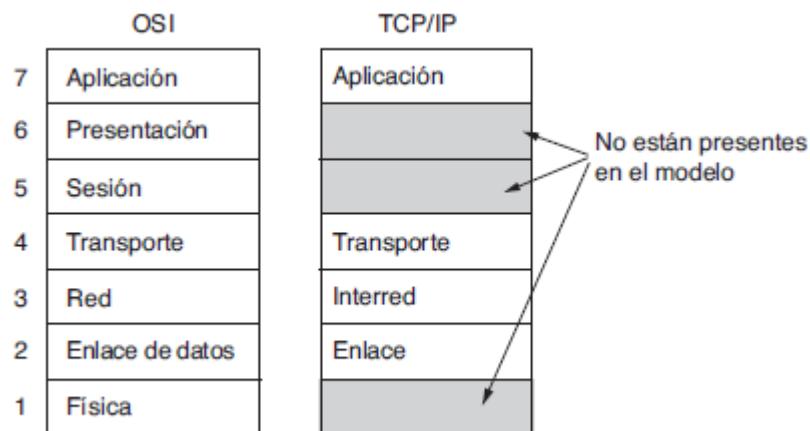


Figura 4.5: Comparativa entre modelo OSI y TCP/IP

El modelo OSI, define tres partes básicas:

1. Servicios
2. Interfaces
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.

## 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(Internet Protocol, protocolo de internet), se encuentra en el estándar RFC 791(Request for Comments<sup>1</sup> ver [6]), 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, reenvían cada paquete, a través de más 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 envía 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

<sup>1</sup> serie de publicaciones de Internet Engineering Task Force, son los encargados de redactar protocolos y normas en el uso de internet

el datagrama original. Un datagrama es la información que viaja en una red, y está 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

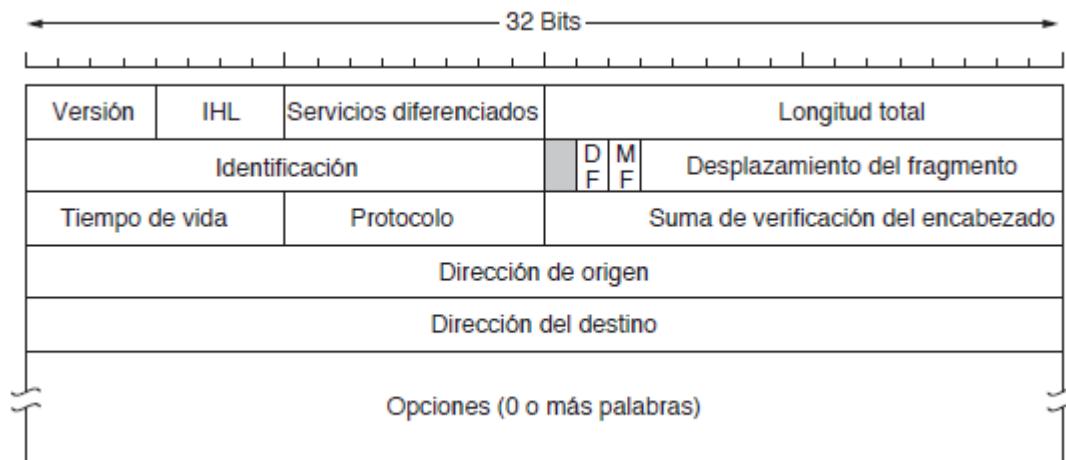


Figura 4.6: Datagrama IP

Este datagrama, se compone de dos partes: encabezado y carga útil. El encabezado, se compone de 20 bytes de longitud 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, están definidos en el estándar 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

Se observa en la figura 4.6, se observa que hay 32 bits(o 4 bytes) para indicar esta dirección. 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. 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 estándar RFC 2131[7] un protocolo de asignación dinámicas de ips, conocido como DHCP(Dynamic Host Configuration Protocol, Protocolo de Configuración Dinámica de Host). En el estándar, 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 está disponible, envía la dirección en un paquete denominado DHCPOFFER. Un paquete DHCP está conformado por los bits que se muestran en la siguiente figura:

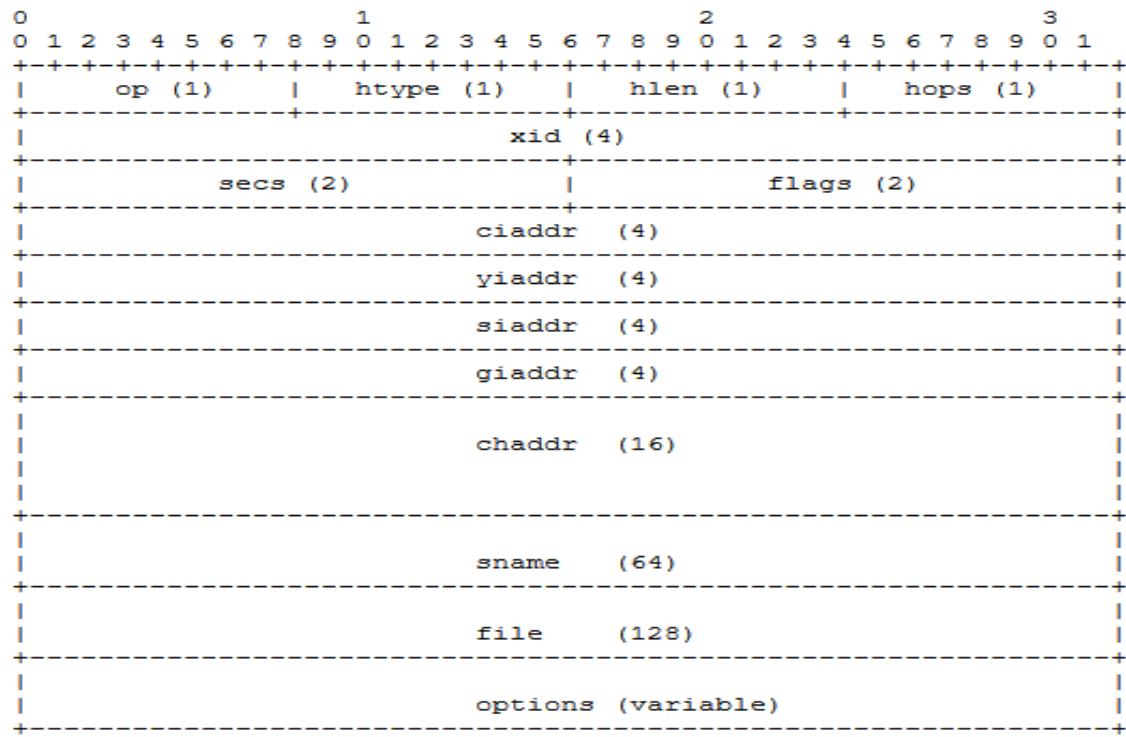


Figura 4.7: Datagrama para intercambio de mensajes del protocolo DHCP

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 estándar indicado en el primer párrafo de 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(transmission control protocol, protocolo de control de transmisión) 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(puertos). 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 estándar, por ejemplo, el puerto 80 para el protocolo HTTP. Estos puertos conocidos, van desde 0 a 1024. Los puertos desde el 1024 hasta el 49151 son puertos que se pueden registrar en IANA<sup>2</sup>, 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

<sup>2</sup> es una organización que se encarga de la numeración y asignación de nombres, y puertos únicos para cada organización que lo solicite

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 se encuentre instalado. El software elegido para realizar esta visualización de los paquetes se llama Wireshark. Se ha elegido este software por ser software libre, y se puede aplicar filtros por 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 más de una placa de red), 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.

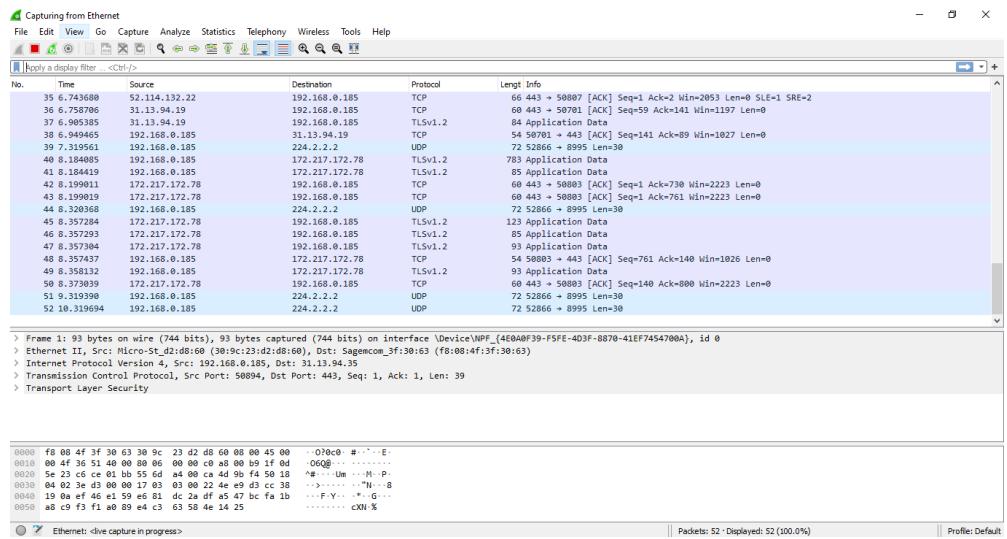


Figura 4.9: análisis de los datos dentro de una red utilizando Wireshark

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 para que realicen seguimiento de satélites. Además, se puede comprobar los protocolos implementados por los programas.

# Interfaz De usuario

## Resumen

Se definen las interfaces sobre las computadoras de la institución. Estas interfaces, se basan en software existente. Para seleccionar las interfaces, se realiza una comparación de varios programas. Luego de esto, se muestra cómo realizar la configuración de las aplicaciones seleccionados.

### 5.1 Introducción

En esta sección, analizamos los programas que son capaces de mostrar las posiciones de los satélites en tiempo real. Luego de un análisis, se seleccionan dos de estos mismos. Los programas 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 cuáles deben ser los pasos a seguir para poder configurar dichos programas.

### 5.2 Interfaz de Usuario

En 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 programas que se analizan son:

- Gpredict
- Stelarium
- Orbitron
- Celestia
- Pass

Hay que destacar, que todos estas aplicaciones, 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 7) 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.

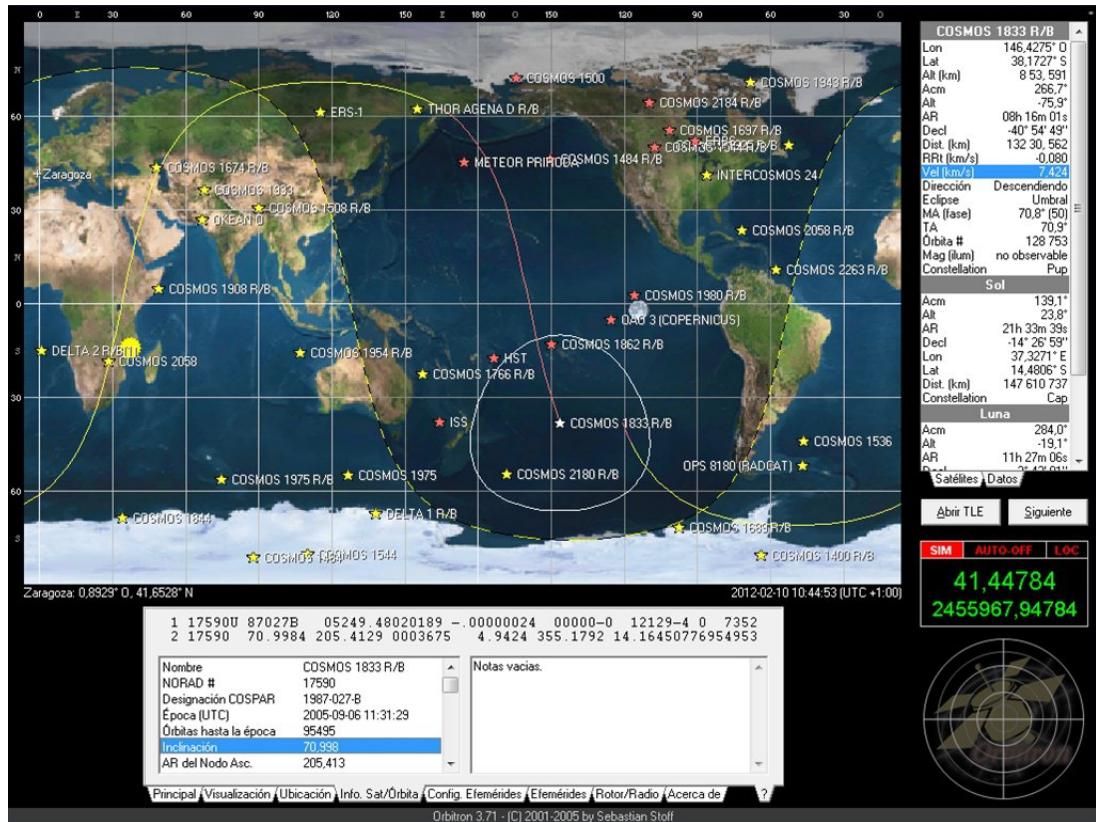


Figura 5.1: Captura de pantalla del software Orbitron

### 5.2.2. Stellarium

El software Stellarium, posee un seguimiento en tiempo real de estrellas, y está pensado para realizar el seguimiento usando telescopios. Tiene distintos módulos, lo que lo hacen personalizable. Tiene soporte para estrellas, y satélites. Estos datos, pueden enviarse a través de la red, para que los reciba un telescopio, y realice el apuntado del telescopio. Existen pluggins, de este software, que permiten enviar las coordenadas a dispositivos mediante la red. Una imagen del software puede visualizarse en la siguiente figura:



Figura 5.2: Imagen al abrir el software Stellarium

### 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.

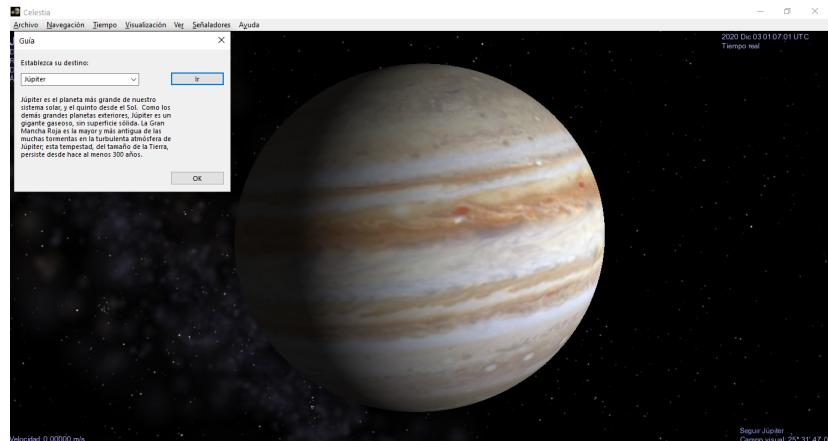


Figura 5.3: Viaje a júpiter a través de celestia

### 5.2.4. Pass

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

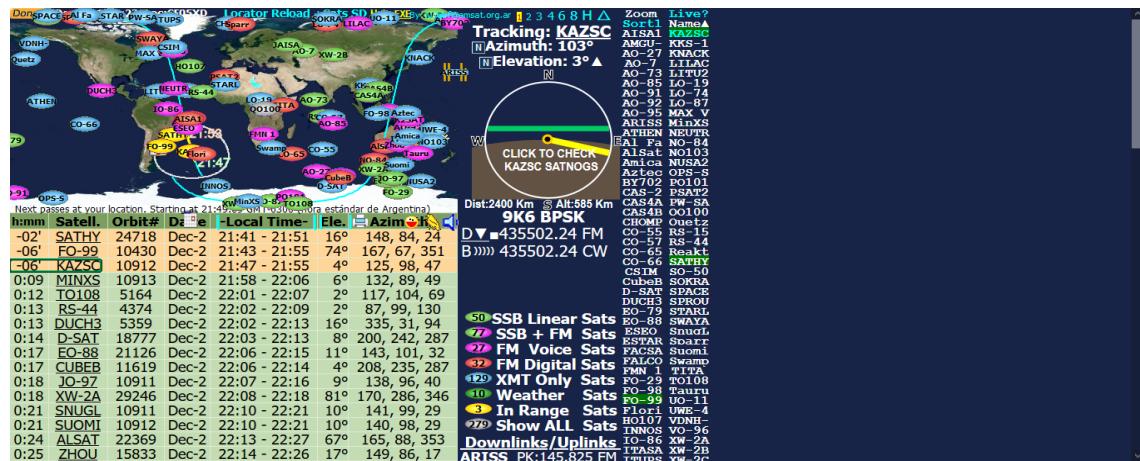


Figura 5.4: Captura de pantalla de la página <http://amsat.org.ar/pass>

Este sitio web, 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 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 su programa 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 siguiente figura:

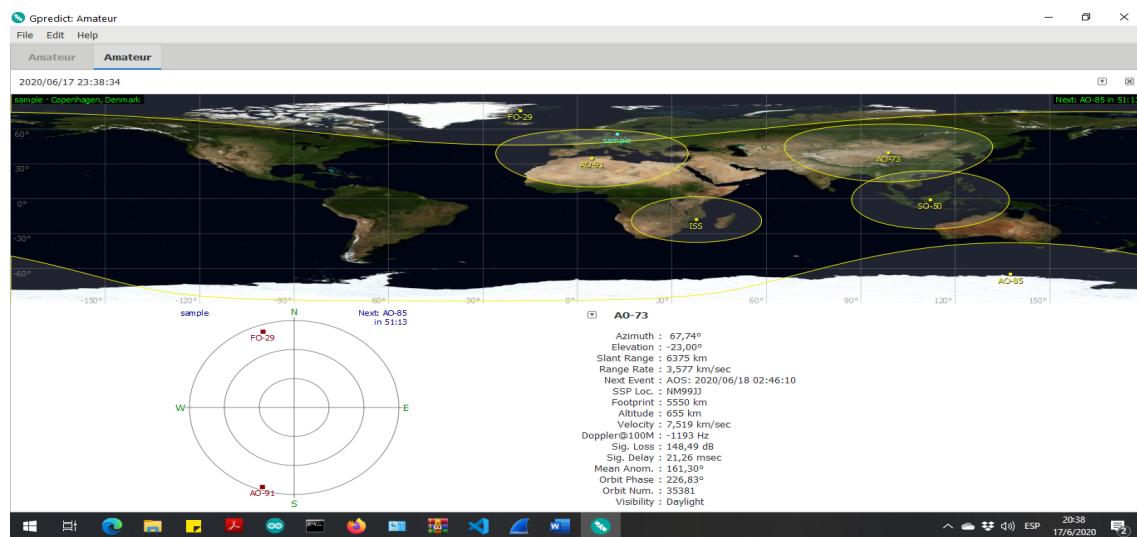


Figura 5.5: Captura de pantalla del software Gpredict

## 5.3 Comparativa de interfaces y selección del software

Una vez, realizado el análisis sobre cada aplicación, se realiza una tabla comparando sus características (ver tabla 5.1)

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

Tabla 5.1: Comparativa entre los distintos software

De la tabla, se observa que hay dos programas que pueden usarse sin ningún complemento adicional: Gpredict y Stellarium. El stellarium es un software pensado para telescopios, pero conociendo su protocolo, puede obtenerse las coordenadas para apuntar una antena, pues el software no es capaz de reconocer el tipo de dispositivo que debe 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 está 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, tiene su propio protocolo de comunicación a través de la red, además tiene funciones relacionadas al ajuste de los receptores de comunicaciones mediante la conexión a la red local.

#### 5.4.1. Libreria Hamlib

Hamlib es una capa de software dedicada al manejo de radios y rotadores<sup>1</sup> 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 aún 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 línea, o conocido como “\n” dentro del código ASCII.

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

<sup>1</sup> Un rotador es un dispositivo que es capaz de mover la antena en la dirección que se le ordene.

Metodos Get		Metodos 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(vea referencia [8]).

#### 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 “módulos”. 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 módulo, debe dirigirse a file → new module, como se muestra en la siguiente imagen:

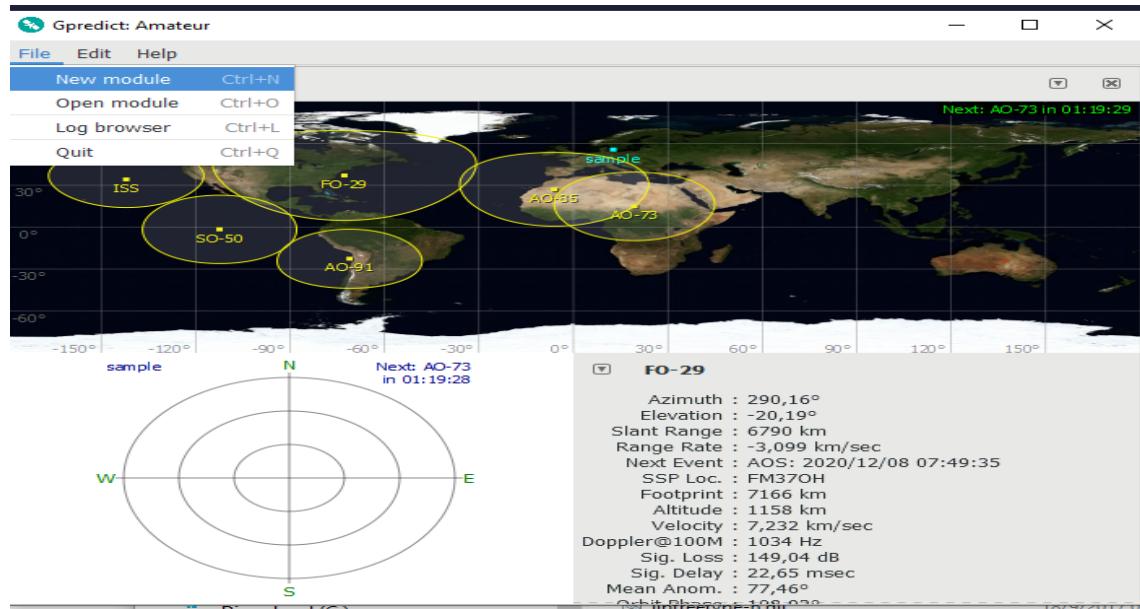


Figura 5.6: Creación de un módulo dentro de Gpredict

Al realizar esto, se abre la ventana, dónde selecciona los satélites a seguir. Esta ventana se muestra en la figura 5.7. En esta ventana, seleccionamos los satélites, y lo nombramos al módulo.

En este caso, se lo denominamos arduino.

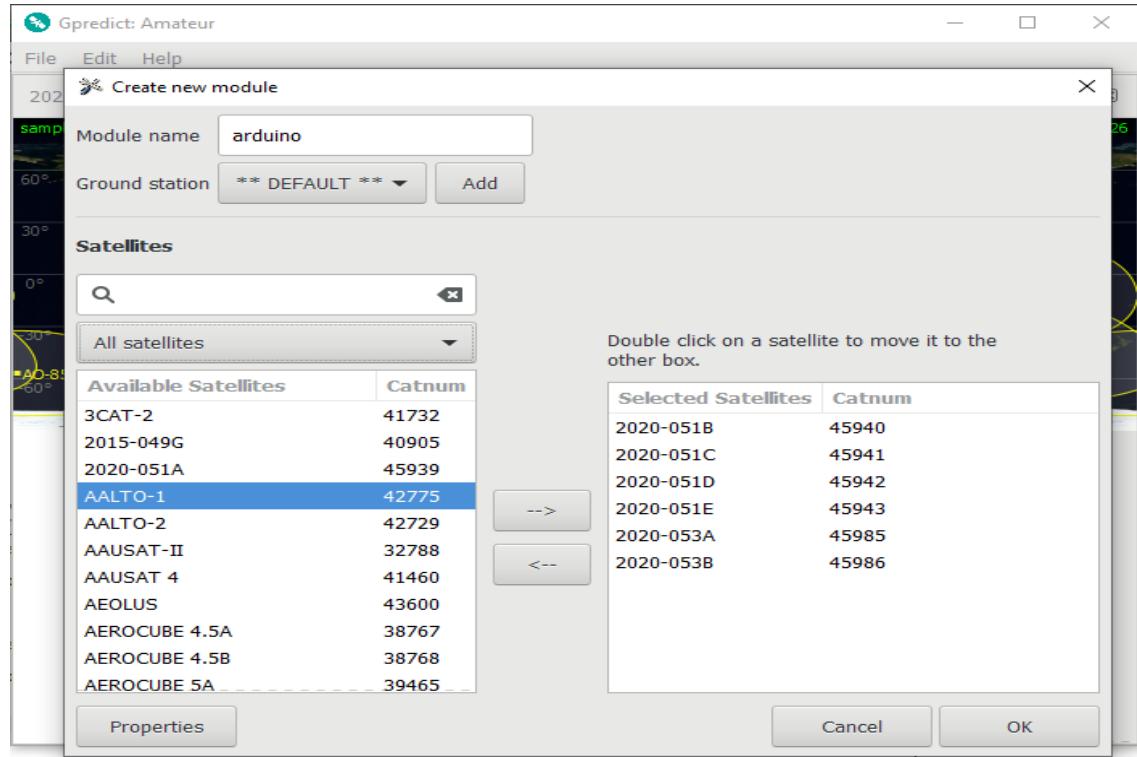


Figura 5.7: Selección de satélites para el nuevo módulo creado

Donde dice “Ground station”, se debe configurar latitud y longitud de la antena. Luego se presiona OK. Después de esto, ya se creó 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:

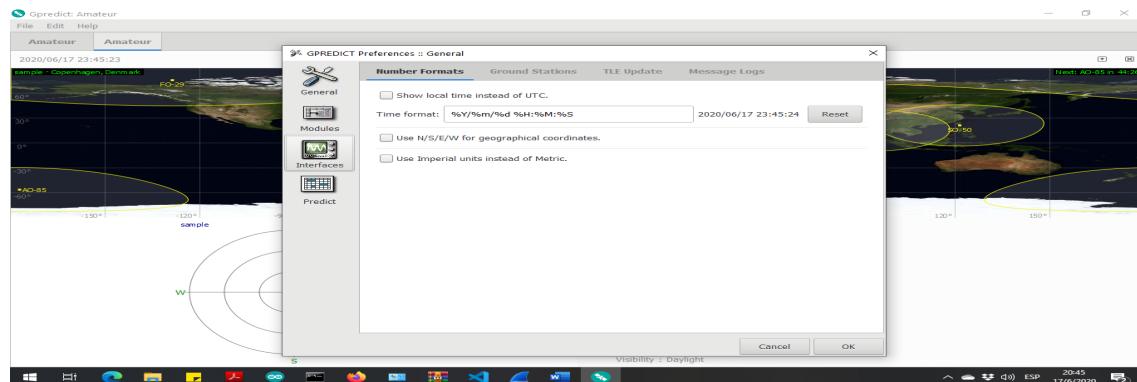


Figura 5.8: Configuración de rotador

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

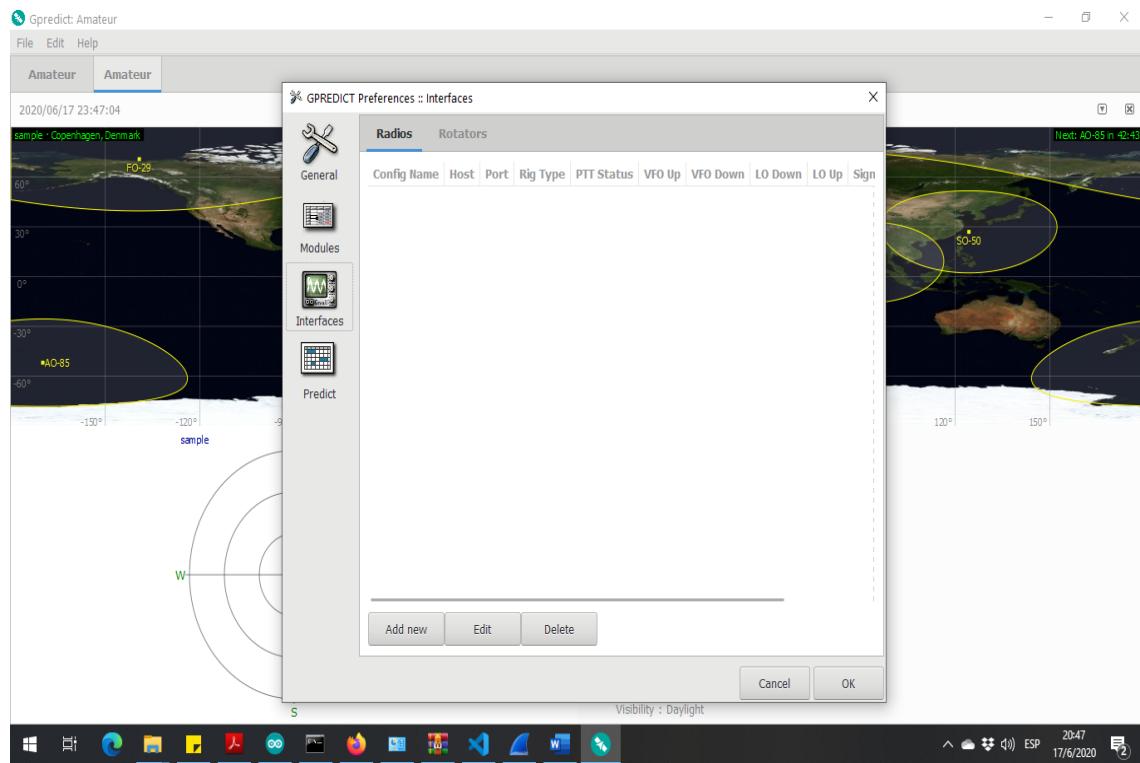


Figura 5.9: Configuración de rotador

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:

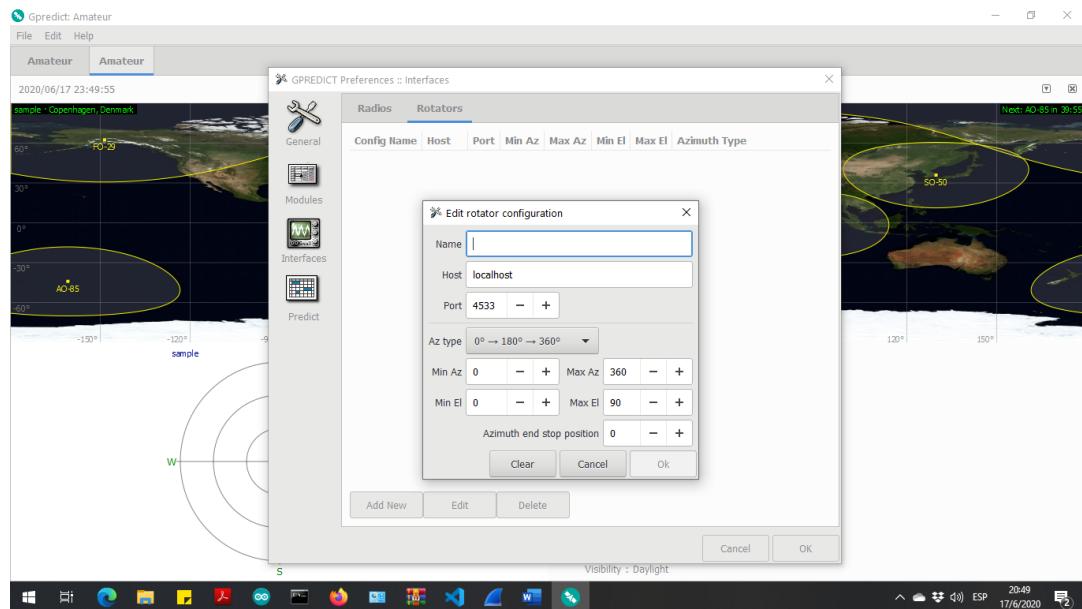


Figura 5.10: Configuración de rotador

Los parámetros a configurar son nombre del rotador(el que desee el usuario), ángulos máximos y mínimos de la antena. Estos parámetros se verán en el siguiente capítulo. 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 defecto (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 deben 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 menú, debe dirigirse a “antena control”, como se ve en la figura 5.11

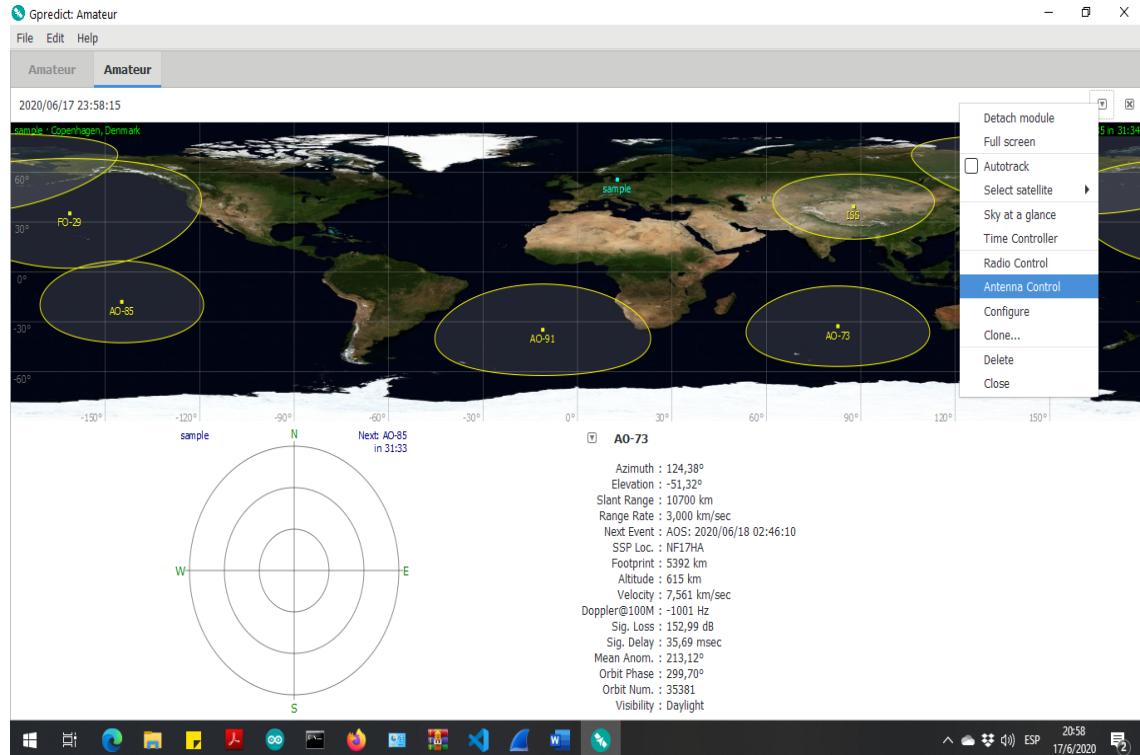


Figura 5.11: Ver el rotador o rotadores configurados

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

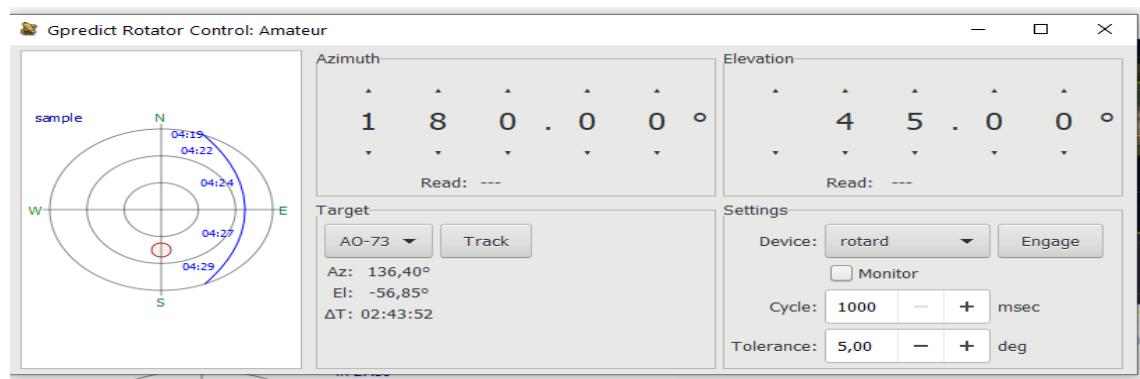


Figura 5.12: Panel de control del rotador de Gpredict

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 es horizontal-azimutal, con el cero en el polo norte. Estas coordenadas son definidas

en la fase 3, donde se muestra el análisis matemático de estas.

Las medidas angulares, corresponden a los ángulos 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 los ángulos y los límites de la misma. Esto se realiza en el siguiente capítulo, también, se muestra cómo se comunica Gpredict con el dispositivo desarrollado en el presente texto.

## 5.5 Software Stellarium

Este programa, es libre, y está 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 del 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. Esta guía ha sido basada en la referencia [9], que es el manual oficial del software.

### 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 intercambiarán 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 configuración, luego presionar en la pestaña pluggins, y en la parte izquierda seleccionar “control del telescopio”. A continuación se deja la imagen de este proceso:

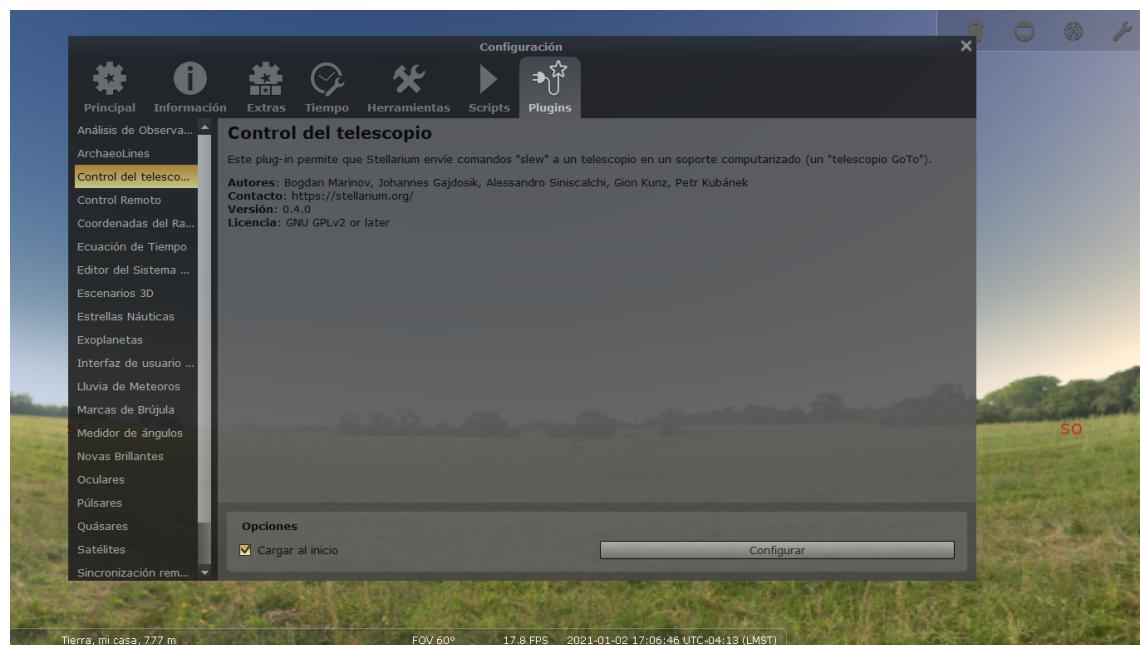


Figura 5.14: Abrir el pluggin control del telescopio en Stellarium

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

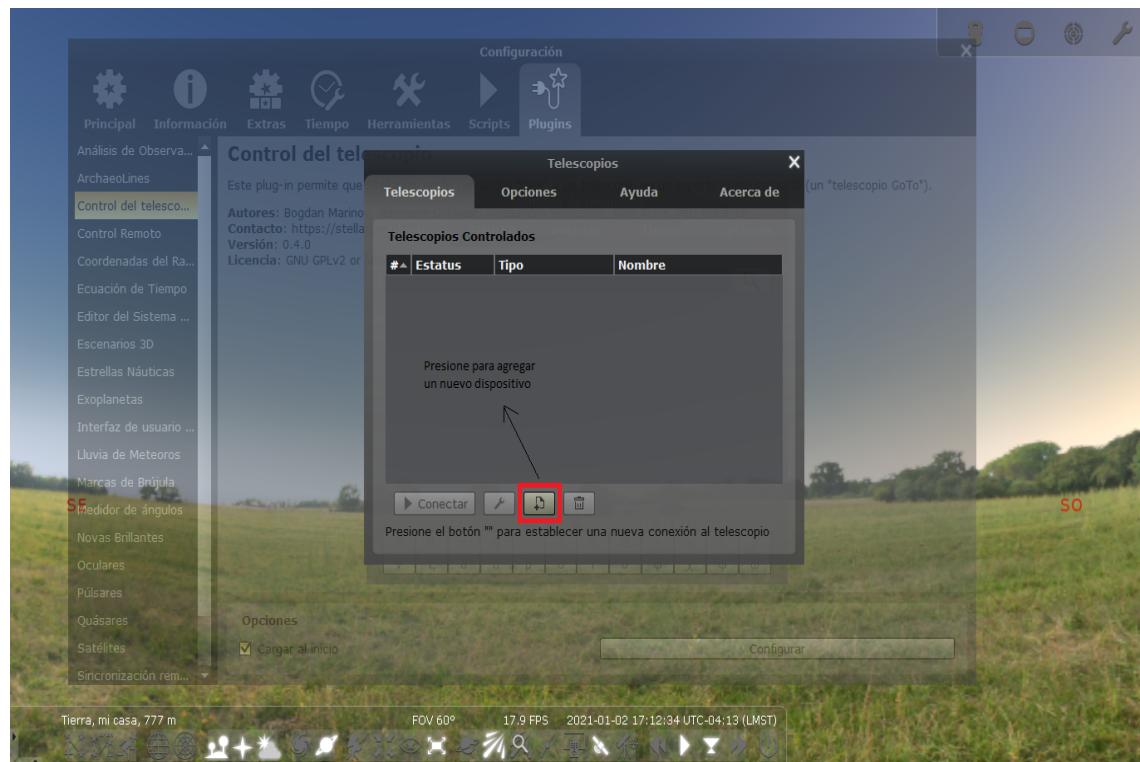


Figura 5.15: Añadir un telescopio dentro del software Stellarium

Luego, debe presionar donde muestra la imagen anterior para empezar a agregar el primer dispositivo. Se realiza un clic 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

enviará los datos.

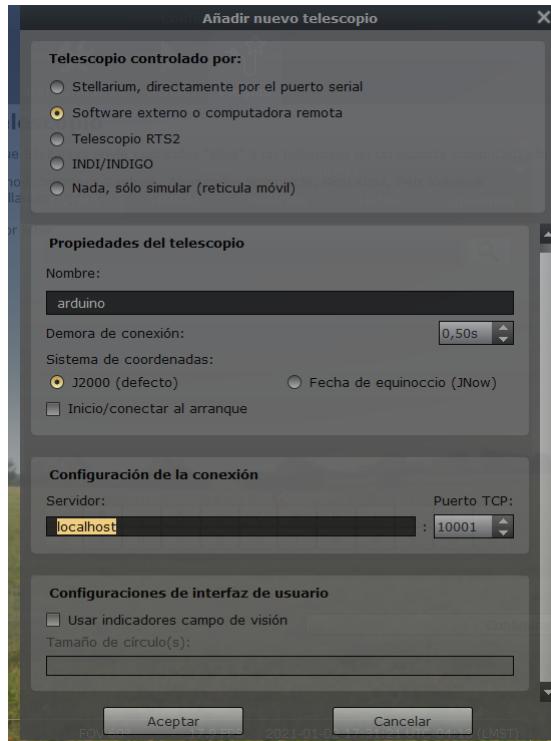


Figura 5.16: Ventana de configuración de red del software Stellarium.

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. Estas coordenadas están relacionadas con la selección de la referencia para las coordenadas ecuatoriales. Estas se discuten en el capítulo 7. Se elige J2000.

Una vez configurado el dispositivo, queda por intentar entender el protocolo, que éste emplea para comunicarse con los dispositivos. Si desea conocer más opciones de este programa, puede ver la referencia [10]

### 5.5.2. Protocolo de comunicación Stellarium

El stellarium, no es capaz de recibir datos en este modo de configuración (al menos, por lo investigado hasta el momento), pero si es capaz de enviar datos al dispositivo conectado en la red. Este emplea 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 están 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) - ascensión 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. Big-Endian y little Endian, se refiere al modo de leer los datos dentro de un registro o posición de memoria dentro de un microprocesador. Esto es similar a los lenguajes, que se leen de izquierda a derecha o de derecha a izquierda. Por ejemplo, suponga el número binario 0111 0000, en el sentido usual, se lee en hexadecimal como 70. Este caso se denomina Little-Endian. En caso de que se represente como Big-Endian se escribe como 07 en hexadecimal. Dada esta situación, los bits que llegan al microcontrolador, se deben reorganizar para que se interpreten correctamente.

# Software del Microcontrolador

## Resumen

Se describe el proceso realizado para programar el software para el microcontrolador ATmega328P, bajo el entorno arduino, y sus librerías. Además, se muestra cómo deben interconectarse los componentes, y los circuitos correspondientes para realizar las pruebas sobre cada parte del software. Cada parte del software desarrollado, se realizan las verificaciones, y se muestran los resultados obtenidos. Una vez, realizadas todas las pruebas, se procede a integrar todo el software, y se realiza una prueba final, y se verifica la funcionalidad. La versión de software presentada en este capítulo, es una parte del desarrollo. En la siguiente fase del proyecto, se implementa e integra la parte restante del mismo (fase 3).

## 6.1 Introducción

El software, requiere del hardware seleccionado en el capítulo 3. Este hardware, debe interconectarse entre sí antes de empezar a programarse sobre el microcontrolador. Una vez realizado este proceso, se desarrolla el software en base a los requerimientos de 1.2. Para la programación, se divide el trabajo en el siguiente orden:

1. Autocalibración: necesaria para que el software sea capaz de reconocer la orientación de la antena.
2. Control de posición: requiere que se realice la autocalibración correctamente. Esta función, decide los movimientos de giro de la antena en base a la autocalibración.
3. Scheduler o planificación: necesaria para realizar el control, a tiempo controlador, y no se utilice un esquema de polling.
4. Conexión del dispositivo a la red. Conectarse con Gpredict y stellarium: en esta parte del desarrollo, se realiza la programación de los protocolos implementados por cada uno de estos programas. Cada programa tiene protocolos distintos.
5. Display LCD: se utiliza para mostrar información.

Luego del desarrollo de cada parte del software, se procede a su integración, y análisis del código, utilizando herramientas provistas por el fabricante del chip ATmega328P (microchip).

## 6.2 Diagrama del sistema

En base a los componentes seleccionados, en el capítulo 3, estos deben interconectarse entre sí, 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 I2C (descripto en el apéndice A). El bloque de drivers de motores, son dos controladores de motores, y dos encoders, pero se ha puesto una único bloque, ya que el sistema de control es el mismo en ambos motores.

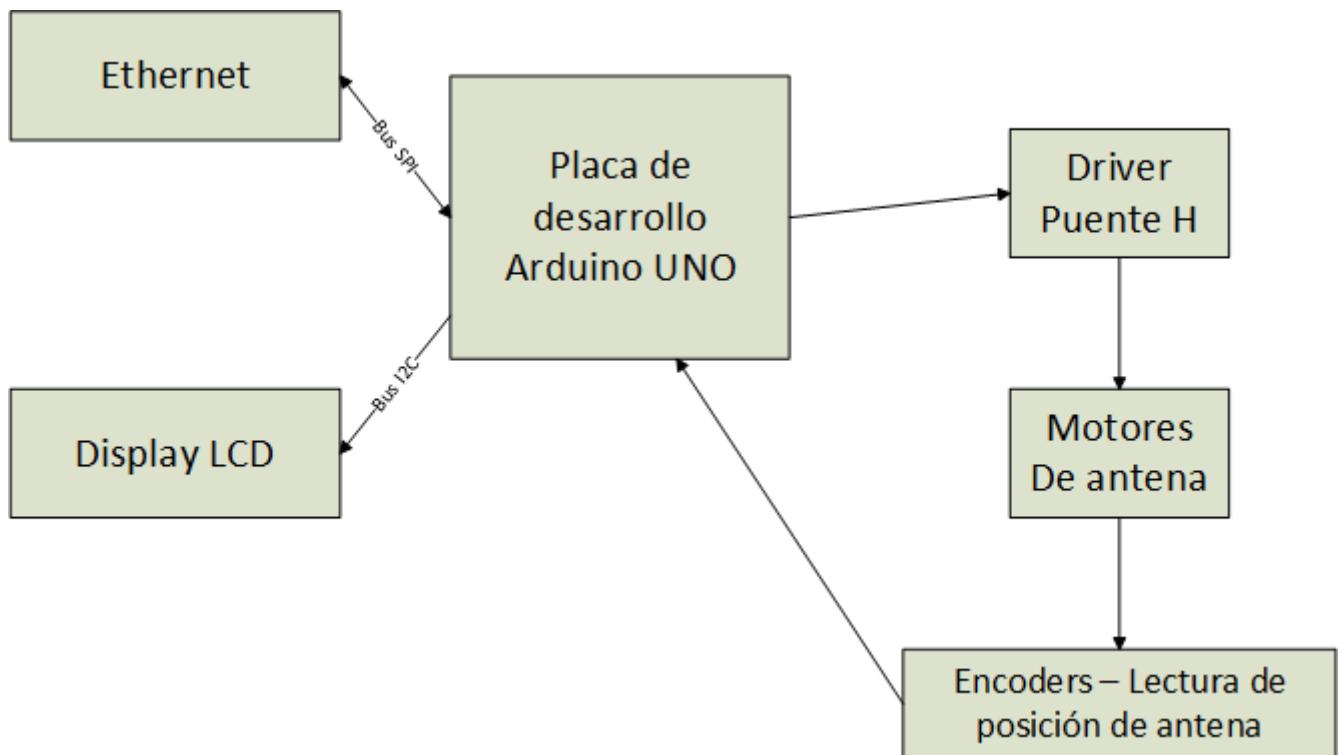


Figura 6.1: Diagrama general del sistema de control

### 6.3 Esquema eléctrico de los componentes

El lenguaje de programación del entorno Arduino es C/C++, la programación se desarrolla en C, mientras, el uso de librerías externas requiere el uso de C++. Estas librerías son la librería Ethernet, display LCD, y la librería para el manejo del puerto serie. El software, debe comunicarse con el display LCD(ver figura 3.2) mediante el protocolo I2C 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 sí, 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 a la placa de desarrollo. Los puertos SPI físicos del de la placa Arduino UNO son los pines 11,12 y 13. El puerto de Slave Select se selecciona mediante software. El chip ethernet W5100, dispone de cinco puertos para conectarse a un microcontrolador. Estos son, el bus SPI(incluido el slave select), y el de reset. El puerto de Slave Select y reset, pueden conectarse a cualquier puerto del microcontrolador. Los únicos puertos que no pueden cambiarse son aquellos relacionados a las señales del bus SPI. Por ende, en el microcontrolador, los puertos 11,12, y 13, se conectan al chip ethernet W5100, sin opción a cambiarse.

La conexión del display 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 por cada motor). Estos, deben poseer modulación por ancho de pulso, para poder realizar un control de

velocidad en próximos 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.

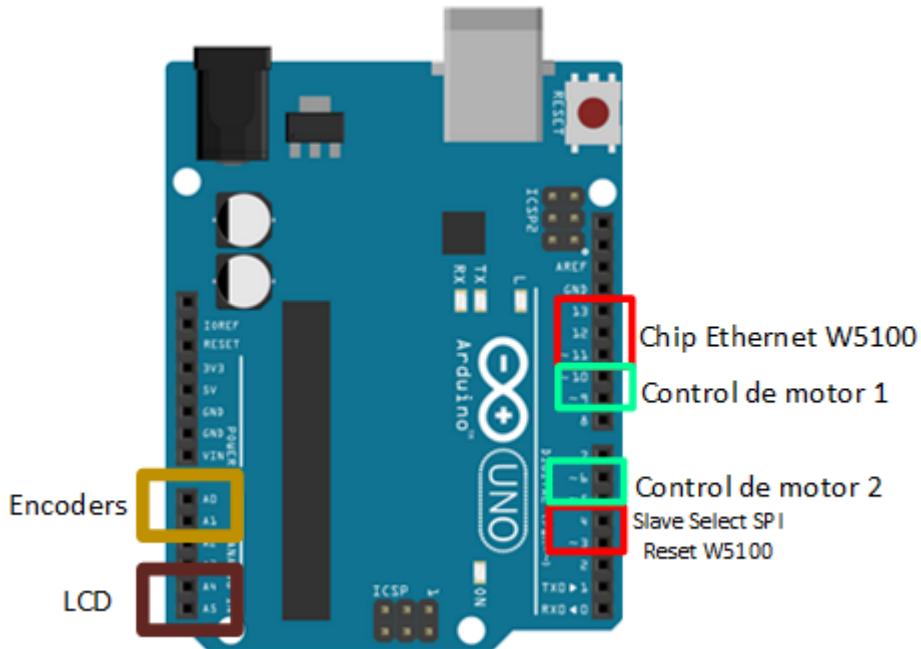
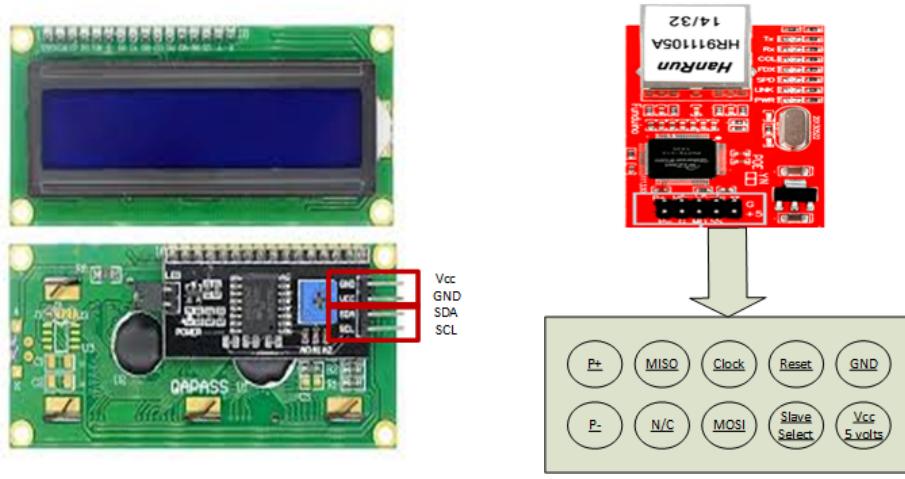


Figura 6.2: Pines seleccionados sobre la placa de desarrollo Arduino UNO para realizar el prototipo

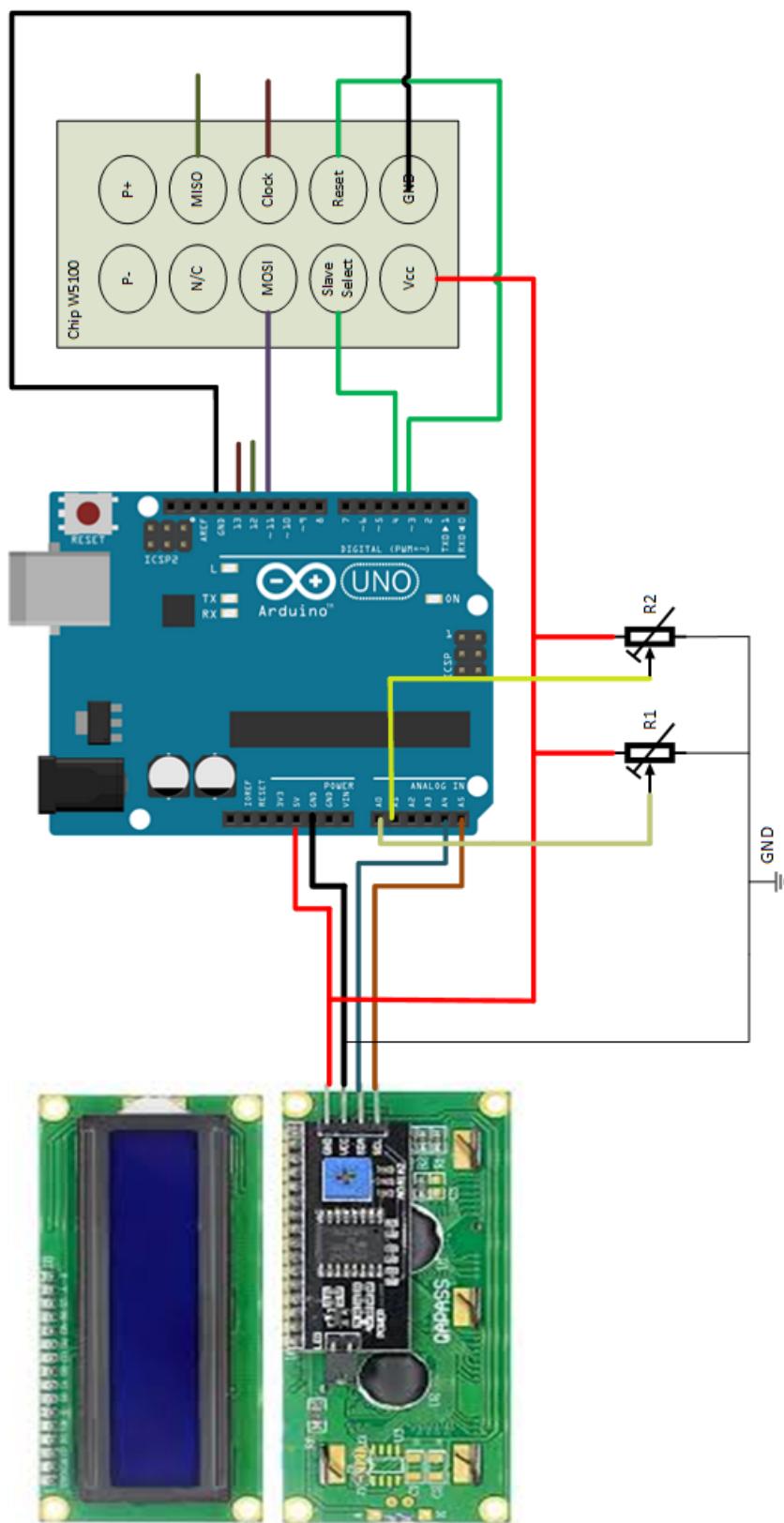
Una vez, definidos los puertos a utilizar, se deben conectar los componentes a la placa de desarrollo. Esta conexión, se realiza usando cables denominados “dupont” en una protoboard. Para saber cómo se deben conectar el display LCD y el ethernet Shield W5100, se deben conocer su disposición de pines,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



(a) Pinout Display LCD

(b) Pinout chip ethernet W5100

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 debe conocer cuáles 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([1]), se obtiene que los pines de la placa de desarrollo para conectar los dispositivos es la siguiente (ver figura 6.4):

- pines del microcontrolador para conectarse con Chip W5100:
  - Pin 13 SCK
  - Pin 12 MISO
  - Pin 11 MOSI
  - Pin 4 Slave Select(SS)
  - Pin 3 Reset

- Puertos para conectarse con el Display LCD:
  - Pin A5 SCL
  - Pin A4 SDA

Por último, 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, cada uno de los led, tiene 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 6.4.

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 6.4.

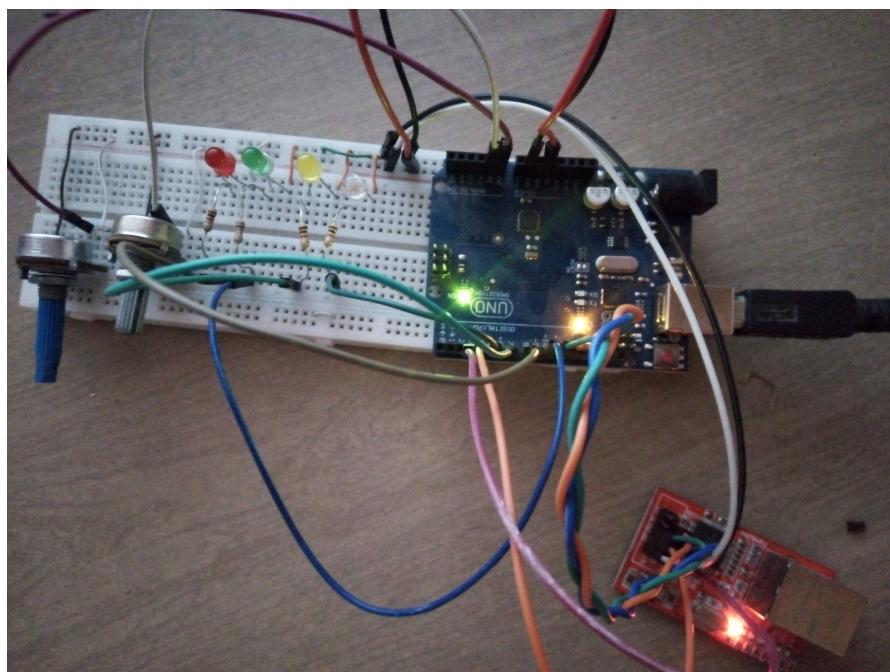


Figura 6.5: Imagén del protoboard armado para realizar las primeras pruebas con el software.

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

- led rojo: puerto 9
- led verde: puerto 10
- led amarillo: puerto 6
- 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, debe 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, debe tener la capacidad de volver a la posición de equilibrio de la antena. Esta posición de equilibrio se denomina cenit. Esto, es un requerimiento

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 en 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 está 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 realice 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 su posición. 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 ángulo de azimut y altura respectivamente, en astronomía de posición, y por eso se le dio ese nombre. En el capítulo 7 hay una descripción con más detalle de estas coordenadas.

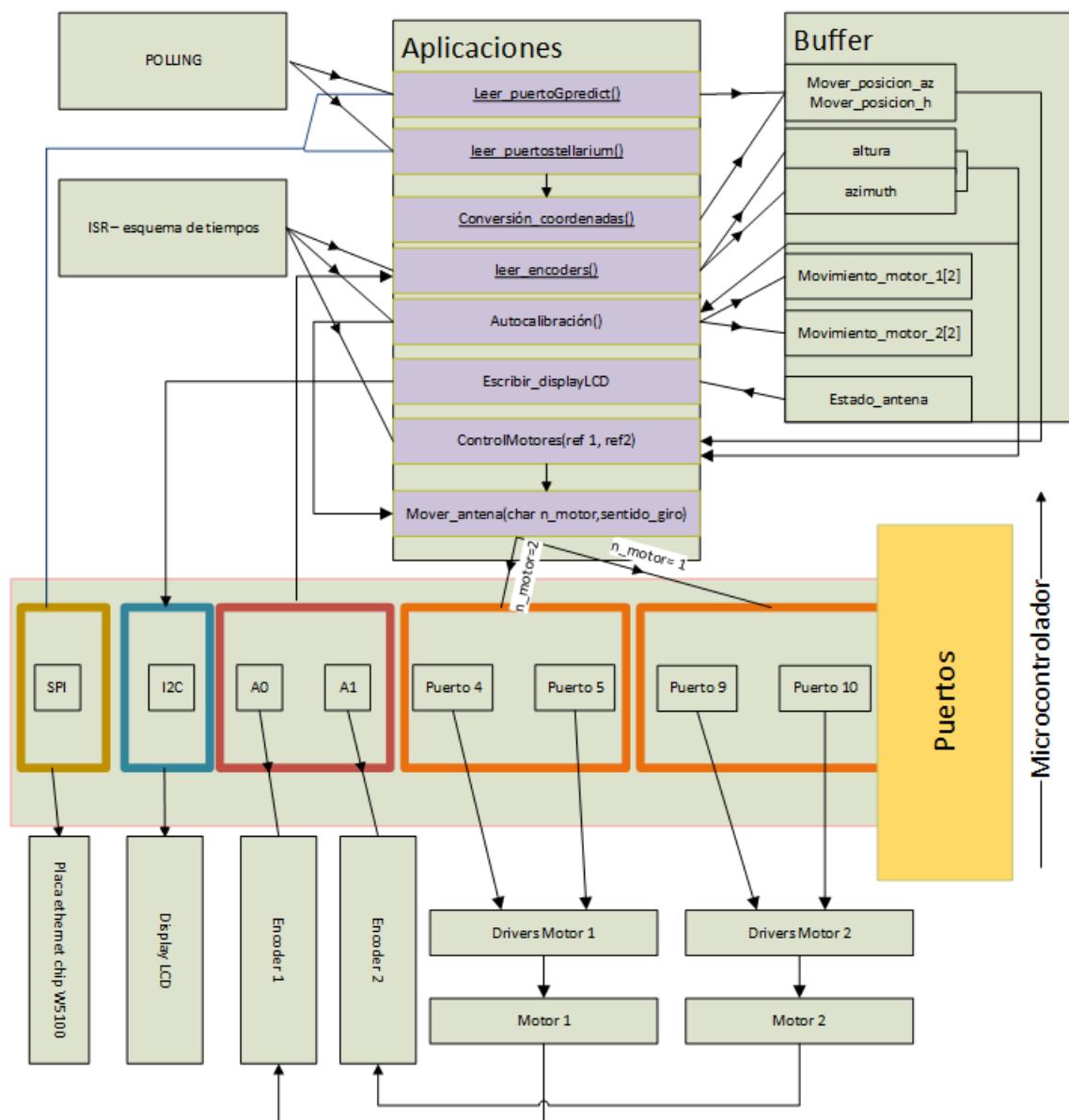


Figura 6.6: Diagrama general con las partes de software y hardware

En la figura se observan dos recuadros, 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 uno de los programas de la PC(Gpredict, o Stellarium) quiere realizar un movimiento sobre la antena.

El cuadro donde dice “Buffer”, son las variables globales que interactúan con las funciones del 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 acimut, sentido Oeste - Este
- movimiento\_motor\_1[1] : ídem, 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] : ídem, 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 y funciones auxiliares que sirven de soporte al programa general. Estas no se muestran 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. Después se continúa 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 teoría 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 que se desarrolla. Se supone que el lector posee conocimientos básicos de programación en C/C++, y cómo 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, mostrados en la figura 6.2. El archivo contiene las sentencias mostradas en el código 1.

Donde las variables TIMERS\_CLOCKS y DEBUG(resaltadas en el código 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 está conectado a la fuente de tensión, 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 la tensión, y esta tensión es la que se mide desde el microcontrolador. En la figura 6.7 se encuentra la imagen de un potenciómetro.

```

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 azimut
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 azimut
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 // depuración de aplicaciones usando puerto serie .
21

```

codigo 1: definición de los puertos del microcontrolador. El nombre del archivo es “pinout\_ard\_uno.h”.

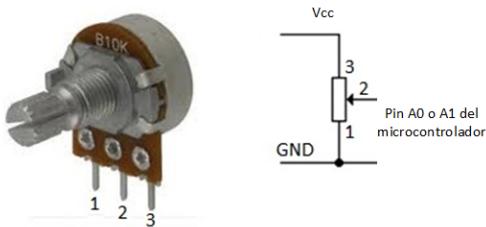


Figura 6.7: Vista de un potenciómetro, y sus respectivas conexiones de tensión, y con el microcontrolador.

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 apendice 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, la función de autocalibración se dará cuenta, cual es el sentido en que se está moviendo la antena(esta definición debe realizarse en ambos ejes de la antena). Esta, se da cuenta revisando si la tensión de entrada, está 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 figura 6.8.

La antena, solo tiene movilidad ESTE- OESTE, apuntando hacia el sur. Por este motivo, se define que la mínima tensión este dada en el oeste, y la máxima tensión en el este, donde se define  $0^\circ$  en el OESTE, y  $180^\circ$  en el ESTE, en sentido antihorario. En sentido del eje horizontal, definimos la mínima tensión, a  $90^\circ$  respecto al piso, y máxima tensión cuando la antena se encuentre a  $0^\circ$  del plano del suelo. Se usa 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.

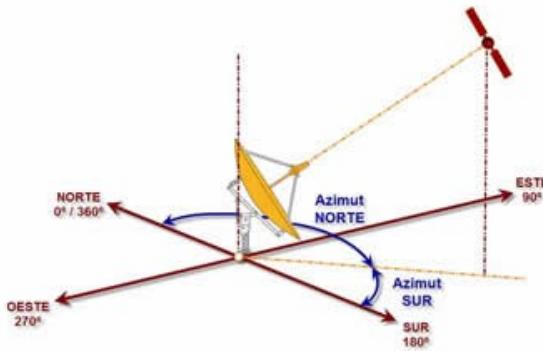


Figura 6.8: Movimientos que puede realizar la antena usando sus dos motores.

Se observa que el código 1, 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 tabla 6.1:

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

Tabla 6.1: Definición del sistema de coordenadas para la antena

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

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 idénticas
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 qué 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 realiza la función que obtiene los datos de la posición actual, desde la lectura de los potenciómetros, y las guarde en las variables azimut y altura, respectivamente. Para esta función, se han creado dos archivos: uno denominado “lectura\_encoders.cpp” y otro denominado “lectura\_encoders.h”. 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”. Los archivos se encuentran en el CD anexo al presente informe.

La única función definida en este archivo es `leer_encoders()`, cuyo propósito es guardar el valor leído por el conversor analógico digital, cuando se está autocalibrando. Si no se está autocalibrando,

deberá devolver las coordenadas correspondientes. Esta parte, se encuentra desarrollada en la siguiente sección. Ambas funcionalidades, se encuentran desarrolladas en el apéndice del presente capítulo.

Una vez realizado el proceso de lectura de los encoders por parte del microcontrolador, ahora resta la función de autocalibración. Esta esta divida 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 azimut si `n_motor` 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, ídem 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, char sentido)`,se observa en el apéndice del presente capítulo.

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 últimos cuatro valores leídos del encoder. En `ult4ad[0]` y `ult4ad[1]` guarda los últimos dos valores del ángulo de azimut, y en `ult4ad[2]` y `ult4ad[3]` guarda los dos últimos valores del ángulo 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 azimut, y el otro al valor de sentido del motor de altura
- `calibracion_encoders[4]`: guarda los valores máximos y mínimos leídos por los encoders. `calibracion_encoders[0]` y `calibracion_encoders[1]` corresponden a los valores del encoder correspondiente al ángulo de azimut. Los otros dos, corresponden al valor del ángulo de altura.

El código desarrollado para las funciones complementarias, se halla en el CD que se encuentra anexado al presente informe. El diagrama de flujo del funcionamiento de la función de autocalibración, y sus funciones auxiliares, se muestra en la figura 6.9. La función `assignar_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.

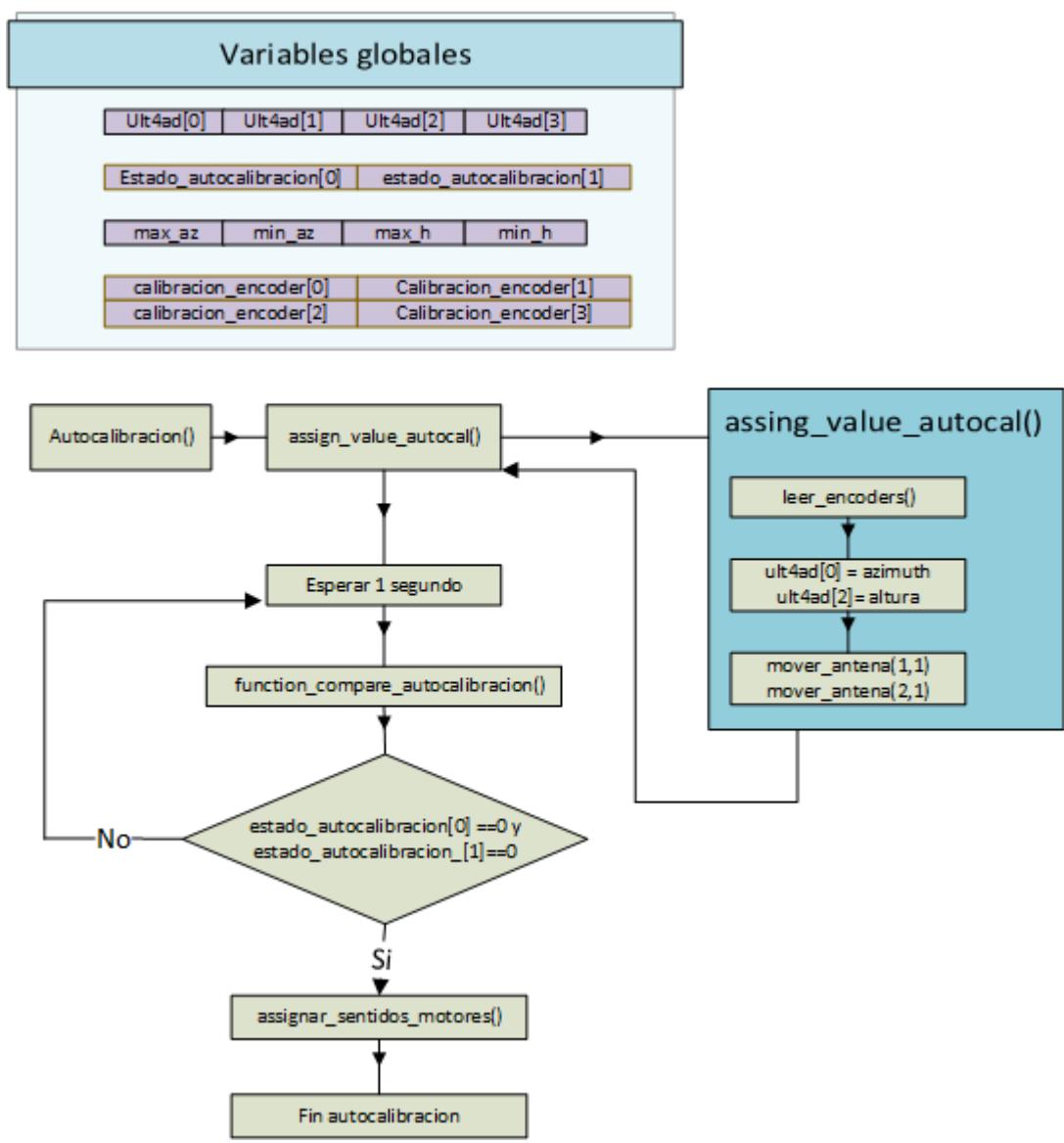


Figura 6.9: Diagrama de flujo de la función de autocalibración.

La idea del algoritmo es la siguiente:

1. Llamar a la función `assign_value_autocal()`. Esta lee los encoders, y pone sentido = 1 en ambos motores. asigna los valores de `calibracion_encoders[0]` y `calibracion_encoders[1]` en uno.
2. Luego espera un segundo, este proceso es necesario, para darle tiempo a la antena de realizar algún movimiento.
3. LLama a la función de comparación `function_compare_autocalibration()`. 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. `asignar_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. Así, sigue hasta que los últimos dos valores son iguales. Luego, la función de autocalibración, posee el siguiente código.

```

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 2: Código de la función de autocalibración. Esta definido en el archivo “control\_motores.cpp”

### 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 6.5, y las conexiones del potenciómetro y los diodos leds, se encuentran debajo de la imagen.

Para realizar la prueba del programa, se coloca el potenciómetro en una posición inicial, y luego se gira el potenciómetro. Hay tres posiciones iniciales, que empiece en uno de los dos extremos, o en un punto medio. Si empiezan en un extremo, solo se debería girar una vez hacia el 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 [2](#). 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.

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.

Al analizar la tabla anterior, se observa, que la función de autocalibración, responde correctamente para ambos motores. Por ejemplo, la primera 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(véase tabla [6.1](#)). 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 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 adosado 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(ver figura [6.8](#)).

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.

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 giran 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 registraron cada 1 milisegundo. Los resultados fueron los siguientes:

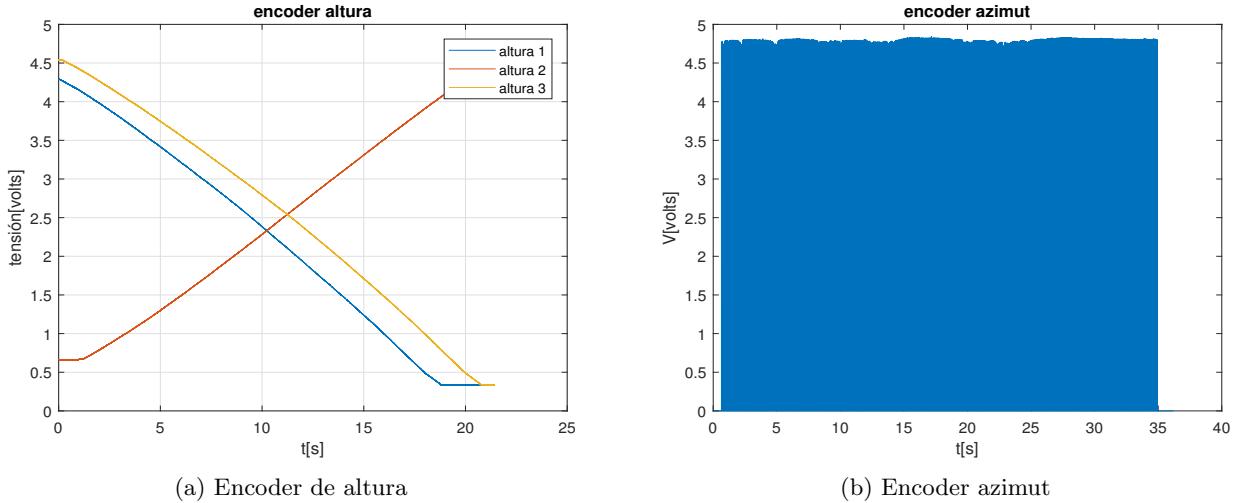


Figura 6.10: Medidas realizadas sobre los encoders de la antena en función del tiempo.

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 azimut, 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 azimut. A partir de estos puntos, obtenemos la relación entre la tensión leída y los ángulos. 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 azimut, se tienen los puntos

$$p_1 = (0^\circ, \min\{calibracion\_encoders[0], calibracion\_encoders[1]\})$$

$$p_2 = (180^\circ, \max\{calibracion\_encoders[0], calibracion\_encoders[1]\})$$

Al tener dos puntos, se puede armar la ecuación de una recta, en términos de  $\theta_{az} = f(AD_0)$ , siendo  $\theta_{az}$  el ángulo de azimut. 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:

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

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

$$y_1 = \min\{\text{calibracion\_encoders}[0], \text{calibracion\_encoders}[1]\}$$

$$y_2 = \max\{\text{calibracion\_encoders}[0], \text{calibracion\_encoders}[1]\}$$

$$\Delta A_d = y_2 - y_1$$

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  $\theta_h$  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

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

$$y_1 = \max\{\text{calibracion\_encoders}[0], \text{calibracion\_encoders}[1]\}$$

$$y_2 = \min\{\text{calibracion\_encoders}[0], \text{calibracion\_encoders}[1]\}$$

$$\Delta A_d = y_2 - y_1$$

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:

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

Cabe destacar, qué como magnitud del error, se ha utilizado la resolución, para realizar pruebas. En realidad, se debe tener en cuenta el ángulo sólido de la antena para realizar el control. Esta discusión, sobre el ángulo sólido de una antena parabólica, rebasa el alcance del presente trabajo. 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 próximas versiones, se va a cambiar el bloque “controlONOFF” por un controlador denominado PID.

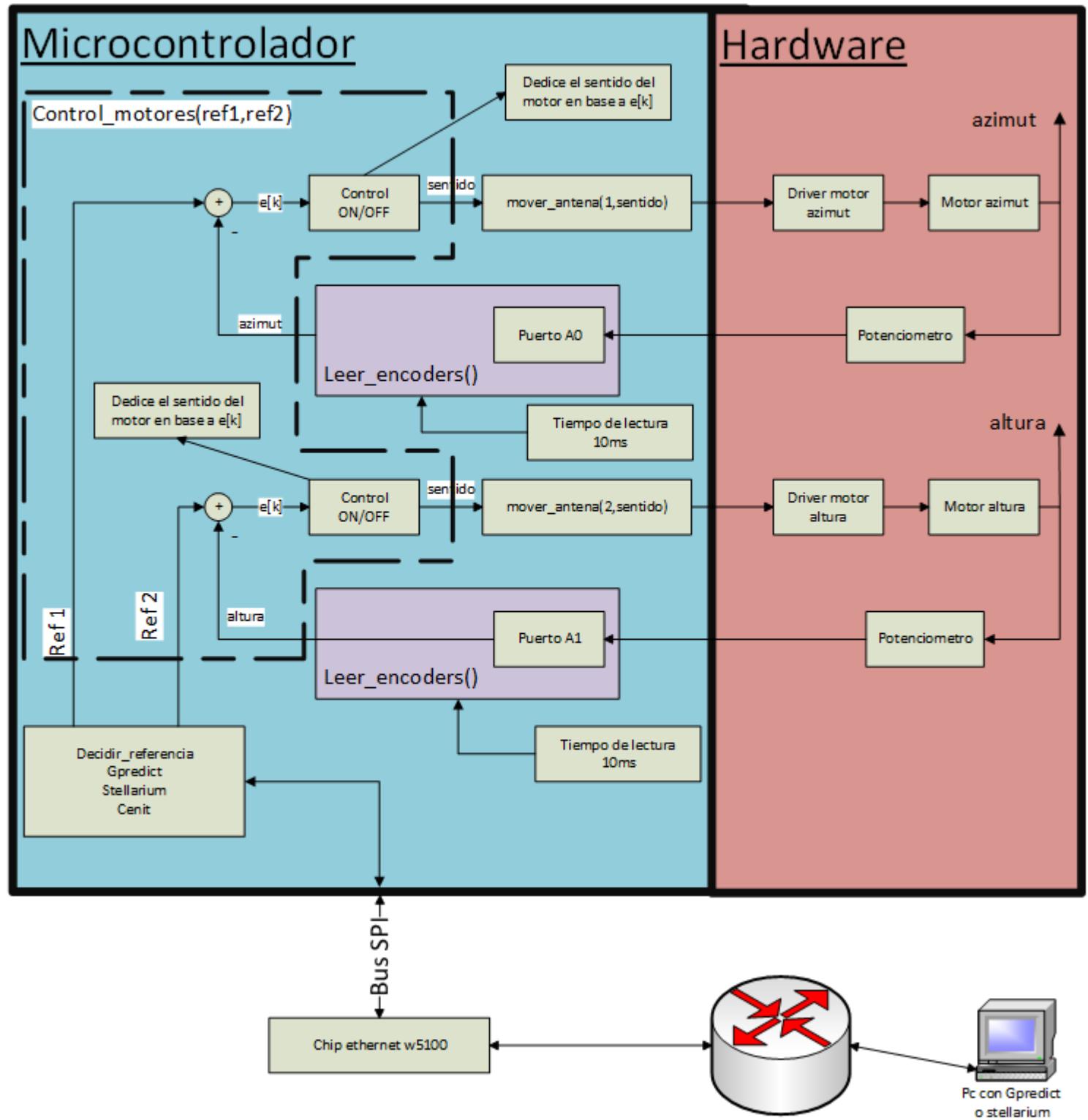


Figura 6.11: Sistema de control microcontrolado implementado en este trabajo.

El control ON/OFF, actúa 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

La medida angular sobre la que se debe realizar el control ON/OFF, se debe basar en el ángulo sólido de la antena, y en la potencia de ruido recibida por el microcontrolador (mas específicamente, la relación SNR, la explicación de esta, rebasa el alcance del presente informe). Esta potencia, nos da un indicativo de cual debe ser el intervalo del error para realizar el frenado de los motores en esa posición. Ambas medidas, no se pueden realizar, ya que el ángulo sólido de la antena, depende de la frecuencia de operación del receptor, mientras la potencia de ruido, depende del ambiente en el cual se coloque el dispositivo.

### 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 leds referidos en la figura 6.5. La resolución, se definió en la sección anterior. El valor de referencia utilizado para realizar las pruebas es el denominado cenit:  $90^\circ$  en altura y  $90^\circ$  en azimut.

En la tabla 6.3, se observa que, la función de control responde como se esperaba, ya que se observa que al cambiar el punto inicial, 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.

## 6.7 Programación de Scheduler o planificador

La programación por scheduler(o planificador por su traducción al español), es la base del sistema operativo en tiempo real(p. ej FreeRtos<sup>1</sup>). 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 – sincrónica. La ejecución sincrónica, 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, definidos por el programador. Luego, se prueban estos relojes, con el uso de un osciloscopio, y midiendo los tiempos, y revisando que coincidan con el código desarrollado.

<sup>1</sup> FreeRtos es un sistema operativo de tiempo real para sistemas embebidos

Motor 1: Motor de azimut - Referencia 90°						
autocalibración			control			
Posición inicial	Primer Giro	Segundo giro	$e > res_{az}$		$e < -res_{az}$	
			puerto on	puerto off	puerto on	puerto off
punto medio	giro izquierda	giro derecha	MOTOR_1.S2	MOTOR_1.S1	MOTOR_1.S1	MOTOR_1.S2
punto medio	giro derecha	giro izquierda	MOTOR_1.S1	MOTOR_1.S2	MOTOR_1.S2	MOTOR_1.S1
extremo izquierdo	giro derecha	x	MOTOR_1.S2	MOTOR_1.S1	MOTOR_1.S1	MOTOR_1.S2
extremo derecho	giro izquierda	x	MOTOR_1.S1	MOTOR_1.S2	MOTOR_1.S2	MOTOR_1.S1

Motor 2: Motor de altura - Referencia 90°						
autocalibración			control			
Posición inicial	Primer Giro	Segundo giro	$e > res_h$		$e < -res_h^1$	
			puerto on	puerto off	puerto on	puerto off
punto medio	giro izquierda	giro derecha	MOTOR_2.S1	MOTOR_2.S2	x	x
punto medio	giro derecha	giro izquierda	MOTOR_2.S2	MOTOR_2.S1	x	x
extremo izquierdo	giro derecha	x	MOTOR_2.S1	MOTOR_2.S2	x	x
extremo derecho	giro izquierda	x	MOTOR_2.S2	MOTOR_2.S1	x	x

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.

<sup>1</sup> 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.

### 6.7.1. Funcionamiento base de tiempo para scheduler

Este se basa en el concepto de interrupción. Las interrupciones se explican en el apéndice B. Se realiza una interrupción por timer es una interrupción sincrónica, 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.

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, y se configura con 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:

- 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 ([11]), 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), de todos los modos, es el que mejor se ajusta a una base de tiempo controlada.

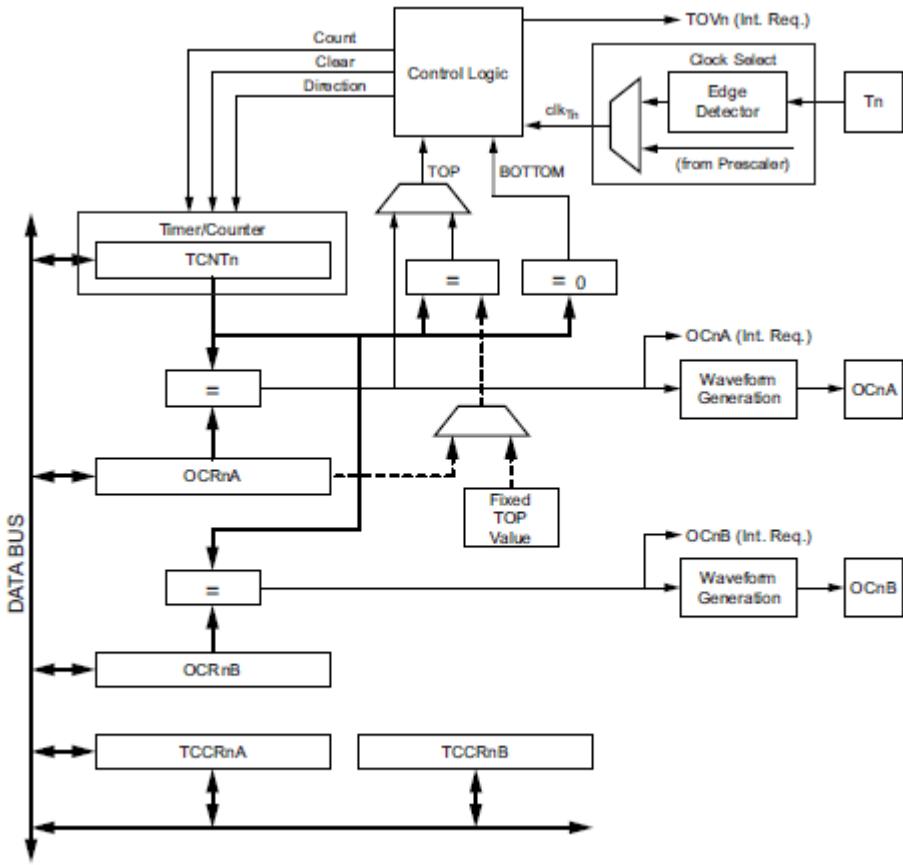


Figura 6.12: Diagrama del timer 2

### 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 ([11]).

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.

En el se ve, que la interrupción se ejecuta cuando TCNT2 alcanza a OCR2A. Para el cálculo 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

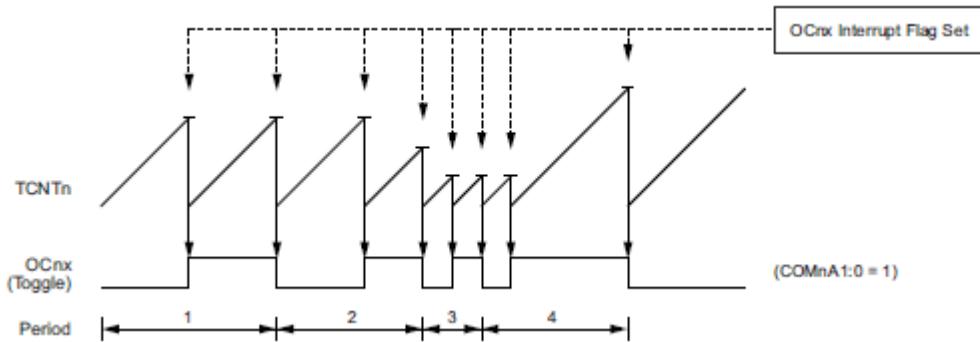


Figura 6.13: Diagrama de tiempos del modo CTC. Extraído de la hoja de datos del microcontrolador

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

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

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

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 multiplicar 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 (véase [1]). 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 ocurrirá, con una frecuencia de 10Khz, o cada 100  $\mu$ s.

Para configurar el valor de N(denominado preescalador, 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 (véase [1]), 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.

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.

```

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 3: Función base de tiempo.

### 6.7.3. Programación del Software scheduler

El software esta compuesto por 8 relojes, cuya cantidad puede programarse a partir del archivo “tiempo.cpp”, el cual tiene una bandera denominada RELOJES, para configurarse la cantidad de relojes deseada. 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 número 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, además, debe crearse una bandera, que diga, que reloj o relojes contaron el tiempo preestablecido por el programador. Esto se logra con un vector de eventos, donde va guardando los relojes que han agotado su tiempo. Se usa un vector, porque puede que exista más de un reloj que se agote su tiempo. Este vector se denomina FlagRepEvent[] dentro del código de programación. Además, creamos un vector que indica cuales relojes están activos, y lo llamamos timer\_activo. Todas estas variables, necesitan, comunicarse entre sí, y la comunicación entre ellas se realiza usando funciones. En la figura 6.14 se muestra un diagrama del software programado.

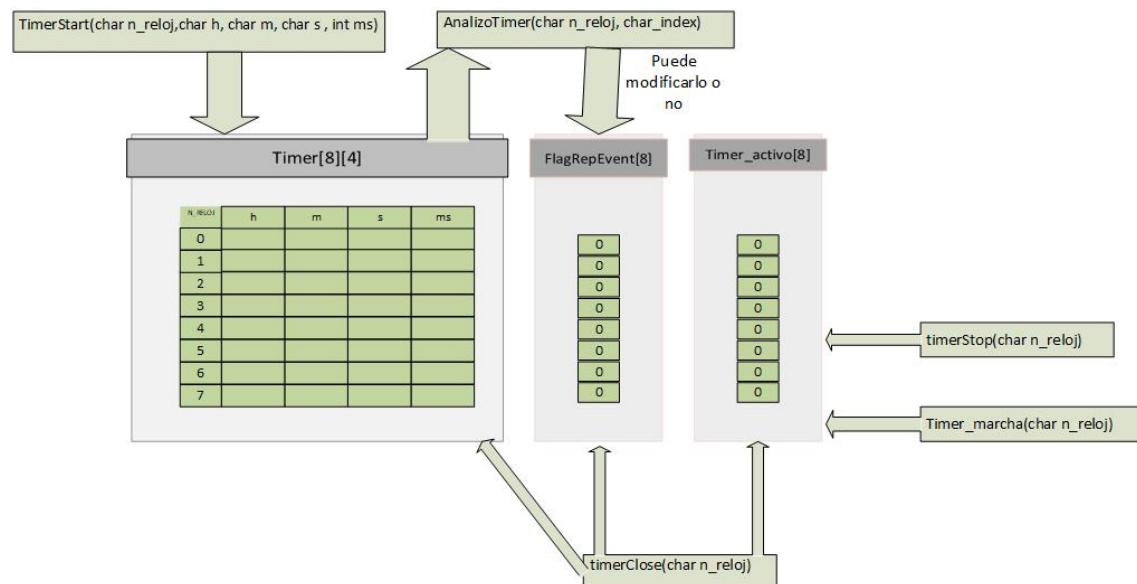


Figura 6.14: Diagrama de software para el software de scheduler

### 6.7.3.1. Funciones

Los archivos tiempo.cpp y .h ofrecen las siguientes funciones, encapsuladas 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)
- timer\_marcha(char n\_reloj)

**TimerStart:** inicializa el número 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 estén activos en el flag timer\_activo. La función decremente igual que lo haría un cronómetro de bolsillo. Es decir, llega ms a cero, y mira los segundos, si los segundos no son cero, decremente en uno el segundo, y pone en 999 la variable de ms en el número 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 cronómetro, 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.

**timerEvent():** esta función, observa la variable flagRepEvent[], y realiza la acción correspondiente, a ese tiempo asignado. En esta función, se asignan las tareas que debe realizar la planificación o scheduler.

### 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 estén activos, y si luego de descontar, todo el reloj es cero,(es decir, la fila n\_reloj es toda cero), activa el 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 número 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 halla en el CD que acompaña al presente documento.

### 6.7.4. Resultados del software de planificación

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

En la primera prueba, se configuran dos relojes de 872ms, y se miden con un osciloscopio. Ambos se lanzan a tiempos distintos, es decir, existe un retardo entre ambos. Los resultados se midieron en el osciloscopio. La segunda prueba, fue configurar dos relojes, parar uno y relanzarlo, mientras el otro permanecía encendido. Los resultados de ambas pruebas sobre el osciloscopio se observan en las siguientes imágenes:



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

Figura 6.15: prueba sobre el software de manejo de scheduler

Los códigos utilizados para medir esta función, no se encuentran 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 primera 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 qué 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, se trata de ver como el software, decide esa referencia. Las referencias provienen de tres fuentes posibles:

- Gpredict.
- Stellarium.
- cenit(ningún software conectado).

Para ello, dentro del programa principal, se crean dos variables: denominadas `ref1` y `ref2` respectivamente. Estas tendrán por defecto el valor del cenit ( $90^\circ$  en ángulo de azimuth y  $90^\circ$

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, qué 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 primer lugar, se debe conectar el chip ethernet w5100, mediante un cable denominado utp, a un switch o router. Este cable es comúnmente conocido como cable de red.

Una vez, el chip ethernet, se ha conectado físicamente a la red, mediante un cable de red, se deben obtener los parámetros de la red, mediante el uso del 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 “ethernet.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 6.4. 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 está 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 4. La dirección IP, que se le asigna mediante el protocolo DHCP, se muestra en el puerto serie, con este código.

```

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

código 4: Obtención de la dirección IP usando el protocolo DHCP

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.

### 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 del eje azimutal, en el polo norte geográfico, y en sentido de las agujas del reloj, aumenta la cantidad de grados, hasta llegar a 360°. Por este motivo, la configuración de Gpredict en el eje azimutal, debe ser entre 90°(corresponde al este), y 270°(corresponde al oeste geográfico). Dicho esto, se configura el rotador como muestra la siguiente figura:

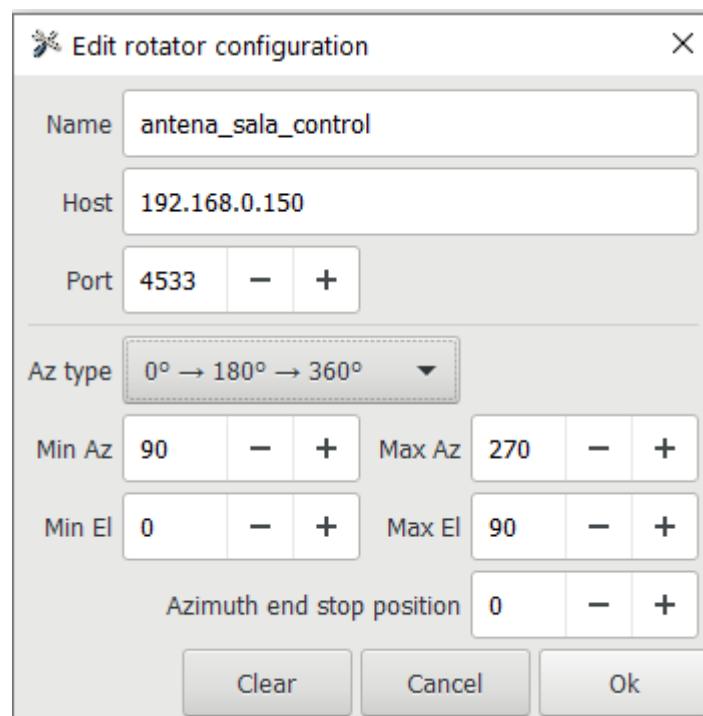


Figura 6.16: Imagen de la configuración del software Gpredict para la antena ubicada en sala de control del IAR.

#### 6.8.2.1. Panel de control del rotador

Si se procede a abrir el rotador recién creado, se observa la siguiente imagen(ver sección 5.4.3): Observamos en la imagen, que se tienen los siguientes partes:

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

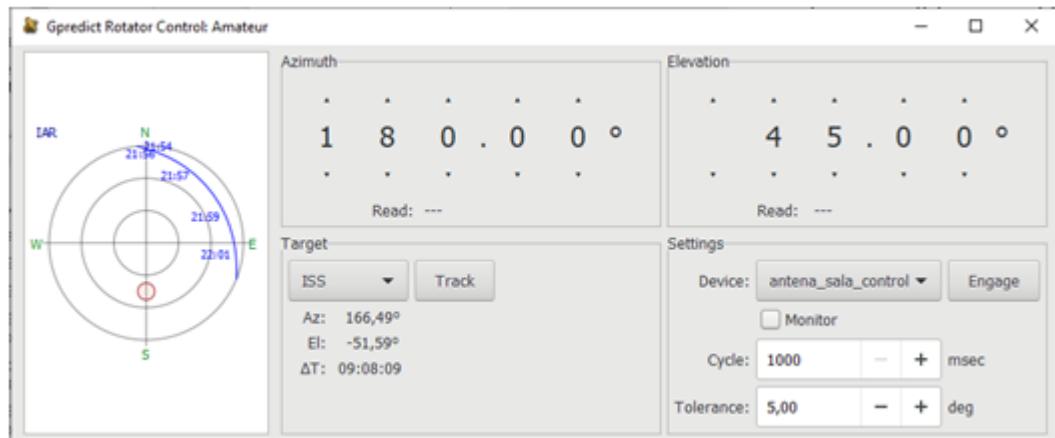


Figura 6.17: Panel de control del rotador en Gpredict

En la parte izquierda de la figura 6.17, se encuentra una gráfica en coordenadas polares. Los círculos concéntricos, indican el ángulo de altura, siendo el círculo exterior un ángulo de altura de  $0^\circ$ , y el centro de  $90^\circ$ . El ángulo desde el norte, en sentido de las agujas del reloj, indica el ángulo de azimut. Este ángulo es el que está representado en la ventana de azimut y elevación, con un círculo rojo. Por ejemplo, en la imagen de la figura 6.17, se observa,  $180^\circ$  azimut, y  $45^\circ$  elevación, y el círculo rojo, está marcado apuntando al sur, y aproximadamente en la mitad del radio del centro, hasta el círculo más grande. La imagen 6.18 aclara la explicación dada. En la figura, se observa que hay un recuadro a rayas rojas, esto indica, la visibilidad de la antena instalada en sala de control en el IAR.

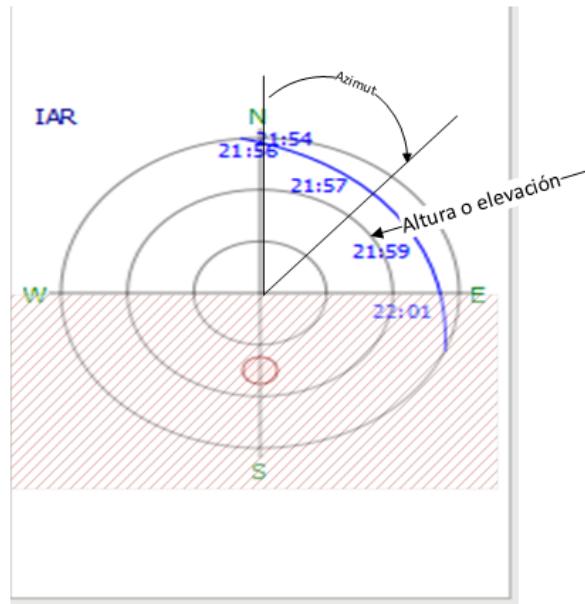


Figura 6.18: Diagrama de gráfico polar en software Gpredict.

De la figura, se observa, qué en el ángulo de azimut, se tiene un desfase de  $270^\circ$ , entre la convención de gpredict, y la convención utilizada en este trabajo. Por este motivo, a la coordenada recibida por el software Gpredict, debe restársele esta cantidad. Si denominamos  $\theta_{azgpr}$  a la coordenada recibida por Gpredict, y  $\theta_{az}$  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 en la tabla 6.4

track	engaje	Descripción.
off	off	No se envían comandos al rotador y tampoco se lee su posición
on	off	La posición del controlador se actualizan con la posición destino pero no se envían comandos al rotador. La posición actual del rotador no se lee. Si el satélite objetivo esta fuera del alcance, las coordenadas del controlador se establecerán en donde se espera que aparezca el satélite.
off	on	Las coordenadas azimut y altura, se envían al rotador, pero no se establece ningún satélite para seguir. Este modo se puede usarse para controlar manualmente el rotador, o realizar una calibración de la antena.
on	on	La posición del satélite a seguir, se envía continuamente al rotador en caso de que exista un satélite dentro del campo de visión de la antena. Si el satélite no se encuentra dentro del campo de visión, se establece la ubicación de la antena para esperarlo, y luego seguirlo

Tabla 6.4: comportamiento al oprimir los botones track y engage

De la tabla, 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 parte del panel donde dice “azimut 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 de manera nativa con el entorno de desarrollo arduino, 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 5: 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, y responder adecuadamente. Recordar que hay dos tipos de comandos, los comandos de tipo Get y de tipo Set. En la siguiente tabla, se ilustra el formato de los datos, tanto en la respuesta como en la llamada:

Comandos tipo Get		Comandos tipo Set	
Protocolo de envío	respuesta	protocolo de envío	respuesta
comando <sup>1</sup>	par1\npar2\n	comando <sup>1</sup> par1 <sup>2</sup> par2 <sup>2</sup> \n	par1\npar2\n

<sup>1</sup> los comandos están definidos en la tabla 5.2

<sup>2</sup> par1 y par2 son las coordenadas de azimut y elevación respectivamente.

Tabla 6.5: Envío de respuesta y comandos entre Gpredict y el microcontrolador.

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 6: captura de paquetes recibidos mediante el software Gpredict

Dentro de las llaves debe ir la acción a realizar según el comando recibido. Los comandos que utiliza Gpredict, son los que se muestran en la tabla 5.2, y el formato para responder a estas peticiones, se encuentran en la tabla 6.5. Para exemplificar: suponga que se lee el valor de "P 155.20 38.3". Esto significa que la antena debe apuntar al punto 155.20° en azimut, y 38.3° en altura, pero debe responder la petición, como indica la tabla 6.5. 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 azimut, y 39° en altura, la respuesta a esta petición será "125.0\n 39°\n", siendo "\n" un carácter de salto de línea y en la pantalla de Gpredict, aparecerán estas coordenadas. En este ejemplo, no se aplicó la corrección de las coordenadas, están 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". El código desarrollado, se encuentra en el CD anexo al presente informe.

#### 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 siguiente figura

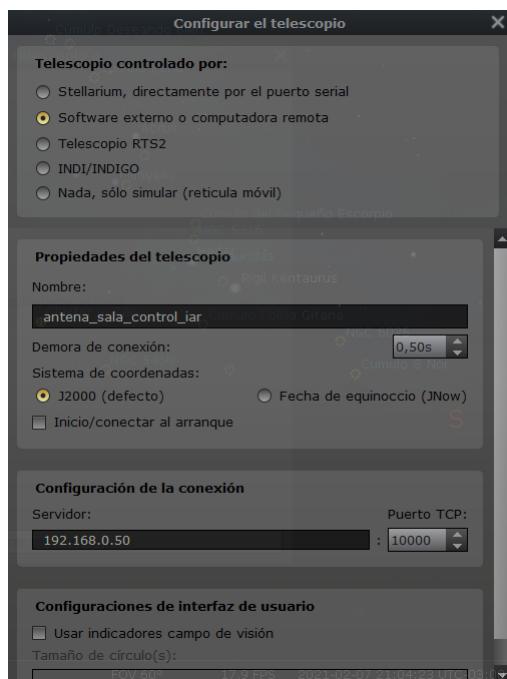


Figura 6.19: configuración del stellarium

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

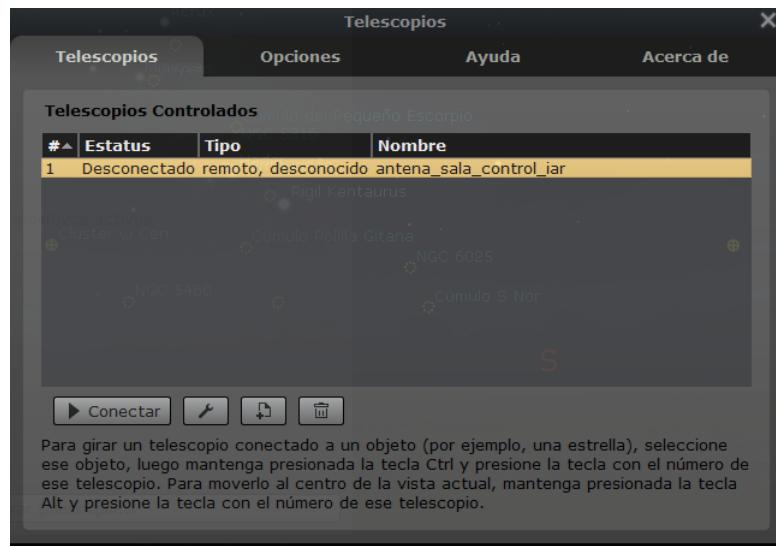


Figura 6.20: Apertura del telescopio recién creado.

Una vez, ahí dentro, presionando el botón conectar, se aprecia la siguiente ventana que aparece en la siguiente imagen:

donde se observan los siguientes botones y opciones:

- Rotar: envía 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 ningún tipo de uso.
- Ascensión recta: tipo de coordenadas para visualizar estrellas.
- declinación: ídem que ascensión 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 qué 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:

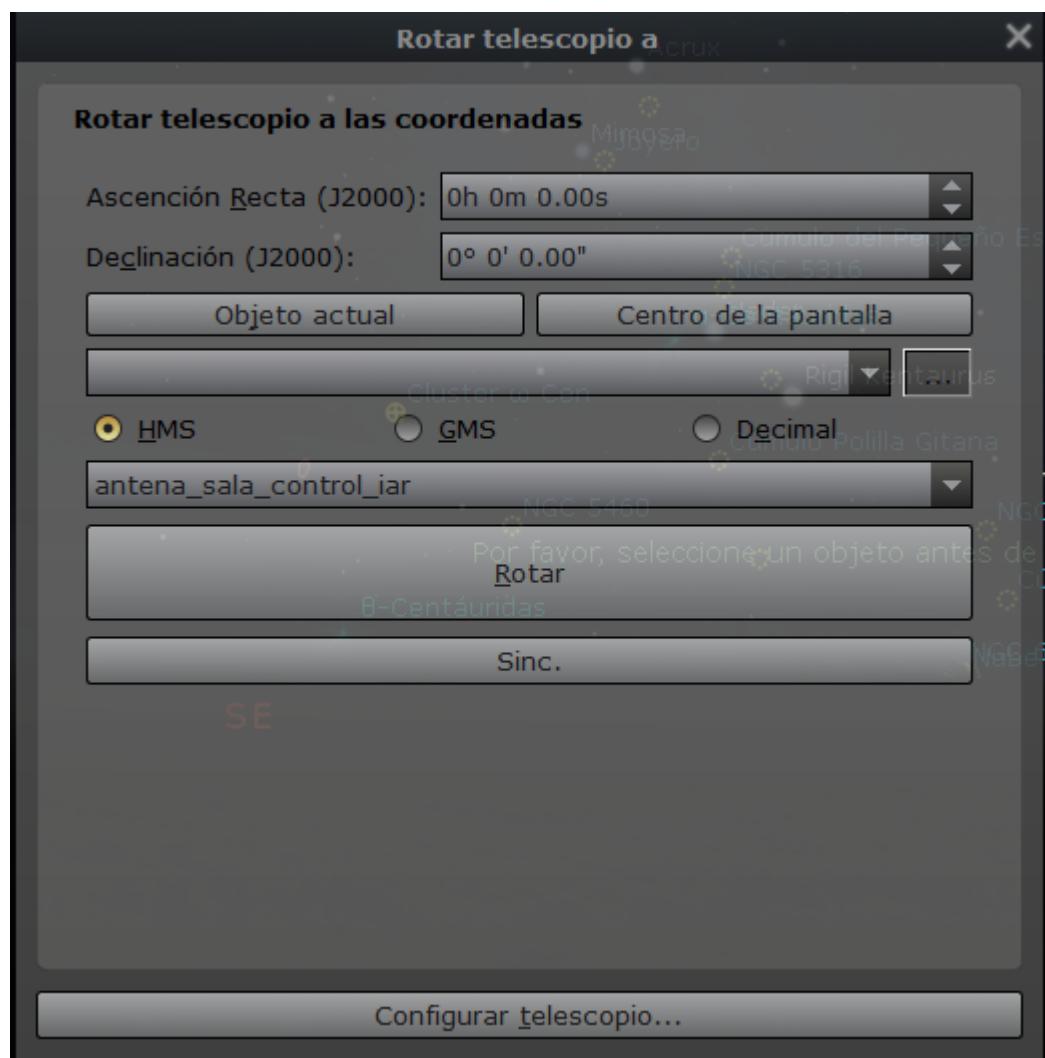


Figura 6.21: Imagen del panel de control del telescopio usando Stellarium

En la sección 5.5.2, se explica como envía los datos el stellarium, es decir, la trama de datos. En esta 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([12]), utiliza los ángulos en un sistema decimal (los siguientes datos estan en sistema hexadecimal):

- ascención recta:
  - ascención recta 0hs = 0x100000000
  - ascención recta 12hs = 0x800000000
  - ascención recta 24hs = 0x000000000
  
- declinación:
  - declinación -90° = -0x400000000
  - declinación 0° = 0x000000000
  - declinación -90° = -0x400000000

```

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

```

codigo 7: definición de objeto servidor para conectarse con el stellarium

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:

```

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);
8

```

codigo 8: Reorganización de los datos recibidos desde el programa stellarium dentro del microcontrolador

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 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 }

```

codigo 9: Parte del software que se encarga de conectarse con el software stellarium programado dentro del microcontrolador.

Luego, dentro de cada bloque if (ver código 6 y código 9), 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 Gpredict, con el microntrolador Atmega328p, utilizando como interface de 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 programas 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 azimut y elevación, se obtienen la posición de la antena (donde dice “read” en la figura 6.17). 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 azimut y altura, marcada con números más grandes en la imagen 6.17.

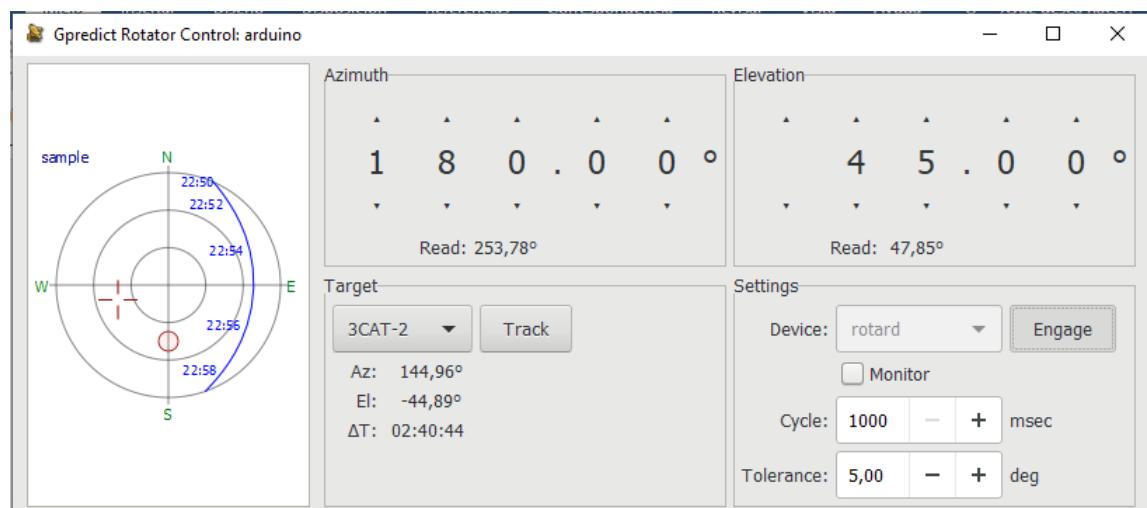


Figura 6.22: Conexión del microcontrolador al software Gpredict mediante el uso del protocolo TCP/IP

En la prueba siguiente, se le dio a la antena, una altura de  $45^\circ$  y ángulo de azimut de  $180^\circ$ . Esta posición está marcada con un círculo rojo dentro de la gráfica polar. La posición que se encuentra el rotador, es de  $253.78^\circ$  en azimut y  $47.85^\circ$  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 desde Gpredict. En los comandos enviados y recibidos, no se ha mostrado ningún tipo de anomalía. En la figura 6.23 se dejan los resultados del panel de control y en la figura 6.24 los resultados del análisis hecho con wireshark.

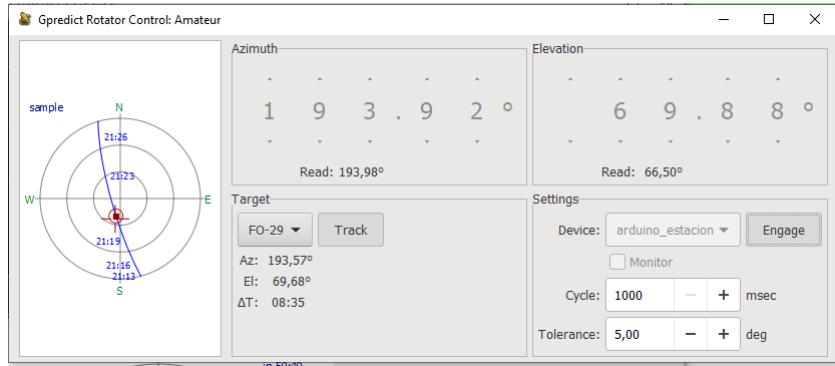


Figura 6.23: Panel de control del monitor realizando un seguimiento satelital

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	10.215948	192.168.0.51	192.168.0.127	TCP	60	4533 → 49786 [PSH, ACK] Seq=78 Ack=56 Win=2033 Len=6
43	10.215948	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=58 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=84 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	11.044172	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	11.169940	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=90 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=98 Ack=122 Win=64119 Len=2
62	12.000847	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	13.781127	192.168.0.51	192.168.0.127	TCP	67	4533 → 49786 [PSH, ACK] Seq=148 Ack=96 Win=2046 Len=13
71	13.791700	192.168.0.127	192.168.0.51	TCP	54	49786 → 4533 [ACK] Seq=96 Ack=141 Win=64090 Len=0

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

### 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 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.

Cabe destacar, que el ángulo de ascensión recta marcado por el stellarium es de  $-109,41^\circ$ , y el del programa desarrollado, es de  $250.58^\circ$ . Este desfasaje, se da, por el signo negativo de la ascensión recta. Este signo negativo, indica que es contrario al sentido establecido por los astrónomos en este tipo de coordenadas. Para transformarla, se debe restarle  $360^\circ$  a ese ángulo, y se obtendrá la

ascensión recta. La cuenta que debe realizarse es:

$$360^\circ - 109,41^\circ = 250,58$$

Se observa, que el ángulo coincide, con el valor predicho, esto se observa en la imagen 6.26.

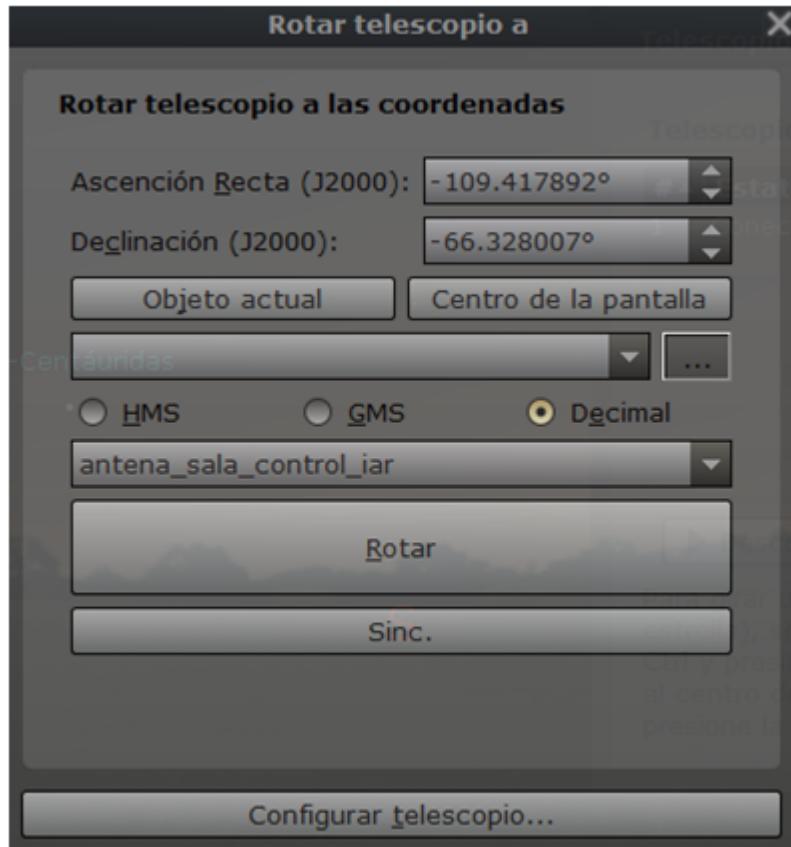


Figura 6.25: Panel de control del rotador del stellarium al momento de realizar las pruebas sobre el software

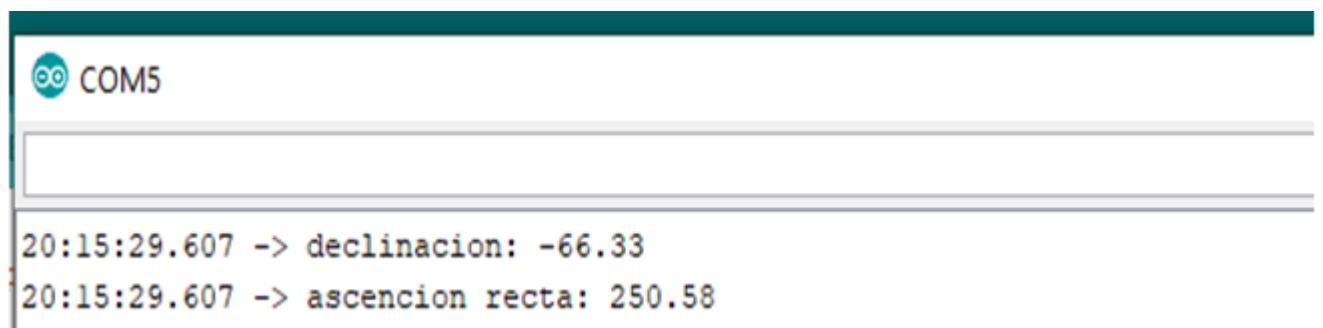


Figura 6.26: Coordenadas recibidas por el stellarium

Además, se revisa la conexión mediante el software wireshark. Los resultados del análisis se muestran en la imagen 6.27.

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 (Port 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 (Port unreachable)
23	20.289959	192.168.0.127	192.168.0.51	NBNS	92	Name query NBSTAT <><><><><><><><><><><><><><>
24	20.299649	192.168.0.51	192.168.0.127	ICMP	70	Destination unreachable (Port unreachable)
28	21.838053	192.168.0.127	192.168.0.51	TCP	68	49898 + 10000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
29	21.849818	192.168.0.51	192.168.0.127	TCP	68	10000 + 49898 [SYN, ACK] Seq=0 Ack=1 Win=2048 Len=0 MSS=1460
30	21.850040	192.168.0.127	192.168.0.51	TCP	54	49898 + 10000 [ACK] Seq=1 Ack=1 Win=64240 Len=0
41	34.055676	192.168.0.127	192.168.0.51	TCP	74	49898 + 10000 [SH, ACK] Seq=1 Ack=1 Win=64240 Len=0
42	34.070840	192.168.0.127	192.168.0.51	TCP	68	10000 + 49898 [ACK] Seq=1 Ack=1 Win=2048 Len=0
43	34.087505	192.168.0.51	192.168.0.127	TCP	74	10000 + 49898 [SH, ACK] Seq=1 Ack=21 Win=2048 Len=20
44	34.119106	192.168.0.127	192.168.0.51	TCP	54	49898 + 10000 [FIN, ACK] Seq=21 Ack=21 Win=64228 Len=0
45	34.138724	192.168.0.51	192.168.0.127	TCP	68	10000 + 49898 [ACK] Seq=21 Ack=22 Win=2048 Len=0
46	34.138724	192.168.0.51	192.168.0.127	TCP	68	10000 + 49898 [FIN, ACK] Seq=21 Ack=22 Win=2048 Len=0
47	34.138852	192.168.0.127	192.168.0.51	TCP	54	49898 + 10000 [ACK] Seq=22 Ack=22 Win=64228 Len=0
48	34.184201	192.168.0.127	192.168.0.51	TCP	68	49899 + 10000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
49	34.194159	192.168.0.51	192.168.0.127	TCP	68	10000 + 49899 [SYN, ACK] Seq=0 Ack=1 Win=2048 Len=0 MSS=1460
50	34.194310	192.168.0.127	192.168.0.51	TCP	54	49899 + 10000 [ACK] Seq=1 Ack=1 Win=64240 Len=0
52	41.054255	192.168.0.127	192.168.0.51	TCP	54	49899 + 10000 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
53	41.065485	192.168.0.51	192.168.0.127	TCP	68	10000 + 49899 [ACK] Seq=1 Ack=2 Win=2048 Len=0
54	41.065485	192.168.0.51	192.168.0.127	TCP	68	10000 + 49899 [FIN, ACK] Seq=1 Ack=2 Win=2048 Len=0
55	41.065610	192.168.0.127	192.168.0.51	TCP	54	49899 + 10000 [ACK] Seq=2 Ack=2 Win=64228 Len=0

Figura 6.27: Coordenadas recibidas por el stellarium

## 6.9 Integración de partes del software

En esta sección se procede a unir todas las partes desarrolladas en esta etapa. Esta integración, se realiza utilizando el esquema de la figura 6.1. La primera parte, debe realizar la función de 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 el código mostrado en la función setup, del archivo “main.cpp” mostrado en el apéndice del presente capítulo. Asimismo, en esta función, se programa, la obtención de la IP por DHCP, y se muestra por el display LCD.

La información angular, el número de IP, la depuración de variables, etc, se realiza mediante el puerto serie. En la próxima sección, se utiliza el display LCD para mostrar la dirección IP, y como se utilizan sus métodos para escribir texto en ella. 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 conectó 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 conectó 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, y no se encontró ningún tipo de Bug o error en el código.

## 6.10 Programación del display LCD

El entorno arduino, provee una librería específica para el manejo del display con el chip PCF8574. Esta se denomina `LiquidCrystal_I2C`. Para utilizarla, se debe incluir en el archivo principal del proyecto. Una vez incluida, se debe crear un objeto LCD, que será el encargado de realizar todas las funciones del display. El objeto LCD se crea de la siguiente forma:

LiquidCrystal\_I2C lcd (ADRRESS\_LCD\_I2C,COLS\_LCD,FILAS\_LCD), donde ADRESS\_LCD tiene el valor 0x27, que es la dirección del display para el uso del I2C(ver apéndice A).

Algunos de los métodos que posee esta clase, son:

- **print(String c):** imprime el String c en el display.
- **setCursor(x,y):** posiciona el cursor en una fila y columna(x es la columna e y es la fila)
- **backlight():** prende el backlight del display LCD.

Estas funciones, serán utilizadas para mostrar información en el display LCD, reemplazando al puerto serie. En el código anexado al presente documento, ya se encuentra implementado.

## 6.11 Análisis del código realizado

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

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

Tabla 6.6: Tabla resumen de las líneas de desarrollo realizadas hasta el momento

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 realizó 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 este método, 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, ha agregado código para aumentar la capacidad de la memoria FLASH. En la siguiente figura se deja una imagen con los resultados del obj dump.

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

## 6.12 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 librerías, 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.

## **Fase III**

# **Sistemas de coordenadas astronómicas**

# Sistemas de coordenadas astronómicas

## Resumen

Se explican, los fundamentos de las coordenadas astronómicas. Además, se explican algunos de los sistemas de coordenadas utilizados en astronomía. Esta descripción es necesaria, para poder realizar la transformación de coordenadas dentro del microcontrolador, en caso que se este siguiendo una estrella o satélite, mediante el programa Stellarium,

## 7.1 introducción

En este capítulo se describe como realizar la transformación de coordenadas. El microcontrolador, recibe las coordenadas en coordenadas ecuatoriales horarias (con el software Stellarium), y debe realizar la transformación de coordenadas a ecuatoriales locales, luego debe realizarse otra transformación, al sistema horizontal local, dado el tipo de montura (altazimutal) que posee la antena. El presente texto, intenta explicar cuáles son los sistemas de coordenadas usados en astronomía, y necesarios para el presente trabajo. Existen más sistemas de coordenadas, pero rebasan el alcance del proyecto, el lector interesado puede ver la referencia [15]. Para ello, se debe empezar conociendo conceptos básicos de la trigonometría esférica, esfera celeste, etc. Al lector que este familiarizado con estos temas, puede omitir este capítulo.

## 7.2 Esfera - Elementos fundamentales

La superficie más usada a lo largo del presente capítulo es la esfera. Por este motivo, se va a definir una esfera, en conjunto con sus elementos fundamentales. A partir de esto, obtendremos algunos resultados de una materia denominada trigonometría esférica. Esta, se plantea en la superficie de una esfera, a partir de los elementos básicos de ella, por ello, el concepto de esfera y sus elementos es importante para comprender el resto del capítulo. Cabe recordar, que una esfera también puede escribirse en coordenadas esféricas.

Definición de esfera[1] Conjunto de puntos del espacio equidistantes de otro punto fijo. Sobre esta definición, al punto fijo, se lo conoce como centro de la esfera y lo vamos a denotar con la letra O. La distancia desde O a cualquier punto, se denomina radio de la esfera (R). Si a la esfera, le hacemos pasar un plano, según este plano contenga a O, o no, podemos definir dos tipos de círculos o circunferencias:

- Circunferencia máxima (o círculo máximo): Circunferencia que se obtiene al cortar la esfera con el plano que contiene a O
- Circunferencia menor (o círculo menor): Si el centro de la esfera (O) no está contenida en el plano que corta a la esfera.

Ahora, para definir las posiciones de un punto, sobre la esfera, es necesario definir círculos de referencia, usando círculos menores y mayores. Si se tiene un círculo máximo, y se traza una recta perpendicular al plano del círculo máximo, y que pase por O, esta recta, corta a la esfera en dos puntos, denominados polos. Además de estos elementos, se tienen los siguientes:

## **Fase IV**

# **Sistema electrónico de posicionamiento - Construcción de prototipo y resultados**

## **construcción del manejador de los motores**

# 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:

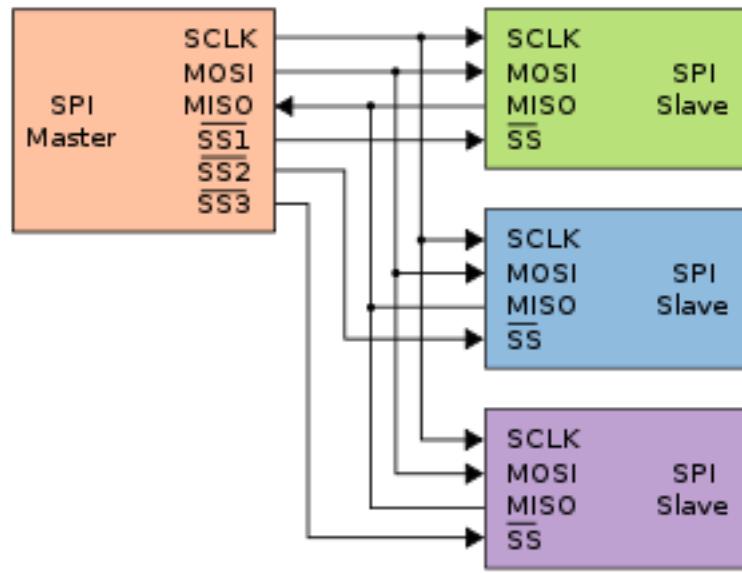


Figura A.1: Esquema de conexiones entre un dispositivo maestro y un dispositivo esclavo usando el protocolo SPI.

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

El dispositivo maestro, es el encargado de seleccionar a que dispositivo se quiere comunicar, mediante la línea 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 sí. La señal de clock, solo está 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. En la siguiente imagen, se muestra un diagrama de tiempos de las señales involucradas en el protocolo SPI.

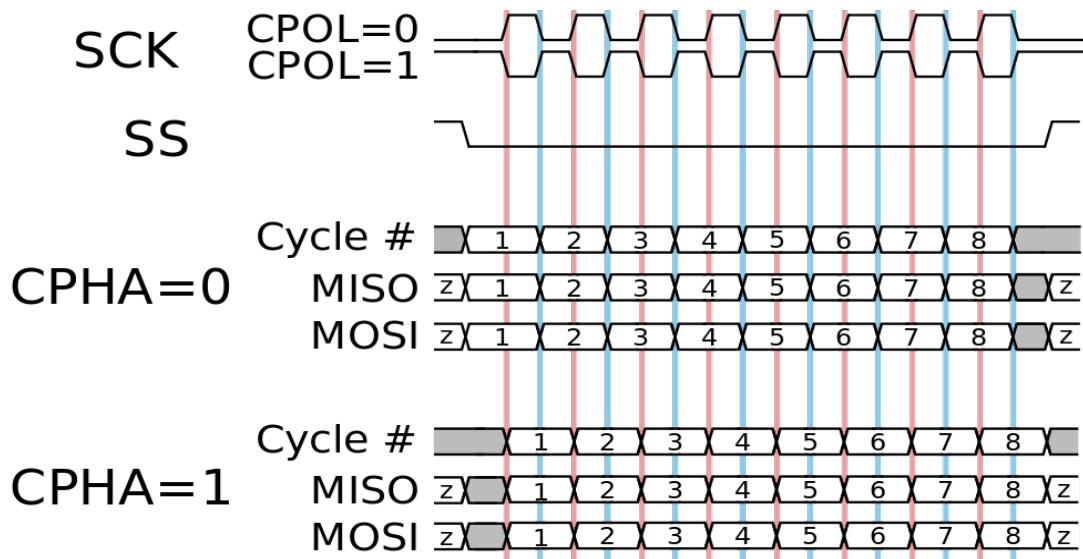


Figura A.2: Señales del protocolo SPI en función del tiempo. La letra z dentro del diagrama indica estado de alta impedancia

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 envían datos cuando hay una transición del estado alto al estado bajo.

- Modo 2: CPOL = 1 y CPHA = 0. El estado lógico del reloj es alto, y se envían los datos en cada transición de bajo a alto(existe un desfase entre el estado lógico y el envío de los datos).
- Modo 3: CPOL = 1 y CPHA = 1. Modo en el cual el 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 lógico y el envío de los datos).

Por último, 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 lo menos tengan un modo en común para que la comunicación pueda llevarse a cabo.

### A.3 Proctocolo 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 bibliografías, 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:

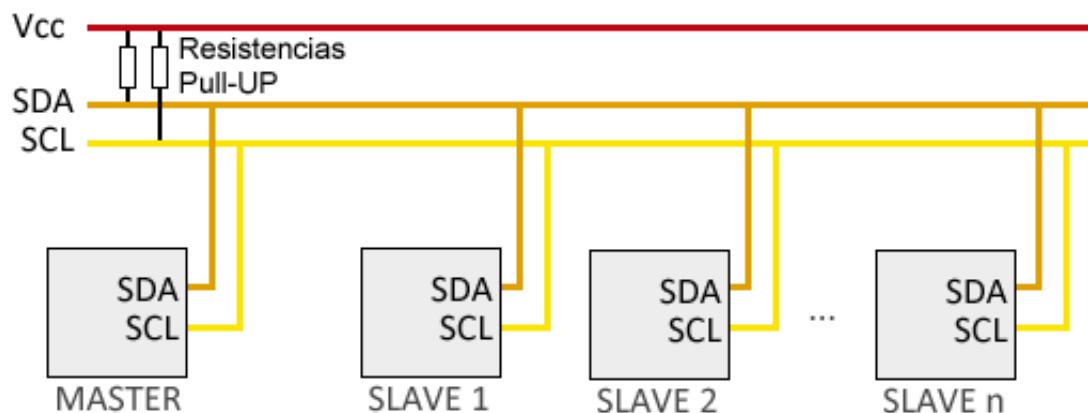


Figura A.3: diagrama de las conexiones entre una maestro y n esclavos mediante el uso del protocolo I2C

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 qué forma se están 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 línea SDA en bajo, eso les indica a los esclavos que se va a iniciar una comunicación por parte del master. Acto seguido, se inicia la

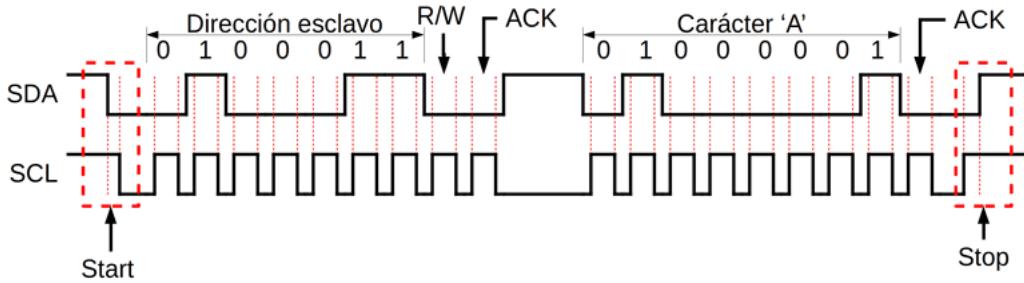


Figura A.4: Diagrama de tiempos que utiliza el protocolo I2C. En este caso, se utiliza una dirección de 7 bits

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 transmite un ACK, poniendo la linea SDA en bajo. Así, cada 8 bits, luego cuando la linea SCL permanece en alto, significa que la comunicación ha finalizado. Si el dispositivo no esta 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: Línea MISO para el protocolo SPI

- pin SCK o CLOCK: Línea 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 línea de red ethernet. Recibe potencia positiva
- pin P-: Power of Ethernet. Cable destinado a recibir alimentación por la línea de red ethernet. Recibe potencia negativa

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

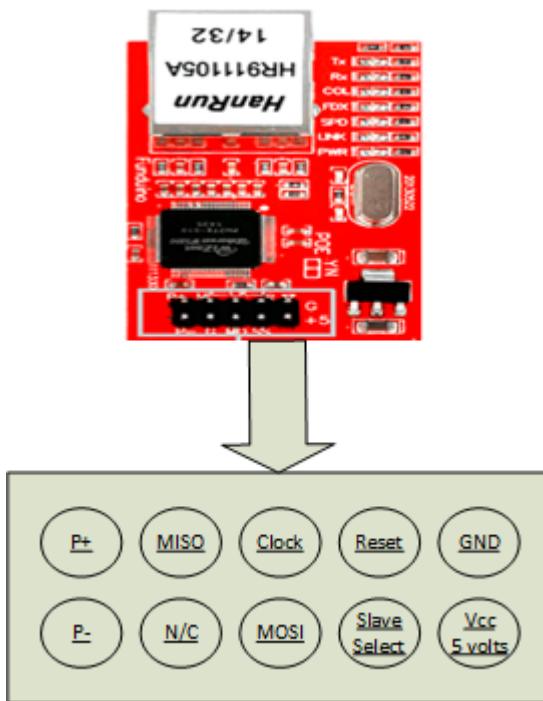


Figura A.5: Esquema de pines que posee el módulo W5100.

Si se observa la hoja de datos del dispositivo, se observan que los modos SPI soportados son los modos 0 y 3 respectivamente(véase pag. 61 de [13]).

## 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(ver [14]), 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 líneas 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, está realizada para aplicarla exclusivamente a los displays LCD, y se provee la librería 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

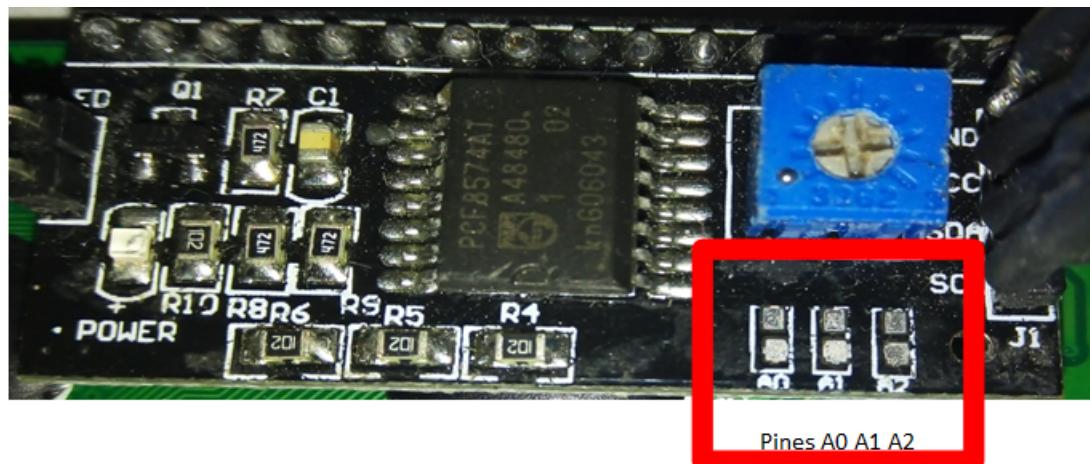


Figura A.6: Foto del display LCD de la parte de atrás, donde se observan los pines A0,A1 y A2 accesibles para poder modificarlos.

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:

Esta placa por defecto, trae todos los pines A0,A1 y A2 en alto. Si queremos pasar algún 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(pág. 13 de [14]), se explica que la dirección posee 7 bits, donde los siete bits vienen dados en 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 ínfima, y no conviene optar por el display LCD sin el chip(la diferencia en pesos argentinos ronda los \$200,en esta época,esto equivale a 1,25 dólares aproximadamente al escribir esta sección del presente documento).

Los puertos que tiene disponibles a la salida, son Vcc Y GND. En el pin de Vcc deben conectarse 5 volts, 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, están implementadas sobre la misma placa.

# Programación sobre plataforma Arduino UNO

## B.1 PlatformIO

En este trabajo, no se utiliza el IDE provisto por la plataforma de Arduino, sino, que se utiliza PlatformIO. Para instalar platformIO, debe instalarse el editor de código VisualStudioCode. Este es gratuito, multiplataforma, y se encuentra disponible en <https://code.visualstudio.com/>.

Una vez instalado el editor de código, se debe instalar el plugin de PlatformIO, el cual es una extensión de Visual Studio Code, que permite programar sobre casi cualquier sistema embebido. Para instalar este plugin, dentro del editor de código, debe realizar un clic en el cuadrado rojo, que se muestra la siguiente imagen:

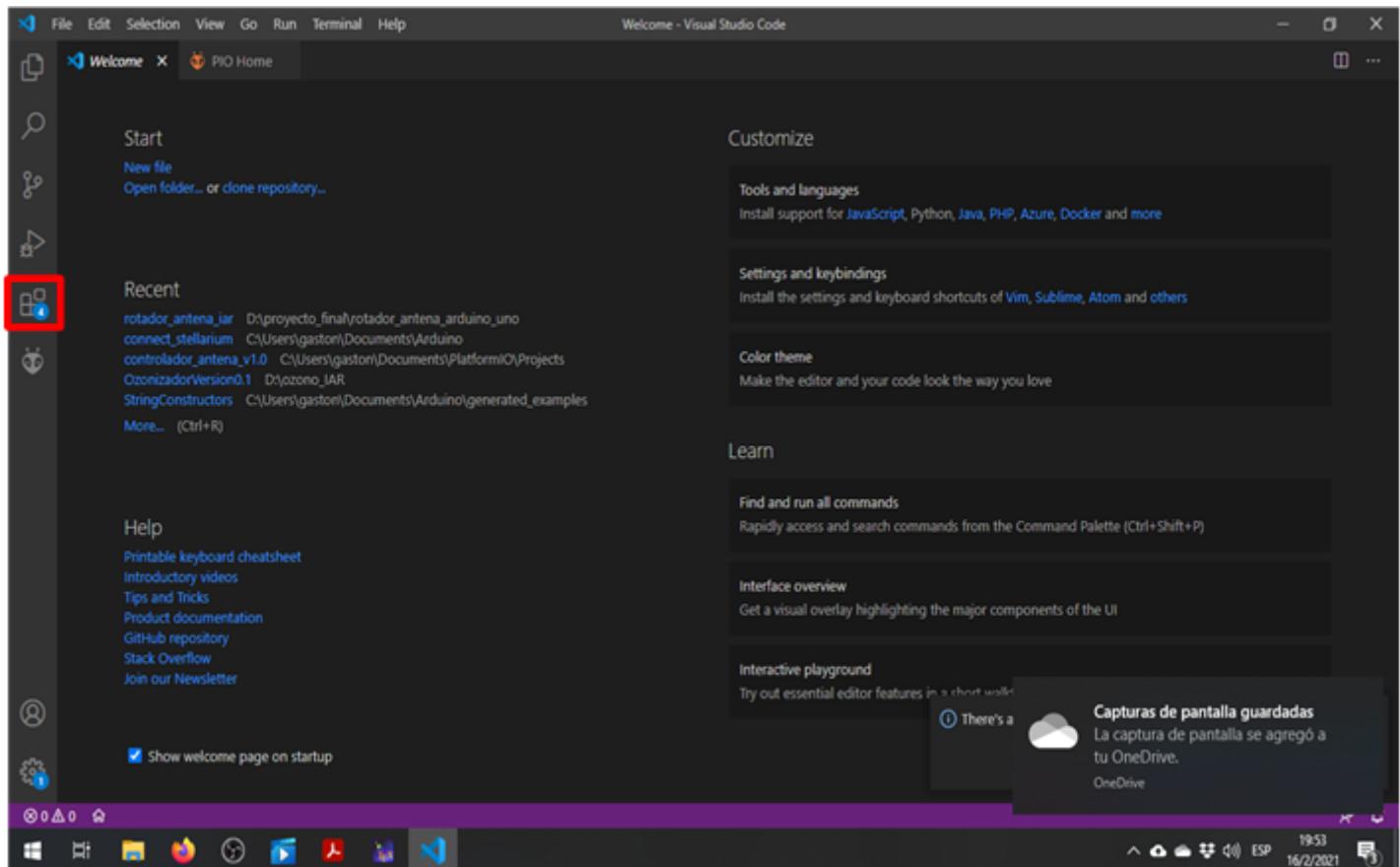


Figura B.1: imagen de Visual Studio Code para añadir extensiones

Luego, se abre una ventana donde puede introducir texto, en esta debe introducir “PlatformIO”, y en la parte derecha de la misma, le aparece, el botón de instalar en azul. Una vez se realicen estos pasos, se puede iniciar un nuevo proyecto dentro de este plugin. La explicación del inicio de un nuevo proyecto, y el proceso de instalación detallado, rebasa el alcance del presente informe.

## B.2 Configuración de platformIO

Una vez, iniciado un nuevo proyecto en platformIO, aparecen, las siguientes carpetas y archivos en el directorio principal del proyecto:

- carpetas:

- .pio
- .vscode
- include
- lib
- src
- test

- archivos:

- .gitignore
- platformio.ini

Las carpetas .pio y .vscode, tienen configuraciones relacionadas al proyecto(compilador, librerías, ruta del compilador, etc). El archivo .gitignore, está relacionado con el sistema de control de versiones git, en este trabajo, se utilizó este sistema de control de versiones. La explicación del mismo, rebasa el alcance de este texto. El otro archivo, se relaciona con la configuración de platformIO. Este, se modifica, para que se pueda realizar la compilación.

Dentro de estas carpetas, las que se utilizan en el proyecto, son las carpetas include, lib y src.

La carpeta include, contiene todos los archivos de cabecera que utiliza el proyecto, estos, en general tienen extensión .h. La carpeta lib, incluyen las librerías, tanto las desarrolladas por terceros, como las que se desarrollen en el proyecto, y la carpeta src, contiene el archivo principal del programa.

En el programa desarrollado en el presente texto, dentro de la carpeta include, se crean dos carpetas, denominadas “lectura\_encoders” y “control\_motores”. Dentro de estas carpetas, se colocan los archivos “lectura\_encoders.cpp”, “lectura\_encoders.h”, “control\_motores.h” y “control\_motores.cpp”(ver anexo del capítulo ?? y código versión final).

Una vez, creados estos archivos, dentro de la carpeta lib, se crea una nueva carpeta, denominada “tiempo”, donde se guardan los archivos “tiempo.cpp” y “tiempo.h”.

Una vez, finalizada la creación de archivos y carpetas, se debe proceder a configurar el entorno de desarrollo. El entorno de desarrollo, se configura, mediante el archivo platformio.ini.

La programación del archivo platformio.ino, puede encontrarse en [19]. En, esta referencia, se explica que un proyecto, puede poseer distintos entornos. En el caso del presente trabajo, se utiliza un solo entorno. La sentencia para crear el entorno es [`env:entorno`], donde entorno, se le debe dar el nombre que el programador desee.

Luego de este paso, se debe configurar, la búsqueda de los archivos dentro de los directorios. Esto, según el propio compilador, debe tener la directiva “-I”. Esta directiva se da dentro del archivo platformio.ini, con la sentencia build\_flags. Esta, permite, adicionar todas las banderas de compilación, que se deseen, donde deben separarse por comas o por saltos de líneas. En este trabajo, se opta por la segunda. Estas, son algunas de las modificaciones del archivo platformIO.ini. El resto

```

1 [env:uno]
2 platform = atmelavr
3 board = uno
4 framework = arduino
5 lib_extra_dirs=D:\proyecto_final\rotador_antena_arduino_uno\rotador_antena_iar\include\
6 build_flags = -Wchar-subscripts
7         -Wunused-variable
8         -DCORE_DEBUG_LEVEL=5
9         -I\include\control_motores
10        -I\include\lecturas_encoders
11
12 lib_deps =
13         paulstoffregen/Ethernet@0.0.0-alpha+sha.9f41e8231b
14         marcoschwartz/LiquidCrystal_I2C@^1.1.4

```

codigo 10: Modificaciones realizadas sobre el archivo platformio.ini

de las modificaciones, están relacionadas con el directorio de trabajo, y con avisos relacionados al compilador. El archivo platformio.ini resulta finalmente de la siguiente manera:

Todos estos archivos y modificaciones, se encuentran en el sistema de control de versiones con soporte para git: github. El enlace al proyecto es [https://github.com/gaston-cb/control\\_antena\\_iar](https://github.com/gaston-cb/control_antena_iar)

### B.3 Carga de software dentro del microcontrolador

Una vez, se ha creado un proyecto, y se configura, el siguiente paso, es crear un primer programa, y cargarlo dentro del microcontrolador. Para esta sección, puede utilizar algún ejemplo que este incluido dentro del software platformio, o puede buscar algún ejemplo en Internet. Una vez realizado este proceso, se crea un nuevo proyecto, o se abre algún proyecto existente dentro de platformIO, y en la parte inferior izquierda de la pantalla, aparecen los iconos para compilar, cargar,etc. Estos iconos son los siguientes:

-  0 △ 4 Marca errores de sintaxis y de compilación.
-  Acceso al menú principal de platformIO.
-  compilar. Genera los archivos binarios. Puede cambiarse el directorio de compilación mediante el archivo platformIO.
-  Sube el archivo a la placa arduino. No requiere seleccionar el puerto. El programa analiza los puertos, y los sube automáticamente.
-  Borra todos los archivos generados por la compilación.
-  Visualizar el puerto serie.

Presionando el botón que tiene una flecha hacia la derecha, con el microcontrolador conectado a la PC, se carga el software de forma automática.

## B.4 Compilación condicional

La compilación condicional consiste en utilizar lo que se denomina directivas de compilación. El código es muy similar a lo que se realiza en programación, pero puede compilar diferentes códigos. Para realizar este tipo de compilación, deben definirse banderas o flags de compilación, y con una estructura similar al “if - else”, puede seleccionar el código que se carga dentro del microcontrolador. Para ilustrar la compilación condicional, se utiliza el siguiente código

```

1 #define BANDERA 5
2
3 void setup()
4 {
5 #if BANDERA<5
6     Serial.begin(9600) ;
7 #else
8     Serial.begin(115200) ;
9 #endif
10 ...
11 }
12
13 void loop()
14 {
15 ...
16 }
```

codigo 11: Código ejemplo para ilustrar la compilación condicional.

En el código anterior, se define la bandera, con el nombre BANDERA, y se le asigna el valor de 5.

Luego dentro de la función setup, se encuentran las sentencias de las líneas 5 a 9. En el código de ejemplo, cambia el baudrate del puerto UART del microcontrolador. En este ejemplo, el baudrate seleccionado es de 115200bps(bits por segundo). Si por el contrario, por ejemplo, se pone la variable BANDERA en 2, el baudrate seleccionado sería el de 9600bps. Esto, puede usarse por ejemplo, para trabajar con dos arquitecturas diferentes, con el mismo código y que trabajen bajo el entorno Arduino.

La compilación condicional, pueden anidarse tantos “if-elsif-else” como se deseen, y puede ser tan complejo como un lenguaje de programación propio. En este documento, la única sentencia que se ha usado de las directivas de compilación es esta, y es la única que se muestra.

## B.5 Sentencias sobre el entorno arduino

El código Arduino, provee de sus funciones, las cuales permiten el manejo del hardware con una mayor flexibilidad, sin entrar en el detalle del conocimiento profundo del hardware. Algunas de las sentencias provistas por su lenguaje propio son:

- **analogRead(PORTPIN)**: devuelve el valor del puerto analógico digital. En PORTPIN, debe ingresar el valor del puerto analógico que se desea leer.
- **pinMode(NUMBER\\_PIN,MODE)**: permite poner el puerto en tres modos distintos, estos son:
  - INPUT: pone el pin como entrada
  - INPUT\_PULLUP: pone el pin como entrada, pero pone una resistencia de pull-up desconocida. Según [11] esta oscila entre 20KΩ y 50KΩ

- OUTPUT: permite poner el puerto como salida.
- **digitalRead(PORTPIN)** y **digitalWrite(PORTPIN,STATE)**: En el caso de digitalRead, lee el puerto de entrada, y devuelve el estado de este. En el caso de digitalWrite, escribe el puerto definido como salida, STATE, puede ser LOW o HIGH.

Estas son solo algunas de las funciones que provee el entorno Arduino para el manejo de puertos del microcontrolador. Para una lista completa, la puede obtener de <https://www.arduino.cc/reference/en/>

## B.6 Conversor Analógico Digital

En el mundo real, las señales electricas, son analógicas. Estas, se pueden digitalizar, para poder procesarlas dentro de un microcontrolador,pc ,etc. Este transpaso de señales eléctricas analógicas a digitales, las realiza un componente de hardware, que se denomina conversor analógico-digital. Este dispositivo, convierte un nivel de tensión eléctrica en un número binario. La cantidad de bits que tenga este número binario, define la resolución del conversor. Además, en caso de trabajos futuros, debe considerarse el error introducido por el mismo, ya que el último bit en general es incierto (esto se denomina 1/2LSB en las hojas de datos generalmente).

En el caso del microcontrolador Arduino UNO, este posee integrado un conversor analógico-digital, lineal, de 10 bits de resolución. Al ser 10 bits, la cantidad de niveles que puede tomar es de  $2^{10} = 1024$ , o sea que va desde 0 a 1023. La tensión de este, va desde 0 a 5volts. Para calcular cual es el mínimo valor de tensión, que hace que se incremente en un bit viene dado por:

$$\frac{5V}{1024} = 0,0048V$$

O sea, que el conversor posee una resolución de aproximadamente 0.5mv.

Para leer el conversor analógico digital, debe usarse la función analogRead(PORTPIN), y luego almacenar ese valor en una variable. Este valor, estará, entre 0 y 1023. El código para leer el conversor analógico digital, por ejemplo, del puerto A0, es el siguiente:

```
1 #define PORT_PIN A0
2
3 Leer puerto analógico A0
4
5 int lect_ad = analogRead(PORT_PIN) ;
```

---

# bibliografía

- [1] Documentación de arduino UNO, disponible en: <https://store.arduino.cc/usa/arduino-uno-rev3>.
- [2] Documentación de EDU-CIAA, disponible en: <http://www.proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:edu-ciaa:edu-ciaa-nxp>.
- [3] Documentación STM32VLdiscovery, disponible en: <https://www.st.com/en/evaluation-tools/stm32-discovery-kits.html>.
- [4] W. Stallings, J. Verdejo, R. Velarde y J. Barrientos, *Comunicaciones y redes de computadores*, ép. Fuera de colección Out of series. Pearson Educación, 2004, ISBN: 9788420541105.
- [5] A. Tanenbaum, *Redes de computadoras*. Editorial Alhambra S. A. (SP), 2003, ISBN: 9789702601623.
- [6] D. A. R. P. Agency. “*Internet Protocol*”, disponible en: <https://tools.ietf.org/html/rfc791>.
- [7] N. W. Group. “*Estandar DHCP*”, disponible en: <https://tools.ietf.org/html/rfc2131>.
- [8] documentación de hamlib, disponible en: <http://hamlib.sourceforge.net/html/hamlib-utilities.7.html#NAME>.
- [9] A. Csete. “*Gpredict User Manual*”, disponible en: [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi62aXNlY7qAhU1GLkGHYohBuMQFjADegQIAxAB&url=https%3A%2F%5C%2Fgithub.com%5C%2Fcsete%5C%2Fgpredict%5C%2Freleases%5C%2Fdownload%5C%2Fv2.2%5C%2Fgpredict-user-manual-2.2.pdf&usg=A0vVaw3NhB1bo3tV\\_opYZX1X8szh](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi62aXNlY7qAhU1GLkGHYohBuMQFjADegQIAxAB&url=https%3A%2F%5C%2Fgithub.com%5C%2Fcsete%5C%2Fgpredict%5C%2Freleases%5C%2Fdownload%5C%2Fv2.2%5C%2Fgpredict-user-manual-2.2.pdf&usg=A0vVaw3NhB1bo3tV_opYZX1X8szh) (visitado 30-06-2019).
- [10] G. Zotti y A. Wolf. “*Stellarium 0.20.4 User Guide*”, disponible en: [https://github.com/Stellarium/stellarium/releases/download/v0.20.4/stellarium\\_user\\_guide-0.20.4-1.pdf](https://github.com/Stellarium/stellarium/releases/download/v0.20.4/stellarium_user_guide-0.20.4-1.pdf) (visitado 16-02-2019).
- [11] *Datasheet ATmega328P*, microchip, 2015. disponible en: [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\\_Datasheet.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf).
- [12] Documentación telescope server, disponible en: [http://svn.code.sf.net/p/stellarium/code/trunk/telescope\\_server/stellarium\\_telescope\\_protocol.txt](http://svn.code.sf.net/p/stellarium/code/trunk/telescope_server/stellarium_telescope_protocol.txt).
- [13] *W5100 Datasheet*, WizNet.Co., LTD, 2008. disponible en: [https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100\\_Datasheet\\_v1\\_1\\_6.pdf](https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf).
- [14] *PCF8574Remote8-Bit I/O Expanderfor I2C Bus*, Texas Instrument, 2015. disponible en: [https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100\\_Datasheet\\_v1\\_1\\_6.pdf](https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf).
- [15] G. Baume, *La esfera celeste*. Editorial de la Universidad Nacional de La Plata (EDULP), 2014.
- [16] J. E. M. Torres, *Elementos de astronomía observacional:la esfera celeste*. INAOE, 2013.

- [17] C. A. Miranda, P. L. Schlesinger, A. D. Valdez, J. A. Chiozza y C. V. Miranda, “Procedimiento para el apuntamiento de una antena terrestre, a un satélite geoestacionario BSS (broadcast sattelite service),” *Extensionismo, Innovación y Transferencia tecnológica*, vol. 4pág. 294, mayo de 2018.
- [18] A. S. Hernandez Cruz, “Diseno e implementacion de un sistema de rastreo y posicionamiento de una antena de reflector parabolico para comunicaciones con satelites geoestacionarios y orbitales,” Tesis doct., UNAM, 1991. disponible en: <http://132.248.9.195/pmis2016/0162663/Index.html>.
- [19] Documentación de platformIO, disponible en: <https://docs.platformio.org/en/latest/what-is-platformio.html>.