

UNIVERSIDAD BLAS PASCAL**ONI²ET 2017****21º Olimpiadas Nacionales de Informática, Innovación, Electrónica y Telecomunicaciones**

TITULO: SUN TRACK "SEGUIDOR SOLAR"

ALUMNOS EXPOSITORES:

- ✓ PANDULLO, AGUSTIN NICOLAS
DNI: 40.124.695
6º AÑO ELECTRONICA
- ✓ PERA, SEBASTIAN EZEQUIEL
DNI: 40.388.503
6º AÑO ELECTRONICA

DOCENTES ORIENTADORES:

- ✓ Garcia Diego
D.N.I. 35.567.305
- ✓ QUIROGA, JONHATAN
D.N.I. 34.569.425
- ✓ ORUE, Jorge
D.N.I.32.337.890

DOCENTES ACOMPAÑANTES:

- ✓ VIDAL BARRIA, Karina Paz
DNI: 18.844.253

ESCUELA: INDUSTRIAL Nº 6 "X BRIGADA AEREA"

PROVINCIA: SANTA CRUZ

LOCALIDAD: GUER AIKE

CIUDAD: RIO GALLEGOS

AÑO: 2017

FECHA DE INICIO: 1/04/2017

DURACION EN SEMANAS: 22 SEMANAS

ESFUERZO EN HORAS: 140

PERSONAS AFECTADAS: 4 PERSONAS AFECTADAS CON UN PROMEDIO DE 5 HS DE TRABAJO SEMANALES





ÍNDICE

Contenido	Página
Introducción	2
Objetivo	2
Alcances/Destinatarios	2
Resumen	2
Antecedentes	2
Etapas de investigación	4
Diagrama de Gantt	4
Funcionamiento del sistema	4
Diagramas de bloque	5
Definición de cada modulo	5
Esquemas o Circuitos eléctricos	6
Placa Arduino MEGA 2560 caracterización de entradas y salidas	6
Circuito de potencia para controlar motores PaP: driver Pololu A4988	6
Sensores: Explicación del funcionamiento de los sensores seleccionados	7
Finales de carrera de calibración, interruptor de encendido o apagado.	7
Circuito de ubicación GPS: Módulo NEO 6M – V2	7
Placa reguladora de tensión Placa para módulos de Arduino	8
Placa para módulos de Arduino A4988 y GPS	9
Mediciones tomadas	10
Esquema del prototipo	11
Costos de proyecto y su desarrollo	13
Bibliografía y recursos de Internet consultados	13
Diagrama modulare y diagramas de flujo	14
Mapa de hardware	22
Codificación	22



Introducción

Objetivo:

Diseñar y construir un prototipo capaz de realizar un seguimiento del sol, el cual permita medir parámetros como, por ejemplo, el espesor óptico de aerosoles, radiación ultravioleta, índice U-V, ozono, entre otras.

Alcances:

El prototipo se considera adaptable y utilizable para cualquier ente público o privado destinado a la investigación y estudio del medio ambiente, como así también en el ámbito meteorológico, el mismo puede ser utilizado en cualquier zona sin importar el clima y pudiendo adaptar sus datos a la ubicación actual y zona horaria.

Destinatarios:

Principalmente el prototipo está destinado a profesionales que se dedican al estudio e investigación del ambiente, como así también, de la meteorología, dando la seguridad que las mediciones obtenidas por el mismo sean precisas y confiables.

Resumen del proyecto:

El proyecto se comenzó a desarrollar en el mes de marzo del corriente año en el Industrial N°6, en vista de desarrollar un prototipo que integre los conocimientos adquiridos durante la carrera, impulsado por la carencia de datos como la “dispersión de aerosoles” que se encuentra en la atmósfera, que afecta directamente al clima y medio ambiente de la Patagonia, razón por la que fue propuesta la realización de un dispositivo automatizado capaz de realizar mediciones referentes a dicha atmosfera.

Después de recolectar información de diversas fuentes y hablar con un profesional que se desempeña en el área de meteorología de la ciudad de Rio Gallegos, se decidió realizar un seguidor solar utilizando una placa Arduino Mega 2560 y sus módulos, GPS, pololu A4988.

El principal objetivo del prototipo es conocer con mayor certeza que partículas componen nuestra atmósfera y que riesgos representan dichas partículas, o precauciones que deben ser tomadas ante los dichos Hoyos de Ozono (que presentan un alto pico de radiación de rayos U-V).

Antecedentes:

El dispositivo está basado en un prototipo de la NASA, el cual tiene la finalidad de medir espesor óptico de las partículas. El mismo se utiliza para estudiar la dispersión de aerosoles (partículas sólidas de distintos tipos, materiales y tamaños) en la atmósfera y como este fenómeno afecta al planeta.

Estudios recientes han mostrado la importancia de los aerosoles atmosféricos como agente modificador del clima. Estos constituyentes atmosféricos pueden afectar al clima tanto directamente, mediante la dispersión y absorción de la radiación, como indirectamente produciendo cambios en la distribución en tamaños y concentraciones de las gotículas que constituyen las nubes.

Se desea diseñar un prototipo similar al anteriormente mencionado con componentes accesibles y precios económicos capaz de realizar mediciones de forma precisa.

Ya realizadas las investigaciones de los dispositivos, los cuales se presentan en la tabla siguiente con la descripción de los mismos, se establecerá a continuación las ventajas y desventajas de nuestro prototipo:


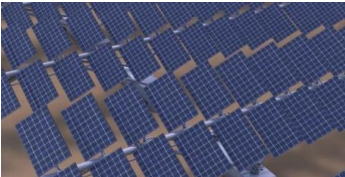


➤ Ventajas del prototipo:



- Es transportable y liviano.
- Cumple su finalidad y es de bajo costo.
- Es de un tamaño considerablemente pequeño.
- Es impermeable al agua.

➤ Desventajas del prototipo:

- Carece de una batería, por lo tanto tiene que estar conectado a la red en todo momento.
- Su cable de datos es corto.
- Al producirse alguna caída, este se puede dañar.

Prototipo – dispositivo	Descripción	Origen
CIMEL Electronique 318A 	Es un sistema de medición, radiómetro espectral de energía solar que mide las radiancias del sol, en una serie de longitudes de onda fijas dentro del espectro visible y del infrarrojo cercano.	Estados Unidos
Seguido solar 1 + 1 ejes horizontales - ADES Tempero Grup 	<p>La superficie se mantiene siempre perpendicular al sol. Existen de dos tipos:</p> <ul style="list-style-type: none"> • <i>Monoposte</i>: un único apoyo central. • <i>Carrousel</i>: varios apoyos distribuidos a lo largo de una superficie circular. 	España
Seguidor solar de eje azimutal - AvantSolar 	La superficie gira sobre un eje vertical, giro de este a oeste con una inclinación fija de 15° al sur.	España
Seguidor solar de 1 eje horizontal – ADES Tempero Grup 	La superficie gira en un eje horizontal y orientado en dirección norte-sur. El giro se ajusta para que la normal a la superficie coincida en todo momento con el meridiano terrestre que contiene al Sol.	España



--	--	--

Etapas de investigación:

Etapa 1: Se realizó la investigación previa al tema de desarrollo del proyecto y con dicha información se planteó la idea del prototipo, la cual constara de un cabezal el cual tendrá sensores de luz, y sensores de espesor óptico, capaz de calcular la posición en la que se encuentra el sol y así realizarle un seguimiento al mismo. Y utilizando como base un prototipo ya existente optimizando sus gastos y mejorando su funcionamiento.

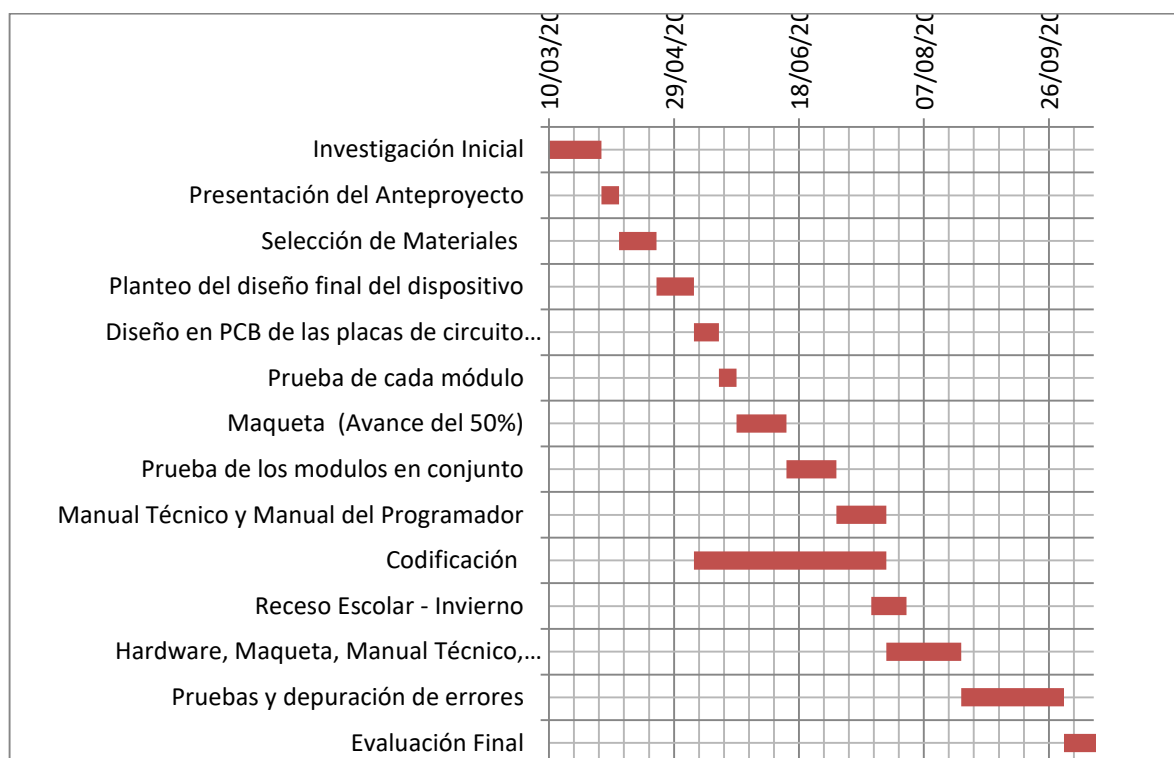
Etapa 2: Superada la etapa anterior, se realizó la selección y posterior compra de los materiales a utilizar, como así también se dio inicio al diseño del prototipo en base a su estructura, al planteo de algoritmos mediante la confección de diagramas modulares y diagramas de flujo, diagramas de bloques principales y secundarios del mismo, evaluación y selección de los circuitos eléctricos y placas de circuito impreso que se utilizarán en el prototipo.

Etapa 3: Una vez definido su diseño físico, como así los circuitos que lo integran, se procedió a realizar la implementación, mediante la construcción y montajes de los mismos, realizando las verificaciones pertinentes a cada módulo individual desarrollado. Por otro lado, se procedió a la codificación del algoritmo que hace al funcionamiento para placas Arduino, definidos en la etapa anterior.

4° Etapa: Se montó todo el dispositivo completo, verificaciones, mediciones reales y simuladas del funcionamiento del dispositivo, para eliminar problemas o errores de programación, aplicando software específico como Livewire, LabView, y Arduino IDE. Paralelamente se realizaron las primeras mediciones sobre el montaje elaborado.

Etapa 5: En esta se realizarán las correcciones pertinentes al informe y manuales del usuario y del programador, de la misma forma se hacen revisiones a la codificación y se pone a punto la maqueta en función a las correcciones antes mencionadas.

6° Etapa: Se realizó el ensamble final y las calibraciones necesarias sobre las posiciones de los motores de azimut y elevación. Así mismo, se efectuaron las mediciones finales y las correcciones de la programación.





Funcionamiento del sistema:

El dispositivo inicia al conectar el “cable serial” (el cable de comunicación entre la placa Arduino y el ordenador) y cargar la codificación mediante la plataforma Arduino IDE, una vez que cargada la codificación se debe cerrar el switch de la maqueta para dar paso a la alimentación. Ambos cables, tanto el de alimentación como el de comunicación deben estar siempre conectados.

El programa comenzara cuando la codificación sea cargada correctamente a la placa Arduino, mostrando un mensaje de bienvenida en la pantalla del ordenador mediante el monitor serial de Arduino IDE y quedando a la espera del ingreso de la tecla “A” a través del teclado del ordenador, al ingresar la letra ya mencionada el prototipo hará su calibración girando un motor a la vez hasta que cada uno toque su respectivo final de carrera.

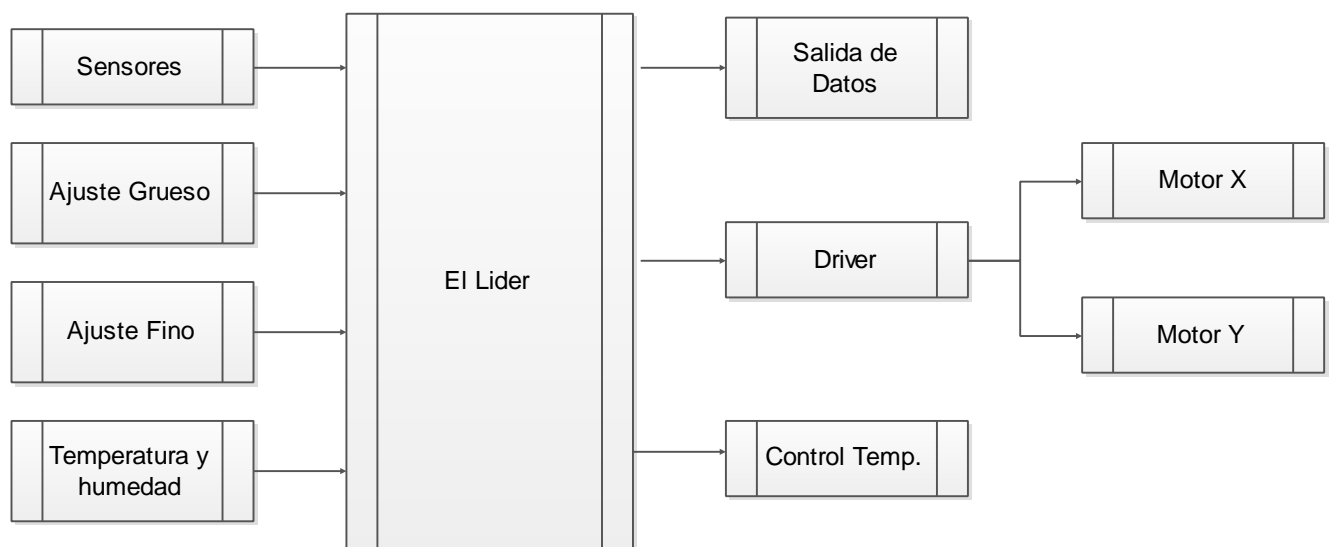
Finalizada la calibración el programa se mantendrá a la espera del ingreso de la letra “B”, cuando sea ingresada la plataforma LabView utilizara los datos del módulo GPS y realizara un algoritmo del cual la resultante son los valores en grados que deben moverse cada motor para apuntar en dirección al sol.

Después de este proceso, al presionar la tecla “C” se muestran en pantalla los valores de azimut y elevación y se acciona el movimiento de los motores hasta posicionarse cada uno en los ángulos recién mencionados de azimut y elevación.

Registrará en todo momento la temperatura interna del dispositivo, con la finalidad de proteger los componentes, y la humedad del aire externo del mismo, asegurando que el clima sea adecuado para el correcto funcionamiento del prototipo. En caso de que la temperatura interna del mismo sea muy elevada se pondrá en marcha el sistema de refrigeramiento del dispositivo. Y dado el caso de que la humedad en el exterior sea alta no se realizara medición ni se accionara el movimiento de los motores, mostrando en pantalla “condiciones no aptas para realizar la medición”.

Una vez realizadas las calibraciones y mediciones necesarias con respecto al clima, se realizará la introducción de datos referentes a la localización actuales en los que se encuentra el dispositivo de forma automática, extrayendo los valores de latitud, longitud, altitud, hora y fecha el dispositivo del módulo GPS.

Diagrama en bloque principal:



Definicion de cada modulo:

- **Sensores:** Capaces de captar la longitud de onda (LO) de una señal electromagnética y generar una señal proporcional a la LO absorbida
- **Ajuste grueso:** Permite el ingreso/lectura de los parámetros tales como longitud, latitud, altitud, fecha y hora del lugar en donde se encuentra el prototipo.



- Ajuste fino: Sensa la intensidad lumínica del sol para realizar un ajuste más preciso con el fin de seguir la trayectoria del sol.
- Temperatura y humedad: Mide temperatura interna del prototipo y humedad exterior.
- El Líder: Realiza los procesos y cálculos para controlar las salidas que permiten el posicionamiento del seguidor mediante los datos de entrada.
- Salida de datos: Se encarga de proveer los datos procesados para ser visualizados por una pantalla o PC.
- Driver: Brinda el control a los motores para el ajuste de la posición seguidor.
- Motor X: Controla la rotación horizontal del sensor.
- Motor Y: Controla la rotación vertical del sensor.
- Control Temp: Busca mantener la temperatura interior del prototipo en un rango estable calentando o refrigerando según corresponda.

Esquemas o circuitos eléctricos:

Placa Arduino MEGA 2560 caracterización de entradas y salidas:

La automatización del prototipo se definió por medio de una placa programable Arduino Mega 2560. Esta placa posee un micro controlador ATmega2560, que realiza los procesos que le hayan sido programados. Como entradas tenemos ocho, la información que nos brindan los valores del GPS (RXPin2) y la calibración por la cual se utilizarán dos finales de carrea (inPin2 e inPin1), el sensado de temperatura y humedad (STH), y la información correspondiente del ajuste fino (Itsensor, rtsensor, ldsensor y rlsensor).

Como salidas se tiene en cuenta los dos drivers controladores de motores, para controlar la cantidad de pasos que realizara los dos motores están los pines (Step1 y Step2), para controlar la dirección del sentido de los motores se utilizan los pines (Dir1 y Dir2) y para que el driver sepa si mandarle corriente al motor están los pines (Enable1 y Enable2). Incluidas en las salidas se tiene el control de temperatura por medio del pin (Vent) y transmisión de datos al GPS (TXPin2).

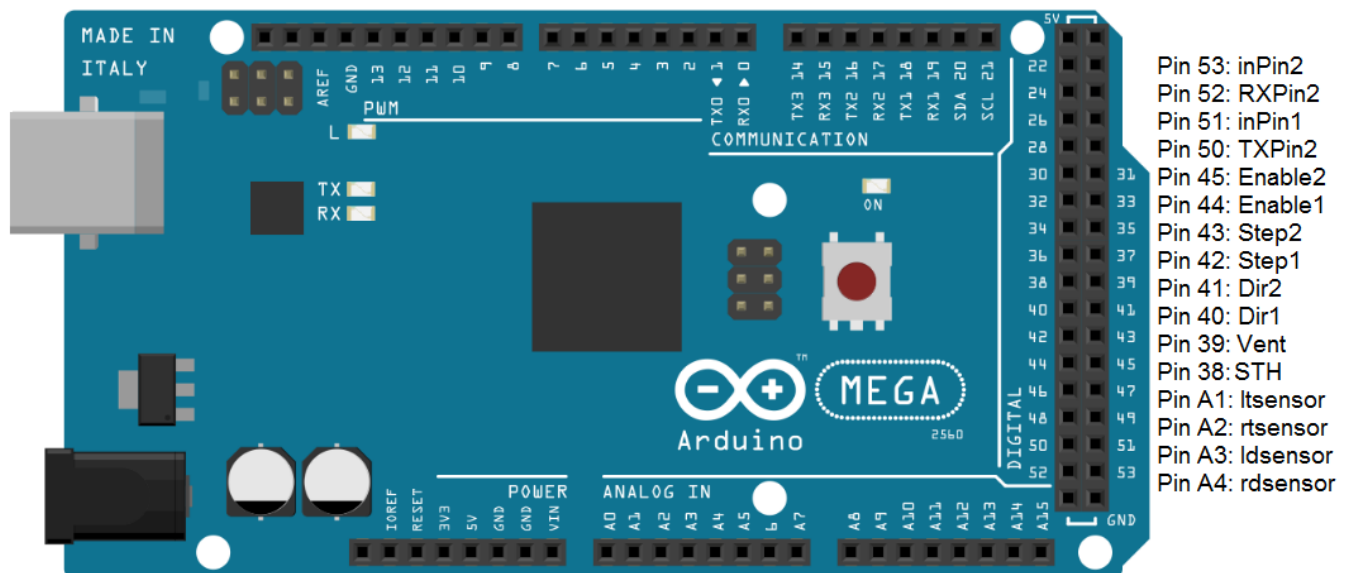


Figura 2- configuración de entradas y salidas de la placa Arduino MEGA 2560

Circuito de potencia para controlar motores PaP: driver Pololu A4988



Al estar limitada la placa Arduino por una potencia máxima que no cubre los requerimientos para el funcionamiento de los motores empleados, se decidió utilizar circuitos externos que sean capaces de trabajar con un mayor consumo de potencia. Por eso se emplea el driver Pololu A4988, que soporta el consumo de los motores, estos mismos se controlan por los puertos (STEP Y DIR), que controlan la cantidad de pasos realizados por el motor y el sentido del mismo. Permitiendo el correcto funcionamiento de los motores y la placa programable.

En la figura 3 se visualiza el montaje superficial del driver controlador de motores PaP A4988.

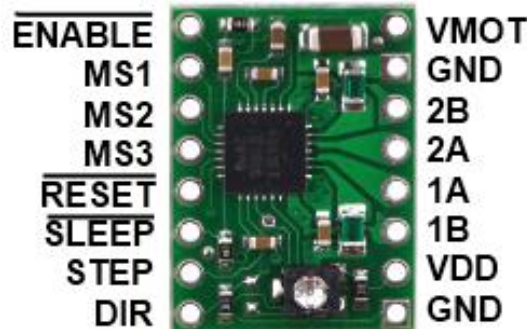


Figura 3 – Driver A4988.

Sensores: Explicación del funcionamiento de los sensores seleccionados

Sensor de temperatura y humedad.

Este sensor se encarga de registrar la temperatura y humedad que hay en un cierto ambiente, aumentando o disminuyendo la tensión de salida que entrega por el pin de SEÑAL. Este sensor es un módulo de Arduino, como se muestran en la Figura 4. Este módulo se alimenta con una tensión de 3.3V o 5V

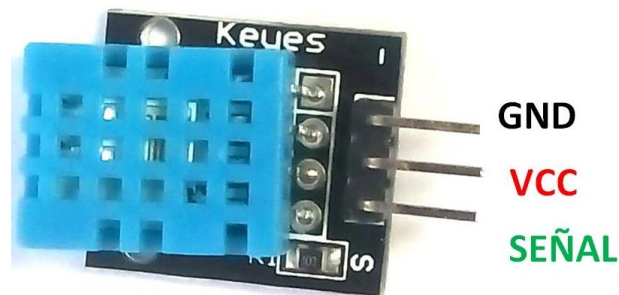


Figura 4 – especificaciones del sensor de temperatura y humedad DHT11

Finales de carrera de calibración, interruptor de encendido o apagado.

Estos dispositivos electromecánicos permiten desviar o interrumpir el curso de una corriente eléctrica, para que alguno de ellos intervenga directamente con placa programable Arduino. En la figura 5 se aprecian varios ejemplos que son utilizados en los circuitos.

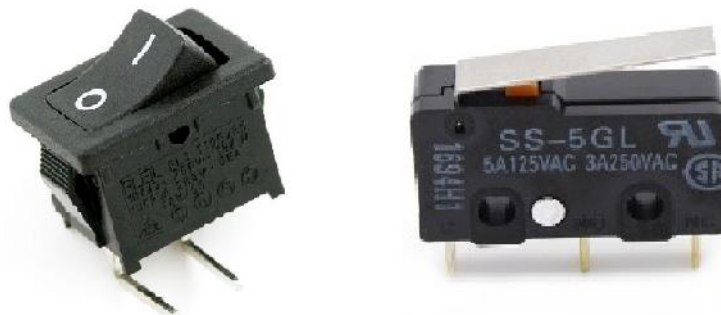


Figura 5 – A la izquierda un interruptor de dos puntos y a la derecha un final de carrera NA

Circuito de ubicación GPS: Módulo NEO 6M – V2

El circuito se puede visualizar en la figura 7, y tiene como funcionalidad establecer una conexión satelital, para extraer valores esenciales, que se utilizaran en el programa.



Posee cuatro pines uno de ellos es su alimentación (Vcc), otro es su conexión a masa (GND) y luego los últimos dos son de comunicación, el pin Tx es de transmisión de datos y Rx es de recepción de datos.

Este módulo funciona a 3V, pero sus parámetros máximos llegan hasta los 5V, también contiene un EEPROM (las memorias de tipo EEPROM es un derivado de la memoria - SD) con su correspondiente pila de tipo “botón” para mantener los datos intactos y una antena de cerámica que permite amplificar las señales recibidas de los satélites con los que se comunica.

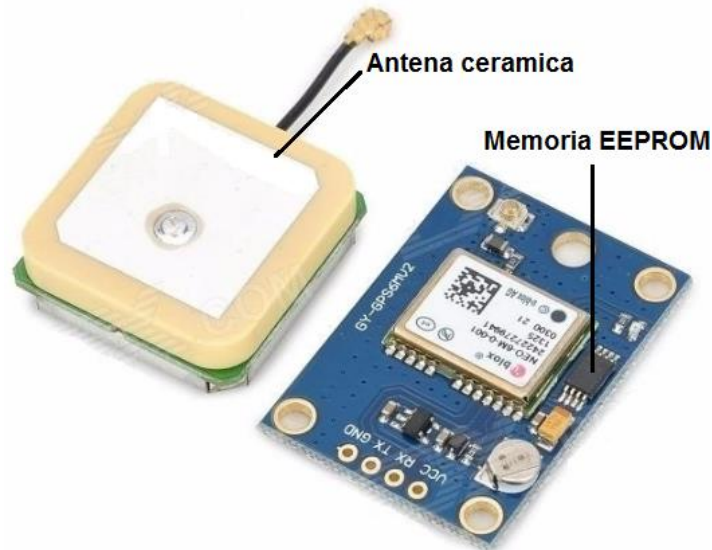


Figura 7 – Módulo NEO6M – V2

Placa reguladora de tensión

Esta placa convierte la tensión de la alimentación a corriente continua (C.C.) y regula sus salidas en 5V, para alimentar los módulos GPS NEO 6, Drivers pololu A4988 y sensores de humedad y temperatura DHT11, y 12V para alimentar los motores paso a paso, proporcionándoles también la corriente necesaria.

La tensión de salida 5V es regulada por un regulador de tensión LM7805.

Para regular la tensión de salida 12V y proporcionar la corriente necesaria que consumen los motores se utilizó un LM350, el cual se regula la tensión mediante un divisor de tensión utiliza los valores de resistencia de: R1= 120Ω; R2= 1000Ω. Tales valores fueron calculados para una determinada tensión de salida según la siguiente formula.

$$V_o = V_{ref} \cdot \left(1 + \frac{R_2}{R_1}\right) + I_{Adj} \cdot R_2$$

La cual se despejo hasta llegar a la siguiente ecuación:

$$R_1 = \frac{V_{ref} \cdot R_2}{V_o - V_{ref} - I_{Adj} \cdot R_2}$$

Calculando finalmente el valor de R1.

$$R_1 = \frac{1,25V \cdot 1000\Omega}{12V - 1,25V - 100\mu A \cdot 1000\Omega} = 117,4\Omega$$

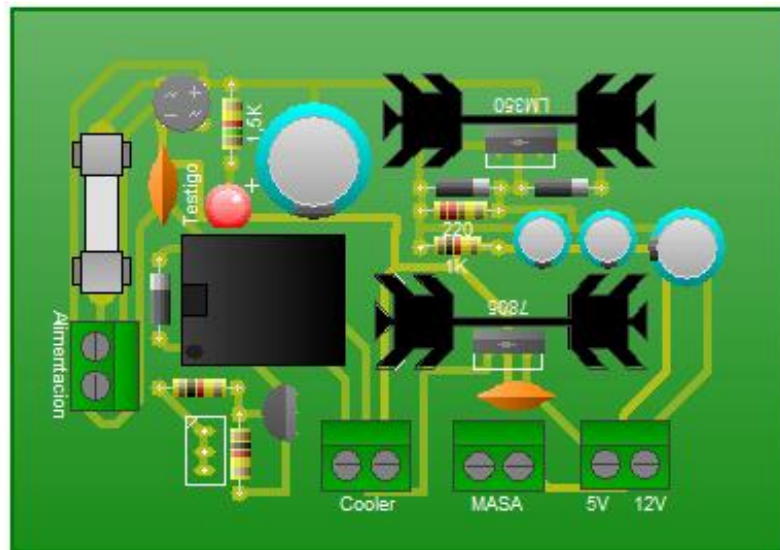


Figura 8 – Visualización del montaje superficial de la placa de alimentación.

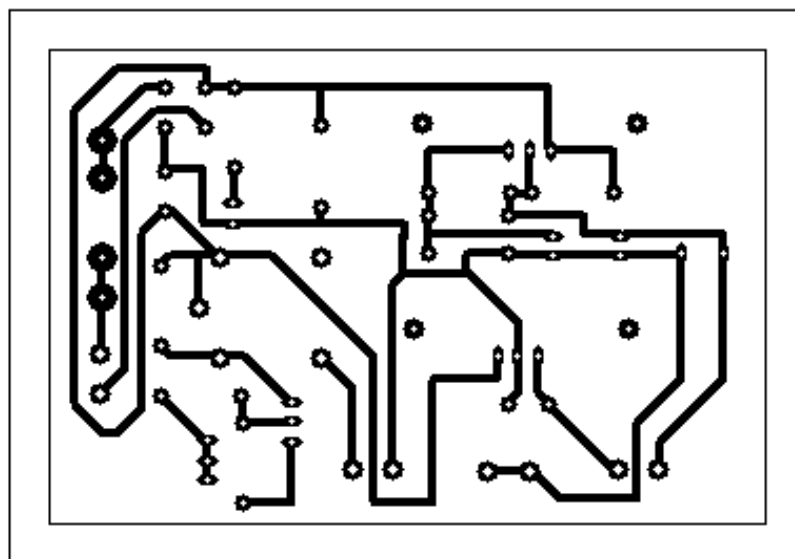


Figura 9 – diseño de la placa de alimentación

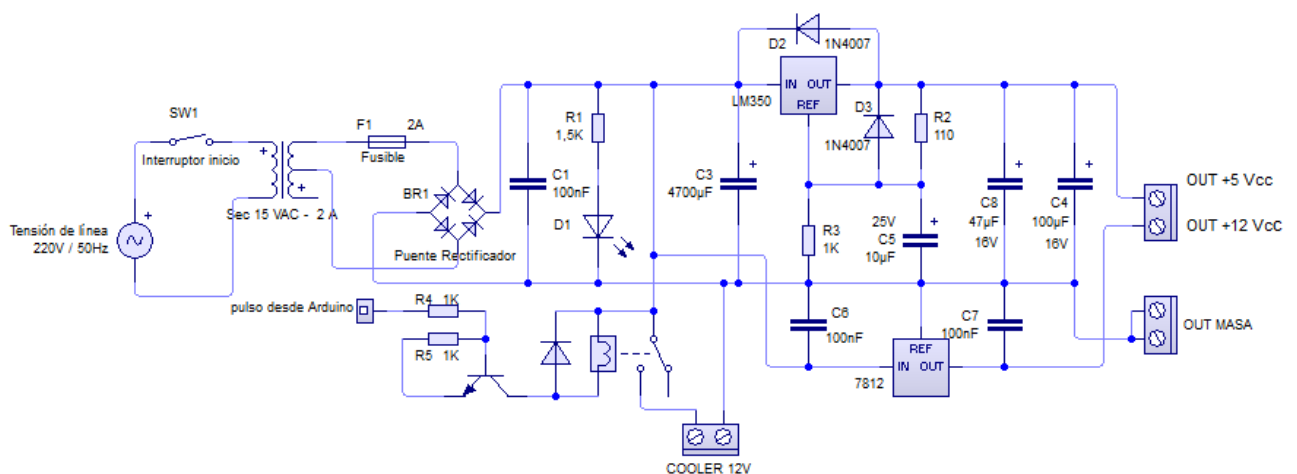


Figura 10 – esquema eléctrico de la placa de alimentación

Placa para módulos de Arduino A4988 y GPS

Placa diseñada para facilitar la conexión entre los módulos ya mencionados que utilizan el dispositivo, y la placa programable encargada de controlar las acciones de cada módulo.

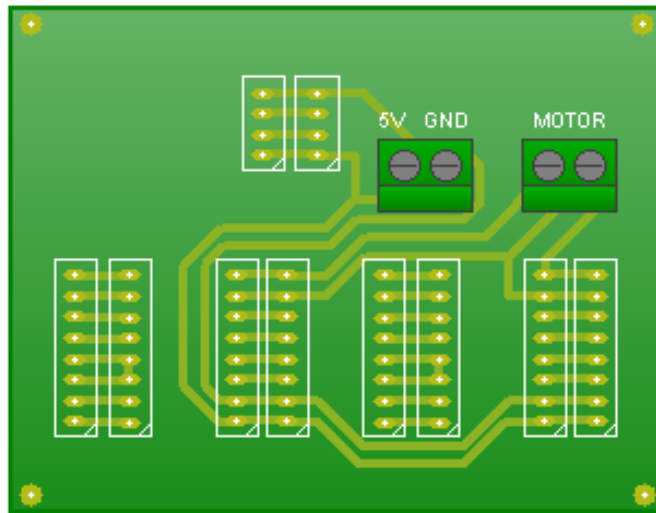


Figura 11 – visualización del montaje de la placa del A4988 y GPS

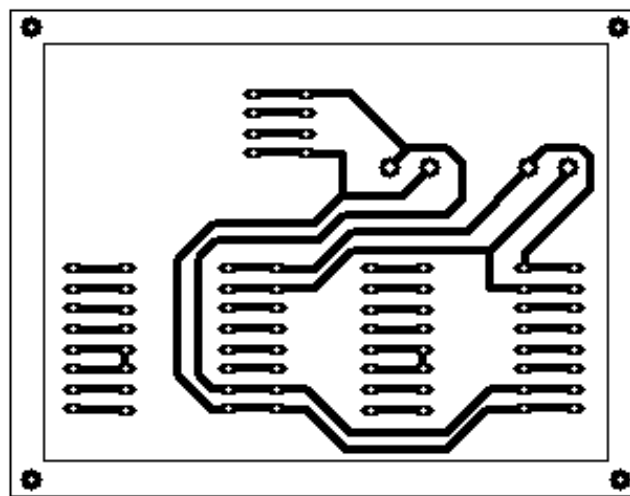


Figura 12 – Diseño de placa del A4988 y GPS

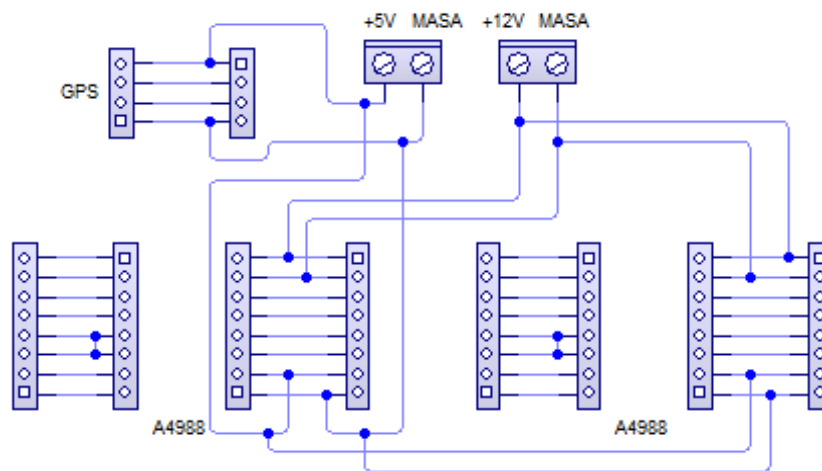


Figura 13 – Esquema eléctrico de la placa A4988 y GPS

Mediciones tomadas:

En la tabla 1 se presentan las mediciones tomadas en los diferentes puertos de entrada y salida, donde se pueden visualizar los valores de tensión que entrega cada uno.

Puerto	Conexión del	Tipos de datos	Tipos de señal	Medición
--------	--------------	----------------	----------------	----------



	Hardware	Salidas	Entradas	Digital	Analógica	
53	inPin2		+	+		0V a 5V
52	RXPin2	+		+		Datos
51	inPin1		+	+		0V a 5V
50	TXPin1		+	+		Datos
45	Enable2	+		+		0V a 5V
44	Enable1	+		+		0V a 5V
43	Step2	+		+		0V a 5V
42	Step1	+		+		0V a 5V
41	Dir2	+		+		0V a 5V
40	Dir1	+		+		0V a 5V
39	Vent	+		+		0V a 5V
38	STH		+	+		Datos
A1	ltsensor		+		+	0V a 5V
A2	rtsensor		+		+	0V a 5V
A3	ldsensor		+		+	0V a 5V
A4	rdsensor		+		+	0V a 5V

Tabla 1 – Mediciones en los puertos de entrada y salida

Esquema de prototipos:

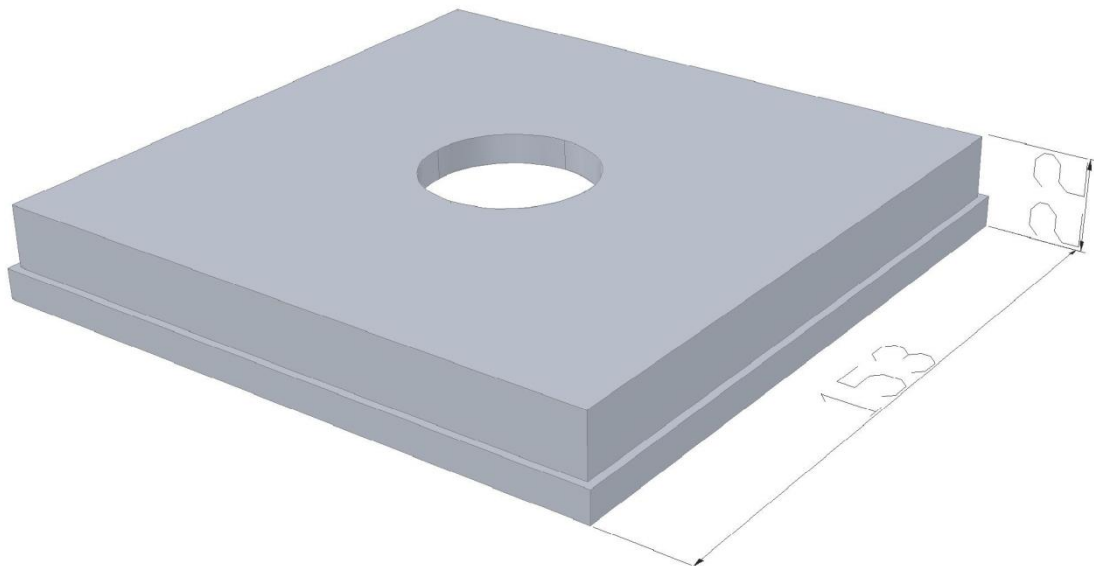


Figura 15 – Vista isométrica de la tapa del gabinete

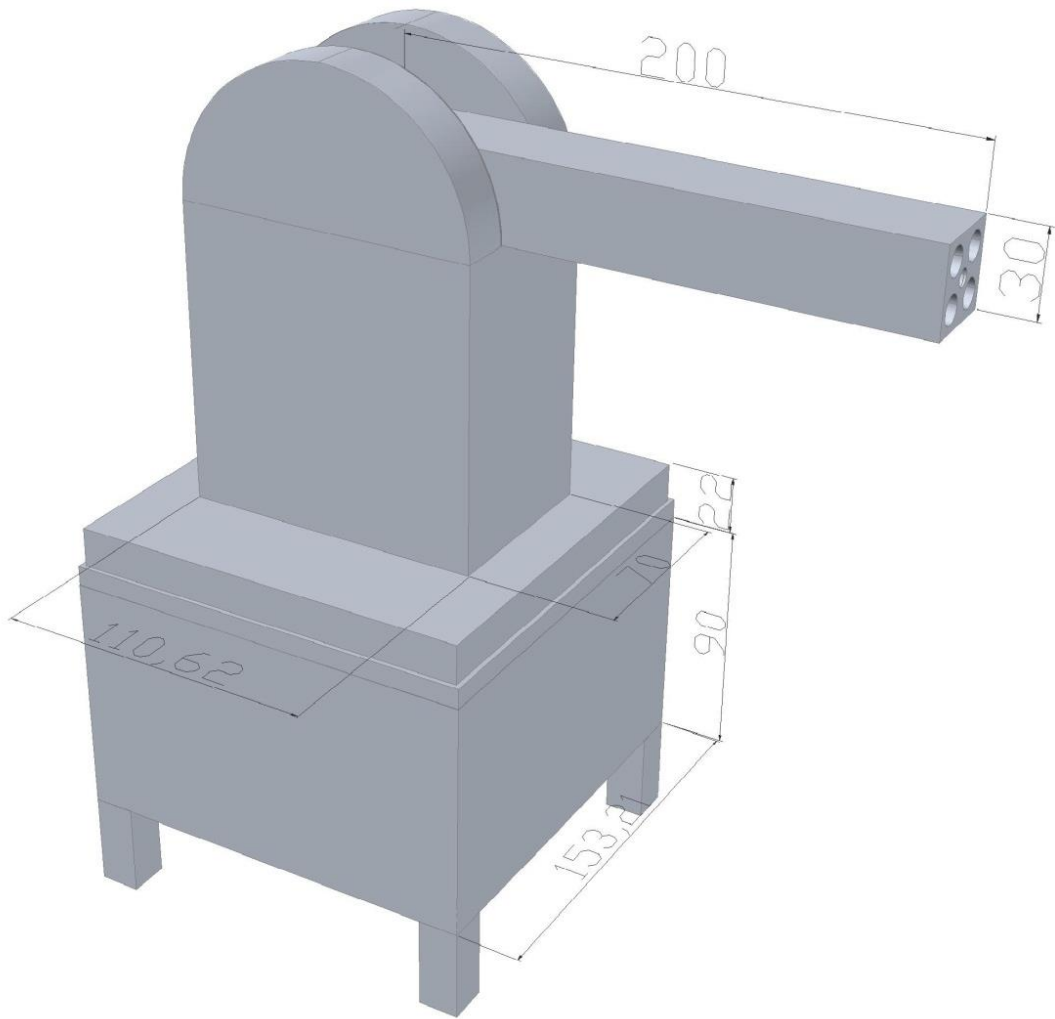


Figura 14 – Vista isométrica del prototipo

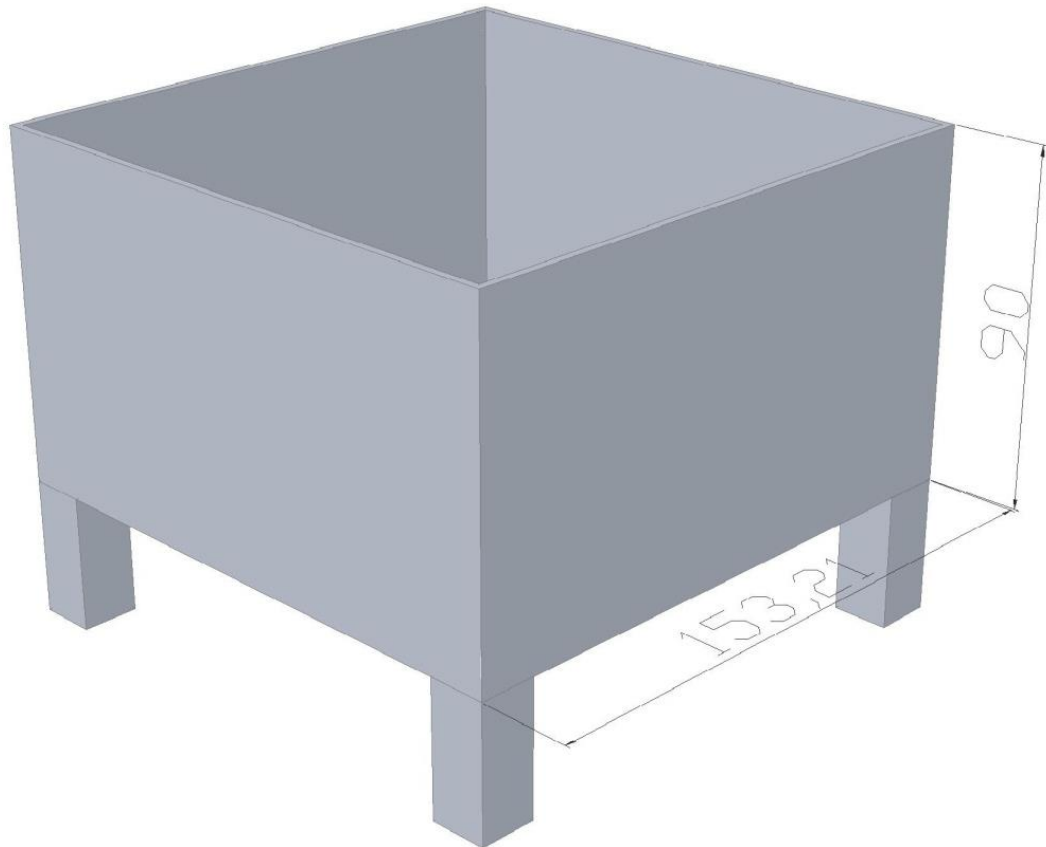


Figura 16 – vista isométrica del gabinete

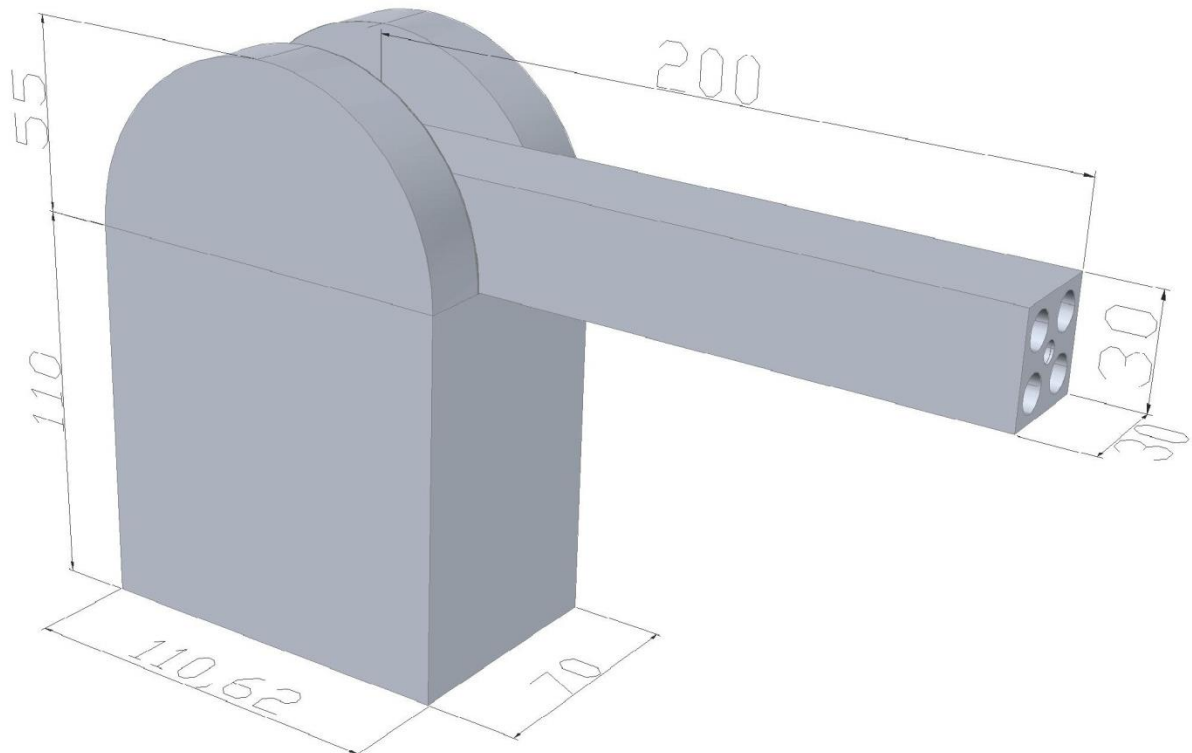


Figura 17 – vista isométrica del cabezal del prototipo

Costos del prototipo:



Cantidad	Componentes / materiales	Precio unidad	Precio total
1	Placa Arduino Mega 2560	\$335,00	\$335,00
1	Módulo Arduino GPS	\$425,00	\$425,00
1	Caja de paso estanco 15x15x10	\$340,20	\$340,20
1	Placa impresa 15 x 15	\$80,00	\$80,00
2	Modulo Driver Motor PaP	\$39,80	\$79,80
1	Transformador 220V a 15V	\$250,00	\$250,00
2	Motor PaP Bipolar	\$440,00	\$880,00
1	Rollo filamento impresora 3D	\$400,00	\$400,00
Total			\$2.790

Tabla 2 – costos del prototipo

Costos de desarrollo:

En la siguiente tabla se presentan los costos estimados correspondientes a la investigación, estudio, montaje, ensamblado, ensayo y mediciones del prototipo. Se tuvo en cuenta que el valor del trabajo de un técnico electrónico por hora es de 93 pesos argentinos. El tiempo empleado para el desarrollo del mismo se encuentra representado en horas. Cabe mencionar, que este último dato puede observarse de manera más detallada en el diagrama de Gantt (Figura 3).

	Tiempo aplicado [Hs]	Valor Hora de trabajo de un técnico electrónico [\$/Hs]	Subtotal [€]	Total [€]
Investigación y estudio	12 semanas a un promedio de 16 horas por semana.		17.856	
Montaje y ensamblado del prototipo	9 semanas a un promedio de 16 horas por semana.	93	13.392	41.013
Ensayo y mediciones	7 semanas a un promedio de 15 horas por semana.		9.765	

Tabla 3 – Costos de desarrollo.

Bibliografía y recursos de Internet consultados:

<https://tallerarduino.com/2012/12/24/sensor-dht11-humedad-y-temperatura-con-arduino/>

<https://www.promotec.net/sensores-dht11/#>

<http://www.avantsolar.com/somos.php>

<http://www.adessolar.com/energias-renovables/energia-solar/seguidor-solar>

<https://pubs.giss.nasa.gov/abs/gi03000u.html>

<https://en.wikipedia.org/wiki/AERONET>

<http://energiasolar.charito.com.ar/productos.html>

<https://www.pololu.com/product/1182>

<https://www.staticboards.es/blog/drv8825-vs-a4988/>

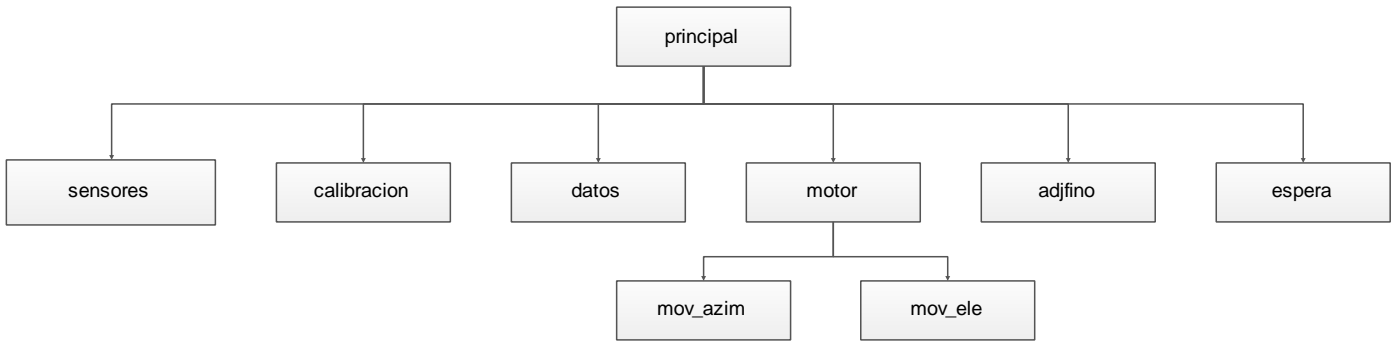
<http://www.todomicro.com.ar/investigacion-desarrollo-y-prototipado/251-modulo-gps-gy-neo6mv2-con-antena.html>

<http://miarduinounotieneunblog.blogspot.com.ar/2016/02/geolocalizacion-con-modulo-gps-gy.html>

<https://store.arduino.cc/usa/arduino-mega-2560-rev3>

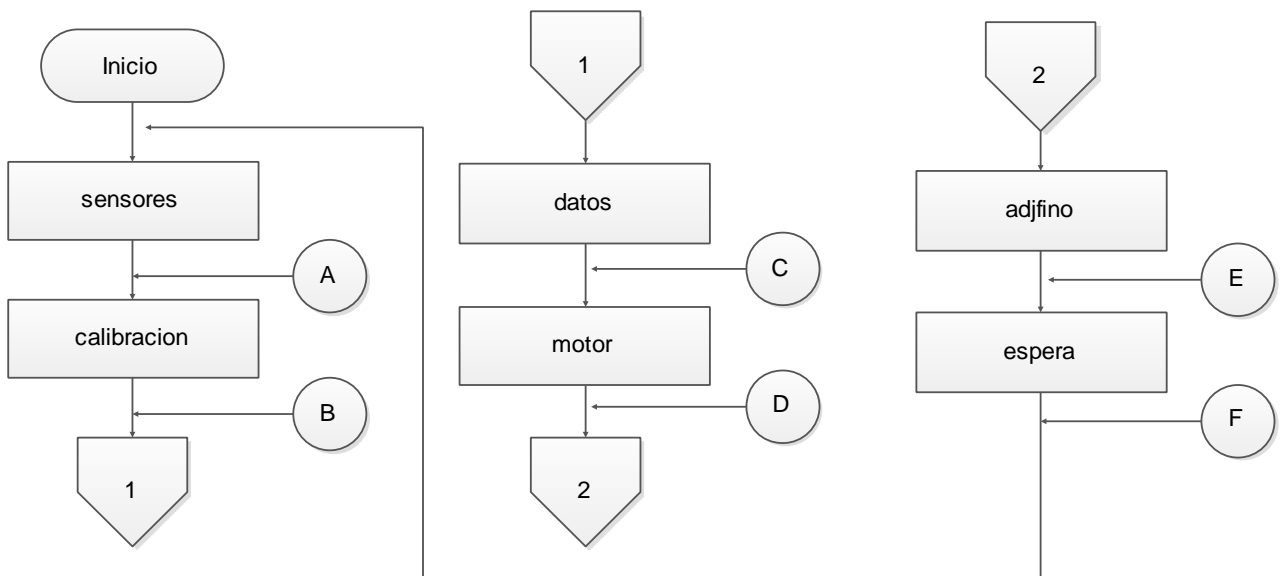
<https://www.arduino.cc/en/Guide/ArduinoMega2560>

Diagrama modular:

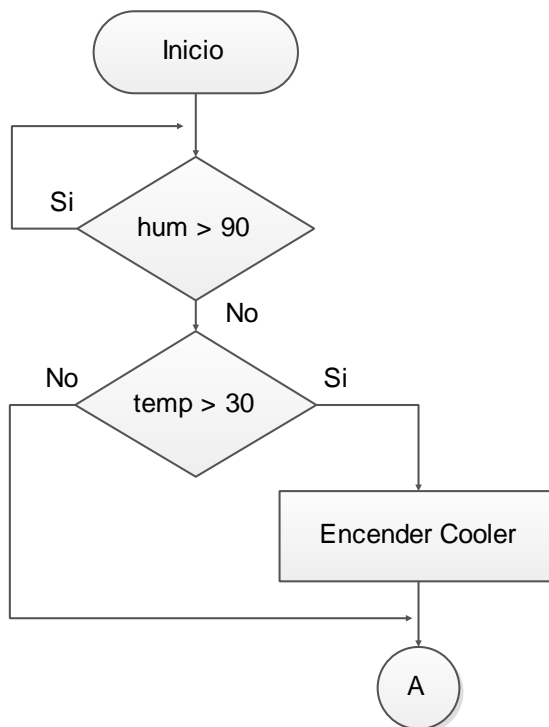


Diagramas de flujo:

principal:

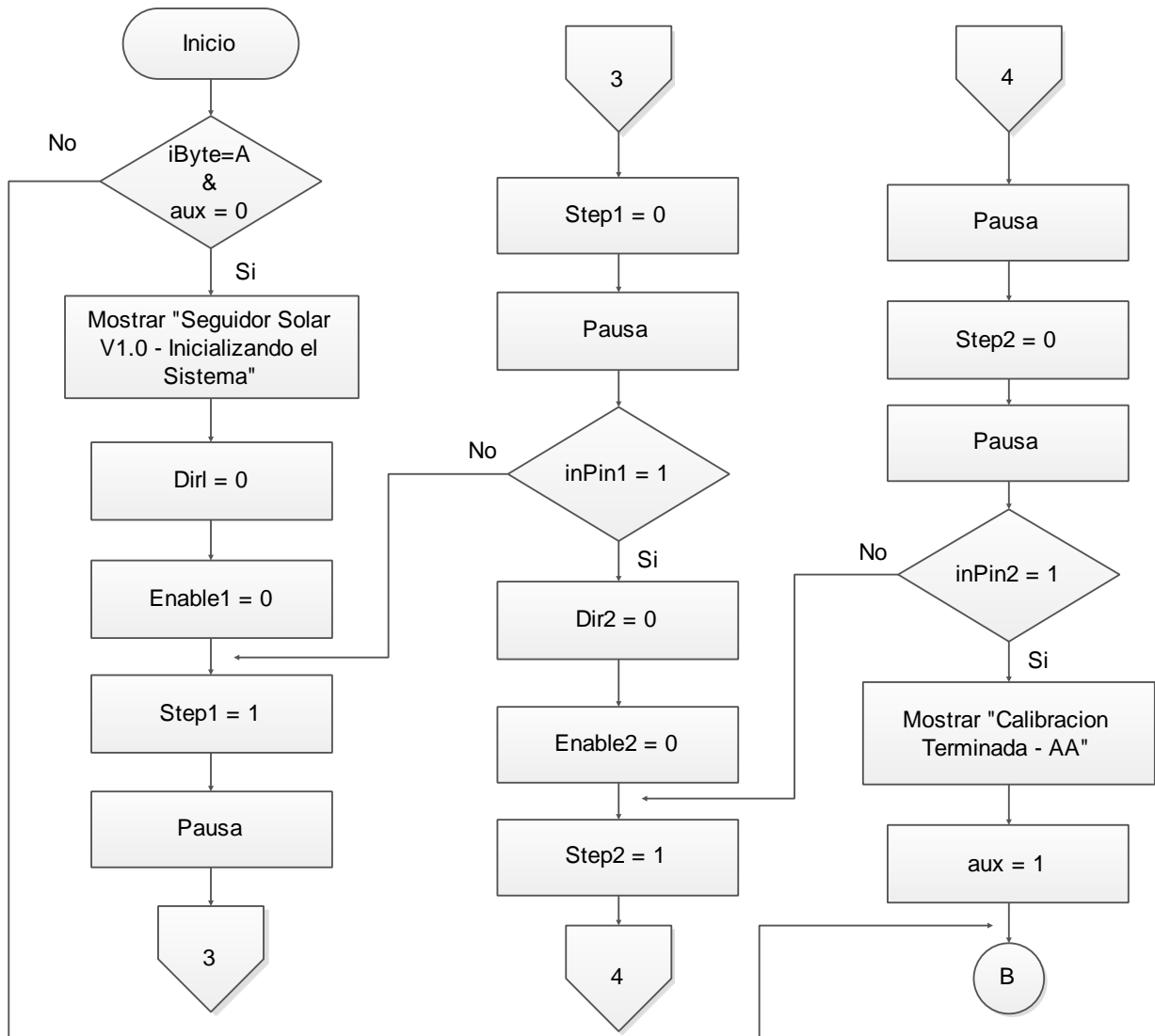


Sensores:



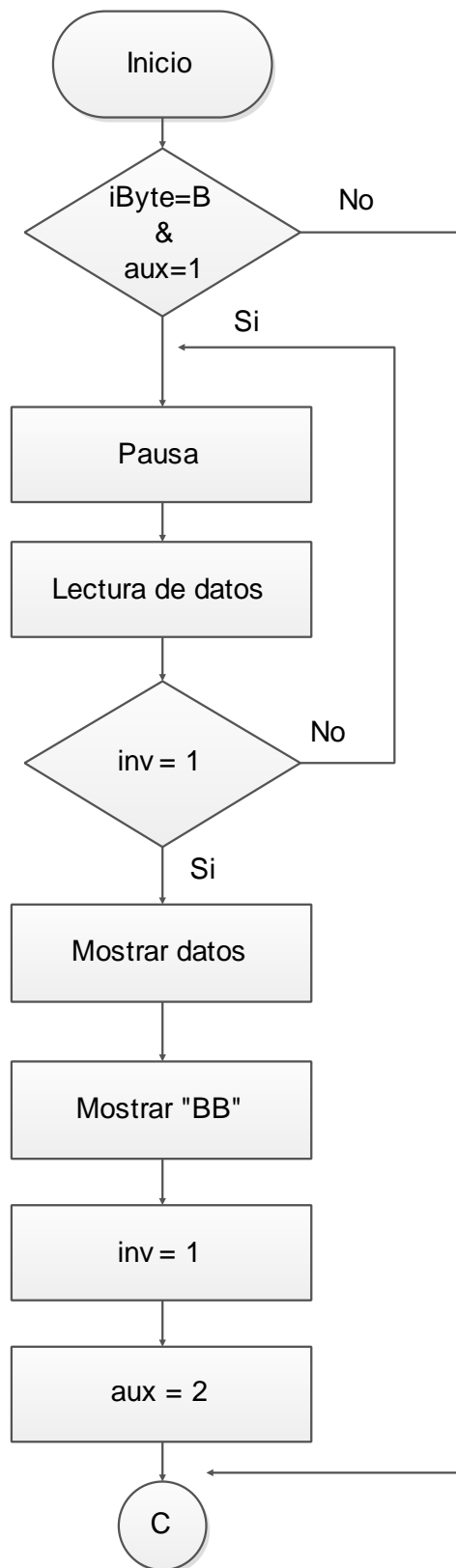


Calibración:



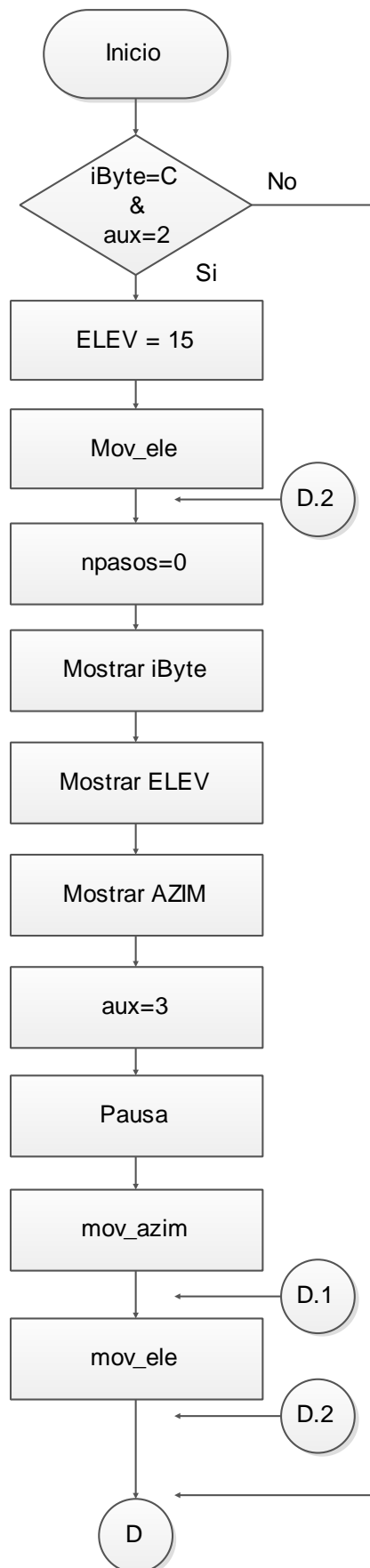


Datos:



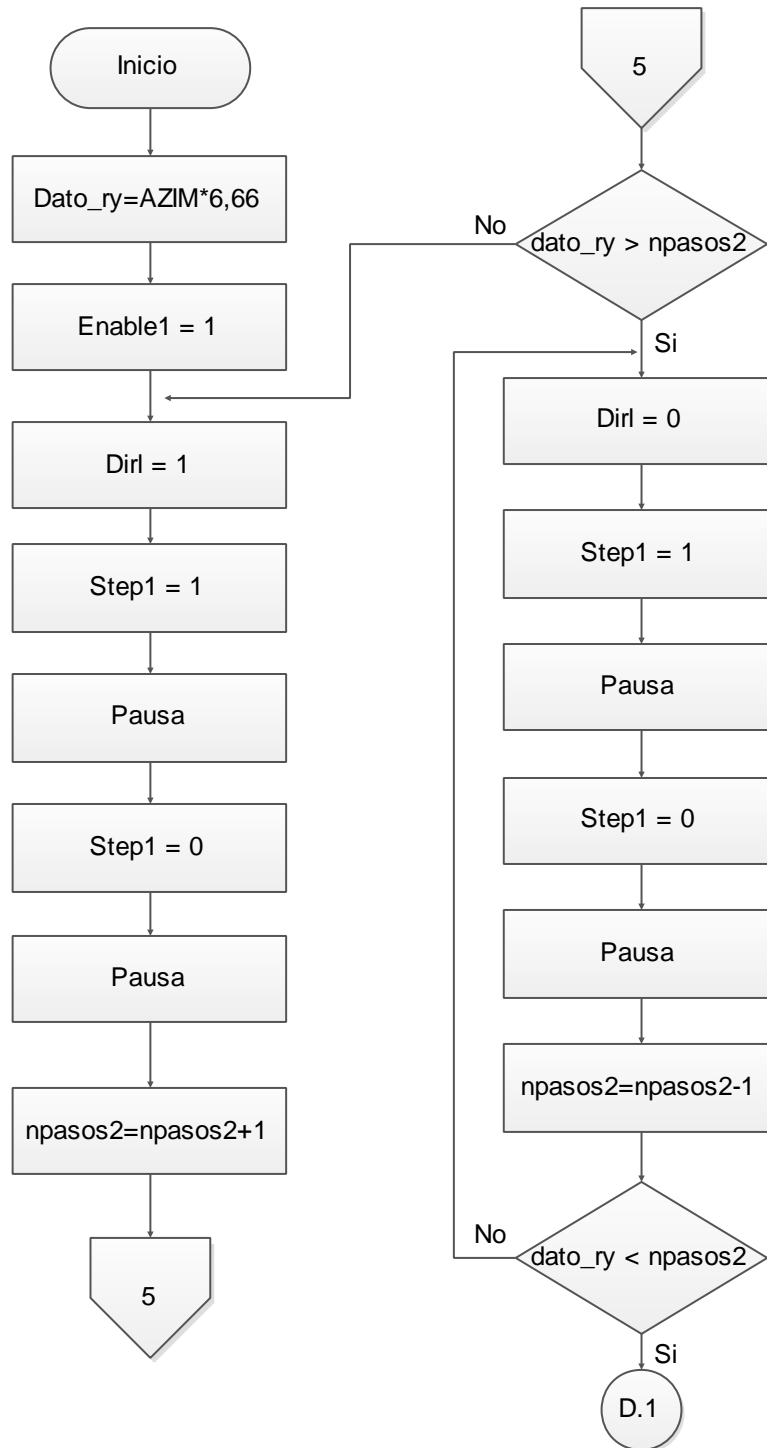


Motor:



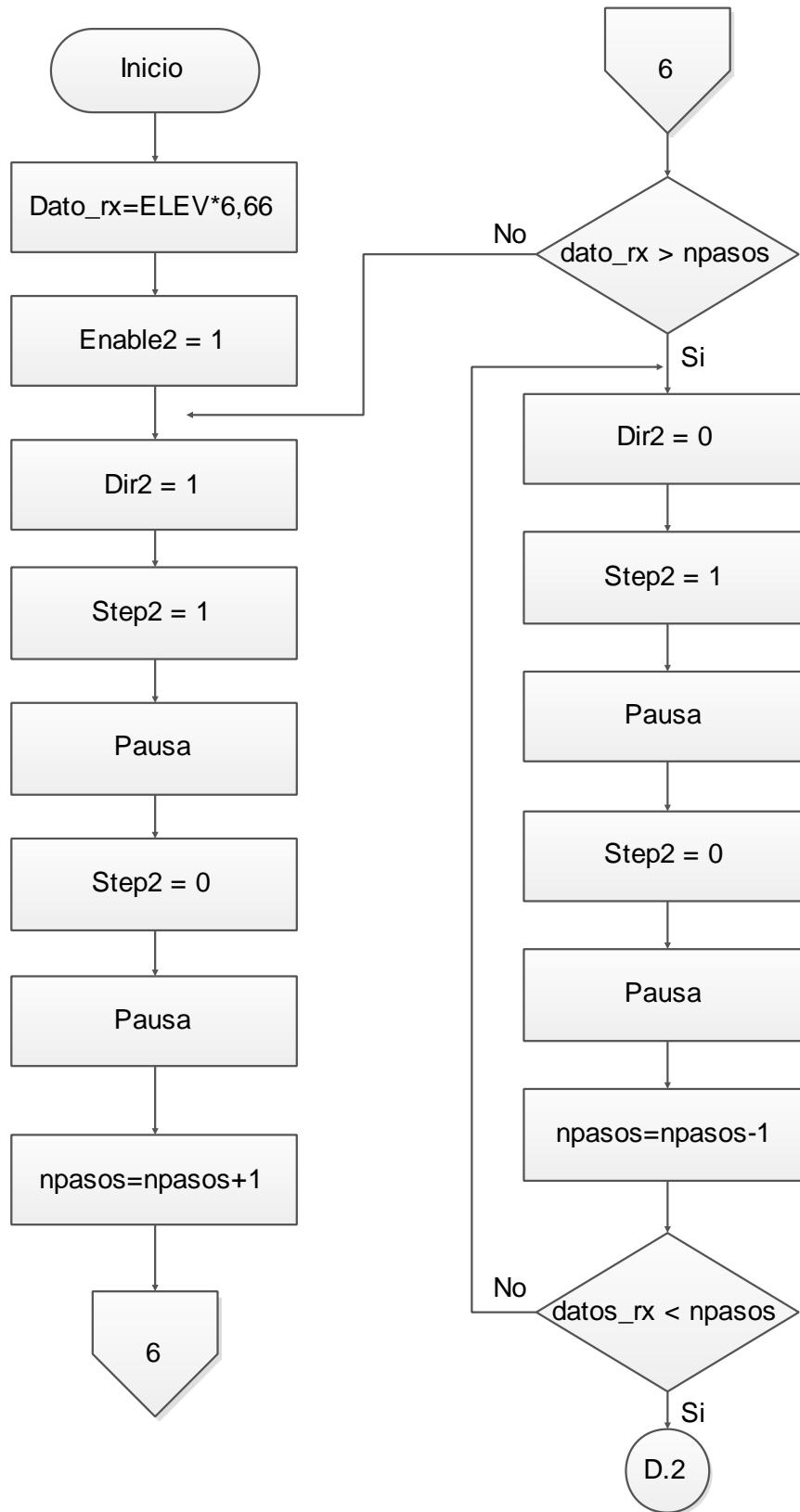


Mov_azim:



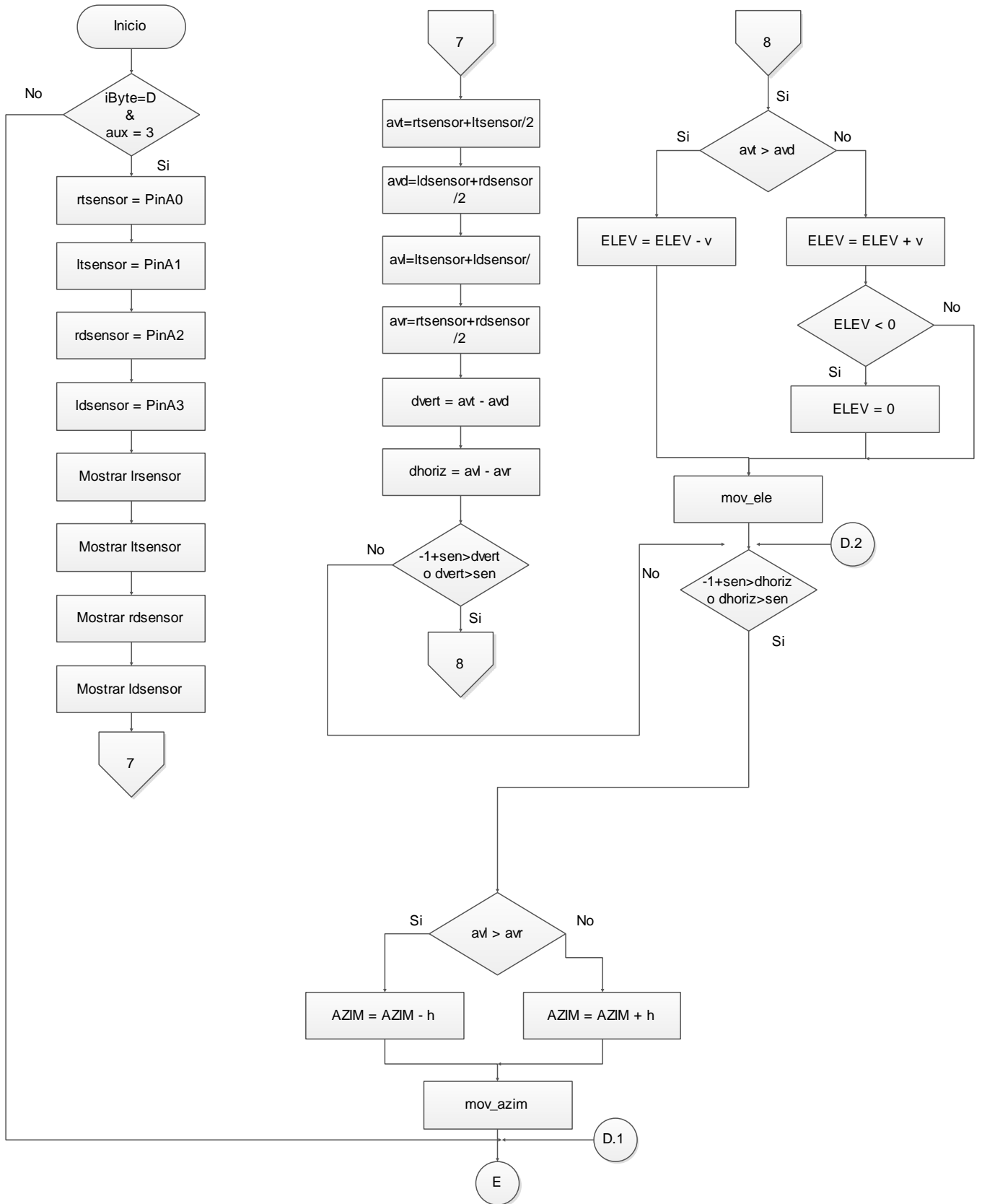


Mov_ele:



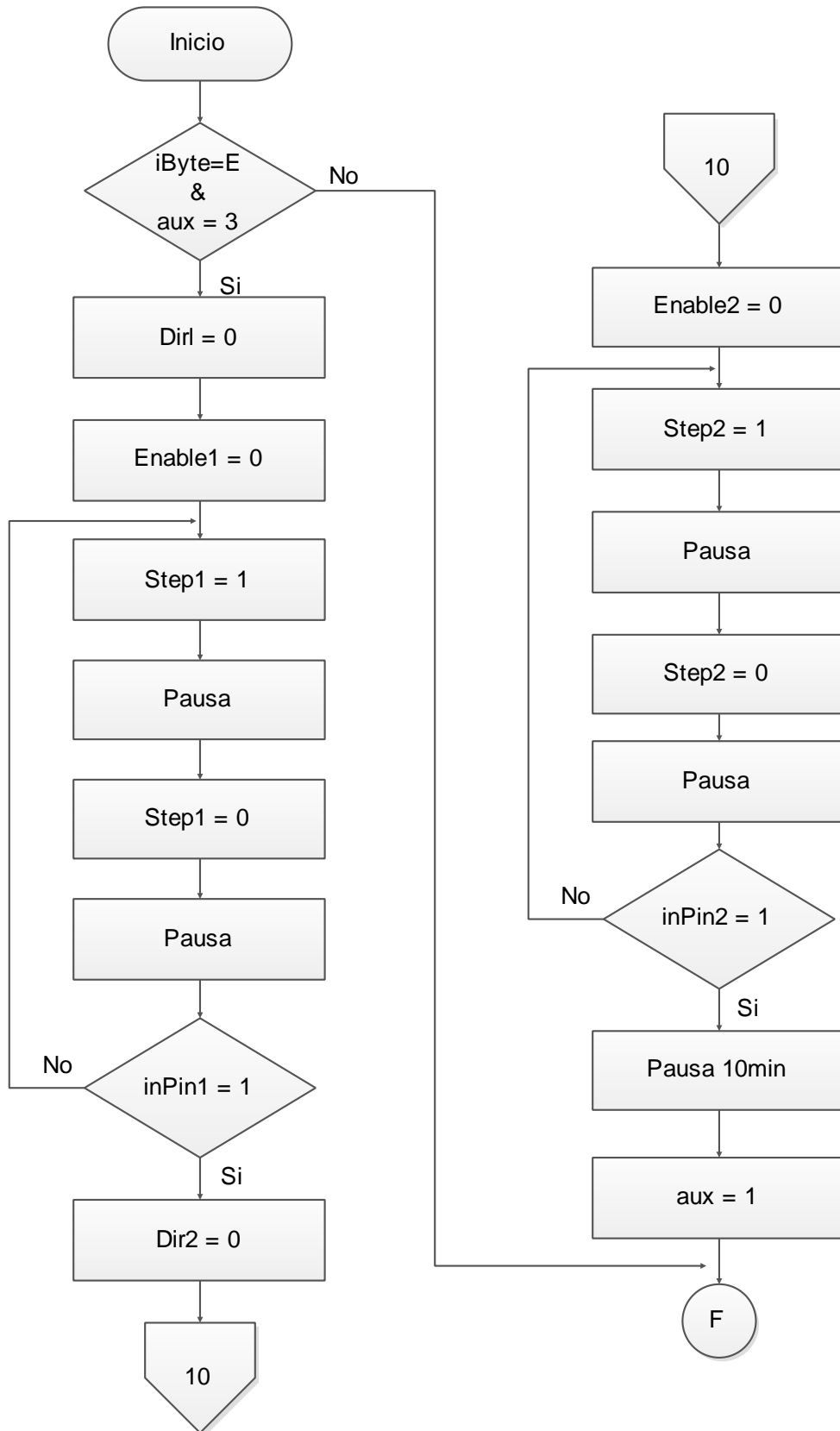


Adjfino:



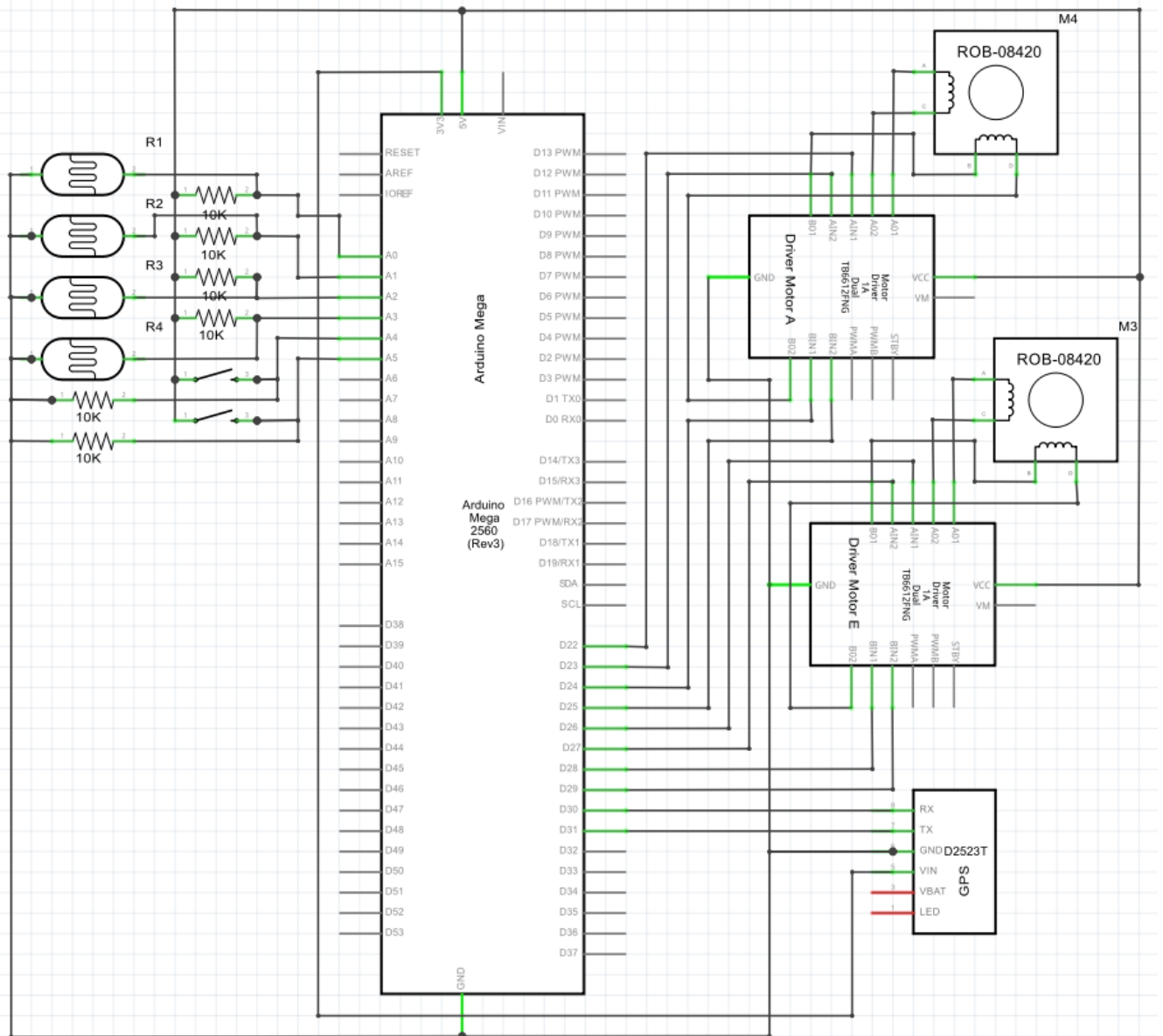


Espera:





Mapa de Hardware:



```
#include <SoftwareSerial.h>
```

```
#include <TinyGPS.h>
```

```
#include <DHT11.h>
```

```
#include <math.h>
```

```
//Declaracion de Constantes//
```

```
static const int TXPin = 50, RXPin = 52; //Puertos del GPS NEO-6M
```

```
static const uint32_t GPSBaud = 9600; //Baudios de comunicacion con LABVIEW
```

```
const int inPin1 = 51; //Puerto Fin de carrera motor 1
```




```
const int inPin2 = 53; //Puerto Fin de carrera motor 2

int Enable1 = 44; //Puerto de habilitacion motor 1

int Step1 = 42; //Puerto de pulsos de motor 1

int Dir1 = 40; //Puerto de sentido de giro motor1

int Enable2 = 45; //Puerto de habilitacion motor 2

int Step2 = 43; //Puerto de pulsos de motor 2

int Dir2 = 41; //Puerto de sentido de giro motor 2

int retardo = 2; //Retardo por step

int aux = 0; //Valor auxiliar

int numero_pasos = 0; //Valor en grados donde se encuentra el motor 1

int numero_pasos2 = 0; //Valor en grados donde se encuentra el motor 2

//int pin = 33; //Salida al cooler

//////////LDR-----//////////

int prom; //Promedio de los cuatroLDR

float h = 0.5; //Pasos ejecutados por el motor horizontal

float v = 0.5; //Pasos ejecutados por el motor vertical

int ltsensor; //Valor del LDR superior izquierdo

int rtsensor; //Valor del LDR superior derecho

int rdsensor; //Valor del LDR inferior izquierdo

int ldsensor; //Valor del LDR inferior derecho

int sen = 100; //Sensibilidad

int dil; //Promedio del conjunto de LDR izquierdo

int dit; //Promedio del conjunto de LDR arriba

int dir; //Promedio del conjunto de LDR derecho

int did; //Promedio del conjunto de LDR abajo

int diff; //Diferencia entre los LDR de arriba con los abajo

int diff2; //Diferencia entre los LDR de la izquierda con los de la derecha

int pup; //Interruptor superior

int pdown; //Interruptor inferior
```



```
//Declaracion de Variables//

static void smartdelay(unsigned long ms);

static void print_float(float val, float invalid, int len, int prec);

static void print_int(unsigned long val, unsigned long invalid, int len);

static void print_date(TinyGPS &gps);

static void print_str(const char *str, int len);

char incomingByte;

float ELEV; //Valor de Elevacion

float AZIM; //Valor de Azimutal

int dato_rx; //Valor recibido en grados

int dato_ry; //Valor recibido en grados

//float temp; //Valor de temperatura

//float hum; //Valor de humedad

TinyGPS gps; //Inicia GPS

SoftwareSerial ss(TXPin, RXPin); //Habilita el puerto serial del GPS

//DHT11 dht11(pin);

void setup(){

  Serial.begin(115200);

  ss.begin(GPSBaud);

  pinMode(inPin2, INPUT); //Fin de carrera Motor 2

  pinMode(inPin1, INPUT); //Fin de carrera Motor 1

  pinMode(Step2, OUTPUT); //Step motor 2

  pinMode(Dir2, OUTPUT); //Dir Motor 2

  pinMode(Enable2, OUTPUT); //Enable Motor 1

  pinMode(Step1, OUTPUT); //Step motor 2

  pinMode(Dir1, OUTPUT); //Dir Motor 2

  pinMode(Enable1, OUTPUT); //Enable Motor 1

  pinMode(Itsensor, INPUT); //LDR1

  pinMode(rtsensor, INPUT); //LDR2
```



```
pinMode(ldsensor, INPUT); //LDR3
pinMode(rdsensor, INPUT); //LDR4
// pinMode(pin, OUTPUT); //Cooler
digitalWrite(Enable1, HIGH);
digitalWrite(Enable2, HIGH);
}

void loop(){
  incomingByte = Serial.read();
  mako();
  calibracion();
  datos();
  motor();
  adjfino();
  espera();
}

void mako(){
  int err;
  float temp, hum;
  if((err = dht11.read(hum, temp)) == 0) { // Si devuelve 0 es que ha leído bien
    if (hum > 90){
      mako();
    }
    if (temp > 30){
      digitalWrite(pin, HIGH);
    }
  }
}

void calibracion(){
```



```
if (incomingByte == 'A' && aux == 0){  
    delay(100);  
    Serial.println("Seguidor Solar V1.0 - Inicializando el Sistema");  
    digitalWrite(Dir1, LOW);  
    digitalWrite(Enable1, LOW);  
    Azimut();  
    digitalWrite(Dir2, LOW);  
    digitalWrite(Enable2, LOW);  
    Elevacion();  
    delay(100);  
    Serial.println("Calibracion Terminada - AA");  
    aux = 1;  
}  
}
```

```
void Azimut(){  
    while (digitalRead(inPin1) != HIGH){  
        digitalWrite(Step1, HIGH); // Step del motor a la izq.  
        delay(retardo);  
        digitalWrite(Step1, LOW); // Step del motor a la izq.  
        delay(retardo);  
    }  
}
```

```
void Elevacion(){  
    while (digitalRead(inPin2) != HIGH){  
        digitalWrite(Step2, LOW); // Step del motor a la izq.  
        delay(retardo);  
        digitalWrite(Step2, HIGH); // Step del motor a la izq.  
        delay(retardo);  
    }  
}
```



```
}
```

```
void datos(){  
  if (incomingByte == 'B' && aux == 1){  
    delay(1000);  
    smartdelay(1000);  
    float flat, flon;  
    unsigned long age, date, time, chars = 0;  
    unsigned short sentences = 0, failed = 0;  
    static const double LONDON_LAT = 51.508131, LONDON_LON = -0.128002;  
    gps.f_get_position(&flat, &flon, &age);  
    print_float(flat, TinyGPS::GPS_INVALID_F_ANGLE, 11, 6);  
    print_float(flon, TinyGPS::GPS_INVALID_F_ANGLE, 11, 6);  
    print_date(gps);  
    print_float(gps.f_altitude(), TinyGPS::GPS_INVALID_F_ALTITUDE, 7, 2);  
    Serial.println("BB");  
    aux = 2;  
  }  
}
```

```
void motor(){  
  if (incomingByte == 'C' && aux == 2){  
    ELEV = 15;  
    mov_ele();  
    numero_pasos=0;  
    Serial.println(incomingByte);  
    ELEV = Serial.parseFloat();  
    AZIM = Serial.parseFloat();  
    Serial.println(ELEV);  
    Serial.println(AZIM);  
    aux = 3;
```



```
delay(1000);

mov_azim();

mov_ele();

}

}

//Función "mov"

void mov_azim(){

dato_ry = (AZIM * 6.666666666); // Ajuste de 512 vueltas a los 360 grados

digitalWrite(Enable1, LOW);

while (dato_ry > numero_pasos2) { // Giro sentido horario en grados.

digitalWrite(Dir1, HIGH); // Direccion de giro en sentido horario.

digitalWrite(Step1, HIGH); // Accionamiento del motor.

delay(retardo);

digitalWrite(Step1, LOW); // Parar el motor.

delay(retardo);

numero_pasos2 = numero_pasos2 + 1;

}

while (dato_ry < numero_pasos2) { // Giro sentido antihorario en grados.

digitalWrite(Dir1, LOW); // Direccion de giro en sentido antihorario.

digitalWrite(Step1, HIGH); // Accionamiento del motor.

delay(retardo);

digitalWrite(Step1, LOW); // Parar el motor.

delay(retardo);

numero_pasos2 = numero_pasos2 - 1;

}

}

void mov_ele(){

dato_rx = (ELEV * 6.666666666); // Ajuste de 512 vueltas a los 360 grados
```



```
digitalWrite(Enable2, LOW);

while (dato_rx > numero_pasos) { // Giro hacia arriba en grados.

    digitalWrite(Dir2, HIGH); // Direccion de giro hacia arriba.

    digitalWrite(Step2, HIGH); // Accionamiento del motor.

    delay(retardo);

    digitalWrite(Step2, LOW); // Parar el motor.

    delay(retardo);

    numero_pasos = numero_pasos + 1;

}

while (dato_rx < numero_pasos) { // Giro hacia abajo en grados

    digitalWrite(Dir2, LOW); // Direccion de giro hacia abajo.

    digitalWrite(Step2, HIGH); // Accionamiento del motor.

    delay(retardo);

    digitalWrite(Step2, LOW); // Parar el motor.

    delay(retardo);

    numero_pasos = numero_pasos - 1;

}

}

void adjfino(){

    if (incomingByte == 'D' && aux == 3) {

        while (aux = 3) {

            Itsensor = analogRead(2); //(La constante es para calibrar las fotorresistencias)

            rtsensor = analogRead(1);

            ldsensor = analogRead(4);

            rdsensor = analogRead(3);

            Serial.println(Itsensor);

            Serial.println(rtsensor);

            Serial.println(ldsensor);

            Serial.println(rdsensor);

            int avt =(Itsensor + rtsensor) / 2;
```



```
int avd =(ldsensor + rdsensor) / 2;
int avl =(ltsensor + ldsensor) / 2;
int avr =(rtsensor + rdsensor) / 2;
int dvert = avt - avd;
int dhoriz = avl - avr;
if (-1*sen > dvert || dvert > sen){
  if (avt > avd){
    ELEV=ELEV-v;
  }
  else if (avt < avd){
    ELEV=ELEV+v;
    if (ELEV < 0){
      ELEV=0;
    }
  }
  mov_ele();
}
if (-1*sen > dhoriz || dhoriz > sen){
  if (avl > avr){
    AZIM=AZIM-h;
  }
  else if (avl < avr){
    AZIM=AZIM+h;
  }
  else if (avl == avr) {
    // nothing
  }
  mov_azim();
}
delay(10);
}
```




```
}  
}
```

```
void espera(){  
  if (incomingByte == 'E' && aux == 3){  
    digitalWrite(Dir1, LOW);  
    digitalWrite(Enable1, LOW);  
    Azimut();  
    digitalWrite(Dir2, LOW);  
    digitalWrite(Enable2, LOW);  
    Elevacion();  
    delay(3000);  
    aux = 1;  
  }  
}
```

```
static void smartdelay(unsigned long ms){  
  unsigned long start = millis();  
  do {  
    while (ss.available())  
      gps.encode(ss.read());  
  }  
  while (millis() - start < ms);  
}
```

```
static void print_float(float val, float invalid, int len, int prec){  
  if (val != invalid) {  
    Serial.print(val, prec);  
    int vi = abs((int)val);  
    int flen = prec + (val < 0.0 ? 2 : 1); // . and -  
    flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
```



```
for (int i = flen; i < len; ++i)
    Serial.print(' ');
}
smartdelay(0);
}
```

```
static void print_int(unsigned long val, unsigned long invalid, int len) {
    char sz[32];
    if (val != invalid)
        sprintf(sz, "%ld", val);
    sz[len] = 0;
    for (int i = strlen(sz); i < len; ++i)
        sz[i] = ' ';
    if (len > 0)
        sz[len - 1] = ' ';
    Serial.print(sz);
    smartdelay(0);
}
```

```
static void print_date(TinyGPS &gps) {
    int year;
    byte month, day, hour, minute, second, hundredths;
    unsigned long age;
    gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
    if (age != TinyGPS::GPS_INVALID_AGE) {
        char sz[32];
        sprintf(sz, "%02d/%02d/%02d %02d:%02d:%02d ",
            month, day, year, hour, minute, second);
        Serial.print(sz);
    }
    smartdelay(0);
}
```



```
}
```

```
static void print_str(const char *str, int len){
```

```
    int slen = strlen(str);
```

```
    for (int i = 0; i < len; ++i)
```

```
        Serial.print(i < slen ? str[i] : ' ');
```

```
    smartdelay(0);
```

```
}
```