UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE INGENIERÍA

Microcontroladores y Electrónica de Potencia

Trabajo integrador:

Helióstato doméstico

Alumno: Víctor David Silva

Legajo: 10988

Índice

Introducción

En el presente trabajo integrador se muestra una idea de un dispositivo doméstico capaz de aprovechar la luz natural mediante reflexión para iluminar ambientes que de otra manera no podrían tener este tipo de luz, como se observa en la figura 1. Para poder calcular los ángulos que debe

Figura 1. Aplicación del helióstato doméstico

moverse el helióstato se le debe dar los ángulos de azimut y altura solar (figura 2), que en este trabajo se le va a llamar zenit.

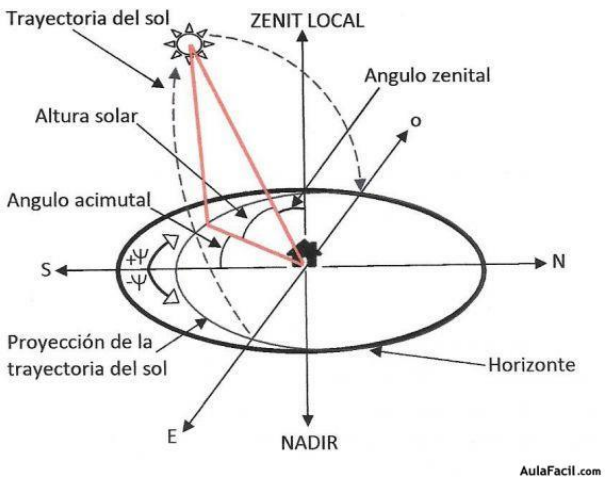


Figura 2. Ángulos relacionados al sol

Esquema tecnológico

El diagrama en bloques del proyecto se observa en la figura 3. Esto consta de una PC que se comunica con el Arduino UNO, donde mediante la UART puede recibir diferentes instrucciones a realizar. Este se comunica mediante SPI con un módulo micro SD de donde recibe la información con la que va a trabajar. A partir de esto controla mediante la cantidad de pulsos y la dirección correspondiente a dos drivers de motor paso a paso A4988 los cuales controlan cada uno su respectivo motor. Cuando se realice la maniobra de homing se presionará un pulsador final de carrera para cada uno de los motores lo que dará la posición de referencia.

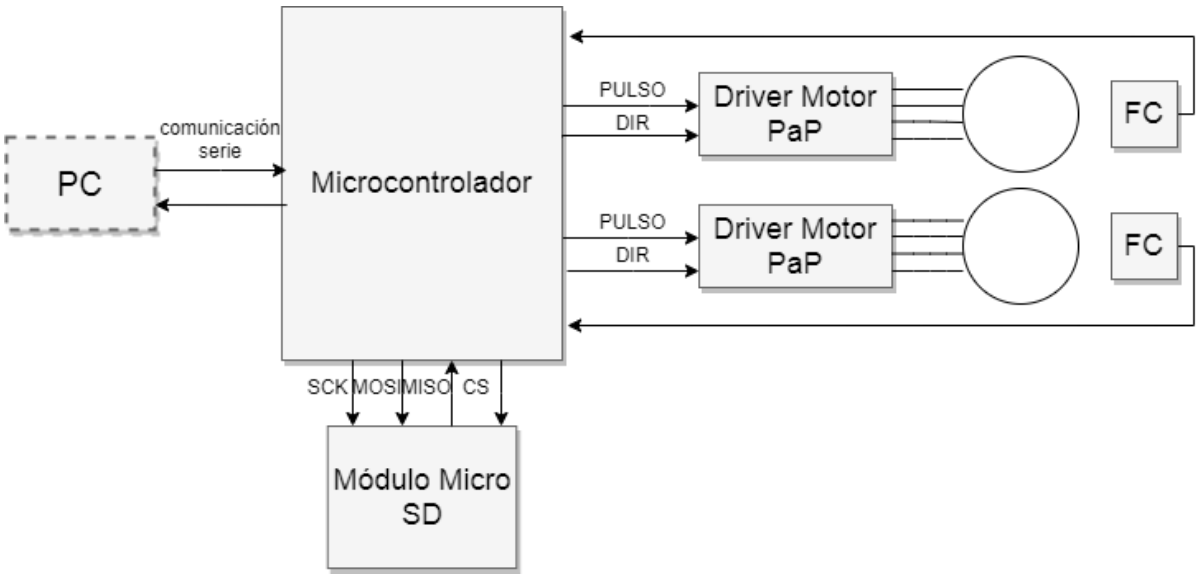


Figura 3. Diagrama de estados

Detalle de módulos

Fines de carrera

Estos fines de carrera (figura 4) tienen el objetivo de que cuando se realice la maniobra de homing y los motores lleguen a la posición de referencia inicial sean presionados por la estructura del helióstato de modo tal que paren los motores y luego se muevan unos pocos pasos hasta dejar de presionarlos en el otro sentido. En el proyecto se utilizaron pulsadores comunes para simularlo.

Figura 4. Interruptor final de carrera

Driver de motor A4988

Para el control de los motores paso a paso se utiliza un driver A4988 (figura 5) para cada uno de ellos. Estos drivers reciben las señales del microcontrolador de la cantidad de pulsos y el sentido de rotación requerido. Se deben conectar a una fuente con la tensión y amperaje necesario para los motores. En los pintes 2B, 2A, 1A y 1B se conecta el motor una vez identificado los bobinados. Para la aplicación se va a usar el paso normal que tiene el motor ya que el micro paso no es necesario.

Figura 5. Driver A4988

Motor paso a paso

Como actuador se utilizarán dos motores unipolares (figura 6) uno de 5 cables y otro de 6 cables. Antes de realizar las conexiones al driver es necesario identificar los bobinados para conectarlos correctamente al driver.

Figura 6. Motor paso a paso

Módulo tarjeta micro SD

La información de los ángulos de zenit y azimut se guardan en una tarjeta micro SD como un archivo de extensión txt, esta será leída por el microcontrolador gracias al uso del módulo para tarjetas micro SD (figura 7), el cual se comunica con el microcontrolador mediante protocolo SPI.

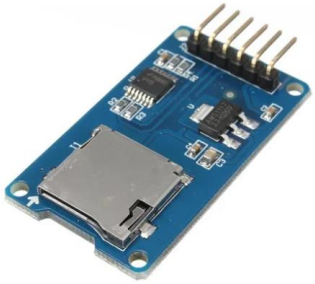


Figura 7. Módulo micro SD

Funcionamiento general

El programa comienza declarando las variables, inicializándolas y configurando los periféricos. Se observa en el diagrama de estados (figura 8) que luego el programa pasa al estado *Desactivado* donde

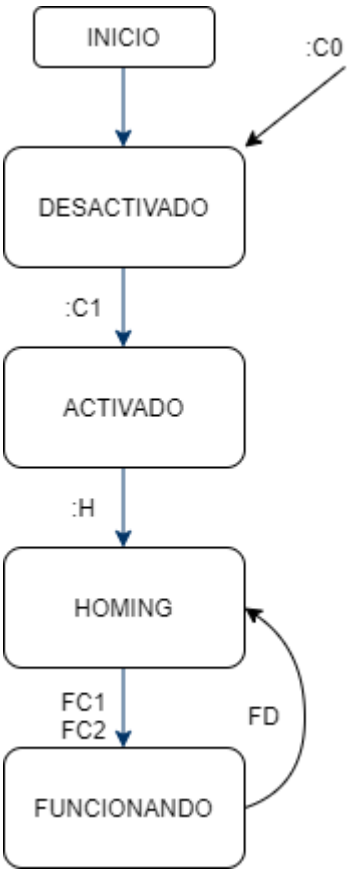


Figura 8. Diagrama de estados

muestra un mensaje de si desea pasar al estado *Activado* debe ingresar por la terminal :C1. En cualquier momento se puede pasar al estado *Desactivado* desde cualquier otro estado ingresando por la terminal :C0. En el estado *Activado* se realiza la configuración de algunos de los parámetros como del ángulo deseado donde se quiere reflejar según el valor de zenit y azimut que se le dé, si está apuntando al norte o al sur la flecha azul que tiene la plataforma del dispositivo (figura 9), el tiempo en el que se desea comenzar dependiendo

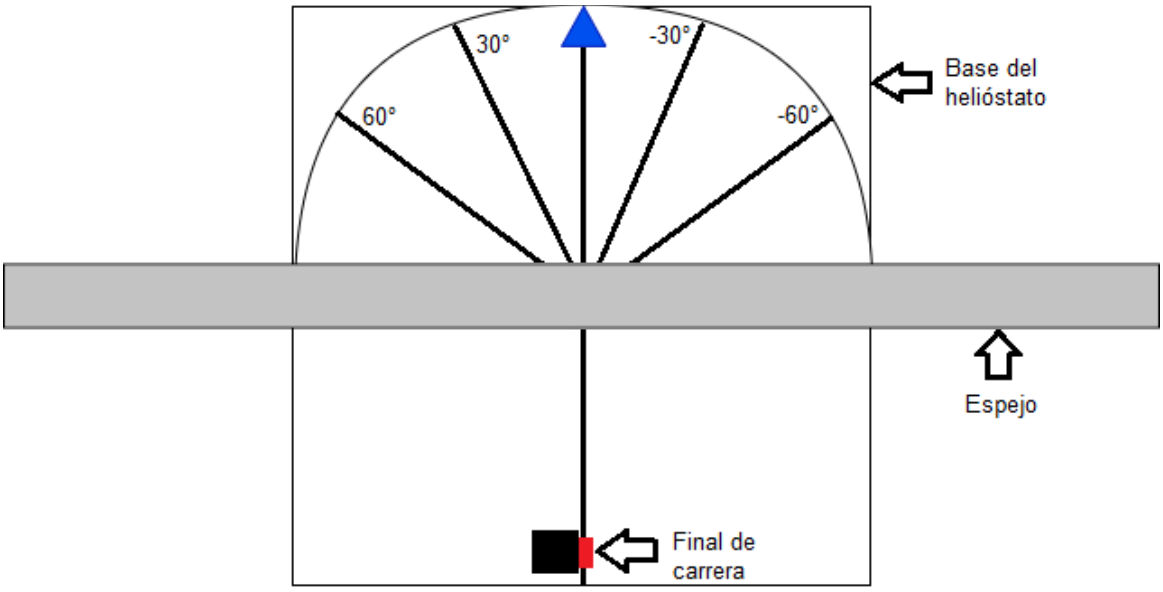


Figura 9. Vista superior del helióstato

a que hora se haga la configuración y por último se debe ingresar por la terminal :H para hacer un homing inicial, esto permitirá pasar al siguiente estado. En el estado de *Homing* llama a la función homing donde los motores comienzan a retroceder hasta tocar cada uno su final de carrera donde se tendrá la respectiva posición de referencia (en el caso de azimut es el rojo que se observa en la figura 8, en el caso de zenit está en el soporte y se presionará cuando esté completamente vertical), luego de esto pasa al siguiente estado. Para el estado *Funcionando* se activa el timer que será el que lleve el conteo del tiempo para que este sea siempre de 15 minutos, aquí se desarrollará el funcionamiento principal del dispositivo donde se calcularán los ángulos que debe tener el helióstato para reflejar de la forma deseada. Seguirá en este estado hasta que el día llegue a su fin, donde se realizará la maniobra de homing nuevamente cuando se termine el día.

A continuación se muestran algunos de los diagramas (figuras 10 , 11 ,12 y 13) para el cálculo de los ángulos tanto para el caso de beta_azimut como de beta_zenit.

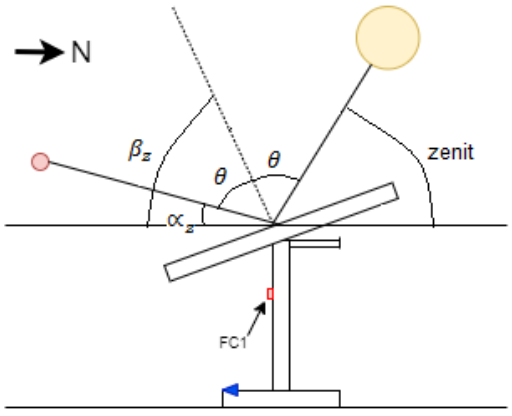


Figura 10. Diagrama para beta zenit sur

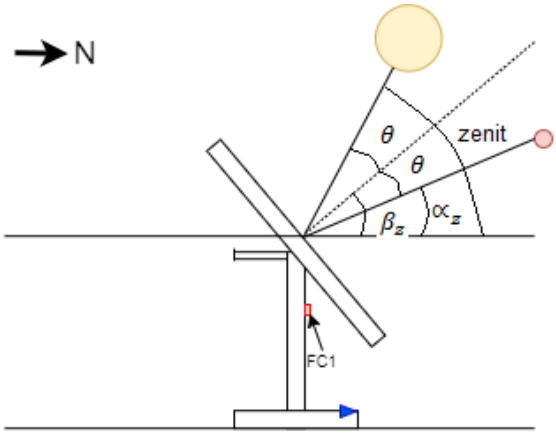


Figura 11. Diagrama para beta zenit norte

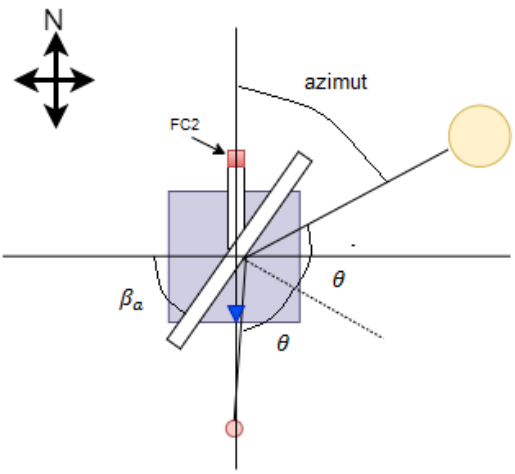


Figura 12. Diagrama para beta azimut sur

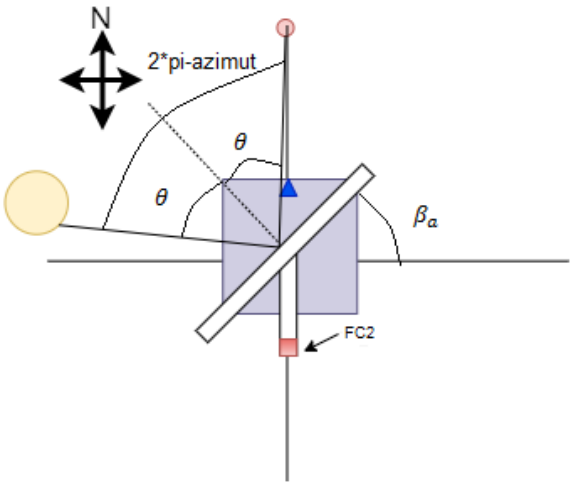


Figura 13. Diagrama para beta azimut norte

Programación

A continuación, se detallará la configuración de los periféricos del controlador:

GPIO

```
DDRB = (1<<DDB1)|(1<<DDB2);
DDRD = (1<<DDD6)|(1<<DDD2)|(1<<DDD3)|(1<<DDD4)|(1<<DDD5);
```

Para el PORTB se configuran el PIN1 y PIN2 como salidas. Para el PORTD se configuran el PIN2, PIN3, PIN4, PIN5 y PIN6 como salidas.

```
DDRB|=(1<SS)|(1<<MOSI)|(1<<SCK);
```

Para la comunicación SPI definimos el PINB0 como SS, el PINB3 como MOSI, el PINB4 como MISO y el PINB5 como SCK. Definimos los SS, MOSI y SCK como salida, quedando MISO como entrada.

TIMER

```
#define sNO 0
#define sTOGGLE 1
#define sCLEAR 2
#define sSET 3
```

Se define el valor correspondiente a cada uno.

```
TCCR1A=(sNO<<COM1A0)|(sNO<<COM1B0)|(0<<WGM11)|(0<<WGM10);
TCCR1B=(0<<WGM13)|(1<<WGM12)|(0<<CS12)|(0<<CS11)|(0<<CS10);
TIFR1 =(1<<OCF1A)|(0<<TOV1)|(0<<ICF1);
TIMSK1=(1<<OCIE1A)|(0<<TOIE1)|(0<<ICIE1);
```

El timer 1 se utilizará como la unidad de tiempo del sistema. Para el registro TCCR1 se programó que COM1A y COM1B tengan el valor de sNO, es decir que estarán desconectados. El modo de generación de onda es el 4, WGM13:0=0100 , modo CTC con OCR1A como TOP, el cual tiene un valor de 62499. En cuanto al prescaler comienza con cero, pero cuando se active el programa este será de 256 ya que CS12 tendrá un 1. Para el registro TIFR1 se borran los flags de compare Match y Overflow antes de habilitar la interrupción. Para el registro TIMSK1 se habilita la interrupción por compare match A. Por lo tanto el tiempo es de 1s.

```
TCCR2A=(sNO<<COM2A0)|(sNO<<COM2B0)|(1<<WGM21)|(0<<WGM20);
TCCR2B=(0<<WGM22)|(0<<CS22)|(0<<CS21)|(0<<CS20);
TIFR2 =(1<<OCF2A)|(0<<TOV2);
TIMSK2=(1<<OCIE2A)|(0<<TOIE2);
```

El timer 2 se usará para establecer el tiempo en alto y bajo de los pulsos que se envían a los drivers. Se configura como CTC que es el modo 2 para este timer, y se usará un prescaler de 1024. En el registro TIFR2 se borran el flag de compare match A antes de habilitarlo. Para el registro TIMSK2 se le habilita la interrupción por compare match A. Este timer es de 8 bits, por lo que se usará OCR2A=249, dando así un tiempo es de 16ms.

UART

```
UBRR0 = F_CPU/16/brate-1;
UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);
UCSR0B = (1<<RXEN0)|(1<<TXEN0);
if (IntRX)
```

```
UCSR0B|= (1<<RXCIE0);
if (IntTX)
UCSR0B|= (1<<TXCIE0);
```

Para la UART tenemos que F_CPU tenemos un cristal de 16MHz y el Baud rate que se le ha dado es de 57600, por lo que UBRR0=16,36=16. Esto da BaudRate=F_CPU/16/(16+1)=58826,53. Esto tiene un error de 2,08%. Para el registro UCSR0C, UCSZ0 valdrá 011 ya que en el registro UCSR0B UCSZ02=0, por lo que la longitud del dato será de 8bits. En el registro UCSR0B se habilita la recepción. Como IntRX se le dio un valor de 1 en el main se le da un 1 a RXCIE0, que habilita interrupción por RXCn.

SPI

Se inicializan los puertos con la función SPI_Init como se mencionó anteriormente. Además de eso se configuran sus registros:

```
SPCR= (1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0);
```

```
Se habilita el SPI, se lo configura como master y se usa un prescaler de 1024, por lo
que la cantidad de Mbps es de 16000000/1024=15625.
```

La función SPI_ SPI_master_Enviar permite transferir datos, se cargan los datos al registro SPDR y luego espera a que se complete la transmisión. La otra función que tenemos es SPI_ SPI_master_Recibir, es una función que nos permite recibir un byte del bus en contraparte a la función anterior, se envía un dummy byte para que el esclavo nos transmite y luego se espera hasta que se complete, finalmente se devuelve el valor de SPDR.

Módulo micro SD

Se comienza inicializando la SD mediante la función SD_Init, según del datasheet (SD Especificaciones, ver referencias) en la página 91 indica que al inicializarla hay que darle 74 pulsos como se indica en la figura 14, en el código se le envían 80. Se hace uso de las funciones SPI_master_Enviar y SD_comando, donde

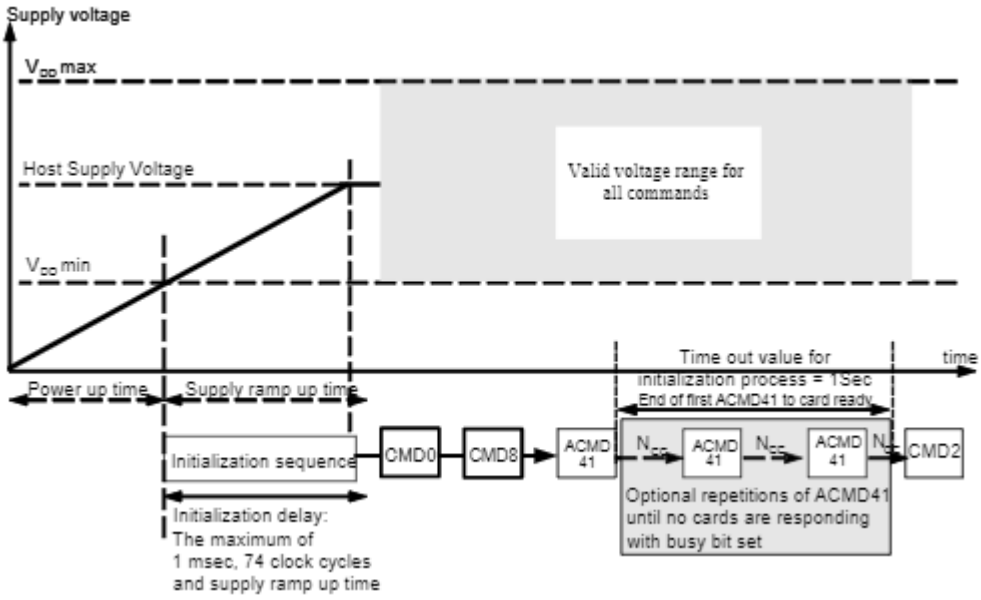


Figura 14. Diagrama de Power up

la última es una función para pasarle comandos a la memoria SD según se indiquen en el datasheet. El formato de los comandos se observa en la tabla de la figura 15.

Table 4-15 shows the format of CMD8.

Bit position	47	46	[45:40]	[39:20]	[19:16]	[15:8]	[7:1]	0
Width (bits)	1	1	6	20	4	8	7	1
Value	'0'	'1'	'001000'	'00000h'	x	x	x	'1'
Description	start bit	transmission bit	command index	reserved bits	voltage supplied (VHS)	check pattern	CRC7	end bit

Figura 15. Formato de comando

Entre los comandos usados en este proyecto podemos encontrar CMD0, CMD1, CMD17 y CMD24 (del cual no se usa en el proyecto). Para estos comandos hay que tener en cuenta que según la tabla de la figura 15 el primer bit es siempre 0 y el siguiente 1, por lo que el número correspondiente a esos comandos hay que agregarle esos dos bits. Por lo tanto, para el caso de CMD0 el command index es '000000', en binario sería 0b00100000, y en hexadecimal 0x40. Para CMD17 el command index es '010111', en binario sería 0b01010111, y en hexadecimal 0x51. Para CMD24 el command index es '100100', en binario sería 0b01100100, y en hexadecimal 0x58. CMD1 es uno bit mayor que CMD0. Todos estos comandos se pasan como primer argumento de la función SD_comando.

Como se observa en la figura 14 el comando que se le debe enviar para inicializarla luego de los pulsos es CMD0, que en la tabla de la figura 16 se puede ver que este resetea la tarjeta de memoria SD. Luego se le envía el comando CMD1 que tiene la misma función que ACMD41, este envía información de la capacidad de soporte del host y activa el proceso de inicialización de la tarjeta proceso.

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD0	Yes	[31:0] stuff bits	R1	GO_IDLE_STATE	Resets the SD Memory Card
CMD1	Yes[1]	[31]Reserved bit [30]HCS [29:0]Reserved bits	R1	SEND_OP_ COND	Sends host capacity support information and activates the card's initialization process. HCS is effective when card receives SEND_IF_COND command. Reserved bits shall be set to '0'.

Figura 16. Tabla comandos CMD0 y CMD1

Para leer la tarjeta de memoria SD se usa la función SD_Leer_Bloque, donde inicialmente se llama al comando CMD17 que en la tabla de la figura 17 se observa que lee un bloque de memoria, donde según la hoja de datos la máxima longitud de este es de 512 bytes que es del tamaño que se usará en la aplicación. Se realizan algunas comprobaciones y se reciben los bytes de la memoria SD, los cuales se vuelcan en un buffer. Para escribir en la memoria se tiene a la función SD_Escribir_Bloque, pero esta no fue usada en la aplicación, en ella se llama al comando, luego se realiza la comunicación por SPI y se hacen algunas comprobaciones para asegurarse de que se ha realizado de forma correcta.

CMD17	Yes	[31:0] data address[10]	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.[3]
CMD18	Yes	[31:0] data address[10]	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a STOP_TRANSMISSION command.
CMD19	reserved				
CMD20	No				
CMD21... CMD23	reserved				
CMD24	Yes	[31:0] data address[10]	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[4]

Figura 17. Tabla comandos lectura y escritura

Funciones

Leerbuffer():
Esta función interpretará cada uno de los caracteres de la trama y le asignará a cada variable el valor numérico que les corresponda según la trama. Cuando halle un caracter '0' (en el salto de línea), saldrá del bucle donde está la función.

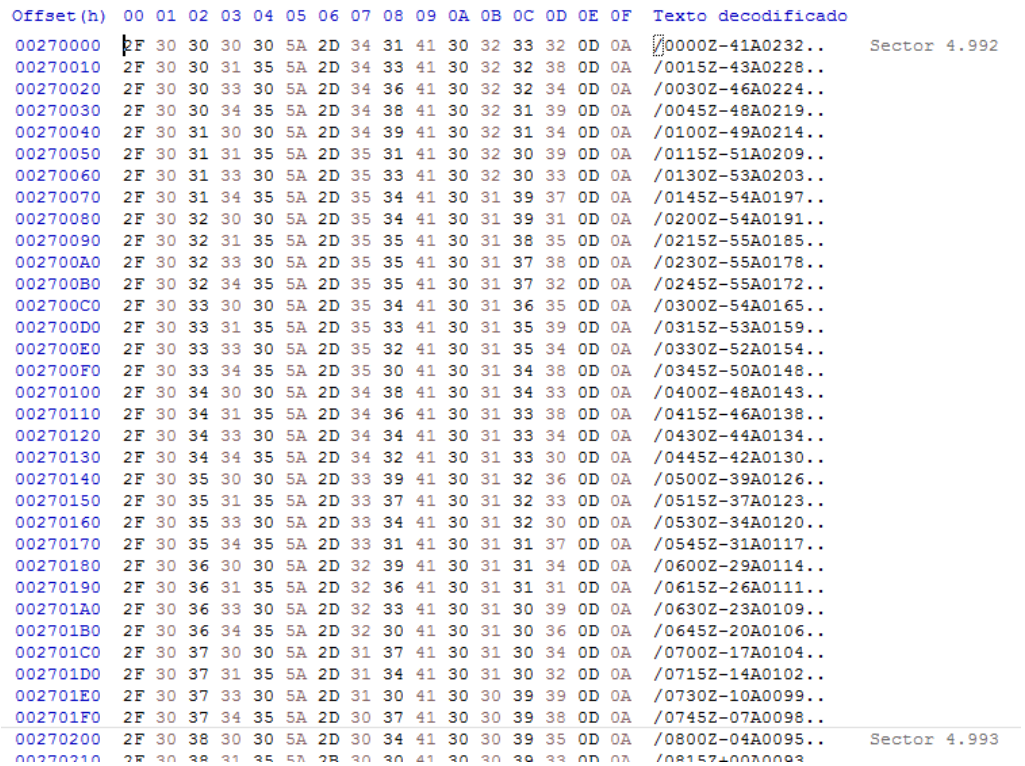
La trama tiene la forma:

/HHHHZ-XXAXXXX\r\n

Se tiene la siguiente información:

/HHHH: es la hora la cual se escribe en los dos primeros bytes la hora y en los dos siguientes los minutos. Para ahorrar espacio en la memoria y como los movimientos son pequeños, la información es por intervalos de 15 minutos.
Z-XX: es el ángulo de zenit que tendrá el motor 1, este puede ser positivo o negativo, así que se considera un caracter para el signo.
AXXXX: es el ángulo de azimut que tendrá el motor dos, este varía entre 0° y 360°. Solo tiene signo positivo.
\r: retorno de carro.
\n: salto de línea.

En la figura 18 se observa como está la información contenida dentro de la tarjeta de memoria, se puede ver que se divide en sectores y que cada uno de estos de 512 bytes.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texto decodificado	
00270000	2F	30	30	30	30	5A	2D	34	31	41	30	32	33	32	0D	0A	/0000Z-41A0232..	Sector 4.992
00270010	2F	30	30	31	35	5A	2D	34	33	41	30	32	32	38	0D	0A	/0015Z-43A0228..	
00270020	2F	30	30	33	30	5A	2D	34	36	41	30	32	32	34	0D	0A	/0030Z-46A0224..	
00270030	2F	30	30	34	35	5A	2D	34	38	41	30	32	31	39	0D	0A	/0045Z-48A0219..	
00270040	2F	30	31	30	30	5A	2D	34	39	41	30	32	31	34	0D	0A	/0100Z-49A0214..	
00270050	2F	30	31	31	35	5A	2D	35	31	41	30	32	30	39	0D	0A	/0115Z-51A0209..	
00270060	2F	30	31	33	30	5A	2D	35	33	41	30	32	30	33	0D	0A	/0130Z-53A0203..	
00270070	2F	30	31	34	35	5A	2D	35	34	41	30	31	39	37	0D	0A	/0145Z-54A0197..	
00270080	2F	30	32	30	30	5A	2D	35	34	41	30	31	39	31	0D	0A	/0200Z-54A0191..	
00270090	2F	30	32	31	35	5A	2D	35	35	41	30	31	38	35	0D	0A	/0215Z-55A0185..	
002700A0	2F	30	32	33	30	5A	2D	35	35	41	30	31	37	38	0D	0A	/0230Z-55A0178..	
002700B0	2F	30	32	34	35	5A	2D	35	35	41	30	31	37	32	0D	0A	/0245Z-55A0172..	
002700C0	2F	30	33	30	30	5A	2D	35	34	41	30	31	36	35	0D	0A	/0300Z-54A0165..	
002700D0	2F	30	33	31	35	5A	2D	35	33	41	30	31	35	39	0D	0A	/0315Z-53A0159..	
002700E0	2F	30	33	33	30	5A	2D	35	32	41	30	31	35	34	0D	0A	/0330Z-52A0154..	
002700F0	2F	30	33	34	35	5A	2D	35	30	41	30	31	34	38	0D	0A	/0345Z-50A0148..	
00270100	2F	30	34	30	30	5A	2D	34	38	41	30	31	34	33	0D	0A	/0400Z-48A0143..	
00270110	2F	30	34	31	35	5A	2D	34	36	41	30	31	33	38	0D	0A	/0415Z-46A0138..	
00270120	2F	30	34	33	30	5A	2D	34	34	41	30	31	33	34	0D	0A	/0430Z-44A0134..	
00270130	2F	30	34	34	35	5A	2D	34	32	41	30	31	33	30	0D	0A	/0445Z-42A0130..	
00270140	2F	30	35	30	30	5A	2D	33	39	41	30	31	32	36	0D	0A	/0500Z-39A0126..	
00270150	2F	30	35	31	35	5A	2D	33	37	41	30	31	32	33	0D	0A	/0515Z-37A0123..	
00270160	2F	30	35	33	30	5A	2D	33	34	41	30	31	32	30	0D	0A	/0530Z-34A0120..	
00270170	2F	30	35	34	35	5A	2D	33	31	41	30	31	31	37	0D	0A	/0545Z-31A0117..	
00270180	2F	30	36	30	30	5A	2D	32	39	41	30	31	31	34	0D	0A	/0600Z-29A0114..	
00270190	2F	30	36	31	35	5A	2D	32	36	41	30	31	31	31	0D	0A	/0615Z-26A0111..	
002701A0	2F	30	36	33	30	5A	2D	32	33	41	30	31	30	39	0D	0A	/0630Z-23A0109..	
002701B0	2F	30	36	34	35	5A	2D	32	30	41	30	31	30	36	0D	0A	/0645Z-20A0106..	
002701C0	2F	30	37	30	30	5A	2D	31	37	41	30	31	30	34	0D	0A	/0700Z-17A0104..	
002701D0	2F	30	37	31	35	5A	2D	31	34	41	30	31	30	32	0D	0A	/0715Z-14A0102..	
002701E0	2F	30	37	33	30	5A	2D	31	30	41	30	30	39	39	0D	0A	/0730Z-10A0099..	
002701F0	2F	30	37	34	35	5A	2D	30	37	41	30	30	39	38	0D	0A	/0745Z-07A0098..	
00270200	2F	30	38	30	30	5A	2D	30	34	41	30	30	39	35	0D	0A	/0800Z-04A0095..	Sector 4.993

Figura 18. Contenido de la tarjeta de memoria

InterpreraComando():
Mediante el uso de UART se pueden introducir mediante teclado a través de una terminal externa diferentes comandos, estos se comunicarán con el microcontrolador mediante el protocolo de comunicación serie RS232. Entre estos comandos podemos hallar:

:H realiza la maniobra de homing.
:TXXXX indica la hora a la cual se comenzará a operar, esta se define al principio del programa y su valor por defecto es 0.
:NX indica si la flecha está apuntando hacia el norte (X=1) o si está apuntando hacia el sur (X=0)
:AXX permite elegir el ángulo de alpha azimut donde se desea reflejar.
:ZXX permite elegir el ángulo de alpha zenit donde se desea reflejar.

calcular_angulos():
Esta función toma como parámetros el valor de zenit y azimut, y a partir de ellos y de la definición de los ángulos deseados donde se quiere la reflexión, se calculan los ángulos que debe tener el helióstato.

inicio():
Muestra por pantalla los comandos posibles para realizar la configuración.

homing():
En esta función se realiza la maniobra de homing, donde hace que los motores retrocedan pasos hasta tocar cada uno su respectivo fin de carrera y luego avanzan unos pocos pasos hasta que dejan de presionarlos, luego se establece el valor de referencia para los ángulos del helióstato.

main():
Luego de inicializar las variables y periféricos, se comporta como una máquina de estados donde el funcionamiento principal se da en el estado funcionando. Aquí se muestra por pantalla el tiempo, los respectivos ángulos de las variables donde zenit es el ángulo de elevación del sol y se muestra el ángulo

de azimut del sol. También se muestran los ángulos que tiene el helióstato y el ángulo anterior con el subíndice aux, que sería la posición actual. Si el zenit es menor a cero no se mueve. Se realizan restas para determinar la cantidad de pasos a dar y dependiendo de cuál ángulo es mayor girará en un sentido o en otro. Finalmente, cuando se lea todo el buffer aumentará un contado hexadecimal para que se lean los siguiente 512 bytes de la memoria.

Interrupciones

Se tiene varias interrupciones en el programa debido a los diferentes periféricos usados.

```
PCICR|=(1<<PCIE0);
PCMSK0|=(1<<PCINT1)|(1<<PCINT2);
```

En el registro PCICR se hace la habilitación Pin Change Interrupt en pines PCINT[7:0]. El registro PCMSK0 habilita individualmente los pines PCINT1 y PCINT2.

```
ISR(TIMER1_COMPA_vect)
```

Al producirse la interrupción por Compare Match se ingresa a la función donde se obtiene el tiempo final entre lecturas, donde cada vez que se cumpla ese tiempo entra en un while donde se vacía el buffer y se inicializa la SD con la función SD_Init(), luego le llama a la función SD_Read_Block donde se lee la tarjeta por SPI y se vuelcan los datos al buffer. Luego entra a un for donde si es la primera vez que se inicia el programa buscará la hora de inicio seleccionada en la configuración y comenzará a partir de ahí, en caso de que no esté en los primeros 512bytes seguirá leyendo la tarjeta y buscando esa hora. Luego comienza a leer el buffer asignando los valores correspondientes a las variables hasta hallar un salto de línea.

```
ISR(TIMER2_COMPA_vect)
```

Al producirse la interrupción por compare match para el timer 2 el Contador de activar_motor aumenta en 1.

```
ISR (PCINT0_vect)
```

Al presionarse alguno de los fines de carrera se produce la interrupción por cambio de pin lo que llama a esta función, dependiendo de si está presionado el interruptor será el valor de FC1 y FC2.

```
ISR(USART_RX_vect)
```

Se produce una interrupción a recibir un dato, aquí se analiza la trama y se llama la función InterpretaComando.

Etapas de montaje y ensayos realizados

A medida que se desarrollaba el código se iban realizando distintas pruebas y se procedía a realizar correcciones y mejoras. En la figura 19 se observa el circuito del helióstato, donde se pueden ver el microcontrolador, el módulo micro SD, los pulsadores que hacen de fines de carrera, algunas resistencias y LEDs, de los cuales los de la izquierda (amarillo y verde) son los que dan los pulsos a los drivers y los de la derecha (rojo y verde), son lo que establecen la dirección de giro de los motores y el LED del medio (amarillo) es un testigo que cambia de estado cada vez que se realiza una lectura de trama del buffer.

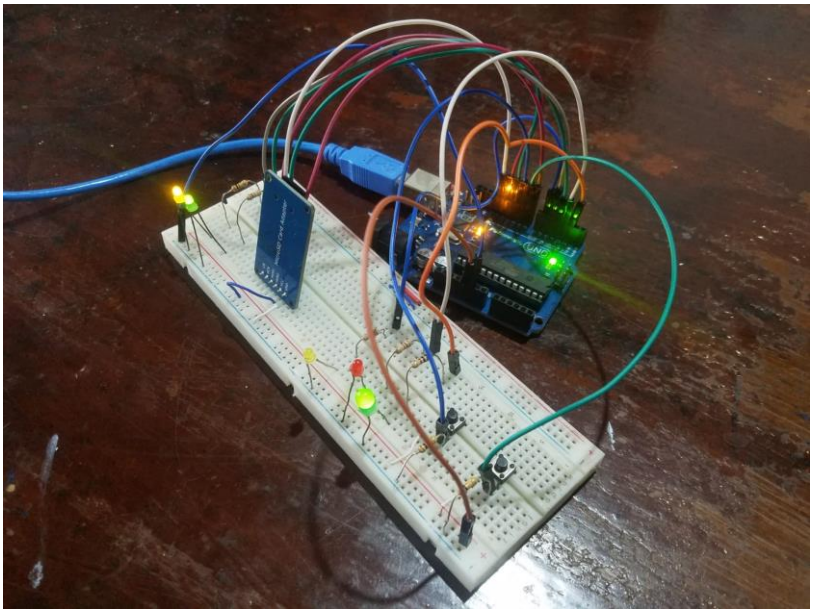


Figura 19. Montaje del circuito parte 1

Luego se le agregó la parte de potencia con los drivers y los motores paso a paso como se observa en la figura 20 quedando el montaje del circuito completo con una fuente de alimentación para la parte de potencia (figura 21).

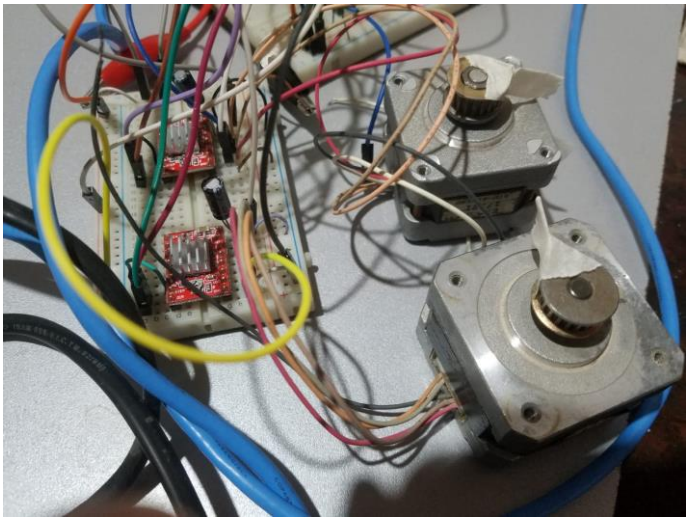


Figura 20. Montaje del circuito parte 2

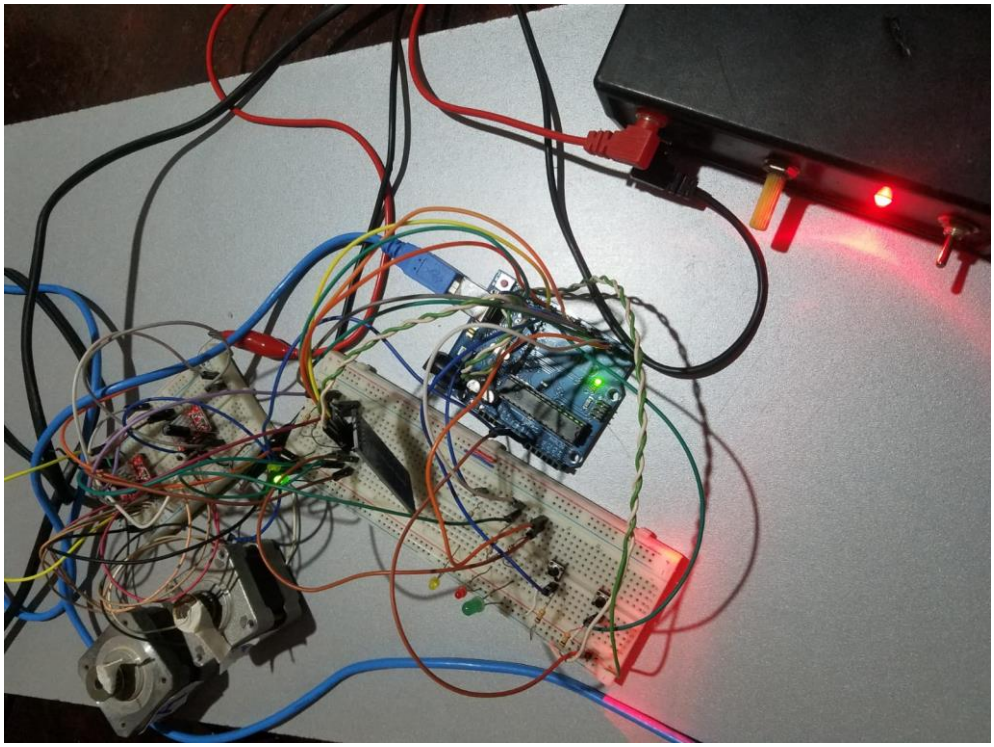


Figura 21. Montaje del circuito completo

Para simular el programa se ha definido “multiplicador” como igual a 3, es decir que se realizará cada 3 segundos una lectura de la trama (ya que el timer 1 tiene un tiempo de 1 segundo). Por lo tanto para que sea le una trama cada 15 minutos, multiplicador debe valer 900.

Al comenzar a ejecutarse el programa muestra que está desactivado (figura 22) y que para activarlo hay que ingresar el comando :C1.

```
————Heliostato domestico————

Se encuentra desactivado
(Pulse :C1 para activar y :C0 para desactivar)
```

Figura 22. Estado desactivado

En el estado activado (figura 23) se mostrará un menú de configuración, del ángulo de zenit y azimut deseado, si apunta al norte o al sur y el tiempo donde se quiere comenzar. Luego de eso se debe realizar la maniobra de homing para continuar con el programa.

```
:C1
comenzar:1

Iniciando...

Configure los agulos deseados donde quiere realizar la reflexion (:Z_ _->alpha_z, :A_ _ _->alpha_a)

Quiere reflejar hacia el N o hacia el S? (:N1->N , :NS->S)

El tiempo inicial sera el dado por :T_ _ _ _

Para continuar realize primero la maniobra de homing (:H)
```

Figura 23. Estado activado

En el estado homing (figura 24) se quedará esperando a que se presionen los finales de carrera de cada motor

```
Para continuar realize primero la maniobra de homing (:H)

:Z30
alpha_z:30.000000

:A0
alpha_a:0.000000

:N1
norte:1

:T0
tiempo aux:0

:H
Realizando maniobra de homing...
```

Figura 24. Realizando homing

Una vez que se haya completado pasará al estado funcionando donde se leerá el primer sector de la memoria (figura 25) micro SD comenzando en el registro 0x00270000. Se puede ver el contenido del buffer en el monitor serie de Termite.

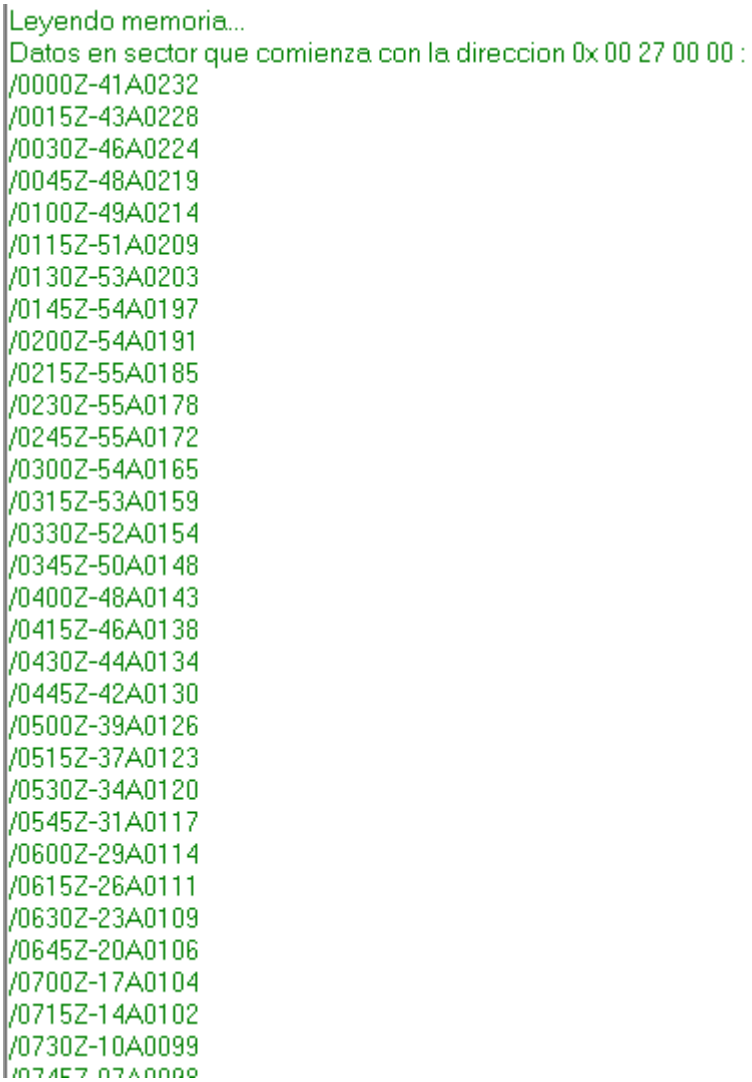


Figura 25. Contenido de la memoria en el sector de la dirección 0x00270000

Se observa en la figura 26 los ángulos de zenit y azimut que tiene el sol, los ángulos de beta_zenit y beta_azimut que serán los ángulos que tendría el helióstato en función de los ángulos del sol y los establecidos en la configuración (alpha_a y alpha_z), los ángulos de beta_zenit_aux y beta_azimut_aux que son los ángulos actuales del helióstato, se observa que el beta_azimut_aux es igual a -90° debido a la maniobra de homing. También se observa la cantidad de pulsos que se movió en el valor mov_motor1 y mov_motor2, no se moverá hasta que zenit será mayor a cero ya que no tiene sentido que se mueva si no hay luz solar. Se tiene información cada 15 minutos, comenzando a las 00:00, siguiendo 00:15, 00:30, 00:45, 01:00, etc.

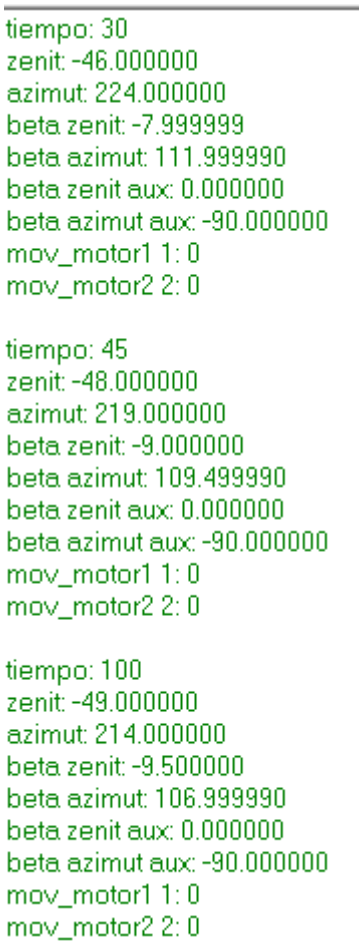


Figura 26. Ángulos para tiempo 30, 45, 100

En la figura 27 se observa que se lee la última trama del buffer, luego de eso se leerá el siguiente sector de memoria y se volcará en el buffer para luego ser mostrado.

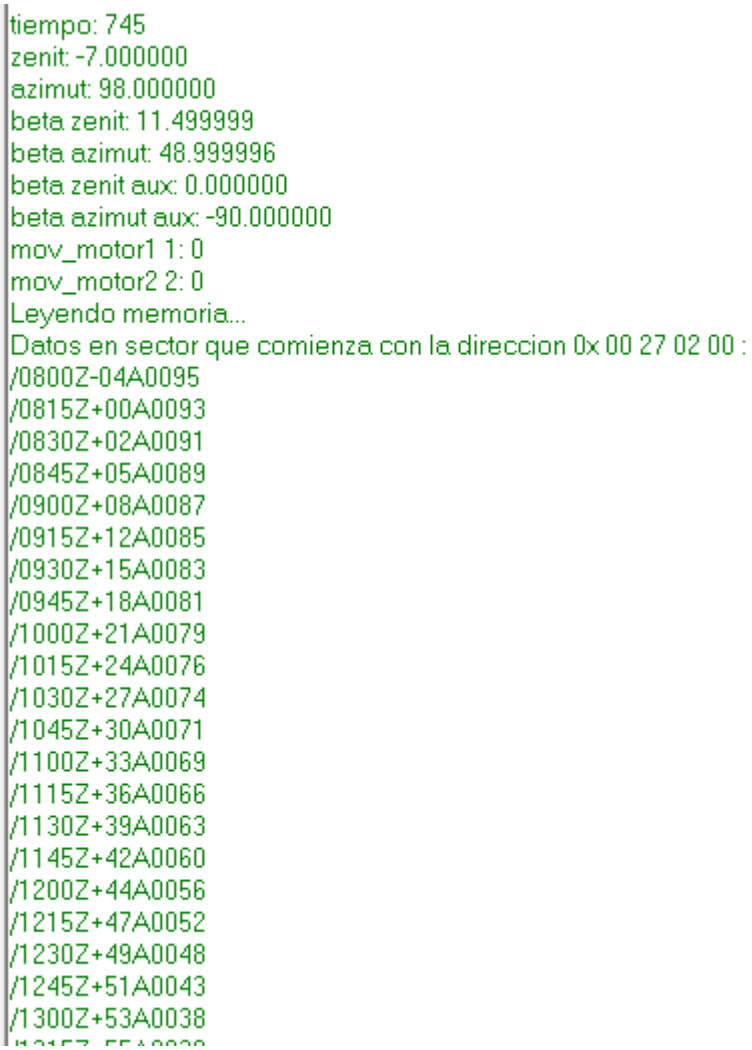


Figura 27. Contenido de la memoria en el sector de la dirección 0x00270200

En la figura 28 se aprecia que el ángulo de zenit deja de ser negativo y pasa a valer cero, por lo que se comenzarán a mover los motores una cantidad de pasos igual a la diferencia entre zenit y zenit aux dividido el paso del motor y lo mismo para azimut y azimut aux. Se observa como se van actualizando las posiciones de cada motor.

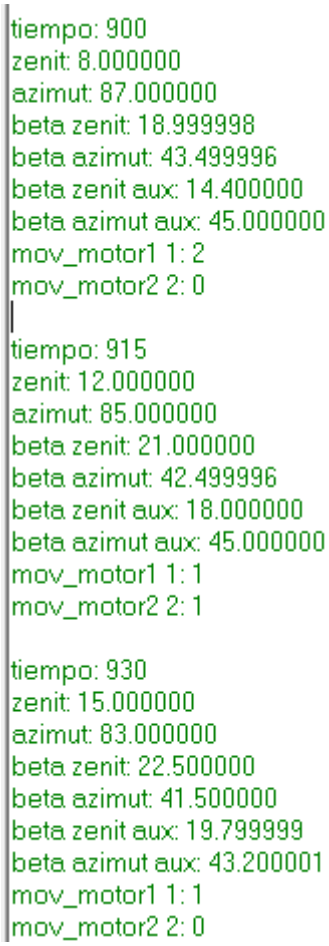


Figura 28. Ángulos para tiempo 900, 915, 930

El programa sigue leyendo la memoria, hasta el tercer sector y al llegar a la última trama del día (figura 29) se pasa al estado de homing nuevamente, luego de que se presionen los fines de carrera vuelve nuevamente al estado funcionando y así sucesivamente cada día.

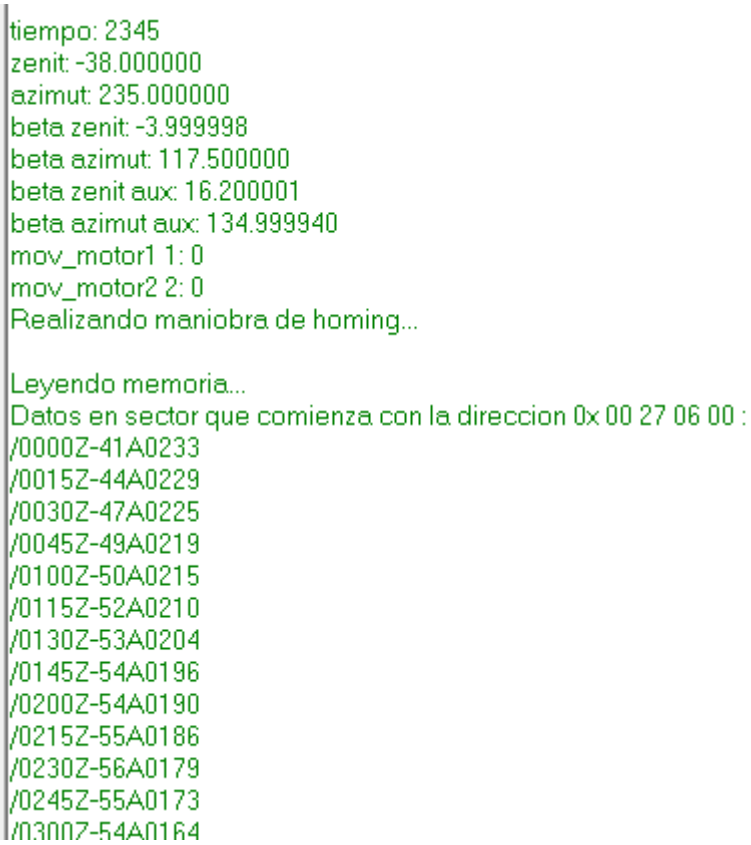


Figura 29. Contenido de la memoria en el sector de la dirección 0x00270600

Si en la configuración inicial se elige que comience en otro tiempo distinto de 0, como por ejemplo 500, que sería el tiempo en el que se lo configura e inicia por primera vez, el programa comenzaría a partir de ahí como se observa en la figura 30.

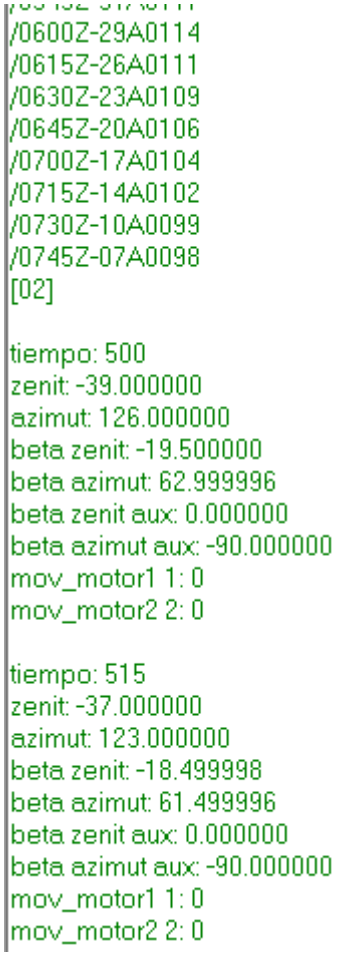


Figura 30. Comenzar en un tiempo distinto de cero

Resultados, especificaciones finales

Se pudieron cumplir con las metas propuestas inicialmente, se pudo llegar a pasar a código toda la idea del proyecto, adicionando detalles como los de la configuración inicial. A medida que se avanzada en el mismo, se iban incorporando ideas para desarrollar a futuro.

Este sistema al realizar un movimiento de un motor, ambos o ninguno cada 15 minutos conlleva un bajo consumo de energía. Se requiere una fuente capaz de alimentar ambos motores y con un regulador de voltaje esta se la puede llevar a que trabaje a niveles de tensión adecuados para el microcontrolador. La precisión es la del paso normal de los motores, pero esta puede ser mejorada trabajando con micro pasos, lo que aumentaría la cantidad de pasos necesaria y un aumento del consumo de energía.

Conclusiones. Ensayo de ingeniería de producto.

Debido a la realización del prototipo del helióstato doméstico se pudo comprender como se pueden obtener los ángulos que va describiendo el Sol en relación a la Tierra y como estos pueden ser usados para desarrollar distintos dispositivos como helióstatos, paneles solares que sigan la trayectoria, etc. También se pudo desarrollar un programa donde se controló dos motores paso a paso para tal fin. Se pudo comprender como es la estructura interna de una memoria micro SD y como es que se almacena la información en ella, para luego ser leída mediante el protocolo de comunicación SPI.

El prototipo planteado tiene varias mejoras posibles que se le pueden implementar las cuales se detallarán a continuación:

1. **Sistema de archivos**: en el prototipo no se utilizó sistema de archivos y se leía directamente en la dirección de memoria correspondiente a donde se hayan guardado las tramas, pero esto supone un problema ya que no se guardan en forma sucesiva, sino que hay saltos dentro de la memoria micro SD, lo cual se solucionaría utilizando esto. Además de que sería más sencillo el programa.
2. **Más días:** para hacer las pruebas se utilizó un día y parte del siguiente, pero si uno quisiera realizar un producto de este prototipo se debería incrementar la cantidad de días posibles, cubriendo hasta al menos una década de información guardada dentro de la tarjeta micro SD. Esto se podría con las ecuaciones correspondientes a los ángulos en los que se encontrará el sol y un algoritmo desarrollado para obtener estos datos.
3. **Trama que incluya fecha:** como se ha probado con un solo día, este solo de referencia temporal incluye las horas y minutos, como el buffer es de 512 bytes, la trama debería en lugar de ser de 16 bytes ser de 32 bytes.
4. **Configuración:** la configuración el prototipo se hace al comienzo de su instalación comunicando la PC con el microcontrolador mediante la UART, pero como producto si el usuario decidiera moverlo de lugar o cambiarle los ángulos no podría, por ello es necesario hacer una aplicación móvil mediante una interfaz friendly user para que pueda cambiar el valor de estas variables.
5. **GPS**: también se puede agregar un GPS para que los valores de los ángulos que son calculados en función de las coordenadas donde se encuentre el helióstato, se pueden obtener en cualquier parte del mundo.

Referencias

Bibliografía y datasheets:

Apuntes de cátedra (aula abierta)

11 de enero de 2017, por Narod Stream. Publicado en Programación AVR. Lección 33.
https://narodstream.ru/avr-urok-33-spi-karta-sd-fat-chast-1/

ATmega 328P datasheet :
https://html.alldatasheet.com/html-pdf/313560/ATMEL/ATmega328P/3969/25/ATmega328P.html

Driver A4988 datasheet:
https://www.alldatasheet.com/datasheet-pdf/pdf/338780/ALLEGRO/A4988.html

SD Card datasheet:
https://curtocircuito.com.br/datasheet/modulo/cartao_micro_SD.pdf

SD Especificaciones:
https://narodstream.ru/avr/download/LESSON33/Simplified_Physical_Layer_Spec.pdf

Herramientas:

Atmel Studio

Termite:
https://www.compuphase.com/software_termite.htm

HxD:
https://en.wikipedia.org/wiki/HxD

Solartopo (aplicación para obtener las coordenadas):
http://www.solartopo.com/orbita-solar.htm

Anexos

Componentes para la realización del proyecto:
1x Arduino UNO
2x Driver A4988
2x Motor paso a paso
1x Módulo micro SD Arduino
2x Pulsadores (Fines de carrera)
5x Resistencias de 1kohm
2x Resistencias de 10khom
2x Capacitor de 100microFaradios
5x LEDs
2x Placa experimental
1x Fuente de alimentación regulable
Cables

Junto al informe y código se anexa el archivo “coordenadas.txt”