

IOT – Creación de prototipos con Raspberry Pi

Localizador de astros

MEMORIA

Autor: Josué López
Director: Joan Masdemont
Convocatoria: 06/2019

CIFO LA VIOLETA



Resumen

El objetivo de nuestro proyecto es diseñar una herramienta de astronomía de código abierto y bajo coste que pueda ayudar en la localización de estrellas.

Efectivamente, el mapa del cielo puede ser algo difícil de interpretar cuando alguien se inicia en astronomía. Por esta razón, se ha diseñado y construido un puntero láser motorizado de 2 ejes. El usuario elige un astro en Stellarium (software de astronomía de código abierto) y el puntero láser lo localiza en el cielo nocturno real.

Se ha desarrollado un código de Python que:

1. toma la hora actual + coordenadas GPS + orientación + astro objetivo
2. traduce esta información en ángulos de destino
3. convierte estos ángulos en comandos *Goto* para los motores paso a paso.

Sumario

<u>RESUM.....</u>	1
<u>SUMARI.....</u>	2
<u>1. GLOSSARI.....</u>	3
<u>2. PREFACI.....</u>	5
2.1. Origen del projecte.....	5
2.2. Motivació.....	5
2.3. Requeriments previs.....	5
<u>3. INTRODUCCIÓ.....</u>	7
3.1. Objectius del projecte.....	7
3.2. Abast del projecte.....	7
<u>4. TEXT DEL CAPÍTOL 4 [CTRL + SHIFT + 1].....</u>	9
4.1. Text de l'apartat 1 del capítol 4 [Ctrl + Shift + 2].....	9
4.1.1. text del subapartat 1 del apartat 1 del capítol 4 [Ctrl + Shift + 3].....	9
<u>CONCLUSIONS.....</u>	11
<u>AGRAÏMENTS.....</u>	13
<u>BIBLIOGRAFIA.....</u>	15
Referències bibliogràfiques.....	15
Bibliografia complementària.....	16

1. Glosario

2. Introducción

No ha de repetir o parafrasejar el resum, ni donar detalls de la teoria, l'experimentació, el mètode o els resultats, ni anticipar les conclusions o les recomanacions.

2.1. Objetivos del proyecto

El objetivo de nuestro proyecto es diseñar una herramienta de astronomía de código abierto y bajo coste que pueda ayudar en la localización de estrellas.

Se trata de diseñar y construir un puntero láser motorizado de 2 ejes que localiza en el cielo nocturno real el astro elegido por el usuario.

2.2. Alcance del proyecto

Se desarrollará un código de Python que:

1. toma la hora actual + coordenadas GPS + orientación + astro objetivo
2. traduce esta información en ángulos de destino
3. convierte estos ángulos en comandos *GoTo* para los motores paso a paso.

3. Conceptos básicos

3.1. Sistemas de coordenadas

3.1.1. Conceptos básicos

El Sol, debido al movimiento real de la Tierra, describe una trayectoria aparente sobre la esfera celeste denominada, al igual que el plano que la contiene, trayectoria eclíptica. A la línea perpendicular a dicho plano se le denomina eje de la eclíptica y la oblicuidad de la eclíptica es el ángulo que forma la eclíptica con el ecuador celeste, que actualmente vale $23^{\circ} 26'$.

La línea de equinoccios es la intersección del ecuador con la eclíptica. Al punto donde se proyecta el Sol al pasar del Hemisferio Sur al Hemisferio Norte se le denomina punto vernal o punto Aries.

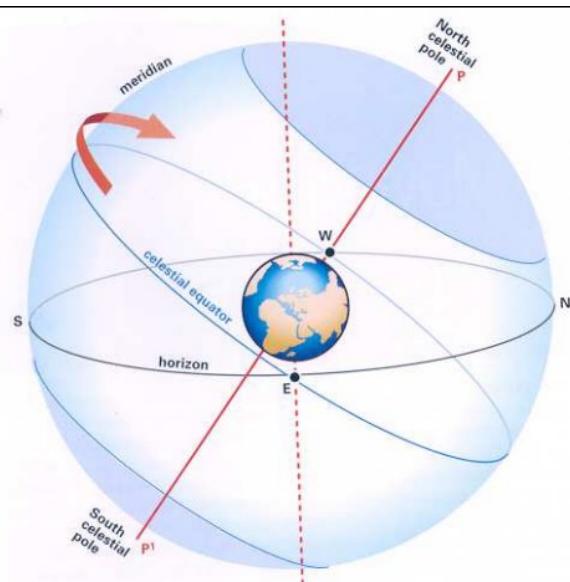


Figura F.1: xxx

3.1.2. Coordenadas horizontales

El sistema de coordenadas horizontales puede ser usado para situar una dirección de vista (el ángulo de acimut) y la altura en el cielo (el ángulo de altitud). Este sistema de coordenadas es atractivo por ser intuitivo. Sin embargo, los valores de altitud y acimut para un objeto en el cielo cambian con el tiempo y con la ubicación del observador. Esto significa que dichas coordenadas son locales.

El ángulo de **altitud** se mide hacia arriba desde el horizonte. Mirando verticalmente hacia arriba (al céñit), será de 90° ; a medio camino entre el céñit y el horizonte, será de 45° , etc. El punto opuesto al céñit se llama nadir (ver figura F.1).

El ángulo de **acimut** es el ángulo medido sobre el horizonte celeste que forman el punto cardinal Norte y la proyección vertical del astro sobre el horizonte del observador . Se mide en grados desde el punto cardinal Norte (norte geográfico) en el sentido de las agujas del reloj, o sea Norte-Este-Sur. Por tanto, el del norte es 0° , el del este 90° , el del suroeste 135° , etc.

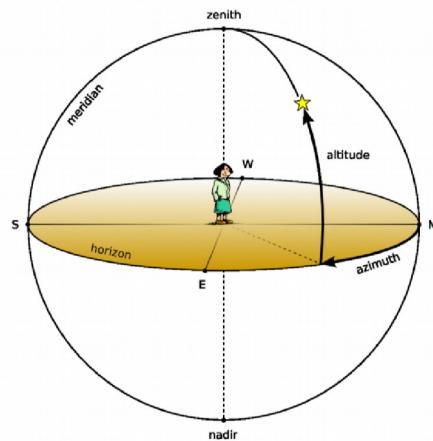


Figura F.1: Ángulos altitud/acimut

El acimut podrá ser magnético si se mide respecto al norte magnético (Azm). En este caso se le denomina rumbo. Para convertir un rumbo a un acimut es necesario primero conocer la declinación magnética. La declinación magnética en un punto de la Tierra es el ángulo comprendido entre el norte geográfico (o norte verdadero) y el norte magnético local. En otras palabras, es la diferencia entre el norte verdadero y el indicado por una brújula.

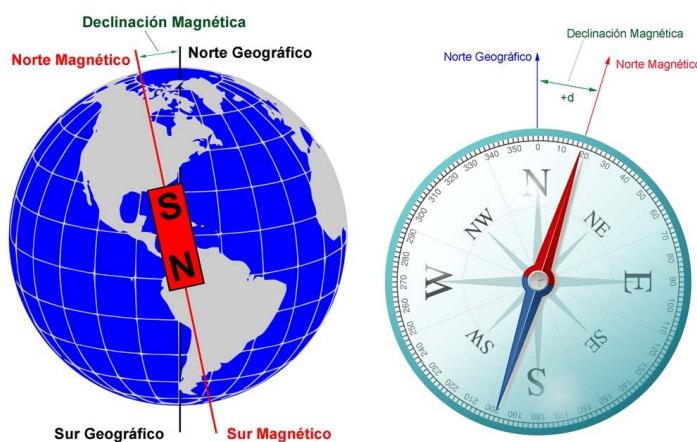


Figura F.1: Declinación magnética

La declinación magnética depende de la ubicación y puede variar sensiblemente de un lugar

a otro. Por ejemplo, un viajero que se traslade de la costa occidental a la costa oriental de Estados Unidos puede registrar una variación de veinte a treinta grados. Si bien la declinación magnética también varía con el transcurso del tiempo, su valor en un área dada cambia muy lentamente. Es posible que cada cien años la velocidad de cambio llegue a ser de 2 a 2,5 grados.

Por convención, a la declinación se le considera de valor positivo si el norte magnético se encuentra al este del norte verdadero, y negativa si se ubica al oeste. De esta forma:

- si la declinación magnética es al Este, entonces el acimut va a ser el rumbo más la declinación magnética ($Az = Rm + Dm$)
- si la declinación magnética es al Oeste entonces el acimut es igual al rumbo menos la declinación magnética ($Az = Rm - Dm$).

Ejemplo: necesito encontrar el acimut en un punto donde el rumbo es de 60° y la declinación magnética es de 5° Oeste (-5°). Utilizando la fórmula: $Az = Rm + Dm = 60^\circ + (-5^\circ) = 55^\circ$

En nuestro caso, en el que las coordenadas de ubicación del CIFO La Violeta son ($41^\circ 25' 01.5''$ [Norte], $2^\circ 08' 01.1''$ [Este]), el valor de la declinación magnética es de $0^\circ 53'$ [Este].

3.1.3. Coordenadas ecuatoriales

El sistema de coordenadas ecuatoriales utiliza dos ángulos para describir posiciones en el cielo, ascensión recta y declinación. Estos ángulos son medidos desde puntos estándar en la esfera celeste.

La ascensión recta, abreviadamente AR y denotada por α , es el ángulo, medido sobre el ecuador celeste, abarcado entre el Punto Aries (equinoccio vernal) y el meridiano que pasa por el objeto observado. Equivale a la longitud geográfica. Su sentido positivo es el antihorario. Sus unidades son las angulares, expresadas en horas: 24 horas se corresponden a 360° . Es decir que 1 hora equivalen a 15° , ó 1° equivale a 4 minutos horarios.

La declinación, denotada por δ , es el ángulo que forman el ecuador celeste y el objeto. Equivale a la latitud geográfica. Para objetos situados entre el ecuador y el polo norte, la declinación es positiva y, en caso contrario, negativa. Así, el polo norte celeste tiene una declinación de 90° , el ecuador celeste de 0° , y el polo sur celeste de -90° .

La figura F.3 ilustra las coordenadas ecuatoriales de ascensión recta/declinación.

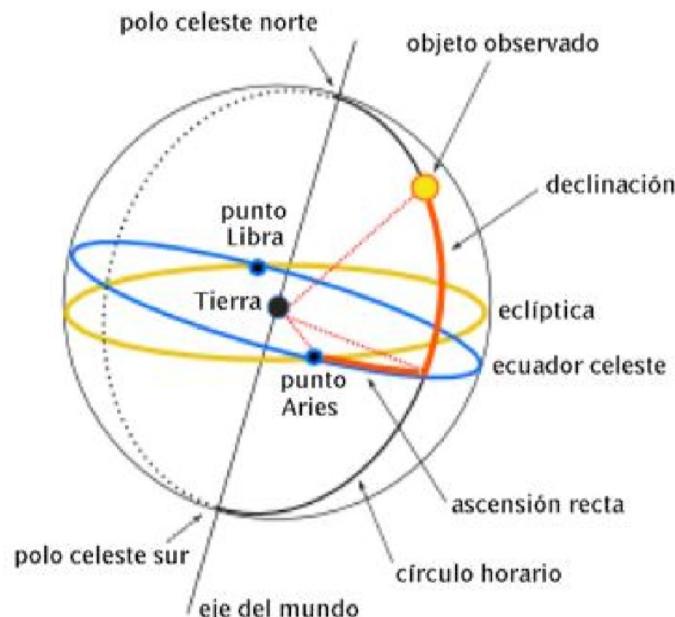


Figura F.3: Ascensión recta/declinación

Las coordenadas ecuatoriales de una estrella no cambian con la posición del observador ni con el transcurso del día debido a la rotación de la Tierra.

3.1.4. Coordenadas eclípticas

Las coordenadas eclípticas permiten determinar la posición de un objeto celeste respecto al plano de la eclíptica y al Punto Aries.

Las dos coordenadas son la longitud celeste, denotada por λ , medida sobre la eclíptica a partir del punto Aries y en sentido antihorario, y la latitud celeste, denotada por β , que es el ángulo que forma el astro con la eclíptica.

3.1.5. Conversión de coordenadas ecuatoriales a coordenadas eclípticas

Las fórmulas para convertir las coordenadas ecuatoriales en coordenadas eclípticas son:

- $\sin \lambda \cos \beta = \sin \delta \sin \epsilon + \cos \delta \cos \epsilon \sin \alpha$ (1)
- $\cos \lambda \cos \beta = \cos \delta \cos \epsilon$ (2)
- $\sin \beta = \sin \delta \cos \epsilon - \cos \delta \sin \epsilon \sin \alpha$ (3)

, donde α es la ascensión recta, δ es la declinación, ϵ es la oblicuidad de la eclíptica y vale $\epsilon = 23^\circ 26'$, λ es la longitud celeste y β es la latitud celeste.

3.1.6. Coordenadas terrestres

La **latitud** es el arco contado desde el ecuador al punto donde se encuentra el observador. Las líneas de latitud (paralelos) rodean la circunferencia de la tierra en el plano horizontal. La línea de partida es el ecuador (latitud 0°) que divide a la Tierra en los hemisferios boreal y austral. Desde este punto se dibujan paralelos a éste cada 15° siendo su numeración convencionalmente positiva hacia el polo Norte (latitud 90°) y negativa hacia el polo Sur.

Las líneas de **longitud** (meridianos) van de polo a polo y dividen la circunferencia de la Tierra (el Ecuador) en 24 horas. La referencia donde está la hora 0 pasa por el meridiano de Greenwich. Desde este punto los meridianos y las horas avanzan hacia el este. Para medir el meridiano se mide el ángulo entre el meridiano 0 hasta donde está el observador.

La figura F.2 ilustra las coordenadas de latitud/longitud.

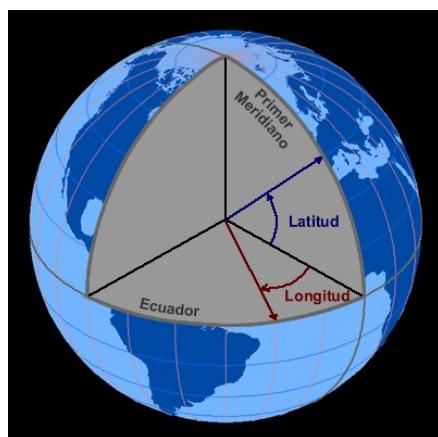


Figura F.1: Ángulos latitud/longitud

3.1.7. Coordenadas de Google Maps

Supongamos que queremos localizar con Google Maps la ubicación de un observador. Obtenemos 2 coordenadas: X, Y.

La coordenada X coincide exactamente con la latitud. Efectivamente, es el arco contado desde el ecuador (latitud 0°) al punto donde se encuentra el observador. Su numeración es positiva hacia el polo Norte (latitud 90°) y negativa hacia el polo Sur (latitud -90°).

La coordenada Y coincide exactamente con la longitud, pero de signo contrario. Efectivamente, es el arco contado desde el meridiano de Greenwich (longitud 0°) al punto donde se encuentra el observador. Su numeración es positiva hacia oriente y negativa hacia occidente.

Ejemplo: Supongamos que queremos localizar con Google Maps la ubicación del CIFO La

Violeta. Obtenemos las siguientes coordenadas: 41.417106°, 2.133646°.

Entonces, tenemos:

- latitud = 41°25'01", longitud = -2°08'01"

3.2. Unidades

3.2.1. Distancia

Los astrónomos utilizan una serie de unidades para la distancia que dan sentido a la vastedad del espacio.

1. Unidad astronómica (UA)

Es la distancia media entre la Tierra y el Sol. Aproximadamente, unos 150 millones de kilómetros. La UA se utiliza principalmente para objetos del sistema solar – por ejemplo, la distancia de los planetas al Sol.

2. Año luz

Es la distancia que la luz recorre en un año. La velocidad de la luz es de, aproximadamente, 300.000 km/s. Esto significa que un año luz es una distancia de unos 9,5 billones de km. Los años luz se utilizan frecuentemente para describir la distancia de las estrellas y galaxias, o los tamaños de grandes objetos, como galaxias y nebulosas.

3.2.2. Tiempo

En astronomía se le llama día solar al tiempo que tarda el Sol en viajar desde el punto más alto del cielo a mediodía, hasta ese mismo punto al día siguiente (i.e. 24h). Sin embargo, en ese tiempo la Tierra no solo gira, también se mueve ligeramente alrededor de su órbita. En consecuencia, en un día solar la Tierra gira más de 360° sobre su eje.

En astronomía se le llama día sideral al tiempo que tarda la Tierra en girar exactamente 360°, decir, en completar una revolución en el cielo. Un día sideral dura unas 23h, 56 minutos y 4 segundos. Los astrónomos encuentran útil el tiempo sideral para la observación

La figura F3 ilustra el movimiento de la Tierra alrededor del Sol. El triángulo rojo sobre la tierra representa la ubicación de un observador. La figura muestra la Tierra en cuatro tiempos:

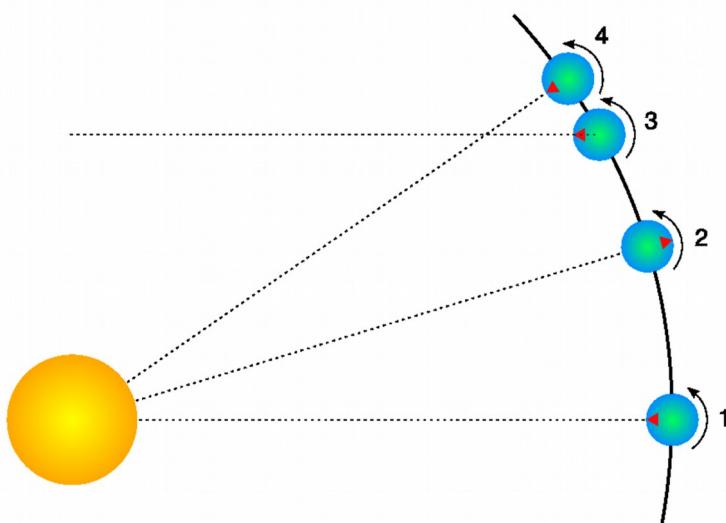


Figura F.3: Días solares y días siderales

1. El Sol está directamente encima. Es mediodía.
2. Han pasado 12 h desde 1. La Tierra ha rotado y el observador está ahora en el lado opuesto al Sol. Es medianoche. La Tierra se ha movido también un poco alrededor de su órbita.
3. La Tierra ha girado exactamente 360° . Ha pasado un día sideral exacto desde 1.
4. Es mediodía de nuevo. Ha pasado exactamente un día solar desde 1. Observe que la Tierra ha girado más de 360° desde 1.

3.2.3. Ángulos

Los astrónomos normalmente usan grados para medir ángulos. Puesto que muchas observaciones requieren medidas muy precisas, el grado está subdividido en sesenta minutos sexagesimales o minutos de arco, también conocidos como arco-minutos. Cada minuto de arco se subdivide, a su vez, en sesenta segundos sexagesimales, segundos de arco o arco-segundos. Así, un grado es igual a 3.600 segundos de arco.

Los grados se denotan mediante el símbolo $^\circ$ tras un número. Los minutos de arco, se denotan con el símbolo $'$, y los segundos de arco con $''$.

Los ángulos frecuentemente se escriben en dos formatos:

1. Formato DMS (i.e. grados, minutos y segundos). Por ejemplo, $90^\circ 15' 12''$. Cuando se requiere más precisión los segundos incluyen una parte decimal (e.g. $90^\circ 15' 12,432''$).

2. Grados decimales. Por ejemplo: 90,2533º.

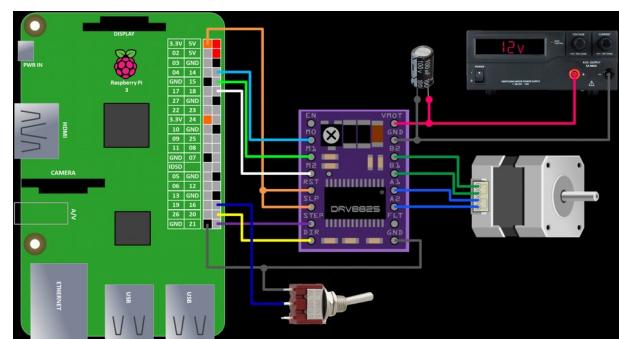
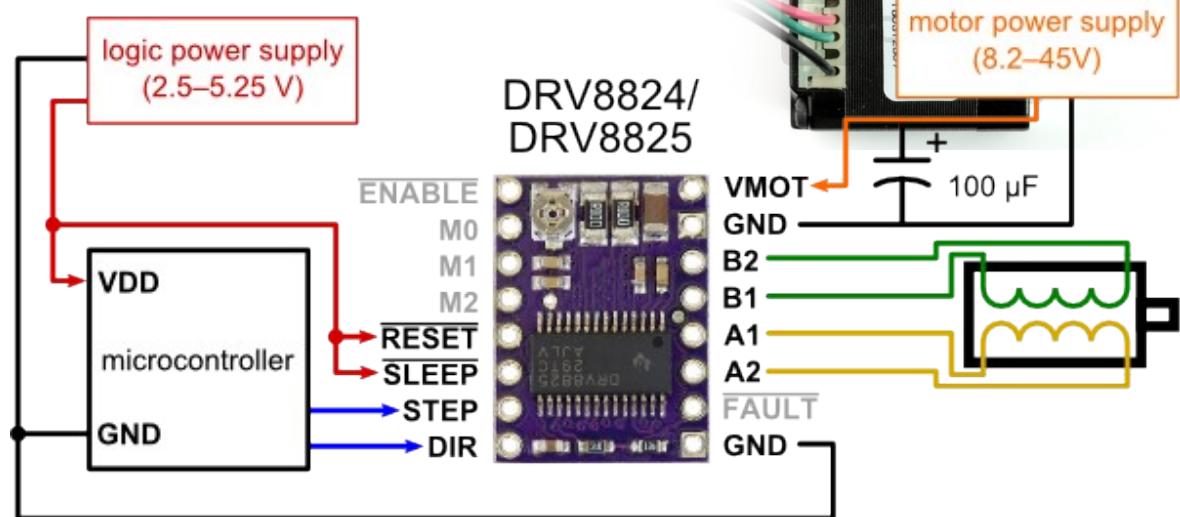
4. Hardware

4.1. Componentes mecánicos

xxx

4.2. Componentes electrónicos

xxx



5. Software

5.1. xxx

5.1.1. xxx

xxx

6. Utilización de los sensores de Android

6.1. Introducción

El objetivo es hacer uso de los sensores de un teléfono inteligente como IMU (Inertial Measurement Unit). Se trata de conectar el terminal a la Raspberry mediante un cable USB y enviarle los datos de los sensores (i.e. GPS, orientación) a través de un socket UDP (User Datagram Protocol)*. Incluso podríamos conectarnos mediante wifi a través del teléfono Android. En ese caso no sería necesario ni *dongle* wifi ni cable Ethernet.



Figura F.3: Conexión del teléfono inteligente a la Raspberry

*UDP (User Datagram Protocol) <http://www.chuidiang.org/clinux/sockets/udp/udp.php>

Los sockets UDP son sockets no orientados a conexión. Esto quiere decir que un programa puede abrir un socket y ponerse a escribir mensajes en él o leer, sin necesidad de esperar a que alguien se conecte en el otro extremo del socket.

El protocolo UDP, al no ser orientado a conexión, no garantiza que el mensaje llegue a su destino. Parece claro que si mi programa envía un mensaje y no hay nadie escuchando, ese mensaje se pierde. De todas formas, aunque haya alguien escuchando, el protocolo tampoco garantiza que el mensaje llegue. Lo único que garantiza es, que si llega, llega sin errores.

¿Para qué sirve entonces?. Este tipo de sockets se suele usar para información no vital, por ejemplo, envío de gráficos a una pantalla. Si se pierde algún gráfico por el camino, veremos que la pantalla pierde un refresco, pero no es importante. El que envía los gráficos puede estar dedicado a cosas más importantes y enviar los gráficos sin preocuparse (y sin quedarse bloqueado) si el otro los recibe o no.

Durante la fase de configuración necesitaremos alguna forma de conexión a la Raspberry. Esto significa que la Raspberry deberá estar conectada mediante un cable HDMI y su

propio teclado (y el mouse, si usamos el entorno gráfico), o bien mediante una conexión wifi/ethernet para acceder remotamente a ella mediante protocolo ssh (*Secure SHell*).

También necesitaremos disponer de una red de área local o LAN (*Local Area Network*) a la que conectaremos la Raspberry y el teléfono inteligente a través de un enrutador habilitado para wifi. Con pequeños cambios (en teoría, solo las direcciones IP utilizadas cambiarían), se podría preparar una pequeña LAN inalámbrica utilizando la función de punto de acceso móvil para teléfonos inteligentes.

Las diferentes marcas de teléfonos inteligentes pueden tener menús de configuración ligeramente diferentes, aunque deberían ser muy similares.

6.2. Activación del enlace USB en el teléfono Android

Lo primero que haremos será configurar el teléfono y la Raspberry para que esta última tenga comunicación LAN con el teléfono. De esta manera se podrá acceder a ella (al menos parcialmente) desde la misma LAN a la que el teléfono inteligente está conectado vía wifi. El procedimiento es el siguiente:

- Arrancaremos la Raspberry.
- Conectaremos el cable USB entre la Raspberry y el teléfono inteligente.
- En el teléfono, iremos a *Configuración -> Conexiones -> Punto de acceso móvil y conexión* y activaremos la conexión USB.

Este paso puede ser diferente en cada teléfono. Buscaremos en Internet la conexión compartida o USB en el modelo específico y la versión de Android.

La Raspberry la detectará automáticamente y configurará una nueva interfaz de red mediante USB y con una dirección IP asignada dinámicamente por el teléfono.

- Iremos a la Raspberry (pantalla física o ssh) y verificaremos la dirección IP que le ha sido asignada a través del USB. Para hacer esto, simplemente usaremos la consola/xterminal o ssh y llamaremos a *ifconfig* (vea imagen F.xx). Buscaremos la sección que describe la interfaz llamada *usb0*. En esta sección se verá la palabra *inet* seguida de una dirección IP. Esta es la dirección que el teléfono asignó a la Raspberry. En la siguiente imagen, la dirección 192.168.42.100 ha sido asignada a la Raspberry.

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b8:27:eb:4a:3e:0f txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1 (Local Loopback)
        RX packets 30 bytes 5372 (5.2 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 30 bytes 5372 (5.2 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.42.100 netmask 255.255.255.0 broadcast 192.168.42.255
        inet6 fe80::7bf2:98c1:16a1:518d prefixlen 64 scopeid 0x20<link>
            ether 92:e6:5f:18:63:3e txqueuelen 1000 (Ethernet)
            RX packets 70 bytes 10145 (9.9 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 80 bytes 15105 (14.7 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~ $ |
```

Figura F.x: Salida del comando *ifconfig* en la consola de mi Raspberry

6.3. Configuración de una IP estática para la Raspberry

En teoría, cada vez que activemos el módem USB, el teléfono podría asignar una IP diferente a la Raspberry. Si esto es así, cada vez que se conecta al teléfono deberemos buscar la IP de Raspberry y cambiar su configuración en el teléfono para usar la nueva IP y poder enviarle los datos de los sensores. Lógicamente, esto no es conveniente. Por suerte, puede evitarse indicando a la Raspberry que establezca una IP estática específica. El procedimiento es el siguiente:

- Modificaremos el archivo */etc/dhcpcd.conf* y agregaremos las siguientes líneas al final:

```
interface usb0
static ip_address=192.168.42.100
```

- Reiniciaremos la Raspberry.

- Volveremos a activar el módem USB en el teléfono una vez que se reinicie la Raspberry.

Si todo salió bien, se debería ver esta IP en la interfaz *usb0* de Raspberry cuando llamemos a *ifconfig*.

En cualquier caso, la dirección IP que establecimos como estática no tiene que ser necesariamente la misma que el teléfono le asignó dinámicamente.

6.4. Habilitación de la conexión ssh a la Raspberry a través de la wifi del teléfono

En este punto, la Raspberry debería tener un conjunto de IP conocido (192.168.42.100) y debería estar correctamente conectada al teléfono. Sería bueno habilitar conexión ssh a través de la wifi del teléfono, pero la Raspberry no parece ser accesible desde dispositivos conectados a la misma wifi que el teléfono. Afortunadamente, hay aplicaciones que permiten realizar *port forwarding* en tu dispositivo Android.

En nuestro caso usaremos la aplicación llamada *Fwd: the port forwarding* de Elixsr Ltd y que podemos encontrar en Google Play Store (<https://play.google.com/store/apps/details?id=com.elixsr.portforwarder>). El procedimiento es el siguiente:

- Agregaremos una regla que indique a la aplicación que las conexiones que llegan al teléfono a través de un puerto no restringido deben reenviarse a otro puerto en la Raspberry. En nuestra Raspberry (IP 192.168.42.100) tenemos un servicio ssh en ejecución que está conectado al puerto predeterminado 22. Por lo tanto, la regla a establecer consistirá en reenviar las conexiones que llegan a un puerto (e.g. port 52000) del teléfono al puerto 22 de la Raspberry.
- Estableceremos la interfaz en el teléfono que está recibiendo la conexión. En nuestro caso será la interfaz wifi (i.e. *wlan0*):

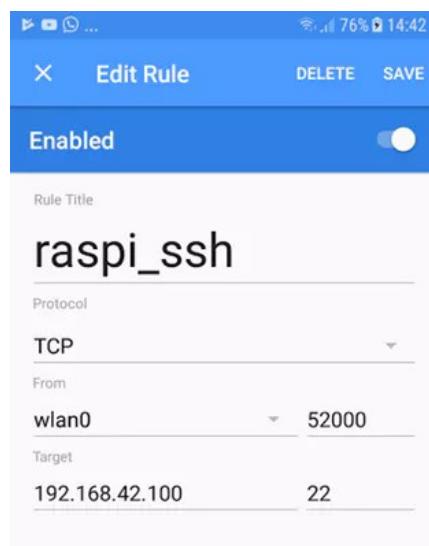


Figura F.3: Configuración de reglas en la aplicación *Fwd: the port forwarding*

Esto podría leerse como "reenviar conexiones a la interfaz wifi de este teléfono (wlan0) en el puerto 52000, al puerto 22 de la Raspberry (IP 192.168.42.100)".

Una vez que se crea la regla (importante guardar y habilitar), simplemente regresamos y pulsamos inicio.

En este punto, cualquier conexión al teléfono vía wifi a través del puerto 52000 se reenviará al puerto 22 de la Raspberry. Eso significa que podremos conectarnos a la Raspberry mediante protocolo ssh a través de la IP wifi del teléfono usando el puerto 52000.

Esto podemos verlo en *Configuración -> Acerca de este teléfono -> Estado -> dirección IP*. Si usamos el comando estándar de Linux ssh, debería ser algo así (asumiendo que la IP del teléfono es 192.168.1.56 y el usuario de Raspberry es pi):

```
> ssh -p 52000 pi@192.168.1.56
```

Si usamos otras aplicaciones ssh como Putty o Bitvise, cambiaremos el puerto a 52000, porque estas aplicaciones solo intentarán conectarse al puerto predeterminado 22 si no se especifica lo contrario.

Ahora ya podemos desconectar de la Raspberry el monitor y teclado, y quitar el *dongle* de Ethernet y wifi, dejando solo el teléfono como proveedor de conexión.

6.5. Envío de los datos de los sensores a la Raspberry a través de UDP

La aplicación "Sensor UDP", desarrollada por Pasan Weerasing y que podemos encontrar en Google Play Store, permite enviar datos de los sensores de su dispositivo Android a través de un socket UDP a la Raspberry.

Esta aplicación lee la información de los sensores que definamos y la envía a un host y puerto especificados utilizando el protocolo UDP. Solamente necesitamos configurar la IP del host y el puerto al que deseamos enviar los datos. En nuestro caso, queremos enviar a Raspberry (192.168.42.100) y al puerto 50000 (por ejemplo). Introduciremos estos valores y presionaremos el botón "ENVIAR DATOS".

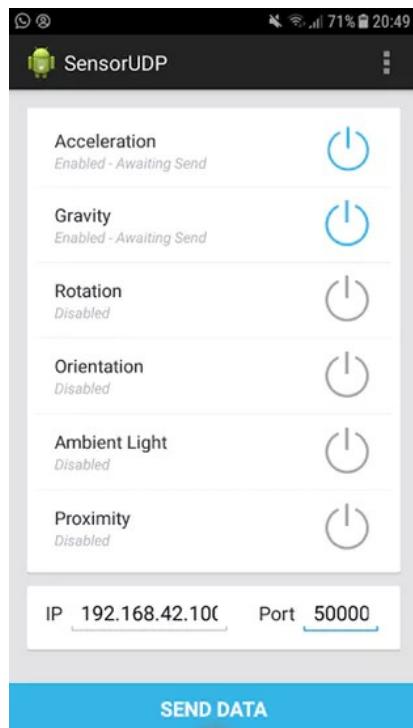


Figura F.3: Configuración de la aplicación *Sensor UDP*

- Includes support for Linear Acceleration, Gravity, Rotation, Orientation, Ambient Light and Proximity sensors
- Displays sensor values on screen during wireless transmission
- Checks device for sensor availability. It is important to note that not all devices support all sensors.
- Includes a keyboard to 'create notes' for performance
- Values are stored in a float array, with ordering consistent to the user interface.

USAGE INSTRUCTIONS:

The data is stored within the app as a float array. See the format below.

Format: [X Acceleration, Y Acceleration, Z Acceleration, X Gravity, Y Gravity, Z Gravity, X Rotation Rate, Y Rotation Rate, Z Rotation Rate, X Orientation (Azimuth), Y Orientation (Pitch), Z Orientation (Roll), deprecated, deprecated, Ambient Light, Proximity, Keyboard Buttons 1 - 8]

6.6. Lectura de los datos de los sensores en la Raspberry

En este punto, los datos deberían enviarse al puerto 50000 de Raspberry pero aún necesitamos leerlos de alguna manera. Para ello un simple *script* de Python es suficiente. El código es muy simple y se tomó del enlace en la página de la aplicación del sensor UDP en Google Play Store. El código se encuentra en el anexo y se llama *sensors.py*. Debemos copiarlo en la Raspberry y ejecutarlo invocando *python sensors.py* en la carpeta donde lo copiamos.

```
> cd <folder_where_you_copied_the_sensors.py_file>
> python sensors.py
```

La salida a la consola es solo un flujo continuo de líneas con los valores de sus sensores., Cambiaremos los sensores activados en la aplicación *Sensor UDP* para saber qué campos en las cadenas corresponden a qué sensores

```
0.0000 0.0000 0.0000
received message: -0.0001 -0.0007 0.0013 0.0071 -0.0059 -1.0006 0.0011 -0.0011 -0.0009
119.1120 -0.3314 0.4089 0.0000 0.0000 36.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
received message: -0.0001 -0.0007 0.0013 0.0071 -0.0059 -1.0006 0.0023 -0.0017 -0.0009
119.1120 -0.3314 0.4089 0.0000 0.0000 36.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
received message: -0.0001 -0.0007 0.0013 0.0071 -0.0059 -1.0006 0.0023 -0.0017 -0.0009
119.1120 -0.3314 0.4089 0.0000 0.0000 36.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
received message: -0.0003 -0.0004 0.0015 0.0071 -0.0059 -1.0006 -0.0007 -0.0011 -0.0028
119.2156 -0.3241 0.4157 0.0000 0.0000 36.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
received message: -0.0003 -0.0004 0.0015 0.0071 -0.0059 -1.0006 -0.0007 -0.0011 -0.0028
119.2156 -0.3241 0.4157 0.0000 0.0000 36.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
received message: -0.0003 -0.0004 0.0015 0.0071 -0.0059 -1.0006 0.0011 0.0014 -0.0022 1
19.2156 -0.3241 0.4157 0.0000 0.0000 36.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
received message: 0.0004 0.0018 0.0065 0.0071 -0.0059 -1.0006 -0.0001 0.0001 0.0009 119
.1524 -0.3793 0.4004 0.0000 0.0000 36.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
received message: 0.0004 0.0018 0.0065 0.0071 -0.0059 -1.0006 -0.0001 0.0001 0.0009 119
.1524 -0.3793 0.4004 0.0000 0.0000 36.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

Figura F.3: Salida del script *sensors.py*

El *script* muestra cómo obtener los datos que se envían a la Raspberry. Se trata ahora de aislar aquellos fragmentos que son de nuestro interés. La función clave de python es *unpack_from* en el módulo *struct*.(<https://docs.python.org/2/library/struct.html>)

6.7. Acceso a los sensores de ANDROID con QPython

<https://masquesig.com/2015/03/04/qpython-programa-con-tu-terminal-android-acceso-al->

gps-y-mucho-mas/

7. Motores paso a paso

7.1. Introducción

En esta sección cubriremos el principio de funcionamiento de los motores paso a paso y las tipologías de construcción. Además, explicaremos los modos de control de los motores paso a paso bipolares en una Raspberry Pi con Python utilizando un controlador de motor paso a paso DRV-8825.

Los motores paso a paso son motores de corriente continua sin escobillas cuya rotación se divide en un número finito de pasos que proporciona un control posicional y una repetibilidad muy precisos sin ningún sensor de retroalimentación, lo que posibilita un control de lazo abierto. Esto los hace muy populares para impresoras 3D, enrutadores CNC y robótica.

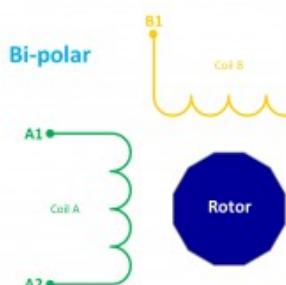


Figura F.3: Motor HY200 1713 0033 BX04 (izquierda) y bobinado bi-polar (derecha)

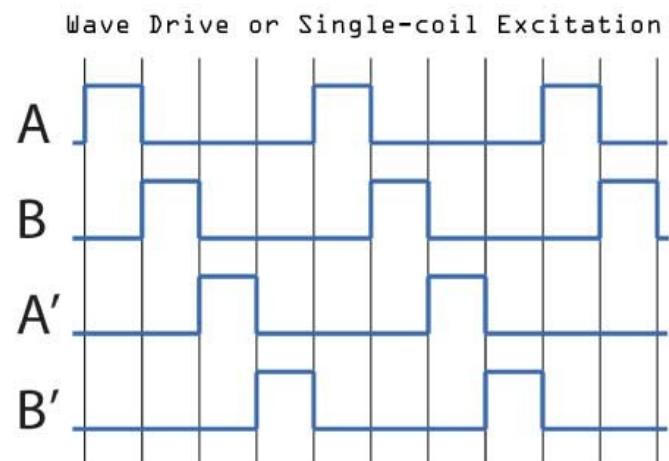
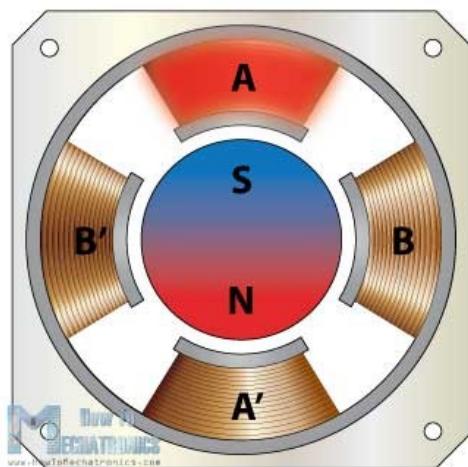
En nuestro caso disponemos de un motor HY200 1713 0033 BX04 de Offanengo 19.4 oz-in indica el par de retención. Esta medida de fuerza significa que el motor estacionario (i.e. sin girar) puede sostener un peso de 19 onzas en una polea de 1 pulgada de radio. Cuando el motor gira su par se reduce. De hecho, cuanto más rápida sea la rotación del motor menor será el par. A la hora de seleccionar el motor se debe prestar especial atención a la curva par-velocidad especificada en la hoja de datos del motor paso a paso (ver anexo).

El motor HY200 1713 0033 BX04 de Offanengo es un motor bipolar de 4 cables y 2 fases (i.e. 2 grupos de bobinas). Efectivamente, un motor paso a paso tiene un determinado número de bobinas conectadas en grupos llamados "fases". Todas las bobinas de una fase se energizan (i.e. activan) juntas. Y es el controlador quien controla su activación para cambiar los polos magnéticos y así hacer girar el rotor.

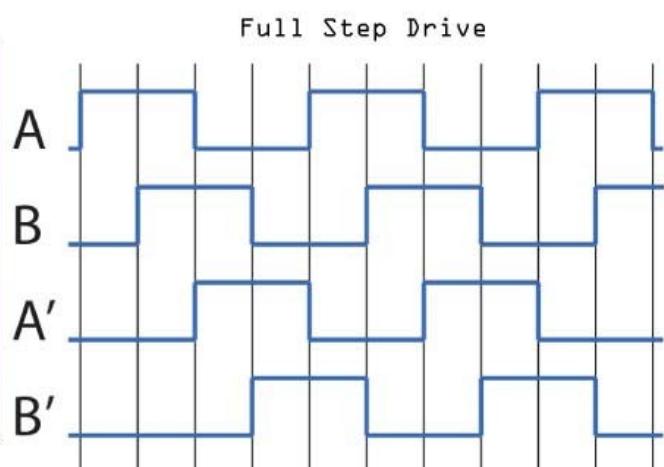
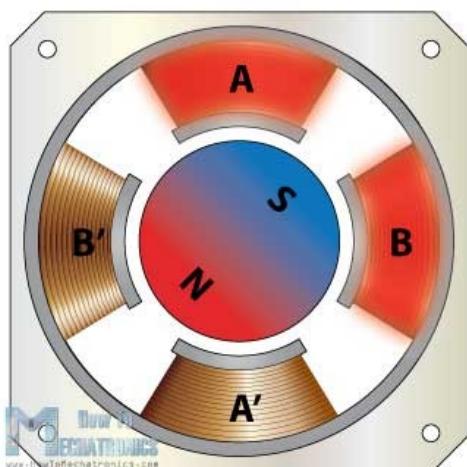
7.2. Principio de funciomamiento

El motor paso a paso consiste en un rotor que generalmente es un imán permanente y que está rodeado por los devanados del estator. A medida que dejamos que una corriente fluya a través de los devanados en un orden particular, estos se magnetizarán y crearán polos electromagnéticos que causarán el giro del motor.

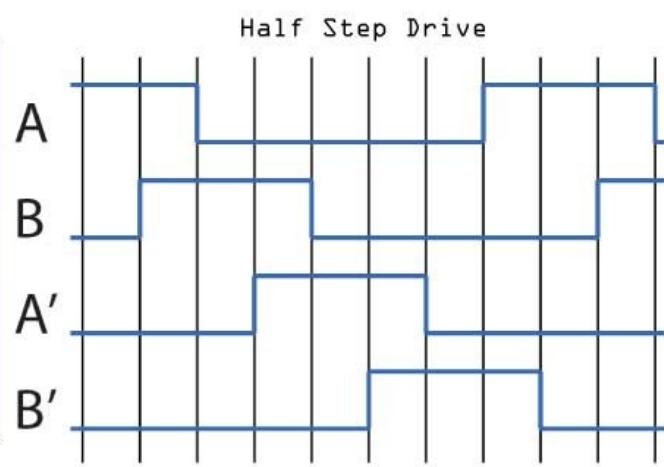
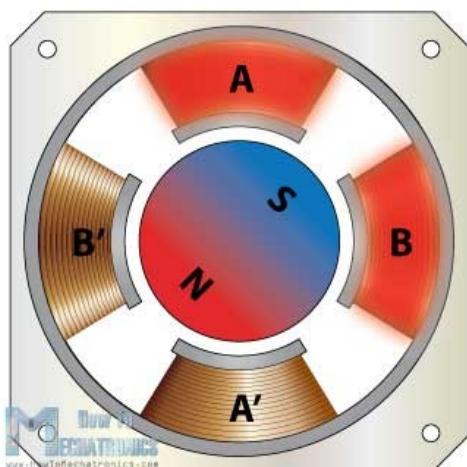
Hay varias formas diferentes de controlar un motor paso a paso. El primero es el *wave drive* o excitación de bobina simple. En este modo activamos solo una bobina a la vez, lo que significa que para el caso de un motor con 4 bobinas el rotor completará un giro de completo en 4 pasos.



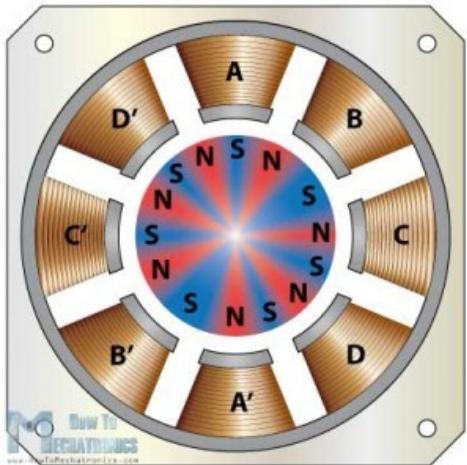
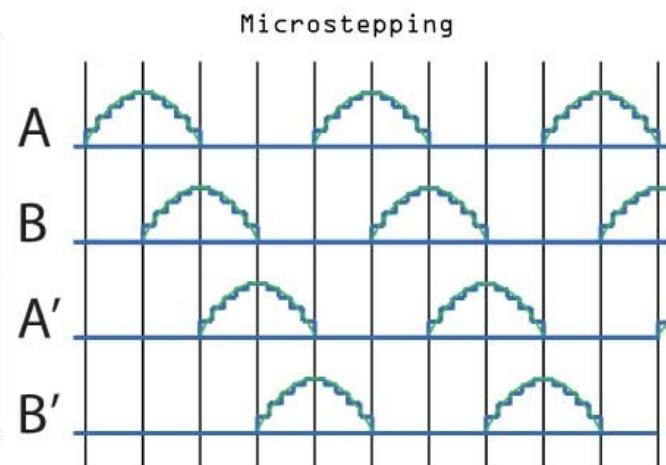
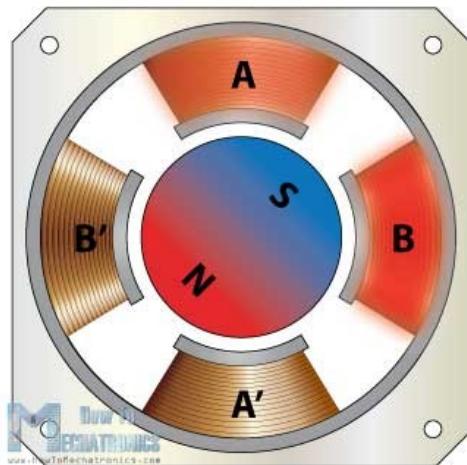
El siguiente modo es el *full step* o paso completo. Este modo proporciona aproximadamente un 30% a un 40% más de par porque siempre tenemos 2 bobinas activas. Sin embargo, requiere el doble de potencia del controlador y no mejora la resolución, ya que nuevamente el rotor completará un ciclo completo en 4 pasos.



Para aumentar la resolución utilizamos el modo *half step drive*, que es en realidad una combinación de los dos modos anteriores. Aquí tenemos una bobina activa seguida de 2 bobinas activas y luego nuevamente una única bobina activa seguida de 2 bobinas activas y así sucesivamente. Así que con este modo obtenemos el doble de resolución con la misma construcción ya que ahora el rotor hará un ciclo completo en 8 pasos. Sin embargo, el par proporcionado es aproximadamente un 15% menor que con dos fases activas en todo el paso.



Sin embargo, el método más común para controlar los motores paso a paso es el *microstepping*. En este modo proporcionamos corriente controlada variable a las bobinas en forma de onda sinusoidal. Esto genera un movimiento suave del rotor, disminuirá la tensión de las piezas y aumentará la precisión del motor.

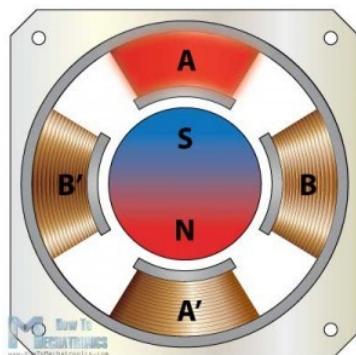


Otra forma de aumentar la resolución del motor paso a paso es aumentar los números de los polos del rotor y los números del polo del estator.

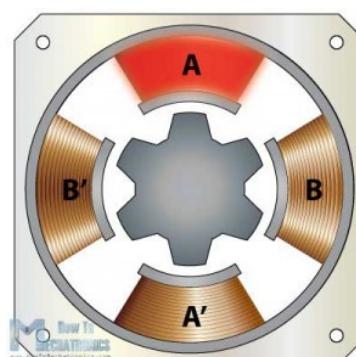
7.3. Tipos de motores

Por construcción, hay 3 tipos diferentes de motores paso a paso: de imán permanente, de reluctancia variable y sincrónico híbrido.

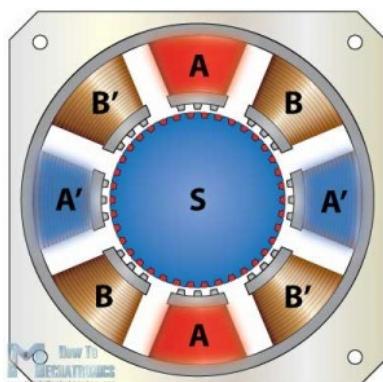
El motor paso a paso de imán permanente tiene un rotor de imán permanente que es accionado por los devanados del estator. Estos crean polos de polaridad opuesta en comparación con los polos del rotor impulsándolo.



El motor paso a paso de imán permanente tiene un rotor de imán permanente que es accionado por los devanados del estator. Estos crean polos de polaridad opuesta en comparación con los polos del rotor impulsándolo.



El motor paso a paso de **reluctancia variable** utiliza un rotor con dientes desplazados del estator. Así, cuando activamos los devanados en un determinado orden, el rotor se mueve en consecuencia para minimizar la distancia entre el estator y los dientes del rotor.



El motor síncrono híbrido es una combinación de los dos anteriores. Tanto el rotor como el estator son dentados. Se trata de un rotor de imán permanente, con dos secciones opuestas en polaridad y dientes desplazados.

7.4. Controlador DRV-8825

Esta polaridad cambiante explicada en el apartado anterior requiere un circuito complejo. Por suerte, existen soluciones de bajo costo, como el controlador de pasos bipolar DRV-8825:

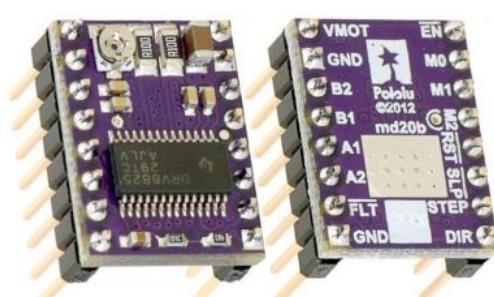


Figura F.3: Controlador DRV-8825

Esta placa de bajo costo puede controlar un solo motor paso a paso bipolar de hasta 2,2 amperios y 45 voltios. La salida de corriente máxima es ajustable, lo que permite utilizar voltajes por encima del voltaje nominal del motor paso a paso para lograr tasas de paso más altas (*chopper driver*). Además, tiene protección a sobrecalentamiento, sobrecorriente, caída de voltaje y cortocircuito. En esta placa existen 6 resoluciones de pasos, desde *full step* hasta 1/32, y tiene un regulador de 3.3 V incorporado que facilita la interfaz con la Raspberry Pi.

7.5. Conexionado

Conectar el DRV8825 es muy simple. El cableado mínimo requiere solo 2 pines GPIO. En realidad 1 si no importa la dirección. En nuestro caso, el pin de paso está conectado a [GPIO 21](#) y el pin DIR (o pin de dirección) está conectado a [GPIO 20](#). El pin de tierra está conectado a una tierra en el Pi. No se requiere suministro de voltaje lógico porque el DRV8255 tiene un regulador de voltaje de 3.3V incorporado. Sin embargo, los pines RST (reset) y SLP (sleep) debemos conectarlos a un pin de 3.3 V en la Pi porque sus estados bajos predeterminados son reiniciar y dormir. El valor por defecto del pin EN (enable) es el de habilitado, por lo que se puede dejar desconectado. El pin opcional FLT (fault) se utiliza para detectar eventos de sobrecorriente o apagado térmico. En nuestro caso lo dejaremos desconectado.

Para controlar el modo de paso programáticamente, conectaremos M0, M1 y M2 a [GPIO 14](#), [GPIO 15](#) y [GPIO 18](#) respectivamente. Podemos dejarlos desconectados si solo pretendemos usar el modo *full step*.

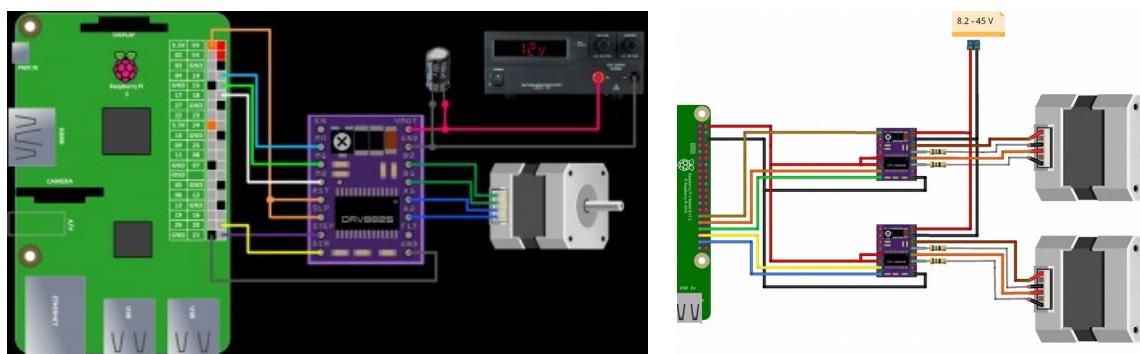


Figura F.3: Conexionado

El DRV8825 requiere una fuente de alimentación de entre 8.2 y 45 voltios. Usaremos una fuente de alimentación externa porque la Raspberry Pi no puede suministrar suficiente voltaje/corriente. Hemos de asegurarnos de que la fuente de alimentación esté apagada antes de conectarla. El pin VMOT (voltaje del motor) estará conectado al terminal positivo de la fuente de alimentación y el pin de tierra estará conectado al terminal negativo.

Importante:

- No hay protección contra tensión inversa, así que es importante revisar dos veces las conexiones.
- El cable positivo de la fuente de alimentación del motor no puede tocar nada que no sea el pin VMOT. Se trata de por lo menos 8.2 voltios lo que destruiría la Raspberry Pi.

El DRV8825 es susceptible a picos de voltaje destructivos. Para proteger la placa, debemos mantener cortos los cables de alimentación del motor. Además, es importante colocar un condensador electrolítico de mínimo 47 μF a través de la fuente de alimentación del motor lo más cerca posible del DRV8825. Elegiremos un condensador con el doble de la tensión nominal. Es decir, si tenemos una fuente de alimentación de 12 V utilizaremos un condensador diseñado para trabajar a una tensión nominal de al menos 24 V.

Una bobina del motor estará conectada a A1 y A2 y la otra bobina a B1 y B2. Hemos de asegurarnos de establecer la corriente máxima antes de conectar el motor y de que la fuente de alimentación del motor está apagada. Conectar o desconectar un motor paso a paso mientras el controlador está encendido puede dañar o destruir el controlador. Hemos de asegurarnos también de establecer la corriente máxima antes de conectar el motor.

La corriente máxima del controlador DRV-8825 es igual al doble de la referencia V_{ref} o de voltaje.

$$I_{limit} = V_{ref} \times 2$$

Se puede medir V_{ref} utilizando V_{ref} o la V_{ref} del potenciómetro. En primer lugar, conectaremos la sonda negativa del multímetro a la tierra DRV-8825. En segundo lugar usaremos un clip de cocodrilo para conectar la sonda positiva del medidor a un destornillador de metal pequeño. Ahora ya podremos leer V_{ref} mientras hacemos los ajustes necesarios.

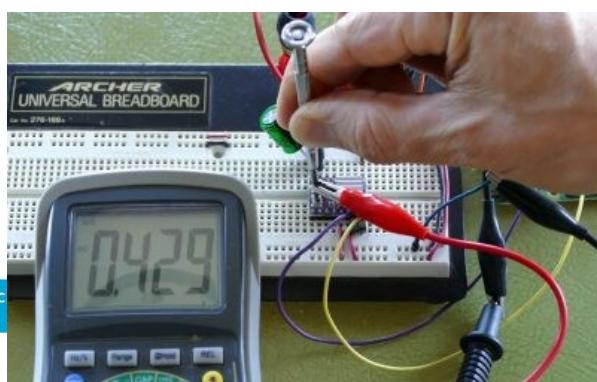


Figura F.3: Limitación de la corriente máxima

Al establecer V_{ref} a **0.429V**, de acuerdo con la fórmula anterior, se establece la corriente máxima del motor a **0.858A**. Hemos de consultar la hoja de datos de nuestro motor paso a paso para determinar la máxima corriente pico.

La hoja de datos del controlador DRV8825 indica que, en el modo *full step*, la corriente está limitada al 71% de la corriente máxima establecida. Por lo tanto, se puede configurar la corriente máxima más alta que el pico para conseguir un mayor par de torsión siempre y cuando nos mantengamos en el modo *full step*.

Mode 2	Mode 1	Mode 0	Step Mode
0	0	0	full step (2-phase excitation) with 71% current
0	0	1	1/2 step (1-2 phase excitation)
0	1	0	1/4 step (W1-2 phase excitation)
0	1	1	8 microsteps/step
1	0	0	16 microsteps/step
1	0	1	32 microsteps/step

Siempre es una buena idea controlar la temperatura del motor paso a paso y de la placa del conductor. Aplicaremos la regla de los 5 segundos: *Si puedo tocar el motor y la placa del conductor durante al menos 5 segundos sin quemarme los dedos, entonces la corriente probablemente esté bien*. Tendremos en cuenta que la corriente también puede estar limitada por la resistencia de la bobina del motor debido a la ley de Ohm ($V = R \times I$). Así, si la resistencia de la bobina es de 23.9Ω y la fuente de alimentación es de 9.6 V, la corriente se limitará a 0.4 A ($9.6 \text{ V} = 0.4 \text{ A} \times 23.9 \Omega$).

Una vez establecida la corriente máxima se puede conectar el motor paso a paso. Los cables del motor paso a paso a menudo no están etiquetados. Un método fácil para identificar qué cables van con qué bobina es usar un LED. Colocaremos un LED en puenteando una pareja cualquiera de cables del motor (la dirección no importa) y haremos girar el motor con la mano. Si el LED parpadea, los 2 cables son una bobina. También podemos usar un multímetro para verificar la continuidad de la bobina si no disponemos de un LED o si el motor paso a paso es demasiado pequeño para alimentar el LED.

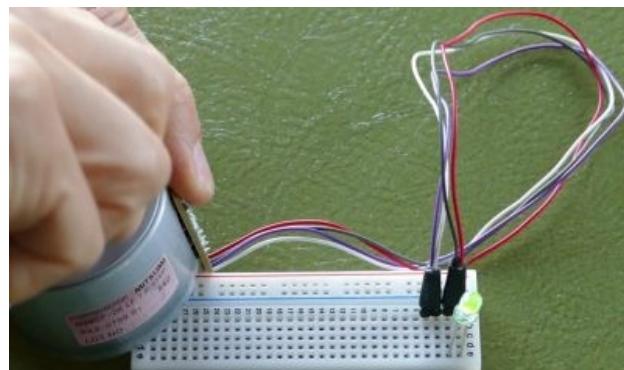


Figura F.3: Identificación de cables

7.6. Código

Antes de escribir cualquier software, nos hemos de asegurar de que nuestra Raspberry Pi está actualizada:

```
sudo apt-get update && sudo apt-get upgrade
```

Lo recomendable es comenzar con una Raspberry Pi "limpia" con la última versión de Raspbian instalada para asegurarnos de que tiene todo el software necesario.

El primer ejemplo de Python hace girar un motor de 48 steps/rev. (pasos por revolución) una vez en el sentido de las agujas del reloj y luego retrocede en el sentido contrario a las agujas del reloj utilizando la biblioteca RPi.GPIO. Los pines DIR y STEP están configurados como salidas. El pin DIR se establece *high* para girar en el sentido de las agujas del reloj. Luego, un bucle *for* cuenta hasta 48. Cada ciclo alterna el pin STEP *high* durante 0.0208 segundos y *low* durante 0.0208 segundos. El pin DIR se establece *low* para girar en sentido contrario a las agujas del reloj y se repite el bucle *for*.

```

from time import sleep
import RPi.GPIO as GPIO

DIR = 20 # Direction GPIO Pin
STEP = 21 # Step GPIO Pin
CW = 1 # Clockwise Rotation
CCW = 0 # Counterclockwise Rotation
SPR = 48 # Steps per Revolution (360 / 7.5)

GPIO.setmode(GPIO.BCM)
GPIO.setup(DIR, GPIO.OUT)
GPIO.setup(STEP, GPIO.OUT)
GPIO.output(DIR, CW)

step_count = SPR
delay = .0208

for x in range(step_count):
    GPIO.output(STEP, GPIO.HIGH)
    sleep(delay)
    GPIO.output(STEP, GPIO.LOW)
    sleep(delay)

sleep(.5)
GPIO.output(DIR, CCW)
for x in range(step_count):
    GPIO.output(STEP, GPIO.HIGH)
    sleep(delay)
    GPIO.output(STEP, GPIO.LOW)
    sleep(delay)

GPIO.cleanup()

```

Este código puede resultar en vibraciones del motor y movimientos bruscos, especialmente a bajas velocidades. Una forma de atenuar este fenómeno es con microstepping. El siguiente fragmento de código se agrega al código anterior. Los pines GPIO de modo se establecen como salidas. Un dict mantiene los valores apropiados para cada formato de pasos. GPIO.output establece el modo en 1/32. Tendremos en cuenta que el contador de pasos se multiplica por 32 porque ahora cada rotación requiere 32 veces más ciclos, lo que resulta en un movimiento más fluido. El retraso se divide por 32 para compensar los pasos adicionales.

```

MODE = (14, 15, 18) # Microstep Resolution GPIO Pins
GPIO.setup(MODE, GPIO.OUT)
RESOLUTION = {'Full': (0, 0, 0),
              'Half': (1, 0, 0),
              '1/4': (0, 1, 0),
              '1/8': (1, 1, 0),
              '1/16': (0, 0, 1),
              '1/32': (1, 0, 1)}
GPIO.output(MODE, RESOLUTION['1/32'])

step_count = SPR * 32
delay = .0208 / 32

```

El microstepping tiene algunos inconvenientes. A menudo hay una pérdida sustancial de paso que puede llevar a una pérdida de precisión.

Un problema con el último programa es que se basa en el método `sleep` de Python para la sincronización, lo que no es muy fiable. En el siguiente ejemplo, usaremos la biblioteca PiGPIO que proporciona sincronización PWM basada en hardware. Antes de usarla, iniciaremos el demonio PiGPIO:

```
sudo pigpiod
```

La primera parte del siguiente código es similar al primer ejemplo. La sintaxis se modifica para la biblioteca PiGPIO. El método `set_PWM_dutycycle` se usa para configurar el ciclo de trabajo del PWM. Esto es el porcentaje de pulso que es *high* y *low*. El valor 128 lo establece en 50%. Por lo tanto, las partes de encendido y apagado del ciclo son iguales. El método `set_PWM_frequency` establece el número de pulsos por segundo. El valor 500 establece la frecuencia a 500 Hz. Un bucle `while` infinito verifica el `switch` y alterna la dirección de manera apropiada.

```
from time import sleep
import pigpio

DIR = 20    # Direction GPIO Pin
STEP = 21   # Step GPIO Pin
SWITCH = 16  # GPIO pin of switch

# Connect to pigpiod daemon
pi = pigpio.pi()

# Set up pins as an output
pi.set_mode(DIR, pigpio.OUTPUT)
pi.set_mode(STEP, pigpio.OUTPUT)

# Set up input switch
pi.set_mode(SWITCH, pigpio.INPUT)
pi.set_pull_up_down(SWITCH, pigpio.PUD_UP)

MODE = (14, 15, 18)  # Microstep Resolution GPIO Pins
RESOLUTION = {'Full': (0, 0, 0),
              'Half': (1, 0, 0),
              '1/4': (0, 1, 0),
              '1/8': (1, 1, 0),
              '1/16': (0, 0, 1),
              '1/32': (1, 0, 1)}
for i in range(3):
    pi.write(MODE[i], RESOLUTION['Full'][i])

# Set duty cycle and frequency
pi.set_PWM_dutycycle(STEP, 128) # PWM 1/2 On 1/2 Off
pi.set_PWM_frequency(STEP, 500) # 500 pulses per second

try:
    while True:
        pi.write(DIR, pi.read(SWITCH)) # Set direction
        sleep(.1)

except KeyboardInterrupt:
    print ("\nCtrl-C pressed. Stopping PIGPIO and exiting...")
finally:
    pi.set_PWM_dutycycle(STEP, 0) # PWM off
    pi.stop()
```

El método PiGPIO *set_PWM_frequency* está limitado a valores de frecuencia específicos por frecuencia de muestreo tal y como se especifica en la siguiente tabla:

Sample Rate	Hertz										
1: 40000 20000 10000 8000 5000 4000 2500 2000 2000 1600 1250 1000 800 500 400 250 200 100 100 50											
2: 20000 10000 5000 4000 2500 2000 1250 1000 800 625 500 400 250 200 125 100 50 25											
4: 10000 5000 2500 2000 1250 1000 625 500 400 313 250 200 125 100 63 50 25 13											
5: 8000 4000 2000 1600 1000 800 500 400 320 250 200 160 100 80 50 40 20 10											
8: 5000 2500 1250 1000 625 500 313 250 200 156 125 100 63 50 31 25 13 6											
10: 4000 2000 1000 800 500 400 250 200 160 125 100 80 50 40 25 20 10 5											

Default sample rate of 5 is set when PiGPIO daemon is started. -s to specify a different rate

Podemos cambiar la frecuencia de muestreo utilizando la opción de configuración *-s* cuando iniciamos el demonio PiGPIO. La frecuencia de muestreo predeterminada es 5, que tiene valores comprendidos entre 8000 y 10 hertz (ver la fila verde arriba). Para el ejemplo del código, hemos elegido 500 Hz. Si hubieramos especificado 600 Hz, se habría reducido automáticamente a 500 Hz. Si hubieramos especificado 700 Hz, se habría aumentado automáticamente a 800 Hz.

Si necesitamos usar una frecuencia que no se encuentra en la tabla, PiGPIO también permite usar el PWM de hardware integrado de Pi, que solo está disponible en el GPIO 18. En ese caso, en lugar de *set_pwm* se usaría *hardware_PWM*, que toma parámetros para GPIO, frecuencia y ciclo de trabajo.

pi.hardware_PWM(18, frequency, duty_cycle)

Un inconveniente del segundo ejemplo de Python es que solo tenemos control sobre la velocidad y la dirección. No permite especificar la cantidad de pasos a realizar. Además, cambiar bruscamente la dirección de un motor o acelerar demasiado rápido puede provocar pérdida de pasos. Lo recomendable es acelerar y desacelerar suavemente, en forma de rampa. Esto se puede abordar con las formas de onda PiGPIO. Actualmente hay un error en la biblioteca PiGPIO relacionado con la sincronización de la forma de onda. Una solución es iniciar el demonio PiGPIO con la opción *-t* (reloj periférico) establecida en cero para PWM en lugar de la PCM predeterminada.

sudo pigpiod -t 0

Esto corrige la sincronización de la forma de onda. Hay que tener en cuenta que, aunque corrige la sincronización de la forma de onda, interfiere con la temporización PWM en el ejemplo anterior e impide el uso de *hardware_PWM()*.

El siguiente método *generation_ramp* toma una rampa variable que es una lista de parejas

de frecuencias y pasos. El método genera una cadena de formas de onda correspondiente a los valores pasados. Se han introducido algunas pequeñas modificaciones al método, como la codificación de GPIO y el borrado de las ondas existentes al inicio del método, en lugar de bloquear el hilo al final para limpiarlo.

```
def generate_ramp(ramp):
    """Generate ramp wave forms.
    ramp: List of [Frequency, Steps]
    """
    pi.wave_clear()      # clear existing waves
    length = len(ramp)   # number of ramp levels
    wid = [-1] * length

    # Generate a wave per ramp level
    for i in range(length):
        frequency = ramp[i][0]
        micros = int(500000 / frequency)
        wf = []
        wf.append(pigpio.pulse(1 << STEP, 0, micros)) # pulse on
        wf.append(pigpio.pulse(0, 1 << STEP, micros)) # pulse off
        pi.wave_add_generic(wf)
        wid[i] = pi.wave_create()

    # Generate a chain of waves
    chain = []
    for i in range(length):
        steps = ramp[i][1]
        x = steps & 255
        y = steps >> 8
        chain += [255, 0, wid[i], 255, 1, x, y]

    pi.wave_chain(chain) # Transmit chain.
```

El siguiente fragmento de código llama a `genera_ramp` con 6 niveles de rampa para acelerar lentamente de 320 Hz a 2000 Hz. Los pasos por nivel de rampa son exagerados con el propósito de demostración. Si una determinada frecuencia causa resonancia será necesario aumentar la aceleración para pasar rápidamente por dicha frecuencia.

```
# Ramp up
generate_ramp([[320, 200],
               [500, 400],
               [800, 500],
               [1000, 700],
               [1600, 900],
               [2000, 10000]])
```


Conclusiones

Les conclusions han de ser un reflex clar i ordenat de les deduccions fetes com a conseqüència del treball descrit al llarg del nucli del document. S'hi poden incloure dades quantitatives però no s'haurien de donar detalls de cap argument o resultat.

Les recomanacions són manifestacions concises d'alguna acció futura que sembli necessària, com a resultat directe de les conclusions o d'alguna experiència feta en el curs del treball objecte del projecte. No són necessàries, tret que estiguin completament justificades pel treball descrit.

Aquest capítol i els següents no formen part del nucli del document i no necessiten portar numeració de capítol.

Anexo

sensors.py: Script de Python para mostrar valores de sensores recibidos en la Raspberry

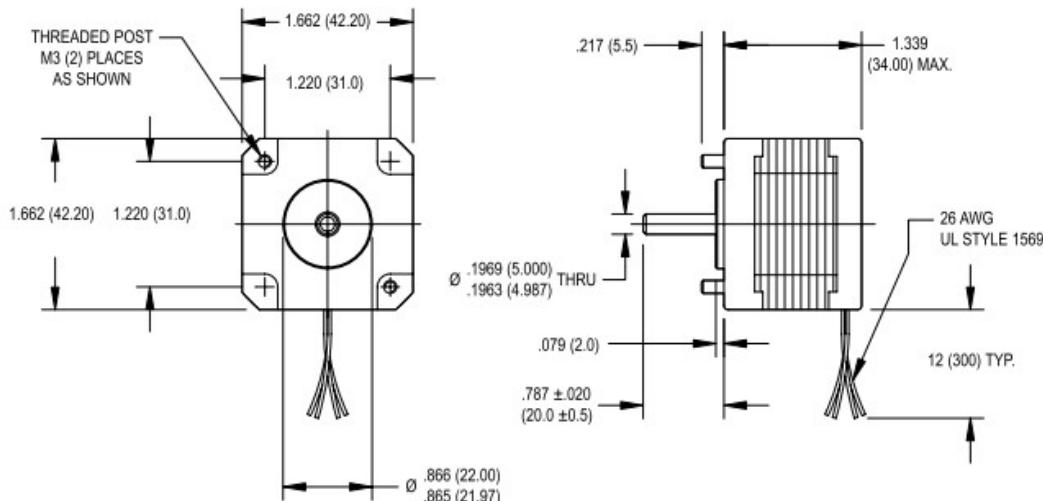
```
import socket
from struct import *

# Prepare the UDP connection
UDP_IP = "192.168.42.100"
print "Receiver IP: ", UDP_IP
UDP_PORT = 50000
print "Port: ", UDP_PORT
sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

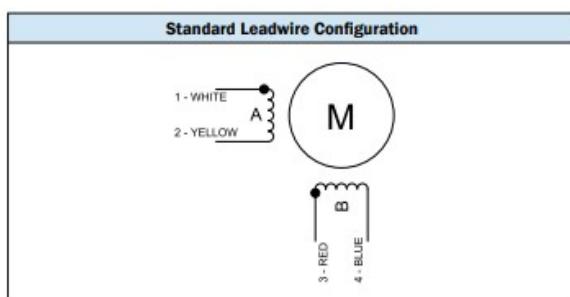
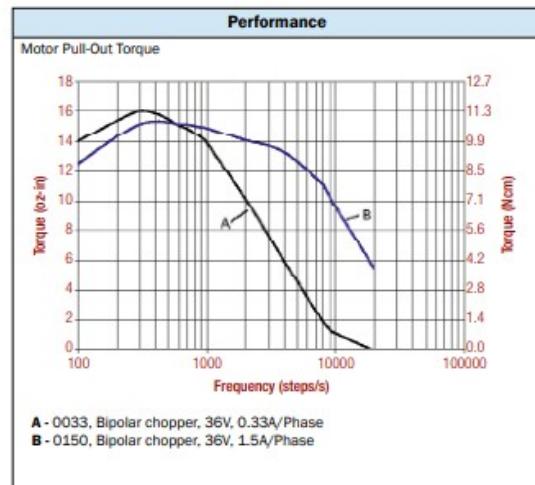
# Continuously read from the UDP socket
while True:
    data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
    # The next line just chops a lot of floats from the received data chunk
    # Just activate and deactivate sensors in the Sensor UDP app to locate
    # the values you want
    print "received message: ",\
        "%1.4f" %unpack_from ('!f', data, 0),\
        "%1.4f" %unpack_from ('!f', data, 4),\
        "%1.4f" %unpack_from ('!f', data, 8),\
        "%1.4f" %unpack_from ('!f', data, 12),\
        "%1.4f" %unpack_from ('!f', data, 16),\
        "%1.4f" %unpack_from ('!f', data, 20),\
        "%1.4f" %unpack_from ('!f', data, 24),\
        "%1.4f" %unpack_from ('!f', data, 28),\
        "%1.4f" %unpack_from ('!f', data, 32),\
        "%1.4f" %unpack_from ('!f', data, 36),\
        "%1.4f" %unpack_from ('!f', data, 40),\
        "%1.4f" %unpack_from ('!f', data, 44),\
        "%1.4f" %unpack_from ('!f', data, 48),\
```

```
"%1.4f" %unpack_from ('!f', data, 52),\  
"%1.4f" %unpack_from ('!f', data, 56),\  
"%1.4f" %unpack_from ('!f', data, 60),\  
"%1.4f" %unpack_from ('!f', data, 64),\  
"%1.4f" %unpack_from ('!f', data, 68),\  
"%1.4f" %unpack_from ('!f', data, 72),\  
"%1.4f" %unpack_from ('!f', data, 76),\  
"%1.4f" %unpack_from ('!f', data, 80),\  
"%1.4f" %unpack_from ('!f', data, 84),\  
"%1.4f" %unpack_from ('!f', data, 88),\  
"%1.4f" %unpack_from ('!f', data, 92)
```



MOTOR PASO A PASO OFFANENO HY200 1713 0033 BX04**SIZE 17 STEPPER MOTOR DATA**

Specification	Units	HY 200 1713	
		0033	0150
Rated Phase Current	A	0.33	1.50
Phase Resistance	Ω	23.9	1.0
Phase Inductance	mH	28.9	1.2
Holding Torque Unipolar	oz-in Ncm	— —	— —
Holding Torque Bipolar	oz-in Ncm	19.4 13.7	18.4 13.0
Detent Torque	oz-in Ncm	2.4 1.7	2.4 1.7
Rotor Inertia	oz-in-s ² ×10 ⁻⁴ g-cm ²	2.5 18	2.5 18
Motor Weight (Mass)	lb kg	0.4 0.2	0.4 0.2
Maximum Voltage	V	40	40
Std. No. of Leads	—	4	4



Standard Features
• Step angle: 1.8°
• Step angle accuracy: 5%
• Insulation class: B (130°C)
• NEMA 17 mounting configuration
• Neodymium magnets
• Additional windings and customization options available
Complementary Products
• Gearboxes
• Encoders

Bibliografía

Referencias bibliográficas

Exemples de llibres, articles, catàlegs, material informàtic i material obtingut a la xarxa:

- [1] ASHRAE, American Society of Heating, Refrigerating and Air-Conditioning Engineers, *Fundamentals Volume (S.I. edition.)*. Atlanta: 2001, p. 104-121
- [2] BOSSER, J. *Vademécum de mecánica de fluidos y maquinas hidráulicas*, Barcelona: ETSEIB - CPDA . 1985.
- [3] CARRIER AIR CONDITIONING CO.: *Manual de Aire Acondicionado*. Barcelona: Marcombo, 1974, p. 57-64
- [4] GUTOWSKI, T.G., DYM, C.L. *Propagation of ground vibration: a review. Journal of Sound and Vibration*. Vol. 49(2)*, 1976, p. 179-193. *Revista Volum (Número).
- [5] UNIVERSITAT POLITÈCNICA DE CATALUNYA. SERVEI DE LLENGÜES I TERMINOLOGIA *. *Guia lingüística práctica 2*. Barcelona, Servei de Publicacions de la UPC , 1996. *Institució. Departament(s).
- [6] UNIVERSITAT POLITÈCNICA DE CATALUNYA. SERVEI DE LLENGÜES I TERMINOLOGIA. *Guia lingüística práctica 2*. Barcelona, 1998.[<http://www.upc.es/slct/cat/publicacions/gl2/gl2.htm>, 21 de setembre de 2000]*. *[URL, data de consulta].
- [7] FUNDACIÓ SERVEIS DE CULTURA POPULAR. *Introducció a la geometria descriptiva, II: la perspectiva cònica*. Barcelona, 1993. [Vídeo]*. *[Suport, tipus de material].

Bibliografía complementaria

- [8] <https://es.slideshare.net/IvanCoal/latitud-y-longitud-azimut-y-altitud>
- [9] https://wiki2.org/es/Conversi%C3%B3n_de_coordenadas_ecuatoriales_a_coordenadas_ecl%C3%ADpticas
- [10] <http://delegacion.topografia.upm.es/wp-content/uploads/2016/03/Astronom%C3%ADA.pdf>

- [11] <https://www.ign.es/web/ign/portal/gmt-declinacion-magnetica>
- [12] <https://www.hackster.io/jm4rc0/android-sensors-as-imu-for-raspberry-pi-2-via-usb-and-udp-452bad>
- [13] <https://www.rototron.info/raspberry-pi-stepper-motor-tutorial/>
- [14] <https://www.hackster.io/rs2vn/drv8825-stepper-motor-driver-on-rpi-3-win10-iot-75e011>
- [15] <https://howtomechatronics.com/how-it-works/electrical-engineering/stepper-motor/>
- [16] <https://masquesig.com/2015/03/04/qpython-programa-con-tu-terminal-android-acceso-al-gps-y-mucho-mas/>