Escuela Politécnica Superior de Linares

Grado en Ingeniería Eléctrica

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CONTROL DE MOTORES DE CORRIENTE CONTINUA MEDIANTE TARJETAS DE ADQUISICIÓN DE DATOS DE BAJO COSTE

Alumno: Carlos Domínguez Parra

Tutor: Manuel Valverde Ibáñez

Dpto: Ingeniería Eléctrica

Septiembre, 2017

Escuela Politécnica Superior de Linares

Grado en Ingeniería Eléctrica

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CONTROL DE MOTORES DE CORRIENTE CONTINUA MEDIANTE TARJETAS DE ADQUISICIÓN DE DATOS DE BAJO COSTE

Alumno

Carlos Domínguez Parra

Tutor

Manuel Valverde Ibáñez

NOMBRE VALVERDE IBAÑEZ MANUEL - NIF 26230208L

Firmado digitalmente por NOMBRE VALVERDE IBAÑEZ MANUEL - NIF 26230208L
Nombre de reconocimiento (DN): c=ES, o=FNMT, ou=FNMT Clase 2 CA, ou=501072765, cn=NOMBRE VALVERDE IBAÑEZ MANUEL - NIF 26230208L
Fecha: 2017.09.02 19:23:02 +02'00'

Septiembre, 2017

Índice

1. Introducción

En este trabajo vamos a realizar el estudio de cómo funcionan las máquinas de corriente continua, en especial los motores de corriente continua como motores con escobillas, motores paso a paso y servomotores, además de realizar diferentes ensayos para la utilización de estos controlados a través de tarjetas de adquisición de datos de bajo coste, como es el caso de las placas arduino Uno, arduino Mega y Raspberry Pi 2, con otras complementarias a estas y específicas para el control de los motores de corriente continua como la placa Shield Motor para el arduino, y la placa Gertbot para la Raspberry. Se utilizarán todos estos motores con las diferentes placas y con diferentes programas, para manipular su control.

El trabajo constará, de información a acerca de todos estos componentes, de instalaciones, ensayos y análisis e identificación de errores que se puedan observar durante el trabajo.

Y finalmente daremos unas conclusiones que nos informarán de la mejor opción entre las ensayadas en el trabajo, considerando las partes positivas y negativas de cada motor, cada programa y de cada método de control.

2. Análisis teórico de la máquina de corriente continua.

2.1 Voltaje inducido en una espira giratoria.

Una máquina, tanto de corriente continua como de corriente alterna, estará constituida principalmente por los devanados principales que tiene un estator, que da soporte a la máquina, los cuales pueden ser imanes permanentes o devanados con hilo de cobre sobre un núcleo de hierro, y un rotor, que es la parte giratoria de la máquina que está formada normalmente de forma cilíndrica y también con un devanado introducido en unas ranuras labradas en un núcleo de hierro, Entre ambas partes se proporciona un espacio que se le llamara entrehierro de anchura constante. En las terminaciones del devanado, se conectará al exterior con unas escobillas, que es la parte más sensible de la máquina, ya que está en continuo rozamiento con los extremos del bobinado y esto hace que se deteriore con facilidad, sin embargo si la presión de esta con el rotor es muy pequeña, el rotor tiende a apoyarse ligeramente en el colector y ocurre una gran cantidad de chisporroteo.

Suponemos un devanado de una espira, que tiene una sola vuelta alrededor de su rotor mientras que se le aplica un campo magnético y una velocidad al rotor igual que aparece en la figura 201

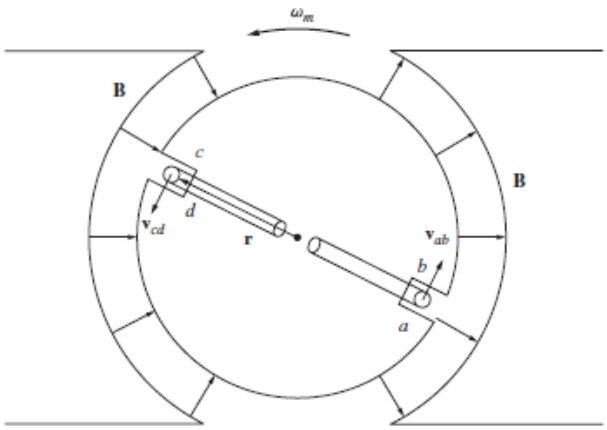


Figura 102: *Rotor de una espira en su rotor.*

Se va a producir en los terminales de esta espira una tensión generada, debido a la siguiente relación.

$$e_{ind} = (v \; x \; B) \cdot l$$

Podemos determinar esta tensión con más facilidad separando el devanado de la figura 101 en cuatro partes. Aquella que es perpendicular al plano de la figura ab y cd, y los que son paralelos a la figura bc y da.

En el segmento ab tiene una velocidad tangencial a la trayectoria de rotación, y tiene un campo magnético tangencial a esta, por lo que su valor de tensión es el producto de la velocidad, campo magnético y longitud del mismo. El lado cd realizaría las mismas condiciones que el lado explicado anteriormente, pero tendría un sentido contrario. En los lados bc y da, funcionaria de la misma manera que los lados anteriores, pero puesto que el producto v x B es perpendicular a la longitud, la tensión se hará 0 en toda la rama. Cuando la espira gire 180 grado sobre sí misma, los cálculos de tensión serán exactamente los mismos, salvo con la objeción de tener el lado ab con el sentido de cd, y al contrario. Por lo tanto conseguiremos una tensión que quiere parecerse a una onda sinusoidal parecida a la de la figura 202 con el valor máximo de $e = 2vBl$ que es la suma del lado ab y cd.

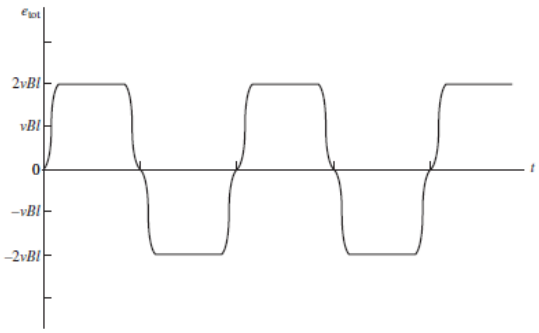


Figura 202: *Valor de la tensión en los extremos de la espira del rotor.*

En la figura 203 se puede apreciar de mejor manera las fuerzas que tiene el rotor según su posición. Teniendo en cuenta que el magnetismo del rotor circula de izquierda

a derecha, y que la bobina tiene un sentido de circulación determinada, se producirá una fuerza, que hará que el rotor gira hasta la posición de la imagen b, cuando el rotor sobrepasa esa posición debido al impulso, se produce un cambio de sentido de la corriente en la bobina, que es lo que ocurre en la imagen c, y hace que el sentido de la fuerza también cambie, y consiga un movimiento continuo y constante.

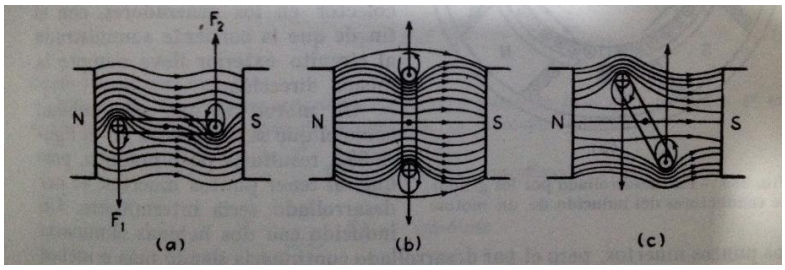


Figura 203: *Sentido de la fuerza en la bobina según la posición del rotor sobre el estator.*

Como hemos observado, en este proceso conseguimos unos valores que alternan entre una tensión positiva y negativa que es lo que se caracteriza como una corriente alterna, pero sin embargo, si queremos obtener una corriente continua en los extremos de nuestro devanado, lo que deberemos colocar son dos segmentos conductores semicirculares a los extremos de los devanados y que hagan contacto con las escobillas, cada una en cada segmento, y se establece que en el cambio de signo de la tensión, coincida con el cambio de la polaridad a la salida del mismo, de esta manera, cada vez que la espira cambie de sentido, los contactos también lo harán y la salida siempre sea del mismo signo. Este cambio se le llamara conmutación.

2.2 Maquina DC

Un motor de corriente continua, por el contrario del caso anterior, es capaz de general energía mecánica a partir de energía eléctrica introduciéndole una tensión en los extremos del devanado, y con el campo magnético generado por el estator, producirá una fuerza, que hace que el rotor gire sobre sí mismo.

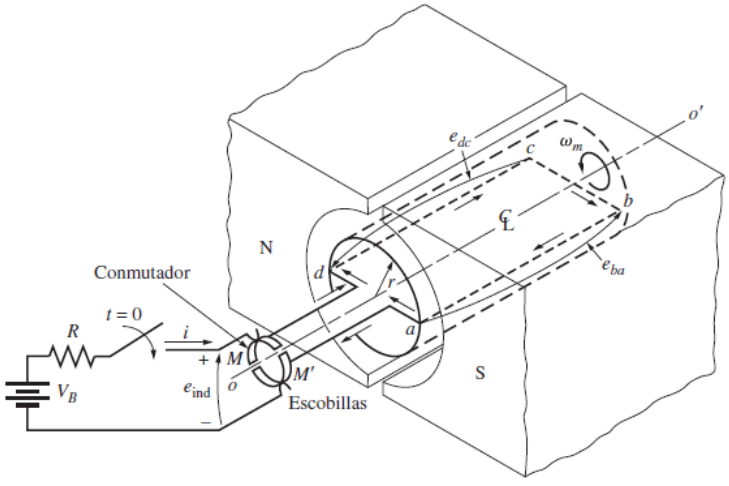


Figura 204: *Esquema básico de una máquina de corriente continua.*

El funcionamiento de este motor se debe principalmente a la ley de la fuerza.

$$F = B \cdot L \cdot I \cdot sen\ (\emptyset) = i \cdot (\ l\ x\ B\ )$$

Siendo:

- F: Fuerza en Newton.

- I: Intensidad que circula por el rotor en amperios.

-L: Longitud del conductor.

-B: Densidad del flujo magnético.

-Ø: Ángulo que forma entre I y B.

En el estator se va hacer circular una intensidad en un sentido. Esta intensidad dentro de un campo magnético va a producir una fuerza perpendicular a ambos, que hará que la bobina, y el rotor gire sobre sí mismo en una dirección, hasta que gracias a los segmentos de conmutación produzcan el intercambio de polaridad, y hará que el rotor no pare de girar sobre sí mismo.

Se debe realizar correctamente esta conmutación, debido a que justo en el cambio de polaridad se va a producir un pequeño cortocircuito. Hay diferentes formas de realizar la conexión de estas espiras a sus segmentos de conmutación, estas formas variarán según el número de cambios de corriente que existe en el rotor, el voltaje que se utiliza y el número y posición de las escobillas.

Una bobina está formada por varias vueltas del conductor aisladas entre ellas sobre unas ranuras del rotor también aisladas. La distancia entre los segmentos del conmutador que están conectados a los extremos de la bobina, se le llama paso del conmutador. Si el extremo en el que termina una bobina se conecta en la siguiente posición al que se conectó el comienzo de esta, el devanado se llamara devanado progresivo, sin embargo, si se conecta el extremo final en la posición anterior del extremo inicial, se llamara devanado regresivo.

Los devanados de un rotor también se pueden clasificar en simple, siendo único y cerrado, devanado doble con dos conjuntos de devanados completos e independientes, estando uno de ellos en todos los segmentos pares, y otro en los impares. También hay devanados triples que de la misma manera que el doble tendrá tres conjuntos de devanados completos e independientes. Todos los rotores que tengan más de un conjunto de devanados se les puede llamar también múltiples.

Las conexiones de los devanados que se realizarán en el rotor se pueden clasificar en dos principales:

- Devanados imbricados.

También llamado devanado en serie sencillo, estos tendrán tantas trayectorias de corriente a través de la maquina como polos en ella. Por este motivo hace que este tipo de conexión sea la opción ideal para máquinas de corriente alta y voltaje bajo, ya que la corriente se puede dividir entre los diferentes caminos que existen.

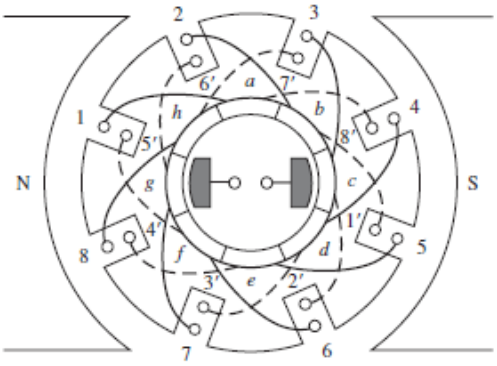


Figura 205: Rotor con devanado imbricado.

Sin embargo puede tener un problema serio, cuando la maquina lleve tiempo utilizándose, y por lo tanto es posible que el rotor tenga una tendencia a descolgarse, y tener el entrehierro más estrecho en la parte inferior que en la parte superior de este, como consecuencia de esto, habrá más tensión en la parte inferior del rotor, que en el superior, y por lo tanto se pueden producir grandes corrientes y por lo tanto un gran calentamiento y desgaste de las escobillas. Para poder remediarlo habría que colocar compensadores o devanados de compensación. Este produce un cortocircuito en puntos con el mismo nivel de tensión en caminos paralelos, y gracias a esto, evita que estas elevaciones de corrientes, entren en cortocircuito, y dañen las escobillas.

Devanado ondulado.

También llamado en serie, en este tipo de conexión solo habrá dos caminos de corriente, y por lo tanto se conectarán a un segmento de conmutación una vez sí y otra no. Gracia a este tipo de conexión no se va a poder producir ningún tipo de desequilibrio de voltaje, ya que las tensiones de salida son la suma de cada polo. Este tipo de conexión es favorable para una máquina de una alta tensión, ya que se permite acumular en los devanados una mayor tensión.

2.3 Posibles pérdidas de una maquina eléctrica.

Los diferentes tipos de pérdidas que se pueden producir en una máquina de corriente continua se pueden dividir en unas 5 principales.

- *Pérdidas en el cobre, o pérdidas eléctricas.*

Estas se pueden dar en los dos devanados, tanto en el devanado inducido, como en el devanado de campo de la máquina, las cuales dependen tanto de la intensidad que circula por el cobre, como de la resistencia de este, con las siguientes ecuaciones:

Para el inducido $P_A = I_A^2 \cdot R_A$

Para el campo $P_F = I_F^2 \cdot R_F$

Normalmente este tipo de pérdidas no suelen ser muy significantes, y se pueden reducir con otro grosor de cable.

- *Pérdidas en las escobillas.*

Estas son debidas a lo ya nombrado anteriormente, al rozamiento y desgaste de las escobillas, y en cuanto peor estado estén estas, más pérdidas se producirán. Se pueden cuantificar según la caída de tensión que tengan en las escobillas y la corriente que pase a través de él, se representarán con la siguiente fórmula:

$$P_{CE} = V_{CE} \cdot I_A$$

Pero si normalmente se podrá suponer que habrá una caída de unos 2 voltios.

- *Perdidas en el núcleo.*

Son la resultante de la suma de las pérdidas producidas por histéresis y por corrientes parásitas, y se localizan en el metal del motor.

Las pérdidas por histéresis de la chapa metálica, son aquellas producidas por la habilidad del material a conservar el magnetismo al realizar en ellas una variación de la inducción magnética (B).

Y las pérdidas por Foucault debidas a la oposición del material al cambio del flujo de inducción. En realidad se puede decir que estas pérdidas son iguales a las perdidas por efecto Joule.

- *Pérdidas mecánicas.*

Estas son debidas a los elementos físicos de una máquina, debido a la fricción y al rozamiento del motor. La parte de las pérdidas de fricción es por las escobillas ya nombradas antes, mientras que la parte de rozamiento con el aire, es las partes movimiento del rotor. Estas pérdidas variarán según la velocidad que tenga la máquina.

- *Pérdidas dispersas.*

También llamadas pérdidas misceláneas. Estas serán todas aquellas pérdidas que tenga una máquina pero que no se pueda incorporar en ninguna de las categorías anteriores. Aproximadamente se dice que estas pérdidas podrán alcanzar un 1% de la plena carga.

Se pueden representar en un diagrama de Sankey, todas las pérdidas que se tendrán en cuanta para calcular la potencia de un motor.

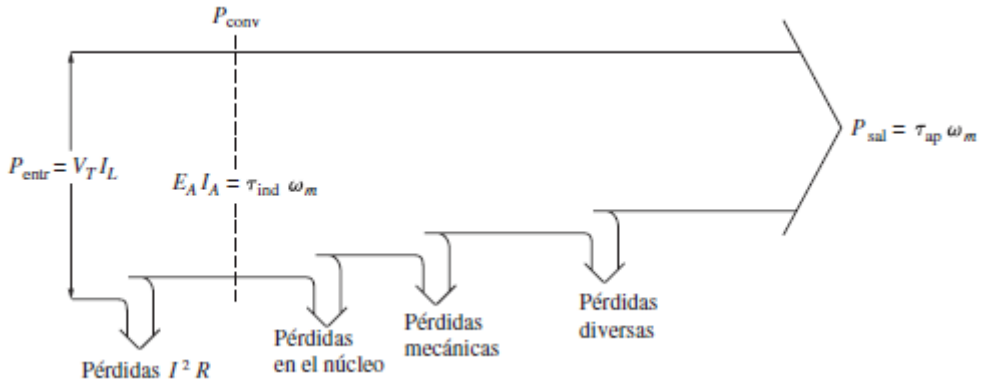


Figura 206: Diagrama de Sankey con todas las pérdidas de potencia de un motor.

2.4 <u>Circuito equivalente de un motor de corriente continua.</u>

Como ya hemos nombrado varias veces, el motor está formado principalmente por un estator y un rotor, los cuales van a tener unas bobinas de campo y unas bobinas del inducido respectivamente. El circuito equivalente de un motor representado en el figura 207 va a constar principalmente de una fuente de tensión llamada, E_A y una resistencia variable R_A la cual representa el circuito equivalente Thevenin que engloba toda la estructura del rotor con sus bobinas, interpolos y devanados de compensación si lo tuviera. La parte de la caída de tensión se podría representar como una fuente de tensión con la polaridad cambiada al sentido del flujo de corriente. El campo magnético del estator estará representado por las bobinas de campo y están simbolizadas por L_F y un resistor R_F. Además se le añade un resistor independiente ajustable llamado R_{ajus} que se utilizará para lograr un control de la corriente de campo.

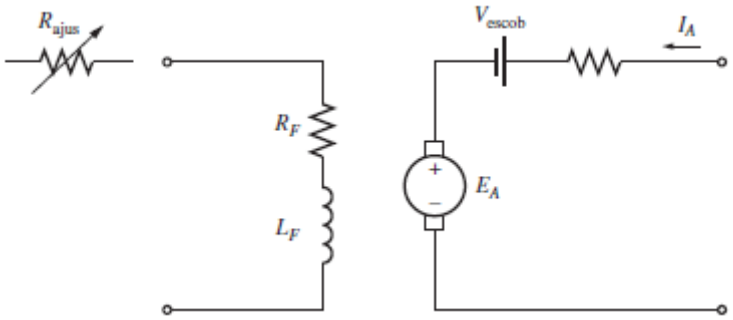


Figura 207: Circuito equivalente de un motor de corriente continua.

Este sistema se puede simplificar colocando la fuente de tensión E_A y la caída de tensión de las escobillas como una sola fuente, también se agrupa la resistencia de campo, tanto la fija del circuito como la variable, y se determina como una sola llamada R_F. Además se simplifica todas las bobinas de campo en una sola si hubiese varias. De esta manera, el circuito equivalente básico quedaría tal que así:

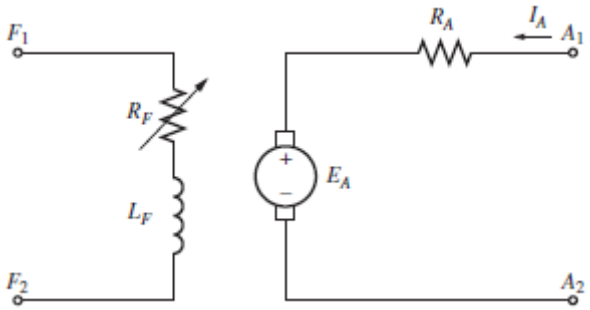


Figura 208: Circuito equivalente simplificado de un motor de corriente continua.

2.5 Curva de magnetización.

La tensión interna que tiene el motor E_A estará caracterizada por la siguiente fórmula:

$$E_A = K \cdot \emptyset \cdot \omega_m$$

Se puede decir que la tensión generada internamente en la maquina es proporcional al flujo de esta y la velocidad en la que gira el rotor. Se produce una fuerza magnetomotriz en la parte del circuito de campo, gracias a la corriente que circula por ella.

$$\Im = N_F I_F$$

Esta fuerza magnetomotriz, varia con respecto al flujo de acuerdo a la siguiente gráfica.

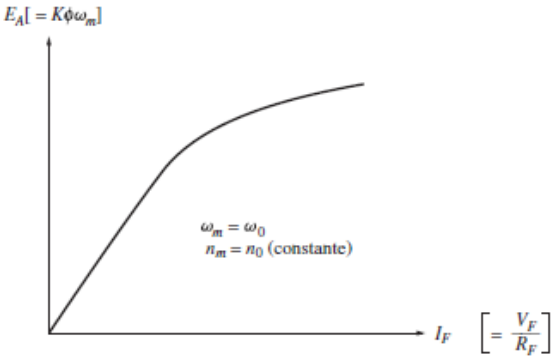


Figura 209: Gráfico que representa la tensión interna de un motor con respecto a la corriente para una velocidad constante.

Debido a que la corriente de campo es proporcional a la fuerza magnetomotriz, y que la tensión interna generada lo es también con respecto al flujo, se podrá representar gráficamente de forma casi idéntica de la tensión interna E_A con respecto la velocidad ω_m como se muestra en la figura 210. Esta gráfica será la llamada curva de magnetización o curva de saturación la cual se va a caracterizar que con un gran incremente del flujo dará lugar a una pequeña variación de la fuerza magnetomotriz, pero después de un cierto puntos a pesar de un gran aumento de la fuerza

magnetomotriz, la variación del flujo serán más pequeños hasta llegar a un momento, en el cual, el flujo casi será constante con respecto a la fuerza electromagnética.

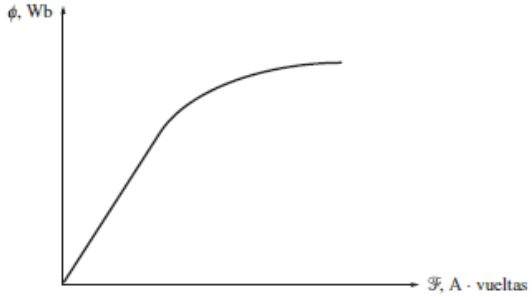


Figura 210: Gráfico que representa el flujo de un motor con respecto a la fuerza magnetomotriz.

Además debemos decir que en los motores se intentará diseñar para que siempre funcionen en el momento de saturación de esta curva, aunque se tenga que incrementar la corriente de campo para subir el nivel de tensión interna.

2.6 Par

Cuando un mecanismo que gire alrededor de un eje central, es necesaria una fuerza tangencial que mantenga esa rotación, el producto de esta fuerza por la distancia que existe entre el centro del eje hasta la recta de acción de la fuerza se le llamara par. El hecho de realizar un rotor con una sola bobina, será poco efectivo ya que habrá intermitencias en la producción de par cuando la fuerza ejercida en la bobina sea cero, ese será uno de los motivos por el cual se dispondrá un rotor de varias vueltas y con varios bobinados en su interior.

El par inducido producido por una maquina se determinará de la siguiente manera:

$$\tau = r \cdot F \ \sin\theta$$

Teniendo en cuenta que la r es la distancia desde el eje central y la posición donde se producirá la fuerza de rotación, y θ es el ángulo entra la distancia r y la fuerza. Desarrollando esta fórmula para cada una de las caras de las bobinas, sumándolas y teniendo en cuenta que el valor de la fuerza es igual al campo magnético, por la longitud y por la intensidad que circula por la bobina. La fórmula principal que tendrá el par magnético será:

$$\tau_{ind} = K \cdot \emptyset \cdot I_A$$

2.7 Tipos de motores

Pueden encontrarse distintos tipos de disposiciones en los que conectar un motor o un generador, en ambos casos son los mismos.

- *Motor con excitación independiente y en derivación.*

En estos dos tipos de conexiones se pueden encontrar una gran diferencia principal. El motor con excitación independiente será el que su circuito de campo se alimentará de una fuente de tensión independiente, mientras que una conexión en derivación, este circuito de campo se conecta en paralelo con el circuito del inducido, y se alimenta de este.

Sus circuitos equivalentes serán los siguientes.

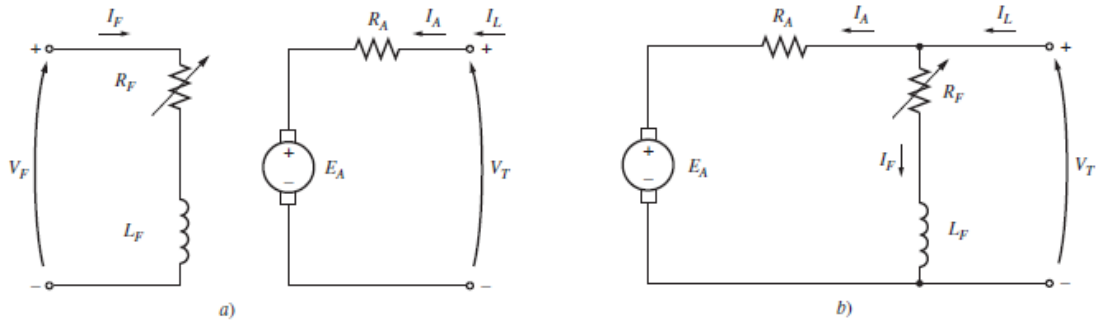


Figura 211: Circuito equivalente con excitación independiente (a) y en derivación (b).

Como se muestra en la figura 211, las relaciones de tensiones e intensidades pueden variar según el método que utilicemos. Por lo tanto si tenemos un circuito con excitación independiente (a), tendremos una única intensidad en el circuito del inducido, llamada I_A o I_L que son valores iguales. La intensidad de campo se puede calcular como el cociente entre la tensión que se suministrara el circuito de campo, entre la resistencia total de esta.

$$I_F = \frac{V_F}{R_F}$$

Y la tensión del inducido que será igual a la tensión interna del motor más la caída de tensión que existe en ese circuito según la ecuación de la ley de Kirchhoff.

$$V_T = E_A + I_A \cdot R_A$$

Sin embargo en un motor en derivación, las intensidades estarán unas relacionadas con las otras con la siguiente fórmula:

$$I_L = I_A + I_F$$

Siendo la I_A la intensidad del circuito inducido, la I_F la intensidad de campo, y la I_L es la intensidad sumada en paralelo. La intensidad de campo se calculará igual que la forma anterior pero teniendo en cuenta que la tensión que tiene en el inducido es la

misma que en la de campo. Y la fórmula que se desarrolla con la ley de Kirchhoff es exactamente igual que en la conexión de excitación independiente.

- *Motor en serie.*

Este tipo de conexión se puede caracterizar diciendo que todos sus componentes están en la rama del inducido conectados en serie. Se encontrará una única intensidad que circula a través de este circuito como se muestra en la figura 212, por lo tanto $I_A = I_S = I_L$.

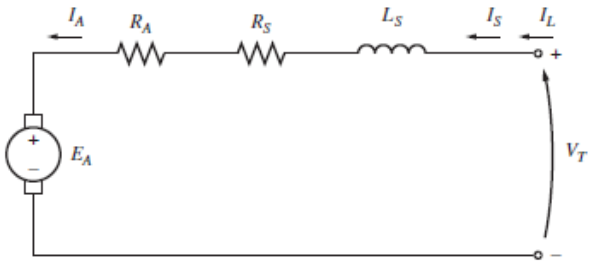


Figura 212: Circuito equivalente de un motor conectado en serie.

Nos encontramos en esta rama, la resistencia del inducido además de otra resistencia en serie, y una inductancia, además de la fuente de tensión interna. Según Kirchhoff, la ecuación que se obtendrá será la siguiente:

$$V_T = E_A + I_A \cdot (R_A + R_S)$$

- *Motor compuesto.*

Como bien dice el nombre, este tipo de motor es una composición entre el motor derivación y el motor serie, ya que será igual que una conexión en derivación pero con una segunda resistencia R_S y con otra bobina en serie con el inducido. Además se podrán diferenciar dentro de este tipo, unos dos dependiendo de donde se realiza la conexión de la rama de campo. Podrá ser motor compuesto con conexión de derivación larga (figura 213 a)) que tendrá la rama de campo antes de todos los componentes del inducido, o con conexión de derivación corta (figura 213 b)), la cual se pondrá esta rama después del bobinado del inducido y de la resistencia R_S.

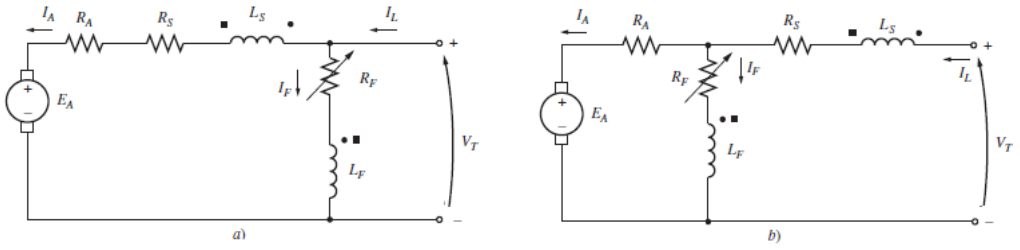


Figura 213: Circuito equivalente de un motor compuesto con conexión larga (a) o con conexión corta (b).

Los puntos y cuadrados que se observan en la imagen anterior representan otros dos tipos de circuitos dentro del motor compuesto en cada una de ellos. Podrán ser compuesto acumulativo o compuesto diferencial. Si la corriente circula hacia los puntos de los bobinados tanto de campo como el inducido, la fuerza electromotriz resultante se

sumarán para hacerla más grande, esta será el compuesto acumulativo que está representada en la imagen con los puntos. Pero sin embargo si la corriente fluye hacia el punto de una de las bobinas, y hacia afuera en la otra, este será compuesto diferencias representado en la imagen con los cuadrados y en este caso se restarían las fuerzas electromotrices y obtendríamos otra más pequeña.

2.8 Control de motor de DC en derivación

Las características principales que debemos saber para controlar perfectamente un motor son el par de salida y la velocidad del eje de rotación

En un primer lugar supondremos que vamos a disminuir la carga que tiene la parte del inducido, es decir, el rotor. En este caso obtendremos una disminución del par del inducido. En un maquina eléctrica el par tanto del inducido como el de carga, deberán de ser iguales o próximos, por lo tanto se realizaran unas variaciones hasta que lo consiga. En este caso al disminuir el par del inducido, se reducirá la velocidad del rotor, y con ello la tensión interna como se observa en la formula.

$$E_A \downarrow = K \cdot \emptyset \cdot \omega_m \downarrow$$

A su vez la corriente del inducido aumentará con la disminución de la tensión interna,

$$V_T = E_A + I_A \cdot R_A \;\; \rightarrow \;\; I_A \uparrow = \frac{V_T - E_A \downarrow}{R_A}$$

Y por lo tanto conforme aumenta la corriente del inducido también lo hará con él el par del inducido, consiguiendo el par anterior, pero con un valor de velocidad de rotación más bajo.

$$\tau_{ind} \uparrow = K \cdot \emptyset \cdot I_A \uparrow$$

Por lo tanto podemos concluir que si reducimos la carga del inducido, reduciremos la velocidad del rotor.

Podremos obtener una ecuación única de la velocidad con respecto al par si desarrollamos y sustituimos entre ellas las ecuaciones anteriores.

$$\left.\begin{matrix} V_T = E_A + I_A \cdot R_A \\ E_A = K \cdot \emptyset \cdot \omega_m \end{matrix}\right\} \left.\begin{matrix} V_T = K \cdot \emptyset \cdot \omega_m + I_A \cdot R_A \\ I_A = \frac{\tau_{ind}}{K \cdot \emptyset} \end{matrix}\right\} V_T = K \cdot \emptyset \cdot \omega_m + \frac{\tau_{ind}}{K \cdot \emptyset} \cdot R_A \rightarrow \boxed{\omega_m = \frac{V_T}{K \cdot \emptyset} - \frac{R_A}{(K \cdot \emptyset)^2} \tau_{ind}}$$

Gracias a esto, tendremos una ecuación de una recta descendente del tipo:

$$y = A - B \cdot x$$

Por lo tanto cuanto mayor sea el valor de B mayor inclinación tendrá, y el valor servirá como referencia del inicio del eje "y". Entonces podemos darnos cuenta que la variación de la velocidad con respecto al par se puede representar en una línea de forma lineal recta negativa. Pero también se puede dar el caso de producirse una reacción en el inducido, y por lo tanto se va a producir una reducción del debilitamiento del flujo al aumentar la carga, por lo tanto en este caso la velocidad no se reducirá con respecto al par de forma recta como se muestra en la figura 214, a no ser que en el motor tengamos un devanado de compensación que gracias a este podemos olvidar cualquier debilitamiento.

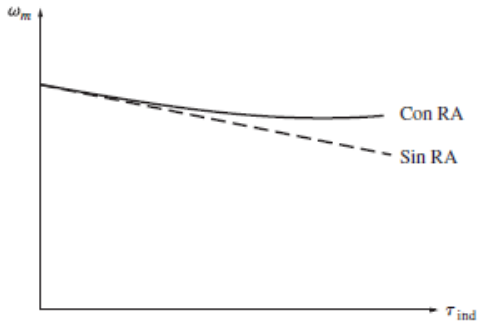


Figura 214: Gráfica que representa la velocidad con respecto el par inducido.

Debido a que el flujo, o lo que es lo mismo, la tensión interna generada E_A no tiene una función lineal con la fuerza electromagnética debemos observar la variación de estas según la curva de magnetización ya nombradas anteriormente, ya que no se puede realizar un cálculo analíticamente.

Si en una reacción del inducido se reduce el flujo, la fuerza electromotriz será aquella que se genera en el devanado de campo menos la producida por esta reacción, y debido a que en la curva de magnetización se compara la tensión interna con la corriente de campo, habrá que desempeñar esta variación de fuerza electromagnética en función de la corriente de campo equivalente.

$$\Im_{net} = N_F \cdot I_F - \Im_{RA} = I_F^* \cdot N_F \quad \rightarrow \quad I_F^* = \ I_F - \frac{\Im_{RA}}{N_F}$$

Pero debido a que generalmente la curva de magnetización que tengamos será de unas características específicas de velocidad, deberemos realizar un factor de conversión de la tensión interna que obtenemos de la curva con esa velocidad final que se va a utilizar, para así obtener la tensión real que tenemos.

$$E_A = K \cdot \emptyset \cdot \omega_m \ \rightarrow \quad (K \cdot \emptyset = cte) \quad \rightarrow \quad \frac{E_A}{E_{A0}} = \frac{n_m}{n_o}$$

Para poder realizar un control de la velocidad del rotor se podrá conseguir gracias a tres variables de la máquina.

- Al modificar la cantidad de resistencia de campo R_f.
- Variando la tensión de los terminales externos del inducido.
- Y añadiendo una resistencia en serie, que conseguirá el mismo paso anterior de variar la tensión

- *Variación de la resistencia de campo.*

La variación que realizamos con la resistencia de campo producirá unos efectos parecidos a los explicados anteriormente con la resistencia del inducido.

Si bajamos el valor de la resistencia de campo, lograremos una disminución de la intensidad de campo a la vez del flujo. Esto provocará una reducción de la tensión interna de la máquina y compensará un aumento de la corriente del inducido.

$$I_F \downarrow = \frac{V_T}{R_F \uparrow} \quad \rightarrow \quad E_A \downarrow = K \cdot \emptyset \downarrow \cdot \omega_m \quad \rightarrow \quad I_A \uparrow = \frac{V_T - E_A \downarrow}{R_A}$$

Por lo tanto con la fórmula del par inducido, observamos que hay dos términos que variarán en esta misma operación la corriente del inducido, y el flujo. Pero en esta ocasión la variación de la corriente predominará ante la variación del flujo, por lo tanto el par aumentará también.

$$\tau_{ind} \uparrow = K \cdot \emptyset \downarrow \cdot I_A \uparrow$$

La velocidad el motor aumentara con respecto al par. Después de esto, la tensión interna aumentará con respecto a la velocidad, y disminuirá la intensidad del inducido, con esto consigue que el par del inducido vuelva a igualarse con el par de campo.

$$\tau_{ind} > \tau_{carga}$$

$$E_A \uparrow = K \cdot \emptyset \cdot \omega_m \uparrow \ \rightarrow \ I_A \downarrow = \frac{V_T - E_A \uparrow}{R_A} \ \rightarrow \ \tau_{ind} \downarrow = K \cdot \emptyset \cdot I_A \downarrow \ \rightarrow \ \tau_{ind} = \tau_{carga}$$

Finalmente podremos definir lo que ocurre en rasgos generales cuando se modifica la resistencia de carga.

- Si la resistencia de campo (R_f) aumenta la velocidad (ω_m) aumenta también.

Hay que tener en cuenta que una disminución del par de la maquina va a aumentar la velocidad de la máquina de forma gráficamente recta pero según el valor de la resistencia de campo la pendiente de esta recta será diferente como se muestra en la figura 215.

Si la resistencia aumenta la pendiente será mayor. Se puede demostrar lo que ocurre con la fórmula que hemos desarrollado antes de la velocidad.

$$\omega_m = \frac{V_T}{K \cdot \emptyset} - \frac{R_A}{(K \cdot \emptyset)^2} \cdot \tau_{ind}$$

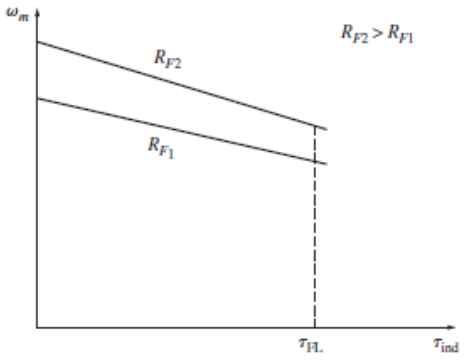


Figura 215: Gráfico que representa la variación de la variación respecto al par inducido, con dos valores de resistencia de campo.

Sin embargo podremos observar que esta variación de pendiente hará que se junten en un punto de la gráfica, y se observará que a unos bajos valores de velocidades un incremento de la resistencia de campo hará que la velocidad disminuya y no al contrario como ocurre en valores normales de velocidad, esto ocurre debido a la formula explicada anteriormente del par inducido ya que la variación de intensidad será menos predominante que la variación del flujo.

$$\tau_{ind} = K \cdot \emptyset \cdot I_A$$

- *Cambio de tensión en el inducido.*

Para poder realizar el control de la velocidad de esta manera, debemos realizar el control de la tensión de la parte del inducido, pero sin modificar la tensión de campo. Para ello debemos incorporar un control de tensión variable después del circuito de campo. Por lo tanto para un aumento de la tensión del inducido V_A, la tensión de esta parte aumentará al mantenerse fija la resistencia, gracias a esto, el par inducido aumentará con este, por lo que habrá una desigualdad del par de carga y el par inducido.

$$I_A \uparrow = \frac{(V_A \uparrow - E_A)}{R_A} \rightarrow \tau_{ind} \uparrow = K \cdot \emptyset \cdot I_A \uparrow \rightarrow \tau_{ind} > \tau_{carga}$$

Esto hará que aumente la velocidad del motor, y con ella la tensión interna para que vuelva a disminuir la tensión del inducido y con esto vuelva el par inducido a un valor igualado al par de carga.

$$E_A \uparrow = K \cdot \emptyset \cdot \omega_m \uparrow \rightarrow I_A \downarrow = \frac{(V_A - E_A \uparrow)}{R_A} \rightarrow \tau_{ind} \downarrow = K \cdot \emptyset \cdot I_A \downarrow \rightarrow \tau_{ind} = \tau_{carga}$$

- Por lo tanto con un aumento de la tensión del inducido, aumentara la velocidad.

En este caso obtendremos gráficamente una recta descendiente como en el caso anterior, pero al realizar una variación de tensión como hemos hablado, la pendiente será la misma, por lo tanto todas las líneas de todas las posibles tensiones serán paralelas como se muestra en la figura 216.

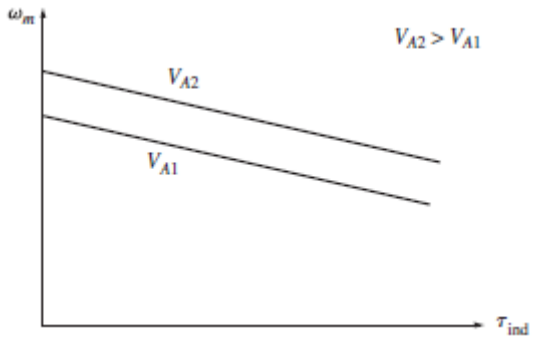


Figura 216: Gráfica que representa la variación de velocidad con respecto al par inducido con diferentes tensiones en el inducido.

- *Añadiendo una resistencia en serie en el inducido*

En este caso lo que producirá este cambio es el aumento de pendiente descendente a medida que vamos aumentado la resistencia en serie de la gráfica de la velocidad con respecto a la par. Por lo tanto cuando más cargado se encuentre el inducido, menos será la velocidad con el mismo valor de par del inducido.

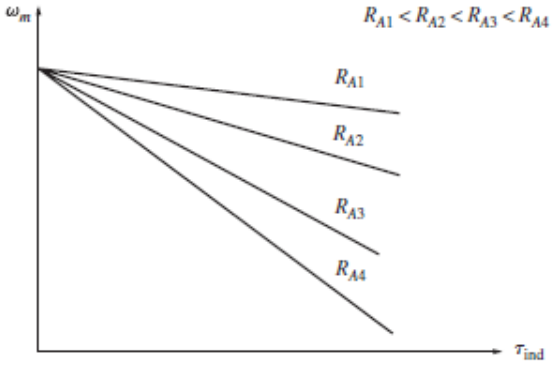


Figura 217: Gráfica que representa la velocidad con respecto al par magnético con varios valores de la resistencia en serie del inducido.

Se puede demostrar matemáticamente con la fórmula de la velocidad demostrada anteriormente.

$$\omega_m = \frac{V_T}{K \cdot \emptyset} - \frac{R_A}{(K \cdot \emptyset)^2} \cdot \tau_{ind}$$

Y como ya hemos dicho antes el segundo término hace que dependa la inclinación de la recta, y el valor de la resistencia nombrada está solo en este término mientras que el valor del primer término siempre será constante.

Pero debido a que este método produce muchas pérdidas a la hora de trabajar, será el que menos se utilice.

3. Materiales.

3.1 Motor paso a paso.

Utilizaremos un motor paso a paso de dos bobinas un total de 4 cables, con un par de 44Ncm, tensión nominal de 2,8V y corriente nominal de 1,68A

Figura 301: Motor paso a paso RS 5350401 de 400 pasos.

Una bobina está formada por el cable rojo y el azul, mientras que la otra por el cable verde y negro.

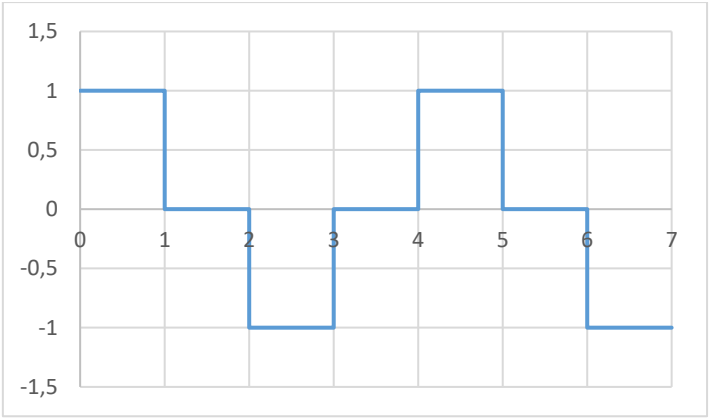
Se puede realizar el control de este tipo de motores de las siguientes formas:

- *Paso simple*

  Este está formado por los 4 siguientes pasos

  1. Bobina A con intensidad positiva y bobina B sin intensidad
  2. Bobina A sin intensidad y Bobina B con intensidad positiva
  3. Bobina A con intensidad negativa y bobina B sin intensidad
  4. Bobina A sin intensidad y Bobina B con intensidad negativa

  Quedaría de tal manera representada en intensidades de 1 amperio



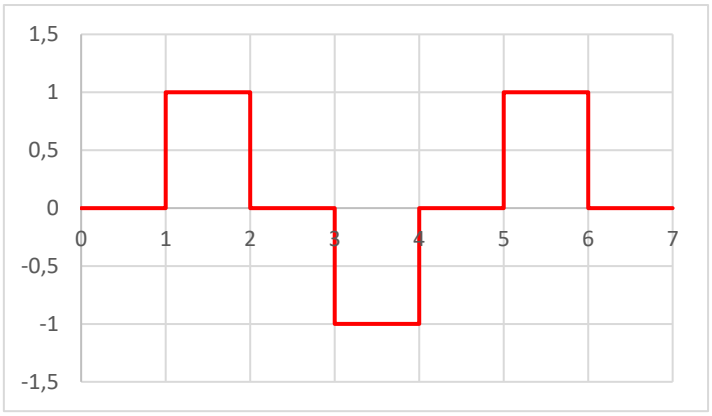


Figura 302: Paso simple. Niveles de intensidad para la bobina A (Azul) y la bobina B (Rojo) en un eje X de valores de 10^{-2}

Este va a ser el que realizaremos en los ensayos de arduino.

- Paso doble.
  1. Bobina A con intensidad positiva y bobina B con intensidad positiva
  2. Bobina A con intensidad negativa y Bobina B con intensidad positiva
  3. Bobina A con intensidad negativa y bobina B con intensidad negativa
  4. Bobina A con intensidad positiva y Bobina B con intensidad negativa

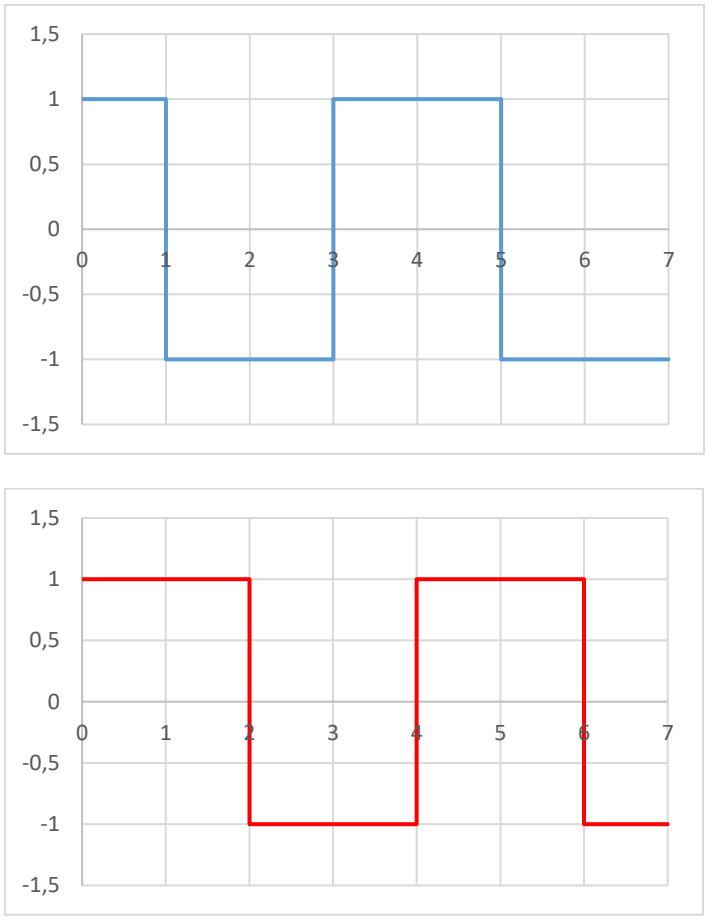


Figura 303: Paso doble. Intensidad para la bobina A (Azul) y bobina B (Rojo) en un eje X de valores de 10^{-2}

La diferencia que nos encontramos en este paso doble será en el aumento de fuerza que tendrá el rotor en el giro, debido a que las dos bobinas están activadas constantemente.

- Medio paso.

Estará formado por 8 pasos.

1. Bobina A con intensidad positiva y bobina B sin intensidad
2. Bobina A con intensidad positiva y bobina B con intensidad positiva
3. Bobina A sin intensidad y bobina B con intensidad positiva
4. Bobina A con intensidad negativa y bobina B con intensidad positiva
5. Bobina A con intensidad negativa y bobina B sin intensidad
6. Bobina A con intensidad negativa y bobina B con intensidad negativa
7. Bobina A sin intensidad y bobina B con intensidad negativa
8. Bobina A con intensidad positiva y bobina B con intensidad negativa

Se podría representar de la siguiente manera:

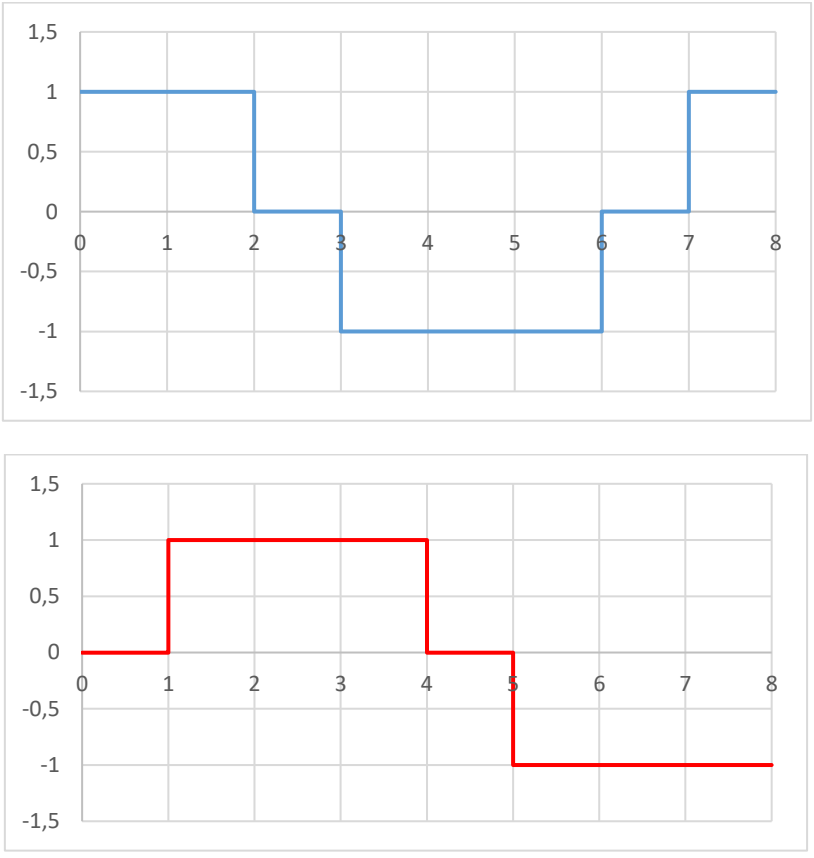


Figura 304: Medio paso. Niveles de intensidad para la bobina A (Azul) y la bobina B (Rojo) en un eje X de valores de 10^{-2}

Este destacará por la combinación de los dos anteriores, así tendremos mayor precisión en el recorrido del rotor ya que tenemos el doble de pasos.

3.2 Motor DC brushed

Es un motor de corriente continua cepillado básico, el cual podemos modificar su velocidad básicamente modificando la tensión de alimentación del mismo, o bien la fuerza de campo magnético. Estos motores suelen ser utilizados para propulsión eléctrica. Estos motores están tendiendo a desaparecer debido al desgaste de las escobillas por la utilización prolongada del mismo, sustituyéndose por otros motores sin escobillas.

Este tipo de motor funciona exactamente como se explicó anteriormente en el apartado 2.2 (Motores DC)

Figura 305: Motor brushed de corriente continua.

El motor está compuesto básicamente por dos entradas de tensión, una positiva y otra negativa, que da a nuestras dos escobillas, que hace el rozamiento con las bobinas de su interior. Además de su eje de giro, que atraviesa el motor, y sobresale por sus dos extremos.

3.3 Servomotor

Podremos encontrarnos dos tipos de servomotores, un servomotor convencional con un giro controlado por el ángulo con un eje que tiene un movimiento de 180 grados máximo, y un servomotor de rotación continua con un giro de 360 grados la cual controlamos su velocidad para sus dos posibles direcciones. Nosotros utilizaremos en los ensayos un servomotor de rotación continua, tal y como se muestra en la siguiente imagen.

Figura 306: Servomotor de movimiento perpetuo.

Las características y el control de estos dos son muy similares entre ellos. Nuestro servomotor será el SM-S4303R. Se necesita una tensión de alimentación entre 4,8 a 7,2V y una corriente entre 200mA y 250mA. Lamentablemente no disponemos de un control preciso de la velocidad de giro de este servo debido a que no es lineal el cambio de valor del tiempo con la velocidad.

Este tipo de servomotor tiene tres cables, en nuestro caso de califican de la siguiente manera.

Negro	GND
Blanco	Señal
Rojo	5V

Tabla 1: *Utilidad de los tres cables del servomotor.*

Para poder realizar el movimiento del servo se debe conocer que este necesita impulsos de tensiones de entre 0,5 a 2,5 milisegundos con respecto a un periodo de 20ms que son para 50 Hz.

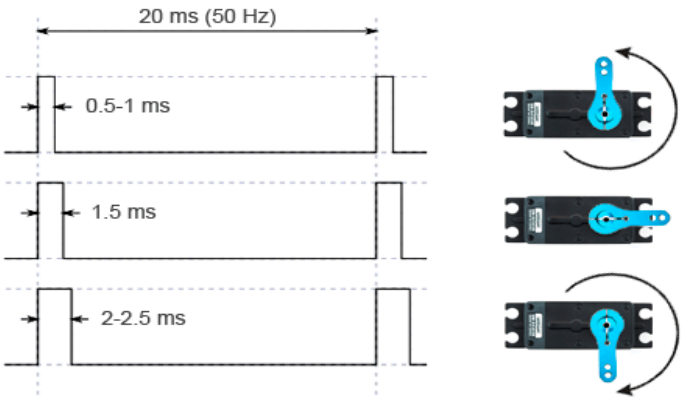


Figura 307: Impulsos de tensión para el movimiento del servomotor

Por lo tanto para pulsos entre 0.5 y 1 ms girara en un sentido, para 1,5 se parara, y de 2 a 2.5 avanzara en sentido contrario.

3.4 Potenciómetro

En algunos ensayos podremos añadirle un potenciómetro que haga que varíe algún valor determinado a través de un potenciómetro.

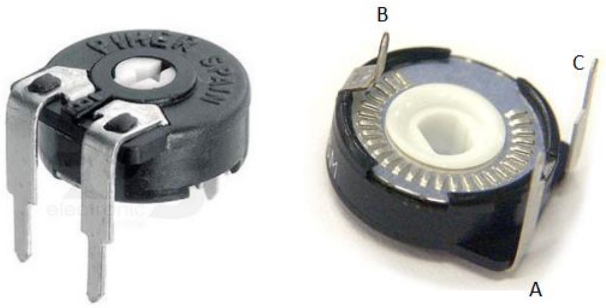


Figura 308: Potenciómetro. Parte superior e inferior.

Este está formado de tres patillas, dos en un lado y otra patilla en el otro, y se podrá representar su interior con la siguiente figura.

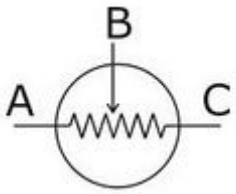


Figura 309: Esquema unifilar del potenciómetro.

Por lo tanto en la patilla A y C conectaremos la tensión y la tierra, independientemente del orden entre estos dos, y en B será la señal que queremos recibir.

4. Arduino

La empresa Arduino se encarga de producir placas computadoras, también llamados microcontroladores de placas simples formadas por circuitos impresos y desarrolladas para facilitar el mundo de la electrónica, el diseño y la programación para aficionados, estudiantes e ingenieros.

Podemos separar este tipo de placas en dos partes:

4.1 Hardware

Consiste en la placa de circuito impreso que integra generalmente, un microprocesador Atmel AVR, con una entrada USB para su alimentación y transferencia de datos con la computadora, puertos digitales y entradas y salidas analógicas, además de una entrada de tensión auxiliar. Estos componentes irán variando según el modelo de arduino que tengamos.

Podemos destacar los más representativos y utilizados como:

Arduino Uno y Arduino Mega 2560 que serán los que utilizaremos nosotros en nuestro trabajo.

Figura 401: *Arduino Galileo*

Figura 402: *Arduino Zero*

Figura 403: *Arduino Leonardo*

Figura 404: *Arduino Due*

Figura 405: *Arduino Micro*

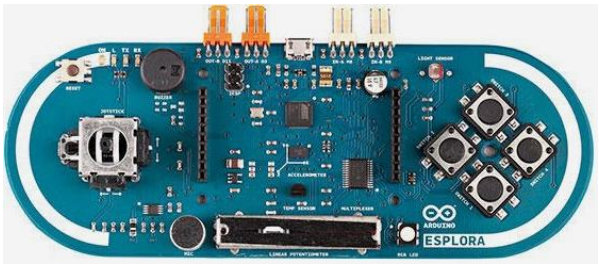


Figura 406: *Arduino Esplora*

Y otros muchos como: Arduino Yún, Tre (En Desarrollo), Mega ADK, Ethernet, Robot, Mini, Nano, LilyPad Arduino Simple, LilyPad Arduino SimpleSnap, LilyPad Arduino, LilyPad Arduino USB, Pro Mini, Fio, Pro, MKR1000/Genuino MKR1000, MICRO/Genuino MICRO, 101/Genuino 101, Gemma.

Todos estos modelos se podrán comparar y encontrar diferencias, esencialmente sobre la cantidad de pines de entrada y de salida, el tipo de microprocesador integrado y entre otros, el tipo de entrada de comunicación a la computadora siendo USB con los siguientes modelos de clavijas:

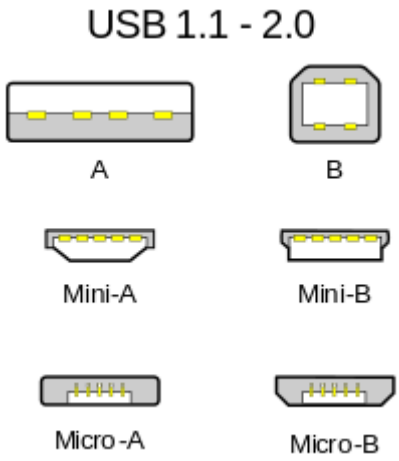


Figura 407: *Modelos de clavija de arduino*

Las placas arduino que utilizaremos en nuestro caso serán como ya hemos dicho antes: El arduino Uno y el Arduino Mega.

- *Arduino UNO:*

Trabaja habitualmente con una tensión de funcionamiento de 5V, aunque permite tensiones límites entre 7 y 12 voltios y con 20 mA para cada pin. Tiene una memoria flash de 32KB, con unas medidas de 68.6 x 53.4 mm siendo este tipo de placas de coste reducido.

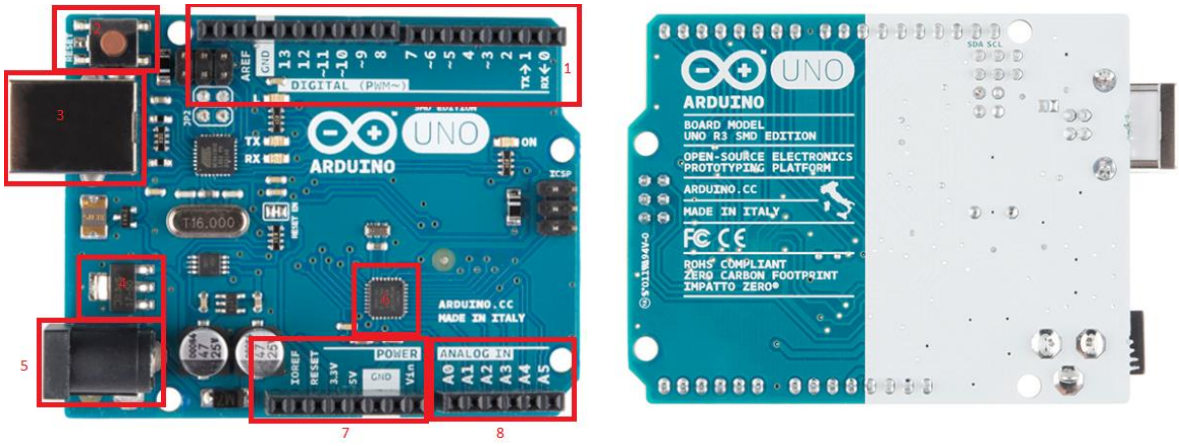


Figura 408: *Arduino Uno con sus partes principales..*

Formado por:

1 -> 14 pines digitales de entrada y de salida (6 de ellos con posibilidad de salida PWM).

2 -> Boton de reinicio de la programación.

3 -> Conexión USB tipo B

4 -> Regulador de tensión a 5 Voltios.

5 -> Entrada de alimentación externa con limite entre [6-20V] siendo lo recomendado entre [7-12 V]. Con entrada hembra Jack de alimentación.

6 -> Microcontrolador ATmega328P

7 -> Pines de energía (5V, 3.3V, dos pines de ground, Vin)

8 -> Son 6 pines analógicas de entrada o de salida.

- *Arduino Mega 2560:*

Trabaja habitualmente con una tensión de funcionamiento de 5V y con 20 mA para cada pin. Tiene una memoria flash de 256KB, con unas medidas de 101.52 x 53.3 mm.

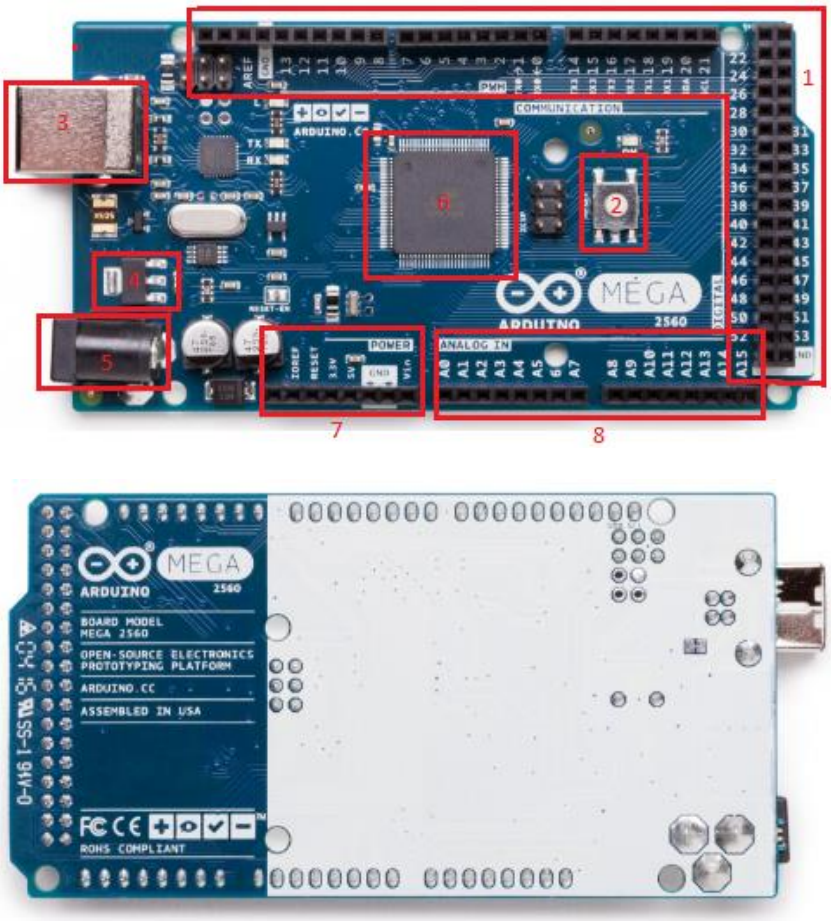


Figura 409:Tar*jeta Arduino Mega con sus partes principales.*

Formado por:

1 -> Con 54 pines digitales de entrada y de salida (15 de ellos con posibilidad de salida PWM).

2 -> Boton de reinicio del programación.

3 -> Conexión USB tipo B

4 -> Regulador de tensión a 5 Voltios.

5 -> Entrada de alimentación externa con limite entre [6-20V], recomendado entre [7-12 V]. Entrada hembra Jack de alimentación.

6 -> Microcontrolador ATmega2560

7 -> Pines de energía (5V, 3.3V, dos pines de ground, Vin)

8 -> Son 16 pines analógicas de entrada o de salida.

- *Shields*

Dentro de la parte del hardware también podemos añadirle los componente que utilizaremos como es una placa de extensión llamada shields.

Podemos encontrarnos diferentes placas shields que pueden encajar con nuestro arduino tanto el Uno como el Mega que realizan diferentes funciones como una pantalla, un adactador de entradas o salidas, para añadirle camara de fotos, una extensión wifi,… Pero nosotros utilizaremos una placa expecífica para el control de motores, llamada Motor Shield

La placa Motor Shield está diseñada para poder manejar cargas inductivas como relés, solenoides, motores DC y motores paso a paso. En este caso podemos introducir un máximo de dos motores de corriente continua, o un motor paso a paso de dos bobinas. Gracias a este podemos controlar la velocidad y dirección del motor de cada uno de ellos de forma independiente y sencilla. Tendrá una tensión entre unos 5 y 12 voltios, y unos 2 y 4A según si le añadimos una alimentación externa. En nuestro caso utilizaremos una tensión de 12V y 2 A. Se podrá realizar la alimentación, tanto por la entrada 5 de la figura 408 y 409, por los pines 7 de esas dos mismas figuras, como por las borneras 1 de la figura 410

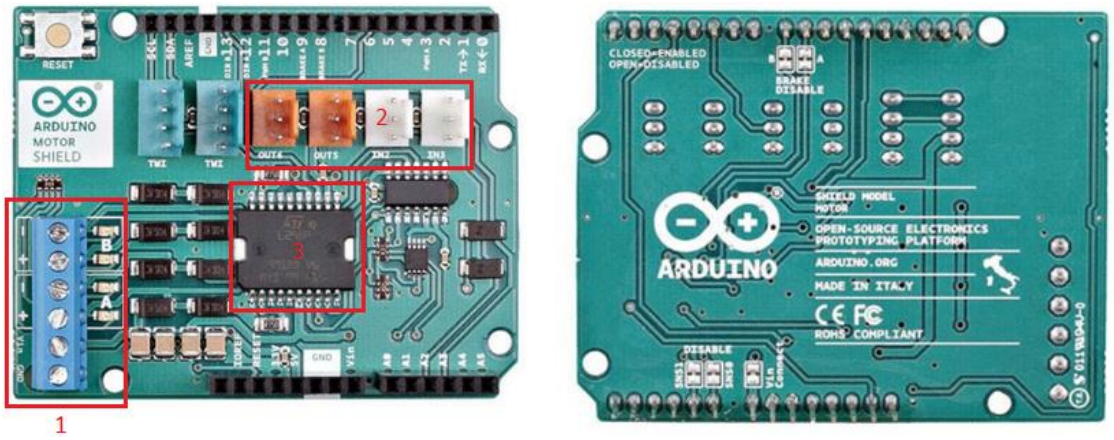


Figura 410: *Partes principales de la tarjeta MOTOR SHIELD de Arduino.*

La placa esta formada por:

1 -> 6 borneras. 2 para la bobina o motor A, 2 para la bobina o motor B (tanto positivo como negativo), una entrada de tensión, y un de tierra, los cuales es otra alternativa de alimentación externa.

2 -> 4 conexiones molex que sirven para introducir de forma sencilla servomotores (dos de ellos como entradas y otros dos para salidas) los cuales tienen tres pines machos, uno para la tensión, otro para tierra, y otro para la señal de funcionamiento. Cada uno corresponde con los pines 6 y 5 de salida, 2 y 3 de entrada.

3 -> Microcontrolador L298 basado en un puente completo doble. Este permite acoplar las potencias eléctricas de un motor o un par de motores con el microcontrolador del arduino.

El shield está formado por dos canales independientes, llamados A y B, y cada uno tiene predeterminado 4 pines del arduino para controlar el motor. En total hay unos 8 pines utilizados para este shield. Son los siguientes:

	A	B
Dirección	12	13
Velocidad (PWM)	3	11
Freno	9	8
Sentido de corriente	A0	A1

Tabla 2: *Canales y pines de la shield empleada para el control de motores.*

Los valores que pueden tomarse para los pines de la dirección y del freno son HIGH o LOW (1 o 2), determinando las dos sentidos posibles como para activar o desactivar el freno. Para la velocidad tomaremos valores entre 0-255. Esto realmente lo que nos hará modificar será el ciclo util (duty cycle) de la señal y se podría comparar con la siguiente figura 411.

Por lo tanto cuando modificamos el valor del pin 3 o 11 de nuestro arduino no modificamos tal cual la tensión enviada a nuestro motor, sino el ciclo de este. Si tenemos un valor de 255 tendremos un duty del 100% y por lo tanto si tenemos por ejemplo un valor de 100 nuestro duty seria de 39,21%.

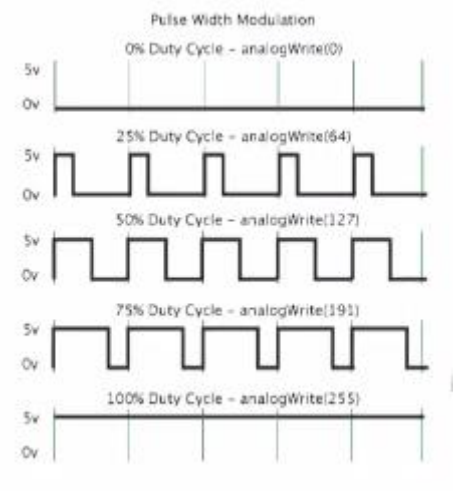


Figura 411: *Variación del ciclo util con el cambio del valor entre 0-255 del pin de velocidad*

4.2 Software

En la siguiente experimentación vamos a realizar el control de los motores a través del propio programa de arduino, y a través de Matlab.

- *Arduino 1.8.1*

El software, basado para arduino en Wiring, soporta todas las funciones de C y algunas de C++, se crea el código que se va a traspasar a nuestra placa arduino a través de una comunicación serial. Utilizaremos la versión arduino 1.8.1 el cual se puede descargar gratuitamente en la página https://www.arduino.cc/en/main/software en nuestro caso para Windows.

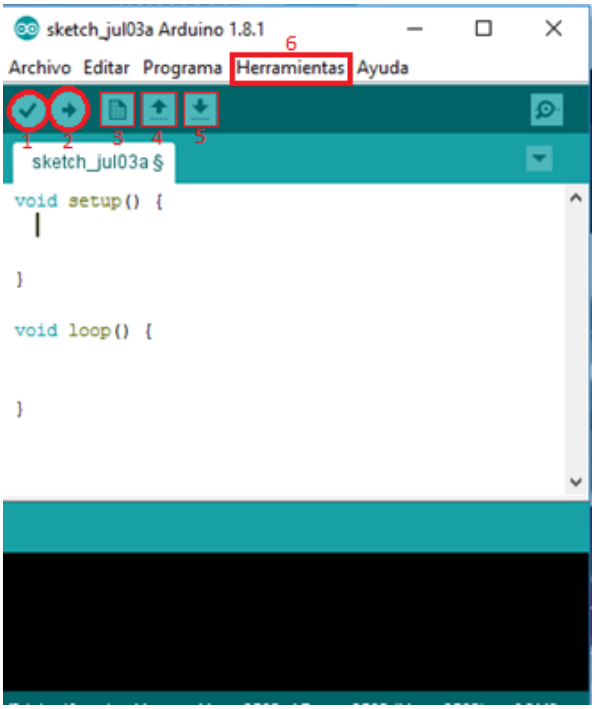


Figura 412: Pantalla principal del programa Arduino 1.8.1.

Una vez descargado, y abierto el programa obtendremos una página igual a la figura 319.

En 1, podremos verificar que el código este bien escrito, y en 2 enviarlo y guardarlo en el arduino. En 3 podemos abrir un nuevo archivo, en 4 abrir uno guardado anteriormente, y en 5 guardar el que está abierto.

En primer lugar se debe de configurar el programa conectando el arduino a través del puerto USB al ordenador. Se debe identificar en que puerto COM se conecta este, esto solo ocurrirá correctamente si tenemos el programa arduino instalado en el pc de lo contrario, no detectará el arduino como un puerto COM. Se podrá comprobar el puerto COM conectado dependiendo del Windows que tengamos.

Si es Windows 10 “Panel de control > Sistema y seguridad > Sistema > Administrador de dispositivos > Puertos (COM y LPT)”

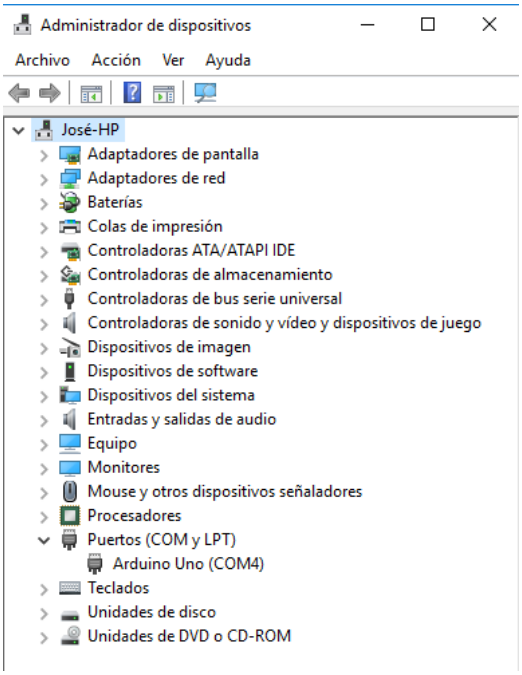


Figura 413: *Carpeta de administrador de dispositivos de Windows.*

Este valor de puerto COM estará determinado automáticamente por el pc, y pondrá siempre el COM inferior que este libre, sabiendo que existirán algunos que estarán siendo utilizados por el mismo sistema Windows y por lo tanto no tendrá por qué ser siempre el mismo.

Una vez conocido el puerto, volvemos al programa, y seleccionaremos en la barra de comandos el cuadro seleccionado en la imagen (6), “Herramientas > Placa”, tal y como se observa en la figura 414 y saldrá un listado de las clases de arduino existentes, después, en la misma barra de herramientas seleccionamos “Herramientas > Puerto” en este, habrá una lista de puertos disponibles del ordenador y por lo tanto seleccionaremos el indicado anteriormente. Ya estaremos listos para escribir el código.

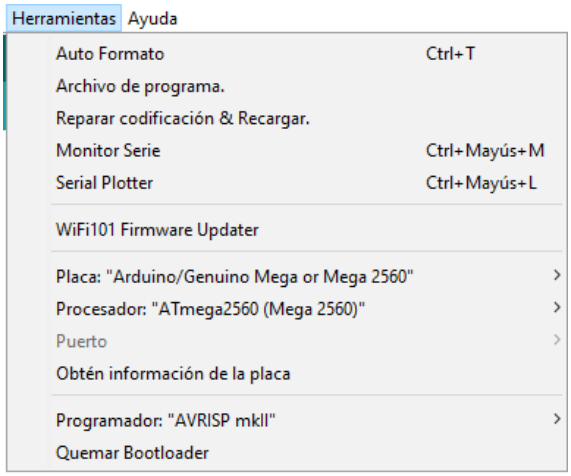


Figura 414: *Pestaña de la barra de herramientas para la preinstalción de arduino.*

En la barra de herramientas (6) de la figura 412 también podemos encontrarnos con el llamado monitor serie que mostrará valores que predeterminaremos en las líneas de código y el "Serial Plotter" que hace lo mismo, pero representado en una gráfica a través del tiempo de simulación.

El código se va a poder dividir en dos partes principales:

-La parte "*void setup() { }*" Todo el código que se coloque entre los corchetes se leerá y reproducirá en el arduino solamente una vez.

-Y la parte *void loop() { }* Por el contrario, el código escrito entre estos corchetes, se leerá en bucle de forma infinita.

A continuación de las líneas anteriores se pueden poner cualquier comando que queramos transferir al arduino. Explicaremos los comandos más importantes, y los que hemos estado utilizando en nuestro proyecto.

pinMode (1,2); -> Esta función activará los pines individualmente que utilizaremos en nuestro programa, en 1 pondremos el número de pin, y en 2 escribiremos "*OUTPUT*" si es un pin de salida, o "*INPUT*" si es en caso contrario.

int 1=2; -> Este tipo de línea se deberá colocar al inicio del programa, y fuera de los dos "*void*". Servirá para establecer una constante. Colocándose en 1 el nombre que queramos de nuestra constante, y en 2 lo que queremos guardar en ella. También podremos utilizar esta función de la siguiente manera: "*int 1*" para establecer una constante que utilizaremos más adelante poniendo en 1 el nombre, como ya hemos dicho. El programa dará error si lee una constante que no se ha establecido previamente.

digitalWrite(1,2); -> Podremos dar pulsos a los pines digitales poniendo en 1 el número de pin, y en 2 pondremos *HIGH* para el pulso positivo, y *LOW* en estado nulo.

analogWrite(1,2); ->Esta es la misma función que en el caso anterior, pero en este caso en 2 pondremos un valor comprendido entre 0-255 ya que estamos hablando de un pin analógico.

Hay que saber que en arduino tiene 10 Bits de entrada en un pin analógico y 8 Bits de salida, por lo tanto:

$$N = 10\ Bits \rightarrow\ \ 2^{10} = 1024$$

$$N = 8\ Bits \rightarrow\ \ 2^{8} = 256$$

Por esta razón decimos que el valor debe quedar entre 0 y 256, y más adelante para la salida será entre 0 y 1024.

digitalRead(1); -> Esta hará la función de recibir una señal de pulsos del arduino al ordenador, el valor se podrá guardar en una variable si la determinamos previamente como ya hemos explicado. Para ello debemos ponerlo de la siguiente manera *2=digitalRead(1);* siendo 2 el nombre de la variable.

analogRead(1); -> Este será igual que el anterior pero de una señal analógica, y como ya hemos explicado recibiremos valores entre 0 y 1024.

1=map(2,3,4,5,6); -> Gracias a esta función podremos transformar un valor o variable (2) de una escala entre (3) y (4) a otro valor o variable (1) con otra escala diferente entre (5) y (6).

delay(1); -> si escribimos esto, conseguiremos un tiempo de espera de lectura del código de valor (1) teniendo en cuenta que está en valores de milisegundos.

if(1==2){ 3 } -> Acción que comprueba una condición para activar el programa guardado en (3). Esta condición pueden ser las siguientes: "==" igual que, "¡=" diferente que, ">" mayor que, "<" menos que, entre otras. En caso de ser falsa la condición del "*if"*, accionará las funciones que hay dentro de la función "*else { 4 }".* Si no existe esta última, ignorara estas líneas.

Serial.begin(9600); -> Este comando es necesario si se desea abrir un puerto serie, que nos servirá para averiguar los valores que estamos recibiendo o enviando del arduino. Este fija la velocidad para la transmisión de datos en bits por segundo (baudio). Es importante que este valor coincida tanto en la línea de programación, como en el monitor serie, ya que si no se realizara, o no coincidiera no funcionaría.

Serial.available() -> Es el nombre de variable donde se guarda el valor recibido del puerto serie.

1=2.toInt(); -> Consigue guardar en (1) la parte entera del valor (2).

- *Matlab*

En este apartado vamos realizar las mismas operaciones anteriores con el programa Matlab en nuestro caso la versión 2015a. Para ello debemos descargar un paquete de soporte de hardware que se puede obtener de forma gratuita o bien en el apartado de las barras de herramientas "HOME>Resources>Add-Ons>Get Hardware Support Packages", o bien a través de la página oficial www.mathworks.com/hardware-support/arduino-matlab.html

Una vez realizado los pasos de la primera opción, nos saldrá la siguiente ventana emergente, en la cual nos da la opción de descargarlo e instalarlo a través de internet, de solo descargarlo de internet, de instalarlo de un archivo ya descargado, o de desinstalar, por lo tanto elegimos según la opción que hemos desempeñado previamente y le damos a "Next".

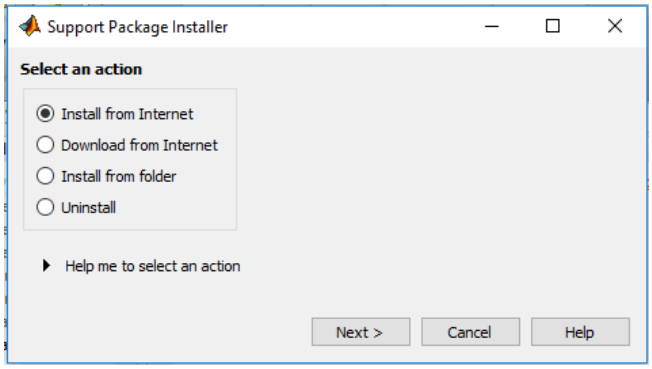


Figura 415: *Primera pestaña de instalación del paquete arduino para matlab.*

Si le hemos dado a descargar e instalar directamente de internet, nos aparecerá la siguiente ventana con un listado de paquetes de diferentes tarjetas que están disponibles para Matlab.

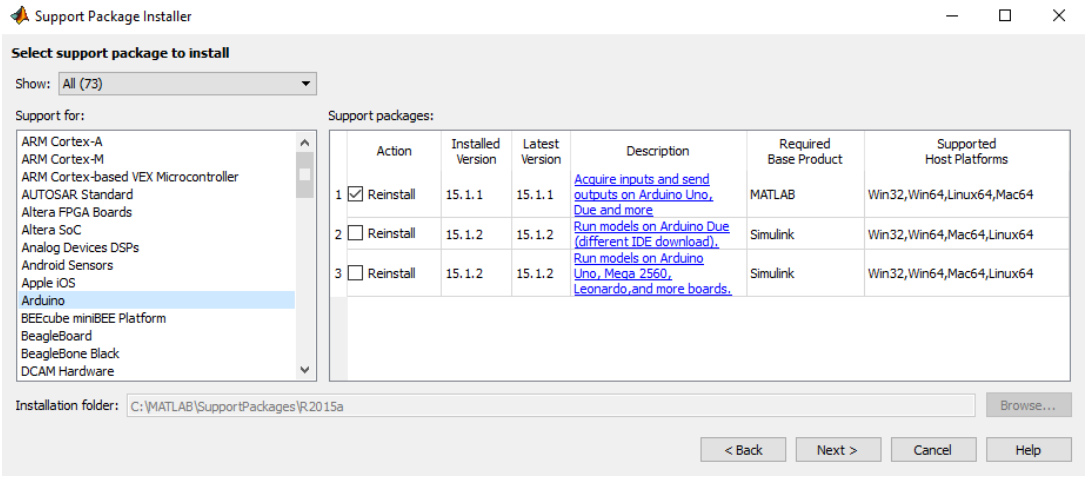


Figura 416: *Pestaña de diferentes paquetes de targetas de instalación para matlab.*

Aunque en el apartado de Arduino aparezcan tres archivos diferenciados es recomendable instalarlas todas para un buen funcionamiento de este. Una vez dado a "Next" solo te pedirá aceptar los términos de funcionamiento, y comenzará a

descargarse e instalarse solo. Es importante observar donde se va a guardar estos archivos instalados porque será necesario para el siguiente paso.

Una vez instalado, debemos poner en el directorio del Matlab la carpeta de este nuevo paquete, sino no podrá funcionar. Y ya estaremos listos para programar Arduino con Matlab.

En esta descarga hemos añadido un paquete en el sistema interno de Matlab, Simulink, por lo que tendremos unos bloques prediseñados listos para su utilización en una carpeta llamada "Simulink Support Package for Arduino Hardware" con tres subcarpetas: "Common", "Ethernet Shield" y "Wifi Shield". En nuestro caso utilizaremos la carpeta "Common" donde se encuentran como su nombre dice, los bloques comunes de arduino.

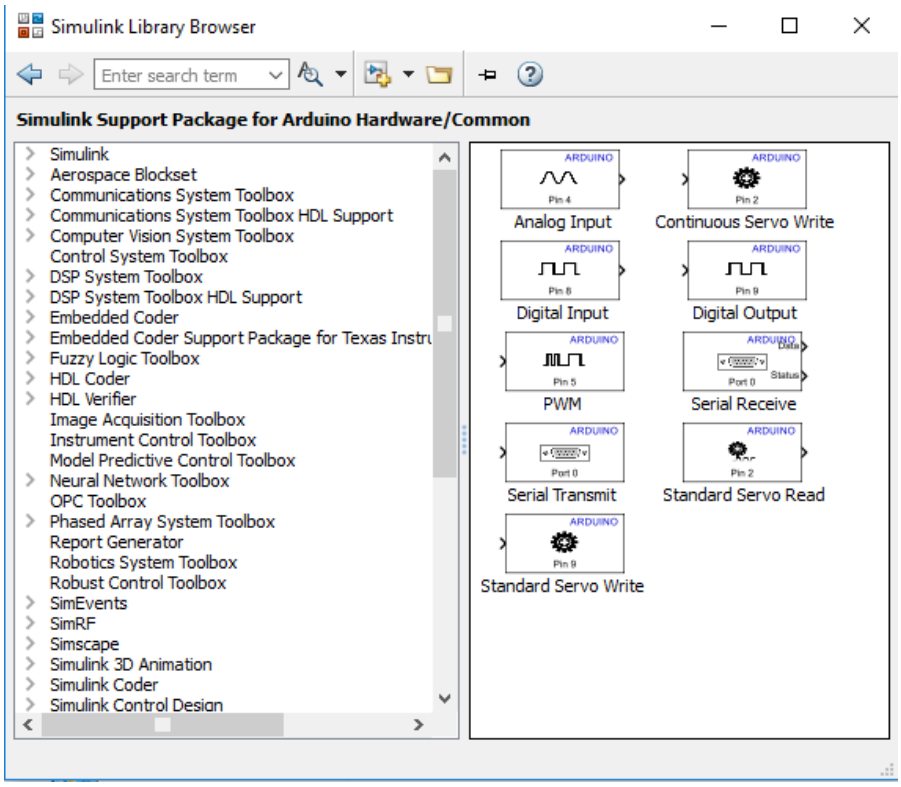


Figura 417: *Apartado de arduino de la librería de bloques de simulink.*

En todos estos bloques lo que debes especificar una vez que lo uses es el número de pin del arduino que vamos a comunicar la información. Se debe decir que existen más tipos de bloques descargables para arduino para versiones más recientes de Matlab.

Los siguientes bloques son:

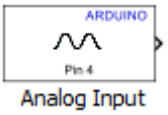
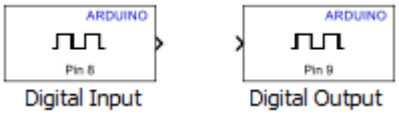
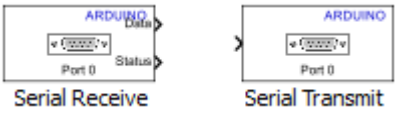
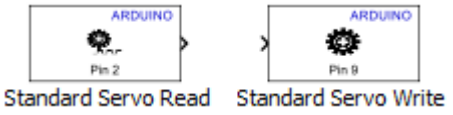
Analog Input	Para recibir señales analógicas del arduino. Se realiza en relación con el voltaje de referencia que tiene el pin analógico, sabiendo como hemos dicho anteriormente que tiene unos 10 bits, y por lo tanto el valor oscilara entre 0 y 1023. Si la tensión medida es igual a la tensión de tierra, dará un valor de 0, sin embargo si la medida es la tensión de referencia (5V) valdrá 1023.	
Digital input and output	Para recibir y enviar valores lógicos, si este valor es bajo, la entrada o salida será igual a 0, y si por el contrario si es alto, valdrá 1	
PWM	Este bloque permite una salida digital con una gama de diferentes niveles de potencia, similar a la de una salida analógica, con un rango valido de 0 a 255.	
Serial Receive or Transmit	Envia o recibe datos almacenados temporalmente al puerto sere expecíficado.	
Standard Servo Read or Write	Es un bloque más específico donde ajustará la dirección y la velocidad de un servo motor de rotación continúa. Siendo el valor de -90 el valor máximo de rotación en una dirección, 90 en la otra dirección, y 0 se detendrá. Pero sin embargo, este tipo de bloques no servirán en moto externo que es el que utilizaremos.	

Tabla 4: *Bloques de Simulink para Arduino*

4.3 Ensayos

En estos ensayos controlaremos los motores mencionados en un apartado 3 de materiales para los dos programas explicados.

4.4.1 Motor DC brushed

- *Arduino 1.8.1*

En primer lugar deberemos de realizar las conexiones adecuadas al ensayo, como se muestra en la figura 418 o la figura 419.

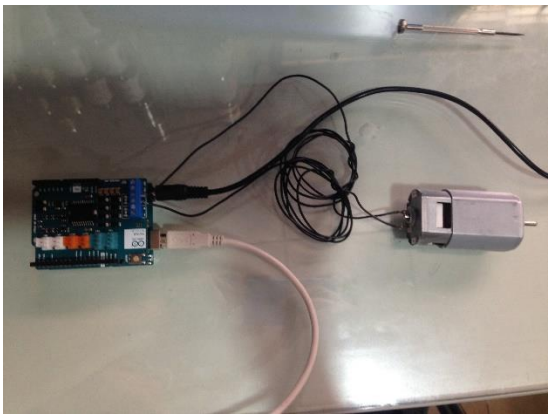


Figura 418: *Motor DC brushed controlado con un arduino Uno*

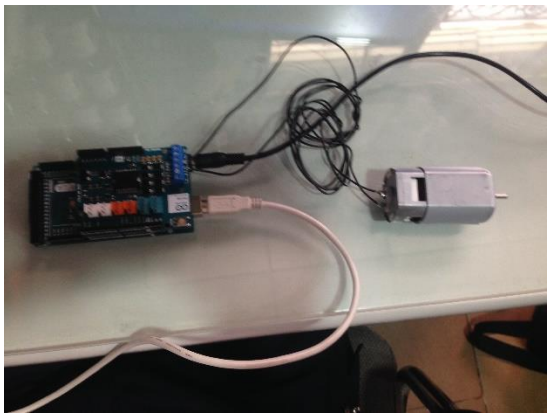


Figura 419: *Motor DC brushed controlado con un arduino Mega*

Deberemos colocar el Motor Shield encima del arduino, tanto el arduino Uno como el arduino Mega, teniendo en cuenta de encajar las clavijas correspondiente de ambos. A continuación conectaremos dos cables, independientemente de donde colocamos el positivo y el negativo del motor a los bornes azules del Motor Shield, ya que lo único que influirá será en la dirección de giro del motor una vez finalizado el ensayo, tal y como se muestra en la imagen. En nuestro ensayo utilizaremos los bornes A.

Da igual el tipo de arduino que escojamos entre los dos que tenemos para este ensayo, porque los dos funcionan exactamente igual sin necesidad de cambiar la programación. Aunque si debemos predeterminar el arduino que utilizamos en el programa como ya se explicó en la figura 319 en la barra de herramientas (6).

Una vez preparado todo lo mencionado, empezaremos a escribir la programación.

Haremos una combinación de comandos para que realice un movimiento giratorio a una velocidad igual a 100 de 255 durante unos cinco segundos, detendremos el motor un segundo, volveremos a activarlo a una velocidad de 50 de 255 durante otros 5 segundos y finalmente lo volveremos a detener un segundo más.

Todos estos valores se pondrán cambiar a voluntad de lo que se necesite y en tantos pasos como se plazca. Esta información se traspasará a la memoria del arduino, y se activará en bucle hasta que se cargue otro programa en él o mientras tenga

alimentación. A pesar de la desconexión del arduino de la alimentación externa, mantendrá la programación guardada en su memoria hasta que vuelva a alimentarse de nuevo. Esta se eliminará simplemente al sobrescribirse otro programa en su interior.

Por lo tanto las líneas de código escrito será el siguiente:

```
    //MOTOR DE UNA BOBINA
  //Variables
int vA=3; //Variable de la VELOCIDAD de la bobina A enviada al pin 3
int fA=9; //Variable de la FRENO de la bobina A enviada al pin 9
int dA=12; //Variable de la SENTIDO de la bobina A enviada al pin 12


void setup() {

  //Establecer los pines de la bobina a como pines de salida
pinMode(vA,OUTPUT);
pinMode(fA,OUTPUT);
pinMode(dA,OUTPUT);
}

void loop() {

    //Paso 1    Giro del motor a 100 sobre 255 durante 5 segundos en dirección positiva

  //Liberamos el freno
digitalWrite(fA,LOW);

  //Sentido positiva de la corriente
digitalWrite(dA,HIGH);

  //Recibimos el valor de velocidad del monitor.
analogWrite(vA,100);

  //Tiempo de espera de 5 segundos
delay(5000);


    //Paso 2   Parada de un segundo

  //Freno
digitalWrite(fA,HIGH);

  //tiempo de espera de 1 segundos
delay(1000);

    //Paso 3 Giro del motor a 50 sobre 255 durante 5 segundos en dirección negativa
```

```
  //Liberamos el freno
digitalWrite(fA,LOW);

  //Sentido negativa de la corriente en A
digitalWrite(dA,LOW);

  //velocidad
analogWrite(vA,50);

  //Tiempo de espera de 5 segundos
delay(5000);

    //Paso 4    Parada de un segundo

  //Freno
digitalWrite(fA,HIGH);

  //Tiempo de espera de 1 segundos
delay(1000);
}
```

Si decidimos incorporarle en los extremos del motor un osciloscopio, y capturamos la imagen, el resultado será el siguiente.

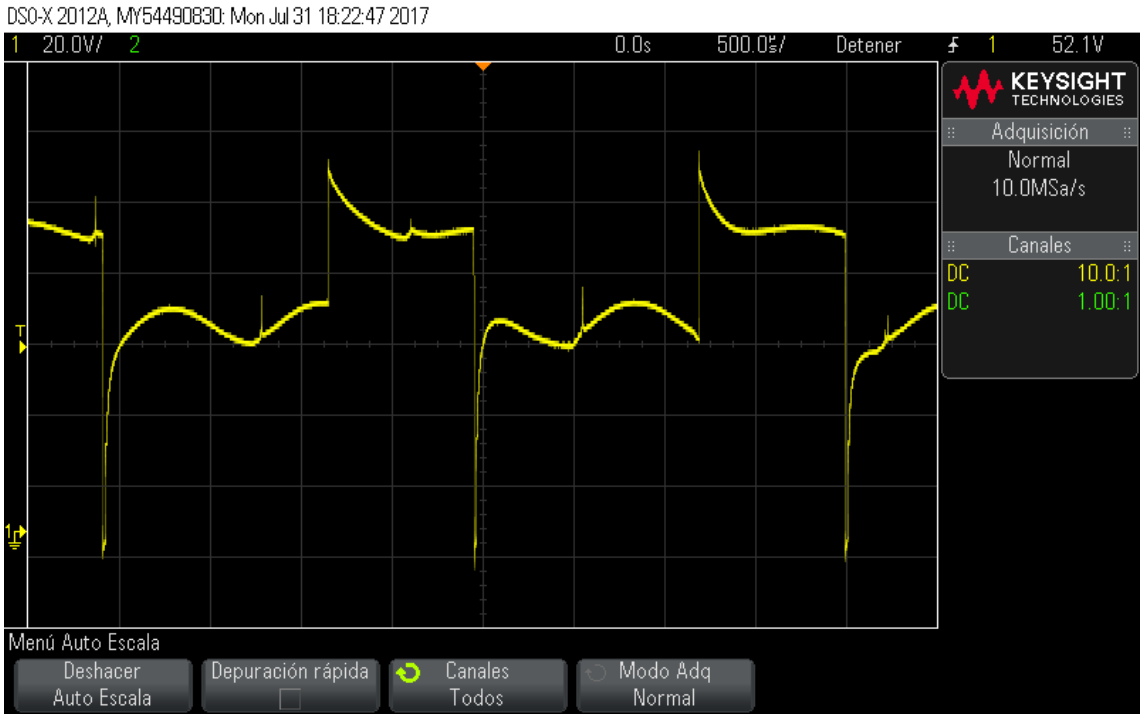


Figura 420: Capturación en el osciloscopio para motor con cepillado con arduino 1.8.1

Podríamos decir de esta imagen, que las variaciones de valor de tensión que realizamos en el motor son irregulares, no conseguimos los valores adecuados, sobre todos en los picos de bajada que son más pronunciados que los de subida.

También podemos añadirle una ampliación en el programa de manera que podamos modificar la velocidad a tiempo real al motor sin tener que cambiar la programación y subirla al arduino constantemente. Este es la ya llamada anteriormente comunicación serial.

Para utilizarla deberemos de añadir las siguientes líneas:

```
String val="";
```

Este comando es necesario para la creación de la variable que utilizaremos en la comunicación serial. En el cual, *val* será el nombre donde se guardará el valor.

```
Serial.begin(9600);
```

Necesario para iniciar la comunicación serial con unos 9600 baudios. Este se colocará en el interior del *void setup()* debido a que solo necesitamos que se ejecute una vez.

```
if(Serial.available()>0){

  val=Serial.readString();
  vel=val.toInt();
}
```

Gracias a estas líneas colocadas en el interior del *void loop ()* cuando reciba un valor mayor de 0, la escribirá en la variable establecida anteriormente *val* y guardara la parte entera de este valor en la variable *vel*.

Con esto, la línea de código se reducirá considerablemente, ya que solo tendremos que escribir un paso, que es la activación del motor a la velocidad guardada en *vel* por lo tanto deberemos establecer este un valor inicial antes de recibir el primer valor en el monitor serie de la siguiente manera.

```
  //MOTOR DE UNA BOBINA CON MONITOR
 //Variables
int vA=3; //Variable de la VELOCIDAD de la bobina A enviada al pin 3
int fA=9; //Variable de la FRENO de la bobina A enviada al pin 9
int dA=12; //Variable de la SENTIDO de la bobina A enviada al pin 12

String val="";  // escribe en val el valor del monitor
int vel=10;     // Valor de la velocidad


void setup() {

Serial.begin(9600);  //Abrimos comunicación serial con velocidad de 9600 bit por segundo
 //Establecer los pines de la bobina A como pines de salida
pinMode(vA,OUTPUT);
pinMode(fA,OUTPUT);
pinMode(dA,OUTPUT);
```

```
}

void loop() {

if(Serial.available()>0){

  val=Serial.readString();
  vel=val.toInt();
}   //Nos permite el cambio de valor de la variable de velocidad en el monitor serial

//Liberamos el freno
digitalWrite(fA,LOW);

//Sentido positiva de la corriente
digitalWrite(dA,HIGH);

//Recibimos el valor de velocidad del monitor.
analogWrite(vA,vel);
}
```

En este caso obtendremos la misma imagen en el osciloscopio, pero podrá variar su valor de velocidad escribiéndolo en el monitor serial (figura 421)

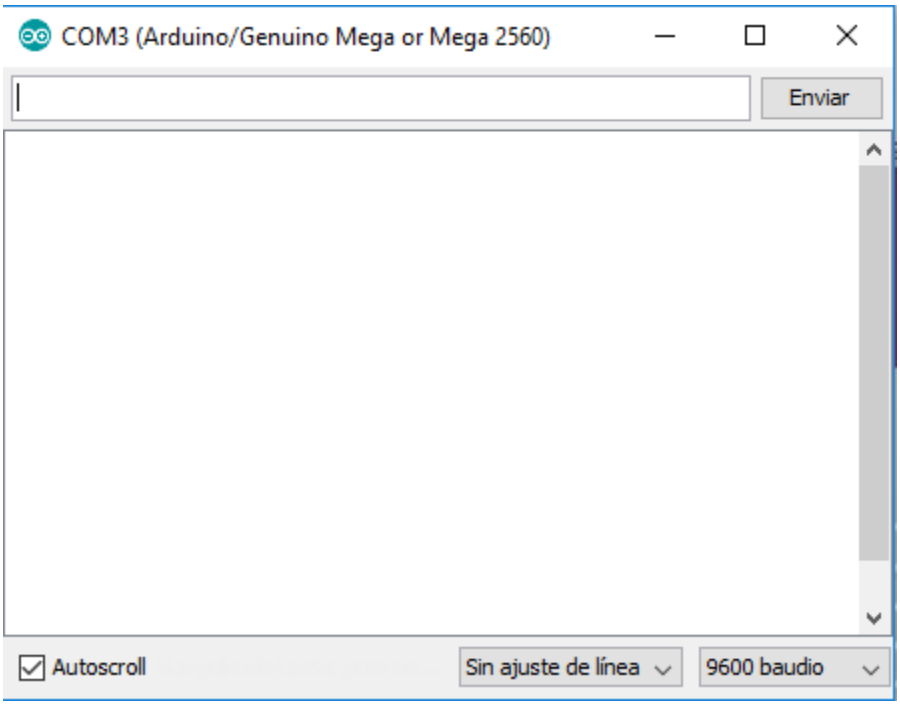


Figura 421: Ventana el monitor serial.

Se podría realizar el ensayo con dos de este tipo de motores simplemente conectando el otro motor en las otras borneras, en nuestro caso en el B tanto positivo como negativo como se muestra en la figura 422 y la figura 423 y añadiéndole la

programación adecuada, teniendo que definir las tres nuevas variables (freno, velocidad y dirección) y darle valores según se quiera.

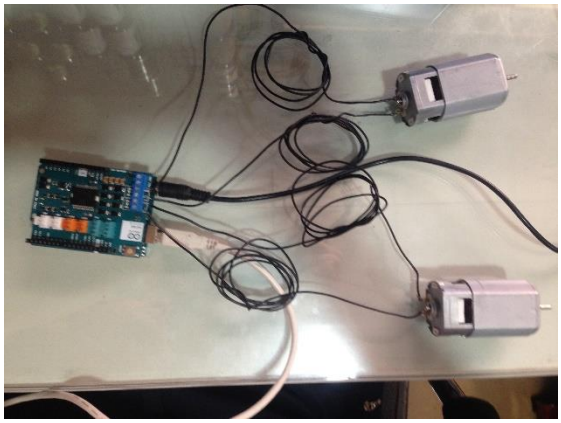


Figura 422: Dos m*otor sin cepillado de una bobina controlados con un arduino Uno*

Figura 423: Dos m*otores sin cepillado de una bobina controlados con un arduino Mega*

- *Matlab*

Ahora realizaremos esta misma práctica pero a través del programa simulink de Matlab

En primer lugar, en cuanto abrimos un nuevo modelo, lo que debemos hacer es especificar el hardware que se va a instalar en el ordenador donde vamos a enviar nuestro programa. En este caso en arduino, por lo tanto seguiremos los siguientes pasos "Tools -> Run on Target Hardware -> Prepare to Run…"

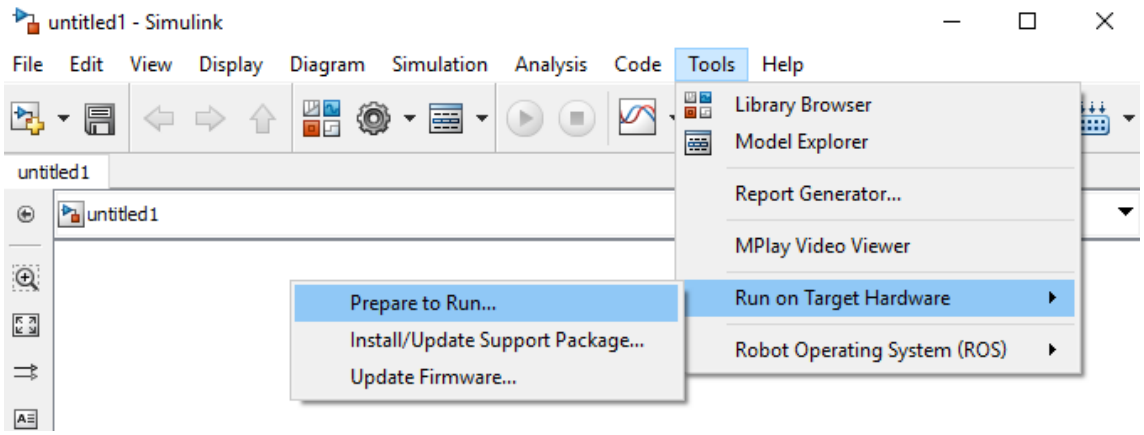


Figura 424: Pasos para la preparación de nuestro arduino en simulink

Después saldrá una pantalla emergente en la cual deberemos modificar los siguientes parámetros.

En "Run on Target Hardware" en el apartado "Target hardware selection" deberemos seleccionar nuestro arduino, en nuestro caso pondremos el arduino Mega

2560, debido a que el arduino Uno no nos deja realizar simulaciones externas, es decir, que no podremos hacer simulaciones y cambiar valores al instante y en tiempo real, aunque si nos deja cargar y guardar los programas en el arduino. Por ese motivo no lo haremos con el arduino Uno, debido a la lentitud de la carga del programa.

Una vez seleccionado el arduino que utilizamos, esperamos unos segundos a que los valores cambien. Como podremos ver en este programa el puerto COM se seleccionará solo automáticamente.

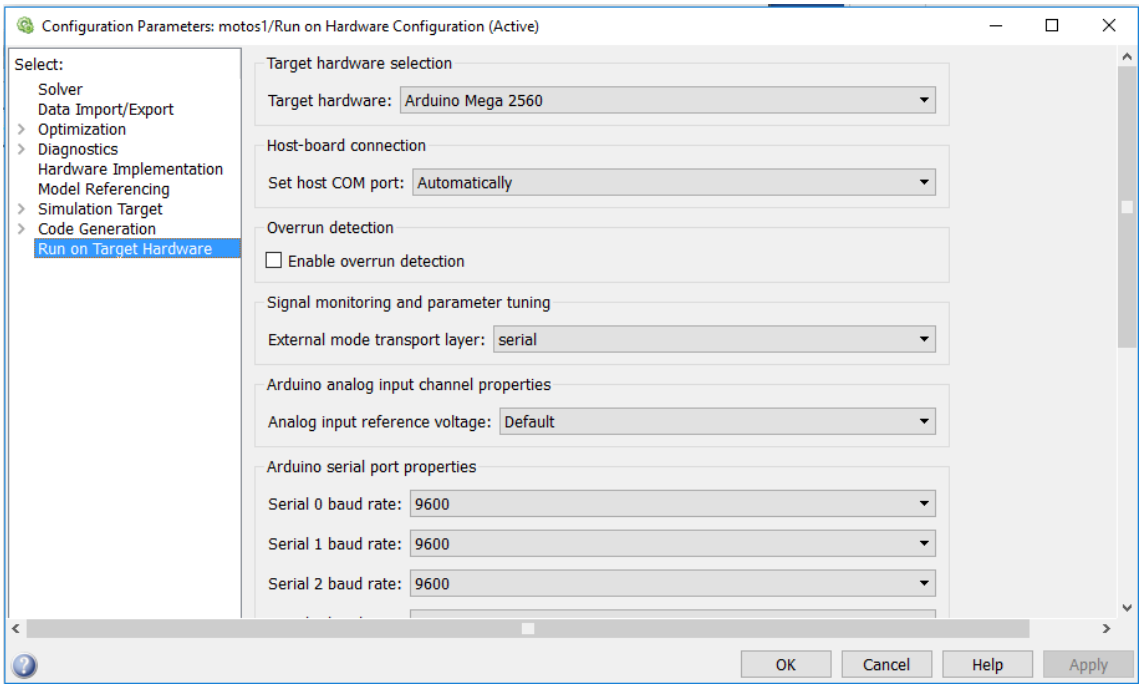


Figura 425: Pasos para la preparación de nuestro arduino. Configuración de parámetros del hardware.

A continuación en la misma ventana, le daremos a la sección de “Solver” y escribiremos en “Stop time” el valor de tiempo que debe parar nuestra simulación, nosotros pondremos “inf” para que la simulación no pare a no ser que nosotros la detendremos manualmente. Y además podremos realizar un cambio más en “solver options -> Solver:” ahí deberemos modificarlo y poner “discrete”, añadirle en “Fixed-step size” un tiempo de “10e-3”, y por último en “Tasking mode for periodic sample times” pondremos la opción de “SingleTasking”. Esto es debido a un gran número de pulsos que se deben de tener que realizar en un tiempo muy pequeño y esto no es capaz de hacerlo Matlab sin realizar este cambio. Por este motivo con este motor que estamos explicando y el servomotor, es opcional, pero es un campo obligatorio si queremos utilizar el motor paso a paso, sino se realiza, se llegará a enviar un valor constante al arduino y el motor no se moverá.

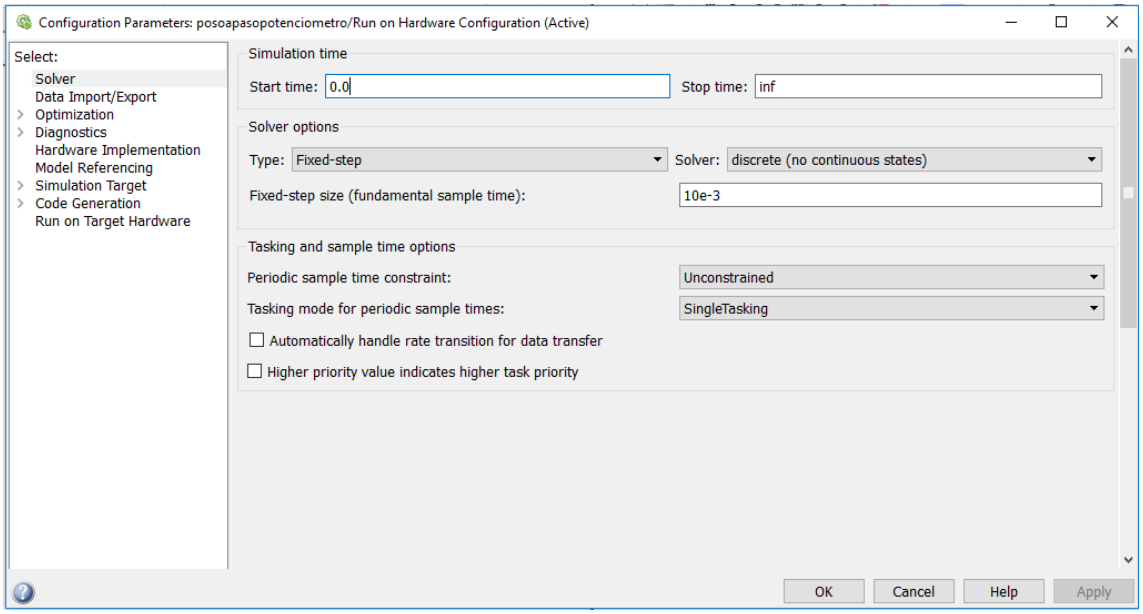


Figura 426: Pasos para la preparación de nuestro arduino. Configuración de parámetros del tiempo de muestreo.

Una vez realizada la configuración inicial, conectaremos un motor en los molex B como hemos explicado, desarrollaremos nuestro modelo de simulink, explicando lo que debemos de poner, de manera inicial serán los tres bloques que mandarán la información necesaria al motor a través del arduino, que serán dos bloques "Digital output" para el freno y la dirección, debido a que solo se recibirán señales de pulsos con valores de 1 o 0, y un bloque "PWM" para la velocidad porque recibirá valores entre 0 y 255. Si clicamos en cada uno de estos bloques, aparecerá una pantalla emergente que nos pide el pin del arduino en la que mandará la información que reciba. Debemos poner los pins correspondientes a donde hemos conectado nuestro motor según la tabla 2. Con este motor, lo único que deberemos hacer será añadirle una constante unida a cada uno de los bloques mencionados antes, y ponerles un valor que nos venga bien según lo que queremos hacer con el motor.

Como observamos en la figura 427 enviaremos un 0 al freno para desactivarlo, aunque realmente esto está predeterminado, por lo tanto no será necesario, para la dirección el 0 será un sentido, y 1 la contraria, y finalmente en la velocidad ponemos el valor que nos parezca en el rango establecido.

Si en vez de colocar un bloque PWM ponemos un bloque Digital, o simplemente, si enviamos un valor diferente de 0 y 1 a un bloque digital, se sobrecalentará el microcontrolador del arduino en pocos segundos.

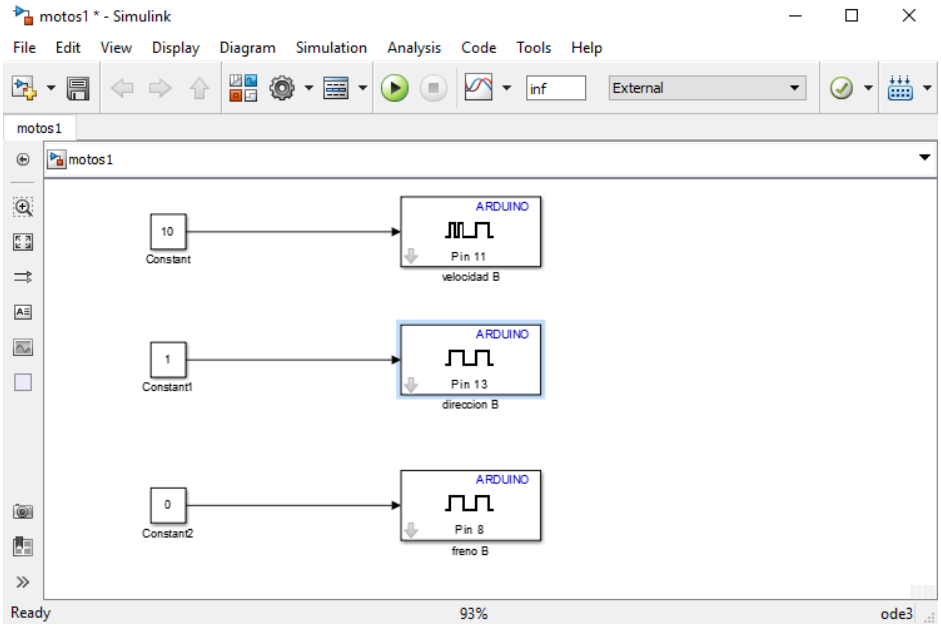


Figura 427: Modelo de simulink para motor de una bobina controlado por arduino.

Para poder enviar y simular el modelo, deberemos poner en la barra de herramientas la opción “External” donde antes aparecía “Normal”, y ya simplemente darle al símbolo de Play si es un arduino distinto al Uno, debido a que como ya hemos dicho, este no acepta modo externo. Si queremos utilizar el arduino Uno con simulink, deberemos darle al icono de desplegar el hardware con forma de cuadrado azul y tres flechas en la barra de herramientas de la figura anterior.

Ambos tipos de procesos tardarán un tiempo en ejecutarse, dependiendo del tipo de pc que tengamos.

Si lo hacemos con el modo “External” podremos modificar los valores de las constantes a la vez y verse este cambio a tiempo real. Y la señal del osciloscopio en este caso será de la siguiente manera.

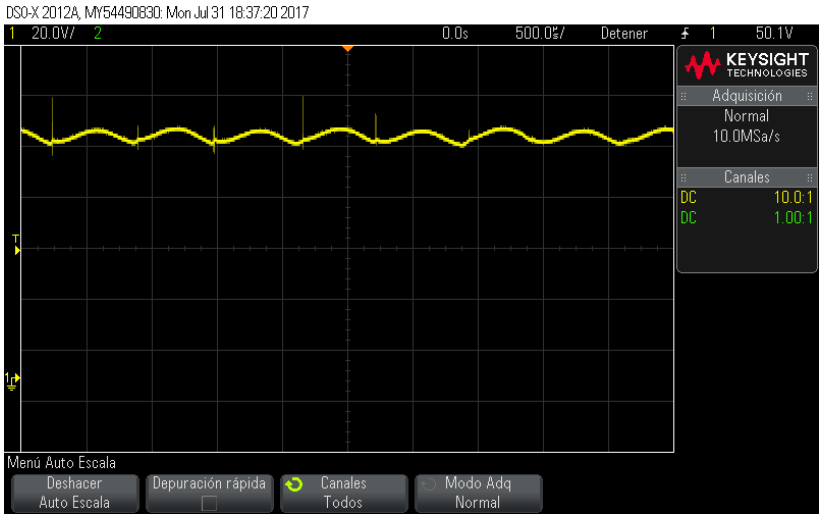


Figura 428: Señal del osciloscopio para un motor DC controlado por arduino con un programa de Matlab

Este tipo de ensayo se puede perfeccionar instalando una o varios potenciómetros para que controles los valores de estas constantes, como hemos realizado a continuación.

Nosotros añadiremos un potenciómetro en primer lugar en sustitución de la constante de la velocidad.

En nuestro potenciómetro tendremos que realizar unas conexiones previas al funcionamiento del mismo, conectando los extremos A y C en el pin de 5V y el GND que se localiza en la sección 7 de la figura 408 del arduino Uno y figura 409 del arduino Mega. Y en B conectaremos a cualquier pin analógico del arduino uno (del 0 al 5) o el arduino mega (del 0 al 15) en la sección 8 de las misma figuras nombradas antes. Este tipo de conexión quedara de la siguiente manera.

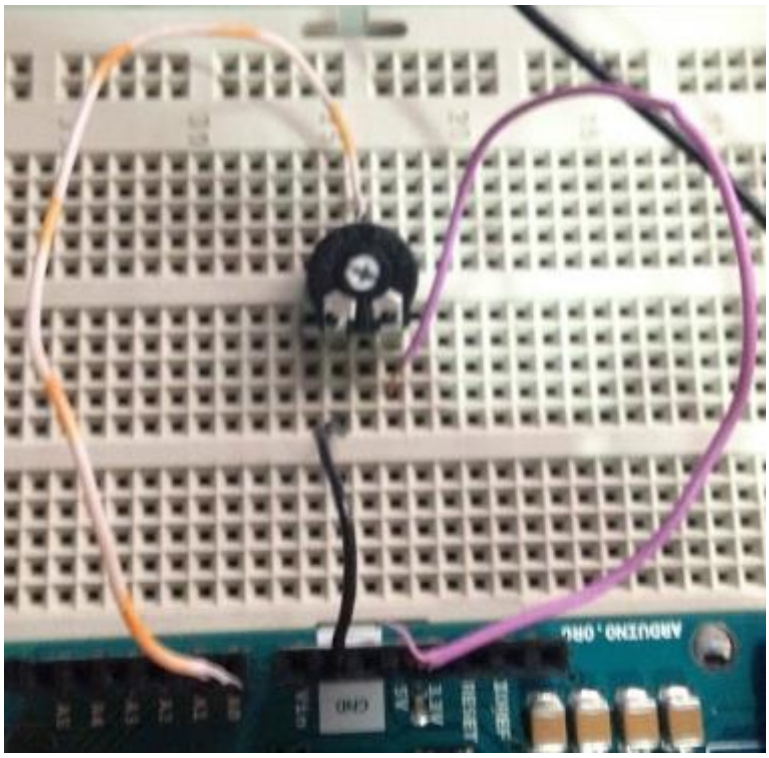


Figura 729: Conexión del potenciómetro al arduino.

Hay que decir que da igual los ohmios que tenga nuestro potenciómetro ya que lo que mediremos es la tensión que llegue a la patilla B, con una señal de 10 Bit. Por lo tanto para los 5V tendremos una señal de 1024, y para 0V una señal de 0.

Una vez conocida esta información, diseñamos nuestro modelo modificando nuestra constante, por tres componentes. El bloque “Analog Input” con el pin donde hemos conectado nuestro potenciómetro, un bloque de multiplicación que hará multiplicar nuestra señal entrante, por una constante, para poder modificar el rango que tenemos de 0-1024 a 0-255, para eso usaremos una constante con un valor igual a 0.219. También podremos sustituir el bloque de la multiplicación y el de la constante por un bloque de ganancia, que hace el papel de estos dos.

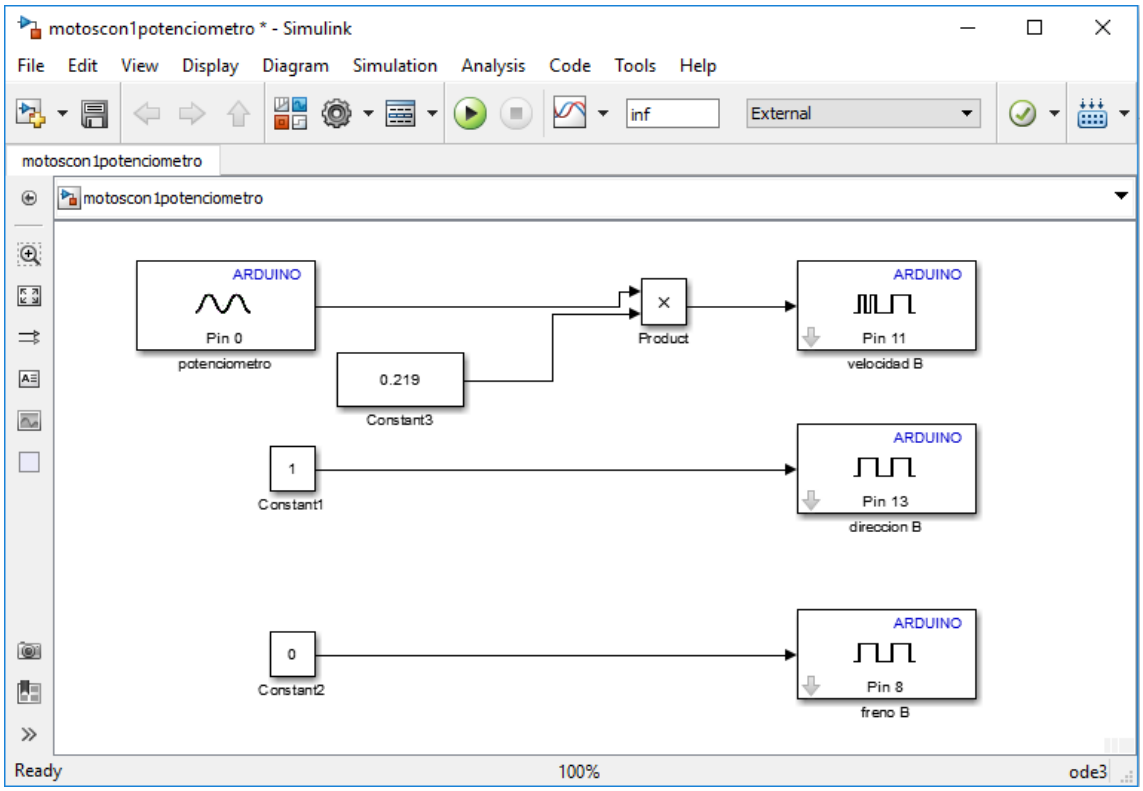


Figura 730: Modelo de simulink para motor de una bobina controlado por arduino y un potenciómetro externo.

Ahora añadiremos un segundo potenciómetro a cualquiera de nuestras dos constantes restantes, en nuestro caso la constante de la dirección. Ya que solo necesitamos dos valores, o un uno o un cero para enviar a nuestro pin, separaremos el potenciómetro en dos, entre 0 y 500 dará un valor de uno, y entre 501 y 1024 obtendremos un cero. Lo conseguiremos poniendo el mismo bloque anterior para recibir la señal del potenciómetro, además de una constante con valor de 500, y un nuevo bloque llamado “Relational Operator” en el cual comparará los dos valores, tanto el del potenciómetro, como el de la constante, y dependiendo de verificar la condición que imponemos en este bloque, enviará a nuestro pin o un uno o un cero tal y como se observa en la figura 431.

Este último potenciómetro funciona a la perfección, pero con el inconveniente de un pico de intensidad a la hora del cambio de dirección, debido al cambio brusco que se produce. La consecuencia de esto, es un salto del motor físicamente si este no está sujeto.

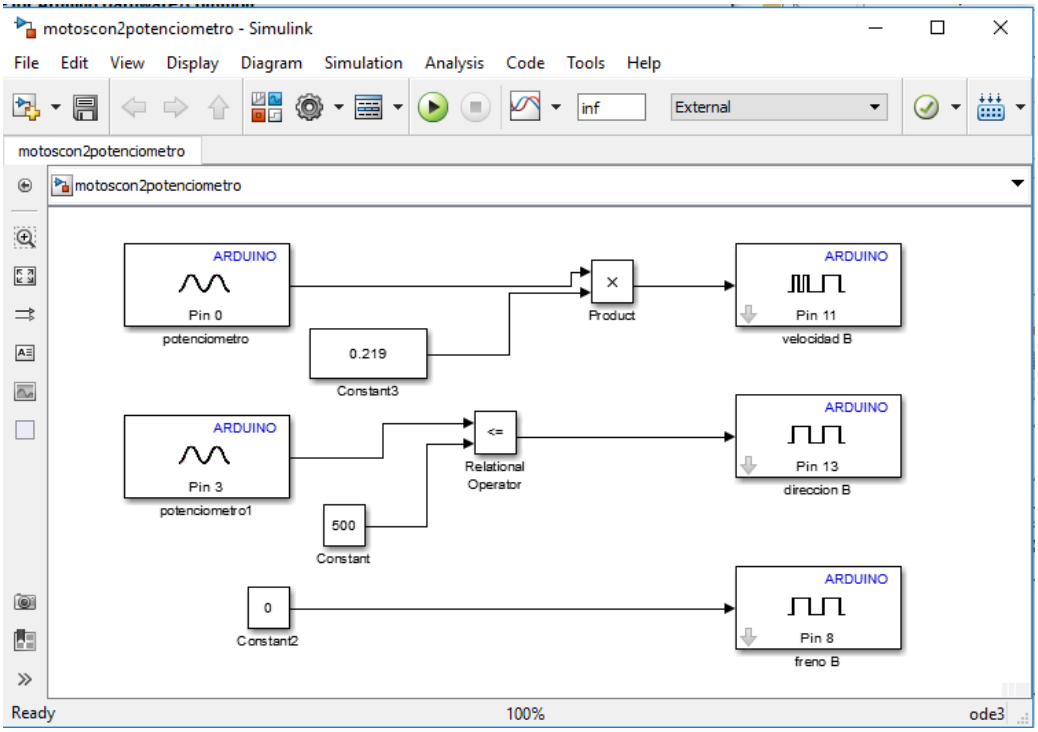


Figura 431: Modelo de simulink para motor de una bobina controlado por arduino y dos potenciómetros externos.

Y como observamos en la siguiente imagen, no se produce diferencias alguna observándolo desde el osciloscopio comparándolo con el experimento sin potenciómetros.

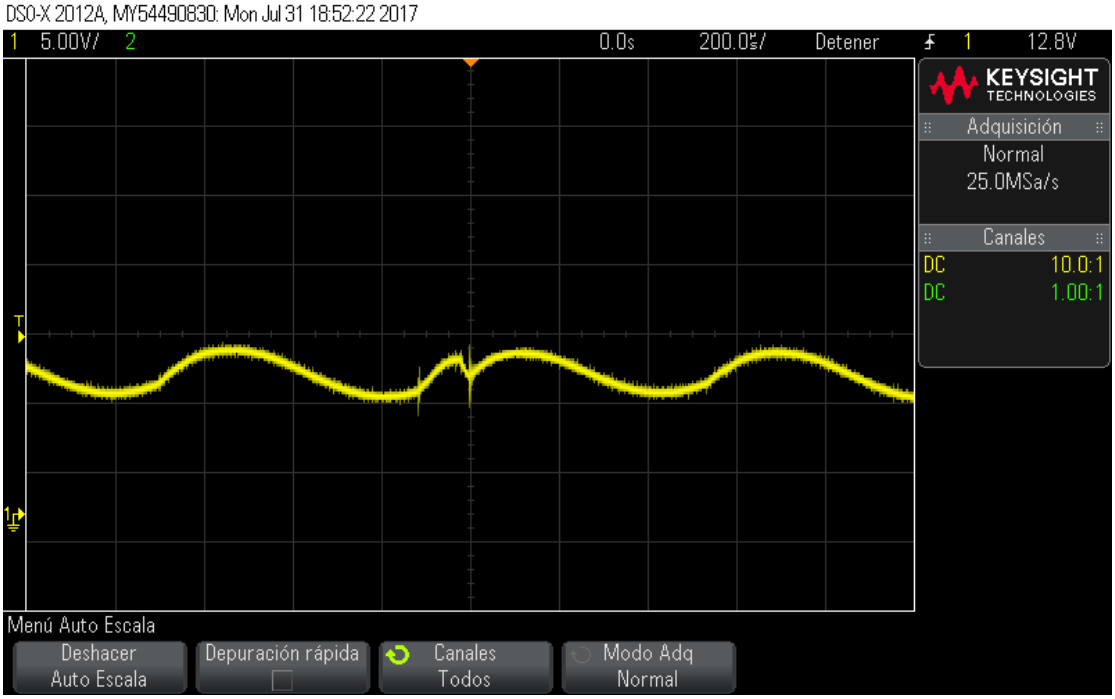


Figura 432: Señal del osciloscopio para un motor DC controlado por arduino con un programa de Matlab y con dos potenciómetros externos.

4.3.2 Motor paso a paso de dos bobinas

- *Arduino 1.8.1.*

En este caso tendremos que conectar los dos extremos de cada una de las bobinas a cada una de las conexiones en bornes de la siguiente manera y como se muestra en la figura 433 y 434.

Motor	Conexión
Bobina +1	Bornera +A
Bobina -1	Bornera –A
Bobina +2	Bornera +B
Bobina -2	Bornera –B

Tabla 5: *Conexión del motor paso a paso de dos bobinas*

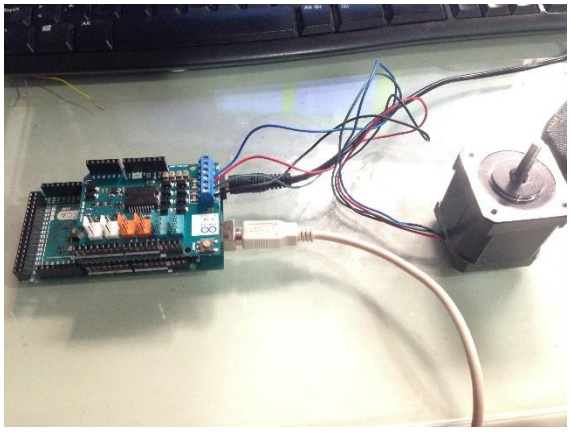


Figura 433: *Motor paso a paso controlado con un arduino Mega*

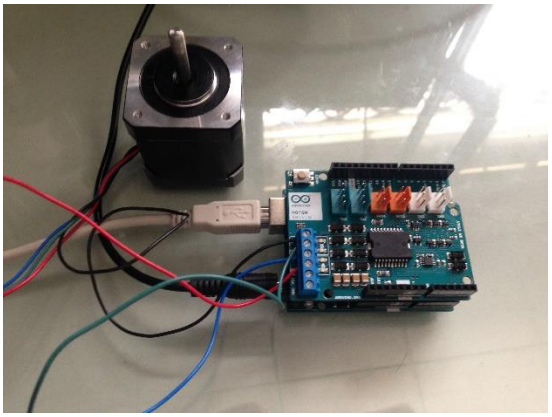


Figura 434: *Motor paso a paso controlado con un arduino Uno.*

Una vez explicado cómo funciona este tipo de motores en el apartado 3 de materiales necesitaremos que la programación realice el cambio de intensidad y de tensión según lo establecido. Para ellos daremos las órdenes a nuestro programa para que haga estas modificaciones en un periodo pequeño de tiempo, debido a que si reducimos este valor aumentaremos la velocidad. Por lo tanto los pines que estaban establecidos por el Motor Shield como velocidad esta vez no actuará sobre ella, sino por el par que se ejercerá sobre el motor, la dirección indicara el cambio de dirección de la corriente que necesitamos para que consiga funcionar, pero el freno si podrá funcionará como tal.

Utilizaremos como ya hemos dicho un paso simple, con unos cambios en la intensidad de las dos bobinas, tal y como se muestra en la figura 302.

Una vez conocido esta información, debemos adaptarlo a los tres pines de nuestra Shield (dirección, velocidad y freno).

- Freno

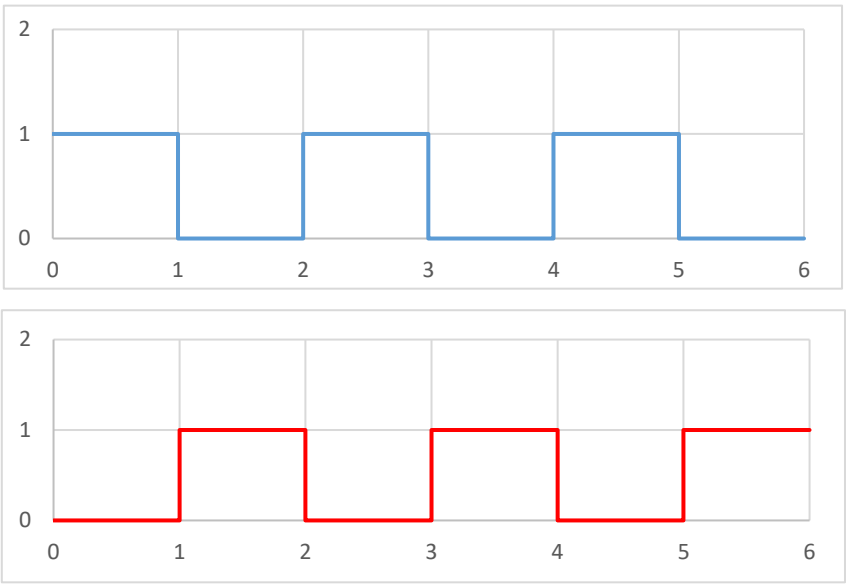


Figura 435: Señal enviada al pin de freno. Niveles de intensidad para la bobina A (Azul) y la bobina B (Rojo) en un eje X de valores de 10^{-2}

De esta manera tenemos los disparos de cada bobina en el momento determinado, y ahora es necesario enviar la mitad de estas señales en una dirección de la intensidad, y la otra mitad en dirección opuesta, tal y como se muestra en el siguiente gráfico:

Figura 436: Señal enviada al pin de dirección. Niveles de intensidad para la bobina A y B con valores de 10^{-2}

El valor que enviaremos por el pin de velocidad será constante a un valor establecido, en nuestro caso será igual a 100, aunque puede llegar a niveles de 255.
Este tiempo de disparo será el que deseamos hacer modificaciones, porque será el que hará variar la velocidad de nuestro motor.
Aquí demostramos como se realizaría la programación de nuestro motor paso a paso:

```
//MOTOR PASO A PASO
//Variables de la bobina A
int vA=3; //Variable de la VELOCIDAD de la bobina A enviada al pin 3
int fA=9; //Variable de la FRENO de la bobina A enviada al pin 9
int dA=12; //Variable de la SENTIDO de la bobina A enviada al pin 12
//Variable de la bobina B
int vB=11; //Variable de la VELOCIDAD de la bobina B enviada al pin 3
int fB=8; //Variable de la FRENO de la bobina B enviada al pin 8
int dB=13; //Variable de la SENTIDO de la bobina B enviada al pin 13
```

```
String val=""; // Escribe en val el valor del monitor
int vel=10;     // Valor de la velocidad


void setup() {

  Serial.begin(9600); //Abrimos comunicación serial con velocidad de 9600 bit por segundo
  //Establecer los pines de la bobina A como pines de salida
pinMode(vA,OUTPUT);
pinMode(fA,OUTPUT);
pinMode(dA,OUTPUT);

  //Establecer los pines de la bobina B como pines de salida
pinMode(vB,OUTPUT);
pinMode(fB,OUTPUT);
pinMode(dB,OUTPUT);
}

void loop() {

if(Serial.available()>0){

   val=Serial.readString();
   vel=val.toInt();
} //Nos permite el cambio de valor de la variable de velocidad en el monitor serial

    //Paso 1

  //Liberamos A y frenamos B
digitalWrite(fA,LOW);
digitalWrite(fB,HIGH);

  //Dirección positiva de la corriente en A
digitalWrite(dA,HIGH);

  //velocidad del A
analogWrite(vA,100);

  //Tiempo de espera
delay(vel);

     //Paso 2

  //Liberamos B y frenamos A
digitalWrite(fA,HIGH);
digitalWrite(fB,LOW);
```

```
  //Dirección negativa de la corriente en B
digitalWrite(dB,LOW);

  //velocidad del A
analogWrite(vB,100);

  //Tiempo de espera
delay(vel);

   //Paso 3

  //Liberamos A y frenamos B
digitalWrite(fA,LOW);
digitalWrite(fB,HIGH);

  //Dirección negativa de la corriente en A
digitalWrite(dA,LOW);

  //velocidad del A
analogWrite(vA,100);

  //Tiempo de espera
delay(vel);

   //Paso 4

  //Liberamos B y frenamos A
digitalWrite(fA,HIGH);
digitalWrite(fB,LOW);

  //Dirección positiva de la corriente en B
digitalWrite(dB,HIGH);

  //velocidad del A
analogWrite(vB,100);

  //Tiempo de espera
delay(vel);
}
```

Observamos que colocaremos un tiempo inicial de 10 milisegundos ya que aún no habría recibido ningún valor del monitor serial.

Una vez realizada esta simulación, podremos apreciar en el osciloscopio la siguiente imagen:

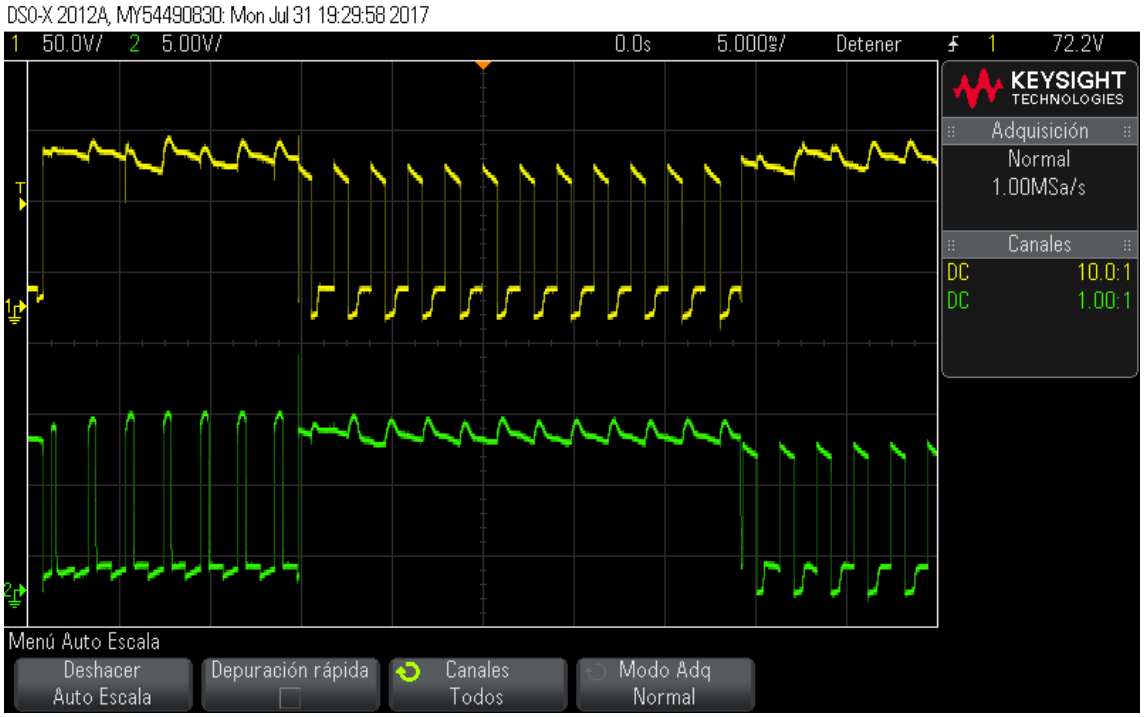


Figura 437: Imagen del osciloscopio para un motor paso a paso conectado al arduino.

Como podemos comprobar la señal que tenemos no es muy precisa a la que nosotros necesitamos para el motor, pero sin embargo, el motor gira perfectamente, y a distintas velocidades. Estas variaciones es lo que hemos estado explicando anteriormente a nuestra señal duty, debido a que tenemos un valor de velocidad de 100, tenemos un duty de 39,21. Si nosotros modificamos el valor de la velocidad y ponemos 255, esta gráfica si sería casi perfecta e igual a la figura 435.

Además lo que hemos podido hacer con este ensayo es añadirle un potenciómetro al arduino tal como lo hicimos en el ensayo anterior, para que este reciba un valor determinado, y así modificarse el valor del tiempo espera y de esta manera modifique la velocidad del motor de modo externo. La instalación será igual que antes, tal y como se muestra en las figuras.

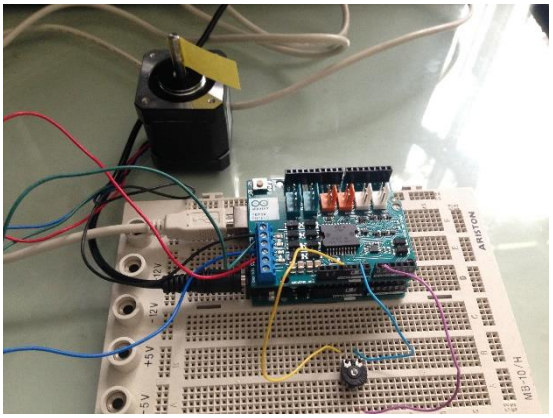


Figura 438: *Motor paso a paso controlado con un*

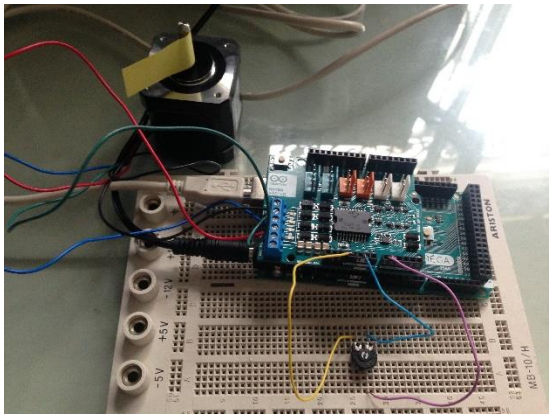


Figura 439: *Motor paso a paso controlado con un*

arduino Uno y un potenciómetro. *arduino Mega y un potenciómetro.*

El valor que recoja el potenciómetro lo guardaremos en una variable con esta función dentro del “void loop”.

valor=analogRead(A0);

En este caso podremos añadirle esta A de analógico de forma opcional a continuación del número de pin que introduciremos nuestro potenciómetro.

Por lo tanto, con un simple factor de conversión podremos convertir este rango de 0 a 1024 al rango de 5 a 50 milisegundos, que un rango adecuado para el buen funcionamiento del motor, utilizando la siguiente función:

vel=map(valor,0,1023,5,50);

Además hemos querido que este valor del potenciómetro se muestre en nuestra pantalla del ordenador utilizando la comunicación y el siguiente comando.

Serial.println(vel);

La “*ln*” de nuestra palabra “*println”* es opcional, lo que hará será colocar un valor debajo de otro y no uno a continuación del siguiente lo que nos podría producir confusión a la hora de leerlo en nuestro monitor serie. Aunque aquí podremos utilizar también nuestro Serial Plotter lo cual nos dará en una gráfica nuestro valor frente al tiempo de simulación en tiempo real. Tal y como se muestra en la siguiente figura 440.

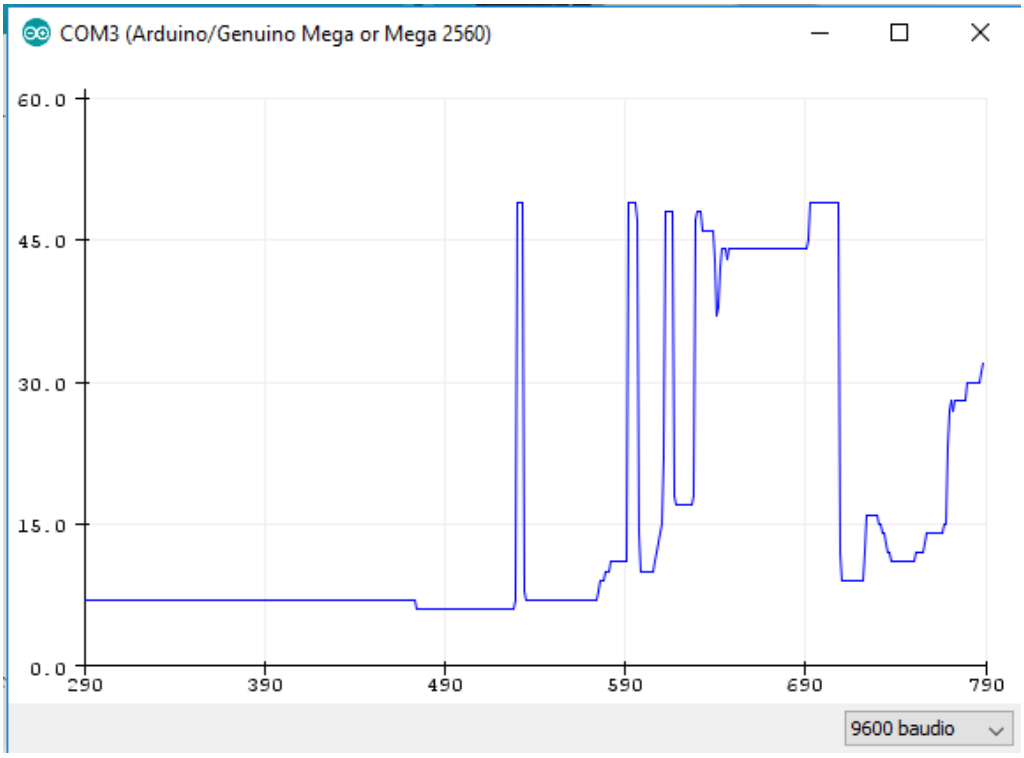


Figura 440: Gráfico que muestra el valor de la velocidad enviada de forma gráfica.

- *Matlab*

Ahora realizaremos el ensayo de un motor paso a paso con el programa Simulink de Matlab.

En primer lugar, se deben de realizar las mismas configuraciones previas que se realizaron con el motor anterior.

Para que funcione correctamente el motor, se debe de enviar a cada pin correspondiente del arduino sus valores adecuados, para ello, en primer lugar deberemos de poner cuatro bloques "Digital Output" para los pines del freno A y B (9 y 8) y los pines de la dirección A y B (12 y 13) en los cuales, como ya hemos dicho antes, solo enviaremos pulsos entre cero y uno. Y dos bloques más para la velocidad de A y B (3 y 11) con valores entre 0 y 255.

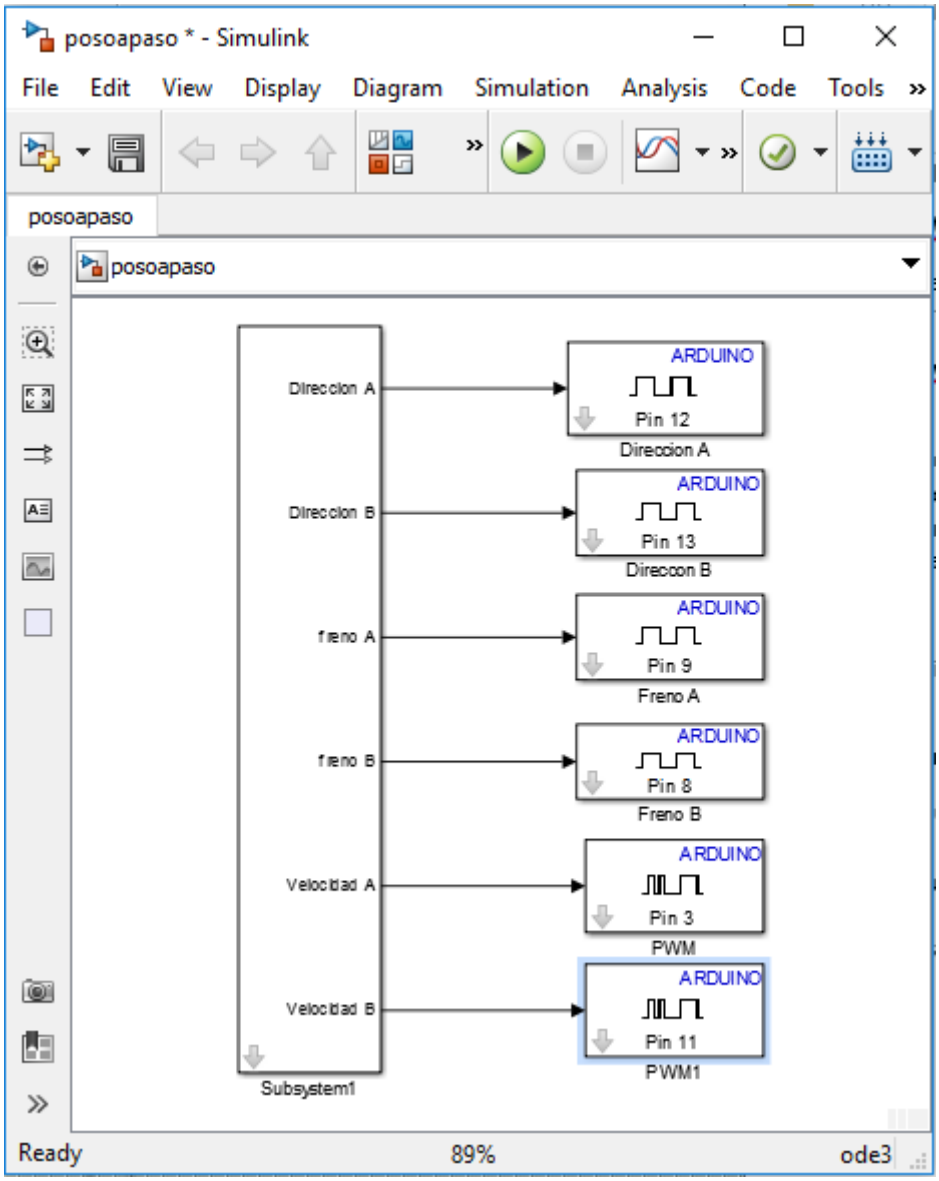


Figura 441: Modelo de simulink para motor paso a paso de dos bobinas controlado por arduino.

Crearemos un subsistema dentro de este modelo, para poder crear una máscara con un parámetro, que será nuestro tiempo de ancho de pulso, gracias a él podremos cambiar la velocidad del mismo simplemente clicando en el subsistema y modificando el valor en la siguiente pantalla.

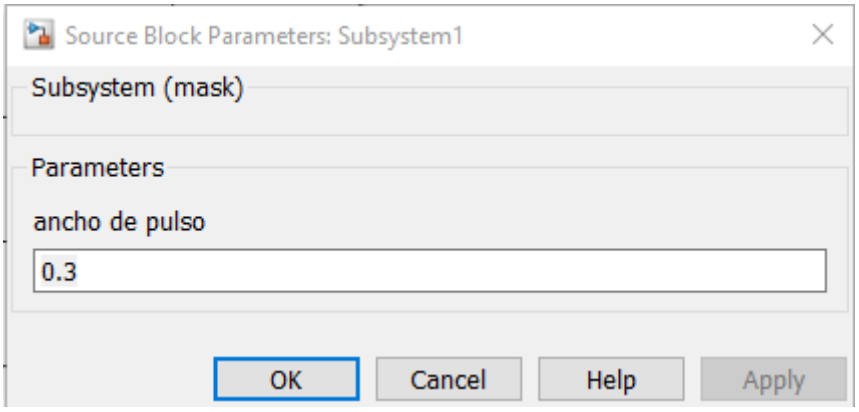


Figura 442: Ventana emergente del subsistema para modificar un parámetro.

Para poder crear un subsistema, lo que debemos hacer es seleccionar todos los bloques que deben de quedar en su interior, clicar botón derecho, y seleccionar "Create Subsystem from Selection" o dándole a Ctrl+G. Y para crear una máscara, clicaremos de nuevo con el botón derecho encima del subsistema y seleccionamos la opción "Mask -> Create Mask" o dándole a Ctrl+M. Nos aparecerá una ventana emergente, en la cual deberemos darle a la pestaña "Parameters & Dialog" (1) y después en controles le damos a "Edit" (2) de esta manera aparecerá un nuevo parámetro (3), el cual deberemos de darle nuestras características, como nombre pondremos "ancho de pulso" y llamaremos a la variable "T"

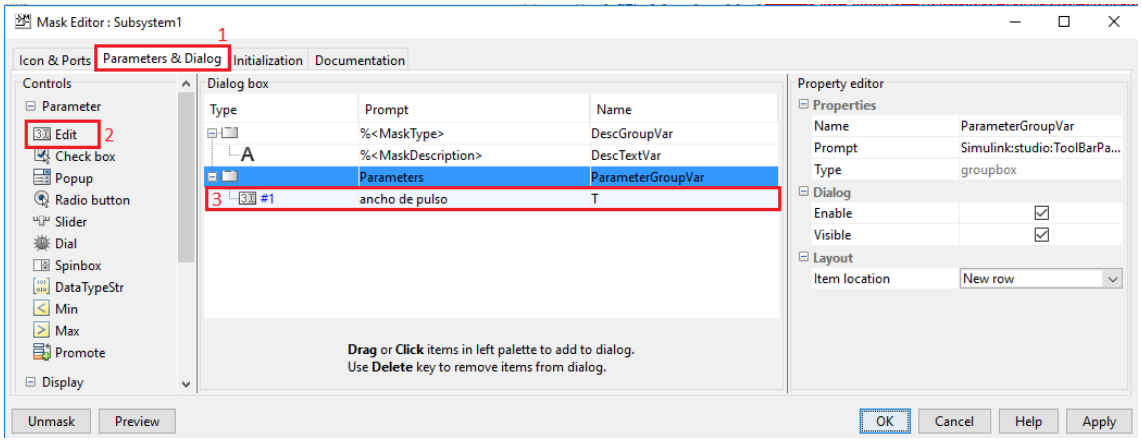


Figura 443: Ventana emergente del subsistema para crear un parámetro en la máscara.

Ahora nos dedicaremos a crear las órdenes necesarias para simular unos gráficos como la figura 435 y la figura 436.

Para ello, podremos poner simplemente un bloque de generador de pulsos para cada uno de los pines. Para la dirección debido a que el pulso va a ser el mismo para la bobina A como la B, nos bastara por un generador único con las siguientes características.

Amplitud	1
Periodo	2*T
Ancho de pulso	La mitad del periodo, por lo tanto 50%

Tabla 6: *Valores para el generador de pulsos de la dirección*

Para el freno, si necesitaremos dos generadores, ambos opuestos entre ellos, con las siguientes características.

	A	B
Amplitud	1	1
Periodo	T	T
Ancho de pulso	50%	50%
Retardo de fase	0	T/2

Tabla 7: *Valores para el generador de pulsos del freno*

Y por último los dos pines de la velocidad le daremos un valor constante de 100.

Dentro del subsistema quedara de la siguiente manera:

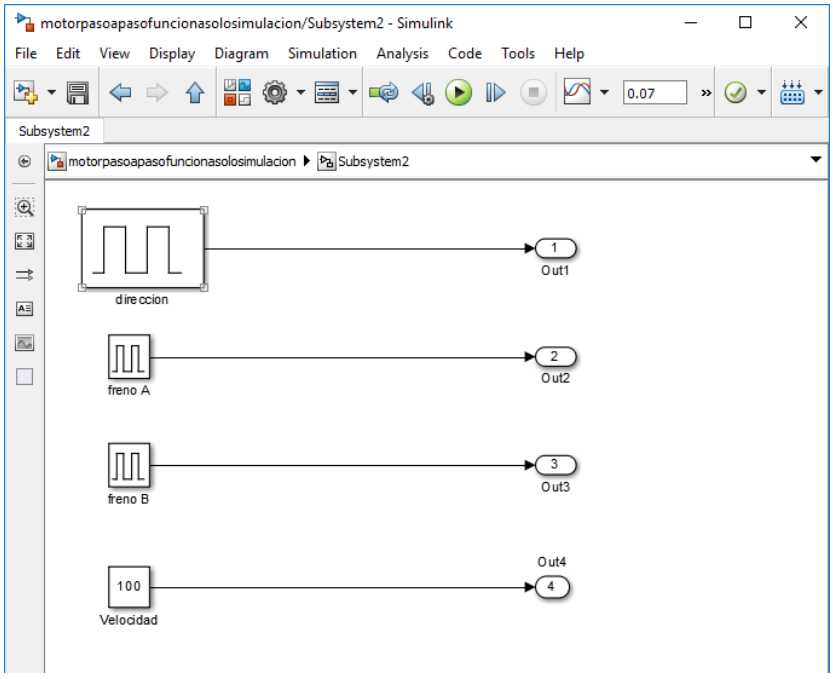


Figura 444: Modelo del subsistema con generadores de pulsos.

Luego podemos realizar este mismo sistema de forma parecida al segundo potenciómetro que añadimos en el motor anterior.

En este caso utilizaremos un bloque llamado “Repeating Sequence”, el cual dará unos pulsos con pendientes de subida. Le daremos unos valores para la dirección de la siguiente tabla.

Valor de tiempo	[0 T]
Valor de salida	[0 2]

Tabla 8: *Valores para el repetidor de la dirección*

Y para el freno daremos los siguientes valores.

Valor de tiempo	[0 T]
Valor de salida	[0 2]

Tabla 9: *Valores para el repetidor de la velocidad*

Tendremos que añadir un “Relational Operator”, comparándolo esta vez con una constante de valor la unidad, de esta manera podemos dar un pulso para la mitad del periodo “2T” para la dirección, y dos “Relational Operator” más, uno que compare con el mayor, y otro con el menos. Para que finalmente quede de la siguiente manera:

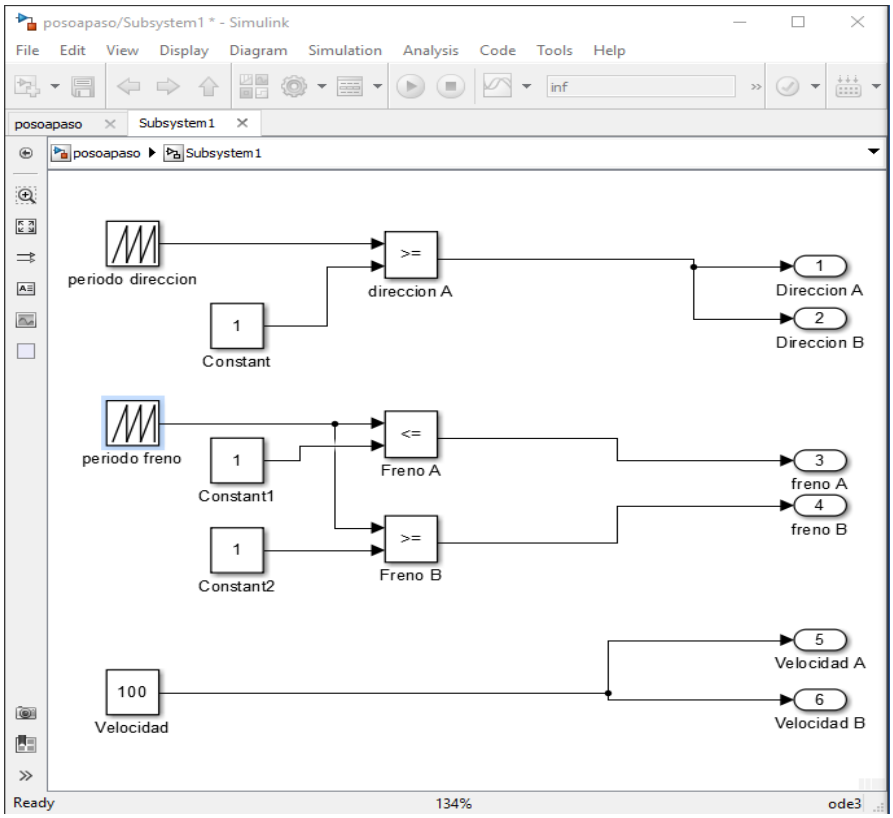


Figura 445: Modelo del subsistema con repetidor de secuencias.

Si colocamos un “scope” para que analice la señal de estas tres salidas del subsistema, dará el siguiente gráfico.

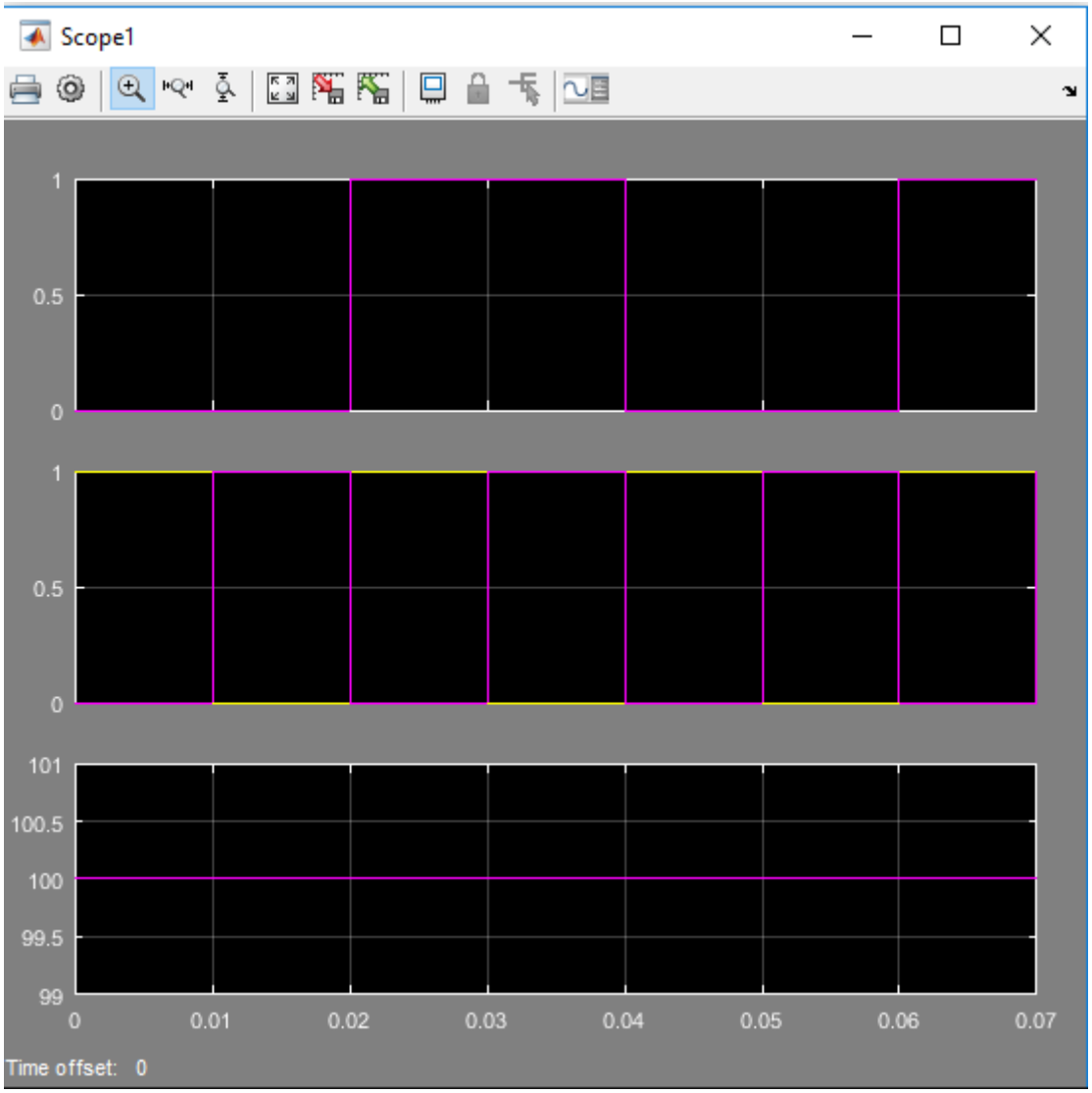


Figura 446: Señal ideal enviada al arduino

Como observamos, la señal es idéntica a la que necesitamos para el arduino, pero debido a tantas repeticiones en tan poco tiempo, la señal no será uniforme, aunque nos sirve y es adecuada para el motor.

Por lo tanto la señal real que enviamos al arduino, será la siguiente.

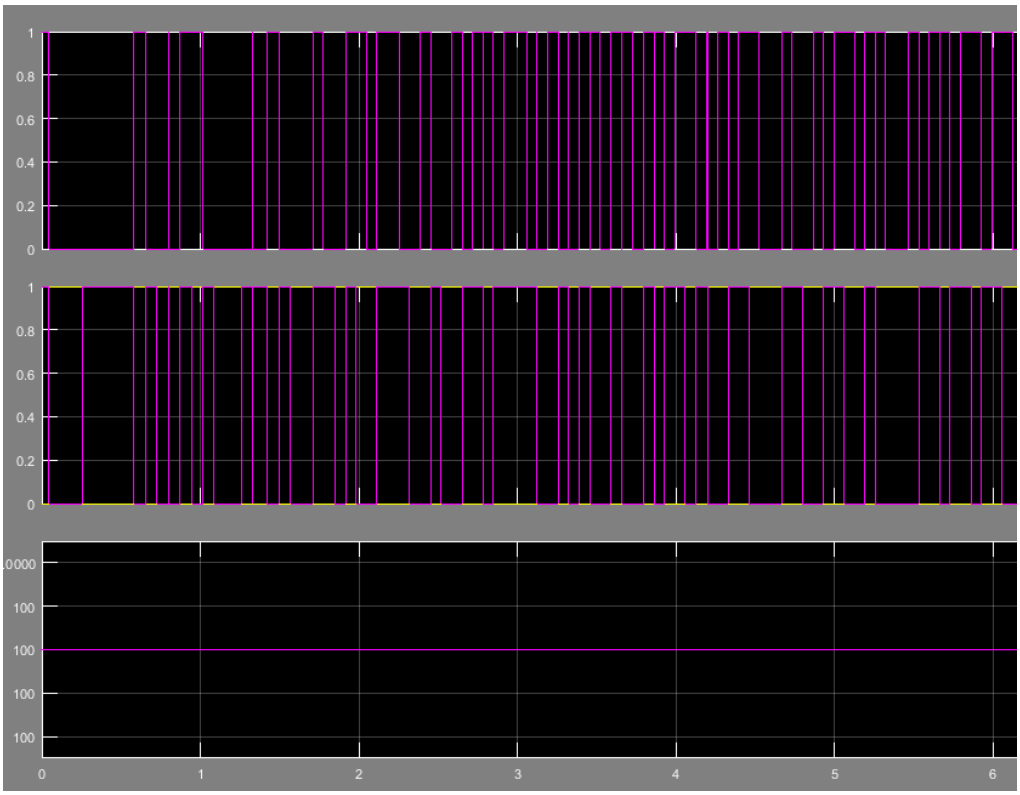


Figura 447: Señal real enviada al arduino

Observamos que esta real no es la correcta, pero funciona perfectamente, y el motor gira adecuadamente, esto es debido a que el programa no es capaz de asimilar la gran velocidad de pulsos que se envían.

Y la señal real recogida por el osciloscopio.

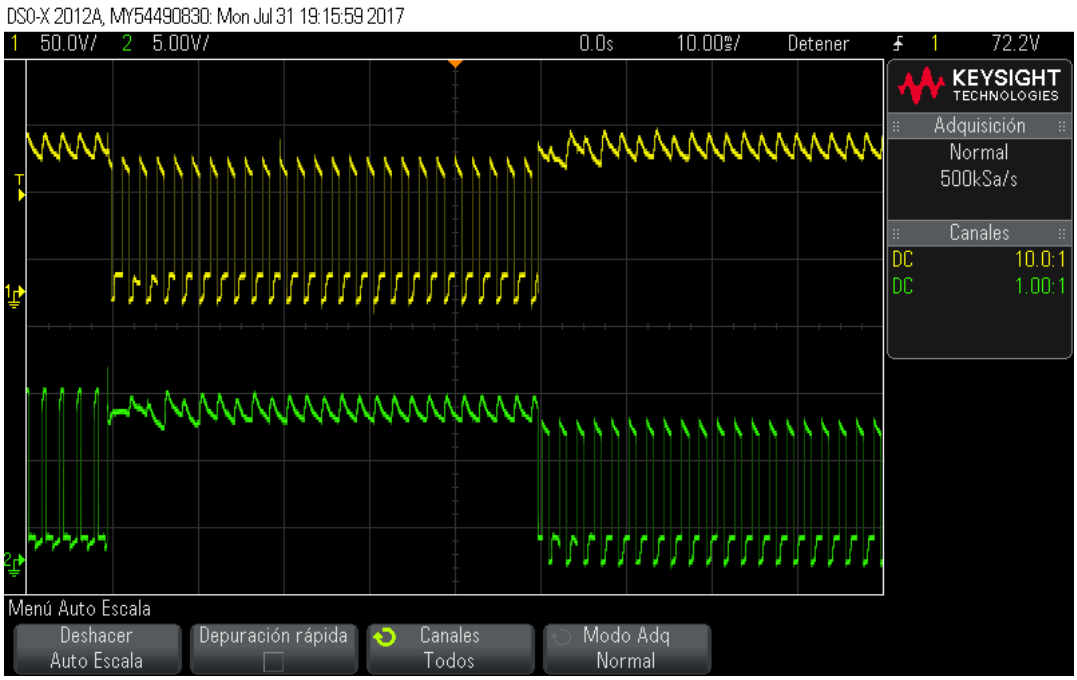


Figura 448: Señal del osciloscopio para un motor paso a paso con arduino y Simulink.

El cambio realizado del bloque del generador de pulsos al bloque del repetidor de secuencias se debe sencillamente, a que este bloque se puede modificar en su interior para poder acoplarle un potenciómetro como en casos anteriores, pero sin embargo el generador de pulsos no.

Por lo tanto, en esta ocasión, vamos a modificar el subsistema para poder cambiar el tiempo de ancho de pulso a través de un osciloscopio.

Deberemos de cambiar las constantes con la unidad que teníamos por un "Analog Input" con el pin adecuado, y una ganancia que haga modificar el rango de la señal de entre 0 y 1025 a 0 y 0.02, por lo tanto, la ganancia tendrá un valor de 0.00001955. Además deberemos añadir un bloque llamado "Goto" que nos servirá para copiar el valor de la señal del potenciómetro y enviarla a otro lado. En su interior deberemos modificar su visibilidad y ponerlo en modo global. La conexión que tenemos con la parte del freno se necesitara una ganancia que divida la señal del potenciómetro entre dos antes de realizar la comparación.

Debemos crear un nuevo subsistema con una máscara exactamente igual al bloque "Repeating Sequence", clicamos con el botón derecho encima de este bloque y le damos a "Mask -> Look Under Mask" de esta manera, observaremos que tiene en el interior de este bloque. Ya simplemente habrá que copiarlo y pegarlo en nuestro nuevo subsistema, debido a que no nos deja modificar el bloque. Y por último, añadiremos una máscara idéntica a la que tiene el bloque.

Las modificaciones que realizaremos en este nuevo subsistema comparado con el bloque real serán las siguientes.

Lo que necesitamos es conseguir que el valor T este en función de nuestro "Analog Input" para ello en primer lugar, debemos editar los parámetros de su máscara como "[0 1]" tanto para el valor del tiempo, como para valor de salida. A continuación dentro del subsistema observamos que hay una constante con el nombre de "period" en el cual averiguamos que el parámetro que sale de la constante es el valor del tiempo de la máscara, en nuestro caso entre 0 y 1, por lo tanto lo que hacemos será multiplicar esa constante con la señal del potenciómetro, que estará formado por el bloque "From" que lo que hace será recibir la señal del bloque "goto", a continuación de la misma ganancia que antes.

Hay que crear dos como estos, con la única diferencia de añadirle en uno de ellos otra ganancia que multiplique el periodo por 2, debido a que el periodo de la dirección es el doblo que el periodo del freno.

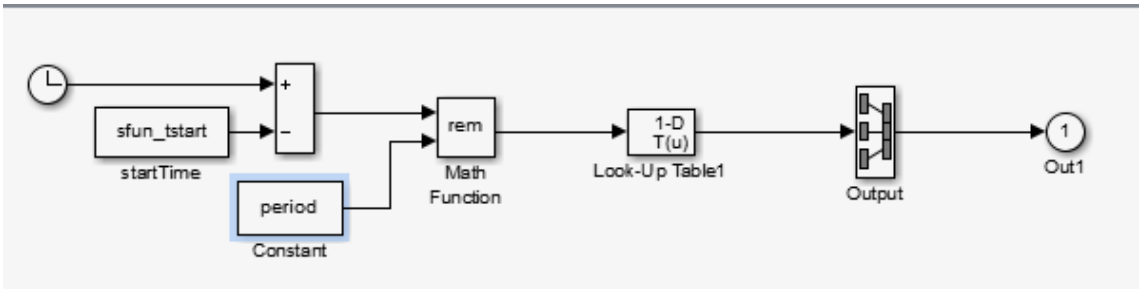


Figura 449: Contenido original de un bloque "Repeating Sequence"

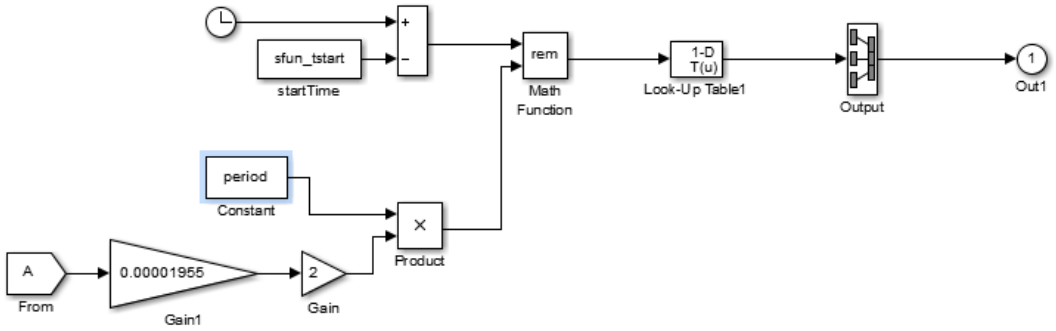


Figura 450: Contenido con la modificación de un bloque "Repeating Sequence"

Por lo tanto el primer subsistema se quedara después de todas estas modificaciones de esta manera.

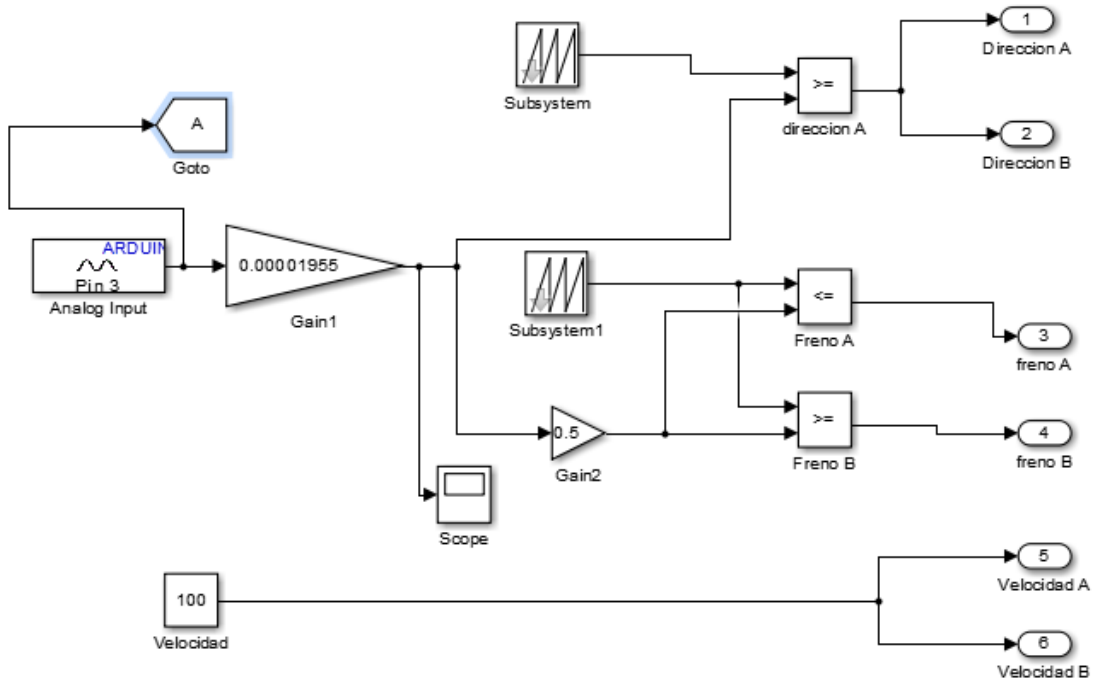


Figura 451: Modelo del subsistema con repetidor de secuencias modificado y potenciómetro.

Si lo realizamos de esta manera, podemos asegurarnos que podemos regular la velocidad de nuestro motor paso a paso con el potenciómetro y finalmente la señal del osciloscopio es exacta a la de la figura 448.

4.3.3 Servomotor

- *Arduino 1.8.1*

Realmente el servomotor no necesita el Motor Shield, pero ya que tiene los molex específicos para este tipo de conexión lo utilizaremos. Podremos utilizar en este caso el PWM del pin 5 y 6 establecidos para los molex como se muestra en las siguientes imágenes.

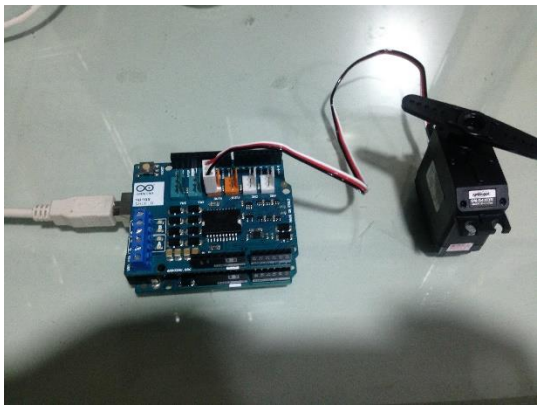


Figura 452: Servomotor controlado por arduino Uno

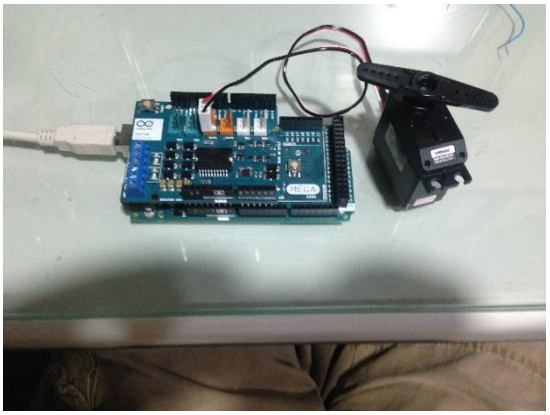


Figura 453: Servomotor controlado por arduino Mega.

También podremos separar los cables del servo, y conectarlos individualmente directamente en el arduino sin la placa Motor Shield, la programación será exactamente la misma. Para este tipo de motor que no se necesita una tensión ni corriente muy elevada no necesita tampoco la alimentación externa que hemos estado usando antes para otros motores.

Debemos añadir de primeras una librería necesaria al arduino, tecleando esta línea.

#include <Servo.h>

Gracias a esta librería conseguiremos establecer en nuestro arduino un rango de valores entre 0 a 180. Siendo 90 para que el servo este detenido, 0 una velocidad máxima hacia un sentido y 180 la velocidad máxima en el otro sentido.

A continuación determinaremos un nombre cualquiera a nuestro servomotor de la siguiente manera.

Servo 1;

Colocaremos en 1 el nombre especifico aleatoriamente, nosotros lo llamaremos *ser*, por lo tanto ponemos *Servo ser;*

Seguiremos añadiendo en void loop() la siguiente línea para vincular el servomotor a un pin del arduino en este caso en el pin 6.

ser.attach(6);

Y ya por último añadimos todos los movimientos que queremos que realice nuestro servomotor escribiendo:

ser.write(x);

Poniendo en "x" el valor ya nombrado entre 0 y 180. Para una mayor comodidad a la hora de establecer el movimiento de nuestro servo, podemos hacer como en el caso anterior y leer este valor del monitor serial, por lo tanto el código quedara tal que así.

```
#include <Servo.h>

Servo ser; //Llamamos al servomotor "ser"

String val=""; //val será nuestra variable obtenida del monitor serial
int vel=0; //nombre de la velocidad del servo
        // para 90, el servo estará parado, para 180 será velocidad máxima a
derechas
        //para 0 será velocidad máxima a izquierdas


void setup() {
  Serial.begin(9600); // Velocidad del puerto serial
ser.attach(6); //Vinculamos el servo al pin 6
}

void loop() {
if(Serial.available()>0){

   val=Serial.readString();
   vel=val.toInt();
}//Nos permite el cambio de valor de la variable de velocidad en el monitor
serial


  ser.write(vel); //mandamos el valor de la velocidad al servomotor
  delay(150); //esperamos 0,15 segundos para enviar otro valor
}
```

Aunque también podremos hacerlo como lo hicimos con el paso a paso, y establecer el valor del movimiento con un potenciómetro, por lo tanto, deberemos de

añadir lo ya explicado en ese apartado a la programación y realizar las conexiones como se observan.

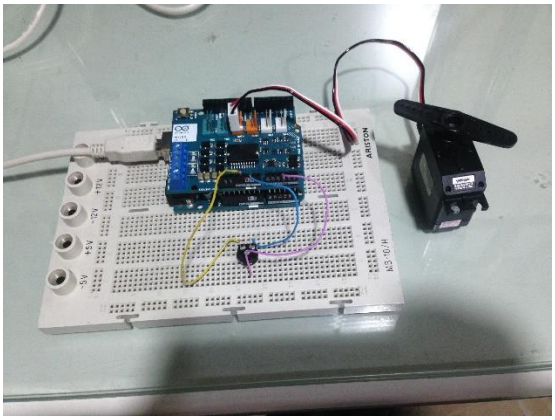


Figura 454: Servomotor controlado por arduino Uno y un potenciómetro.

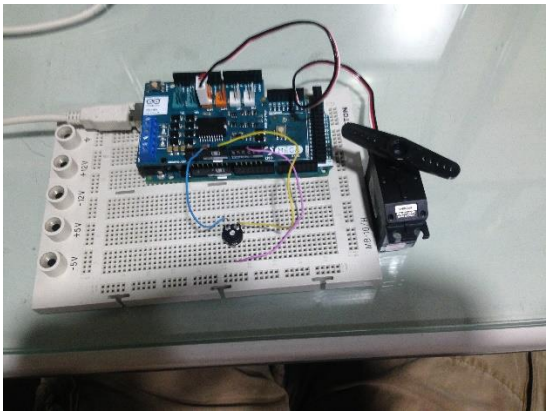


Figura 455: Servomotor controlado por arduino Mega *y un potenciómetro.*

```
#include <Servo.h>

Servo ser; //Llamamos al servomotor "ser"

int vel=0; //nombre de la velocidad del servo
        // para 90, el servo estará parado, para 180 será velocidad máxima a
        //derechas para 0 será velocidad máxima a izquierdas

void setup() {
ser.attach(6); //Vinculamos el servo al pin 6
}

void loop() {
  vel=analogRead(3); //Recibimos el valor analógico del pin A3
  vel=map(vel, 0, 1023, 0, 180); //cambiamos la escala de 0-1023 a 0-180

  ser.write(vel); //mandamos el valor de la velocidad al servo
  delay(10); //esperamos 10 segundos para mandar otro valor
}
```

Una vez realizado el ensayo, y conectarlo al osciloscopio, podremos observar que la señal es casi perfecta, y se observa el cambio del acho de pulso si modificamos el valor con el potenciómetro o con el monitor serial.

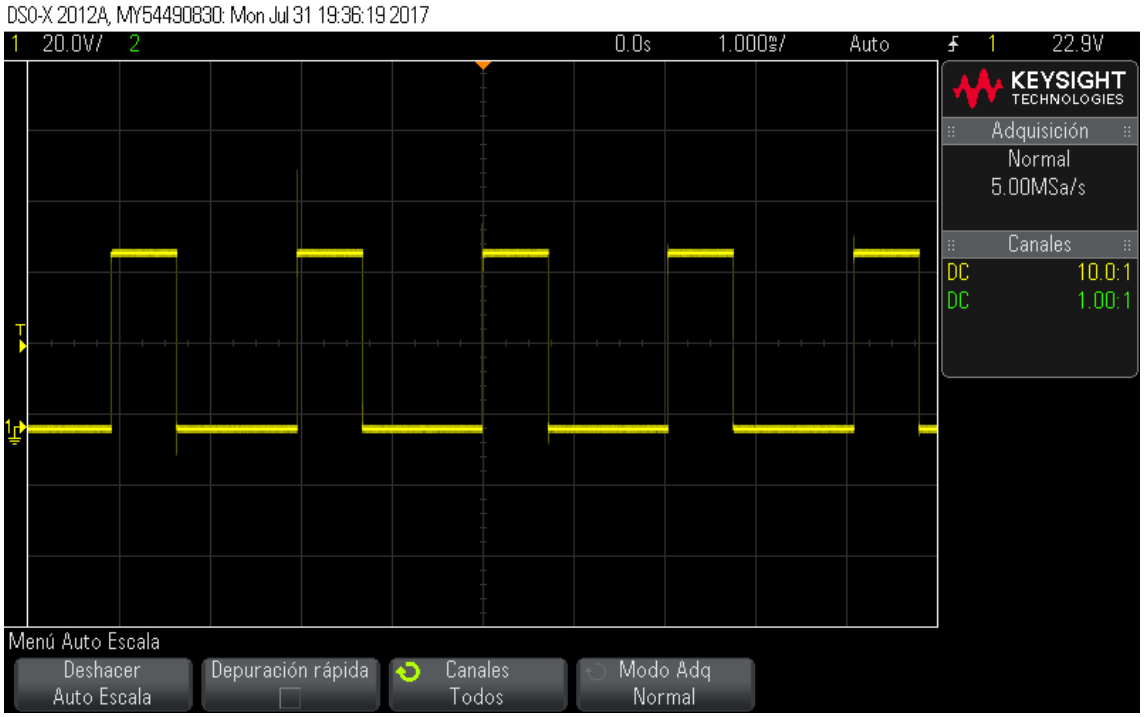


Figura 456: Imagen del osciloscopio para un servomotor conectado al arduino.

- *Matlab*

Para crear un modelo de simulink para un servomotor, simplemente habría que añadir un PWM con el pin correspondiente a donde se conectó el servo y unirle una constante a la cual le añadiremos valores entre 0 y 180 de la siguiente manera.

Figura 457: Modelo en Simulink para el control de un servomotor con arduino.

Existe un inconveniente, y es que no encuentra diferencia con los distintos valores desde el 1 al 180, por lo tanto, con este programa solo lograremos o bien arrancarlo o pararlo con el valor 0, simplemente a la hora de la simulación externa, habría que modificar el valor de la constante.

Pero también podríamos añadirle el potenciómetro para lograr controlarlo de forma externa.

Para ello solo se añadiría el "Analog input" con su pin correspondiente, además de una constante con el valor de 500 que es la mitad de la máxima señal de entrada del

arduino, y un comparador, que da valor de 1 o 0, si la señal recibida es mayor o menos de los 500.

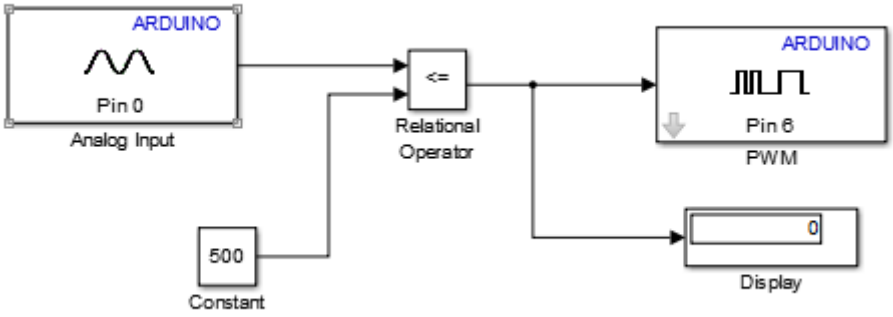


Figura 458: Modelo en Simulink para el control de un servomotor con arduino con potenciometro.

Finalmente obtendremos una imagen del osciloscopio bastante parecida a la conseguida por el programa arduino.

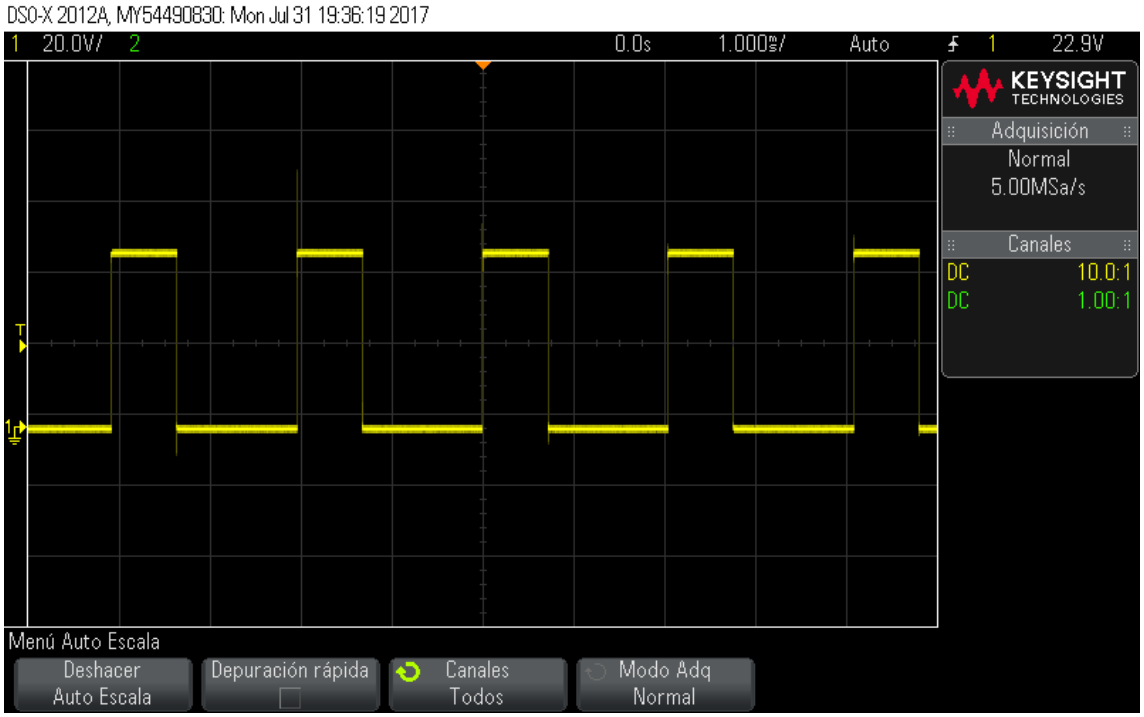


Figura 459: Señal del osciloscopio del arduino controlando un servomotor a través de Matlab

5. Raspberry Pi

5.1 Hardware

La Raspberry Pi se puede decir que es una computadora de tamaño muy reducido, en el cual se puede conectar a su televisión, un teclado y un ratón para su funcionamiento. Es capaz de realizar cualquier proyecto de electrónica y cualquier cosa que haría un ordenador de escritorio.

Podemos encontrarnos diferentes modelos.

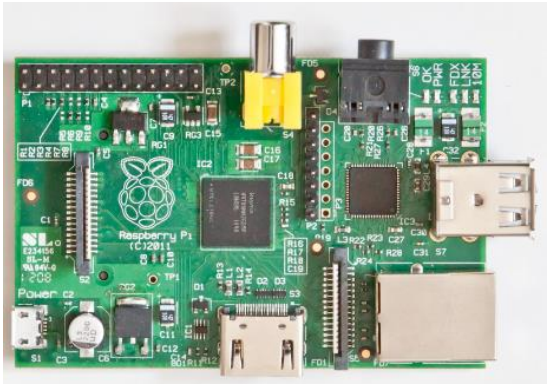


Figura 501: Raspberry Pi Modelo A

Figura 502: Raspberry Pi Modelo B+

Figura 503: Raspberry Pi 2 Modelo B

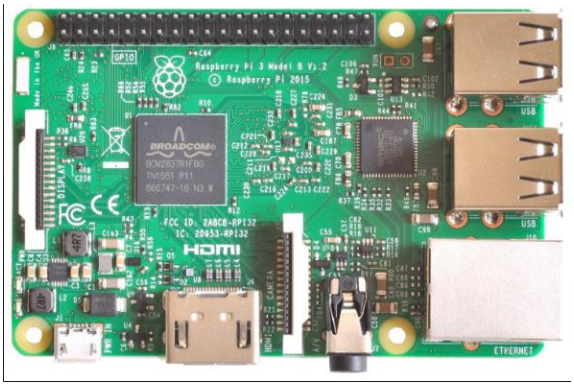


Figura 504: Raspberry Pi 3 Modelo B

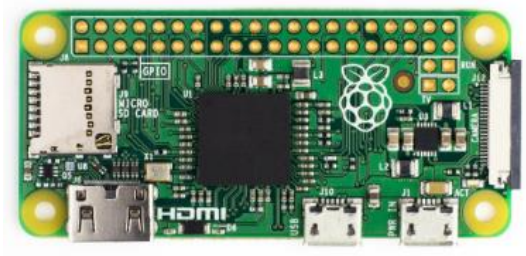


Figura 505: Raspberry Pi Zero

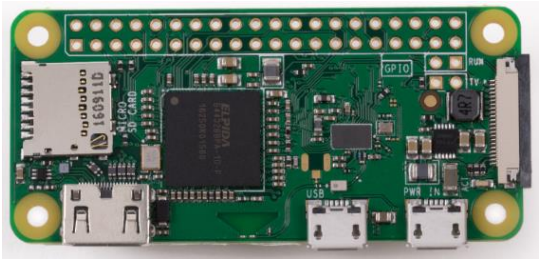


Figura 506: Raspberry Pi Zero W

Las principales diferencias que podemos encontrar entre las distintas placas son las expresadas en la siguiente tabla.

Tarjeta	SoC	Velocidad	Memoria Ram	Nº USB	Entrada internet	Tarjeta de almacenamiento
A	BCM2835	700Mhz	256 MB	1	No	SD
B+	BCM2835	700Mhz	512 MB	4	Si	Micro SD
2 B	BCM2836	900Mhz	1 GB	4	Si	Micro SD
3 B	BCM2837	1200Mhz	1 GB	4	Si	Micro SD
Zero	BCM2835	1000Mhz	512 MB	1	No	Micro SD
Zero W	BCM2835	1000Mhz	512 MB	1	No	Micro SD

Tabla 10: Modelo y caracteristicas de los distintas tarjetas de Rapsberry.

Además también encontramos una gran diferencia de tener wifi incorporado en el interior como en la Raspberry pi 3 y el Pi Zero W, los demás no los incorporan pero si puedes introducir un dispositivo wifi de USB.

5.1.1 Raspberry Pi 2 Modelo B

En nuestro caso utilizaremos este modelo de Raspberry con una tensión de alimentación de entre 8 y 18 voltios y 1,8 amperios. Con una memoria RAM de 1 GB una velocidad de 900 MHz.

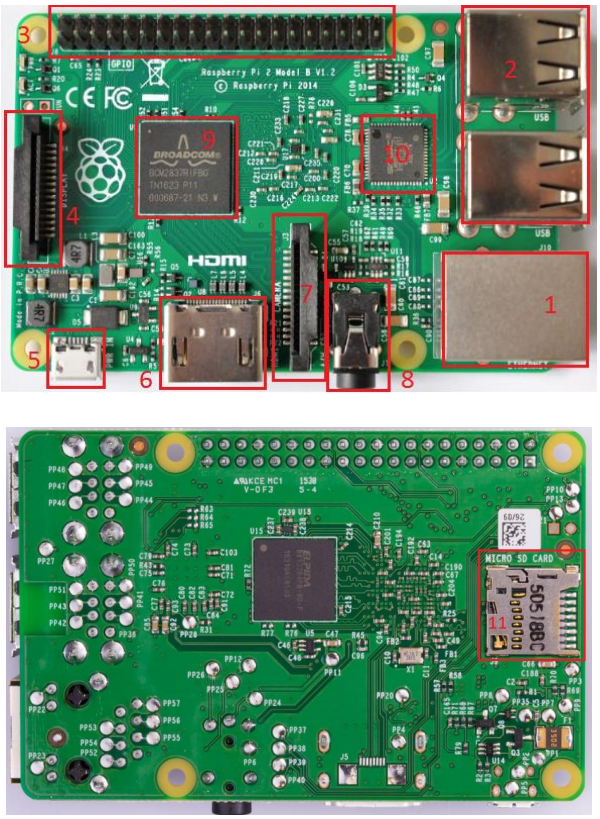


Figura 507:Tar*jeta Raspberry Pi 2 modelo B con sus partes principales.*

Según las partes que aparecen en la figura anterior se pueden diferenciar las diferentes partes.

1 -> Entrada de Ethernet RJ-45

2 -> 4 puertos USB (USB tipo A)

3 -> 40 pin (Entrada / salida de uso general)

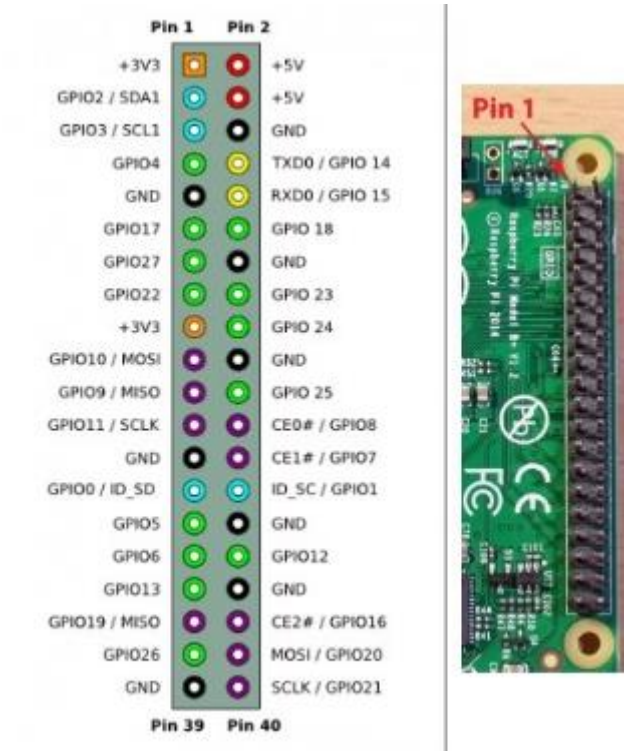


Figura 508: Funcionalidad de cada pin GPIO.

4 -> Interfaz de pantalla (DSI)

5 -> Entrada de tensión de alimentación (micro USB)

6 -> Puerto HDMI

7 -> Interfaz de cámara (CSI)

8 -> Audio Jack de 3,5 mm compuesto por video.

9 -> Tarjeta del sistema operativo BCM2836RIFBG

10 -> HUB USB (concentrador de dispositivos USB)

11 -> Puerto para tarjeta de memoria Micro SD

5.1.2 Gertbot

También utilizaremos una placa acoplable a la Raspberry llamada Gertbot, que nos ayudará al realizar el control de los motores.

Por lo tanto es una placa capaz de conducir cargas inductivas y capacitivas. Puede controlar motores paso a paso, motores con cepillado.

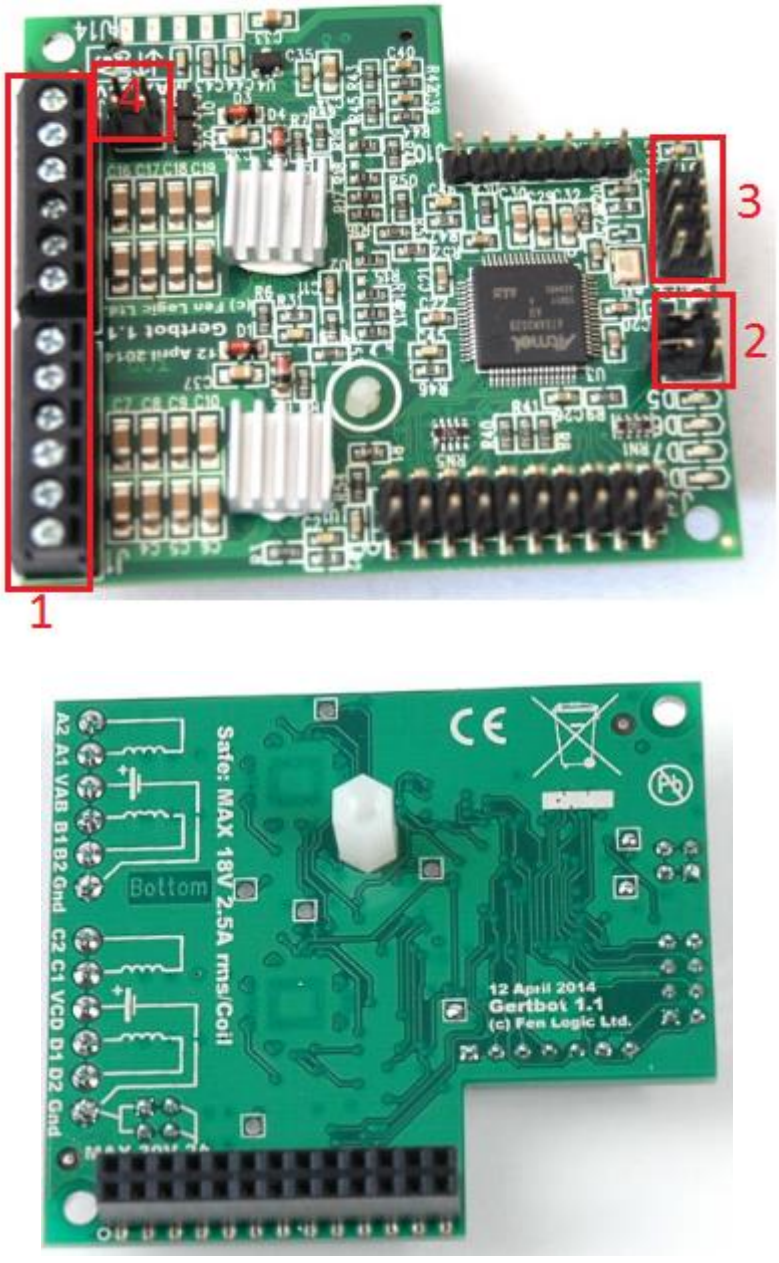


Figura 509:Tar*jeta Gertbot.*

Formadas por estos tres componentes fundamentales.

1 -> 12 borneras. 2 para cada bobina de los motores, con un total de cuatro bobinas, y dos borneras para la alimentación positiva y negativa para cada dos bobinas.

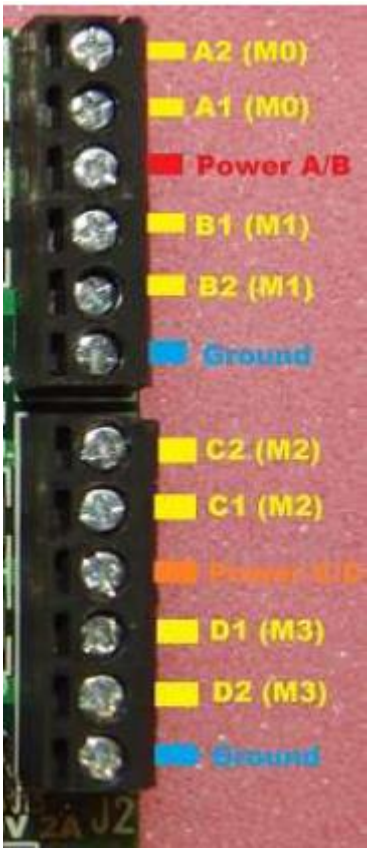


Figura 510: *Utilidad para cada bornera de nuestro Gertbot*

2-> Puerto para la identificación de varias placas Gertbot. Según la unión de pares de estos jumpers, la tarjeta se denominara como la 0, la 1, 2 o 3.

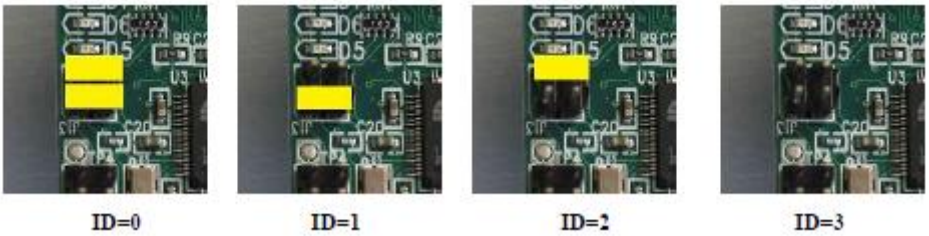


Figura 511: *variedad de union de los jumpers para la identificación de la placa*

3-> Puerto en cascada para controlar hasta cuatro tarjetas a la vez

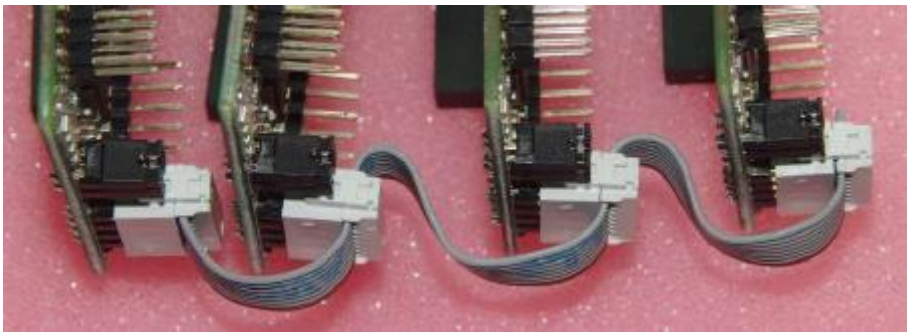


Figura 512: *Unión en cascada de 4 placas Gertbot (Esta imagen tiene las Jumpers de indentificacion mal colocadas.)*

4 -> Dos drenajes abiertos para realizar una alimentación externa para toda la placa de 30 voltios y 3 amperios.

5.2 Software

- *Instalación*

En primer lugar vamos a necesitar una tarjeta micro SD para poder guardar en ella la programación que va a necesitar nuestra Raspberry. Esta tarjeta debe tener un tamaño mínimo de 4GB aunque se recomienda utilizar una de 8 GB.

Debemos de descargar de la página https://www.raspberrypi.org/downloads/ el archivo denominado Raspbian, el cual es el sistema operativo que utilizaremos para Raspberry, que está basado en Linux. También existen otros sistemas que se pueden descargar y se pueden utilizar para la Raspberry. Una vez descargado este archivo deberemos descomprimirlo y nos dará un archivo imagen del tipo .img.

Incorporaremos la tarjeta micro SD a nuestro PC, deberemos de formatear esta tarjeta previamente, y utilizar un programa gratuito llamado win32diskimager para poder grabar la imagen del Raspbian en la tarjeta. Una vez el programa abierto, nos pedirá el archivo que queremos introducir, y el dispositivo donde lo guardamos, y ya simplemente darle a “write”. Este proceso tardará unos minutos.

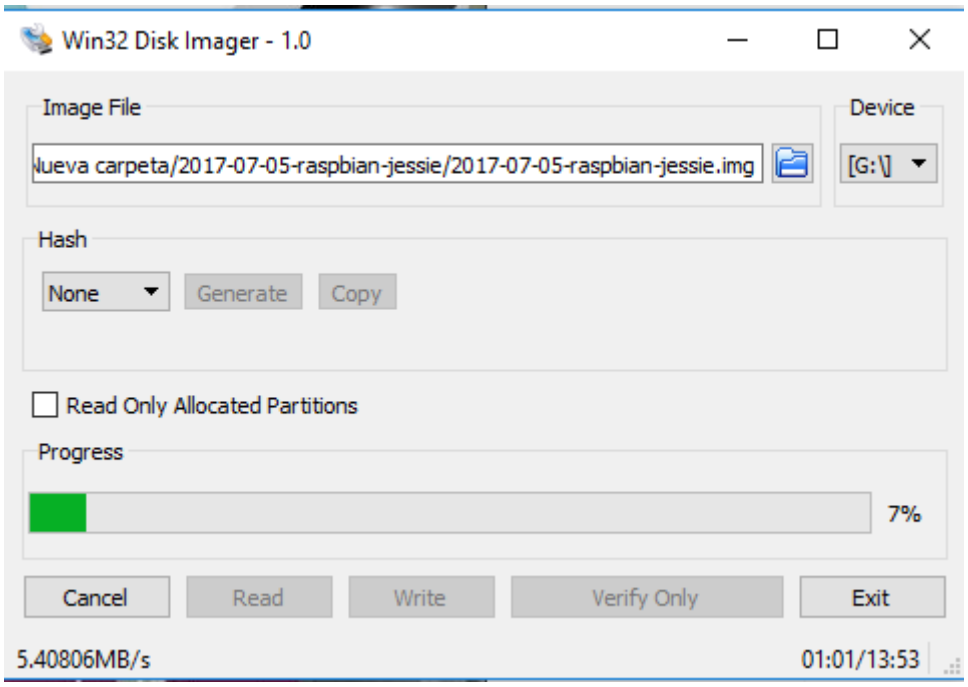


Figura 513: Programa win32diskimager grabando la imagen de Raspbian en una tarjeta micro SD.

Una vez estos pasos solo debemos introducir la tarjeta a la Raspberry e incorporarle a una pantalla a través de la entrada HDMI, además de un teclado y un ratón a través de USB. También necesitamos una entrada de Ethernet con una señal adecuada y una entrada de tensión a través del micro USB con la ayuda de un trasformador que tenga una salida de 5 voltios.

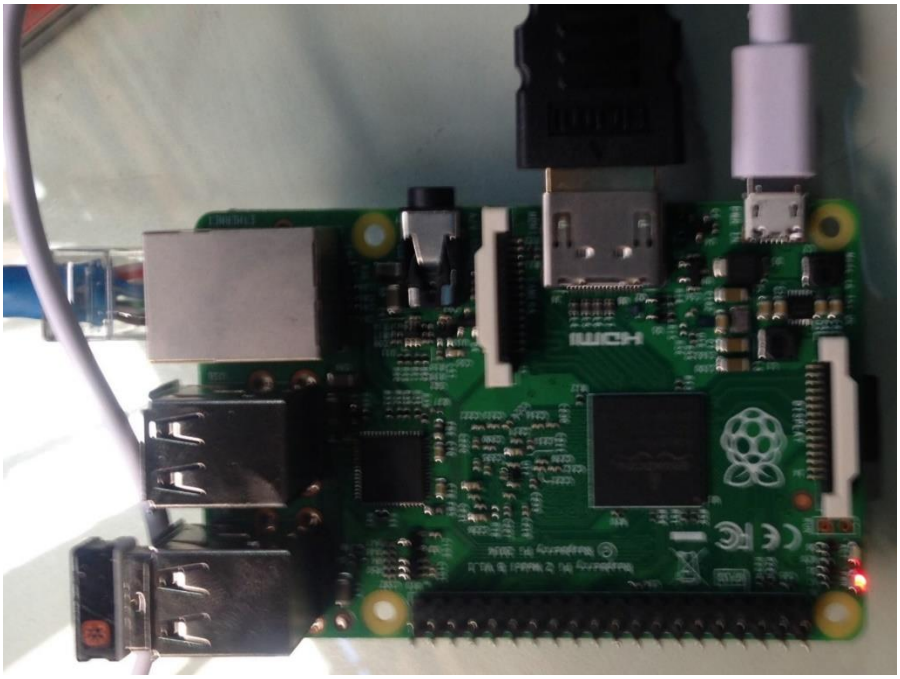


Figura 514: Raspberry con todos sus componentes conectado (Pantalla, teclado, raton, internet y corriente)

Simplemente si lo conectamos a la corriente se encenderá solo. Y aparecerá una pantalla de configuración inicial.

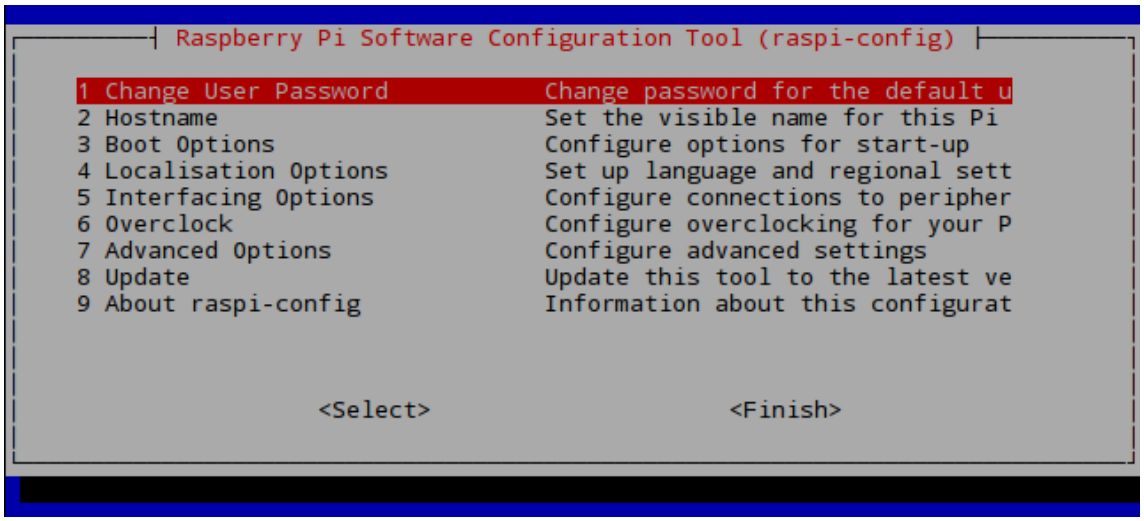


Figura 515: Pantalla de configuración pra la Raspberry.

En esta pantalla tenemos:

1 Una opción muy recomendable para poder utilizar la totalidad de la tarjeta de memoria del micro SD. En Raspbian es necesario para que el sistema no ocupe toda la memoria de la tarjeta.

2 Esta segunda opción nos servirá para poder cambiar la contraseña de nuestra Raspberry Pi. De forma predeterminada todas las Raspberry tendrán el nombre del usuario “pi” y su contraseña es “Raspberry”. Por lo tanto es necesario cambiarla por motivos de seguridad.

3 Esta nos da la opción de predeterminar la manera de inicio de nuestra Raspberry. Hay tres opciones, la primera que se abra con la línea de comando directamente, que es la forma que iniciará nuestra Raspberry sino lo modificamos, la segunda se abrirá con el escritorio gráfico, y la tercera opción con un programa determinado. Como nosotros vamos a utilizar el escritorio gráfico, marcaremos la segunda opción “Desktop log in as user ´pi´ at the graphical desktop”. Si no realizamos este cambio, cuando nos aparezca la línea de comandos la próxima vez que arranquemos nuestra Raspberry, deberemos escribir “startx” para iniciar el escritorio gráfico.

4 Es necesario para poder cambiar la zona horaria, el idioma, y el tipo de teclado que utilizaremos

5 Para habilitar o deshabilitar partes del Software.

5.1 “Camera” Para la configuración de la conexión de la cámara.
5.2 “SSH” Para habilitar o deshabilitar la línea de comandos SSH
5.3 “VNC”
5.4 “SPI”
5.5 “I2C”

5.6 “Serial” Para habilitar o deshabilitar las conexiones predeterminadas de algunos pin de nuestra Raspberry, esta opción si es necesario que realicemos un cambio, debido a que si no se realiza no aceptará nuestra Gertbot. Tendremos que entrar y darle en la primera pregunta “No” y a la segunda “Yes”.
5.7 “1-Wire”
5.8 “Remote GPIO” Para habilitar o deshabilitar el control de los Gpio.

6 Aquí tenemos una opción para poder aumentar el rendimiento de la Raspberry Pi

7 Para opciones avanzadas.

Aquí descubrimos diferentes apartados.
7.2 “Expand Filesystem” Para poder expandir los archivos en la micro SD.
7.2 “Overscan” Para poder modificar el tamaño de la pantalla que utilizaremos.
7.3 “Memory Split” Cambiar la memoria GPU.
7.4 “Audio” Para poder forzar la salida del audio a través del HDMI.
7.5 “Resolution” Para modificar la resolución de la Raspberry.
7.6 “GL Driver”

8 Para actualizar a la última versión.

9 Para obtener información a acerca de esta lista de configuración.

Una vez terminada toda la configuración que necesitamos para nuestro Raspberry simplemente tendremos que darle a finalizar, y nos pedirá reiniciar.

Después del reinicio, aparecerá directamente el escritorio digital como hemos dicho.

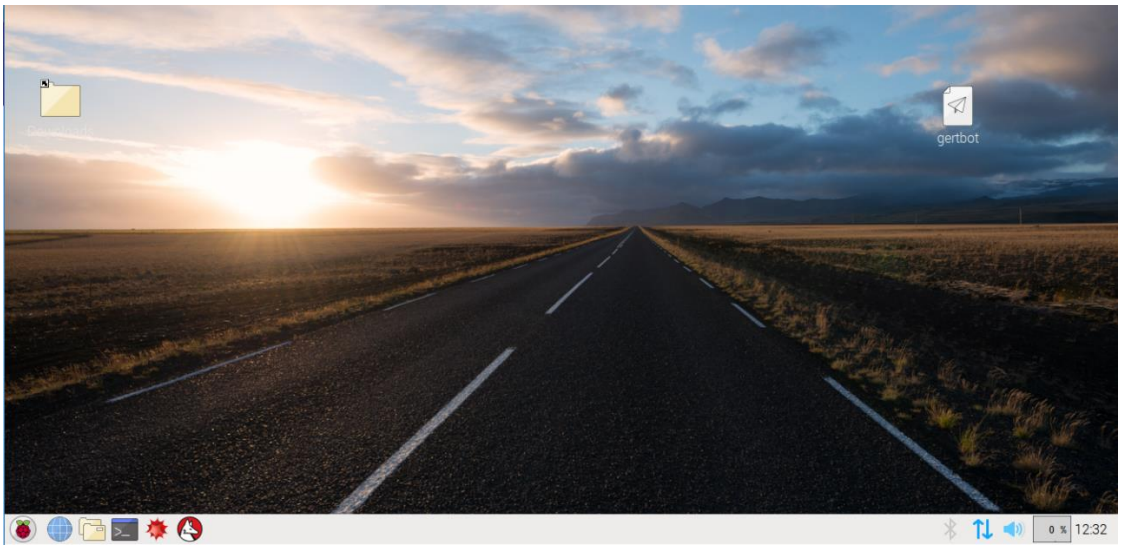


Figura 516: Pantalla principal de Raspbian.

Si después del reinicio, queremos volver a modificar un parámetro de la configuración, tendremos que escribir en el terminal “Sudo raspi-config”, y nos volverá a aparecer la figura 515.

- *Conexión a escritorio remoto*

Sino queremos tener una pantalla, un teclado y un ratón independiente, podemos conectar la Raspberry a nuestro ordenador con Windows. Para ello, necesitamos escribir en nuestra línea de comandos de nuestra Raspberry esta línea

“sudo apt-get install xrdp”

A continuación nos va a pedir nuestra contraseña, que si no se ha modificado como se explicó antes será “Raspberry”. Después de la descarga y la instalación de este programa, deberemos de escribir:

“ifconfig –a”

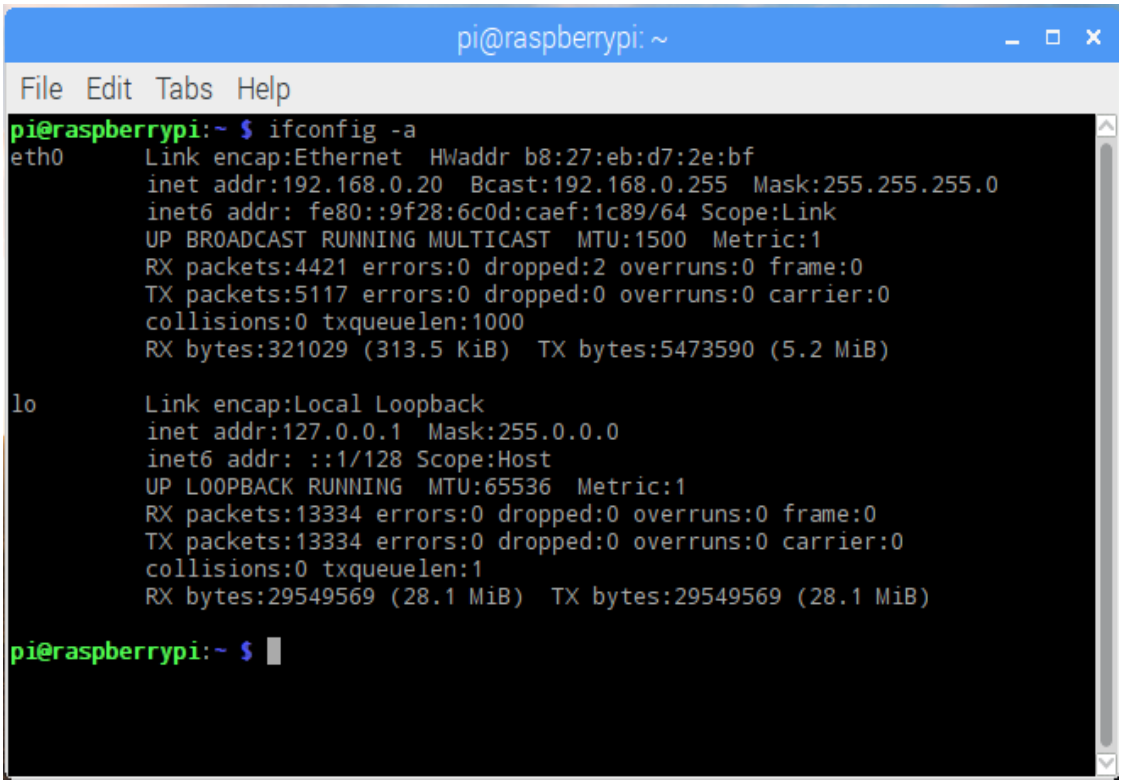


Figura 517: Comando “ifconfig –a”

En la segunda línea obtendremos la dirección ID que tiene nuestra Raspberry conectada a esa red.

A continuación en nuestro ordenador, deberemos buscar el programa que viene predeterminado en nuestro Windows llamado “Conexión a escritorio remoto” y deberemos de copiar en equipo la dirección IP obtenida antes, y ya solo darle a conectar.

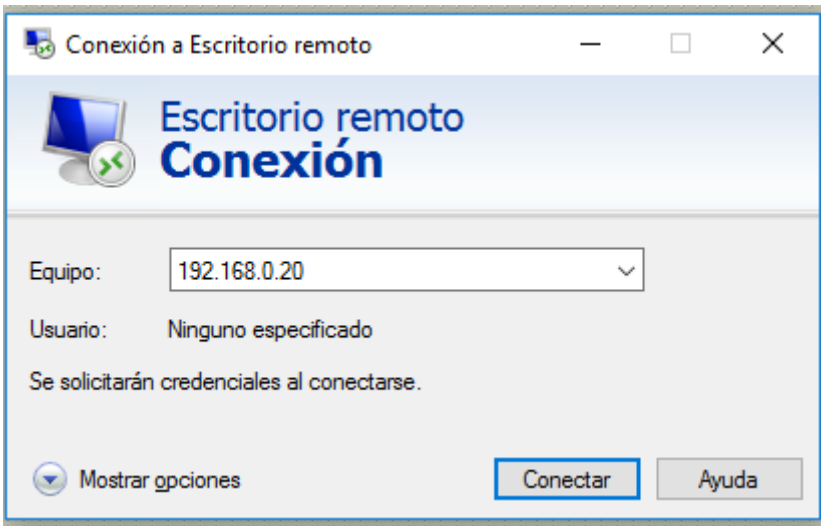


Figura 518: Programa conexión a escritorio remoto

Gracias a este programa podremos manejar nuestra Raspberry pi sin necesidad de conectarlo a una pantalla HDMI, a un teclado y ratón auxiliar.

Al darle a conectar, solo tendremos que introducir nuestro usuario y nuestra contraseña, y aparecerá de nuevo nuestro escritorio de Linux.

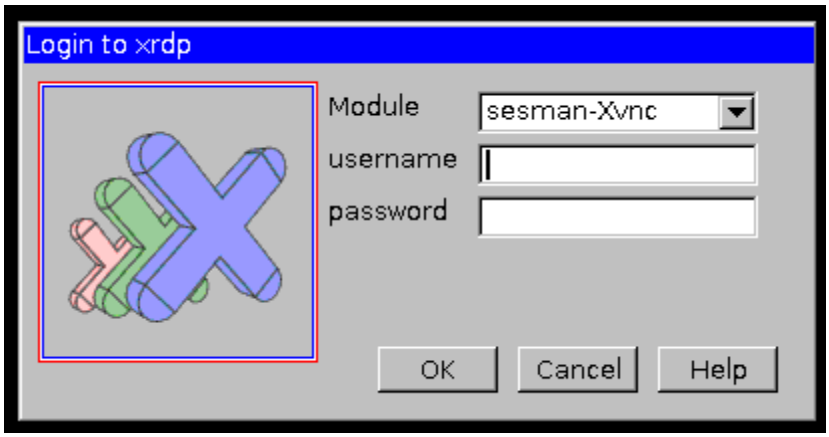


Figura 519: Ventana para entrar a la Raspberry con nuestro usuario y contraseña

Existe un problema que es habitual con este procedimiento. La IP de nuestra Raspberry puede variar cada vez que se conecte a la red, por eso mismo, explicaremos como podemos fijar esta IP.

Para ello solo debemos saber cuál es la IP oficial que tiene el rúter que conectaremos nuestra placa, para ello, con cualquier ordenador conectado a la red, tenemos que abrir el programa “símbolo del sistema” y escribir “ipconfig”, y aparecerá lo que hay escrito en la figura 520, y deberemos de copiar la IP escrita en la línea de Puesta de enlace predeterminada.

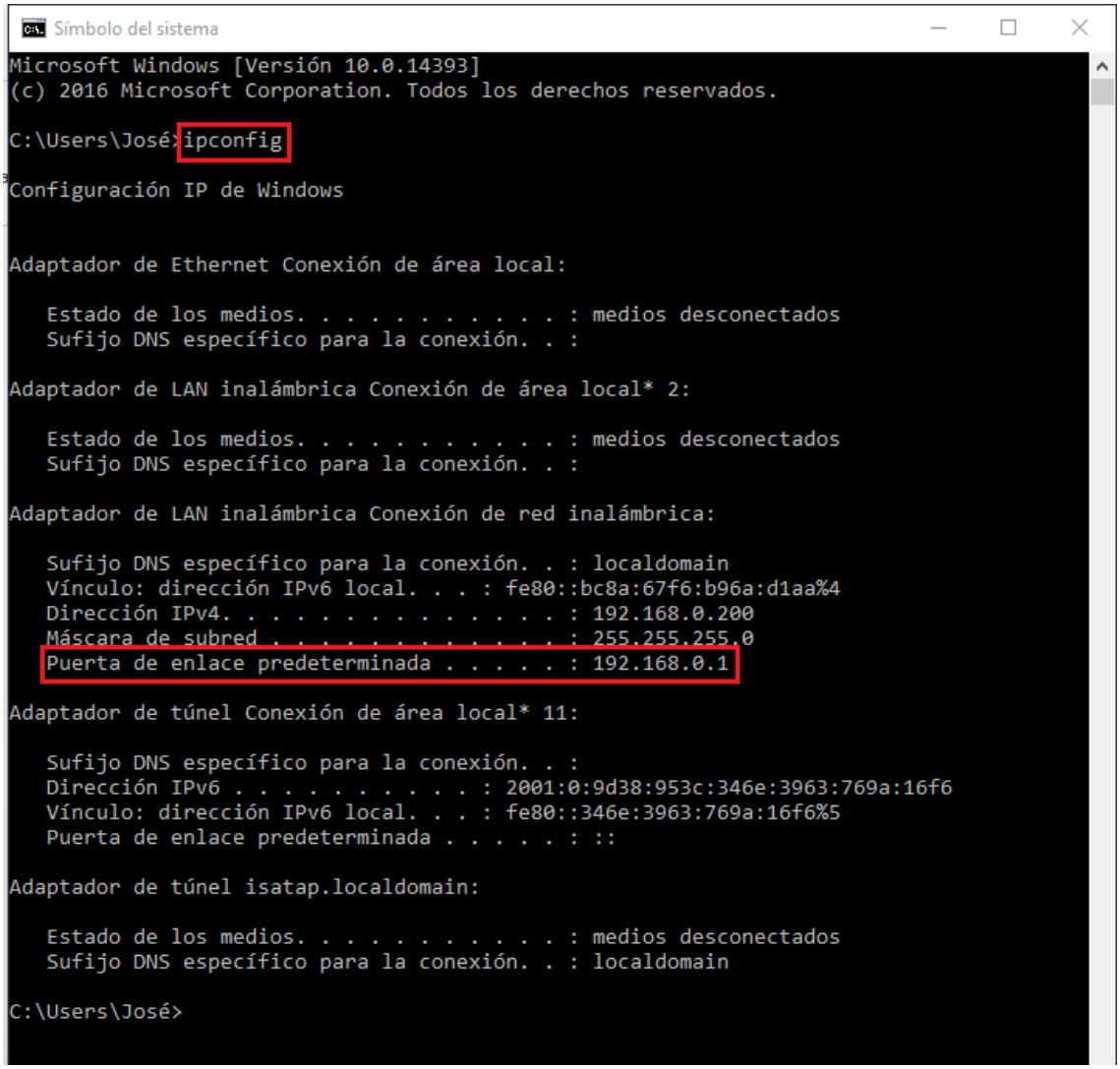


Figura 520: Averiguar la IP de nuestra red con Símbolo de sistema.

Seguiremos colocando el ratón encima del icono de la red en la barra de inicio de la pantalla gráfica de Linux y darle al botón derecho, y seleccionar la opción de "Wireless & wired Network Settings".

Se nos abrirá una pantalla emergente, donde colocaremos los datos anteriores como se observa en la figura 521.

Lo único que tendremos que hacer entonces es aplicar y reiniciar nuestra tarjeta, abrir esta vez el control del escritorio remoto con la nueva y fija IP que hemos establecido.

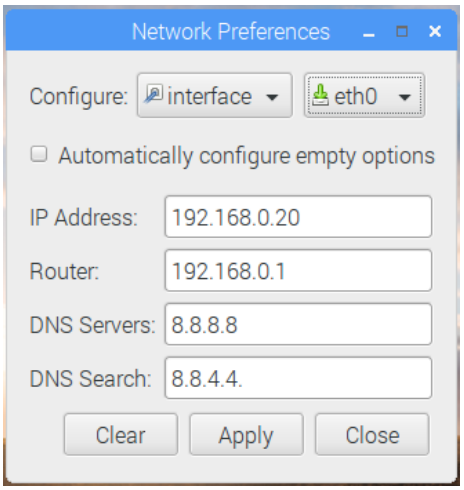


Figura 521: Ventana emergente para fija la IP.

Una forma de comprobar que este procedimiento lo hemos hecho de forma correcta, seria escribir en la línea de comandos "ping www.google.es" y acto seguido pulsaremos ctrl+C. Si se recibe la misma cantidad de señales de las que se envía, tendremos buena conexión. El comando anterior se podrá realizar con cualquier página web.

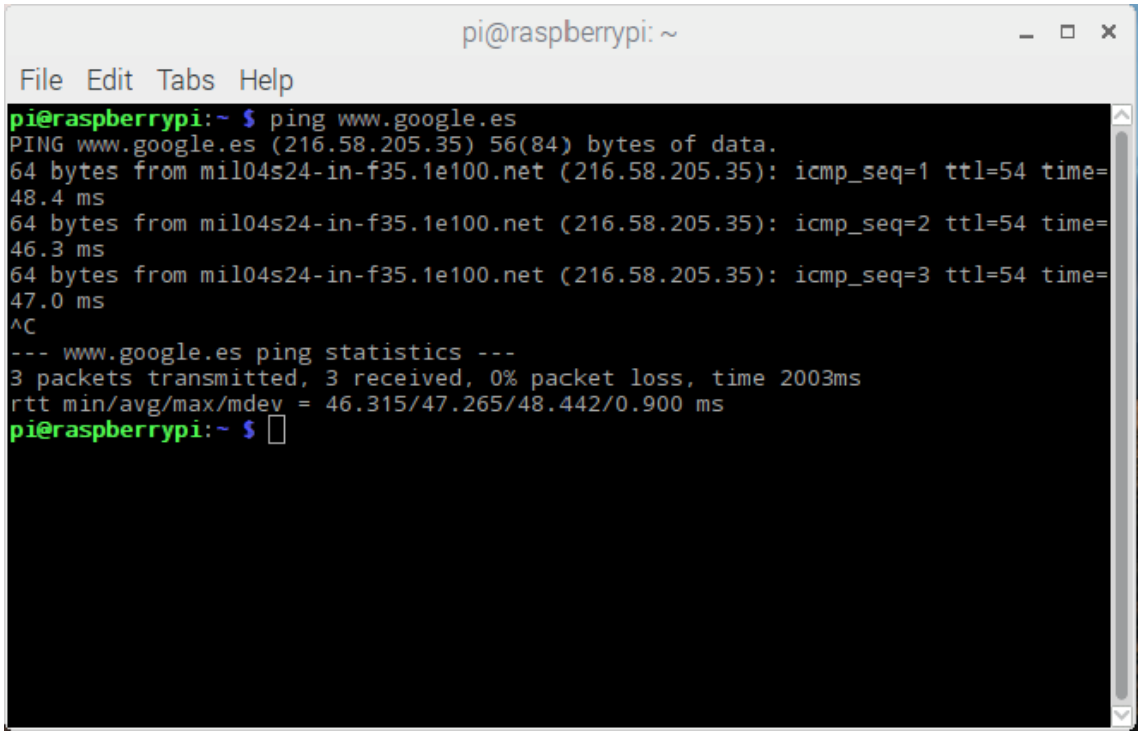


Figura 522: Comprobación de la conexión a la red.

Hay que explicar que si utilizamos el escritorio remoto y no tenemos buena conexión a la red, la información que enviamos a nuestra Raspberry para manejar el motor, no llegará.

- *Programación Gertbot*

Cuando tengamos todo esto controlado, podremos empezar por descargarnos el programa de nuestro Gertbot. Para ello, entramos en internet, y vamos a esta página: https://www.gertbot.com/download.html aquí podremos encontrar diferentes tipos de manuales, drivers, ejemplos, y lo que estamos interesados nosotros, que es la GUI del programa. Se puede hacer de dos formas, uno con un pc independiente mandando información a nuestra Raspberry a través de una entrada USB que se conecta a los pines de la tarjeta, o con un ejecutable para Raspbian. La primera opción, tiene muchos inconvenientes, y se necesita muchos pasos para poder lograrlo, debido a que el programa necesitara unos drivers instalados independientemente y unos Uart descargables de esta misma página, pero que están incompletos. Además de la extremada sencillez que tenemos en la segunda opción, que solamente tenemos que descargar el ejecutable llamado "Gertbot debug GUI (.tgz)".

Una vez descargado, solamente debemos abrir el programa, y ejecutarlo.

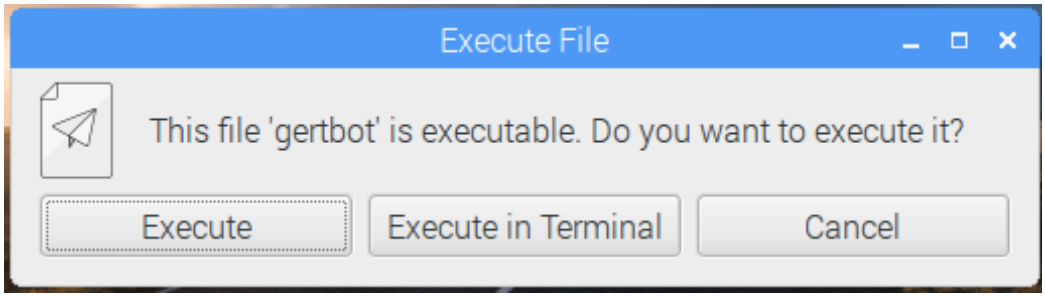


Figura 523: Ventana emergente para ejecutar nuestra GUI.

Y ya se abrirá automáticamente nuestro programa.

Figura 524: Ventana inicial de nuestra GUI del Gertbot

Si tenemos conectada ya el Gertbot, podemos darle al botón “connect”.

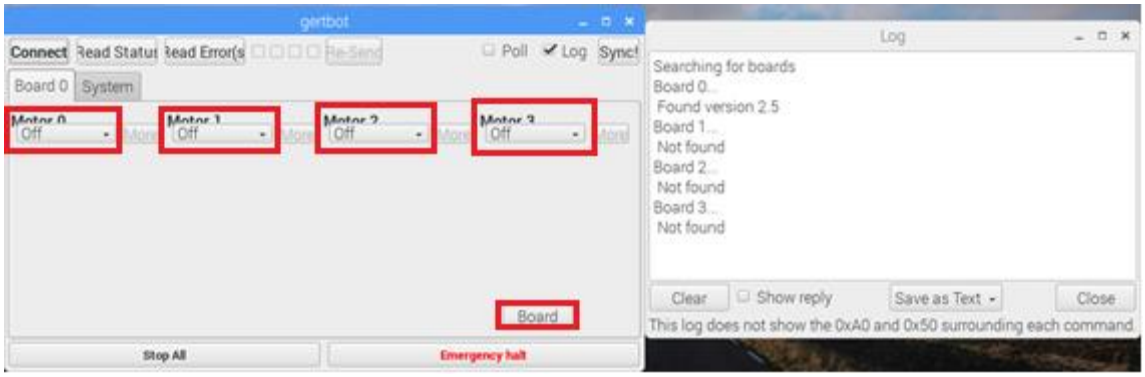


Figura 525: Ventana principal de nuestra GUI del Gertbot al comenzar.

Como se observa en la figura 525 nos aparecerá una segunda ventana donde establecerá la conexión, según como tengamos conectados los jumpers de identificación, estaremos en la placa 0, 1, 2 o 3, como ya hablamos anteriormente. Si tenemos más de un Gertbot conectado, nos aparecerán diferentes pestañas para el control de cada placa.

A continuación, deberemos de conectar de forma física el motor que deseemos controlar, como observábamos en la figura 510, además de la alimentación externa. Esta alimentación deberá de ser entre unos 8 y 18 voltios, y un máximo de 2,5 amperios. Hemos conectado una fuente de tensión externa, para poder regular estos dos valores.

Tendremos una opción en esta figura, que es la de “board” gracias a ella, nos saldrá una ventana con valores característicos de nuestra placa, con la tensión que tiene a tiempo real cada uno de los pines de los conectores J3. Esta ventana será muy útil si utilizamos para nuestro control de motores unos interruptores externos, llamado en este programa H, De esta manera podemos habilitarlos, y saber en qué pin deberemos de conectarlos.

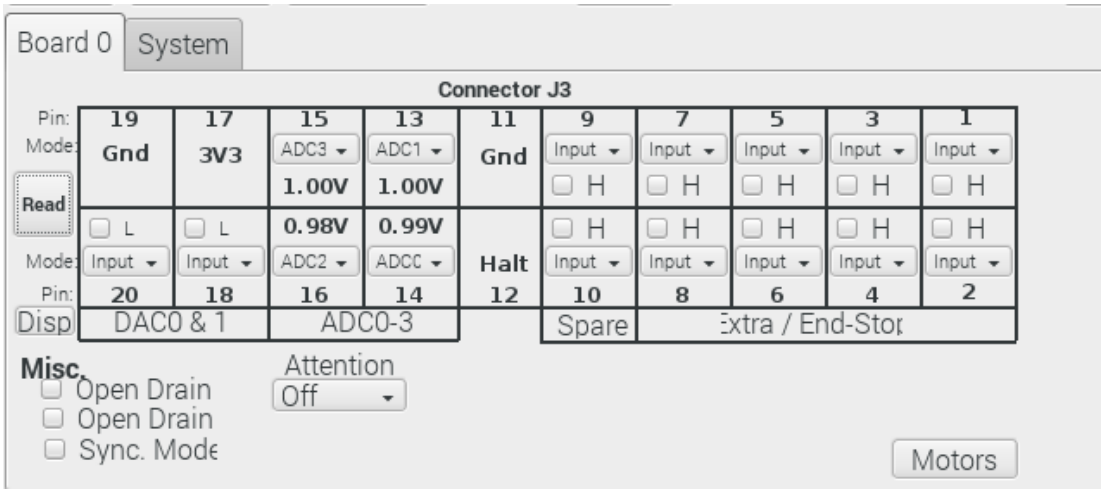


Figura 526: Ventana para la configuracion de los pin de nuestra gertbot.

La utilización del Gertbot no es compatible con el programa Matlab, debido a que el programa Matlab mandará las órdenes a los pines que utilizamos, pero al no haber relación destacable entre cada uno de las entradas de los motores, con los pines, como lo había en el arduino, no podremos controlarlo adecuadamente.

5.3 Ensayos

- *Motor DC brushed*

Después de realizar las conexiones adecuadas del motor en la placa Gertbot, simplemente tendremos que seleccionar el tipo de motor que hemos utilizado en los recuadros de la figura 525 según el puesto de la tarjeta hemos conectado físicamente el motor.

En este ensayo tendremos que utilizar la opción "DC/brushed". En ese instante, saldrá todos los valores que podremos modificar para el funcionamiento del motor.

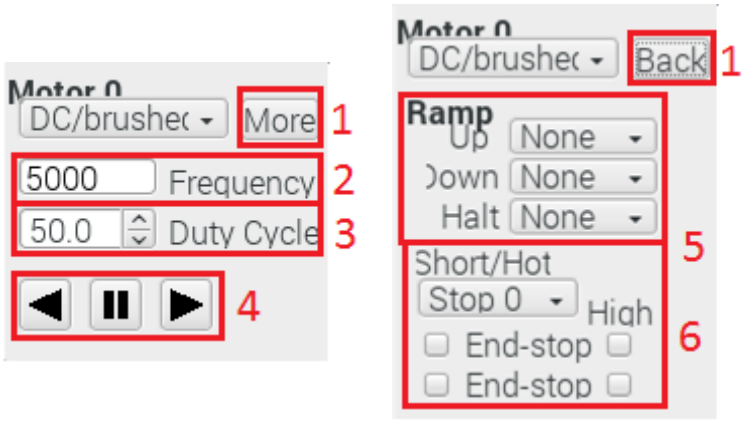


Figura 527: Parametros para el control del motor DC Brushed.

Según la figura anterior, tenemos dos imágenes independientes, que se puede acceder entre ellas dándole a "more" o "back" (1)

Encontramos en 2 la frecuencia enviada al motor, en 3 el duty que se explicó en el apartado del arduino, que básicamente lo que hará, será nivelar el valor de tensión que circulará al motor, y en este caso, variará la velocidad del mismo.

Simplemente dándole a los botones del 4, el motor arrancará, variando el sentido de giro según el lado que le demos.

También hay unos parámetros secundarios, en 5 podremos regular un tiempo de arranque o de frenado del motor, para que no se produzca estos procesos bruscamente, y evitar los picos de corriente. Y finalmente en 6 lograremos establecer interruptores externos que se pueden conectar a través de los pines de nuestro Gertbot además, de poder establecer donde se realizara una parada de emergencia por si se produce un cortocircuito en las conexiones.

En una sola placa Gertbot se pueden lograr conectar 4 motores DC Brushed a la vez, y se necesitará una alimentación externa con las características anteriores para cada 2 motores.

Si realizamos este ensayo con un motor conectado en A1 y A2, corresponderá al motor 0 del programa, y sal señal de osciloscopio al poner en marcha el motor es de esta manera.

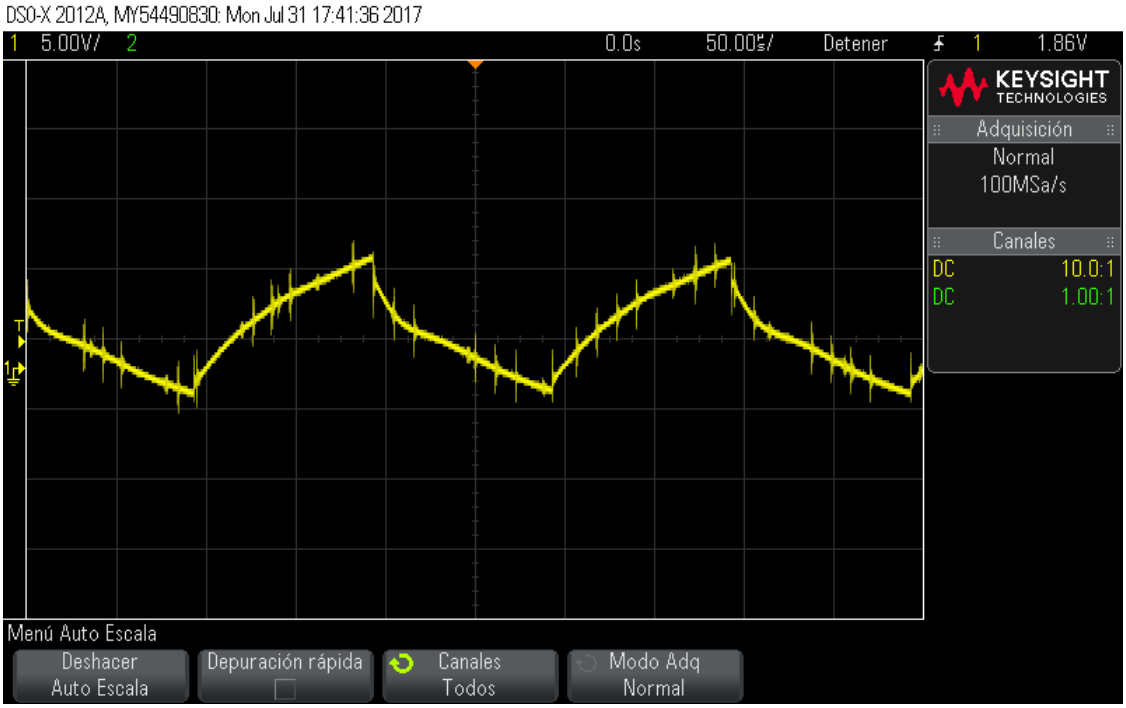


Figura 528: Señal del osciloscopio concetado a un motor DC empleando Gertbot

Con esta placa, podremos manejar un total de 4 motores de esta clase de forma independiente. Siendo los conectores A para el motor 0, el B para el motor 1, C para el motor 2, y el D para el motor 3 de nuestra GUI, teniendo en cuanta que necesitamos una fuente de alimentación de las características nombradas para el motor 0 y 1, y otra fuente igual para el 2 y 3.

También hemos utilizado dos motores DC brushed para demostrar que funcionan perfectamente aunque se ponga más de un motor, y con la misma tensión de alimentación externa y observamos la figura 529 que es la señal del osciloscopio.

Observamos que aunque tenga la misma tensión de alimentación la placa, el motor 0 no se ve afectado al conectarse un motor 1, y aunque en la figura de este ensayo se ha realizado con las mismos parámetros para ambos, se puede modificar, la tensión, como la frecuencia, y los tiempos de arranque y frenado.

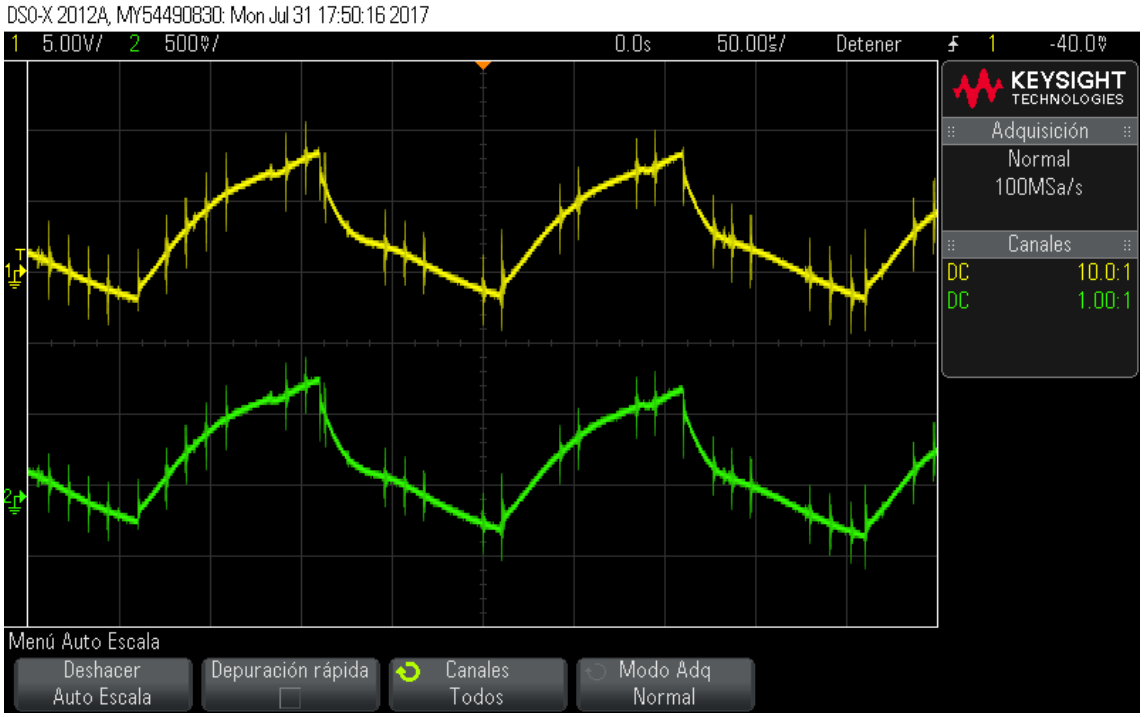


Figura 529: Señal del osciloscopio concetado a dos motor DC empleando Gertbot

- *Motor paso a paso de dos bobinas.*

Con el programa de nuestro Gertbot, simplemente debemos de seleccionar este motor, en el lugar donde lo vamos a conectar, que en este caso, podremos conectar dos motores en total de este modelo, e irán conectadas uno de las bobinas en A y la otra en B, o una bobina en C y la otra en D.

Por lo tanto, si seleccionamos en los tipos de motores, el "Step Gray" en el motor 0, se deshabilitará el complementario, es decir el motor 1, y lo mismo si lo seleccionamos en el 2, se deshabilita el motor 3. Esto es debido a que en estas dos bobinas, mandará las mismas ordenes de funcionamiento que pongamos en el motor 0 o en el 2.

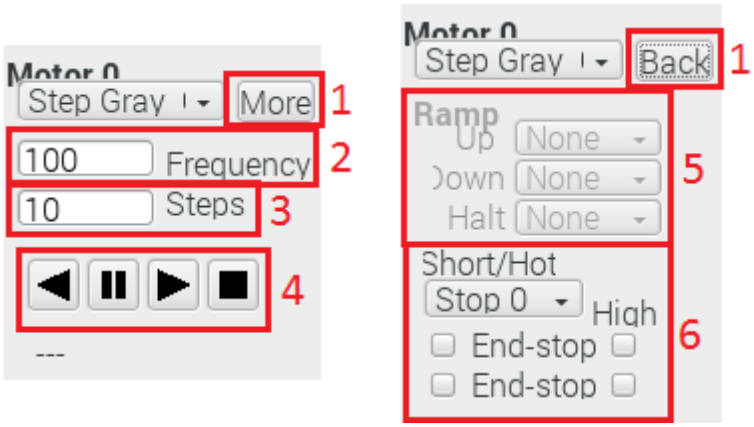


Figura 530: Parámetros para el control de un motor paso a paso de dos bobinas.

Como en el caso anterior, nos podremos encontrar con dos ventanas de parámetros que podremos cambiar seleccionando el botón de “more” y “back” (1).

Aquí también nos encontramos en 2, el valor de la frecuencia que se le dará al motor, y en tres, en este caso no se podrá modificar la tensión, sino los pasos que queremos que realice. En el momento que haga esa cantidad de pasos, el motor se parará, por eso mismo deberemos de poner un valor muy alto de pasos, pero que se mantenga activo durante un tiempo.

En 5 están los valores de tiempo de las rampas de arranque y frenado, pero que para este tipo de motor, esta opción está bloqueada y no se puede modificar. Y por último en 6 tendremos como antes, las opciones para lograr poner unos interruptores externos, y para determinar que motor debe de para si se produce algún cortocircuito.

Una vez realizado este ensayo, lograremos que el motor funcione perfectamente en ambas direcciones, y obtenemos la siguiente señal del osciloscopio.

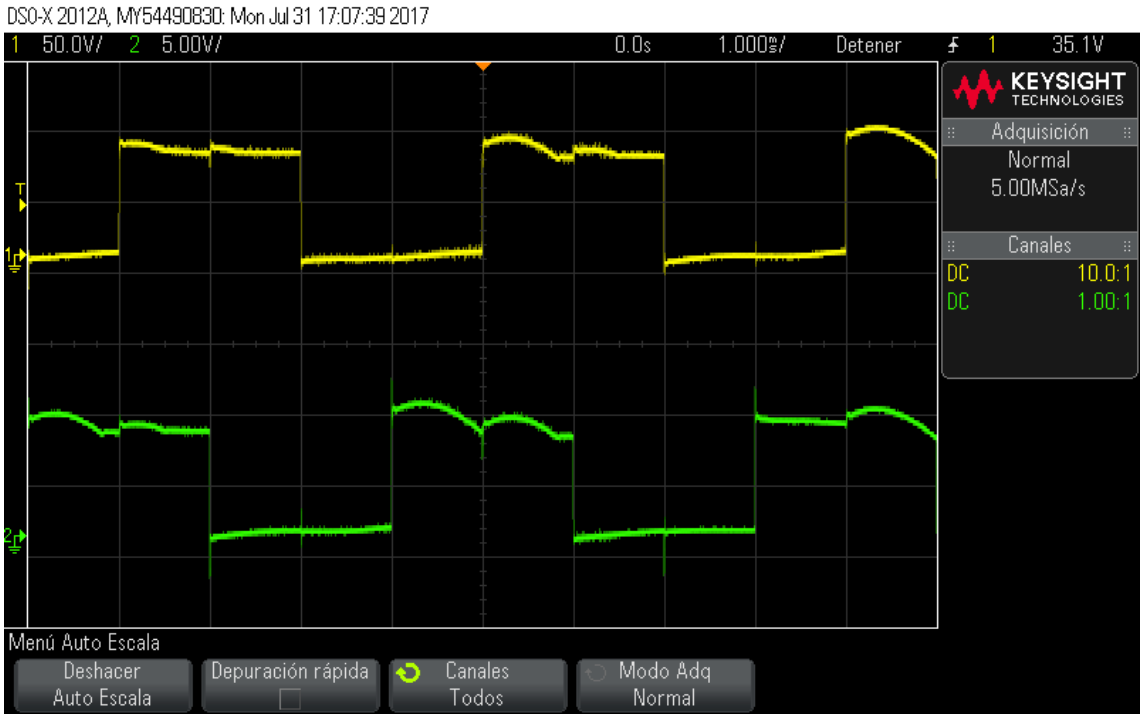


Figura 531: Señal del osciloscopio concetado a un motor paso a paso de dos bobinas empleando Gertbot

Si nos fijamos en el tipo de señal que conseguimos, nos damos cuenta que la placa Gertbot, dará al motor un paso doble el cual explicamos en el apartado 3 de materiales.

También hemos ensayado la placa con toda su funcionalidad, conectando este motor paso a paso en A y B, y dos motores DC brushed en los otros dos conectores. Por lo tanto hemos utilizado dos alimentaciones auxiliares y ha funcionado perfectamente. Al tener un osciloscopio con solo dos entradas, solo hemos podido medir una bobina de nuestro paso a paso, y un motor DC brushed, y este es nuestro resultado.

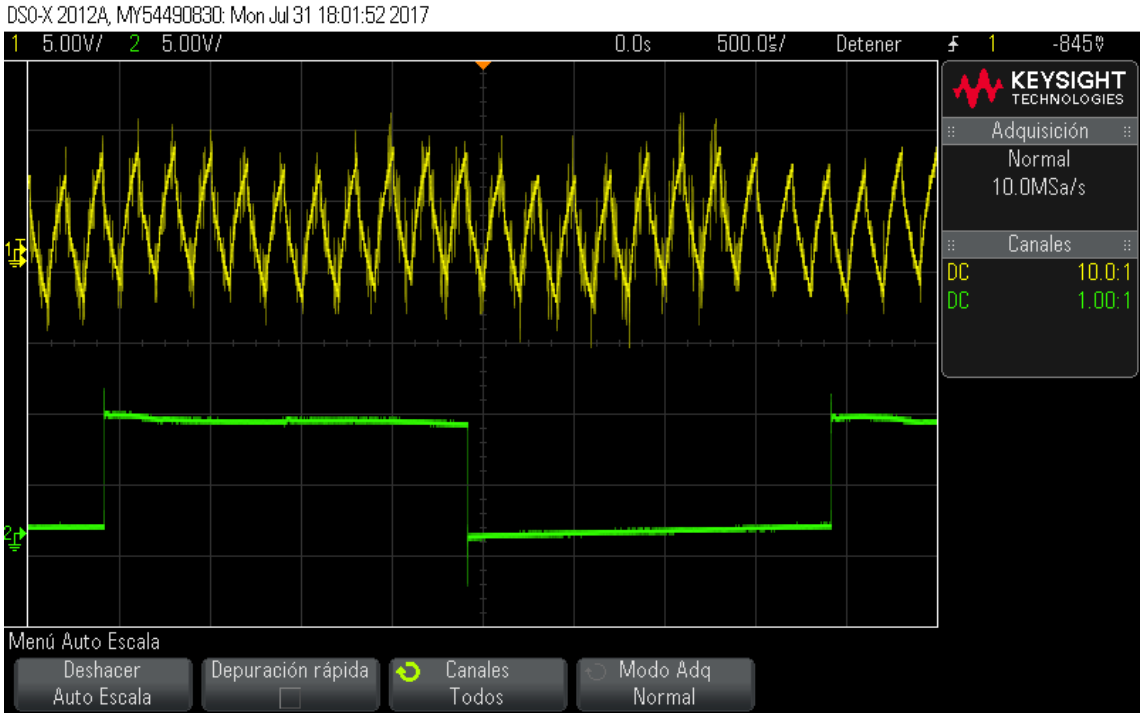


Figura 532: Señal del osciloscopio concetado a una bobina de un motor paso a paso de dos bobinas y un motor DC Brushed empleando Gertbot

La señal de nuestra bobina del motor paso a paso, es diferente a la mostrada anteriormente porque hemos aumentado la escala horizontal, y el motor DC Brushed se puso otro valor mayor de duty, y por lo tanto va a mayor velocidad nuestro motor.

- *Servomotor*

Para poder realizar el control de un servomotor, tendremos que realizarlo con la placa Raspberry solamente, debido a que la placa Gertbot no está diseñada para el control de los servos.

Tendremos que instalar el servo con los pines J3 de la figura 509. El cable de la tensión al pin 1 que da una tensión de 3,3 voltios. El cable del neutro al pin 6 y finalmente el cable de la señal, tendrá que conectarse a un “gpio”, en nuestro caso lo conectaremos al pin 12 que es el “gpio 18”.

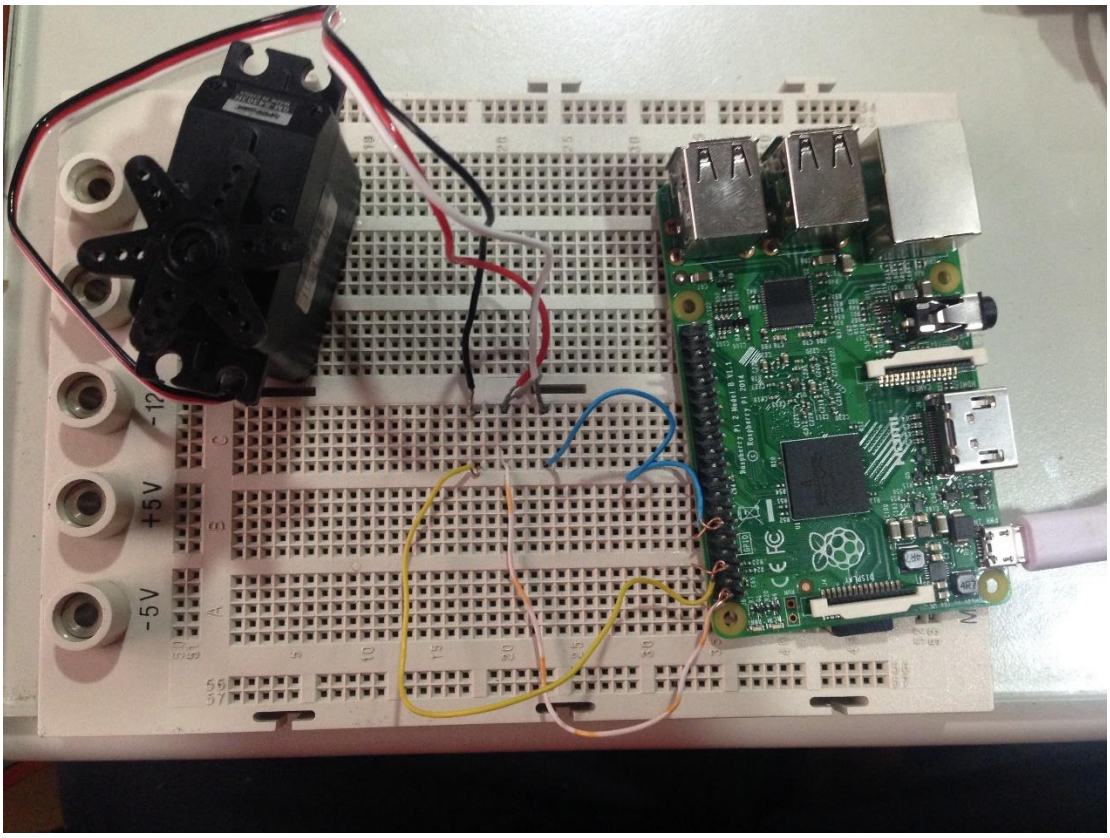


Figura 533: Conexión del servomotor con la tarjeta Raspberry Pi.

En el software deberemos de instalar un programa básico a través del terminal poniendo "sudo apt-get install –y wiringpi". Esto servirá, para que la Raspberry obedezca las órdenes que daremos desde el terminal para manejar el servo. Seguiremos estableciendo la frecuencia que tendrá el envío del "gpio", la cual deberá de ser de 50Hz, que es la que necesita el servomotor, para ello deberemos escribir estas tres líneas en el terminal.

```
gpio pwm - ms
gpio pwmc 192
gpio pwmr 2000
```

Y ya a partir de aquí solamente tenemos que dar las órdenes desde el terminal poniendo:

"gpio –g pwm 1 2"

Poniendo en 1 el número de gpio que estamos utilizando, y en 2, la velocidad a la que queremos que gire el servo que será un rango de entre 50 a 250.

6. Conclusiones

Para el tipo de motor DC brushed, hemos descubierto, que tiene una forma sencilla de controlar su velocidad, que es simplemente con la modificación de la tensión de alimentación. Tanto en este como en los demás motores que hemos utilizado, la corriente variará sobre todo por la impedancia que tiene en su interior el motor, además de la carga que va a soportar al girar el motor, debido a que se va a necesitar más par y por lo tanto, más intensidad en este.

El motor paso a paso es algo más complejo de controlar que el anterior, debido a que necesitamos los cambios de valores de tensión contantemente y en un breve tiempo.

Tenemos que decir que si no queremos utilizar el Shield motor, o el Gertbot para el control de estos motores anteriores, lo que debemos de realizar es una conexión aparte con un transistor característico para un motor determinado, o bien un puente-H, además de unas ciertas resistencias.

Y hemos visto la ventaja de utilizar el servomotor, ya que solo necesitamos una variable para su control total. Pero a la vez también tenemos nuestra dificultad para controlarlo con ciertos programas.

El potenciómetro utilizado, no ha sido el adecuado para ciertos ensayos, sobre todo para el servomotor, debido a que el arduino, no conseguía recibir los valores adecuados y correspondientes de cada posición de este.

6.1. Arduino

6.1.1. Arduino 1.8.1

A la hora de utilizar las tarjetas de adquisición para el control del motor DC brushed, no ha habido demasiados problemas. La dificultad no está en la variación de los valores de control del motor, sino en el conocimiento de las conexiones del motor en la placa, de la instalación de la placa en la computadora, y la instalación de drivers y de programas del mismo. En el caso del programa Arduino 1.8.1, se debe de tener un conocimiento previo de programación en C para poder controlarlo, además hay que decir que el programa necesita tiempo para poder volcar la información al arduino, sino se utiliza en la programación el monitor serial que nos ayudara a cambiar los valores mientras funciona y solo hará falta volcarlo una vez. Pero sin embargo, el programa una vez realizado es completamente fiable, y funciona a la perfección.

Para el motor paso a paso, es exactamente como hemos hablado anteriormente, es sencillo si tenemos el conocimiento previo de programación, y se necesita tener las ideas claras de cómo funciona el motor, para poder darles las ordenes adecuadas, además, del cambio de utilización de los pines del arduino. Pero por lo demás, los pulsos se realizan correctamente en el tiempo necesario y establecido, y funciona correctamente.

El servomotor, está todo bien establecido antes de comenzar a utilizarlo, y es igual para todos los servos. Hay que mencionar que para este motor, no se necesitaría la tarjeta de Shield motor, y utilizando el programa de Arduino 1.8.1 es todo muy sencillo, ya que al realizar su descarga, obtenemos también unos ejemplos de la utilización del programa, y uno de ellos es para el control de un servomotor.

6.1.2. Matlab

Para el programa de Matlab con el motor DC brushed, lo más aparatoso de utilizarlo es el peso del programa en nuestra computadora, que hará que el funcionamiento de nuestro ordenador sea lento y por lo tanto el programa también. El mayor inconveniente para utilizar este programa es el tiempo que tarda en trasladar la información de nuestra computadora a la tarjeta, que es mucho mayor al del arduino 1.8.1, pero una vez volcada esta información inicial, el cambio de valores de control en modo externo es bastante más rápido. Por eso mismo, hemos utilizado solo el arduino Mega en el programa Matlab, ya que con el arduino Uno al no poder utilizarse el modo externo el proceso de experimentación sería un proceso largo y tedioso. Podemos decir a favor de este programa, que no necesitamos un conocimiento previo de programación, y que es una visión del control mucho más gráfico, con la facilidad de utilizar bloques "scope" para ver si se enviaría una señal correcta al arduino.

Sin embargo para el motor paso a paso, hemos necesitado algo más de ingenio, sobre todo a la hora de utilizar el potenciómetro para poder controlar la velocidad de forma física y externa. En este caso ha existido un gran problema, debido a que una vez realizado la estructura de bloques, y mandada la información al arduino, no funcionaba el motor. Esto es debido a la velocidad que necesitaba el programa para enviar esos pulsos en un tiempo tan reducido. Y hemos podido arreglarlo gracias a la modificación del tamaño de paso fijo o "Fixed-step size". Podemos decir que el valor impuesto en este apartado es el adecuado, porque si lo aumentamos, quedará como estaba y no funcionará, y si lo disminuimos más, se podrá realizar un sobrecalentamiento del microcontrolador de la placa.

Para el servomotor, es muy sencillo, ya que solo utilizamos dos bloques. Pero el resultado no es el que queríamos, ya que no podemos controlar la velocidad, solo podremos arrancarlo, o pararlo.

6.2. Raspberry Pi

Para la tarjeta gráfica de Raspberry Pi para el motor DC brushed todo ha sido mucho más sencillo al existir el programa de Gertbot, ya que está especializado para este tipo de motores, dando la facilidad de utilizar rampas de tensión de inicio, y de parada. El único inconveniente que se le puede poner, es la instalación previa de la Raspberry, aunque no es demasiado complicado. Además la utilización de interruptores como hemos explicado antes a través de los pines, no es una instalación sencilla, y debe

de ser permanente, debido a que se debe de modificar físicamente algunos de los componentes de la tarjeta Gertbot, por ese motivo no lo hemos experimentado.

Si queremos manejar un motor paso a paso, será mucho más sencillo a pesar de que no está completa para la utilización de un motor de estas características. También se intentó realizar un ensayo con un motor paso a paso de 4 bobinas, porque esta placa es la única que acepta esa cantidad de motores, pero no funciona correctamente, debido a que no se puede establecer el orden de pulsos para las 4 bobinas a la vez, y si de dos en dos, como hemos demostrado.

Y para el servomotor, no hemos podido utilizar el Gertbot, ya que no es compatible para el servomotor, pero hemos podido controlarlo con la Raspberry en solitario, y otorgando impulsos a los pines de este. Es mejor que utilizar el Matlab, porque si podremos manejar el cambio de velocidad.

Finalmente podemos concluir diciendo que el programa Arduino 1.8.1 es el más adecuado de utilizar para el control de esos motores, ya que es el más compatibles para nuestros motores y el que mejor se adapta a los cambio que queremos realizar, pero sino tenemos ese conocimiento de programación, nos resultará más dificultoso. Simulink es el programa más fácil para comprender el control de los motores, a pesar de ser el menos flexible, y más errores se experimentan. Y la Raspberry es la forma más sencilla de utilizar los motores.

7. REFERENCIAS

- Stephen J. Chapman (2012). Máquinas eléctricas (5 ed.) México: Bogotá.

-Dawes (1991). Tratado de electricidad, 1 Corriente continua.

. Página oficial del programa Matlab

https://es.mathworks.com/

-Información sobre las pérdidas en el hierro de los motores.

https://es.scribd.com/doc/43461447/Perdidas-en-el-hierro

- Tipos de paso para el motor paso a paso.

http://server-die.alc.upv.es/asignaturas/lsed/2002-03/MotoresPasoaPaso/ftomotpap.htm

- Información de nuestro motor paso a paso.

http://es.rs-online.com/web/p/motores-paso-a-paso/5350401/

-Información de la placa Motor Shield

https://www.youtube.com/watch?v=PKcd_EuK_5c

https://www.youtube.com/watch?v=SNaWrEwgH20

-Información para la conexión de los motores con la placa Motor Shield

http://www.prometec.net/motor-shield-stepper/

http://www.instructables.com/id/Arduino-Motor-Shield-Tutorial/step6/Stepper-Motor/

-Datos sobre el funcionamiento de Arduino con Matlab

https://es.mathworks.com/company/newsletters/articles/motor-control-with-arduino-a-case-study-in-data-driven-modeling-and-control-design.html

-Comparación entre distintas placas Arduino

http://manueldelgadocrespo.blogspot.com.es/p/arduino-due.html

-Control del servomotor con Arduino.

https://www.luisllamas.es/controlar-un-servo-de-rotacion-continua-con-arduino/

-Tipos de Raspberry

https://www.raspberrypi.org/help/faqs/#introWhatIs

-Descarga de Raspbian

https://www.raspberrypi.org/downloads/

-Lista de configuración de la Raspberry

https://www.raspberrypi.org/documentation/configuration/raspi-config.md

-Tutorial para conectar la Raspberry a un ordenador corriente con Windows

http://www.frambuesapi.co/2013/09/09/tutorial-4-parte-2-conexion-del-raspberry-pi-directamente-al-pc/

https://www.youtube.com/watch?v=AJ7skYS5bjI

-Control remoto

https://raspberryparatorpes.net/instalacion/xrdp-escritorio-remoto-en-raspberry/

-Fijar dirección IP

https://raspberryparatorpes.net/instalacion/poner-la-direccin-ip-fija-en-raspbian/

https://raspberryparatorpes.net/instalacion/poner-la-direccin-ip-fija-en-raspbian-pixel-modo-grfico-gui/

-Página de descargas de la tarjeta Gertbot.

https://www.gertbot.com/

-Manual Gertbot. Rev 1.0, 8 Septiembre 2014. Copyright

-Manual GUI Gertbot. Rev 2.3, 27 Octubre 2014. Copyright

-Control del servomotor con shh de Raspberry

https://learn.adafruit.com/adafruits-raspberry-pi-lesson-8-using-a-servo-motor?view=all