UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

**TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALE**

# DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL PARA MOTORES DE CONTINUA MEDIANTE UN MICROCONTROLADOR ARDUINO


AUTOR: IGNACIO IRANZO JUAN

TUTOR: ALBERTO PÉREZ JIMÉNEZ

Seleccion NOMBRE DEL COTUTOR


**Curso Académico: 2020-21**

# Resumen

En el presente trabajo se ha abordado el reto de diseñar e implementar de un sistema de control que permita regular tanto la velocidad de rotación como el ángulo de giro de un motor de continua mediante un microprocesador Arduino.

Para conseguir dicho objetivo se optado por programar dos PID digitales en el Arduino, ejecutándose uno de ellos si se requiere controlar la posición, y otro si se requiere controlar la velocidad.

A fin de permitir la entrada de datos por parte del usuario se ha incluido un teclado y un encoder con pulsador (le permitirán configurar el controlador e introducir la referencia), también se ha incorporado una pantalla LCD que permite la comunicación controlador-usuario.

Además de las funcionalidades necesarias para cumplir con el objetivo de controlar la velocidad y ángulo de giro del motor de continua se ha optado por introducir otras como puede ser el control de la intensidad, tensión y potencia consumida.

Con tal de crear un dispositivo robusto y estético todos los elementos han sido introducidos en una caja.

# Resum

En el present treball s'ha abordat el repte de dissenyar i implementar d'un sistema de control que permeta regular tant la velocitat de rotació com l'angle de gir d'un motor de contínua mitjançant un microprocessador Arduino.

Per a aconseguir aquest objectiu s'optat per programar dos PID digitals en el Arduino, executant-seun d'ells si es requereix controlar la posició, i un altre si es requereix controlar la velocitat.

A fi de permetre l'entrada de dades per part de l'usuari s'ha inclòs un teclat i un encoder amb polsador (li permetran configurar el controlador i introduir la referència), també s'ha incorporat una pantalla LCD que permet la comunicació controlador-usuari.

A més de les funcionalitats necessàries per a complir amb l'objectiu de controlar la velocitat i angle de gir del motor de contínua s'ha optat per introduir altres com pot ser el control de la intensitat, tensió i potència consumida.

Amb la condició de crear un dispositiu robust i estètic tots els elements han sigut introduïts en una caixa.

# Abstract

In this project, it is taken the challenge of designing and implementing a control system that allows regulating both the rotation speed and the rotation angle of a DC motor by means of an Arduino microprocessor has been addressed.

To achieve this objective, it was decided to program two digital PIDs in the Arduino, executing one of them if it is required to control the position, and another if it is required to control the speed.

In order to allow data entry by the user, a keyboard and a push-button encoder have been included (they will allow the user to configure the controller and enter the reference), as well as an LCD screen that allows the communication system-user.

In addition to those functionalities necessary to meet the objective of controlling the speed and angle of rotation of the DC motor, it has been decided to introduce other such as the control of the current, voltage and power consumed.

To create a robust and aesthetic device, all the elements have been placed in a box.

# Índice general:

DOCUMENTO I: MEMORIA

ANEXO: CÓDIGO DE PROGRAMACIÓN

# Documento I:
# MEMORIA

# Índice

## Índice de las ilustraciones

# 1 Introducción

## 1.1 Objetivos

El objetivo principal de este TFG es el diseño e implementación de un sistema que permita al usuario controlar tanto la posición como la velocidad de giro en motores de corriente continua.

Para poder llegar a dicho objetivo final, se abordan los siguientes retos:

- Diseñar un sistema electrónico e informático que nos permita medir el giro de un motor de corriente continua.
- Diseñar una interfaz con el usuario.
- Diseño e implementación de un bucle de control.
- Construcción de una caja que permita contener todos los componentes electrónicos otorgándole robustez y una mejora estética al producto.

A fin de abordar todos estos retos va ser necesario aplicar los conocimientos adquiridos en la carrera en diversos campos como la informática, electrónica, mecánica, automática, programación… Además de ahondar en muchos de ellos con tal de adquirir herramientas nuevas que se puedan aplicar al proyecto.

## 1.2 Antecedentes

Una de las ventajas que tienen los motores de corriente continua respecto a los de corriente alterna es la relativa facilidad a la que se puede variar su velocidad de giro, para hacerlo simplemente basta con variar la tensión a la que son alimentados. Es por ello que de manera natural surge la necesidad de crear un sistema que sea capaz de controlar de manera precisa su velocidad y o posición.

Con el objetivo de lograr esto se ha optado por un bucle de control en lazo cerrado con un PID. Esta herramienta se emplea en la amplia mayoría de los casos en los que se aplica un control en lazo cerrado por su facilidad de uso y eficacia.

La herramienta sobre la que se va a realizar la programación necesaria para implementar el controlador va a ser una placa Arduino Mega 2560, las placas Arduino han ganado fama y usuarios en los últimos años, el ser un hardware y software libre los convierte en una solución barata y accesible a muchos usuarios, además de ser simple de usar y existir mucha información sobre ella en la red.

## 1.3 Justificación

Además de permitirnos plasmar los conocimientos adquiridos en gran parte de las asignaturas de la carrera y poder profundizar en muchos de ellos, este proyecto conlleva multitud de aplicaciones para el mundo laboral, sobre todo en la industria donde el control de procesos es

algo básico y necesario. La experiencia adquirida en el diseño e implementación de este controlador PID podrá extrapolarse a otros controles, no simplemente al de motores de corriente continua puesto que los métodos empleados para su diseño son muy parecidos a los de otros lazos de control (ya sean de temperatura, presión, caudal…).

# 2 Controlador PID

Lo que busca un controlador es que el proceso que controla se comporte de la manera que se desea, en este caso que nuestro motor llegue a la velocidad o posición requerida por el usuario (también llamado control por referencia), para conseguir esto, el controlador que se va a implementar es un controlador PID (empleado hasta en el 95% de los procesos de control en bucle cerrado) por su facilidad de programar y su efectividad. Pese a que su principal objetivo es que nuestro motor siga la referencia dictada por el usuario, también será capaz de corregir posibles perturbaciones en el sistema.

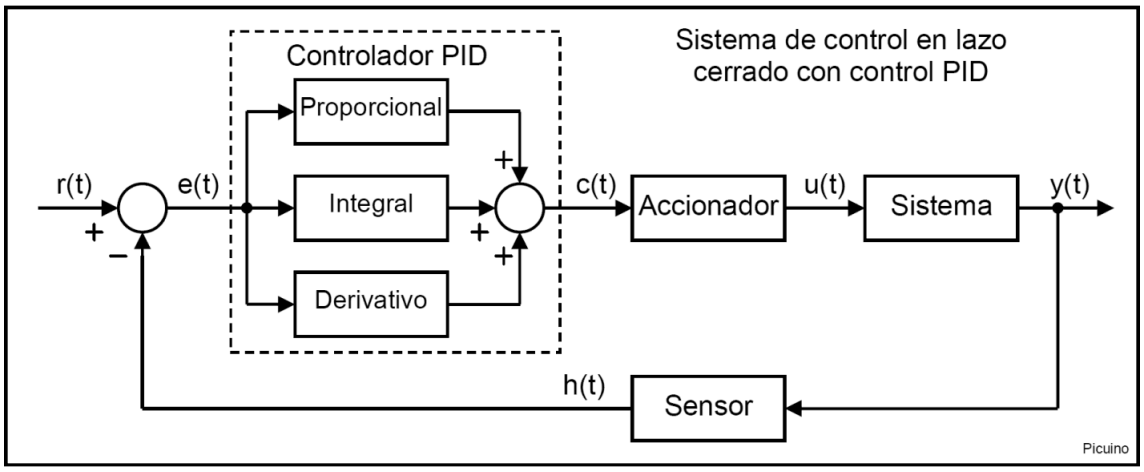



*Ilustración 1. Diagrama bucle cerrado PID. Fuente: Pardo (2021)*

Para que un controlador PID funcione va a necesitar una entrada denominada error, que debe de ser actualizada cada cierto tiempo (tiempo de muestreo) cuando se realiza la implementación digital. El error no es más que la diferencia entre el estado de referencia del sistema (introducido por el usuario), y el valor medido por el sensor, que nos indica el estado actual del sistema. Una vez este valor entra en el PID llega a las tres 'ramas' que lo conforman:

-Acción proporcional: Para obtener la salida multiplica el error por una constante (Kp), esta rama solo tiene en cuenta el estado actual del sistema, sin tener en cuenta cómo está cambiando el mismo a lo largo del tiempo. A medida que se incrementa el valor (Kp), el error en régimen permanente disminuye y se alcanza de manera más rápida, pero si este valor es demasiado elevado nuestro sistema tendrá sobreoscilaciones y puede llegar a tener problemas de inestabilidad.

Ec. continua:

$$u_p(t) = K_p \cdot e(t)$$

Ec. discreta:

$$u_p(k) = K_p \cdot e(k)$$

-Acción derivativa: Para obtener la salida considera la velocidad a la que varía el error (derivada del error), pudiendo en cierto modo predecir cómo será el error en el futuro. El aumento del valor de Kd otorgará al sistema una mayor estabilidad con una disminución de la rapidez a la que alcanzará el régimen permanente. Sin embargo, si este valor es demasiado elevado, puede producir sobreoscilaciones.

Ec. continua:

$$u_i(t) = K_d \cdot \frac{de(t)}{dt}$$

Ec. discreta:

$$u_i(t) = \frac{e(k) - e(k-1)}{T} \cdot K_p \cdot T_d$$

-Acción integral: En el cálculo de la salida considera el error acumulado (integral del error), su objetivo es asegurar que el error permanente sea 0. Un valor demasiado elevado de Ki producirá sobreoscilaciones.

Ec. continua:

$$u_i(t) = K_i \cdot \int_0^t e\,(t)dt$$

Ec. discreta:

$$u_i(t) = \frac{Kp \cdot T}{Ti} \cdot e(k) + u_i(k\text{-}1)$$

Tras el cálculo del valor de la salida de cada una de las tres ramas, estos se suman para obtener la salida total del PID. Como se puede observar, existen ciertos valores a la hora de configurar el PID que van a tener que ser ajustados dependiendo del sistema que se vaya a controlar (en este caso dependiendo de las características del motor que se deba controlar), estas constantes serán la Kp, Ti, Td y T.

El método Ziegler-Nichols permitirá obtener los valores de Kp, Ti y Td de una forma empírica lo cual ahorra una gran cantidad de cálculos (obtención de las funciones de transferencia del motor y del controlador PID). Puesto que se quieren controlar dos sistemas diferentes (la velocidad y la posición de nuestro motor), va a ser necesario el diseño de dos controladores PID distintos.

## 2.1 Método 1 Ziegler-Nichols

Para aplicar este método se debe obtener una representación gráfica de nuestro sistema ante una entrada del tipo escalón en bucle abierto.

El método 1 de Ziegler-Nichols es efectivo para controlar sistemas que tienen un comportamiento en forma de S, como es el caso de la velocidad de giro de un motor DC.

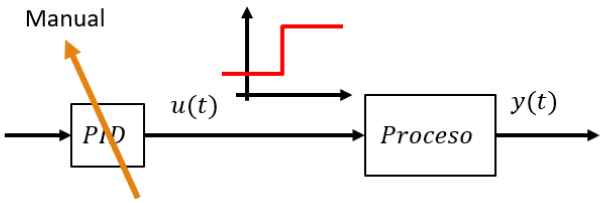



*Ilustración 3. Diagrama entrada escalón bucle abierto. Fuente: Giraldo (2019)*

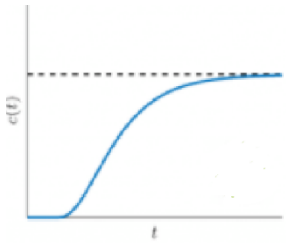


*Ilustración 2. Respuesta en forma de S. Fuente: Giraldo (2019)*

Para obtener el valor de las constantes del PID a implementar se deben obtener tres valores de la gráfica (K, L y T) tal y como muestra la figura siguiente:

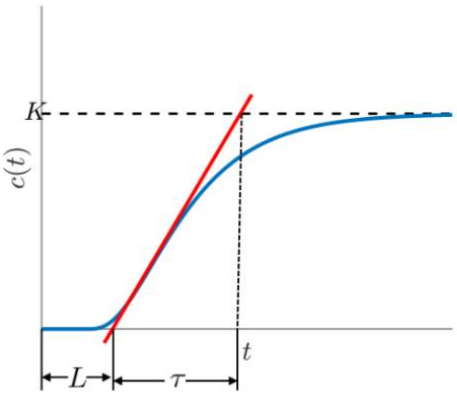



*Ilustración 4. Toma de parámetros Z-N método 1. Fuente: Giraldo (2019)*

Una vez obtenidos dichos parámetros aplicamos las siguientes tablas para obtener la implementación digital de nuestro PID.

| | Parámetro | Método Escalón (B.Abierto) | Respuesta Sostenida (B.Cerrado) |
|---|---|---|---|
| P | *Kp* | $\frac{T}{K*L}$ | 0,5*Kc |
| PI | Kpi | $0{,}9*\frac{T}{K*L}$ | 0,4*Kc |
| | Ti | 3*L | 0,8*Tc |
| PID | Kpid | $1{,}2*\frac{T}{K*L}$ | 0,6*Kc |
| | Ti | 2*L | 0,5*Tc |
| | Td | $\frac{L}{2}$ | 0,125*Tc |

*Ilustración 5. Tabla parametrización PID. Fuente: Elaboración propia a partir de Fernández (2021)*

| | p0 | p1 | p2 |
|---|---|---|---|
| P | $Kp$ | 0 | 0 |
| PD | $Kpd * \frac{Ts+Td}{Ts}$ | $-Kpd * \frac{Td}{Ts}$ | 0 |
| PI | $Kpi$ | $Kpi * \frac{Ts-Ti}{Ti}$ | 0 |
| PID | $Kpid * \frac{Ts+Td}{Ts}$ | $Kpid * (-1 + \frac{Ts}{Ti} - 2 * \frac{Td}{Ts})$ | $Kpid * \frac{Td}{Ts}$ |

*Ilustración 6. Tabla digitalización PID. Fuente: Elaboración propia a partir de Fernández (2021)*

En este caso la implementación digital del PID será:

$u(k)=p_0e(k) + p_1e(k-1) + p_2e(k-2) + u(k-1)$

Una vez obtenidos los valores preliminares de las constantes K, Ti y Td, se realizarán diferentes pruebas para comprobar su funcionamiento, variando si es necesario dichos valores en fin de afinar el controlador.

## 2.2 Método 2 Ziegler-Nichols

El método 2 de Ziegler-Nichols es el método empleado para sistemas que no tienen comportamiento tipo S en bucle abierto ante entrada de tipo escalón (como es el caso de la posición de un motor DC ante una entrada en escalón de voltaje, siendo su respuesta una rampa), es por ello que se debe implementar un bucle cerrado.

Para obtener la gráfica deseada, el PID de control debe ser de tipo proporcional, con una Kp que debe ser elevada hasta que se obtenga una respuesta oscilatoria de amplitud constante. De esta manera se obtendrá el valor de dos variables Pu (periodo entre cada oscilación) y Ku (ganancia necesaria para que nuestro sistema entre en un estado oscilatorio de amplitud constante).

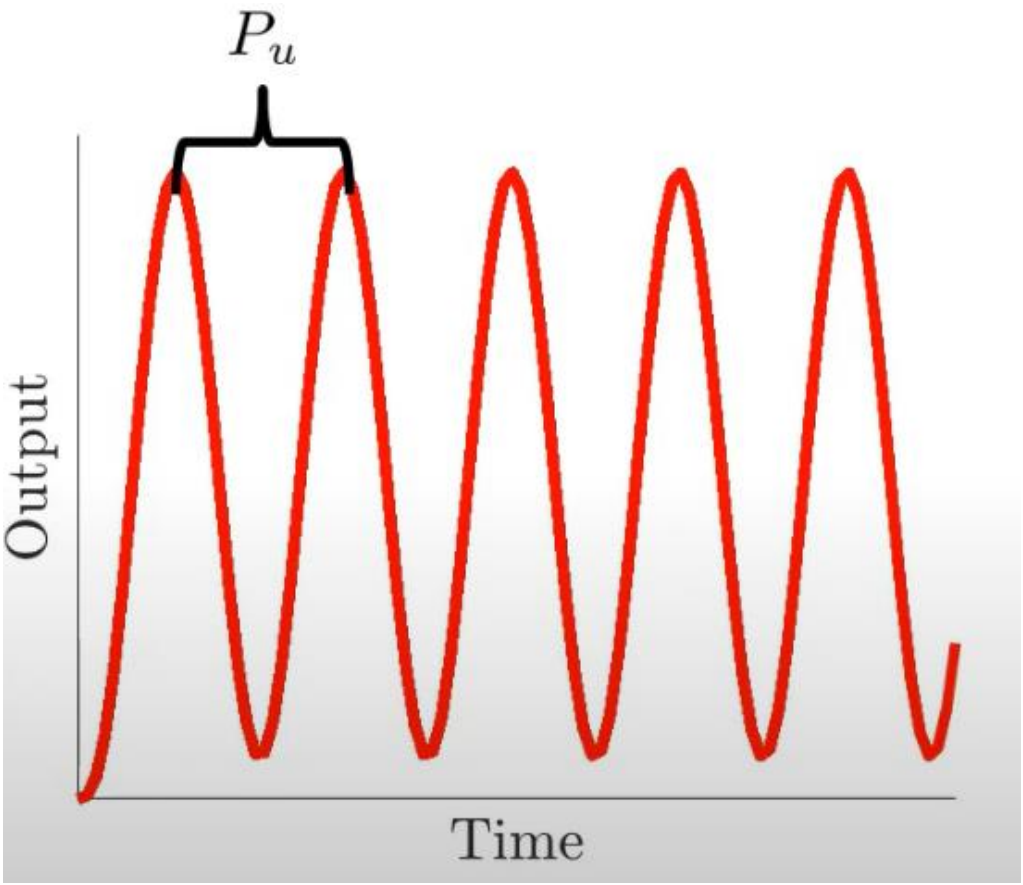



*Ilustración 7. Toma de parámetros Z-N método 2. Fuente: Giraldo (2019)*

Una vez obtenidos estos valores se aplican las siguientes tablas para obtener la ecuación preliminar del PID a implementar.

| | Parámetro | Método Escalón (B.Abierto) | Respuesta Sostenida (B.Cerrado) |
|---|---|---|---|
| P | *Kp* | $\frac{T}{K*L}$ | 0,5*Kc |
| PI | Kpi | $0{,}9*\frac{T}{K*L}$ | 0,4*Kc |
| | Ti | 3*L | 0,8*Tc |
| PID | Kpid | $1{,}2*\frac{T}{K*L}$ | 0,6*Kc |
| | Ti | 2*L | 0,5*Tc |
| | Td | $\frac{L}{2}$ | 0,125*Tc |

*Ilustración 8. Tabla parametrización PID. Fuente: Elaboración propia a partir de Fernández (2021)*

| | Parámetro | Método Escalón (B.Abierto) | Respuesta Sostenida (B.Cerrado) |
|---|---|---|---|
| **P** | *Kp* | $\frac{T}{K * L}$ | 0,5*Kc |
| **PI** | Kpi | $0{,}9*\frac{T}{K*L}$ | 0,4*Kc |
| | Ti | 3*L | 0,8*Tc |
| **PID** | Kpid | $1{,}2*\frac{T}{K*L}$ | 0,6*Kc |
| | Ti | 2*L | 0,5*Tc |
| | Td | $\frac{L}{2}$ | 0,125*Tc |

*Ilustración 9. Tabla digitalización PID. Fuente: Elaboración propia a partir de Fernández (2021)*

En este caso la implementación digital del PID será:

$u(k)=p_0e(k) + p_1e(k-1) + p_2e(k-2) + u(k-1)$

Una vez obtenidos los valores preliminares de las constantes K, Ti y Td, se realizarán diferentes pruebas para comprobar su funcionamiento, variando si es necesario dichos valores en fin de afinar el controlador.

# 3 Hardware

## 3.1 Componentes

### 3.1.1 Arduino MEGA 2560


*Ilustración 10. Arduino MEGA 2560. Fuente: Arduino.cc (2021)*

La Arduino MEGA 2560 es una placa electrónica basada en el microcontrolador ATmega2560, tanto el hardware como el software de la placa con libres y sencillos de usar.

Las características principales de la placa son:

- 54 pines digitales de entrada/salida (15 de ellos pueden emplearse como salidas PWM y 6 como interrupciones).
- 16 pines de entradas analógicas.
- 4 puertos serie.
- Voltaje de operación de 5V.
- El rango de voltajes de entrada máximo y mínimo es de 6V a 20V, aunque el recomendado es de 7V a 12V.
- Memoria Flash de 256 KB.
- Superficie 101.52 mm x 53.3 mm.
- Peso de 37g.
- Botón de Reset.
- Entrada Jack de alimentación.
- Puerto conexión USB (tanto para alimentación como para transferencia de datos, por él se carga el programa en la placa).
- Frecuencia del reloj de 16MHz.
- 2 pines que soportan comunicación I2C (SDA y SCL).

Algunas de las posibles alternativas a esta placa hubiesen sido:

- Arduino Uno: Pese a ser una alternativa que hubiese tenido prácticamente las mismas características que la opción MEGA 2560 a un precio más barato y con un tamaño más reducido, existe una diferencia que hace que no sea viable su uso, y es que para la realización del programa ha sido necesario incluir cuatro señales provenientes del hardware como interrupciones, por lo tanto nuestra placa debía incluir un mínimo de 4 pines que pudiesen ser programados cocn este fin, y la placa Uno cuenta solo con 2.
- Raspberri Pi: Otra posible alternativa a la placa empleada hubiese sido la Raspberri Pi, y si bien es cierto que hubiese sido eficaz a la hora de realizar el proyecto, no hubiese sido para nada eficiente, la placa Raspberri Pi es básicamente un ordenador, y fue diseñada para la realización de proyectos más complejos, lo que se ve reflejado en su precio y complejidad de uso (ambos mayores que en la Arduino MEGA 2560).

### 3.1.2 Motor DC (+ encoder cuadratura)

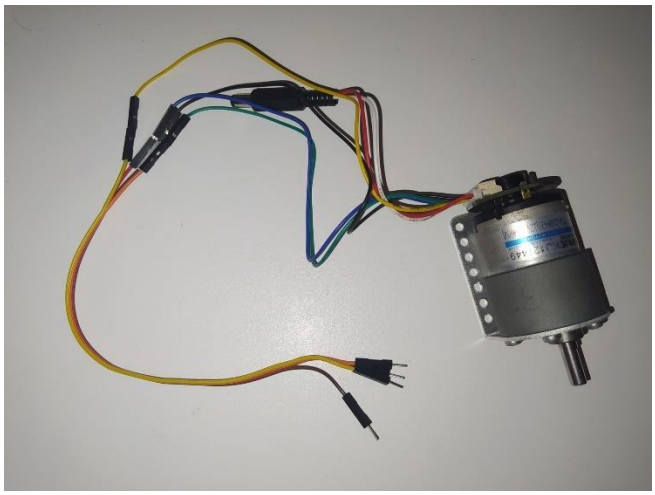


*Ilustración 11 Motor CC + encoder cuadratura*

- Motor DC

  Éste será el elemento a controlar (como se verá más adelante, no tendrá que ser este motor en concreto, sino que el controlador tendrá un abanico de posibles motores a los que se pueda aplicar), un motor de corriente continua no es más que una máquina que convierte energía eléctrica en energía mecánica (reflejada en la rotación de un rotor). A diferencia de los motores de corriente alterna síncronos cuya velocidad de giro viene determinada por la frecuencia de la alimentación, la velocidad de giro de los motores DC, viene dada por la tensión a la que se alimentan. Para una misma carga

(par motor), cuanto mayor sea la tensión de alimentación, mayor será la velocidad de giro del motor.

Las características de este tipo de motores los hacen ideales para aplicaciones donde los voltajes de alimentación no sean elevados y se deban realizar cambios en la velocidad de giro del motor.

Especificaciones de motor:
-Voltaje de alimentación (Máx.): 12V
-Vel. Máx. sin carga: 320 RPM a 0.15A
-Máx. potencia: 4.0kg.cm/160RPM/6.7W/1.2A
-Reductora: 1 : 30

La alimentación del motor se realizará por los pines M1 y M2, que han sido conectados a un conector de tipo jack macho.

- Encoder rotativo

Se utilizará para poder medir la velocidad de giro y posición del motor.

Especificaciones:

- Nº de polos: 22 polos.
- Pulsos por revolución: 11x30 = 330 PPR.

El encoder cuenta con 4 pines:

- Dos pines para su alimentación: serán los pines VCC y GND, deben conectarse a 5V y a ground respectivamente.
- Dos pines de salida: serán los pines C1 y C2 cada uno de ellos proporcionando una señal digital cuadrada (de ahí el nombre encoder de cuadratura) desfasadas una respecto a la otra 90 grados.

Para poder justificar la utilización de este componente y comprender el código empleado para poder obtener valores útiles de sus lecturas es necesario tener una comprensión básica de su funcionamiento:

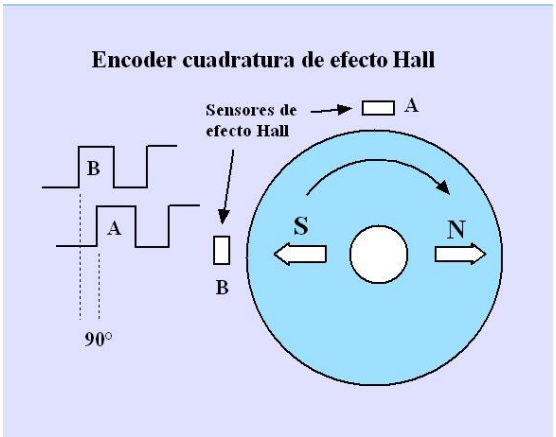



*Ilustración 12. Funcionamiento encoder cuadratura. Fuente: Flotante (s.f.)*

Cada salida del encoder es la salida de uno de los dos sensores de efecto Hall que contiene, producen una señal eléctrica ante la influencia de una campo magnético variable (provocado por la rotación de un imán de ferrita) de esta manera, la frecuencia de la señal cuadrada proporcionada por cada uno de los dos sensores será la frecuencia de giro del imán multiplicado por el número de polos que contiene (el encoder de estudio contiene un imán de ferrita con 22 polos).

El contar tan solo con un sensor de efecto Hall sería suficiente para poder obtener la velocidad o ángulo de giro del motor, pero no se podría saber el sentido, es para ello para lo que se introduce un segundo sensor, desfasado 90º del primero. De esta manera, dependiendo de que cuál de las dos señales se active primero, se podrá saber el sentido de giro:

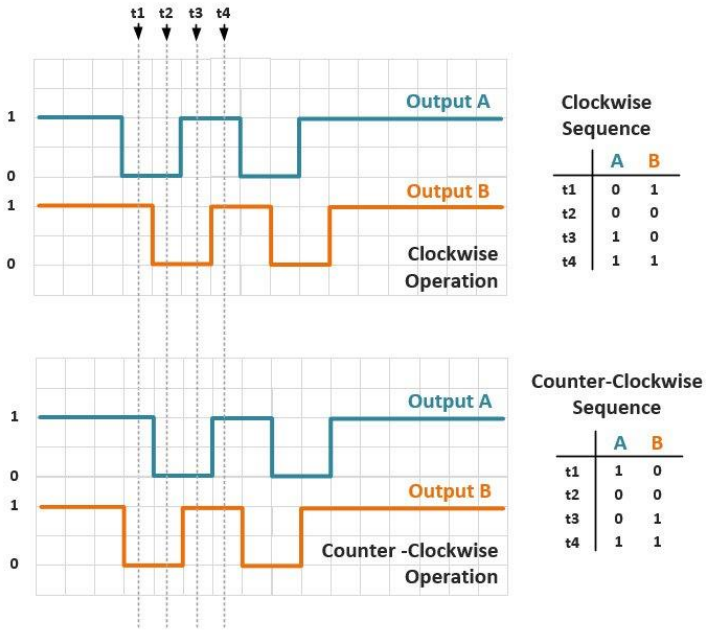



*Ilustración 13. Señal encoder rotativo. Fuente: NPH22 (2021)*

Como se verá más adelante, el tener dos señales desfasadas 90º nos duplicará la resolución del sensor.

### 3.1.3 Driver L298N

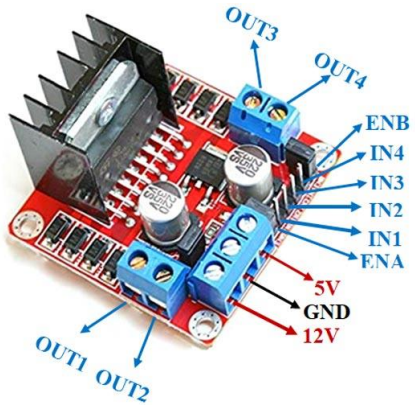



*Ilustración 14. Driver L298N. Fuente: components101 (2021)*

El Driver L298N permite controlar tanto la velocidad como el sentido de giro del motor. El chip L298N cuenta con dos puentes en H completos que permiten controlar un motor paso a paso o dos motores de corriente continua (como se verá a continuación puede configurarse para que controle un solo motor permitiendo administrarle hasta el doble de la potencia máxima recomendada para el driver).

Especificaciones:

- Chip L298N
- 2 canales
- Alimentación parte lógica a 5V
- Voltaje de potencia 5-35V
- Intensidad máxima 2A (picos de 3A)
- Potencia máxima de 25W (puede configurarse para 50W)
- Dimensiones de 43x43x27mm
- Peso de 30g

El driver cuenta con los siguientes pines:

- Pines de entrada:
  - ENA y ENB controlarán el voltaje que llega a los canales 1 y 2 respectivamente, recibirán una señal PWM de la placa Arduino MEGA 2560.
  - Pines IN1, IN2, IN3 y IN4 recibirán una señal digital HIGH o LOW, la cual determinara si la salida OUT1, OUT2, OUT3 y OUT4 será 0V o los voltios indicados por ENB.
  - +12V alimentará la parte de potencia pudiendo variar el voltaje de alimentación de 5 a 35V, será alimentado por un adaptador de corriente.

- +5V alimentará la parte lógica este pin será alimentado por la placa Arduino MEGA 2560, existe la posibilidad de configurar el driver para que la propia alimentación de la parte de potencia alimente la parte lógica, pero tan solo puede emplearse este método con voltajes de alimentación de hasta 12V, puesto que el objetivo es aprovechar el máximo potencial del driver, para poder tener un abanico más grande de motores que pueda controlar se ha optado por no elegir esta opción.
- GND, debe conectarse al GND, tanto lógico como de la fuente de intensidad.

- Pines de salida:
  - OUT1, OUT2, OUT3 y OUT4 serán las salidas que alimentarán los motores, se activará o no cada una de ellas dependiendo del estado de los pines digitales IN1, IN2, IN3 y IN4 respectivamente.

Puesto que el driver empleado permite controlar hasta dos motores DC, pero nuestra aplicación tan solo emplea uno se ha configurado para poder aprovecharlo al máximo.

El objetivo será que la potencia suministrada al motor provenga de ambos canales, y no de tan solo uno de ellos, pudiendo así duplicar tanto la intensidad máxima que es capaz de proporcionar al motor, como la potencia máxima, esto se conseguirá de la siguiente manera:

- Realizar dos puentes en los pines de salida, uno que conecte OUT1 con OUT4, y otro OUT2 con OUT3, de esta manera el motor será alimentado por ambos canales.
- Realizar tres puentes en los pines de entrada que nos aseguren que tanto OUT1 y OUT4 como OUT3 y OUT2 tengan el mismo voltaje, de lo contrario podríamos producir un cortocircuito, o como mínimo que los dos canales no alimentasen por igual el motor. Esto se conseguirá puenteando ENB y ENA, IN4 y IN1, y por último IN3 y IN2.

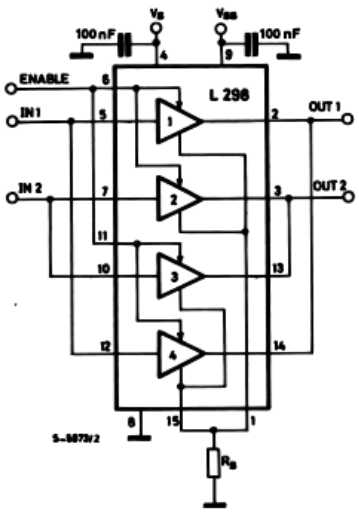



*Ilustración 15. Conectar en paralelo ambos canales. Fuente: STMicroelectronics (2000)*

Estos cambios son sencillos de implementar y permitirá sacarle el máximo partido al driver.

### 3.1.4 Pantalla LCD (+ adaptador IIC)

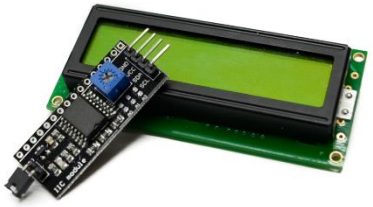


*Ilustración 16. Pantalla LCD + adaptador IIC. Fuente: inven (s.f.)*

Dispositivo clave para la interfaz con el usuario, le permitirá configurar algunos parámetros del PID y a través de ella se le mostraran diversos datos, ya sea estado del motor, referencia que ha introducido…

Especificaciones:

- Alimentación: 5V
- Filas: 2
- Caracteres por fila: 16
- Tamaño: 80x36x12 mm

No se trata de una pantalla muy grande ni de una alta resolución, pero es más que suficiente para desempeñar la función que se desea. El utilizar un adaptador IIC facilita mucho la conexión de la pantalla con la placa Arduino, puesto que en vez de 16 pines, emplearemos 4:

- 2 pines serán la alimentación, VCC que se deberá conectar a 5V de la placa Arduino, y GND que se deberá conectar al ground de la placa.
- 2 pines para la comunicación, los pines SDA y SCL que se deberán conectar a los pines con el mismo nombre de la placa Arduino.

### 3.1.5 Encoder rotativo con pulsador


*Ilustración 17. Encoder rotativo con pulsador. Fuente: iberobotics (2021)*

Segundo dispositivo empleado para generar una interfaz con el usuario será empleado para cambiar la referencia del sistema (uno de los dos métodos empleados).

Su funcionamiento sigue el mismo principio que el del encoder incorporado en el motor, a diferencia de que este incorpora un botón, lo cual añade una salida más a este componente, lo cual hace que tenga un total de 5 pines:

- Dos pines para su alimentación: serán los pines VCC y GND, deben conectarse a 5V y a 0V respectivamente.
- Dos pines de salida del encoder: serán los pines CLK y DT cada uno de ellos proporcionando una señal digital cuadrada (de ahí el nombre encoder de cuadratura) desfasadas una respecto a la otra 90 grados.
- Un pin de salida del botón, denominado SW, la señal es 1 cuando no se pulsa el botón, y 0 al ser pulsado.

Una alternativa que nos hubiese dado la opción de hacer cambios en la referencia de manera incremental (incrementa o disminuye el valor de la referencia a medida que se gira el encoder) hubiese sido un potenciómetro, que también hubiese sido más barato, pero lo que hubiésemos ganado en costes y sencillez lo hubiésemos pagado en menor precisión a la hora de introducir la referencia (se hubiese tenido que hacer una lectura analógica del potenciómetro, y a esa lectura aplicarle una función que nos diese una posible referencia dentro de un campo ya predefinido…). Siendo la referencia un dato clave para el funcionamiento de un controlador PID, se ha decidido invertir en un encoder para realizar esta tarea.

### 3.1.6 Keypad 4x4

*Ilustración 18. Keypad 4x4. Fuente: elfa (2021)*

Se trata de un teclado de 16 teclas, 4 filas y 4 columnas, junto con la pantalla y el encoder conforman los elementos empleados para la interfaz con el usuario. El keypad contiene 8 pines, 4 correspondientes a cada una de las 4 columnas, y otros 4 correspondientes a cada una de las 4 filas. Estos 8 pines deben ser conectados a 8 pines digitales de la placa Arduino a fin de obtener lecturas del componente.

El teclado se empleará para que el usuario pueda realizar una configuración rápida del PID una vez comience el programa, y para si así lo indica, pueda hacer cambios en la referencia a través de este.

### 3.1.7 Pulsador

*Ilustración 19. Pulsador. Fuente: cespedes (s.f.)*

El pulsador se empleará para que el usuario pueda resetear el programa en caso de que lo vea necesario (desea hacer cambios en la configuración del PID, cambio de la variable manipulada (velocidad o posición), cambio en el método de introducción de la referencia (teclado o encoder), o simplemente haya surgido algún problema y quiera cortar la alimentación al motor).

El pulsador al ser presionado unirá el neutro con el pin Reset de la placa Arduino, lo que hará que el programa se resetee y vuelva a preguntar al usuario sobre la configuración del PID.

### 3.1.8 Sensor de corriente ACS712

*Ilustración 20. Sensor de corriente ACS715. Fuente: Brico Geek (2021)*

Conectado en serie con el motor para medir la corriente que circula por este.

El sensor tiene 5 pines:

- 2 pines de alimentación: 5V, que se conectará a los 5V de la placa Arduino, y GND, que se conectará al ground de la placa.
- 1 pin de salida: se trata de una salida analógica cuyo voltaje será proporcional a la corriente que circula por el sensor, se conectará a un pin de entrada analógica de la placa Arduino,
- 2 pines de paso de corriente: Deben conectarse en serie al elemento del cual se desea conocer la intensidad que lo atraviesa, puesto que la lectura que ofrece el sensor tiene signo deberemos de ser conscientes de ello para saber el sentido en el que conectamos el sensor.

Especificaciones:

- Rango de lectura: -5 / 5A
- Centro: 2.5V
- Sensibilidad: 185mV/A

### 3.1.9 Resistencias

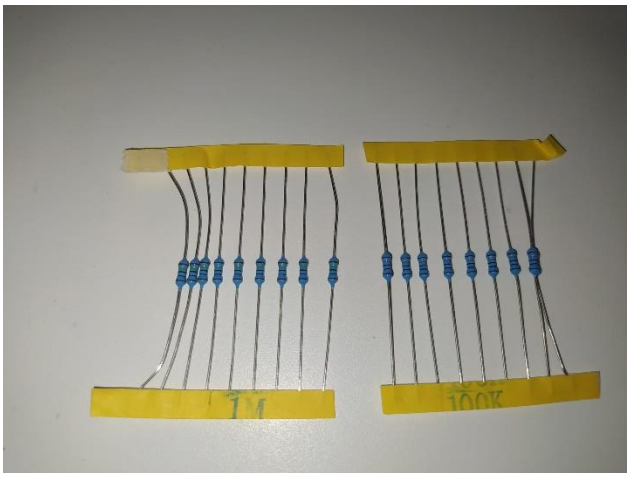



*Ilustración 21. Resistencias 1M y 100K Ω*

Se emplearán dos resistencias, una de 1MΩ y otra de 100KΩ, con el objetivo de obtener un sensor de tensión, ambas resistencias en serie se pondrán el paralelo a la alimentación del driver, esto evitará que se empleen tensiones excesivas (mayores de 35V) para alimentar el controlador.

Para poder obtener la tensión, lo que se hará es la lectura de la caída de tensión que se producirá en la resistencia de 100KΩ, conectando un borne a GND y el otro a una entrada analógica de la placa, al multiplicar este valor por 11, obtendremos la rensión de la alimentación.

Se emplean estas dos resistencias ya que consumirán muy poca potencia, en el caso más desfavorable (que se esté alimentando con 35V), consumirán 1,11 mW.

### 3.1.10 Cable USB

*Ilustración 22. Cable USB Tipo B 2.0. Fuente: taloselectronics (2021)*

Será necesario tanto para la comunicación computadora-Arduino como para la alimentación de la placa, si se desea puede sustituirse por otro de mayor largarie.

### 3.1.11 Cableado

*Ilustración 23. Cableado*

Se emplearán dos tipos de cables para hacer todas las conexiones:

- En primer lugar cables Dupont para todas las conexiones de la parte lógica de nuestro prototipo (por los que van a circular intensidades del orden de los miliamperios).
- En segundo lugar, cables de 1mm de diámetro por los que van a circular intensidades de 0 a 4 amperios.

### 3.1.12 Adaptador de corriente

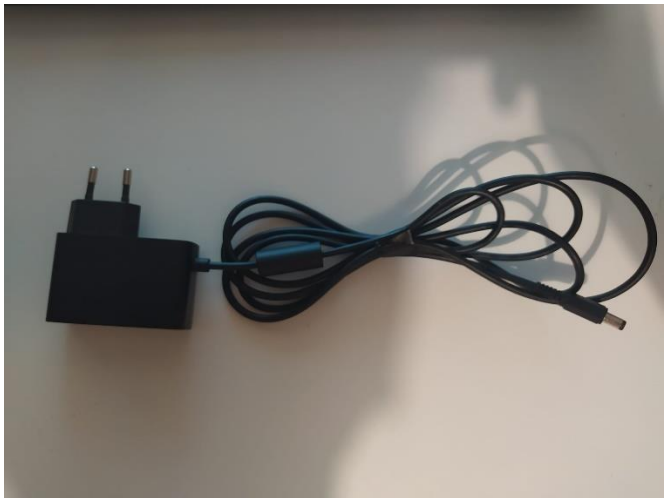


*Ilustración 24. Adaptador de corriente*

Específico para el motor que se va a emplear, el adaptador convertirá los 230V de alterna de la red a 12V de continua que alimentarán el driver, que a su vez alimentará al motor. El extremo que proporciona la alimentación DC ha sido conectado a un conector de tipo Jack macho.

### 3.1.13 Conectores tipo Jack

*Ilustración 26 Conector Jack macho. Fuente: 12v14v products (2021)*

*Ilustración 25. Conector Jack hembra. Fuente: sfond294 (2021)*

Harán las conexiones mucho más robustas y seguras, se emplearán dos conectores macho, y dos conectores hembra, una conexión será la de adaptador de corriente-driver y la otra driver-motor.

### 3.1.14 Caja de conexiones

*Ilustración 27. Caja de conexiones. Fuente: Bauhaus (2021)*

Este elemento contendrá a todos los demás, haciendo que el prototipo sea más fácil de transportar, más robusto y estético. Se trata de una caja de conexiones estanca, a la que se le han realizado diferentes ranuras y agujeros con tal de albergar los demás componentes (la distribución y montaje de estos en la caja se explicará en el subapartado de montaje).

Especificaciones:

- Dimensiones: 230x150x85 mm
- Antiestática e impermeable
- Material: plástico ABS
- Peso: 407 g

Alternativas:

Una posible alternativa hubiese sido el diseñar e imprimir con una impresora 3D una caja específica para el prototipo. La idea fue barajada, pero finalmente se desechó puesto que las posibles ventajas no compensaban el tiempo y recursos que hubiesen sido necesarios invertir. Por otro lado, al emplear la caja de conexiones simplemente hubo que agujerearla y realizarle el ranurado con una miniamoladora.

### 3.1.15 Placa de prototipos

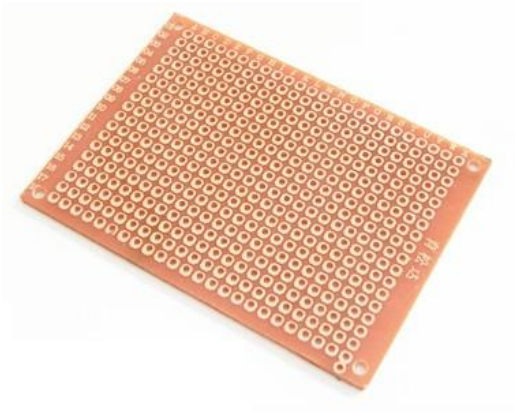


*Ilustración 28. Placa perforada. Fuente: Cetronic (s.f.)*

Placa de circuito impreso perforada, en ella estarán conectados los componentes electrónicos que se encuentren en el interior de la caja, esto evita que puedan producirse desconexiones al transportar el prototipo, es decir, es otro elemento que proporciona robustez.

Especificaciones:

- Tamaño: 100x160 mm
- Paso entre orificios: 2.54 mm

## 3.2 Montaje

A la hora de realizar el montaje de los elementos que conforman el prototipo en la caja se han tenido que hacer tres distinciones:


*Ilustración 29. Vista conexiones de alimentación del controlador y del motor (izq) y vista conexiones del encocer del motor (der)*

- Elementos que se no se encuentran en la caja: Se trata de el convertidor de corriente y del motor, ambos elementos se encuentran en el exterior ya que deben de ser fácilmente sustituibles en caso de que se desee controlar un motor distinto evidentemente, se deberá cambiar el motor, y también el convertidor de corriente así lo requiere el nuevo motor. Es por ello que la caja dispondrá de dos conectores de tipo jack hembra en su exterior, permitiendo la conexión de ambos elementos con el driver. Los pines del encoder del motor podrán conectarse al extremo de cuatro cables Dupont ubicados en una de las paredes de la caja.


*Ilustración 30. Vista de la tapa de la caja con el teclado, encoder y pantalla*

- Elementos que conforman la interfaz con el usuario: Se trata del teclado, la pantalla, el encoder y el pulsador. Todos estos elementos se encontrarán en la parte exterior de la caja, incrustados en ella, de manera que sean accesibles para el usuario. Para ello se han creado ranuras con las dimensiones adecuadas para cada uno de ellos, y se encuentran sujetos ya sea a la tapa o a las paredes de la caja.

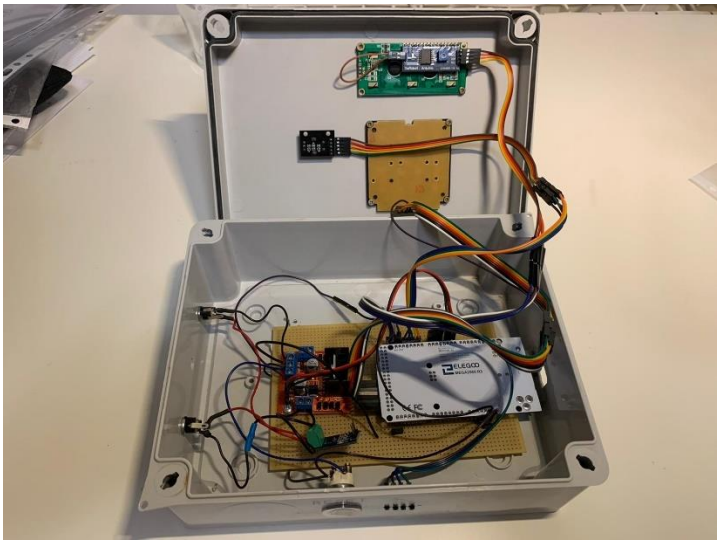


*Ilustración 31. Vista del interior de la caja, con el resto de componentes y conexiones*

- Elementos que se encuentran en el interior de la caja: Todos los demás elementos se encuentran en el interior de la caja, tan solo la placa Arduino requerirá conexión con el exterior (además del driver como se ha mencionado antes), es por ello que la conexión USB se encontrará en un orificio, permitiendo así su conexión con la computadora y o fuente de alimentación para la placa.

# 4 Programación

## 4.1 Software

Para la programación de la placa se ha empleado el programa, se ha empleado el programa Arduino. Como se ha dicho con anterioridad cuando se ha hablado de la placa Arduino Mega 2560, Arduino es un software libre fácil de utilizar y de el que se puede obtener mucha información y cursos gratis en internet.

Al ser un software libre los usuarios han realizado multitud de librerías que se pueden encontrar y descargar fácilmente de la red, como se verá más adelante, esto va a resultarnos de mucha utilidad a la hora de manejar ciertos componentes o implementar ciertas funciones.

## 4.2 Código para cada componente

### 4.2.1 Driver

Sencillo de manejar, sus pines IN3 e IN4I se declaran como salidas digitales, tras esto, dependiendo de si el estado de uno u otro es HIGH o LOW, podremos controlar el sentido de giro del motor.

Su tercer pin será ENB, este debe ser declarado como una salida analógica, y ser conesctado a un pin digital PWM, una vez hecho esto, dependiendo de el valor (entre 0 y 255), se regulará la tensión que le llega al motor, y con ello su velocidad de giro.

### 4.2.2 Sensor de corriente ACS712

Bastará con leer su señal de salida (OUT), esta deberá estar conectada a un pin analógico, y ser declarado como entrada analógica. Una vez leída esta señal se le debe aplicar la función de transferencia que nos permita pasar su lectura a Amperios.

La función de transferencia, siendo OUT la lectura proporcionada por el sensor, será:

I=abs((5.0*(OUT/1023.0)-2.5)/0.185)

Tomaremos el valor absoluto, ya que es el que nos interesa para saber si se está produciendo una sobrecarga, y su signo cambiará según el sentido de paso de la corriente (dato que no necesitamos para esta aplicación).

### 4.2.3 Pantalla LCD

Para facilitar su manejo se ha incorporado a nuestro programa las librerías <LiquidCrystal_I2C.h> y <Wire.h>.

De ese modo la escritura en la pantalla será tan sencilla como escribir **lcd.print("Lo que se desee escribir")** y su borrado **lcd.clear()**.

### 4.2.4 Keypad 4x4

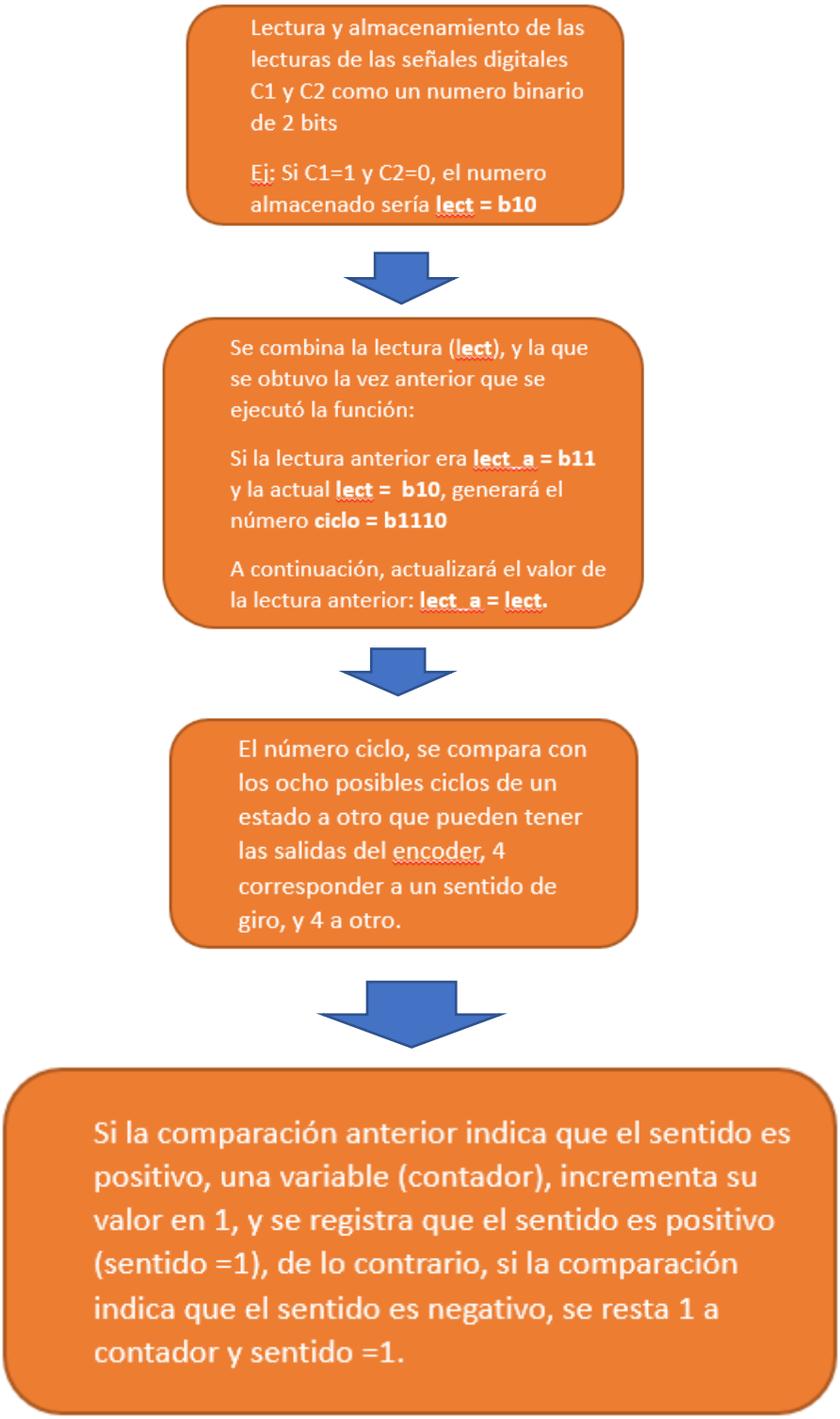
Otro de los componentes en los que nos podemos beneficiar de incluir una librería, en este caso la librería #include <Keypad.h>.

Tras incluirla, simplemente deberemos indicar qué pines se han empleado para la conexión del teclado, y qué carácter corresponde a cada tecla.

La lectura se realizará con el comando **teclado.getKey(),** el cuál devolverá el carácter correspondiente tecla que se ha empleado, o **NO_KEY** si no se ha pulsado ninguna.

### 4.2.5 Encoder incorporado en el motor

Para la programación de este dispositivo se han empleado interrupción, una herramienta que se permite ligar a la señal que entra por ciertos pines de la placa, en este caso se ha indicado que produzca una interrupción tipo **CHANGE** es decir si las señales provenientes de las salidas C1 o C2 den encoder tienen un cambio, el programa interrumpirá la tarea que se este ejecutando y llamará a la función que le hayamos indicado, en este caso void (contar). El empleo de este método cuadriplica los PPR, es decir, que en un giro completo del eje del motor nuestra función contar contará hasta 1320 lo que dará mucha más precisión a las medidas de velocidad y posición.


Lectura y almacenamiento de las lecturas de las señales digitales C1 y C2 como un numero binario de 2 bits
Ej: Si C1=1 y C2=0, el numero almacenado sería lect = b10
Se combina la lectura (lect), y la que se obtuvo la vez anterior que se ejecutó la función:
Si la lectura anterior era lect_a = b11 y la actual lect = b10, generará el número ciclo = b1110
A continuación, actualizará el valor de la lectura anterior: lect_a = lect.
El número ciclo, se compara con los ocho posibles ciclos de un estado a otro que pueden tener las salidas del encoder, 4 corresponder a un sentido de giro, y 4 a otro.
Si la comparación anterior indica que el sentido es positivo, una variable (contador), incrementa su valor en 1, y se registra que el sentido es positivo (sentido =1), de lo contrario, si la comparación indica que el sentido es negativo, se resta 1 a contador y sentido =1.

### 4.2.6 Encoder con pulsador

La programación será similar a la que se ha realizado para el encoder que incorpora el motor, pero si que es necesario hacer distinciones.

Este encoder, al contrario que el anterior tiene posiciones de reposo, es decir que el movimiento que se puede hacer al girarlo no es continuo, sino que se debe hacer de una posición de reposo a otra (cada posición de reposo coincide con que ambas señales provenientes del encoder, DT y CLK, se encuentran en 1). Esto hace si se emplease el mismo método para contar el movimiento del encoder que en el anterior caso, contaríamos de 4 en 4.

Para solucionar este problema, se suele reducir el número de señales que producen interrupción a una, y en lugar de producirse con un cambio (interrupción de tipo change), se produce cuando la señal está en uno de sus dos posibles estados (en este caso sería LOW (0), ya que el esta 1 es su estado por defecto. Teóricamente con hacer estos cambios en el programa sería suficiente, cuando una de nuestras señales fuese 0, se procedería a leer la otra, y dependiendo de si fuese 0 ó 1, se sabría el sentido de giro. El problema viene al llevarlo a la práctica, y es que esas posiciones de "reposo mecánico" que tiene el encoder produce un rebote y eso puede producirnos lecturas incorrectas.

Tras diversas pruebas la solución a la que se ha llegado es emplear exactamente el mismo programa que para controlar el encoder del motor, pero en lugar de existir cuatro "ciclos" para cada sentido de giro, tan solo hay uno, el que se encuentra entre las posiciones de "reposo", de esta manera se evitan las lecturas incorrectas por rebotes.

En cuanto al botón, se ha implementado un PULL-UP en el pin al que está conectado y un delay de 10ms desde que se deja de pulsar el botón hasta que puede volver a ser pulsado, de ese modo también evitaremos rebotes al pulsar el botón.

### 4.2.7 Resistencias como sensor de tensión

La programación para la lectura de las mismas es sencilla, simplemente se leerá la entrada analógica que indica la caída de tensión en la resistencia de 100KΩ y se le aplica la siguiente función de transferencia:

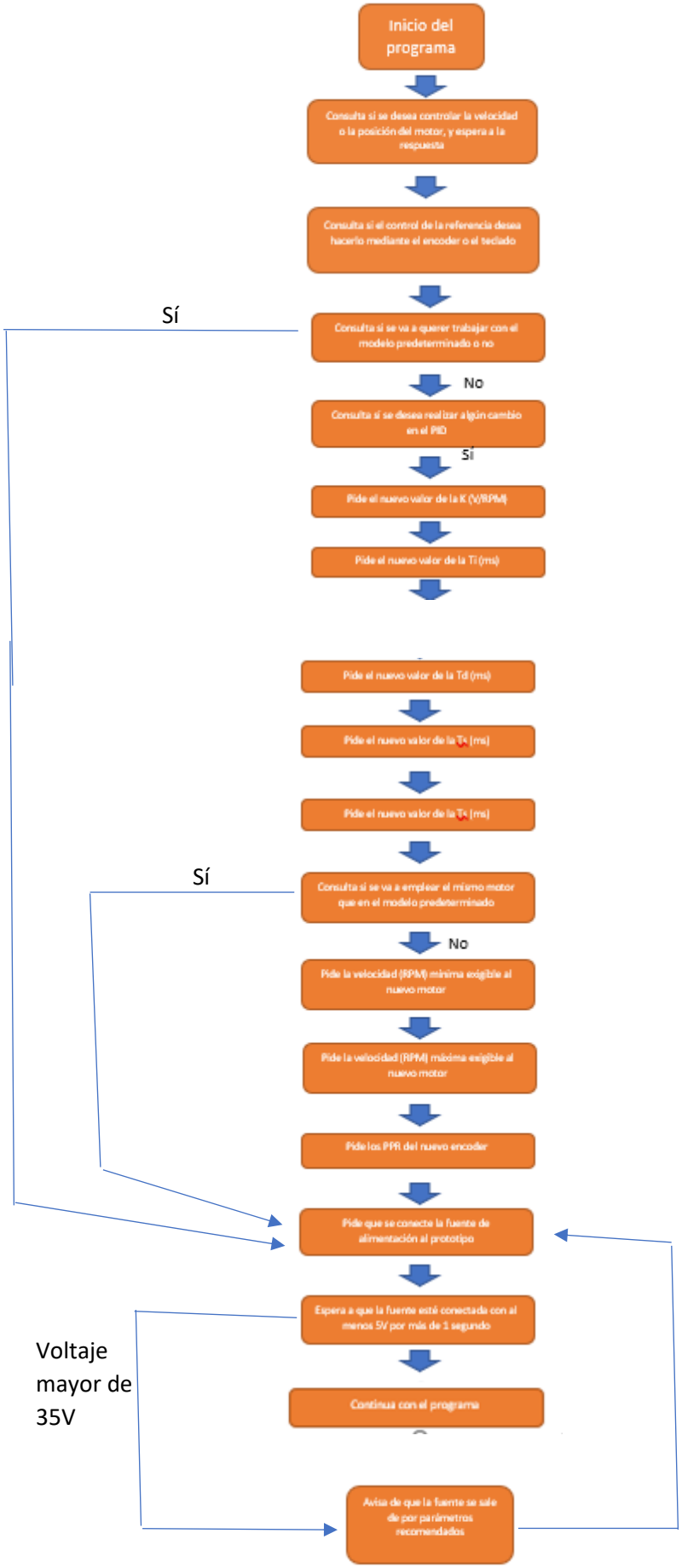
Siendo V_r la lectura de la tensión en la resistencia, y V_in la tensión de alimentación del driver:

V_in = (V_r/1023.0)*5.0*11.0

## 4.3 Programación funciones empleadas

### 4.3.1 Funciones comunes al control de posición y al de velocidad

- Inicialización parámetros controlador


Inicio del programa
Consulta si se desea controlar la velocidad o la posición del motor, y espera a la respuesta
Consulta si el control de la referencia desea hacerlo mediante el encoder o el teclado
Sí
Consulta si se va a querer trabajar con el modelo predeterminado o no
No
Consulta si se desea realizar algún cambio en el PID
Sí
Pide el nuevo valor de la K (V/RPM)
Pide el nuevo valor de la Ti (ms)
Pide el nuevo valor de la Td (ms)
Pide el nuevo valor de la Ts (ms)
Pide el nuevo valor de la Ts (ms)
Sí
Consulta si se va a emplear el mismo motor que en el modelo predeterminado
No
Pide la velocidad (RPM) mínima exigible al nuevo motor
Pide la velocidad (RPM) máxima exigible al nuevo motor
Pide los PPR del nuevo encoder
Pide que se conecte la fuente de alimentación al prototipo
Espera a que la fuente esté conectada con al menos 5V por más de 1 segundo
Voltaje mayor de 35V
Continua con el programa
Avisa de que la fuente se sale de por parámetros recomendados

- Control de la potencia,tensión e intensidad máxima

En caso de que la lectura proveniente del lector de intensidad supere los 4A (máxima intensidad que puede proporcionar el driver de forma segura), resetea el programa de forma automática para evitar el deterioro del controlador. Lo mismo ocurre si detecta que la fuente de alimentación es desconectada o excede los 35V, o si la potencia suministrada al motor excede los 50W.

Para calcular la potencia suministrada se sigue la siguiente fórmula:

P=I*V

La I, proviene de la lectura del sensor de intensidad, y la V, es la tensión que está suministrando el controlador al motor, que se calculará:

V=V_in*(u/255.0) donde V_in es la tensión de alimentación del controlador, y u la acción de control proveniente del PID (la cual varía entre 0 y 255).

- Cambios en la referencia

Existen dos formas de realizar cambios en la referencia:

1. Cambios mediante el teclado: Si así lo ha indicado el usuario, el valor introducido por pantalla será la nueva referencia, ya sea posición o velocidad del motor. Dicho cambio se podrá realizar en cualquier momento mientras se esté ejecutando el bucle principal.
2. Cambios mediante el encoder y pulsador: Si el usuario ha indicado que la referencia la quiere manejar desde el encoder, este incrementará o disminuirá el valor de la referencia (de 1 en 1) a medida que se vaya girando, en caso de ser pulsado, añadirá o disminuirá 50rpm a la velocidad (dependiendo del sentido del último giro del encoder), y pondrá la referencia a 0 en caso de que se trate de un control de posición.

- Actualización de pantalla

Por la pantalla LCD al usuario se le muestra la referencia que ha introducido y el estado actual del motor, estos valores se actualizan cada vez que el usuario hace un cambio en la referencia, o cada vez que pasa una cierta cantidad de tiempo desde la ultima vez que se actualizó la pantalla.

Se consigue aplicando la función lcd.clear() para limpiar la pantalla, y posteriormente lcd.print() para poder mostrar los datos en ella.

### 4.3.2 Funciones empleadas en el control de la posición

- Cálculo de la posición del motor

El cálculo de la posición actual del motor se realiza mediante una fórmula de transferencia que tiene en cuenta el conteo de la función cuenta(), y las PPR con las que cuenta nuestro encoder;

Posición(º) = 360*cuenta/(4.0*PPR)

- Función PID de posición

Esta función es la que implementa el PID para controlar la posición, cada vez que la unterrupción que se ha ligado a un temporizador mediante la librería <TimerOne.h> lo indica (la frecuencia a la que esto ocurre depende del tiempo de muestreo que se haya indicado), este PID calcula el error y a partir de él, calcula la acción de control (señal que debe llegar al driver), para que el movimiento sea suave y fluido la nueva referencia se le suministra al PID en forma de rampa, de esa manera el error no sufre cambios bruscos y se evitan movimientos violentos del motor.

### 4.3.3 Funciones empleadas en el control de la velocidad

- Cálculo de la velocidad

La llamada a esta función ocurre exactamente igual que la llamada al PID de posición, cada vez que la interrupción ligada al temporizador se activa, esta función se ejecuta.

Para el cálculo de la posición simplemente sigue la fórmula velocidad = espacio/tiempo, evidentemente esta fórmula no es exacta, pero el error que produce es mínimo, la mejor manera de disminuirlo es aumentar la velocidad, el tiempo de muestreo o los polos del encoder utilizado. La Ts empleada en el modelo comporta un buen compromiso entre posición y tiempo de respuesta del controlador (incrementar la Ts demasiado en un sistema tan rápido podría producir problemas de tiempo de respuesta y estabilidad).

V (RPM) = 60*(cuenta)/(Tm(ms)*PPR*4.0/1000.0)

- Función PID de velocidad

Esta función se ejecuta a continuación de ejecutarse la función de calculo de la velocidad, simplemente calcula el error entre la referencia y la velocidad calculada, tras ello aplica el PID y obtiene la acción de control que traslada al driver.

### 4.3.4 Función setup

Inicializa los pines necesarios para la toma de datos del teclado y comunicación con la pantalla LCD, a continuación ejecuta la función de inicialización de parámetros del controlador e inicializa el resto de pines y rutinas de interrupción necesarias para el control que se desea realizar (ya sea de posición o de velocidad).

### 4.3.5 Función void

En ella existen cuatro bucles de control (control de velocidad mediante encoder, control de velocidad mediante teclado, control de posición mediante encoder y control de posición mediante teclado). Entrará en uno de los cuatro bucles según loq eu haya indicado en usuario en la inicialización, una vez en él de forma continua realizará las siguientes tareas:

- Recogida de datos del usuario (ya sea por el teclado o por el encoder).
- Control de tensión, intensidad y potencia.
- Refresco de la pantalla con actualización de datos, mediante un contador se evita que el refresco de pantalla ocurra en menos de un periodo de 500 ms, ya que esto complica su lectura.
- Una vez se activen las rutinas de interrupción pertinentes, tomará una muestra del estado del motor (velocidad o posición), e implementará el PID.

# 5 Resultados

Este apartado se abordará la sintonización de los dos PID que se han implementado.

Si bien es cierto que la herramienta creada en este proyecto permite al usuario elegir los parámetros a implementar, permitiéndole controlar cualquier motor de continua que desee (siempre que se encuentre en el rango de motores que puede soportar el driver), el prototipo contendrá una configuración predeterminada, que controlará el motor empleado en este trabajo. Para realizar tal sintonización se ha empleado el método empírico Ziegler-Nichols, del cual ya se ha hablado en los subapartados 2.1 y 2.2.

El proceso seguido para la sintonización de cada PID ha sido la siguiente:

## 5.1 PID velocidad

Como se ha indicado en el subapartado 2.1, se empleará el método 1 de Ziegler-Nichols, el cual requiere obtener una gráfica que muestre la respuesta del motor en bucle abierto ante una entrada tipo escalón. El escalón ha sido de 9.41 V.

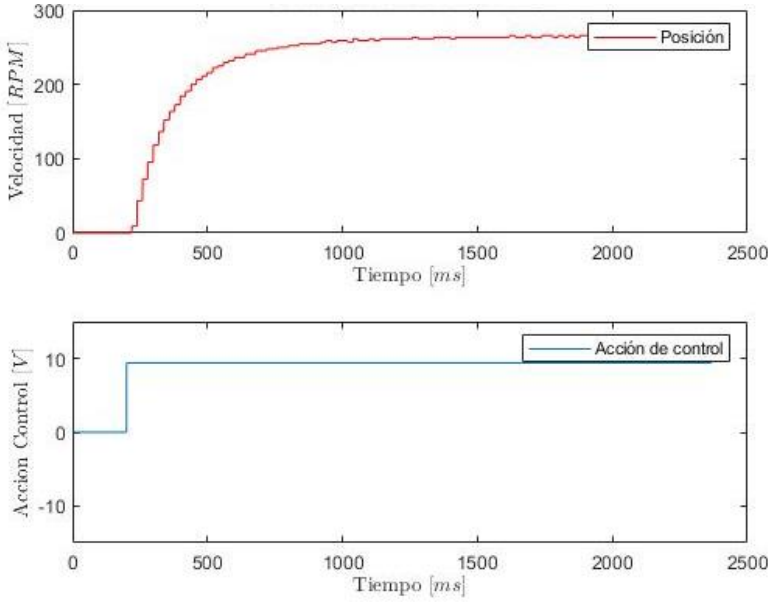



*Ilustración 32. Velocidad bucle abierto ante entrada de tipo escalón de 9.4V*

A partir de esta gráfica podemos obtener los siguientes parámetros:

K = 260/9.41 = 27.63 rpm/V

L = 25 ms (puesto la respuesta del sistema es prácticamente instantánea, se toma como retraso el 50% del tiempo de muestreo)

T = 150 ms

Nota: El tiempo de muestro empleado para este PID será de 50ms, aumentar este valor pese a disminuir el error en la medida de las velocidades podría dar problemas de estabilidad, al acercarse demasiado el tiempo de muestreo al tiempo que tarda el sistema en alcanzar el régimen permanente. Disminuir el tiempo de muestreo comprometería demasiado la fiabilidad de la medida de la velocidad.

Con estos valores si se les aplica la siguiente tabla se obtiene que:

| | Parámetro | Método Escalón (B.Abierto) | Respuesta Sostenida (B.Cerrado) |
|---|---|---|---|
| P | $Kp$ | $\frac{T}{K*L}$ | 0,5*Kc |
| PI | Kpi | $0{,}9*\frac{T}{K*L}$ | 0,4*Kc |
| | Ti | 3*L | 0,8*Tc |
| PID | Kpid | $1{,}2*\frac{T}{K*L}$ | 0,6*Kc |
| | Ti | 2*L | 0,5*Tc |
| | Td | $\frac{L}{2}$ | 0,125*Tc |

*Ilustración 33. Tabla parametrización PID. Fuente: Elaborada a partir de Fernández (2021)*

Kpid = 0.261 V/rpm

Ti = 50 ms

Td = 12.5 ms

Implementado un PID de la siguiente forma:

$u(k)=p_0e(k) + p_1e(k-1) + p_2e(k-2) + u(k-1)$

Donde:

| | p0 | p1 | p2 |
|---|---|---|---|
| P | $Kp$ | 0 | 0 |
| PD | $Kpd * \frac{Ts+Td}{Ts}$ | $-Kpd * \frac{Td}{Ts}$ | 0 |
| PI | $Kpi$ | $Kpi * \frac{Ts-Ti}{Ti}$ | 0 |
| PID | $Kpid * \frac{Ts+Td}{Ts}$ | $Kpid * (-1 + \frac{Ts}{Ti} - 2 * \frac{Td}{Ts})$ | $Kpid * \frac{Td}{Ts}$ |

*Ilustración 34. Tabla digitalización PID. Fuente: Elaborada a partir de Fernández (2021)*

Se obtiene la siguiente respuesta del motor:

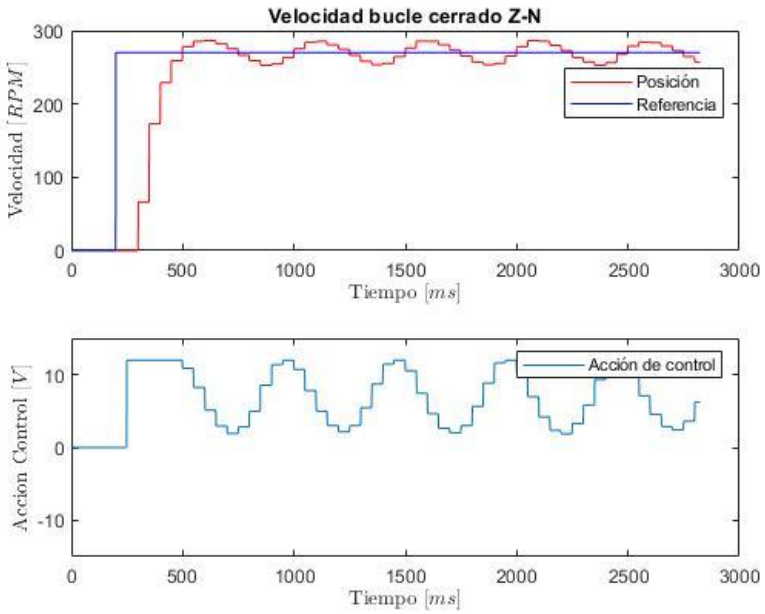



*Ilustración 35. Velocidad bucle cerrado con PID obtenido por Ziegler-Nichols*

El problema salta a la vista, el sistema no estabiliza. Como hemos visto en el apartado 2 existen distintos motivos por los que esto puede ocurrir, y por tanto diferentes soluciones.

En este caso en concreto se han optado por tres:

- Eliminar el término derivativo (Td=0): Este sistema se comporta de forma rápida de forma natural (sin aplicar ningún controlador), por tanto carece de sentido el aplicar este término.
- Disminución del valor Kpid (Kpid=0.085 V/rpm): Este cambio viene dado por la misma razón que el anterior, disminuir la Kpid, simplemente hará que el sistema sea más lento, pero le otorgará mayor estabilidad.
- Disminución del efecto del término integral (Ti=200 ms): El incrementar el valor de Ti hará disminuir que peso que tiene el error acumulado sobre el cálculo de la ación de control, ralentizará el sistema, pero disminuirá la sobreoscilación.

El valor exacto de la Kpid y de la Ti a la que el controlador mostraba unos resultados satisfactorios se han empleado además las bases teóricas antes explicadas como guía, el método prueba y error.

Tras aplicar estos cambios obtenemos una respuesta suave y sin sobreoscilaciones sin ser lenta:

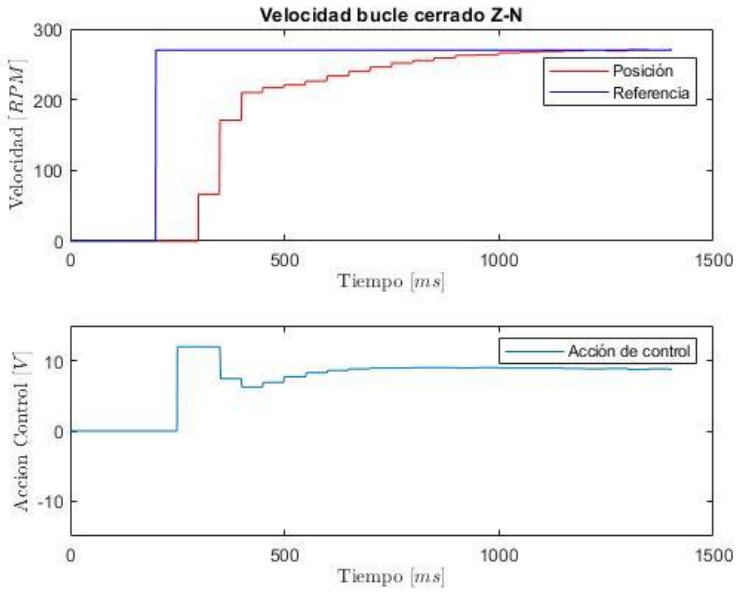



*Ilustración 36. Velocidad bucle cerrado con PID afinado*

## 5.2 PID posición

Como se ha indicado en el subapartado 2.2, se empleará el método 2 de Ziegler-Nichols, el cual requiere obtener una gráfica que muestre la respuesta del motor en bucle cerrado ante un cambio en la referencia, con una ganancia que haga que el sistema tenga una respuesta oscilatoria de amplitud constante. El cambio en la referencia ha sido de 50 cuentas del encoder (13.64 º), el sistema alcanza este estado cuando la Ku ha llegado a los 0.47 V/rpm.

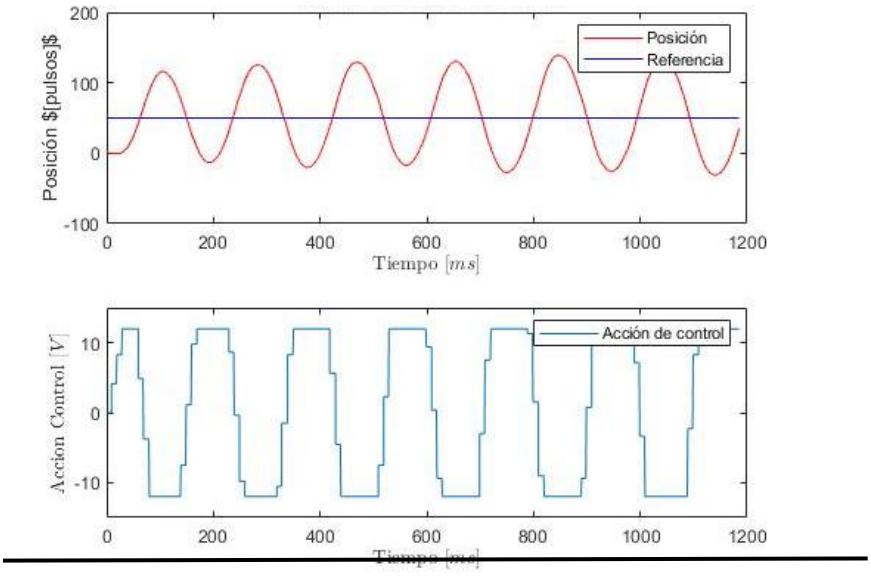



*Ilustración 37. Posición bucle cerrado con respuesta oscilatoria de amplitud constante*

De esta gráfica se han obtenido los siguientes parámetros:

Ku = 0.47 V/rpm

Pu = 177 ms

Nota: El tiempo de muestreo empleado para este PID será de 10ms, puesto que una disminución del tiempo de muestreo no aumenta el error en las medidas de posición, este valor se hará lo más pequeño posible, siempre y cuando no sature el microprocesador e impida que se ejecuten las demás funciones del programa.

Con estos valores si se les aplica la siguiente tabla se obtiene que:

| | Parámetro | Método Escalón (B.Abierto) | Respuesta Sostenida (B.Cerrado) |
|---|---|---|---|
| **P** | *Kp* | $\frac{T}{K * L}$ | 0,5*Kc |
| **PI** | Kpi | $0{,}9*\frac{T}{K*L}$ | 0,4*Kc |
| | Ti | 3*L | 0,8*Tc |
| **PID** | Kpid | $1{,}2*\frac{T}{K*L}$ | 0,6*Kc |
| | Ti | 2*L | 0,5*Tc |
| | Td | $\frac{L}{2}$ | 0,125*Tc |

*Ilustración 38. Tabla de parametrización PID. Fuente: Elaborada a partir de Fernández (2021)*

Kpid = 0.282 V/cuenta

Ti = 88.5 ms

Td = 22.125 ms

Implementado un PID de la siguiente forma:

$u(k)=p_0e(k) + p_1e(k-1) + p_2e(k-2) + u(k-1)$

Donde:

| | p0 | p1 | p2 |
|---|---|---|---|
| P | $Kp$ | 0 | 0 |
| PD | $Kpd * \frac{Ts+Td}{Ts}$ | $-Kpd * \frac{Td}{Ts}$ | 0 |
| PI | $Kpi$ | $Kpi * \frac{Ts-Ti}{Ti}$ | 0 |
| PID | $Kpid * \frac{Ts+Td}{Ts}$ | $Kpid * (-1 + \frac{Ts}{Ti} - 2 * \frac{Td}{Ts})$ | $Kpid * \frac{Td}{Ts}$ |

*Ilustración 39. Tabla digitalización PID. Fuente: Elaborada a partir de Fernández (2021)*

Se obtiene el siguiente comportamiento:

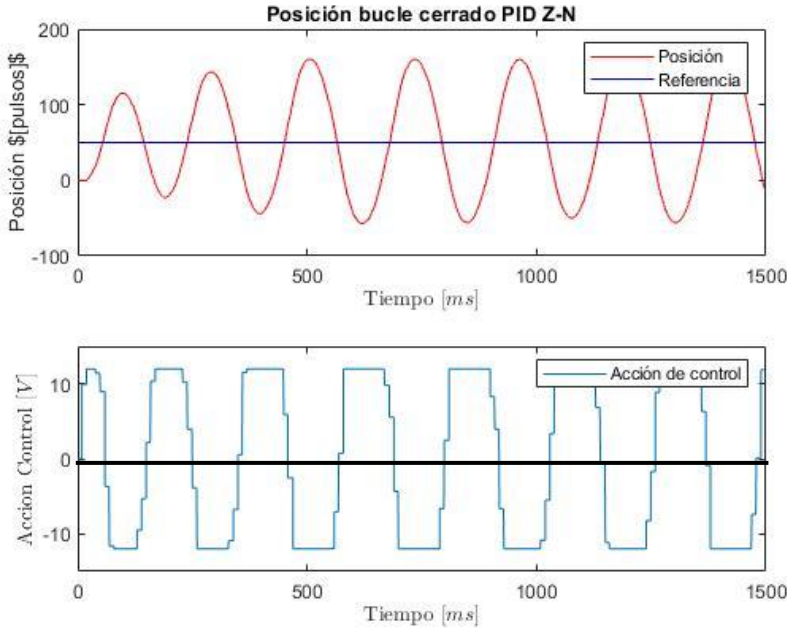



*Ilustración 40. Posición bucle cerrado con PID obtenido por Ziegler-Nichols*

Esta respuesta no cumple ninguno de los requisitos que debe tener un controlador, no es capaz de estabilizar, y encima su comportamiento genera cambios bruscos de velocidad y sentido de giro en el motor lo que puede causar un desgaste excesivo del ismo y de los elementos que lo rodean.

Para poder empezar a buscar soluciones debemos primero aclarar ciertas peculiaridades de este PID, para empezar, está tratando de controlar la posición de una forma extremadamente precisa (el error en la medida del giro del motor será menor a 0.14º), la naturaleza de los motores de corriente continua hace muy difícil conseguir un movimiento fluido a bajas revoluciones (este tipo de motores tienen lo conocido como zona muerta, esta es una rango de tensiones a las que el motor pese a estar siendo alimentado no se mueve).

Sabiendo esto se pueden emplear las siguientes herramientas para conseguir un controlador efectivo:

- Limitar el error: Esto se conseguirá introduciendo los cambios de referencia mediante una rampa, y no de golpe, lo cual hace reaccionar al controlador de forma brusca.
- Disminuir el valor de la Td (Td = 11.375ms), en este caso, que el motor sea un sistema rápido por naturaleza es una desventaja a la hora de controlar su posición, esto se debe a que durante el transcurso de un periodo de muestreo, la posición habrá cambiado considerablemente.
- Disminuir el efecto de la acción integral (Ti=1365ms), aunque sea necesario el contar con la acción integral, ya que permite asegurar el que error final sea 0, darle demasiado peso provocará sobreoscilaciones en el sistema, las cuales serán lentas de corregir, además, el aumento de este valor ralentizará más al sistema.
- Aumento de la Kpid (Kpid = 0.705 V/cuenta), aunque parezca extraño el realizar este cambio (el aumento de Kpid inestabiliza al sistema), es necesario que el motor sobrepase la zona muerta y sea capaz de moverse cuando el error sea muy pequeño (error de una cuenta son 0.28º).

Al igual que al sintonizar el PID para el control de la velocidad, los calores exactos de Kpid, Ti y Td se han obtenido de forma experimental mediante el método de prueba y error.

El resultado obtenido ha sido el siguiente:

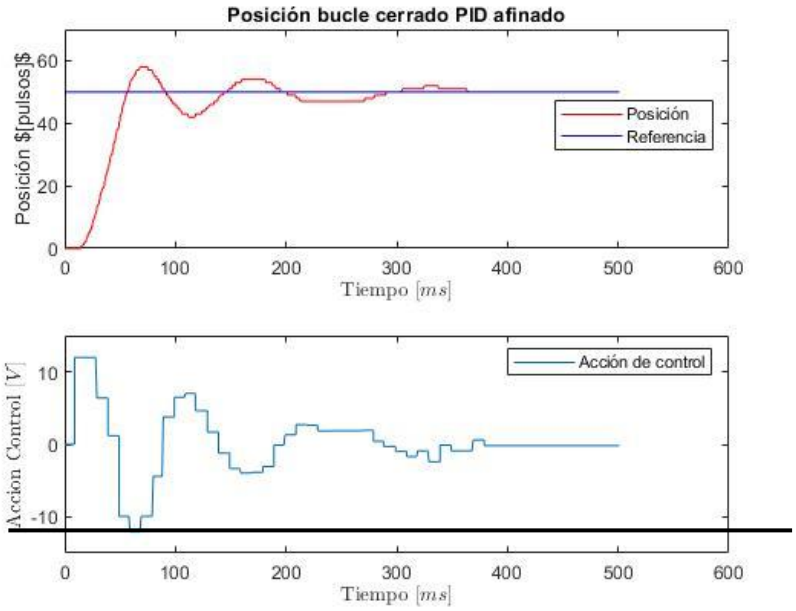



*Ilustración 41. Posición bucle cerrado PID afinado, referencia 50 cuentas (aproximadamente 15º) cambio de referencia tipo rampa*

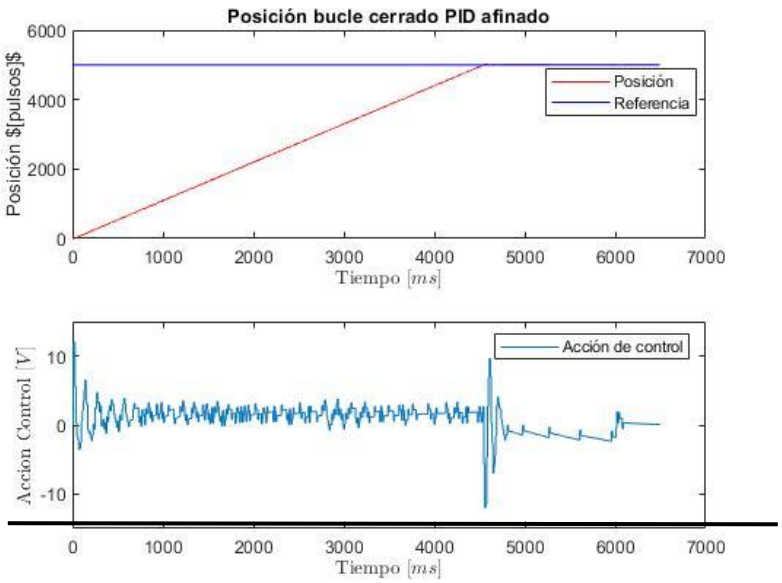



*Ilustración 42. Posición bucle cerrado PID afinado, referencia 5000 cuentas (aproximadamente 4 vueltas completas)cambio de referencia tipo rampa*

Como se puede observar, tanto con cambios de referencia grandes como pequeños, se consigue una respuesta suave y con apenas sobreoscilaciones. La posición en el caso de la referencia a 5000 cuentas tiene forma de rampa ya que es así como el programa introduce el error en el PID, su efecto se puede observar mejor ante cambios de referencia grandes. La ventaja de emplear este método es que no importa lo grande que sea el cambio de referencia, el error siempre dependerá de la pendiente de la rampa que hayamos programado para introducir el error en el PID.

Cabe a destacar que a la hora de digitalizar ambos PID se les ha aplicado un Antiwindup, que consiste en limitar el valor máximo que puede alcanzar la acción de control a el valor al que satura el controlador (en este caso 12V). Esto evitará casos de retardo en la respuesta del sistema en caso de que la acción de control calculada por el PID exceda las capacidades de nuestros actuadores.

# 6 Manual para el usuario

El producto que recibirá el usuario será la caja que contendrá tanto el hardware necesario, como el programa ya cargado en la placa Arduino. Pala la utilización del producto deberá hacer los siguiente:

1. Conectar el motor con el que desee trabajar a la caja. Nótese que deberá conectar tanto la alimentación del motor como los pines del encoder, en la propia caja indica donde debe conectarse cada cable.
2. Conectar la placa a un puerto USB a través del cable USB, puesto que el programa ya está cargado no es necesario que se conecte a una computadora, basta con que la alimentación sea de 5V de corriente continua.
3. Una vez la placa esté conectada el sistema se iniciará automáticamente, y le solicitará que indique qué aspecto del motor desea controlar (posición o velocidad), y si desea emplear el modelo ya existente. La pantalla le irá indicando que debe hacer en cada momento. Al introducir los valores numéricos el botón '*' actuará como punto decimal, y el botón '#', indicará que ha terminado de escribir el valor, la pantalla solo mostrará dos decimales del valor escrito, y si en algún momento se equivoca en la introducción de alguno de ellos no tiene más que accionar el pulsador Reset, y podrá volver a confirgurar el controlador.
4. Antes de comenzar con el control del motor se le indicará que conecte la fuente de alimentación a la caja, esto se debe a que de hacerlo una vez haya introducido una referencia, conectar la fuente de tensión puede provocar un arranque brusco del motor. (El programa también se asegurará de que la fuente de alimentación proporcione una tensión adecuada, de no ser así, se lo indicará al usuario).
5. Una vez iniciado el control, podrá introducir la referencia como haya indicado con anterioridad (mediante el teclado o mediante el encoder).
    - Control de posición mediante el encoder: El programa está diseñado para que el eje del motor siga el giro del encoder, si se acciona el botón del mismo, el motor volverá a la posición inicial.
    - Control de la velocidad mediante el encoder: Si gira el encoder en sentido horario, la velocidad del motor aumentará, si lo gira en el otro sentido disminuirá (nótese que el motor puede girar en ambos sentidos), si acciona el botón del encoder, la velocidad del motor aumentará o disminuirá en 50 rpm, dependiendo del último sentido en el que giró el encoder.
    - Control de la velocidad o posición mediante el teclado: Introduzca la referencia por el teclado, (rpm en caso de velocidad, y º en el caso de la posición), una vez escrito el valor, presione '*' si este es positivo, o '#' si es negativo.
      Notas: La velocidad de giro del motor (valor absoluto), está limitada por lo que el usuario haya indicado al programar el controlador, el propio programa impedirá que se soliciten dichas velocidades.
    - En todo momento se le indicará al usuario tanto la velocidad o posición del motor, como la referencia que ha introducido a través de la pantalla.
6. Si en algún momento desea parar el programa solamente debe de accionar el pulsador Reset. El programa parará automáticamente por motivos de seguridad si se

desconecta la fuente de alimentación o se conecta una que exceda los 35V, si la intensidad que le llega al motor los 4A, o se consumen más de 50W.
Modelo predeterminado:

PID vel:

- Tm = 50 ms
- K = 0.085 V/rpm
- Ti = 200 ms
- Td = 0

PID pos:

- Tm = 10ms
- K = 0.705 V/cuenta
- Ti = 1365 ms
- Td = 11.375 ms

Motor:

- V = 12V
- PPR = 330
- PRM max = 285 rpm
- RPM min = 10 rpm

# 7 Conclusiones

Como punto final sería conveniente hacer una valoración del trabajo realizado y comentar las conclusiones que se han podido obtener del mismo.

Todos los objeticos que se perseguían desde un comienzo se han podido cumplir satisfactoriamente:

- El sistema al que se ha llegado controla de forma precisa, suave y rápida la velocidad y posición del motor.
- La interfaz creada para que la máquina y el usuario puedan comunicarse es clara y sencilla de utilizar, permitiendo programar el controlador como se desee o implementando el modelo predeterminado.
- El valor de la referencia puede ser introducido por dos métodos distintos (pantalla y encoder), lo cual puede ser útil dependiendo del tipo de aplicación que se le esté dando al motor cuyo estado está siendo constantemente monitoreado y postrado por la pantalla LCD. Todo ello facilitando el uso del controlador al usuario.
- Cada componente ha sido aprovechado al máximo de sus apacidades (ej. modificaciones hechas al Driver para permitir duplicar la intensidad máxima que puede tolerar, programación de código que cuadriplicar la precisión del encoder integrado en el motor…) todo ello ha llevado a un sistema no solo eficaz sino eficiente, abaratando los costes de los componentes.
- Se han introducido funcionalidades que velan por la integridad y longevidad del controlador (ej. Control de la tensión de entrada, intensidad que llega al motor y potencia absorbida).
- Si bien es cierto que el código seguramente podría ser optimizado, disminuyendo el uso del microprocesador que emplea el programa, o espacio en memoria que requiere el programa es robusto y es capaz de cumplir con todas las funcionalidades requeridas sin saturar el microprocesador.

El proyecto desde un inicio ha sido interesante y divertido de realizar permitiéndome libertad a la hora de pensar en soluciones para los problemas que iban surgiendo y desarrollo de nuevas funcionalidades que considerase útiles. A lo largo de la realización del trabajo he podido aumentar mis conocimientos en muchas ramas de la ingeniería que aunque en un principio de la carrera no habían llamado mucho mi atención (como es el caso de la programación) han resultado ser fascinantes como es el caso de Arduino, que ha realizado una labor increíble acercando el mundo de la electrónica y la programación a la población general lo cual he podido comprobar investigando la multitud de proyectos y guías que existen en la red.

# 8 Esquema eléctrico

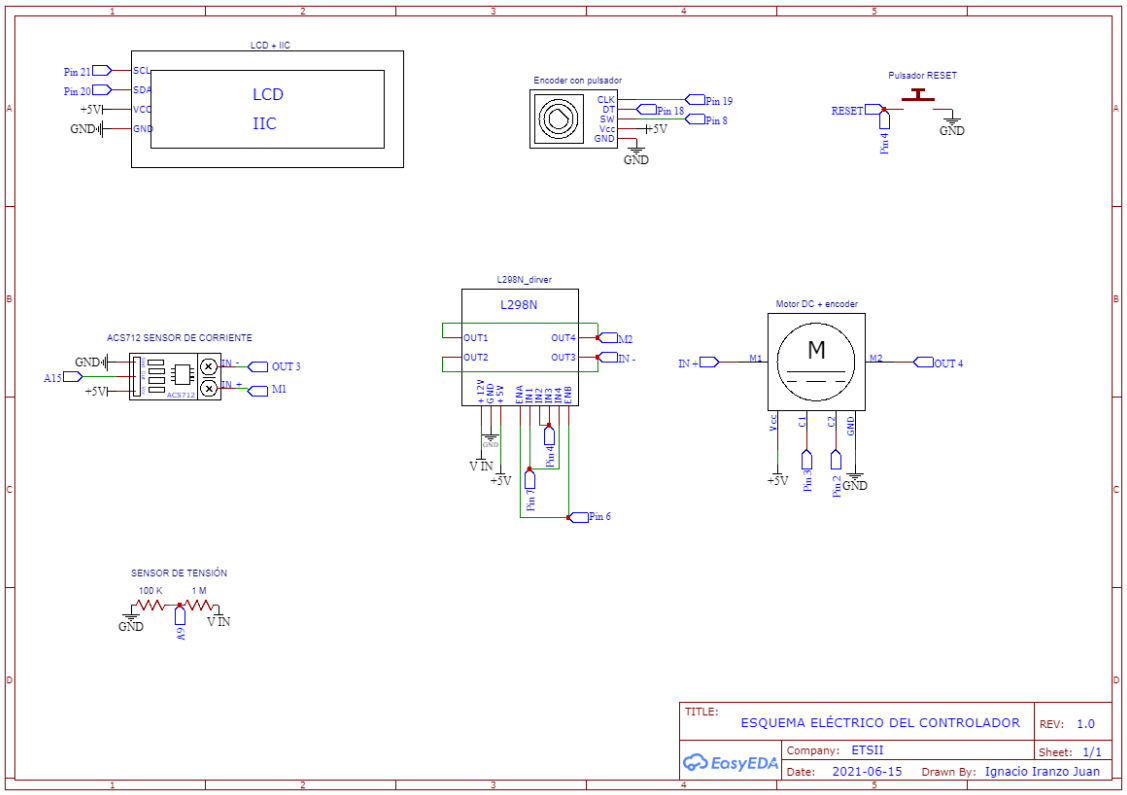



*Ilustración 43. Esquema eléctrico*

# 9 Presupuesto

## 9.1 Introducción

En este apartado se tratará el presupuesto para la realización del proyecto, el cual se desglosará según el presupuesto en materiales, herramientas y mano de obra.

## 9.2 Presupuesto de los materiales

Se desglosará por el costo de cada componente y material empleado en el proyecto

| Unidad | Descripción | Costo unitario (€) | Cantidad | Costo total (€) |
|---|---|---|---|---|
| u | **Arduino MEGA 2560** | 13,99 | 1 | 13,99 |
| u | **Motor DC** | 17,50 | 1 | 17,50 |
| u | **Driver L295N** | 7,99 | 1 | 7,99 |
| u | **Pantalla LCD** | 6,99 | 1 | 6,99 |
| u | **Adaptador IIC** | 0,35 | 1 | 0,35 |
| u | **Encoder con pulsador** | 2,40 | 1 | 2,40 |
| u | **Keypad 4x4** | 9,99 | 1 | 9,99 |
| u | **Pulsador** | 3,85 | 1 | 3,85 |
| u | **Sensor de corriente ACS712** | 4,80 | 1 | 4,80 |
| u | **Resistencia 100KΩ** | 0,02 | 1 | 0,02 |
| u | **Resistencia 1MΩ** | 0,02 | 1 | 0,02 |
| u | **Cable USB** | 0 | 1 | 0 |
| u | **Cables Dupont (Pack 120)** | 6,49 | 1 | 6,49 |
| m | **Cable 1mm** | 1,00 | 2 | 2,00 |
| u | **Adaptador de corriente** | 11,98 | 1 | 11,98 |
| u | **Conector tipo Jack (pareja)** | 2,65 | 2 | 5,30 |
| u | **Caja de conexiones** | 11,10 | 1 | 11,10 |
| u | **Placa perforada estañada** | 13,99 | 1 | 13,99 |
| u | **Tornillos 3mm** | 0,05 | 8 | 0,40 |
| u | **Pegamento termofusible** | 0,18 | 1 | 0,18 |
| m | **Tubo termoretráctil** | 4,49 | 0,20 | 0,90 |
| u | **Cinta aislante** | 0,95 | 1 | 0,95 |
| m | **Estaño** | 0,11 | 2 | 0,22 |
| | **Total** | | | 131,41 |

*Ilustración 44. Presupuesto parcial de los Materiales*

## 9.3 Presupuesto de las herramientas

En este apartado se tendrá en cuenta el precio de las herramientas y maquinaria que han sido empleadas durante el desarrollo del proyecto, debe aclararse que estas herramientas y maquinaria van a poder y serán utilizadas en otros proyectos, por tanto, podrán ser amortizadas.

El presupuesto en herramientas y maquinaria ha sido:

| Unidad | Descripción | Costo unitario (€) | Cantidad | Costo total (€) |
|---|---|---|---|---|
| u | **Destornillador plano pequeño** | 4,80 | 1 | 4,80 |
| u | **Destornillador de estrella pequeño** | 4,80 | 1 | 4,80 |
| u | **Cuter** | 11,29 | 1 | 11,99 |
| u | **Pelacables** | 12,49 | 1 | 12,49 |
| u | **Miniamoladora** | 37,99 | 1 | 37,99 |
| u | **Pistola de pegamento termofusible** | 7,99 | 1 | 7,99 |
| u | **Soldador de estaño** | 15,99 | 1 | 15,99 |
| u | **Total** | | | 83,56 |

*Ilustración 45. Presupuesto parcial de las Herramientas*

## 9.4 Presupuesto de la mano de obra

En este apartado se tendrán en cuenta las horas destinadas por los distintos profesionales a la realización del proyecto.

Para la realización de este proyecto ha sido necesaria la participación de dos profesionales, un ingeniero industrial (trabajando como tutor de la UPV), y un ingeniero técnico en prácticas (estudiante de la UPV realizando el Trabajo de Fin de Grado). Para calcular el costo por hora trabajada de cada uno de ellos se ha tenido en cuenta que:

- El ingeniero industrial tutor en la UPV tiene un salario anual de unos 60.000 €, y un ingeniero técnico en prácticas en prácticas unos 25.000 €.
- Un año cuenta con 250 días laborables, con jornadas de 8h al día.

Con los datos anteriores se puede calcular el costo por hora de cada uno de ellos:

Ingeniero técnico en prácticas (I.T.): 25.000 (€/año) / (250 (días/año) x 8 (h/día)) = 12,5 €/h

Ingeniero industrial (I.I.): 60.000 (€/año) / (250 (días/año) x 8 (h/día)) = 30 €/h

El tiempo dedicado a este proyecto se ha repartido de la siguiente manera:

| Concepto | Horas (h) | Participantes |
|---|---|---|
| **Reuniones** | 15 | (I.T.) (I.I.) |
| **Compras** | 5 | (I.T.) |
| **Montaje** | 45 | (I.T.) |
| **Programación** | 135 | (I.T.) |
| **Elaboración de documentos** | 100 | (I.T.) |
| **Total** | (I. T.): 300 (I.I.): 15 | |

*Ilustración 46. Distribución de las horas de trabajo*

Con los datos calculados con anterioridad, podemos obtener el presupuesto de la mano de obra:

| Unidad | Descripción | Costo unitario (€) | Cantidad | Costo total (€) |
|---|---|---|---|---|
| **h** | Ingeniero técnico en prácticas | 12,50 | 300 | 3750 |
| **h** | 15 | 30 | 15 | 450 |
| | Total | | | 4200 |

*Ilustración 47. Presupuesto parcial de la Mano de Obra*

## 9.5 Presupuesto total

Mediante la suma de los presupuestos parciales, más el cálculo de los Gastos Generales (13%), Beneficio Industrial (6%) e IVA (21%) podremos obtener el Presupuesto Base de Licitación o lo que es lo mismo, el coste total.

| | |
|---|---|
| **Presupuesto Parcial de Materiales** | 131,41 |
| **Presupuesto Parcial de Maquinaria** | 83,56 |
| **Presupuesto Parcial de Mano de obra** | 4200 |
| **Presupuesto de Ejecución Material** | 4414,97 |
| **Gastos Generales (13 %)** | 573,95 |
| **Beneficio Industrial (6 %)** | 264,90 |
| **Presupuesto Ejecución por Contrata** | 5255,82 |
| **IVA (21%)** | 1103,72 |
| **Presupuesto Base de Licitación** | 6359,54 |

*Ilustración 48. Presupuesto Base de Licitación*

El trabajo tendrá un coste de 6.359,54 €.

(SEIS MIL TRESCIENTOS CINCUENTA Y NUEVE EUROS CON CINCUENTA Y CUATRO CÉNTIMOS)

# 10 Bibliografía


12v14v products. (2021). *12v24vproducts.org*. Obtenido de https://www.12v24vproducts.org/es/conector-jack-12v

Arduino.cc. (2021). *store.arduino.cc*. Obtenido de https://store.arduino.cc/arduino-mega-2560-rev3?queryID=undefined

Bauhaus. (2021). *bauhaus.es*. Obtenido de https://www.bauhaus.es/cajas-de-conexiones/famatel-caja-de-superficie-para-estancias-con-humedad/p/23944488

Brico Geek. (2021). *tienda.bricogeek.com/*. Obtenido de https://tienda.bricogeek.com/sensores/1147-sensor-de-corriente-acs712-5a.html?gclid=CjwKCAjwrPCGBhALEiwAUl9X0677aAWXCuykRqy7VLEcsQowuFtAEDs3mQvPenMOvyJcmsvJrNXMuBoC0PkQAvD_BwE

cespedes. (s.f.). *cespedes.es*. Obtenido de https://www.cespedes.es/spa/item/1396658.html

Cetronic. (s.f.). *cetronic.es*. Obtenido de https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=303060020&cPath=1407&gclid=CjwKCAjwieuGBhAsEiwA1Ly_nSb5HUFYQXllFOa2L6F9RFO0zoDFpGTHEDlZUmgzohtg5JVjsG1KNRoC6O4QAvD_BwE

components101. (2021). *https://components101.com/*. Obtenido de https://components101.com/modules/l293n-motor-driver-module

elfa. (2021). *elfa.se*. Obtenido de https://www.elfa.se/en/4x4-matrix-keypad-adafruit-3844/p/30139178

Fernández, Á. V. (2021). Seminario: Introducción al Control de Robots. Universidad Politécnica de Valencia.

Flotante, P. (s.f.). *puntoflotante.net*. Obtenido de https://www.puntoflotante.net/FUNCIONAMIENTO-ENCODER-CUADRATURA-EFECTO-HALL.htm#:~:text=Los%20conocidos%20como%20'encoders%20cuadratura,del%20c%C3%ADrculo%20de%20360%C2%B0.

Giraldo, S. A. (2019). *controlautomaticoeducacion*. Obtenido de https://controlautomaticoeducacion.com/control-realimentado/ziegler-nichols-sintonia-de-control-pid/

iberobotics. (2021). *iberobotics.com*. Obtenido de https://www.iberobotics.com/producto/modulo-codificador-rotatorio-pulsador/

inven. (s.f.). *inven.es*. Obtenido de https://inven.es/product/pantalla-lcd-16x2-adaptador-i2c/

NPH22. (2021). *instructables.com*. Obtenido de https://www.instructables.com/Rotary-Encoder-Understand-and-Use-It-Arduinoother-/

Pardo, C. (2021). *picuino*. Obtenido de https://www.picuino.com/es/arduprog/control-pid.html

sfond294. (2021). *sfond294.blogspot.com/*. Obtenido de https://sfond294.blogspot.com/2021/01/view-20-conector-dc-hembra-hembra.html

STMicroelectronics. (2000). DUAL FULL-BRIDGE DRIVER.

taloselectronics. (2021). *https://www.taloselectronics.com/*. Obtenido de https://www.taloselectronics.com/products/cable-usb-tipo-a-tipo-b-2-0-para-arduino-uno-y-mega

```
#include <Key.h>
#include <Keypad.h>
#include <TimerOne.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
int j=0;
int z=0;
int y=0;
int h=0;
int decimal=0;
float pot=0.00;
#define encoderPin1  19 //Encoder Output 'CLK'.
#define encoderPin2  18 //Encoder Otput 'DT'.
#define encoderPinSW  8 //Encoder Out put 'SW'.
volatile float Ti;
volatile long Tm;
int tec = 0;
volatile float Td;
volatile long T_int=0;
volatile float V=12.0;
volatile int RPM_MIN=10;
volatile int RPM_MAX=280;
volatile long speedo=0;
volatile int BPR=330;
volatile float speedo_f=0;
volatile float speedo_ff=0;
volatile float K;
long grados;
const byte FILAS = 4;
const byte COLUMNAS = 4;
char keys [FILAS][COLUMNAS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
```

```
  {'*','0','#','D'},
};
volatile int V_r;
volatile float V_in;
byte pinesFilas[FILAS] = {39,41,43,45};
byte pinesColumnas[COLUMNAS] = {47,49,51,53};
char TECLA;
Keypad teclado = Keypad(makeKeymap(keys), pinesFilas,
pinesColumnas, FILAS, COLUMNAS);
volatile int lastEncoded = 0b11; // Posición anterior del
encoder.(manual)
volatile long encoderValue = 0; // Posición actual del
encoder.(manual)
volatile long triggers = 0; // Las veces que el encoder
del motor ha sido activado.
volatile short sentido = 0; // Valdrá 1 si está avanzando
el encoder y -1 si retrocede
volatile bool b = 1; // Nos permitirá saber si el valor
de la cuenta ha cambiado, y así imprimirlo
volatile float e = 0.0;
volatile float e_1 = 0.0;
volatile float e_2 = 0.0;
volatile int u = 0;
volatile float u_1 = 0.0;
volatile float u_f = 0.0;
volatile float u_ff = 0.0;
volatile float rpm_f = 0.0;
volatile float rpm_f_d = 0.0;
volatile long Tr =0;
volatile bool p =1;
volatile bool n =1;
#define intPin  A15 //Entrada intensidad
float intensidad=0.0;
int sensorInt;
#define IN3  7
```

```
#define IN4  6
#define ENB  5
#define C1  3
#define C2  2
int rpm = 0;
bool quieto = 0; // En un futuro la usare
volatile long c = 0;
uint32_t oldtime = 0;
uint32_t T = 0;
uint32_t Tl = 0;

int l=1;
volatile long d=0;
long v=0;
volatile int lastEncoder = 0;
volatile long encodedValue = 0;
volatile long encoderV = 0;
int i=0;
int rpm_a = 0;


void setup() {
  pinMode(A9,INPUT);
 lcd.init();
 lcd.backlight();
 lcd.clear();
 lcd.print("A: Velocidad");
 lcd.setCursor (0,1);
 lcd.print("B: Posicion");
 while (j!=1) {
    TECLA=teclado.getKey();
  if(TECLA!=NO_KEY){
  if (TECLA=='A') {
    lcd.clear();
    lcd.print("Velocidad");
```

```
   delay(1000);
    z=0;
    j=1;
 }
 if (TECLA=='B') {
   lcd.clear();
   lcd.print("Posicion");
   delay(1000);
    z=1;
    j=1;
 }}}
lcd.clear();
lcd.print("C: Manual");
lcd.setCursor (0,1);
lcd.print("D: Teclado");
while (j!=2) {
   TECLA=teclado.getKey();
 if(TECLA!=NO_KEY){
 if (TECLA=='C') {
   lcd.clear();
   lcd.print("Manual");
   delay(1000);
    y=0;
    j=2;
 }
 if (TECLA=='D') {
   lcd.clear();
   lcd.print("Teclado");
   delay(1000);
    y=1;
    j=2;
 }}}
 j=0;
if ((z==0)&&(y==0)){
   Tm=50;
```

```
    K=1.8*255.0/(12.0);
    Ti=200;
    Td=0;
    inicializarConstantes();
    T_int=1000*Tm;
    K=K*V/255.0;
    pinMode(encoderPin1, INPUT_PULLUP);
    pinMode(encoderPin2, INPUT_PULLUP);
    pinMode(encoderPinSW, INPUT_PULLUP);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(intPin, INPUT);
    digitalWrite(encoderPin1, HIGH);   //pullup on
    digitalWrite(encoderPin2, HIGH); //pullup on
    pinMode(C1, INPUT_PULLUP);
    pinMode(C2, INPUT_PULLUP);
    digitalWrite(C1, HIGH); //pullup on
    digitalWrite(C2, HIGH); //pullup on
   attachInterrupt(digitalPinToInterrupt(encoderPin1),
updateEncoder_v, CHANGE);
   attachInterrupt(digitalPinToInterrupt(encoderPin2),
updateEncoder_v, CHANGE);
   attachInterrupt(digitalPinToInterrupt(C1), contar,
CHANGE);
   attachInterrupt(digitalPinToInterrupt(C2), contar,
CHANGE);
    lcd.init();
    lcd.backlight();
    Timer1.initialize(T_int);
    Timer1.attachInterrupt(vel);
 }
 if((z==1)&&(y==0)) {
    Tm=10;
    K = 15*255.0/12.0;
```

```
    Ti = 15*91;
    Td = 0.125*91;
    inicializarConstantes();
    T_int=1000*Tm;
    K=K*V/255.0;
    pinMode(encoderPin1, INPUT_PULLUP);
    pinMode(encoderPin2, INPUT_PULLUP);
    pinMode(encoderPinSW, INPUT_PULLUP);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(intPin, INPUT);
    pinMode(C1, INPUT_PULLUP);
    pinMode(C2, INPUT_PULLUP);
    digitalWrite(encoderPin1, HIGH); //pullup on
    digitalWrite(encoderPin2, HIGH); //pullup on
    digitalWrite(C1, HIGH); //pullup on
    digitalWrite(C2, HIGH); //pullup on
   attachInterrupt(digitalPinToInterrupt(encoderPin1),
updateEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(encoderPin2),
updateEncoder, CHANGE);
   attachInterrupt(digitalPinToInterrupt(C1), contar,
CHANGE);
   attachInterrupt(digitalPinToInterrupt(C2), contar,
CHANGE);
    lcd.init();
    lcd.backlight();
    Timer1.initialize(T_int);
    Timer1.attachInterrupt(pid_p);
 }
  if ((z==0)&&(y==1)){
    Tm=50;
    K=1.8*255.0/(12.0);
    Ti=200;
```

```
    Td=0;
    inicializarConstantes();
    T_int=1000*Tm;
    K=K*V/255.0;
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(intPin, INPUT);
    pinMode(C1, INPUT_PULLUP);
    pinMode(C2, INPUT_PULLUP);
    digitalWrite(C1, HIGH); //pullup on
    digitalWrite(C2, HIGH); //pullup on
   attachInterrupt(digitalPinToInterrupt(C1), contar,
CHANGE);
   attachInterrupt(digitalPinToInterrupt(C2), contar,
CHANGE);
    lcd.init();
    lcd.backlight();
    Timer1.initialize(T_int);
    Timer1.attachInterrupt(vel);
  }
  if ((z==1)&&(y==1)){
    Tm=10;
    K = 15*255.0/12.0;
    Ti = 15*91;
    Td = 0.125*91.0;
    inicializarConstantes();
    T_int=1000*Tm;
    K=K*V/255.0;
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(C1, INPUT_PULLUP);
    pinMode(C2, INPUT_PULLUP);
    digitalWrite(C1, HIGH); //pullup on
```

```
    digitalWrite(C2, HIGH); //pullup on
    lcd.init();
    lcd.backlight();
   attachInterrupt(digitalPinToInterrupt(C1), contar,
CHANGE);
   attachInterrupt(digitalPinToInterrupt(C2), contar,
CHANGE);
    Timer1.initialize(T_int);
    Timer1.attachInterrupt(pid_p);
  }
}

void loop() {
while ((z==0)&&(y==0)) {
  V_r = analogRead(A9);
 V_in=V_r*55.0/1023.0;
 sensorInt=analogRead(intPin);
intensidad=abs(((5.0*sensorInt/1023.0)-2.5)/0.185);
 pot=V_in*(u/255.0)*intensidad;
   if((V_in>35.5)||(V_in<5)||(intensidad>4)||(pot>50)){
    pinMode(4,OUTPUT);
    digitalWrite(4,LOW);
  }

  if(b==1) {
   lcd.clear();
   lcd.print("Vel_ex: ");
   lcd.print(encoderValue);
   lcd.setCursor (0,1);
   lcd.print("Vel_real: ");
   lcd.print(rpm);
   b=0;
  }
 if ((!digitalRead(encoderPinSW))) {
   encoderValue = encoderValue + 50*sentido;
```

```
    if (encoderValue>RPM_MAX) {
      encoderValue=RPM_MAX;
    }
    if (encoderValue<-RPM_MAX) {
      encoderValue=-RPM_MAX;
    }
      if
((encoderValue>-RPM_MIN)&&(encoderValue<RPM_MIN)){
        encoderValue=0;
      }

    while (!digitalRead(encoderPinSW))
    delay(10);
    b=1;
  }
  if (p==1) {
    PID(K, Td, Ti);
    p=0;
  }
  if (d>=10) {

    b=1;
    d=0;
  }
}
while((z==1)&&(y==0)) {
  V_r = analogRead(A9);
 V_in=V_r*55.0/1023.0;
 sensorInt=analogRead(intPin);
intensidad=abs(((5.0*sensorInt/1023.0)-2.5)/0.185);
 pot=V_in*(u/255.0)*intensidad;
   if((V_in>35.5)||(V_in<5)||(intensidad>4)||(pot>50)){
    pinMode(4,OUTPUT);
    digitalWrite(4,LOW);
  }
```

```
   d=encoderValue*(BPR/5);
   if ((!digitalRead(encoderPinSW))) {
    d=0;
    encoderValue=0;
   while (!digitalRead(encoderPinSW))
    delay(10);
    b=1;
  }
 if ((v>=50)||(b==1)) {
  lcd.clear();
  lcd.print("Pos_ex: ");
  lcd.print(d);
  lcd.setCursor (0,1);
  lcd.print("Pos_real: ");
  lcd.print(encodedValue);
  b=0;
  v=0;
  }
}
 while ((z==0)&&(y==1)){
 V_r = analogRead(A9);
 V_in=V_r*55.0/1023.0;
 sensorInt=analogRead(intPin);
intensidad=abs(((5.0*sensorInt/1023.0)-2.5)/0.185);
 pot=V_in*(u/255.0)*intensidad;
   if((V_in>35.5)||(V_in<5)||(intensidad>4)||(pot>50)){
    pinMode(4,OUTPUT);
    digitalWrite(4,LOW);
  }
    if((b==1)&&(h==0)) {
  lcd.clear();
  lcd.print("Vel_ex: ");
  lcd.print(encoderValue);
  lcd.setCursor (0,1);
  lcd.print("Vel_real: ");
```

```
 lcd.print(rpm);
 b=0;
 }
TECLA=teclado.getKey();
if(TECLA!=NO_KEY){
  tec=TECLA;
  if ((tec<=57)&&(tec>=48)){
    speedo=speedo*10+(tec-48);
  lcd.clear();
  lcd.print(speedo);
  h=1;
 }
if ((TECLA=='*')||(TECLA=='#')) {
  if (speedo>RPM_MAX) {
    speedo=RPM_MAX;
   }
  if ((speedo<RPM_MIN)&&(speedo!=0)) {
    speedo=RPM_MIN;
   }
  if (TECLA=='*') {
    encoderValue=speedo;
   }
  if (TECLA=='#') {
    encoderValue=-1*speedo;
   }
  speedo=0;
  h=0;
 }
 }

if (p==1) {
  PID(K, Td, Ti);
  p=0;
 }
if (d>=10) {
```

```
   b=1;
   d=0;
 }
 }
 while ((z==1)&&(y==1)){
 V_r = analogRead(A9);
V_in=V_r*55.0/1023.0;
sensorInt=analogRead(intPin);
intensidad=abs(((5.0*sensorInt/1023.0)-2.5)/0.185);
pot=V_in*(u/255.0)*intensidad;
  if((V_in>35.5)||(V_in<5)||(intensidad>4)||(pot>50)){
   pinMode(4,OUTPUT);
   digitalWrite(4,LOW);
 }
   TECLA=teclado.getKey();
   if(TECLA!=NO_KEY){
     tec=TECLA;
   if ((tec<=57)&&(tec>=48)){
     speedo=speedo*10+(tec-48);
     lcd.clear();
     lcd.print(speedo);
     h=1;
 }
 if ((TECLA=='*')||(TECLA=='#')) {
   if (TECLA=='*') {
    d=(speedo*(4.0*BPR/360.0)+0.5);
     encoderValue=speedo;
   }
   if (TECLA=='#') {
    d=-1*(speedo*(4.0*BPR/360.0)+0.5);
     encoderValue=speedo*-1;
   }
   speedo=0;
   h=0;
```

```
  }
    }

  if (((v>=50)||(b==1))&&(h==0)) {
    lcd.clear();
    lcd.print("Pos_ex: ");
    lcd.print(encoderValue);
    lcd.setCursor (0,1);
    if(encodedValue>=0){
   grados=(encodedValue*360.0/(BPR*4.0))+0.5;
    } else {
     grados=(encodedValue*360.0/(BPR*4.0))-0.5;
    }
    lcd.print("Pos_real: ");
    lcd.print(grados);
    b=0;
    v=0;
    }
  }
}
void vel() {
  rpm_f = 60.0*encodedValue/(Tm*BPR*4.0/1000.0);
//https://www.banggood.com/12V-320rpm12V-107rpm6V-160rpm-DC-Gear-Motor-Encoder-Motor-with-Mounting-Bracket-and-Wheel-p-1059966.html?cur_warehouse=CN&ID=522021
  rpm_f_d=rpm_f;
  if (rpm_f>=0){
    rpm_f = rpm_f + 0.5;
  }
  if (rpm_f<0){
    rpm_f = rpm_f - 0.5;
  }
  rpm = rpm_f;
  encodedValue = 0;
  p=1;
```

```
  d=d+1;
}

void PID (float K, float Td, float Ti) {
  long Ts;
  Ts = millis()-Tr;
  float p0=K*(Ts+Td)/Ts;
  float p1=K*(-1+Ts/Ti -2*Td/Ts);
  float p2=K*Td/Ts;
 e=encoderValue-rpm_f_d;
  u_f = p0*e + p1*e_1 + p2*e_2 + u_1;
  if (u_f>255) {
  u_f=255;
  }
  if (u_f<-255) {
    u_f=-255;
  }
  u_1=u_f;
  if (u_f>=0){
    u_f = u_f + 0.5;
    u=u_f;
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, HIGH);
    analogWrite (ENB,u);
  }
  if (u_f<0){
    u_f = u_f - 0.5;
    u=u_f;
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    analogWrite (ENB,abs(u));
  }
  e_2=e_1;
  e_1=e;
  Tr=millis();
```

```
}
void contar(){
  int MB = digitalRead(C1); //MB = most significant bit
  int LB = digitalRead(C2); //LB = least significant bit

  int encoder = (MB << 1) |LB; //converting the 2 pin
value to single number
  int sm  = (lastEncoder << 2) | encoder; //adding it to
the previous encoded value

  if(sm == 0b1101 || sm == 0b0100 || sm == 0b0010 || sm
== 0b1011) encodedValue++;
  if(sm == 0b1110 || sm == 0b0111 || sm == 0b0001 || sm
== 0b1000) encodedValue --;

  lastEncoder = encoder; //store this value for next time

}

void updateEncoder(){
  int MSB = digitalRead(encoderPin1); //MSB = most
significant bit
  int LSB = digitalRead(encoderPin2); //LSB = least
significant bit

  int encoded = (MSB << 1) |LSB; //converting the 2 pin
value to single number
  int sum  = (lastEncoded << 2) | encoded; //adding it to
the previous encoded value

  if(sum == 0b0010) {
    if (encoderValue>-RPM_MAX)
    {
    encoderValue --;
    }
```

```
    b=1;
    sentido=-1;
  }
  if(sum == 0b1000) {
    if (encoderValue<RPM_MAX) {
    encoderValue ++;
    }
    b=1;
    sentido = 1;
    }

  lastEncoded = encoded; //store this value for next time

}
void updateEncoder_v(){
  int MSB = digitalRead(encoderPin1); //MSB = most
significant bit
  int LSB = digitalRead(encoderPin2); //LSB = least
significant bit

  int encoded = (MSB << 1) |LSB; //converting the 2 pin
value to single number
  int sum  = (lastEncoded << 2) | encoded; //adding it to
the previous encoded value

  if(sum == 0b0010) {
    if
((encoderValue>-RPM_MAX)&&((encoderValue>RPM_MIN)||(encode
rValue<=-RPM_MIN)))
    {
    encoderValue --;
    } else if (encoderValue==0) {
      encoderValue=-RPM_MIN;
    } else {
      encoderValue=0;
```

```
    }
    b=1;
    sentido=-1;
  }
  if(sum == 0b1000) {
    if
((encoderValue<RPM_MAX)&&((encoderValue<-RPM_MIN)||(encode
rValue>=RPM_MIN))) {
    encoderValue ++;
    } else if (encoderValue==0) {
      encoderValue=RPM_MIN;
    } else {
      encoderValue=0;
    }
    b=1;
    sentido = 1;
    }

  lastEncoded = encoded; //store this value for next time

}
void pid_p() {
  float p0=K*(Tm+Td)/Tm;
  float p1=K*(-1+Tm/Ti -2*Td/Tm);
  float p2=K*Td/Tm;
  if ((encoderV-d)>=(BPR/30)) {
    encoderV=encoderV-(BPR/30);
  } else if ((encoderV-d)<=-(BPR/30)) {
    encoderV=encoderV+(BPR/30);
  } else {
    encoderV=d;
  }
  e = encoderV-encodedValue;
  u_f = p0*e + p1*e_1 + p2*e_2 + u_1;
  if (u_f>255) {
```

```
    u_f=255;
  }
  if (u_f<-255) {
    u_f=-255;
  }
    u_1=u_f;
    if (u_f>=0){
    u_f = u_f + 0.5;
    u=u_f;
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, HIGH);
    analogWrite (ENB,u);
  }
  if (u_f<0){
    u_f = u_f - 0.5;
    u=u_f;
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    analogWrite (ENB,abs(u));
  }
  e_2=e_1;
  e_1=e;
  v++;
}
void inicializarConstantes(){
  int tec;
   lcd.clear();
   lcd.print("A: Modelo pred.");
   lcd.setCursor (0,1);
   lcd.print("B: Cambios");
   do{
  TECLA=teclado.getKey();} while((TECLA!='A')&&(TECLA!
='B'));
    if (TECLA=='A') {
      lcd.clear();
```

```
    lcd.print("MODELO PRED");
    delay(1000);
  } else if (TECLA=='B') {
    lcd.clear();
    lcd.print("Cambios en PID");
    lcd.setCursor (0,1);
    lcd.print("A: NO   B:SI");
    do{
   TECLA=teclado.getKey();} while((TECLA!='A')&&(TECLA!
='B'));
    if (TECLA=='A') {
      lcd.clear();
      lcd.print("MODELO PRED");
      delay(1000);
    } else if (TECLA=='B') {
      lcd.clear();
      lcd.print("CAMBIOS PID");
      delay(1000);
      j=0;
      lcd.clear();
      lcd.print("NUEVA K (V/RPM)");
      while (j==0){
        TECLA=teclado.getKey();
        if(TECLA!=NO_KEY){
          tec=TECLA;
          if ((tec<=57)&&(tec>=48)&&(decimal==0)){
            speedo=speedo*10+(tec-48);
            lcd.clear();
            lcd.print(speedo);
          } else if (TECLA=='*') {
            decimal=1;}
            if(TECLA!=NO_KEY){
              tec=TECLA;
             if ((tec<=57)&&(tec>=48)&&(decimal!=0)){
             speedo_f=speedo_f*10+(tec-48);
```

```
            decimal++;
            speedo_ff=speedo_f;
            for (int i=1;i<decimal;i++) {
            speedo_ff=speedo_ff/10.0;}
            lcd.clear();
            lcd.print(speedo+speedo_ff);
            } else if (TECLA=='#') {
              K=speedo+speedo_ff;
              decimal=0;
              speedo=0;
              speedo_f=0;
              speedo_ff=0;
              j=1;
            }
       }

  }
}
j=0;
lcd.clear();
lcd.print("NUEVA Ti (ms)");
while (j==0){
    TECLA=teclado.getKey();
    if(TECLA!=NO_KEY){
      tec=TECLA;
      if ((tec<=57)&&(tec>=48)&&(decimal==0)){
        speedo=speedo*10+(tec-48);
        lcd.clear();
        lcd.print(speedo);
      } else if (TECLA=='*') {
        decimal=1;}
        if(TECLA!=NO_KEY){
          tec=TECLA;
          if ((tec<=57)&&(tec>=48)&&(decimal!=0)){
          speedo_f=speedo_f*10+(tec-48);
```

```
            decimal++;
            speedo_ff=speedo_f;
            for (int i=1;i<decimal;i++) {
            speedo_ff=speedo_ff/10.0;}
            lcd.clear();
            lcd.print(speedo+speedo_ff);
            } else if (TECLA=='#') {
              Ti=speedo+speedo_ff;
              decimal=0;
              speedo=0;
              speedo_f=0;
              speedo_ff=0;
              j=1;
            }
       }

  }
}
j=0;
lcd.clear();
lcd.print("NUEVA Td (ms)");
while (j==0){
    TECLA=teclado.getKey();
    if(TECLA!=NO_KEY){
      tec=TECLA;
      if ((tec<=57)&&(tec>=48)&&(decimal==0)){
        speedo=speedo*10+(tec-48);
        lcd.clear();
        lcd.print(speedo);
      } else if (TECLA=='*') {
        decimal=1;}
        if(TECLA!=NO_KEY){
          tec=TECLA;
          if ((tec<=57)&&(tec>=48)&&(decimal!=0)){
          speedo_f=speedo_f*10+(tec-48);
```

```
            decimal++;
            speedo_ff=speedo_f;
            for (int i=1;i<decimal;i++) {
            speedo_ff=speedo_ff/10.0;}
            lcd.clear();
            lcd.print(speedo+speedo_ff);
            } else if (TECLA=='#') {
              Td=speedo+speedo_ff;
              decimal=0;
              speedo=0;
              speedo_f=0;
              speedo_ff=0;
              j=1;
            }
       }

   }
}
j=0;
lcd.clear();
lcd.print("NUEVA Tm (ms)");
while (j==0){
    TECLA=teclado.getKey();
    if(TECLA!=NO_KEY){
      tec=TECLA;
      if ((tec<=57)&&(tec>=48)){
        speedo=speedo*10+(tec-48);
        lcd.clear();
        lcd.print(speedo);
      } else if (TECLA=='#') {
            Tm=speedo;
            speedo=0;
            j=1;
       }
     }
```

```
}
}
lcd.clear();
lcd.print("DIST MOT/ENC");
lcd.setCursor (0,1);
lcd.print("A: NO   B:SI");
do {
  TECLA=teclado.getKey();
} while ((TECLA!='A')&&(TECLA!='B'));
if (TECLA=='A') {
  lcd.clear();
  lcd.print("MOT Y ENCO PRED");
  delay(1000);
} else if (TECLA=='B') {
  j=0;
  lcd.clear();
j=0;
lcd.clear();
lcd.print("NUEVO RPM_min");
while (j==0){
    TECLA=teclado.getKey();
    if(TECLA!=NO_KEY){
      tec=TECLA;
      if ((tec<=57)&&(tec>=48)){
        speedo=speedo*10+(tec-48);
        lcd.clear();
        lcd.print(speedo);
      } else if (TECLA=='#') {
            RPM_MIN=speedo;
            speedo=0;
            j=1;
       }
    }
}
j=0;
```

```
lcd.clear();
lcd.print("NUEVO RPM_max");
while (j==0){
    TECLA=teclado.getKey();
    if(TECLA!=NO_KEY){
      tec=TECLA;
      if ((tec<=57)&&(tec>=48)){
        speedo=speedo*10+(tec-48);
        lcd.clear();
        lcd.print(speedo);
      } else if (TECLA=='#') {
            RPM_MIN=speedo;
            speedo=0;
            j=1;
       }
     }
}
j=0;
lcd.clear();
lcd.print("NUEVO BPR enc");
while (j==0){
    TECLA=teclado.getKey();
    if(TECLA!=NO_KEY){
      tec=TECLA;
      if ((tec<=57)&&(tec>=48)){
        speedo=speedo*10+(tec-48);
        lcd.clear();
        lcd.print(speedo);
      } else if (TECLA=='#') {
            BPR=speedo;
            speedo=0;
            j=1;
       }
     }
}
```

```
    }
   }
   j=0;
   lcd.clear();
   lcd.print("CONECTAR TENSION");
   while(j==0){
   V_r = analogRead(A9);
  V_in=V_r*55.0/1023.0;
  if((V_in>=5)&&(V_in<=35.5)){
    delay (1000);
    V_r = analogRead(A9);
    V_in=V_r*55.0/1023.0;
    if((V_in>=5)&&(V_in<=35.5)){
      V=V_in;
      j=1;
     }
    if(V_in>35.5){
      lcd.clear();
      lcd.print("TENSION>35V");
      delay(1000);
      lcd.clear();
      lcd.print("CONECTAR TENSION");
     }
   }
   }
}
```