UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

MODELADO Y DISEÑO DE CONTROL DE TRAYECTORIA DE UN VEHÍCULO MEDIANTE ARDUINO

AUTOR: ROBERT CONSTANTIN ANTONEAC ANTONEAC

TUTOR: ANTONIO GONZÁLEZ SORRIBES

Curso Académico: 2019-20

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

AGRADECIMIENTOS

A mis padres y hermana que me han apoyado para poder seguir formándome a lo largo de estos años, y porque sin ellos no habría llegado hasta donde he llegado y donde llegaré.

Asimismo, me gustaría mucho agradecer la oportunidad que se me ha ofrecido de estudiar esta carrera en una de las universidades con más prestigio de España. Por lo que agradezco a todos mis profesores y compañeros que me han acompañado a lo largo de este camino, en donde he aprendido muchas más cosas que la teoría sino a un modo de pensar distinto hasta ahora.

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

RESUMEN

En el presente trabajo se realiza la creación y diseño de un método de control el cual nos permitirá controlar de modo eficiente la velocidad de cuatro motores que forman parte de un vehículo gracias a un controlador , en concreto el Arduino MEGA 2560. El objetivo final será la de ser capaces de mantener el vehículo a una distancia de seguridad con respecto a un obstáculo.

Para poder desarrollar este proyecto será necesario la construcción de un prototipo a partir del cual se realizará el estudio del modelo físico que expliqué el comportamiento real del conjunto del sistema formado por él vehículo con los motores.

Con todos estos elementos se diseñará el controlador para que cumpla unas especificaciones determinadas. En una primera fase se lleva a cabo la identificación del modelo dinámico del vehículo, y en una segunda fase se diseña el controlador para conseguir mantener el vehículo suficientemente alejado del obstáculo.

Finalmente, con el fin de validar el diseño, se compara los resultados obtenidos en lazo abierto y en lazo cerrado tanto mediante herramientas de simulación como con el prototipo experimental

Palabras Clave: PID, controlador, Arduino, lazo abierto, lazo cerrado, control, motor CC y programación.

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

RESUM

En el present treball es realitza la creació i disseny d'un mètode de control el qual ens permetrà controlar de manera eficient la velocitat de quatre motors que formen part d'un vehicle gràcies a un controlador, en concret el Arduino MEGA 2560. L'objectiu final serà la de ser capaços de mantenir el vehicle a una distància de seguretat respecte a un obstacle.

Per poder desenvolupar aquest projecte serà necessari la construcció d'un prototip a partir d'el qual es realitzarà l'estudi de el model físic que vaig explicar el comportament real del conjunt de el sistema format pel va vehicular amb els motors.

Amb tots aquests elements es dissenyarà el controlador perquè compleixi unes especificacions determinades. En una primera fase es porta a terme la identificació de el model dinàmic de el vehicle, i en una segona fase es dissenya el controlador per aconseguir mantenir el vehicle prou allunyat de l'obstacle.

Finalment, per tal de validar el disseny, es compara els resultats obtinguts en llaç obert i en llaç tancat tant mitjançant eines de simulació com amb el prototip experimental

Paraules clau: PID, controlador, Arduino, llaç obert, llaç tancat, control, motor CC i programació.

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

ABSTRACT

In this work, the creation and design of a control method is carried out which will allow us to efficiently control the speed of four motors that are part of a vehicle thanks to a controller, specifically the Arduino MEGA 2560. The final objective will be that of being able to keep the vehicle at a safe distance from an obstacle.

In order to carry out this project, the construction of a prototype will be necessary, from which the study of the physical model will be carried out, explaining the actual behavior of the entire system formed by the vehicle with the engines.

With all these elements, the controller will be designed to meet certain specifications. In a first phase, the identification of the dynamic model of the vehicle is carried out, and in a second phase, the controller is designed to keep the vehicle sufficiently far from the obstacle.

Finally, in order to validate the design, the results obtained in open loop and closed loop are compared both by means of simulation tools and with the experimental prototype

Keywords: PID, controller, Arduino, open loop, closed loop, control, DC motor and programming.

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

Documentos contenidos en el TFG

- Memoria.
- Presupuesto.
- Anexo I: Programa Principal.
- Anexo II: Programa obtención de velocidad.

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

DOCUMENTO I:

MEMORIA

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

ÍNDICE DE LA MEMORIA

ÍNDICE DE LAS ILUSTRACIONES

1. INTRODUCCIÓN

1.1 Objetivo del proyecto.

El objetivo principal de este documento es explicar los pasos seguidos en el diseño de un controlador PID con los cuales seamos capaces de controlar dentro de unos parámetros predefinidos un sistema. El sistema a controlar consiste en un vehículo robot con cuatro ruedas motrices. El vehículo robot y su entorno será el sistema a controlar con la finalidad de obtener el funcionamiento deseado en base a unas especificaciones de comportamiento.

Los objetivos que se pretenden con este tipo de recomendaciones son los siguientes:

- Diseño y montaje de un vehículo móvil con todos los elementos necesarios para su funcionamiento.
- Aplicar los conocimientos físicos y técnicos adquiridos a lo largo de la carrera para la aplicación de un modelo que se ajuste a la realidad del vehículo móvil.
- Diseño teórico del controlador PID con el cual podemos cumplir el objetivo de distancia de seguridad.
- Programación con el programa Arduino (2020) con el cual sea posible aplicar el controlador PID diseñado para poder controlar efectivamente el vehículo móvil.

Se desarrollará cada uno de los puntos anteriores gracias a los conocimientos adquiridos en los diferentes campos: automática, electrónica, física, eléctrica, informática, etc.

1.2 Antecedentes.

Desde la aparición de los primeros coches se han ido creando sistemas de seguridad. Gracias al desarrollo de la electrónica se han podido crear sistemas de seguridad cada vez más sofisticados como sensores digitales en cada una de las partes mecánicas, sensores de aparcamiento, regulación de la oscuridad en los retrovisores, detección de vehículos en los puntos muertos. Con este trabajo se busca el implementar un elemento más de seguridad desarrollando la idea y el concepto a pequeña escala con un vehículo robot el cual sea capaz de detectar un obstáculo mediante un sensor y sea capaz de actuar en consecuencia para evitar colisiones. Para implementar esta idea se va a utilizar el concepto del controlador PID pues es la herramienta más utilizado mundialmente para el control de procesos.

1.3 Motivación.

La principal motivación que he tenido para realizar este trabajo surge del gran interés en la informática y automática. Este interés surge de los dos años de bachillerato en los cuales tuve contacto con la informática. Para ser más concreto, en el conocimiento de programación en C++. Cuando empecé a interesarme en la programación no era consciente de la gran implicación que esta tiene en el día a día, ya que únicamente generaba pequeños programas como calculadoras o mini programas con puertas lógicas de decisión.

Así pues, desde el primer curso del grado he ido descubriendo cada vez más el mundo de la programación y la forma en la cual somos capaces de controlar desde los procesos más simples hasta los más complejos. Gracias a asignaturas como ''Sistemas Automáticos'' en la cual se introducía el concepto de modelar un sistema real y transformar ese mismo sistema en ecuaciones. Ya en ''Tecnologías Automáticas'' ya aprendíamos a crear controladores para cada uno de los sistemas. Por último, cabe destacar la asignatura de ''Tecnologías Industriales'' en la cual ya juntaban perfectamente ambos mundos entre la programación y el control de sistemas.

Por lo que el motivo de elegir este trabajo fue por un lado el carácter multidisciplinario, ya que para poder modelar cualquier sistema hay que conocer las leyes y ecuaciones por la cual se rige. Y por otro lado el concepto de ser capaz de programar cualquier programa, el cual se puede ver en acción y funcionamiento en la vida real más allá de una pantalla.

1.4 Justificación.

La principal justificación para realizar el trabajo académico sobre este tema reside como se ha mencionado anteriormente en el atractivo de la informática y automatización.

Siendo esta una excelente oportunidad para indagar en el funcionamiento del controlador PID y de su implementación en cualquier sistema gracias a la programación del microcontrolador Arduino Mega 2560.

Por lo que respecta al interés tecnológico y social es muy alto, puesto que el objetivo final se lograra construir e implementar un sistema más de seguridad para vehículos, con el cual se pondrán evitar accidentes que anteriormente eran impensables que se podrían evitar.

Asimismo, al trabajar con Arduino Mega 2560 se logra un mayor conocimiento sobre el funcionamiento de este. Este hecho facilitará la construcción de diferentes sistemas tanto en el ámbito industrial como académico.

2. FUNDAMENTOS TEÓRICOS.

Karl J. Åström y Tore Hägglund definieron el PID como: ‘’Una implementación simple de la idea de realimentación. Tiene la capacidad de eliminar errores en estado estacionario mediante la acción integral, y puede anticipar el futuro con la acción derivativa’’. (CONTROL PID AVANZADO, PEARSON EDUCACIÓN, 2009)

Por lo tanto, el PID es un controlado con el cual podemos obtener un estado final deseado. Para lograr el PID este objetivo como su nombre indica el PID está formado por tres partes: Proporcional, Integral y Derivado.

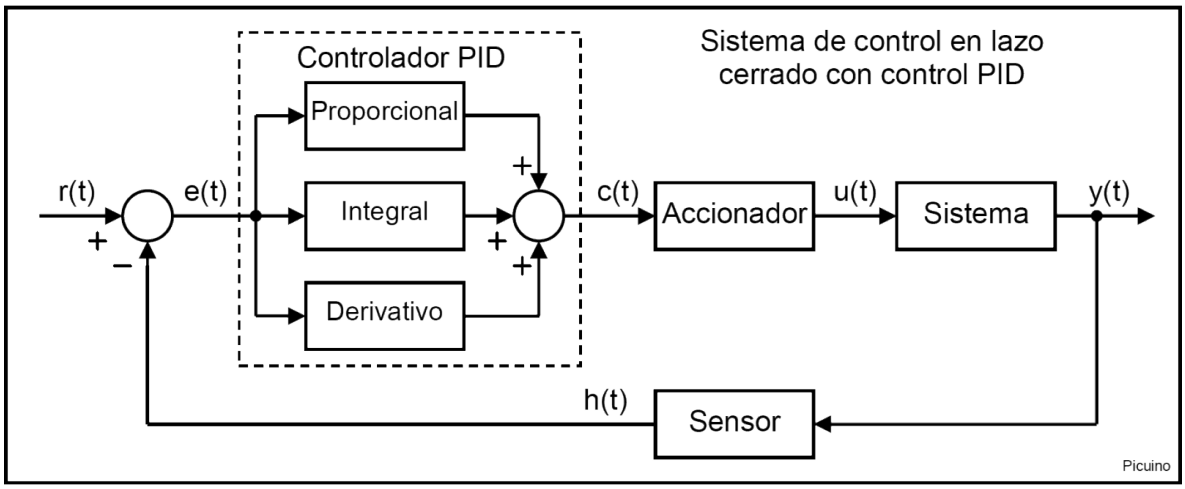

Figura 1.Funcionamiento del controlador PID

Una vez explicado brevemente que es el PID, el cual se verá con más detalle, vamos a ver los demás elementos del sistema de control:

Sensor: Es el que será el que mida aquel valor que se quiere controlar como temperatura, velocidad, distancia, caudal, luminosidad. Existen diferentes tipos de valores a controlar como sensores existen en la actualidad para cada parámetro. Este sensor dará una información en tiempo real de lo que está sucediendo en el sistema a controlar.

Error: Una vez que se tiene el valor medido por el sensor se ha de comparar con el valor objetivo. La diferencia entre ambos valores es el error, que será utilizada por el PID para calcular las acciones que se han de tomar.

Accionador: En lazado a lo anterior el actuador será aquella máquina, mecanismo o subconjunto de estos que recibirán las órdenes del controlador PID y actuarán en consecuencia sobre el sistema. Al haber una actuación de los accionadores sobre el sistema los más obvio y normal será que habrá una variación en el valor que el sensor mide así cerrando el bucle de retroalimentación.

2.1 Controlador PID.

Como se ha mencionado anteriormente el PID está formado por Proporcional, Integral y Derivado. En este apartado definiremos y explicaremos en detalle cada una de las partes.

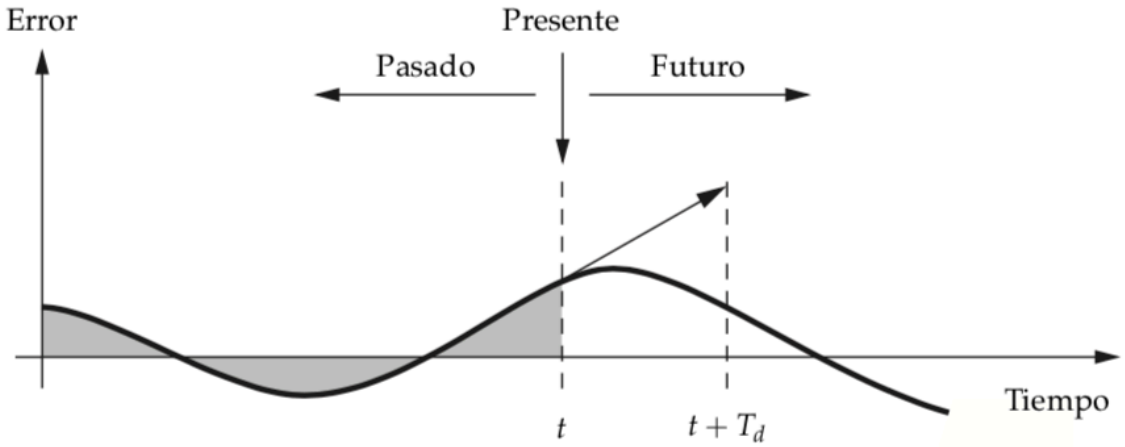

Figura 2. Variación del error en el tiempo

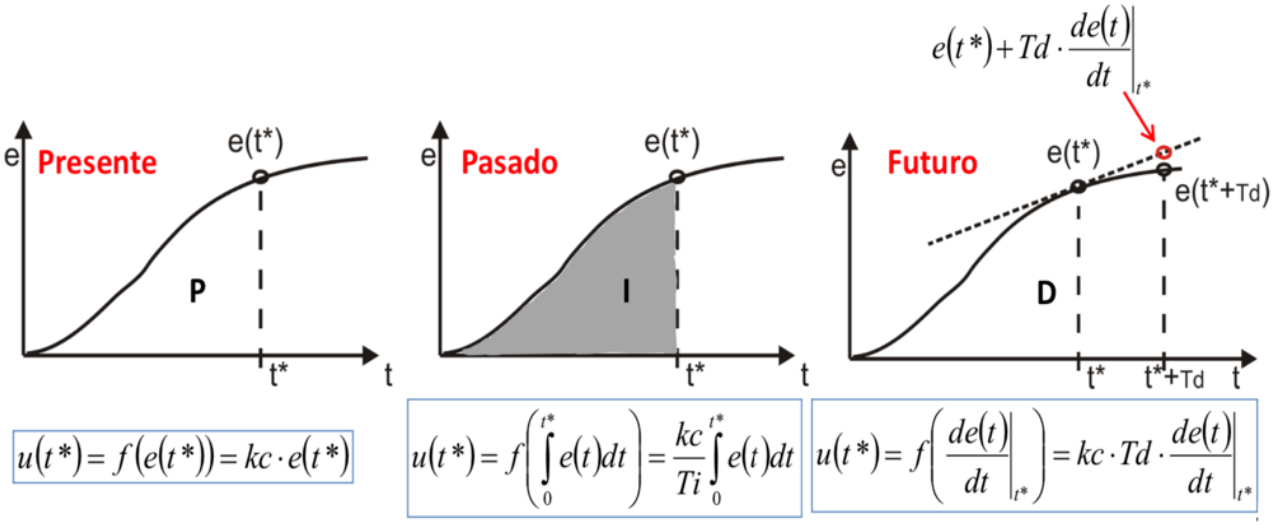

Figura 3. Tipos de controlador P, I y D

2.1.1 Acción Proporcional.

La acción proporcional repercute directamente sobre el error en el instante actual. Es por eso que solo influye en el presente y no tiene en cuenta el tiempo en el pasado o futuro. Viene definida como:

$$u(t) = kc * e(t) \qquad [1]$$

Donde K es la ganancia del controlador. A mayor ganancia mayor brevedad y contundencia se actúa sobre el sistema. Pero debido a esto mismo, el sistema será cada vez más inestable.

2.1.2 Acción Integral.

La acción integral ya tiene en cuenta el tiempo. Este tipo de acción controla el error acumulado a lo largo del tiempo por lo que se centra concretamente en el pasado. Siendo su expresión:

$$\mathrm{u(t)} = \frac{kc}{Ti}\int_0^t e(t)dt \qquad [2]$$

Suponiendo que estamos en el régimen permanente no hay variación de u (salida) por lo que implica la siguiente ecuación:

$$\frac{u(t)}{dt} = \frac{kc}{Ti}\int_0^t e(t)dt \qquad [3]$$

$$0 = \frac{kc}{Ti}\, et \qquad [4]$$

De esta ecuación se saca la conclusión que al aplicar una acción integral el error en el estado estacionario será siempre cero.

2.1.3 Acción Derivada

Este tipo de acción calcula la velocidad con la cual está cambiando el error. Esto permite predecir futuros valores del error. Por tanto se podrá evitar que el error vaya creciendo y por ende realizar cambios para minimizar el error. Su expresión es la siguiente:

$$u(t) = kc * Td * \frac{de(t)}{dt} [5]$$

El principal inconveniente a este tipo de acción es el hecho de que es muy sensible al ruido existente en el sistema, ya que lo amplifica y al mínimo error la ganancia aumenta significativamente provocando así oscilaciones en la respuesta.

La suma de los tres tipos de acciones constituye lo que se conoce como PID;

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \qquad [6]$$

2.2 La función de transferencia.

Cuando se controla un sistema para ser más concretos un valor de este, se tiene que modelar y representar físicamente este sistema con la ayuda de las ecuaciones y leyes en las cuales se basan su funcionamiento. Así pues, la función de transferencia es la relación entre la señal de entrada respecto a la señal de salida en el dominio de Laplace.

$$Gr = \frac{U(s)}{Y(s)} \qquad [7]$$

En este caso particular Gr es la función de transferencia siendo U(s) la señal de salida e Y(s) la señal de entrada. Se denomina polos a las raíces del denominador de la función de transferencia y ceros las raíces del nominador.

2.3 Obtención de valores del PID.

Para el ajuste del cero de la acción integral y derivada vamos a explicar los dos criterios que se tiene (Argumento y Cancelación). Por otro lado, también se tiene el criterio del módulo para ajustar la acción proporcional.

2.3.1 Criterio de Cancelación.

Mediante el criterio de cancelación se pretende que el cero del PID sea igual a uno de los polos de la función de transferencia. Con este criterio lo que se consigue es eliminar uno o dos de los polos de la función de transferencia. Podremos eliminar uno o dos polos según hayamos elegido un controlador PI, PD o PID.

No podremos cancelar los polos positivos ni a aquellos polos que no cumplan con las especificaciones dinámicas.

2.3.2 Criterio del Argumento.

Mediante el criterio del argumento lo que se quiere conseguir es la colocación del cero , para que los polos dominantes pasen por el límite de las especificaciones dinámicas a un valor concreto del valor proporcional. Se tiene que colocar en el plano todos los polos y ceros que se conozcan del PID como de la función de transferencia.

Una vez colocado todos los polos y ceros se ha de calcular los ángulos que forman cada uno con el punto entre la intersección de la especificación de tiempo de establecimiento y sobreoscilación.

Para calcular la posición del cero del controlador se obtiene mediante la siguiente ecuación:

$$\sum_{i=0}^{n} \alpha_i - \sum_{j=0}^{n} \beta_i - \beta_a = \pi \qquad [8]$$

Siendo $\sum_{i=0}^{n} \alpha_i$ cada uno de los polos $\sum_{j=0}^{n} \beta_i$ es cada uno de los ceros y β_a es el ángulo del polo que se tiene que averiguar. Una vez que se averigua el ángulo simplemente mediante ecuaciones trigonométricas se logra colocar el polo en el plano.

Cabe destacar que los criterios de cancelación y argumento para el ajuste de la acción integral y/o derivada no son excluyentes, ya que, si podemos realizar la cancelación de uno de los ceros del controlador con uno de los polos por cumplir con las condiciones de estar dentro de las especificaciones dinámicas, podremos cancelar y posteriormente realizar el criterio del argumento.

2.3.3 Criterio del Módulo

Básicamente para la realización del criterio del módulo tenemos que aplicar la siguiente ecuación:

$$K_i = \frac{\prod |s + p_i|}{\prod |s + z_j|}\Bigg|_{s=P_d} \quad [9]$$

Siendo p cada uno de los polos y z cada uno de los ceros que previamente hemos calculado. Mientras que Pd es un punto que podemos asegurar que pertenece al lugar de las raíces. Debido a todo esto el criterio del módulo se aplica una vez que se hayan aplicado los criterios de cancelación y/o del argumento.

2.4 Control mediante puente en H y PWM.

El puente en H es un conjunto de cuatro transistores colocados estratégicamente mediante los cuales al activarlos en secuencias concretas se logra controlar el sentido de la corriente para así poder controlar el sentido de giro del motor al cual está conectado.

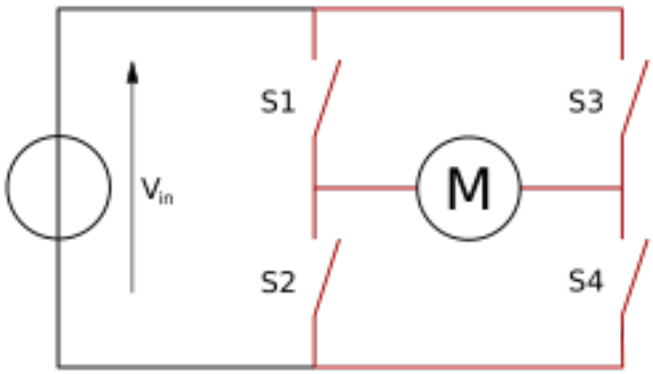

Figura 4. Esquema de puente en H.

Al estar los transistores S1 y S4 cerrados mientras que S2 y S3 están abiertos logramos que el sentido de la intensidad que recorre el circuito sea horario como se puede apreciar en la siguiente figura:

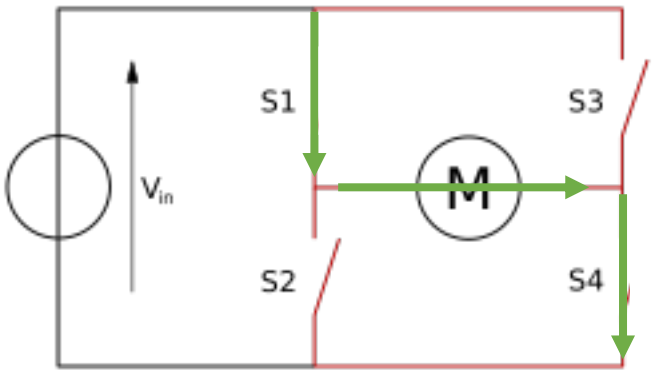

Figura 5. Esquema de puente en H sentido horario.

Si por lado contrario se necesita que el motor gire en sentido contrario la secuencia de transistores será la de S2 y S3 cerrados mientras que S1 y S4 abiertos para lograr que la intensidad que recorre el circuito sea antihoraria:

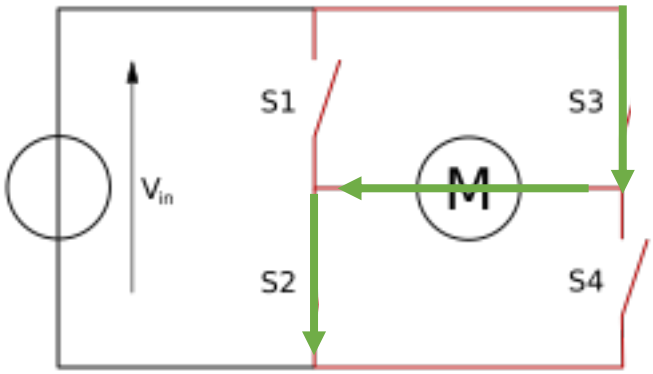

Figura 6. Esquema de puente en H sentido anti horario.

Por otro lado, si al mismo tiempo que se cierra la secuencia de transistores necesarios según el sentido deseado del motor se puede realizar que esos transistores se cierren y abran con una frecuencia deseada. Siendo así que si la frecuencia es muy alta obtenemos toda la potencia de la fuente mientras que si disminuye la frecuencia del transistor sé ira decrementando en proporción la potencia obtenida en los motores.

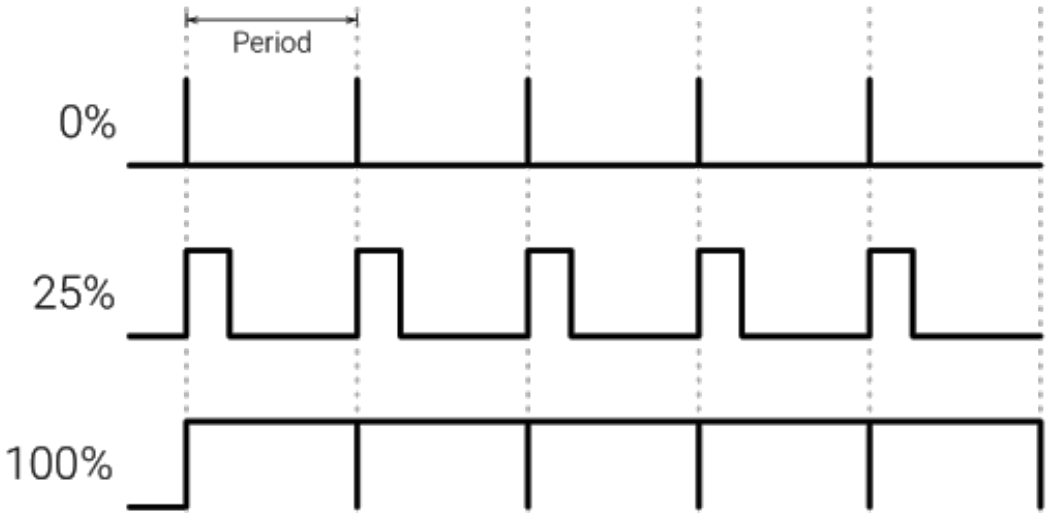

Figura 7. Gráfico de control mediante PWM.

3. HARDWARE

3.1 Componentes

3.1.1 Arduino Mega 2400

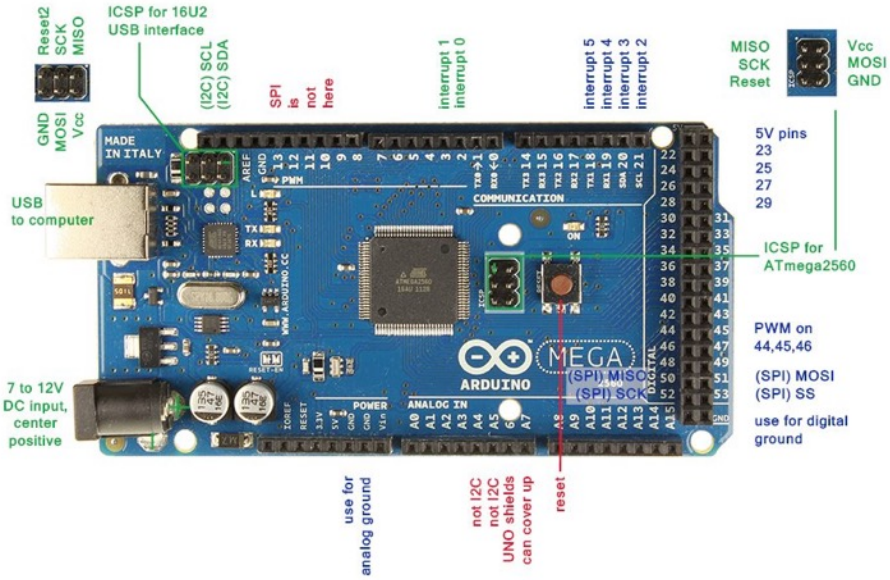

Figura 8. Arduino Mega 2560

El concepto de Arduino surgió de la necesidad de crear microcontroladores para el ámbito de la enseñanza a un precio asequible. Con el paso del tiempo se han ido desarrollando y mejorando a partir de Arduino UNO, el primer Arduino en crearse. Siendo así que el Arduino Mega 2560 es un microcontrolador siendo la versión ampliada de la tarjeta Arduino UNO. Todos los Arduinos están basadas en el concepto en un hardware y software libre, por lo que es sencillo de utilizar y muy versátil a la hora de su aplicación.

Algunas de las características de este microcontrolador son:

- Voltaje Operativo: 5V
- Voltaje de Entrada(límites): 6-20V.
- Pines digitales de Entrada/Salida: 54 (de los cuales 15 proveen salida PWM)
- Pines análogos de entrada: 16
- Corriente DC por cada Pin Entrada/Salida: 40 mA
- Corriente DC entregada en el Pin 3.3V: 50 mA
- Memoria Flash: 256 KB (8KB usados por el bootloader)
- Clock Speed: 16 MHz

El controlador contiene un botón de Reset. El Arduino Mega 2560 también contiene una conexión Jack hembra para alimentación mediante fuente externa no siendo recomendando la alimentación con un voltaje superior al voltaje operativo de 5V. Al sobrepasar este límite existe el riesgo de destruir el microcontrolador es por eso por lo que será necesario de un módulo de expansión el cual controle los motores a mayores voltajes y en caso de error únicamente se romperá el módulo de expansión y no todo el circuito con el Arduino incluido. En el siguiente apartado detallaremos el funcionamiento de este módulo.

3.1.2 L293D Shield

Es un módulo de expansión que utilizaremos para controlar los motores con seguridad de no comprometer todo el circuito pudiendo conectar y controlar 4 motores DC directa o 2 motores de paso a paso o 2 servomotores. Dentro de las posibles opciones se ha elegido el controlar 4 motores DC directos a la vez.

La forma de conectar este módulo se realiza ajustando los pines machos del Shield con los pines hembras del Arduino Mega de tal modo que todo los pines macho del Shield queden conectados del modo que aparece a continuación:

Figura 9. Conexión Arduino Mega 2560 con L293D Shield

Dentro de las características de este módulo que más nos interesan son las siguientes:

- Voltaje de potencia (motores): 4.5V-24V DC
- Incorpora 2 circuitos integrados L293D proporcionando 4 puentes-H completos
- Diodos de protección internos contra voltajes inversos generados por las cargas inductivas
- 4 canales (M1, M2, M3 y M4) para controlar igual número de cargas inductivas como motores DC o 2 motores paso a paso unipolares o bipolares de entre 4.5 V y 24 V
- Corriente máxima continua en cada una de las salidas M1, M2,M3 y M4: 600 mA (0.6 A)
- Corriente máxima pico no repetitivo en cada una de las salidas M1, M2, M3 y M4: 1.2 A

Como se aprecia las características del Shield es posible alimentarlo con hasta 24 V con una fuente externa, lo que da mayor potencia de alimentación de los motores respecto a como si se hubiera alimentado directamente con el Arduino Mega 2560. Para poder alimentar con una fuente externa se tiene que desactivar el Jumper del módulo.

El Shield también cuenta con diodos de protección internos por lo que se convierte en un elemento de protección de todo el sistema frente a sobrecargas de tensión.

El Shield al incorporar 2 circuitos integrados L293D proporciona 4 puentes-H lo que hace posible el controlar los 4 motores mediante modulación de ancho de pulso (PWM) como se ha visto anteriormente.

3.1.3 Módulo HC-SR04 sensor de distancia por ultrasonidos.

El módulo HC-SR04 es un elemento mediante el cual sé pude medir la distancia, así como la presencia de cualquier obstáculo. Los fundamentos en los cuales se rige este módulo es en la existencia de un emisor y un receptor de ultrasonidos. El pulso de ultrasonido es generado y viaja por el medio hasta chocar con un objeto u obstáculo. Lo que genera un pulso de rebote el cual será captado por el receptor. El módulo puede medir el tiempo que tarda el pulso en chocar y rebotar, al conocer la velocidad de este pulso que es constante en el aire, por regla de tres se puede medir la distancia real hasta el un objeto u obstáculo.

Figura 10. Módulo HC.SR04

Como se puede apreciar en la imagen anterior este módulo está caracterizado por sus dos cilindros que sobresalen. Se puede identificar que el cilindro con la T es el transmisor mientras que el cilindro con la R es el receptor. La alimentación de este módulo es con una tensión de 5V la cual puede ser suministrada directamente por el Arduino Mega. Dentro de sus características de medición se puede medir una distancia de entre 2 cm hasta 400 cm y tiene un ángulo de visión menor a 15°. Por lo que respecta a la resolución del módulo es de 0,3 cm de variación entre la distancia real y la medida.

3.1.4 Motor de continua con rueda.

Figura 11. Motor reductor con rueda.

El motor es el elemento al cual se realizará el control mediante el microcontrolador programado con el PID diseñado. Para este tipo de motores no se tiene unas especificaciones técnicas en detalle lo que implica la necesidad calcular los parámetros que rigen el motor. Dentro de las características que conocemos de este tipo de motor son las siguientes:

- Voltaje de Operativo: 3V-12V (recomendado 6 a 8 voltios)
- Torque Máximo: 2 kg/cm
- Reductora: 48: 1
- Radio de rueda: 32 mm

Este tipo de motores tiene un eje alargado que sobresale por ambas partes del motor. Con este eje alargado se puede conectar en una parte la rueda motriz mientras que en el otro lado se conectara un encoder con agujeros.

Figura 12. Encoder con agujeros.

Este encoder al estar conectada al mismo eje motriz girar a la misma velocidad pues esa es su función, la de permitir calcular la velocidad del eje.

3.1.5 FC-33 Sensor de medición de infrarrojo fotoeléctrico

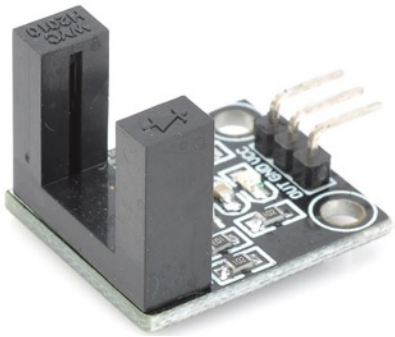

Figura 13. FC-33 Sensor Infrarrojo fotoeléctrico.

Para poder calcular la velocidad se hace pasar con un infrarrojo un haz de fotos los cuales son captados por un fotorreceptor. Cuando el fotorreceptor recibe la luz emite una señal eléctrica analógica la cual será captada por el Arduino Mega. Siendo así, si conocemos el número de agujeros que tiene el encoder y la dimensión de este podemos calcular la velocidad de giro al medir cuantos agujeros se han medido en un determinado intervalo de tiempo. De este modo, se podrá calcular la velocidad de giro.

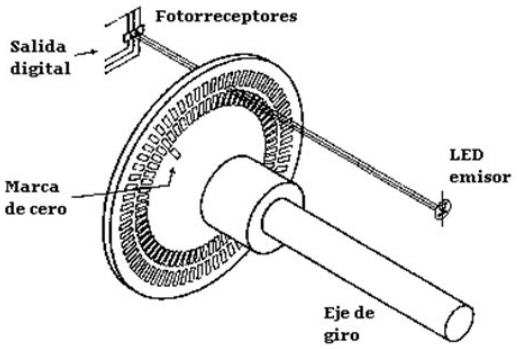

Figura 14. Esquema de medición sensor infrarrojo.

3.1.6 Chasis del coche

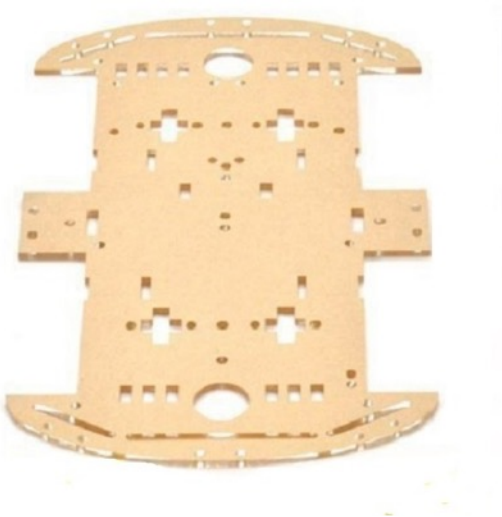

Figura 15. Chasis del Coche.

El chasis del motor está formado por dos placas los cuales se conectarán entre ellas dejando un espacio entremedio. Este será el elemento principal sobre el cual se montará los diferentes elementos por lo que necesitamos que esta sea rígida y resistente. Al mismo tiempo las dos placas tienen agujeros incorporados de forma nativa, lo que facilita el montaje con tuercas.

3.1.7 Cables de conexión

Será necesario la interconexión de los diferentes elementos será necesario cables que realicen las conexiones eléctricas. Para esta función utilizaremos los cables ``Dupont''. Este tipo de cable son de hilo de cobre envueltos con aislante de diferentes colores lo que facilita su identificación.

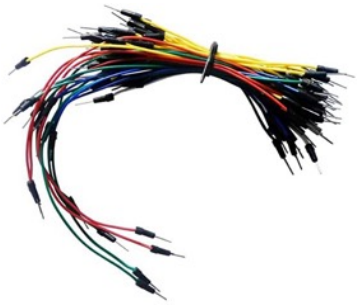

Figura 17. Cable Dupont Macho/Macho

Figura 16. Cable Dupont Macho/Hembra

Usaremos dos tipos de cables. El primer tipo será los cables Macho/Hembra, principalmente este tipo de cables lo usaremos para conectar cada uno de los diferentes módulos/sensores con el Arduino Mega.

El otro tipo de cable utilizado será el Macho/Macho el cual se usará para conectar el Shield con cada uno de los motores los cuales transmitirán toda la potencia a estos.

3.1.8 Adaptador de corriente.

La alimentación del Shield se realizará mediante un adaptador de corriente de voltaje variable. Mediante este adaptador podemos alimentar con una tensión de entre 3V-12V. Siendo que elegiremos una tensión de 9V para no comprometer el sistema eléctrico.

Figura 18. Adaptados de corriente.

3.2 Montaje.

Para la explicación del montaje sé ira explicando pasa a paso en detalle:

Para comenzar se ha conectado el Arduino Shield L293D al Arduino Mega, esto es un proceso sencillo pues se ajusta a la perfección los pines machos del Shield con los pines hembras del Arduino Mega como se ha visto anteriormente para ser más concretos en la Fig. 9.

El siguiente paso que se ha realizado ha sido la de colocar el encoder en cada uno de los motores teniendo en cuenta que este encoder tiene que quedar en la parte interna del vehículo. Así mismo también se ha conectado y soldado dos cables al motor para poder conectarlos al Arduino Shield L293D a posteriori.

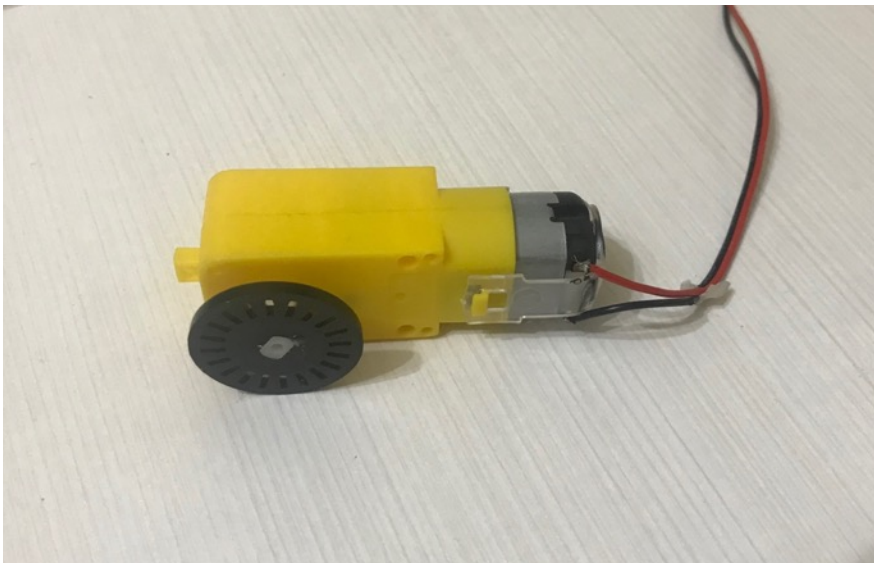

Figura 19. Motor con encoder y cables soldados.

Una vez que se tiene todos los motores con sus respectivos cables soldados y con el encoder colocado estratégicamente, se realiza la colocación de esos motores sobre el chasis inferior del vehículo. Esto se realiza gracias a que el chasis tiene unos agujeros en donde se colocan dos barrillas que al colocarlas en el chasis se quedan inmóviles. Cada una de las barrillas van a cada lado del motor y se sujetan a este mediante tornillo y tuerca.

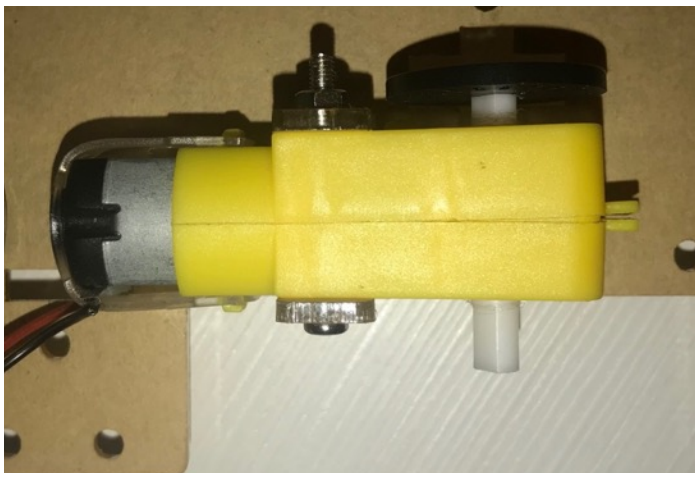

Figura 20. Vista superior de la colocación del motor.

En las dos próximas figuras se verá en detalle la colocación de los tornillos y tuercas.

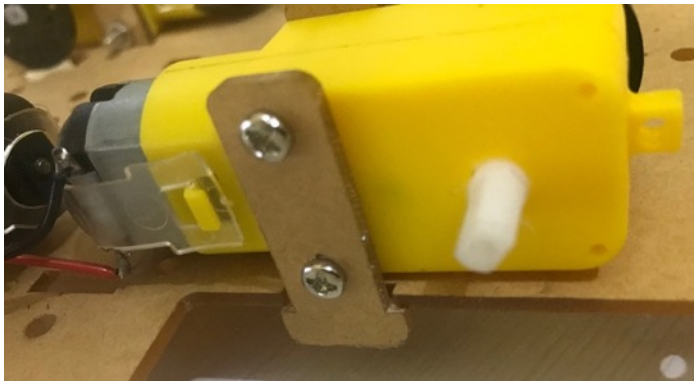

Figura 21.Vista del lateral derecho de la colocación del motor.

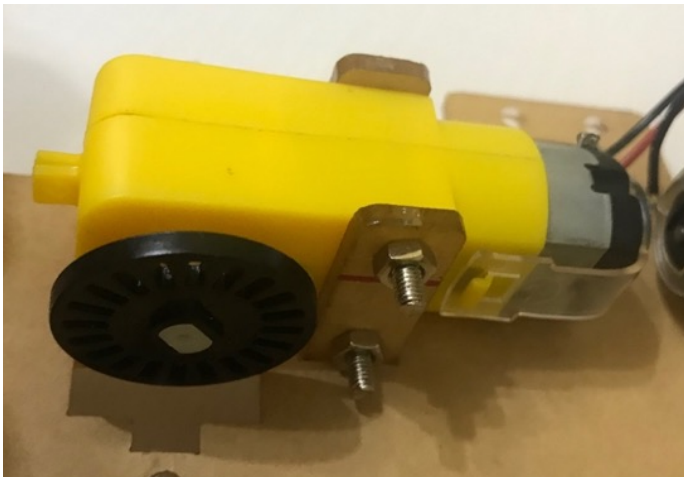

Figura 22. Vista del lateral izquierda de la colocación del motor.

Una vez sujeto todos los motores al chasis inferior se realiza la colocación de cada una de las ruedas al motor

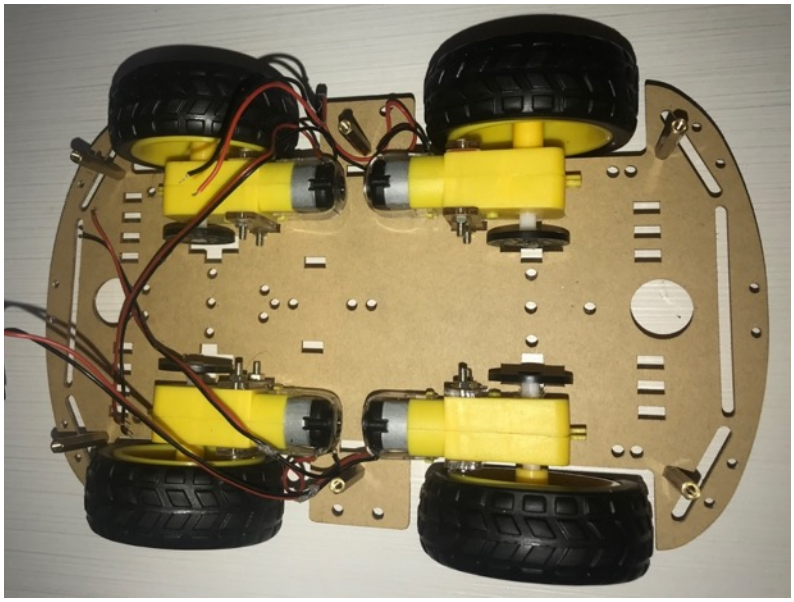

Figura 23. Chasis inferior montado.

Por lo que respecta al chasis superior se realiza la colocación de dos sensores FC-33 infrarrojo fotoeléctricos. Se ha decidido instalar dos sensores para medir la velocidad de ambas ruedas y poder extraer conclusiones de los datos obtenidos.

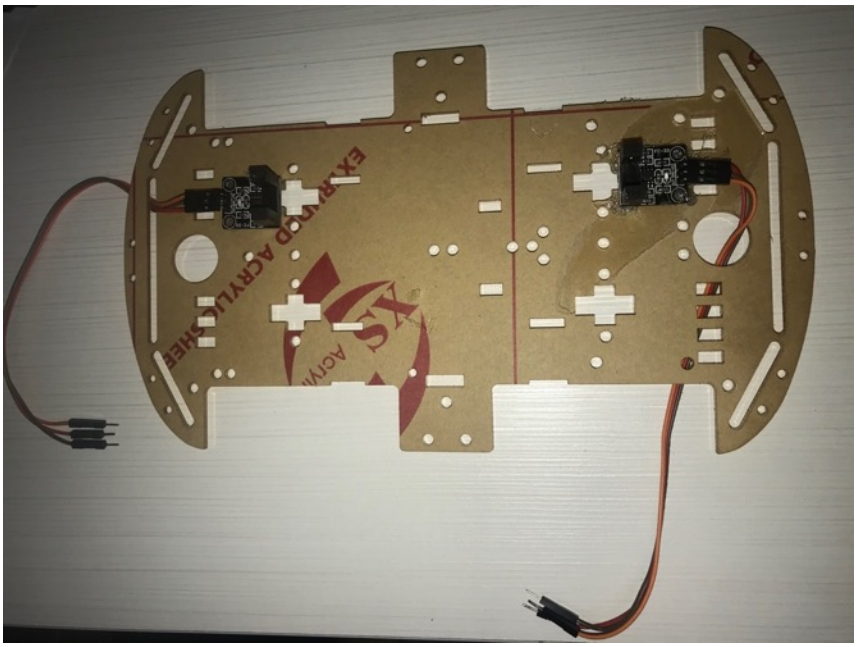

Figura 24. Colocación de sensores FC-33.

Cabe destacar que estos sensores únicamente se usaran en la parte de obtención de valores para la determinación de las variables que nos interesan. Estos dos sensores se pegan al chasis superior con una pistola de silicona y se mide con precisión para que el encoder de las ruedas queden justamente en medio del sensor con lo que se pueda hacer la medición, tal y como se pude apreciar en la Fig.25.

Figura 25. Ajuste perfecto entre encoder y sensores FC-33.

En la parte superior al chasis superior se realiza la colocación con la pistola de silicona del sensor HC-SR04 sensor de distancia por ultrasonidos el cual será fundamental para nuestro proceso de control. Su colocación es tal como se aprecia en la Fig.26.

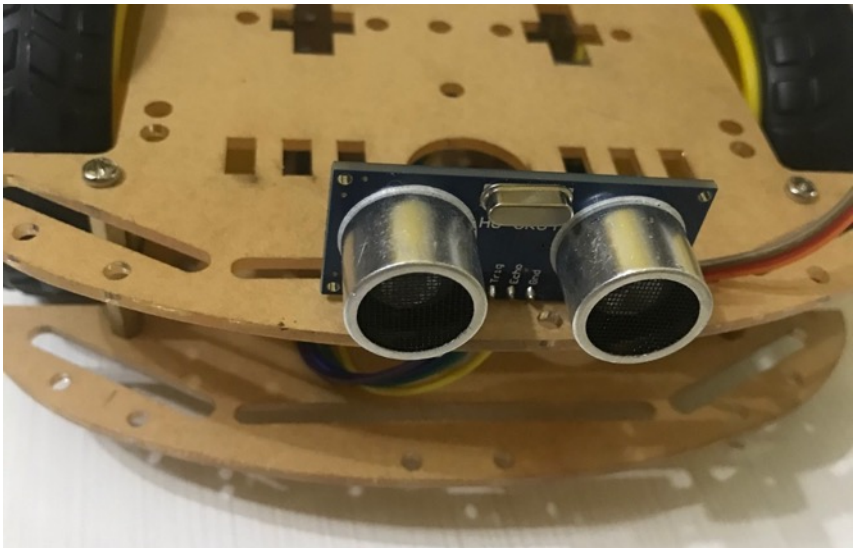

Figura 26. Colocación de sensor HC-SR04.

Al mismo tiempo que se ha realizado la colocación del sensor de ultrasonidos se hacen pasar los cables a través del chasis superior y se hace las sujeciones entre el chasis superior y el inferior mediante seis pilares las cuales sujetan ambos chasis mediante tornillo. Dando como resultado la Fig.27.

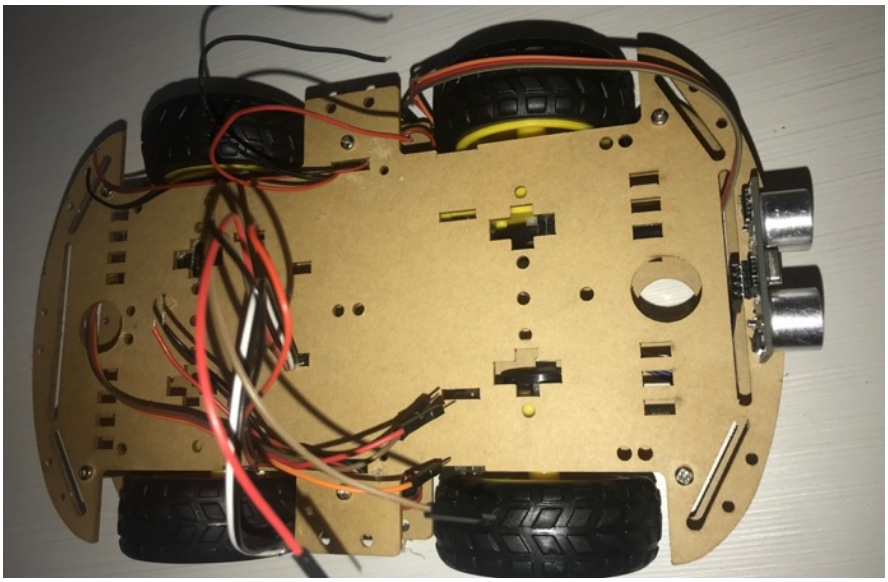

Figura 27. Resultado de la conexión entre chasis superior e inferior.

El siguiente paso en el montaje es la colocación y fijación del controlador Arduino Mega al cual está conectado el Arduino Shield L293D. Al igual que en los pasos anteriores se realiza mediante una pistola de silicona.

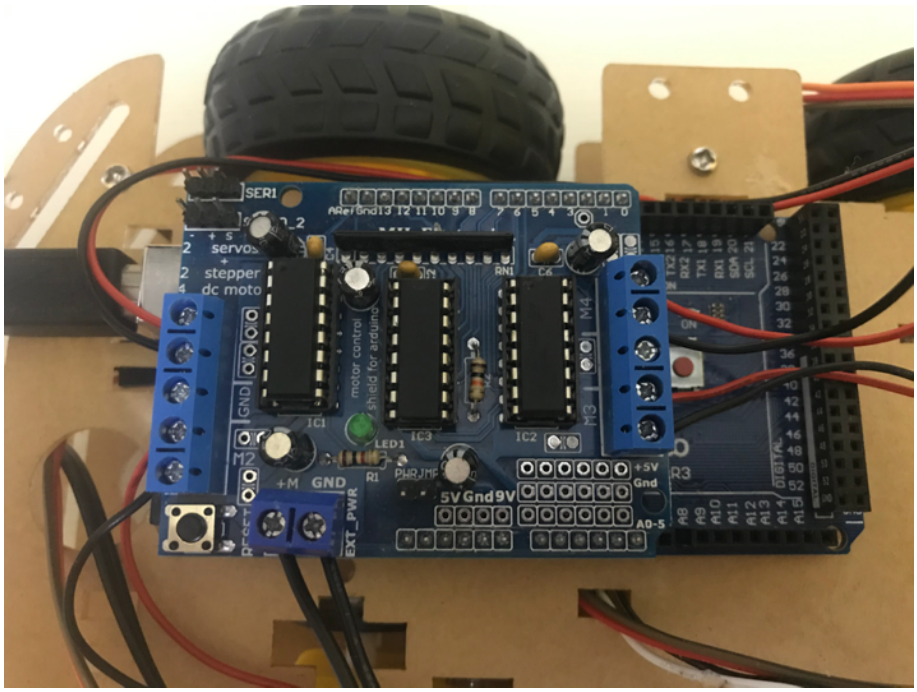

Figura 28. Conexiones del Arduino Shield L293D en detalle.

Como se puede apreciar en la Fig. 28. anterior también se ha realizado las conexiones de los motores con el Arduino Shield L293D de tal modo que el motor delantero derecho será M4, el motor delantero izquierdo M3, el motor trasero derecho M2 y el motor trasero izquierdo M1. También se ha conectado la fuente de alimentación.

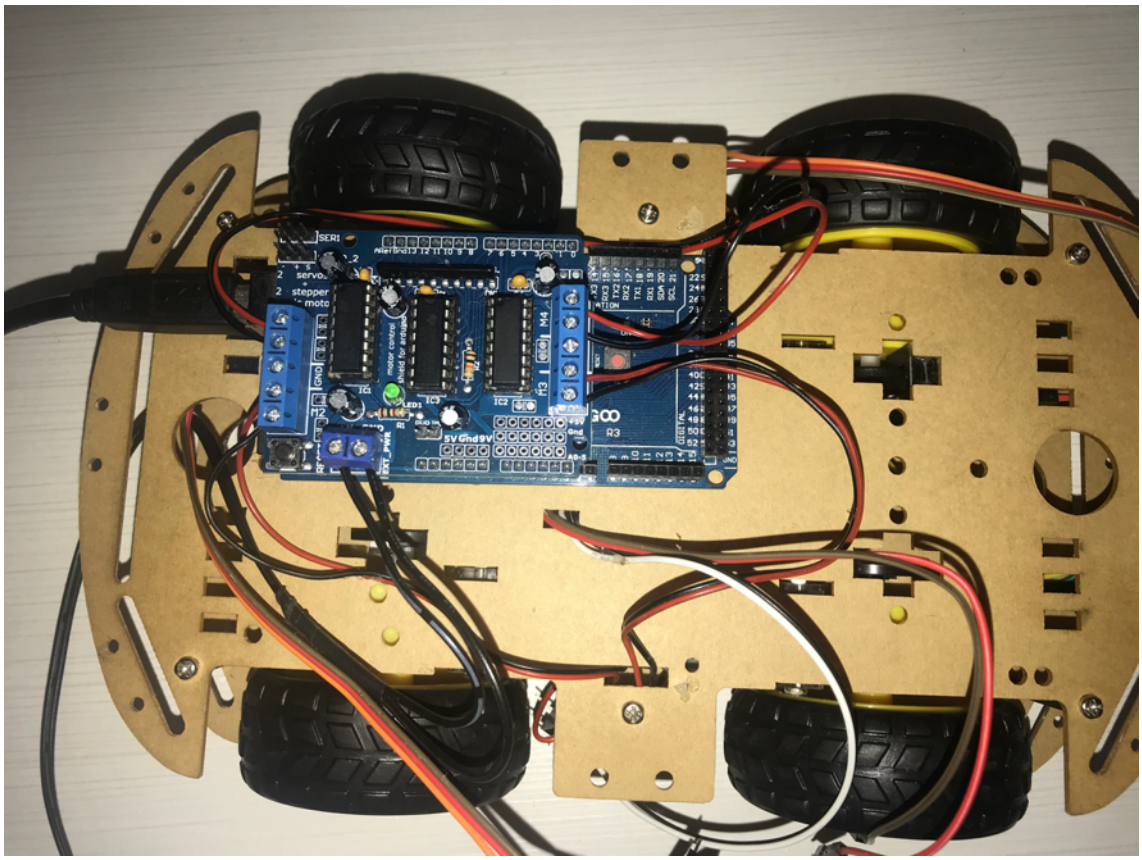

Figura 29.Conexiones del Arduino Shield L293D.

En el último paso del proceso de montaje se ha realizado las conexiones de los distintos sensores con el Arduino Mega. Para este proceso ha sido necesario utilizar un pequeño breadborder, en la cual hemos elegido dos líneas en las cuales colocaremos la tensión de 5 V saliente del Arduino y la tierra también conectada al Arduino para así poder conectarlos varios sensores a la vez, ya que el Arduino Mega solo le quedan dos pines para fuente de tensión a 5 V y dos pines de tierra.

El primer sensor que debe conectarse será el de ultrasonido a través de dos cables correspondientes al breadborder para obtener la energía para su funcionamiento. Por otro lado, se ha conectado sus dos cables de emisión y recepción de datos al Arduino siendo así que el sensor receptor de la onda de ultrasonido se conecta al pin 44 y el sensor de emisión de la onda de ultrasonidos al pin 42. Por último, se conecta los sensores FC-33 infrarrojo fotoeléctricos, este sensor tiene tres cables dos de ellos corresponde con la tierra y el positivo por los cuales se va a recibir la energía necesaria para su funcionamiento. El último cable es mediante el cual se realizará el envío de datos al Arduino Mega, siendo así que los dos sensores se conectan a los pines 21 y 22. Cabe destacar que para cada uno de los sensores ha sido necesario la colocación de un condensador de 100 nF para un correcto funcionamiento. Debido a que si no se conecta este condensador no se obtiene un pulso cuadrado perfecto que es lo que necesitamos para poder hacer la medición, sino que se recibe un pulso que se parece a uno cuadrado, pero con rebotes al final de este lo que hace que se obtenga un valor erróneo de la medición.

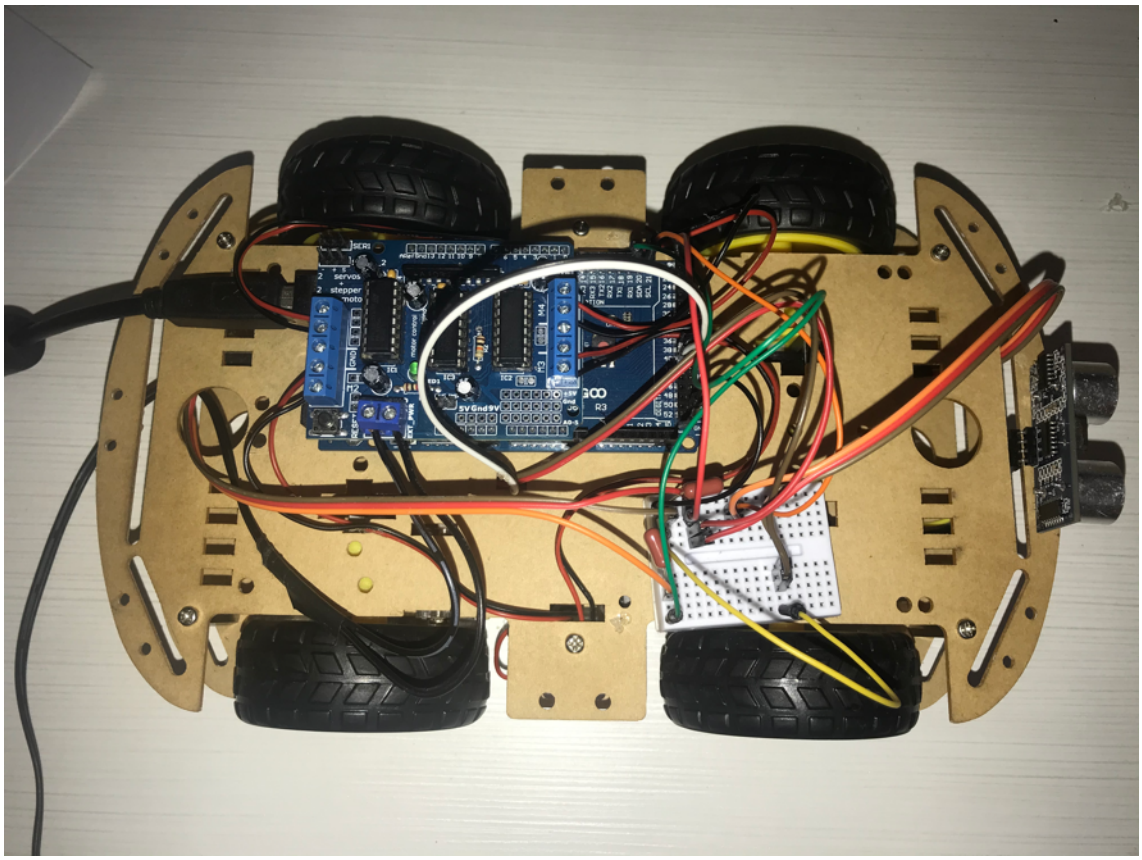

Figura 30. Resultado final del proceso de montaje.

4. MODELIZACIÓN DE PLANTA

4.1 Modelización teórica del sistema

El modelo del sistema se obtiene como dos subsistemas conectados en serie. Un primer subsistema estará formado por el circuito del motor y por la rueda/eje. El segundo subsistema contiene las leyes de Newton que relacionan todas las fuerzas existentes con la aceleración.

Sistema del Motor.

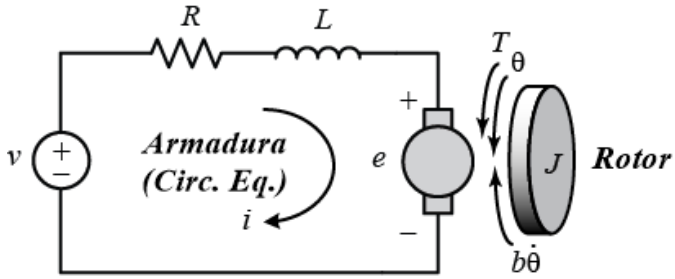

Figura 31. Sistema del Motor

Subsistema eléctrico del motor con el cual será alimentado a un de los motores.

$$w = \frac{dx}{dt}\frac{1}{r} \quad [10]$$

$$V = Ra * I + E_a \quad [11]$$

$$E_a = K_2 * w \quad [12]$$

$$V = Ra * I + K_2 * w + La\ \frac{dI}{dt} \quad [13]$$

$$V(s) = Ra * I(s) + K_2 * w(s) + La * s * I(s) \quad [14]$$

Subsistema mecánico del motor:

$$T_m = K_1 I \quad [15]$$

$$J\frac{dw}{dt} + Bw += T_m \quad [16]$$

$$Jws + Bw = T_m \quad [17]$$

$$w = \frac{T_m}{(Js + B)} \quad [18]$$

Sistema del Motor en su conjunto:

$$V(s) = Ra * \frac{T_m}{K_1} + K_2 * \frac{T_m}{(Js + B)} + Ls * \frac{T_m}{K_1} \quad [19]$$

$$V(s) = \left(\frac{Ra}{K_1} + \frac{K_2}{(Js + B)} + \frac{Ls}{K_1}\right) T_m \quad [20]$$

$$V(s) = \left(\left(\frac{Ra + Ls}{K_1}\right) + \left(\frac{K_2}{(Js + B)}\right)\right) T_m \quad [21]$$

$$V(s) = \left(\frac{(Ra + Ls) * (Js + B) + K_1 * K_2}{K_1 * (Js + B)}\right) T_m \quad [22]$$

$$\frac{T_m}{V(s)} = \left(\frac{K_1(Js + B)}{(Ra + Ls) * (Js + B) + K_2K_1}\right) \quad [23]$$

$$\frac{F * r_{ext}}{V(s)} = \left(\frac{K_1(Js + B)}{LJs^2 + (RaJ + LB)s + RaB + K_2K_1}\right) \quad [24]$$

$$\frac{F}{V(s)} = \left(\frac{K_1(Js + B)}{LJs^2 + (RaJ + LB)s + RaB + K_2K_1}\right) * \frac{1}{r_{ext}} \quad [25]$$

La expresión de arriba se corresponde con la función de transferencia según la cual se obtendrá cuantos Newtons de fuerza aporta cada uno de los motores para poder mover el vehículo. Al tener 4 ruedas y cuatro motores, los Newtons totales obtenidos será multiplicar la anterior función de transferencia por cuatro,

Sistema Físico del vehículo.

Sobre el vehículo se le están aplicando la gravitacional (N,P). Por otro lado, la Fm será la fuerza aprovechada para el movimiento del vehículo. En el presente trabajo únicamente estudiaremos el caso en el cual el vehículo está en un plano horizontal.

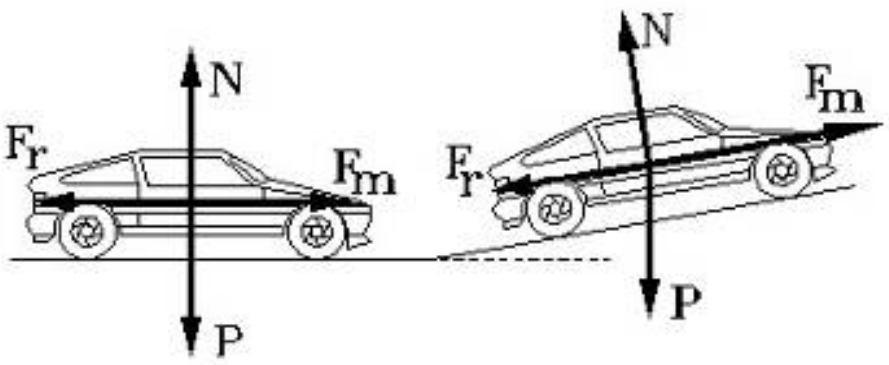

Figura 32. Fuerzas aplicadas al coche

Trasladamos todas estas fuerzas a la rueda.

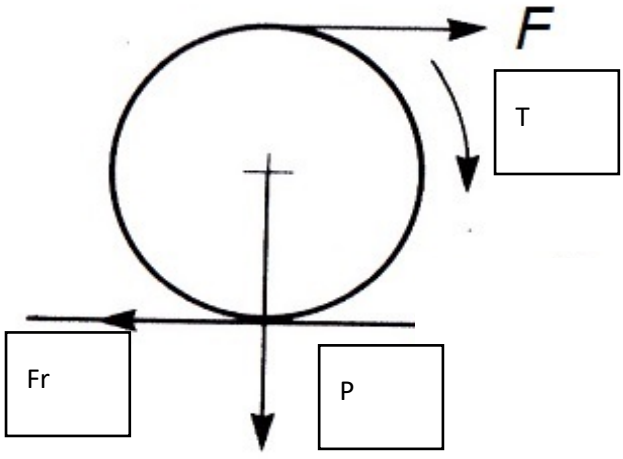

Figura 33. Fuerzas aplicadas a la rueda

$$T = F * r_{ext} = \frac{m}{4} * a * r_{ext} \quad [26]$$

$$a = \frac{dv}{dt} \quad [27]$$

$$F = \frac{m}{4} * \frac{dv}{dt} \ [28]$$

$$\frac{v}{F} = \frac{1}{\frac{m}{4} * s} \ [29]$$

4.2 Determinación de los parámetros.

Una vez obtenido la ecuación de Laplace que rige el sistema es hora de ir sustituyendo las variables que hay en el modelo para obtener el modelo matemático.

Ra: Es la resistencia interna del motor que mediante un voltímetro hemos medido que es de 6 Ω.

$\boldsymbol{r_{ext}}$: Corresponde al radio de la rueda que midiendo con regla obtenemos que es de 32 mm.

$\boldsymbol{r_{int}}$: Corresponde al radio de la rueda que midiendo con regla obtenemos que es de 12 mm.

m: Es la masa total de todo el dispositivo siendo que pesa 585 g.

mrueda: Es la masa de una de las ruedas del dispositivo siendo que pesa 28 g

g: Es la constante de la aceleración de la gravedad que es de 9.81 m/s^2.

J: Corresponde a la inercia que presenta el motor considerando la inercia del eje y la de la rueda. Despreciaremos la inercia del eje, ya que es mucho menor a la inercia que presenta la rueda:

$$J = \frac{1}{2} * mrueda * \left(r_{ext}^2 + r_{int}^2\right) = 0.000016352\, kg/m^2 \quad [30]$$

$\boldsymbol{K_2}$: La constante electromotriz surge del hecho de cuando un motor eléctrico se encuentra en rotación se genera una tensión proporcional a esta. Es decir que la velocidad a la cual gira el motor será proporcional a la tensión con una constante K. Para la obtención de esta K se ha realizado un experimento en el cual giramos el motor a diferentes voltajes y midiendo la velocidad angular de este. Cabe destacar que se ha realizado este experimento sin haber colocado antes la rueda al motor, ya que el peso y la inercia de esta nos daría unas velocidades menores

Tabla 1. Valores de K_2 para distintos parámetros de Voltaje y RPM.

E_a (Volts)	N(rpm)	K_2
4,04	288	0,0140277
3,44	240	0,014333
3,26	234	0,014298
3,13	222	0.014099

$$K_2 = \frac{E_a}{n} \quad [31]$$

En la tabla 1 podemos ver distintas mediciones del número de RPM del eje a distintos voltajes los cuales han sido medidos con un voltímetro. En la tercera columna se encuentra el resultado de haber aplicado a los anteriores valores la Ecuación [31].

Ponderamos los 4 valores obteniendo así $\boldsymbol{K_2}$**=0,0142**

$\boldsymbol{K_1}$: Es la constante de par entrega en la cual se está relacionando directamente el flujo en el entrehierro que es proporcional con la intensidad con el par obtenido. Por lo que la K_1 relaciona directamente el par obtenido con la intensidad proporcionada. Hay que remarcar que K_1 y K_2 tienen el mismo valor.

$$K_1 = \frac{T_m}{i} \quad [32]$$

$\boldsymbol{K_1}$=0,0142

B: Es la constante de fricción de Coulomb. Esta constante establece la fuerza de rozamiento que se ejerce sobre el eje que se opone al movimiento.

Partiendo de la ecuación [6]:

$$J\frac{dw}{dt} + Bw = T_m \quad [33]$$

Sustituimos T_m mediante la ecuación [5] obteniendo así:

$$J\frac{dw}{dt} + Bw = \frac{i}{K_1} \quad [34]$$

De esta ecuación [18] se desconoce B pero mediante el método de regresión lineal seremos capaces de obtener B. Para poder realizar este método primero se obtiene los datos experimentales, los cuales los obtendremos mediante un experimento en el cual ponemos en funcionamiento el motor de corriente continua a distintitos voltajes y midiendo la intensidad y velocidad cuando esta se haya estabilizado. Al medir a velocidad constante ("estable"). De la ecuación [18] al no haber variación en la velocidad angular $\frac{dw}{dt} = 0$. Así mismo añadimos el concepto de par de arranque para que la ecuación se ajuste a la gráfica:

$$Bw + Tf = Tm = i * K_1 \quad [35]$$

Tabla 2. Par Revoluciones primeros datos

$Tm = i * K_1$	N(rpm)
0,001136	81
0,001562	96
0,001704	108
0,003266	129
0,00355	141
0,004118	168

Tabla 3. Par revoluciones segundos datos

$Tm = i * K_1$	N(rpm)
0,001136	84
0,001562	92
0,001704	102
0,003266	126
0,00355	144
0,004118	165

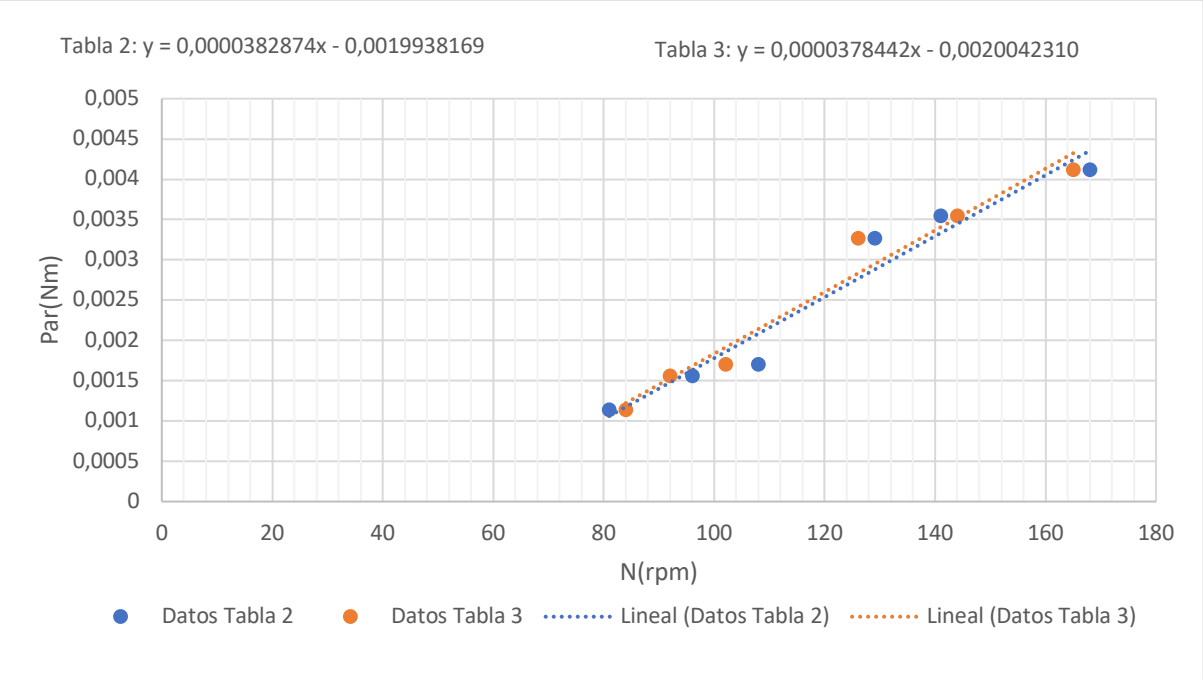

Figura 34. Gráfico relación par con revoluciones

En la Fig.34 se representan los datos de las tablas 2 y 3. Para cada conjunto de datos se ha creado una línea de tendencia lineal con su respectiva ecuación. Y como se puede apreciar son correctos, ya que ambas rectas obtenidas de los datos de la tabla 2 y 3 son casi coincidentes. Obtenemos las pendientes de cada recta de las cuales se ha realizado la media entre ambas para obtener el valor B.

$$B = \frac{0{,}0000382874+0{,}0000378442}{2} = 0.0000380658 \text{ Nms} \quad [36]$$

4.3 Función de Transferencia.

La modelización del vehículo sería la siguiente da como resultado el siguiente sistema lineal.

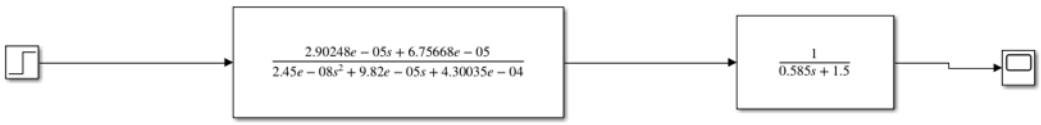

Figura 35. Esquema del sistema lineal.

El diagrama de bloques correspondiente al modelo lineal del vehículo se representa en la Fig. 35. Se puede apreciar dos sistemas lineales agrupados en serie, donde la función de transferencia del primer bloque representa la relación entre la entrada en voltios y la fuerza del motor expresada en Newtons:

$$\frac{F}{V(s)} = \left(\frac{2{,}90248e{-}05s{+}6{,}75668e{-}05}{2{,}45e{-}08s^2{+}9{,}82e{-}05s{+}4{,}30035e{-}04}\right) \qquad [37]$$

El segundo bloque representa el sistema físico del vehículo, donde la entrada es la fuerza en Newtons y la salida es la velocidad del vehículo:

$$\frac{v}{F} = \left(\frac{1}{0{,}585s{+}1{,}5}\right) \qquad [38]$$

En la ecuación [37] se ha añadido un término que ha sido calculada gracias a los datos obtenidos experimentalmente. Se ha obtenido un valor del parámetro que ofrezca una razonable aproximación a los datos reales. El término calculado empíricamente corresponde a las fuerzas no consideradas hasta el momento como es la fuerza de rozamiento.

Podemos apreciar que la salida del primer sistema lineal corresponderá a la entrada del segundo sistema lineal por eso que al juntar estos dos sistemas lineal independientes obtenemos el sistema lineal de todo el proceso como hemos visto anteriormente.

Una vez que obtenida la función de transferencia comprobamos si se ajusta la función con el funcionamiento real del vehículo.

$$G(s) = \frac{v}{V} = \left(\frac{2{,}90248e{-}05s{+}6{,}75668e{-}05}{1{,}435\text{e}{-}08s^3{+}5{,}747\text{e}{-}05s^2{+}3{,}988\text{e}{-}04s{+}6{,}451\text{e}{-}04}\right) \qquad [39]$$

Para la validación del modelo experimental se ha realizado la obtención de datos del modelo real ante un escalón de 9 V. Se ha podido realizar esta obtención de datos gracias a la instalación de dos sensores FC-33 infrarrojo fotoeléctricos las cuales pueden leer un encoder que está sujeto al eje del motor. Por otro lado, se ha realizado la simulación mediante SimuLink de la respuesta de la velocidad en lazo abierto ante un escalón de amplitud 9 V también. El resultado se pude visualizar en la siguiente Fig.36.

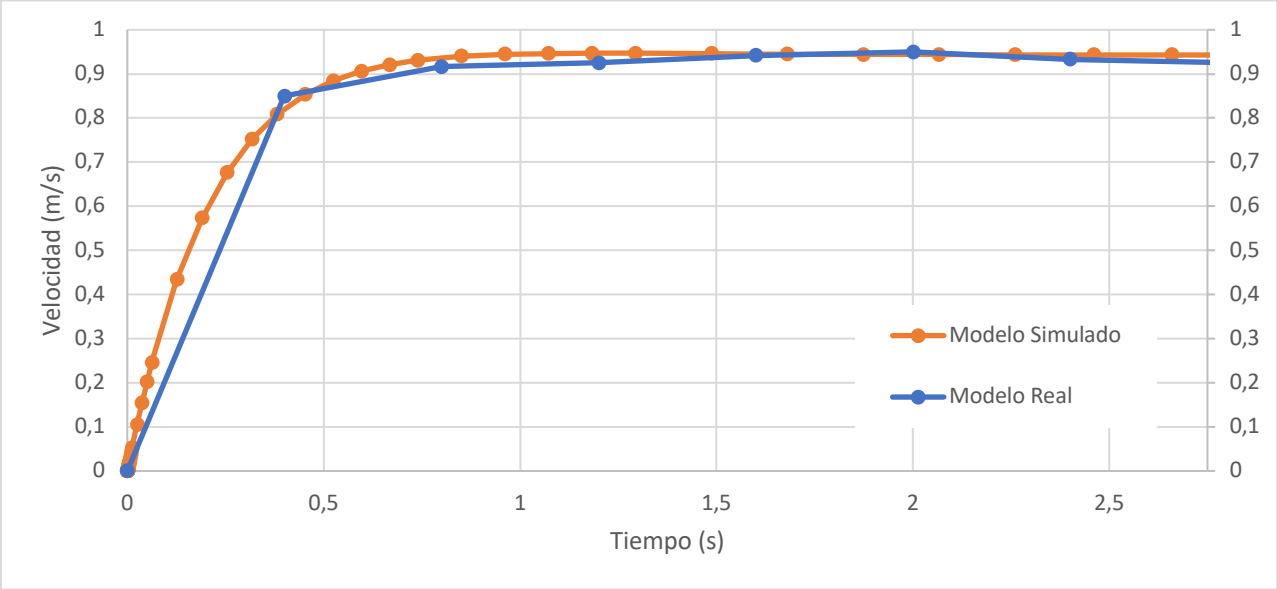

Figura 36.Respuesta de la velocidad en lazo abierto del modelo real y simulado.

Observando la comparación entre las curvas de la Fig. 36 , donde se ha representado la velocidad obtenidas del modelo simulado con una curva naranja y la velocidad obtenida del modelo real con una curva azul. Se puede apreciar que el comportamiento del modelo y del sistema real son prácticamente similares con un pequeño error debido principalmente a la dinámica no modelada e incertidumbre paramétrica.

Para reafirmar el modelo se realiza la medición de la posición gracias al sensor de ultrasonidos y compararlo con el modelo simulado que se ha modelizado.

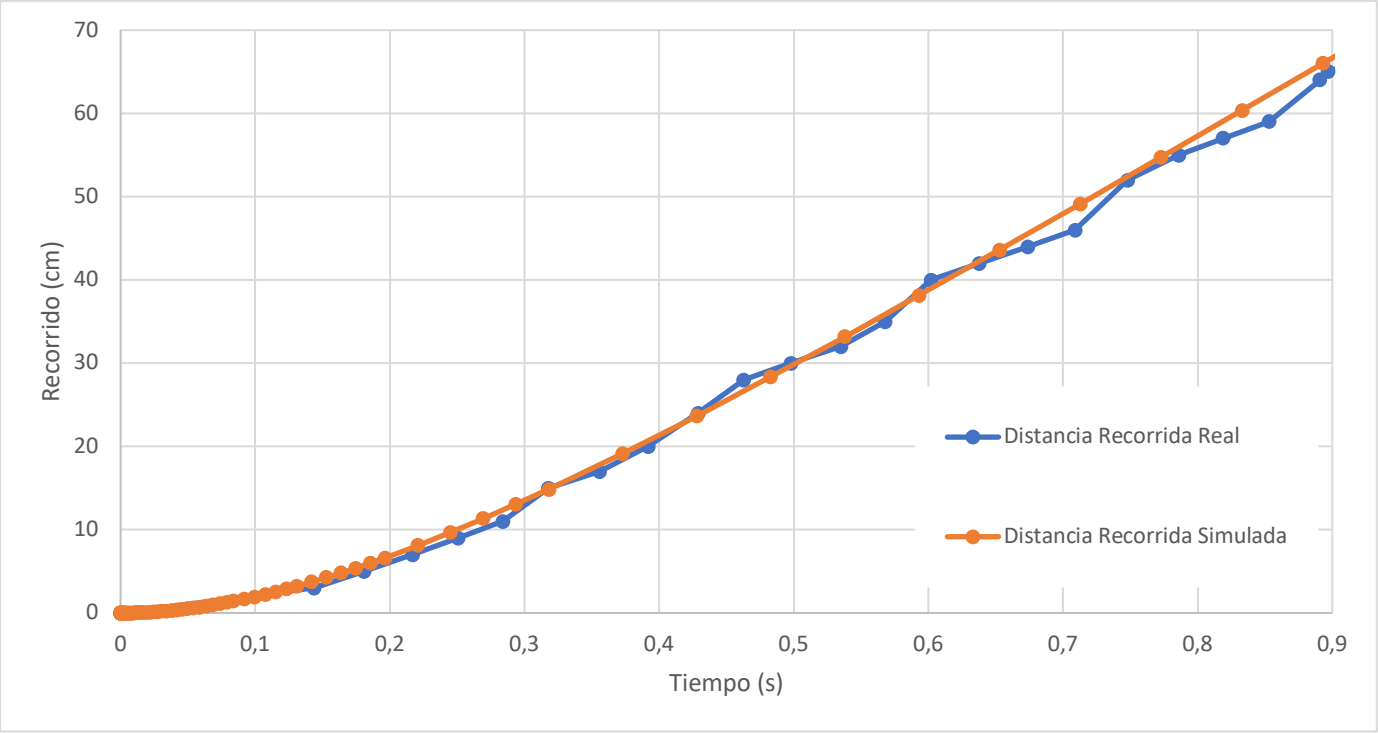

Figura 37.Respuesta de la posición en lazo abierto del modelo real y simulado

En la Fig. 37 se ha representado la curva en naranja la respuesta simulada de la posición del vehículo ante un escalón de 9 V. Mientras que la curva azul representa los datos obtenidos del vehículo al aplicar una tensión de 9 V a cada uno de sus motores. Como se puede apreciar al igual que en la respuesta de la velocidad obtenemos dos curvas que son muy similares, aunque con un pequeño error debido principalmente a la dinámica no modelada e incertidumbre paramétrica las cuales afectaban igualmente en la respuesta de la velocidad.

Se puede afirmar que a partir de la comparación del modelo real y el simulado de tanto la velocidad como la posición, el modelo de planta realizado en el aparatado anterior es correcto, ya que el modelo simulado tiende a explicar el comportamiento real del vehículo.

Una vez que se ha comprobado la fiabilidad de la función de transferencia obtenida del modelado de la planta, se realiza el cálculo de los polos y los ceros de la función de transferencia obteniendo la siguiente representación del lugar de las raíces.

Tabla 4. Polos y ceros de la función de transferencia de Planta.

Polos	-3997.92 -2.57 -4.38
Ceros	-2.3279

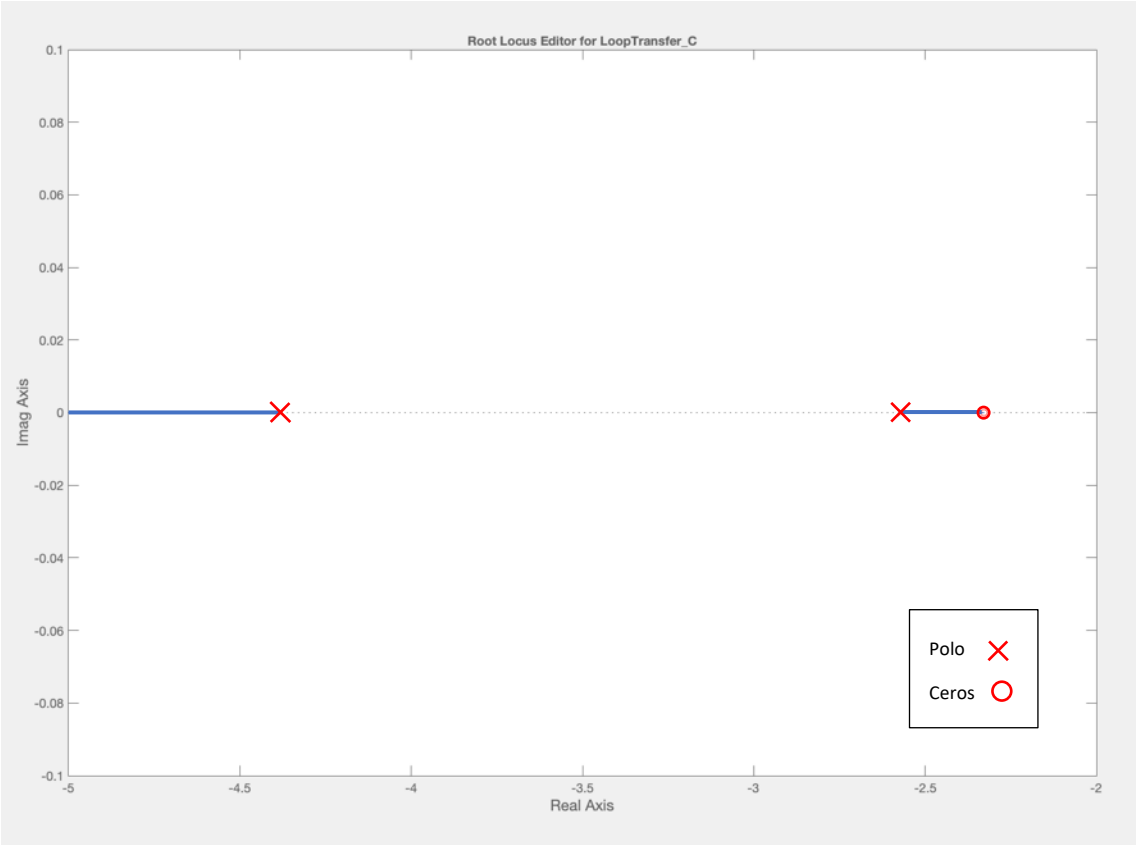

Figura 38.Representación en el plano de ceros y polos en bucle abierto.

Al observar los datos de la Fig. 38. se extrae las siguientes conclusiones. El polo -3997,92 al estar muy a la izquierda respecto a los otros polos dominantes este se puede despreciar pues no afecta prácticamente nada a la función. Este polo es tan grande debido a la existencia de una inductancia muy pequeña en el sistema eléctrico que no afecta por lo que podríamos haberlo despreciado desde un primer momento.

Una vez colocado los polos y ceros en el plano es hora de indicar las especificaciones dinámicas y representarlas en el plano.

Como especificaciones de funcionamiento de nuestro vehículo he decidido que el sistema no tenga una sobreoscilación mayor a al 4.3% lo que es el caso particular. Por otro lado, el tiempo de establecimiento hemos decidido que sea menor a 0,5 segundo, por lo tanto:

$$t_e = \frac{4}{\sigma} \qquad [40]$$

$$\sigma = \frac{4}{t_e} = \frac{4}{0.5} = 8 \; [41]$$

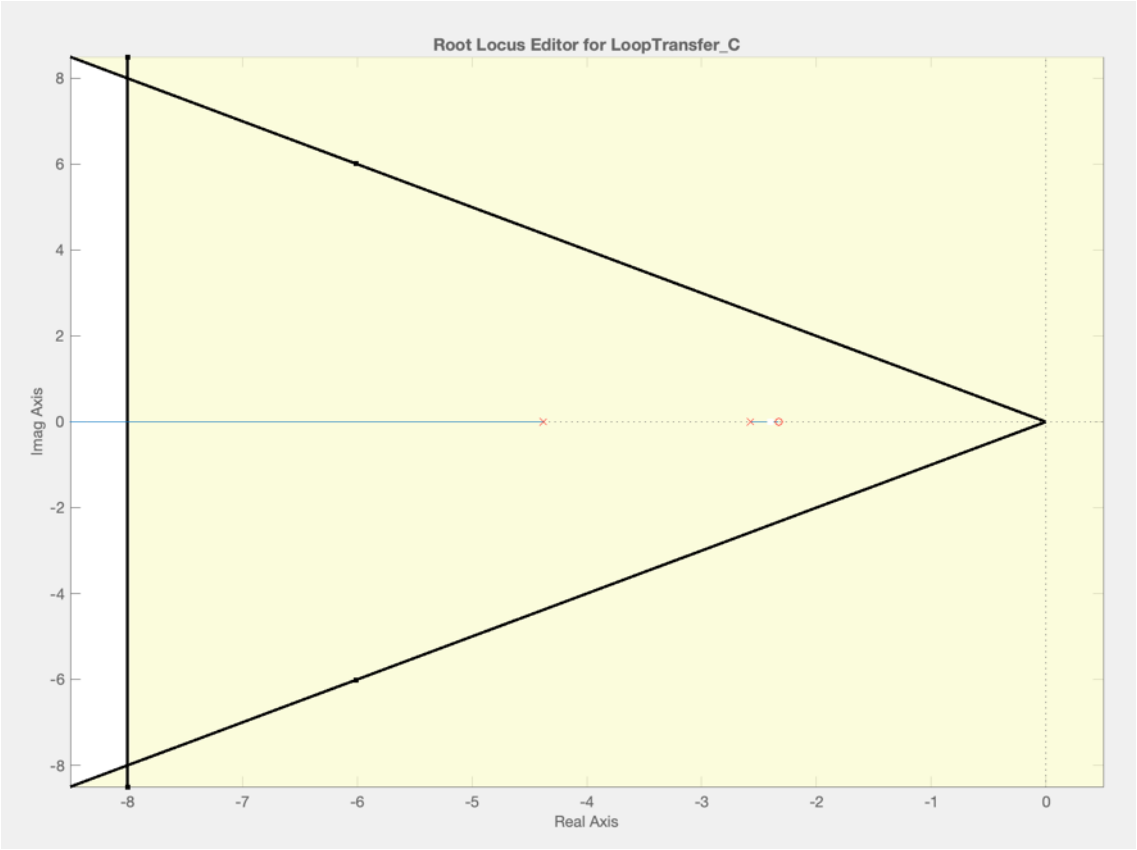

Figura 39. Representación de especificaciones dinámicas en bucle abierto.

En la gráfica anterior se ha representado las restricciones dinámicas. Estas restricciones están representadas mediante una línea negra la cual se considera que es el límite de las restricciones mientras que la zona en amarillo es la zona en la cual no se cumple una o ninguna de las restricciones. Como apreciamos nuestro polo o cumple con las especificaciones dinámicas. Por lo que estamos obligados a aplicar un PID para que se ajuste a nuestro deseo.

4.4 Valores del PID.

El controlador que diseñaremos será del tipo PI debido a que el error de posición ante referencia como ante perturbación de 0. La acción Integral es la que nos proporciona el error en el estado estacionario cero. Por lo que en un principio consideramos en utilizar un PI o un PID. Se ha desechado la idea de utilizar el controlador PID porque no interesa la acción derivada, ya que no se desea ningún cambio brusco y obtendremos el controlador más sencillo posible.

Al utilizar un controlador PI existe un polo en 0 y un cero el cual se deberá ajustar mediante la técnica del argumento:

$$Gr = kc + \frac{kc}{Ti \cdot s} = kc\frac{\left(s + 1/Ti\right)}{s} \quad [42]$$

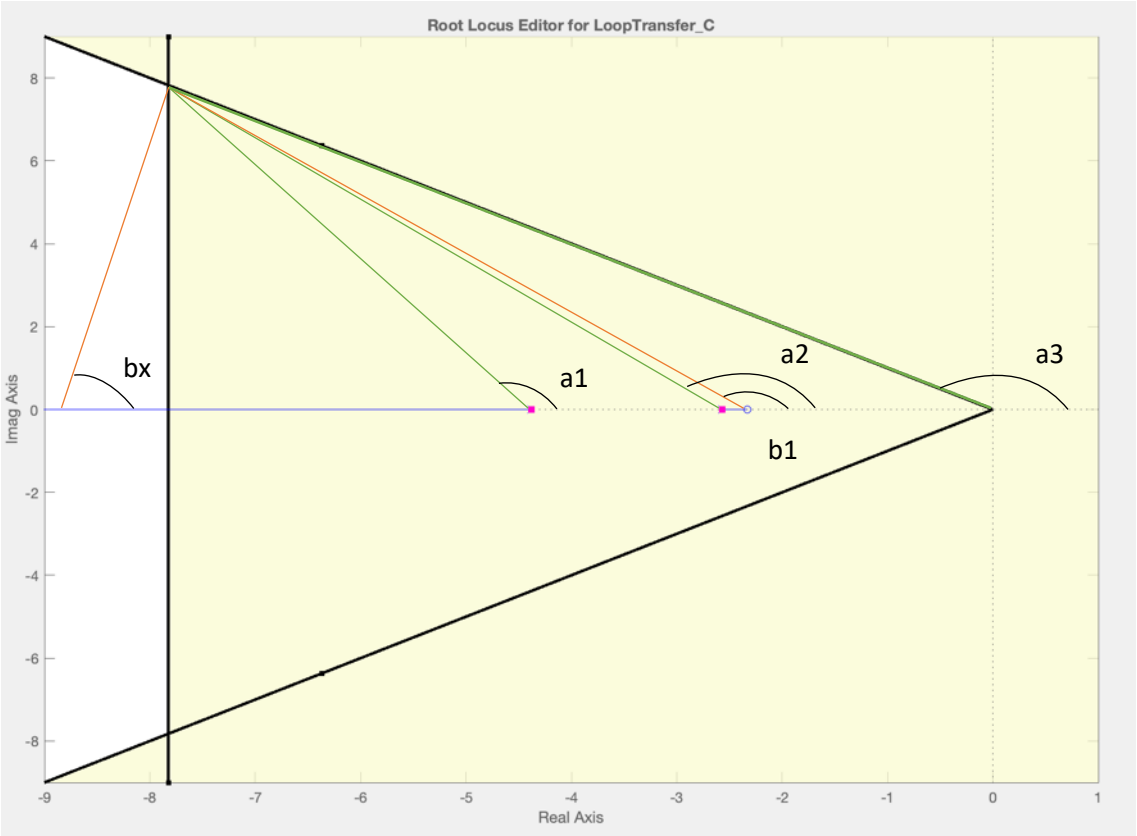

Figura 40. Gráfica con ángulos de los ceros y polos en bucle abierto.

$$a1 + a2 + a3 = b1 + bx + 180 \quad [43]$$

$$a3 = 45° + 90° = 135° \quad [44]$$

$$a2 = 180 - arctg\left(\frac{8}{2{,}57}\right) = 107{,}809° \quad [45]$$

$$a1 = 180 - arctg\left(\frac{8}{4.38}\right) = 118{,}7° \quad [46]$$

$$b1 = 180 - arctg\left(\frac{8}{2{,}3279}\right) = 106{,}22° \quad [47]$$

$$118{,}7° + 107{,}809° + 135° = 106{,}22° + 180° + bx \quad [48]$$

bx=75,289°

$$arctg\left(\frac{8}{cerox'}\right) = 75{,}289° \quad [49]$$

cerox`=2,2

cerox=cerox'+8 [50]

cerox =10,2

Con el criterio del argumento se ha logrado colocar el cero del PI en una posición tal que la recta formada por los ceros y polos tanto del controlador como del sistema pasen por los puntos límites en los cuales se cumplen las especificaciones dinámicas. Siendo estos puntos el $-8 \pm 8j$.

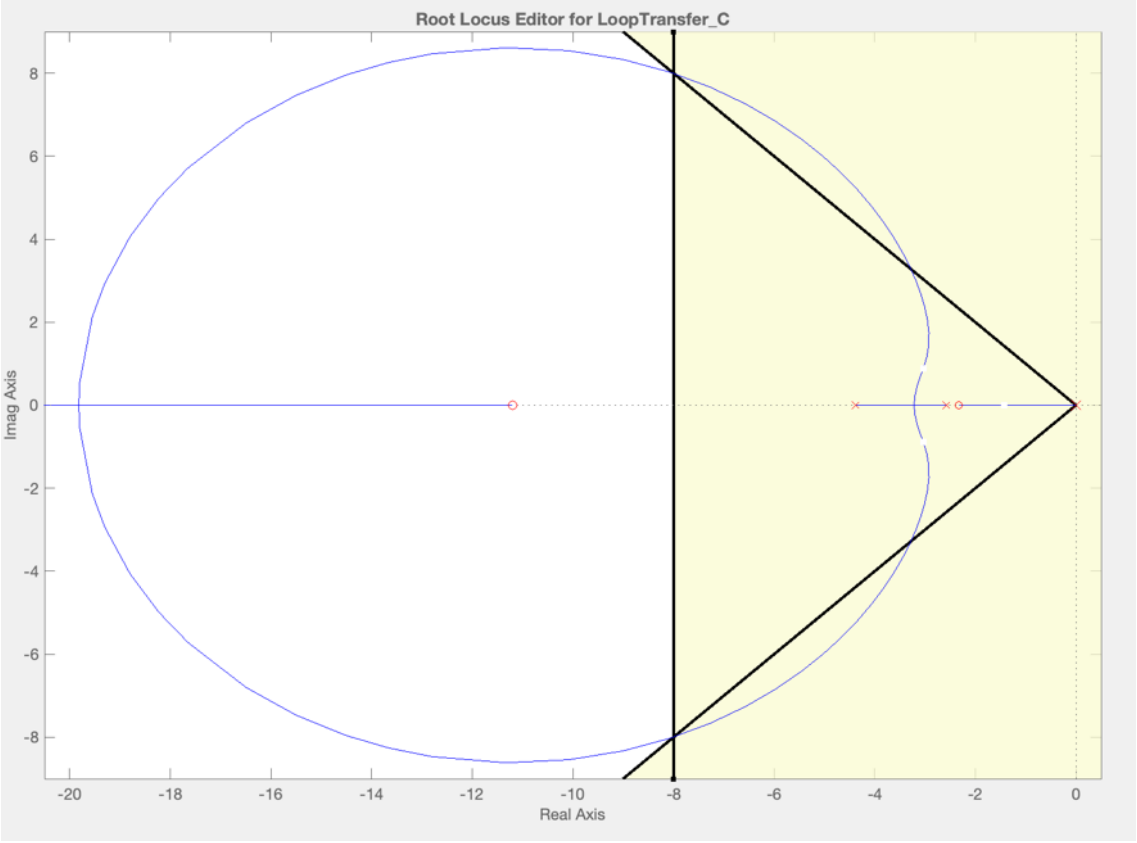

Figura 41. Gráfica forma de los polos y ceros en bucle abierto

Una vez calculado el cero del PI faltaría ajustar la K del proporcional a los puntos de funcionamiento para los límites de las especificaciones dinámicas las cuales corresponden con la intersección de los límites de ambas restricciones con la curva del lugar de las raíces en azul.

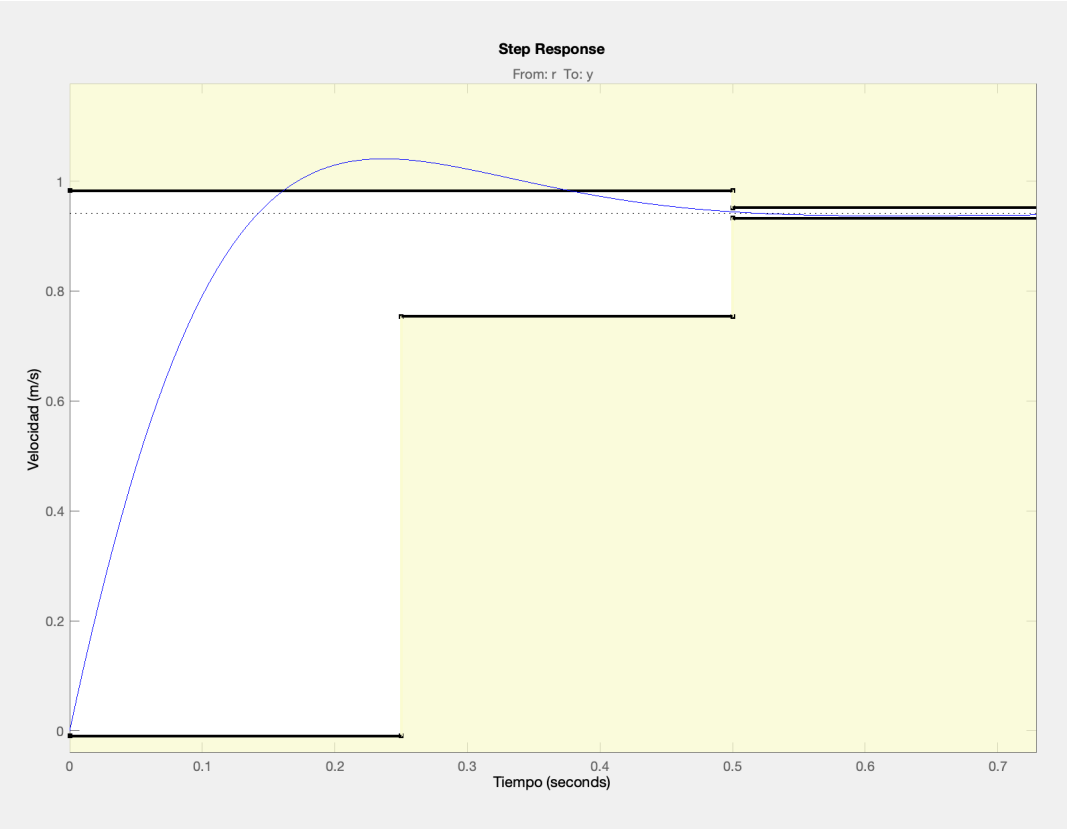

Figura 42. Gráfica respuesta ante escalón en bucle abierto simulado.

En la gráfica de la Fig. 41. se ha representado la respuesta cuando se realiza el control para obtener una velocidad de 0.942 m/s en bucle cerrado con el punto de funcionamiento en los límites de las especificaciones dinámicas. Las líneas horizontales corresponden a los límites dinámicos de la sobreoscilación mientras que la zona amarilla es la zona donde se incumple esta sobreoscilación. Al ajustarlo manualmente en los puntos límites para obtener un valor aproximado observamos que la respuesta ante un escalón no cumple con la especificación de sobre oscilación, ya que la respuesta ante escalón invade la zona amarilla.

Debido a esta razón, aunque se calcule de modo teórico la ganancia (acción proporcional) con la técnica de módulos no se obtendrá un controlador que se ajuste a las especificaciones indicadas.

Aunque se haya ajustado los polos dominantes para trabajar en los límites de las especificaciones dinámicas no se cumple la restricción debido a la existencia de un cero muy próximo a los polos dominantes lo que genera una mayor sobreoscilación. Por lo tanto, se deberá ir reduciendo la ganancia y el cero, pero debemos tener cuidado de no incumplir con la especificación de error=0. Es por eso por lo que se ajustara manualmente para lograr un consenso entre la sobreoscilación y el error. Una posible solución donde cumple con las especificaciones es:

$$PI = \frac{17{,}517(S+7{,}445)}{S} \qquad [51]$$

$$PI = 130{,}41\,\frac{(1+0{,}13s)}{S} \qquad [52]$$

$$Kp = 130{,}41 \qquad Ki = \frac{1}{Ti} = \frac{1}{7{,}4} = 0{,}13 \qquad Kd = 0$$

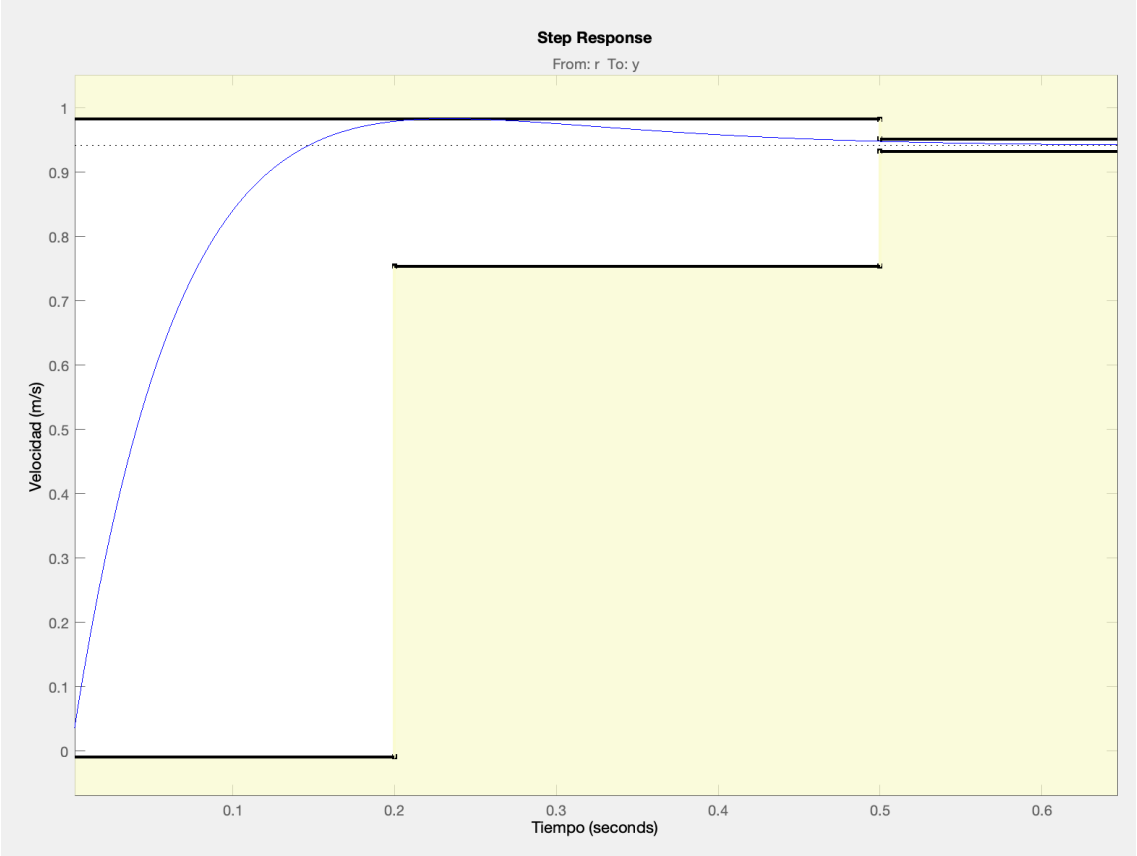

Figura 43. Respuesta ante escalón en bucle abierto simulado.

En la Fig.43 se representa la respuesta cuando se realiza el control con el nuevo controlador para una velocidad de 0.942 m/s en bucle cerrado. En este caso la respuesta ante escalón en ningún momento entra en la zona amarilla en la cual se incumpliría la sobreoscilación deseada. Por lo tanto, la respuesta cumple la sobreoscilación del 4,3% así como un tiempo de establecimiento de 0,5 s. Por otro lado, si miramos en la gráfica ante error podemos apreciar que el error ante una perturbación está dentro de las especificaciones que hemos indicado.

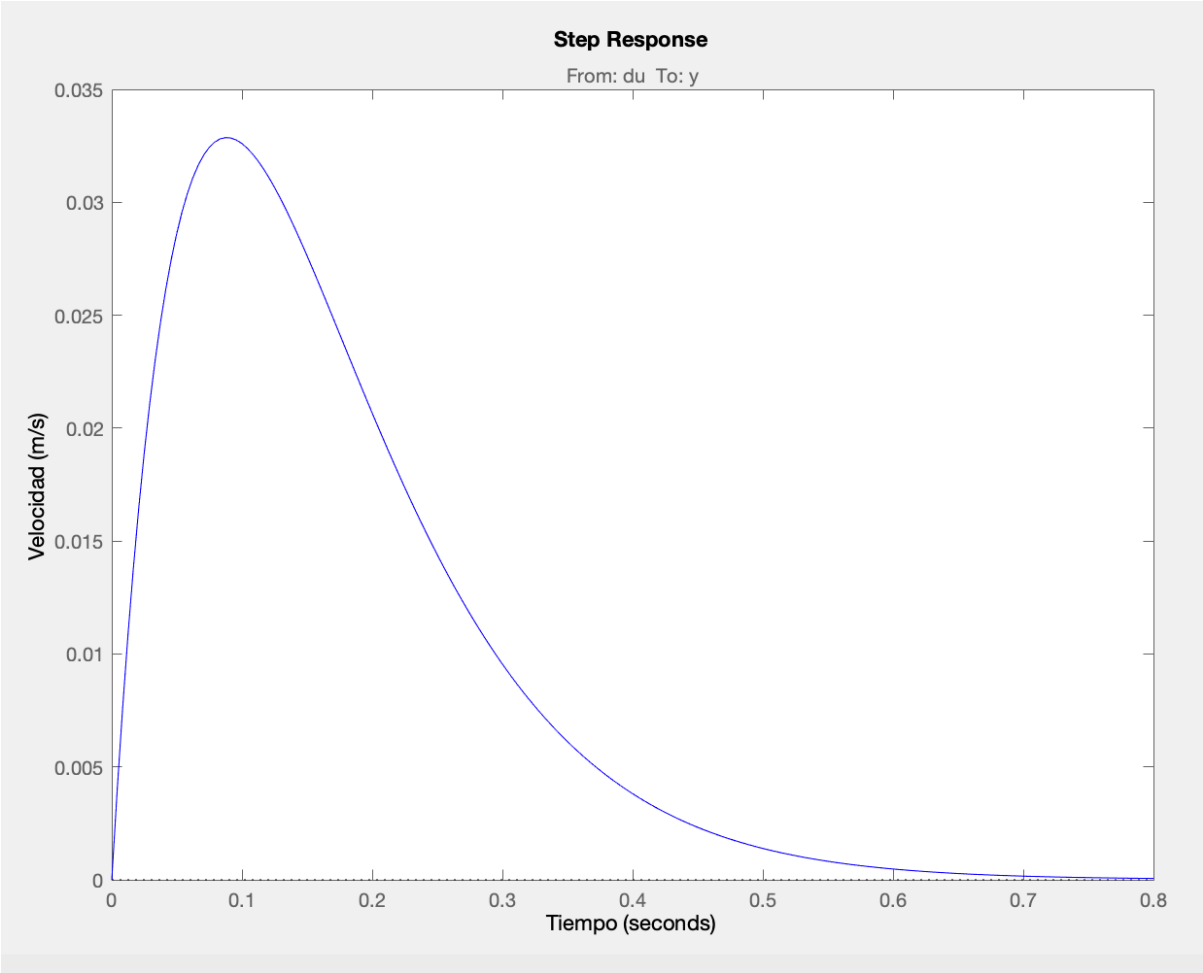

Figura 44. Gráfica del error en bucle lazo abierto simulado.

Como se puede apreciar en la Fig. 44. mediante el PID anteriormente se obtiene un error de posición igual a 0 lo cual es lógico por estar utilizando un PI.

5. PROGRAMACIÓN.

5.1 Software.

Se ha utilizado el programa Arduino para la programación del controlador Arduino Mega con el objetivo de controlar la distancia entre el vehículo y el obstáculo. El programa Arduino está basado en el lenguaje de programación C++.

El programa consta básicamente de tres partes bien diferenciadas: IDE (Entorno de Desarrollo), Serial Monitor y Consola de Error.

La primera de las partes es el IDE siendo esta la parte que más ocupa de la interfaz del programa. En este entorno de desarrollo es la herramienta mediante la cual programamos escribiendo las instrucciones.

Mediante la consola de errores existe una comunicación directa entre el programa y el programador , en donde se indican los errores que aparecen al compilar o cargar cualquier programa. Así mismo en esta consola de errores aparte de indicarnos los errores también nos indican cuando se ha compilado y/o cargado exitosamente el programa.

Por último, tenemos el Serial Monitor. Esta parte de la aplicación es donde se mostraría los datos e información del programa. También mediante este Serial Monitor se puede solicitar cualquier dato que será utilizado por el programa. Haciendo un símil con un ordenador, esta parte del programa correspondería a lo que es actualmente a la consola de comandos de cualquier ordenador.

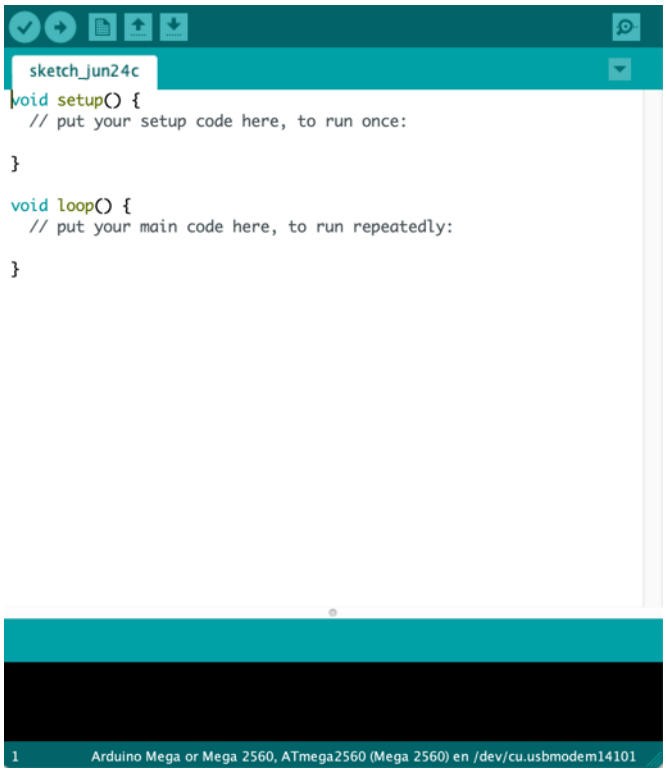

Figura 45. Interfaz del programa Arduino.

5.2 Librerías.

Cualquier programa que estemos programando tanto en C++ como en otros lenguajes de programación existen librerías los cuales facilitan la programación por parte del programador. Las librerías son un conjunto de códigos e instrucciones pre-programadas con anterioridad. Con estas librerías podemos introducir funciones en nuestro programa sin la necesidad de programar cada una de las funciones, ya que con el programa de Arduino únicamente podemos crear funciones básicas. Por ejemplo, si necesitamos hacer cálculos trigonométricos si no añadiésemos ninguna librería tendríamos que introducir y programas cada una de las ecuaciones trigonométricas. Al introducir la librería de trigonometría ya están creadas todas las ecuaciones en el lenguaje de programación y únicamente tenemos que hacer una llamada a la librería.

Existen infinidad de librerías creadas por distintos usuarios en internet las cuales podemos descargar gratuitamente. El archivo descargado deberá ser un archivo tipo ZIP.

Para introducir las de diferentes librerías en el Arduino hay que ir a la siguiente ubicación:

Programa/Incluir Librería/Añadir biblioteca .ZIP.

Una ve aquí se nos abrirá un seleccionador donde tenemos que buscar y seleccionar el archivo ZIP descargado anteriormente.

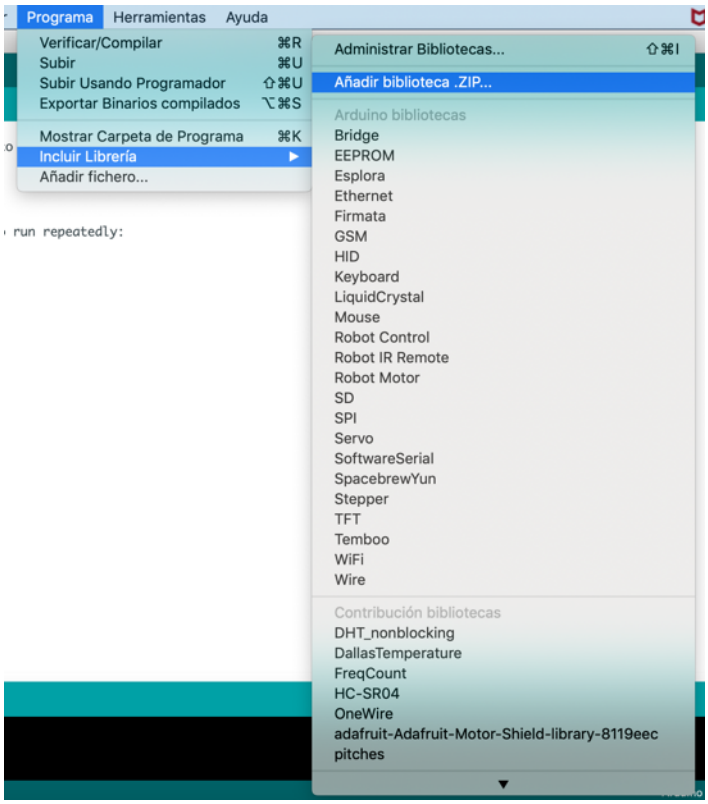

Figura 46. Camino para añadir librerías formato .ZIP .

5.3 Programa Principal.

Cualquier programa está separado en diferentes instrucciones y funciones claramente delimitados para facilitar la compresión del funcionamiento del programa. En este apartado seguiremos esta misma estructura del programa para ir explicando cada conjunto de instrucciones y funciones separadamente.

5.3.1 Instrucciones *"#include"*

Este conjunto de instrucciones hace referencia al conjunto de librerías que serán necesarias a lo largo del programa. Cabe destacar que cada una de estas librerías ha tenido que haber sido instalado como se indica en los apartados anteriores.

```
#include <PID_v1.h>
#include <AFMotor.h>
#include <SR04.h>
```

Figura 47. Instrucciones Include del programa principal.

- PID_v1.h: Esta librería ha sido descargada directamente desde internet, ya que ha sido creado por un usuario particular y nos sirve para nuestra finalidad. El objetivo de esta librería es la de simplificar la creación del PID que utilizaremos como controlador. Al añadir esta librería el controlador PID se simplifica únicamente a una función en la cual tendremos que especificar las constantes Kp, Ki, Kd anteriormente calculadas y a un comando para hacer ejecutar el controlador PID.

- AFMotor.h: Tiene como objetivo el controlar el Shield con lo cual se podrá hacer girar los motores de continua. En conclusión, la finalidad de esta librería es la aplicar y simplificar el control mediante puente en H y PWM.
- SR04.h: Sirve para la comunicación entre el sensor Módulo HC-SR04 y el Arduino Mega, realizando las conversiones convenientes para obtener la distancia en cm.

5.3.2 Instrucciones *"#define"*

Este tipo de instrucciones sirve para definir las constantes y darles un valor que permanecerán invariables a lo largo de todo el programa.

```
#define TRIG_PIN 44
#define ECHO_PIN 42
```

Figura 48. Define del programa principal.

Los dos valores que hemos definido corresponden al pin que está conectado el Arduino Mega con el receptor(TRIG_PIN) y emisor(ECHO_PIN) del sensor Módulo HC-SR04.

5.3.3 Funciones de las Librerías.

En estas líneas se están definiendo los parámetros de las distintas librerías.

```
AF_DCMotor Motor3(3);
AF_DCMotor Motor2(2);
AF_DCMotor Motor4(4);
AF_DCMotor Motor1(1);
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
PID pidController(&Dist, &Vd , &Ref, Kp, Ki, Kd, DIRECT);
```

Figura 49. Funciones de las librerías del programa principal.

- AF_DCMotor: Aquí se define cada función correspondiente al pin que ocupa cada motor.
- SR04(RIG_PIN, ECHO_PIN): Se está definiendo los pines del Arduino Mega que conectan el emisor y el receptor del sensor Módulo HC-SR04.
- PID: Se están definiendo los valores de entrada y salida del controlador, así como el punto de referencia al que queremos trabajar. También se están indicando las constantes necesarias para el controlador.

5.3.4 Variables.

En esta parte del programa se han creado las variables de tipo doble float que se necesitaran a lo largo de todo el programa. Hemos creado todas las variables del tipo doble float debido a las exigencias de las librerías siendo que estas variables tienen doble precisión.

- Kp: Corresponde al parámetro proporcional del PID diseñado.
- Ki: Corresponde al parámetro integral del PID diseñado.
- Kd: Corresponde al parámetro derivado del PID diseñado.
- Dist: En esta variable se almacena la distancia en tiempo real de cada iteración. Esta variable según la librería instalada esta en cm.
- Vd: Es la variable de salido del controlador PID el cual entrara a los motores.

- Ref: Esta variable corresponde al punto de referencia al cual se quiere llegar. Se ha decidido a priori que la distancia de seguridad sea de 20 cm.

```
double Kp=130.41, Ki=0.13, Kd=0, Dist, Va, Ref=20;
```

Figura 50. Variables creadas para el programa principal.

5.3.5 Función void setup().

Este tipo de función se encarga principalmente en para inicializar variables que no han sido inicializadas anteriormente, configurar el modo de los pines y funciones o configurar la comunicación de los datos indicando la velocidad en baudios del puerto de serie.

```
void setup()
{ Serial.begin(9600);
   pidController.SetMode(AUTOMATIC);
}
```

- Serial.begin(9600): Se abre el puerto de serie y se fija la velocidad en 9600 baudios.
- pidController.SetMode(AUTOMATIC): Se pone en modo automático el controlador PID.

5.3.6 Función void loop().

Esta función como su propio nombre indica está en un bucle(“loop”) infinito durante el funcionamiento del programa. De no estar en un bucle la secuencia de comandos se realizaría una única vez y el programa habría llegado a su fin por lo que el programa solo habría hecho su propósito una única iteración. Al colocarlo en el bucle el programa se estará ejecutando continuamente hasta que hubiéramos desconectado. Ahora iremos explicando cada línea de comando programado:

```
void loop()
{
   Dist=sr04.Distance();
   Serial.println(Dist);
   pidController.Compute();

  Motor3.run(BACKWARD);
  Motor3.setSpeed(Vd);
  Motor2.run(BACKWARD);
  Motor2.setSpeed(Vd);
  Motor1.run(BACKWARD);
  Motor1.setSpeed(Vd);
  Motor4.run(BACKWARD);
  Motor4.setSpeed(Vd);
}
```

Figura 51. Función void loop del programa principal.

- Dist=sr04.Distance(): En esta línea de comando se está llamando a la función sr04.Distancia que está incluida en la librería SR04.h. El valor de salida de esta función será la distancia entre el vehículo y el obstáculo que será guardada en el parámetro Dist.

- pidController.Compute(): Se está realizando la llamada a la función pidController.Compute(): incluida en la librería PID_v1.h. Únicamente tenemos que hacer la llamada a la función para hacer funcionar el controlador, ya que se predefinió sus valores de entrada y salida. Cabe destacar que Vd es el valor que sale de la función del PID y estará en un rango entre [0,255].
- Motor3.run(BACKWARD): Se define el sentido de giro del motor, siendo así que si quisiera el sentido contrario se tendría que haber escrito ("FORWARD")
- Motor3.setSpeed(Vd): Se define la velocidad de gira del motor que coincidirá con el valor de salida de la función del controlador PID. De estos dos tipos de funciones se puede ver que tenemos cada par para cada poder controlar cada motor.

5.4 Programa Obtención de Velocidad.

En este apartado se explicará el programa creado para la obtención de la velocidad de giro de uno de los motores gracias a al sensor FC-33 Sensor de medición de infrarrojo fotoeléctrico y un encoder conectado al eje del motor el cual también está conectado la rueda. El programa principalmente es un medidor de interrupciones. Esto quiere decir que el sensor se activa y desactiva cada vez que mide cada una de las muescas del encoder. Es por eso por lo que el programa se encargara principalmente en medir cada vez que hay un flanco de bajada (de estar activado a desactivado).

5.4.1 Instrucciones *"#include"*

En este programa únicamente será necesario utilizar la librería AFMotor.h explicado anteriormente.

```
#include <AFMotor.h>
```

Figura 52.Instrucción Include del programa obtención de velocidad

- AFMotor.h: Tiene como objetivo el controlar el Shield con lo cual se podrá hacer girar los motores de continua. En conclusión, la finalidad de esta librería es la aplicar y simplificar el control mediante puente en H y PWM.

5.4.2 Funciones de las Librerías.

En estas líneas se están definiendo los parámetros de las distintas librerías.

```
AF_DCMotor Motor3(3);
AF_DCMotor Motor2(2);
AF_DCMotor Motor4(4);
AF_DCMotor Motor1(1);
```

Figura 53.Funciones de las librerías del programa obtención de velocidad.

- AF_DCMotor: Estamos definiendo cada función con el pin que ocupa cada motor.

5.4.3 Variables.

Como en el programa principal se han creado y definido las variables que necesitaremos a lo largo del programa.

```
int PinSensor1 = 21;
unsigned int N1,M1;
volatile byte n;
unsigned long timeold, t;
unsigned int S = 20;
const int diametro = 64;
float velocidad1 = 0;
int x=255;
```

Figura 54- Variables creadas para el programa obtención de velocidad.

- PinSensor1: Se define en que pin está conectado el sensor FC-33 al tener montados dos sensores únicamente mediremos un sensor cada vez haciendo así variar entre el pin 21 o el pin 20. Cabe destacar que estos pines son unos especiales pues pueden detectar interrupciones lo cual será clave para el correcto funcionamiento.
- N1: En esta variable se guardará el número de revoluciones.
- M1: Esta variable se utilizará como ayuda para hacer el cálculo del número de revoluciones.
- n: En esta variable se guardará el número de interrupciones en un plazo de tiempo definido.
- timeold,t: Estas variables se encarga de guardar el tiempo para poder calcular cuánto tiempo ha transcurrido al medir las interrupciones.
- S: Es el número de muescas que contiene el encoder.
- velocidad1: Se guardará el valor de la velocidad calculada en (m/s)
- x: Por último, esta variable será la que se irá cambiando para ir modificando la velocidad de los motores. Cabe destacar que va desde un rango entre [0, 255] siendo que para 255 se entrega la máxima tensión de la fuente de tensión a los motores siendo así que para el máximo 255 equivale a 9V en nuestro caso y para valores menores de 255 se pude obtener por regla de tres los Voltios entregados.

5.4.4 Función void counter()

La finalidad de esta función es la ejecutar esta función cada vez que hay un flaco de bajada lo que hace subir en una unidad la variable n donde se guardara el total de interrupciones que han ha habido en un intervalo de tiempo.

```
void counter(){
      n++;
}
```

Figura 55. Función counter del programa obtención de velocidad.

5.4.5 Función void setup()

```
void setup(){
   Serial.begin(9600);
   pinMode(PinSensor1, INPUT);
   n = 0;
   N1 = 0;
   timeold = 0;
```

Figura 56. Función void setup del programa obtención de velocidad.

- Serial.begin(9600): Se abre el puerto de serie y se fija la velocidad en 9600 baudios.
- pinMode(PinSensor1, INPUT): Se define el pin PinSensor1 un pin de entrada de datos
- n=0, N=0, timeold=0: Se inicializan estas tres variables que no habían sido inicializados anteriormente.

5.4.6 Función void loop().

```
void loop(){

 Motor3.run(BACKWARD);
 Motor3.setSpeed(x);
 Motor2.run(BACKWARD);
 Motor2.setSpeed(x);
 Motor1.run(BACKWARD);
 Motor1.setSpeed(x);
 Motor4.run(BACKWARD);
 Motor4.setSpeed(x);

  attachInterrupt(2, counter, FALLING);

  do {
  }while(millis() - timeold <= 1000);
   detachInterrupt(0);
   t=millis();


   N1 = (n*60); M1=((t-timeold)*S)/1000; N1=N1/M1; velocidad1 = N1 * 3.1416 * diametro / (60000);

   n = 0;
   timeold = millis();
   detachInterrupt(digitalPinToInterrupt(PinSensor1));

   Serial.print(velocidad1);

}
```

Figura 57. Función void loop() del programa obtención de velocidad.

- Motor3.run(BACKWARD): Se define el sentido de giro del motor, siendo así que si quisiera el sentido contrario se tendría que haber escrito (“FORWARD”)
- Motor3.setSpeed(Vd): Se define la velocidad de gira del motor que coincidirá con el valor de salida de la función del controlador PID. De estos dos tipos de funciones se puede ver que tenemos cada par para cada poder controlar cada motor.
- attachInterrupt(2, counter, FALLING): Esta función configura el conteo de las interrupciones siendo así que 2 indica el pin mediante el cual se medirá estas interrupciones, cabe destacar que para esta función la enumeración de pines es distinta a la de la placa Arduino Mega siendo así que 2 corresponde con el pin 21. Counter hace referencia que cada vez que se da una interrupción se ejecuta la función counter anteriormente definida. Por último, FALLING corresponde que únicamente mediremos cuando la interrupción sea un flanco de bajada.

- do{}while(millis()-timeold<=1000): Mediante esta línea lo que se consigue es medir las interrupciones durante un segundo. Sabemos que ha pasado un segundo gracias a la condición (millis()-timeold<=1000) donde millis es tiempo actual y timeold el tiempo que se guardó de la anterior iteración.
- detachInterrupt(0): Se parará el conteo de las interrupciones para poder realizar el cálculo.
- t=millis(): Se guarda en t el momento en el cual se para el conteo de las interrupciones.
- N1 = (n*60); M1=((t-timeold)*S)/1000; N1=N1/M1; velocidad1 = N1 * 3.1416 * diámetro / (60000): Mediante este conjunto líneas se consigue calcular la velocidad la cual se guardará en la variable velocidad1
- n = 0: Se elimina el valor del conteo anterior y se inicializa para poder realizar un nuevo conteo en la siguiente interacción.
- timeold = millis(): Se guarda el instante en el cual se ha reiniciado el conteo y se va a iniciar el nuevo conteo.
- detachInterrupt(digitalPinToInterrupt(PinSensor1)); Se reinicia el conteo de interrupciones.
- Serial.print(velocidad1): Se muestra por el Serial Monitor el valor de velocidad 1.

6. RESULTADOS OBTENIDOS.

Una vez creado y diseñado el controlador comprobamos el funcionamiento de este comparando la respuesta que obtendríamos al simular el modelo.

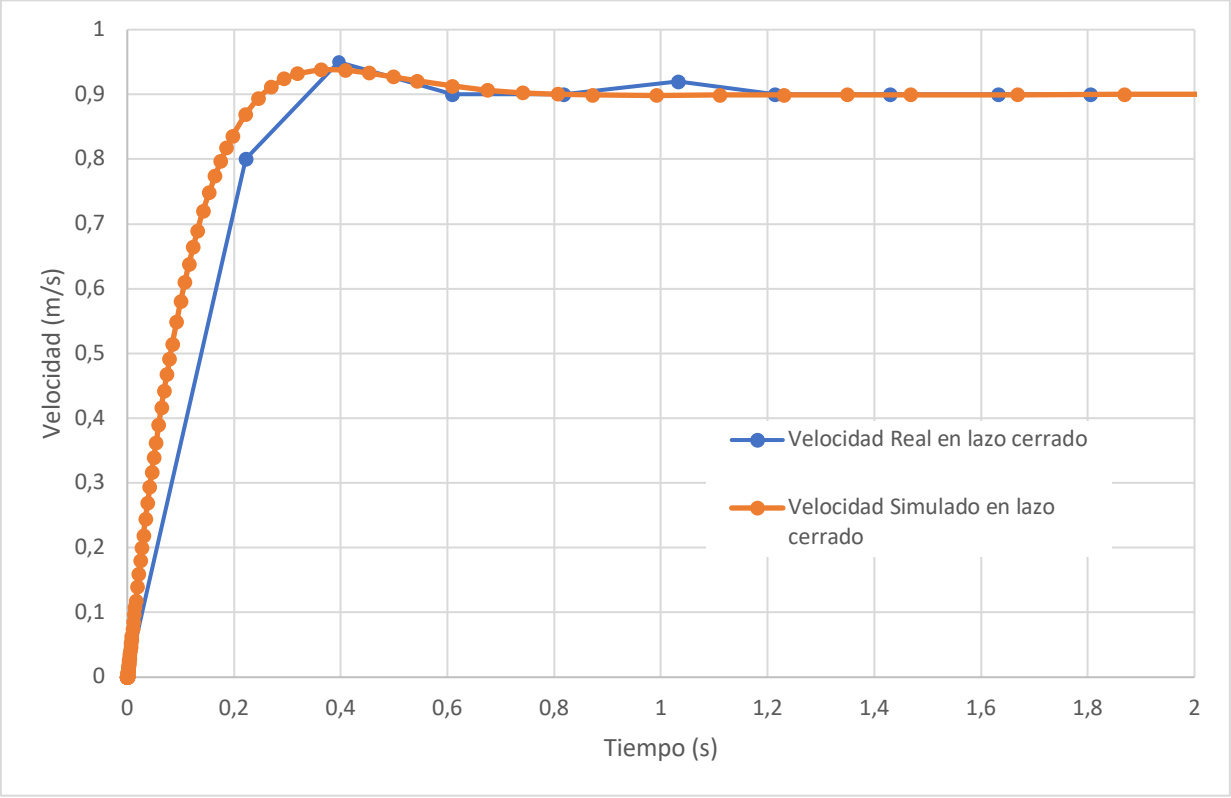

Figura 58. Comparación velocidad Real y Simulada en lazo cerrado.

En la Fig.58 se representan dos curvas las cuales corresponden a la velocidad en cada instante de momento ante un escalón de 20 cm constantes con la cual se obtiene la velocidad máxima. Siendo por eso que su velocidad en régimen permanente es distinta a 0 m/s y es de aproximadamente 0,9 m/s. Siendo que la curva azul corresponde a la velocidad que hemos obtenido midiendo la velocidad del vehículo con los sensores e introduciendo manual mente la distancia para obtener un error de 20 cm. Mientras que la curva naranja corresponde a la velocidad simulada siendo la velocidad que deberíamos obtener si el sistema fuera perfecto. Ambas curvas no son coincidentes debido a que existen error debido a la incertidumbre de parámetros o debido al margen de error del sensor. Aun debido a la existencia de errores mínimos se aprecia que ambas curvas son muy similares lo cual nos indica que el controlador aplicado ejerce su función correctamente.

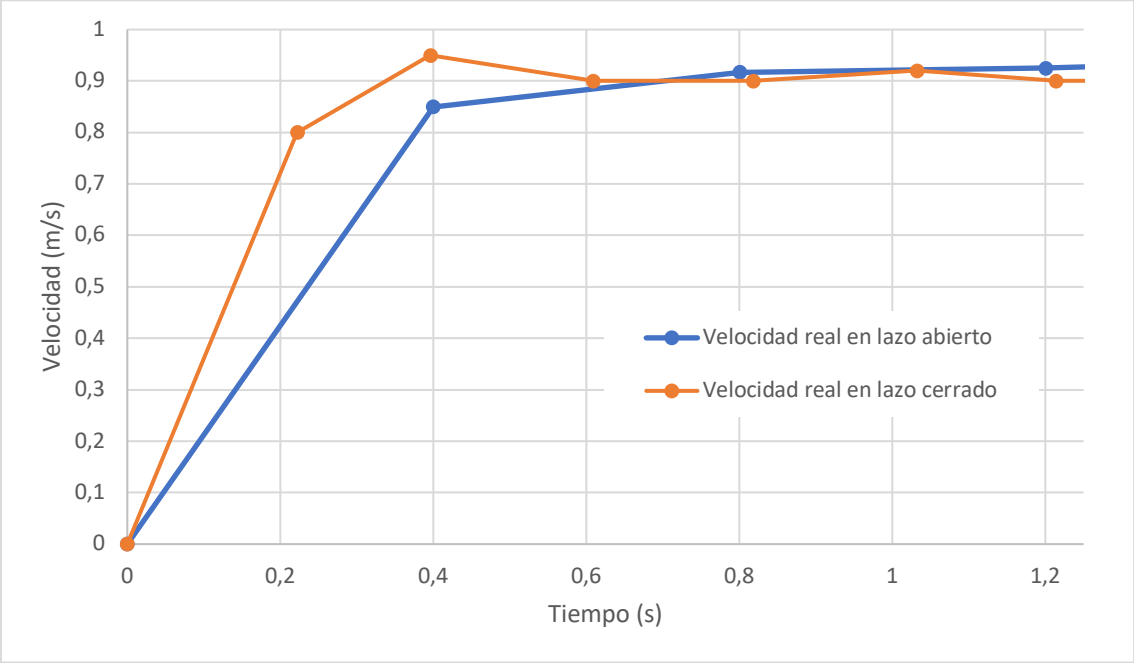

Figura 59. Velocidad real en lazo abierto y cerrado.

En la Fig.59 se ha representado la respuesta real del sistema cuando está en lazo cerrado y en lazo abierto para obtener la misma velocidad máxima de 0,9 m/s. Podemos apreciar que el controlador en el lazo cerrado hace posible un tiempo de establecimiento de 0,5 s mientras que el mismo sistema en lazo abierto el tiempo de establecimiento sería mayor (aproximadamente 1 s).

Una vez que hemos comprobado el buen funcionamiento de tanto el programa como del controlador que este programa está implementando se realiza la comparación del tiempo que se tarda en alcanzar en el punto de referencia del modelo en lazo cerrado y en lazo abierto.

En la primera prueba se comprobará el funcionamiento y la respuesta del vehículo en lazo cerrado con un controlador PI el cual fue diseñado anteriormente y se lleva explicando a lo largo de todo el documento. Se realizará un experimentó en el cual estaremos acercando un obstáculo en un intervalo de tiempo intentado que este obstáculo se tenga una velocidad constante.

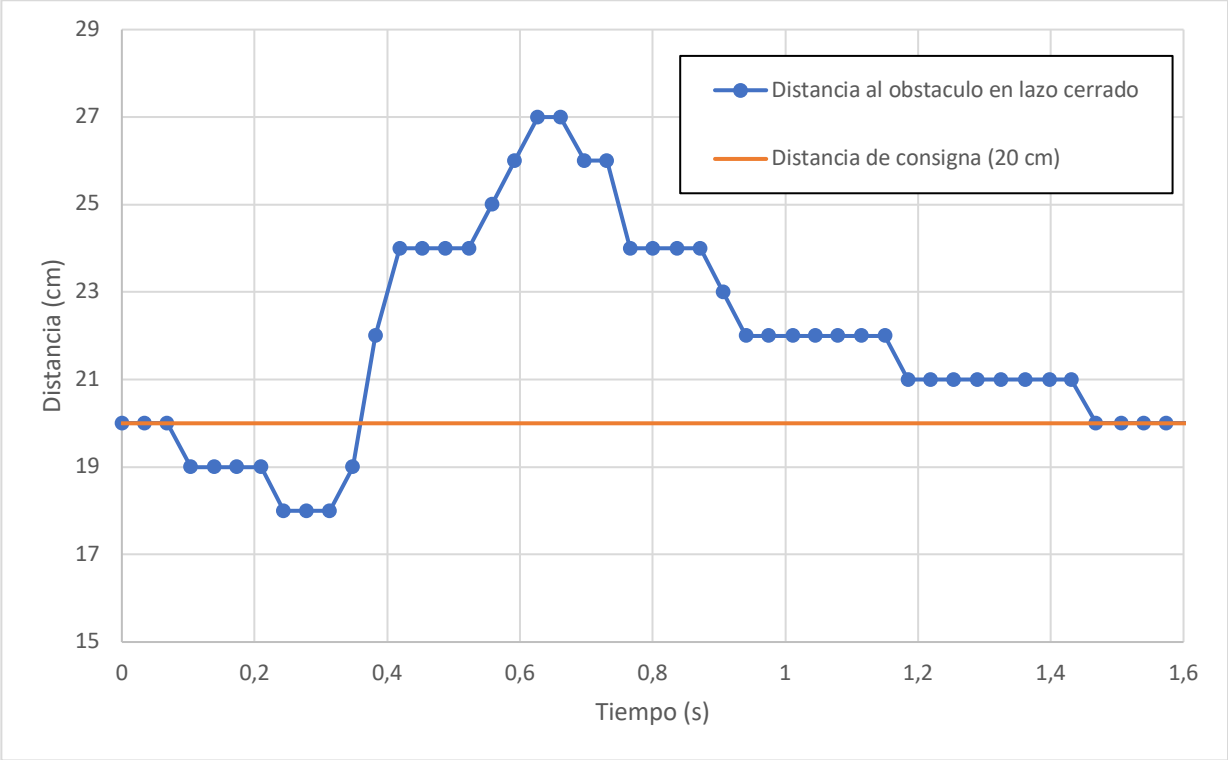

Figura 60. Respuesta de la distancia al obstáculo en lazo cerrado.

En la Fig.60. se observa la respuesta del vehículo al variar la distancia entre el obstáculo y el vehículo, ya que el obstáculo se está acercando a velocidad constante hasta un punto en el cual empieza a frenarse y detenerse el obstáculo. Justamente en el instante 0,068 s se empieza a mover el obstáculo. El controlador calcula el error entre el dato del sensor y el dato de referencia generando así una respuesta proporcional al cambio en el sistema (el obstáculo acercándose).

Siendo así que una vez que la variación en el sistema se detiene (el obstáculo deja de moverse) el controlador al calcular la diferencia entre el punto de referencia y el dato del sensor (error) es igual a cero ha llegado de nuevo al punto de funcionamiento deseado. El controlador ya habrá realizado su función y dejará de actuar. Lo anteriormente comentado se puede aplicar a la Fig.59 puesto que el inicio de la variación en el sistema se realiza en el instante 0,068 s manteniéndose esta variación (acercamiento del obstáculo) hasta el instante 1,506 s. A partir del instante 0,731 s la distancia de seguridad se está reduciendo, ya que el controlador con el sensor detecta que el obstáculo se está frenando con lo que se genera una respuesta cada vez más tenue hasta el punto de que se ha conseguido llegar al punto de referencia y el obstáculo se ha frenado. El punto de referencia deseado se alcanza en el instante 1,468 s.

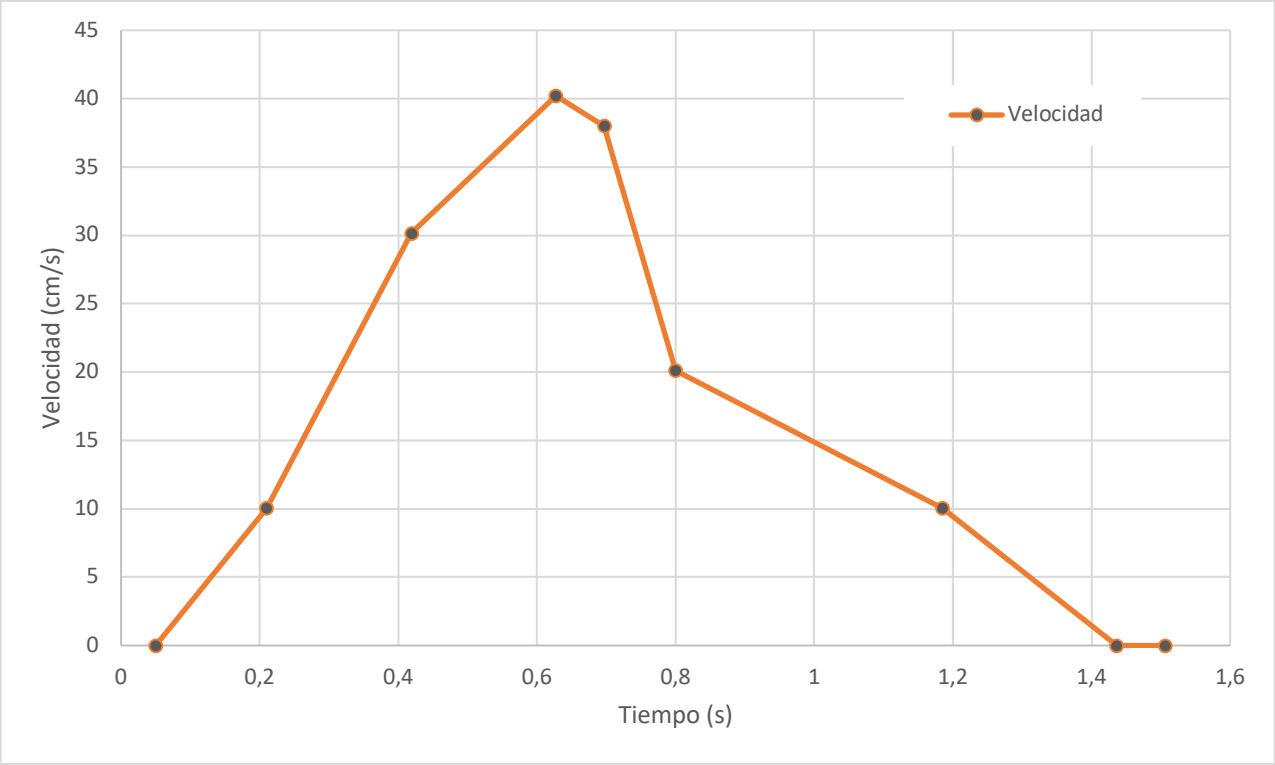

Figura 61. Velocidad real en lazo cerrado.

En la Fig. 61 se representa la velocidad en cada instante del anterior experimentó con el cual conseguimos extraer la Fig.59. Enlazando a lo anteriormente dicho se puede volver a comprobar que la variación al empezar en el instante 0,068 s se empieza a generar una respuesta proporcional. Se aprecia que la curva de la velocidad comprendida entre el intervalo (0,068 s y 0,731) tiene muchas similitudes con la curva de la Fig.58, ya que tiende a tener la misma forma con una ligera sobretensión al principio y un parecido tiempo de establecimiento (0,5 s). Aunque a partir del instante 0,731 s se aprecia que la velocidad va disminuyendo lo que afirmaría lo anteriormente dicho que el obstáculo se está frenando y por tanto se generara una respuesta cada vez más tenue reduciendo así la velocidad de los motores.

En la Fig.62 se ha representado y superpuesto las Fig.61 y Fig.60 de la velocidad y distancia del vehículo. Se aprecia que se disminuye la velocidad de los motores se empieza a disminuir la distancia de seguridad al compás que se va frenando el obstáculo.

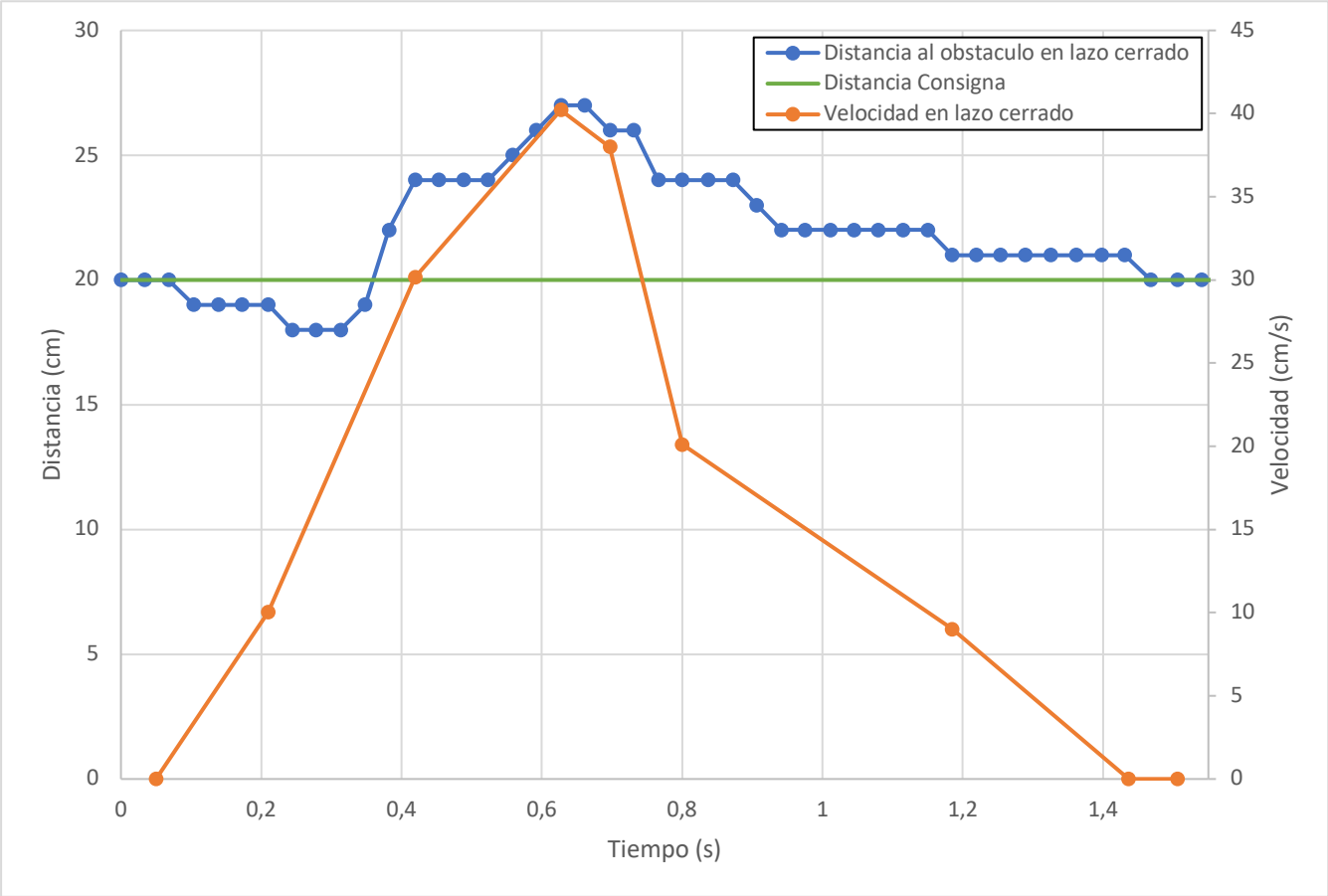

Figura 62. Comparación velocidad y distancia en bucle cerrado.

7. CONCLUSIÓN.

En este último apartado se comentará las conclusiones extraídas a lo largo del documento.

En primer lugar, destacar que se ha logrado alcanzar satisfactoriamente todos los objetivos que se habían impuesto en un principio, los cuales se resumen a continuación:

- Diseño y montaje de un vehículo móvil con todos los elementos necesarios para su funcionamiento.
- Aplicar los conocimientos físicos y técnicos adquiridos a lo largo de la carrera para la aplicación de un modelo que se ajuste a la realidad del vehículo móvil.
- Diseño teórico del controlador PID con el cual podemos cumplir el objetivo de distancia de seguridad.
- Programación con el programa Arduino (2020) con el cual podemos aplicar el controlador PID diseñado para poder controlar efectivamente el vehículo móvil.

Se ha logrado crear un vehículo móvil mediante al cual hemos podido medir los distintos parámetros como velocidad, posición, tensión. Etc. Se ha realizado una mejora en el tiempo de respuesta de los motores gracias a la implantación de un lazo cerrado con un controlador PI, gracias al haber conseguido un modelo que se ajustaba correctamente al mundo real. Esta misma mejora en la respuesta en los motores se refleja en una mayor rapidez para que el vehículo alcance la distancia de seguridad (punto de referencia).

A pesar de tener un excelente nivel en la programación en lenguaje en C++, he necesitado un tiempo de adaptación del lenguaje utilizado por el Arduino a pesar de que este está fundamentado en el lenguaje C++, ya que en Arduino se incluyen funciones nunca vistas. También aparecieron complicaciones a lo largo del proyecto a la hora de programar y hacer que funcionase tanto los motores como el resto de los sensores según mi propuesta de diseño, pero básicamente se aprende con el clásico método de prueba/error.

Gracias a este proyecto he visto que soy capaz de resolver cada problema. Informándome y recabando la información suficiente para detectar dónde está el error (troubleshooting) y pensar, de una forma muy estructurada y lógica, de las posibles herramientas y opciones que tengo para dar solución. Al mismo tiempo me reconforta el hecho de haber creado a pequeña escala un elemento de seguridad que puede ser implementado en cualquier otro sistema.

8. BIBLIOGRÁFICA.

Arduino(2020). Recuperado 13 abril de 2020 de:

https://www.arduino.cc

Características Arduino MEGA 2560 (2013). Recuperado el 13 abril de 2020 de:

http://panamahitek.com/arduino-mega-caracteristicas-capacidades-y-donde-conseguirlo-en-panama/

Descripción Shield Motor Driver L293D.(2020). Recuperado 30 de abril de 2020 de:

https://naylampmechatronics.com/arduino-shields/86-shield-motor-driver-l293d.html

Características sensor HC-SR04. Recuperado 5 mayo de 2020 de:

https://www.hwlibre.com/hc-sr04/

Rectificación de pulsos malos.(2016). Recuperado 8 de mayo de 2020 de:

http://androminarobot.blogspot.com/2016/07/en-este-tutorial-mostramos-como-usar-el.html

Determinación de los parámetros motor de corriente continua (2009).Recuperado 20 de abril 2020 de :

https://repository.upb.edu.co/bitstream/handle/20.500.11912/504/digital_17633.pdf?sequence=1

Hacer un encoder óptico con un optointerruptor y Arduino.(2016). Recuperado 27 de abril de 2020 de:

https://www.luisllamas.es/usar-un-optointerruptor-con-arduino/

Benjamin C. Kuo(1996). Capitulo 4.Modelado matemático de sistemas físicos. Sistemas de control automático(pp.179-178)Nueva Jersey, USA:PEARSON PRENTICE HALL.

Karl J. Åström Tore Hägglund (2009). Control PID avanzado. Department of Automatic Control Lund Institute of Technology Lund University.

Alicia Esparza (2019) Apunte asignatura Sistemas Automáticos. Departamento de Ingeniería de Sistemas y Automática. Valencia: Universidad Politécnica de Valencia.

Pedro García Gil (2019). Apuntes asignatura Tecnología Automática. Departamento de Ingeniería de Sistemas y Automática. Valencia: Universidad Politécnica de Valencia.

DOCUMENTO II:

PRESUPUESTO

Modelado y diseño de control de trayectoria de un vehículo mediante Arduino

ÍNDICE DEL PRESUPUESTO.

1. INTRODUCCIÓN.

En el presente documento se va a realizar el desglose del presupuesto de todo el Trabajo Final de Grado.

En primer lugar, será calculado el precio de mano de obra de los integrantes necesarios para el desarrollo de proyecto, así como las horas de dedicación.

A continuación, será calculado los costes de los materiales necesarios para la elaboración del proyecto, también se incluirán los costes de las herramientas necesarios para desarrollar el proyecto.

Por último, será calculado el presupuesto final incluyendo todas las partes anteriores, así como los gastos generales , el beneficio industrial y el IVA.

2. MANO DE OBRA.

En este apartado se hará el cálculo de todos los costes de las horas de trabajo empleadas en el proyecto.

En este proyecto ha contribuido un ingeniero industrial(tutor de la UPV) con un salario medio de 50000€/año, también ha contribuido un ingeniero técnico en prácticas (estudiante) con un salario medio de 20000€/año.

Se realiza el cálculo de salario en €/h considerando que un año laboral incluyendo vacaciones es de 250dias/anuales. Así mismo se admite que un trabajador trabajo 8h/día siendo este el límite legal. Por tanto:

Ingeniero Industrial

$$50000\frac{€}{año} * \frac{1}{250}\frac{año}{dias} * \frac{1}{8}\frac{dia}{horas} = 25\ €/h$$

Ingeniero técnico en prácticas

$$20000\frac{€}{año} * \frac{1}{250}\frac{año}{dias} * \frac{1}{8}\frac{dia}{horas} = 10\ €/h$$

Resumiendo, se obtiene que:

Tabla 1. Coste de mano de obra.

	Coste €/h
Ingeniero Industrial	25
Ingeniero técnico en prácticas	10

En la tabla de a continuación se reflejará las horas dedicadas a cada actividad a lo largo de todo el proyecto siendo así:

Tabla 2. Horas por cada actividad.

	h
Reuniones	10
Planificación	10
Búsqueda de información	30
Montaje	10
Diseño del sistema	40
Programación	20
Elaboración de los documentos	100
Total	220

Dentro de todas estas actividades únicamente la actividad de reunión ha sido realizada tanto por el Ingeniero Industrial como por el Ingeniero Técnico en Prácticas. Todas las demás actividades han sido realizadas por el Ingeniero Técnico en Prácticas exclusivamente.

El presupuesto de mano de obra será:

Tabla 3. Presupuesto parcial Mano de Obra.

Unidades	*Descripción*	*Cantidad*	*Precio*	*Importe*
h	Ingeniero Industrial	20	25	500
h	Ingeniero técnico en prácticas	220	10	2200
			Total	2700

3. COSTES DE MATERIALES

Se realiza el desglose en detalle de los materiales que han sido necesarios para la elaboración de este proyecto.

Cabe destacar que el precio del adaptador de corriente es superior a la media de los adaptadores de corriente en el mercado, puesto que este adaptador de corriente incluye la opción la poder variar la tensión que entregue mediante una ruleta.

Tabla 4. Presupuesto parcial de Materiales.

Unidades	*Descripción*	*Cantidad*	*Precio*	*Importe*
Ud	Arduino MEGA 2560+Cables	1	13,99	13,99
Ud	Controlador Arduino Shield L293D	1	8,79	8,79
Ud	Kit Robot Chasis+Motores+Encoder	1	16,99	16,99
Ud	Sensor HC-SR04	1	2,94	2,94
Ud	FC-33 Sensor infrarrojo fotoeléctrico	2	1,99	3,98
Ud	Pack de cables	1	6,99	6,99
Ud	Adaptador de corriente	1	12,99	12,99
			Total	66,67

4. COSTES DE LAS HERRAMIENTAS NECESARIAS PARA EL PROYECTO.

En este apartado se especificarán todos los costes de tanto de las herramientas necesarias para el montaje como de las herramientas necesarias para la obtención de datos. Se incluyen estos costes en un nuevo aparatado, ya que estas herramientas no se han tenido que adquirir con el único objetivo para realizar este proyecto, sino que se podrán utilizar en otros proyectos o por el contrario ya se tenían estas herramientas de proyectos anteriores.

Tabla 5.Presupuesto parcial de Herramientas.

Unidades	*Descripción*	*Cantidad*	*Precio*	*Importe*
Ud	Kit de soldador eléctrico, 60W	1	19,99	19,99
Ud	Kit de herramientas	1	9,99	9,99
Ud	Multímetro Digital MESTEK	1	8,61	8,61
Ud	BreadBoard 170 puntos	1	1,75	1,75
			Total	40,34

5. PRESUPUESTO TOTAL

Por ultimo se realiza el cálculo de todo el presupuesto en su conjunto. Se sumarán todos los presupuestos parciales anteriores con lo que se obtendrá el Presupuestó de Ejecución Material. A este Presupuesto de Ejecución Material hay que tener en consideración las suma de los gastos generales (13%) y el beneficio industrial (6%). Por último se deberá tener en cuenta también el IVA (21%) obteniendo así el Presupuesto Base de Licitación.

Tabla 6. Presupuesto Final.

Presupuesto parcial de Mano de Obra	2700
Presupuesto parcial de Materiales	66,67
Presupuesto parcial de Herramientas	40,34
PRESUPUESTO EJECUCIÓN MATERIAL	2807,01
Gastos Generales (12%)	336,84
Beneficio Industrial (6%)	168,42
PRESUPUESTO EJECUCIÓN POR CONTRATA	3312,27
IVA (21%)	695,58
PRESUPUESTO BASE DE LICITACIÓN	4007,85

Se obtiene que el Proyecto tiene un coste de 4.007,85 €

(CUATRO MIL SIETE EUROS CON OCHENTA Y CINCO CÉNTIMOS)

ANEXO I:

PROGRAMA PRINCIPAL

```
#include <PID_v1.h>
#include <AFMotor.h>
#include "SR04.h"

#define TRIG_PIN 44
#define ECHO_PIN 42

AF_DCMotor Motor3(3);
AF_DCMotor Motor2(2);
AF_DCMotor Motor4(4);
AF_DCMotor Motor1(1);
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);

double Kp=130.41, Ki=0.13, Kd=0,Dist, Va, Ref=20;
PID pidController(&Dist, &Va , &Ref, Kp, Ki, Kd, DIRECT);

void setup()
{ Serial.begin(9600);
   pidController.SetMode(AUTOMATIC);}

void loop()
{
  Dist=sr04.Distance();
  pidController.Compute();
  Motor3.run(BACKARD);
  Motor3.setSpeed(Va);
  Motor2.run(BACKARD);
  Motor2.setSpeed(Va);
  Motor1.run(BACKARD);
  Motor1.setSpeed(Va);
  Motor4.run(BACKARD);
  Motor4.setSpeed(Va); }
```

ANEXO II:

PROGRAMA OBTENCIÓN DE VELOCIDAD

```
#include <AFMotor.h>

AF_DCMotor Motor3(3);
AF_DCMotor Motor2(2);
AF_DCMotor Motor4(4);
AF_DCMotor Motor1(1);

int PinSensor1 = 21;
unsigned int N1, M1;
volatile byte n;
unsigned long timeold, t;
unsigned int S = 20;
const int diametro = 64;
float velocidad1 = 0;
int x=255;

 void counter(){
     n++;    }

void setup(){
  Serial.begin(9600);
  pinMode(PinSensor1, INPUT);
  n = 0;
  N1 = 0;
  timeold = 0;

 void loop(){
 Motor3.run(BACKARD);
 Motor3.setSpeed(x);
 Motor2.run(BACKARD);
 Motor2.setSpeed(x);
 Motor1.run(BACKARD);
 Motor1.setSpeed(x);
 Motor4.run(BACKARD);
 Motor4.setSpeed(x);
 attachInterrupt(2, counter, FALLING);

  do {
  }while(millis() - timeold <= 1000);
   detachInterrupt(0);
   t=millis();

   N1 = (n*60); M1=((t-timeold)*S)/1000; N1=N1/M1; velocidad1 = N1 * 3.1416 * diametro / (60000);

   n = 0;
   timeold = millis();
   detachInterrupt(digitalPinToInterrupt(PinSensor1));

   Serial.println(velocidad1);}
```