ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

OPTIMIZACIÓN DE UN MONTACARGAS

Autor: Sergio Méntrida Zamarra
Director: Richard Gourdeau

Madrid

Junio, 2018

Lista de Documentos

DOCUMENTO I: MEMORIA

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Sergio Méntrida Zamarra

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: OPTIMIZACIÓN DE UN MONTACARGAS,

que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que

a) pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.
b) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 15 de Junio de 2018

ACEPTA

Fdo. Sergio Méntrida Zamarra

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Optimización de un montacargas

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2017/2018 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Sergio Méntrida Zamarra Fecha: 2018/ 05 /08

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Richard Gourdeau Fecha: 2018 05 08

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

OPTIMIZACIÓN DE UN MONTACARGAS

Autor: Sergio Méntrida Zamarra
Director: Richard Gourdeau

Madrid

Junio, 2018

Resumen:

OPTIMIZACIÓN DE UN MONTACARGAS

Autor: Méntrida Zamarra, Sergio.

Director: Gourdeau, Richard.

Entidad Colaboradora: Université de Montréal, École Polytechnique.

RESUMEN DEL PROYECTO:

Introducción

En la Universidad Politécnica de Montreal se utilizan dos plantas piloto de forma didáctica para las prácticas en laboratorio de la asignatura "Commande des processus industriels". Esta asignatura enseña sobre el control y la automatización de procesos industriales. Cada planta piloto está compuesta de un montacargas: un carro que se desplaza en horizontal a lo largo de un raíl y que transporta una carga que se desplaza verticalmente con una cuerda.

La Universidad quiere modernizar las plantas piloto, debido al elevado coste de mantenimiento de los equipos del sistema actual, y a problemas de seguridad con el alumnado a lo largo de las prácticas. Aprovechando el reemplazo de los equipos que se han quedado anticuados se quiere optimizar el funcionamiento del montacargas, añadiendo un sistema de seguridad más completo y cambiando el sistema de control del carro. Es un proyecto grupal, compuesto por dos profesores, un técnico y mi aportación. Nuestro trabajo se centra principalmente en dar respuesta al sistema de seguridad.

Con el sistema actual, los desplazamientos del carro a gran velocidad no están limitados hasta que éste llega al extremo del raíl por lo que la carga y el carro pueden impactar violentamente contra las paredes del laboratorio. Ante este problema, se pretende diseñar un sistema de seguridad que funcione de forma independiente y que limite la velocidad del carro para que no choquen ni el carro ni la carga que transporta contra los extremos del raíl.

Para ello, los objetivos que se han fijado para el proyecto son los siguientes:

- Diseñar un sistema de seguridad basado en el control del voltaje de un motor que desplaza un carro mediante la medición de la posición de éste último.
- Implementar la comunicación entre un motor y una herramienta de software, pudiendo controlar el voltaje y sentido de giro del motor. La comunicación se va a efectuar mediante un módulo UART y también mediante un potenciómetro que regule el voltaje del motor.
- Medir mediante tecnología láser la posición absoluta de un carro móvil y poder manipular los datos de distancia obtenidos.
- Adquirir un montaje final que simule el funcionamiento de la futura planta piloto provista de un sistema de seguridad completo.

Una vez se tenga un montaje con un sistema de seguridad funcional, se podrá probar el diseño en la planta piloto final que servirá a futuros alumnos para la realización de sus prácticas.

Metodología

La herramienta principal de software que se ha usado para la realización del proyecto es Matlab-Simulink, y para otras tareas subordinadas se ha utilizado el software Arduino. El diseño del sistema de seguridad se ha realizado en una pequeña maqueta, compuesta por un motor que desplaza un carro por un rail de 1m de longitud. No se ha podido implementar el sistema de seguridad diseñado en la planta piloto final por motivos de calendario, pero los resultados extraídos de la maqueta servirán para su implementación en la planta piloto.

Resultados

Los resultados finales del proyecto se han obtenido experimentando en la maqueta diseñada a efectos del proyecto. Los pasos para realizar los objetivos marcados han sido los siguientes.

En primer lugar, se ha logrado realizar la comunicación entre el software Matlab-Simulink y el motor que desplaza el carro. Con una placa de control de motor de corriente continua, un microcontrolador y un potenciómetro lineal se ha implementado un modelo en el cual el desplazamiento del carro es dirigido por el potenciómetro. Con el sistema que se ha montado, variando un voltaje de referencia (de 3V) con el potenciómetro se puede decidir cuál es el sentido y la potencia de giro del motor. De modo que un extremo del potenciómetro (0V) representa un punto de trabajo máximo del motor con un sentido de giro, y el otro extremo (3V) representa también un punto de trabajo máximo del motor pero con el otro sentido de giro. Siendo la potencia de giro del motor proporcional al voltaje generado por el potenciómetro. A parte de este modelo se ha introducido otra forma de generar el mando, sustituyendo el potenciómetro por un módulo UART (Universal Asynchronous Receiver Transmitter) para comandar ganancias al motor desde el propio Matlab-Simulink.

Tras obtener un procedimiento por el cual controlamos la velocidad del carro y su sentido de desplazamiento, se ha constituido la medición de la posición del carro mediante tecnología láser. Se ha buscado un método económico y preciso en el contexto de nuestra aplicación. Desde el software Arduino se han captado los datos provenientes del láser y se ha extraído la posición del carro medida en todo momento.

Una vez reunidas todas las partes, se ha logrado implementar un sistema de seguridad funcional, comunicando los software de Matlab-Simulink y Arduino con el motor que desplaza el carro. Se ha obtenido un modelo en el cual el mando es controlado por un potenciómetro y el programa de Arduino es el que toma el control del motor cuando el carro se sitúa en una situación de riesgo. Se ha dividido el raíl en distintas zonas, de modo que hemos definido dos situaciones de riesgo:

- Primeramente cuando el carro se desplaza a gran velocidad y cerca del extremo, en una zona que hemos denominado zona de frenado. En esta zona se compara el mando generado por el potenciómetro con un patrón de referencia y si la ganancia del mando es mayor que la del patrón entonces el patrón de referencia toma el control del motor. Este patrón es una señal PWM generada por el propio Arduino que varía en función de la posición del carro. De forma que la señal PWM disminuye su ciclo de trabajo de forma lineal según se aproxima el carro al extremo del raíl. Con este sistema, en la zona de frenado no podremos superar

ciertas velocidades debido a la limitación del patrón de referencia, que se ha comprobado que es suficientemente seguro.

- La segunda situación de riesgo se da cuando el carro ya se encuentra muy próximo al extremo. En estos casos, la influencia del programa de Arduino obliga al motor a detenerse por completo. Esta zona que hemos definido como zona de detención se sitúa más al extremo que la zona de frenado, y asegura que el carro se detenga por completo para evitar su impacto con el fin del raíl.

Se ha puesto a prueba el sistema de seguridad y los resultados han sido satisfactorios. El carro jamás impacta contra ninguno de los extremos, y el sistema diseñado puede adaptarse a otro tipo de plantas. Los únicos parámetros que se deben variar para implementar este sistema de seguridad en otras plantas son la delimitación de las zonas de frenado y detención fijadas en el programa de Arduino. Si se desea reemplazar el mando que se genera con el potenciómetro se debe modificar el modelo Matlab-Simulink a efectos de estos cambios.

Conclusiones

Los resultados obtenidos del proyecto han sido verificados exclusivamente en una maqueta montada a efectos de este mismo. Sin embargo, el sistema diseñado es adaptable a otros montacargas bajo la modificación de pocos parámetros. El futuro de este proyecto es que se implemente el sistema de seguridad diseñado en la planta piloto de la Universidad de Montreal.

Sin embargo, podemos afirmar que es un sistema aplicable en plantas piloto similares, y por tanto viable para ser utilizado en la industria. Las principales ventajas de nuestro sistema son que es muy sencillo, flexible y fácilmente modificable. Este proyecto puede dar solución de una forma sencilla a la implantación de sistemas de seguridad en plantas industriales que empleen montacargas, ayudando a que el lugar de trabajo y los equipos que se manejen sean más seguros.

Abstract

OPTIMIZATION OF A HOIST

Introduction

At the University Polytechnic of Montreal there are currently in use two hoist simulations for laboratory tests of the subject “Commande des processus industriels”. This course teaches students about the control and automation of industrial processes. Each hoist simulation is composed of a car that moves horizontally along a rail and transports a load that moves vertically thanks to a rope.

The University wants to modernize the laboratory, due to the high cost of maintenance of the equipment in the current system, and due to safety problems with the students throughout the practices. Taking advantage of the replacement of the outdated equipment, we want to optimize the hoist simulation, adding a broader safety system and changing the control system of the car.

It is a group project, composed by two teachers, a technician and my contribution. This work focuses mainly on responding to the security system.

With the current system, the displacements of the car at high speed are not limited until it reaches the end of the rail so the load and the car can impact violently against the walls of the laboratory. To solve this problem, the intention was to design a safety system that works independently and limits the speed of the car so that neither the car nor the load will collide against the ends of the rail.

The objectives set for the project are the following:

- To design a safety system based on the voltage control of an engine that moves a car by measuring its position.
- To implement the communication between an engine and a software tool, being able to control the voltage and direction of rotation of the motor. The communication will be made through a UART module and also a potentiometer that regulates the voltage of the motor.
- Measuring by laser technology the absolute position of a mobile car and being able to manipulate the obtained distance data.

- Acquiring a final assembly putting together all the components; simulating the functioning of the future laboratory equipped with a complete safety system.

Once the montage with the functional safety system is made, experiments will be done in order to test the design. This system shall be tested in the final laboratory that will serve future students to carry out their practices.

Methodology

The main software tool that has been used for the realization of the project is Matlab-Simulink. For other subordinate tasks, the Arduino software has been used. The design of the security system has been made in a small model, composed of a motor that moves a car by a rail of 1m in length. It has not been possible to implement the safety system designed in the final laboratory due to schedule reasons, but the results extracted from the model will be used for its implementation.

Results

The final results of the project have been obtained by experimenting with the model designed for the project. The steps to accomplish the objectives have been the following.

First of all, communication between the Matlab-Simulink software and the motor that moves the car has been achieved. With a DC motor control board, a microcontroller and a potentiometer; a model has been implemented in which the displacement of the car is controlled by the potentiometer. With the system that has been created, by varying a reference voltage (3V) with the potenciometer, the direction and the power of rotation of the motor can be elected. One extreme point of the potentiometer (0V) represents a maximum working point of the motor with one direction of rotation, and the other extreme point (3V) also represents a maximum working point of the motor, but on the other direction of rotation. Therefore, the power of rotation of the motor is proportional to the voltage generated by the potentiometer.

Apart from this model, a similar one has been introduced to generate the command to the motor, replacing the potentiometer for a UART module (Universal Asynchronous Receiver Transmitter) to command gains to the motor from the Matlab-Simulink itself.

After obtaining a procedure by which we control the speed of the car and its direction, we will move onto the measurement of the position that has been constituted by a laser. An inexpensive and precise method has been sought in the context of our application. It has been possible to capture the absolute position of the car by communicating the software Arduino with the laser.

Once all the parts have been assembled, a functional safety system has been implemented, communicating Matlab-Simulink and Adruino software with the motor that moves the car. A model has been obtained in which the command is controlled by a potentiometer and the Arduino program is the one that takes control of the engine when the car is placed in a risky situation. The rail has been divided into different zones, so we have defined two risky situations:

- Firstly, when the car moves at high speed and near the end of the rail, in an area that we have called the braking zone. In this zone, the command generated by the potentiometer is compared with a reference pattern and if the gain of the command is greater than the pattern one, then the reference pattern takes control of the motor. This pattern is a PWM signal generated by the Arduino itself that varies depending on the position of the car. Due to this, the PWM signal decreases its duty cycle linearly as the car approaches the end of the rail. With this system, in the braking zone we cannot exceed certain speeds due to the limitation of the reference pattern, which has been checked to be safe.
- The second risky situation occurs when the car is already very close to the end of the rail. In these cases, the influence of the Arduino program forces the motor to stop completely. This area, which we have defined as the stopping area, is located more in the extreme than the braking zone, and ensures that the car stops completely to avoid its impact with the end of the rail.

The security system has been tested and the results have been satisfactory. The car never hits any of the extremes, and the designed system can be adapted to other types of hoist. The only parameters that must be changed to implement this

security system into other hoists are the delimitation of braking and stopping zones, set in the Arduino program. If a replacement of the command system (the potentiometer in this case) is needed, the Matlab-Simulink model must be modified in consequence.

Conclusions

The results obtained from the project have been verified exclusively in a model assembled for the purposes of the project. Nevertheless, the designed system is adaptable to other hoist under the modification of a few parameters. The future of this project is to implement the security system in the future laboratory of the University of Montreal.

However, we can affirm that this system is applicable in similar models, and therefore viable to be used in the industry. The main advantages of our system are that it is very economical, simple and easily modified. This project can provide a simple solution to the implementation of security systems in industrial plants that use mobile hoist, helping to make the workplace and the equipment handling safer.

A mis padres,

Sin ellos nada de esto hubiera sido posible

Sólo cabe progresar cuando se piensa en grande,

Sólo es posible avanzar cuando se mira lejos.

JOSÉ ORTEGA Y GASSET

Agradecimientos

Quisiera dar las gracias a todas las personas que me han apoyado a lo largo de mi carrera. En primer lugar a quienes han participado directamente en este proyecto: el coordinador Aurelio García Cerrada, el director Richard Gourdeau, Saad Chidami y David Fecteau.

Doy las gracias a la Universidad Pontificia de Comillas (ICAI) y la Universidad Politécnica de Montreal por darme las herramientas necesarias y permitirme hacer este proyecto. Gracias al conjunto de profesores que me han acompañado durante mi vida académica. Ha sido un camino difícil hasta la redacción de este documento, el esfuerzo y la dedicación han sido las claves para conseguirlo.

El agradecimiento más profundo y sentido es para mis padres: Jesús Méntrida y Dolores Zamarra. Me veo obligado a decir que todo el trabajo que he realizado ha sido gracias a ellos, les debo todo. A más componentes de la familia: Inés, Leire, Jose Miguel, María, Guillermo, Ana, Loly, Paco.

Dedico una mención especial a mis amigos por su apoyo incondicional, me han enseñado a ser feliz y nunca bajar los brazos.

DOCUMENTO I

MEMORIA

Índice de Contenido

Índice de Figuras

Índice de Tablas

Índice de códigos

Capítulo I:

Introducción

1.1. Contexto del proyecto

En la Universidad Politécnica de Montreal se utilizan dos plantas piloto de forma didáctica para enseñar a los alumnos sobre el control y la automatización de procesos industriales. Las dos plantas piloto son iguales, se componen de un carro que simula el funcionamiento de un montacargas, y cada una de ellas está compuesta por dos motores independientes que permiten los desplazamientos en X e Y de una carga. Cada planta piloto tiene un rail de 14 pies de largo que está unido al techo. Este carril asegura el desplazamiento y el seguimiento de la carga, que puede moverse por todo el rail y desciende hasta un máximo de 9 pies. Esto se logra con un motor fijo en un extremo de la planta que tira de un cable de acero. En el otro extremo está el codificador de posición horizontal, y en entre ambos está el carro que está enganchado al cable. Junto con el codificador horizontal, el conjunto tiene otros dos codificadores que permiten medir la posición vertical y el ángulo de la carga con respecto a la vertical. En las figuras siguientes se observan dos vistas del carro junto con los codificadores de la planta piloto 1 y el conjunto completo de la planta piloto 2 [1]:

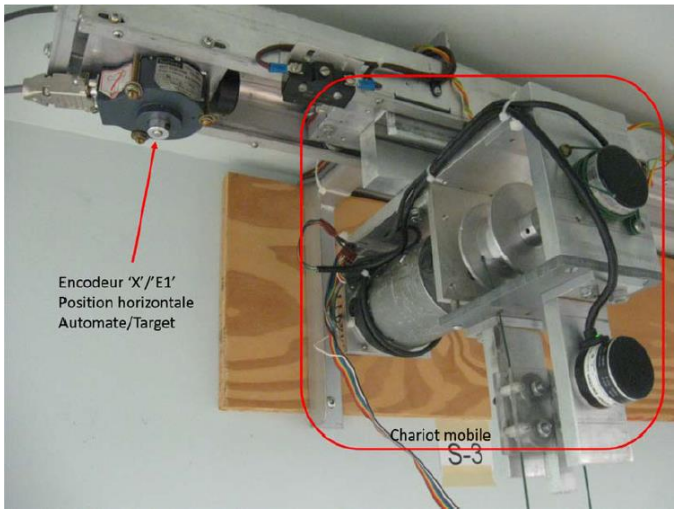

Figura 1: Carro y codificador horizontal de la planta piloto 1

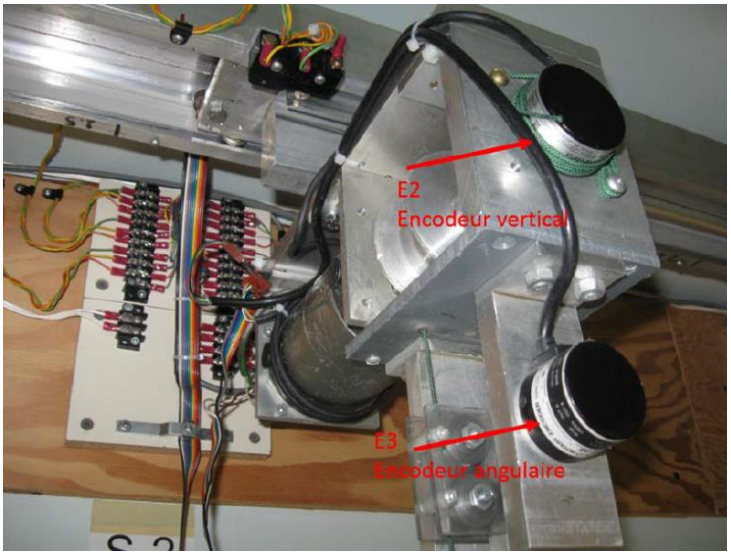

Figura 2: Carro y codificadores vertical y angular de la planta piloto 1

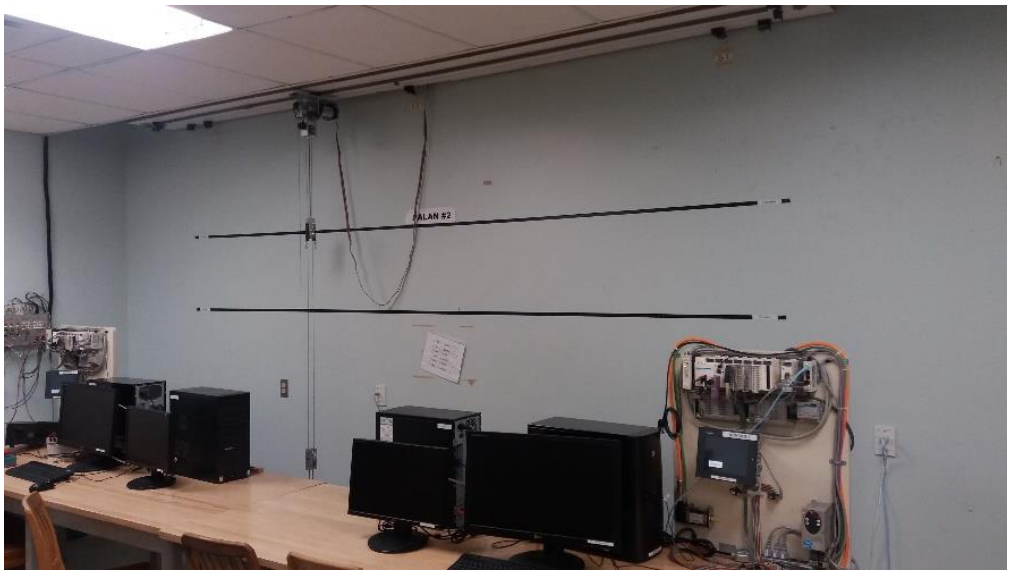

Figura 3: Conjunto de la planta piloto 2

En el esquema siguiente podemos observar el recorrido que efectúa el carro y las zonas que limitan su desplazamiento [2]. S6 y S5 son dos interruptores que cortan la corriente del motor cuando el carro los alcanza. Los otros interruptores (S1, S2 y S3) no tienen ningún efecto sobre el motor. Los tanques SP0, SP1, SP2 y SP3 simulan el funcionamiento de una planta industrial con un montacargas, son simplemente unas cajas situadas en el suelo.

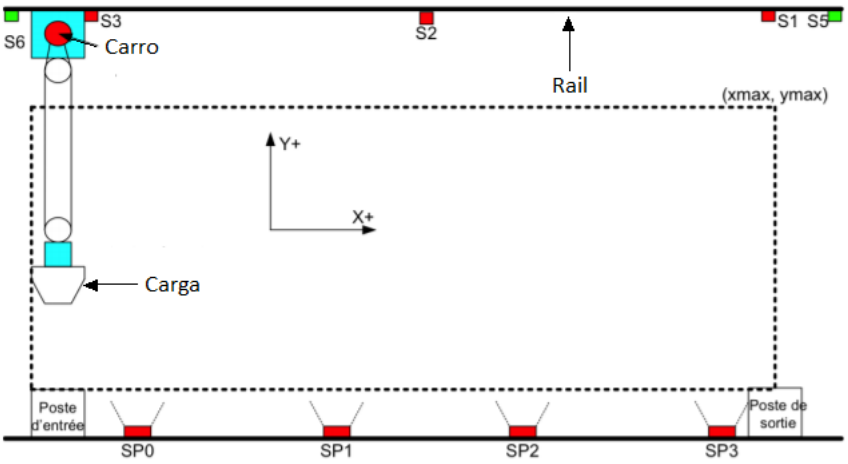

Figura 4: Esquema de la planta piloto

Las plantas se utilizan principalmente para unas sesiones de laboratorio de una asignatura, y se dividen en dos bloques: en el primer bloque se controla el carro de forma secuencial, con un autómata industrial programable (Telemecanique Modicon M340) y en el segundo con un sistema de prototipado rápido en tiempo real utilizando Matlab-Simulink-xPCTarget. El programa xPCTarget es la interfaz directa entre la planta piloto y el ordenador de control.

En vista de este sistema actual, se pretende hacer una actualización de los equipos que se han quedado obsoletos. Los motivos principales son los siguientes:

-La concepción de la planta piloto actual presenta ciertos inconvenientes: en primer lugar, el carro ligado a una cuerda no es independiente del rail que lo transporta, por lo que cuando sucede un fallo todo el conjunto debe desmantelarse. Para los técnicos esto supone un problema grave ya que se requiere la presencia de al menos dos técnicos y horas de ajuste para lograr una tensión de cable adecuada una vez está resuelto el problema. Con la estructura actual, los cables eléctricos están presentes por todo el conjunto, lo que supone una molestia para trabajar en el mantenimiento del equipo. A los problemas conceptuales se añaden otros problemas como la falta de piezas de repuesto en algunos de los equipos (principalmente los motores), y el tiempo excesivo que la planta actual requiere en tareas de reparación y mantenimiento.

-En cuanto a la seguridad del alumnado, el problema principal proviene del hecho que la oscilación de la carga no está limitada, de modo que puede llegar a golpear el propio

carro o incluso el techo si el carro se desplaza a gran velocidad. Los dispositivos de seguridad actuales son unos interruptores en cada extremo del rail que cortan la potencia de motor si el carro está demasiado cerca de dichos extremos, y una parada de emergencia manual que igualmente corta la potencia del motor horizontal. Al depender de los reflejos del usuario, es evidente que el sistema de seguridad es incompleto. Además, a pesar de los interruptores que limitan los extremos, el carro puede impactar contra un extremo del rail si éste se desplaza a gran velocidad ya que el carro tiene una masa importante y el cable de acero desliza entre las poleas.

-El programa xPCTarget se ha quedado obsoleto, y virtualmente es irremplazable, por lo que se requiere adaptar el sistema actual a un sistema integrado, utilizando un microcontrolador como principal interfaz entre la planta piloto y los ordenadores de control.

Ante estos problemas, la universidad se ha planteado remodelar la planta piloto, y sustituir el modelo actual por un taller de pórtico (figura 5) [3].

Figura 5: Taller de pórtico

Con esta estructura la parte móvil (el carro) y la parte fija (el soporte) quedan separadas, lo que facilita las tareas de mantenimiento. Junto con un cambio en la estructura, se va a proceder a cambiar los motores y codificadores actuales, y por último se debe reemplazar la solución de control actual que utiliza xPCTarget.

El grupo encargado de optimizar el montacargas está compuesto por un técnico, dos profesores y mi aportación. Este documento es referente a nuestra tarea, que se

concentra en diseñar un sistema de seguridad para la planta piloto final. Por tanto, no nos centraremos en los cambios a los motores ni en la nueva solución de control. El presente documento es referente al diseño del sistema de seguridad y no al proyecto global.

1.2. Estado del arte

Las soluciones tecnológicas que se plantean al problema se basan en dar solución al sistema de seguridad. Al tratarse de la seguridad de los alumnos que van a emplear el nuevo sistema, es la parte más importante del proyecto global. Dividiremos nuestro trabajo en diferentes etapas:

En primer lugar, nos centraremos en diseñar conceptualmente un sistema de seguridad acorde a la futura planta piloto. Se pretende diseñar un sistema de control del voltaje y la potencia del motor mediante la medición de la posición absoluta del carro. La posición se va a medir con tecnología láser, y está previsto que el sistema de control funcione mediante una placa computadora a través de Matlab-Simulink.

Una vez esté listo el diseño del sistema de seguridad, se va a proceder a realizar la comunicación entre un motor y el ordenador utilizando el software que va a reemplazar xPCTarget: Matlab-Simulink. Queremos obtener un sistema en el cual podamos enviar órdenes al motor desde el ordenador, y a su vez regular la potencia que llega al motor con un potenciómetro. Esto nos permitirá poner a prueba más adelante el sistema de seguridad, y establece la comunicación que habrá en el proyecto global entre el sistema de control y el motor.

Cuando la comunicación con el motor sea posible y podamos controlar la velocidad y el sentido de desplazamiento del carro, nos centraremos en obtener medidas de la posición absoluta del carro con tecnología láser. El software a emplear para la medición de las distancias es Arduino.

Finalmente, reuniendo todas las partes obtendremos un montaje final en una pequeña maqueta que simule el funcionamiento del futuro sistema. En este montaje pondremos a prueba el sistema de seguridad diseñado y evaluaremos su viabilidad en el sistema final.

1.3. Motivación

El proyecto se hace para dar solución a los problemas descritos en la introducción. Estos problemas se resumen en un sistema de seguridad incompleto que puede poner en peligro la salud y seguridad de los alumnos. Lo más importante es no permitir que la carga impacte contra la pared, limitando su velocidad cuando el carro se sitúa en ciertas zonas del rail. Por lo tanto, nuestro objetivo es que las próximas sesiones de laboratorio se efectúen con la mayor seguridad posible, y el menor coste. En definitiva, que los alumnos disfruten de un equipo de alta calidad para el proceso de aprendizaje en el control y la automatización de procesos industriales.

Las conclusiones de este proyecto pueden servir para otros sistemas de seguridad de plantas similares en la industria. Nuestro sistema es pequeño y de muy bajo coste pero veremos que es perfectamente funcional.

1.4. Objetivos del proyecto

Los objetivos del proyecto son:

- Diseñar un sistema de seguridad basado en el control del voltaje de un motor que desplaza un carro mediante la medición de la posición de este último.
- Implementar la comunicación entre un motor y el ordenador, pudiendo controlar el voltaje y sentido de giro del motor. La comunicación se va a efectuar mediante un módulo UART y también mediante un potenciómetro que regule el voltaje del motor.
- Medir mediante tecnología láser la posición absoluta de un carro móvil y poder manipular los datos de distancia obtenidos.
- Adquirir un montaje final que simule el funcionamiento de la futura planta piloto provista de un sistema de seguridad completo.

Capítulo II:

Implementación del sistema de seguridad

En este capítulo vamos a analizar el sistema de seguridad diseñado para la planta piloto siguiendo las diferentes partes del sistema completo. Empezaremos con un diseño conceptual del sistema requerido, analizaremos como podemos controlar el motor desde Simulink, mediremos la posición absoluta del carro con un láser y finalmente en un montaje final podremos observar todo el conjunto.

2.1. Diseño del sistema de seguridad

El objetivo principal del sistema de seguridad es asegurar que el carro no impacte con violencia los extremos del rail, por lo que debemos limitar la tensión que le llega al motor según dónde se encuentre el carro en el rail. El diseño que vamos a plantear consiste en la división del área de desplazamiento por zonas. El esquema siguiente muestra los desplazamientos permitidos y limitados según la posición del carro y su dirección:

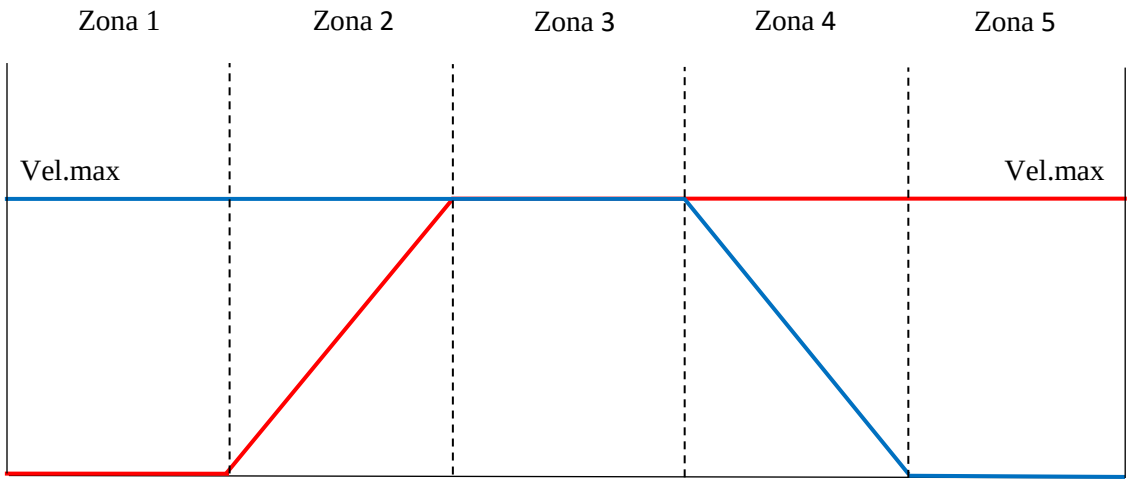

Figura 6: Esquema de la velocidad permitida en función de la posición y la dirección del carro

El trazado rojo representa la velocidad permitida hacia la izquierda, y el trazado azul la velocidad permitida hacia la derecha. Por lo que si nos acercamos a uno de los extremos a gran velocidad el carro debería frenarse en primer lugar de forma lineal (según su posición) y finalmente el carro debería frenarse por completo. Con este sistema

podemos definir las zonas más adelante de acuerdo con nuestro modelo, es decir que a base de ensayar la potencia del motor podremos delimitar las zonas de frenado para que bajo ningún caso el carro o la carga que transporta impacte contra los extremos.

En la zona 3 el carro puede desplazarse a velocidad máxima, es decir que no limitaremos su velocidad. Las zonas 2 y 4 son de frenado y serán las zonas más importantes de nuestro sistema. Últimamente las zonas 1 y 5 sirven para garantizar que el motor no recibe más voltaje en el sentido correspondiente.

Con este diseño, vamos a necesitar poder manipular el motor en los dos sentidos y medir la posición absoluta del carro para identificar la zona en la que se encuentra. Nos centramos en dar respuesta a la comunicación entre el motor y el ordenador.

2.2 Comunicación entre el motor y Matlab-Simulink

Disponemos de un pequeño motor acoplado a un carro para hacer pruebas y preparar un montaje que nos sirva para el diseño final. En las imágenes siguientes se puede observar la maqueta con la que trabajaremos: se compone de un carro movido por un pequeño motor y un rail de un metro de largo por el cual el carro se desplaza. Tenemos acceso a la tensión de alimentación del motor (cables rojo y negro).

Figura 7: Vistas de la planta y el alzado del carro

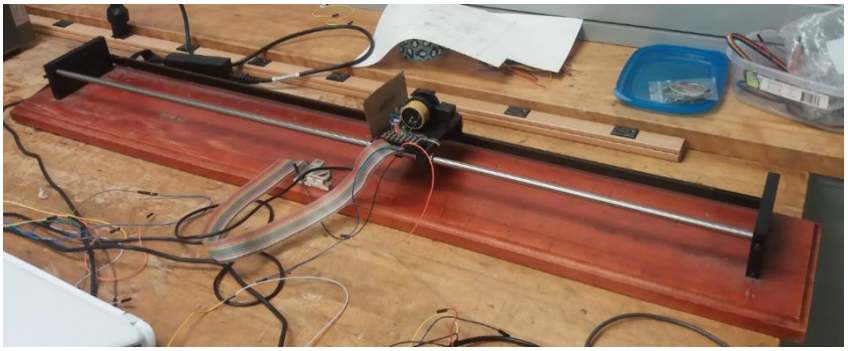

Figura 8: Vista del carril por el que se desplaza el carro

Para alimentar el motor disponemos de una fuente de alimentación de hasta 50V. En la figura siguiente podemos ver una vista frontal de la fuente de alimentación.

Figura 9: Vista frontal de la fuente de alimentación

Nuestro primer objetivo es poder controlar el sentido de giro del motor. Para ello empleamos un módulo: una placa de control de motor DC.

2.2.1 Implementación de la placa de control de motor DC

Para el control de la velocidad de rotación y el sentido del motor vamos a emplear un módulo integrado con dos BTN8982TA. Se puede encontrar su hoja de características en [4]. El BTN8982TA es un medio puente integrado de alta corriente que se usa para accionamientos de motores. Es un módulo específico para aplicaciones con cambios en

el sentido de giro de un motor y regulación por PWM, *Pusle width modulation* (explicaremos más adelante lo que es una señal PWM). Entre las numerosas características del módulo destacamos la protección contra cortocircuitos y sobrecalentamientos, limitación de la corriente y protección contra polaridad inversa. Soporta un máximo de 40V de entrada, y una corriente media de 30A restringida debido a la disipación de potencia de los puentes integrados.

Con este módulo se pueden manejar dos motores de corriente continua unidireccionales o un solo motor bidireccional. Los motores se manejan mediante PWM o modulación por ancho de pulsos. Consiste en modificar el ciclo de trabajo de una señal periódica de forma que podemos controlar la potencia entrante en un motor. Vamos a trabajar con ondas cuadradas, a las que modificaremos el ciclo de trabajo para obtener mayor o menor potencia en el motor. En la imagen siguiente podemos ver 4 ejemplos de una señal cuadrada trabajando a distintos ciclos [5]:

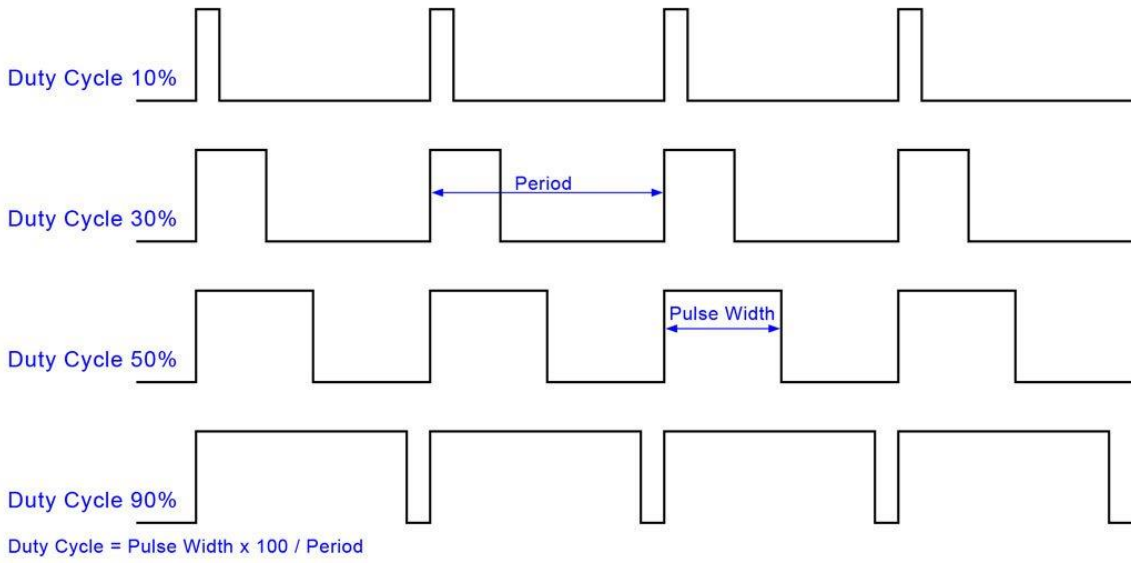

Figura 10: Representación de 4 valores diferentes de ciclos de trabajo (Duty cycle) de una señal cuadrada

Como podemos observar en la imagen tenemos una señal cuadrada con una frecuencia determinada a la que vamos modificando su anchura (Pusle Width) en función del ciclo de trabajo. Cuanto mayor es el ciclo de trabajo mayor será el voltaje en el motor y a mayor velocidad podremos desplazar el carro.

El módulo es capaz de controlar motores con señales PWM de alta frecuencia, de hasta 30 kHz.

La placa es la siguiente: un shield de control de motor DC con BTN8982TA [6]:

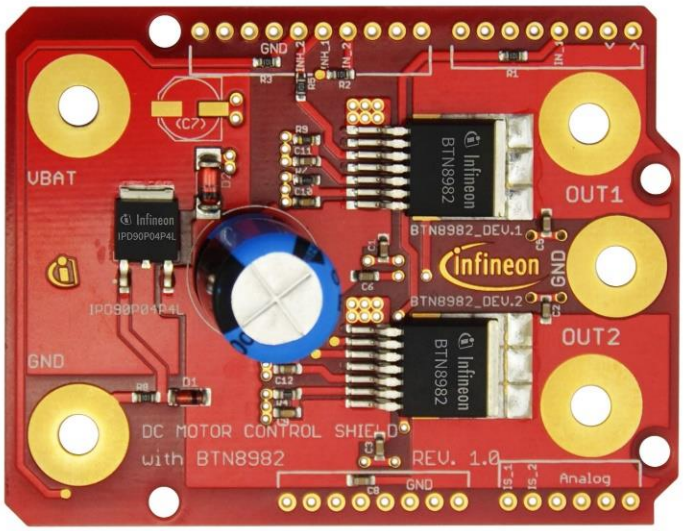

Figura 11: Placa de control de motor DC con BTN8982TA de la marca infineon

La placa está diseñada para integrarse con una placa computadora de tipo Arduino Uno o un microcontrolador XMC1100, el fabricante (Infineon) proporciona códigos para distintas aplicaciones posibles del módulo en estas placas. El código proporcionado por la compañía se puede encontrar en [6]. Nosotros no vamos a utilizar este código, pero siguiendo la hoja de características y las posibles aplicaciones del módulo vamos descifrando las funciones de cada pin que utilizaremos. Por lo que haciendo un trabajo de traducción del código, destacamos las siguientes funcionalidades:

Los terminales Vbat y GND sirven para la alimentación de la placa, que según la hoja de características de la misma tendremos que alimentar entre 8 y 18V. Los pins INH_1 e INH_2 accionan el modo de ahorro de energía cuando no reciben alimentación, pero en nuestra aplicación no los vamos a usar por lo que mantendremos alimentadas esas entradas. Los pins IN_1 e IN_2 activan un sentido u otro del motor, requieren una señal PWM para estar activos y mandar un voltaje al motor. En función del ciclo de trabajo del PWM podremos controlar la velocidad de giro del motor, y con este sistema con dos pins podremos controlar su sentido. Los pins IS_1 e IS_2 sirven para medir la

intensidad que circula por los puentes, pero no los vamos a utilizar en nuestra aplicación. Finalmente, los terminales OUT1 y OUT2 son nuestras salidas que van al motor. El terminal situado entre ambos denominado GND no se utiliza a no ser que queramos usar el módulo para aplicaciones unidireccionales.

En la tabla siguiente resumimos las funciones de cada pin:

PIN	Función
GND	Tierra
IN_1	Activa un sentido del motor, requiere una señal PWM
IN_2	Activa el otro sentido del motor, requiere una señal PWM
INH_1	Suspende la corriente del motor en el sentido 1 cuando está alimentado
INH_2	Suspende la corriente del motor en el sentido 2 cuando está alimentado
OUT_1	Salida del módulo y entrada al motor
OUT_2	Salida del módulo y entrada al motor
IS_1	Mide la intensidad circulando por el medio puente 1
IS_2	Mide la intensidad circulando por el medio puente 2
Vbat	Alimentación de la placa (entre 8 y 18V)

Tabla 1: Funciones de los diferentes pins de la placa de control DC

Para entender mejor el funcionamiento de la placa en la imagen siguiente podemos observar el circuito eléctrico que sirve para una aplicación bidireccional de un motor [4].

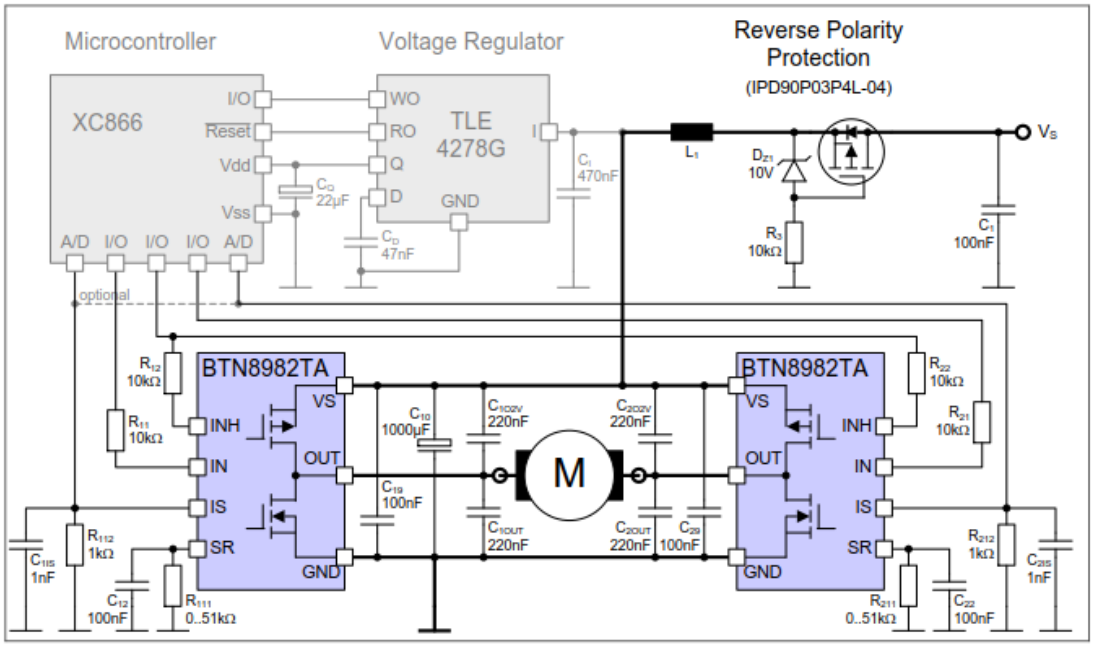

Figura 12: Esquema del circuito eléctrico de la placa de control de motor DC con BTN8982TA

No nos centramos en la aplicación sugerida por el fabricante por lo que podemos ignorar el microcontrolador XC866 y el regulador de voltaje 4278G conectados a la placa.

Destacamos los dos medio puentes integrados BTN8982TA que funcionando conjuntamente permiten el giro del motor en un sentido u otro en función de la entrada IN de cada puente. Es importante que las dos entradas IN de cada puente no estén alimentadas a la vez, sino que sea una u otra quién determine el sentido de giro del motor.

Los dos puentes configuran lo que denominamos un puente en H. Es un circuito electrónico que permite aplicar una tensión positiva en el motor de dos formas distintas, haciendo girar el motor hacia los dos sentidos. Esto se consigue con el cambio de la polaridad, es decir, cambiando el sentido con el que la corriente pasa a través del motor [7].

El medio puente BTN8982TA contiene un transistor MOSFET de tipo p y otro de tipo n que permiten generar los dos voltajes sobre el motor. En función de la alimentación de las entradas de los medios puentes, obtendremos una tensión positiva generando el giro del motor en un sentido u otro.

Lo más importante a subrayar es que los medio puentes se controlan desde los pins IN e INH. Para girar el motor en un sentido debemos mantener una entrada IN alimentada y la otra sin tensión, al mismo tiempo que alimentamos las dos entradas INH. Para cambiar el sentido de giro tendremos que invertir la alimentación de las entradas IN.

En vistas de las características de la placa, vamos a construir un modelo que nos permita controlar la velocidad y el sentido de giro del motor. Para ello, utilizaremos un microcontrolador: el STM32F4DISCOVERY.

2.2.2 Control de velocidad y sentido de giro de un motor con un microcontrolador, un potenciómetro y un módulo ADC

Nuestro objetivo es poder manejar el motor del carro desde el ordenador, más específicamente desde un modelo de Matlab-Simulink. En este apartado damos

respuesta a la comunicación entre el motor y el modelo Matlab-Simulink, con una aplicación de control de velocidad y sentido de giro con un potenciómetro que nos servirá más adelante para nuestro sistema de seguridad.

El microcontrolador STM32F4DISCOVERY nos servirá como interfaz entre nuestro modelo de Matlab-Simulink y la placa de control de motores DC. Para emplear el software de Matlab con nuestro microcontrolador empleamos una librería específica para nuestro microcontrolador. La librería "Waijung blockset" nos permite generar el código C traducido de un modelo de Matlab-Simulink a nuestro microcontrolador. Esta librería implementa bloques de Simulink que funcionan conjuntamente con el microcontrolador. En la fuente [8] se pueden encontrar múltiples aplicaciones de esta librería junto con las definiciones y características de los bloques que se pueden emplear. En esa misma fuente se puede descargar la librería, para este proyecto se ha utilizado la versión de Matlab 2015a para todos los modelos creados.

Por lo que con esta herramienta podemos cargar un programa al microcontrolador con un simple modelo de Matlab-Simulink. El programa que cargaremos en cada ocasión lo definiremos como "Target".

En la imagen siguiente podemos ver el microcontrolador que empleamos [9]:

Figura 13: Microcontrolador STM32F4DISCOVERY

Se puede encontrar la hoja de características de la placa STM32F4DISCOVERY en la fuente [9]. De esta hoja destacamos las siguientes características: tiene 1MB de memoria flash, 192KB de RAM; se alimenta a través del bus USB o una fuente de alimentación externa de 5V; sensores de audio y movimiento; un botón de reset de color azul visible en la figura anterior; alimentación de 5V y 3,3V; 8 puertos de entradas/salidas de 16 bits y un puerto de 12 bits; 14 timers, 6 módulos UART, 3 puertos SPI, 3 puertos I2C, 3 módulos ADC y 2 módulos DAC.

Con esta placa junto con la placa de control de motor DC queremos controlar la velocidad y el sentido de giro del motor con un potenciómetro. Por lo tanto necesitaremos generar pulsos variables que enviaremos a los pins IN_1 e IN_2 en función del sentido de giro deseado. Con la ayuda de un potenciómetro haremos variar un voltaje de referencia Vref para poder pasar de un sentido de giro a otro, y siendo las tensiones Vmáx y Vmin generadas por el potenciómetro los puntos extremos de trabajo del motor. Vmáx será el voltaje que haga trabajar a máxima potencia el motor en un sentido, y Vmin en el otro sentido.

Para que sea lo más sencillo posible, vamos a emplear tensiones negativas para un sentido de giro y tensiones positivas para el otro, siendo 0V el punto intermedio.

El potenciómetro que vamos a emplear es lineal, en la figura siguiente podemos observar una vista de la planta y una vista inferior del potenciómetro:

Figura 14: Vistas de la planta e inferior del potenciómetro lineal

Su funcionamiento es muy sencillo: el potenciómetro varía el voltaje de referencia Vref conectado al terminal superior (cable rojo) en función de la rotación del mismo.

Utilizaremos un Vref de 3V aproximadamente proporcionados por el microcontrolador, e iremos variando los 3V para ir cambiando el sentido de giro y la potencia con la que gira el motor. Como hemos mencionado previamente, para hacerlo más sencillo empleamos tensiones negativas para un sentido de giro y tensiones positivas para el otro. De forma que de 0V a 1,5V vamos en una dirección (siendo 1,5V el 100% del ciclo de trabajo del PWM) y de 0V a -1,5V en la otra dirección (siendo -1,5V el 100% del ciclo de trabajo del otro PWM).

El modelo que utilizamos de Matlab-Simulink es el siguiente:

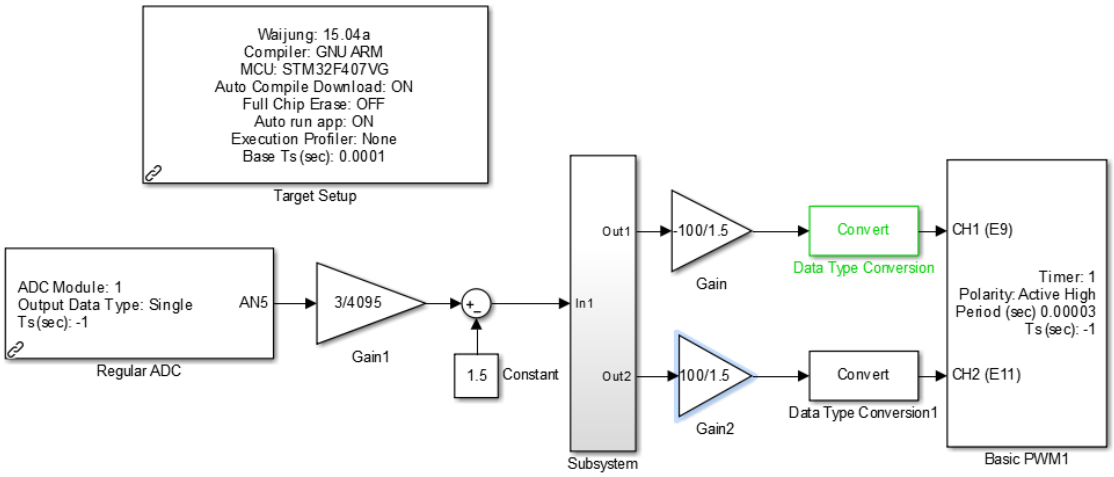

Figura 15: Modelo Simulink para el control de velocidad y sentido de giro de un motor con un potenciómetro

En primer lugar, vamos a describir las funcionalidades de los principales bloques:

-Target Setup: Es el bloque más importante de la librería "Waijung", genera el código en C para el microcontrolador deseado, por lo que es la interfaz directa entre el microcontrolador y el modelo Simulink. Por lo tanto, es el bloque que genera y carga la "Target" en el microcontrolador. Podemos seleccionar el compilador a utilizar junto con otras funciones como ejecutar el programa una vez esté descargado en el microcontrolador. Este bloque funciona para nuestro microcontrolador por defecto por

lo que no cambiaremos su configuración en ninguno de los modelos que vamos a ver en este proyecto.

-Regular ADC: Es un bloque de la librería “Waijung”, implementa un convertidor analógico/digital utilizando uno de los 3 convertidores disponibles del microcontrolador. El funcionamiento es el siguiente: el microcontrolador recibe un voltaje por el pin deseado y lo transforma en un dato de tipo “single” (es un decimal) que va desde 0 hasta 4095, siendo 4095 el voltaje máximo.

-Basic PWM: También es un bloque de la librería “Waijung”, genera hasta 4 señales PWM de una frecuencia determinada en distintas salidas del microcontrolador. Cada generador de pulsos recibe un ciclo de trabajo de entre 0 y 100, siendo estos ciclos las entradas del bloque. En este modelo empleamos 2 generadores de pulsos de 33,333 kHz que enviamos a los pins E9 y E11.

-Convert: Concierte los datos de tipo “single” en datos de tipo “double” que utiliza el bloque Basic PWM.

Por lo que el funcionamiento es el siguiente: el módulo ADC convierte el voltaje que variamos con el potenciómetro a un dato de tipo “single” entre 0 y 4095. Convertimos el dato para obtenerlo en V multiplicando el valor tomado por 3V/4095. Le restamos a ese valor 1,5V para lograr la situación deseada: de 0 a 1,5V una dirección y de 0 a -1,5V la otra, siendo 1,5V y -1,5V los puntos de trabajo máximo del motor. El subsistema del modelo es el encargado de activar uno de los pins del PWM o el otro en función de si el voltaje recibido es superior o inferior a 0V.

En la figura siguiente observamos el subsistema en detalle:

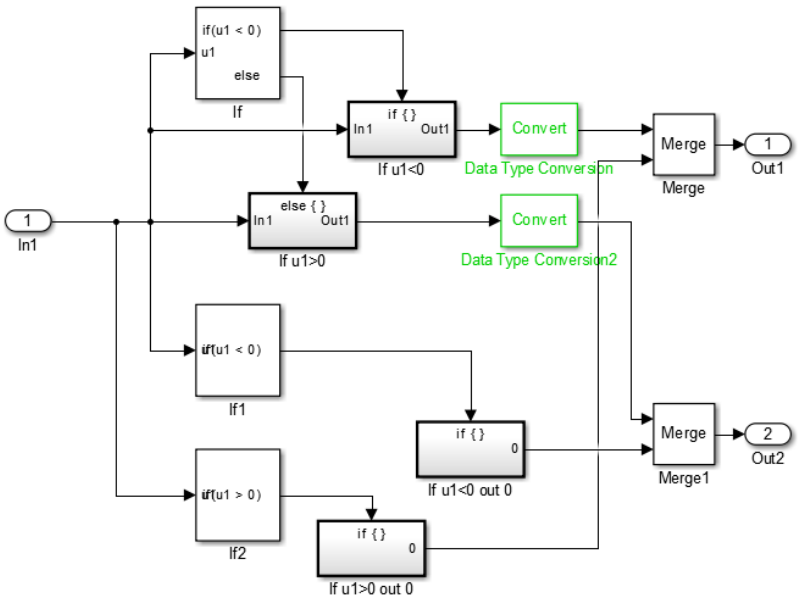

Figura 16: Subsistema del modelo Simulink para el control de velocidad y sentido de giro de un motor con un potenciómetro

Cuando la entrada que llamaremos u1 es negativa, u1 sale por la salida out1 y out2 pasa a ser 0. Y cuando la entrada es positiva, sale por out2 y out1 se pone a 0. Es esencial la utilización de los bloques Merge ya que nos permiten dar como salida al bloque la última compilación efectuada en el modelo. Por lo que cuando se cumple una de las condiciones y se activa el subsistema correspondiente, el bloque Merge tiene en cuenta que esa fue la última condición validada. Esto nos permite mantener sólo una de las salidas con un valor y mantener a 0 la otra.

Una vez sale el voltaje por la salida correspondiente multiplicamos ese voltaje por una ganancia para obtener un valor entre 0 y 100. En el caso de la salida out1 multiplicamos por 100/1,5 la salida out2 y por -100/1,5 la salida out1 porque corresponde a las entradas negativas. El bloque Basic PWM necesita un valor entre 0 y 100 para generar una señal PWM con un ciclo de trabajo igual a ese mismo valor. Por lo que en los pins E9 y E11 estamos generando dos señales PWM con ciclos de trabajo dependientes del voltaje que enviemos por el potenciómetro, y según este modelo no podremos obtener las dos señales al mismo tiempo.

Así, si por ejemplo enviamos un voltaje de 3V desde el potenciómetro, por la salida out2 salen 1,5V y el bloque Basic PWM genera una señal PWM con un ciclo de trabajo del 100% en el pin E11. Si enviamos un voltaje de 0,5V desde el potenciómetro, por la salida out1 salen -1V y el bloque Basic PWM genera una señal PWM con un ciclo de trabajo del 66,67% en el pin E9.

Por último, montamos nuestro circuito:

-La parte correspondiente al mando (potenciómetro) es muy sencilla, conectamos a tierra y a Vref=3V el potenciómetro y su pin intermedio va conectado al pin A5 del microcontrolador que resulta ser el módulo ADC.

-Conectamos los pins E9 y E11 a IN_1 e IN_2 de la placa de control, y alimentamos los pins INH_1 e INH_2 a los 3V proporcionados por el microcontrolador para no accionar el modo de ahorro de energía en ningún momento.

En las siguientes imágenes podemos observar un conjunto del montaje:

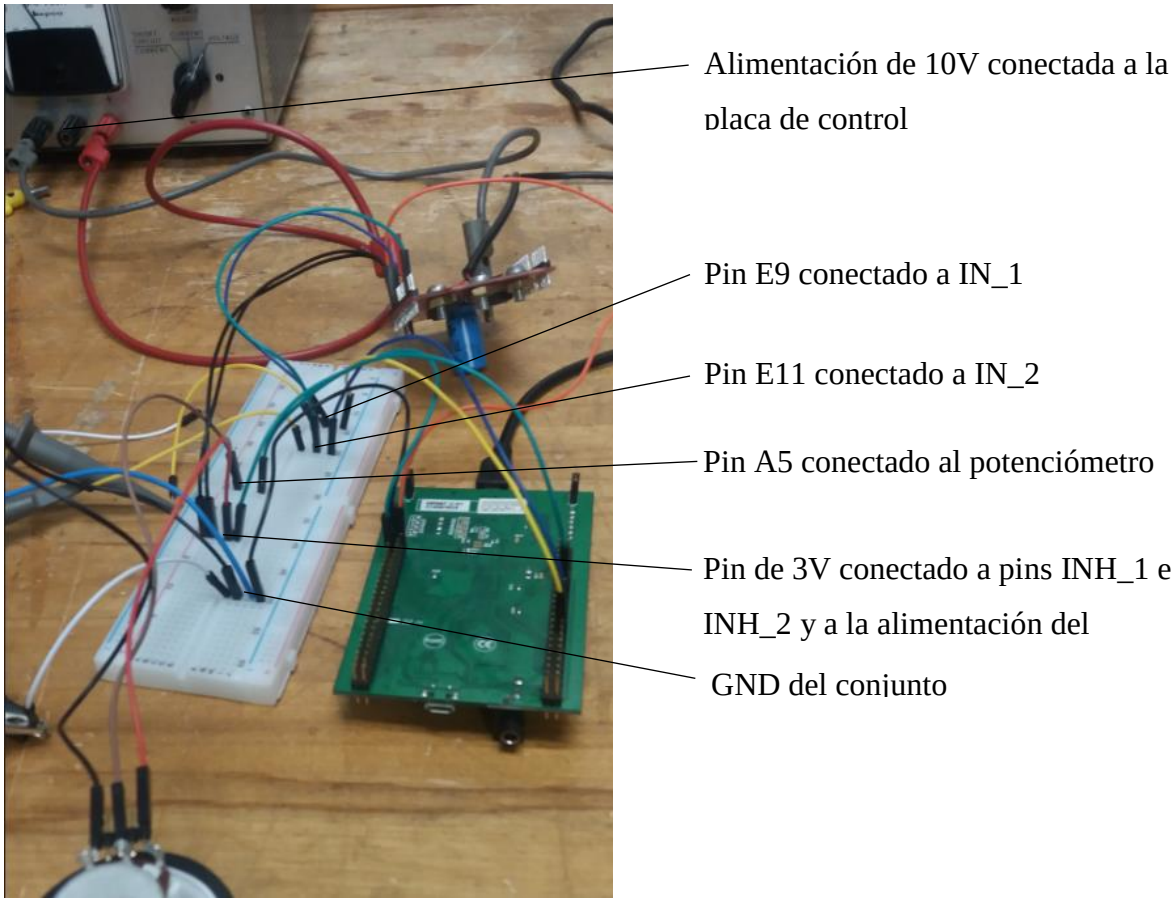

Figura 17: Montaje para el control de velocidad y sentido de giro con un potenciómetro (vista 1)

Tomamos otra fotografía para ver en detalle las conexiones en la placa de control de motor DC:

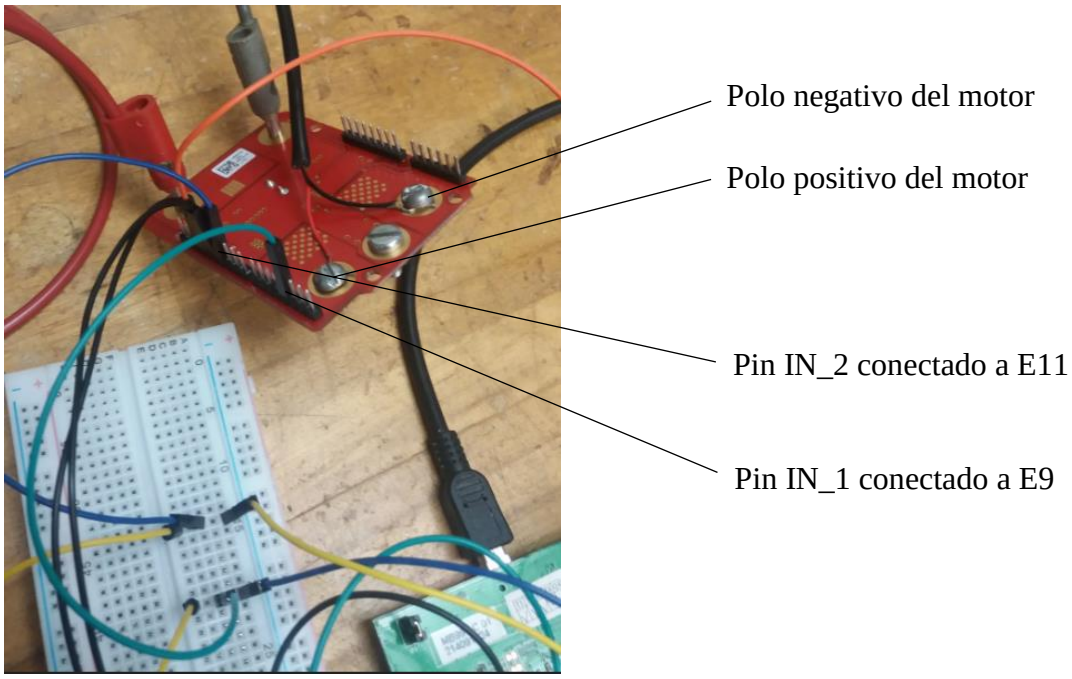

Figura 18: Montaje control de velocidad y sentido de giro con un potenciómetro (vista 2)

Con la ayuda de un osciloscopio verificamos que estamos generando las señales deseadas, conectamos las dos tomas del osciloscopio a los pins E9 y E11. El canal 1 del osciloscopio (amarillo) está conectado al pin E11 y el canal 2 (azul) al pin E9. Para valores positivos de entrada (superiores a 1,5V) deberíamos generar un PWM en el pin E11 y para valores negativos (inferiores a 1,5V) en el pin E9. Posteriormente alimentaremos el motor para verificar que somos capaces de mover el carro a distintas velocidades y cambiando el sentido.

En las dos imágenes siguientes observamos las dos situaciones posibles: generando un PWM en el pin E11 y generando un PWM en el pin E9.

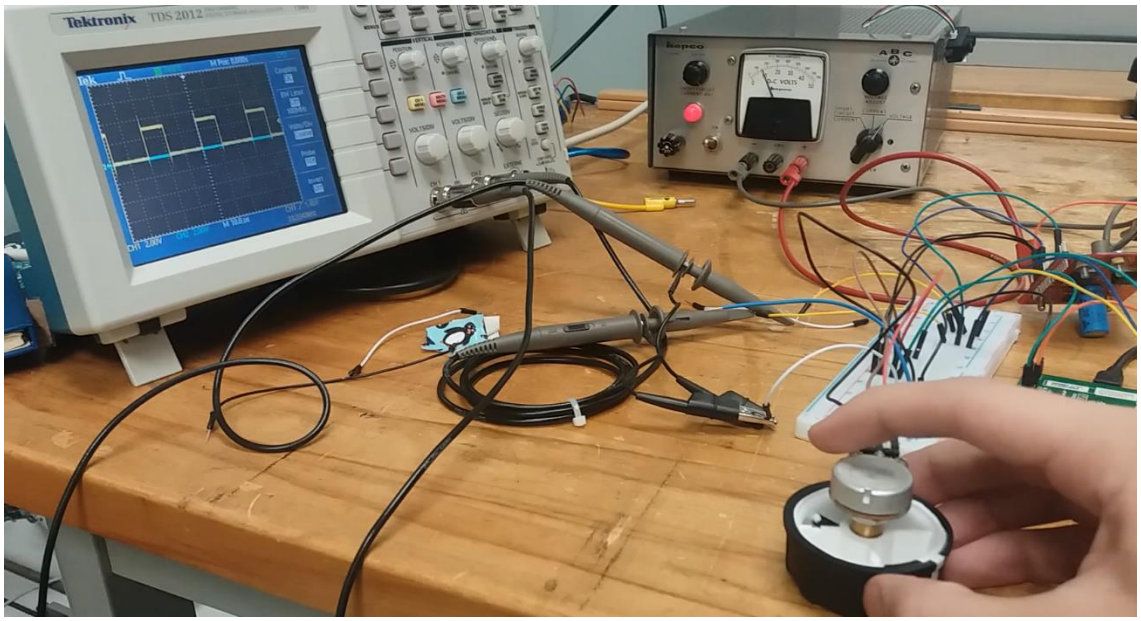

Figura 19: Generador de pulsos en el pin E11 cuando el potenciómetro es girado en sentido horario

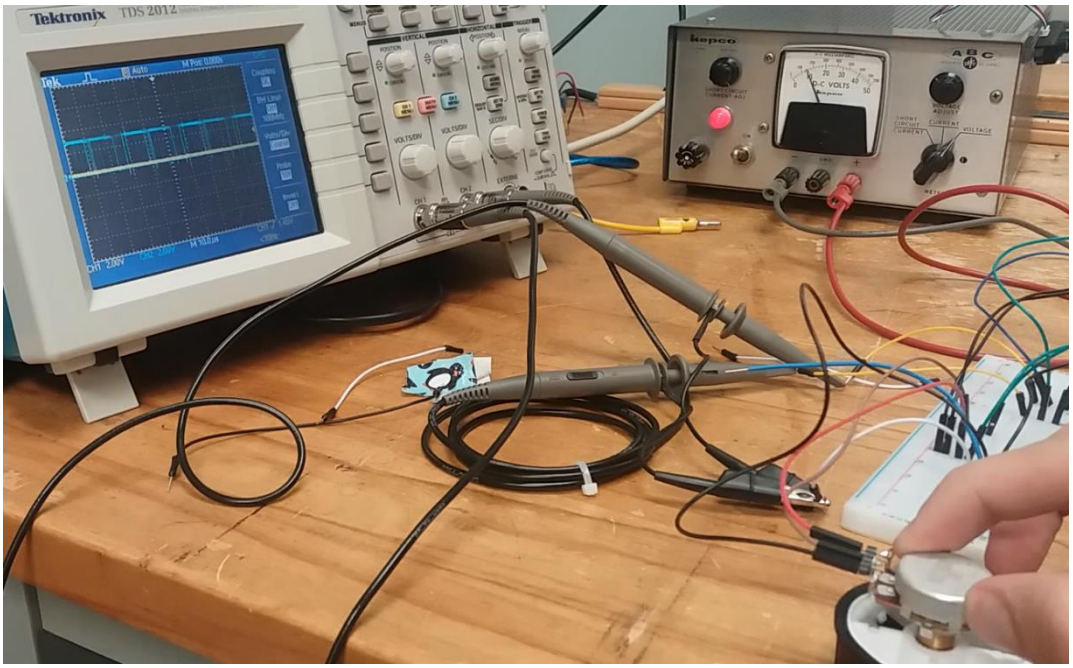

Figura 20: Generador de pulsos en el pin E9 cuando el potenciómetro es girado en sentido antihorario

Lo que vemos en las imágenes corresponde con el modelo, ya que al girar el potenciómetro en sentido horario estamos disminuyendo la resistencia del potenciómetro y el voltaje aumenta progresivamente. En nuestro modelo de Simulink el voltaje que entra al subsistema pasa a ser mayor que 0 y por lo tanto la señal se envía al pin E11.

Del mismo modo, al girar el potenciómetro en sentido antihorario disminuimos el voltaje hasta 0V como tope, lo que hace que en el subsistema entren voltajes negativos y generemos pulsos en el pin E9.

De este modo, somos capaces de mover el carro en un sentido u otro en función del giro que manejamos sobre el potenciómetro.

Hemos logrado realizar un control de velocidad y sentido de giro de un motor con un microcontrolador y un potenciómetro. En esta aplicación, nuestro modelo funciona con un tiempo de muestreo definido de 0,0001 segundos, y la "Target" se ejecuta en el microcontrolador indefinidamente. Queremos ir más allá y poder manejar los datos del modelo Simulink a tiempo real, para ello necesitaremos tener dos modelos Simulink: una "Target" y un "Host" donde poder analizar datos del programa a tiempo real. A tal fin, vamos en primer lugar a sustituir el mando generado por el potenciómetro por un mando dirigido por el ordenador, con un módulo UART (Universal Asynchronous Receiver-Transmitter). Y después con este módulo UART podremos recuperar datos a tiempo real desde nuestro modelo "Target" de Simulink.

2.2.3 Control de velocidad y sentido de giro de un motor con un módulo UART

Primeramente, empleando el modelo anterior vamos a sustituir la parte del mando generado por el potenciómetro para poder generar un mando a tiempo real desde el propio Simulink. Para ello requerimos de un módulo UART que nos permita enviar a tiempo real datos desde Simulink que reciba el microcontrolador. Así, tendremos un programa que denominaremos "Host" que enviará datos a través del módulo UART a la "Target" del microcontrolador.

El módulo UART que vamos a emplear es el siguiente:

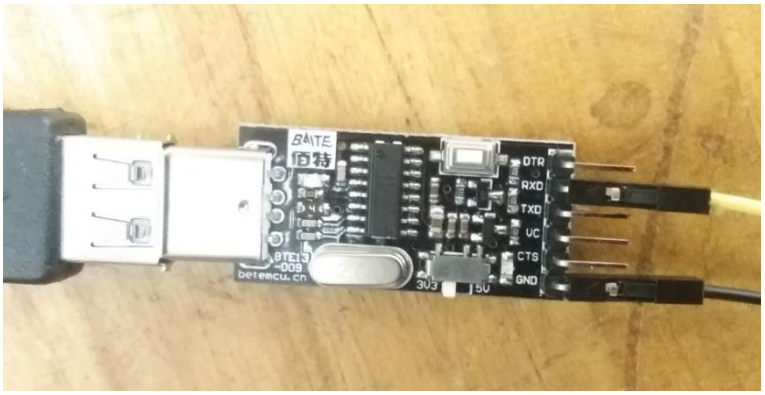

Figura 21: Modelo Simulink "Target" para el control de velocidad y sentido de giro de un motor

Conectaremos el módulo UART al ordenador mediante un USB y al mismo tiempo al microcontrolador. En nuestro modelo el cable que aparece en la imagen conectado a Rx irá conectado a Tx ya que queremos transmitir desde el UART al microcontrolador.

Los modelos Matlab-Simulink que vamos a emplear son los siguientes:

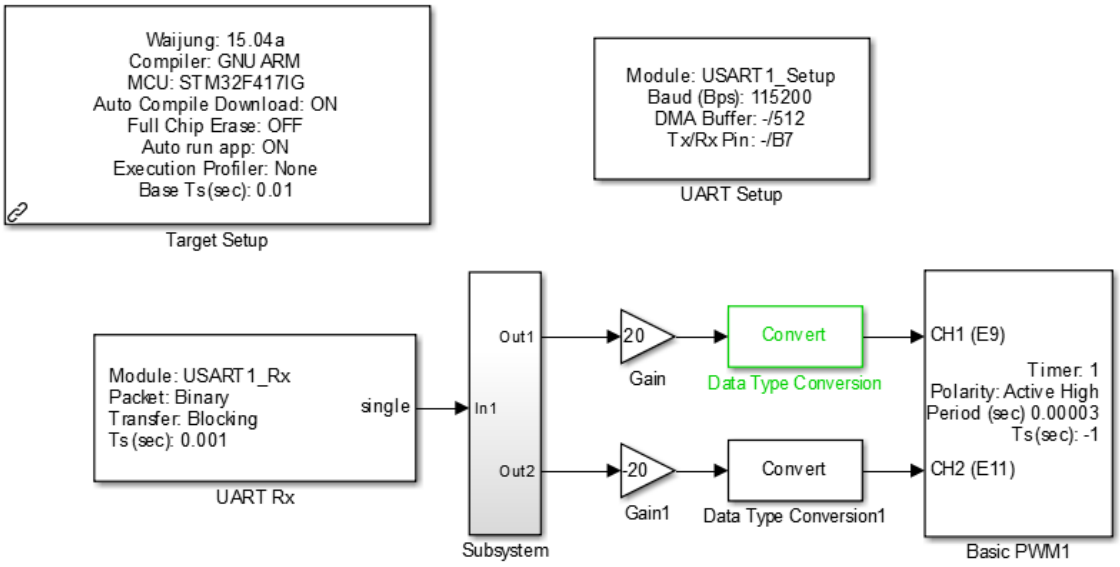

Figura 22: Modelo Simulink "Target" para el control de velocidad y sentido de giro de un motor

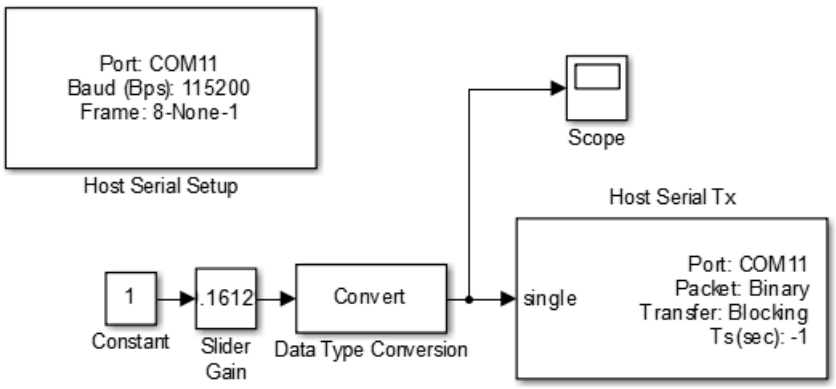

Figura 23: Modelo Simulink "Host" para el control de velocidad y sentido de giro de un motor

En el modelo de la "Target" cabe destacar que hemos remplazado el módulo ADC por un bloque denominado UART Rx y se ha añadido el bloque UART Setup.

-UART Setup: Este bloque está proporcionado por la librería de Waijung y nos permite configurar un módulo UART si requerimos de una aplicación que necesite enviar o recibir datos mediante un protocolo UART. Configuramos el bloque para únicamente recibir datos, con una velocidad de transmisión de 115200 bits por segundo y el pin B7 será el que reciba los datos.

-UART Rx: Este bloque también está proporcionado por la librería de Waijung, se utiliza para recibir datos de un protocolo UART. El bloque recibe los datos del módulo UART seleccionado y procesa los datos. En nuestro caso recibiremos datos de tipo "single", y seleccionamos el modo Blocking que obliga al proceso de compilación a esperar a recibir el paquete de datos completo antes de ejecutar los otros bloques. Para evitar problemas en este sentido, aumentamos el tiempo de muestreo del bloque UART Rx para que los datos recibidos estén disponibles con mayor frecuencia.

El modelo del "Host" es el que ejecutaremos a tiempo real. Nos encontramos con dos bloques proporcionados nuevamente por la librería Waijung:

-Host Serial Setup: Este bloque es parecido al UART Setup pero está diseñado específicamente para ser utilizado en un programa con la funcionalidad de un "Host". En este bloque configuramos el puerto COM del ordenador (en nuestro caso el COM11) que vamos a emplear para la comunicación del protocolo UART y la velocidad de la transmisión (115200 bits/s).

-Host Serial Tx: Este bloque es muy parecido al UART Rx, tiene las mismas funcionalidades pero transmite los datos en vez de recibirlos. Tenemos que asegurarnos que hemos puesto correctamente el puerto COM, y una vez hecho esto la comunicación a través del módulo UART debería funcionar.

El funcionamiento de los modelos es muy parecido al que hemos visto en el apartado anterior. Vamos a enviar una ganancia entre -5 y 5 desde el "Host" al programa del microcontrolador a tiempo real con el bloque Slider Gain. Para valores comprendidos entre 0 y 5 iremos en un sentido y para valores entre 0 y -5 iremos en el otro. Cabe destacar que hemos cambiado los signos dentro del subsistema con respecto al apartado anterior, para notar si hay grandes cambios. Por lo que para valores de entrada positivos la entrada sale hacia el pin E9 y para valores de entrada negativos la entrada sale hacia el pin E11.

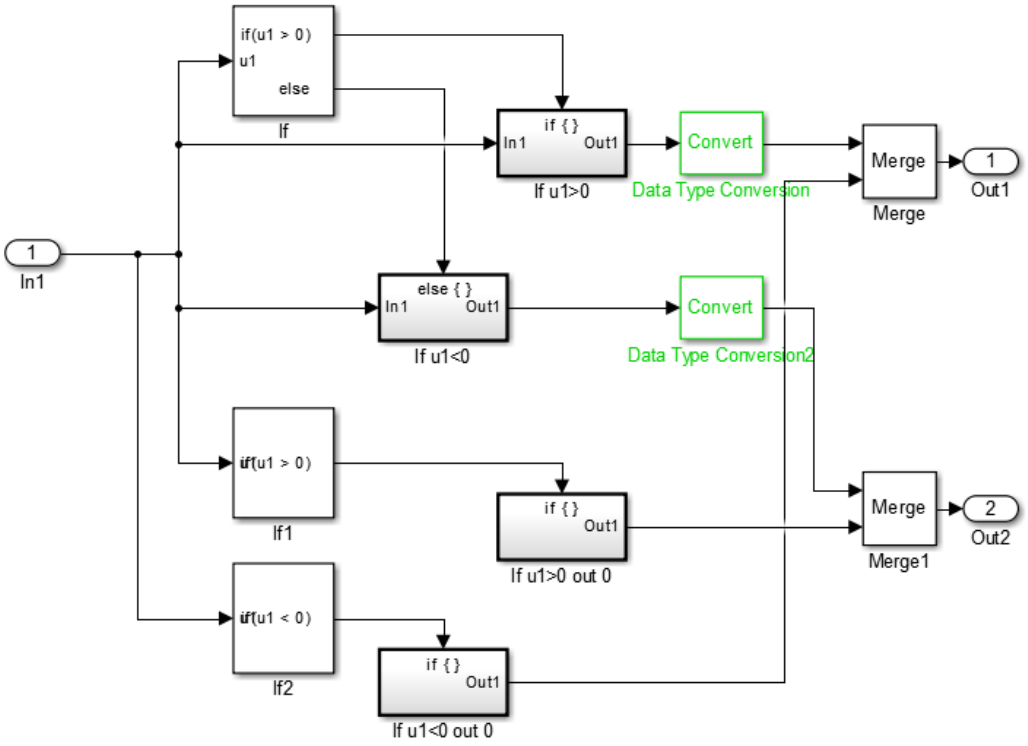

Figura 24: Subsistema modelo Simulink para el control de velocidad y sentido de giro de un motor con un potenciómetro

Las ganancias por las que multiplicamos la salida del subsistema han cambiado de valor y signo con respecto al apartado anterior, para poder enviar al bloque Basic PWM

valores entre 0 y 100. Como nos entran constantes entre -5 y 5, multiplicamos sus valores por 20 (si entra un número positivo) y -20 (si entra un número negativo).

Ahora los valores de entrada negativos corresponden al pin E11 y los positivos al pin E9.

Montamos el circuito: las conexiones referentes a la placa de control y los pins del microcontrolador no varían, pero en esta ocasión prescindimos del potenciómetro y en su lugar únicamente conectamos el pin B7 del microcontrolador al pin Tx del módulo UART. De modo que transmitimos a través del UART hacia el microcontrolador.

En la siguiente imagen podemos ver una fotografía del conjunto del montaje:

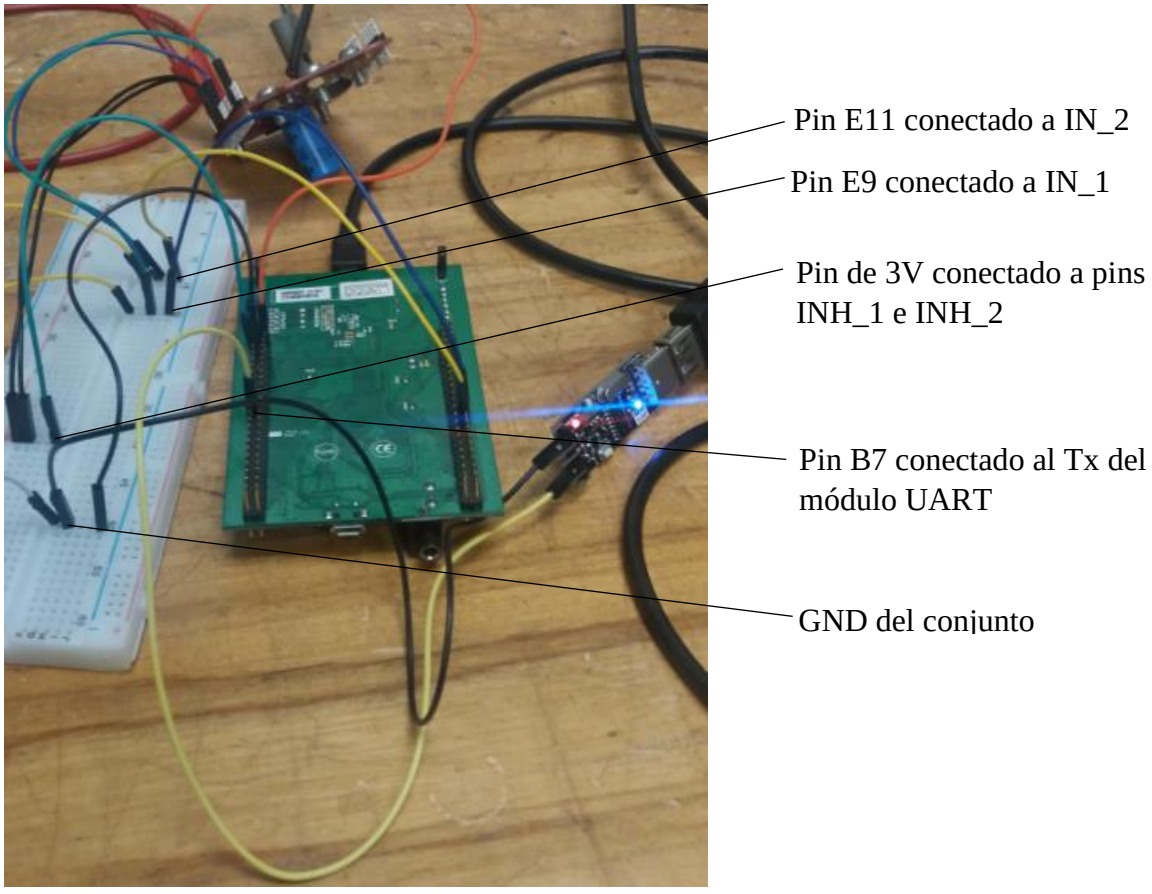

Figura 25: Montaje control de velocidad y sentido de giro con un módulo UART

Siguiendo con el ejemplo del apartado anterior, verificamos con el osciloscopio que estamos generando las señales deseadas. En las dos imágenes siguientes vemos dos posiciones del Slider Gain distintas y las señales resultantes en los pins E9 y E11. Como en el apartado anterior, la señal en azul (CH2) corresponde al pin E9 y la señal en amarillo (CH1) al pin E11.

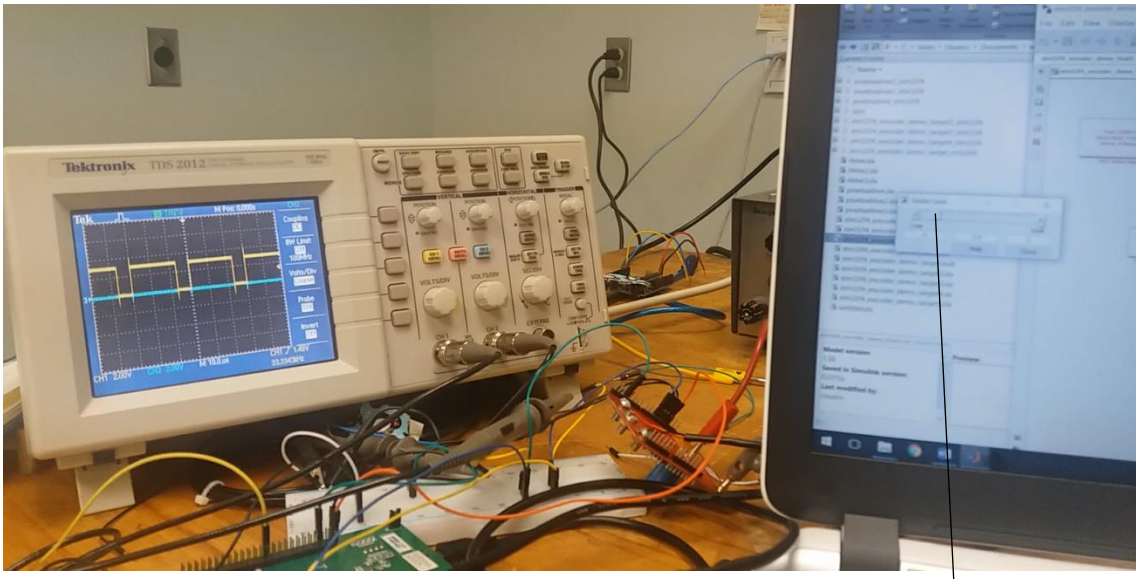

Figura 26: Generador de pulsos en el pin E11 cuando el Slider Gain da valores negativos

Posición del Slider Gain

En el modelo "Host" de Simulink que se está ejecutando a tiempo real podemos observar lo siguiente:

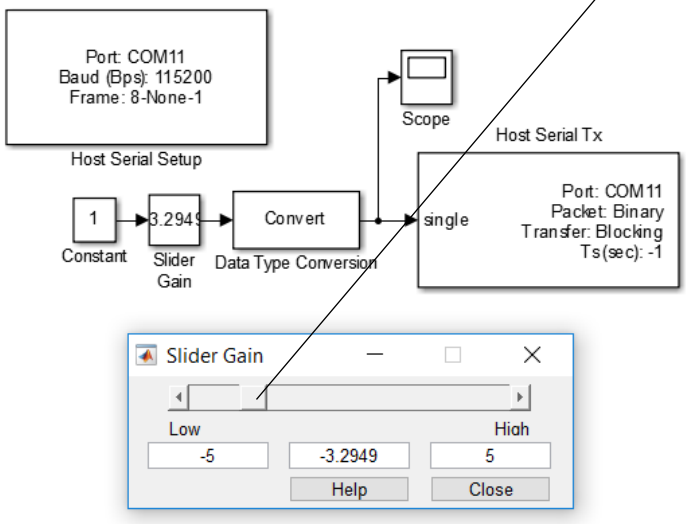

Figura 27: Modelo "Host" de Simulink ejecutándose cuando el Slider Gain da valores negativos

Para la generación de pulsos en el pin E9:

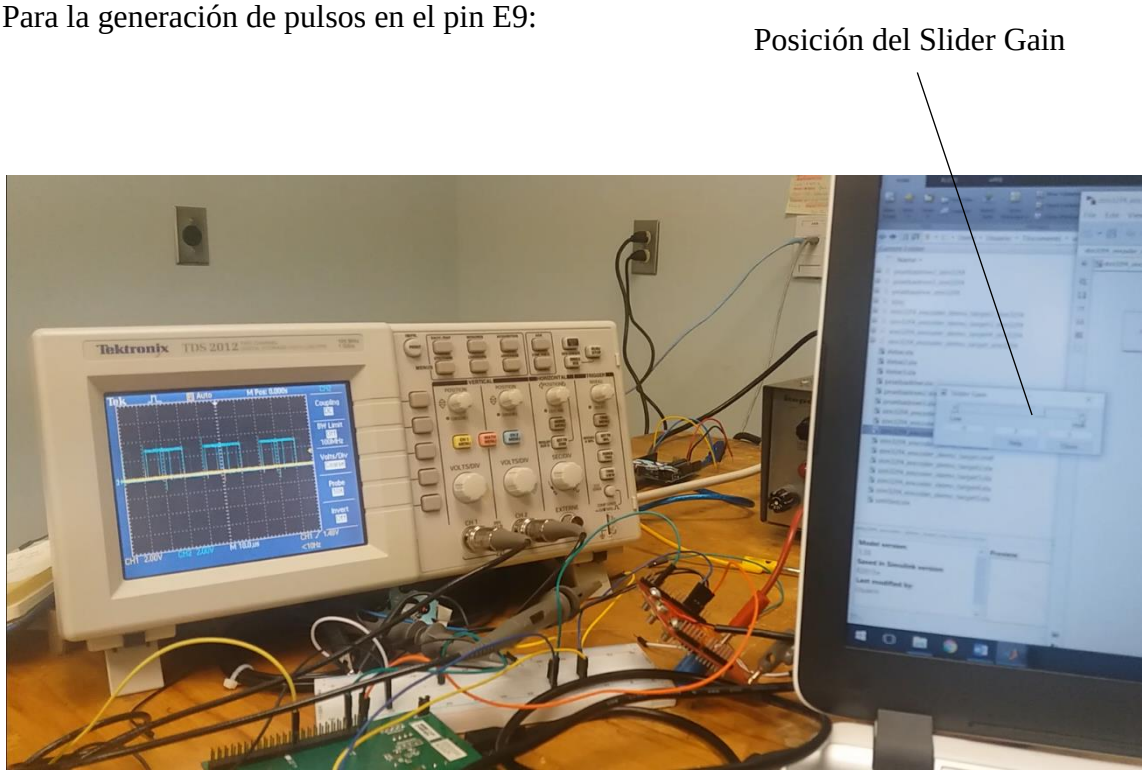

Figura 28: Generador de pulsos en el pin E9 cuando el Slider Gain da valores positivos

Al igual que en el apartado anterior, lo que vemos en las imágenes corresponde con el modelo ya que al enviar una ganancia positiva medimos una señal PWM proporcional a esa misma ganancia en el pin E9. Y en la figura 26 vemos que al enviar una ganancia negativa generemos una señal PWM en el pin E11.

Hemos conseguido controlar la velocidad y el sentido de desplazamiento del carro a través un módulo UART, enviando las ganancias deseadas desde un modelo de Simulink.

Con el concepto de “Target” y “Host” podríamos emplear más módulos UART para recuperar datos que se ejecutan en la “Target”: poniendo varios bloques Tx en la “Target” y otros Rx en el “Host” podríamos leer a tiempo real los datos que se ejecutan en el modelo. No nos centramos en realizar una aplicación en este sentido ya que nuestro objetivo principal es realizar el sistema de seguridad.

Tras haber implementado el sistema con el que comunicamos el motor con el modelo de Simulink, ahora nos centramos en poner en marcha el sistema de seguridad. El primer

paso es ser capaces de medir la posición del carro para luego poder limitar sus desplazamientos.

2.3 Medición de la posición con un láser

Vamos a medir la posición del carro con un láser. El láser a emplear es un TFmini Micro LIDAR Module, su hoja de características puede encontrarse en [10], funciona con infrarrojos, muy preciso y capaz de detectar objetos moviéndose a grandes velocidades. Su principal defecto es que es muy sensible a vibraciones y oscilaciones angulares: el objeto a medir debe situarse entre 0,3 y 12 metros, y debemos tener en cuenta que el campo de visión del láser es de 2,3º por lo que en largas distancias el láser necesita medir objetos de tamaño grande. En nuestra aplicación el objeto no se va a alejar más de un 1,5m por tanto según la hoja de características, a 1,5m el objeto debe ser de 40mm de largo para que el láser lo reconozca correctamente.

En la imagen siguiente vemos una fotografía tomada del láser TFmini [11]:

Figura 29: Láser TFmini Micro LIDAR Module

El láser se comunica con el protocolo UART a través de los cables verde y blanco que vemos en la figura anterior, por lo que vamos a emplear una placa de desarrollo de hardware para poder recibir sus datos. La placa que vamos a utilizar es un Arduino Genuino Uno. En la hoja de características nos indican cómo podemos manejar los

datos recibidos: el láser ofrece datos de salida hexadecimales, cada dato está codificado con 9 bytes.

La tabla siguiente, extraída de la hoja de características, nos muestra la información que contiene cada byte:

Byte1-2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9
0x59 59	Dist_L	Dist_H	Strength_ L	Strength_ H	Reserved	Raw.Qual	CheckSum_ L
Data encoding interpretation							
Byte1	0x59, frame header, all frames are the same						
Byte2	0x59, frame header, all frames are the same						
Byte3	Dist_L distance value is a low 8-bit. Note: The distance value is a hexadecimal value, for example, Distance 1,000cm = 03 E8 (HEX)						
Byte4	Dist_H distance value is a high 8-bit.						
Byte5	Strength_L is a low 8-bit.						
Byte6	Strength_H is a high 8-bit.						
Byte7	Reserved bytes.						
Byte8	Original signal quality degree.						
Byte9	Checksum parity bit is a low 8-bit, Checksum = Byte1 + Byte2 + ... + Byte8, Checksum is the sum of the first 8 bytes of actual data; here is only a low 8-bit.						

Tabla 2: Descripción detallada de la codificación de datos del módulo TFmini LIDAR

Los datos que nos interesa recuperar son los bytes 4, 5,6 y 7. Combinaremos los bytes 4 y 5 para obtener la distancia, y los bytes 6 y 7 para obtener la fiabilidad de la medida.

Nos inspiramos en los códigos de Arduino de las fuentes [12] y [13]. Son códigos proporcionados para la medición de la distancia con el láser. El código que usamos para nuestra aplicación es el siguiente:

```
#include<SoftwareSerial.h>// Librería SoftwareSerial

SoftwareSerial Serial1(2,3); // Definimos los pins 2 y 3 como Rx
y Tx respectivamente

void setup()
{
    Serial1.begin(115200);
    // Conf. de la transmisión UART para los pins 2 y 3
    Serial.begin(115200);
    // Configuración de la transmisión serial USB
}

void loop()
{
 //Declaración e inicialización de variables
    unsigned int d1 = 0;
    unsigned int d2 = 0;
    unsigned int s1 = 0;
    unsigned int s2 = 0;
  while(1)
  {
        while(Serial1.available()>=9)
        //Si el láser lee los 9 bytes
        {
       if((0x59 == Serial1.read()) && (0x59 == Serial1.read()))
       //Comprobamos el header (bytes 1 y 2)
            {
                 d1 = Serial1.read();
                 //Leemos el byte 3 (Dist_L)

                 d2 = Serial1.read();
                 //Leemos el byte 4 (Dist_H)

                 d2 <<= 8;
                 d2 += d1; //Sumamos para obtener la dist. total
```

```
                    Serial.print("Distance: ");
                    Serial.print(d2);
                    Serial.print('\t');

                    s1 = Serial1.read();
                    //Leemos el byte 5 (Strength_L)

                    s2 = Serial1.read();
                    //Leemos el byte 6 (Strength_H)

                    s2 <<= 8;
                    s2 += s1;
                    //Sumamos para obtener la fiabilidad total

                    Serial.print("Strength: ");
                    Serial.println(s2);

                    for(int i=0; i<3; i++)
                    //Leemos el resto de bytes
                    {
                        Serial1.read();
                    }
                }
            }
        }
}
```

Código 1: Obtención de la distancia medida por el láser TFmini Micro LIDAR

El código es sencillo: en primer lugar configuramos los pins 2 y 3 como Rx (Receiver) y Tx (Transmitter) respectivamente para nuestra comunicación vía UART. Después, siempre comprobando que el láser está transmitiendo correctamente los datos, vamos recuperando los datos de distancia y fiabilidad de la medida en un ciclo infinito.

Las variables d1 y d2 sirven para obtener el dato de distancia combinando los bytes 3 y 4, y las variables s1 y s2 para el dato de fiabilidad combinando los bytes 5 y 6.

Las funciones Serial.print() sirven para imprimir en el puerto serie del Arduino los datos que obtenemos. Podemos leer estos datos abriendo el puerto serie o con el plotter proporcionado por Arduino, para leer los datos en forma de gráfica.

El dato obtenido de distancia es extraído en cm.

El láser debe estar conectado de la siguiente manera [10]:

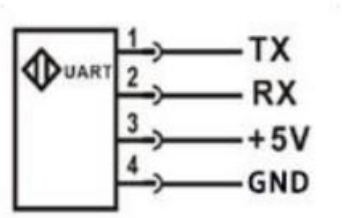

Figura 30: Esquema de los pins del láser TFmini

Por lo que conectaremos el pin 2 del Arduino (configurado como Rx) al pin 1 del láser, y el pin 3 del Arduino (configurado como Tx) al pin 2 del láser. Alimentamos el láser con los 5V que proporciona la placa Arduino.

Una vez tenemos el módulo láser instalado y funcional, ya podemos medir la posición del carro. Para finalizar tenemos que juntar todos los componentes en un montaje final e integrar el sistema de seguridad que hemos considerado al inicio del proyecto.

2.4 Montaje final

Utilizamos la maqueta que hemos visto anteriormente para el control de la velocidad del carro con el potenciómetro. Para la medición de la posición con el láser tenemos algunos inconvenientes: El láser no va a estar completamente fijo, ya que al tratarse de una maqueta no queremos deteriorar lo más mínimo los componentes. Además, hemos montado una placa metálica encima del carro de forma que la trayectoria del láser impacte con la superficie plana de la placa, pero el montaje no es demasiado preciso y no podemos garantizar que la trayectoria incidente del láser y la placa con la que

impacta estén a 90º. Hemos visto en el apartado anterior que el objeto que vamos a medir debería medir 40mm de alto aproximadamente y nuestra placa mide 10mm, por lo que es posible no tengamos una gran precisión de la posición exacta del carro. Pero al tratarse de una maqueta no requerimos de un montaje excesivamente detallista.

Nuestro primer paso es delimitar por zonas el raíl de 1m por donde se desplaza el carro. Si situamos el láser a 30mm del inicio del carril cómo nos indicaba la hoja de características de láser para una correcta medición, nos quedan las siguientes zonas:

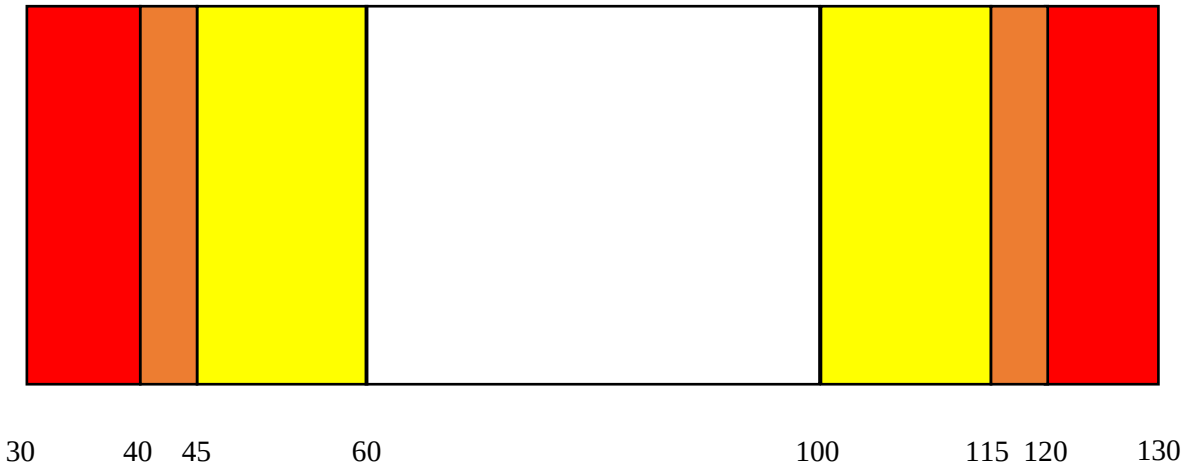

Figura 31: Esquema de la delimitación por zonas del raíl (en cm)

Zona 1: Entre 30 y 45 cm.

Zona 2: Entre 40 y 60 cm.

Zona 3: Entre 60 y 100 cm.

Zona 4: Entre 100 y 120 cm.

Zona 5: Entre 115 y 130 cm.

Las zonas 1 y 2, y las zonas 4 y 5 comparten 5mm. Lo hacemos para asegurar que el motor se pare completamente cuando está llegando hacia uno de los extremos. Así, según el esquema tenemos dos zonas claramente definidas: las zonas rojas y naranjas donde el carro no puede desplazarse lo más mínimo hacia el extremo; y las zonas amarillas donde el carro va a frenarse proporcionalmente según se acerque a la zona naranja. Si recuperamos la representación que hemos visto al inicio de este proyecto, tendríamos un esquema de la siguiente forma:

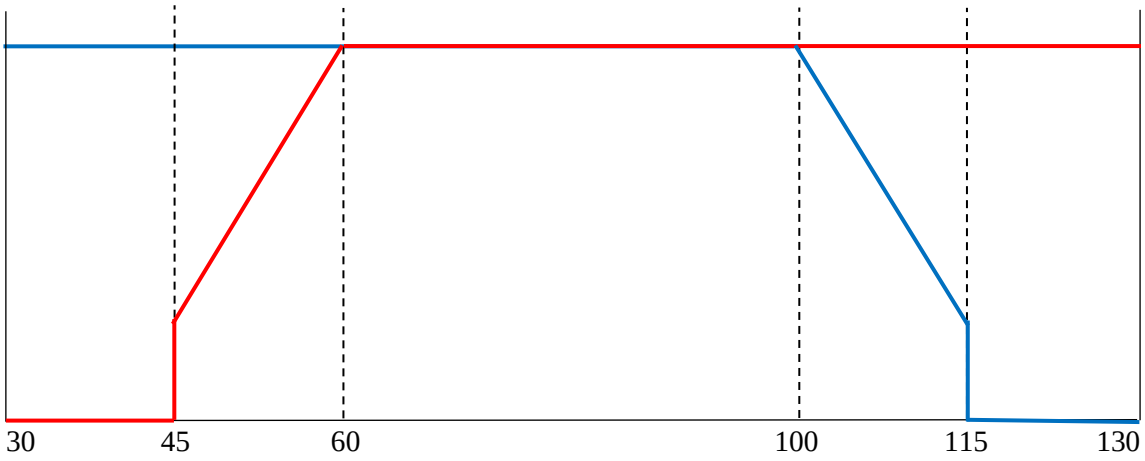

Figura 32: Esquema de la velocidad permitida en función de la posición (en cm) y la dirección del carro (Versión 2)

Como hemos visto anteriormente, el trazado rojo representa la velocidad permitida hacia la izquierda, y el trazado azul la velocidad permitida hacia la derecha. Por lo que este sistema nos va a garantizar que el carro se frene progresivamente hasta que alcance una zona de riesgo donde se detenga por completo.

Una vez tenemos el raíl delimitado, completamos el código en Arduino de medición de distancias con el láser para generar señales en función de la posición del carro. En función de las señales generadas haremos que el Simulink que controla la potencia del motor reciba estas señales y modifique la ganancia del motor para frenarlo.

Vamos a comentar distintos apartados de este código presente en el Anexo de este documento, referente al montaje final.

En primer lugar nos centramos en la parte más importante, la delimitación de las diferentes zonas y la generación en consecuencia de un digital output con la función digitalWrite(pin, valor) en diferentes pins de la placa Arduino. Esta función genera una señal de 5V o 0V si el valor del pin está en HIGH o LOW.

Por ejemplo, para la zona 1 tenemos las siguientes líneas de código:

```
//Zone 1
 if(x1 <= 45)
 {
     digitalWrite(12, HIGH);
     digitalWrite(13, LOW);
     digitalWrite(11, LOW);
     digitalWrite(8, LOW);
 }else{
     digitalWrite(12, LOW);
 }
```

Figura 33: Extracto del código del Anexo que delimita la zona 1

Cuando el carro está situado en la zona 1, se pone a HIGH el pin 12 y se apaga el resto de pins (13, 11 y 8).

Siguiendo este ejemplo delimitamos todas las zonas que hemos definido anteriormente. La tabla siguiente relaciona los digital output generados en los distintos pins en función de la zona en la que se sitúa el carro, es decir que cuando el carro está en una zona, el valor del pin correspondiente pasa de LOW a HIGH.

Zona 1	PIN 12
Zona 2	PIN 11
Zona 3	Ningún PIN
Zona 4	PIN 8
Zona 5	PIN 13

Tabla 3: Generación de señales de 5V en los pins en función de la zona del raíl

También cabe destacar del código implementado en Arduino que generamos dos señales PWM en los pins 9 y 10, en función de la posición del carro. Estos generadores de pulsos nos servirán para las zonas 2 y 4, donde queremos que el carro se desplace a una velocidad proporcional a su posición con respecto a los extremos. En nuestra placa de Arduino, los PWM generados son de 500Hz de frecuencia de forma estándar [5].

Para la generación de pulsos utilizamos la función analogWrite(pin, posición), donde la variable “posición” es la distancia medida por el láser. De modo que se genera una señal

PWM función de la posición, que mapeamos con la función map() entre las distancias 40 y 60 cm para la zona 2 y entre las distancias 100 y 120 cm para la zona 4. Es necesario usar la función map() ya que la función analogWrite() requiere de una posición entre 0 y 255.

De modo que cuando el carro está en la zona 2 (entre 40 y 60 cm), se genera un PWM en el PIN 9 de 100% de ciclo de trabajo en 60cm y baja proporcionalmente hasta 0% en 40cm. Del mismo modo, cuando el carro está en la zona 4 (entre 100 y 120 cm), se genera un PWM en el PIN 10 de 100% de ciclo de trabajo en 100cm y baja proporcionalmente hasta 0% en 120cm.

Nos centramos en la obtención de la velocidad a partir de la medición de la posición con el láser. Como medimos distancias cada 0,01 segundos con el láser (ya que el láser muestrea a 100Hz), podemos obtener una velocidad estimada a la que se mueve el carro con la simple fórmula:

$$vel = \frac{x2 - x1}{0{,}01}$$

Siendo x1 la posición tomada con anterioridad a x2.

Este dato de velocidad no está filtrado, y varía cada 0,01 segundos. Para obtener un dato de velocidad estimada menos variable pero que sea fiable diseñamos un filtro de Kalman. Este filtro nos permite eliminar el ruido que afecta a la obtención del dato de velocidad a través del sensor láser. Es un estimador matemático que utiliza un proceso iterativo para eliminar el error cuadrático medio de un conjunto de datos. Se basa en la predicción de un conjunto de variables (en nuestro caso simplemente la velocidad) [15].

El filtro usado proviene de la fuente [16], es un filtro de Kalman básico para modelos unidimensionales. El código que implementamos en Arduino para la generación de un filtro es el siguiente:

```
#include <SimpleKalmanFilter.h>
SimpleKalmanFilter simpleKalmanFilter(5, 5, 0.01);
estimated_value = simpleKalmanFilter.updateEstimate(vel);
```

Figura 3435: Extracto del código del Anexo que genera un filtro de Kalman y obtiene la velocidad del carro estimada

La variable `vel` es el dato de velocidad obtenido como hemos visto anteriormente. Para diseñar un filtro simplemente se deben rellenar los 3 parámetros en la función `simpleKalmanFilter()`. El primer argumento es la incertidumbre de medición, se traduce por la variable que mide cuánto esperamos que varíe nuestra medición. El segundo argumento es la incertidumbre de estimación, puede iniciarse con el mismo valor que la incertidumbre de medición ya que es el propio filtro el que ajustará su valor. Y el tercer argumento es la varianza del proceso, que generalmente es un número pequeño comprendido entre 0,001 y 1.

Para los valores seleccionados obtenemos un filtro aceptable, que nos permite estimar la velocidad a la que se mueve el carro.

No hemos visto demasiado en detalle esta sección ya que para nuestra aplicación no hemos considerado la velocidad como un parámetro importante. Los parámetros que marcan la limitación de la velocidad del carro han sido las delimitaciones de las diferentes zonas.

Una vez tenemos delimitadas las zonas del raíl en función de la generación de señales desde el Arduino, diseñamos un modelo de Simulink utilizando el modelo que hemos usado para el control de velocidad y sentido de giro de un motor con un potenciómetro. Tenemos que añadirle a este modelo nuestro control desde el Arduino, para que sea este el que limite la ganancia enviada al motor cuando estemos en una situación de riesgo. Las situaciones de riesgo son aquellas donde nos vemos obligados a limitar los desplazamientos del carro. Podemos definirlas como dos: el caso en el cual el carro se aproxima a gran velocidad a un extremo debido a que el mando generado es elevado (zona de frenado), y cuando el carro está ya pegado al extremo y limitamos completamente su desplazamiento (zona de detención). En estos casos, será el Arduino el que tome el control del motor.

Este es el modelo completo:

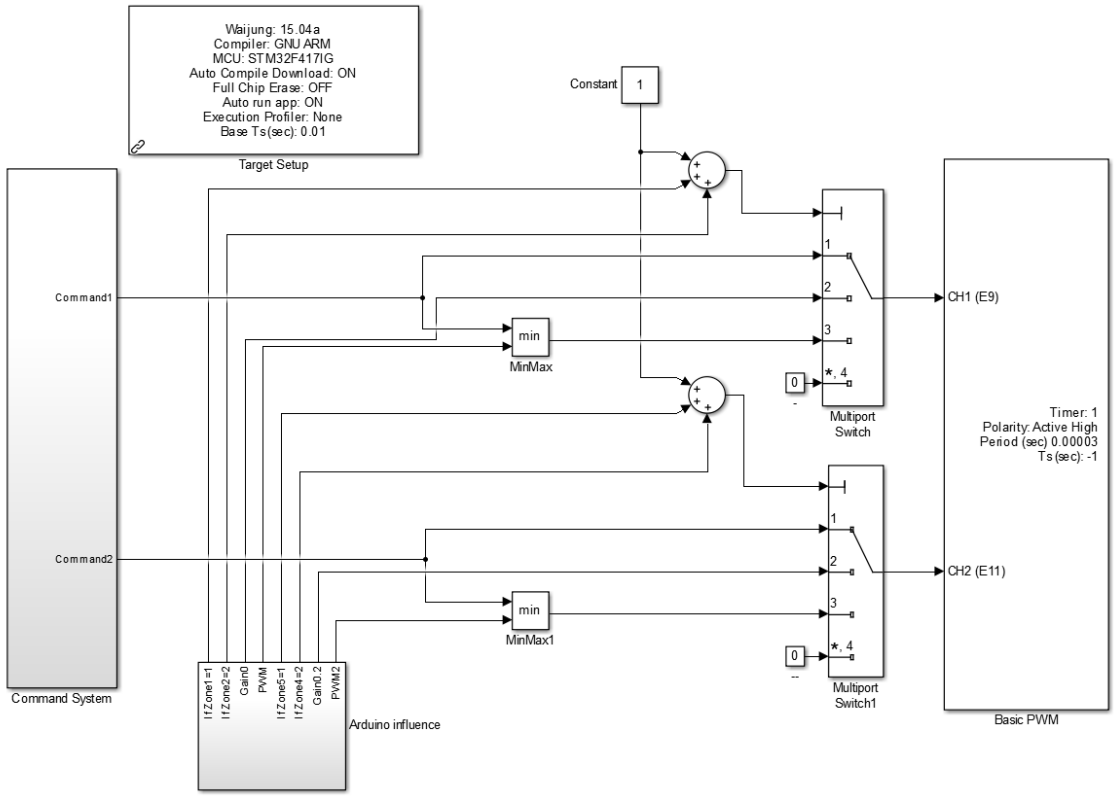

Figura 36: Modelo Simulink para el control de velocidad limitada y sentido de giro de un motor con un potenciómetro

En primer lugar, el subsistema Command System (presente en el Anexo) es el mismo que el de la figura 15 vista previamente en este documento. Es el encargado en mandar una ganancia entre 0 y 100 al bloque Basic PWM a una de sus dos entradas según queramos movernos en un sentido u otro. Los dos bloques Multiport Switch son los encargados de decidir si el sistema envía la ganancia pedida por el potenciómetro o si es el Arduino quién da la ganancia deseada. Cada Multiport Switch se encarga de limitar los desplazamientos a la izquierda (Switch superior) o a la derecha (Switch inferior). Su funcionamiento es el siguiente: en función de la primera entrada de estos bloques se escoge una de las entradas entre las distintas posibles. Si la entrada es un 1, el bloque da como salida la entrada del puerto 1, y así sucesivamente. Así, algunas de las salidas del bloque "Arduino Influence" influyen en la selección del Multiport Switch. Para ver cada una de las salidas y por lo tanto de las opciones del Multiport Switch vamos a ver en detalle el subsistema de "Arduino Influence".

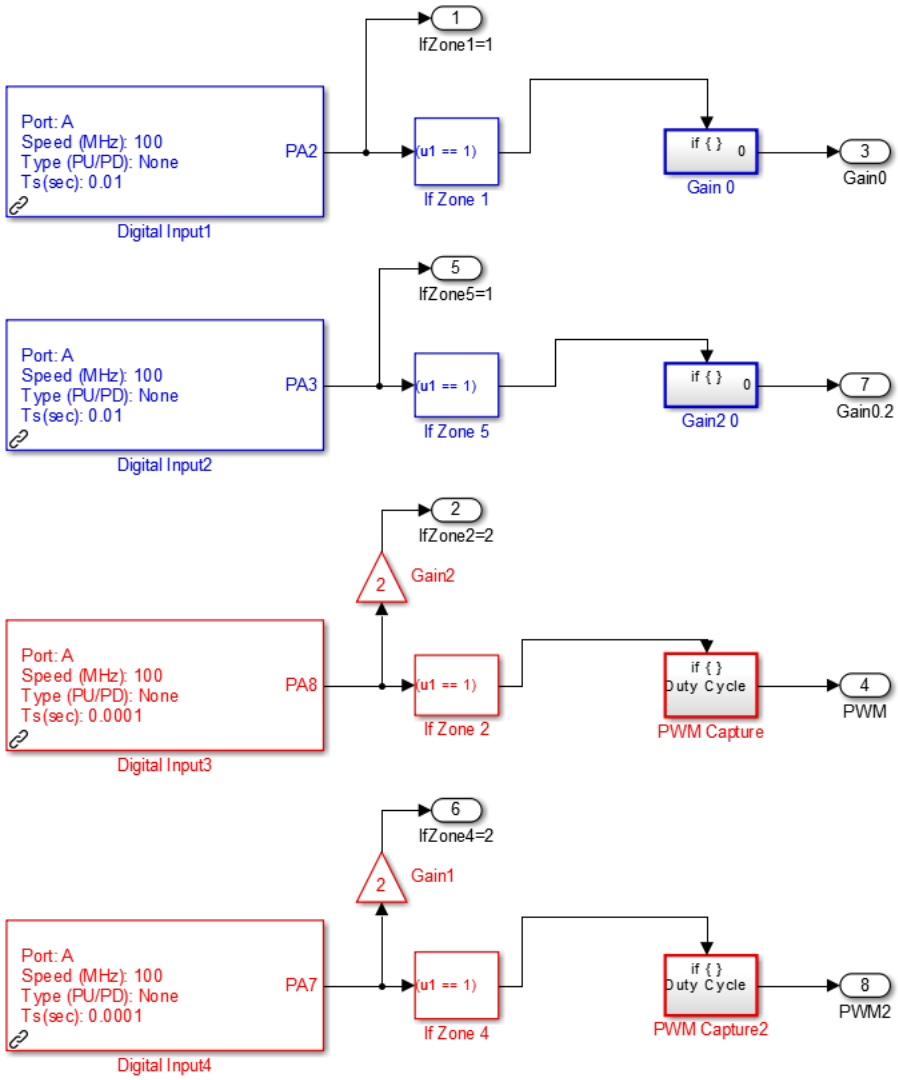

Figura 37: Subsistema "Arduino Influence" del modelo Simulink para el control de velocidad limitada y sentido de giro de un motor con un potenciómetro

A su vez, capturamos otra imagen de los subsistemas de tipo if() denominados "PWM Capture" y "PWM Capture2". En los otros subsistemas simplemente tenemos constantes de valor 0 salientes (Gain 0).

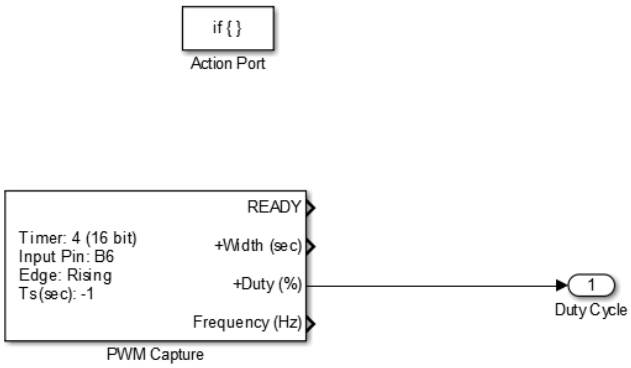

Figura 38: Subsistema "PWM Capture" de tipo if del modelo Simulink "Arduino Influence"

Primeramente, vamos a describir los bloques más importantes para entender correctamente el funcionamiento del subsistema "Arduino Influence".

-Digital Input: es un bloque de la librería Waijung, implementa un módulo Digital Output para generar un 1 si recibe 1 y un 0 en caso contrario. Podemos colocar todos los bloques Digital Input que queramos mientras configuremos diferentes pins para cada uno de ellos. Así, si el primer bloque está configurado con el pin A2 del microcontrolador, cuando en el pin A2 llegue un 1 lógico el bloque genera otro 1 lógico.

-PWM Capture: también es un bloque de la librería Waijung, sirve para analizar señales PWM entrantes. Se selecciona un input Pin y el bloque nos da la frecuencia y el ciclo de trabajo de la señal. En este caso usamos el bloque para extraer el ciclo de trabajo de un generador de pulsos. Las señales PWM entrantes serán las generadas por el Arduino que hemos visto previamente.

Para sincronizar la generación de las señales del Arduino con nuestro modelo Simulink, tenemos que conectar los pins donde se generan las señales directamente con el microcontrolador. De tal forma que cada zona está ligada a un pin en Arduino y a su vez a un pin del microcontrolador.

Recuperamos la tabla 2 vista previamente para completarla con los pins que usaremos en el microcontrolador:

Zona	PIN Arduino	PIN STM32F4DISCOVERY
Zona 1	PIN 12	PIN A2
Zona 2	PIN 11	PIN A8
Zona 3	Ningún PIN	Ningún PIN
Zona 4	PIN 8	PIN A7
Zona 5	PIN 13	PIN A3

Tabla 4: Generación de señales de 5V en los pins de Arduino y conexión con los pins del microcontrolador STM32F4DISCOVERY en función de la zona del raíl

Observando el conjunto del sistema, podemos ver que tenemos 4 casos definidos:

-Estamos en la zona 1: el multiport switch superior escoge la entrada 2, que es de ganancia 0. Por lo que para cualquier ganancia enviada desde el potenciómetro, el pin E9 sólo recibe ganancia 0 mientras que el carro esté en la zona 1. Así, el carro sólo podrá desplazarse a la derecha hasta que salga de la zona.

-Estamos en la zona 5: el multiport switch inferior escoge la entrada 2, que es de ganancia 0. Por lo que para cualquier ganancia enviada desde el potenciómetro, el pin E11 sólo recibe ganancia 0 mientras que el carro esté en la zona 5. Así, el carro sólo podrá desplazarse a la izquierda hasta que salga de la zona.

-Estamos en la zona 2: el multiport switch superior escoge la entrada 3. Esta entrada es la señal mínima resultante entre el mando (ganancia dirigida por el potenciómetro) y el generador de pulsos que genera el Arduino en función de la posición del carro. El bloque MinMax compara los ciclos de trabajo de las dos señales y coge el más pequeño de los dos. Así, si el mando pide una ganancia superior a la permitida (es decir, con un ciclo de trabajo superior al ciclo de trabajo del PWM del Arduino) es la señal PWM del Arduino la que toma el control del motor. En esta situación, el PWM generado por el Arduino es la referencia que no podemos sobrepasar y al ser una señal que disminuye cuando el carro se aproxima al extremo, hemos conseguido obtener nuestra zona de frenado.

-Estamos en la zona 4: el multiport switch inferior escoge la entrada 3. Es el mismo caso que el caso anterior, pero esta vez se limitan los desplazamientos a la derecha.

Cuando estemos en dos casos a la vez (recordemos que es posible ya que existe un pequeño margen donde se solapan dos zonas), los multiport switch escogen las entradas 4, que como podemos observar es un bloque de constante 0. Precisamente es como si estuviéramos en las zonas más extremas, y mandamos al motor una ganancia 0 para que frene completamente. Con este sistema respetamos la condición que hemos impuesto de priorizar las zonas más al borde sobre el resto.

Con este modelo hemos logrado obtener un sistema que respete el diseño inicial, con dos zonas de frenado y dos zonas de detención. Lo más destacable es el hecho de que es el programa de Arduino el que tiene prioridad sobre la generación del mando con el potenciómetro. Es por tanto el limitador de nuestro sistema, funcionando de forma independiente y actuando sólo cuando es necesario.

Ahora vamos a poner a prueba nuestro sistema, y a corregir el modelo si es necesario.

En la imagen siguiente podemos ver un conjunto del montaje:

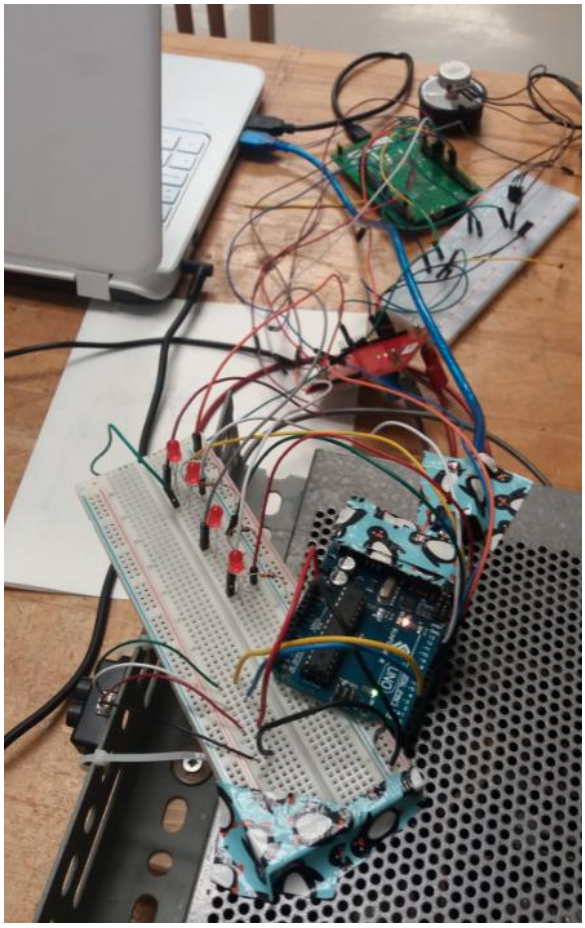

Figura 39: Montaje control de velocidad limitada y sentido de giro de un motor con un potenciómetro

Tenemos todos los componentes que hemos visto de forma separada anteriormente: el láser TFmini, la placa Arduino Uno, y el montaje que hemos usado para el control de velocidad y sentido de giro del motor con el potenciómetro.

En la imagen podemos observar que hemos añadido 4 LEDs para poder visualizar cuando se activan nuestros pins al detectar la posición del carro en una de las zonas definidas. Ya que los LEDs están directamente conectados con los pins de salida 8, 11, 12 y 13 que se encienden o apagan en función de la zona del raíl como hemos visto anteriormente.

Para poner a prueba la medición de la posición con el láser, movemos el carro para ver si los LEDs se encienden en función de la zona en la que se encuentra el carro. Hemos tomado dos fotografías:

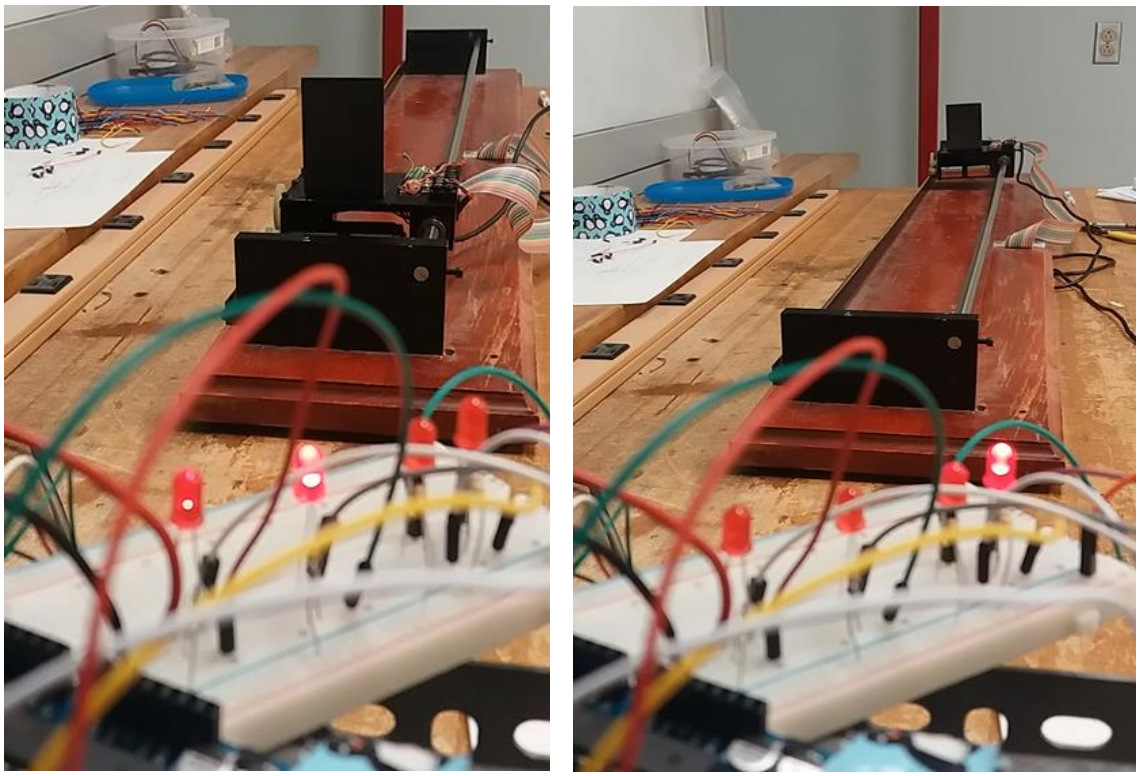

Figura 40: Fotografías de dos posiciones distintas del carro e iluminación de los LEDs correspodientes

En la primera captura vemos que el carro está entre la zona 1 y 2 (entre 40 y 45 cm) y por lo tanto se han encendido los dos primeros LEDs. Esto significa que los pins 12 y 11 están alimentados. Esto es debido a que cómo la delimitación de las zonas interiores (2 y 4) se efectúa más tarde en el código de Arduino, es posible que tengamos dos pins alimentados a la vez. No nos supone un problema ya que como hemos visto en el

modelo de Simulink, ante estos casos la ganancia que se envía al motor es de 0, priorizando así las zonas más exteriores.

En la segunda captura lo que observamos es que el carro está en la zona 5, sin contacto con la zona 4 ya que sólo tenemos un LED encendido.

Por lo tanto, nuestro sistema de medición de la posición del carro y la generación de señales en consecuencia funciona correctamente.

Ponemos en marcha el sistema completo, cargando en el microcontrolador nuestro modelo de Simulink y alimentando el motor. Variando el potenciómetro movemos el carro de un lado a otro del raíl, pero este jamás impacta contra los extremos. Si nos aproximamos a un extremo con una ganancia elevada el Arduino toma el control del motor enseguida, al entrar en la zona de frenado. Pero si nos aproximamos al extremo despacio con una ganancia poco elevada entonces el Arduino sólo toma el control del motor cuando ya se encuentra en la zona de detención.

Podemos modificar los valores que delimitan las diferentes zonas para agrandar la zona de frenado si la inercia del carro es demasiado grande, o darles más importancia a las zonas de detención ampliando sus límites.

También hay que tener en cuenta que en nuestra maqueta no teníamos una carga junto con el carro, por lo que es probable que haya que dar un mayor protagonismo a las zonas de frenado cuanto mayor sea la carga transportada.

2.5. Resultados

En primer lugar, cabe destacar que los resultados obtenidos han sido satisfactorios. El carro no llega a impactar contra ninguno de los extremos en ningún caso. La zona de frenado permite al carro disminuir su velocidad, si es demasiado alta, de forma suave y controlada. La zona de detención garantiza que el carro no se aproxime más al extremo en ninguna circunstancia. Lo único que podría hacer fallar el sistema de seguridad es una falsa medición de la posición del carro, falseando de ese modo la delimitación de las zonas y en consecuencia la limitación de la velocidad. Pero en la planta piloto que se ha previsto implantar este sistema, los alumnos no tendrán acceso a la zona de

desplazamiento del carro y por tanto no podrán interferir en la medición de la posición de este último.

A pesar de que los resultados obtenidos sólo se han verificado en nuestra maqueta, nuestro modelo final se puede emplear en todo tipo de carros (con un tamaño considerable para ser medidos por el láser) que se muevan con un motor y en raíles delimitados entre 0.3 y 12 metros. Esta limitación es debida al láser que hemos empleado, pero con otro dispositivo de medición se puede ampliar la zona de desplazamiento del carro. Sólo se tienen que variar los valores de las diferentes zonas definidas en el código del programa de Arduino.

El futuro de este proyecto es que se implemente el sistema de seguridad diseñado en la planta piloto de la Universidad de Montreal. Sin embargo, se puede implementar este sistema en plantas piloto similares, presentes en la industria. Nuestro sistema diseñado es sencillo y económico y no requiere de montajes adicionales, salvo fijar el láser en algún lugar para que tome correctamente la posición del carro que transporta la carga.

Como aplicaciones complementarias al proyecto, se pueden añadir nuevas condiciones en las cuales se limita la velocidad del carro. Es decir, añadir nuevas situaciones de riesgo, añadiendo otra zona de detención si la velocidad medida es muy elevada, o teniendo en cuenta la velocidad del carro en las distintas zonas. Esto podría proveer a la planta piloto de un sistema de seguridad más completo.

Capítulo III:

Anexo

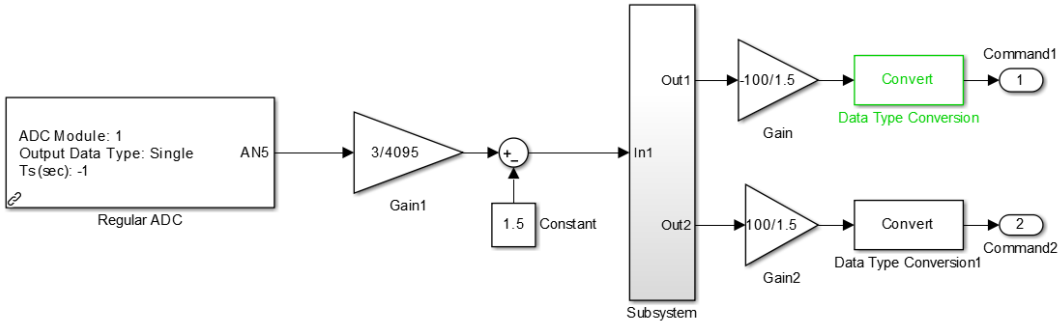

Figura 41: Subsistema "Command System" del modelo Simulink para el control de velocidad limitada y sentido de giro de un motor con un potenciómetro

Código del montaje final en Arduino:

```
//
//Author: Sergio Méntrida Zamarra
//Description: This program is a distance measurer through a
laser for the security system of a hoist. Designed for Arduino
Uno and STM32F4 Microntroller.
//Version: v1.1, 11/05/2018

#include <Arduino.h>
#include<SoftwareSerial.h>// soft serial port header file
#include <SimpleKalmanFilter.h>

//Connections of MicroLidar:
// - Black = GND (connected to GND)
// - Red = 5V (4.5 - 6.0V) (connected to 5V on Arduino)
// - White = TFmini RX (aka. connect to microcontroller TX)
// - Green = TFmini TX (aka. connect to microcontroller
RX)charriot
// NOTE: for this sketch you need a microcontroller with
additional serial ports beyond the one connected to the USB
cable
```

```
SoftwareSerial Serial1(2,3); // define the soft serial port as
Serial1, pin2 as RX, and pin3 as TX
//Connect pin2 to Tx TFmini and pin3 to Rx Tfmini

//Declaration and initialization of variables
signed int vel = 0;
signed int x1 = 0;
signed int x2 = 0;
float estimated_value = 0;
int pos = 0;
int pos2 = 0;

//Simple Kalman Filter (Measurement Uncertainty, Estimation
Uncertainty, ProcessVariance)
SimpleKalmanFilter simpleKalmanFilter(5, 5, 0.01);


void setup()
{
    Serial1.begin(115200); // HW Serial for TFmini
    Serial.begin(115200); // Serial output through USB to
computer

    //Declaration and initialization of PINS,and connections
between Arduino and Microcontroller
    pinMode(12, OUTPUT); //PIN 12 connected to A2
    pinMode(13, OUTPUT); //PIN 13 connected to A3
    pinMode(11, OUTPUT); //PIN 11 connected to A8
    pinMode(8, OUTPUT); //PIN 8 connected to A7
    digitalWrite(12, LOW);
    digitalWrite(13, LOW);
    digitalWrite(11, LOW);
    digitalWrite(8, LOW);

    delay (100); // Give a little time for things to start
}
```

```
void loop()
{
   while(1){ // Keep going for ever

        while(Serial1.available()>=9)
    {
        if((0x59 == Serial1.read()) && (0x59 == Serial1.read()))
        //Byte1 & Byte2
        {
            signed int t1 = Serial1.read(); //Byte3
            x1 = Serial1.read(); //Byte4
            x1 <<= 8;
            x1 += t1;

            if((x1 < 500) && (x1 > 0))
            //Filter for extreme measures
            {
              Serial.print("Distance: ");
              Serial.print(x1);
              Serial.print('\t');
            }

            if((x1 < 500) && (x1 > 0))
            //Filter for extreme measures
            {
                vel=(x2-x1)/(0.01);
                //speed=distance/time   time=1/100Hz (TFmini
                samples at 100 Hz)
                Serial.print(vel);
                Serial.print('\t');

                x2=x1;
                //Estimated value from Kalman Filter:
                estimated_value =
                simpleKalmanFilter.updateEstimate(vel);
```

```
            Serial.print("Estimated speed: ");
            Serial.print(estimated_value);
            Serial.println('\t');
        }
      /* Code for reading other values from the TFmini
        unsigned int t3 = Serial1.read(); //Byte5
        unsigned int t4 = Serial1.read(); //Byte6

        t4 <<= 8;
        t4 += t3;

        if(t4 < 2000)
        {
          Serial.print("Strength: ");
          Serial.println(t4);
        }


        for(int i=0; i<3; i++)
        {
            Serial1.read(); ////Byte7,8,9
        }
        */
    }

  //Delimitation of the different zones
  //Note: Zones 1 and 5 have priority in Zones 2 and 4
  //Zone 1
  if(x1 <= 45)
  {
      digitalWrite(12, HIGH);
      digitalWrite(13, LOW);
      digitalWrite(11, LOW);
      digitalWrite(8, LOW);
  }else{
      digitalWrite(12, LOW);
  }
```

```
//Zone 5
if((x1 >= 115) && (x1 <= 500))
{
    digitalWrite(13, HIGH);
    digitalWrite(12, LOW);
    digitalWrite(11, LOW);
    digitalWrite(8, LOW);
}else{
    digitalWrite(13, LOW);
}

//Zone 2
if( ((x1 >= 40) && (x1 <= 60)) )
{
    pos=map(x1,40,60,0,255);
    analogWrite(9,pos);
    //PWM with duty cycle depending on the position
    digitalWrite(11, HIGH);
    digitalWrite(13, LOW);
    digitalWrite(8, LOW);
}else{
  pos=0;
  analogWrite(9,pos);
  digitalWrite(11, LOW);
}

//Zone 4
if( (x1 >= 100) && (x1 <= 120) )
{
    pos2=map(x1,120,100,0,255);
    analogWrite(10,pos2);
    //PWM with duty cycle depending on the position
    digitalWrite(8, HIGH);
    digitalWrite(12, LOW);
    digitalWrite(11, LOW);
}else{
```

```
          pos2=0;
          analogWrite(10,pos2);
          digitalWrite(8, LOW);
        }


      }
    }
}
```

Código 2: Programa Final para el control de la velocidad de un carro desplazándose por un raíl delimitado por zonas

Capítulo IV:

Bibliografía

[1] **David Fecteau,** *Documentation relative aux palans du local A429.4,* Département de Génie Électrique, École Polytechnique, Université de Montréal, 2017, pp.1-3.

[2] **Richard Gourdeau,** *ElE4202 Commande des processus industriels, Cahier des laboratoires,* Département de genie électrique, École Polytechnique de Montréal, 2016, pp.7.

[3] **David Fecteau,** *Projet de modernisation des palans du laboratoire A429.4,* Département de Génie Électrique, École Polytechnique, Université de Montréal, 2017, pp.1-5.

[4] **Infineon,** *High Current PN Half Bridge BTN8982TA Data Sheet*, Infineon [Online] Disponible: https://www.infineon.com/dgdl/Infineon-BTN8982TA-DS-v01_00-EN.pdf?fileId=db3a30433fa9412f013fbe32289b7c17 [Última visita el 10/06/2018].

[5] ___, *Entradas y Salidas Analógicas Arduino. PWM,* Aprendiendo Arduino [Online]. Disponible: https://aprendiendoarduino.wordpress.com/category/pwm/ [Última visita el 13/06/2018].

[6] ___, *Library of the infineon DC Motor Control Shields with BTN8982TA for Arduino,* GitHub [Online]. Disponible: https://github.com/Infineon/DC-Motor-Control-BTN8982TA [Última visita el 20/06/2018].

[7] **Antony García González,** *El puente H: Inviertiendo el sentido de giro de un motor con Arduino,* Panamahitek [Online]. Disponible: http://panamahitek.com/el-puente-h-invirtiendo-el-sentido-de-giro-de-un-motor-con-arduino/ [Última visita 11/06/2018].

[8] **Aimagin Co., Ltd,** *Waijung Blockset,* Aimagin [Online]. Disponible: http://waijung.aimagin.com/ [Última visita el 24/06/2018].

[9] **STMicroelectronics,** *User manual 1472*, ST [Online]. Disponible: https://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf [Última visita 20/06/2018].

[10] **Benewake (Beijing) Co. Ltd,** *TFmini Infrared Module Specification,* Document No.: SJ-GU-TFmini-01, Version A00 [Online] Disponible: https://www.robotshop.com/media/files/pdf/benewake-tfmini-micro-lidar-module-ip65-12-m-datasheet.pdf#page=5 [Última visita el 15/06/2018].

[11] ___, *TFMini – Micro LiDAR Module (12m),* Cytron, Sensors, Laser Range Finder [Online]. Disponible: https://www.cytron.io/p-sn-lidar-tfmini [Última visita el 19/06/2018].

[12] ___, *Grove-TF Mini LiDAR,* Wiki Grove [Online]. Disponible: http://wiki.seeedstudio.com/Grove-TF_Mini_LiDAR/ [Última visita el 18/06/2018].

[13] ___, *BENEWAKE TFMINI - INEXPENSIVE LIDAR WITH TEENSY 3.5,* Instructables [Online]. Disponible: http://www.instructables.com/id/Benewake-TFmini-Inexpensive-LiDAR-With-Teensy-35/ [Última visita el 20/06/2018].

[14] **Sébastien Parent-Charette,** *Benewake TFMINI Micro LIDAR Module IP65 (12 m) connection,* RobotShop inc. [Online]. Disponible: https://www.robotshop.com/forum/benewake-tfmini-micro-lidar-module-ip65-12-m-connection-t16743 [Última visita el 15/06/2018].

[15] **María Esther Aranda Romasanta,** *Estudio y aplicación del Filtro de Kalman en fusión de sensores en UAVs,* Trabajo de Fin de Grado, Grado en Ingeniería Aeroespacial, Departamento de Ingeniería Electrónica, Universidad de Sevilla, 2017, pp.6.

[16] ___, *Simple Kalman Filter,* GitHub [Online]. Disponible: https://github.com/denyssene/SimpleKalmanFilter [Última visita el 17/06/2018].