Trabajo Fin de Grado
Grado de Ingeniería en Tecnologías Industriales

Diseño y realización de un sistema automatizado de clasificación por colores basado en brazo robótico y el TMS320F28335

Autor: Andrés Jordán Gamito

Tutor: Federico José Barrero García

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado de Ingeniería en Tecnologías Industriales

Diseño y realización de un sistema automatizado de clasificación por colores basado en brazo robótico y el TMS320F28335

Autor:

Andrés Jordán Gamito

Tutor:

Federico José Barrero García

Profesor titular

Dpto. de Ingeniería Electrónica

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2020

Proyecto Fin de Carrera: Diseño y realización de un sistema automatizado de clasificación por colores basado en brazo robótico y el TMS320F28335

Autor: Andrés Jordán Gamito
Tutor: Federico José Barrero García

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Sevilla, 2020

A todo aquel que me apoyó en este camino

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia, especialmente a mis padres, pues siempre me aportaron la confianza necesaria para creer que conseguiría todo lo que me propusiera. Sin su apoyo y ayuda en los momentos más duros de este largo camino, jamás habría podido llegar hasta el final.

A mis amigos de toda la vida y a los que me ha traído la ETSI, con los que tanto he compartido penas y alegrias a lo largo de mi vida universitaria, y gracias a los que estos años han sido los mejores de mi vida.

A todos los profesores de la ETSI, pues gracias a su esfuerzo y dedicación he conseguido progesar mucho, tanto a nivel académico como personal.

Y, por último, a mi tutor, Federico, y a Jose María, cuya pasión por la electrónica, que bien me supieron transmitir siendo mis profesores, ha sido fundamental para el desarrollo de este proyecto.

Andrés Jordán Gamito
Sevilla, 2020

Resumen

Un brazo robótico es un brazo mecánico articulado, capaz de manipular objetos de una forma similar a la que lo hace un brazo humano. Hoy en dia, su uso esta muy generalizado en el mundo de la industria, ya que su gran versatilidad les permite ser utiles en infinidad de aplicaciones. Realizar operaciones con estos aparatos presentan numerosas ventajas respecto respecto a los trabajos realizados por un ser humano, como pueden ser, por ejemplo, su gran precisión o su capacidad para llevar a cabo una tarea de forma ininterrumpida.

En este TFG se pretende desarrollar un sistema que, mediante un brazo robótico, sea capaz de clasificar objetos según el color de estos. Para simular el entorno de trabajo, se ha construido una pequeña cinta transportadora por la que van pasando objetos de distinto color. El reconocimiento de color se hace a través un sensor de color, y, además, se han incluido unos detectores de presencia para conocer la posición de los objetos en la cinta.

En definitiva, el proyecto se fundamenta en la coodinación y gestion del trabajo de un grupo de sensores (sensor de color, sensor de presencia), y actuadores (cinta transportadora, brazo robótico), para realizar la función anteriormente descrita. Esta tarea de coordinación y gestión se realiza gracias al DSP TMS320F28335 de Texas Instrument, de forma que, pese a tratarse de una maqueta, las ideas y conceptos que se implementan tienen luego una aplicación industrial directa.

Abstract

A robotic arm is an articulated mechanical arm, capable of manipulating objects like a human arm would do it. Nowadays, their use is very generalized in the industrial world, since their great versatility allows them to be useful in countless applications. Carrying out operations with these devices have a lot of advantages over the work done by a human, for example, their great precision or their ability to carry out a task continuously.

In this TFG it is intended to develop a system that, using a robotic arm, is capable of classifying objects according to their color. To simulate the work environment, a small conveyor belt has been built through which objects of different colors pass. Color recognition is done through a color sensor, and, in addition, presence detectors have been included to know the position of objects on the conveyor belt.

In short, the project is based on the coordination and management of the work of a group of sensors (color sensor, presence sensor), and actuators (conveyor belt, robotic arm), to perform the function described above. This coordination and management task are carried out with the DSP TMS320F28335 from Texas Instrument, so that, despite being a model, the ideas and concepts that are implemented then have a direct industrial application.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

E/S	Entrada y salida
PCB	Printed circuit board (placa de circuito imperso)
LDR	Light Dependant Resistor (Resistencia dependiente de la luz)
ALU	Unidad aritmético – lógica
PWM	Pulse Width Modulator (Modulador por anchura de pulsos)
DSP	Digital Signal Processor (Procesador de señales digitales)
DSC	Digital Signal Controller (Controlados de señales digitales)
DMA	Direct Memory Access (Acero directo a memoria)
CPU	Central Process Unit (Unidad central de procesos)
RAM	Random Access memory (memoria de acceso aleatorio)
CAD	Convertidor analógico digital
rpm	Revoluciones por minuto
CCS	Code composer studio

1 INTRODUCCIÓN

"Un hombre sin técnica, es decir, sin reacción contra el medio, no es un hombre"

- Ortega y Gasset -

Desde tiempos inmemoriares, el ser humano ha desarrollado y perfeccionado su capacidad de modificar su entorno para conseguir aquello que desea. Así, desde el uso por parte de los primeros homínidos de rudimentarias herramientas, lo avances en la ciencia y la técnica se han sucedido a lo largo de la historia, facilitando a cada paso un poco más la vida de las personas. Estos avances alcanzaron su punto culmen en la revolución industrial, hecho que supuso el cambio de una economía rural, basada en el esfuerzo del hombre, a una economía industrial, sustentada en la fuerza de las máquinas.

A mediados del sigo XX llegaría lo que se conoce como tercera revolución industrial, la cual se produjo gracias a los grandes avances de la época en electrónica, que trajeron de la mano otros en informática, computación y comunicación. Es a partir de este momento cuando el ámbito de la automatización, que hasta la fecha solo había tenido una pequeña implementación con sistemas mecánicos, cobra especial relevancia [1].

En el mundo industrial se empieza a implementar entonces lo que se conoce como producción automatizada, que no es otra cosa que el uso de estas nuevas tecnologías para la realización, el control y el monitoreo de actividades industriales, todo ello de una forma autónoma, reduciéndose así, por tanto, la intervención humana drásticamente.

Esta automatización industrial ha cambiado por completo la forma de operar de las fábricas desde aquella primera revolución industrial. Un claro ejemplo de ello lo podemos ver en la industria del automóvil, cuyas líneas de producción hoy en día están fundamentalmente basadas en maquinaria que trabaja de forma autónoma.

Figura 1-1. Fábrica de coches automatizada. [1]

[1] Imagen tomada de https://www.ersilias.com/fabrica-de-coches-robotizada/. Fecha de consulta: Agosto de 2020

Tras esta breve reseña histórica, podemos concluir que hoy en día, la automatización de tareas es uno de los aspectos mas importantes en el ámbito de la industria. El proyecto que nos ocupa se enmarca dentro de esta necesidad actual de conseguir que un determinado proceso sea realizado de forma automática. Concretamente, el objetivo es desarrollar un sistema capaz de clasificar objetos que son transportados por una cinta según su color, mediante el uso de un brazo robótico.

Es decir, partiendo de una necesidad que puede surgir dentro de una fábrica actual, como puede ser la detección del color de objetos y su manipulación y clasificación con un brazo articulado, se pretende elaborar paso a paso una solución, aunque a pequeña escala, de esta situación, de manera que se afronten los problemas y retos que se podrían plantear en un escenario real.

Con el fin de lograr dicho objetivo, los pasos a seguir serán los siguientes:

1. Análisis del problema a resolver y planteamiento de la solución de forma general.
2. Estudio de mercado, seleccionando y adquiriendo todo el hardware necesario para llevar a cabo la solución antes planteada. Si se detectara que alguna de las partes de la solución propuesta es inviable, volver al paso 1 y modificar lo que sea necesario.
3. En caso de no encontrar en el mercado alguno de los componentes necesarios o de necesitar componentes a medida, investigar y llevar a cabo su fabricación.
4. Análizar cada uno de los componentes obtenidos de forma individualizada. Puesta en marcha de cada parte por separado.
5. Acometer el funcionamiento común de todos ellos, consiguiéndose una primera aproximación funcional del proyecto.
6. Realizar correcciónes de posibles errores y pulir detalles del funcionamiento, obteniendo así el proyecto final.

Finalizada esta breve introducción, en la que se ha situado el proyecto en su contexto ingenieril, y se ha hablado sobre sus objetivos y la estructura de trabajo, se pasará al **capítulo 2**, denominado “Estado del arte”. En este segundo apartado se comentará cual es la situación de este tipo de proyectos de automatización en la actualidad, intentando conectar así las ideas y conceptos que aquí se desarrollarán con el mundo real. Así, se entrará en un análisis pormenorizado de cada una de las partes del trabajo, indicando cuales son sus aplicaciones reales hoy en día, y sus perspectivas de futuro dentro del mundo de la industria.

Además, se realizará una revisión del mercado, analizando difrentes opciones ante las cuestiones que este trabajo pretende resolver y justificando la elección de unas frente a otras. Se detallarán pues todos los criterios seguidos para tales decisiones, los cuales pueden ser muy diversos, desde la disponibilidad hasta el criterio económico, pasando por criterios de carácter ingenieril.

El **capítulo 3** se centrará en la parte hardware del dispositivo. Se entrarán a describir las características de cada uno de los componentes elegidos, partiendo de la base de que la comprensión de su funcionamiento es el pilar fundamental sobre el que se sustentará el posterior éxito del proyecto.

Al existir partes del proyecto que se han diseñado y fabricado expresamente para él, como son la cinta transportadora y el PCB, en este capítulo también se describirá paso a paso el proceso de obtención de los mismos, incluyéndose referencias a todos los materiales y recursos usados. El objetivo de esta parte es doble, por un lado, describir todas las tareas realizadas, y por otro, servir de guía para aquellos que quisieran reproducir el proyecto.

Posteriormente, en el **capítulo 4** se hablará sobre el software. Este apartado comieza con una explicación sobre el software con el que se ha desarrollado este trabajo, el programa Code Composer Studio, enfocado a la programación de dispositivos de Texas Instrument. Tras esto, se pasará a una descripción de todas las pruebas individuales realizadas con los difrentes dispositivos, de forma que se pueda llegar a comprender las entrañas de este proyecto.

Se incluirán también explicaciones sobre el código principal, así como los pertinentes diagramas de flujos y esquemas, presentándose de esta forma una visión global de como se realizan las operaciones necesarias para llevar a cabo la clasificación por colores de los objetos.

En el **capítulo 5**, una vez descrito todo el proyecto, se analizarán sus posibles ampliaciones, de forma que, partiendo de la base aquí descrita, se pueda continuar el trabajo, haciendolo más rico y complejo. También se incluirán una serie de aspectos a mejorar de él, los cuales no se abordaron por falta de tiempo o por quedar fuera de los límites del mismo, pero que sería interesante revisar o corregir con el fin de perfeccionar lo más posible el trabajo realizado.

Y ya, por último, en el **capítulo 6** se expondrán las conclusiones que se han extraído tras la realización del proyecto, así como todas las enseñanzas y lecciones aprendidas durante la elaboración de este Trabajo Fin de Grado.

2 ESTADO DEL ARTE

Una vez definido el sistema a desarrollar, podemos concretar que, dentro del mundo ingenieril, nuestro proyecto se enmarca dentro de lo que se conoce como sistema empotrado o embebido. Desde su desarrollo, este tipo de sistemas se han adentrado a gran velocidad en el mundo de la automatización industrial, debido a que mejoraban en muchos aspectos a los anteriores controles centralizados. Gracias a ellos se ha conseguido, por tanto, pasar a un modelo de control descentralizado, evitando que un fallo en el control central ocasione el paro de un gran número de procesos, además de ser un método económico y efectivo para la resolución de tareas [4].

Al comienzo de este capítulo, se buscará aportar una visión general sobre los sistemas empotrados, así como un repaso a cada una de las partes que los componen, de forma que pueda comprenderse su funcionamiento. Posteriormente se entrarán en detalles sobre nuestro sistema embebido, siguiendo de esta forma una estructura que va desde lo general a lo particular.

Un sistema empotrado o embebido *es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real* [2]. Es decir, son un tipo de sistemas diseñados para funciones concretas, y que están basados en ordenadores, pero no en el sentido que los entendemos en nuestro día a día, sino que usan pequeñas computadoras, con capacidad de procesamiento reducida, las cuales se integran dentro del propio sistema.

En la actualidad, el uso de este tipo de sistemas esta muy generalizado, debido a que permiten desempeñar infinidad de tareas, y siempre con un coste muy ajustado. Por este motivo, podemos encontrarlos en multitud de aplicaciones, desde pequeños aparatos electrónicos de uso diario hasta en aeronaves, pasando por el control de procesos dentro de una industria.

Figura 2-1. Difrentes campos de aplicación de los sistemas empotrados. [2]

[2] Imagen tomada de https://www.exa.unne.edu.ar/. Fecha de consulta: Agosto de 2020

Las principales características de este tipo de sistemas, por tanto, son:

- Número limitado de funciones.
- Precio ajustado con respecto a la tarea a desempeñar.
- Funcionamiento en tiempo real.
- Funcionamiento autónomo.
- Bajo consumo de energía.
- Estrecha relación entre hardware y software.
- Requerimiento de dispositivos de E/S específicos a la aplicación.
- Robustos y con un consumo de energía reducido

Son dos fundamentalmente las partes que integran un sistema empotrado: computadora y transductores. La computadora, de la que ya hemos hablado un poco, es la parte que se encarga del manejo y tratamiento de los datos. Por tanto, es la parte del sistema que toma las decisiones, teniendo así el control sobre el resto de los elementos.

Por otro lado, un transductor es un *dispositivo que transforma el efecto de una causa física, como la presión, la temperatura, la dilatación, la humedad, etc., en otro tipo de señal, normalmente eléctrica, o viceversa* [3]. Su función principal es, de este modo, permitir la interacción del sistema empotrado con el medio. Esta tarea la consigue gracias a su capacidad para transformar un determinado tipo de energía en energía eléctrica, o viceversa, toda vez que la energía eléctrica es la única que entiende una computadora. En resumen, los transductores son el nexo de unión de la computadora con el exterior.

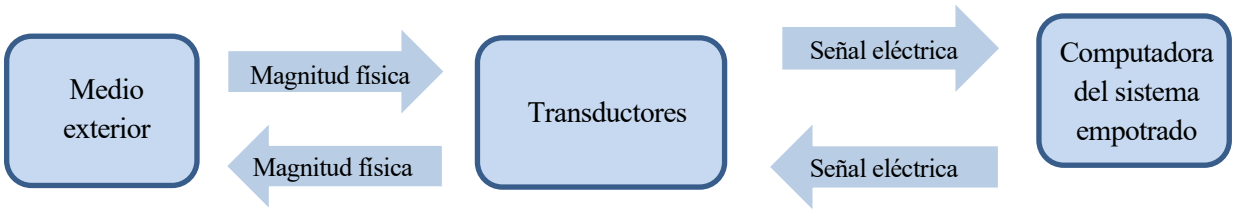


Figura 2-2. Función de los transductores en un sistema empotrado.

Y atendiendo a su función concreta dentro del sistema empotrado, podemos decir que los transductores se clasifican en 4 grupos:

- **Transductores sensores:** son aquellos cuya principal tarea es la de recoger alguna magnitud física del exterior y transformarla en una magnitud eléctrica entendible por el computador. Así, para cada magnitud física que podamos imaginar, hay un sensor asociado, existiendo, por ejemplo, sensores de presión, de proximidad, de luz, acústicos, de temperatura….
- **Transductores de mando:** en esencia, son parecidos a los transductores sensores, es decir, capturan de alguna forma información del exterior y la transfieren al procesador de datos. La principal diferencia radica en la finalidad de estos dispositivos, y es que no trabajan con ninguna magnitud física medible, sino que su objetivo es transladar al controlador algún estímulo proveniente de un ser humano. Como ejemplos podemos mencionar un potenciómetro o una botonera.

- **Transductores indicadores:** también conocidos como de monitorización, sirven, del mismo modo que los anteriores, para la comunicación entre un ser humano y la computadora, pero esta vez en el otro sentido, es decir, transmitiendo de algún modo información interna del sistema hacia el exterior. Su principal exponente es la pantalla.

- **Transductores actuadores:** son los encargados de llevar a cabo en el mundo físico las órdenes del controlador. Dentro de este grupo, podemos mencionar los motores, las bombas, las válvulas, los brazos robóticos….

No todos los sistemas embebidos tienen porqué tener integrados todos los tipos de transductores, al contrario, es bastante raro que lo hagan. Un claro ejemplo lo podemos ver con los aparatos de medida. En este caso, los dispositivos poseerán transductores sensores, para detectar aquello que se desea medir, y transductores indicadores, de forma que la medida sea comunicada a la persona que la realiza. Podrían tener transductores de mando, posibilitando configurar de algún modo la medida, pero en ningún caso tendrían actuadores, ya que no esta en la naturaleza de estos aparatos realizar modificaciones en el medio.

Para el caso de los sistemas orientados a la automatización, como es el que nos ocupa, serán imprescindibles tanto sensores como actuadores, pues sin la capacidad de recibir información del exterior, y en base a esta, realizar modificaciones en el medio, la tarea de automatizar sería imposible. La inclusión del resto de tipo de transductores sería opcional, de forma que una persona pudiera monitorear y influir de algún modo en el proceso. Este tipo de cosas no serán consideradas en nuestro proyecto, por lo que en él nos encontraremos únicamente con sensores y actuadores.

Nuestro sistema constará, por tanto, de dos tipos de sensores, de presencia y de detección de color, dos actuadores, como son la cinta y el brazo robótico, y por supuesto, una unidad de procesado de información. Durante los siguientes subapartados, trataremos en profundad cada uno de estos elementos, haciendo una descripción general de de ellos, para posteriormente elegir los modelos que se integraran en nuestro diseño.

2.1. Actuadores

El mundo de los actuadores es tremendamente basto y complejo, ya que, para poder desarrollar un actuador, es necesario tener amplios conocimientos del medio sobre el que va a actuar. Por ejemplo, para desarrollar un motor eléctrico será imprescindible basarse en conceptos de electricidad y de mecánica, de la misma forma que para diseñar un elemento que actue sobre la temperatura serán necesarios conocimientos de termodinámica o para construir una bomba hay que tener una base sólida en mecánica de fluidos. Con esto se quiere remarcar que se trata de un campo tremendamente multidisciplinar, en el que la electrónica suele jugar el papel de control de los dispositivos, no encargándose nunca del desarrollo desde cero de los mismos.

Antes de comentar los actuadores que serán parte de nuestro sistema, combiene hacer un breve apunte sobre otras opciones que se podrían haber implementado, y con las que se hubiera alcanzado de igual forma el objetivo propuesto.

En lo referente a la clasificación en si de los objetos, el brazo robótico, podría haberse sustituido, por ejemplo, por un grupo de pistones que sacaran a los objetos de la cinta, de forma que cada pistón empujaría solo los objetos de un determinado color, agrupándolos. Otra opción hubiera sido la implementación de una plataforma giratoria al final de la cinta, la cual se posicionaría según el color del objeto que fuera a caer en ese instante de tiempo.

Tanto el brazo robótico como las opciones alternativas están basadas en el mismo concepto, el uso de motores para seleccionar las piezas, por lo que finalmente se escogió la opción del brazo por resultar más vistosa para el proyecto, a parte de añadirle más dificultad.

La cinta transportadora, por su parte, es el elemento de transporte de objetos entre distancias pequeñas por antonomasia, implementado en la práctica totalidad de las líneas de producción, por lo que no se han encontrado alternativas a este elemento a la hora de desarrollar el trabajo.

2.1.1 Brazo robótibo

Según la Federación Internacional de Robótica (IFR), **un brazo robótico** o robot manipulador es *una máquina de manipulación automática, reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos especiales para la ejecución de trabajos diversos en las diferentes etapas de la producción industrial, ya sea en una posición fija o en movimiento* [5].

Esto es, estamos hablando de un brazo mecánico articulado y programable, que trata de reproducir lo más fielmente posible los movimientos de un brazo humano. Esta formado por un número de eslabones unidos por articulaciones, las cuales permiten el movimiento relativo de unos frente a otros. En la punta posee una pieza que se denomina efector final, mas comúnmente conocido como “mano”, que es con la que el brazo robótico realiza la tarea que se le ha encomendado. Este efector final cambia según la aplicación para la que queramos emplear al robot.

Figura 2-3. Brazo robótico industrial.[3]

Los robots manipuladores más extendidos son los de 3 eslabones y 3 articulaciones, como el de la figura 2-3. En la práctica, se usan dos tipos difrentes de articulaciones para esta clase de robots, rotacionales y prismáticas (hay más, pero no se usan), por lo que existe una clasificación de los brazos robóticos en función de sus articulaciones y los ejes que estas siguen:

[3] Imagen tomada de http://www.reporteroindustrial.com/temas/Brazos-roboticos+50002448. Fecha de consulta: Agosto de 2020

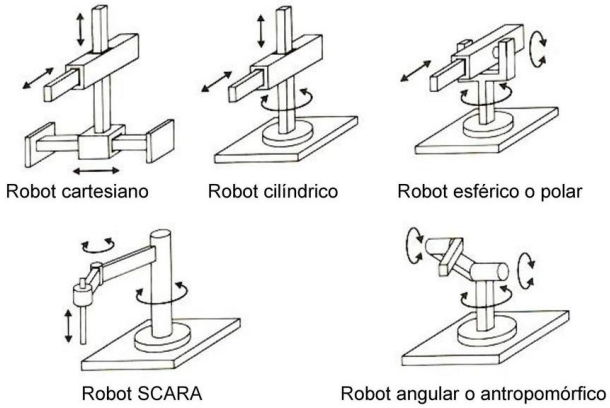


Figura 2-4. Clasificación de los manupuladores según sus acticulaciones.[4]

El hecho de que tanto las articulaciones como la mano sean modificables, y de que sean programables, hace que estos dispositivos tengan una amplia implementación en el mundo industrial, debido a su gran capacidad para adaptarse a cualquier tarea. Así, por ejemplo, podemos destacar su uso en campos como la industria automovilística (Figura 1-1), en el desarrollo de componentes electrónicos, en cadenas de pintura, en la distribución y empaquetado de almacenes, en tareas de "pick an place" [5] e incluso en el campo aeroespacial, donde famosos manipuladores espaciales como el Canadarm o Canadarm 2 son los máximos exponentes de este sector.

El uso de este tipo de máquinas aporta una serie de ventajas respecto al trabajo manual, entre las que podemos destacar:

- Capaces de realizar tareas repetitivas sin cansancio.
- Realización de tareas de forma rápida y precisa.
- Pueden trabajar en condiciones en las que no podría hacerlo un humano.
- Pueden realizar tareas potencialmente peligrosas para los humanos.
- Pueden escalarse según el tamaño necesitado, llegando los más grandes a desplegar potencias mucho mayores de las que un humano puede manejar.
- Reprogramables, por lo que son adaptables a un gran número de tareas.
- Reducción de costes.

El estudio del movimiento de los brazos robóticos, es decir, de como conseguir que sigan una trayectoria deseada con orientación correcta, es algo bastante complejo, por lo que, dentro de la robótica, existe una rama que se dedica íntegramente a esta tarea. Así, mediante estudios cinemáticos y dinámicos basados en la morfología y articulaciones de los mismos, es posible llegar a controles fluidos y muy precisos de este tipo de elementos. Esta tarea, por su complejidad, queda fuera de los límites de este trabajo, sin embargo, es importante tener presente que este estudio debería ser tratado en caso de abordar un trabajo más profundo sobre brazos robóticos.

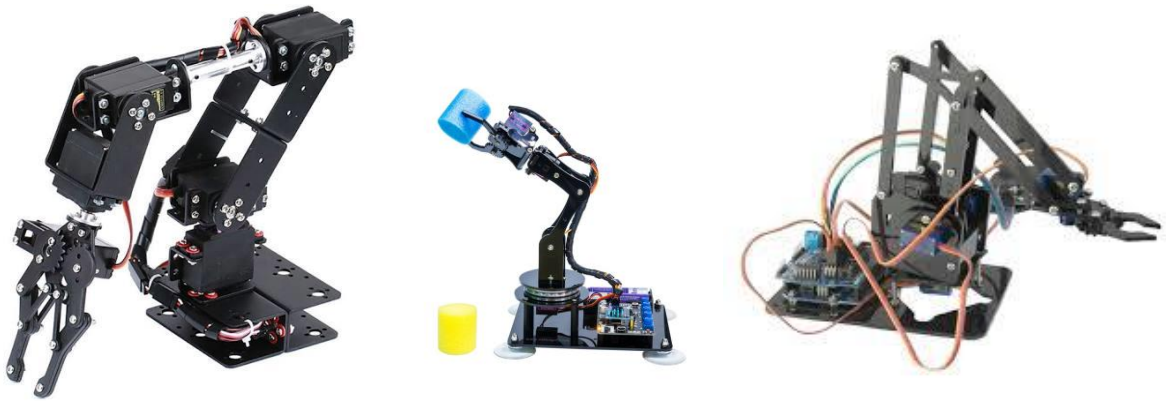
[4] Imagen tomada de https://docplayer.es/16277041-Introduccion-tema-2-morfologia.html. Fecha de consulta: Agosto de 2020
[5] Tarea de recoger objetos de un lugar y colocarlos en otro, de forma precisa. Numerosas industrias llevan a cabo este tipo de tareas en sus líneas de producción.

Una vez realizada una revisión sobre este tipo de elementos, pasaremos a realizar una búsqueda en el mercado de aquellos que se adecuen al proyecto, de forma que se pueda elegir el que se va a implementar. Debido a que nuestro objetivo es realizar una maqueta a pequeña escala, el brazo buscado deberá tener dimensiones reducidas. Además, otro requisito fundamental es que su actuador final sea tipo pinza, de forma que se puedan realizar la manipulación de objetos pretendida.

Con estas restricciones, tras rastrear un poco la oferta disponible, se encuentran varias opciones que podrían ser válidas:

Tabla 2-1. Comparativa de los brazos robóticos

MARCA	MODELO	DIMENSIONES	MATERIAL	¿SERVOS?	PRECIO
MBEN	6 DOF BRAZO LIBRE [6]	43 x 10 x 10 cm	Metal	No	49,53 €
ADEPT	Brazo Robótico 5DOF [7]	35 x 7 x 9 cm	Metacrilato	Si	77,69 €
EBOTICS	Arm robot [8]	6.8 x 5,3 x	Metacrilato	Si	52,43 € (en oferta)



(a) (b) (c)

Figura 2-5. Imágenes de los brazos encontrados

Finalmente, el brazo seleccionado para nuestro proyecto será el de la marca Ebotics. Esta decisión se ha basado fundamentalmente en el criterio económico, ya que tanto las otras opciones aquí presentadas, como otros muchos modelos, cumplian las especificaciones buscadas, pero en todos los casos a un precio mayor. Otro punto a favor de este dispositivo es que incluye los servomotores necesarios para el funcionamiento del robot, lo que ahorra el esfuerzo de buscar unos que se adaptaran al modelo elegido.

2.1.2 Cinta transportadora

Una **cinta o banda transportadora** es un *dispositivo mecánico formado por una banda móvil que se mueve entre dos tambores y cuya función es trasladar mercancías, equipajes, personas, etc.* [9]. Estos componentes se emplean cuando existe la necesidad de transportar una gran cantidad de objetos entre dos posiciones fijas. La mayoría están movidos mecánicamente, aunque existen algunos tipos donde es únicamente la fuerza de la gravedad la que empuja los objetos a través de la cinta.

Figura 2-6. Cinta transportadora industrial [6]

Como características principales de este tipo de dispositivos, podemos destacar:

- Permiten el transporte automatizado de elementos.
- Ocupan un lugar fijo, estableciento una ruta que conecta dos puntos.
- Suelen estar apoyadas sobre el firme, aunque existen otras configuraciones, por ejemplo, montadas en el techo.
- Generalmente mueven cargar discretas, aunque pueden prepararse para el transporte de cargas continuas.
- El flujo de materiales sobre ellas suele ser unidireccional.

Existen multitud de tipos de cintas transportadoras, de rodillos, de cintas planas, de cadenas, con ruedas, con listones, etc. Sin embargo, la idea que subyace a todas ellas es la misma, y es simplemente el tensar algún tipo de cinta y conseguir su movimiento mediante el giro de unos rodillos. Esta simpleza es una de las características fundamentales por las que hoy en dia son uno de los elementos más usados en todo tipo de industrias. Podemos destacar su uso en la industria agrícola, alimentaria, farmacéutica, en el campo de la minería, así como en cualquier fábrica que tenga sistemas de producción en línea.

En cuanto a la cinta transportadora necesaria para llevar a cabo el proyecto, su principal requisito es que sea de dimensiones reducidas, y que su precio no sea demasiado desorbitado. Tras una intensa búsqueda de un producto de estas características, encontramos que en el mercado actual no existe, pues todas las cintas transportadoras, pese a que su uso no sea industrial, tiene unos precios demasiado elevados, no bajabando en ningún caso de los 200 €.

Solo una empresa colombiana tenia en su catálogo un kit de una cinta transportadora a un precio razonable. Se trata del modelo MIGUE de la empresa ZTROBOTIC [10]. Esta es una cinta transportadora de 30 cm de largo, que además cuenta con una estructura para colocar sensores, elemento que resultaría muy útil en este proyecto. Al cambio, el precio de este producto es de 9 euros.

[6] Imagen tomada de https://www.ultimationinc.com/es/replacement-parts/cinta-transportadora/cinta-transportadora-24-pulg-ancho-x-20-pulg-ancho-x-11-pies/. Fecha de consulta: Agosto de 2020

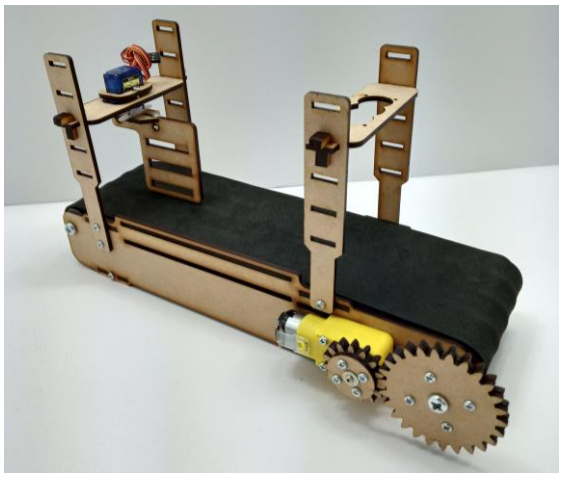


Figura 2-7. Cinta transportadora MIGE

Sin embargo, sería imposible finalmente incorporar este producto al proyecto, debido a que, tras una serie de gestiones, el precio para su envio a España rondaría los 120 €, cantidad que de nuevo hace inviable desde el punto de vista económico su adquisición.

Tras investigar proyectos de electrónica similares al que aquí se desarrolla, se llegaría a la conclusión de que la única forma de poder tener una cinta a un precio competitivo es fabricándola, luego esta sería la opción final elegida. En capítulos posteriores se comentará el proceso seguido para la construcción de esta cinta, así como el resultado final obtenido.

2.2. Sensores

Al igual que ocurría con los actuadores, el campo de los sensores es también una disciplina que implica a otras ramas del conocimiento a parte de la electrónica. En este caso, son conceptos físicos los que se esconden detrás de estos aparatos. Así, la idea sobre la que se sustenta un sensor será una propiedad física que haga que una determinada magnitud eléctrica sea sensible a la magnitud que se desea medir. La propiedad física por la cual un sensor funciona es lo que se conoce como su **fundamento físico**.

Cada tipo de sensor tendrá su propio fundamento físico, existiendo incluso muchos sensores que miden una misma magnitud, pero sustentados en principios difrentes. De esta forma, por ejemplo, el fundamento físico que hay detrás de un sensor de luz (LDR) es el cambio de resistencia de un material al ser iluminado, mientras que un sensor de proximidad puede estar basado en conceptos inductivos, capacitivos, ópticos, ultrasónicos y electromagnéticos, teniendo cada uno propiedades difrentes, por lo que serán apropiados para difrentes aplicaciones.

Por su parte, el papel que juega la electrónica en un sensor es el de traducir esta sensibilidad ante la magnitud medida en algo entendible por el sistema de control. Existe así, dentro de la disciplina de la instrumentación electrónica (la cual de encarga de los aparatos de medida), una parte dedicada íntegramente a hacer esta traducción, denominada acondicionamiento de señal.

Las estapas necesarias para llevar a cabo un acondicionamiento de señal serían las siguientes:

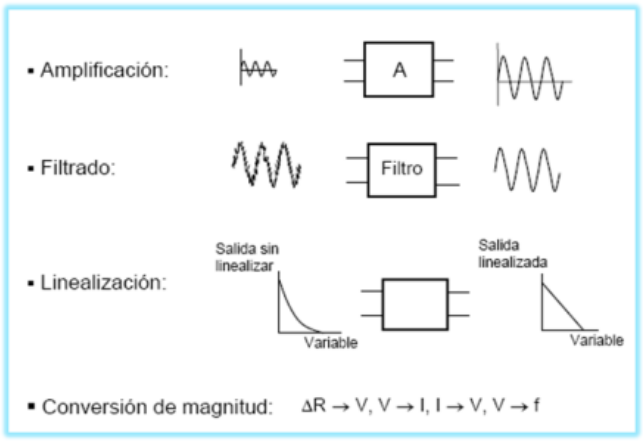


Figura 2-8. Estapas de un acondicionamiento de señal [7]

- Amplificación: Se realiza cuando la intensidad de la señal de salida del sensor es demasiado baja. Gracias a ella, se consigue que la señal medida se encuentre en un rango de valores aceptables para el posterior trabajo en el procesador. Este trabajo lo realiza por medio del uso de amplificadores operaciones.

- Filtrado: Su tarea es la eliminación del ruido, siempre presente en las medidas eléctricas. Así, gracias al uso de difrentes tipos de filtros, se eliminan aquellas componentes de frecuencia no deseadas, quedando la señal medida mucho más limpia.

- Linealización: muchos sensores poseen dependencias exponenciales respecto a la magnitud medida, por lo que en esta etapa se transforman dichas dependencias a lineales, con el fin de ser mejor manejadas posteriormente.

- Conversión de magnitud: en multitud de ocasiones es conveniente transformas el tipo de variación eléctrica que provoca la magnitud medida. Claro ejemplo lo tenemos en las antes comentadas LDR, cuya variación de resistencia deberá ser traducida a una variación de tensión, por ejemplo, para poder ser manejada.

La inclusión de unas u otras etapas en un sensor dependerá de las características de su medida, no siendo necesarias, por tanto, todas ellas en algunos casos.

Suele existir una etapa más antes de pasar de la medida al controlador, y es la conversión analógico-digital. En ciertos casos, la salida después del resto de etapas es una variable continua, y como es bien sabido, los procesadores solo entienden variables discretizadas, por lo que la misión de esta parte será realizar dicha conversión. Esta etapa no suele venir integrada en el propio sensor, sino que suele ser uno de los periféricos del controlador donde se encuentra el procesador.

Para clarificar este concepto, en la siguiente figura podemos observar como sería la discretización (parte roja) de una señal continua (parte gris):

[7] Imagen tomada de https://instrumentacionelectronicai.wordpress.com/tag/instrumentacion/. Fecha de consulta: Agosto de 2020

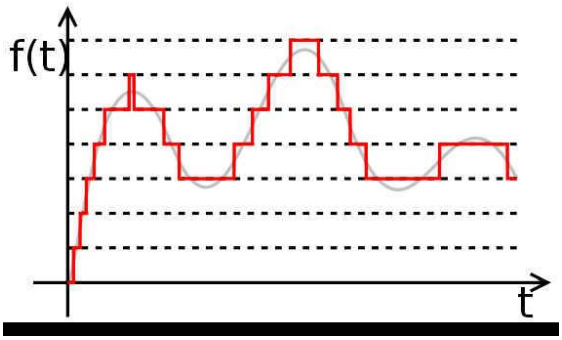


Figura 2-9. Conversión analogico-digital de una señal [8]

La mayoría de los sensores que se venden actualmente en el mercado ya vienen con todos estos temas resueltos, de forma que vienen en unos encapsulados que permiten conectarlos directamente al dispositivo de control, y empezar a hacer mediciones de forma no demasiado compleja. Es por eso que no se entra en un estudio demasiado pormenorizado de esta parte, pues los sensores que se obtengan para desarrollar el proyecto ya vendrán preparados electrónicamente para empezar a realizar mediciones.

2.2.1 Sensor de color

Haciendo una búsqueda sobre difrentes sensores para medir el color de los objetos, nos encontramos con que, para este tipo de proyectos, existen dos opciones fundamentalmente: El TCS3200 y el TCS43725.

Ambos ofrecen una medida RGB, es decir, no te dicen directamente de que color se trata, sino que ofrecen la "cantidad" de cada color primario presente en el objeto. Así midiendo un objeto rojo, ofrecerán unos altos valores de R (red - rojo), y unos valores bajos de G (green - verde) y de B (blue - azul). Si en cambio medimo algo de color morado, que es mezcla de rojo y azul, dará unos altos valores en rojo y azul, y bajos en verde. De esta forma se pueden detectar multitud de colores, aunque tampoco será posible afinar demasiado al alejarnos mucho de los colores primarios.

Finalmente, el sensor elegido para desarrollar el proyecto será el TCS3200 [11], pues al ser algo más antiguo que el otro, existe mayor cantidad de información disponible sobre él, por lo que su implementación en el proyecto será mas sencilla.

2.2.2 Sensor de presencia

Existen una gran cantidad de sensores de presencia en el mercado, cada uno de ellos basado en un fundamento físico difrente, como ya se ha comentado con aterioridad.

Tras analizar sus características, se llega a la conlcusión de que el más apropiado es el módulo de detección de obstáculos por infrarrojos IR FC-51 [12]. Posee varias ventajas que lo difrencian notablemente del resto, como su reducido precio, su sencilla implementación, o su capacidad para regular la distancia de detección del objeto de una forma muy sencilla, mediante un potenciómetro.

2.3. Unidad de control

El componente más importante de un sistema embebido es su unidad de control, pues se trata de la pequeña computadora que es responsable de procesar los datos y dirigir todas las tareas dentro del sistema. El microcontrolador [13] es la unidad de control por antonomasia, y esta formado por las siguientes partes:

[8] Imagen tomada de https://www.puntoflotante.net/CONVERTIDORAD.htm. Fecha de consulta: Agosto de 2020

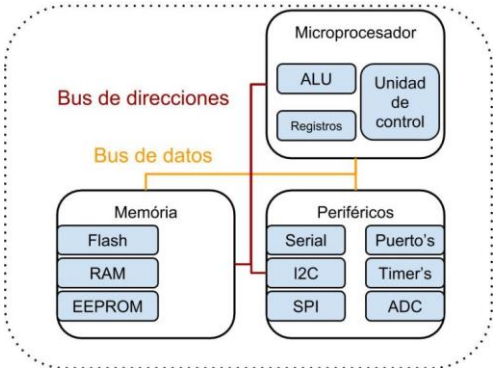


Figura 2-10. Esquema de un microcontrolador [9]

- Microprocesador: Es la parte que interpreta las instrucciones, mediante la realización de operaciones básicas de aritmética, lógica y de E/S del sistema. A su vez, esta formada por:
    - ALU: su función principal es el realizar las operaciones, mediante el uso de circuitos electrónicos digitales del tipo combinatorios (compuertas, sumadores, multiplicadores)
    - Unidad de control: su función es la de extraer las instrucciones de memoria, las descodifica y las ejecuta, llamando a la ALU cuando sea necesario.
    - Registros: Son la memoria del microprocesador.
- Memoria: Es el lugar donde se almacenan los datos dentro del microcontrolador. Existen varios tipos:
    - FLASH: es una memoria lenta, aunque relativamente grande, pues en ella se guardarán los programas para su ejecución.
    - RAM: es mas rápida y con menos capacidad que la FLASH. Su función es básicamente almacenar temporalmente aquellos datos que este usando el microprocesador, de forma que se le posibilite un acceso rápido a ellos.
- Periféricos de entrada y salida: Son los que permiten la cominucación del microcontrolador con el exterior. Existen difrentes módulos y no todos los microcontroladores deben tener de todos:
    - Comunicación: existen difrentes accesos para la comunicación con otros dispositivos. Dentro de los módulos de comunicación destacamos la UART (transmisión serie asíncrona), el I2C (comunicación serie síncrona) o el SPI (comunicación serie síncrona, difrentes caracteristicas).
    - Adecuación de señales: son el conversor analógico-digital y el conversor digital-analógico, encargados de la comunicación con el exterior en caso de que las variables que se reciben o se envíen deban ser analógicas.
    - Periféricos generales: dentro de este grupo tenemos tanto E/S de propósito general, como otros, por ejemplo, timers (permiten temporizar eventos) o PWM (modulación por anchura de pulso, permiten generar señales de salida con difrentes periodos de activación-desactivación).

La principal diferencia de este tipo de dispositivos respecto a los ordenadores de propósito general es que estos poseen menos recursos y capacidad de procesamineto. Otra diferencia fundamental es la de la arquitectura en la que están basados. Así, mientras que la mayoría de los ordenadores convencionales siguen la arquitectura de Von Neumann, los microcontroladores normalmente poseen una arquitectura tipo Harvard. Sin entrar mucho en

[9] Imagen tomada de https://hetpro-store.com/TUTORIALES/microcontrolador/. Fecha de consulta: Agosto de 2020

esta cuestión, la diferencia entre las dos es que, aunque en ambas memoria de programa y de datos están difrenciadas, en la arquitectura de Von Neumann solo existe un aceso para las memorias desde el procesador, debiendo compartir ambas memorias este acceso, mientras que en la de Harvard cada memoria tiene difrente aceso desde la CPU. Esto hace que la arquitectura de Harcard sea mucho más rápida que la de Von Neumann, sin embargo, para ordenadores de propósito general, donde el procesador tiene su propio encapsulado, y no se encuentra en el mismo que la memoria, como ocurre en los microcontroladores, implmentar una arquitectura de Harvard seria demasiado costodo en lo que se refiere a líneas de E/S del procesador.

Existe un tipo especial de microcontrolador llamado DSP, el cual esta preparado para el tratamiento de señales en tiempo real, sobre todo provenientes de un conversor analógico-digital. Estos dispositivos están caracterizados por ser capaces de realizar operaciones numéricas complejas a alta velocidad. De esta forma, porseen mayor capacidad de procesado que los microcontroladores convencionales, y un mayor gasto energético, además de ser mas complejos en cuanto a su manejo. Sus principales campos de aplicación son en sistemas de audio, video y en transmisión de datos digitales.

En cuanto a la unidad de control de nuestro proyecto, analizando lo que se necesitaría para el manejo de los elementos antes mencionados, encontramos que sería necesario disponer de un buen número de pines de propósito general, asi como módulos PWM y timers, y también la posibilidad de alimetar tanto a 3.3V como a 5 V.

Con un microprocesador no demasiado complejo, seríamos capaces de cubrir todas estas necesidades. Existen en el mercado una amplia gama de dispositivos de este tipo, como los de la marca ARDUINO o de la gama MSP430 de Texas Instrument, que además son bastante económicos.

Sin embargo, finalmente se ha optado por implementar un dispositivo DSP, concretamente el TMS320F28335 de la ya mencionada Texas Instrument. Pese al precio elevado de estos dispositivos, existía la posibilidad de obtenerlo temporalmente gracias a la universidad, consiguiendo asi que el proyecto fuera mucho mas complejo y con un mayor fundamento que si hubiera sido llevado a cabo con un dispositivo más sencillo, por lo que este ha sido el motivo fundamental para su elección.

3 HARDWARE

Durante todo este capítulo se analizarán los componentes que previamente se han seleccionado para llevar a cabo el proyecto. Este análisis detallado de cada componente será básico para comprender su funcionamiento, y así, poder lograr el funcionamiento correcto del sistema completo. Se comenzará por comentar todos los elementos que han sido adquiridos, dejando para el final la descripción de aquellos que han sido fabricados expresamente para este trabajo.

3.1 Sensor de color TCS3200

El sensor TCS3200 es un sensor óptico de reconocimiento de color. Posee una matriz de 64 fotodiodos, 16 con un filtro rojo, 16 con un filtro verde, 16 con un filtro azul y 16 sin filtro. Con cada grupo de fotodiodos realiza la medida del color correspondiente, ya que los fotodiodos, al tener el filtro, son sensibles solo a la luminosidad de su color. Los resultados son, por tanto, tres componentes, R (rojo), G (verde) y B (azul), cada una con la "cantidad" de este determinado color que detecte en la superficie.

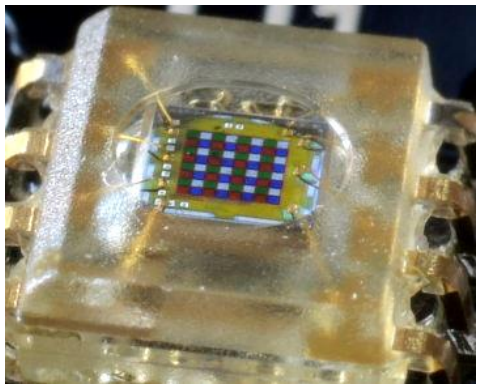


Figura 3-1. Detalle de la matriz de filtros del sensor[10]

La respuesta de los fotodiodos ante la luz es una variación de corriente, sin embargo, ante la dificultad que acarrea el trabajo con corrientes, este dispositivo incorpora una etapa de conversión de magnitud. Así, dentro del propio encapsulado, el sensor cuenta con un módulo conversor de corriente a frecuencia, de tal forma, que la salida del dispositivo será, según su propio datasheet [14], una onda cuadrada, activa el 50% del tiempo y desactivada el otro 50%, y cuya frecuencia será mayor cuanta mayor luminosidad se haya detectado de un color determinado.

En el datasheet podemos también encontrar las conexiones que posee el dispositivo para su configuración y para obtener las medidas, y son las siguientes:

[10] Imagen tomada de http://robots-argentina.com.ar/didactica/arduino-reconocer-colores-con-el-modulo-tcs230/. Fecha de consulta: Agosto de 2020

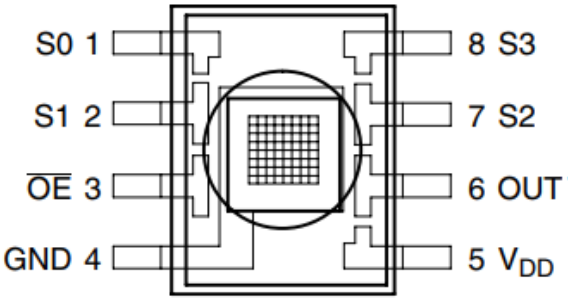


Figura 3-2. Pinout del sensor TCS3200 [14]

Los pines de Vdd y GND son para alimentación (5V) y tierra respectivamente. El pin de OUT será por el que saldrá esa señal cuadrada con la frecuencia dependiente del color, de la que hemos hablado anteriormente. Por su parte, el pin OE servirá para habilitar la medida. Es de activación negada, por lo que cuando queramos medir, deberemos ponerlo a 0.

El resto de pines, S0 – S4 servirán para configurar la medida. Los dos primeros, S0 y S1 se usan para seleccionar el escalado de frecuencias deseado a la salida, pues el conversor de magnitud integrado incorpora esta opción. Por su parte, S2 y S3 se usan para seleccionar el color a medir, porque solo se puede realizar la medida de un color simultáneamente. Los valores que deben tomar estos pines aparecen reflejados en las siguientes tablas:

Tabla 3-1. Configuración del escalado de frecuencia del TCS3200

Escalado de frecuencia	S0	S1
Sin escalado	Bajo	Bajo
2%	Bajo	Alto
20 %	Alto	Bajo
100%	Alto	Alto

Tabla 3-2. Selección del color a medir del TCS3200

Escalado de frecuencia	S0	S1
Rojo	Bajo	Bajo
Azul	Bajo	Alto
Sin filtro	Alto	Bajo
Verde	Alto	Alto

Al comprar el sensor, el encapsulado viene incrustado en un PCB, en el que también vienen integrados 4 diodos led. Estos se encienden al activar el pin de medida, iluminando la superficie de la que se quiere obtener el color, de forma que el resultado final sea lo más fiel posible a la realidad. Además, esta placa cuenta con un pin conectado a cada pin del sensor, pero más separados entre sí, de forma que es mucho mas fácil acceder a ellos para conectarlos.

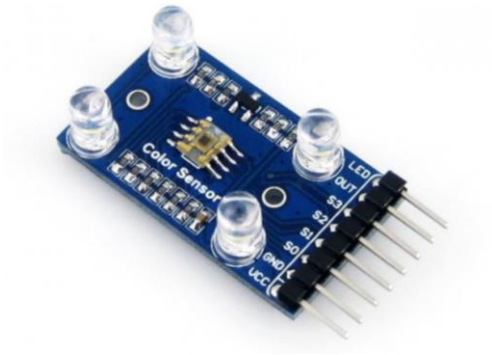


Figura 3-3. Sensor TCS3200 en su PCB [11]

3.2 Sensor de presencia infrarrojo IR FC-51

El fundamento teórico de este sensor de presencia por infrarrojos [15] es bastante sencillo, ya que se basa únicamente en la reflexión de la luz al chocar con un obstáculo. Así, este dispositivo cuenta con un LED emisor de luz infrarroja (invisible para el ojo humano) y un fotodiodo que detecta esta luz reflejada.

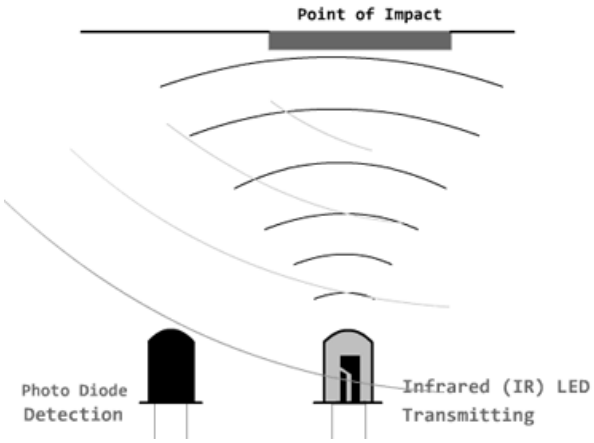


Figura 3-4. LED emisor de infrarrojo y fotodiodo receptor [11]

A parte de estos elementos, el sensor cuenta con un comparador, concretamente el LM393. Este elemento sirve para discernir si una detección de luz infrarroja por parte del fotodiodo es producida por un objeto que se encuentra más cerca del umbral deseado o no, según la intensidad de la radiación infrarroja recibida. Así, cuando un objeto se encuentre por debajo del límite deseado el sensor sacará un nivel lógico bajo por la salida, siendo alto mientras esto no se produzca. Este umbral de detección se puede modificar mediante un potenciómetro que actúa sobre el comparador, siendo los límites máximo y mínimo de detección de 2 cm a 30 cm.

La placa sobre la que se encuentra el sensor posee únicamente 3 pines, los correspondientes a alimentación (3V – 5V) y tierra, y el de salida, que seguirá el comportamiento antes descrito. Cuenta además con 2 LED's de cierta utilidad, uno que indica que el dispositivo está alimentado correctamente, y otro que se enciende siempre que se este detectando un objeto.

[11] Imagen tomada de https://www.luisllamas.es/detectar-obstaculos-con-sensor-infrarrojo-y-arduino/. Fecha de consulta: Agosto de 2020

3.3 Brazo robótico

El kit brazo robótico adquirido viene con las piezas del armazón pretroqueladas en metacrilato, con una pequeña capa de protección sobre ellas. Así, al extraer las piezas de la placa en la que se alojan, obtenemos lo siguiente:

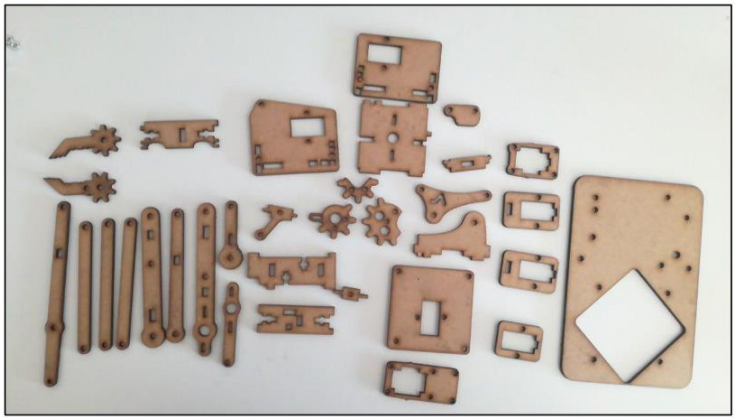


Figura 3-5. Piezas del armazón del brazo robótico [12]

Tras retirar la capa de protección, y siguiendo las instrucciones proporcionadas por el propio kit, se consigue de manera muy sencilla el montaje del todo el robot con sus motores integrados. Cabe destacar que para que estos puedan mover de forma conveniente al robot, es necesario realizar algunas uniones entre piezas con cierta holgura, de forma que la fricción sea mínima, pues sino estos pequeños motores no serán capaces de vencerla. El resultado final obtenido es el siguiente:

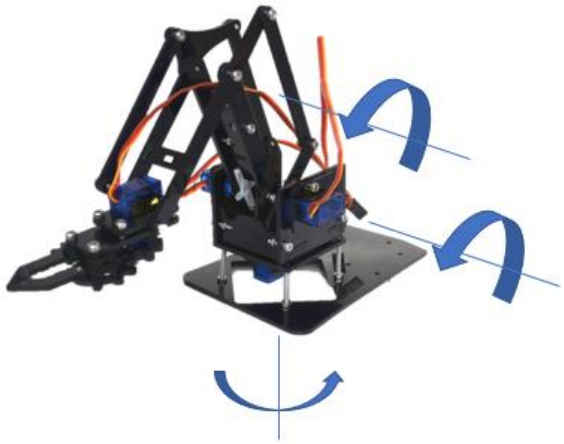


Figura 3-6. Brazo robótico montado con ejes de movimiento indicados

En la figura anterior también se han indicado los ejes de giro del robot. Como vemos, cuenta con 3 articulaciones rotacionales, una que gira según un eje perpendicular a la base, y dos que lo hacen con sus ejes paralelos a la misma. Además, tiene el movimiento de apertura y cierre de la garra. Con el control de estos 4 movimientos, podremos ser capaces de realizar cualquier operación de recogida y manipulación de elementos, que es lo que se pretende.

[12] Imagen tomada de http://bibing.us.es/proyectos/abreproy/92610/fichero/TFG-2610+SOSA+ALEM%C3%81N.pdf. Fecha de consulta: Agosto de 2020

De esta forma, el control del robot se basa en saber manejar a nuestro antojo los motores que incluye, que en este caso son los servomotores SG90. En el siguiente subapartado, por tanto, se realizará un análisis sobre este tipo de dispositivos.

3.3.1 Servomotores SG90

Un servomotor es un tipo especial de motor que no está pensado para el movimiento continuo, sino para colocar su eje según un ángulo determinado y luego mantenerse fijo en esa posición. Estos dispositivos poseen una circuitería electrónica interior para el control de la posición, basada en el uso de un potenciómetro conectado al eje. Este potenciómetro es el que indica la posición del motor, pues forma un divisor de voltaje cuya salida varía en función del ángulo en que se encuentre el eje. Para introducirle al control del motor el ángulo deseado, se le inyecta una señal PWM con unas características determinadas, cuya anchura de pulso determinará la posición en que se quiere fijar el eje.

Acoplado al motor también encontramos un juego de engranajes, que permiten elevar el torque en detrimento de la velocidad, ya que en estos dispositivos interesará más que el movimiento sea potente que rápido. En la siguiente figura se puede apreciar como es el interior de un servomotor:

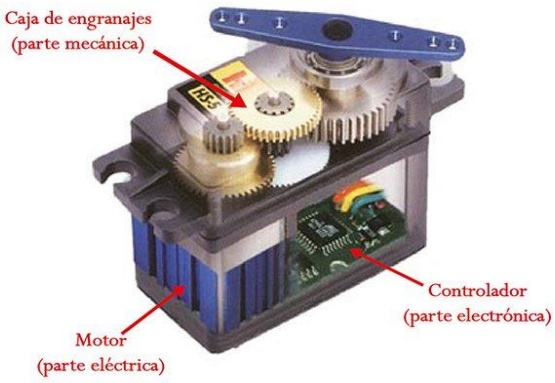


Figura 3-7. Interior de un servomotor [13]

Las características fundamentales de los servomotores acoplados al brazo robótico, el modelo SG 90, según su datasheet [16], son las siguientes:

- Peso: 9g
- Dimensiones: 22.2 x 11.8 x 31 mm
- Torque: 1.8 Kgf/cm
- Velocidad de operación: 0.1s/60 grados
- Voltaje de operación: 4.8 V (~ 5V)
- Temperatura de funcionamiento: 0 - 55º
- Rotación: 180º (90º en cada dirección).

Algunas de estas características serán importantes a la hora del desarrollo del proyecto, como, por ejemplo, la de la velocidad de operación, que es bastante elevada. Eso podría ocasionar problemas tanto a la hora de

[13] Imagen tomada de http://panamahitek.com/que-es-y-como-funciona-un-servomotor/. Fecha de consulta: Agosto de 2020

conseguir un movimiento fluido y sin estridencias del robot, como para la manipulación de objetos. En el siguiente capítulo comentaremos como resolver este problema desde el control del dispositivo.

Otro de los aspectos importantes será el de la rotación, ya que nuestro robot, por su propia construcción, no permite una rotación de 180 º de todos los motores. Por consiguiente, cuando se programe el movimiento del robot, deberá prestarse especial atención en no llevar a los motores a posiciones imposibles para ellos, ya que esto podría ocasionar la rotura de los mismo.

El servomotor solo dispone de 3 conexiones, las de alimentación y tierra, y una tercera por la que se le debe introducir la señal PWM de control de la posición. Las características de esta señal y el cómo generarla son temas que serán abordados en el siguiente capítulo, dedicado al software.

3.4 DSC TMS320F28335

Aunque con anterioridad hemos indicado que este dispositivo es un DSP, realmente su fabricante, Texas Instrument, lo denomina como DSC (Digital Signal Controller), pues considera que se trata de un dispositivo a caballo entre un DSP y un microcontrolador. De DSP posee su gran capacidad de cáculo, y de microcontrolador, su amplia gama de periféricos, así como un buen manejo de las interrupciones.

Según su datasheet [17], sus principales características son:

- Tecnología CMOS, con reloj de 150 Mhz.
- CPU de 32 bits, con arquitectura Harvard, procesamiento y respuesta rápida a interrupciones y código eficiente en C/C++.
- Control DMA (direct memory access) para seis canales. Esto implica una transferencia de datos directa entre memoria y periféricos, sin intervención de la CPU, lo que hace el proceso más rápido.
- Interfaz asíncrona externa de 16/32 bits, para ampliación.
- 256 kb de memoria Flash y 34 kb de RAM..
- Cuenta con una Boot ROM de 8 kb, así como con difrentes modos de arranque.
- Cerradura de seguridad de 128 bits que protege bloques de OTP/Flash/RAM, para evitar ingeniería inversa.
- 88 pines de propósito general multiplexados programables individualmente.

La arquitectura interna del dispositivo es la siguiente:

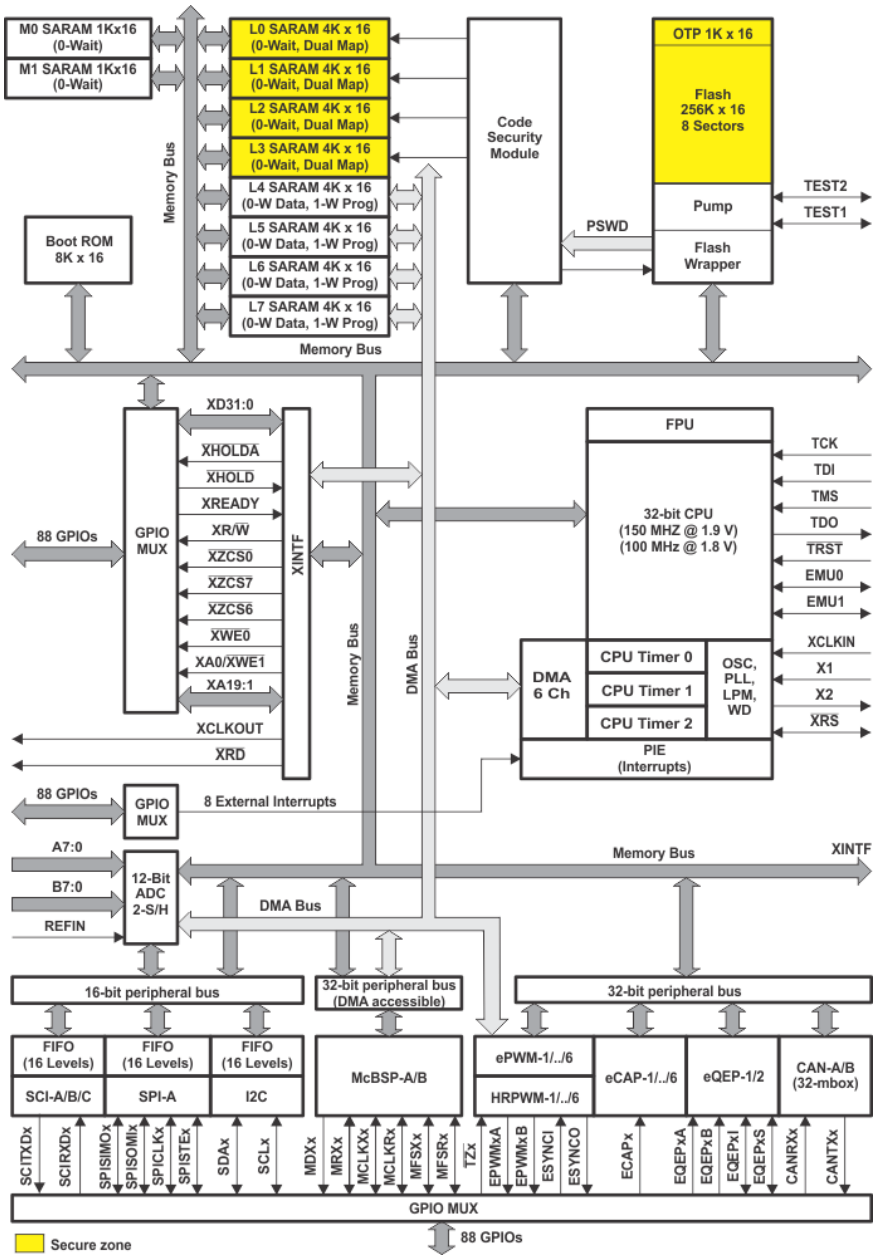


Figura 3-8. Arquitectura interna TMS320F28335

En la parte superior del diagrama podemos observar las memorias, la RAM, la boot ROOM y la flash. Los bloques que se encuentran en amarillo son aquellos protegidos por la cerradura de seguridad antes comentada. En la parte central derecha vemos la CPU y en la parte inferior tenemos situados todos los periféricos. El diagrama también nos permite ver los buses que interconectan todos los elementos. Este dispositivo cuenta con 2 buses internos para lectura de instrucciones (uno de 22 bits de direcciones y otro de 32 bits de datos) y 3 buses internos para lectura y escritura de datos (32 bits).

En cuando a los periféricos, el TMS320F28335 tiene integrados:

- 3 Timers de 32 bits cada uno: Sirven para temporizar eventos, generando una interrupción cuando su cuenta llega a 0.
- Watchdog: Timer especial que esta siempre funcionando, y que deberá ser refescado periódicamente. Si esto no se hace, provoca que la CPU se reinicie. Es un dispositivo de control para evitar que el sistema se quede colgado.

- CAD de 12 bits de resolución: Sirve para convertir datos analógicos a digitales, y asi poder ser procesados por el DSP. Posee 16 canales y 2 “sample and hold”. Su ratio de conversión es de 80 ns.
- 6 PWM: cada uno con 2 salidas, lo que hace un total de hasta 12 salidas PWM. Sirven para crear salidas con pulsos modulados.
- 88 pines de propósito general: pueden configurarse como de entrada o de salida, para leer o escribir valores lógicos. El resto de periféricos usan estos mismos pines, por lo que todos se encontrarán multiplexados y tendrán varias funcionesselccionables.
- eCAP: su función es la de medir tiempos de eventos externos.
- eQEP: sirve para trabajar con encoders incrementales de máquinas rotatorias.
- Periféricos de comunicación: sirven para implementar difrentes tipos de comunicación del DSP con otros dispositivos. A saber, I2C, SPI, SCI (incluye 3), CAN (incluye 2) y McBPS (incluye 2).

El DSP viene acompañado de un PCB, denominado Experimenter Kit, donde se pincha la Control Card, que es donde se aloja el TMS320F28335. Este PCB sirve para tener mas accesibles todos los pines del dispositivo, así como para alimentarlo de una forma mas cómoda.

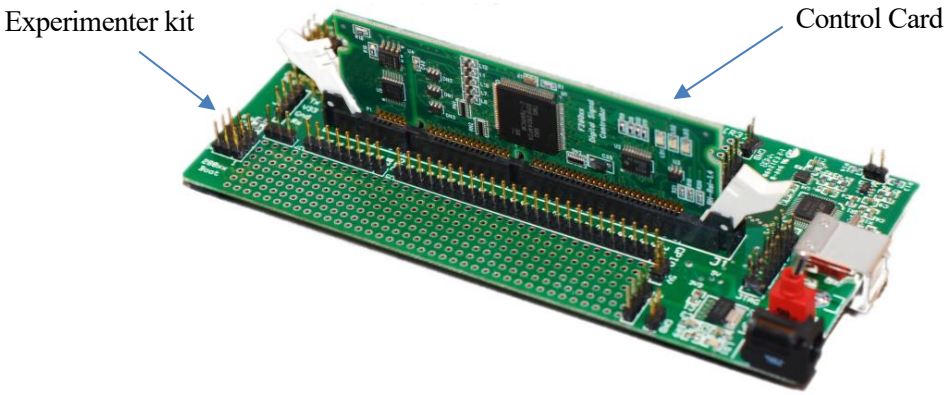


Figura 3-9. TMS320F28335 con Experimenter kit [14]

3.5 Cinta transportadora

Para la fabricación de la cinta transportadora se ha usado de referencia la siguiente guía encontrada en internet [18], mientras que el diseño se ha inspirado en la cinta transportadora de este proyecto [19]. Tras todo el proceso de fabricación, que se explicará con posterioridad, el resultado obtenido es el siguiente:

[14] Imagen tomada de https://www.ti.com/tool/TMDSDOCK2808. Fecha de consulta: Agosto de 2020

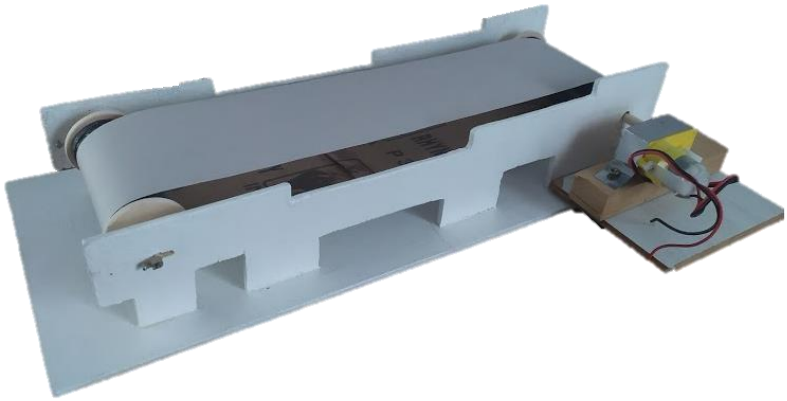


Figura 3-10. Cinta transportadora fabricada

Los pasos que se han seguido para la construcción de este elemento son:

1. **Fabricación del soporte**: tanto para la fabricación de las paredes como de la base, se ha usado madera, concretamente, contrachapado fino. Este material fue cortado con una segueta y lijado, según la forma que se observa, teniendo una longitud total de 350 mm. Antes del pegado, se realizaron dos taladros en cada pared para el soporte de los ejes. Es importante que estos agujeros estén alineados, pues si esto no ocurriera, los soportes podrían provocar mucha resistencia al giro de la cinta.

   Tras esto, se procedió al pegado de las paredes sobre la base a 110 mm de distancia una de la otra., con silicona termofusible. Para asegurar la perpendicularidad de las mismas, así como para conferir a la estructura una mayor resistencia, se añadieron ridigidizadores a lo largo de todas las paredes, unidos del mismo modo que los elementos anteriores. Una vez todo estuvo pegado, se pintó de blanco.

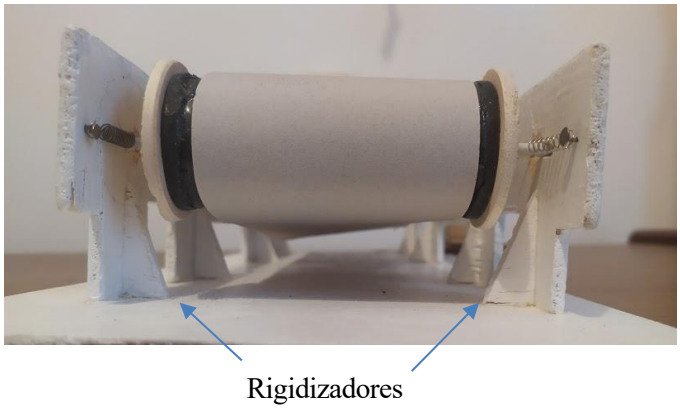


Figura 3-11. Detalle interior cinta donde se observan los rigidizadores

2. **Fabricación de rodillos:** Los rodillos se consiguieron cortando dos tramos de 75 mm de largo de un tubo de PVC de 30 mm de diámetro. Posteriormente se pintaron con pintura negra y, mientras esta se secaba, se fabricaron cuatro circunferencias de contrachapado, de 40 mm de diámetro cada una. En el centro de estas circunferencias se realizaron unos taladros con el mismo diámetro que los realizados con anterioridad.

   Finalmente, de nuevo con silicona termofusible, se unen las circunferencias a cada extremo de los tubos, teniendo especial cuidado en dejar los agujeros en el centro y alineados. El hecho de que las circunferencias tengan mayor diámetro que los tubos, permite que la madera sobresalga unos milímetros. Este pequeño saliente será muy importante, pues servirá de límite para la cinta, evitando que esta se salga de los rodillos.

3. **Inclusión de los ejes:** Diversas posibilidades fueron probadas para hacer de ejes. La que finalmente mejor funcionó fue la de tomar una varilla de madera para la implementación del eje del motor, y un trozo de percha cortado para el eje del otro rodillo.

   En el eje que no posee motor, se colocaron unas pequeñas abrazaderas en cada extremo, de forma que no permitieran que se saliera de su posición. En el eje del motor, sin embargo, no haría falta poner ningún elemento de este tipo, pues este va fuertemente unido al motor, que a su vez habría de fijarse luego a la estructura. Para la unión del eje de la cinta con el eje del motor, se utilizó un tubo de goma, dentro del que se introdujeron ambos ejes, quedando firmemente fijados por el apriete del mismo.

4. **Fijado del motor a la estructura:** al igual que en el paso anterior, múltiples fueron las opciones probadas hasta dar con una que funcionara correctamente. De esta forma, se intentó fijar primero el motor a la pared lateral, por medio de un taco de madera con la medida apropiada que se pegara a ambas partes. Sin embargo, al poner en funcionamiento el motor, este taco se despegaba.

   Finalmente, se optó por fijar el motor a la base. Esto se consiguió uniendo a la misma varias piezas de madera mediante tornillos, que dejaban al motor a la altura apropiada para conectarse al eje. Para la unión del motor con estos trozos de madera, se utilizó una pequeña tira de chapa, que se adaptó a la forma del motor, y que contaba con unas pestañas por las que se introdujeron tornillos que llegan hasta la madera.

5. **Fabricación de la cinta:** La cinta que permite el movimiento de los objetos está fabricada usando un trozo de tela fina. Así, se cortó con unas tijeras una tira de este material de 65 mm de ancho. La longitud de esta tira se fue ajustando de forma que quedara tensa entre los rodillos, y cuando se consiguió la adecuada, se pegó con pegamento de secado rápido.

6. P**rueba y corrección de errores:** Tras la realización de diversas pruebas de movimiento, se comprobó que el tensado de la cinta no era el suficiente como para no doblarse cuando se colocaban objetos sobre ella. Sin embargo, si se tensaba más, los ejes sufrían demasiado en el giro, provocando que el movimiento fuera muy irregular.

   La solución adoptada a esta situación fue alargar los taladros del eje que no tenía motor, de forma que se creó un pequeño carril que permitía el movimiento de este eje de forma paralela a la base. Para conseguir entonces el tensado de la cinta se añadieron unos pequeños muelles, fijados a las paredes mediante tornillos, y que tiran del eje en sentido contrario a donde se encuentra el motor. Gracias a este método, se consiguió un tensado óptimo de la cinta, funcionando el dispositivo de forma satisfactoria.

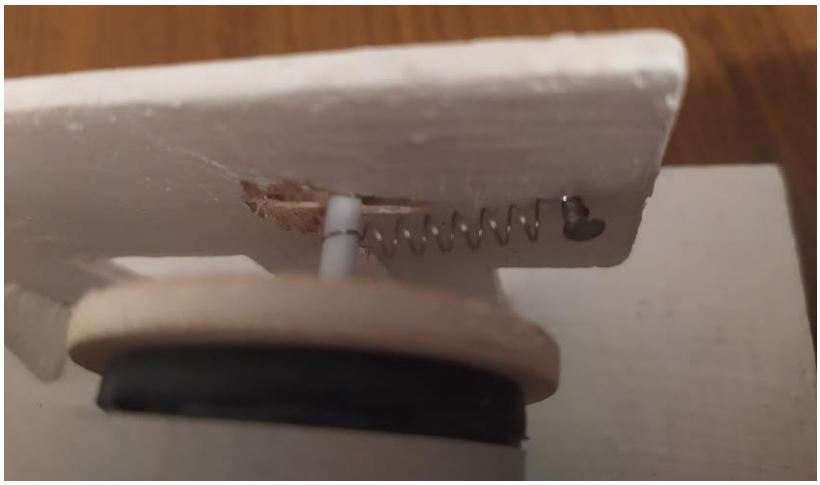


Figura 3-12. Detalle del tensor de la cinta

Existe dos elementos auxiliares imprescindibles para el funcionamiento de la cinta de los que aún no se ha comentado nada, como són el motor y el controlador de este. En los siguientes subapartados se hablará sobre ellos, comentando los modelos elegidos, asi como sus características.

3.5.1 Motor de corriente continua

Un motor de corriente continua no es otra cosa que un dispositivo que transforma una corriente eléctrica de tipo continuo en el movimiento rotatorio de un eje. Existen una amplia gama de motores, siendo la característica fundamental de estos la tensión a la que son alimentados, pues esta es la propiedad que indica la cantidad de potencia que son capaces de manejar. Para nuestra aplicación buscaremos un motor de dimensiones reducidas, de forma que pueda acoplarse fácilmente a la cinta, y que no necesite demasiada tensión de alimentación, ya que no necesitamos disponer de mucha potencia para mover la cinta.

Haciendo una búsqueda de dispositivos que cumplan estos parámeteos en el mercado, encontramos que hay diversos tipos, incluso hay algunos que traen una reductora incorporada entre el motor y el eje de salida. Este tipo de motores son muy beneficiosos para nuestros intereses, pues conseguiremos con ellos un motor donde el par es más importante que la velocidad. El modelo finalmente elegido será el motor con reductora DFRobot de 6V [20].

Figura 3-13. Motor de DC DFRobot

Podemos destacar las siguientes características de este motor:

- Rango de funcionamiento: de 3 V a 7.5 V
- Voltaje nominal: 6V.
- Velocidad de giro (sin carga): 180 rpm.
- Corriente de bloqueo: 2.8 A.
- Diámetro del eje: 5.5 mm.
- Máxima salida de torque: 0.8 kgf * cm.
- Relación de engranajes: 1:120.
- Dimensiones: 45.1 x 22.7 x 48 mm.
- Peso neto: 45g.
- Temperatura de funcionamiento: -10º a 60º.

Todas estas características hacen que se tarte de un dispositivo muy apropiado para nuestro trabajo. Sin embargo, el funcionamiento del motor alimentado a su tensión nominal sigue siendo demasiado rápido para lo que se pretende, por lo que necesitaremos incorporar un método, tanto para el control de velocidad, como para poder ponerlo en marcha y paralo a nuestro antojo. Ahí es donde entra en juego el controlador de motor del que hablaremos en el siguiente punto.

3.5.2 Controlador del motor

Antes de hablar sobre el controlador del motor, merece la pena comentar otras opciones que fueron estudiadas para llevar a cabo esta tarea. De esta forma, el mismo problema podría haberse resuelto con la integración conjunta de un relé y un potenciómetro. Un relé no es más que un interruptor de activación eléctrica, por tanto, este elemento podría haberse usado para controlar el encendido y apagado del motor. Por su parte un potenciómetro es un dispositivo basado en una resistencia variable, y que sirve para regular la tensión manualmente. Así, a través de la regulación de la tensión que le llega al motor, podría controlarse la velocidad de giro del mismo.

Finalmente, se decidió la implementación en el proyecto de un controlador de motor, pues es un dispositivo que aúna la funcionalidad de ambos, pero en un solo aparato. Además, posee otras características como el ajuste de velocidad por software, y no manuelmente como el potenciómetro, y la posibilidad de cambiar el sentido de giro de la cinta, todo esto a un precio bastante reducido. El modelo elegido es el Pololu DRV8835 Dual Motor Driver Kit [21].

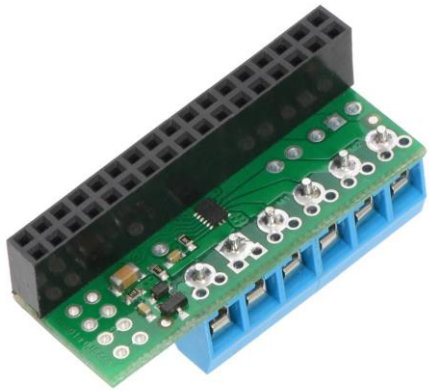


Figura 3-14. Controlador de motor el Pololu DRV8835 Dual Motor Driver Kit

Según la información que el fabricante nos proporciona del producto [22], con él podemos controlar simultáneamente 2 motores CC o un motor bipolar paso a paso. Puede ser alimentado a 3.3 V o a 5 V, mientras que su tensión de salida hacia los motores puede varias de 1.5 V hasta 11 V. Cuenta además con protecciones frente a sobrecorriente y sobretensión.

Posee 2 modos de funcionamiento, el modo PHASE / ENABLE (predeterminado: un pin para determinar la dirección y un PWM para ajustar la velocidad) o IN / IN (salidas en su mayoría entradas espejo). El que nos interesará es el primero, pues es el que nos permitirá controlar el motor de una forma muy sencilla. Para explicar como controlar un motor según este modo, veamos un esquema de los pines del dispositivo:

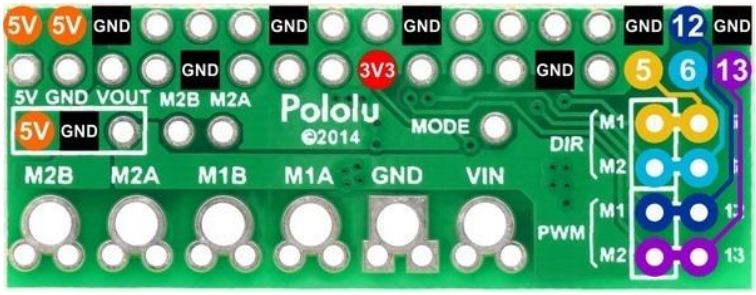


Figura 3-15. Esquema de pines del Polulo DRV8835 Dual Motor Driver Kit

Así, lo que deberemos hacer es alimentar correctamente la placa, (a 5 V o 3.3 V), conectar las bornas del motor a M1A y M1B (o M2A y M2B, en caso del motor 2), y enchufarle la fuente de alimentación de los motores, en los conectores grandes que aparecen en la parte inferior, VIN y GND. Tras esto, deberemos conectar una señal que se ponga a nivel lógico alto o bajo (indica un sentido u otro) en el pin 5, que aparece de amarillo, para controlar la dirección del motor 1 (o en el pin 6, para controlar la dirección del 2), y otra señal PWM en el pin 12 (o en el pin 13, en el caso del control de velocidad del motor 2). A partir de aquí, dependiendo del ciclo de trabajo del PWM que le estamos introduciendo y el valor del pin de dirección, nuestro motor girará más rápido o más lento, y en un sentido u otro.

3.6 PCB

El objetivo de diseñar un PCB es el de poder reducir la cantidad de cableado del dispositivo, lo que hace que las interferencias entre los mismos sean menores, así como poder trabajar de una forma más cómoda y clara. El PCB se ha diseñado, por tanto, para colocarse sobre el Experimenter kit, accediendo a todos los pines de este, y llevando las conexiones de los que sean de utilidad cerca del lugar donde deben ser usados.

Para el diseño del PCB se ha usado el software Altium Designer, que es un programa especialmente diseñado para este fin. El diseño de placas de circuito impreso en este programa se basa en 3 pasos, que son los que se han seguido:

1. **Creación de componentes:** lo primero que deberemos hacer para diseñar un PCB es crear una librería con los componentes que se van a incluir en la placa. Por defecto, Altium trae librerías con multitud decomponentes electrónicos, sin embargo, en nuestro PCB solo tendremos conectores de pines macho o hembra, sobre los que se pincharan los elementos del sistema, por tanto, deberemos crear nuestros propios componentes.

   Un componente en Altium esta formado por 2 elementos, su representación esquemática y su representación real. Crear la representación esquemática es bastante sencillo, pues solo deberemos realizar un dibujo simple del componente donde indicar los pines que posee y sus nombres. Por su parte, en la representación real, que en el programa se denomina “footprint” (huella), incluiremos la disposición real de los pines. Los “footprint” serán lo que luego se coloque sobre el diseño del PCB físico.

   Será muy importante, por tanto, respetar todas las distancias reales del dispositivo. La distancia entre pines consecutivos esta estandarizada, y es 2.45 mm. De esta forma, todas las distancias entre pines suelen ser un multiplo de este número, aunque se encuentren separados en distintos grupos. Se deberán indicar en este parte, de nuevo, los nombres de los pines, de forma que queden relacionados con el esquemático antes realizado. Si, a parte de los pines, sobre nuestro PCB queremos tener alguna delimitación del componente o algun dibujo para identificar los pines, deberemos colocarlo en su huella, de forma que aparezca luego al crear el componente.

De esta forma, para el desarrollo de nuestro PCB se crearon hasta 5 componentes, uno para del DSP, otro para el controlador del motor, otro para los sensores de presencia, otro para el sensor de color, y uno más para para las conexiones de los servomotores del brazo robótico.

2. **Desarollo del esquemático:** Una vez desarrolladas todas las librerías, el siguiente paso será crear un esquemático del circuito completo. En este esquema incluiremos los esquemáticos de todos los componentes que integren nuestro dispositivo, y, mediante pistas, uniremos aquellos pines que luego deberán estar físicamente conectados. Tras realizar todo el esquemático de nuestro dispositivo, el resultado obtenido es el siguiente:

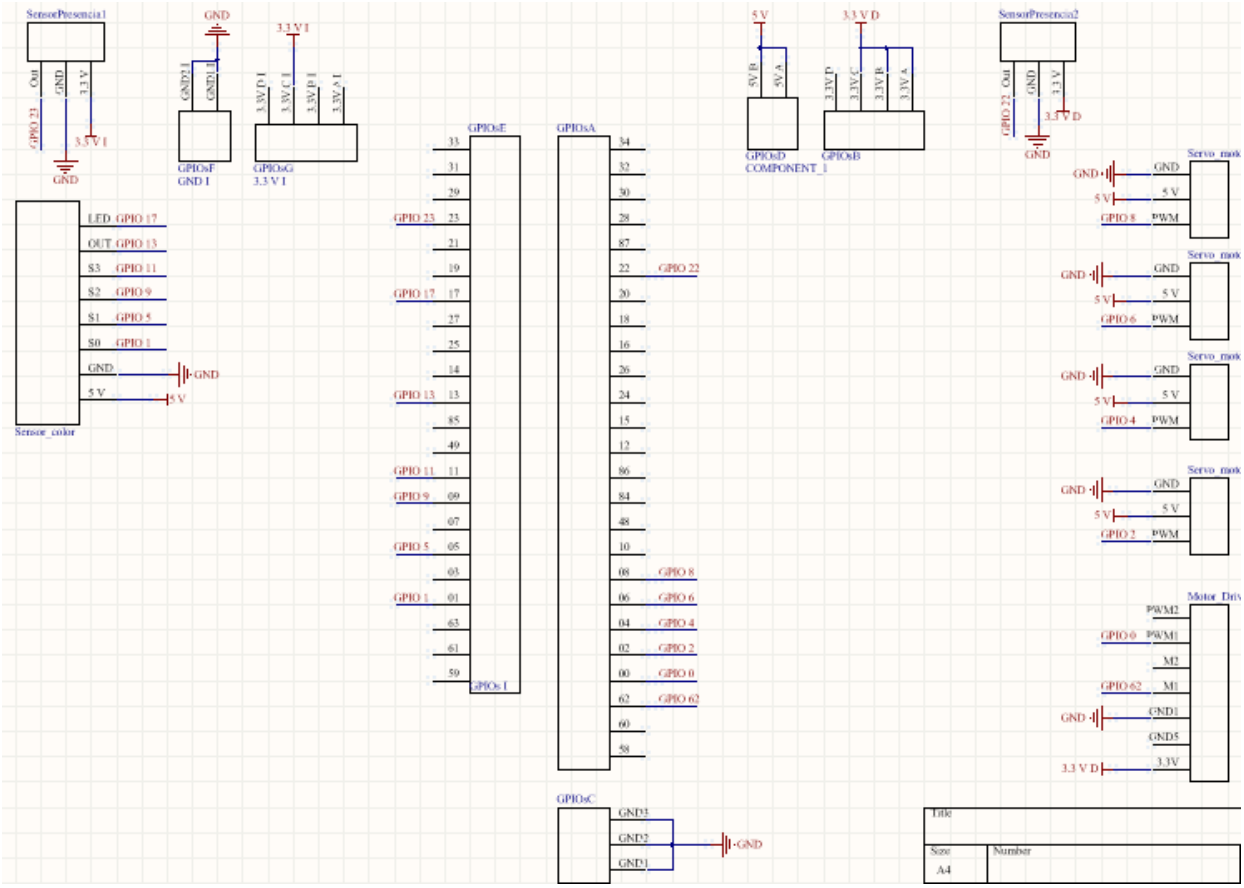


Figura 3-16. Esquemático del PCB desarrollado

Como podemos ver, en la parte central se encuentra el esquemático del DSP, en las esquinas, tenemos los dos sensores de presencia, en la parte izquierda el sensor de color, y en la parte derecha las conexiones de los motores del brazo robótico y del controlador del motor de la cinta. Para las conexiones, en lugar de pistas, se han usado etiquetas. De esta forma, aquellos pines marcados con la misma etiqueta serán los que estén conectados. Esto se ha hecho para intentar mantener la limpieza dentro del esquemático.

Será importante, una vez llegado a este paso, tener claro donde irá conectado cada pin de cada dispositivo. Por ejemplo, si el pin de un determinado elemento debe estar conectado a una señal PWM, habrá que unirlo con un pin del DSP que pueda configurarse como PWM, o si un elemento requiere alimentación a 5V, unirlo a una alimentación de 5V y no de 3.3 V.

3. **Diseño del PCB:** Una vez todas las conexiones están listas en el esquemático, importaremos este diseño al editor de PCB de Altium. En esta pantalla, nos aparecerán las huellas de todos los componentes que se encuentren en el esquemático. También serán visibles unas pequeñas líneas amarillas que unen

los pines que se hayan conectado entre sí en dicho esquemático. Gracias a estas, sabremos cuales de ellos deberemos unir.

Tras colocar las huellas de todos los componentes en el lugar donde posteriormente queremos que estén situados en el PCB, procederemos a unir mediante pistas los pines indicados. Existe en Altium la posibilidad de colocar pistas en varios niveles, de forma que estas puedan cruzarse sin tocarse. Para nuestro diseño, distribuiremos las pistas entre la capa superior del PCB (top layer) y la inferior (Bottom layer).

Tanto a la hora de colocar las pistas, como para introducir taladros u otro tipo de elementos en el PCB deberemos tener muy presentes las reglas de diseño. Estas son un conjunto de restricciones que no debemos violar para poder asegurar que nuestro PCB posteriormente sea fabricable, tales como distancia mínima entre un pin y una pista, o entre dos agujeros para pines. Estas reglas son modificables desde los menús de Altium, de manera que es posible adaptarlas al proceso de fabricación que luego vayamos a seguir. También existe la posibilidad de importar reglas desde un archivo, que es lo que realizamos nosotros, ya que la empresa donde posteriormente se mandará a fabricar el PCB oferce sus propias reglas de fabricación a disposición de todos sus clientes en internet.

Para delimitar las medidas y la forma de la placa, se utiliza la capa de diseño “Keep-out layer”. Así, dibujando un polígono cerrado mediante líneas que incluya todos los componentes de nuestro diseño dentro, e indicándole que este pertenece a la capa anteriormente mencionada, el programa entiende que con esto estamos marcando la forma de nuestro PCB. Será importante en este paso también prestar atención a las distancias entre elementos, pues no queremos que, al colocar el PCB sobre nuestro DSP, este choque con algún elemento y no se pueda realizar la conexión correctamente. Debido a que necesitamos acceder a pines de ambos lados del experimenter kit, es necesario dejar un hueco en nuestra placa por el que pueda sobresalir la control card del dispositivo. Esta es la razón del hueco central del PCB que posteriormente veremos en su diseño.

Con el fin de conseguir un funcionamiento eléctrico más estable del PCB, también se incluyen en nuestra placa de circuito impreso los conocidos como planos de tierra. Mediante la inclusión de estos, estamos convirtiendo toda una capa en la masa del dispositivo, en lugar de tener conectado cada elemento a tierra mediante pistas. Es decir, los pines de tierra de todos los elementos estarán directamente unidos a una capa, formando un gran plano de tierra, de ahí su nombre. Elegiremos como plano de tierra, tanto las capas superior como inferior, de forma que durante todo el PCB habrá distribuidos unos taladros de conexión de ambas capas, que aseguran que se encuentren al mismo potencial.

Tras la realización de todos estos pasos, el diseño final del PCB es el siguiente:

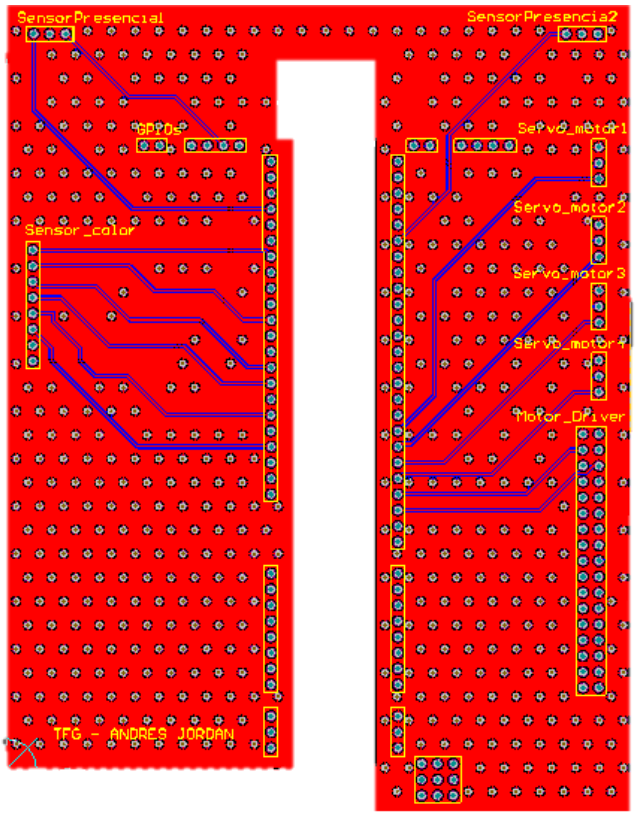


Figura 3-17. Top layer del PCB diseñado

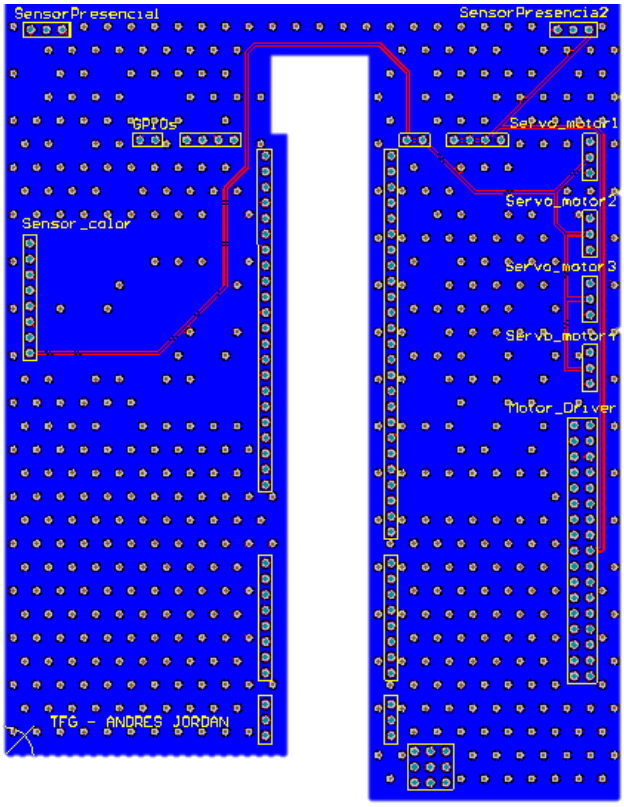


Figura 3-18. Bottom layer del PCB diseñado

Para poder encargar la fabricación de nuestro PCB, necesitamos una serie de ficheros conocidos como gerbers. En la página de la empresa donde lo vamos a encargar, JLCPCB [23] encontramos una sencilla guía donde

muestra los pasos a seguir para obtener todos estos ficheros desde un proyecto de Altium terminado. Una vez conseguidos, se lo mandamos a la compañía, a través de su página web, junto con una serie de especificaciones. Con esto, la empresa ya será capaz de abordar la fabricación del PCB. Tras finalizarla, la empresa se encargará de mandarlo a nuestro domicilio.

Una vez recibido, deberemos soldarle conexiones macho y hembra, según corresponda, en todos los huecos de los pines, para poder conectarle elementos. El resultado final, una vez montado sobre el DSP es el siguiente:

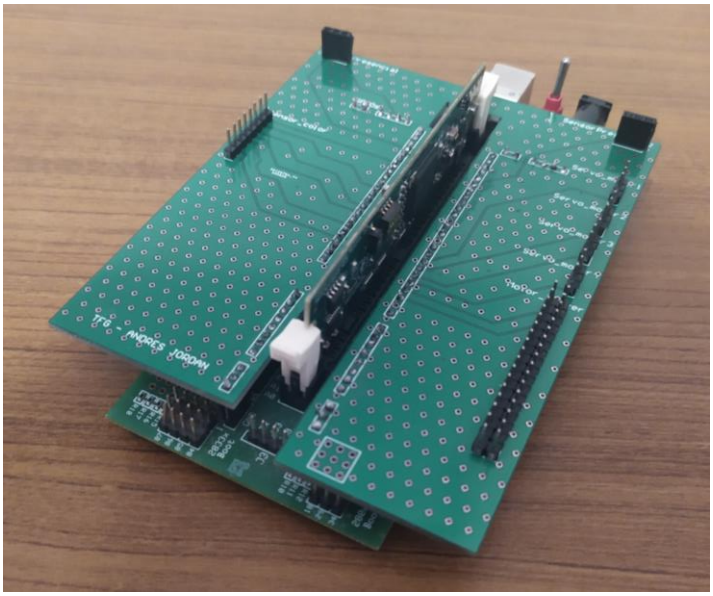


Figura 3-19. PCB montado sobre el DSP

4 SOFTWARE

El sotfware usado para el desarrollo del todo el poryecto es Code Composer Studio 9.2, que es el programa que proporciona el fabricante del DSP, Texas Instrument, para la programación de sus dispositivos. El manejo de este programa es bastante sencillo, basándose fundamentalmente en dos modos, el modo edición, en el que podemos modificar todos los archivos pertenecientes al proyecto en el que nos encontremos, y el modo ejecución, en el que carga el programa del proyecto que tengamos activo en el DSP y lo ejecuta.

El CCS permite además establecer breakpoints a lo largo del código, así como la posibilidad de ejecutar paso a paso los programas y monitorizar el valor de las variables en tiempo real. Todas estas opciones lo convierten en una herramienta muy cómoda para escribir, probar y depurar código.

Durante los siguientes subapartados, analizaremos las difrentes pruebas de funcionamiento y manejo de todos los elementos que componen nuestro proyecto, para acabar con una explicación sobre el código principal que hace que todo el sistema funcione.

4.1 Sensor de presencia

Conseguir la detección de objetos mediante el sensor de presencia es una tarea relativamente sencilla. Según lo explicado anteriormente, lo único que deberemos hacer es, una vez alimentado correctamente y conectado a tierra, configurar un pin como GPIO, es decir, como de propósido general, y posteriormente indicarle que es un pin de entrada. Este pin es el que va conectado a la salida OUT del sensor.

En el código de prueba de este dispositivo, por tanto, introduciremos las instrucciones necesarias para leer el valor de este pin, y si tenemos en él un nivel lógico bajo, estaremos detectando un objeto, y si en el pin encontramos un 1, no habrá objeto delante del sensor. Incluyendo la lectura del pin en un bucle infinito, de forma que estemos mirando constantementesu valor y, poniendo una variable a 1 si hay objeto, o a 0 si no lo hay, se obtienen los siguientes resultados:

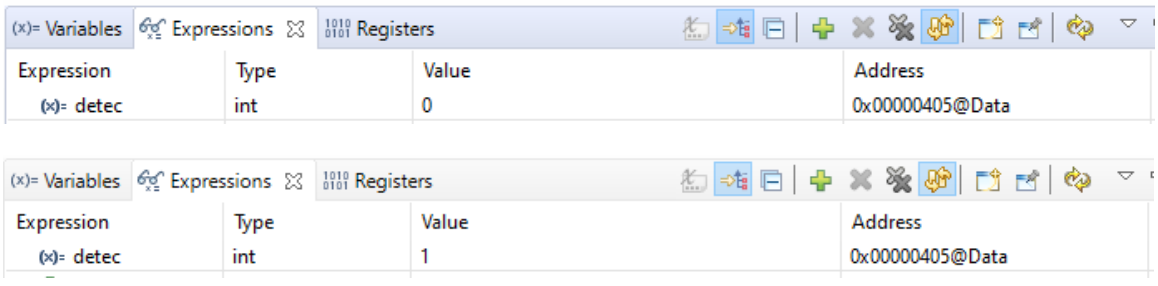


Figura 4-1. Prueba funcionamiento sensor presencia: (a) sin objeto delante (b) con objeto delante

Como vemos, conseguimos obtener un cambio en una variable según coloquemos objetos o no, por lo que somos capaces de manejar este sensor satisfactoriamente. Sin embargo, para nuesto dispositivo, necesitaremos un funcionamiento más rico de este dispositivo, en el sentido de que no queremos detectar que haya objeto o no, sino que para nuestro sistema necesiatamos saber el momento exacto en que un objeto se coloca en una posición, es decir, necesiatamos capturar el instante en que esta variable que indica la posición cambia de cero a 1. Esto es lo que se conoce como detección de flanco, en este caso, de flanco de subida.

La necesidad de esta detección de flanco radica en que la condición para realizar la media cuando un objeto se encuentre en posición de medir su color no puede ser que el sensor este detectando objeto. Si esto fuera así, una vez realizada la medida, el sensor sigue detectando objeto, luego al volver el bucle del sistema al punto donde se decide si se debe realizar medida, la respuesta sería sí, pese a que ya se ha realizado, y el sistema quedaría colgado en este punto, repitiendo la medida constantemente.

Para la implementación de la detección de flanco, se necesita una segunda variable interna, llamada detec_ant. En esta variable guardaremos el valor de la detección en el ciclo anterior, de forma que al llegar a la decisión de si realizar medida, la pregunta es si el sensor esta detectando objeto, y en el ciclo anterior no lo detectó. De esta forma, solo se realiza una medida en el instante preciso en que el objeto llegue a la posición, no llevandose a cabo más hasta que otro objeto entra a la posición de medida. Para guardar la medida del ciclo anterior, deberemos asignarle el valor leído del pin al final del ciclo, justo antes de iniciar uno nuevo. Implementando este código y realizando una prueba, se determina que funciona de manera satisfactoria.

4.2 Sensor de color

Según lo antes comentado de este sensor, necesitaremos hasta seis GPIO´s para manipularlo y conseguir una medida. Cinco de ellos se configurarán como salidas, los correspondientes a LED (habilitación de medida), S3, S2, S1 y S0, y, el restante, que es el que va conectado a OUT, deberá definirse como de entrada. Una vez hecho esto, colocaremos el pin conectado a S0 a nivel alto, y el pin conectado a S1 a nivel bajo. De esta forma, fijamos un escalado de frecuencias a la salida del 20% para siempre.

La idea de la medida del color es que deberemos crear una estructura *for* que se ejecute tres veces, cada una de ellas para la medida de una de las componentes en que se descompone el color, rojo, verde y azul. Así, cada vez que se entre en el bucle for, en función del valor de la variable de actualización del bucle, y mediante una estructura *case*, se selecciona el valor de los pines de salida conectados a S2 y S3 para realizar la medida de un parámetro concreto. Según lo programado, en la primera iteración se medirá la intensidad de rojo, después la de verde y por último la de azul.

Teniendo ya la estructura para cambiar el color a medir, necesitamos ahora ser capaces de calcular la frecuencia para cada uno de los tres casos. Para ello, se utilizará un *timer*, y el concepto anteriormente comentado de la detección de flancos. De esta forma, una vez elegido el color que toca medir en ese instante, se activa la medida poniendo a cero el pin correspondiente a LED. Después, se inicia un temporizador, previamente programado para saltar a los 300 ms. Tras poner en marcha el temporizador, se entra en un bucle *while,* cuya condición de salida se activa precisamente al saltar el temporizador, o lo que es lo mismo, un bucle *while* que se ejecutará durante 300 ms. En el interior de este bucle utilizamos el concepto anteriormente explicado de detección de flancos de subida, y cada vez que se detecte uno en el pin conectado a OUT, se incrementa una variable llamada *cont*.

Al salir del bucle, por tanto, tendremos almacenado en esta variable *cont* el número de flancos que se han detectado en 300 ms, y tras realizar una pequeña cuenta, podremos obtener el número de flancos en un segundo, o lo que es lo mismo, la frecuencia de la señal, que era lo que buscábamos. Antes de repetir el ciclo de medida para otro color, se almacena cada valor de frecuencia obtenido en la posición apropiada de un vector de longitud tres, llamado RGB. Al finalizar el ciclo de medida de los tres colores, en dicho vector estará almacenada la frecuencia de cada una de las componentes del color, con lo que ya podremos trabajar con él para discernir de que color se trata.

Tras implementar este código, y realizar pruebas midiendo objetos rojos, verdes y azules, los resultados obtenidos son:

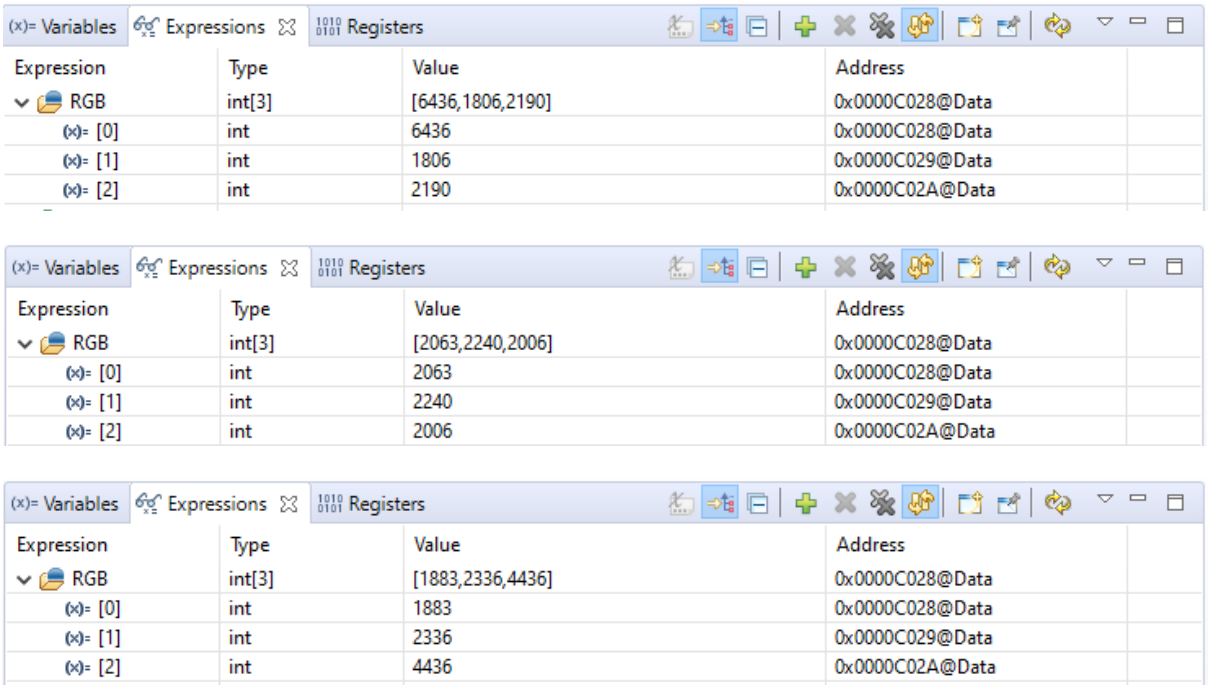


Figura 4-2. Pruebas sensor de color con objeto: (a) rojo, (b) verde, (c) azul.

En estos resultados podemos observar que conseguimos medir el color, pues siempre se obtiene la frecuencia mayor para el color del objeto que se mida en ese momento. La medida del verde no la realiza con tanta precisión como las otras dos, pues, como vemos, los valores obtenidos tanto para rojo como para azul son ampliamente superiores en sus respectivas medidas, no sucediendo así en el segundo caso.

Una posible explicación para esto es que el verde de los objetos que se están midiendo no se corresponda demasiado con lo que el sensor entiende como verde, siendo demasiado claro el que se mide. Aún así, daremos por buenas estas pruebas de funcionamiento, debido a que tanto las medidas de rojo como las de azul las realiza de forma muy satisfactoria. En el apartado dedicado al código principal se explicará como se distinguirán los colores según estas medidas, y que hacer para solucionar los posibles pequeños errores en la medida del verde.

Cabe mencionar también que estas pruebas se han hecho en condiciones de luz baja, que es la apropiada para la medida del sensor. En caso de realizarse con más luz, las medidas en general serán mas parecidas unas a otras, aunque se seguirá notando la diferencia. Es por ello que la medida de este sensor, y el reconocimiento de color posterior requerirán de una calibración, según las condiciones de luz en que vaya a operar.

4.3 Control del motor de la cinta

Para el control del motor de la cinta, necesitamos configurar 2 pines, uno como GPIO de salida y otro como PWM. En el puente H, estos pines GPIO y PWM se conectan respectivamente a la dirección y al PWM del motor uno. También necesitaremos conectar a las pestañas azules del Pololu el portapilas de 4 pilas, cuyos terminales se conectan en VIN y GND, y las bornas del motor, que se conectan en las pestañas correspondientes al motor uno. Tras realizar estas conexiones, estamos listos para desarrollar el código y probar el funcionamiento de la cinta.

Debido a que en nuestra operación no desearemos cambiar la dirección de giro de la cinta, el GPIO de salida deberá colocarse a uno o a cero, dependiendo de cual sea el correcto para nosotros según en que orden hayamos colocado las bornas del motor, y mantenerse en este valor durante toda la ejecución.

Por su parte, el PWM requerirá ser configurado para su uso. Para ello usaremos el esqueleto de la función Setup_pwm, que es una función que se encuentra en los ejemplos de desarrollo de código que nos proporciona Texas Instrument. Dentro de esta función, deberemos ir dándole valor a los difrentes registros de configuración del PWM, para obtener el comportamiento deseado.

Así, colocando un 2 en el registro CTRMODE, conseguimos un PWM en modo *up-down*, es decir, que realiza la cuenta de su valor interno de forma ascendente y luego descendente. Esta cuenta ascendente y descendente deberá realizar commutaciones en la salida cuando su valor sea igual al registro CMPA. Las acciones a realizar se modifican mediante la configuración del módulo *action qualifier*, donde debemos seleccionar que cuando la cuenta interna se encuentre con el valor de CMPA en la subida, debe ponerse a 1, y cuando se lo encuentre en la bajada debe ponerse a 0. El resto de registros los colocamos a 0, con lo que dejamos desactivadas una serie de funcionalidades del PWM que no necesitamos.

Para la obtención de la frecuencia de operación deseada, se debe configurar el módulo *time base.* De esta forma, si colocamos el registro CLKDIV a 0 y HSPCLKDIV a 1, conseguimos un escalado de la mitad de la frecuencia de reloj del sistema. La frecuencia deseada para la salida del PWM es de 1 kHz, por lo que necesitaremos calcular, según este preescalado de reloj y esta frecuencia, la cuenta máxima del valor interno del PWM, que es TBPRD. Esto se hace con la siguiente operación:

$$TBPRD = \frac{T_{PWM} * CLK_{sistema}}{2 * HSPCLKDIV * CLKDIV} = \frac{0.001 * 150e^6}{2 * 2 * 1} = 37500$$

Con toda esta configuración, obtenemos un PWM de 1 kHz de frecuencia y cuya salida tiene esta forma:

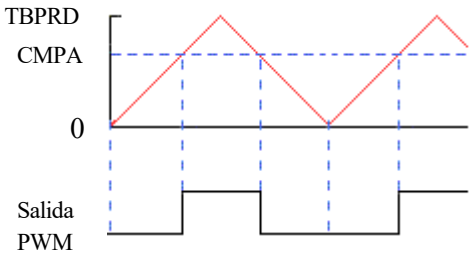


Figura 4-3. Esquema del PWM del motor de la cinta

Modificando el valor de CMPA podemos conseguir un ciclo de trabajo (tiempo que se encuentra a uno la salida) mayor o menor, con lo que obtendremos un funcionamiento más rápido o lento de la cinta. Uno de nuestros objetivos es poder parar la cinta, luego si colocamos CMPA = TBPRD el ciclo de trabajo será 0 (nunca se pone a 1 la salida), luego la cinta estará parada. Esta será, por tanto, la sentencia de parado de cinta. Por su parte, cuando este en marcha queremos que realice un giro lento, por lo que, realizando difrentes pruebas, llegamos a la conclusión que colocando CMPA= TBPRD*0.9 (es decir, programando un ciclo de trabajo del 10%), obtenemos una velocidad de cinta idónea.

El código para realizar estas pruebas no es más que la configuración antes explicada de GPIO ´s y PWM, y una serie de sentencias modificando el valor de CMPA, y viendo como esto afecta a la velocidad de la cinta y a su funcionamiento o parada. Al realizarse y comprobarse que todo funciona, estamos en condiciones de incluir la cinta en el sistema completo.

4.4 Control del brazo robótico

Para el control del brazo robótico, necesitaremos de nuevo configurar señales PWM, en este caso, cuatro, una por cada servomotor que debemos manejar. Contaremos con la ventaja de que todos ellos tienen las mismas

características, por lo que su configuración será la misma, y solo deberemos copiarla de unas a otros. Según la información disponible de estos servos, para establecer un ángulo de giro, los PWM deben tener la siguiente forma:

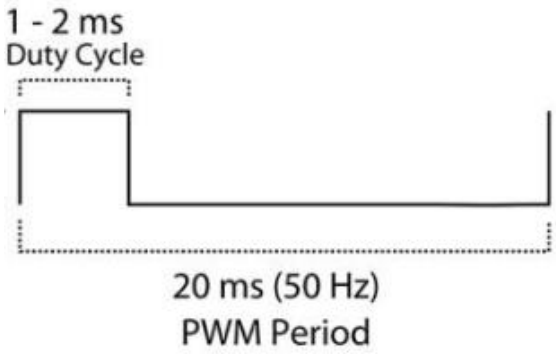


Figura 4-4. PWM necesario para control de los servomotres.

En el caso anterior usamos el modo *up-down* para generar las señales, sin embargo, como vemos, ahora necesitamos que el pulso se encuentre en la parte inicial del ciclo, y no en la parte central de este, como ocurria anteriormente. Es por esto que los PWM se configurarán en modo *up,* poniendo el registro CTRMODE a 0, y en el módulo *action qualifier* seleccionaremos que la salida se ponga en nivel lógico alto cuando la cuenta interna sea cero, y que pase al nivel lógico bajo al llegar al valor de CMPA.

Para conseguir la frecuencia indicada, colocamos los registros CLKDIV a 5 (escalado de 32) y HSPCLKDIV a 1 (escalado de 2), con lo que obtenemos un preescalado total del reloj del sistema de 64. En este caso, como la cuenta está modo *up,* la operación que deberemos realizar para calcular el valor de TBPRD es la siguiente:

$$TBPRD = \frac{T_{PWM} * SYSCLKOUT}{HSPCLKDIV * CLKDIV} - 1 = \frac{0.02 * 150e^{6}}{32 * 2} - 1 = 46874$$

Con esto, hemos configurado un PWM de de 50 Hz con la siguiente forma de salida:

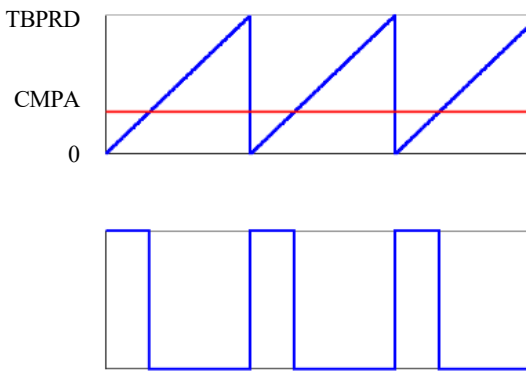


Figura 4-5. Esquema del PWM de los servomotores

De nuevo, tocando el valor del registro de CMPA conseguimos que el tiempo de activación sea uno u otro. Para mover el servo, necesitamos unos tiempos de activación precisos, que provoquen un valor alto de entre 1 ms, lo que el servo interpretará como que debe colocarse en un extremo, y 2 ms, que significará que el servo se encuentra en el otro.

De esta forma, según la configuración descrita, existe un rango de valores de CMPA que llevan el motor desde una de sus posiciones extremo hasta la otra. Haciendo unas sencillas operaciones, determinamos que este rango es [2343.7, 4687.4]. Luego, el código de prueba del brazo robótico, tras realizar las configuraciones, asigna a los difrentes PWM´s valores en este rango de forma secuencial, consiguiendo así el movimiento de los motores a

posiciones concretas. Programando los valores concretos y en el orden correcto, se consigue que el robot realice una secuencia de movimiento determinada.

Cabe mencionar que este rango de operación no será completo, ya que por la propia estructura del robot, habrá servomotores que no puedan alcanzar ciertas posiciones, lo que obliga a tener mucho cuidado a la hora de asignar los valores de CMPA a cada PWM.

4.4.1 Control de la velocidad de movimiento del brazo robótico

Tras la realización de diversas pruebas, se concluye que el movimiento del robot es demasiado brusco, pues que los servomotores se mueven muy rápido al recibir una referencia, tal y como se vaticinó al analizar las características de estos. Es por esto que se ha ideado una funcion que consigue un movimiento más suave del robot, y es la siguiente:

```
void movimientox(int pos, int des){

    int i;
    if(pos>des){
        for(i=pos;i>des;i--){
        EPwmxRegs.CMPA.half.CMPA = i ;
        delay_loop(3000);
        }
    }
    else{
        for(i=pos;i<des;i++){
        EPwmxRegs.CMPA.half.CMPA = i ;
        delay_loop(3000);
        }
    }
}
```

Figura 4-6. Función de control de velocidad del brazo robótico

Para el uso de esta función, necesitamos conocer el valor de CMPA en el que se encuentra al inicio (pos), y al que se desea mover (des). Dentro de la función, si *pos* es mayor que *des,* el valor de CMPA se disminuyendo de uno en uno hasta alcanzar *des,* estableciendo un pequeño retraso cada vez que se disminuye. Este retraso se consigue mediante la función *delay_loop*. Por el contrario, si *pos* es menor que *des,* se realiza la misma opración, pero esta vez aumentando el valor hasta llegar al deseado. Será necesario construir una de estas funciones por cada PWM, de manera que cada una actue sobre el registro CMPA de cada señal.

Tras realizar diversas pruebas con estas estructuras, se comprueba que funcionan de forma satisfactoria. La velocidad de movimiento del robot puede ajustarse mediante el valor de entrada de la función *delay_loop,* pues este indica el retraso que se establece entre una referencia y la siguiente. De esta forma, colocando una secuencia de estas funciones, prestando mucha atención a sus posiciones iniciales y de destino para no cometer errores, se consigue un movimiento bastante satisfactorio del brazo.

4.5 Funcionamiento de todo el sistema en conjunto

Una vez se han comentado todas las pruebas necesarias para el manejo de cada elemento del proyecto, en este subapartado comentaremos el código que hace que todo funcione conjuntamente, consiguiendo la clasificación por colores de los objetos que pasen por la cinta. El pseudocódigo del programa desarrollado es el siguiente:

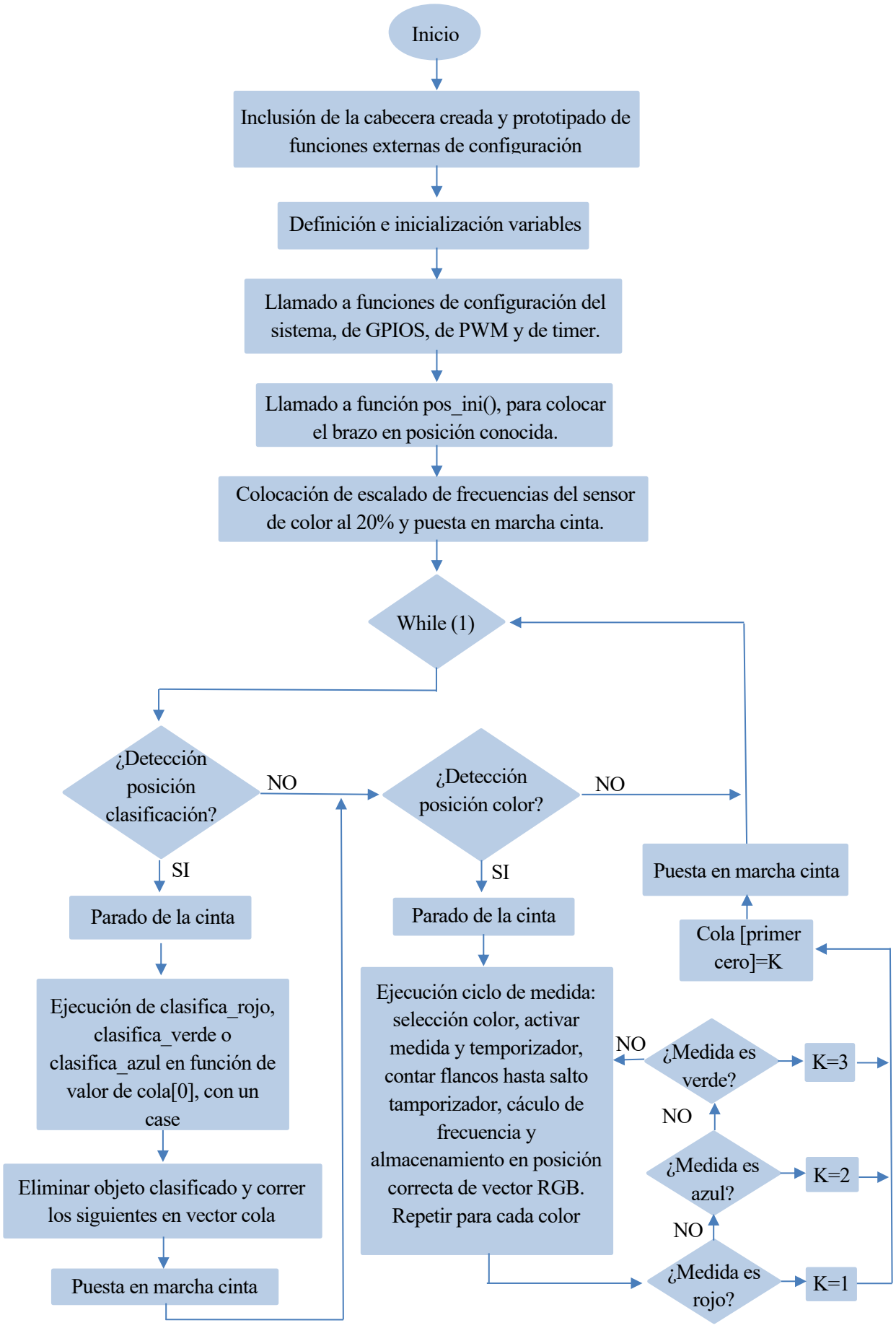


Figura 4-7. Diagrama de flujo del algoritmo de control

Realmente, no se ha sido totalmente fiel en el diagrama al código implementado, ya que se han resumido ciertas partes y otras se han modificado levemente, con el fin de hacer el esquema más compacto y que no se extendiera en exceso. Sin embargo, esto no será ningún impedimento para que nos sirva de referencia a la hora de hacer comentarios del programa desarrollado.

La creación de la cabecera tiene por objetivo que no sea necesario escribir toda la inclusión de librerías y declaración de funciones creadas al principio de cada archivo del proyecto, sino que, con la inclusión de la cabecera, todo esto se encuentra ya implicito. Es una cuestión básicamente de claridad, ya que nos ahorrará algunas líneas de código al principio de cada archivo de programación. El proyecto esta formado por tres archivos: uno donde se desarrollarán las funciones de movimiento del brazo robótico, otro donde se encontrarán las de configuración y, por último, el *main,* que es donde se encuentra el código principal que llama al resto de funciones.

La llamada a función de configuración de los GPIOS, configura cada GPIO que se usará según su función concreta. En la siguiente tabla se observa los pines usados, donde se conectan y su función:

Tabla 4-1. Conexión de pines del sistema

PIN	Función	Conectado a
00	PWM1A	Controlador motor cinta, PWM1
01	GPIO, salida	Sensor color, S0
02	PWM2A	PWM servomotor 1
04	PWM3A	PWM servomotor 2
05	GPIO,salida	Sensor color, S1
06	PWM4A	PWM servomotor 3
08	PWM5A	PWM servomotor 4
09	GPIO,salida	Sensor color, S2
11	GPIO,salida	Sensor color, S3
13	GPIO,entrada	Sensor color, OUT
17	GPIO,salida	Sensor color, LED
22	GPIO,entrada	Sensor presencia 1, OUT
23	GPIO,entrada	Sensor presencia 2, OUT
62	GPIO,salida	Controlador motor cinta, DIR1

La función de configuración de PWM ejecuta las instrucciones explicadas en los puntos anteriores de control de la cinta y de los servos para cada PWM, consiguiendo por tanto que estas señales tengan la forma requerida para cada caso. Por su parte, la configuración del *timer,* lo inicializa, activa su interrupción, y lo configura para 300 ms, dejandolo a la espera de ser activado.

Tras esto, y antes de entrar en el bucle infinito de control, será necesario ejecutar robot_ini(), función que hemos creado para colocar al robot en su posición inicial. Esto será necesario para trabajar con las funciones antes descritas de movimiento del robot controlando su velocidad, ya que un pequeño inconveniente de esta forma de

operar es que necesitaremos conocer en todo momento la posición inicial de un servomotor para poder moverlo a otra distinta. Durante el funcionamiento normal, esto no será un problema, puesto que el motor se encontrará en aquella posición que lo dejamos la última vez que lo movimos. Sin embargo, en el instante inicial los servos pueden están en cualquier estado no conocido, y esta es la razón por la que se debe ejecutar esta función.

Una vez en el bucle infinito, es interesante hacer notar que no se realizará ninguna otra acción mientras no se detecte un flanco de subida en las posiciones de clasificación o de medida de color. Una vez se produzca un flanco en el sensor de la posición del color, se realizará toda la secuencia de medida de color explicada en su correspondiente apartado.

Al finalizar, se entra en una estructura de *if´s,* que sirven para discernir el color. Las condiciones de estos *if´s* son las que hacen que el sistema entienda que el objeto se trata de un color u otro, según la medida obtenida. Para realizar la calibración del sensor según unas condiciones de trabajo, lo que se debe hacer es realizar diversas medidas de colores, y según los valores obtenidos, modificar estas condiciones, de forma que el sistema pueda relaccionar una determinada medida con un color.

Cuando se entra en uno de estos *if´s*, se guarda en el primer lugar del vector cola que sea cero, el valor correspondiente al color detectado (Rojo=1, azul=2, verde=3). Si, por el contrario, no se cumple ninguna condición, la medida se considerará indeterminada, y se procederá a realizar una nueva, activando la variable repite_medida. Solo en caso de que la medida sea concluyente, se puede salir de este bucle. Con esta operación se evita que comportamientos extraños por parte del sensor den lugar a clasificaciones erróneas, así como solucionar de alguna forma la dificultad del mismo a la hora de medir el verde.

Por su parte, si se detecta un flanco en la posición de clasificación, se mira el primer elemento del vector cola, y según el valor de este, se ejecuta la función de clasificación que corresponda. Estas funciones de clasificación están definidas en el archivo func_mov.c y están formadas simplemente por la secuencia de funciones de movimiento correctas para llevar el brazo a una posición u otra.

Al clasificarse un objeto, se borra de la cola, y se recorre el vector adelantando un lugar cada valor almacenado. Con esto se consigue que el primer lugar del vector sea el que indique cual es el color del siguiente objeto a clasificar, manteniendo en el resto de las posiciones las siguientes clasificaciones. Esta implementación sirve para que pueda haber más de un objeto a la vez en la cinta, y el sistema sea capaz de clasificarlos en su orden correcto.

5 MEJORAS Y FUTUROS TRABAJOS

TRas explicar todos los entresijos y procedimientos realizados para llevar a cabo el proyecto, merece la pena dedicar un capítulo a hablar de posibles aspectos a mejorar del mismo, así como sobre otros proyectos que pudieran estar basados en este trabajo.

- **Uso del módulo eCAP:** tras haber programado la medida del color de forma "manual", mediante GPIOS, se notó que el DSP trae un módulo integrado que puede usarse para la medida de periodos de eventos externos, el eCAP. Este podría haberse implementado para el cálculo de las frecuencias de los colores. Esta posibilidad fue ampliamente estudiada, sin embargo, ya estaba listo el PCB y no era posible configurar el pin al que se conectaba la salida del sensor como de este tipo, así que, como el otro método funcionaba, se decidió no modificarlo. El uso de este módulo posiblemente permita un cálculo más preciso de los colores.

- **Incorporación de botón de marcha-paro:** mediante la incorporación de un pulsador conectado a un GPIO se podría haber configurado la marcha-paro del sistema, haciendo que no siempre que esté el DSP encendido esté todo en funcionamiento.

- **Mejora del movimiento del brazo:** con la programación actual, el brazo robótico solo mueve un servomotor simultáneamente, limitando las trayectorias que el brazo puede seguir, además de tardar un tiempo considerable en realizar cada movimiento. Conseguir mover varios motores simultáneamente y de la forma que se desea, es algo bastante complejo. Tanto es así, que existe toda una rama en la robótica que se dedica a ello, como ya se ha comentado con anterioridad. Pese a su dificultad, si se implementara, el sistema ganaría mucho en eficacia y vistosidad.

- **Mejora diseño de la cinta:** pese a que se podría pensar que la fabricación de una cinta funcional es algo sencillo, tras la fabricación de esta se ha comprobado que no es así. Hay muchos aspectos que interfieren en su buen funcionamiento, y pulirlos todos es una tarea que requiere mucho tiempo. De la que aquí se ha fabricado, se podría mejorar la forma de los rodillos, la sujeción del motor, o la alineación de los ejes, todo ello para un funcionamiento más fluido y sin estridencias.

- **Funcionamiento en paralelo de la cinta y el brazo:** sería posible programar el sistema de forma que brazo y cinta funcionaran de forma "independiente". Esto consistiría en poder manejar la cinta de forma que, aunque el brazo estuviera realizando una operación de clasificado, ella estuviera funcionando, permitiendo que el resto de piezas pudieran avanzar en el proceso mientras esto fuera posible. Para conseguir esto, habría que programar la ejecución de multiprocesos, motivo por el que esta posibilidad quedó fuera totalmente del alcance del proyecto.

- **Incorporación de más colores:** la precisión del sensor de color es limitada, sin embargo, definiendo unas condiciones de trabajo concretas, el sensor siempre ofrece medidas parecidas. Es por esto que, mediante una secuencia de calibraciones, podrían llegar a distinguirse muchos más colores de los que en este trabajo se hacen. Incluso sin demasiados esfuerzos, detectar el negro (todas las medidas muy bajas) y el blanco (todas las medidas muy altas), es una tarea bastante sencilla y que ofrecería mayor variabilidad al proyecto.

- **Método de decisión distinto:** manteniemdo la actual estructura de trabajo, podría sustituirse el sensor de color, por ejemplo, por un lector de código de barras, y el brazo realizaría la clasificación según esta lectura. Esta podría ser una implementación más cercana a algo que podría verse en una industria real.

Otra posibilidad podría ser la inclusión de una cámara con algún tipo de software de procesado de imágenes, para, por ejemplo, realizar un control de calidad de los objetos que pasen por la cinta, y aquellos que no lo cumplan, serían retirados por el brazo robótico.

- M**étodo de clasificación distinto:** el brazo robótico podría ser reemplazado por un juego de pistones, o por una plataforma giratoria que se encontrara al final de la cinta, y que girara según el color del objeto a recibir. Serían proyectos con la misma funcionalidad, pero sin la complejidad que aporta el manejo del brazo.

6 CONCLUSIONES

La realización de este sistema embebido, mediante todas las operaciones aquí descritas, demuestra la gran complejidad que acarrea abordar proyectos de electrónica de estas características, donde se manejan un grupo de sensores y actuadores.

Cada elemento posee sus propias características y necesidades para un correcto funcionamiento, por lo que al integrar todos ellos en un mismo sistema, la cantidad de problemas a abordar crece exponencialmente. Sin embargo, como se justificó en el segundo capítulo, los sistemas embebidos están muy presentes en nuestra vida actual, por eso, es importante comprender y aprender a resolver las dificultades que entraña desarrollarlos, pese a que luego no tengan una aplicación directa real. De esta forma, los conceptos y aptitudes aprendidos durante el desarrollo del presente proyecto servirán de valiosa enseñanza a la hora de afrontar problemas de mayor envergadura en la vida real.

El DSP elegido, el TMS320F28335 de Texas Instrument, posee unas características más que suficientes para el control de todo el proceso. El uso de este dispositivo aporta también esta complejidad de la que estamos hablando, pues se ha elegido por delante de otras opciones mucho más fáciles de manejar, pero que no están orientadas a la fabricación de sistemas complejos.

En conclusión, se ha conseguido el objetivo del trabajo, desarrollando el sistema que pretendiamos con éxito, y adquiriendo durante el proceso un gran número de nuevas y valiosas enseñanzas.

REFERENCIAS

[1] Caro Márquez, E., & Rodríguez Gutiérrez, M. (2017). *La cuarta revolución industrial*. El autor.

[2] https://es.wikipedia.org/wiki/Sistema_embebido

[3] https://dle.rae.es/transductor

[4] https://www.exabyteinformatica.com/uoc/Informatica/Sistemas_empotrados/Sistemas_empotrados_(Modulo_1).pdf

[5] http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/industrial.htm

[6] https://www.amazon.es/MBEN-mec%C3%A1nica-Bricolaje-mand%C3%ADbula-Educativo/dp/B082V9P7GJ

[7] https://www.robotshop.com/es/es/kit-brazo-robotico-5dof-para-arduino-uno-r3-adeept.html

[8] https://www.tresding.com/ebotics/501-ebotics-arm-robot-kit-dyi-robotica-y-programacion-brazo-robot-y-mando-de-doble-joystick-8427542085517.html

[9] https://dle.rae.es/cinta

[10] https://ztrobotic.com/product/banda-transportadora-mige/

[11] https://www.robotshop.com/es/es/sensor-color-tcs3200.html

[12] https://www.ebay.es/itm/Modulo-detector-obstaculos-IR-sensor-infrarrojos-arduino/221687935501?hash=item339da1720d:g:nqIAAOSwk~ZaAeK4

[13] https://hetpro-store.com/TUTORIALES/microcontrolador/

[14] https://www.electronicoscaldas.com/datasheet/TCS3200-TCS3210_Taos-ams.pdf

[15] https://www.luisllamas.es/detectar-obstaculos-con-sensor-infrarrojo-y-arduino/

[16] http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

[17] https://www.ti.com/lit/ds/sprs439o/sprs439o.pdf?ts=1599264786615&ref_url=https%253A%252F%252Fwww.google.com%252F

[18] https://www.youtube.com/watch?v=7UsmJgHU6wk

[19] https://www.youtube.com/watch?v=AWoMMpc49hE

[20] https://www.robotshop.com/es/es/dfrobot-6v-180rpm-motor-engranaje-cc-micro-con-eje-trasero.html

[21] https://www.robotshop.com/es/es/juego-controlador-dual-motores-drv8835-para-raspberry-pi-b-.html

[22] https://www.pololu.com/product/2753

[23] https://jlcpcb.com/

ANEXO I: CABECERA "CLASIFICADORA.H"

```
#ifndef CLASIFICADOR_H_
#define CLASIFICADOR_H_

#include "DSP2833x_Device.h"

void Gpio_select(void);
void Setup_ePWM(void);
void delay_loop(long);

interrupt void cpu_timer0_isr(void);

void pos_ini(void);
void coge_objeto(void);
void clasifica_rojo(void);
void clasifica_verde(void);
void clasifica_azul(void);

void movimiento1(int , int);
void movimiento2(int, int );
void movimiento3(int, int );
void movimiento4(int, int );


#endif /* CLASIFICADOR_H_ */
```

ANEXO II: FUNCIÓN PRINCIPAL

```
#include "clasificador.h"

// Prototipado de funciones externas
extern void InitSysCtrl(void);
extern void InitCpuTimers(void);
extern void ConfigCpuTimer(struct CPUTIMER_VARS *,float ,float );
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);

//Variables, todas declaradas como globales
double cont;
int j,k=0,val_ant,flag,flag2,dec1_ant,dec2_ant,repite_medida=0;
int RGB[3]={0,0,0};
int cola[10]={0,0,0,0,0,0,0,0,0,0};

void main (void){

    InitSysCtrl();                          // Inicializacion del sistema
(DSP2833x_SysCtrl.c)

    DINT;                                   // Deshabilitación de todas las
interrupciones

    Gpio_select();                          // Seleccion de los GPIO que se usaran

    Setup_ePWM();                           // Configuración de los PWM

    InitCpuTimers();                        // Configuración inicial timer

    ConfigCpuTimer(&CpuTimer0,150,300000);  // configura timer con periodo de 300 ms

    InitPieCtrl();

    InitPieVectTable();

    EALLOW;
    PieVectTable.TINT0 = &cpu_timer0_isr;   // Relación salto del timer con rutina de
                                            // interrupción
    EDIS;

    PieCtrlRegs.PIEIER1.bit.INTx7 =1;

    IER = 0x0001;

    GpioDataRegs.GPASET.bit.GPIO1 = 1;      // Escalado de frecuencia al 20%
    GpioDataRegs.GPACLEAR.bit.GPIO5 = 1;

    GpioDataRegs.GPBCLEAR.bit.GPIO62 = 1;   // Pone el pin 62 de modo que la cinta gire
                                            // hacia un lado

    pos_ini();                              // Necesario para establecer posición
                                            // inicial conocida

    EPwm1Regs.CMPA.half.CMPA = 37500*0.9;   // Pone cinta a funcionar
```

```
while(1){

if(GpioDataRegs.GPADAT.bit.GPIO22==0 && dec2_ant==1 ){

  // Detectado objeto en posición de clasificacion

  EPwm1Regs.CMPA.half.CMPA = 37500;  // Para cinta

  // Se mira el primer elemento de la cola, para saber donde se clasifica

  switch(cola[0]){
                    case 1 :
                        coge_objeto();
                        clasifica_rojo();
                        break;
                    case 3 :
                        coge_objeto();
                        clasifica_azul();
                        break;
                    case 2 :
                        coge_objeto();
                        clasifica_verde();
                        break;
                    default:
                        // en caso de error, se vera como la cinta sufre un
                        //pequeño paro
                        delay_loop(2000000); //
                        break;
                }

     //Elimina primer elemento de la cola, y corre un lugar el resto

     for(k=0;k<9;k++){
     cola[k]=cola[k+1];
    }

    EPwm1Regs.CMPA.half.CMPA = 37500*0.9;  // Pone cinta a funcionar

}

if((GpioDataRegs.GPADAT.bit.GPIO23==0 && dec1_ant==1) || repite_medida==1){

    // Detectado objeto en posición de deteccion color

    EPwm1Regs.CMPA.half.CMPA = 37500;  // Para cinta


    repite_medida=1;

    /* habilita las interrupciones, para que pueda saltar el timer,
     debe hacerse cuando ya se haya parado la cinta, pues interfiere
     con el PWM. Antes de ponerla en marcha habrá que desactivasrlas*/

    EINT;

    while(repite_medida==1){ // si medida no concluyente, se repite

    for (j=0;j<3;j++){

        // case donde se selecciona el color a medir, según número de iteración
```

```
            switch(j){
                case 0 :
                    GpioDataRegs.GPACLEAR.bit.GPIO9 = 1;    // Medir rojo
                    GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
                    break;
                case 1 :
                    GpioDataRegs.GPASET.bit.GPIO9 = 1;      // Medir verde
                    GpioDataRegs.GPASET.bit.GPIO11 = 1;
                    break;
                case 2 :
                    GpioDataRegs.GPACLEAR.bit.GPIO9 = 1;    // Medir azul
                    GpioDataRegs.GPASET.bit.GPIO11 = 1;
                    break;
                default:
                    GpioDataRegs.GPASET.bit.GPIO9 = 1;      // Medir claro
                    GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
                    break;
            }

            flag=0;
            cont=0;

            GpioDataRegs.GPASET.bit.GPIO17 = 1;              //habilita medida

            val_ant=GpioDataRegs.GPADAT.bit.GPIO13;          // guarda primer valor
                                                             // para hacer la
comparacion
                                                             // luego

            CpuTimer0Regs.TCR.bit.TSS = 0;                   // inicio de temporizador

            while(flag==0){
                if (GpioDataRegs.GPADAT.bit.GPIO13==1 && val_ant==0) cont++;
                val_ant=GpioDataRegs.GPADAT.bit.GPIO13;
            }
            CpuTimer0Regs.TCR.bit.TSS = 1;                    //Para temporizador

            RGB[j]=cont*10/3;                    // Halla la frecuencia, calculando el
                                                 // numero de flancos medidos en un
segundo

            GpioDataRegs.GPACLEAR.bit.GPIO17 = 1;    // deshabilita medida

            delay_loop(5000000);
            }

           flag2=0;
           repite_medida=0;

           // Una vez realizada medida, se comprueba si se corresponde con algun color,
           // si es así, se guarda en el vector, sino, se activa repite_medida. Las
           // condiciones de estos ifs serán las que se deben modificar en caso de
           // recalibrar el sensor para otras condiciones de medida

            for(k=0;k<10 && flag2==0;k++){
                if(cola[k]==0){
                    flag2=1;

                    if(RGB[0]>5000){
                        cola[k]=1;
```

```
                    }else if(RGB[2]>4500){
                        cola[k]=3;
                    }else if((RGB[1]>RGB[0])&&(RGB[1]>RGB[2])){
                        cola[k]=2;
                    }else repite_medida=1;
                }
            }
        }

        DINT;                                   // Deshabilitación de todas las
                                                // interrupciones

        EPwm1Regs.CMPA.half.CMPA = 37500*0.9;   // Pone cinta a funcionar

        delay_loop(10000000);                   // ajustar para que de tiempo
                                                // a salir a objeto antes de flanco
    }

    dec1_ant=GpioDataRegs.GPADAT.bit.GPIO23;    // Actualizacion de valor anterior,
                                                //para deteccion de flancos
    dec2_ant=GpioDataRegs.GPADAT.bit.GPIO22;
    }

}

    interrupt void cpu_timer0_isr(void){

        /* En la rutina de interrupción del timer simplemente
         * se activa una variable que provoca la salida del
           bucle de que cuenta los flancos del sensor de color
         */

        flag=1;
        PieCtrlRegs.PIEACK.all = 0x0001;

    }
```

ANEXO III: FUNCIONES DE MOVIMIENTO DEL ROBOT

```
//###########################################################################
// Funciones de movimiento del brazo robotico
//###########################################################################

#include "clasificador.h"

void pos_ini(void){

    int ini1,ini2,ini3,cerrado;

    cerrado=5900;
    ini1=2100;              //motor derecho
    ini2=4150;              //motor base
    ini3=5400;              //motor izquierdo

    EPwm2Regs.CMPA.half.CMPA = ini1;
    delay_loop(20000);

    EPwm3Regs.CMPA.half.CMPA =ini2 ;
    delay_loop(20000);

    EPwm4Regs.CMPA.half.CMPA = ini3;
    delay_loop(20000);

    EPwm5Regs.CMPA.half.CMPA = cerrado;
    delay_loop(20000);

}

void coge_objeto(void){

    // Movimiento que recoge el objeto del punto de clasificación de objetos de
    // la cinta y lo deja sobre esa misma posición, elevado, esperando para ser
    // enviado a su punto de clasificación

    int ini1,des1,ini3,des3,cerrado_fuerte,abierto,des3_2,des1_2;

    abierto=6500;
    ini1=2100;
    ini3=5400;

    des3=4600;
    des3_2=4300;

    des1=2800;
    des1_2=2900;

    cerrado_fuerte=3000;

    EPwm5Regs.CMPA.half.CMPA = abierto;
    delay_loop(2000000);
```

```
    movimiento3(ini3,des3);
    delay_loop(3000000);

    movimiento1(ini1,des1);
    delay_loop(3000000);

    movimiento3(des3,des3_2);
    delay_loop(3000000);

    movimiento1(des1,des1_2);
    delay_loop(3000000);

    movimiento4(abierto,cerrado_fuerte);
    delay_loop(3000000);

    movimiento3(des3_2,ini3);
    delay_loop(2000000);

}

void clasifica_rojo(void){

    // Movimiento que parte del objeto elevado sobre la posición de
    // clasificación, lo envia al lugar de clasificación rojo, lo suelta,
    // y vuelve a la  posición inicial, la de cuando se inicia el código,
    // para que despues pueda ejecutarse coge_objeto de nuevo

    int ini1,ini1_fin,abierto,cerrado,des1;

    ini1=2750;

    abierto=6500;
    des1=4300;
    cerrado=5900;

    ini1_fin=2100;

    movimiento1(ini1,des1);
    delay_loop(2000000);

    EPwm5Regs.CMPA.half.CMPA = abierto;
    delay_loop(3000000);

    EPwm5Regs.CMPA.half.CMPA = cerrado;
    delay_loop(2000000);

    movimiento1(des1,ini1_fin);
    delay_loop(2000000);

}

void clasifica_azul(void){

    // Movimiento que parte del objeto elevado sobre la posición de
    // clasificación, lo envia al lugar de clasificación azul, lo suelta,
    // y vuelve a la  posición inicial, la de cuando se inicia el código,
    // para que después pueda ejecutarse coge_objeto() de nuevo

    int ini1,ini1_fin,abierto,cerrado,des1,ini2,des2;

    ini1=2750;
```

```
    abierto=6500;

    des1=4300;
    cerrado=5900;

    ini1_fin=2100;
    ini2=4150;
    des2=2900;

    movimiento2(ini2,des2);
    delay_loop(2000000);

    movimiento1(ini1,des1);
    delay_loop(2000000);

    EPwm5Regs.CMPA.half.CMPA = abierto;
    delay_loop(3000000);

    EPwm5Regs.CMPA.half.CMPA = cerrado;
    delay_loop(2000000);

    movimiento1(des1,ini1_fin);
    delay_loop(2000000);

    movimiento2(des2,ini2);
    delay_loop(2000000);


}

void clasifica_verde(void){

    // Movimiento que parte del objeto elevado sobre la posición de
    // clasificación, lo envia al lugar de clasificación verde, lo suelta,
    // y vuelve a la  posición inicial, la de cuando se inicia el código,
    // para que después pueda ejecutarse coge_objeto() de nuevo

    int ini1,ini1_fin,abierto,cerrado,des1,ini2,des2;

    ini1=2750;
    abierto=6500;

    des1=4300;
    cerrado=5900;

    ini1_fin=2100;
    ini2=4150;
    des2=1700;

    movimiento2(ini2,des2);
    delay_loop(2000000);

    movimiento1(ini1,des1);
    delay_loop(2000000);

    EPwm5Regs.CMPA.half.CMPA = abierto;
    delay_loop(3000000);

    EPwm5Regs.CMPA.half.CMPA = cerrado;
    delay_loop(2000000);
```

```
    movimiento1(des1,ini1_fin);
    delay_loop(2000000);

    movimiento2(des2,ini2);
    delay_loop(2000000);

}


void movimiento1(int pos, int des){

    // Función que ejecuta paso a paso el
    // movimineto del servomotor de la derecha,
    // el que controla el movimiento hacia
    //delante o hacia detrás

    int i;

    if(pos>des){

        for(i=pos;i>des;i--){
        EPwm2Regs.CMPA.half.CMPA = i ;
        // Modificar para movimiento más lento o más rápido
        delay_loop(4000);
        }

    }
    else{

        for(i=pos;i<des;i++){
        EPwm2Regs.CMPA.half.CMPA = i ;
        // Modificar para movimiento más lento o más rápido
        delay_loop(4000);
        }

    }
}

void movimiento2(int pos, int des){

    // Función que ejecuta paso a paso el
    // movimineto del servomotor de la base,
    // el que controla el movimiento de giro
    //sobre sí mismo del robot

    int i;

    if(pos>des){

        for(i=pos;i>des;i--){
        EPwm3Regs.CMPA.half.CMPA = i;
        // Modificar para movimiento más lento o más rápido
        delay_loop(4000);
        }

    }
    else{

        for(i=pos;i<des;i++){
        EPwm3Regs.CMPA.half.CMPA = i ;
        // Modificar para movimiento más lento o más rápido
        delay_loop(4000);
```

```
        }

    }
}

void movimiento3(int pos, int des){

    // Función que ejecuta paso a paso el
    // movimineto del servomotor de la izquierda,
    // el que controla el movimiento de subida
    // y bajada del robot

    int i;

    if(pos>des){

        for(i=pos;i>des;i--){
        EPwm4Regs.CMPA.half.CMPA = i ;
        // Modificar para movimiento más lento o más rápido
        delay_loop(4000);
        }

    }
    else{

        for(i=pos;i<des;i++){
        EPwm4Regs.CMPA.half.CMPA = i ;
        // Modificar para movimiento más lento o más rápido
        delay_loop(4000);
        }

    }
}
void movimiento4(int pos, int des){

    // Función que ejecuta paso a paso el
    // movimineto del servomotor de la pinza.

    // Para operaciones de abiero y cerrado sin
    // coger objeto no se usará, pues en estos
    // casos el movimiento de la cinta no
    // importa. Si  será importante a la hora de
    // coger un objeto, pues si lo hace demasiado
    // rápido, lo golpeará em lugar de atraparlo

    int i;

    if(pos>des){

        for(i=pos;i>des;i--){
        EPwm5Regs.CMPA.half.CMPA = i ;
        // Modificar para movimiento más lento o más rápido
        delay_loop(4000);

        }
    }
    else{

        for(i=pos;i<des;i++){
        EPwm5Regs.CMPA.half.CMPA = i ;
        // Modificar para movimiento más lento o más rápido
```

```
        delay_loop(4000);
        }

    }
}
```

ANEXO IV: FUNCIONES DE CONFIGURACIÓN

```
//###########################################################################
// Funciones de configuración
//###########################################################################

#include "clasificador.h"

 void delay_loop(long end){

     //Función que establece un pequeño retrado, dejando al DSP
     // ejecutando un bucle for que depende del valor introducido

    long i;

    for(i=0;i<end;i++){
        asm(" NOP");
    }
}

 void Gpio_select(void)
 {

     EALLOW;

     GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;     // GPIO1 como GPIO

     GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 0;     // GPI5 como GPIO

     GpioCtrlRegs.GPAMUX1.bit.GPIO9 = 0;     // GPIO9 como GPIO

     GpioCtrlRegs.GPAMUX1.bit.GPIO11 = 0;    // GPIO11 como GPIO

     GpioCtrlRegs.GPAMUX1.bit.GPIO13 = 0;    // GPIO13 como GPIO

     GpioCtrlRegs.GPAMUX2.bit.GPIO17 = 0;    // GPIO17 como GPIO


     GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;      // GPIO1 como salida

     GpioCtrlRegs.GPADIR.bit.GPIO5 = 1;      // GPIO5 como salida

     GpioCtrlRegs.GPADIR.bit.GPIO9 = 1;      // GPIO9 como salida

     GpioCtrlRegs.GPADIR.bit.GPIO11 = 1;     // GPI11 como salida

     GpioCtrlRegs.GPADIR.bit.GPIO13 = 0;     // GPIO13 como entrada

     GpioCtrlRegs.GPADIR.bit.GPIO17 = 1;     // GPIO17 como salida


     GpioCtrlRegs.GPAPUD.bit.GPIO22 = 0;     // Activa pullup en GPIO22

     GpioCtrlRegs.GPAMUX2.bit.GPIO22 = 0;    // GPIO22 como GPIO

     GpioCtrlRegs.GPADIR.bit.GPIO22 = 0;     // GPIO22 como entrada
```

```
    GpioCtrlRegs.GPAPUD.bit.GPIO23 = 0;     // Activa pullup en GPIO23

    GpioCtrlRegs.GPAMUX2.bit.GPIO23 = 0;    // GPIO23 como GPIO

    GpioCtrlRegs.GPADIR.bit.GPIO23 = 0;     // GPIO23 como entrada


    GpioCtrlRegs.GPBMUX2.bit.GPIO62 = 0;    // GPIO62 como GPIO

    GpioCtrlRegs.GPBDIR.bit.GPIO62 = 1;     // GPIO62 como salida


    GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;     // GPIO0 como EPWM1A

    GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1;     // GPIO2 como EPWM2A

    GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1;     // GPIO4 como EPWM3A

    GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 1;     // GPIO6 como EPWM4A

    GpioCtrlRegs.GPAMUX1.bit.GPIO8 = 1;     // GPIO8 como EPWM5A


    EDIS;

}

void Setup_ePWM(void)
{
    // configuración de los PWM: El uno es el que controla el motor de la cinta
    // y tiene una configuración difrente, el resto, 2 ,3 ,4 y 5, que son para
    // los servos, tienen la misma

    // Configuración del módulo Time-Base

    EPwm1Regs.TBCTL.bit.CLKDIV = 0;         // Pre-escalador CLKDIV = 1
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = 1;      // Pre-escalador HSPCLKDIV = 2
    EPwm1Regs.TBCTL.bit.CTRMODE = 2;        // Modo up - down
    EPwm1Regs.TBCTL.bit.PRDLD = 0;          // Shadow activado
    EPwm1Regs.TBCTL.bit.SYNCOSEL = 0;       // Sincronización deshabilitada
    EPwm1Regs.TBPRD = 37500;                // Periodo para 1KHz de fPWM

    EPwm2Regs.TBCTL.bit.CLKDIV = 5;         // Pre-escalador CLKDIV = 16
    EPwm2Regs.TBCTL.bit.HSPCLKDIV = 1;      // Pre-escalador HSPCLKDIV = 2
    EPwm2Regs.TBCTL.bit.CTRMODE = 0;        // Modo up
    EPwm2Regs.TBCTL.bit.PRDLD = 0;          // Shadow activado
    EPwm2Regs.TBCTL.bit.SYNCOSEL = 0;       // Sincronización deshabilitada
    EPwm2Regs.TBPRD = 46874;                // Periodo para 50 Hz de fPWM

    EPwm3Regs.TBCTL.bit.CLKDIV = 5;         // Pre-escalador CLKDIV = 16
    EPwm3Regs.TBCTL.bit.HSPCLKDIV = 1;      // Pre-escalador HSPCLKDIV = 2
    EPwm3Regs.TBCTL.bit.CTRMODE = 0;        // Modo up
    EPwm3Regs.TBCTL.bit.PRDLD = 0;          // Shadow activado
    EPwm3Regs.TBCTL.bit.SYNCOSEL = 0;       // Sincronización deshabilitada
    EPwm3Regs.TBPRD = 46874;                // Periodo para 50 Hz de fPWM

    EPwm4Regs.TBCTL.bit.CLKDIV = 5;         // Pre-escalador CLKDIV = 16
    EPwm4Regs.TBCTL.bit.HSPCLKDIV = 1;      // Pre-escalador HSPCLKDIV = 2
    EPwm4Regs.TBCTL.bit.CTRMODE = 0;        // Modo up
    EPwm4Regs.TBCTL.bit.PRDLD = 0;          // Shadow activado
```

```
EPwm4Regs.TBCTL.bit.SYNCOSEL = 0;       // Sincronización deshabilitada
EPwm4Regs.TBPRD = 46874;                // Periodo para 50 Hz de fPWM

EPwm5Regs.TBCTL.bit.CLKDIV = 5;         // Pre-escalador CLKDIV = 16
EPwm5Regs.TBCTL.bit.HSPCLKDIV = 1;      // Pre-escalador HSPCLKDIV = 2
EPwm5Regs.TBCTL.bit.CTRMODE = 0;        // Modo up
EPwm5Regs.TBCTL.bit.PRDLD = 0;          // Shadow activado
EPwm5Regs.TBCTL.bit.SYNCOSEL = 0;       // Sincronización deshabilitada
EPwm5Regs.TBPRD = 46874;                // Periodo para 50 Hz de fPWM

// Configuración del módulo Compare-Counter

EPwm1Regs.CMPCTL.bit.SHDWAMODE = 0;     // shahow activado para CMPA
EPwm1Regs.CMPCTL.bit.SHDWBMODE = 0;     // shahow activado para CMPB
EPwm1Regs.CMPCTL.bit.LOADAMODE = 0;     // shadow de CMPA cargado en TBCTR = 0
EPwm1Regs.CMPCTL.bit.LOADBMODE = 0;     // shadow de CMPB cargado en TBCTR = 0

EPwm2Regs.CMPCTL.bit.SHDWAMODE = 0;     // shahow activado para CMPA
EPwm2Regs.CMPCTL.bit.SHDWBMODE = 0;     // shahow activado para CMPB
EPwm2Regs.CMPCTL.bit.LOADAMODE = 0;     // shadow de CMPA cargado en TBCTR = 0
EPwm2Regs.CMPCTL.bit.LOADBMODE = 0;     // shadow de CMPB cargado en TBCTR = 0

EPwm3Regs.CMPCTL.bit.SHDWAMODE = 0;     // shahow activado para CMPA
EPwm3Regs.CMPCTL.bit.SHDWBMODE = 0;     // shahow activado para CMPB
EPwm3Regs.CMPCTL.bit.LOADAMODE = 0;     // shadow de CMPA cargado en TBCTR = 0
EPwm3Regs.CMPCTL.bit.LOADBMODE = 0;     // shadow de CMPB cargado en TBCTR = 0

EPwm4Regs.CMPCTL.bit.SHDWAMODE = 0;     // shahow activado para CMPA
EPwm4Regs.CMPCTL.bit.SHDWBMODE = 0;     // shahow activado para CMPB
EPwm4Regs.CMPCTL.bit.LOADAMODE = 0;     // shadow de CMPA cargado en TBCTR = 0
EPwm4Regs.CMPCTL.bit.LOADBMODE = 0;     // shadow de CMPB cargado en TBCTR = 0

EPwm5Regs.CMPCTL.bit.SHDWAMODE = 0;     // shahow activado para CMPA
EPwm5Regs.CMPCTL.bit.SHDWBMODE = 0;     // shahow activado para CMPB
EPwm5Regs.CMPCTL.bit.LOADAMODE = 0;     // shadow de CMPA cargado en TBCTR = 0
EPwm5Regs.CMPCTL.bit.LOADBMODE = 0;     // shadow de CMPB cargado en TBCTR = 0

// Configuración del módulo Action Qualifier

EPwm1Regs.AQCTLA.all = 0;
EPwm1Regs.AQCTLA.bit.CAD = 1;           // EPWM1A a low en CMPA down
EPwm1Regs.AQCTLA.bit.CAU = 2;           // EPWM1A a high en CMPA up
EPwm1Regs.AQCTLB.all = 0;
EPwm1Regs.AQCTLB.bit.CBD = 1;           // EPWM1B a low en CMPB down
EPwm1Regs.AQCTLB.bit.CBU = 2;           // EPWM1B a high CMPB up

EPwm2Regs.AQCTLA.all = 0;
EPwm2Regs.AQCTLA.bit.ZRO = 2;           // EPWM2 a high en 0
EPwm2Regs.AQCTLA.bit.CAU = 1;           // EPWM2A a low en CMPA

EPwm3Regs.AQCTLA.all = 0;
EPwm3Regs.AQCTLA.bit.ZRO = 2;           // EPWM3 a high en 0
EPwm3Regs.AQCTLA.bit.CAU = 1;           // EPWM3A a low en CMPB

EPwm4Regs.AQCTLA.all = 0;
EPwm4Regs.AQCTLA.bit.ZRO = 2;           // EPWM4 a high en 0
EPwm4Regs.AQCTLA.bit.CAU = 1;           // EPWM4A a low en CMPB

EPwm5Regs.AQCTLA.all = 0;
EPwm5Regs.AQCTLA.bit.ZRO = 2;           // EPWM a high en 0
EPwm5Regs.AQCTLA.bit.CAU = 1;           // EPWM5A a low en CMPA
```

```
    // Configuración del módulo Dead Band

    EPwm1Regs.DBCTL.bit.OUT_MODE = 0;       // Módulo activado
    EPwm2Regs.DBCTL.bit.OUT_MODE = 0;       // Módulo activado
    EPwm3Regs.DBCTL.bit.OUT_MODE = 0;       // Módulo activado
    EPwm4Regs.DBCTL.bit.OUT_MODE = 0;       // Módulo activado
    EPwm5Regs.DBCTL.bit.OUT_MODE = 0;       // Módulo activado

    // Configuración del módulo PWM-Chopper

    EPwm1Regs.PCCTL.bit.CHPEN = 0;          // Módulo descativado
    EPwm2Regs.PCCTL.bit.CHPEN = 0;          // Módulo descativado
    EPwm3Regs.PCCTL.bit.CHPEN = 0;          // Módulo descativado
    EPwm4Regs.PCCTL.bit.CHPEN = 0;          // Módulo descativado
    EPwm5Regs.PCCTL.bit.CHPEN = 0;          // Módulo descativado


    // Configuarción del módulo Trip Zone

    EPwm1Regs.TZCTL.bit.TZA = 3;            // No hacer nada
    EPwm1Regs.TZCTL.bit.TZB = 3;            // No hacer nada
    EPwm1Regs.TZEINT.all = 0;               // interrupción deshabilitada

    EPwm2Regs.TZCTL.bit.TZA = 3;            // No hacer nada
    EPwm2Regs.TZCTL.bit.TZB = 3;            // No hacer nada
    EPwm2Regs.TZEINT.all = 0;               // interrupción deshabilitada

    EPwm3Regs.TZCTL.bit.TZA = 3;            // No hacer nada
    EPwm3Regs.TZCTL.bit.TZB = 3;            // No hacer nada
    EPwm3Regs.TZEINT.all = 0;               // interrupción deshabilitada

    EPwm4Regs.TZCTL.bit.TZA = 3;            // No hacer nada
    EPwm4Regs.TZCTL.bit.TZB = 3;            // No hacer nada
    EPwm4Regs.TZEINT.all = 0;               // interrupción deshabilitada

    EPwm5Regs.TZCTL.bit.TZA = 3;            // No hacer nada
    EPwm5Regs.TZCTL.bit.TZB = 3;            // No hacer nada
    EPwm5Regs.TZEINT.all = 0;               // interrupción deshabilitada

    // Configuración del módulo Event Trigger

    EPwm1Regs.ETSEL.bit.SOCAEN = 0;         // Generación de SOCA deshabilitado
    EPwm1Regs.ETSEL.bit.SOCBEN = 0;         // Generación de SOCB deshabilitado
    EPwm1Regs.ETSEL.bit.INTEN = 0;          // Generación de señal de interrupción
deshabilitado

    EPwm2Regs.ETSEL.bit.SOCAEN = 0;         // Generación de SOCA deshabilitado
    EPwm2Regs.ETSEL.bit.SOCBEN = 0;         // Generación de SOCB deshabilitado
    EPwm2Regs.ETSEL.bit.INTEN = 0;          // Generación de señal de interrupción
deshabilitado

    EPwm3Regs.ETSEL.bit.SOCAEN = 0;         // Generación de SOCA deshabilitado
    EPwm3Regs.ETSEL.bit.SOCBEN = 0;         // Generación de SOCB deshabilitado
    EPwm3Regs.ETSEL.bit.INTEN = 0;          // Generación de señal de interrupción
deshabilitado

    EPwm4Regs.ETSEL.bit.SOCAEN = 0;         // Generación de SOCA deshabilitado
    EPwm4Regs.ETSEL.bit.SOCBEN = 0;         // Generación de SOCB deshabilitado
    EPwm4Regs.ETSEL.bit.INTEN = 0;          // Generación de señal de interrupción
deshabilitado
```

```
     EPwm5Regs.ETSEL.bit.SOCAEN = 0;          // Generación de SOCA deshabilitado
     EPwm5Regs.ETSEL.bit.SOCBEN = 0;          // Generación de SOCB deshabilitado
     EPwm5Regs.ETSEL.bit.INTEN = 0;           // Generación de señal de interrupción
deshabilitado
 }
```