

MICROCONTROLADORES Y ELECTRÓNICA DE POTENCIA

INFORME DE PROYECTO

Robot de almacén con 3 grados de libertad

Profesores:

- Iriarte, Eduardo
- Cruz, Martín

Autor:

- Aleo, Rodrigo - Legajo: 12106

TABLA DE CONTENIDOS

1. INTRODUCCIÓN	1
2. ESQUEMA TECNOLÓGICO	2
2.1. Comunicación PC-Bluepill	2
2.2. Drivers DRV8825 y motores PAP	5
2.3. Servomotor	5
2.4. Finales de carrera	6
3. FUNCIONAMIENTO GENERAL Y PROGRAMACIÓN	6
3.1. Comandos principales	6
3.2. Recepción de comandos	7
3.3. Ingreso de productos	7
3.4. Reacomodado	7
3.5. Retiro	8
3.6. Movimiento de motores a paso	8
3.7. Movimiento de servomotor	10
4. ETAPA DE MONTAJE Y ENSAYOS REALIZADOS	10
5. RESULTADOS	11
6. ENSAYO DE INGENIERÍA DE PRODUCTO	12
7. REFERENCIAS	14

1. INTRODUCCIÓN

En el presente proyecto, se ha desarrollado un robot de almacenamiento capaz de moverse en tres direcciones distintas para recoger y depositar objetos en lugares específicos. Este tipo de robot es muy útil en ambientes de almacenamiento y logística, ya que permite una manipulación precisa y rápida de productos, lo que a su vez aumenta la eficiencia y productividad en la gestión de inventarios.

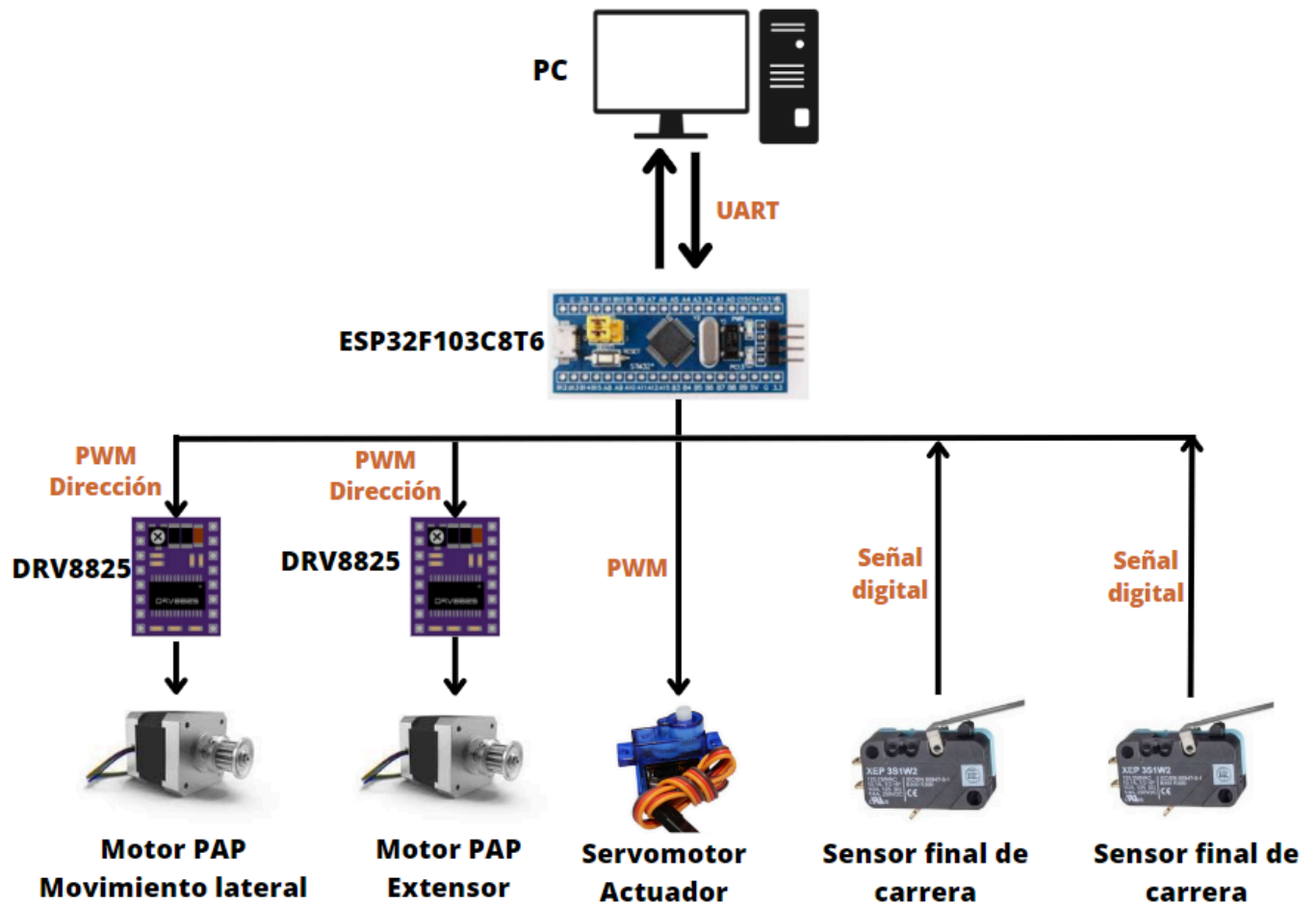
Entre las ventajas de este tipo de robot se encuentran la precisión y la rapidez en el manejo de objetos, la reducción de errores humanos, la disminución del tiempo de inactividad, el aumento de la productividad, y la optimización del espacio de almacenamiento.

Debido a la infinidad de usos que tienen esta clase de robots, se optó por enfocarlo a una aplicación específica: administrar el almacén de una farmacia.

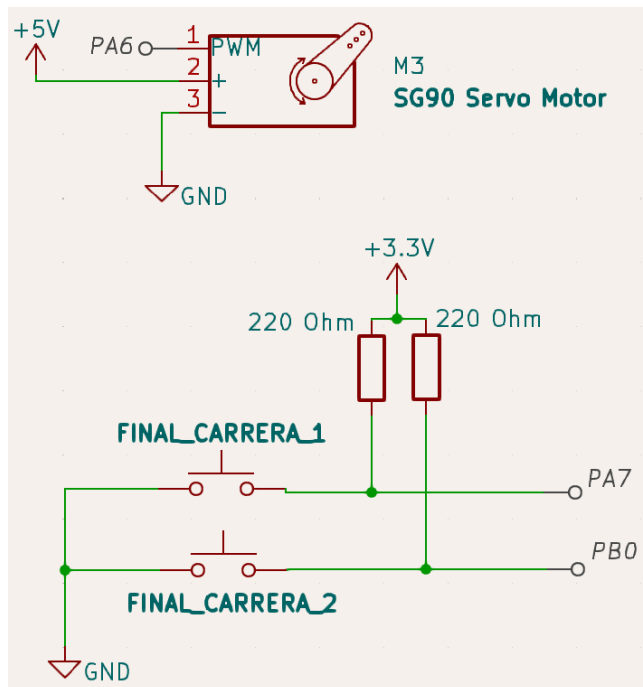
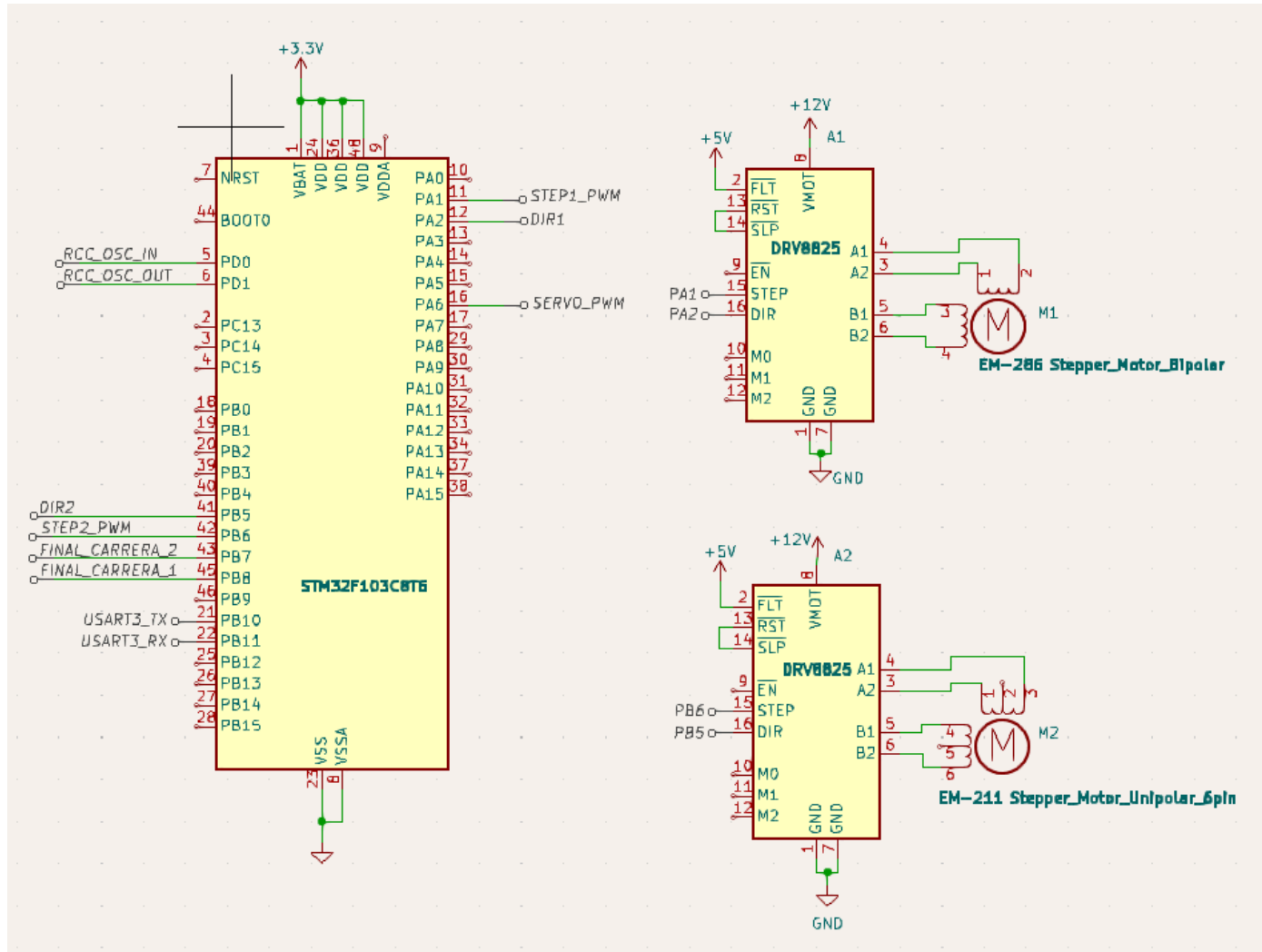
El proyecto fue llevado a cabo utilizando el microcontrolador ARM STM32F103C8T6, también llamado comunmente "Bluepill". Además, el robot en cuestión tiene un movimiento lineal, controlado por un motor paso a paso que le permite desplazarse por el almacén; también posee otro motor paso a paso que permite acercar al actuador a la estantería donde se encuentra el producto y, finalmente, un actuador de tipo pinza de mordazas paralelas accionado por un servo motor que será el encargado de sujetar las cajas que contienen a los diferentes productos.

2. ESQUEMA TECNOLÓGICO

En el siguiente esquema se puede observar la interrelación entre los diferentes sistemas y módulos utilizados en el proyecto:



Se adjunta, además, imágenes representativas de las conexiones realizadas:



2.1. Comunicación PC-Bluepill

La comunicación entre la PC y la Bluepill es necesaria ya que se utilizará para que el operador pueda ingresar los productos al sistema. Para ello se utiliza comunicación serie asíncrona utilizando los pines de la UART3 del microcontrolador y utilizando un conversor de TTL-USB para poder conectarlo a la computadora. Para la comunicación se utiliza una velocidad de 9600 baudios, sin bit de paridad, 1 bit de parada por cada byte de datos.

Por medio de una terminal serie, el operador puede enviar comandos al microcontrolador y tener feedback para comprobar si han, o no, habido errores en el proceso.

2.2. Drivers DRV8825 y motores PAP

Para controlar cada motor paso a paso de forma más eficiente que simplemente ir alimentando bobina a bobina desde el microcontrolador, se utiliza un driver DRV8825. Al mismo se le conectan las 2 bobinas de los motores paso, y se lo alimenta con la tensión que requiere dicho motor. Lo único que se conecta al microcontrolador es el pin STEP y el DIR. Variando la tensión entre un valor alto (3,3V) y bajo (0V) en el pin DIR se puede controlar la secuencia de alimentación de las bobinas, lo que permite invertir el sentido de giro del motor. Además, dicha secuencia se da cuando por el pin STEP se reciben pulsos en forma de onda cuadrada. Variando la frecuencia de esta onda se puede variar la velocidad del motor. Se adjunta en “Referencias” la hoja de datos del driver.

Con respecto a los motores PAP utilizados, se tienen las diferentes características:

Motor PAP	EM-286	EM-336
Función en el proyecto	Movimiento lateral	Extender brazo para alcanzar estantería.
Pasos por vuelta	96	200
° por paso	3,75°	1,8°
R por fase	17 Ω	7 Ω
V alimentación	12V	12V
Imax	Desconocida	Desconocida

Debido a que se desconoce la corriente máxima que pueden utilizar estos motores, se procede a colocar una $V_{ref} = 0,3V$ en el driver. Es un valor conservador ya que mantiene baja la corriente que es suministrada a los motores. Para tener una referencia, un motor Nema 17 de características aproximadas a los utilizados, tiene una corriente máxima de 700 mA. En este caso, entre ambos motores se consumen tan solo 400 mA de la fuente de alimentación.

Cuando se aplica una corriente más alta, se genera un campo magnético más fuerte en las bobinas del motor, lo que a su vez produce más fuerza en el rotor y, por lo tanto, más torque. Además se permite que el motor alcance mayores velocidades. Sin embargo, es importante tener en cuenta que el

aumento de la corriente también genera más calor en el motor, lo que puede ser un problema si no se disipa adecuadamente.

2.3. Servomotor

Se utiliza el servomotor SG90. Es de bajo torque, por lo que no es práctico en una aplicación de este calibre, pero sirve para simular el comportamiento de un servomotor de mayores dimensiones. Es alimentado por medio de 5V. El modo de utilización de este servo, es que por el cable de señal se debe enviar una señal PWM de 50Hz. Variando el duty cycle entre 2,5% y 12,5% se logra modificar su ángulo entre 0° y 180°. Se adjunta su hoja de datos en el apartado de “Referencias”.

2.4. Finales de carrera

Son necesarios para poder inicializar la posición de los motores paso a paso. Para ello se utilizan unos pulsadores normal abierto conectados a 5V que generan la señal que recibe el microcontrolador. Los pines se configuran para estar conectados a una resistencia interna de pull-up para enviar un 1 lógico. Cuando se hace la maniobra de referencia, al ser pulsados, envían un 0 lógico al controlador.

3. FUNCIONAMIENTO GENERAL Y PROGRAMACIÓN

3.1. Comandos principales

Al inicializar el microcontrolador, envía un mensaje a la PC que especifica los diferentes comando que se pueden realizar.

Inicialmente, estos son 4:

- Ingresar *nombre_producto* *prioridad_producto*
- Retirar *nombre_producto*
- Home
- Almacén
- Reiniciar
- Guardar
- Cargar
- Rpm *velocidad_en_rpm*

El comando “ingresar” debe recibir dos parámetros, el nombre del producto y el valor de prioridad. El nombre del producto debe estar dentro de la base de datos previamente definida en el programa del microcontrolador. Esto es porque en dicha base de datos se especifican las dimensiones de cada producto, para que pueda saberse cuánto debe cerrarse la pinza para poder sujetar la caja.

La prioridad del producto ayuda a decidir qué posición ocupará el producto en el almacén. Este dato es representativo. En la aplicación farmacéutica, por ejemplo, podría tratarse de una fecha de vencimiento del producto.

El comando “retirar” se utiliza para extraer el producto de dicho tipo que se encuentre más cercano a la entrada. Como los productos son ordenados por su prioridad, se extraerá entonces el producto con fecha de vencimiento más próxima.

“Home” se encarga de inicializar la posición de los motores paso a paso haciéndolos desplazarse en la dirección del sensor fin de carrera a una velocidad moderada. Una vez accionado el sensor, se puede definir la posición 0. Los motores luego podrán moverse a su velocidad normal. Esta operación es necesario realizarla al comienzo de la operación y de forma ocasional por las dudas de que el motor se haya salteado pasos y haya perdido su referenciación.

“Almacén” muestra por la consola un resumen de los productos almacenados actualmente en el sistema.

“Guardar” sirve para escribir en la memoria flash el estado actual del almacén. Dicho estado puede luego recuperarse usando el comando “cargar”.

“Rpm” permite variar la velocidad de rotación del motor principal (el que genera el movimiento lateral).

3.2. Recepción de comandos

Cuando el usuario ingresa un comando por consola, los datos se reciben por la UART del microcontrolador y se activa la interrupción correspondiente. En dicha interrupción el comando es almacenado en una lista llamada “cola de comandos” y activa un flag que indica que hay comandos en cola. Finalmente se vuelven a activar las interrupciones por UART para recibir nuevos comandos y colocarlos en cola.

En el while principal, si se detecta que está el flag en 1, se activa la función “intérprete de comandos” que es la encargada de derivar el comando a la función correspondiente para atenderlo.

3.3. Ingreso de productos

El almacén tiene un tamaño que puede ser configurado al ajustar una variable.

La posición de entrada y salida de los productos es la posición 0. De 1 hasta n serán las posiciones disponibles en el almacén. Se podría haber configurado además que la entrada y salida sean por posiciones diferentes para evitar confusiones, pero para este prototipo no es necesario.

Al ingresar un producto al sistema, se busca el primer espacio vacío en el almacén donde despacharlo. No se tiene en cuenta la prioridad en esta fase y esto es porque, suponiendo que hay una cadena de productos por ingresar, irlos acomodando al tiempo que ingresan puede derivar en realizar más movimientos de los necesarios. Por simplicidad, se prefiere primero ingresar todos los productos y luego acomodarlos, sin embargo no se descarta que la otra opción sea también válida en otras circunstancias.

Es importante aclarar que este es el criterio elegido para el ingreso, pero podría haberse optado por cualquier otro que pueda requerir un cliente a la hora de comprar un robot de este estilo.

3.4. Reacomodado

La forma en que se puede gestionar el inventario es modificable según las necesidades de la aplicación. En este caso en particular, se optó por que los productos se reordenen teniendo en cuenta su prioridad y la variedad de productos. En el caso en cuestión, se toma que la prioridad es un número entero entre 1 y 8, donde 1 es la máxima prioridad y 8 la peor. Los espacios vacíos se consideran con prioridad 9. Estos números son arbitrarios, podría elegirse un rango más amplio, pero sí es importante que los espacios vacíos tengan la última prioridad.


El reacomodado se da dentro del while principal cuando se ha cumplido con todos los comandos, el flag se pone en 0 y se abandona la función “intérprete de comandos”. En este momento el robot, en vez de entrar en inactividad, ingresa en una función llamada “Intentar reacomodar”, donde a cada elemento se le calcula un valor denominado “pseudoprioridad” que está basado en el tipo de producto y en la prioridad que tiene.

La pseudoprioridad tendrá valor 1 o 2. Los productos de cierto tipo que tengan la prioridad más alta, tendrán pseudoprioridad 1, mientras que el resto del mismo tipo o de peor prioridad tendrán valor 2. Los productos con 1 serán reacomodados en los primeros lugares cerca de donde se despachan los productos.

Este método nos asegura que en las primeras posiciones tendremos por lo menos el elemento de cada tipo con la mejor prioridad.

A continuación se muestra un ejemplo del reacomodado antes y después:

Pos	Producto	Prioridad	Pseudo prioridad
1	Ibuprofeno	1	1
2	Ibuprofeno	1	2
3	Paracetamol	2	1
4	Vacío		
5	Vacío		



Pos	Producto	Prioridad	Pseudo prioridad
1	Ibuprofeno	1	1
2	Paracetamol	2	1
3	Vacío	-	-
4	Ibuprofeno	1	2
5	Vacío	-	-

Como se puede ver, ya hay un ibuprofeno en posición 1, por lo tanto, al ibuprofeno en la posición 2, no importa que tenga prioridad 1, se lo pasa a la posición vacía en 4 y se arrima el paracetamol a la posición 2. Se le da más importancia a la variedad. Si se ingresara un paracetamol de prioridad 1 en el espacio 3, el paracetamol de prioridad 2 se movería al espacio 5 para dejarle el lugar al de mejor prioridad.

Es importante aclarar que debe haber por lo menos un lugar vacío en el almacén para poder reacomodar. No es deseable que el robot esté utilizando el lugar por donde se ingresan/despachan los productos para hacer esta operación ya que puede derivar en confusiones.

3.5. Retiro

Es importante que el retiro de los productos se haga en base a su prioridad, por lo tanto el robot sacará siempre el producto de mayor prioridad de su inventario. Generalmente, como el robot en su tiempo libre reacomoda las cajas, la de mayor prioridad suele ser la más cercana a la entrada/salida.

3.6. Movimiento de motores a paso

La dirección de los motores paso a paso se maneja mediante una salida digital del microcontrolador.

Cuánto gira y a qué velocidad son manejados por una salida PWM de los timers de propósito general que posee el microcontrolador.

Ambos motores paso a paso tienen comportamientos similares, por lo que tienen aproximadamente la misma configuración. Cada uno está conectado a un timer diferente, sin embargo también podría haberse utilizado el mismo timer ya que para la aplicación no se requiere que todos los motores estén funcionando al mismo tiempo.

La frecuencia del oscilador es de 72 MHz. Se desea tener una salida de 1 MHz en cada timer (es un valor arbitrario establecido por conveniencia ya que podría funcionar con otras frecuencias), por lo tanto se calcula el prescaler necesario de la siguiente manera:

$$prescaler = \frac{F_{osc}}{F_{timer}} - 1 = 71$$

Se configuran los pines correspondientes como salidas PWM. En el modo PWM el timer de 16 bits cuenta de 0 a 65535, y se establece un valor apartir del cuál la salida estará en 3,3V. Mientras el contador esté por debajo de dicho valor, la salida está en 0. Contar hasta 65535 con dicha frecuencia demorará:

$$T_{pwm} = \frac{1}{F_{pwm}} * 65535 = 0.0655 s/pulso = 6,55 ms/pulso$$

Cada ciclo de pulsos máximo demorará 6,55 ms, lo cual define la velocidad mínima del motor paso a paso. Se opta por esta velocidad mínima porque sirve para poder contar fácilmente a ojo la cantidad de vueltas que da el motor (para hacer pruebas y una demostración visual de la implementación del robot previa a la implementación mecánica). En la implementación práctica de todas formas, se los termina utilizando a mayores velocidades, por lo que podría haberse elegido una frecuencia de timer más alta. Este valor de ciclo de PWM puede ser modificado al cambiar el valor del registro ARR del timer en cuestión, que hace referencia al valor al cual genera el desborde. Se puede elegir un valor entre 0 y 65535 donde mientras menor sea, logrará ciclos más cortos y, por lo tanto, el motor girará más rápido.

El usuario puede cambiar la velocidad de los motores. El rango de velocidades está siendo definido, pero el valor de ARR para conseguir dicha velocidad está dado por la siguiente fórmula:

$$ARR = \frac{60 * F_{timer}}{rpm * N_{pasos}} - 1$$

Donde rpm es la velocidad definida por el usuario, y Npasos es la cantidad de pasos que debe dar el motor PAP para lograr una revolución.

El valor por comparación para cambiar el estado de la salida, se almacena en el registro CCTx (x es el número de canal del timer). Para generar una onda cuadrada, este valor debe ser siempre la mitad de ARR, por lo que se trabaja siempre con un duty cycle del 50%.

Para dejar el motor inhabilitado basta con definir a ARR como 0. Cuando ARR se le define en un valor diferente y a CCTx como la mitad de este valor, el motor comenzará a girar. Siempre es importante también reiniciar el contador (registro CNT) para evitar conteos inexactos.

Cuando el valor del conteo alcanza a ARR, se activa la interrupción de desborde de timer, en el cual se aumenta en 1 una variable que lleve el conteo general de los pulsos enviados al motor. Al cumplir la cantidad de pulsos necesarias para hacer girar al motor cierto ángulo, ARR se define nuevamente en 0 para detener el motor.

Distancias con modelo mecánico

En el modelo mecánico, se propone una división de 6 casillas (1 para entrada y salida y 5 para almacenar productos). Se establece que cada casilla está a 250 pasos de motor una de otra (un poco más de una vuelta, considerando que el motor necesita 200 para generar una revolución). Con esta distancia, se logra la subdivisión necesaria.

Para el motor que alarga el brazo, se determinó prácticamente que 50 pulsos son los necesarios para hacer que el brazo se extienda por completo.

3.7. Movimiento de servomotor

Para el servomotor se utilizó también un timer con frecuencia de 1MHz. Como para el servomotor SG90 se necesita enviar una onda PWM con frecuencia de 50 Hz, se modifica el ARR utilizando la siguiente fórmula:

$$ARR = \frac{F_{timer}}{F_{pwm}} - 1 = 19999$$

Con este valor se genera un ciclo de PWM de 20 ms (50 Hz). Colocando a CCTx a ese valor se genera un duty cycle del 100%. Teniendo esto en consideración se pueden calcular los valores de CCTx para lograr los duty cycle que establecerán el ángulo al que girará el servo:

Ángulo	Duty Cycle	CCTx
0°	2,5%	500
90°	7,5%	1500
180°	12,5%	2500

Como puede verse, los 180° se reparten entre 2000 posibles valores de CCTx, por lo tanto el servo utilizado tiene una resolución de 0,9°.

Ángulos en modelo mecánico

La pinza está en estado abierto a los 180°.

Se estableció que hay dos tipos de cajas que el robot debe agarrar. Para el tipo de caja 1, el servomotor girará 90°, mientras que para el tipo de caja 2, alcanzará los 0°. Este valor se mantiene mientras la caja está siendo sujeta. El valor de los 0° (pinza completamente cerrada) no es práctico, sólo es demostrativo para que se denote la diferencia entre los dos tipos de caja.

3.8. Guardado en memoria flash

De los 128 sectores de memoria es importante no utilizar los primeros porque es allí donde se guarda el programa que ejecuta el microcontrolador. Es por ello que se utilizó el sector de memoria 126. El espacio que ocupa el guardado de la información es de unos pocos bytes ya que solo se almacena el nombre de los productos en el almacén y su prioridad. Esta operación ahora es accionada por el usuario, pero debe darse de forma automática al ocurrir un evento que ponga en peligro la alimentación del microcontrolador con el objetivo de no perder la información hasta el momento. No conviene estar haciendo escritura en memoria flash de forma tan frecuente, porque la memoria EEPROM se desgasta luego de varios miles de ciclos y además es una operación que demora varios ciclos de reloj en realizar.

4. ETAPA DE MONTAJE Y ENSAYOS REALIZADOS

Para el montaje del prototipo, se optó primero por probar el correcto funcionamiento de cada componente por separado.

Conexión a PC

El STlink-V2 utilizado no es uno original, por lo que cargar el programa desde el STM32CubeIDE no era posible. Se procedió a compilar el programa y armar un archivo .hex que luego era cargado a la Bluepill por medio del STM32CubeProgrammer.

El proceso de debug se realizó de forma manual enviando mensajes por consola utilizando la comunicación serie por medio del conversor TTL-USB. Cabe aclarar que el conversor tenía su carcasa dada vuelta y, por lo tanto, el esquema de pines también estaba invertido, lo que causaba que el dispositivo no funcione y que además caliente bastante al estar conectado. Afortunadamente no se dañó ningún componente, y al dar cuenta el error, se pudo corregir y utilizar normalmente.

Motores PAP

Como los motores paso a paso fueron obtenidos de impresoras viejas, no se encontraron hojas de datos oficiales, por lo que en una primera instancia se tuvo que conocer estos motores midiendo su resistencia interna, su modo de conexión, contar la cantidad de pasos requeridos para dar una vuelta y ajustar la tensión del driver a valores que no generen altas temperaturas en el motor. Debido a que se desconoce la corriente máxima que pueden soportar estos motores, se optó por una tensión de referencia del driver baja, pero que, al menos sin la implementación mecánica, permiten un fluido movimiento de los motores a altas velocidades. Sí se presentan pitidos agudos debido a ruido eléctrico generado por los drivers DRV8825. Hay drivers más sofisticados como los TMC2100 que permiten al motor funcionar emitiendo mucho menos ruido.

Servomotor

Con respecto al servomotor, al ser muy común, si se cuenta con hojas de datos oficiales, por lo que su configuración fue mucho menos tediosa. Sí ocurre que el servo SG90 utilizado es muy sensible al ruido eléctrico, por lo que frecuentemente se pueden ver fluctuaciones en la pinza sin que la señal PWM sea modificada en el programa.

5. RESULTADOS

Al unificar todas las partes y realizar el código principal, se logró tener un prototipo funcional del robot en el cual se pueden observar todas sus partes moviéndose de forma coordinada y respondiendo de forma exitosa a los comandos ingresados por el operador, el cuál era el principal objetivo del proyecto.

Los motores, en su configuración actual tienen un consumo de 400 mA a 12V entre ambos de forma constante cuando están conectados sin girar. Esto provoca calentamiento en los motores pero no se los puede desenergizar para no perder la referencia. Una posible mejora es que si están mucho tiempo inactivos, manejando el pin SLEEP, se haga entrar al driver en un modo de bajo consumo para disminuir la corriente en las bobinas del motor. También se podría mejorar la refrigeración del motor con algún ventilador.

Al motor 1 (movimiento lateral) se lo pudo hacer alcanzar una velocidad de 200 rpm en la aplicación mecánica. La correa utilizada no corresponde al engranaje del motor, por lo que es susceptible a perder pasos cuando se supera dicha velocidad. Al motor 2 (extensión de brazo), debido a que se utilizan un sistema engranaje-cremallera impreso en 3D, para evitar roturas, se lo limita a una velocidad de 50 rpm. Ambos motores, en vacío lograron alcanzar velocidades de hasta 600 rpm, pero como se ha visto, éstas no son velocidades prácticas. Para lograr una mayor velocidad habría que utilizar rampas de velocidad, o incluso rampas de aceleración (también a la hora de realizar el frenado). Si se logra evitar perder pasos, estos motores tienen una alta precisión para la aplicación en cuestión.

Con respecto al servomotor, se observó que tiene eventuales fluctuaciones al reajustar el ángulo. Por un lado también habría que aplicar las rampas de velocidad, pero además lo que se realizó fue no hacer girar al servo hasta sus valores extremos. Se le dió un margen de error para que el servo defina correctamente el ángulo al cual dirigirse. Con esto se solucionó en gran parte las fluctuaciones en los extremos (en vez de manejar CCTx entre 500 y 2500, se lo hizo entre 600 y 2400). El servomotor tiene un consumo de 100mA a 5V, por lo que es de muy bajo consumo.

6. ENSAYO DE INGENIERÍA DE PRODUCTO

A estos tipos de robot se los denomina mini-load AS/RS (Automated Storage and Retrieval Systems) para productos de bajo peso, ya que también pueden utilizarse para transportar grandes productos.

- **Mecánica**

Se realiza la instalación de unas guías mecánicas en el suelo o en el techo de la habitación a las que el robot sería anclado y mediante un actuador se movería por el camino prefijado. Esta alternativa aporta robustez y sencillez para la aplicación deseada, sin embargo, tiene poca flexibilidad ya que una vez

instalado no se puede modificar y además su precio es más elevado al de otras alternativas (por ejemplo instalar cinta negra en el suelo para dibujar la trayectoria que se desee que realice el robot).

- **Detección de la ubicación**

Se colocan sensores inductivos o de laser para informar al robot de que está pasando por un punto determinado o una bifurcación concreta. Se deben colocar a lo largo de las guías.

- **Motores**

El tipo de motor utilizado en un mini-load AS/RS puede variar dependiendo del diseño y las especificaciones del sistema, pero a continuación se describen algunos de los motores más comunes que se utilizan:

Motores de corriente continua (DC) brushless: se utilizan comúnmente en sistemas mini-load AS/RS debido a su capacidad para proporcionar un alto torque en un tamaño compacto. Estos motores también son fáciles de controlar y pueden proporcionar una velocidad variable.

Motores de corriente alterna (AC): se utilizan debido a su alta eficiencia energética y su capacidad para proporcionar una velocidad variable. Estos motores son ideales para aplicaciones que requieren una alta precisión de velocidad.

Motores de paso a paso: son una opción popular para los sistemas mini-load AS/RS debido a su alta precisión y su capacidad para operar en velocidades bajas y altas. Estos motores también son fáciles de controlar y proporcionan un torque constante hasta cierta velocidad.

Motores servo: debido a su alta precisión y capacidad de proporcionar una velocidad y torque variables. Estos motores son ideales para aplicaciones que requieren un control de posición preciso y rápido.

Es importante tener en cuenta que la selección del motor adecuado dependerá de varios factores, como la carga útil del sistema, la velocidad requerida, el torque y la precisión de posicionamiento necesaria.

En las aplicaciones farmacéuticas, el robot puede tener un peso entre los 150 y los 300 kg y moverse a velocidades de 3000 rpm con suma precisión.

- **Supervisión remota**

El sistema se comunica en todo momento con una aplicación externa que supervisa su funcionamiento y almacena la información de forma externa para poder recuperarla en caso de cortes de alimentación del robot.

- **Ingreso autónomo**

El operario simplemente envía todas las cajas juntas y el robot es el encargado de realizar la separación y clasificación utilizando una cámara en el actuador que le permite leer el código de barras del producto e ingresarlas a la base de datos para que sean posteriormente almacenadas.

- **Grados de libertad**

Para aprovechar en mayor medida el espacio físico del almacén, el robot debe poder desplazarse de forma planar (para los costados y hacia arriba). Además se suele hacer que el robot pueda girar 180° la dirección del actuador para poder trabajar con estanterías adelante y atrás.

- **Sensor de proximidad**

Tener un sensor en el actuador para detectar cuándo se está a la distancia correcta de una caja y poder asegurar un buen agarre.

- **Medidas de detección de errores**

Se los utiliza junto a sistemas de alimentación auxiliar que permite que continuen funcionando a pesar de un corte en la alimentación principal. Puede utilizarse para realizar una parada de segura.

Colocar un encoder al eje del motor paso a paso permite detectar cuando el mismo se ha bloqueado y ha perdido pasos.

Además se pueden colocar sensores de temperatura para detectar un sobrecalentamiento y detener al motor antes que el daño sea permanente. Los drivers más complejos también permiten detectar picos de voltaje y sobrecalentamiento para enviar una señal que detenga la operación.

Debido a las altas velocidades que se manejan, debe haber sensores que detecten la entrada de un operario en la habitación para realizar una parada de emergencia y evitar posibles lesiones.

- **Posicionamiento de mayor precisión**

Para aplicaciones donde las distancias que tenga que recorrer el robot sean mayores, conviene utilizar motores DC con encoder, ya que tiene ciertas ventajas como lo son mayores velocidades y trabajan en lazo cerrado, por lo que se puede conocer en todo momento la posición del motor y es más fácil detectar cuando están sobrecargados.

7. REFERENCIAS

- **Hoja de datos STM32F103C8T6**

RM0008 REFERENCE MANUAL: STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs. (s. f.). ST.

https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf

- **Hoja de datos DRV8825**

alldatasheet.com. (s. f.). *DRV8825 pdf, DRV8825 Description, DRV8825 Datasheet, DRV8825 view* :::

ALLDATASHEET :: <https://pdf1.alldatasheet.com/datasheet-pdf/view/432263/TI/DRV8825.html>

- **Hoja de datos TowerPro SG90 Servomotor**

RM0008 REFERENCE MANUAL: STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs. (s. f.). ST.

https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf