

# Microcontroladores y Electrónica de Potencia

Trabajo integrador:

*Control de actuadores de robot SCARA*

Año: 2021

Año de cursado: 2020

Integrantes:

Costarelli, Ignacio Agustín (10966)

Reinoso, Maximiliano Gabriel (11754)

## Resumen

En este proyecto se logra implementar un sistema maestro-esclavo entre 4 microcontroladores Atmega 328p los cuales controlan el movimiento de los actuadores. Se desarrolla un esquema tecnológico con análisis de sus componentes. Se implementa protocolo de comunicación UART con el usuario y comunicación SPI entre microcontroladores, también se hace uso de interrupciones y timers que poseen los 328p.

Se desarrolla una lógica de funcionamiento la cual es implementada en lenguaje C, se prueba el correcto funcionamiento desarrollando simulación en Proteus para su posterior implementación física.

# Índice

<b>Resumen</b>	<b>2</b>
<b>Índice</b>	<b>3</b>
<b>1. Introducción</b>	<b>4</b>
<b>2. Esquema tecnológico</b>	<b>4</b>
<b>3 Detalle de módulos</b>	<b>5</b>
3.1 Arduino UNO	5
3.2 Motor PaP Unipolar 6 hilos	6
3.3 Driver DRV8825	6
<b>4. Funcionamiento general</b>	<b>7</b>
4.1 Diagrama de flujo	7
<b>5. Programación</b>	<b>10</b>
5.1 Información simulación de Matlab.	10
5.2 Main Maestro.	10
5.3 Main esclavos.	12
<b>6. Etapas de montaje y ensayos realizados</b>	<b>13</b>
6.1 Ensayos preliminares en Proteus.	13
6.2 Armado físico del circuito.	14
<b>7. Resultados, especificaciones finales</b>	<b>16</b>
<b>8. Conclusiones</b>	<b>16</b>
<b>9. Referencias</b>	<b>16</b>

# 1. Introducción

En el siguiente trabajo se ha desarrollado un sistema de control para los motores de un robot SCARA el cual estará realizando una tarea secuencial de pick and place entre dos puntos fijos los cuales han sido llamados punto A y punto B. Este tipo de tareas son de repetitividad y requieren optimización de tiempos para poder mover el mayor volumen posible de elementos. Por lo cual se ha planteado la utilización de múltiples microcontroladores ATMEGA328p en sistema maestro-esclavo con el fin de poder realizar un control distribuido y poder controlar múltiples acciones en simultáneo. Se busca la implementación del conocimiento adquirido en el cursado de la cátedra haciendo uso de programación en C, el manejo de motores, implementación de comunicación UART, SPI, así como también la electrónica básica para llevar adelante la conexión entre los distintos componentes del sistema.

# 2. Esquema tecnológico

Un primer acercamiento para explicar el proyecto se hace mediante el esquema tecnológico de la figura 1, que introduce la función de cada elemento y cómo se relacionan entre sí.

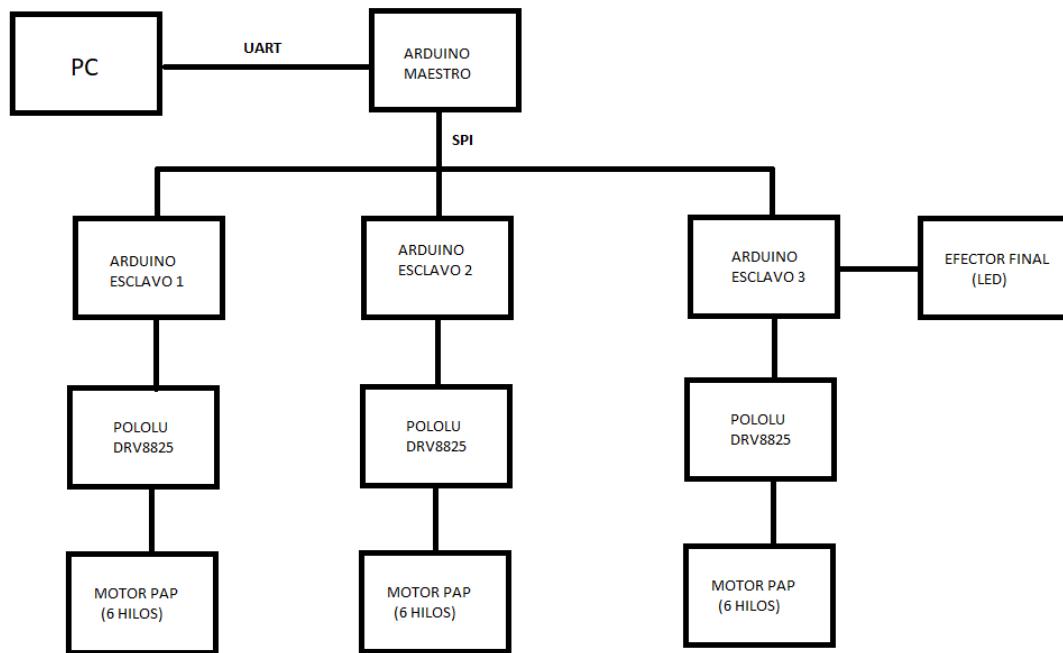


Figura 1: Esquema simplificado del proyecto

**PC:** Es la encargada de funcionar como interfaz para la comunicación usuario maestro, una vez iniciado el programa el usuario podrá ingresar parámetros y recibir información sobre el funcionamiento del mismo.

**ARDUINO 328p - Maestro:** Este microcontrolador se encarga de recibir las indicaciones del usuario y comenzar con la tarea siguiendo un orden establecido de indicaciones que envía a los distintos esclavos por comunicación SPI.

**ARDUINO 328p Esclavo:** Estos microcontroladores cuando reciban la orden del maestro, se encargarán de ejecutar la función que controla el movimiento de los motores y del efecto final.

**DRV8825:** Son los drivers utilizados para los motores PaP, estos simplifican la forma de controlar los motores.

**MOTOR PaP:** Motores paso a paso unipolares de 6 hilos, son los actuadores que mueven de forma directa cada uno de los ejes del brazo SCARA.

**EFFECTOR FINAL:** Se encarga de sujetar los elementos a mover en la tarea. Puede ser de distintos tipos como una pinza, un succionador neumático o del tipo magnético. Se encuentra en el extremo del brazo, en nuestro caso representado por una luz led la cual indicará el estado libre o en carga.

## 3 Detalle de módulos

A continuación se detalla un poco más en profundidad los distintos módulos principales que componen este proyecto. Los cuales son 3, en la parte de anexo se puede encontrar los datasheet del fabricante que fueron utilizados para extraer la información necesaria.

### **3.1 Arduino UNO**

Este arduino tiene un Atmega328p que es un microcontrolador AVR 8-bit basado en arquitectura RISC, en la figura 2 se puede ver la funcionalidad de los 28 pines que posee. Este arduino usa un clock externo provisto por un cristal de 16MHz. Para este proyecto se hará uso de la comunicación UART, SPI, timers y pines de Entrada/Salida de Propósito General (GPIO). La programación del mismo se realizó en lenguaje C en el entorno de Atmel Studio.

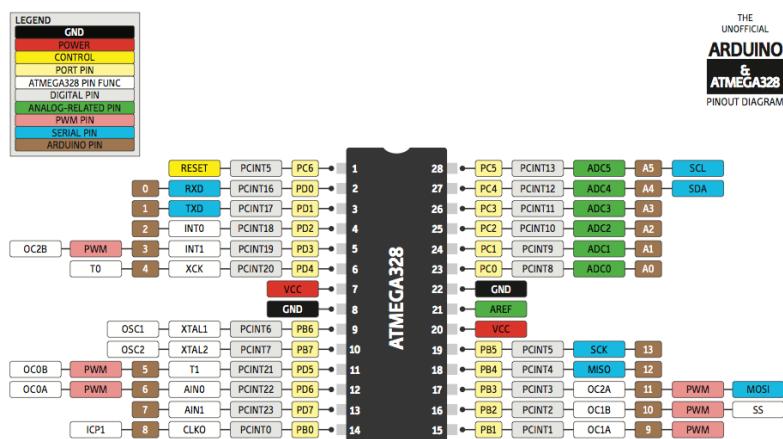


Figura 2: Microcontrolador ATMEGA328

### 3.2 Motor PaP Unipolar 6 hilos

Los motores utilizados son unipolar de 6 hilos dado que fueron reciclados de una fotocopiadora fuera de uso. Para la aplicación solo es necesario 4 de los 6 hilos disponibles, dejando sin uso los que corresponden a los punto medios de cada una de las bobinas. Haciendo uso de un tester se midió resistencia y se reconoció cuales eran los extremos de cada bobina y los puntos medios. Los pasos del motor son de  $1,8^\circ$ , es decir, que son necesarios 200 para dar una revolución completa. En nuestro caso contamos con disponibilidad de dos tipos de motores distintos: dos de 1,45A y otro de 1,2A. Dicha información se pudo extraer de la hoja del fabricante y de lo señalado en los mismos motores. Estos datos son necesarios para la calibración de los drivers y trabajar sin dañar los motores.

La idea de utilizar estos motores es que pueden ser controlados a lazo abierto con precisión mediante el uso de un driver. Al saber el ángulo por paso al mover el eje podemos saber en donde se encuentra, la desventaja que tendríamos es que son motores con un movimiento más lento. Si quisieramos hacer uso de un motor distinto sería necesario la implementación de sensores de posición para conocer la ubicación de los mismos.

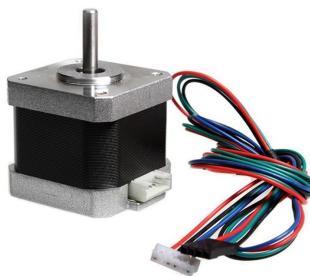


Figura 3: Motor PaP

### 3.3 Driver DRV8825

Para el control de los motores PaP se utilizan los driver DRV8825, con estos se simplifica el control de los mismos, el esquema de conexión se muestra en la figura 3. Desde nuestro arduino le enviaremos al driver con pulsos cuadrados al pin STEP la cantidad de pasos a dar y este driver se encargará de aplicar la tensión a las bobinas del motor para realizar el movimiento.

Se cuenta con 3 pines que dan la posibilidad de trabajar con micros pasos que nos darían movimientos menores a los  $1,8^\circ$  del motor, estos pines se dejarán sin conexión dado que se decidió trabajar con pasos completos. La dirección de giro estará dada por un alto o bajo en el pin de DIR. A su vez el driver tiene que estar ,por un lado, alimentado por una tensión de 5V y ,por el otro, con una alimentación que será la que energice al motor, en nuestro caso es una tensión de 12V. Luego, en el pin de ENABLE cuando se pone en alto se deshabilita la tensión de los motores.

Por otro lado, el driver cuenta con un potenciómetro el cual se utiliza para regular la corriente máxima que circulará por las bobinas del motor, esto debe ser ajustado para no dañar el motor con

sobrecorriente. La corriente máxima soportada por nuestros motores PaP son de 1,2A y 1,45A. Utilizando la fórmula extraída del datasheet calculamos la Vref a la cual debe ser ajustado.

$$I_{max} = \frac{V_{ref}}{5 * R_{sense}}$$

$$V_{ref} = 5 * R_{sense} * I_{max}$$

A su vez, cuando se utiliza el modo full step hay que utilizar el 70% de la corriente máxima con lo cual para nuestros motores nos da un voltaje de referencia de:

$$V_{ref} = 5 * 0,1\Omega * 1,2A * 0,7 = 0,42V$$

$$V_{ref} = 5 * 0,1\Omega * 1,45A * 0,7 = 0,51V$$

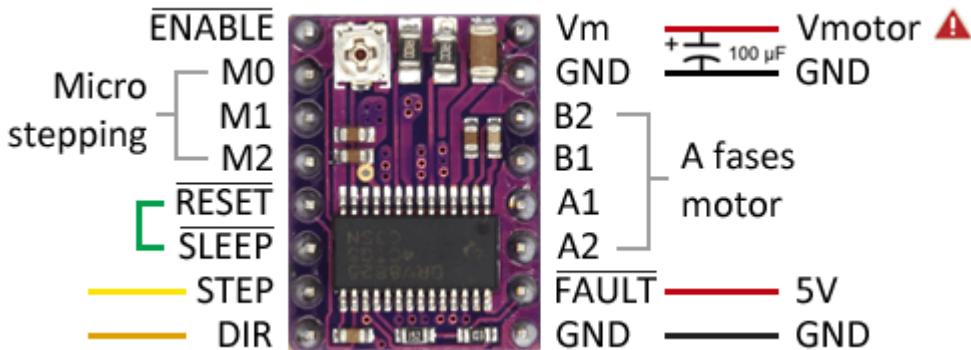


Figura 4: Conexión DRV8825

## 4. Funcionamiento general

El sistema está compuesto por 4 microcontroladores de los cuales 3 son esclavos y 1 es el maestro. Mediante la terminal que se comunica vía UART con el maestro se le pueden ingresar parámetros como pasos, velocidades de operación en carga y en vacío. La tarea definida para pick and place hace uso de una cantidad de pasos específicos en cada motor, estos mismos están precargados en nuestro código pero igual existe la posibilidad de modificarlos. Como no se cuentan con sensores de posición para saber la ubicación angular de los motores se da por asumido que el robot siempre comienza en la posición B. La secuencia a seguir por el robot es la que se observa en el siguiente diagrama de flujo. La tarea continuará sin interrupción hasta que se accione el botón físico de parada.

### 4.1 Diagrama de flujo

En el diagrama de flujo de la figura 5 y 6 se puede observar el funcionamiento secuencial de la tarea. Mediante el botón físico se da inicio al programa. Por consola se notifican los parámetros pre cargados, los cuales se pueden modificar o utilizarlos dando el comando de inicio de programa. Con la

confirmación por consola se ejecutará el programa que dará inicio a la tarea de pick and place la cual se compone de los siguientes pasos:

1-Mover en simultáneo motores eje 1 y 2 de posición B a A.

2-Mover motor eje 3 de posición arriba a abajo.

3-Cerrar efecto.

4-Mover motor eje 3 de posición abajo a arriba.

5-Mover en simultáneo motores eje 1 y 2 de posición A a B.

6-Mover motor eje 3 de posición arriba a abajo.

7-Abrir efecto.

8-Mover motor eje 3 de posición abajo a arriba.

9- Volver a 1.



Figura 5: Diagrama de flujo inicio de programa

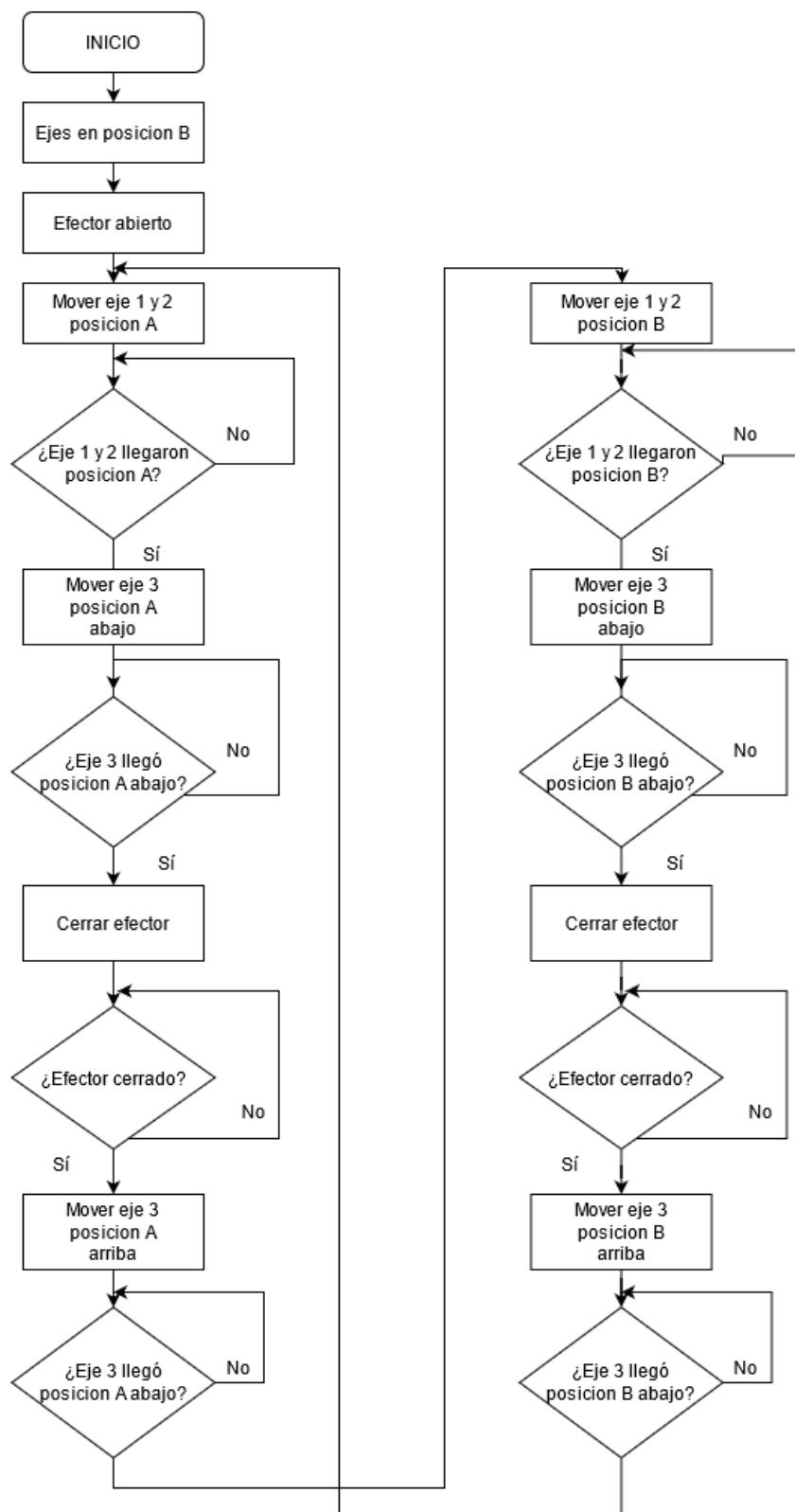


Figura 6: Diagrama de flujo ejecución de tarea secuencial

## 5. Programación

### 5.1 Información simulación de Matlab.

Haciendo uso del Toolbox de Peter Corke se simuló un brazo SCARA modelo ABB IRB9 10 SC. Se planteó que las coordenadas cartesianas en metros de los puntos A y B iban a ser respectivamente (-0.2m,-0.45m, 0.12m) y (0.36m, 0.31m, 0.12m). El origen del sistema de coordenadas se encuentra en la base del robot y cumple las condiciones de ortonormalidad y sistema dextrógiro.

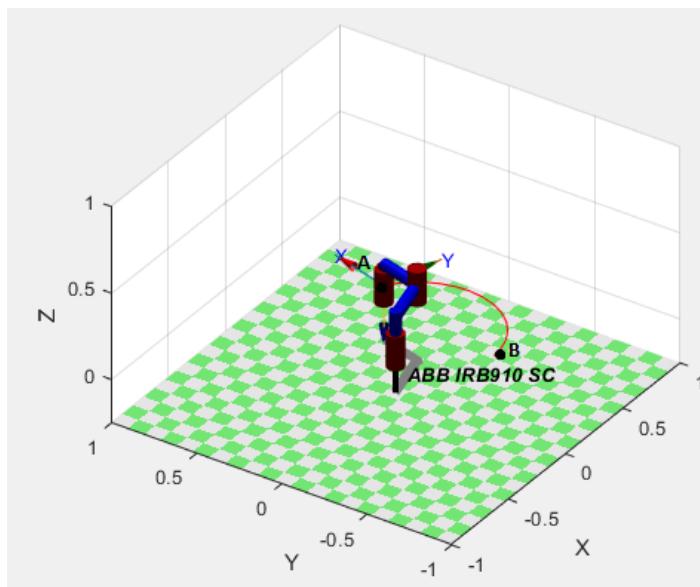


Figura 7: Modelo robot SCARA entorno Matlab

Para tener en cuenta, el efecto final cuando se encuentra totalmente arriba está a una distancia de 0.195m del suelo. Es decir que para la tarea de pick and place tiene que descender 0.075m

Para la rutina considerando que el brazo parte del punto B necesitan moverse los siguientes ángulos y respectivos pasos en el motor PaP para poder alcanzar la posición A.

Eje 1	102.65°	57 pasos
Eje 2	117°	65 pasos
Eje 3	28.8°	16 pasos

### 5.2 Main Maestro.

El Maestro se encuentra cargado con un programa que cuando inicia ejecuta funciones que configura la comunicación UART, la comunicación SPI que realiza con los 3 esclavos y los pines de entrada, salida y los registros de interrupción.

La comunicación UART está configurada con un baud rate de 19200. Mediante interrupción de recepción cuando ingresan por este medio el carácter ":" se comenzarán a guardar los siguientes datos hasta que llegue el comando "\r". Este carácter lo que hace es activar un flagComandoUART el cual indica que hay que revisar un comando ingresado por UART y cuando el programa realice polling sobre la función InterpretaUART() interpreta los comandos ingresados y modifica las variables relacionadas al movimiento.

La comunicación SPI es por conexión independiente. Mediante dos funciones, SPI\_configSPIMaster() y SPI\_MasterInit() se configuran los registros, pines que serán utilizados y se inicializa la comunicación que sucede a 1 MHz, polaridad 0 y fase con flanco de subida.

El maestro es quien en todo momento almacena los parámetros que corresponden a los movimientos de los ejes y los utiliza para enviarlos como consignas a los esclavos.

Para dar inicio con el programa hay que hacer uso de un pulsador físico, esto se considera el botón de arranque de la máquina. El programa tiene precargados los parámetros correspondientes a la cantidad de pasos y velocidad de cada uno de los ejes para realizar la tarea de pick and place, si se desean utilizar estos se ingresa por consola el comando ":0\r" y la ejecución de movimientos comenzará. Pero existe la posibilidad de modificar estos parámetros ingresando una trama con la siguiente forma.

**:NAXXX/r**

Todo mensaje tiene que comenzar con el carácter ":".

El valor de N hace referencia al esclavo que corresponde el parámetro a ingresar. Las opciones son 1, 2 o 3. Cualquier otro valor nos dará un mensaje de que el esclavo determinado no es correcto.

La A corresponde al parámetro en sí que se quiere modificar los cuales son:

V	Velocidad en vacío del motor, el valor que se ingresa es la duración del semiperiodo en ms.
C	Velocidad en carga del motor, el valor que se ingresa es la duración del semiperiodo en ms.
P	Cantidad de pasos que tiene que dar el motor.

La X representa el número que será guardado en la variable del parámetro, se puede ingresar valores entre 1 y 999.

Cuando se presiona enter ("\r"), la función interpreta UART se encarga de analizar el mensaje y modificar la variable deseada, en caso de que algún carácter sea incorrecto un mensaje por consola lo notifica. Ejemplo de comandos serían: ":1P25\r", ":2V50\r", ":3C35\r".

Se pueden modificar los parámetros tantas veces como se quiera una vez terminado esto para continuar con la ejecución se debe ingresar el comando ":0\r".

A partir de ahora el programa entra en un polling el cual primero revisa si hay algún mensaje por UART dado que durante la ejecución del programa se pueden enviar comandos para volver a modificar

parámetros. En caso de que no haya comandos ingresados por UART se procede a ejecutar la función EjecutaTarea() la cual consiste en un ciclo de carga y descarga como se muestra en el diagrama de flujo de la figura 6.

Dentro de EjecutaTarea(), la lógica para realizar el movimiento se realiza mediante dos if que realizan el movimiento de B->A y de A->B, se hacen uso de distintos flags para controlar la posición en la que se encuentran los ejes y si están en movimiento o lo han concluido.

La función que se encarga de enviar los mensajes a los esclavos es ComandoEje(eje, parámetro, valor) la cual carga a los respectivos ejes los distintos parámetros que son velocidad en carga y vacío, dirección, pasos, abrir y cerrar efecto. Primero se setean los parámetros correspondientes al movimiento que tiene que hacer el eje y luego mediante un comando indicado como ComandoEje(eje,R,1) se le da la orden al esclavo de efectuar el movimiento. Para el envío de estas instrucciones vía SPI, se hace uso de dos funciones, primero se interpreta a qué esclavo se quiere comunicar y mediante SPI\_Eje\_tx(datos), la cual pone en bajo el pin de SS correspondiente y, a su vez, hace uso de la función SPI\_Maestro\_tx(datos) que realiza el envío del comando. La función queda en un bucle a la espera de que el eje correspondiente avise mediante una interrupción externa la recepción del dato para poder continuar con la rutina.

Durante el ciclo de movimiento si se ingresa un comando que comience con :0 y cualquier letra distinta a I, como por ejemplo “:0t\r”, el intérprete reconoce que se desean modificar parámetros. Recién cuando se termine el movimiento y ha vuelto a la posición de origen B, ahí la tarea se detiene y se pueden modificar los parámetros. Una vez finalizada la modificación, ingresando el comando “:0I\r” nuevamente se reinicia la tarea y el movimiento continúa.

### 5.3 Main esclavos.

El esclavo al igual que el maestro hace uso de funciones para configurar la comunicación UART, SPI, las interrupciones y los pines de propósito general. En este caso, la UART, no tiene un intérprete de comandos dado que no espera que se ingresen valores. Solamente se utiliza con la finalidad de mostrar mensajes del proceso por consola (o terminal). Esto fue pensado como medio de control para verificar el correcto funcionamiento del programa en cada esclavo durante las pruebas, y posteriormente, como posible señal de fallo en caso de avería.

Cada esclavo cuenta con variables internas que son pasos, dirección y tiempo. Mediante un intérprete de comando SPI que recibe tramas con la misma forma que las utilizadas para ingresar por comunicación UART, mencionada previamente. El intérprete analiza el mensaje que llega y modifica la variable correspondiente. Para el caso del eje 3 el intérprete también puede reconocer un parámetro adicional el cual es del efecto abierto o cerrado, esto lo que modifica es la salida de un pin que, mediante un LED, indica cuando el efecto está cerrado o abierto.

Cuando el intérprete SPI reconoce que llega la orden de realizar el movimiento el esclavo hace uso de la función control\_pap(pasos, dirección, tiempo) la cual hace uso de 2 timers del microcontrolador, uno de 8 bits y otro de 16 bits.

El timer de 16 bits se configura en modo CTC OCR1A, el preescaler se calcula en función del tiempo ingresado. Este timer generará pulso que llegarán hasta el pin de dirección del DRV8825, cada pulso será

un paso y la velocidad de estos estará dada por la duración de los pulsos hecho en función del parámetro ingresado de tiempo. El timer de 8 bit está cableado con el timer de 16 bits y está configurado para contar los pulsos generados, cuando se llega a la cantidad deseada por comparación se genera una interrupción que desactiva el timer generador de pulsos y envía mediante la función TareaEnd() una interrupción al maestro para dar aviso de que la tarea se ha finalizado.

## 6. Etapas de montaje y ensayos realizados

### 6.1 Ensayos preliminares en Proteus.

Como primera etapa para verificación del correcto funcionamiento del programa se realizó una simulación en proteus montando el esquema que se puede observar en la figura 8. Mediante el uso de osciloscopio se pudo comprobar la cantidad de pulsos y la frecuencia de los mismos para asegurar el correcto funcionamiento. Las terminales virtuales fueron implementadas para verificar la comunicación SPI y UART y los leds para ver el comportamiento de las salidas relacionadas a dirección del movimiento y estado del efector final.

Gracias a esta simulación es que se pudo ir depurando el programa y arreglando todos los errores detectados antes de implementar el sistema de manera física.

Si bien el software de simulación no necesita de la definición de determinadas estrategias y/o componentes como resistencias pull-up o pull-down, así como también de las resistencias en serie para la conexión de los leds; en el circuito físico real se realizó la incorporación de los mismos para garantizar el correcto funcionamiento del sistema.

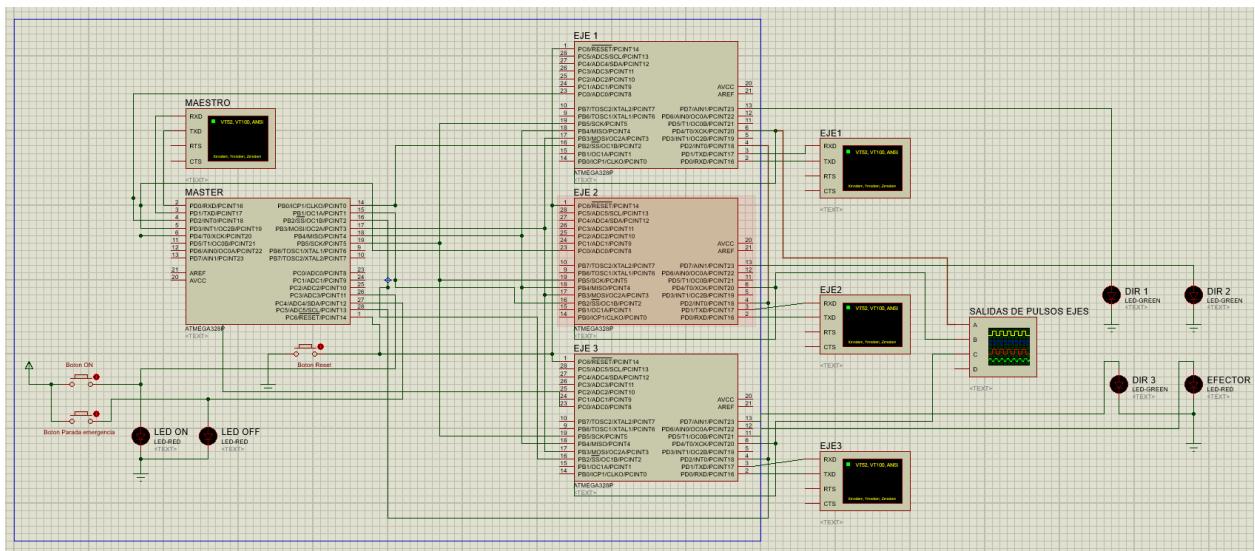


Figura 8: Simulación Proteus

## 6.2 Armado físico del circuito.

Para el armado físico del circuito la conexión entre microcontroladores es como la mostrada en la figura 8 al cual se le suma el esquema de conexión de la figura 4. La alimentación utilizada para este sistema fue dada por una fuente que proveía 5V y 12V. Las salidas del timer de los esclavos junto con las de dirección fueron conectados a los DRV8825. Para la parada de emergencia en el sistema fisico tambien se cableo el pin de ENABLE del DRV, cuando se activa la parada a este pin le llega una señal de alto y se bloquea la salida de tensión a las bobinas de los motores deteniendolos. Como se puede ver en la figura 10 a los pulsadores se les incorporó resistencias de 10Kohm para pull-down, pull-up y, en el caso de los leds, una resistencia de 1Kohm para no dañarlos.



Figura 9:Sistema físico completo

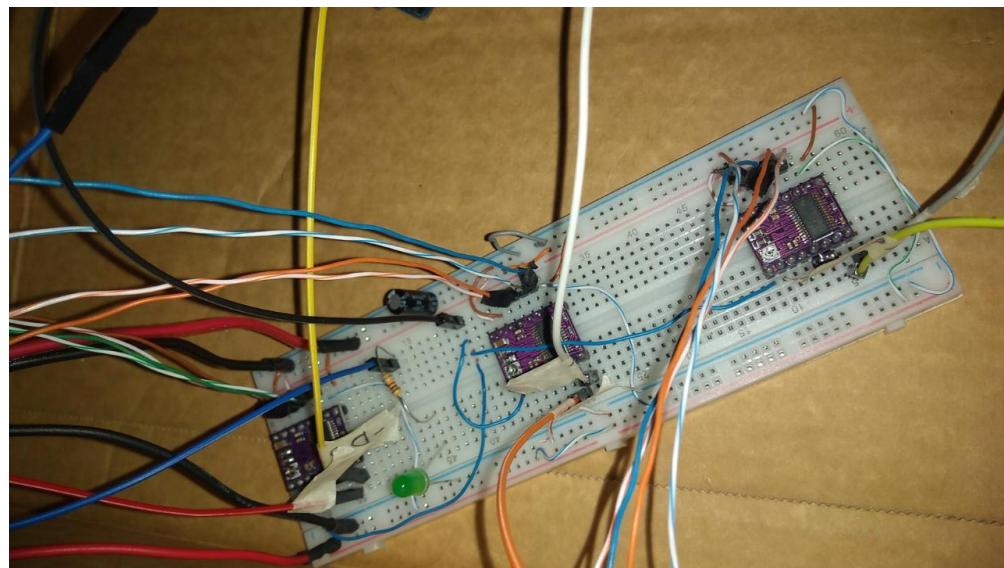


Figura 10: Sistema físico parte DRV8825

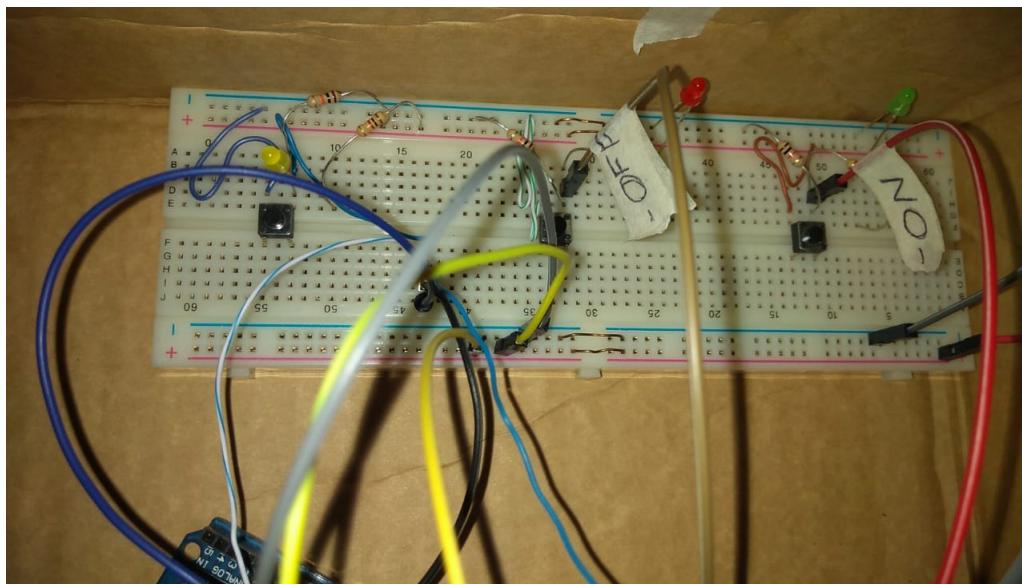


Figura 11: Sistema físico parte pulsadores

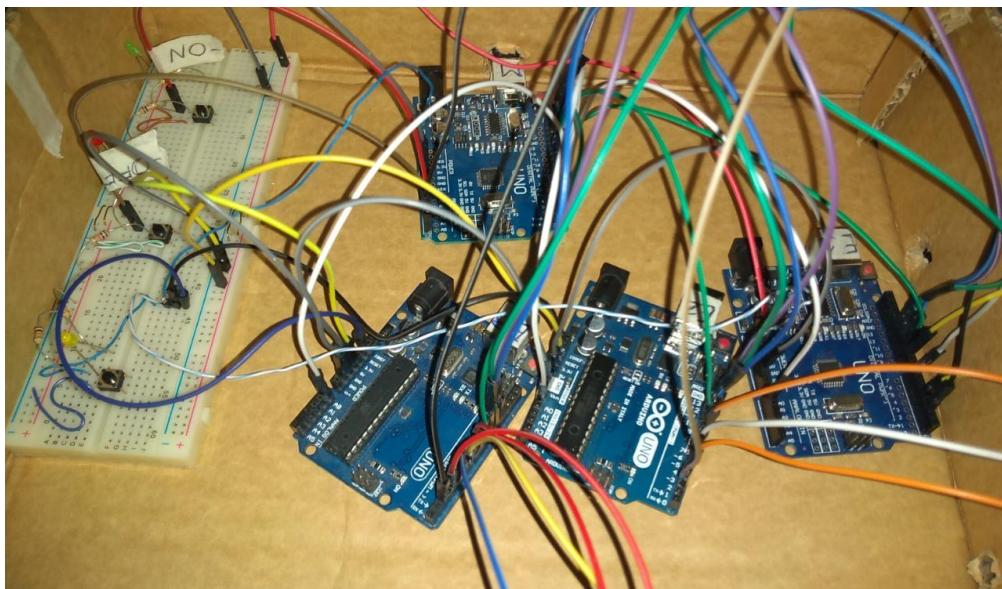


Figura 12: Sistema físico parte maestro-esclavo

## 7. Resultados, especificaciones finales

En primer instancia cuando se montó el sistema físico no funcionó la comunicación SPI entre maestro y esclavos. Para encontrar el problema primero se revisaron todas las conexiones, se hizo un intercambio de quien era el microcontrolador maestro pero en todos los casos la comunicación no sucedía. Hicimos pruebas haciendo un pequeño código con el IDE de Arduino y la comunicación SPI funcionaba. Concluimos que el problema estaba en el código aunque en la simulación andaba sin ningún problema, así que consultando distintas estrategias implementamos un cambio en la función SPI, modificamos cuando se inicializan los pines y donde se escribían los registros de la comunicación. Nada de eso resolvió el problema. En un principio los pines del maestro usados para SS de los esclavos eran PC0, PC1 y PC2. Decidimos cambiar de lugar, además de los respectivos cambios de código, estos pines a PB0, PB1 y PB2, esto resolvió el problema sin tener una explicación de qué fue lo que sucedió.

Luego, con la comunicación SPI establecida, se notó en ocasiones que los ejes de los motores al iniciar los microcontroladores, antes de dar inicio del programa con el botón de ON, generaban un pequeño movimiento de rotación en los ejes, en ocasiones muy pocos pasos sin motivos. Respecto a esto concluimos que puede ser debido a ruido presente en los pines de STEP y DIR en el momento de la inicialización. A fin de eliminar dichos problemas realizamos inicialmente desde el maestro el bloqueo de los motores mediante el pin de ENABLE hasta el momento de la activación de los ejes con el botón de ON.

Una vez solucionados estos problemas se ejecutó la tarea y pudimos ver como realizaba los movimientos esperados de pick and place. Se realizaron pruebas cambiando la cantidad de pasos, las velocidades en carga y en vacío. Si bien no teníamos un método para medir los pasos exactos, cuando asignamos 200 pasos, que son una revolución completa, se veía que esto sucedía correctamente. Aunque tampoco contamos con un método para medir la frecuencia de los pulsos generados es claramente visible que las velocidades se ven afectadas cuando cambiamos por consola los valores correspondientes.

Algo que sucedió en pocas ocasiones fue que el sistema se quedaba trabado (cuelgue del ciclo), esto puede ser debido a que en nuestra comunicación SPI cuando se perdía un mensaje a un eje, el maestro no puede reconocer que la tarea se realizó y no continua con la ejecución de comandos. Este problema, entre otros mencionados previamente, son en gran parte debidos a la calidad de las conexiones disponibles utilizadas y un posible punto de mejora a futuro.

## 8. Conclusiones

Con este proyecto pudimos implementar satisfactoriamente el control de movimiento de un robot SCARA a un bajo costo, la distribución maestro-esclavo tiene un gran beneficio al poder realizar múltiples tareas en simultáneo como es el movimiento de dos ejes al mismo tiempo. Todas las implementaciones propuestas pudieron ser aplicadas como se deseaba, a pesar de esto existen varias cosas por mejorar para que pueda escalar a un sistema robusto, confiable y de buen desempeño en todos los ambientes posibles.

En nuestro sistema, siempre se asume que se parte de la posición B lo cual en la realidad es una limitación. Como mejora, se plantea el uso de fines de carrera con los cuales se puede programar para que los motores giren en un sentido hasta encontrarlos, esto permitirá ubicar al brazo en una posición de homming.

La pérdida de mensajes en la comunicación es un problema que deja sin funcionamiento al robot, aunque no sea algo que suceda muy frecuentemente es algo que hay que evitar, por lo cual sería necesario el desarrollo de un sistema de control de mensajes para asegurar que no ha habido pérdida de información en la comunicación.

Otra mejora al sistema sería el reemplazo de los motores paso a paso por motores de CC de mayor velocidad que le daría mayor eficiencia a la tarea pero para esto también sería necesario la implementación de sensores de posición para conocer los movimientos de los motores dado que ya no tendríamos el control precisión de pasos.

En ocasiones las fallas del sistema se daban por algún problema de contactos entre los cables y los pines del microcontrolador y/o drivers, etc. La electrónica del sistema es uno de los claros puntos a mejorar. Con el uso de placas experimentales perforadas se podría eliminar cables y generar un sistema más compacto y confiable.

Por último, a nivel demostrativo, se ha simplificado la acción del efecto final a encender o apagar una salida (LED), donde consideramos que esto es instantáneo. Esta acción debe ser analizada al momento de la implementación, debido a los tiempos de demora del accionar del efecto final y/o de las señales de activación necesarias del mismo.

## 9. Referencias

[1] Datasheet DRV 8825 -

[https://www.ti.com/lit/ds/symlink/drv8825.pdf?ts=1621868889997&ref\\_url=https%253A%252F%252Fwww.google.com%252F](https://www.ti.com/lit/ds/symlink/drv8825.pdf?ts=1621868889997&ref_url=https%253A%252F%252Fwww.google.com%252F)

[2] Datasheet Atmega 328p -

[https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\\_Datasheet.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf)

[3] Microcontroladores y electrónica de potencia. Material de clases