Microcontroladores y

Electrónica de Potencia

# Trabajo integrador:

## *Control a lazo abierto de un robot SCARA*

Alumno: Olguin Nahuel
Legajo: 12297

# Indice

# 1. Introducción

Este proyecto busca extender el proyecto realizado en la asignatura "Automática y maquinas eléctricas", el cual consistía en controlar una máquina de corriente alterna sincrónica con excitación imanes permanentes (PMSM) para controlar la articulación hombro de un Robot tipo SCARA, como se muestra en la figura 1.


Figura 1 Control de articulación hombro de robot SCARA con motor síncrono

Para este proyecto se aplicará un control sobre los motores en cada una de las articulaciones (hombro, codo y muñeca) a lazo abierto para obtener resultados similares a la situacion real trabajando con equipos de menor potencia.

Para la articulacion hombro se hara uso de un motor PaP debido a que su principio de funcionamiento se basa en los principios teoricos de los motores síncronos y son muy prácticos a la hora de realizar control de posición y velocidad a lazo abierto. Para las articulaciones codo y muñeca se utilizaran servos ya que ofrecen mucha precisión para alcanzar una posición deseada del efector final

Para desarrollar el programa se utilizara el Entorno de desarrollo integrado (IDE) Microchip Studio que permite escribir, construir y depurar aplicaciones en lenguaje C

# 2. Esquema tecnológico

El sistema cuenta con los siguientes módulos:

- Controlador: Placa de microcontrolador Arduino Uno (microchip ATmega328P)
- Actuador (Hombro): Motor PaP unipolar
- Driver ULN2003A para motores PaP unipolar
- Actuador (Codo): Servomotor
- Actuador (Muñeca): Servomotor

Sistema externo:

- Comunicación con PC a través de Puerto Serie

En la figura 2 se muestra un diagrama de bloques que representa los módulos del sistema y los sistemas externos que interactúan con él

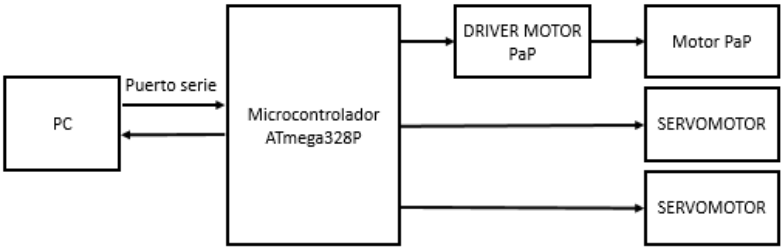



Figura 2 Módulos del sistema

# 3. Detalle de módulos

## 3.1 Arduino Uno (ATmega328P)

El Arduino Uno, como el observado en la figura 3, es una placa de microcontrolador de código abierto basado en el microchip ATmega328P y desarrollado por “Arduino.cc”. La placa está equipada con conjuntos de pines de E/S digitales y analógicas que pueden conectarse a varias placas de expansión y otros circuitos. La placa tiene 14 pines digitales y 6 pines analógicos


Figura 3 Placa Arduino Uno

En este proyecto se harán uso de los siguientes pines:

- 5V: pin que emite 5V
- GND: Pines de tierra.
- Serie/UART: pines 0 (RX) y 1 (TX). Se utiliza para recibir (RX) y transmitir (TX) datos en serie TTL.
- Pines de E/S digitales
- Pines de PWM (modulación de ancho de pulso)

## 3.2 Motor PaP

El motor paso a paso (Stepper) es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de girar una cantidad de grados (paso o medio paso) dependiendo de sus entradas de control.

Tiene un rotor multipolar de hierro y un estator devanado. El motor gira cuando el rotor es atraído por la bobina del estator energizada, por lo tanto se tendrá que gestionar la secuencia en la que excitamos las bobinas para que el motor vaya avanzando de forma continua, como se observa en la figura 4.

| PASO | BOBINA A | BOBINA B | BOBINA C | BOBINA D |
|---|---|---|---|---|
| 1 | ON | OFF | OFF | OFF |
| 2 | OFF | ON | OFF | OFF |
| 3 | OFF | OFF | ON | OFF |
| 4 | OFF | OFF | OFF | ON |

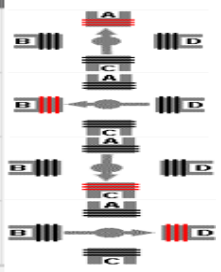


Figura 4 Secuencia de control de motor PaP

En este proyecto se hará uso de un motor paso a paso 28BYJ-48, el cual es unipolar y posee las características presentadas en la figura 5.

| | |
|---|---|
| Rated voltage : | 5VDC |
| Number of Phase | 4 |
| Speed Variation Ratio | 1/64 |
| Stride Angle | 5.625°/64 |
| Frequency | 100Hz |
| DC resistance | 50Ω±7%(25℃) |
| Idle In-traction Frequency | > 600Hz |
| Idle Out-traction Frequency | > 1000Hz |
| In-traction Torque | >34.3mN.m(120Hz) |
| Self-positioning Torque | >34.3mN.m |
| Friction torque | 600-1200 gf.cm |
| Pull in torque | 300 gf.cm |
| Insulated resistance | >10MΩ(500V) |
| Insulated electricity power | 600VAC/1mA/1s |
| Insulation grade | A |
| Rise in Temperature | <40K(120Hz) |
| Noise | <35dB(120Hz,No load,10cm) |
| Model | 28BYJ-48 – 5V |

Figura 5 Características motor 28BYJ-48

El motor es de 4 pasos (Steps), u 8 medios pasos (O half Steps) por vuelta y usa una reductora de 1/64, por lo que se necesitan dar "8 * 64 = 512" impulsos para completar un giro completo a medios pasos.

## 3.3 Driver motor PaP

Cada pin PWM del Arduino Uno puede proporcionar o recibir 20mA según las condiciones de funcionamiento recomendadas. Un máximo de 40mA es el valor que no debe excederse en ningún pin de E/S para evitar daños permanentes al microcontrolador. Como se observa en la figura 5, vemos que el motor PaP consume 55mA por lo que es necesario utilizar un driver adicional

Para controlar el motor PaP se hace un uso de un driver basado en un integrado del tipo ULN2003A, como el observado en la figura 6, el cual es un array de transistores Darlington que soporta hasta 500mA, este dispone de un conector para el motor y de unos pines (IN1 – IN4) para conectar al Arduino.

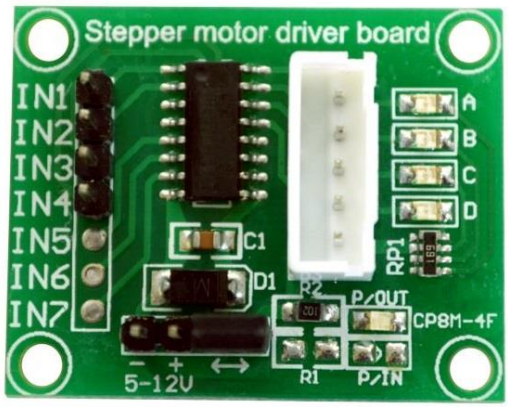



Figura 6 Driver ULN2003A

Básicamente consiste en un conjunto de transistores bipolares que se manejan como uno solo, lo que aumenta mucho la ganancia del transistor resultante y permite la conducción de grandes corrientes y tensiones.

## 3.4 Servomotores

Un servomotor consiste en un motor de corriente continua al que se ha añadido un sistema de control (tarjeta electrónica), un potenciómetro y un conjunto de engranajes. Tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición.

Los servomotores hacen uso de la modulación por ancho de pulsos (PWM), como la observada en la figura 7, para controlar la posición.

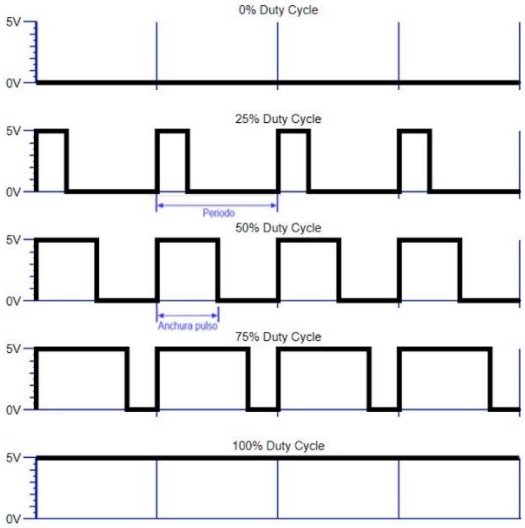



Figura 7 Control PWM

Para este proyecto se utilizara el servo “SG90”, cuyas características se observan en la figura 8:


Figura 8 Características servomotor SG90

Según el Data sheet, la posicion “-90°” (posicion minima) se alcanza con un pulso de “1 ms”, la posicion “0°” (posicion media) con “1.5 ms” y los “90°” (posicion maxima) se alcanza con un pulso de “2ms”. Es decir, que el rango de posisiones va desde pulsos de 1ms a 2ms como se observa en la figura 9

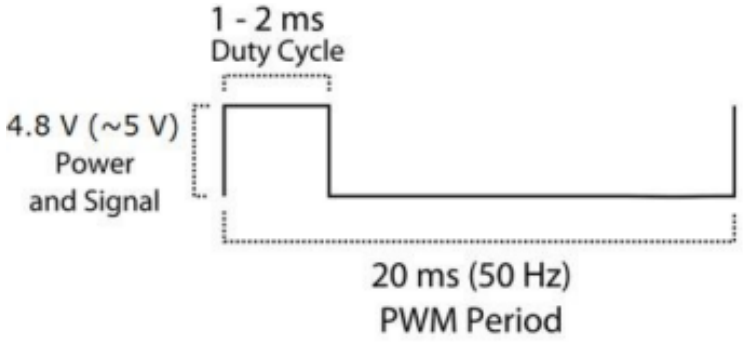



Figura 9 Control PWM de servomotor SG90

Sin embargo, en la práctica se observó que la posición mínima y máxima se obtenían con pulsos de "0.5 ms" y "2.5 ms" respectivamente

# 4. Funcionamiento general

La PC envía a través de la UART distintos comandos, cuya finalidad son las siguientes:

- Posicionar la articulación hombro
- Modificar la velocidad de la articulación hombro
- Posicionar la articulación codo
- Posicionar la articulación muñeca
- Conocer la posición del efector final aplicando cinemática directa
- Proponer consignas de posición del efector final en ejes cartesianos "xyz"
- Aplicar cinemática inversa para que el efector final pueda alcanzar la posición "xyz"

El controlador Arduino Uno procesa los comandos recibidos, los procesa y entrega señales eléctricas controladas al actuador que corresponda. Finalmente devuelve un mensaje a través de la UART de acuerdo a los resultados obtenidos

# 5. Programación

## 5.1 Comunicación entre PC y Controlador a través de la UART

Como hemos dicho anteriormente la PC se comunica con el controlador a través de la UART, por lo que primero se debe inicializarla. Para ello se hace uso del "Registro de Baud Rate" (Figura 10) que contiene el factor de división del clock

**UBRRn — USART Baud Rate register**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | UBRRn[11:8] | | | | UBRRnH |
| | UBRRn[7:0] | | | | | | | | UBRRnL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R/W | |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Figura 10 Registro de Baud Rate

Su valor se determina de la siguiente manera:

$$\text{UBRRn} = \frac{\text{F}_{\text{CPU}}}{16 * \text{Baudrate}} - 1 \qquad \text{Ec(1)}$$

Otro registro que se usa es el "Registro B de control y estado" (Figura 11). En donde se habilita la recepción, transmisión y la interrupción por recepción

**UCSRnB** **USART Control and Status Register B**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | RXCIEn | TXCIEn | UDRIEn | RXENn | TXENn | UCSZn2 | RXB8n | TXB8n | UCSRnB |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**RXCIEn** Habilita interrupción por RXCn.
**TXCIEn** Habilita interrupción por TXCn.
**UDRIEn** Habilita interrupción por UDREn.
**RXEn** Con '1' habilita la recepción
**TXEn** Con '1' habilita la transmisión
**UCSZn2**, junto con los bits UCSZn1:0 (del registro UCSRnC) determina la longitud del dato (5 a 9).
**RXB8n** y **TXB8n** son los novenos bits de Rx y Tx en caso de usar 9 bits de dato.

Figura 11 Registro de control y estado de la UART

La figura 12 muestra cómo se implementaron estos conceptos en el código

```
uart_init(9600,1);  //Inicia la UART, 9600 es la velocidad a transmitir, el 1 habilita la interrupcion

void uart_init(uint32_t bps,uint8_t RX_int){
    UBRR0 = F_CPU/16/bps-1; //se establece la velocidad de transmision
    UCSR0B= (1<<RXEN0)|(1<<TXEN0);  //prendemos la UART, habilitamos la recepcion y la transmision
    if (RX_int==1) UCSR0B|=(1<<RXCIE0); //habilitamos la interrupcion
}
```

Figura 12 Iniciación de la UART en el código

Para enviar o recibir caracteres a través de la UART se hace uso del "Registro de datos" (Figura 13) que cumple la funcion de transmitir o recibir. Esto es posible ya que, cuando se lee se accede al valor en el buffer de recepción RXB, y cuando se escribe se pasa el valor al buffer de transmisión TXB.

**UDRn** **USART Data register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | RXB[7:0] | | | | | | | | UDRn (Read) |
| | TXB[7:0] | | | | | | | | UDRn (Write) |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

```
void putc0(char c){  //Funcion para transmitir caracter por UART
    while((UCSR0A&(1<<UDRE0))==0);  //esperamos a que el buffer de transmision este vacio para continuar (si UDRE0=1 avanzamos)
    UDR0=c;
}

char getc0(void){  //Funcion para recibir caracter por UART
    char c;
    while((UCSR0A&(1<<RXC0))==0); //esperamos a que RXC0 se haga 1 (flag de recepcion completa)
    c=UDR0;                       //se guarda en "c" lo que hay en "UDR0"
    return c;
}
```

Figura 13 Registro de datos y su implementación en el código

## 5.2 Configuración de entradas/salidas de propósito general (GPIO)

Para establecer cuales pines corresponden como salida y cuales como entradas se hace uso del "Registro de dirección de datos" (DDRX), en donde a los pines de salida se les asigna "1" y a las entradas "0". Por otro lado el "Registro de salida de datos" (PORTX) almacena lo que se muestra en cada uno de los pines que fueron asignados como salidas anteriormente. Estos registros y sus funciones se observan en la figura 14

**PORTB – The Port B Data Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x05 (0x25) | PORTB7 | PORTB6 | PORTB5 | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 | PORTB |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**DDRB – The Port B Data Direction Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x04 (0x24) | DDB7 | DDB6 | DDB5 | DDB4 | DDB3 | DDB2 | DDB1 | DDB0 | DDRB |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**PINB – The Port B Input Pins Address**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x03 (0x23) | PINB7 | PINB6 | PINB5 | PINB4 | PINB3 | PINB2 | PINB1 | PINB0 | PINB |
| Read/Write | R | R | R | R | R | R | R | R | |
| Initial Value | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | |

| DDxn | PORTxn | PUD (in MCUCR) | I/O | Pull-up | Comment |
|---|---|---|---|---|---|
| 0 | 0 | X | Input | No | Tri-state (Hi-Z) |
| 0 | 1 | 0 | Input | Yes | Pxn will source current if ext. pulled low. |
| 0 | 1 | 1 | Input | No | Tri-state (Hi-Z) |
| 1 | 0 | X | Output | No | Output low (sink) |
| 1 | 1 | X | Output | No | Output high (source) |

Figura 14 Registros de la GPIO y sus configuraciones

En la figura 15 se observa como se implementaron los registros de GPIO en el código

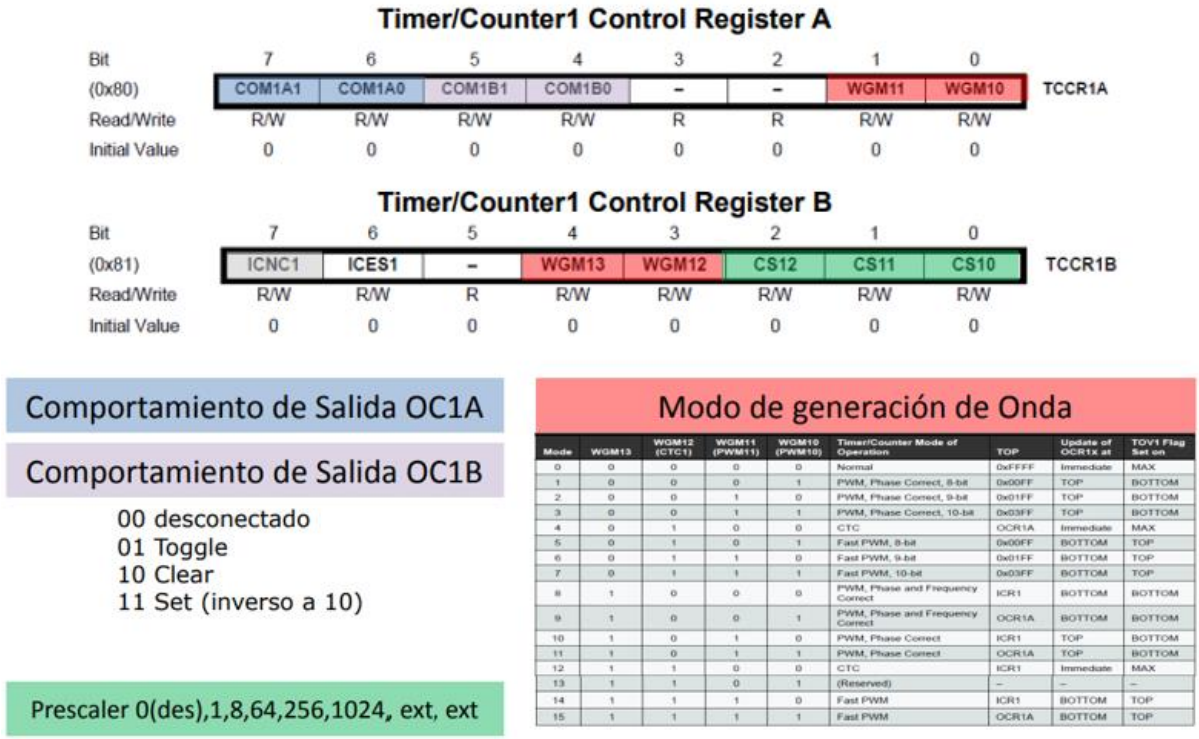
```
/*pin 1 controla codo
  pin 2 controla muñeca
  pin 3 controla la frecuencia de los motores Pap
  pin 4 controla los pulsos del Pap bipolar
  pin 5 controla el sentido de giro del motor Pap bipolar
*/
DDRB=(1<<DDB1)|(1<<DDB2)|(1<<DDB3)|(1<<DDB4)|(1<<DDB5);
//pin4,5,6,7 controlan los pulsos del motor Pap unipolar
DDRD=(1<<DDD7)|(1<<DDD6)|(1<<DDD5)|(1<<DDD4);

//Condiciones iniciales del motor PaP
velocidad=0.5; //velocidad inicial en RPM (valores entre 0.5 y 1.5), luego se multiplica x2 porq se usan pasos y no pasos medios
periodo=1000/velocidad*60/Pasos;  //v=60*f/N°pasos -> v=60/(N°pasos*periodo(en seg))
posicion=0; //posicion real del bipolar
posicionU=0; //posicion real del unipolar
Pin='7'; // comenzamos con el pin7 encendido
semipulso='1'; //comenzamos con la señal del unipolar en alto
PH=0;         //posicion consgna del hombro
PORTD=(1<<7); //activamos inicialmente el pin7
```

Figura 15 Implementación de los registros GPIO

## 5.3 Control de servos

Para controlar los servos se utilizó el Timer1, que permite generar pulsos con duty cycle variados y poder realizar un control PWM. El modo de generación de onda, el comportamiento de las salidas y el valor del prescaler son establecidos por los "Registros de control del Timer" observados en la figura 16




| Mode | WGM13 | WGM12 (CTC1) | WGM11 (PWM11) | WGM10 (PWM10) | Timer/Counter Mode of Operation | TOP | Update of OCR1x at | TOV1 Flag Set on |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Normal | 0xFFFF | Immediate | MAX |
| 1 | 0 | 0 | 0 | 1 | PWM, Phase Correct, 8-bit | 0x00FF | TOP | BOTTOM |
| 2 | 0 | 0 | 1 | 0 | PWM, Phase Correct, 9-bit | 0x01FF | TOP | BOTTOM |
| 3 | 0 | 0 | 1 | 1 | PWM, Phase Correct, 10-bit | 0x03FF | TOP | BOTTOM |
| 4 | 0 | 1 | 0 | 0 | CTC | OCR1A | Immediate | MAX |
| 5 | 0 | 1 | 0 | 1 | Fast PWM, 8-bit | 0x00FF | BOTTOM | TOP |
| 6 | 0 | 1 | 1 | 0 | Fast PWM, 9-bit | 0x01FF | BOTTOM | TOP |
| 7 | 0 | 1 | 1 | 1 | Fast PWM, 10-bit | 0x03FF | BOTTOM | TOP |
| 8 | 1 | 0 | 0 | 0 | PWM, Phase and Frequency Correct | ICR1 | BOTTOM | BOTTOM |
| 9 | 1 | 0 | 0 | 1 | PWM, Phase and Frequency Correct | OCR1A | BOTTOM | BOTTOM |
| 10 | 1 | 0 | 1 | 0 | PWM, Phase Correct | ICR1 | TOP | BOTTOM |
| 11 | 1 | 0 | 1 | 1 | PWM, Phase Correct | OCR1A | TOP | BOTTOM |
| 12 | 1 | 1 | 0 | 0 | CTC | ICR1 | Immediate | MAX |
| 13 | 1 | 1 | 0 | 1 | (Reserved) | – | – | – |
| 14 | 1 | 1 | 1 | 0 | Fast PWM | ICR1 | BOTTOM | TOP |
| 15 | 1 | 1 | 1 | 1 | Fast PWM | OCR1A | BOTTOM | TOP |

Figura 16 Registro de control del Timer1 y sus configuraciones

Para controlar los servos se hace uso del modo "Correct Phase PWM" (PWM-PC-ICR1), tomando como salidas OC1A y OC1B en modo "clear", con un prescaler x8. En este modo el Timer1 cuenta ascendente y descendente de cero a ICR1, conmutando OC1A y OC1B cuando el Timer1 iguala a OCR1A y OCR1B respectivamente (Como se observa en la figura 17)

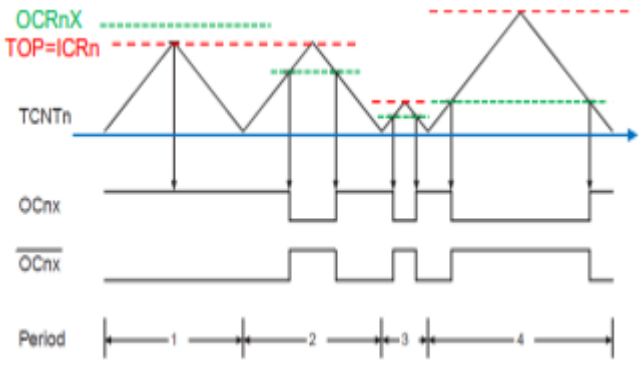



Figura 17 Modo PWM fase correcta

El valor del top del Timer es el registro ICR1, el cual dependerá de la frecuencia a la que trabaja el servo. Se elige el menor prescaler que se puede para obtener precisión en la posición; elegir un prescaler menor a "x8" provoca overflow en el registro ICR1 para los servos de 50 Hz (periodo de 20ms) que se utilizan

$$\text{ICR1} = \frac{\text{F}_{\text{CPU}}}{\text{Prescaler}} * \frac{\text{periodo}_{\text{servo}}}{2} \qquad \text{Ec}(2)$$

El duty cycle que se debe aplicar para alcanzar una determinada posición consigna “Pos”, depende de las características del servomotor

$$\text{DutyCycle} = \frac{(\text{DutyCycle}_{\max} - \text{DutyCycle}_{\min})}{(\text{Pos}_{\max} - \text{Pos}_{\min})}(\text{Pos} - \text{Pos}_0) + \text{DutyCycle}_{\min} \qquad \text{Ec (3)}$$

Para aplicar este duty cycle se debe modificar el registro OCR1x

$$\text{OCR1x} = \text{ICR1} * \frac{\text{DutyCycle}}{100} \qquad \text{Ec}(4)$$

En la figura 18 se observa la implementación del control PWM de los servomotores en el código

```
//Temporizador 1: Modo PWM-PC-ICR1, Salidas OC1A y OC1B "clear", Prescaler x8 (es el menor que podemos elegir sin overflow, ya que exigimos precision en la posicion)
TCCR1A = 0;  // ALWAYS clear this first
TCCR1B = 0;
TCNT1 = 0;
TIMSK1 = 0; // no interrupts needed
TCCR1A=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
TCCR1B=(1<<WGM13)|(0<<CS12)|(1<<CS11)|(0<<CS10);

ICR1=F_CPU/8*servo_periodo/1000/2;  // Recordamos q en PC el conteo es UP/DOWN entonces el tiempo que queremos es la mitad periodo/2
PC=0; //Queremos inicialmente el duty cycle para tener posicion 0°
Duty_Cycle_C=(Duty_Cycle_max-Duty_Cycle_min)*(PC-pos_servo_min)/(pos_servo_max-pos_servo_min)+Duty_Cycle_min;
OCR1A=ICR1*Duty_Cycle_C/100;
PM=0; //posicion inicial del efector final
Duty_Cycle_M=(Duty_Cycle_max-Duty_Cycle_min)*(PM-pos_servo_min)/(pos_servo_max-pos_servo_min)+Duty_Cycle_min;
OCR1B=ICR1*Duty_Cycle_M/100;
```

Figura 18 Implementación control PWM de servos

## 5.4 Control del motor PaP

Para controlar el motor PaP se hace uso del Timer2, cuyos registros de control y configuraciones se observan Figura 19

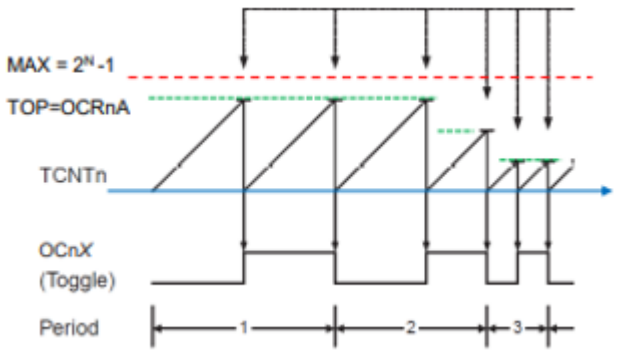



Figura 19 Modo CTC

Se utiliza el modo “Clear Timer on Compare match” (CTC), salida OCR2A "toggle" y prescaler x1024. En este modo el Timer2 cuenta desde 0 y cuando llega al valor OCR2A se resetea y conmuta el flag OC2A (Figura 20)

La velocidad de un motor paso a paso depende de la frecuencia de las señales eléctricas aplicadas y el número de pasos que presenta el motor

$$V = 60 * \frac{f}{Pasos} = 60 * \frac{1}{Pasos * periodo} \qquad Ec(5)$$

**TCCR2A – Timer/Counter Control Register A**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xB0) | COM2A1 | COM2A0 | COM2B1 | COM2B0 | – | – | WGM21 | WGM20 | TCCR2A |
| Read/Write | R/W | R/W | R/W | R/W | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**TCCR2B – Timer/Counter Control Register B**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xB1) | FOC2A | FOC2B | – | – | WGM22 | CS22 | CS21 | CS20 | TCCR2B |
| Read/Write | W | W | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

| Mode | WGM2 | WGM1 | WGM0 | Timer/Counter Mode of Operation | TOP | Update of OCRx at | TOV Flag Set on[1][2] |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Normal | 0xFF | Immediate | MAX |
| 1 | 0 | 0 | 1 | PWM, phase correct | 0xFF | TOP | BOTTOM |
| 2 | 0 | 1 | 0 | CTC | OCRA | Immediate | MAX |
| 3 | 0 | 1 | 1 | Fast PWM | 0xFF | BOTTOM | MAX |
| 4 | 1 | 0 | 0 | Reserved | – | – | – |
| 5 | 1 | 0 | 1 | PWM, phase correct | OCRA | TOP | BOTTOM |
| 6 | 1 | 1 | 0 | Reserved | – | – | – |
| 7 | 1 | 1 | 1 | Fast PWM | OCRA | BOTTOM | TOP |

| CS22 | CS21 | CS20 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source (Timer/Counter stopped). |
| 0 | 0 | 1 | $clk_{T2S}$/(no prescaling) |
| 0 | 1 | 0 | $clk_{T2S}$/8 (from prescaler) |
| 0 | 1 | 1 | $clk_{T2S}$/32 (from prescaler) |
| 1 | 0 | 0 | $clk_{T2S}$/64 (from prescaler) |
| 1 | 0 | 1 | $clk_{T2S}$/128 (from prescaler) |
| 1 | 1 | 0 | $clk_{T2S}$/256 (from prescaler) |
| 1 | 1 | 1 | $clk_{T2S}$/1024 (from prescaler) |

Figura 20 Registro de control del Timer2 y sus configuraciones

La velocidad máxima está limitada por el motor, la cual es 3 RPM y le corresponde un periodo de 10 ms

La velocidad del motor queda definida por el registro OCR2A que representa el top del contador

$$OCR2A = \frac{F_{CPU}}{prescaler} * periodo - 1 \qquad Ec(6)$$

El único prescaler que funciona es el x1024 debido a que el registro OCR2A es de 8 bits y un prescaler más pequeño produce overflow. El efecto de overflow limita la velocidad mínima a la que puede trabajar el motor, la cual es 1 RPM

En la Figura 21 se muestra la implementación del Timer2 en el código

```
//Temporizador 2: Modo CTC-OCR2A, Salida OCR2A "toggle", Prescaler x1024
///(no puedo usar prescaler menor, porq si elijo prescaler 256 -> el periodo maximo para no tener overflow en OCR1B de 8 bits es 8.2ms
//y el periodo minimo para no superar los 1.5 RPM son 10ms, por lo tanto es imposible utilizar un prescaler menor, sin embargo la imprecision solo se vera reflejada en la velocidad)
TCCR2A = 0;  // ALWAYS clear this first
TCCR2B = 0;
TCNT2 = 0;
TIMSK2 = 0; // no interrupts needed
TCCR2A=(1<<COM2A0)|(1<<WGM21);
TCCR2B=(1<<CS22)|(1<<CS21)|(1<<CS20);
OCR2A=F_CPU/1024/1000*periodo/2-1  ;  //Ciclos OCR2A=(F_CPU/valor del prescaler)* Periodo deseado(en seg)/2-1
```

Figura 21 Implementación Timer2

Para controlar la posición se hace uso de la interrupción por compare match de OCR2A mediante los registros TIMSK1 y TIFR1 (Figura 22)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x6F) | – | – | ICIE1 | – | – | OCIE1B | OCIE1A | TOIE1 | TIMSK1 |
| Read/Write | R | R | R/W | R | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x16 (0x36) | – | – | ICF1 | – | – | OCF1B | OCF1A | TOV1 | TIFR1 |
| Read/Write | R | R | R/W | R | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

```
TIFR2|=(1<<OCF1A); //Borra el flag de compare match antes de habilitarlo
TIMSK2=(1<<OCIE1A); //habilita la interrupcion de compare match
```

Figura 22 Registros de interrupción CTC

Esta interrupción permite controlar la secuencia de señales eléctricas aplicadas en el estator que controlan el motor PaP. En resumen el control de posición se realiza de la siguiente forma:

1. Compara la posición actual con la nueva posición consigna para definir el sentido de giro
2. Cada vez que se activa la interrupción del Timer2, ocurre lo siguiente: se apaga el pin actual, se enciende el siguiente pin de la secuencia y se le suma un paso a la variable que guarda la posición en tiempo real del motor
3. Cuando la posición real del motor supera a la posición consigna se dejan de atender a las interrupciones generadas por el Timer2 dejando al motor en esa última posición que se alcanza

## 5.5 Pulsador de fin de carrera y de parada de emergencia

Se implementaran dos pulsadores, uno que funcione como fin de carrera para poder llevar a cabo la maniobra de "homing" y otro para para detener el sistema en caso de emergencias. Para implementar ambos se utilizaran los registros de interrupción externa, se hará uso de la interrupción por flanco de subida, tanto en INT0 como en INT1. En la figura 24 se observa su implementación en el código (Figura 23)

**EICRA – External Interrupt Control Register A**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x69) | – | – | – | – | ISC11 | ISC10 | ISC01 | ISC00 | EICRA |
| Read/Write | R | R | R | R | R/W | R/W | R/W | R/W | |

ISCx1-ISCx0
00 Interrumpe por nivel bajo
01 Interrumpe por cualquier cambio
10 Interrumpe por flanco de bajada
11 Interrumpe por flanco de subida

**EIMSK – External Interrupt Mask Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1D (0x3D) | – | – | – | – | – | – | INT1 | INT0 | EIMSK |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |

**EIFR – External Interrupt Flag Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x1C (0x3C) | – | – | – | – | – | – | INTF1 | INTF0 | EIFR |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |

```
//Boton de parada de emergencia y final de carrera
EICRA=(1<<ISC11)|(1<<ISC10)|(1<<ISC01)|(1<<ISC00); //=0b00001111 interrumpe por flanco de subida de INT0 e INT1
EIMSK=(1<<INT1)|(1<<INT0); //habilita evento de interrupcion externa
EIFR=(1<<INTF1)|(1<<INTF0); //borra flags de eventos previos
```

*Figura 23 Registros de interrupción externa*

# 6 Etapas de montaje y ensayos realizados

Se realizaron diversos ensayos utilizando “Proteus 8”, de esta forma se realizó el control del correcto funcionamiento de cada una de las funciones que forman parte del proyecto facilitando mucho el montaje y los ensayos realizados con los componentes reales. La simulación se observa en la Figura 24

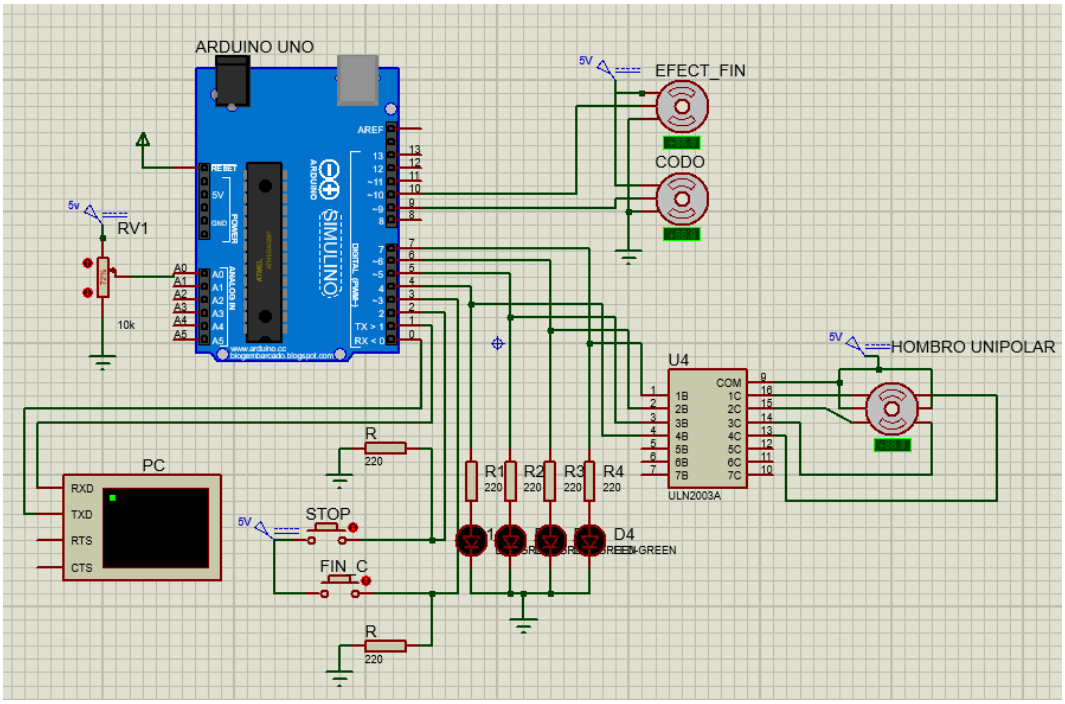



Figura 24 Simulación del sistema

En la figura 25 se muestra el montaje real del sistema

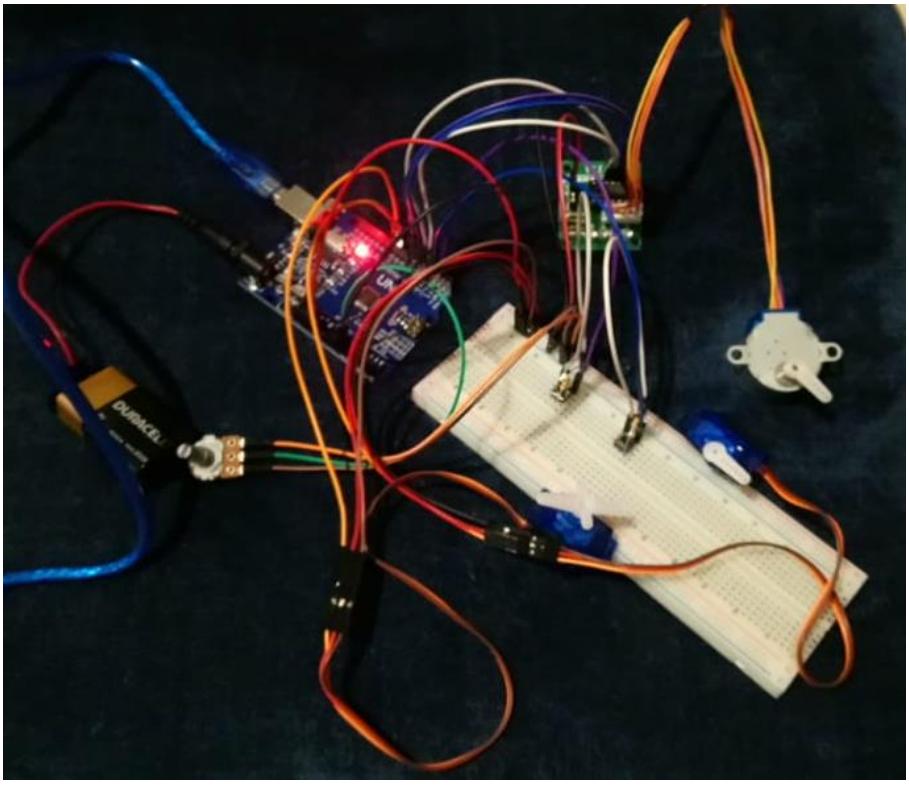


Figura 25 Montaje del sistema

# 7 Ensayo de ingeniería de producto

En este apartado se busca realizar la selección de motores y justificar las decisiones tomadas. Se considera un Robot SCARA modelo "UV-CERMA" cuyas dimensiones son las observadas en la Figura 26. Además se observa como se establece el sistema de ejes de cartesianos de referencia para posicionar el efector final

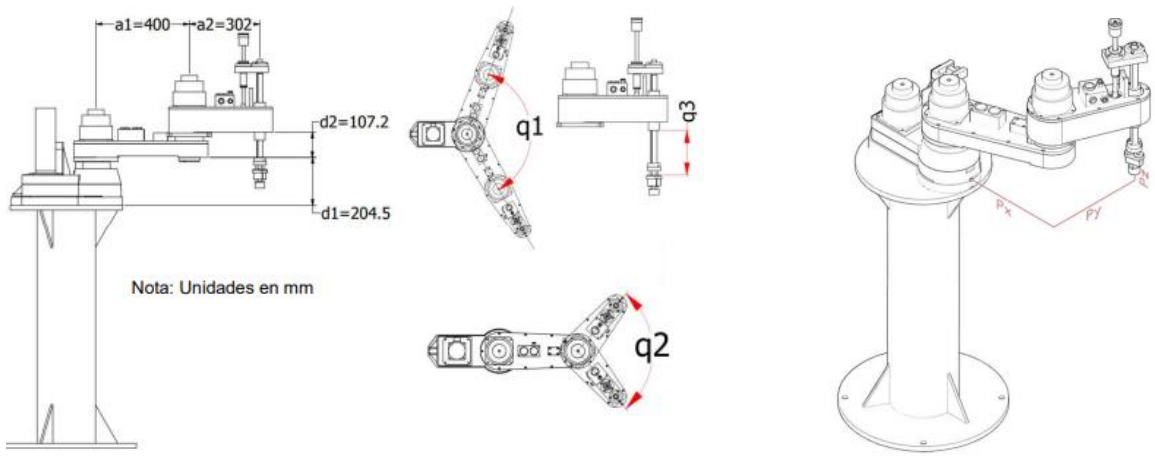



Figura 26 Dimensiones del Robot SCARA, movimientos en coordenadas angulares y cartesianas

Para determinar el torque que deben ofrecer los motores para un correcto funcionamiento, es necesario conocer la masa de cada uno de los eslabones correspondientes a este modelo de SCARA (Figura 27)

| | Eslabón 1 | Eslabón 2 | Eslabón 3 |
|---|---|---|---|
| Peso [Kg] | 22,1 | 16,5 | 0,6 |

*Figura 27 Masa de cada eslabón*

El par resistivo que ofrece el brazo, será el máximo cuando el brazo se encuentra completamente extendido (Se desprecia el peso de motores y otros componentes para simplificar el análisis). El par máximo que deberá ofrecer motor para obtener una velocidad máxima en el hombro de "α=3 RPM=0.31 rad/s²", está dado por:

$$T_{m1} < (J_1 + J_{21} + J_{31}).\alpha$$

Recordamos aspectos relacionados al momento de inercia de barras y el teorema de Steiner (Figura28)

$$I = I_{CM} + MD^2$$

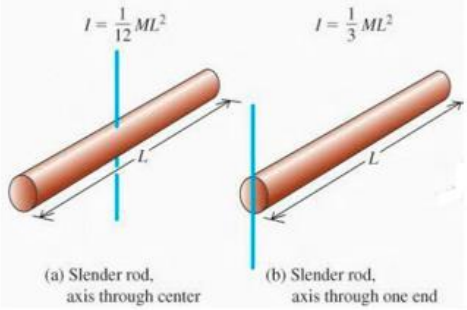



*Figura 28 Teorema de Steiner y momento de inercia en barras*

Donde los momentos de inercia están dados por:

$$J_1 = \frac{1}{3} m_1 a_1^2 = 1.18\ Kg.m^2$$

$$J_{21} = \frac{1}{12} m_2 a_2^2 + m_2.\left(a_1 + \frac{a_2}{2}\right)^2 = 8{,}25\ Kg.m^2$$

$$J_{31} = m_3.(a_1 + a_2)^2 = 0{,}29\ Kg.m^2$$

Por lo que el par máximo que debe entregar el motor está dado por:

$$T_{m1} > (J_1 + J_{21} + J_{31}).\alpha = 3\ Nm$$

Para el hombro se seleccionara un motor paso a paso "EMMS-ST 87 M" (Figura 29) que ofrece un par de 5,9 Nm

**Motores paso a paso EMMS-ST**

Hoja de datos

FESTO

| Datos técnicos generales | | | | |
|---|---|---|---|---|
| Tamaño | | 87-S | 87 M | 87-L |
| Motor | | | | |
| Tensión nominal | [V DC] | 48 | | |
| Corriente nominal | [A] | 9,5 | | |
| Velocidad de giro máxima | [1/min] | 2 130 | 550 | 430 |
| Momento de retención | [Nm] | 2,5 | 5,9 | 9,3 |
| Ángulo de los pasos | [°] | 1,8 ±5% | | |
| Resistencia de la bobina | [Ω] | 0,1 ±10% | 0,23 ±10% | 0,23 ±10% |
| Inductancia de la bobina | [mH] | 0,45 | 2,6 | 2,7 |
| Momento de inercia de salida | [kg cm²] | 1/1,07[1)] | 1,9/1,97[1)] | 3/3,07[1)] |
| Carga radial en el eje | [N] | 200 | | |
| Carga axial en el eje | [N] | 65 | | |
| Momento de inercia de la masa, rotor | [kgcm²] | 1 | 1,9 | 3 |
| | | | | |
| Freno | | | | |
| Tensión de funcionamiento | [V DC] | 24 ±10% | | |
| Potencia eléctrica | [W] | 11 | | |
| Momento de retención | [Nm] | 2 | | |
| Momento de inercia de la masa | [kgcm²] | 0,07 | | |
| Retardo de reacción | | 2 | 2 | 2 |
| Tiempo de separación | | 10 | 25 | 25 |

*Figura 29 Selección de motor para la articulación Hombro*

El par máximo que deberá ofrecer motor para obtener una velocidad máxima en el codo de "α=20 RPM=2 rad/s²", está dado por:

$$T_{m2} < (J_2 + J_{32}).\alpha$$

Donde los momentos de inercia son:

$$J_2 = \frac{1}{3} m_2 a_2^2 = 0.501\ Kg.m^2$$

$$J_{32} = m_3 . {a_2}^2 = 0{,}055\ Kg.m^2$$

Por lo que el par máximo que debe entregar el motor está dado por:

$$T_{m1} > (J_2 + J_{32}).\alpha = 1{,}11\ Nm$$

Para el codo se seleccionara un motor servo "SM060 R40B30_BD" (Figura 30) que ofrece un par de 1,27 Nm

**SM060 / BL060**

| Motor model 060 | | R20B30__BD | R20C30__BD | R20G30__BD | R40B30__BD | R40C30__BD | R40D30__BD | R40G30__BD | R60G30__BD |
|---|---|---|---|---|---|---|---|---|---|
| Voltage | V | 36 | 48 | 220 | 36 | 48 | 60 | 220 | 220 |
| Rated output power | W | | 200 | | | 400 | | | 600 |
| Rated torque | Nm | | 0,637 | | | 1,27 | | | 1,91 |
| Peak torque | | | 1,9 | | | 3,8 | | | 5,7 |
| Rated current | A | 7,5 | 6,5 | 1,9 | 13,8 | 10,25 | 8,5 | 2,8 | 3,7 |
| Peak current | | 31,5 | 27,3 | 6,8 | 42,8 | 31,8 | 26,4 | 9 | 12,4 |
| Constant torque | Nm/A | 0,085 | 0,098 | 0,33 | 0,092 | 0,12 | 0,15 | 0,45 | 0,52 |
| Rated speed | rpm | | | | 3000 | | | | |
| Max. Speed | | | | | 5000 | | | | |
| Motor poles | | | | | 8 | | | | |
| EMF coefficient | mV/rpm | 5,4 | 6,2 | 20 | 5,6 | 7 | 9,2 | 28,8 | 28 |
| Line inductance | mH | 0,9 | 1,14 | 11,79 | 0,44 | 0,67 | 1,3 | 12,54 | 9,97 |
| Line resistance | ohm | 0,33 | 0,35 | 3,2 | 0,16 | 0,27 | 0,38 | 2,75 | 2,35 |
| Moment of inertia | $kg{\cdot}m^2{\cdot}10^{-4}$ | | 0,189 | | | 0,342 | | | 0,419 |
| Weight | kg | | 1,1 | | | 1,7 | | | 1,9 |

*Figura 30 Selección motor para la articulación codo*

Se considera que el motor en la muñeca debe hacer girar una rueda dentada de "20mm" y que se desea alcanzar una aceleración de "α=5 RPM=0,52 rad/s²". El momento de inercia del disco se observa en la Figura 31

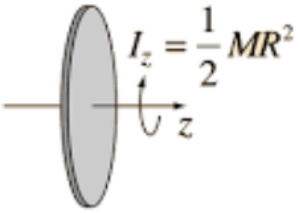



*Figura 31 Momento de inercia de un disco*

$$T_{m2} > I.\alpha = \left(\frac{1}{2}.0{,}6Kg.4e10^{-4}m^2\right).0{,}52\frac{rad}{s^2} = 6{,}24e10^{-5}N.m$$

Para la muñeca se seleccionara un motor servo de la figura 30, en este caso al no requerir de mucho torque seleccionaremos el más pequeño "SM060 R20B30_BD" que ofrece un par de 0,637 Nm

# 8 Conclusiones y mejoras

Como resultado del proyecto se obtuvo un prototipo funcional de control de las articulaciones de un robot SCARA a lazo abierto, presentando gran precisión en cuanto a la posición y la velocidad de las mismas. En conclusión, podemos decir que el proyecto fue exitoso

A continuación mencionare algunas de las mejoras que pueden implementarse en un futuro:

- Implementar un sensor de corriente para aplicar acciones de control que garanticen la seguridad del sistema en caso de sobrecorrientes
- Implementar una memoria que almacene una secuencia de posiciones en bucle que el robot deba seguir
- Mejorar aspectos del código (como por ejemplo recibir las consignas de posición 'xyz' con un solo comando)

# 9 Referencias


- Ing. Iriarte, Eduardo. Apuntes catedra "Microcontroladores y Electrónica de potencia". UNCuyo 2021
- Datasheet Atmega328p https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.
- Datasheet motor PaP 28BYJ-48 https://www.mouser.com/datasheet/2/758/stepd-01-data-sheet-1143075.pdf
- Datasheet_servomotor(SG90) http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf
- Catalogo_motor_PaP http://www.catalogo.sitasa.com/familias/neumatica/festo/25_motores_servomotores/25_2.pdf
- Catalogo servomotores https://docs.gestionaweb.cat/0824/catalogo-servomotores-estandar.pdf
- Información motor PaP https://www.prometec.net/motor-28byj-48/

# 10. Anexos

## 9.1 Cinemática directa

En la Figura 25 vemos que las dimensiones del robot SCARA son las siguientes: el eslabón controlado por el hombro mide 30cm y el eslabón del codo mide 25cm

Como los eslabones controlados por el hombro y el codo se desplazan en el mismo plano, se llamara a este plano "xy" y se proyectaran los eslabones verticalmente (Figura 31)

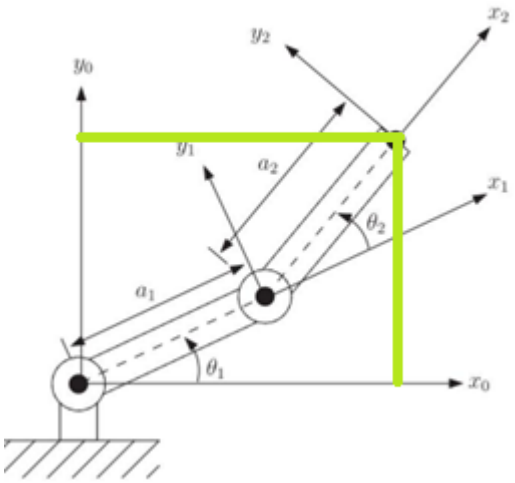



Figura 32 Proyección vertical de las articulaciones hombro y codo

Se observa que la posición del efector final en el plano "xy" es la siguiente:

$$x = a_1.\cos(q_1) + a_2.\cos(q_1 + q_2) \quad \text{Ec(7)}$$

$$y = a_1.\text{sen}(q_1) + a_2.\text{sen}(q_1 + q_2) \quad \text{Ec(8)}$$

La articulación controlada por la muñeca se mueve en el eje "z" de forma ortogonal al plano "xy". A modo de simplificar su funcionamiento, se considera que la salida del conjunto motor-caja reductora posee un radio de "r=3cm" y trabaja sobre un sistema de correa dentada (Figura 27)

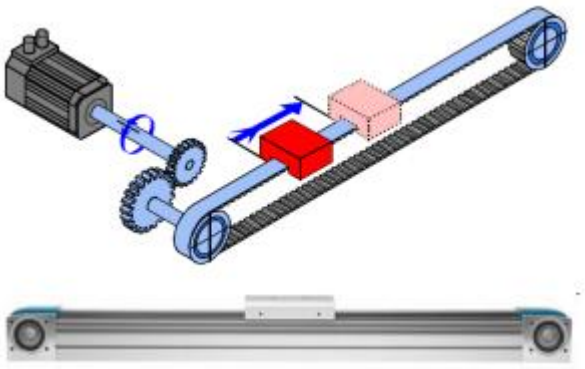


Figura 33 Posicionamiento del efector final

Entonces, la posición en el eje z está dada por la siguiente ecuacion:

$$z = r.q_3 \quad \text{Ec(9)}$$

## 9.2 Cinemática inversa

A partir de consignas de posición del efector final se determina cuanto debe rotar la articulación codo a partir del método observado en la figura 28

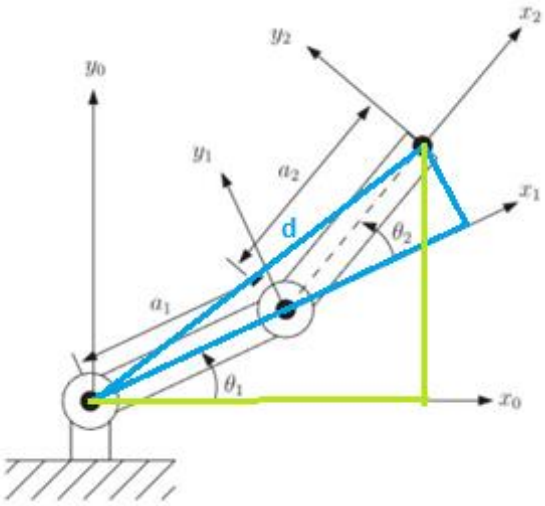



Figura 34 Método de obtención de la posición del codo

$$d^2 = x^2 + y^2 = \left(a_1 + a_2.\cos(q_2)\right)^2 + \left(a_2.\mathrm{sen}(q_2)\right)^2$$

$$x^2 + y^2 = a_1^2 + 2.a_1.a_2.\cos(q_2) + \left(a_2.\cos(q_2)\right)^2 + \left(a_2.\mathrm{sen}(q_2)\right)^2$$

$$x^2 + y^2 = a_1^2 + 2.a_1.a_2.\cos(q_2) + a_2.(\cos(q_2)^2 + \mathrm{sen}(q_2)^2)$$

$$x^2 + y^2 = a_1^2 + 2.a_1.a_2.\cos(q_2) + a_2^2$$

$$q_2 = \mathrm{acos}\left(\frac{x^2 + y^2 - a_1^2 - a_2^2}{2.a_1.a_2}\right) \qquad \text{Ec(10)}$$

Ahora se determina cuanto debe rotar la articulación hombro a partir del método observado en la figura 29

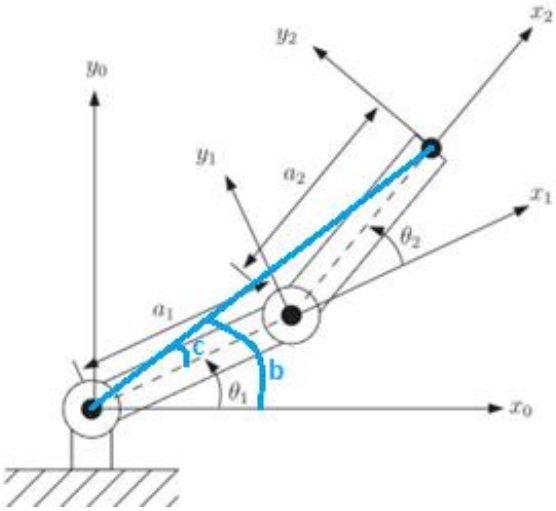



Figura 35 Método de obtención de la posición del hombro

$$q_1 = b - c$$

$$b = \operatorname{atan}\left(\frac{y}{x}\right)$$

$$c = \operatorname{atan}\left(\frac{a_2.\operatorname{sen}(q_2)}{a_1 + a_2.\cos(q_2)}\right)$$

$$q_1 = \operatorname{atan}\left(\frac{y}{x}\right) - \operatorname{atan}\left(\frac{a_2.\operatorname{sen}(q_2)}{a_1 + a_2.\cos(q_2)}\right) \quad \text{Ec}(11)$$

Por último la posición de la articulación de la muñeca se obtiene al despejar la ecuación (9):

$$q_3 = \frac{z}{r} \quad \text{Ec}(12)$$