

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Módulo de alimentación y control de movimiento de servo
motores para pirata animatrónico**

Trabajo de graduación presentado por Arturo Ernesto García Cerezo
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2018

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Módulo de alimentación y control de movimiento de servo
motores para pirata animatrónico**

Trabajo de graduación presentado por Arturo Ernesto García Cerezo
para optar al grado académico de Licenciado en Ingeniería Electrónica

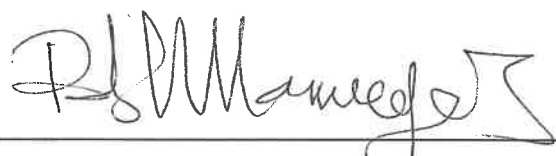
Guatemala,

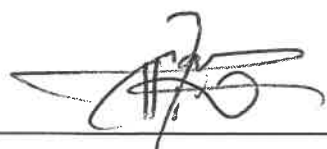
2018

Vo.Bo.:

(f) 
Ing. Pablo Mazariegos

Tribunal Examinador:

(f) 
Ing. Pablo Mazariegos

(f) 
MSc. Carlos Esquit

(f) 
MSc. Miguel Zea

Fecha de aprobación: Guatemala, 5 de diciembre de 2018.

Quiero agradecer primero que nada a mi familia, sin la cual no hubiera sido posible terminar este trabajo. A mi madre Anabella Cerezo, a mi padre Arturo García y a mi hermana Karla Lopez Cerezo. Además, quisiera agradecer a mis compañeros de carrera, ya que sin su apoyo no hubiera logrado llegar hasta esta instancia.

Además, quiero agradecer a mi director de proyecto Luis Pedro Montenegro, que fue una fuente de constante apoyo a lo largo del proyecto. A mi asesor Pablo Daniel Mazariegos, por siempre estar disponible y brindar todo el apoyo necesario. A mis compañeros de megaproyecto y principalmente a mi compañero Jose Francisco Zelada, que brindó apoyo desde la primera etapa de este proyecto y sin el cual no hubiera sido posible completar esta investigación.

Prefacio	V
Lista de figuras	XII
Lista de cuadros	XIII
Resumen	XV
1. Introducción	1
2. Antecedentes	3
3. Justificación	5
4. Objetivos	7
4.1. Objetivo general	7
4.2. Objetivos específicos	7
5. Alcance	9
6. Marco teórico	11
6.1. Comunicación serial	11
6.1.1. Bits de datos	11
6.1.2. Baud Rate	12
6.1.3. Comunicación síncrona y asíncrona	13
6.1.4. Comunicación Half Duplex	13
6.1.5. Estándar non-return-to-zero (NRZ)	14
6.2. Motores utilizados	14
6.2.1. Dynamixel XL-320	14
6.2.2. Dynamixel AX-12A	16
6.2.3. Dynamixel MX-160T	20
6.3. Protocolo de comunicación Dynamixel 1.0	23
6.3.1. Conceptos de comunicación	23
6.3.2. Paquetes de instrucción	23

6.4.	Microcontrolador PIC16F887	24
6.4.1.	Módulo EUSART	24
6.4.2.	Generador de Baud Rate en PIC16F887	26
6.5.	Optoacopladores	26
6.6.	Fuentes de voltaje	27
7.	Metodología	29
7.1.	Desarrollo de software	30
7.1.1.	R+ Manager y Roboplus	30
7.1.2.	Microcontrolador arduino	30
7.1.3.	Movimiento de motores Dynamixel con microcontrolador PIC16F887	34
7.1.4.	Comunicación con módulo de interacción por voz	36
7.2.	Diseño de hardware	37
7.3.	Manufactura	37
8.	Resultados	39
8.1.	Diagramas de flujo	39
8.1.1.	Subrutina setupEUSART	39
8.1.2.	Subrutina transmisión EUSART	40
8.1.3.	Subrutina recepción EUSART	41
8.1.4.	Subrutina de movimiento de motores XL-320	42
8.1.5.	Subrutina cálculo de CRC	43
8.1.6.	Subrutina de movimiento de motores AX-12A	44
8.1.7.	Subrutina de movimiento de motores MX-106T	45
8.2.	Paquetes Transmitidos	46
8.2.1.	Bytes en paquete transmitido para control de motor XL-320	46
8.2.2.	Bytes en paquete transmitido para control de motor AX-12A	47
8.2.3.	Bytes en paquete transmitido para control de motor MX-106T	47
8.3.	Circuitos diseñados	48
8.3.1.	Circuito diseñado para placas de control	48
8.4.	Módulo de comunicación con módulo de interacción por voz	49
8.4.1.	Banderas definidas para instrucciones con módulo de interacción por voz	49
8.4.2.	Diagrama de conexiones de módulo de comunicación con módulo de interacción por voz	50
9.	Discusión	51
9.1.	Subrutinas del módulo EUSART	51
9.1.1.	Subrutina EUSART	51
9.1.2.	Subrutina recepción EUSART	51
9.2.	Análisis de bytes para movimiento de motores	52
9.2.1.	Motores Dynamixel XL-320	52
9.2.2.	Motores Dynamixel AX-12A	53
9.2.3.	Motores Dynamixel MX-106T	53
9.3.	Análisis de bytes transmitidos para movimiento de motores	54
9.3.1.	Bytes transmitidos para motores Dynamixel XL-320	54
9.3.2.	Bytes transmitidos para motores Dynamixel AX-12A	55
9.3.3.	Bytes transmitidos para motores Dynamixel MX-106T	55

9.4. Selección de fuentes de alimentación	55
9.5. Análisis del circuito de aislamiento eléctrico	56
9.6. Análisis de comunicación entre PICS y módulo de interacción por voz	56
10. Conclusiones	57
10.1. Conclusiones de diseño de subrutinas de movimiento	57
10.2. Conclusiones del diseño del módulo de alimentación	57
11. Recomendaciones	59
11.1. Recomendaciones generales	59
12. Bibliografía	61
13. Anexos	63
13.1. Análisis con Osciloscopio y Saleae	63
13.1.1. Capturas de Osciloscopio	63
13.1.2. Pruebas con Saleae	68
13.2. Fotos importantes	75
13.3. Circuitos exportados a PDF	80

1.	Transferencia de bits en comunicación serial [5]	11
2.	Transferencia de datos síncrona [6]	13
3.	Pines digitales Dynamixel XL-320 [10]	15
4.	Direcciones de memoria Dynamixel XL-320 [10]	16
5.	Entradas Dynamixel AX-12A [12]	18
6.	Direcciones de memoria Dynamixel AX-12A [11]	19
7.	Entradas Dynamixel MX106T [13]	21
8.	Direcciones de memoria Dynamixel MX-106T [13]	22
9.	Pinout PIC16F887 [14]	24
10.	Circuito asociado a un optoacoplador [15]	27
11.	Etapas internas de una fuente de voltaje [16]	28
12.	Ejemplo de salida de señal de control de motor XL-320	31
13.	Ejemplo de salida de señal de control de motor AX12A	33
14.	Ejemplo de salida de señal de control de motor MX106T	34
15.	Subrutina para inicializar módulo EUSART	39
16.	Subrutina para iniciar transmisión en módulo EUSART	40
17.	Subrutina para iniciar recepción en módulo EUSART	41
18.	Subrutina para movimiento de motores XL-320	42
19.	Subrutina para el cálculo de CRC	43
20.	Subrutina para movimiento de motores AX-12A	44
21.	Subrutina para movimiento de motores MX-106T	45
22.	Circuito diseñada para placas de control	48
23.	Diagrama de conexiones entre PICs y módulo de interacción por voz	49
24.	Diagrama de conexiones entre PICs y módulo de interacción por voz	50
25.	Prueba motor XL-320, ID 100, Posición 10	63
26.	Prueba motor XL-320, ID 100, Posición 100	64
27.	Prueba motor XL-320, ID 100, Posición 1000	64
28.	Prueba motor AX-12A, ID 2, Posición 0	65
29.	Prueba motor AX-12A, ID 2, Posición 512	65
30.	Prueba motor AX-12A, ID 2, Posición 1023	66

31.	Prueba motor MX-106T, ID 1, Posición 1, Velocidad 100	66
32.	Prueba motor MX-106T, ID 1, Posición 1023, Velocidad 500	67
33.	Prueba motor MX-106T, ID 1, Posición 4095, Velocidad 1023	67
34.	Prueba motor XL-320, ID 1, Posición 10	68
35.	Prueba motor XL-320, ID 1, Posición 100	68
36.	Prueba motor XL-320, ID 1, Posición 1000	68
37.	Prueba motor XL-320, ID 5, Posición 10	68
38.	Prueba motor XL-320, ID 5, Posición 100	69
39.	Prueba motor XL-320, ID 5, Posición 1000	69
40.	Prueba motor XL-320, ID 50, Posición 10	69
41.	Prueba motor XL-320, ID 50, Posición 100	69
42.	Prueba motor XL-320, ID 50, Posición 1000	70
43.	Prueba motor AX-12A, ID 1, Posición 10	70
44.	Prueba motor AX-12A, ID 1, Posición 100	70
45.	Prueba motor AX-12A, ID 1, Posición 1000	70
46.	Prueba motor AX-12A, ID 7, Posición 10	71
47.	Prueba motor AX-12A, ID 7, Posición 100	71
48.	Prueba motor AX-12A, ID 7, Posición 1000	71
49.	Prueba motor AX-12A, ID 120, Posición 10	71
50.	Prueba motor AX-12A, ID 120, Posición 100	72
51.	Prueba motor AX-12A, ID 120, Posición 1000	72
52.	Prueba motor MX-106T, ID 1, Posición 10, Velocidad 10	72
53.	Prueba motor MX-106T, ID 1, Posición 10, Velocidad 100	72
54.	Prueba motor MX-106T, ID 1, Posición 10, Velocidad 1000	73
55.	Prueba motor MX-106T, ID 1, Posición 100, Velocidad 10	73
56.	Prueba motor MX-106T, ID 1, Posición 100, Velocidad 100	73
57.	Prueba motor MX-106T, ID 1, Posición 100, Velocidad 1000	73
58.	Prueba motor MX-106T, ID 1, Posición 1000, Velocidad 10	74
59.	Prueba motor MX-106T, ID 1, Posición 1000, Velocidad 100	74
60.	Prueba motor MX-106T, ID 1, Posición 1000, Velocidad 1000	74
61.	Easy Pic v7 utilizada para pruebas	75
62.	Saleae logic pro 16 utilizado para pruebas	75
63.	Diseño de placa versión 1	76
64.	Soldadura de placa versión 1	76
65.	Diseño de placa versión 2	76
66.	Soldadura de placa versión 2	77
67.	Pruebas de comunicación con módulo de interacción por voz	77
68.	Pruebas con circuito de aislamiento	77
69.	Pruebas con estructura de hombro	78
70.	Pruebas con estructura de cabeza	79

Lista de cuadros

1.	Características físicas del motor Dynamixel XL-320 [10]	15
2.	Características físicas del motor Dynamixel XL-320 [11]	17
3.	Características físicas del motor Dynamixel MX-106T [13]	20
4.	Estructura básica de un paquete de instrucción [7]	23
5.	Características del microcontrolador PIC16F887 [14]	24
6.	Fórmula del cálculo de Baud Rate para PIC16F887 [8]	26
7.	Bytes de paquete de control de motor XL-320	46
8.	Bytes de paquete de control de motor AX-12A	47
9.	Bytes de paquete de control de motor MX-106T	47
10.	Banderas de instrucción transmitidas por el módulo de interacción por voz	50

El megaproyecto “Animatronix” es un robot animatrónico desarrollado por estudiantes de ingeniería Electrónica e ingeniería Mecatrónica de la Universidad del Valle de Guatemala. Este será implementado en el parque de atracciones Irtra, ubicado en el departamento de Retalhuleu.

Para la elaboración del módulo de alimentación y control de movimientos de servo motores, se tuvo como primer requerimiento la familiarización con el equipo y con los materiales con los cuales se iban a trabajar a lo largo del proyecto. Para el movimiento de todas las partes de animatrónico pirata se trabajó con tres tipos de motores: Los servo motores Dynamixel XL-320, los servo motores Dynamixel AX-12A, y los servo motores Dynamixel MX-106T. El movimiento de estos motores se adecuó a mecanismos desarrollados por otros miembros del equipo de trabajo.

Para lograr un movimiento realista para los motores, se desarrollaron códigos de control de movimiento para todos los motores del sistema, con en el software MPLAB X e implementados en microcontroladores PIC16F887. Con estos códigos es posible controlar la posición y, en algunos casos, la velocidad de movimiento de los motores Dynamixel. Además, se desarrolló una interfaz de comunicación entre el módulo de alimentación y control de movimientos, y el módulo de Interacción por voz.

Finalmente, se describe el proceso de diseño del módulo de alimentación de los motores. Este incluye la selección de una fuente de alimentación para todos los actuadores del sistema, circuitos de aislamiento eléctrico para que ruido generado por los actuadores no afecte a los microcontroladores, y drivers de potencia en casos específicos, señales de control del PIC16F887.

CAPÍTULO 1

Introducción

El trabajo de graduación que se presenta a continuación forma parte del megaproyecto “Animatronix”, con el objetivo de diseñar e implementar un robot animatrónico interactivo.

El objetivo general de este módulo es satisfacer las necesidades de voltaje y de corriente de todos los actuadores y plataformas utilizadas para el control de movimientos del animatrónico, además del desarrollo de códigos de control de movimiento de todos los servo motores del sistema.

Para esto se utilizó el software Altium Designer para el diseño e implementación de circuitos de aislamiento eléctrico, y el diseño de placas donde se colocaron los microcontroladores utilizados para el control de movimientos de los servo motores. Además, se utilizó el software MPLAB X para desarrollar códigos de control de movimientos y, en ciertos casos, velocidad de los motores Dynamixel proporcionados por la Universidad del Valle de Guatemala.

Posteriormente se realizaron pruebas de funcionamiento con módulos desarrollados por otros miembros del equipo. Estos incluían estructuras diseñadas para el hombro, muñeca, mano y cabeza del robot animatrónico. También se realizaron pruebas con el módulo de interacción por voz, con el fin de desarrollar un sistema de comunicación entre ambos módulos.

Como antecedentes para este proyecto se tomó el trabajo de graduación *Animatronics: Módulo de Potencia e Interacción Humana*, y las librerías de arduino *Dynamixel XL-320, A servo library for Arduino* [1] para control de movimiento de los motores XL-320, *AX-12A-servo-library*[2] para control de movimiento de motores AX-12A, y *Dynamixel-Serial-Master*[3] para control de movimiento de motores MX-106T.

El trabajo de graduación *Animatronics: Módulo de Potencia e Interacción Humana* se tomó, como su nombre lo indica, como referencia para los aspectos de alimentación y potencia de este trabajo. En este trabajo se utilizaron dos fuentes de alimentación eTopxizu. Estas contaban con salidas que podían suministrar 12 voltios y hasta 30 amperios de corriente. Esto era suficiente para alimentar a los 12 motores en su sistema. Se utilizaron 5 servomotores Dynamixel MX106T, cada uno capaz de consumir hasta 5.2 amperios de corriente, y 7 motores MG996R, con demandas de corriente hasta de 2.5 amperios.

En sus circuitos de aislamiento eléctrico se utilizaron los optoacopladores 4N25. El problema de los 4N25 era el tiempo de caída más pronunciado en la señal de salida con respecto a la de entrada. Además, la velocidad de respuesta no era la adecuada para controlar servomotores que operan a un Baud Rate tan alto como son los Dynamixel MX-106T.

Para el control de movimiento de los motores se utilizaban microcontroladores Arduino Mega. Esto, por la buena integración que tiene con otros sistemas, su velocidad de procesamiento, memoria, conectividad y número de pines I/O. Para lograr el control de movimiento de los motores Dynamixel se utilizó la librería “Dynamixel for Arduino”. Además, se utilizó la interfaz Matlab para establecer comunicaciones con los demás módulos del proyecto.

La librería “Dynamixel XL-320, A servo library for Arduino” [1] fue desarrollada en junio de 2015 por la compañía “HelloSpoon” para el manejo de motores Dynamixel con la plataforma arduino. Esta librería utiliza el protocolo de comunicación serial y los pines TX y RX del arduino para comunicarse con los motores Dynamixel XL-320. Esta incluye subrutinas para la configuración de los motores, asignación de id y configuración de Baud Rate. Además, incluye ejemplos de funciones básicas de movimiento con los motores XL-320 y cambio de

color del LED del motor.

La librería “AX-12A-servo-library”[2] es utilizada para el control de movimiento de los motores Dynamixel AX-12A. Esta fue desarrollada por la organización ThingType en agosto de 2017. La librería cuenta con subrutinas para la asignación de ID y configuración de Baud Rate de motores Dynamixel AX-12A. Dentro de ella se pueden encontrar ejemplos para el uso de los motores en modo servo y en modo de rueda continua. Incluye también ejemplos de parpadeo del LED incluido en cada motor.

Por último, se tomó como antecedentes la librería “Dynamixel-Serial-Master”[3] para control de los motores MX-106T. Esta librería fue desarrollada por Zach Shiner en abril de 2015 para el control motores Dynamixel. Se utilizó solo como referencia para movimiento de motores Dynamixel MX-106T. Esta se basa en el protocolo de comunicación Dynamixel 1.0, y utiliza la interfaz arduino para la comunicación con los motores. Esta incluía subrutinas para configuración de ID y Baud Rate deseado, además de ejemplos de movimiento de motores. El añadido que esta librería presentaba era ejemplos de velocidad de movimiento de motores.

Los enlaces para poder descargar estas librerías se pueden encontrar en la sección de bibliografía.

La animatrónica es una de las técnicas más utilizadas a nivel mundial en la rama de entretenimiento y es de gran atracción para todas las personas cuando son implementados para transmitir un mensaje, campaña publicitaria o son colocados en parque de atracciones y recreación. Lastimosamente en Guatemala no se cuenta con esta tecnología por diversas causas. La principal es que ninguna empresa o entidad en Guatemala ha invertido tiempo y recursos en producir animatrónicos para ninguna causa. Debido a esto, si una empresa guatemalteca desea comprar un animatrónico será necesario importarlo de otro país.

Por eso se plantea una solución de bajo costo que pueda suplir las necesidades de tecnología animatrónica en el país sin tener que depender de empresas extranjeras. Esta solución busca aumentar el atractivo de los parques de recreación y servicios de entretenimiento en general en el país.

Con respecto al módulo de alimentación y control de movimientos de servo motores, es indispensable para el funcionamiento del animatrónico. De no ser por este módulo el animatrónico no sería capaz de moverse porque no tendría una fuente de alimentación capaz de satisfacer sus dependencias de voltaje y corriente. Añadiendo a esto la programación de los motores es indispensable o el animatrónico estaría estático y no simularía vida.

Además, este es un proyecto que integra todas las ramas estudiadas a lo largo de la carrera de ingeniería electrónica y es perfecto para su evaluación final. El proyecto abarca sistemas de control, cálculos de potencia, programación de microcontroladores, diseño e implementación de circuitos y diseño de hardware, todos conceptos muy importantes en la rama de la ingeniería electrónica. Por último, el tema del proyecto es uno que me ha sido de mi interés desde hace mucho tiempo y que, además, espero sea el tema de mi especialización al finalizar mis estudios de pregrado.

4.1. Objetivo general

Satisfacer las necesidades de voltaje y corriente de los motores DC, servo motores y todos los componentes de control del pirata animatrónico de manera eficiente y realizar un control para sus mecanismos que simule lo más posible un movimiento realista y fluido.

4.2. Objetivos específicos

- Implementar un módulo de alimentación para todos los servo motores del pirata animatrónico.
- Diseñar e implementar drivers de potencia para cada servo motor del pirata animatrónico.
- Diseñar y fabricar placas impresas de comunicación y control de servo motores para cabeza, hombros, codos, manos, torso.
- Programar microcontroladores PIC16F887 para control de movimiento de servo motores del pirata animatrónico.
- Implementar módulos de comunicación entre microcontroladores y módulo de reconocimiento de expresiones y gestos.

Este proyecto está dividido en dos partes. El desarrollo de códigos de control de movimiento para los motores Dynamixel XL-320, Dynamixel AX-12A y Dynamixel MX-106T, y el desarrollo de un módulo de alimentación para todos los actuadores y microcontroladores del sistema.

En el módulo de alimentación no se construirá una fuente de voltaje desde cero, si no que se escogerá una preexistente que pueda satisfacer las necesidades de voltaje y de corriente de todos los actuadores del sistema. Ya que son servo motores no serán necesarios drivers de potencia para la alimentación de los mismos. A este módulo se incluyen el diseño de circuitos de aislamiento eléctrico y las placas de control de movimiento para las partes del animatrónico que necesiten una.

Para el diseño de código para control de movimientos, se entiende como el diseño de código que tome una posición, o una velocidad deseada, y mueva el motor a esa posición o velocidad deseada. No se trabajará ninguna técnica de control o se hará alguna implementación de controladores analógicos o digitales. Se trabajará exclusivamente con servo motores Dynamixel, de la serie XL-320, AX-12A y MX106T.

6.1. Comunicación serial

La comunicación serial es un protocolo utilizado frecuentemente para comunicación entre dispositivos. La comunicación es lograda usando tres pines: El pin de transmisión o TX, el pin de recepción o RX y el pin de tierra. Existen otros pines disponibles para el control del flujo de datos, pero no son estrictamente requeridos. Otros estándares como el RS-485 definen funcionalidades adicionales como tasas más altas de transferencia de bits, cables más largos y conexiones de hasta 256 dispositivos [4]

El concepto de comunicación serial se define de la siguiente manera. El puerto serial envía y recibe bytes de información un bit a la vez. Aún y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. [4]

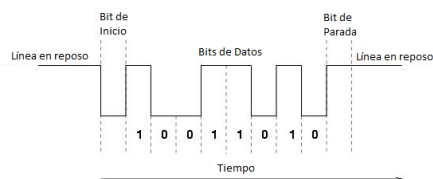


Figura 1: Transferencia de bits en comunicación serial [5]

6.1.1. Bits de datos

Bits de datos se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende del tipo de información que se transfiere. Un paquete se refiere a una

transferencia de bytes, incluyendo los bits de inicio/parada, bits de datos y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se puede usar para referirse a todos los casos.[4]

Bit de inicio

Es utilizado para indicar el inicio de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Mientras más bits de inicio se usen, mayor será la tolerancia a la sincronía de los relojes; sin embargo, la transmisión será más lenta.[4]

Bits de parada

Son usados para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada, además de indicar el fin de la transmisión, otorgan un margen de tolerancia para la diferencia de los relojes. Al igual que con los bits de inicio, mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes y la transmisión será más lenta.[4]

Bits de paridad

Son utilizados para verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados. [4]

6.1.2. Baud Rate

Velocidad de transmisión o “baud rate” indica el número de bits por segundo que se transfieren, y se mide en baudios. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz. Un estándar común para velocidad de transmisión en computadoras es de 9600 baudios por segundo.[4]

6.1.3. Comunicación síncrona y asíncrona

Poseer conocimiento de cuando muestrear y cuando transmitir es crucial para transferencia de datos. Si se viola el baud rate de transmisión o recepción, o no se tiene algún tipo de sincronización al momento de enviar o recibir datos, se puede llegar a muestrear datos incorrectos o perder información.

Comunicación síncrona

La comunicación síncrona es generada cuando la información es transmitida a una velocidad constante sincronizada con una señal de reloj. La información es generada con respecto a los flancos de subida o bajada de un reloj. [6]

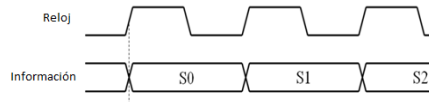


Figura 2: Transferencia de datos síncrona [6]

En este tipo de comunicación un reloj común sincroniza el intercambio de información. La señal de reloj puede ser generada por el dispositivo que este transmitiendo los datos en ese momento o por un reloj externo que esté conectado a ambos dispositivos. La comunicación síncrona permite velocidades más altas de transferencia de información que los métodos asíncronos, porque los bits adicionales usados para marcar el principio y final de cada byte de información no son requeridos [6]

Comunicación asíncrona

La comunicación asíncrona es generada cuando la información se envía por medio de señales acopladas; no es necesaria una señal de reloj. Señales y bit de control sincronizan la comunicación entre los dispositivos. Los requerimientos necesarios para enviar estos bits adicionales causan que la comunicación asíncrona sea más lenta que la comunicación síncrona; pero cuenta con la ventaja de no dar más carga al procesador pues no es necesario manejar bits de control. La mayoría de puertos seriales operan con comunicación asíncrona. [6] Para tener una mejor idea del funcionamiento de la comunicación asíncrona revisar la Figura 1

6.1.4. Comunicación Half Duplex

Half Duplex se refiere a un protocolo de comunicación serial donde los pines Tx y Rx no pueden ser usados al mismo tiempo. Este método generalmente se utiliza cuando varios dispositivos necesitan estar conectados a un mismo bus de datos. Cuando el controlador quiere transmitir este tiene que habilitar su modo de transmisión mientras que los demás dispositivos deben entrar en modo de recepción. Cuando un dispositivo quiere dar retroali-

mentación este debe pasar a modo de transmisión y el controlador a modo de recepción. [7]

6.1.5. Estándar non-return-to-zero (NRZ)

Se refiere a un código binario donde los voltajes positivos representan 1 y la referencia a tierra o cero voltios representa 0. NRZ se refiere al hecho de que bits consecutivos del mismo valor se mantienen en el nivel del ultimo bit sin regresar a un punto neutral. En este estándar cada paquete de transmisión empieza con un bit de inicio, seguido de 8 o 9 bits y termina con 1 o 2 bits de parada. [8]

6.2. Motores utilizados

Los servo motores Dynamixel son actuadores inteligentes desarrollados para ser las conexiones de un robot o estructura mecánica. Los servo motores Dynamixel están diseñados para ser sistemas modulares, utilizados para movimientos robóticos potentes y flexibles. Además, son actuadores de alto rendimiento, de corriente directa, totalmente integrados con caja reductora, controlador, circuito de potencia y sistema de red, todo en un actuador modular. Estos pueden ser monitoreados por medio de un stream de datos enviados por protocolo de comunicación serial asíncrono. [9]

6.2.1. Dynamixel XL-320

El motor Dynamixel XL-320 es un actuador robótico inteligente totalmente integrado que cuenta con caja reductora, controlador, circuito de potencia y sistema de red, todo en un pequeño servo motor DC. El Dynamixel XL-320 utilizan comunicación TTL y puede ser conectado fácilmente en serie con otros motores Dynamixel. [10] Las características físicas del XL-320 se describen en el Cuadro 1

Según el Cuadro 1 se puede destacar que los motores Dynamixel XL-320 trabajan con un protocolo de comunicación serial Half Duplex asíncrono, con data de 8 bits, 1 bit de parada, 1 bit de inicio y ningún bit de paridad. Su protocolo de comunicación será explorado más a profundidad en la sección 4.3 del documento.

Los motores Dynamixel XL-320 cuentan con dos entradas de tres pines cada uno, estas ubicadas a cada lado del motor. Estas pueden ser utilizadas para conectar los motores en serie. Los pines con los que cuenta cada entrada son voltaje de alimentación, tierra y señal de control.[10] Esto puede apreciarse en la Figura 3.

Dynamixel XL-320	
Descripción	Especificación
Peso	16.7 g
Dimensiones	24mm x 36mm x 27mm
Radio de engranajes	238:1
Voltaje de operación	5 V
Torque de parada	0.39 N.m
Velocidad sin carga	114 RPM
Motor	Motor con núcleo de cobre
Ángulo mínimo de control	0.29°x 1024
Rango de operación	Modo actuador: 300° Modo rueda: giro sin fin
Rango de voltaje de operación	5 a 8.4 V
Corriente Máxima	900mA
Temperatura de operación	-5° a 70°
Señal de comando	Paquete digital
Protocolo	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No parity)
Conexión (física)	Conexión TTL (Cable cadena de margarita)
ID	253 ID (0-252)
Baud Rate	7843bps a 1Mbps
Materiales	Recubrimiento: plásticos de ingeniería Engranajes: plásticos de ingeniería

Cuadro 1: Características físicas del motor Dynamixel XL-320 [10]

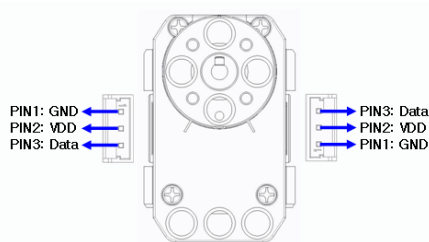


Figura 3: Pines digitales Dynamixel XL-320 [10]

La Figura 4 muestra las direcciones de memoria del motor Dynamixel XL-320. El motor cuenta con 52 localidades de memoria. Estas divididas en memoria RAM y memoria EEPROM. LA diferencia entre memoria RAM y memoria EEPROM es que cuando se desconecta de una fuente de poder la memoria RAM se limpia, mientras que la memoria EEPROM guarda su valor actual. Además de esto las direcciones de memoria pueden ser clasificadas como de lectura o de lectura y escritura. De las direcciones de lectura solo se puede consul-

tar su valor, pero no modificarlo, mientras que de las direcciones de lectura y escritura sí se puede consultar su valor y escribir sobre ellas. El Cuadro además incluye valores mínimos y máximos que puede tener la dirección, y un valor inicial que tendrá la direcciones cuando es conectada por primera vez [10].

Area	Address (Hexadecimal)	Size(byet)	Name	Description	Access	Initial Value	Min	Max
EEPROM	0	2	Model Number	Model number	R	350	-	-
	2	1	Version of Firmware	Information on the version of firmware	R	-	-	-
	3	1	ID	ID of Dynamixel	RW	1	0	252
	4	1	Baud Rate	Baud Rate of Dynamixel	RW	3	0	3
	5	1	Return Delay Time	Return Delay Time	RW	250	0	254
	6	2	CW Angle Limit	clockwise Angle Limit	RW	0	0	1023
	8	2	CCW Angle Limit	counterclockwise Angle Limit	RW	1023	0	1023
	11	1	Control Mode	Control Mode	RW	2	1	2
	12	1	Limit Temperature	Internal Limit Temperature	RW	65	0	150
	13	1	lower Limit Voltage	Lowest Limit Voltage	RW	60	50	250
	14	1	Upper Limit Voltage	Upper Limit Voltage	RW	90	50	250
	15	2	Max Torque	Lowest byte of Max. Torque	RW	1023	0	1023
	17	1	Return Level	Return Level	RW	2	0	2
	18	1	Alarm Shutdown	Shutdown for Alarm	RW	3	0	7
R7AM	24	1	Torque Enable	Torque On/Off	RW	0	0	1
	25	1	LED	LED On/Off	RW	0	0	7
	27	1	D Gain	D Gain	RW	0	0	254
	28	1	I Gain	I Gain	RW	0	0	254
	29	1	P Gain	P Gain	RW	32	0	254
	30	2	Goal Position	Goal Position	RW	-	0	1023
	32	2	Moving Speed	Goal Speed	RW	-	0	2047
	35	2	Torque Limit	Goal Torque	RW	-	0	1023
	37	2	Present Position	Current Position	R	-	-	-
	39	2	Present Speed	Current Speed	R	-	-	-
	41	2	Present Load	Current Load	R	-	-	-
	45	1	Present Voltage	Current Voltage	R	-	-	-
	46	1	Present Temperature	Present temperature	R	-	-	-
	47	1	Registered Instruction	Registered Instruction	R	0	-	-
49	1	Moving	Moving	R	0	-	-	
50	1	Hardware Error Status	Hardware error status	R	0	-	-	
51	2	Punch	Punch	RW	32	0	1023	

Figura 4: Direcciones de memoria Dynamixel XL-320 [10]

6.2.2. Dynamixel AX-12A

El servo motor Dynamixel AX-12A es un actuador inteligente desarrollado por Robotis y utilizado en todo tipo de trabajos de robótica, estructuras mecánicas o hobbies. Como la mayoría de actuadores Dynamixel, este cuenta con caja reductora, controlador, circuito

de potencia y sistema de red, todo en un actuador modular. [11] Las características físicas del servo motor Dynamixel AX-12A pueden apreciarse en el Cuadro 2.

Dynamixel AX-12A	
Descripción	Especificación
Peso	54.6 g
Dimensiones	32mm x 50mm x 40mm
Radio de engranajes	254:1
Voltaje de operación	12 V
Torque de parada	1.5 N.m
Velocidad sin carga	59 RPM
Motor	Motor con núcleo de cobre
Ángulo mínimo de control	0.29°x 1024
Rango de operación	Modo actuador: 300° Modos rueda: giro sin fin
Rango de voltaje de operación	9 a 12 V
Corriente máxima	900mA
Temperatura de operación	-5° a 70°
Señal de comando	Paquete digital
Protocolo	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No pa- rity)
Conexión (física)	Conexión TTL (Cable cadena de margarita)
ID	254 ID (0-253)
Baud Rate	7843bps a 1Mbps
Materiales	Recubrimiento: plásticos de ingenie- ría Engranajes: plásticos de ingenie- ría

Cuadro 2: Características físicas del motor Dynamixel XL-320 [11]

Como los motores XL-320 anteriormente, los motores Dynamixel AX-12A trabajan con un protocolo de comunicación serial Half Duplex asíncrono, con data de 8 bits, 1 bit de parada, 1 bit de inicio y ningún bit de paridad. Por default los motores Dynamixel AX-12A vienen con ID igual a 1, y un Baud Rate de 1Mbps [11].

Los motores Dynamixel AX-12A cuentan con dos entradas de alimentación, de tres pines cada una, en la parte posterior del motor. Estos pines son, de izquierda a derecha, señal, alimentación y tierra [11].

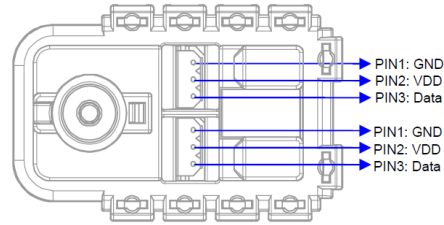


Figura 5: Entradas Dynamixel AX-12A [12]

La Figura 6 muestra la tabla con las direcciones de memoria del motor Dynamixel AX-12A. El motor Dynamixel AX-12A cuenta con 50 direcciones de memoria, divididas en memoria EEPROM y memoria RAM. Estas pueden ser clasificadas también como direcciones de lectura o direcciones de lectura y escritura [11]. En el cuadro puede apreciarse que hay dirección que su nombre termina con una (L) o una (H). Las direcciones terminadas en (L) son los bytes menos significativos, o low, de parámetros que necesitas 2 bytes. Las direcciones terminadas en (H) son los bytes más significativos, o high, de direcciones de memoria que necesitan 2 bytes [11].

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	12 (0X0C)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	1 (0X01)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	3 (0X03)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	70 (0X46)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	140 (0XBE)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36(0x24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36(0x24)
RAM	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	CW Compliance Margin	CW Compliance margin	RW	1 (0X01)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	1 (0X01)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	CCW Compliance slope	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocity)	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocity)	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	32 (0X20)
49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)	

Figura 6: Direcciones de memoria Dynamixel AX-12A [11]

6.2.3. Dynamixel MX-160T

El motor Dynamixel MX-106T es un actuador robótico equipado con un procesador Cortex M3 de 32 bits a 72 MHz, un encoder magnético sin contacto con una resolución hasta 4 veces mayor que la serie AX o XL, y hasta 3Mbps de velocidad de transmisión. Todos los servo motores de la serie MX usan un voltaje de operación de 12v, por lo que las Dynamixels AX y MX pueden mezclarse sin tener que preocuparse por fuentes de alimentación separadas. Además de estas funciones, como los motores anteriores, los motores Dynamixel MX-106T cuentan con caja reductora, controlador, circuito de potencia y sistema de red, todo en un actuador modular[13]. Sus características físicas pueden apreciarse en el Cuadro 3.

Dynamixel MX-106T	
Descripción	Especificación
Peso	153 g
Dimensiones	40.2mm x 65.1mm x 46mm
Radio de engranajes	225:1
Voltaje de operación	12 V
Rango de voltaje de operación	10 a 14.8 V
Torque de parada	8.4 N.m a 12V
Corriente de parada	5.2 A a 12V
Velocidad sin carga	45 RPM a 12V
Motor	Motor Maxon
Ángulo mínimo de control	0.088°x 4096
Rango de operación	Modo actuador: 360° Modo rueda: giro sin fin
Rango de voltaje de operación	10 a 14.8 V
Temperatura de operación	-5° a 80°
Señal de comando	Paquete digital
Protocolo	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No parity)
Conexión (física)	Conexión TTL (Cable cadena de margarita)
ID	254 ID (0-253)
Baud Rate	8000bps a 4.5Mbps
Materiales	Recubrimiento: plásticos de ingeniería Engranajes: plásticos de ingeniería

Cuadro 3: Características físicas del motor Dynamixel MX-106T [13]

Como todos los motores utilizados en este trabajo, los motores Dynamixel MX-106T operan con un protocolo de comunicación serial Half Duplex asíncrono, con data de 8 bits, 1 bit de parada, 1 bit de inicio y ningún bit de paridad. Su protocolo de comunicación será descrito más a profundidad en la sección 4.3 del documento. Por default los motores Dynamixel MX-106T vienen con ID igual a 1, y un Baud Rate de 57600bps [13]. Además de la posición, a los motores Dynamixel MX-106T es posibles controlar su velocidad de

movimiento. Esta variando entre 1 y 1023.[13]

El motor Dynamixel MX-106T cuenta con dos entradas de alimentación, una a cada lado del motor. Los pines de estas entradas son tierra, alimentación, y señal de control. Esto puede apreciarse mejor en la Figura 7.

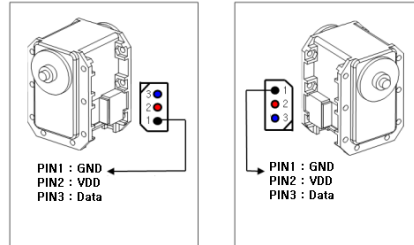


Figura 7: Entradas Dynamixel MX106T [13]

La Figura 8 muestra la tabla de direcciones de memoria del motor Dynamixel MX-106T. El motor Dynamixel MX-106T cuenta con 74 direcciones de memoria que pueden ser clasificadas en memoria RAM o memoria EEPROM dependiendo de su almacenamiento. Además, muestra si la dirección es de lectura o de lectura y escritura, junto con el valor inicial de la dirección. Al igual que las direcciones del motor Dynamixel AX-12A estas cuentas con un indicador si la dirección es del byte más significativo o menos significativo para un parámetro del motor.

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
E E P R O M	0 (0X00)	Model Number(L)	Lowest byte of model number	R	64 (0X40)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	1 (0X01)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	34 (0X22)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	15 (0X0F)
	10 (0X0A)	Drive Mode	Dual Mode Setting	RW	0(0X00)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	160 (0XA0)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)
	20 (0X14)	Multi Turn Offset(L)	multi-turn offset least significant byte (LSB)	RW	0 (0X00)
	21 (0X15)	Multi Turn Offset(H)	multi-turn offset most significant byte (MSB)	RW	0 (0X00)
	22 (0X16)	Resolution Divider	Resolution divider	RW	1 (0X01)
R A M	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	D Gain	Derivative Gain	RW	0 (0X00)
	27 (0X1B)	I Gain	Integral Gain	RW	0 (0X00)
	28 (0X1C)	P Gain	Proportional Gain	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocity)	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocity)	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	0 (0X00)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)
	68 (0X44)	Current(L)	Lowest byte of Consuming Current	RW	0 (0X00)
	69 (0X45)	Current(H)	Highest byte of Consuming Current	RW	0 (0X00)
	70 (0X46)	Torque Control Mode Enable	Torque control mode on/off	RW	0 (0X00)
71 (0X47)	Goal Torque(L)	Lowest byte of goal torque value	RW	0 (0X00)	
72 (0X48)	Goal Torque(H)	Highest byte of goal torque value	RW	0 (0X00)	
73 (0X49)	Goal Acceleration	Goal Acceleration	RW	0 (0X00)	

Figura 8: Direcciones de memoria Dynamixel MX-106T [13]

6.3. Protocolo de comunicación Dynamixel 1.0

Para controlar los motores Dynamixel se tiene que establecer una comunicación con base en el protocolo de comunicación Dynamixel 1.0. En este caso es necesario definir conceptos importantes que servirán a lo largo de este trabajo.

6.3.1. Conceptos de comunicación

Paquete

Es el nombre que se le da a los streams de data enviados y recibidos. Los paquetes pueden ser de dos tipos, paquetes de instrucción y paquetes de status. Los paquetes de instrucción son aquellos mandados por el controlador hacia el motor para controlarlo; y los paquetes de status que son los enviados en respuesta del motor hacia el controlador. [7]

ID

El ID es un número de identificación de cada motor Dynamixel utilizado cuando dos o más motores se conectan en serie a un controlador. Gracias al ID es que el controlador puede diferenciar entre motores, tanto como controlar y recibir información de un motor específico en todo momento. [7]

Tiempo de Byte a Byte

El tiempo de byte a byte se refiere al tiempo de espera entre byte cuando el controlador envía un paquete de instrucción al motor. Si este es mayor a 100mS, el motor reconoce que hubo un problema de comunicación y espera por el inicio del próximo paquete. [7]

6.3.2. Paquetes de instrucción

Dependiendo de la serie de motor utilizado el número de bytes enviados en un paquete de instrucción puede variar, pero todos siguen la misma estructura básica.

Encabezado1	Encabezado2	ID	Ancho	Instrucción
Dirección	Parámetro 1	...	Parámetro N	CHECKSUM 1

Cuadro 4: Estructura básica de un paquete de instrucción [7]

La manera en que se envían los paquetes de instrucción es mediante el envío de bytes consecutivos, cada uno representando los parámetros mostrados en el Cuadro 4.

por medio de comunicación serial. Este puede ser configurado como full-duplex o half-duplex de manera síncrona o asíncrona. [8]

Módulo EUSART modo asíncrono

El módulo EUSART transmite y recibe información usando el estándar non-return-to-zero. Este transmite y recibe primero el bit menos significativo de la información. La transmisión y la recepción del módulo EUSART son funcionalmente independientes, pero comparten el mismo formato de data y Baud Rate. Por el lado de la transmisión, el registro más importante a considerar es el "Transmit Shift Register", o TSR, que no es directamente accesible por software. Este obtiene su valor del registro TXREG. [8]

Transmisión con el módulo EUSART

Para habilitar la transmisión de datos se necesita manipular los valores de los bits TXEN, SYNC y SPEN del registro TXSTA. Al colocar en 1 el bit TXEN se habilita el circuito de transmisión del módulo EUSART. Colocar el bit SYNC en 0 le estamos diciendo al PIC que la transmisión será asíncrona. Y por último al colocar el bit SPEN en 1 configura el pin 25 o TX del PIC16F887 como salida digital. [8]

La transmisión comienza cuando se carga un valor al registro TXREG. Si este es el primer valor dado al registro, o el valor anterior ya fue completamente transmitido del TSR, este es enviado directamente al TSR; si el registro TSR no ha terminado de transmitir el valor anterior este valor es almacenado en el registro TXREG hasta que el bit de parada del valor anterior salga del TSR. Un detalle importante a tomar en cuenta al momento de transmitir con el módulo EUSART es la bandera de interrupción TXIF del registro PIR1. Esta bandera está encendida cuando la transmisión está habilitada, pero no hay ningún valor cargado al registro TXREG para transmisión. La bandera se limpia cuando el registro el TSR está ocupado transmitiendo un valor y ya se cargó un nuevo valor al registro TXREG para transmisión. Esta bandera es muy útil para monitorear el estado de nuestra transmisión y no cambiar el valor a transmitir durante el proceso transmisión. [8]

Recepción con el modulo EUSART

Para Habilitar la recepción de datos se necesita manipular los valores de los bits CREN, SYNC y SPEN del registro TXSTA. Al colocar en 1 el bit CREN se habilita el circuito de recepción del módulo EUSART. Colocar el bit SYNC en 0 le estamos diciendo al PIC que las operaciones realizadas serán asíncronas. Y por último al colocar el bit SPEN en 1 configura el pin 26 o RX del PIC16F887 como entrada digital. El módulo EUSART comienza la recepción de datos en el flanco de bajada del primer bit. Este es el bit de comienzo y su valor siempre es igual a 0. [8]

6.4.2. Generador de Baud Rate en PIC16F887

El generador de Baud Rate es un timer de 8 o 16 bits que se encarga de ayudar en operaciones del módulo EUSART síncrono o asíncrono. Trabaja normalmente en 8 bits, pero puede ser ajustado a 16 en caso de ser necesario. Los registros SPBRGH y SPBRG determinan el período del Baud Rate deseado [8]. La siguiente fórmula es usada para encontrar el valor de los registros SPBRGH y SPBRG para un Baud Rate deseado:

$$BaudRatedeseado = \frac{Fosc}{64([SPBRGH : SPBRG] + 1)} \quad (1)$$

En la ecuación 1 demuestra como calcular el Baud Rate deseado del módulo EUSART utilizando los registros SPBRGH y SPBRG y la frecuencia de oscilación ($Fosc$) del PIC16F887. Esta puede ser generada por un oscilador externo o utilizando el oscilador interno del microcontrolador. Tomando como X el valor de los registros solo tenemos que despejar para X y encontramos el valor de los registros necesario para obtener el Baud Rate deseado. Esto puede verse en la ecuación 2.

$$X = \frac{Fosc}{64 * BaudRatedeseado} - 1 \quad (2)$$

La ecuación 1 puede cambiar dependiendo de los valores de los bits SYNC, BRG16 y BRGH, y dependiendo del modo en que se esté utilizando el módulo EUSART [8]. La tabla 6 muestra las fórmulas a utilizar dependiendo el valor de estos bits y del modo de operación seleccionado.

Baud Rate PIC16F887				
SYNC	BRG16	BRGH	EUSART operación	Formula utilizada
0	0	0	8-bit Asíncrono	$Fosc/[64(n+1)]$
0	0	1	8-bit Asíncrono	$Fosc/[16(n+1)]$
0	1	0	16-bit Asíncrono	$Fosc/[16(n+1)]$
0	1	1	16-bit Asíncrono	$Fosc/[4(n+1)]$
1	0	X	8-bit Síncrono	$Fosc/[4(n+1)]$
1	1	X	16-bit Síncrono	$Fosc/[4(n+1)]$

Cuadro 6: Fórmula del cálculo de Baud Rate para PIC16F887 [8]

6.5. Optoacopladores

Un optoacoplador (llamado también optoaislador o aislador acoplado ópticamente) combina un LED y un fotodiodo en un solo encapsulado. La Figura 10 muestra el circuito de un optoacoplador. Tiene un LED en el lado de entrada y un fotodiodo en el lado de salida. La tensión de la fuente de la izquierda y la resistencia en serie establece una corriente en el LED. Entonces, la luz proveniente del LED incide sobre el fotodiodo, lo que genera una

corriente inversa en el circuito de salida, que produce una tensión en la resistencia de salida. La tensión de salida es igual a la tensión de la fuente menos la tensión en la resistencia [15].

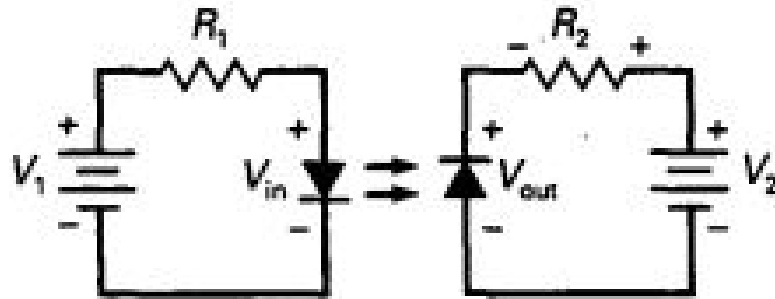


Figura 10: Circuito asociado a un optoacoplador [15]

Si la tensión de entrada varia, la cantidad de luz también lo hará, lo que significa que la tensión de salida cambia de acuerdo con la tensión de entrada. Por ello, la combinación de un LED y un fotodiodo recibe el nombre de optoacoplador. El dispositivo puede acoplar una señal de entrada con el circuito de salida [15].

La ventaja fundamental de un optoacoplador es el aislamiento eléctrico entre los circuitos de entrada y de salida. Mediante el optoacoplador, el único contacto que hay entre la entrada y la salida es un haz de luz. Si se coloca una señal de control del lado del LED, y un circuito de carga del lado del fotodiodo, ambos circuitos estarían entonces, eléctricamente aislados por el optoacoplador [15].

Las señales del circuito de control son transmitidas de manera óptica al circuito de carga, por lo que se encuentran libres de efectos retroactivos. En la mayoría de los casos, esta transmisión óptica es realizada con rayos de luz cuyas longitudes de onda se encuentran entre el rango rojo a infrarrojo, dependiendo de los requerimientos aplicables al optoacoplador [15].

La longitud de onda de la señal a transmitir puede variar desde una señal de voltaje DC constante, hasta una señal con una frecuencia en la banda de los MHz. Un optoacoplador es comparable con un transformador o un relé. Además de tener dimensiones más pequeñas en la mayoría de los casos, las ventajas de los optoacopladores con respecto a los relés son las siguientes: asegura tiempos considerablemente más cortos de conmutación, no tiene contactos mecánicos y no es propenso a interferencia causada por arcos. Gracias a todas estas ventajas, los optoacopladores son más convenientes para circuitos usados en microelectrónica, así como en procesamiento de datos y sistemas de comunicaciones [15].

6.6. Fuentes de voltaje

La fuente de voltaje o alimentación es un dispositivo que se conecta a la red eléctrica y transforma la corriente alterna en continua y proporciona la tensión a la que funcionan dispositivos eléctricos como computadoras, motores, etc. [16]

Debido a avances en la electrónica, las fuentes de alimentación deben proporcionar tensiones e intensidades muy estables que permanezcan insensibles a variaciones externas. Por lo cual suelen utilizarse circuitos estabilizadores o reguladores que mantengan constante la tensión de salida de la fuente, aun existiendo variaciones en la carga o en la tensión del suministro eléctrico. Estas suelen ir equipadas de circuitos eléctricos de protección que son capaces de disminuir la tensión de salida al detectar sobrecorrientes, evitando daños en los componentes a los que alimenta [16] .

Generalmente, las fuentes de alimentación constan de varios elementos o etapas con las siguientes funciones:

- Transformador: reduce la amplitud de corriente eléctrica alterna de entrada para poder manejarla por el resto de etapas [16] .
- Rectificador: elimina los ciclos negativos de la corriente alterna procedente del transformador [16] .
- Filtro: reduce el rizado de la señal de salida del rectificador [16] .
- Regulador: estabiliza la tensión aislándola de las fluctuaciones de la corriente eléctrica de la red [16] .

Estas etapas son visibles en la Figura 11.

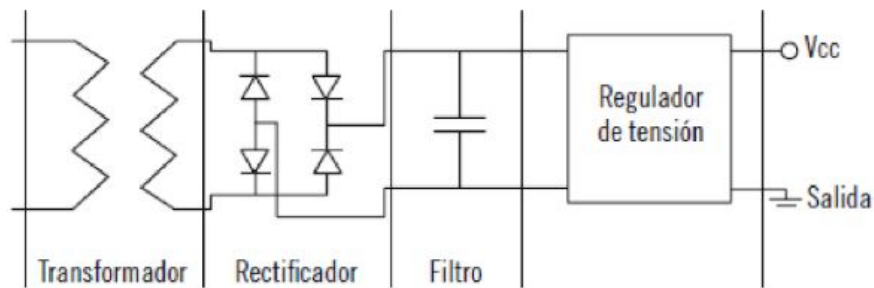


Figura 11: Etapas internas de una fuente de voltaje [16]

El siguiente trabajo se dividió en cuatro etapas principales: investigación, desarrollo de software, diseño de hardware y manufactura. En la etapa de investigación se consultaron libros, artículos científicos, investigaciones previas y manuales electrónicos sobre el funcionamiento de ciertos equipos para tener una idea preliminar del material con el que se estaba trabajando. En esta etapa se determinaron las características físicas y electrónicas de los motores Dynamixel. La teoría aprendida durante esta etapa se encuentra detallada en el marco teórico, por lo que no se discutirá en esta sección del trabajo.

En la etapa de desarrollo de software se trabajó primero con software R+ Manager, especializado para el control de motores Dynamixel. Luego se trabajó con librerías preexistentes de Arduino para el control de los motores sin el uso de un programador físico ni software dedicado al control de motores Dynamixel. Con lo aprendido de las librerías de arduino se desarrollaron códigos de control de movimiento, implementados en PIC16f887, para cada motor del sistema. Además, se desarrollaron códigos para la comunicación con el módulo de Interacción por Voz.

En la etapa de diseño de hardware se tomó todo lo aprendido en las etapas anteriores y se diseñó el módulo de alimentación del sistema. Esto incluyó la selección de la fuente de voltaje y corriente para los motores, el diseño e implementación de circuitos de aislamiento eléctrico entre los PICs y los motores, diseño de placas para los circuitos de control y aislamiento eléctrico.

Por último, en la etapa de manufactura se implementaron las placas impresas para los circuitos y se realizaron pruebas ya con las estructuras finales del pirata animatrónico.

7.1. Desarrollo de software

En la etapa de investigación se descubrió que los motores dynamixel pueden ser controlados con los programas R+ Manager y Roboplus. Ambos son programas desarrollados por Robotis y necesitan del adaptador físico “USB2Dynamixel”.

7.1.1. R+ Manager y Roboplus

El software R+ Manager se utilizó primero para el protocolo Dynamixel 1.0 a los 3 tipos de motores con los que se trabajó. Esto porque el protocolo 1.0 tiene mucha más documentación que el protocolo 2.0, además de ser este el protocolo predeterminado en los motores Dynamixel XL-320. En R+ Manager se les asignó el ID 1 a los 3 motores, y un Baud Rate de 9600 bps para pruebas de funcionamiento preliminares. Con este software se logró el primer movimiento de los motores Dynamixel XL-320, AX-12A y MX-106T. Una desventaja de R+ Manager es que cada vez que se conectaba un motor se tenía que actualizar el software del motor y hacer una recuperación de hardware. Con base en esto se decidió utilizar R+ Manager para solo la primera prueba de cada motor, para asignar un ID y un Baud Rate, y hacer las pruebas de movimiento con el programa Roboplus. En Roboplus era necesario saber el ID y el Baud Rate del motor con el que se estaba trabajando, pero este ya estaba definido para cada motor gracias a las pruebas realizadas con R+ Manager.

R+ Manager y Roboplus dieron un primer entendimiento del funcionamiento de los motores Dynamixel, pero presentaban inconvenientes con su implementación. Dentro de estos se puede destacar el hecho de que el movimiento de los motores se tenía que hacer en tiempo real y no se podían programar subrutinas para su uso posterior. Además, se necesitaba de un componente de hardware, el adaptador USB2Dynamixel, que permitieran la comunicación entre la computadora y los motores. Este también presentaba el problema que solo se podía controlar un tipo de motor a la vez.

Ambos programas se continuaron usando para futura referencia y en casos particulares, cuando se quería cambiar de ID a un motor o revisar su funcionamiento en caso de que no respondiera con otras señales de control (Arduino o PIC16F887).

7.1.2. Microcontrolador arduino

Para las pruebas de movimiento con arduino se trabajaron con tres librerías preexistentes de Arduino. La librería “Dynamixel XL-320, A servo library for Arduino” [1] para control de movimiento de los motores XL-320. La librería “AX-12A-servo-library”[2] para control de movimiento de motores AX-12A. Y por último, la librería “Dynamixel-Serial-Master”[3] para control de movimiento de motores MX-106T.

Movimiento de motores XL-320 con Arduino

Para el control de movimientos de los motores XL-320 con el microcontrolador arduino se utilizó la librería “Dynamixel XL-320, A servo library for Arduino” [1]. Esta proporcionaba

funciones, no detalladas, para el movimiento de los motores. Para usar esta librería primero se especificaba el id del motor, el Baud Rate en el que se estaba trabajando y la posición a la que se quería llevar el motor. Con esto se verificó que todos los motores XL-320 con los que se contaba fueran funcionales. Después de las pruebas preliminares se descubrió que algunos de los motores habían sido dañados en proyectos anteriores y tuvieron que ser descartados del proyecto.

Se realizaron distintas pruebas de movimiento y combinaciones de ID para apreciar mejor el movimiento de los motores. Además de esto se realizaron pruebas para monitorear el funcionamiento de los motores cuando eran conectados en serie. Se descubrió que ciertos motores tenían conexiones dañadas y solo podían alimentarse por uno de sus lados. Esto presentó un problema al principio, pero después de verificar el funcionamiento de ambas terminales en todos los motores disponibles, se delimitó aún más los motores con los que se podía trabajar.

Ya que las funciones para movimiento de los motores no estaban detalladas en las librerías, se buscó otra manera de analizar las funciones de control de movimiento de los motores. Se tenía con base a la investigación previa, la plantilla de los nombres de los paquetes transmitidos, pero no el número ni tampoco los valores que se debían transmitir. Para saber el número de paquetes transmitido por el arduino y los valores que estos representaban se realizó un proceso de ingeniería inversa a la señal de salida del pin TX del arduino. A la salida del pin TX se conectó la punta de un osciloscopio y se analizó la lectura. Se realizaron pruebas variando el número de ID enviado (se trabajó con 10 ID's). A cada uno de esos valores de ID después se le varió el valor de la posición requerido (se trabajó con 15 posiciones por ID). Un ejemplo de este análisis se presenta en la Figura 12.

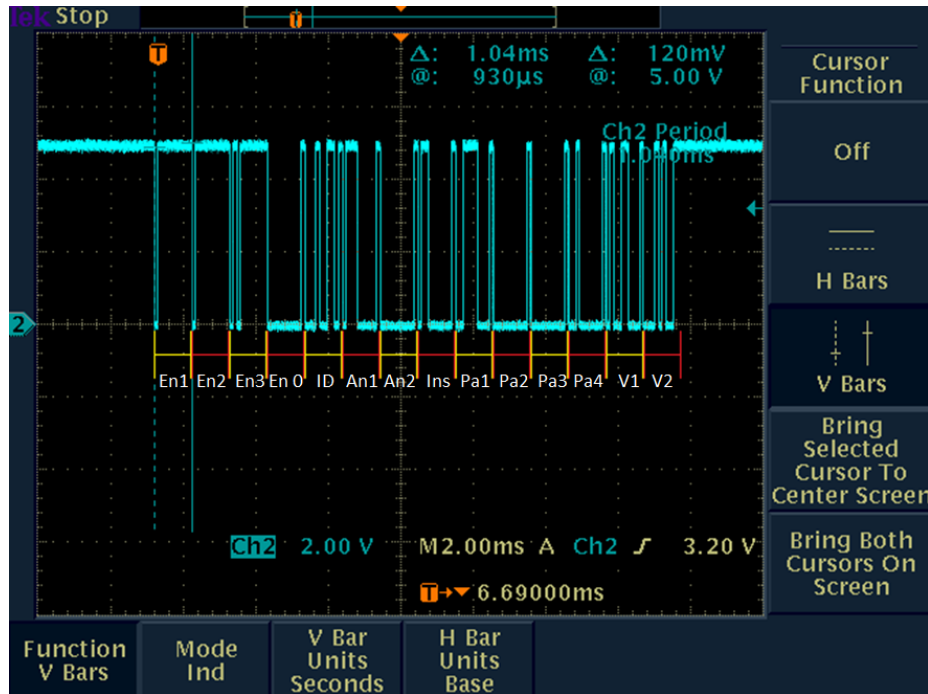


Figura 12: Ejemplo de salida de señal de control de motor XL-320

Gracias a estas pruebas se pudo determinar número de bytes enviados por paquete, y el valor de cada byte en la señal de control. En el caso de los motores XL-320 son 14 bytes de control. Los primeros 4 bytes son encabezados, seguidos de 1 byte de ID, luego 2 bytes de ancho de paquete, 1 byte de instrucción, 4 bytes de parámetros de instrucción y 2 bytes de verificación. Lo único que no se pudo determinar con las pruebas de arduino para los motores XL-320 fue como se calculaban los bytes de verificación en el paquete de control.

Movimiento de motores AX-12A con arduino

Para el control de movimientos de los motores AX-12A con el microcontrolador arduino se utilizó la librería “AX-12A-servo-library”[2]. Al igual que en el caso anterior esta librería contaba con funciones para el movimiento de los motores, pero no contaba con un detalle o una explicación de cómo funcionaba el código. Para el caso de los motores AX-12A los parámetros de entrada para la función eran los mismo que para los motores XL-320. Se necesitaba el ID del motor, el Baud Rate en el cual estaría trabajando el motor y la posición deseada.

Primero se realizaron las mismas pruebas de funcionamiento y movimiento con los motores AX-12A disponibles. En el caso de los AX-12A todos los motores funcionaban correctamente. En las pruebas de conexión en serie todas las terminales de los AX-12A funcionaron.

Se decidió hacer el mismo procedimiento de análisis de salida del arduino para el control de movimiento de los motores AX-12A. Se conectó a la salida del pin TX del arduino la punta del osciloscopio y se analizaron los bytes de salida. Se realizó un menor número de pruebas con los motores AX-12A ya que ya se contaba con una idea del comportamiento de los paquetes de control. Se trabajó con 5 valores de ID diferentes y se realizó un barrido de 10 posiciones diferentes por ID. Un ejemplo de este análisis se presenta en la Figura 13.

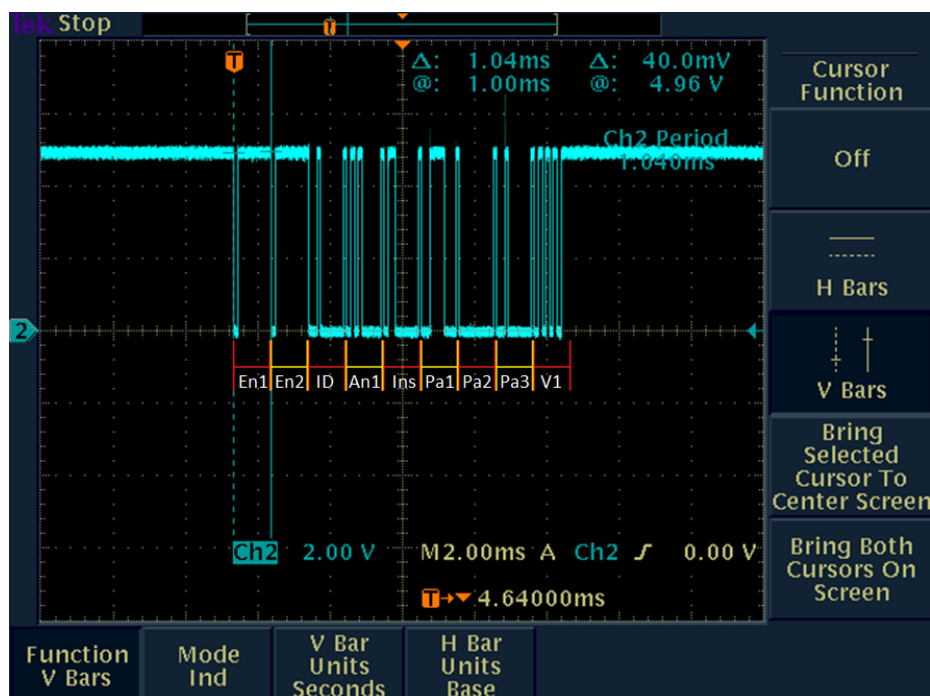


Figura 13: Ejemplo de salida de señal de control de motor AX12A

Con base en estos análisis se pudo determinar el número de bytes enviados y los valores de cada uno. En el caso de los motores AX-12A su paquete de instrucción cuenta con 9 bytes. 2 bytes de encabezado, 1 byte de ID, 3 bytes de instrucción, 2 bytes de parámetros de instrucción y 1 byte de verificación. Al igual que con los motores XL-320, con los motores AX-12A no se pudo determinar una manera para calcular el último byte de verificación.

Movimiento de motores MX-106T con arduino

Para el control de movimientos de los motores MX-106T con el microcontrolador arduino se utilizó la librería “Dynamixel-Serial-Master” [3]. Esta librería variaba en ciertos aspectos con lo visto anteriormente. Además de controlar posición del motor, con esta librería era posible controlar el valor de velocidad de movimiento del motor. Los parámetros que recibía esta función eran, ID del motor a controlar, Baud Rate en el que estaría trabajando, posición deseada y por último velocidad de movimiento.

Como primer paso se realizaron las mismas pruebas de funcionamiento y movimiento con los motores MX-106T, además de agregar pruebas de velocidad de movimiento. Todos los motores MX-106T con los que se trabajó funcionaban correctamente. En las pruebas de funcionamiento en serie todos los motores funcionaron de manera correcta indicando que todas las terminales de todos los motores estaban en buen estado.

En el caso de los motores MX-106T se decidió hacer el mismo análisis de la salida del arduino. Se conectó a la salida del pin TX del arduino la punta del osciloscopio y se

analizaron los bytes de salida. Para las pruebas de movimiento de los motores MX-106T no solo se varió el ID y la posición, sino que, también se realizaron pruebas variando la velocidad de movimiento del motor. Se trabajó con 5 valores de ID diferentes, 5 posiciones por ID, y 5 valores de velocidad por ID y por posición. Con base a estas pruebas se determinó el número de bytes enviados y el valor de cada byte en el paquete. Un ejemplo de este análisis se presenta en la Figura 14.

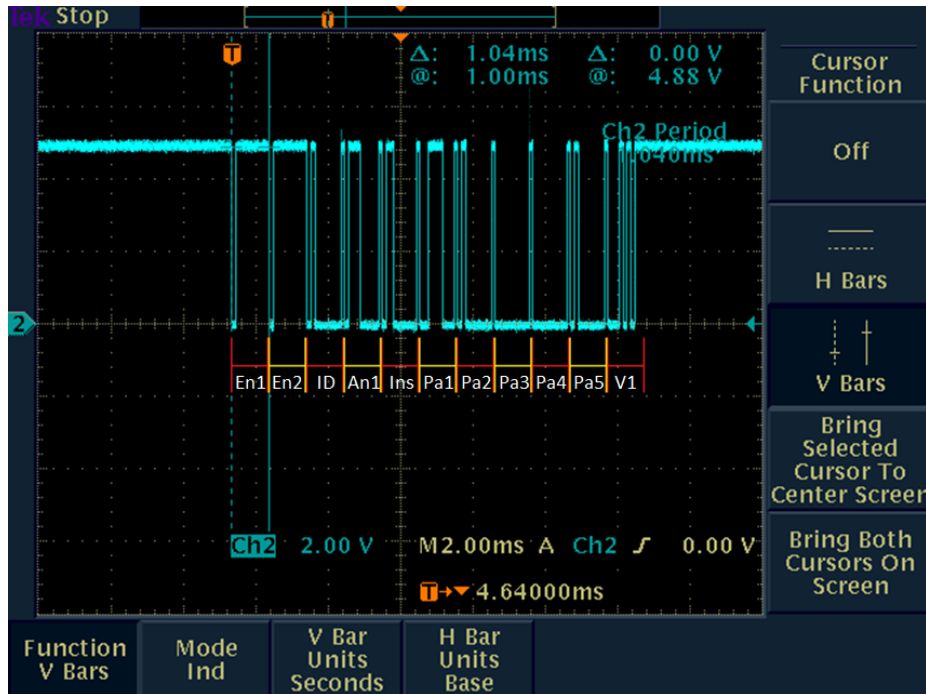


Figura 14: Ejemplo de salida de señal de control de motor MX106T

En el caso de los motores AX-12A su paquete de instrucción cuenta con 11 bytes. 2 bytes de encabezado, 1 byte de id, 1 byte de ancho de paquete, 1 byte de instrucción, 5 bytes de parámetros de instrucción y 1 byte de verificación. Al trabajar con los motores MX-106T, tampoco fue posible determinar una manera de calcular el byte de verificación al final del paquete.

7.1.3. Movimiento de motores Dynamixel con microcontrolador PIC16F887

En esta etapa del proyecto se desarrollaron códigos para el control de movimiento de los motores Dynamixel XL-320, Dynamixel AX-12A y Dynamixel MX-106T utilizando un nuevo microcontrolador. Este fue el PIC16F887, programado en el software MPLAB X. El hardware que se utilizó para programar los PICs y conectarlos a los motores fue el programador Easypic v7. En esta etapa además se contó con un nuevo equipo para análisis de señales digitales. Este fue el Saleae Logic Pro 16, proporcionado por el departamento de ingeniería electrónica de la Universidad del Valle de Guatemala. Esto fue de gran ayuda para acelerar el proceso de análisis ya que el Saleae cuenta con un analizador de protocolo serial incorporado que desplegaba el valor de cada byte recibido directamente en la computadora.

En el software MPLAB X se desarrollaron subrutinas de movimiento para los valores de posiciones encontrados en la etapa anterior. Esto se logró transmitiendo 1 a 1 los bytes del paquete de instrucción. Debido a que el número de bytes en los paquetes de instrucción de los 3 motores es diferente se tuvo que crear una subrutina independiente para cada tipo de motor. El ancho del paquete de instrucciones del motor XL-320 era de 14 bytes, el paquete de instrucción del motor AX-12A era de 9 bytes, y el paquete de instrucción del motor MX-106T era de 11 bytes. Después de una serie de pruebas se determinó que también era posible modificar el valor de la velocidad de movimiento de los motores AX-12A; por lo que el ancho de paquete de instrucción cambio de 9 a 11 bytes.

Una desventaja de las subrutinas creadas para el microcontrolador PIC16F887 era que cada motor solo podía trabajar con un número limitado de IDs y posiciones por ID. Esto presentaba un gran problema de diseño ya que limitaba el rango de movimientos que se podían realizar con los motores. Además, cada vez que se añadiera un nuevo motor al sistema se tendría que encontrar su valor de su byte de verificación. Con base en esto, se decidió hacer más pruebas e investigación acerca del protocolo de control de los motores Dynamixel, para encontrar una manera de calcular los bytes de verificación.

Bytes de verificación motor XL-320

Después de una segunda etapa investigación se encontró una manera de calcular los últimos 2 bytes de verificación para los motores XL-320. Analizando la documentación del protocolo de comunicación Dynamixel 2.0[17] se encontró un algoritmo usado para calcular los bytes de verificación del paquete. El problema con este algoritmo es que dependía de un arreglo de datos que no estaba detallado en esa documentación. Se tenía la idea de cómo calcular los bytes, pero faltaban datos para poder lograrlo. Posteriormente al consultar diferentes investigaciones realizadas con motores XL-320 se encontró una publicación[18] donde se hacía referencia a este arreglo de datos y se daba el detalle de cada uno de los valores utilizados.

Con base en la nueva información encontrada, se trabajó de nuevo en la subrutina de control de movimiento para los motores XL-320. Esta vez el código era capaz de generar todos los bytes del paquete de control solo con ingresar el ID del motor a controlar y la posición deseada. Esto permitió un rango mayor de posibles posiciones para los motores, además de poder ser utilizadas con cualquier ID diferente. Este código también funcionaba con múltiples motores XL-320 conectados en serie.

Byte de verificación motor AX-12A

En el caso de los motores AX-12A se tomó un diferente enfoque para encontrar una manera de calcular el último byte de verificación. Se decidió seguir haciendo pruebas con la librería de arduino que ya se tenía, hasta encontrar una manera de calcular este byte. Al contar con el Saleae el proceso de pruebas se facilitó considerablemente. Después de un segundo período de pruebas se encontró una manera de calcular el último byte del paquete. Con base a esto, se desarrolló un algoritmo para su cálculo.

Con esto se modificó la subrutina de control de movimiento de los motores AX-12A. Esta

versión de código le permitía al programa calcular todos los valores de bytes del paquete de control solo con ingresarle el valor de ID del motor a controlar y la posición deseada. Este código también funcionaba cuando múltiples motores eran conectados en serie.

Además, en un segundo periodo de pruebas de funcionamiento con el PIC16F887, se determinó que también era posible controlar la velocidad de movimiento de los motores AX-12A. Con base a esto, se realizaron las correcciones necesarias al código de control de movimiento de los motores AX-12A.

Byte de verificación motor MX-106T

Por último, para los motores MX-106T, se tomó el mismo enfoque que para los AX-12A. Se decidió continuar haciendo pruebas con el Saleae hasta encontrar una manera de calcular el último byte de verificación. Después de otra etapa de pruebas se logró desarrollar un algoritmo para calcular el valor del último byte del paquete de control.

Con este algoritmo se modificó el código que se había creado anteriormente. El nuevo código calculaba todos los bytes necesarios para el paquete de control, pero necesitaba de más entradas que los códigos anteriores. Esto es debido a que en los motores MX-106T es posible variar su velocidad de movimiento. Las entradas para esta subrutina fueron, ID del motor a controlar, posición deseada y velocidad de movimiento deseada. Esta subrutina permite un movimiento más natural y fluido en los motores ya que, además de tener un control total de la posición del motor y un rango más amplio de posiciones, es posible controlar la velocidad de movimiento del motor.

7.1.4. Comunicación con módulo de interacción por voz

Para esta etapa del proyecto se trabajó en conjunto con el módulo de interacción por voz. El objetivo de esto era llegar a un acuerdo de cómo se enviarían comandos de los módulos de programación de alto nivel al módulo de control de movimientos de los motores. Se decidió trabajar por medio de comunicación serial ya que era algo que ambos módulos podían transmitir y recibir, además de ser un protocolo del que ya se tenían conocimientos previos. Para esto se utilizó un FTDI232 conectado por USB a la computadora, y por cable a los pines TX, RX y GND del PIC16F887. Después de múltiples pruebas se logró establecer comunicación entre ambos. Se definió un set de banderas que se generarían dependiendo de la situación en el módulo de reconocimiento de comandos de voz. Estas generarían una acción específica en el movimiento de los motores dependiendo de la bandera recibida.

Se definió el mismo estándar para el módulo de reconocimiento facial y detección de gestos, pero no se realizó la misma cantidad de pruebas, ni se llevó al mismo nivel de profundidad que con el módulo anterior.

7.2. Diseño de hardware

En esta etapa del proyecto se determinó cuál sería la fuente de alimentación de los motores y microcontroladores. Luego, se diseñaron los circuitos de aislamiento entre la fuente de alimentación y las señales de control de los motores, en este caso los microcontroladores PIC16F887. Lo primero que se consideró fue diseñar y construir desde cero las fuentes de alimentación del sistema. Tras diálogos con el equipo de trabajo y reuniones con el director del proyecto se decidió adquirir una fuente de voltaje para alimentar el sistema en lugar de construir una. Para las pruebas dentro del laboratorio se utilizaron las fuentes de voltaje proporcionadas por la Universidad del Valle de Guatemala. Para la presentación de resultados se adquirieron las fuentes de voltaje X TECH CS850XTK09 y Delta Electronics Dpsn-50eb. La fuente CS850XTK09 fue utilizada para alimentar los motores del sistema. La fuente Dpsn-50eb sería utilizada para alimentar los microcontroladores, esto para que los motores y las señales de control no fueran alimentadas por la misma fuente.

Al conectar los motores Dynamixel directamente a la fuente de voltaje no es necesario implementar un driver que le proporcione la corriente necesaria, ya que puede tomarla directamente de la fuente. Lo mismo se aplica para los microcontroladores ya que su demanda de corriente es baja. En el software de simulación de circuitos Altium Designer se diseñaron las placas donde se colocarían los microcontroladores PIC16F887 para control de los motores. En la primera etapa las placas contaban con cuatro terminales de salida conectadas al pin TX del PIC, una entrada para el voltaje de alimentación del PIC y una terminal para parrear las tierras de la fuente de alimentación de los motores y la fuente de alimentación de los microcontroladores. Con esta placa se realizó la primera serie de pruebas con estructuras construidas por los demás integrantes del equipo de trabajo. Estas incluyeron pruebas de movimiento de cabeza, mano y antebrazo del animatrónico.

En la primera etapa de investigación se escogieron los optoacopladores 6N137 debido a que estos contaban con una velocidad de respuesta adecuada para seguir la señal de control. El problema con la señal de salida de los optoacopladores 6N137 es que está invertida con respecto a la señal de control. Esto se solucionó al conectar a la salida del optoacoplador una compuerta NOT implementada con un transistor 2N2222A. Esta compuerta normalizaría los voltajes de salida del optoacoplador, además de actuar como un seguidor de voltaje.

Con base a esto, se realizaron cambios a los diseños de las placas de control. Se redujo el número de terminales para motores de 4 a 3 y se agregó una terminal conectada al pin RX del PIC para recibir señales seriales. Se agregó el circuito del optoacoplador, el transistor y las resistencias de la compuerta NOT. Además, una terminal para la fuente de alimentación del circuito del optoacoplador y el circuito de la compuerta NOT. Por último, se ajustó el ancho de los tracks en las placas de 10m a 30mm. Este diseño de circuito se utilizó para todas las placas de control; mano, muñeca, hombro y cabeza.

7.3. Manufactura

La última etapa fue manufactura y pruebas de las placas con el sistema. El departamento de Ingeniería Electrónica de la Universidad del Valle de Guatemala cuenta con una

fresadora de circuitos donde se fresaron las placas para los microcontroladores y los circuitos de aislamiento eléctrico.

En la primera etapa se fresó una placa para pruebas de funcionamiento con las estructuras hechas para mano, muñeca, hombro y cabeza del animatrónico. Con esta se probó el funcionamiento del código creado para el control de movimiento de los motores. Esta versión de la placa era funcional pero no contenía una conexión directa al pin RX del PIC, este indispensable para la recepción de datos. Con base a esto se modificó el diseño de las placas para cumplir con todos los requerimientos del sistema, y se fresó nuevamente.

8.1. Diagramas de flujo

8.1.1. Subrutina setupEUSART

La Figura 15 describe los pasos a seguir para activar el módulo EUSART en el PIC16F887. Esta subrutina fue implementada en el software MPLAB X.

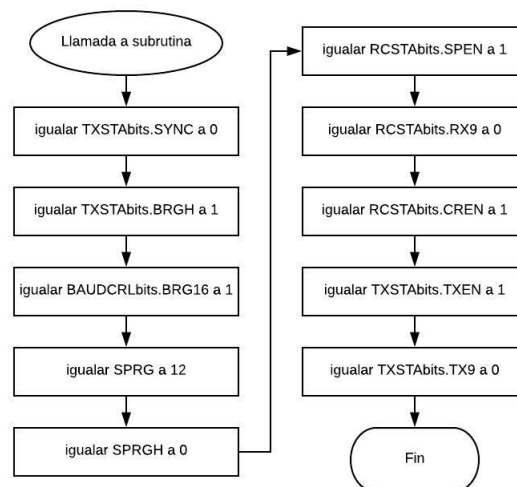


Figura 15: Subrutina para inicializar módulo EUSART

8.1.2. Subrutina transmisión EUSART

La Figura 16 es el diagrama de bloques que describe la subrutina creada para transmitir datos por medio de protocolo serial en el módulo EUSART del PIC16F887. Esta subrutina fue implementada en el software MPLAB X.

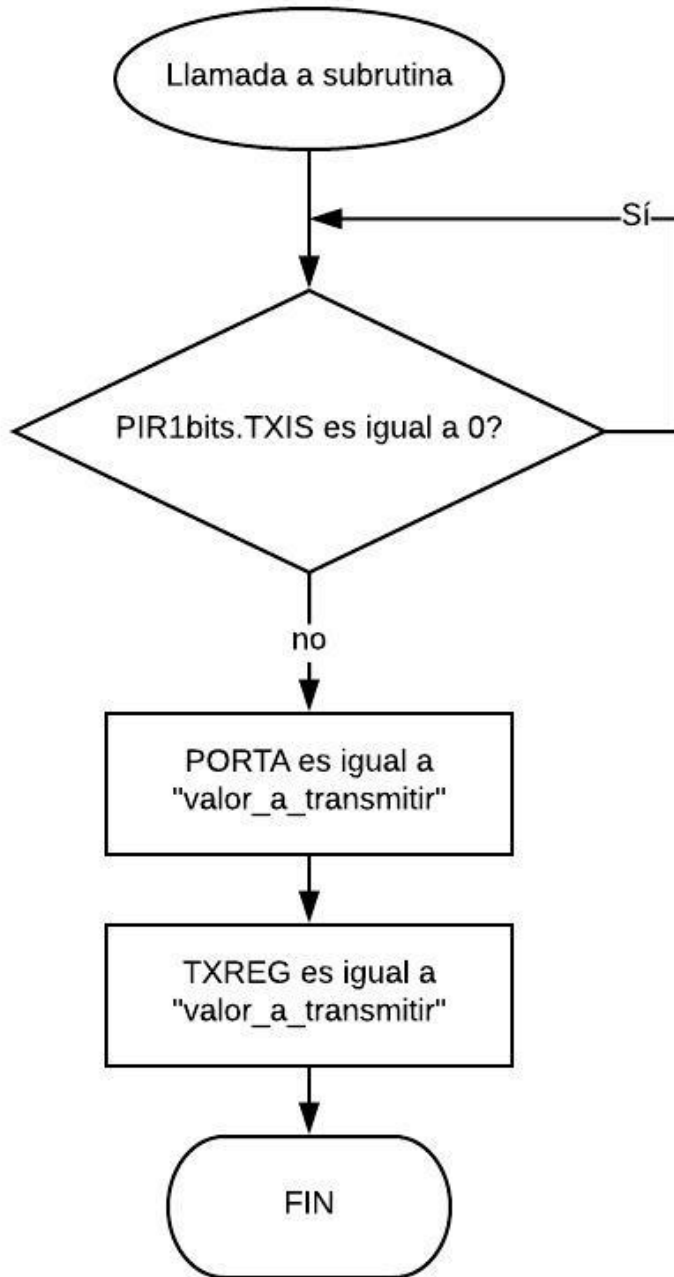


Figura 16: Subrutina para iniciar transmisión en módulo EUSART

8.1.3. Subrutina recepción EUSART

La Figura 17 es el diagrama de bloques que describe la subrutina creada para recibir datos por medio de protocolo serial en el módulo EUSART del PIC16F887. Esta subrutina fue implementada en el software MPLAB X.

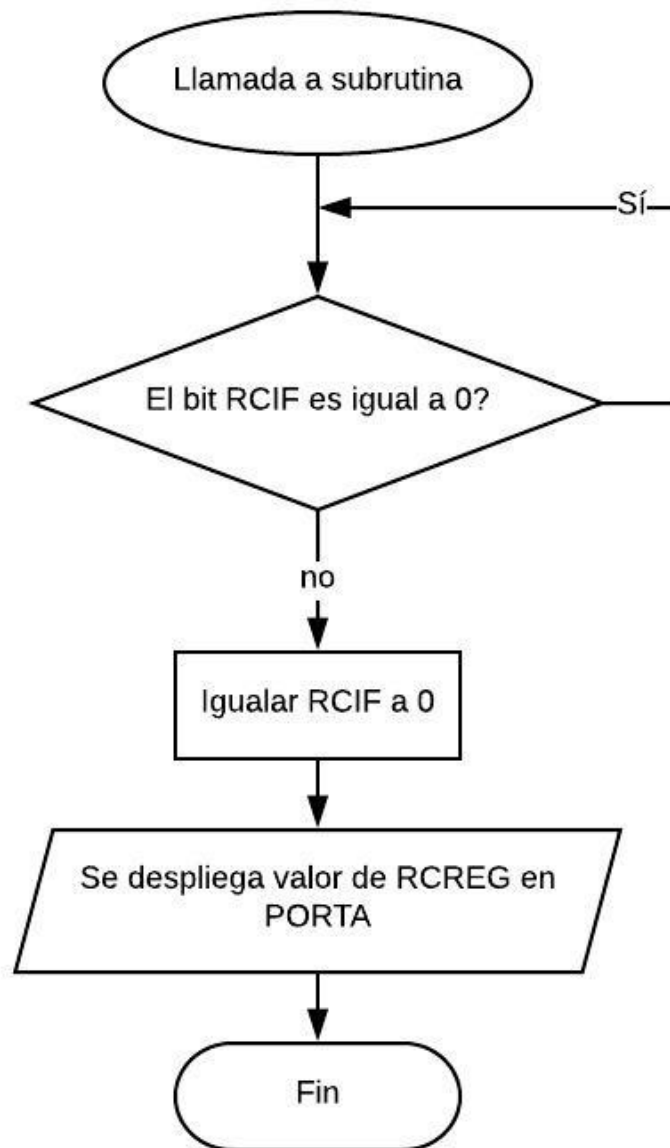


Figura 17: Subrutina para iniciar recepción en módulo EUSART

8.1.4. Subrutina de movimiento de motores XL-320

La Figura 18 es el diagrama de bloques que describe la subrutina creada para el movimiento de los motores XL-320. Esta hace referencia a la Figura 16 y a la Figura 19. Como las demás subrutinas, esta fue hecha en MPLAB X e implementada en microcontroladores PIC16F887.

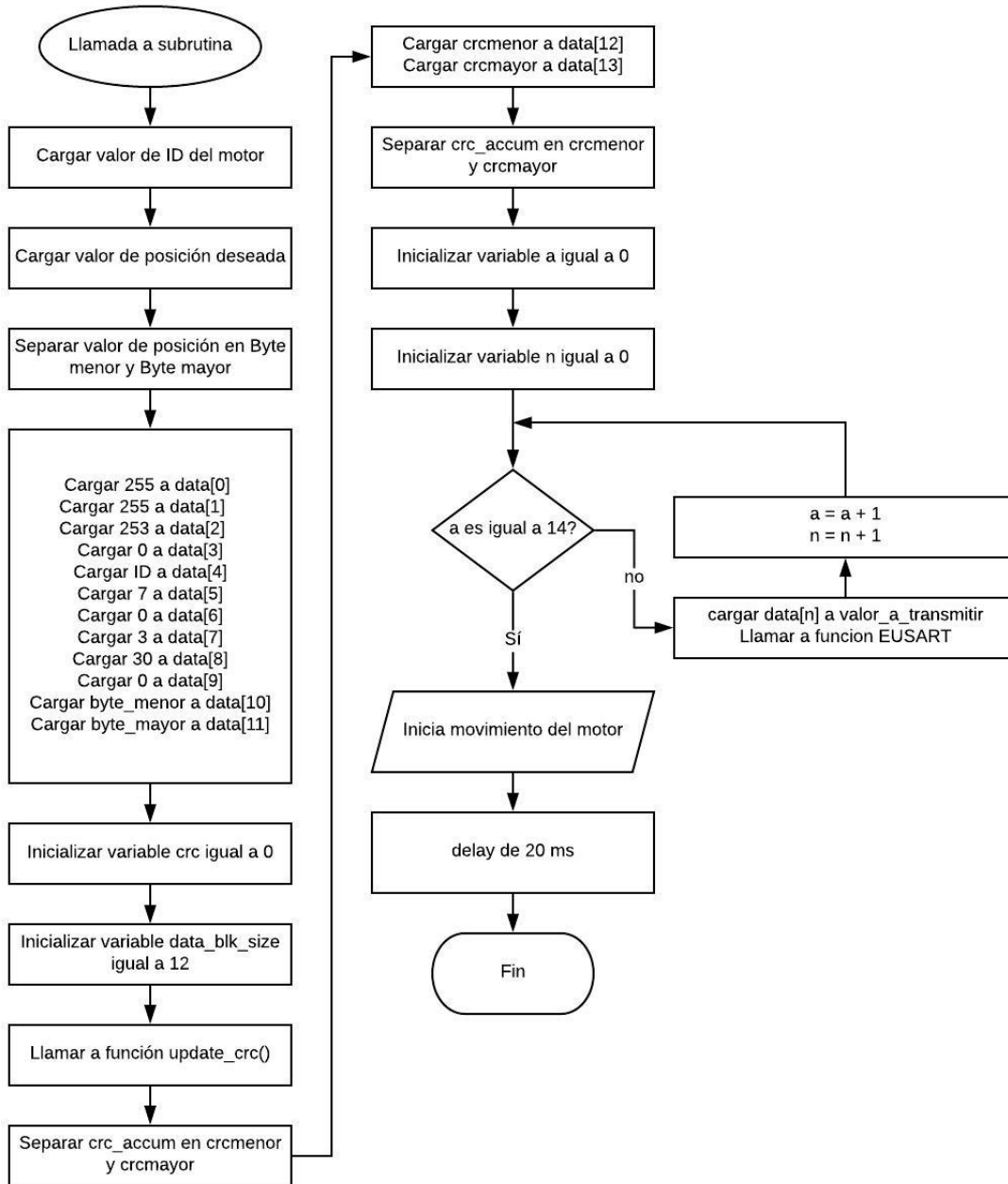


Figura 18: Subrutina para movimiento de motores XL-320

8.1.5. Subrutina cálculo de CRC

La Figura 19 es el diagrama de bloques que describe la subrutina realizada para el cálculo del valor del byte de verificación de los motores XL-320. Esta subrutina se utiliza en la Figura 18. Como las demás subrutinas, esta fue desarrollada en el software MPLAB X e implementada en microcontroladores PIC16F887.

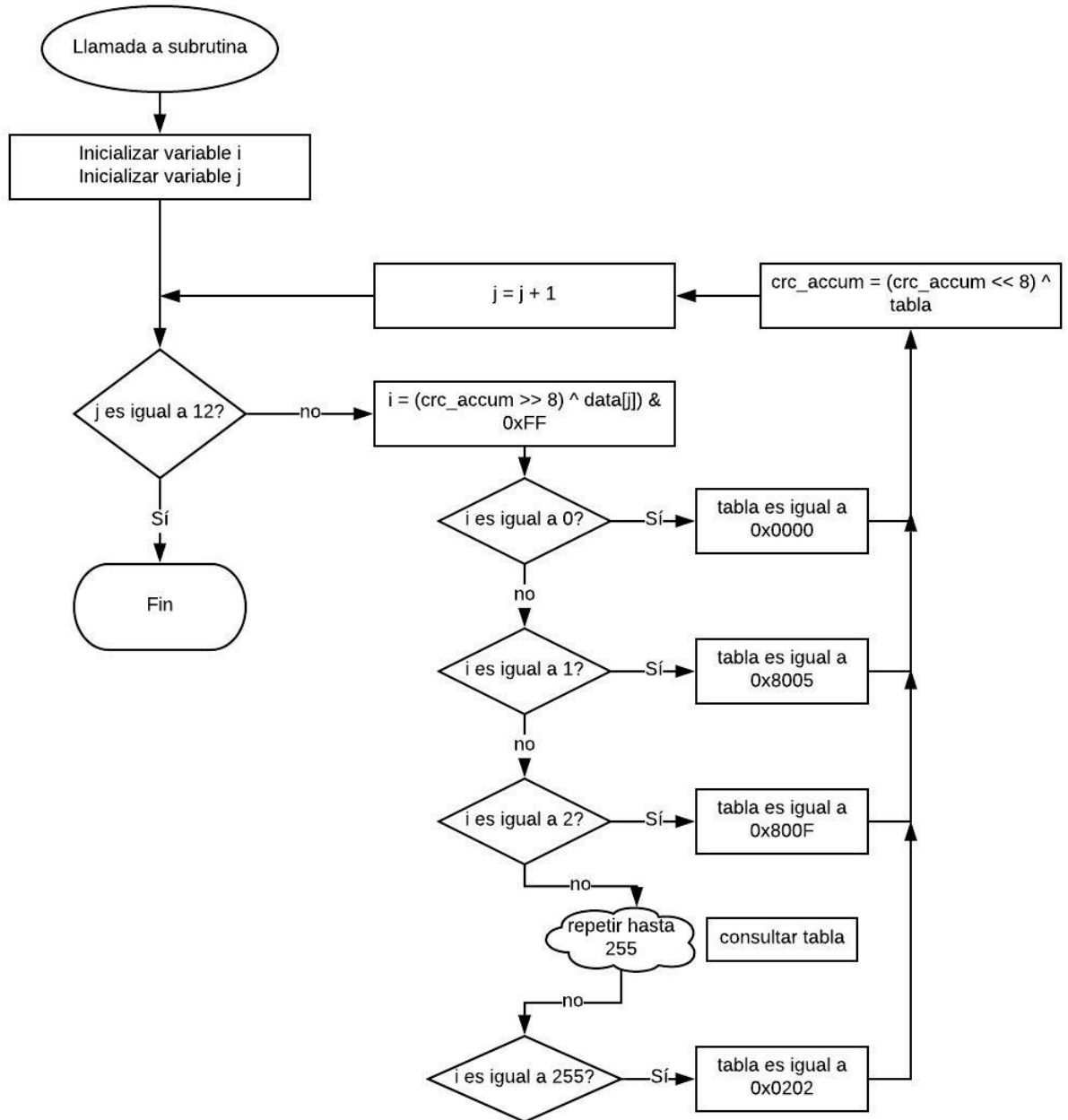


Figura 19: Subrutina para el cálculo de CRC

8.1.6. Subrutina de movimiento de motores AX-12A

La Figura 20 es el diagrama de bloques que describe la subrutina creada para el movimiento de los motores AX-12A. Esta hace referencia a la Figura 16. Como las demás subrutinas esta fue elaborada en MPLAB X e implementada en microcontroladores PIC16F887.

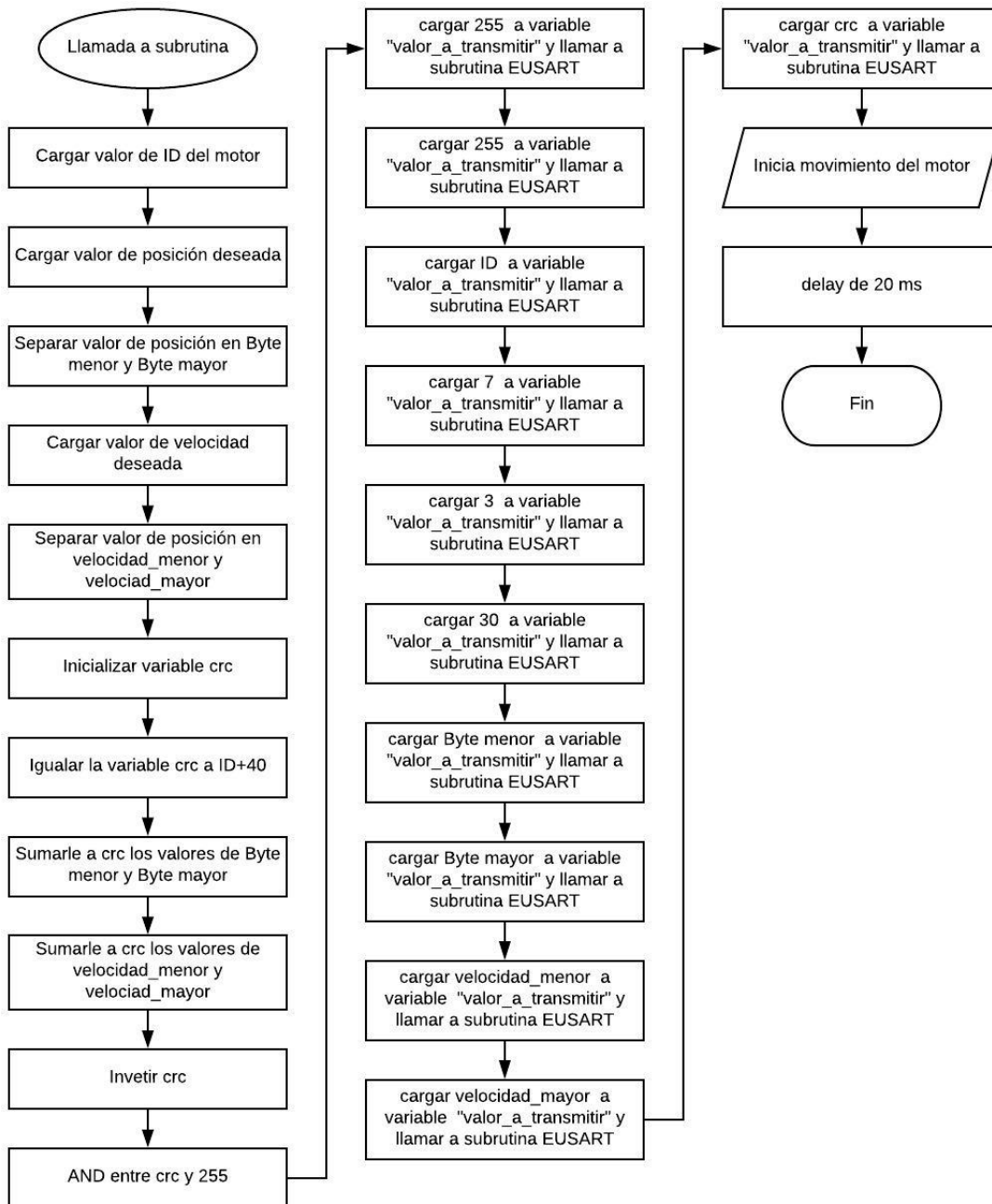


Figura 20: Subrutina para movimiento de motores AX-12A

8.1.7. Subrutina de movimiento de motores MX-106T

La Figura 21 muestra el diagrama de bloques que describe la subrutina creada para el movimiento de los motores MX106T. Esta hace referencia a la Figura 16. Como las demás subrutinas, esta fue elaborada en MPLAB X e implementada en microcontroladores PIC16F887.

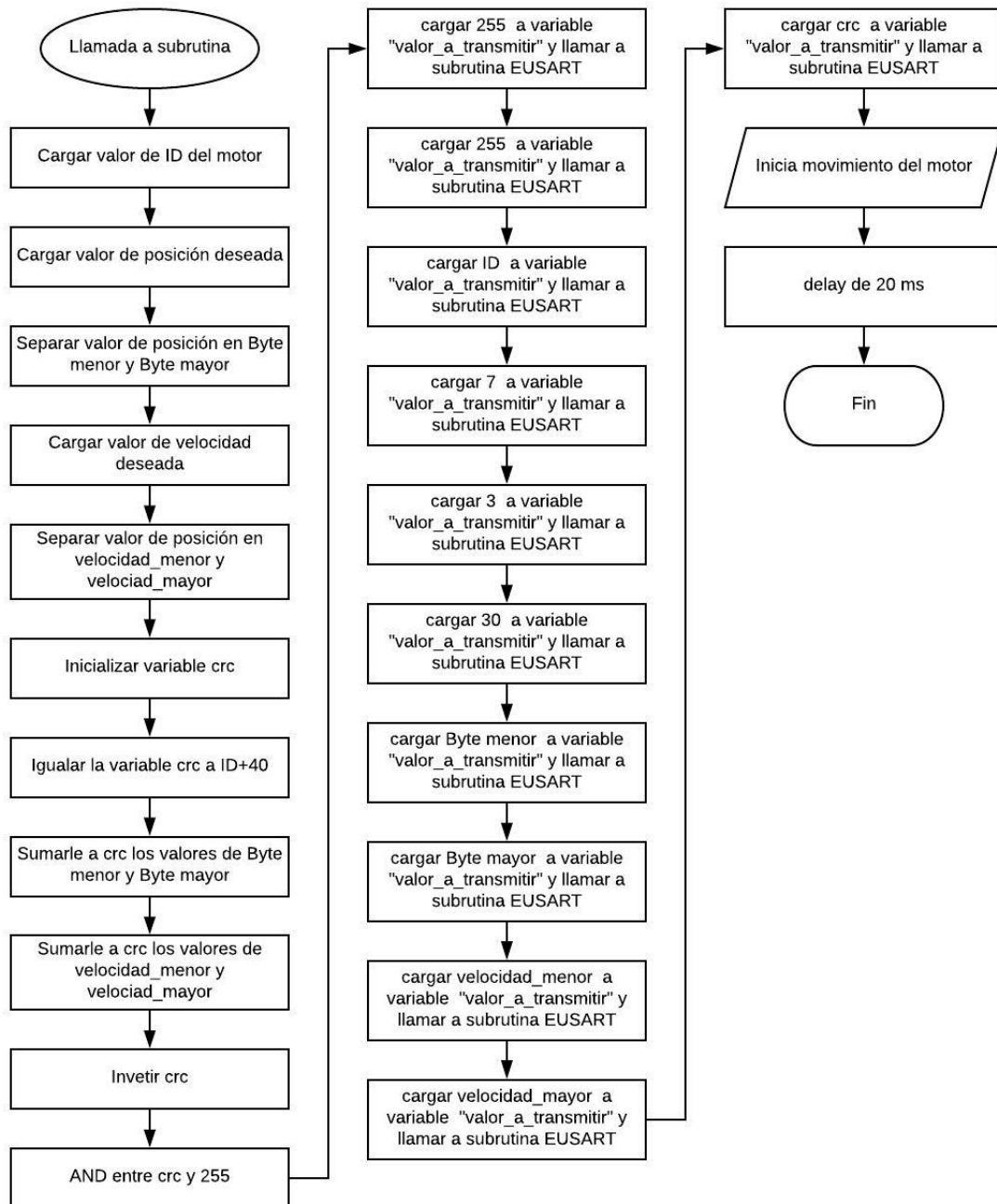


Figura 21: Subrutina para movimiento de motores MX-106T

8.2. Paquetes Transmitidos

En los siguientes cuadros se muestra el número de bytes y su valor transmitido para cada paquete de control de posición de los motores Dynamixel. El Cuadro 7 muestra los bytes de control de los motores Dynamixel XL-320, El Cuadro 8 los bytes de los motores AX-12A, y por último, el Cuadro 9 el paquete de control de los motores MX-106T.

8.2.1. Bytes en paquete transmitido para control de motor XL-320

XL-320	
Byte	Valor
Encabezado 1	255
Encabezado 2	255
Encabezado 3	253
Encabezado 4	0
ID	Valor de ID
Ancho de paquete 1	7
Ancho de paquete 2	0
Instrucción	3
Parámetro 1	30
Parámetro 2	0
Parámetro 3	Posición LS byte
Parámetro 4	Posición MS byte
Verificación 1	CRC calculado
Verificación 2	CRC calculado

Cuadro 7: Bytes de paquete de control de motor XL-320

8.2.2. Bytes en paquete transmitido para control de motor AX-12A

AX-12A	
Byte	Valor
Encabezado 1	255
Encabezado 2	255
ID	Valor de ID
Ancho de paquete	7
Instrucción	3
Parámetro 1	30
Parámetro 2	Posición LS byte
Parámetro 3	Posición MS byte
Parámetro 4	Velocidad LS byte
Parámetro 5	Velocidad MS byte
Verificación	CRC calculado

Cuadro 8: Bytes de paquete de control de motor AX-12A

8.2.3. Bytes en paquete transmitido para control de motor MX-106T

MX-106T	
Byte	Valor
Encabezado 1	255
Encabezado 2	255
ID	Valor de ID
Ancho de paquete	7
Instrucción	3
Parámetro 1	30
Parámetro 2	Posición LS byte
Parámetro 3	Posición Ms byte
Parámetro 4	Velocidad LS byte
Parámetro 5	Velocidad Ms byte
Verificación	crc calculado

Cuadro 9: Bytes de paquete de control de motor MX-106T

8.3. Circuitos diseñados

8.3.1. Circuito diseñado para placas de control

La Figura 22 presenta el diseño del circuito para placas de control de movimiento de los motores Dynamixel, este dividido en 4 secciones. El módulo de control cuenta con el PIC16F887, el circuito de aislamiento es el optoacoplador y la compuerta NOT, las entradas son las fuentes de alimentación, la terminal para parrear las tierras de las fuentes y el pin RX; por último las salidas son las conexiones que van a los motores Dynamixel.

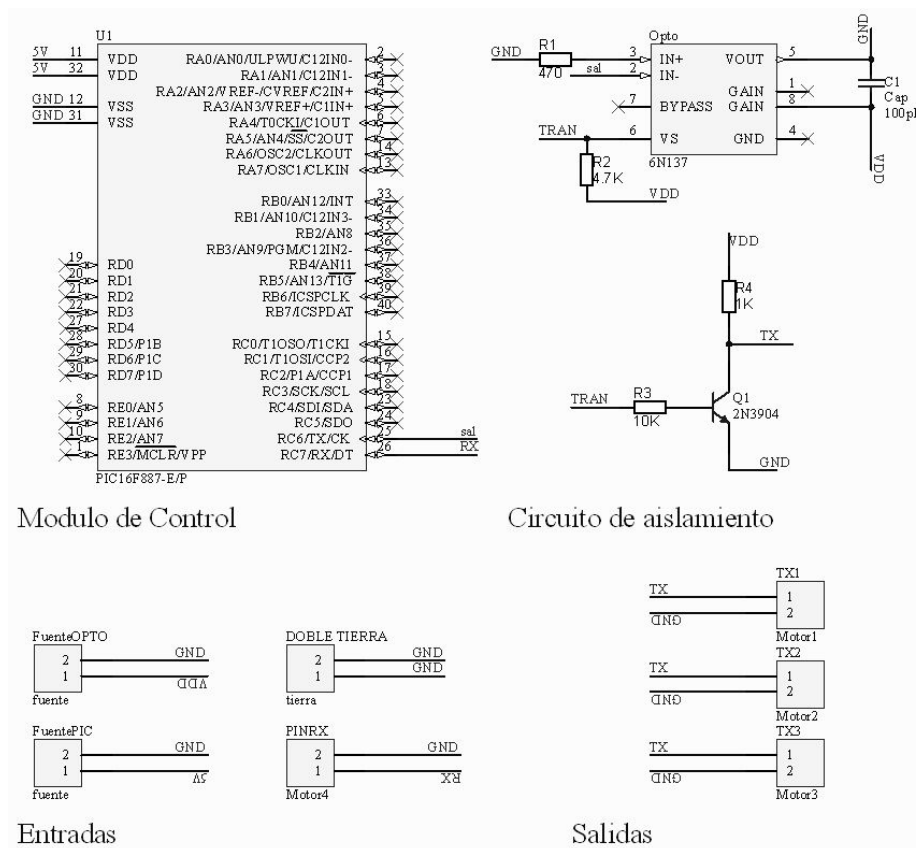


Figura 22: Circuito diseñada para placas de control

La Figura 23 muestra las conexiones entre los componentes colocados en la placa de control. Estas conexiones son el circuito de la figura 22 ya colocados en la placa.

Banderas recibidas	
Valor	Instrucción
0	Terminar conversación
1	Iniciar conversación
2	Mover la muñeca
3	Apuntar derecha
4	Apuntar arriba
5	Apuntar izquierda
6	Asentir
7	Negar
8	Pregunta
9	Confusión
10	Mascara
11	Pistola
12	Oreja
13	Música
14	Pulgar arriba
15	Señal de paz
16	Cara Neutral
17	Felicidad
18	Tristeza
19	Enojo

Cuadro 10: Banderas de instrucción transmitidas por el módulo de interacción por voz

8.4.2. Diagrama de conexiones de módulo de comunicación con módulo de interacción por voz

La Figura 67 muestra las conexiones del módulo de comunicación entre entre los micro-controladores y el módulo de interacción por voz.

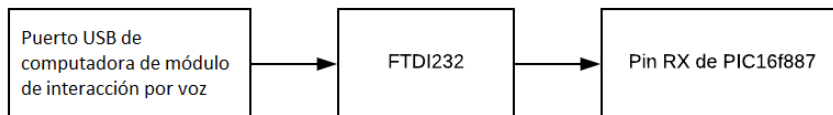


Figura 24: Diagrama de conexiones entre PICs y módulo de interacción por voz

9.1. Subrutinas del módulo EUSART

En esta sección se discutirán las subrutinas importantes, principalmente de transmisión y recepción de datos con el módulo EUSART, mostradas en los diagramas de flujo de la sección de resultados.

9.1.1. Subrutina EUSART

La subrutina EUSART es utilizada para enviar un byte de información por protocolo serial utilizando el módulo EUSART incluido en el PIC16F887. Esta subrutina se encuentra en la Figura 16. Esta es utilizada en todas las subrutinas de movimiento de motores para transmitir el paquete de control de posición. Cuando se llama a la subrutina lo primero que se consulta es el valor del bit TXIF del registro PIR1 para saber el estatus del módulo EUSART. Si la bandera está en 1 significa que aún no ha terminado la transmisión anterior, por lo que hay que esperar a que esta termine o se puede corromper el valor que se está transmitiendo. Cuando esta bandera es igual a 0 se puede continuar. El valor de la variable valor a transmitir se carga en el registro TXREG y se despliega en el puerto A para saber que se está transmitiendo el valor correcto.

9.1.2. Subrutina recepción EUSART

La subrutina recepción EUSART es utilizada para recibir bytes de información enviados por el módulo de Interacción por Voz. Los datos son enviados por protocolo serial y recibidos por el módulo EUSART del PIC16F887. Esta subrutina se muestra en la Figura 17. Cuando

se llama a la subrutina lo primero que se revisa es el valor de la bandera RCIF. Esta monitorea el estado de recepción de datos del módulo EUSART. Si la bandera es igual a 0 se espera debido a que el módulo aún no está libre. Cuando el valor cambia a 1 esta tiene que limpiarse en software. Después se lee el valor del registro RCREG y se hace la acción respectiva dependiendo de la bandera recibida.

9.2. Análisis de bytes para movimiento de motores

Como se discutió en la sección de metodología, el desarrollo de las subrutinas de control de movimiento para los tres motores se basó en el análisis de las salidas generadas por un Arduino. Esto para entender la estructura del paquete de control utilizado para controlar a los motores Dynamixel. Las fotos de estas pruebas están disponibles en la sección de anexos.

9.2.1. Motores Dynamixel XL-320

Subrutina de movimiento de motores Dynamixel XL-320

La Figura 18 muestra el diagrama de flujo que explica la subrutina utilizada para el movimiento de los motores XL-320. La subrutina toma como parámetros de entrada el ID del motor que queremos controlar, y la posición deseada. El ID es un valor entre 0 y 252 para los motores XL-320 y es escrito en solo 1 byte. El valor de la posición puede variar entre 0 y 1023 por lo que se necesitan 2 bytes para poder escribir la posición deseada. Dentro de la subrutina se hace la separación de los 8 bits menos significativos, que son cargados a la variable Byte menor, y los 8 bits más significativos, que son cargados a la variable Byte mayor. Luego tenemos que cargar los valores de el Cuadro 7 a un arreglo llamado data e inicializar la variable CRC en 0. Con esto ya se puede llamar a la función update CRC 19. La función update CRC nos regresa el valor CRC acumulado donde están contenidos los 2 bytes de verificación que necesitamos transmitir al final del paquete de control. El valor de CRC acumulado puede alcanzar hasta 16 bits por lo que hay que separarlo en 2 bytes. Estos valores se añaden al arreglo data en las posiciones 12 y 13.

Lo siguiente es la transmisión de los 14 bytes seguidos por medio del módulo EUSART. Este proceso se hace cargando uno por uno los valores del arreglo data a la variable valor a transmitir y luego llamando a la función EUSART 16. Después de transmitidos los 14 bytes inicia el movimiento del motor. Se aprendió durante las pruebas realizadas con el Saleae que cada byte tiene un ancho de 1.04 milisegundos, tomando eso en consideración se añade un delay al final del diagrama de por lo menos 20 milisegundos ya que aproximadamente esto es lo que tardará todo el paquete en ser transmitido.

Subrutina de cálculo de CRC

La subrutina de cálculo de CRC 19 comienza inicializando dos variables, “i” y “j”. La variable “j” es utilizada como contador dentro del ciclo FOR, y a la variable “i” se utiliza para encontrar el valor de la variable tabla más adelante. Dentro del ciclo FOR se realiza

una operación OR de bits entre los 8 bits más significativos de la variable CRC accum y el byte del arreglo data correspondiente a esta iteración ciclo del FOR. Con el resultado de esta operación se lleva a cabo una operación AND de 8 bits con el número 255, y es almacenado en la variable “i”. Como se discutió anteriormente el PIC16F887 solo cuenta con 256 bytes de memoria para nuestro programa. Para calcular el valor de CRC es necesario una tabla de 256 elementos de 2 bytes cada elemento, y el PIC16F887 no cuenta con esta capacidad de memoria. Para no tener que ingresar una memoria externa donde cupieran los datos, que aumentaría considerablemente la complejidad y el tiempo de respuesta del sistema, se optó por hacer 255 comparaciones diferentes del valor de “i” para poder asignarle un valor a la variable tabla. Después de las 255 comparaciones se realiza una operación OR de bits entre el valor de los 8 bits más significativos de la variable CRC accum y la variable tabla. El resultado es escrito en la variable CRC accum, se le suma 1 al contador y se repite el mismo proceso para cada elemento del arreglo data. Cuando esto se cumple se obtiene el valor final de CRC accum, necesaria en la subrutina 18 de movimiento de los motores XL-320.

9.2.2. Motores Dynamixel AX-12A

En la Figura 20 se muestra el diagrama de flujo que describe la subrutina para el movimiento de los motores AX-12A. Esta subrutina toma como parámetros de entrada el valor de ID del motor a controlar, el valor de la posición deseada y el valor de la velocidad de movimiento deseada. Los motores AX-12A pueden moverse 300 grados que se mapea de 1023 posiciones diferentes. Se hace el mismo proceso de separación en 2 bytes que se hizo en la subrutina de los motores XL-320 para los valores de posición y velocidad deseados. Para el paquete de control de los motores XL-320 también es necesario hacer un cálculo de CRC, este diferente y de menor complejidad al de los motores XL-320 19. Se inicializa la variable CRC en 0, a esta se le suma el valor de ID del motor y un valor de 40. También se le suman los valores de los bytes de posición y velocidad deseadas, por último, se invierte CRC. Se realiza una operación AND de bits entre la variable CRC y el valor 255 para asegurarnos que el resultado solo este en 8 bits. Este proceso se determinó después de una serie de pruebas con el Saleae en las cuales se analizaron diferentes combinaciones de ID y posición deseada hasta encontrar un patrón en el último byte de verificación del paquete de control.

Por último, se transmiten los 11 bytes del paquete de control por medio del módulo EUSART del PIC16F887. Se carga uno por uno los valores de la tabla 8 y se llama a la función EUSART 16. Después de transmitir el paquete de control correctamente iniciará el movimiento del motor. El diagrama 20 también muestra un delay de 20 milisegundos al final de la operación.

9.2.3. Motores Dynamixel MX-106T

La Figura 21 muestra el diagrama de bloques que describe la subrutina de control de movimientos de los motores MX-106T. Esta subrutina toma como parámetros de entrada el valor de ID del motor a controlar, la posición deseada, y el valor de la velocidad de movimiento deseada. Los valores de posición y velocidad deseados tienen que ser separados en 2 bytes cada uno como se ha hecho anteriormente. Se inicializa la variable CRC con valor igual a cero. A esta se le suma el valor de ID del motor a controlar, los 2 bytes de posición,

los bytes de velocidad y por último un valor de 40. El resultado se invierte y se realiza una operación AND con 255 para que el resultado sea de 8 bits. Al igual que con los motores AX-12A este proceso se encontró después de una serie de pruebas con el Saleae analizando la salida del arduino.

Por último, se da la transmisión de los bytes del paquete de control por medio del módulo EUSART del PIC16F887. Se transmite uno por uno los 11 valores de la tabla 9 utilizando la función EUSART 16. Como los demás diagramas este también termina con un delay de 20 milisegundos debido al tiempo que tarda el paquete de datos en transmitirse.

9.3. Análisis de bytes transmitidos para movimiento de motores

9.3.1. Bytes transmitidos para motores Dynamixel XL-320

En el Cuadro 7 se muestra el orden y los valores de los bytes enviados por protocolo serial en el paquete de control de los motores XL-320. Como se dijo anteriormente el paquete consta de 14 bytes enviados consecutivamente. Los primeros 4 bytes son de encabezado, independientemente de la instrucción o parámetros estos bytes siempre tendrán el mismo valor. Los valores son 255, 255, 253 y 0 para los bytes encabezado 1, encabezado 2, encabezado 3 y encabezado 4 respectivamente. El quinto valor transmitido es el ID del motor que nos sirve para distinguir a que motor se está dirigiendo la instrucción. Este es solo 1 byte ya que el valor de id del motor pueden ser de 0 a 252. Luego se esto, siguen los 2 bytes de ancho de paquete, el byte 6 contiene el valor de los 8 bits menos significativos del ancho y el byte 7 los 8 bits más significativos. El ancho de paquete se refiere al número de bytes enviado después de este parámetro. En este caso el valor es 7 que a este le siguen 7 bytes para completar el paquete de instrucción.

El siguiente valor por transmitir es el byte de instrucción. La instrucción de escritura en memoria en los motores Dynamixel es la número 3, por lo que los valores del byte de instrucción en el caso de los 3 tipos de motores serán iguales a 3. Los parámetros 1 y 2 se refieren a la dirección de memoria del motor donde queremos escribir. Según el Cuadro 4, la dirección de memoria de posición deseada es la numero 30, por lo que escribe 30 como valor para el byte menos significativo y 0 para el más significativo.

En los bytes 11 y 12, que representan los parámetros 3 y 4, se escribe el valor de la posición deseada. Son necesarios 2 bytes para el valor de la posición ya que los motores XL-320 son capaces de un movimiento de 300 grados que se mapea a 1023 posiciones diferentes; y el valor 1023 no puede ser escrito en un byte. Por último, se escriben los 2 bytes de verificación. Estos se pueden encontrar utilizando la función de cálculo de CRC que es discutida en la sección 9.2.1. Ya que este valor puede llegar a ser mayor de 255 se necesitan 2 bytes para su escritura. El byte 13 es el menos significativo y el 14 es el más significativo.

9.3.2. Bytes transmitidos para motores Dynamixel AX-12A

El Cuadro 8 muestra el orden y los valores de los bytes enviados por protocolo serial en el paquete de control de posición los motores AX-12A. Este cuenta con 11 bytes que serán transmitidos por protocolo serial. Al inicio se contaba con solo 9 bytes para el paquete de control, pero después de más pruebas realizadas se concluyó que también era posible modificar la velocidad de movimiento de los motores, por lo que el ancho del paquete cambio de 9 a 11. A diferencia de los XL-320, el paquete de control de los motores AX-12A solo cuenta con 2 bytes para encabezado, ambos con valor de 255. El sigue el byte es el valor de ID del motor que se quiere controlar. Este puede variar de 0 a 253, por lo que solo es necesario un byte para el ID. Luego se transmite el ancho de paquete, que también es de solo un byte. El valor de este es igual a 7, ya que después de este se transmiten 7 bytes y se termina el paquete.

El byte de instrucción sigue siendo igual a 3, ya que 3 es la instrucción de escritura en los motores Dynamixel. El byte número 6 es la dirección de memoria que se quiere escribir. En el caso de los motores AX-12A la dirección de memoria 30 contiene el valor de la posición deseada, por lo que esta es la dirección en la que queremos escribir. Los parámetros 2 y 3, correspondientes a los bytes 7 y 8, contienen los valores de posición deseada. Los parámetros 4 y 5 son los bytes que contienen los valores de la velocidad de movimiento del motor, asignados a los bytes 9 y 10. Por último, el número 11 es el byte de verificación calculado.

9.3.3. Bytes transmitidos para motores Dynamixel MX-106T

El Cuadro 9 muestra el orden y los valores de los bytes enviados por medio de protocolo serial en el paquete de control de los motores MX-106T. Como se discutió anteriormente este paquete cuenta con 11 bytes enviados consecutivamente. Al igual que con los motores AX-12A, los primeros 2 byte transmitidos son de encabezado; ambos con un valor de 255. Le sigue el valor de ID del motor a controlar. Solo es 1 byte transmitido ya que el valor de ID de los motores MX-106T solo puede variar entre 0 y 253. El byte número 4 es el ancho de paquete, este es igual a 7 ya que en el resto del paquete se envían 7 bytes.

El byte número 5 es igual a 3, debido a que la instrucción de escritura en los motores Dynamixel es la número 3. El parámetro 1 nuevamente es la dirección de memoria que queremos escribir, en este caso sigue siendo 30. Los parámetros 2 y 3 son los valores de los bytes de posición. A este paquete se le agregan los bytes de parámetro 4 y 5, los cuales toman el valor de la velocidad de movimiento deseada. El parámetro 4 es el byte de velocidad menos significativo y el 5 el más significativo. Por último, tenemos el byte número 11, este es igual al valor de verificación calculado.

9.4. Selección de fuentes de alimentación

Como se discutió anteriormente, se utilizó a fuente de alimentación X TECH CS0850XTK09 para la alimentación de los de los motores Dynamixel, y los circuitos de aislamiento eléctrico. Esto debido a que la fuente cuenta con canales de salida de 5 y 12 voltios, los mismos

voltajes de alimentación de los motores Dynamixel. Ambas salidas pueden otorgar hasta 30 amperios de corriente a su carga máxima de 484W. El Cuadro 1 muestra que el voltaje de operación de los XL-320 es de 5V, y la corriente máxima de alimentación es de 900mA. El voltaje de alimentación de los AX-12A es de 12V, y su corriente máxima es de 900 mA, esto se puede apreciar en el Cuadro 2. Por último, en el Cuadro 3 se denota que el voltaje de operación de los MX-106T es de 12V, y la corriente máxima de alimentación es de 5.2A.

Además, la fuente X TECH CS0850XTK09 alimentó los circuitos de aislamiento eléctrico del sistema. Tanto el optoacoplador 6N137 como los transistores 2N2222A operaban a 5V con necesidades de corriente menores a la de los motores.

La fuente Delta Electronics Dpsn-50eb fue utilizada para la alimentación de los microcontroladores PIC16F887. Esta contaba con dos canales de 5 voltios de salida, capaces de suministrar hasta 9.5A de corriente. Según el Cuadro 5 la corriente máxima de alimentación que pueden demandar los microcontroladores es de 220uA. La fuente Dpsn-50eb era capaz de suministrar esta corriente a todos los microcontroladores del sistema.

El propósito de tener dos fuentes de alimentación es que cualquier ruido o perturbación eléctrica ocasionada por los motores no afecte directamente a los microcontroladores.

9.5. Análisis del circuito de aislamiento eléctrico

Para los circuitos de aislamiento eléctrico se utilizaron optoacopladores 6N137. Se eligieron estos debido a su velocidad de respuesta y a experiencia previa con ellos. El diagrama del circuito puede apreciarse en la Figura 22. La salida del pin TX del microcontrolador es de 5 voltios a un máximo de 25mA, según la data sheet del 6N137 la máxima corriente que soporta el LED del optoacoplador es de 20 mA y en base a eso se hicieron los cálculos para la resistencia del LED.

$$R_{led} = \frac{5V}{20mA} = 250ohms \quad (3)$$

Como medida de seguridad se escogió un valor de 470 ohms para asegurarnos de no alcanzar una corriente mayor a la que el LED del optoacoplador pudiera soportar. Para la resistencia de salida del optoacoplador se escogió el valor del 4.7 ohms.

9.6. Análisis de comunicación entre PICS y módulo de interacción por voz

El Cuadro 10 muestra las banderas, y la instrucción que representa cada bandera, definidas para el módulo de comunicación con el módulo de Interacción por Voz. Se definieron 20 banderas con valores ascendentes de 0 a 19. Estas son transmitidas por medio de protocolo serial a un Baud Rate de 9600bps con la ayuda de un FTDI232. El FTDI232 actuó como puente entre la computadora utilizada para el módulo de Interacción por Voz y los microcontroladores del sistema. Esta conexión se puede apreciar en la figura 67.

10.1. Conclusiones de diseño de subrutinas de movimiento

- Se desarrolló subrutinas de control de movimiento para motores Dynamixel XL-320, AX-12A y MX-106T.
- Se desarrolló subrutinas de control de velocidad de movimiento para los motores Dynamixel AX-12A y MX-106T.
- Se desarrolló una subrutina de cálculo de CRC y bytes de verificación del paquete de control para todos los motores del sistema.
- Las subrutinas creadas pueden ser utilizadas en códigos para controlar múltiples motores a la vez, no necesariamente de la misma clase.
- Se determinó en pruebas realizadas con módulos de cabeza, mano, muñeca y hombro que la variación de velocidad y posición genera un movimiento realista y fluido.
- Se pudo establecer comunicación con módulos de interacción por voz y reconocimiento facial y detección de gestos, por medio de un FTDI232 y subrutinas diseñadas para su uso.

10.2. Conclusiones del diseño del módulo de alimentación

- Al conectar directamente los motores a la fuente de voltaje no fue necesario el desarrollo de drivers de potencia ya que la fuente de alimentación puede entregar hasta 30 amperios de corriente, de 2.5 amperios necesarios para alimentar todo el sistema.
- Se seleccionaron los optoacopladores 6N137 por su velocidad de respuesta al cambio de flanco, siendo esta hasta 100nS comparada con los optoacopladores 4n25 que tenían una respuesta de 2.5uS.

- Fue posible satisfacer las necesidades de voltaje y de corriente de todos los servo motores del sistema.
- Se diseñaron y fresaron placas para los microcontroladores PIC16F887, utilizados para el control de movimiento y comunicación de motores Dynamixel.

11.1. Recomendaciones generales

- Investigar más a fondo el uso de instrucciones en los motores Dynamixel.
- Investigar acerca de la transmisión de datos desde los motores Dynamixel hacia un microcontrolador.
- Volver más eficiente las subrutinas de cálculo de CRC para que pueden trabajar con cualquier tipo de instrucción.
- Buscar una manera de guardar tablas de calculo de CRC dentro del código para usarlas en ciclos y no solo en condicionales.

-
- [1] H. embedded Dev, *Dynamixel XL-320 - A Servo library for Arduino*, 2015. dirección: <https://github.com/hackerspace-adelaide/XL320> (visitado 12-09-2018).
 - [2] ThingType, *Using Dynamixel AX-12A Servo with Arduino*, 2018. dirección: <http://jume-maker.blogspot.com/2018/01/using-dynamixel-ax-12a-servo-with.html> (visitado 14-09-2018).
 - [3] Optimuspi, *Dynamixel MX-106T + Arduino UNO - optimuspi*, 2016. dirección: <https://optimuspi.wordpress.com/2016/07/17/dynamixel-mx-106t-arduino-uno/> (visitado 14-09-2018).
 - [4] National Instruments, *Comunicación Serial: Conceptos Generales - National Instruments*. dirección: <http://digital.ni.com/public.nsf/allkb/03900> (visitado 03-06-2018).
 - [5] Juan González Gómez, *Cuaderno técnico I: Comunicaciones serie (HW)*, 2003. dirección: <http://www.learobotics.com/proyectos/cuadernos/ct1/ct1.html> (visitado 15-09-2018).
 - [6] National Instruments, *Synchronous Communications and Timing Configurations in Digital Devices - National Instruments*. dirección: <http://www.ni.com/tutorial/6552/en/> (visitado 10-09-2018).
 - [7] ROBOTIS, *Protocol 1.0*, 2018. dirección: <http://emanual.robotis.com/docs/en/dxl/protocol1/> (visitado 10-09-2018).
 - [8] Microchip, «PIC16F882/883/884/886/887 Data Sheet», inf. téc., 2007. dirección: <http://ww1.microchip.com/downloads/en/DeviceDoc/41291D.pdf>.
 - [9] ROBOTIS, *DYNAMIXEL All-in-one Smart Actuator*, 2018. dirección: <http://www.robotis.us/dynamixel/> (visitado 10-09-2018).
 - [10] —, *XL-320 eManual*, 2010. dirección: http://support.robotis.com/en/product/actuator/dynamixel%7B%5C_%7Dx/xl%7B%5C_%7Dseries/xl-320.htm (visitado 10-09-2018).

- [11] —, *AX-12/AX-12+/AX-12A*. dirección: http://support.robotis.com/en/product/actuator/dynamixel/ax%7B%5C_%7Dseries/dxl%7B%5C_%7Dax%7B%5C_%7Dactuator.htm (visitado 11-09-2018).
- [12] Robotis, *AX Series pinout*, 2010. dirección: http://support.robotis.com/en/product/actuator/dynamixel/dxl%7B%5C_%7Dax%7B%5C_%7Dmain.htm (visitado 15-09-2018).
- [13] ROBOTIS, *MX-106T / MX-106R*, 2010. dirección: http://support.robotis.com/en/product/actuator/dynamixel/mx%7B%5C_%7Dseries/mx-106.htm (visitado 11-09-2018).
- [14] MICROCHIP, *PIC16F887 - Microcontrollers and Processors - Microcontrollers and Processors*, 2018. dirección: <https://www.microchip.com/wwwproducts/en/PIC16F887> (visitado 11-09-2018).
- [15] A. Malvino, «Dispositivos optoelectrónicos», en *Principios De Electrónica*, West Balley College, 2000, págs. 195-200.
- [16] J. M. Malloza, «Equipos electrónicos», en *Montaje de componentes y periféricos microinformáticos*, ic editorial, 2014, págs. 120-125. dirección: https://play.google.com/store/books/details?id=77sPBAAAQBAJ&rdid=book-77sPBAAAQBAJ&rdot=1&source=gbs_vpt_read&pcampaignid=books_booksearch_viewport.
- [17] ROBOTIS, *Instruction_status packet Protocol 2.0*, 2010. dirección: http://support.robotis.com/en/product/actuator/dynamixel%7B%5C_%7Dpro/communication/instruction%7B%5C_%7Dstatus%7B%5C_%7Dpacket.htm (visitado 15-09-2018).
- [18] Walker Gosrich, *Controlling the Dynamixel XL320 with an Arduino*, 2018. dirección: <http://walker.gosrich.com/posts/xl320.html> (visitado 15-09-2018).

13.1. Análisis con Osciloscopio y Saleae

13.1.1. Capturas de Osciloscopio

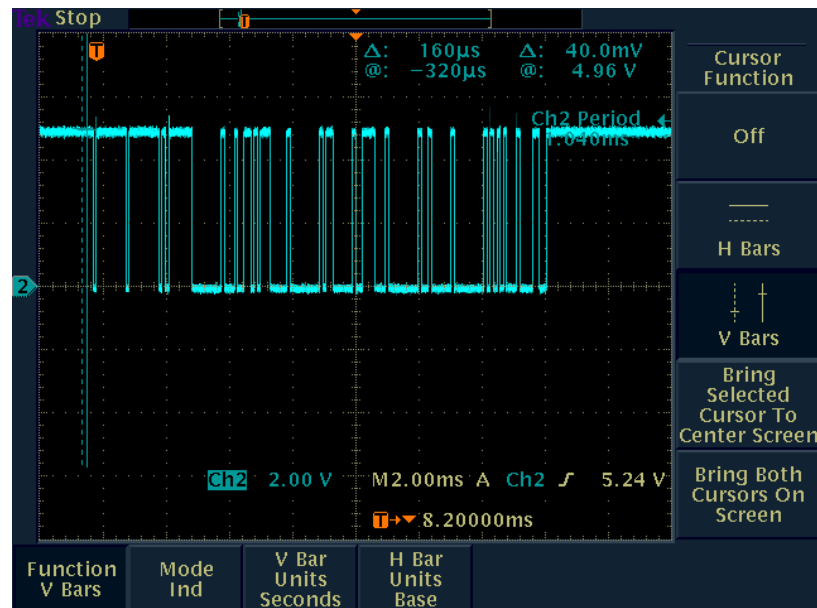


Figura 25: Prueba motor XL-320, ID 100, Posición 10

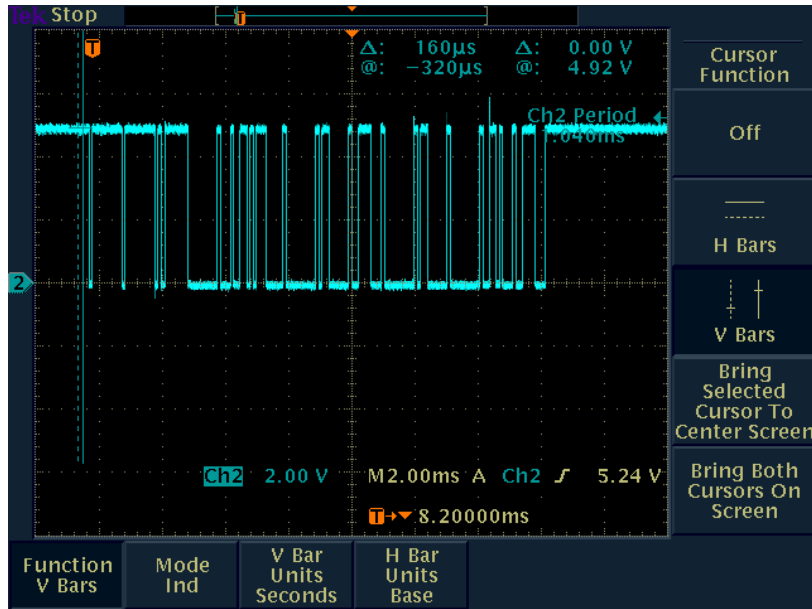


Figura 26: Prueba motor XL-320, ID 100, Posición 100

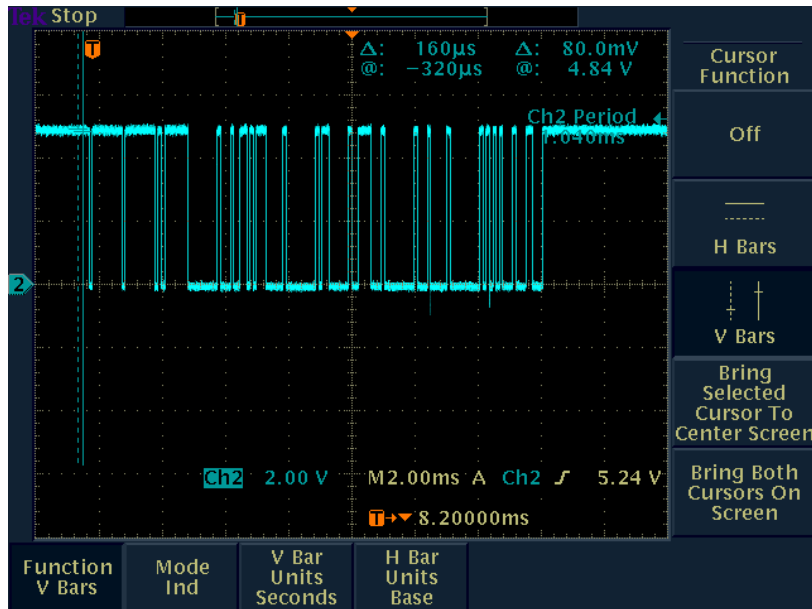


Figura 27: Prueba motor XL-320, ID 100, Posición 1000

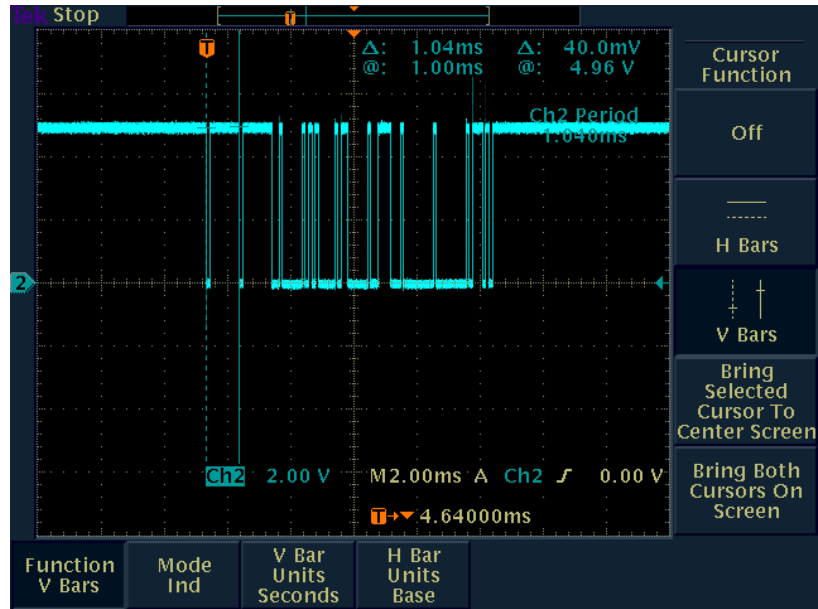


Figura 28: Prueba motor AX-12A, ID 2, Posición 0

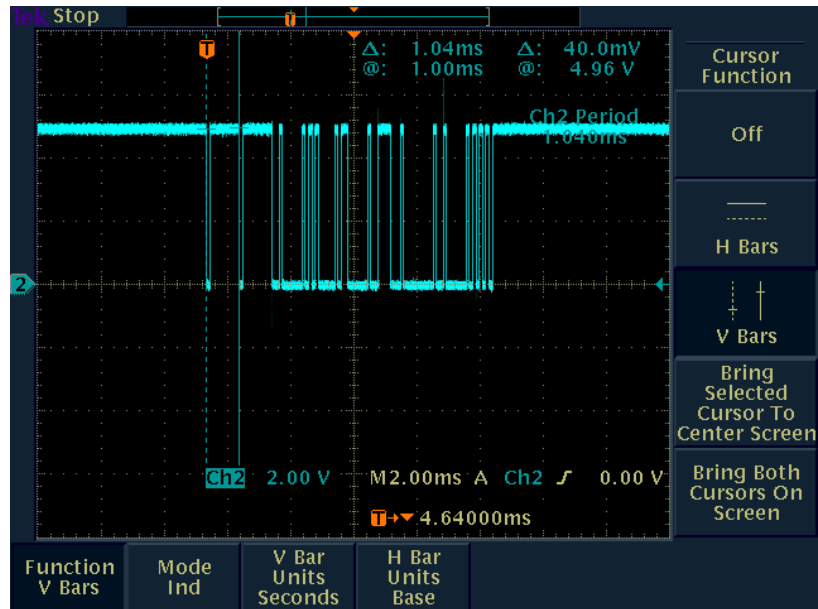


Figura 29: Prueba motor AX-12A, ID 2, Posición 512

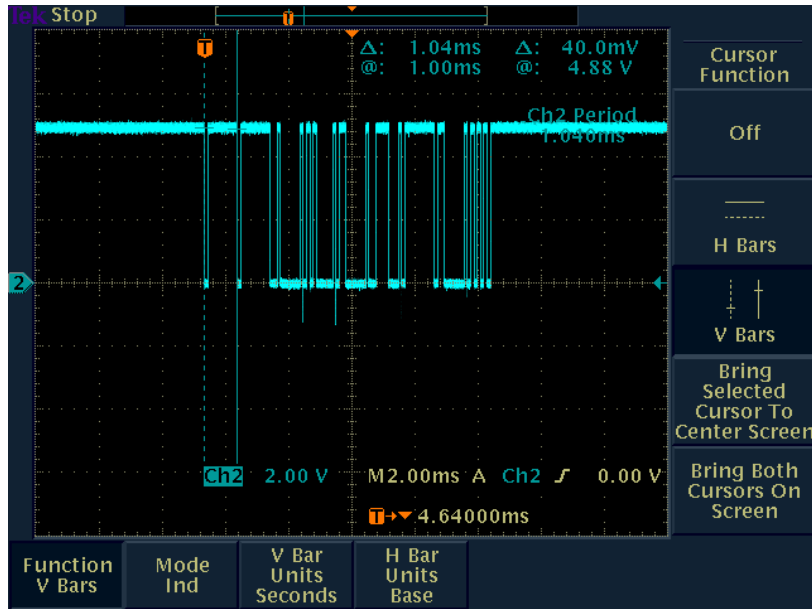


Figura 30: Prueba motor AX-12A, ID 2, Posición 1023

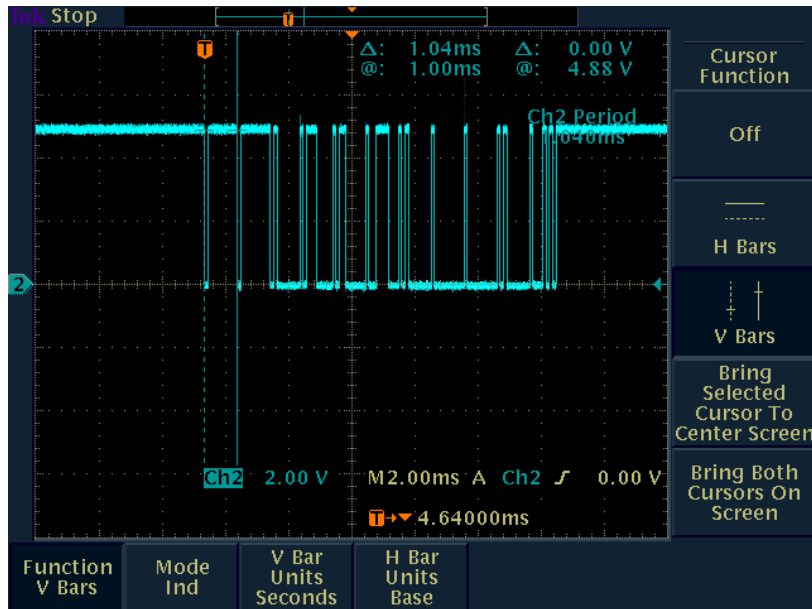


Figura 31: Prueba motor MX-106T, ID 1, Posición 1, Velocidad 100

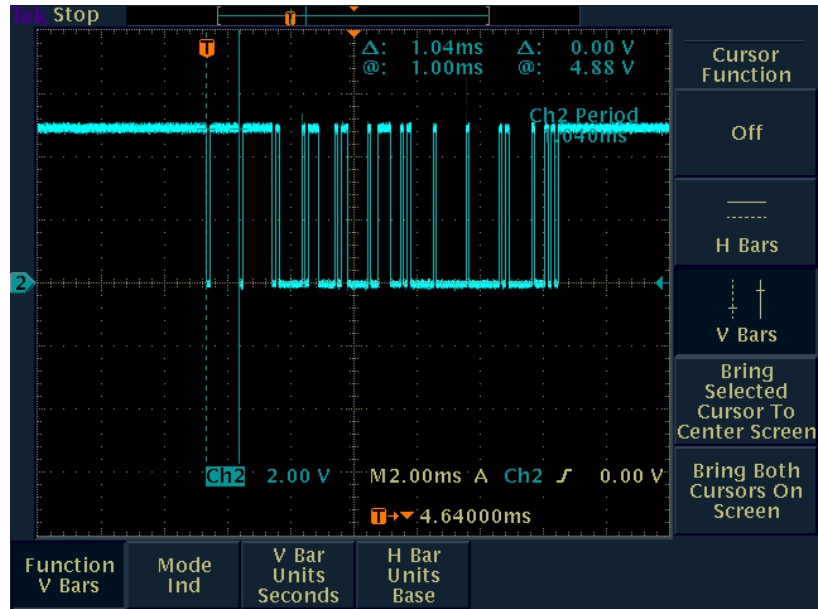


Figura 32: Prueba motor MX-106T, ID 1, Posición 1023, Velocidad 500

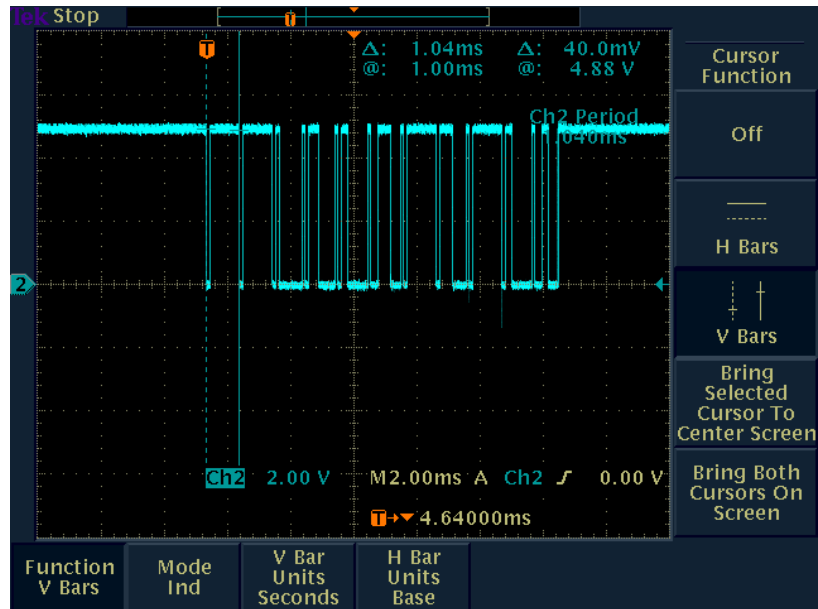


Figura 33: Prueba motor MX-106T, ID 1, Posición 4095, Velocidad 1023

13.1.2. Pruebas con Saleae

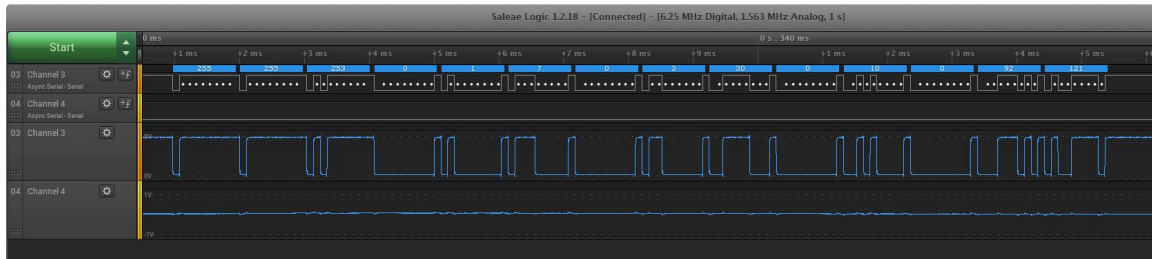


Figura 34: Prueba motor XL-320, ID 1, Posición 10

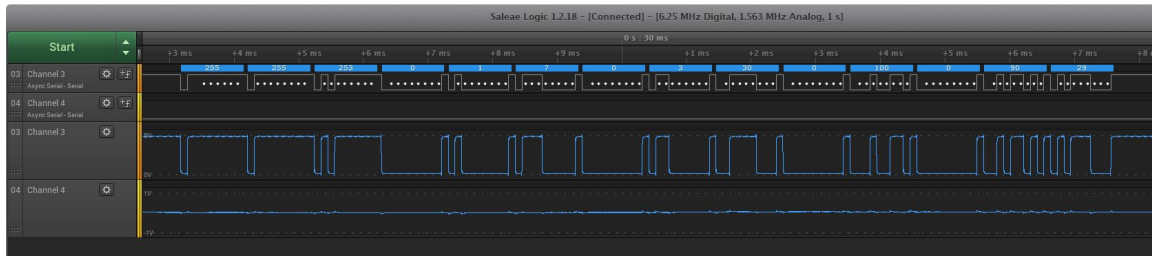


Figura 35: Prueba motor XL-320, ID 1, Posición 100

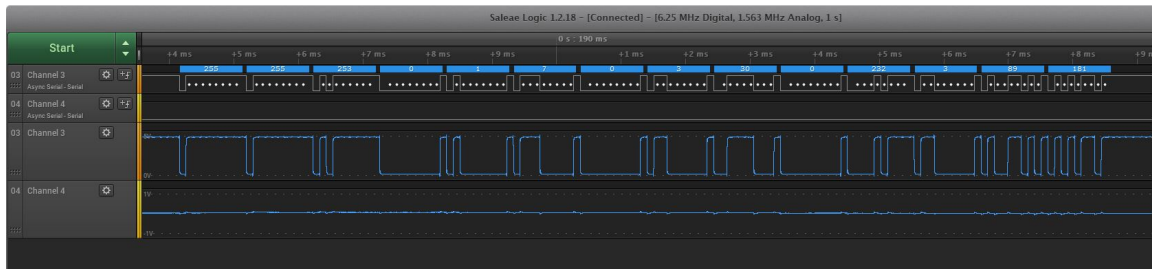


Figura 36: Prueba motor XL-320, ID 1, Posición 1000

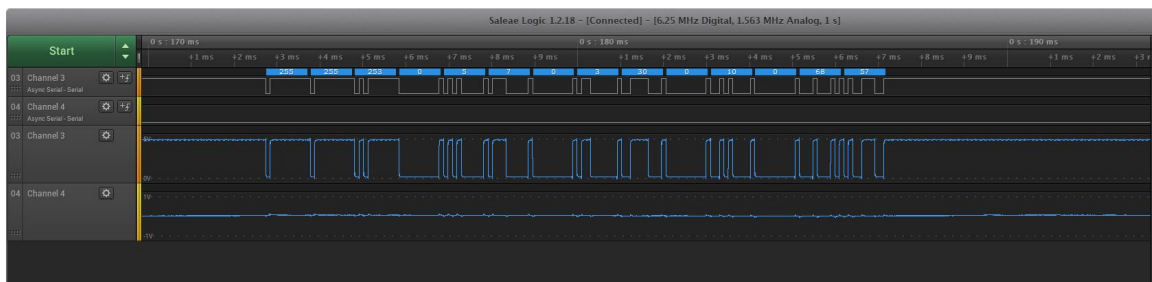


Figura 37: Prueba motor XL-320, ID 5, Posición 10

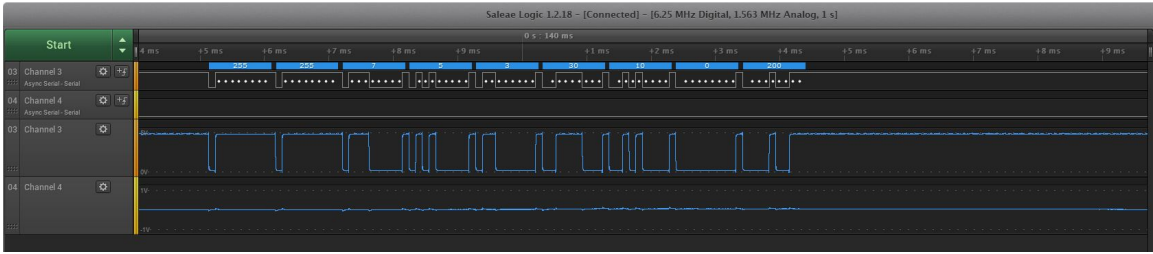


Figura 46: Prueba motor AX-12A, ID 7, Posición 10

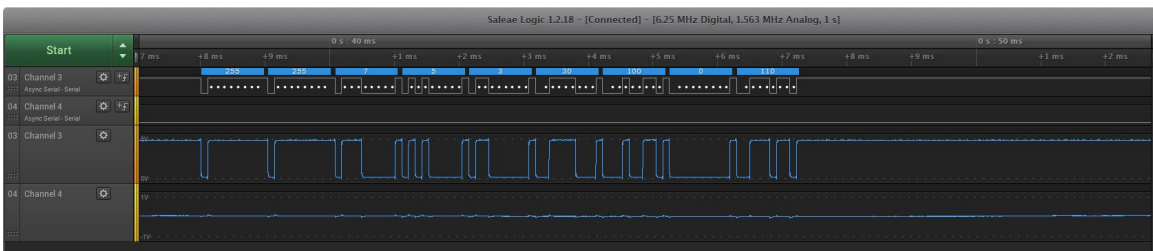


Figura 47: Prueba motor AX-12A, ID 7, Posición 100

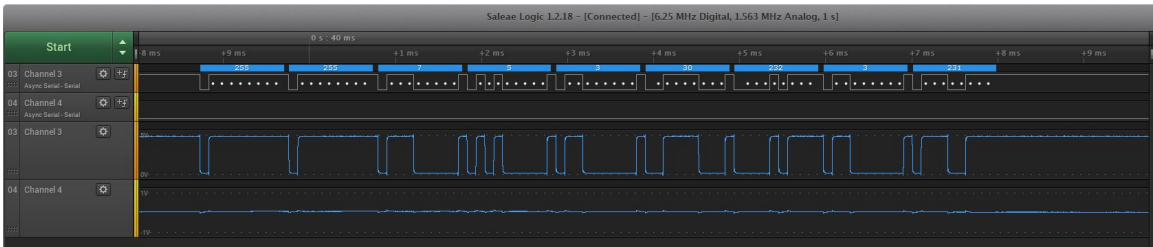


Figura 48: Prueba motor AX-12A, ID 7, Posición 1000

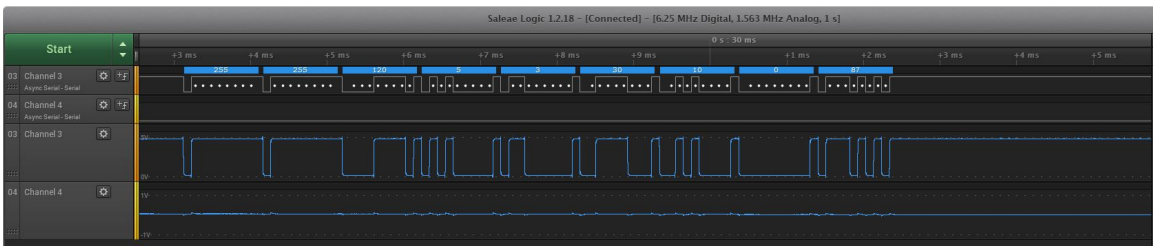


Figura 49: Prueba motor AX-12A, ID 120, Posición 10

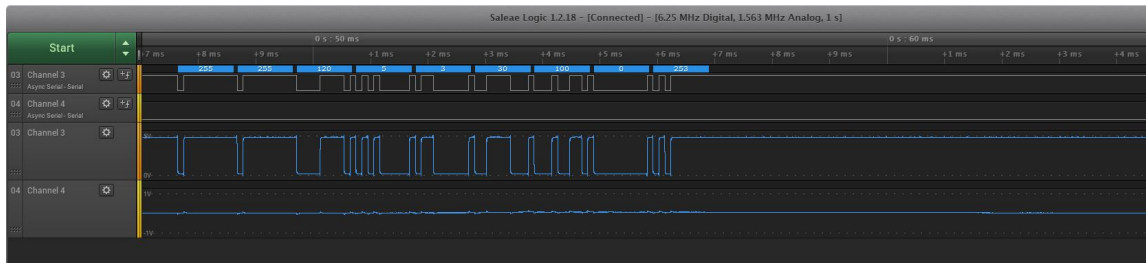


Figura 50: Prueba motor AX-12A, ID 120, Posición 100

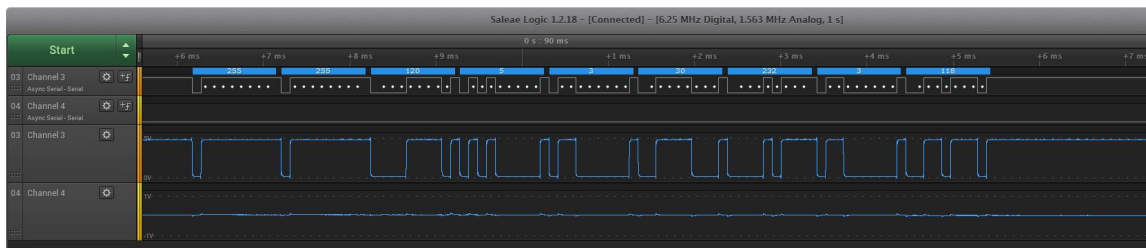


Figura 51: Prueba motor AX-12A, ID 120, Posición 1000

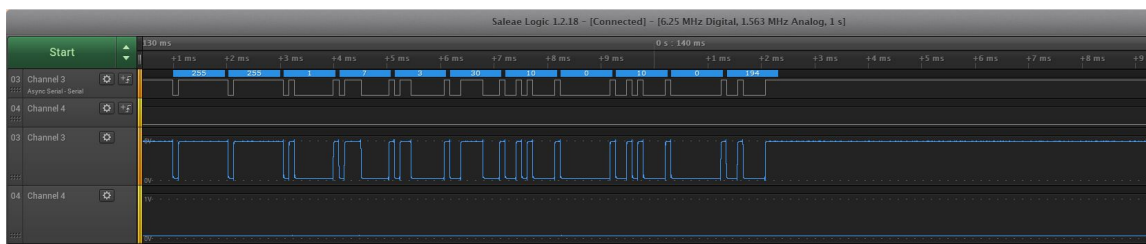


Figura 52: Prueba motor MX-106T, ID 1, Posición 10, Velocidad 10

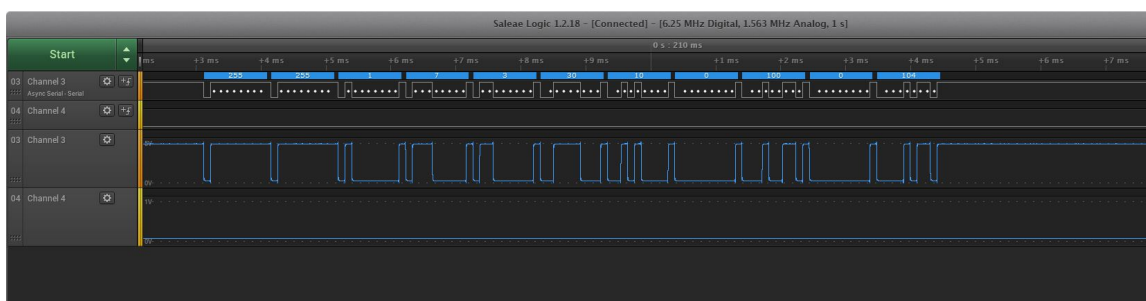


Figura 53: Prueba motor MX-106T, ID 1, Posición 10, Velocidad 100

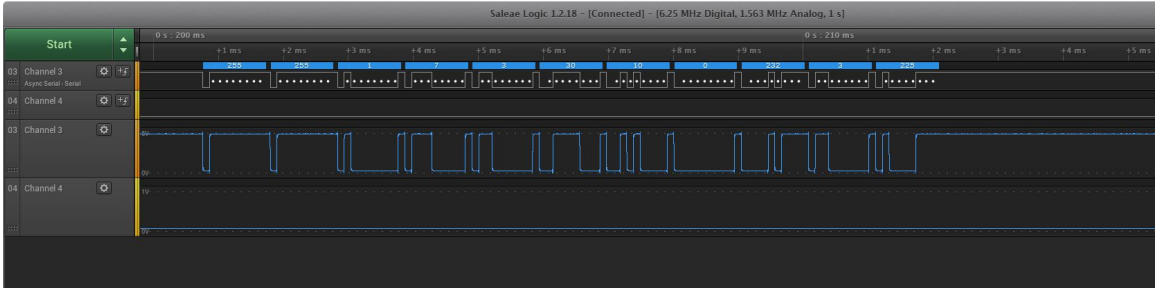


Figura 54: Prueba motor MX-106T, ID 1, Posición 10, Velocidad 1000

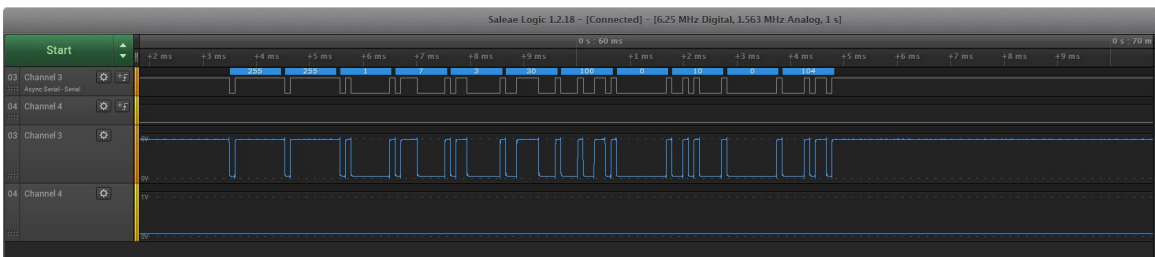


Figura 55: Prueba motor MX-106T, ID 1, Posición 100, Velocidad 10

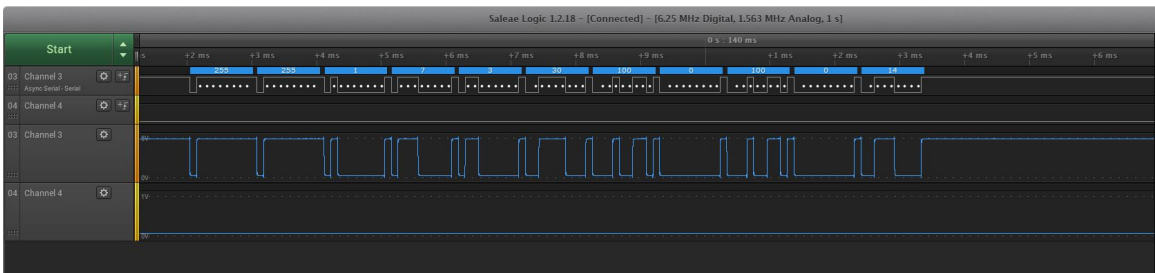


Figura 56: Prueba motor MX-106T, ID 1, Posición 100, Velocidad 100

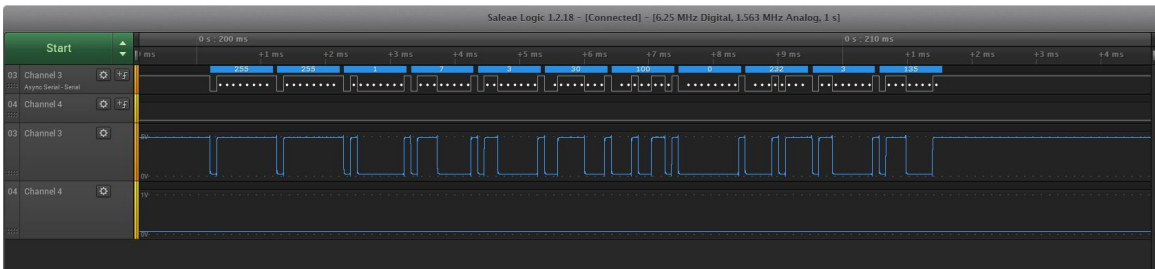


Figura 57: Prueba motor MX-106T, ID 1, Posición 100, Velocidad 1000

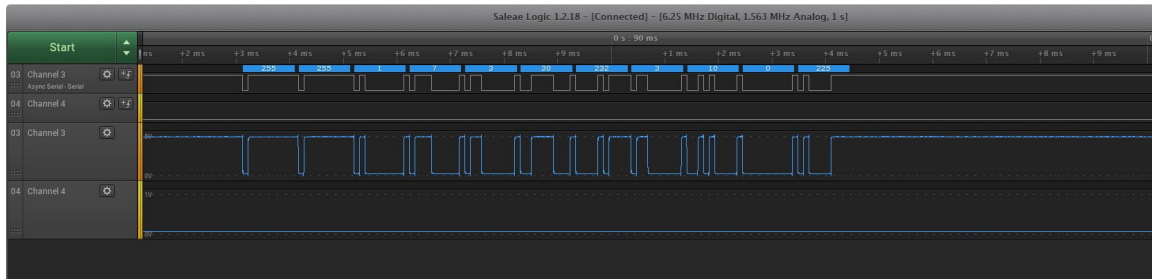


Figura 58: Prueba motor MX-106T, ID 1, Posición 1000, Velocidad 10

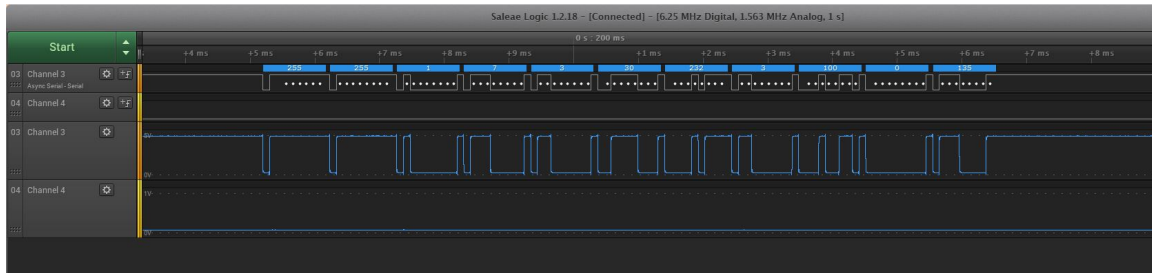


Figura 59: Prueba motor MX-106T, ID 1, Posición 1000, Velocidad 100

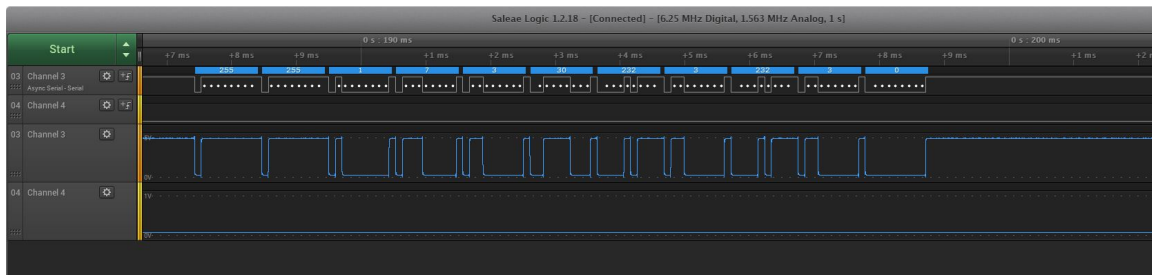


Figura 60: Prueba motor MX-106T, ID 1, Posición 1000, Velocidad 1000

13.2. Fotos importantes

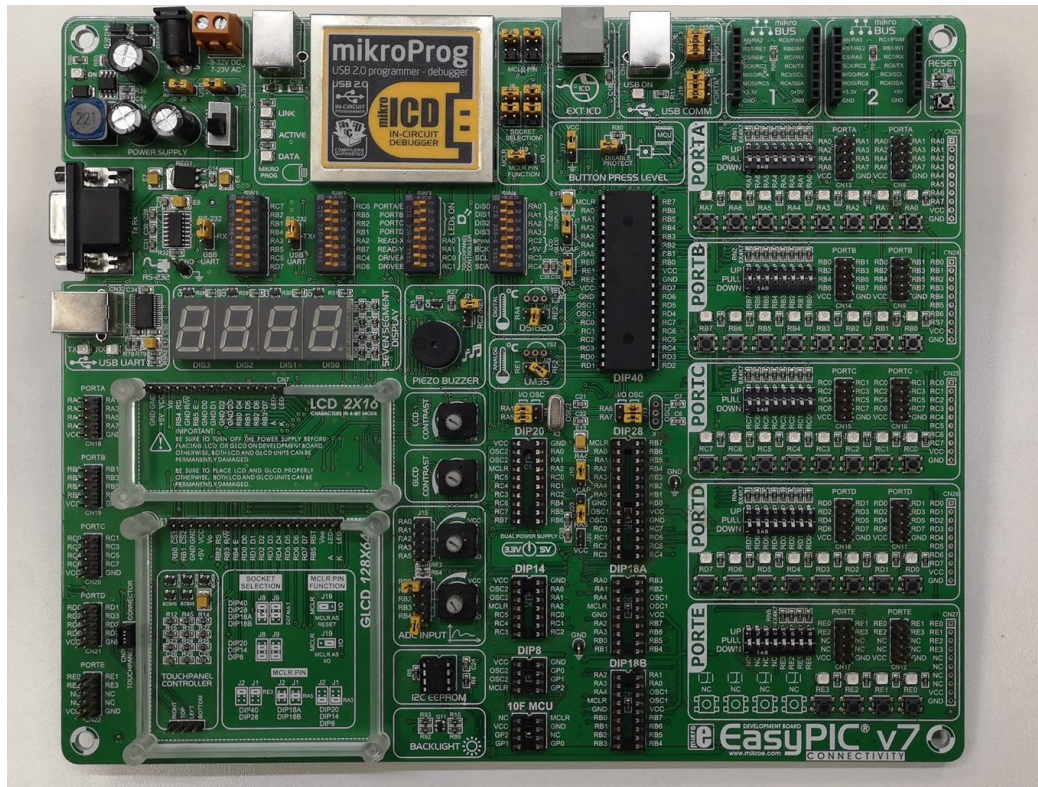


Figura 61: Easy Pic v7 utilizada para pruebas



Figura 62: Saleae logic pro 16 utilizado para pruebas

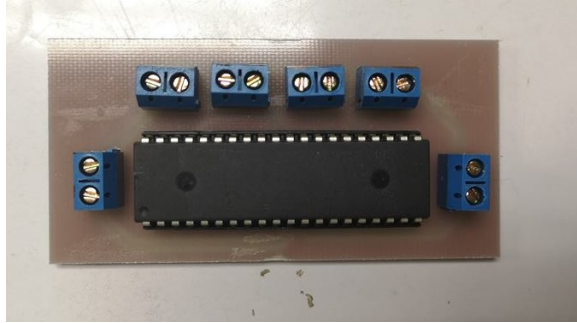


Figura 63: Diseño de placa versión 1

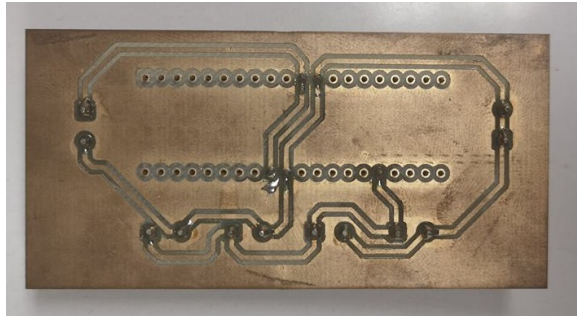


Figura 64: Soldadura de placa versión 1

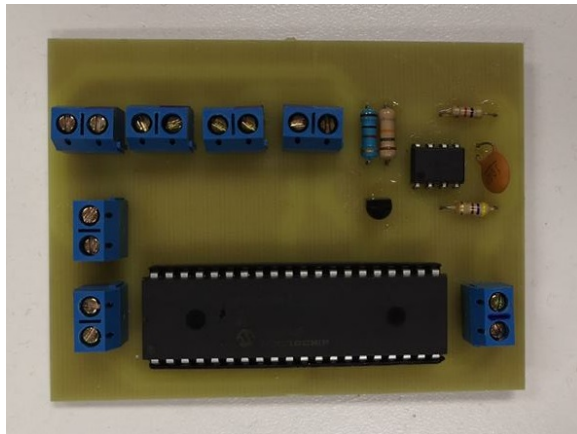


Figura 65: Diseño de placa versión 2

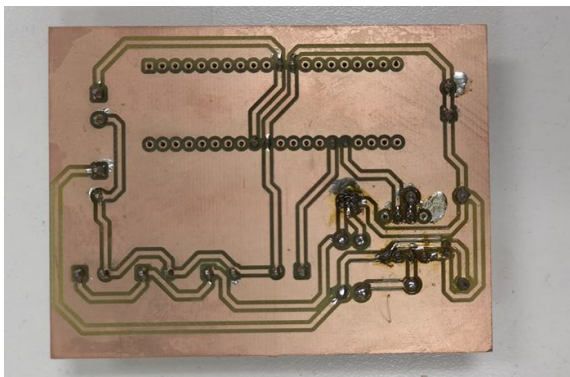


Figura 66: Soldadura de placa versión 2

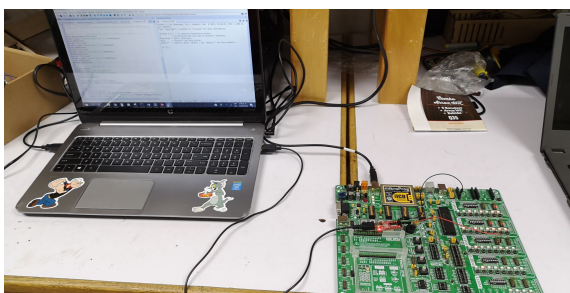


Figura 67: Pruebas de comunicación con módulo de interacción por voz

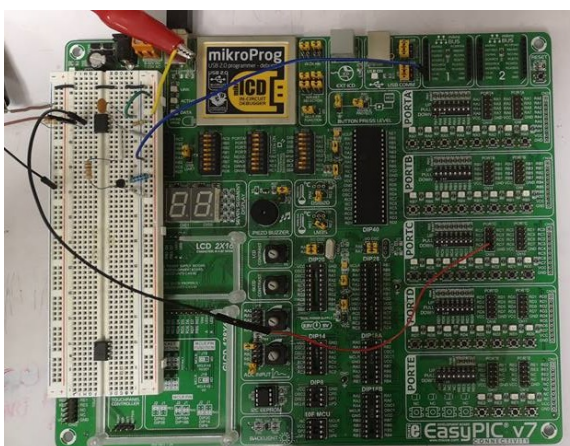


Figura 68: Pruebas con circuito de aislamiento

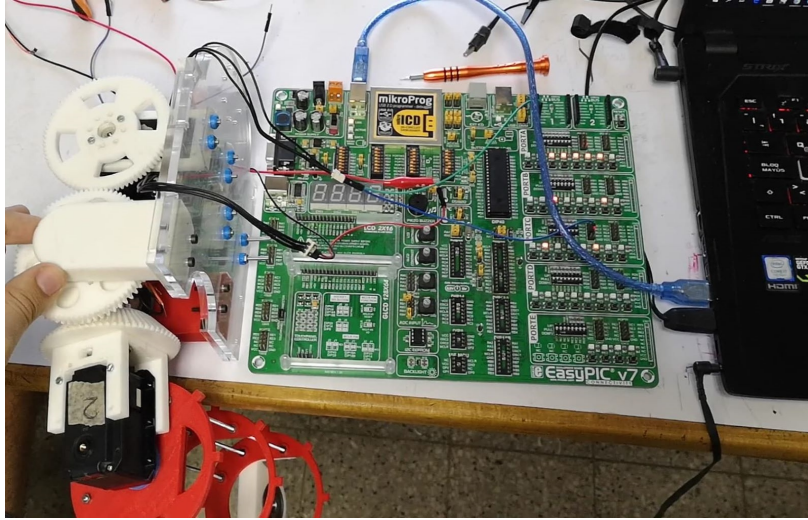


Figura 69: Pruebas con estructura de hombro

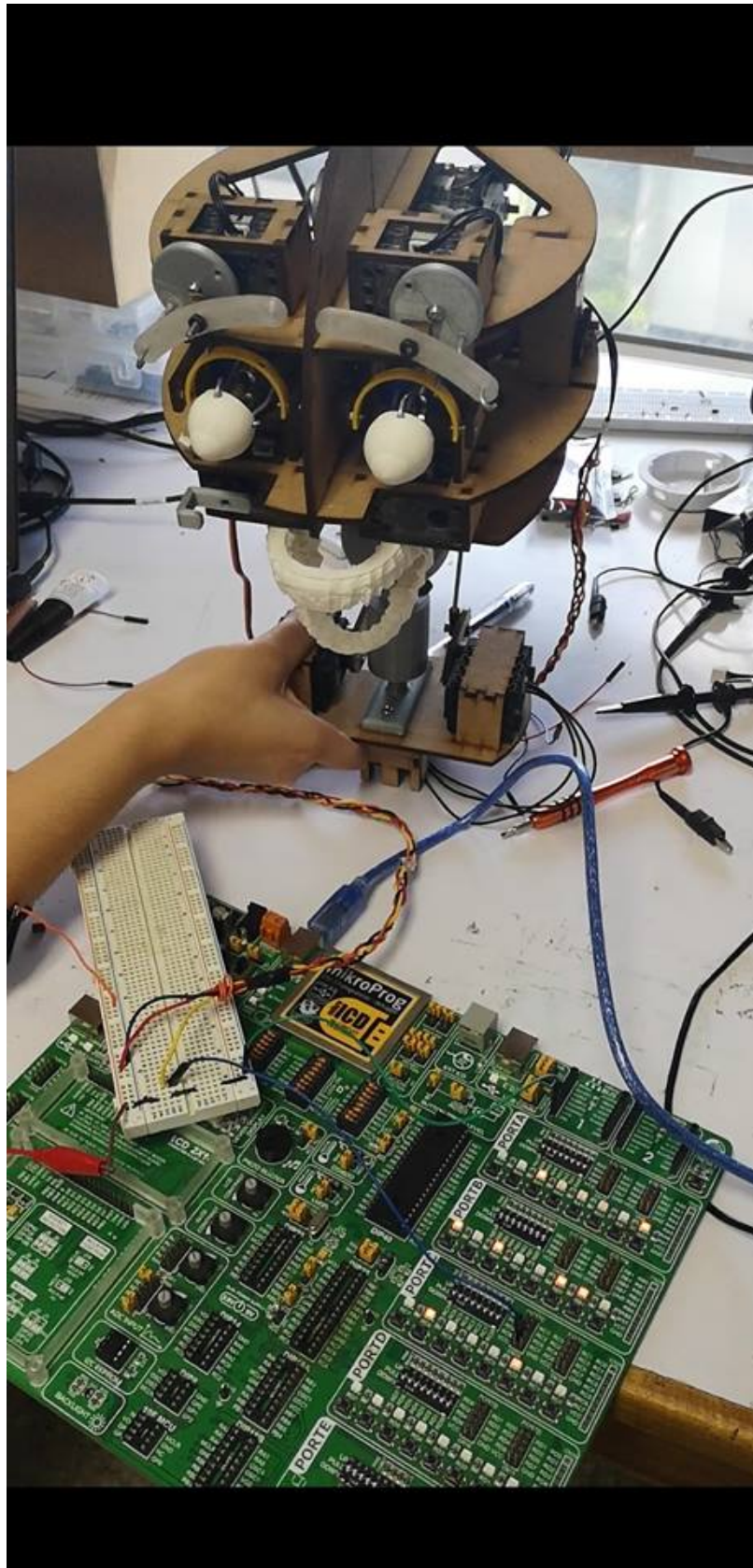
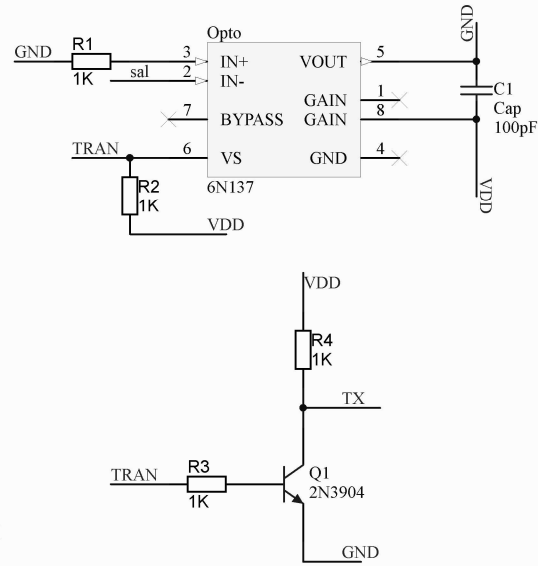
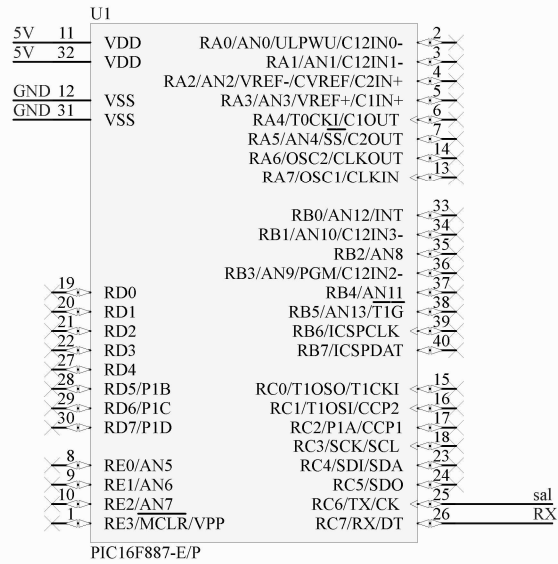


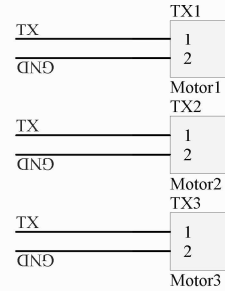
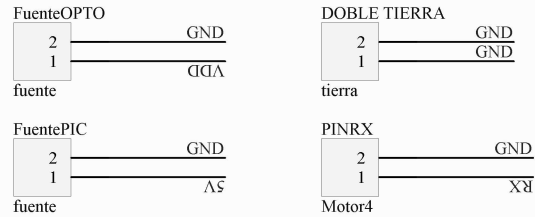
Figura 70: Pruebas con estructura de cabeza

13.3. Circuitos exportados a PDF



Modulo de Control

Circuito de aislamiento



Entradas

Salidas

Title		
Size	Number	Revision
A4		
Date:	24/09/2018	Sheet of
File:	C:\Users\...\Sheet1.SchDoc	Drawn By: