UNSAM
UNIVERSIDAD
NACIONAL DE
SAN MARTÍN

Proyecto Final

Patrón de flujo y volumen espiratorio para la calibración de instrumentos de valoración de la función pulmonar

11/05/2016

FINAL

UNSAM, **NATALIA M. REQUEJO**, **INGENIERÍA EN ELECTRÓNICA**

Trabajo Final

Historial de Revisiones

Rev.	Fecha	Párrafo	Página	Motivo	Firma
01	11/05/2016	-----	-------	Inicio	
02	11/06/2016			WBS y Diagrama de Gantt	
03	12/06/2016			Revisiones para la presentación del primer borrador	
04	25/04/2017			Motor Paso a Paso	
05	04/05/2017			RS232-UART	
06	05/05/2017			Protocolo de Comunicación	
07	20/05/2017			Software CIAA PC	
08	05/06/2017			Diagramas de Flujo y Anexos	
09	10/07/2017			Correcciones	

Lista de Distribución

Director Técnico Responsable:	La Mura Guillermo M.
Codirector del Proyecto:	Romeo Marcelo.
Alumno:	Requejo Natalia M.
Carrera:	Ingeniería en Electrónica.

Contenido

Proyecto Final de Carrera

1. Gestión de Proyecto

1.1 Definición del Alcance y Propósito del proyecto

Se desea desarrollar un controlador, el cual deberá generar un patrón de flujo y volumen espiratorio determinístico, que haga posible la calibración de equipos espiro-métricos, utilizando la CIAA-NXP (Computadora Abierta Argentina) como núcleo principal del instrumento. El proyecto está basado en el desarrollo de un controlador de posición lineal para controlar el desplazamiento de un émbolo. El presente proyecto no incluye las funciones finales que permitan ajustar el patrón de flujo a las curvas de calibración.

Este controlador será parte de un proyecto mayor, el cual posee diferentes partes y componentes. Consiste en un pistón neumático, certificado, una regla digital que medirá con gran exactitud la posición del embolo, un control de temperatura para mantenerla a 37° Centígrados y un transductor de presión. A continuación, se presenta un diagrama sobre el proyecto en general.

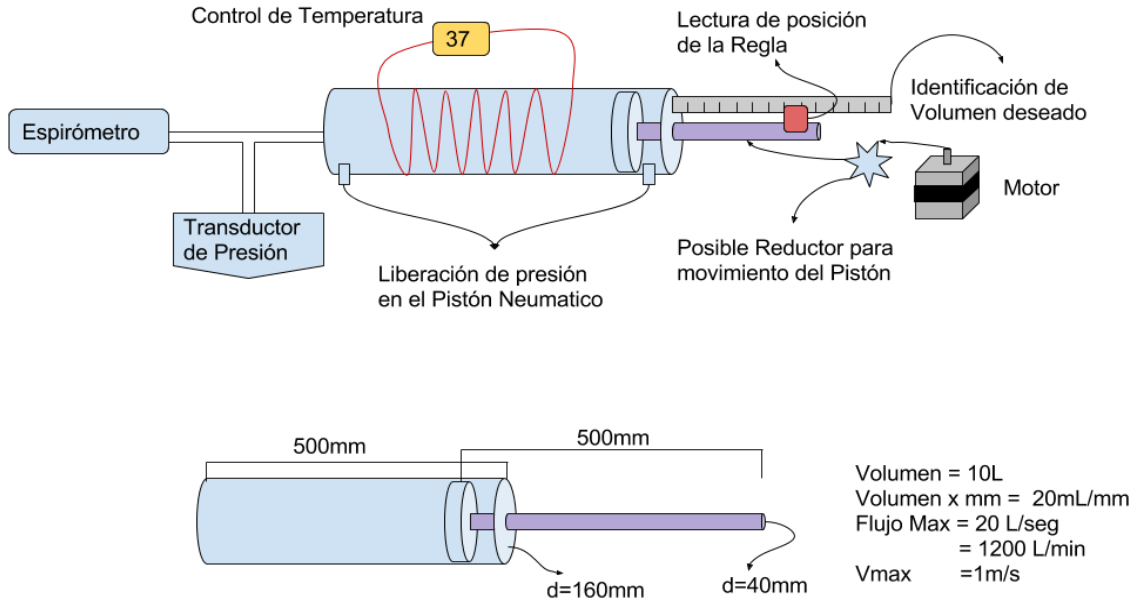


Figura 1: Esquema general del proyecto final.

1.2 Supuestos del proyecto

Se supone que se realizaron los estudios de factibilidad correspondientes, y que el proyecto presentado es posible llevarlo a cabo.

Se supone que se tiene acceso a todas las partes involucradas del equipo, no siendo responsabilidad de los integrantes del proyecto la decisión ni elección de marca o modelo de las partes mecánicas.
Se supone que se cuenta con una CIAA-NXP para el desarrollo del prototipo y su testeo.
Se espera poder generar mediciones y movimientos repetibles que hagan posible una medición exacta y además que permita que el patrón de flujo y volumen sea determinístico.

1.3 Requerimientos

Los requerimientos referentes al controlador no pueden modificarse hasta entregado el proyecto.

1- Grupo de requerimientos
   a. Control de Movimiento
      i. El motor debe poder controlarse de forma exacta, haciendo posible la repetitividad del movimiento.
      ii. El motor debe generar las RPM correspondientes para que el movimiento lineal del pistón llegue a 1m/s.
      iii. Debe seguir una curva cargada en el sistema, y su movimiento debe ser repetible
      iv. Para iniciar el movimiento el motor debe encontrarse en la posición de origen.
      v. Tanto el inicio como la finalización del movimiento deberán poder controlarse con el Software de la PC.
      vi. En caso de llegar al extremo contrario y detectar el final de carrera, el motor deberá detenerse.
2- Grupo de Requerimientos asociados al proyecto
   a. Informe detallado del proyecto.
   b. Manual de usuario.
   c. Manual de Service.

1.4 Definición de los Entregables

- Informe Final del proyecto.
- Documentación.
- Manuales de Uso, y service.

1.4.1 Hardware

- Prototipo funcional.

1.4.2 Firmware

- Código Fuente del firmware desarrollado.
- Diagrama en bloques o estados.

1.4.3 Software

- Código Fuente del Software de PC.

- Diagrama en bloques.

1.5 Desglose en WBS

A continuación, se presenta el desglose en WBS del proyecto presentado. Si bien el desglose de tareas es completo, en lo que se refiere a este trabajo se hará hincapié únicamente en lo correspondiente al software de PC y firmware desarrollado.

1. Gestión de Proyecto
   - 1.1. Definición del Alcance y Propósito del proyecto
   - 1.2. Supuestos del proyecto
   - 1.3. Requerimientos
   - 1.4. Definición de los Entregables
     - 1.4.1.Hardware
     - 1.4.2.Firmware
     - 1.4.3.Software
   - 1.5. Desglose en WBS
   - 1.6. Diagrama de Gantt
   - 1.7. Armado de Informes
   - 1.8. Reuniones
     - 1.8.1.Reunión semanal Definición del proyecto, asignación de roles
     - 1.8.2.Presentación CIAA-NXP
     - 1.8.3.Ajuste de requisitos, Definición del Motor
   - 1.9. Gestión de Riesgos
     - 1.9.1.Identificación de los riesgos
     - 1.9.2.Estimación de ocurrencia
     - 1.9.3.Estimación de consecuencia
   - 1.10. Gestión de Calidad
     - 1.10.1. Grado de Calidad
     - 1.10.2. Costos de Conformidad
2. Documentos
   - 2.1. Manual de Usuario
     - 2.1.1.Definición de los Requerimientos
     - 2.1.2.Diseño
     - 2.1.3.Primer Borrador
     - 2.1.4.Versión Final
   - 2.2. Manual de Calibración
     - 2.2.1.Definición de los Requerimientos
     - 2.2.2.Diseño
     - 2.2.3.Primer Borrador
     - 2.2.4.Versión Final
   - 2.3. Manual de Service
     - 2.3.1.Definición de los Requerimientos
     - 2.3.2.Diseño
     - 2.3.3.Primer Borrador
     - 2.3.4.Versión Final
3. Hardware

- 3.1. Motor actuador
  - 3.1.1. Definición de motor a utilizar
  - 3.1.2. Desarrollo de salidas de potencia
    - 3.1.2.1. Elección de componentes para prototipo
    - 3.1.2.2. Armado de prototipos
  - 3.1.3. Pruebas de fuerza y velocidad
- 3.2. Sensor de detección de final de carrera
  - 3.2.1. Investigación sobre censado mecánico y óptico
  - 3.2.2. Adaptación de señales
    - 3.2.2.1. Elección de componentes para prototipo
    - 3.2.2.2. Armado de prototipos
  - 3.2.3. Montaje de sensores
- 3.3. Prototipo Final en PCB
  - 3.3.1.Armado y pruebas

4. Firmware CIAA
   - 4.1. Control del actuador del motor
     - 4.1.1.Investigación sobre el hardware a utilizar
     - 4.1.2.Desarrollo del software controlador del motor
     - 4.1.3.Relación pasos del motor vs. pasos tabla tiempo
     - 4.1.4.Corrección de los pasos
   - 4.2. Detección de Fin de carrera
     - 4.2.1.Definición del tipo de final de carrera
     - 4.2.2.Investigación, sobre forma de uso
   - 4.3. Control de seguridad del sistema
     - 4.3.1.Interrupción en CIAA (Alta prioridad)
     - 4.3.2.Tarea de comunicación a alto nivel
     - 4.3.3.Función de RESET del sistema
   - 4.4. Protocolo de Comunicación
     - 4.4.1.Recepción de comandos (funcionalidades)
     - 4.4.2.Comunicación con PC
       - 4.4.2.1. Rutina para enviar comandos
       - 4.4.2.2. Rutina para enviar a la PC las posiciones del motor de la última curva
       - 4.4.2.3. Salvado de curvas patrón en la EEPROM
5. Software Visualizador para PC
   - 5.1. Requerimientos
   - 5.2. Diseño
   - 5.3. Aplicación GUI en C#
   - 5.4. Comunicación con la CIAA
     - 5.4.1.Módulo de comunicación RS232 o similar
     - 5.4.2.Rutinas de Lectura de datos
     - 5.4.3.Formateo de datos para postproceso (si vienen en array, uno detrás de otro, cargar distintas estructuras)
     - 5.4.4.Envío de comandos a la CIAA
     - 5.4.5.Carga de curvas Patrón en la CIAA
   - 5.5. módulo para Graficar puntos
   - 5.6. Visualización de curvas patrón
6. Integración
   - 6.1. Verificación de la compatibilidad de las partes

6.2. Ensamble de las partes mecánicas
6.3. Ensamble de las partes electrónicas
6.4. Ensamble del controlador con el pistón
7. Verificación y Validación
7.1. Validación de la documentación
7.1.1. Manuales de Uso
7.1.2. Manuales de Service

1.6 Diagrama de Gantt

El diagrama de Gantt fue generado en base al trabajo de los dos participantes, tomando en cuenta que se trabajará sobre el proyecto de seis a ocho horas semanales. Ver Anexo 1. Este tiempo de trabajo es aproximado y queda abierto a futuros cambios, teniendo en cuenta que es un proyecto educativo y que no se recibirá ninguna remuneración económica por el mismo.

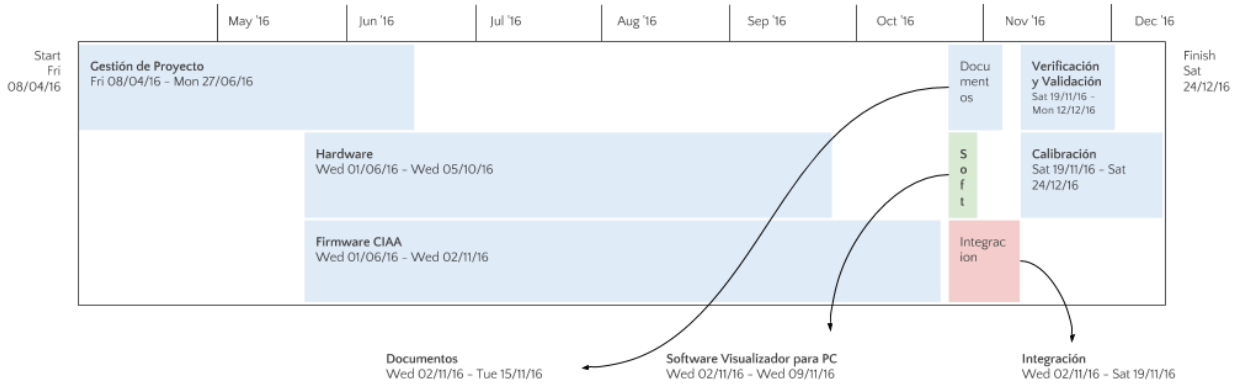


Figura 2: Diagrama de Gantt

1.7 Gestión de Riesgos

1.7.1 Identificación de los riesgos, Estimación de ocurrencia y Consecuencias.

Se identificaron todos los posibles riesgos relacionados al proyecto. Desde la disponibilidad de los recursos humanos, la pérdida de interés de la universidad en el proyecto, como así también cambios en los requisitos del proyecto. Una vez identificados se estimó una probabilidad de ocurrencia y el impacto que podría tener sobre el proyecto.
Esto derivó en subsiguientes reuniones con el personal a cargo para obtener garantías de que el proyecto podría ser llevado a término.
Por último, como consecuencia de estas reuniones se derivó en algunos cambios de requerimientos, y la dilatación de algunas fechas de entrega.
En el Anexo 2 se puede ver la tabla completa con la estimación numérica de cada riesgo.

1.8 Gestión de Calidad

El término gestión de calidad, intenta garantizar que tanto el firmware como el software sea consistente. Respecto al Software de PC el único algoritmo que puede llegar a sufrir de alguna inconsistencia está asociado con la comunicación entre la CIAA-NXP y la PC. Esta inconsistencia

puede deberse a errores en la comunicación, en la línea de transmisión o en la mala decodificación de los paquetes. Los primeros dos se evitan si se utilizan los medios correctos para el conexionado de ambas partes, respecto al tercer punto, el protocolo tiene implementado un mecanismo de detección y corrección de errores, lo que garantiza que los paquetes transmitidos son correctamente recibidos.

Como segunda instancia existe el movimiento del motor, la única manera de verificar y controlar que el grupo de datos enviados para que el motor realice la curva se cumpla, es a través de un estudio estadístico, en el cual se determine la repetibilidad del movimiento.
Para este estudio se utilizó una frecuencia fija de 5000Hz y se recorrieron diferentes distancias.
Se tuvo especial cuidado en que el punto de partida fuera siempre el mismo, y que las mediciones fueran realizadas siguiendo siempre el mismo criterio de observación.
Se configuraron distinta cantidad de pasos a recorrer y se midió la distancia recorrida. Pudiendo de esta forma establecer cuanto es la distancia lineal que se recorre con solo un paso del motor.
El error asociado al movimiento del motor es de 0,0511µm lo que corresponde a un 0,5% del paso.
Ver en Anexo 3 la tabla de datos y el gráfico asociado.

2. Firmware CIAA-NXP

El firmware que se desarrolló para el movimiento del émbolo fue escrito íntegramente en lenguaje C. Se utilizó como entorno de desarrollo el IDE de NXP, LPCxpresso. A continuación, se presenta un diagrama esquemático, con la interconexión de los distintos dispositivos.

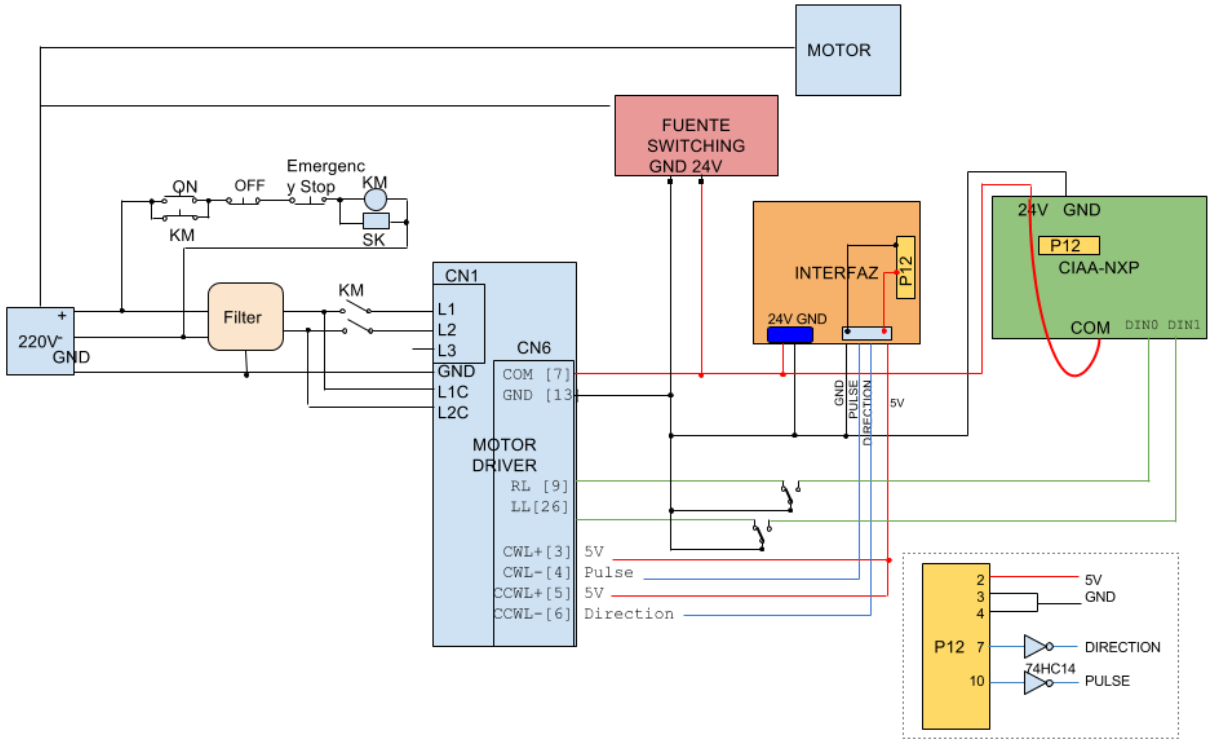


Figura 3: Esquema de conexión de hardware.

En esta primera etapa, la CIAA-NXP tiene el control del motor y los limit-switch e implementa un protocolo de comunicación con la PC, a través del serial port, permitiendo realizar la carga de una curva patrón o realizar movimientos específicos.

2.1 Control del actuador del motor

2.1.1 Investigación sobre el hardware a utilizar

El control de posicionamiento del pistón se realiza a través de un motor paso a paso de alta precisión. Este motor es controlado a través de un tren de pulsos, donde cada pulso define un paso del motor. De acuerdo a la frecuencia de recepción de pulsos, se aumenta o disminuye la velocidad de este.

2.1.1.1 Motor Paso a Paso

El principio básico de un motor paso a paso es la conversión de energía eléctrica en energía mecánica utilizando los principios de electromagnetismo, que consiste en la atracción o repulsión que se ejerce entre distintos polos magnéticos. A diferencia de los motores tradicionales, donde al aplicar una tensión fija, su movimiento es continuo, en el motor paso a paso con cada pulso que se alimenta, el motor gira un determinado ángulo, incrementando su posición de a un paso por vez.
En los motores de corriente continua, es imposible que el motor gire un determinado número de vueltas y pare. Además, tanto al iniciar el movimiento como al finalizar, existe cierto valor de inercia que el motor debe vencer para iniciar su movimiento. Inicialmente no es posible llegar a la velocidad deseada de inmediato, así como tampoco frenar su movimiento al cortar su alimentación, el motor continuará girando debido a la inercia que posee.
En los motores paso a paso por el contrario el movimiento es completamente controlado y permite realizar desplazamientos muy precisos y de forma determinística. Por el contrario, los motores de corriente continua tienen mucha dificultad al momento de controlar su velocidad. Para el control del posicionamiento del émbolo, es de vital importancia un elevado grado de exactitud tanto en la velocidad de giro del motor como en su posicionamiento, siendo posible entonces generar un patrón de flujo bien determinado. Realizar dicho controlador con otro tipo de motor hubiera sido una tarea mucho más compleja. Si bien hacer girar un motor paso a paso es una tarea mucho más compleja, que con un motor de corriente continua. El motor adquirido posee un controlador bien documentado y con una usabilidad relativamente simple, haciendo que la tarea de controlarlo fuera relativamente sencilla.

2.1.1.2 Tipos de Motores paso a paso

Los motores paso a paso están compuestos esencialmente por dos partes, una parte es el estator y la otra el rotor. El estator es la parte fija, donde se alojan las bobinas y el rotor es la parte móvil que generalmente está constituida por un imán permanente.
Los motores paso a paso pueden ser de imán permanente, de reluctancia variable o Híbridos.

2.1.1.2.1 De imán permanente

El movimiento se hace posible gracias a los cables que se enrollan alrededor del estator. Cuando la corriente circula por los distintos estatores, el imán permanente del rotor se orienta en la dirección del campo magnético creado. Teniendo en cuenta que los polos magnéticos del mismo signo se repelen, si existen solo dos estatores y uno se polariza

como Norte y el otro como Sur, el rotor imantado se moverá de forma de alinearse según el equilibrio magnético.
Para motores de más de dos estatores, al introducir impulsos con cierta secuencia lógica, se generarán pasos sucesivos en el motor.
Dependiendo de la cantidad de polos que tenga el motor se obtienen pasos de distinto ángulo. Cuanto mayor sea el número de polos o de pasos, mayor será la resolución del motor.
Si los pulsos de alimentación no se proveen en el orden correcto, el motor no se moverá apropiadamente.

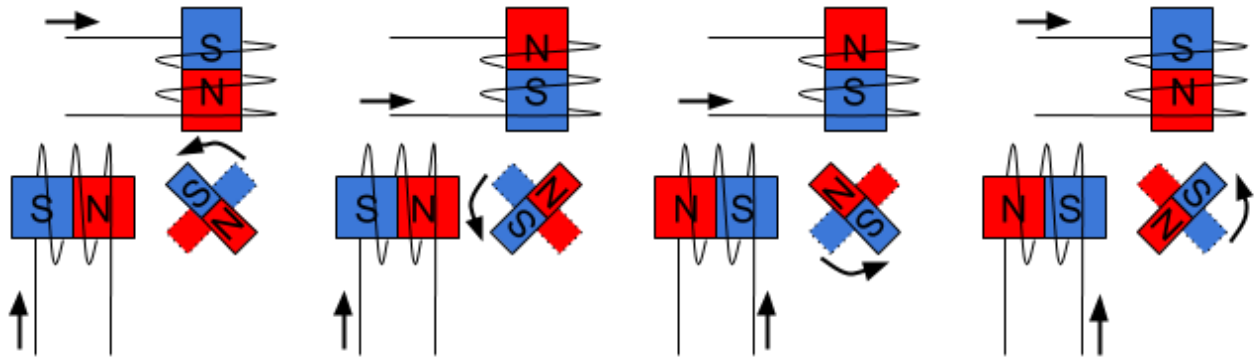


Figura 4: Control de un Motor Paso a Paso Bipolar

2.1.1.2.2 De reluctancia variable

Tanto el estator como el rotor están formados por ruedas dentadas y ante cada impulso eléctrico, el motor gira hasta la posición de reluctancia mínima. El rotor tiende a alinearse con los polos bobinados. Aquí ya no influye el sentido de la corriente, sino que es necesario activar las bobinas contiguas para ir alineando los dientes del rotor con los de la bobina.

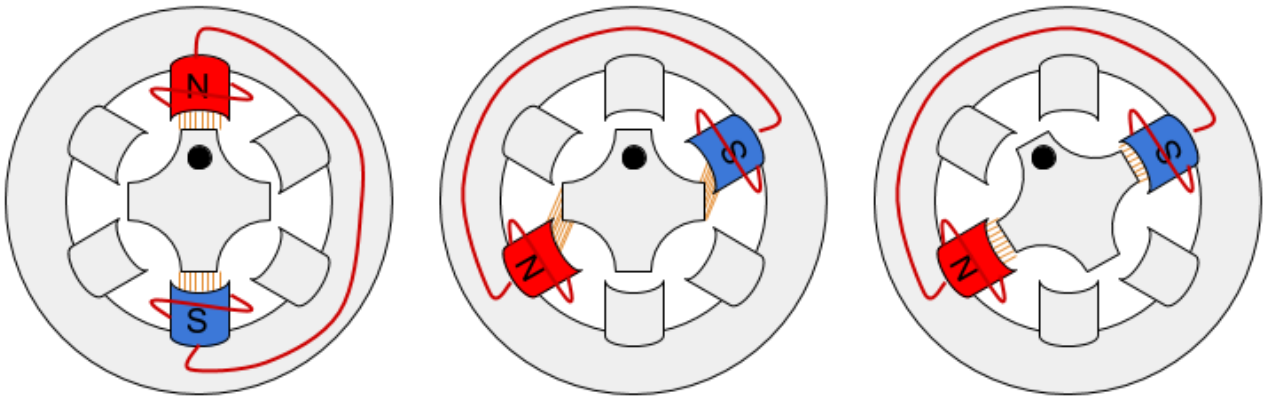


Figura 5: Movimiento secuencial del motor de Reluctancia Variable

2.1.1.2.3 Híbrido

El par se produce mediante la actuación sucesiva de bobinas, buscando la posición de reluctancia mínima. Sin embargo, al campo producido por la excitación le sumaremos el creado por el imán permanente

2.1.1.3 Control de posición

Todos los motores paso a paso tienen la particularidad de poder quedarse fijos en una posición o quedar completamente libres dependiendo de si una o más de sus bobinas está energizada o si no circula corriente por ninguna de ellas. Esto no sucede con los motores de continua, siendo imposible que estos se queden fijos en una posición y por lo tanto se elimina la necesidad de un mecanismo de freno en los motores paso a paso.

Dependiendo de las características del motor y de su controlador, el control de movimiento de estos motores se puede realizar mediante pasos enteros, medio pasos o micro pasos.
El paso entero mueve el rotor de un bobinado a otro, cada vez que estos se polarizan de forma adecuada. Avanzando siempre de a un estator por vez. El medio paso, nos permite movernos entre las posiciones medias entre cada estator, esto nos da una mejor resolución y la distancia entre pasos se reduce a la mitad.
Por último, están los micro pasos, los cuales se generan cuando se alimentan en simultáneo varios bobinados con corrientes medias distintas, logrando que el rotor se posicione prácticamente en cualquier posición deseada. La resolución es obviamente superior que la de pasos medios pero su controlador es bastante más complejo.
El motor que se utilizará para el movimiento del émbolo posee un controlador que permite identificar hasta 10.000 pasos por revolución. Lo cual es una excelente resolución.

Como comentario final los motores paso a paso son dispositivos mecánicos, al igual que los motores de continua deben vencer ciertas inercias. Dependiendo de los tiempos de duración y frecuencia de los pulsos, el comportamiento del motor puede variar. Este debe poder alcanzar el paso antes de que la próxima secuencia de pulsos comience, si la frecuencia es muy elevada puede que el motor no se mueva en absoluto o intente moverse, pero sin llegar a su posición. Por este motivo es recomendable usarlo de forma gradual, comenzando con frecuencias bajas e ir incrementando su frecuencia de a poco. Siempre teniendo en cuenta de no superar la velocidad máxima soportada.

2.1.2 Controlador/Driver

El controlador o también denominado driver de un motor, es un sistema dedicado al control del motor. Generalmente se utiliza para controlar la velocidad, el torque, y la dirección de un motor eléctrico. Cada sistema de control es diferente, depende íntimamente del motor, el fabricante y las funcionalidades implementadas. Sin embargo, hay ciertas funcionalidades comunes asociadas a cualquier motor eléctrico que están siempre presentes, como son la dirección, los micro pasos por vuelta, etc.
El motor que se utilizará posee un sistema de control que nos permite configurar varios parámetros, uno de ellos es la cantidad de pasos por revolución. Pudiendo definir si se utilizaran 10.000 pasos o 20.000. También puede definirse la señal que identificará la dirección de giro (un uno lógico significará un giro en la dirección de las agujas del reloj, un cero lógico, será el giro en sentido contrario)
Estos parámetros son configurados una única vez, utilizando un programa propietario, no siendo posible cambiar dicha configuración desde la CIAA-NXP.

2.1.3 Desarrollo del firmware controlador del motor

2.1.3.1 Timers

Los Timers, son esencialmente temporizadores o contadores, que pueden ser programados para generar interrupciones por hardware. Pueden detener la ejecución del programa, al cumplirse el tiempo establecido y ejecutar la rutina de atención a interrupción (ISR) correspondiente al timer. Este mecanismo, evita tener que monitorear el tiempo transcurrido de forma periódica. Solo se configura al inicio y se espera por las interrupciones.

La CIAA-NXP tiene un cristal de 204 MHz, por lo que podríamos programar una interrupción, teóricamente, cada 1/204.000.000 segundos. Esto no tiene mucho sentido, si tomamos en cuenta que cualquier instrucción que se le pida al micro va a demorar uno o más pulsos de reloj.

Cada Timer tiene un registro que indica cada cuantos ticks debe disparar la interrupción. A este se lo denomina CTR (compare match register). Si tuviéramos un Timer de 16 bits, esto significa que podríamos esperar como máximo, 65536 ticks. Si se hacen las cuentas, el máximo de tiempo que se podría programar sería:
(1/204.000.000) * 65536 = 0.00032 segundos = 0.32 milisegundos.
Seguramente, para más de una aplicación, se desea esperar más tiempo que ese, para esto existen los divisores o prescalers. Donde el timer solo incrementa en un tick el contador, cuando pasan cierta cantidad de ticks programados en el micro.

Una vez el timer llega al match, este se borra o resetea su valor en el siguiente tick y continúa con la cuenta hasta llegar al siguiente match. Seteando el Match y la velocidad a la que se incrementan los tick, se puede controlar la frecuencia de las interrupciones.

2.1.3.2 Implementación en el Firmware

El control del motor se hace mediante pulsos de longitud variable, siendo el flanco ascendente el determinante para incrementar el paso.
El driver permite controlar cada revolución del motor con hasta 20.000 pasos por vuelta. Sin embargo, con 10.000 pasos ya se logra el grado de precisión requerido.

El firmware programado en la CIAA-NXP, se encarga de generar los pulsos necesarios, utilizando el Timer 2. Este timer, convenientemente, tiene la particularidad de definir cierto pin como salida digital. Permitiendo alternar su valor a la salida, cada vez que se llega al match de dicho timer. En conclusión, únicamente cambiando el match del timer, se generan pulsos a distinta frecuencia, sin la necesidad de controlar de forma manual la salida digital.

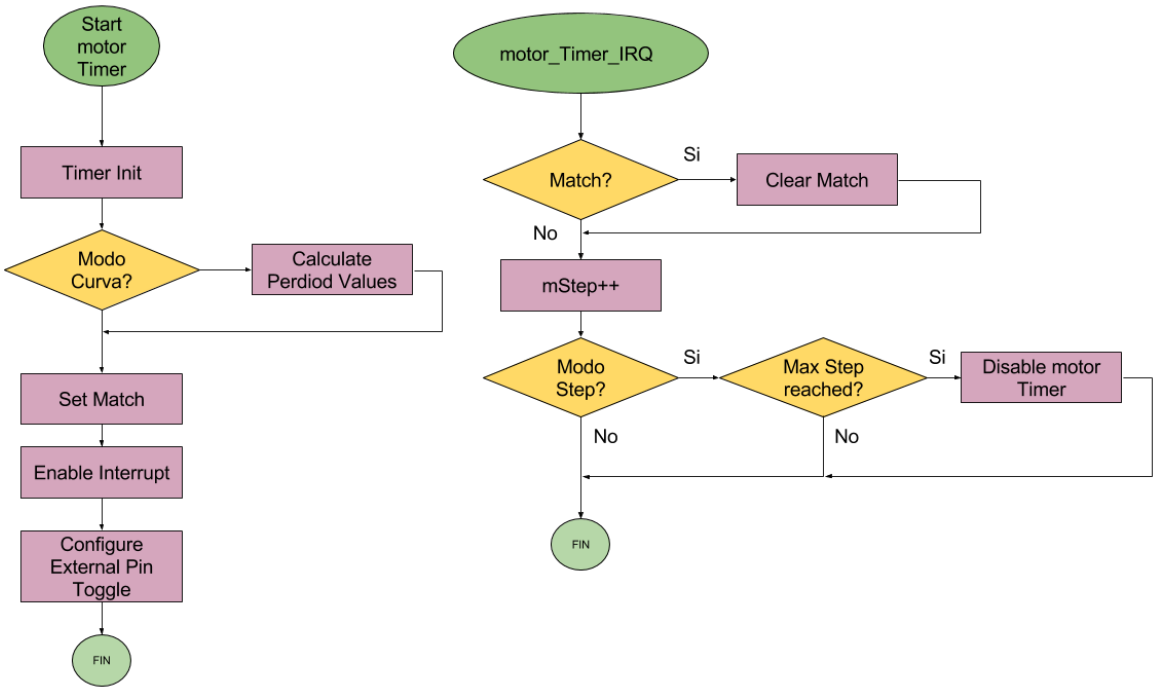


Figura 6: Diagrama de flujo del Timer del motor.

2.1.4 Relación pasos del motor vs. pasos tabla tiempo

La curva deseada se envía a la placa CIAA-NXP utilizando el software de PC desarrollado. El envío se hace utilizando el protocolo de comunicación lo que asegura una óptima recepción de los datos. Esta curva debe contener como máximo 4.000 puntos, siendo cada punto 1ms del tiempo total. Como puede suponerse el tiempo total de cada curva no puede superar los cuatro segundos. Una vez recibida, se la transforma, convirtiendo cada punto directamente al valor necesario para controlar al motor.

Las curvas que generalmente se usan para calibrar los espirómetros, son flujo en función del tiempo. Esta curva para poder ser enviada al motor debe derivarse a posición lineal en función del tiempo, luego la posición lineal corresponderá a una determinada cantidad de pasos del motor en función del tiempo lo que finaliza en frecuencia de pulsos en función del tiempo. Toda esta conversión debe realizarse antes de que la curva sea enviada a la CIAA-NXP.
Para poder realizar dicha conversión es necesario contar con cierta información.
1- Cuantos pasos corresponden a una revolución completa del motor
2- Cuántos mm lineales se desplaza el pistón por vuelta
3- Cuánto es el flujo desplazado por vuelta del motor.
Una vez que se obtuvieron esos datos la única corrección posible y necesaria corresponde a la corrección térmica. Al moverse el émbolo a velocidades muy altas, la compresión del aire dentro del pistón modifica su temperatura. Lo que genera errores en el cálculo teórico del flujo desplazado. Haciendo necesaria una corrección en la curva enviada.

2.1.5 Corrección de los pasos

El firmware desarrollado para la CIAA-NXP, debe ser capaz de recibir la curva de flujo transformada y controlar al motor de forma que su movimiento emule dicha curva. A cada milisegundo el firmware debe verificar la frecuencia deseada para ese punto y la frecuencia actual del timer, corrigiendo esta última de ser necesario. Esta corrección es únicamente para mantener el movimiento del motor dentro de la curva cargada, no se está realizando ninguna corrección por cambio de temperatura, este tipo de correcciones se realizan antes de la carga de la curva a la CIAA-NXP.

El timer de verificación es el Timer 0, durante las interrupciones de este timer se realizan las correcciones del match del Timer 2 el cual genera los pulsos para el motor. El Timer 0 es programado de forma que interrumpa la ejecución del firmware cada 1ms. En el momento de la interrupción, se verifica si el estado en el que se encuentra el match del motor es correcto y de no ser así corrige su valor.

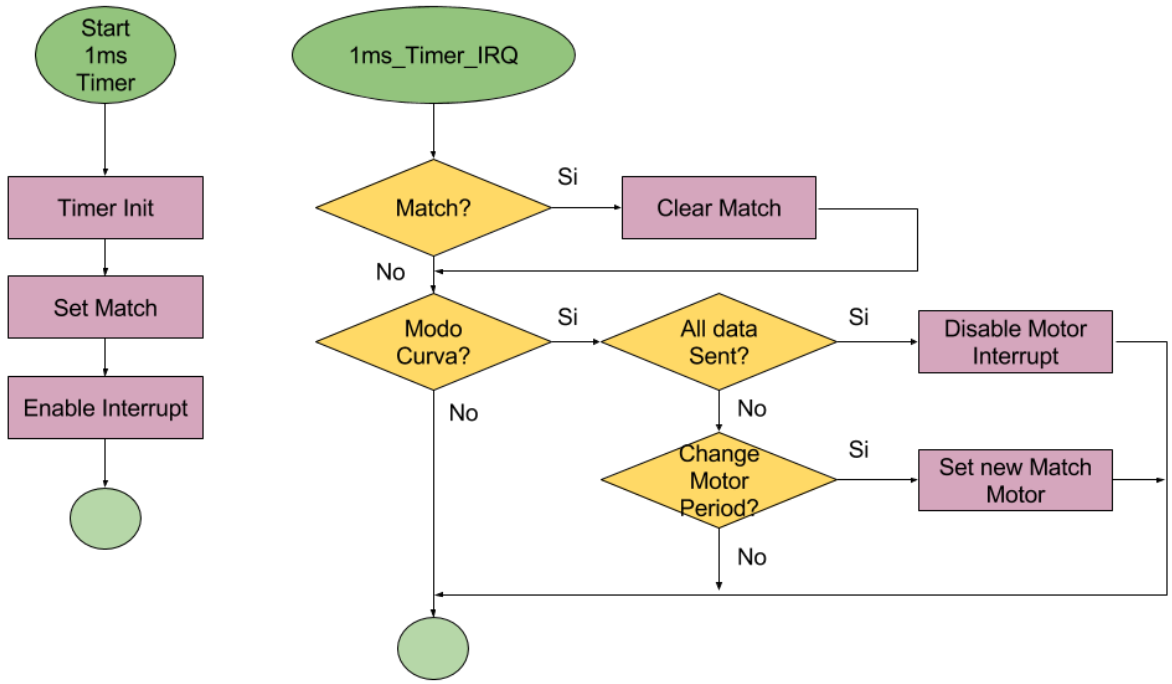


Figura 7: Diagrama de flujo, del Timer controlador del motor.

2.2 Detección de Fin de carrera

2.2.1 Definición del tipo de final de carrera

Los finales de carrera o Limit Switch, son componentes electrónicos, que nos permiten controlar las partes móviles de un dispositivo. Al ser activados, envían una señal al controlador, indicando que entró en contacto con el objeto físico.
Existen Finales de carrera mecánicos, ópticos y neumáticos.
Los finales de carrera que se utilizaron fueron, dos finales de carrera mecánicos, conectados en forma directa al Motor, por cuestiones de seguridad, generando un corte en la alimentación de este

en caso de ser accionados. Estos se encargan de detener el movimiento del motor en caso de sobrepasar el límite mecánico definido.
Además, se instalarán dos sensores ópticos, los cuales serán utilizados por la CIAA-NXP, para la detección de posición. Ambos sensores son de alta precisión, exactitud y repetibilidad.

Los finales de carrera pueden ser Normal Abierto, Normal Cerrado o Conmutados. Los sensores utilizados son Normal Abierto, esto significa que la detección del objeto será indicada a la CIAA-NXP cuando el sensor emita un "1" lógico.

2.2.2 Implementación

La conexión de los finales de carrera a la CIAA-NXP se realizó a través de los pines digitales opto acoplados. Se utilizaron distintos parámetros, en el código implementado, para configurar el estado de detección de estos, dependiendo de si el sensor utilizado era Normal Abierto o Cerrado.
En el Anexo 4, puede visualizarse un Diagrama de flujo, con la inicialización del firmware de la CIAA-NXP, junto con el loop principal en la función de arranque.

2.3 Control de seguridad del sistema

2.3.1 Interrupción en CIAA

Cuando se acciona uno de los finales de carrera, la EDU-CIAA genera una interrupción al controlador deteniendo de forma inmediata el timer que se encarga de controlar el motor.
Al detenerse el timer, dejan de salir pulsos, esto efectiviza el detenimiento del motor.

2.4 Protocolo de Comunicación

Para la comunicación interna entre la CIAA-NXP y el software de PC se implementó un protocolo de comunicación, que será transmitido sobre RS-232.
Para que cualquier sistema se comunique con otro, siempre es recomendable establecer un protocolo. Esto no es más que un conjunto de normas y reglas que permiten regularizar la comunicación entre ambos terminales.

2.4.1 RS-232

RS-232 (Recommended Standard-232) es una norma orientada al intercambio de datos binarios en serie, entre dos equipos. El puerto serial envía y recibe bytes de información de a un bit a la vez. Típicamente se utiliza para transmitir datos en formato ASCII, pero puede usarse con datos binarios. La conexión entre los equipos se hace a través de un conector DB-9. Los equipos modernos ya no tienen este tipo de conectores en sus motherboards, pero al poseer la placa CIAA-NXP un conector DB-9 integrado, se decidió usar ese conector a través de un adaptador USB-To Serial.

Para realizar la comunicación se necesitan como mínimo tres líneas de transmisión: (1) Tierra (o referencia), (2)Tx Transmitir, (3) Rx Recibir. Como la transmisión es asincrónica, es posible enviar datos por una de las líneas mientras se reciben datos por la otra.
Además de estas líneas de transmisión, existen otras líneas estas se enfocan en realizar el handshaking, o intercambio de pulsos de sincronización, pero no es obligatorio su uso, si bien es recomendable.
Para que dos puertos RS-232 puedan comunicarse, es necesario que las características de la comunicación serial sean iguales. Cuando se habla de características, esto hace referencia a la velocidad de transmisión, la cantidad de bits de datos a transmitir, si existe o no paridad, y cuántos bits se utilizan como bits de parada. En particular una configuración típica, es 8N1, 8 bits de datos, sin paridad y 1 bit de stop.

2.4.1.1 Velocidad de transmisión (baud rate)

Indica el número de bits por segundo que se transferirán por la línea. Cuanto mayor sea la velocidad de transferencia, menor debe ser la distancia del cable entre los dos terminales.
Por ejemplo, si se configura una velocidad de 9600, esto representa 9600 bits por segundo.

2.4.1.2 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. El driver de RS-232 se encargará de particionar el mensaje en grupos de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. Si el dato que se va a transmitir es únicamente ASCII estándar (números de 0 a 127), entonces puede utilizarse 7 bits por paquete. Pero si se utilizara el rango extendido de 0 a 255 como en nuestro caso, se deberá utilizar 8 bits.

2.4.1.3 Bits de parada

El bit de parada indica el fin de la transmisión de un paquete, los valores típicos son 1, 1.5 o 2. Generalmente se utiliza como margen de tolerancia en la transmisión, debido a que los equipos no utilizan el mismo reloj, puede que estos no estén sincronizados, cuanto mayor sea el bit de parada mayor será la tolerancia a la sincronización de los relojes.

2.4.1.4 Paridad

Es una forma sencilla de 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. Si la línea de transmisión no tiene ruido entonces no es necesario el uso de la paridad.
Par: El bit de paridad se establece de forma tal que, al hacer el recuento de los bits transmitidos, la cantidad de bits definidos en estado alto sea un número par.
Impar: El bit de paridad se establece de forma que el número de bits definidos en estado alto sea un número impar.
Marcada: Se coloca el bit de paridad en estado alto o “1” lógico.
Espaciada: Se coloca el bit de paridad en estado bajo o ”0” lógico.

Estos últimos no verifican el estado de los bits de datos, sino que fijan la paridad en un estado definido, de forma que el receptor conociendo el valor de este bit de antemano, identifique si existe ruido en la línea o si los relojes no están sincronizados.

La interfaz de la CIAA será configurada a 115200 kbps, con 8 bits de datos, 1 bit de stop, y sin paridad.

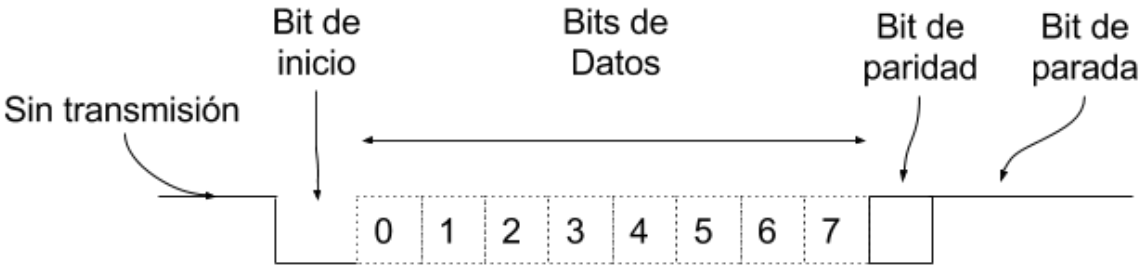


Figura 8: Paquete de transmisión asíncrona de un solo caracter.

2.4.1.5 Handshaking

Como se hizo referencia anteriormente, solo con tres líneas de transmisión es posible establecer una comunicación. Sin embargo, aun cuando ambos terminales están igualmente configurados, como los relojes son diferentes, puede existir cierto tipo de des-sincronización. Así también si uno de los terminales debe enviar un bloque grande de datos, durante un tiempo prolongado, y si el procesador receptor no es capaz de procesarlos a tiempo, este último puede saturarse. Por este motivo es importante el Handshaking. Existen a grandes rasgos dos tipos de handshaking, por software y por hardware.

2.4.1.5.1 Handshaking por software

Se utilizan bytes de datos como caracteres de control. Esto permite al usuario enviar a través de las líneas de transmisión el uso de dos caracteres, XON y XOFF. Cuando el receptor comienza a quedarse sin espacio libre en su buffer de recepción, envía el comando XOFF, para que la transmisión detenga el envío, y al liberarse espacio, envía el XON. El problema con la implementación y uso de este método es que si se envían datos binarios puede que se envié el valor XON y XOFF sin ser esta la intención. Si los valores que se transmitirán serán únicamente ASCII esto no importa mucho ya que estos valores (XON, XOFF) no representan carácter alguno.

2.4.1.5.2 Handshaking por Hardware

La implementación por hardware utiliza las líneas RTS/CTS y DTR/DSR. Estas líneas trabajan de manera conjunta. Cuando el receptor está listo para recibir datos levanta el nivel de la línea RTS (Request to send) que está conectado en el transmisor en la línea CTS (Clear to send) indicándole a este que puede enviar datos.
Cuando el equipo está listo para entablar una comunicación levanta la línea de DTR (Data terminal ready), que es leída por la otra terminal en DSR (Data Set Ready). Generalmente las líneas RTS/CTS se utilizan para paquetes individuales, mientras que DTR/DSR para indicar que el sistema está listo.

2.4.1.6 Hardware

Desde el lado del Microprocesador, la interfaz implementada es la U(s)ART (Universal Synchronous Asynchronous Receiver Transmitter). En la PC, es el puerto serie RS-232 que utiliza comúnmente un conector DB-9.

Si bien en la mayoría de los libros de teoría se habla de UART y RS-232 como si fueran lo mismo, no lo son.

UART es del lado del microprocesador y es el responsable de enviar y recibir una secuencia de bits. A la salida del UART, los bits son representados por tensiones de nivel lógico. Estos bits luego pueden convertirse a cualquiera de las interfaces RS-232, RS-422, RS-485, o alguna propietaria.

Por otro lado, RS-232 (en los computadores) es responsable de especificar los niveles de tensión. Cuanto mayor es el nivel de tensión, la interfaz es más resistente a las interferencias. Es necesario tener presente que la tensión puede alcanzar los ± 15V. Si miramos el UART, el microprocesador no es capaz de generar tales niveles de tensión por sí solo. Por eso se utiliza un componente adicional como el MAX232. Este componente consiste en un controlador de línea RS-232, que genera tensiones de ± 12V.

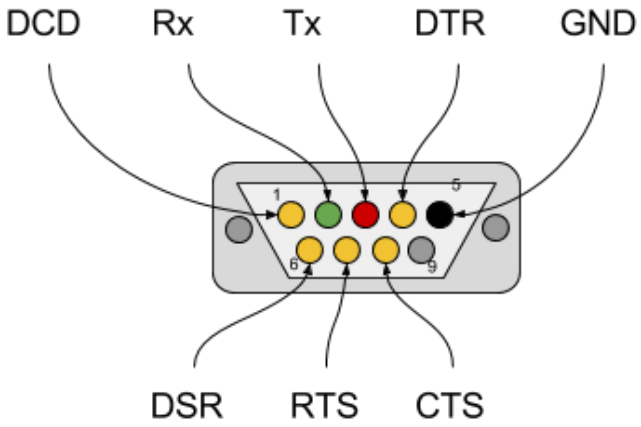


Figura 9: Conector DB-9 para conexión serial con la PC.

El MAX232 (de Maxim) es un circuito que adapta los niveles de tensión TTL que maneja el microprocesador, a niveles bajo la norma EIA/TIA-232E. Los niveles TTL operan entre los 0 y los 5 Volts, mientras que la norma RS-232 utiliza tensiones entre -12V y +12V. Para poder solucionar esta diferencia en los niveles de tensión se implementa un circuito muy sencillo con el circuito integrado MAX232.

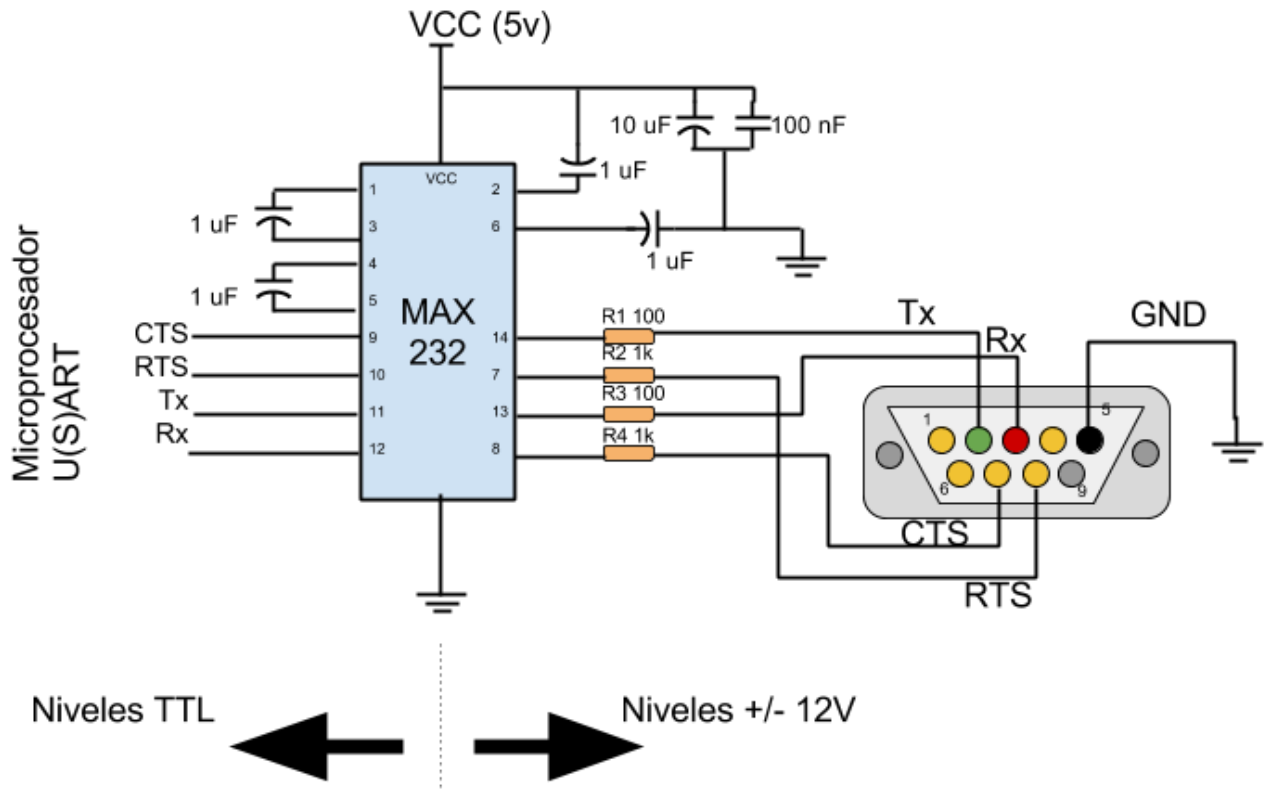


Figura 10: Esquemático de conexión de un MAX232 para adaptar los niveles de tensión.

Dicho circuito la CIAA-NXP ya lo tiene implementado, por lo que no fue necesario realizarlo de forma externa.

2.4.2 Tipos de Protocolo

Hasta ahora se explicó cómo era la conexión física para realizar una comunicación y el paquete básico de una transmisión asincrónica por UART. Sin embargo, el envío de solo un carácter no es para nada útil. Lo interesante sería poder enviar comandos y datos mucho más complejos entre ambas terminales. Esto puede implementarse de diferentes formas, pero lo más conveniente es la creación de un protocolo, donde se definan ciertas reglas y normas que deben cumplirse para que dos terminales se comuniquen.

Existen numerosos tipos de protocolo, en el campo de las redes informáticas la gran mayoría de los protocolos se basan en el modelo de comunicación OSI, como es TCP/IP, SNA, etc. Pero existe una gran variedad de implementaciones distintas.

2.4.2.1 Modelo OSI

Para generar un marco informativo es necesario repasar brevemente como es el modelo OSI. Este modelo fue desarrollado por ISO ("International Standards Organization"), con el objetivo de ser un protocolo estándar, que permite comunicar diferentes terminales o computadores. OSI significa "Open Systems Interconnection" y está compuesto por 7 capas:

2.4.2.1.1 Capa física

Es el medio físico del canal de transmisión, definiendo aspectos mecánicos y eléctricos, como es el valor de tensión que identificará los niveles lógicos “1” y “0”, el tiempo de duración de cada bit y la forma de conexión.

2.4.2.1.2 Capa de enlace

Su principal tarea es la corrección de errores, es responsable de que la transferencia de datos sea fiable a través de la línea de transmisión. En esta capa se identifican los límites de la trama.

2.4.2.1.3 Capa de red

Determina el enrutamiento de los paquetes, para que estos lleguen a destino. Eligiendo diferentes caminos, dependiendo de las condiciones de la línea, evitando la congestión que es responsabilidad de este nivel.

2.4.2.1.4 Capa de transporte

Su función principal consiste en aceptar los datos de la capa de sesión, dividirlos en unidades más pequeñas, pasarlos a la capa de red y asegurar que todos ellos lleguen correctamente al otro extremo de la manera más eficiente.

2.4.2.1.5 Capa de sesión

Proporciona los mecanismos para controlar el diálogo entre las aplicaciones de los sistemas.

2.4.2.1.6 Capa de presentación

Se encarga de codificar los datos según lo acordado previamente, posibilitando la comunicación de diferentes terminales con diferente tecnología.

2.4.2.1.7 Capa de aplicación

Es la capa que le da acceso al usuario al modelo OSI. define los protocolos que utilizan las aplicaciones para intercambiar información.

Si bien el protocolo propuesto para la comunicación con la CIAA-NXP, no es un protocolo de red. El modelo OSI sirve para explicar, cómo será su implementación.
Desde el punto de vista de la comunicación serie RS-232, la **capa física** queda definida por la interfaz serie. Desde el microprocesador la UART define los niveles lógicos TTL y desde el pc, los niveles RS232 de +- 12 V.
Las **capas de enlace y de red** se ven representadas por el paquete básico de una transmisión serial. En la cual se envía de a 1byte por vez con sus respectivas configuraciones (bit de parada, paridad, etc.).
Las **capas de sesión y transporte** se encargan de particionar un mensaje en bytes, y codificar cada byte, en paquetes básicos. Colocando su correspondiente bit de start, stop y paridad.

Por último, las **capas de presentación y aplicación** corresponden a la implementación del protocolo propiamente dicha. Donde se establecen una serie de reglas para el encapsulamiento de cada mensaje, y una configuración determinada.

2.4.3 Trama básica

En líneas generales se tendrán dos tipos de tramas. Aquellas que representen un comando u error, y aquellas encargadas de transmitir datos.
A continuación, se presenta la configuración básica para cada uno de esos paquetes.
Se puede observar que siempre el primer byte corresponde al caracter 0x55, este se denominará Start of Frame, o inicio de trama, y nos sirve para identificar donde inicia la trama. El último byte corresponde al CRC.

Paquete para un Comando/error

SOF	ID	Mode	**Command/Error**	CRC
0x55	**1byte**	**1byte**	**1byte**	**1byte**

Paquete para transmisión de datos.

SOF	ID	Mode	Param	Data			CRC
0x55	**1byte**	**1byte**	**1byte**	**1 byte**	**4 bytes**	**4 bytes**	**1byte**

A continuación, se presenta una serie de tablas, con los valores posibles que puede contener cada campo.

2.4.3.1 Descripción de cada campo

ID	**Descripción**
0x01	CIAA -> PC
0x02	PC -> CIAA

Mode	**Descripción**
0x01	COMMAND
0x02	PARAM
0x03	ERROR

Command	**Descripción**	
0x01	CMD_ACK	CIAA//PC
0x02	CMD_NACK	CIAA//PC

0x03	CMD_INIT_CIAA	PC -> CIAA
0x04	CMD_CIAA_INITIALIZED	CIAA -> PC
0x05	CMD_PC_INITIALIZED	PC -> CIAA
0x06	CMD_START_MOTOR	PC -> CIAA
0x07	CMD_STOP_MOTOR	PC -> CIAA
0x08	CMD_GET_INFO	PC -> CIAA
0x09	CMD_ALL_DATA_RECEIVED	CIAA -> PC

Param	**Descripción**	
0x01	PARAM_SET_MODE_VELOCITY_MAX_STEP	PC -> CIAA
0x02	PARAM_SET_DIRECTION_VELOCITY_STEP	CIAA -> PC
0x03	PARAM_SET_DIRECTION	PC -> CIAA
0x04	PARAM_SET_CURVE_MAX_POINTS	PC -> CIAA
0x05	PARAM_SET_CURVE_DATA	PC -> CIAA

Error	**Descripción**	
0x01	ERROR_CURVE_MAX_POINTS	CIAA -> PC
0x02	ERROR_UNKNOWN	CIAA//PC

Cada comando tiene identificado si es un comando de la PC a la CIAA o viceversa. A continuación, se muestra un ejemplo sencillo para cada uno de los comandos antes representados.

2.4.3.2 CMD_INIT_CIAA

Comando para inicializar la CIAA, se envía desde la PC, al conectarse al puerto COM.

SOF	ID	Mode	Command	CRC
0x55	0x02	0x01	0x03	**XOR**
	PC -> CIAA	COMMAND	CMD_INIT_CIAA	

Respuesta Esperada: CMD_CIAA_INITIALIZED

2.4.3.3 CMD_CIAA_INITIALIZED

Se envía desde la CIAA a la PC, cuando la termina de inicializarse.

SOF	ID	Mode	Command	CRC
0x55	0x01	0x01	0x04	**XOR**

	CIAA -> PC	COMMAND	CMD_CIAA_INITIALIZED	

Respuesta Esperada: CMD_PC_INITIALIZED

2.4.3.4 CMD_PC_INITIALIZED

Una vez recibido el comando indicando que la CIAA esta inicializada, y habiendo la PC terminado de inicializarse, se envía el siguiente comando para terminar con el proceso de inicialización.

SOF	ID	Mode	Command	CRC
0x55	0x02	0x01	0x05	**XOR**
	PC-> CIAA	COMMAND	CMD_PC_INITIALIZED	

Respuesta Esperada: CMD_ACK

A continuación, se presenta un esquema sencillo indicando el proceso de comunicación entre la CIAA y la PC, durante la inicialización.

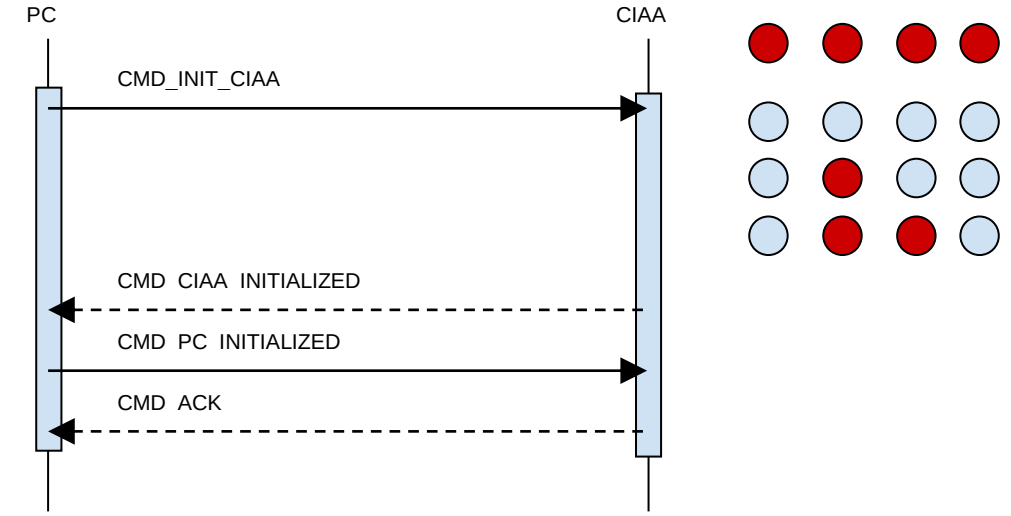


Figura 11: Proceso de inicialización. Los puntos corresponden a los LED de la placa CIAA-NXP

2.4.3.5 CMD_ACK

Este comando se envía cada vez que se recibió uno de forma exitosa. Significa que el mensaje fue recibido correctamente

SOF	ID	Mode	Command	CRC
0x55	0x01//0x02	0x01	0x01	**XOR**
	PC//CIAA	COMMAND	CMD_ACK	

Respuesta Esperada: None

2.4.3.6 CMD_NACK

Significa que el mensaje no fue recibido de forma exitosa y se pide retransmisión.

SOF	ID	Mode	Command	CRC
0x55	0x01//0x02	0x01	0x02	**XOR**
	PC//CIAA	COMMAND	CMD_NACK	

Respuesta Esperada: None

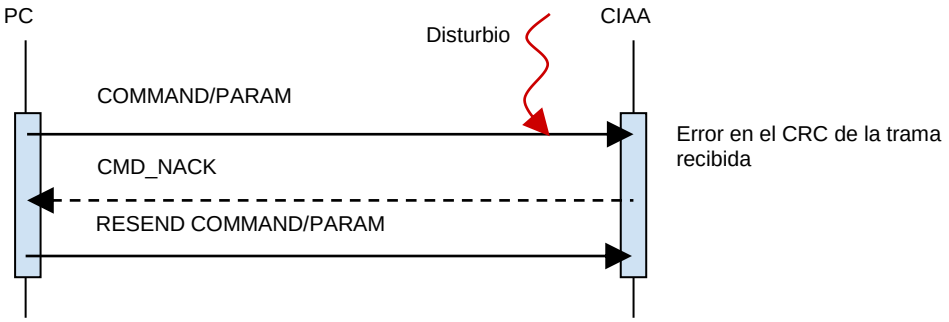


Figura 12: Trama recibida con error, pedido de re transmisión.

2.4.3.7 CMD_START_MOTOR

Se encarga de iniciar el motor. Los parámetros que tenga configurada la CIAA para el movimiento del motor serán los que utilice.

SOF	ID	Mode	Command	CRC
0x55	0x02	0x01	0x06	**XOR**
	PC-> CIAA	COMMAND	CMD_START_MOTOR	

Respuesta Esperada: CMD_ACK

Para un uso correcto, ver PARAM_SET_MODE_VELOCITY_MAX_STEP.

2.4.3.8 CMD_STOP_MOTOR

Detiene el timer del motor. Frenando el motor en el momento de recibir el comando.

SOF	ID	Mode	Command	CRC
0x55	0x02	0x01	0x07	**XOR**
	PC-> CIAA	COMMAND	CMD_STOP_MOTOR	

Respuesta Esperada: CMD_ACK

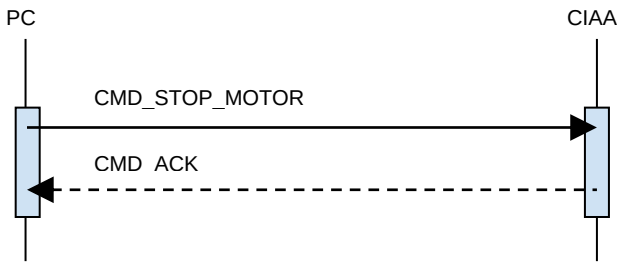


Figura 13: Envio de comando STOP, para frenar el motor.

2.4.3.9 CMD_GET_INFO

Se envía a la CIAA-NXP, solicitando el envío de los parámetros actuales del motor. Pudiendo obtener, dirección de movimiento, velocidad y cantidad de pasos realizados.

SOF	ID	Mode	Command	CRC
0x55	0x02	0x01	0x08	**XOR**
	PC-> CIAA	COMMAND	CMD_GET_INFO	

Respuesta Esperada: PARAM_SET_DIRECTION_VELOCITY_STEP

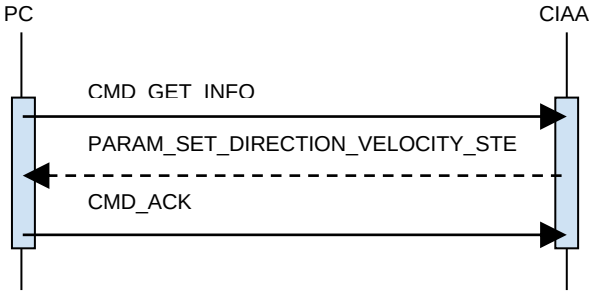


Figura 14: Pedido de parámetros actuales.

2.4.3.10 CMD_ALL_DATA_RECEIVED

Cuando se envía una curva que debe ser representada por el motor, al recibir el último dato, la CIAA responde con el siguiente comando indicando que la curva se recibió completa.

SOF	ID	Mode	Command	CRC
0x55	0x01	0x01	0x09	**XOR**
	CIAA ->PC	COMMAND	CMD_ALL_DATA_RECEIVED	

Respuesta Esperada: None

2.4.3.11 PARAM_SET_MODE_VELOCITY_MAX_STEP

Permite cargar los parámetros de funcionamiento del motor en la CIAA-NXP, el modo (Continuo, step, curve), la velocidad y la cantidad máxima de pasos.

SOF	ID	Mode	Param	Data			CRC
0x55	0x02	0x02	0x01	1 byte	4 bytes	4 bytes	**XOR**
	PC-> CIAA	PARAM	PARAM_SET_MODE_VELOCITY_MAX_STEP	0x01 = CONTINUO, 0x02 = STEP,	VELOCITY	MAX_STEP	

Respuesta Esperada: CMD_ACK

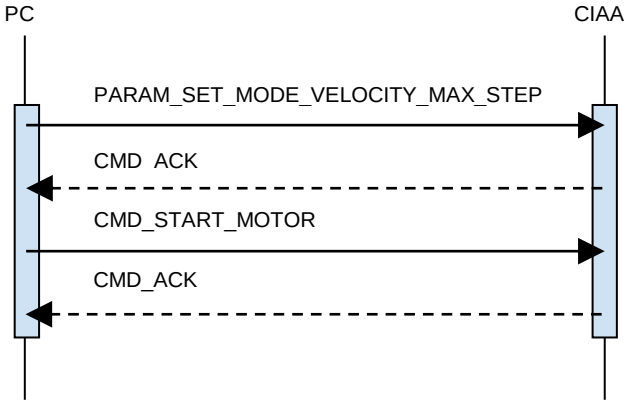


Figura 15: Envío de parámetros de configuración e inicio del motor

2.4.3.12 PARAM_SET_DIRECTION_VELOCITY_STEP

Parámetro enviado por la CIAA cuando se hace un pedido de información, indica dirección, velocidad y pasos realizados.

SOF	ID	Mode	Param	Data			CRC
0x55	0x01	0x02	0x02	1 byte	4 bytes	4 bytes	**XOR**
	CIAA -> PC	PARAM	PARAM_SET_DIRECTION_VELOCITY_STEP	0x01 = LEFT, 0x02 = RIGHT,	VELOCITY	STEP	

Respuesta Esperada: CMD_ACK

2.4.3.13 PARAM_SET_DIRECTION

Parámetro para cambiar la dirección de movimiento del motor. Esto podría haber sido representado por dos comandos distintos.

SOF	ID	Mode	Param	Data			CRC
0x55	0x02	0x02	0x03	1 byte	4 bytes	4 bytes	**XOR**
	PC-> CIAA	PARAM	PARAM_SET_DIRECTION	0x01 = LEFT, 0x02 = RIGHT,	None 0x00	None 0x00	

Respuesta Esperada: CMD_ACK

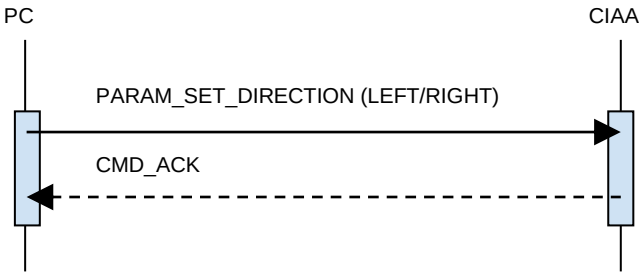


Figura 16: Pedido de cambio de direcciona la CIAA

2.4.3.14 PARAM_SET_CURVE_MAX_POINTS

Al enviar una curva a la CIAA, se indica el máximo número de datos a ser transmitidos, para asegurar una recepción completa de los datos.

SOF	ID	Mode	Param	Data			CRC
0x55	0x02	0x02	0x04	1 byte	4 bytes	4 bytes	**XOR**
	PC-> CIAA	PARAM	PARAM_SET_CURVE _MAX_POINTS	0x03 = CURVE	MAX_POINTS	None 0x00	

Respuesta Esperada: CMD_ACK

2.4.3.15 PARAM_SET_CURVE_DATA

Este parámetro se utiliza para enviar la curva desde la PC a la CIAA, pueden enviarse de a uno o dos parámetros por vez.

SOF	ID	Mode	Param	Data			CRC
0x55	0x02	0x02	0x05	1 byte	4 bytes	4 bytes	**XOR**
	PC-> CIAA	PARAM	PARAM_S ET_CURVE _DATA	0x01 = 1 dato 0x02 = 2 datos	Data 1	Data 2	

Respuesta Esperada: CMD_ACK

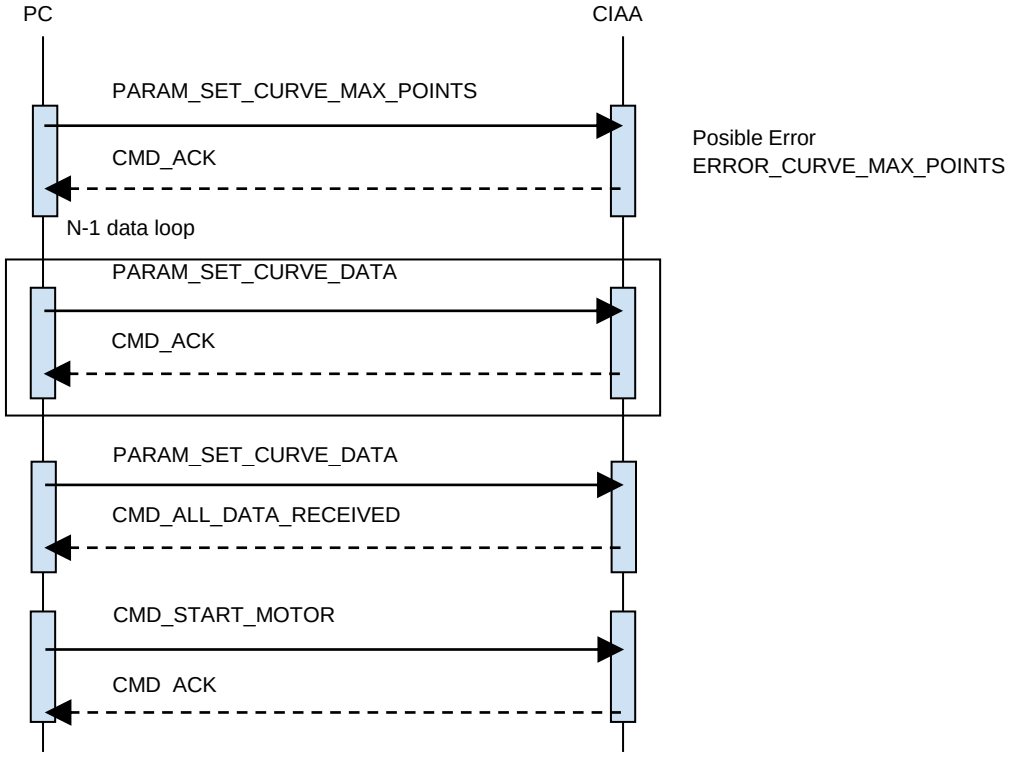


Figura 17: Transmisión de una curva al motor, e inicio de este.

2.4.3.16 ERROR_CURVE_MAX_POINTS

Ocurre cuando se intenta cargar una curva con un número de datos superior al permitido

por la CIAA

2.4.3.17 ERROR_UNKNOWN

Error desconocido, un ejemplo puede ser que no está implementada la trama recibida.

No es un error en el CRC, en esos casos se pide explícitamente una retransmisión con el comando NACK. Si el que envía la trama, sabe que el cliente debería poder manejarla, puede reenviar el paquete.

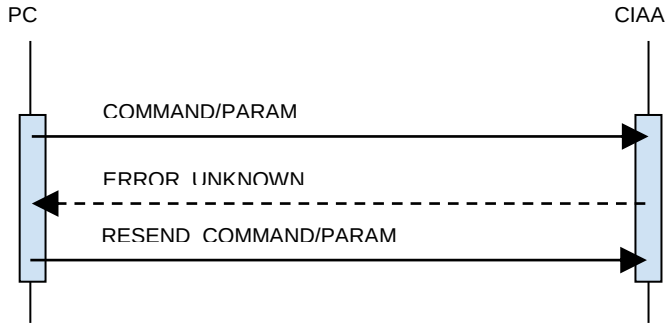


Figura 18: Ante la recepción de un Error_unknown la PC puede decidir hacer una retransmisión

3. Software Visualizador para PC

El software de PC se desarrolló en lenguaje C#, y se utilizó como entorno de desarrollo el Visual Studio 2012. La elección del lenguaje y el entorno se hizo en base a existir cierto conocimiento previo tanto del lenguaje como de la herramienta de compilación. C# es un lenguaje de alto nivel, orientado a objetos. Su sintaxis deriva del C o C++ siendo muy fácil su implementación.
Durante todo el proceso de desarrollo se intercambiaron consultas y alternativas, generalmente por medios virtuales con el coordinador del proyecto. Se utilizó una metodología de desarrollo ágil, sacando versiones tempranas que establecen objetivos específicos y podían ser testeados individualmente. Luego de varias iteraciones se cumplió con todos los requerimientos especificados.

3.1 Requerimientos

Los requerimientos notificados fueron:

- Realizar un movimiento continuo a una velocidad determinada.
- Realizar un movimiento continuo, pero frenar al llegar a la cantidad de pasos deseada.
- Envío de una curva de 4000 puntos a la CIAA-NXP para que ésta reproduzca el movimiento con el motor.
- Movimiento al punto de Origen.
- Detenimiento del motor en cualquier momento.
- Visualización de la curva a cargar.
- Cambiar el sentido de la dirección de movimiento.

Con estos requerimientos en vista se implementó el protocolo previamente explicado, con el que se establecieron los parámetros y comandos necesarios para realizar las operaciones requeridas.

3.2 Implementación de los Requerimientos

Previo a la implementación de los requerimientos se realizó un análisis de la arquitectura y se estableció un plan de trabajo con las prioridades establecidas.
En las primeras versiones el objetivo principal era crear un movimiento lineal en el motor, no siendo de gran importancia el protocolo de comunicación con la CIAA. Por el contrario, en la última versión uno de los puntos de mayor importancia era la robustez del protocolo de comunicación. Siendo de especial importancia, que el envío de cada punto de la curva a reproducir no tuviera errores en su recepción.
Durante todo el proceso de desarrollo se tuvo como objetivo, tener una herramienta capaz de comunicarse con la CIAA, que permitiera mover, detener o cambiar de sentido el movimiento del émbolo.

3.3 Diseño de arquitectura de software

El diseño de la arquitectura de software se refiere a la estructura global del software y a la manera en que ésta estructura, y proporciona integridad al sistema. Especifica una estructura jerárquica de los módulos del programa, la manera en la que estos interaccionan con los distintos componentes y la estructura de los datos usados por los diferentes módulos.
El software diseñado e implementado tiene codificadas tres clases que se interconectan entre sí. La clase GUI, se encarga de la interfaz gráfica, permitiendo mostrar los botones para poder interaccionar con la CIAA, la curva que será enviada, y los mensajes que notifican el estado de la CIAA. La clase protocol, es la que implementa el protocolo de comunicación, aquí se encuentran todas las funciones necesarias para crear y leer los mensajes, tanto los enviados, como los recibidos de la CIAA. Por último, está la clase SerialPortCIAA, la cual se encarga de hacer realmente el envío a través del serial port y de recibir los mensajes de la CIAA.
En el siguiente diagrama se muestra el entorno de participación de cada clase. La clase protocol únicamente interacciona con SerialPortCIAA y esta última con la GUI. De esa forma el código queda dividido en capas, permitiendo separar el comportamiento de cada una.

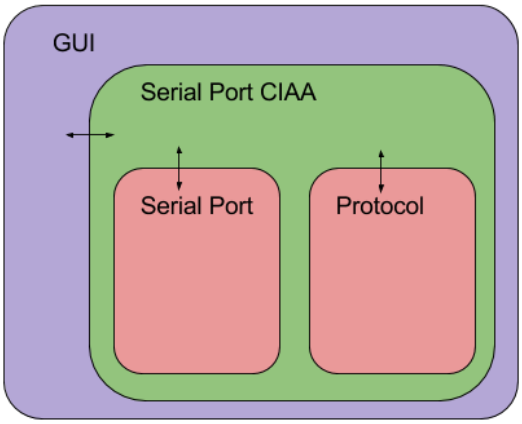


Figura 19: Herencia de las clases.

Siendo un poco más específicos, en el Anexo 5 puede observarse el diagrama de clases implementado y su interconexión. De esta forma se puede apreciar mejor, identificando el nombre de los objetos que tienen interacción con cada clase. Si bien el código no posee grandes

comentarios, es un código auto-comentado, donde simplemente viendo el nombre de la función o de las variables se puede identificar cuál es su función.

3.4 Estructura del Software

Un modelo básico de interacción que tendrá el usuario es el siguiente.

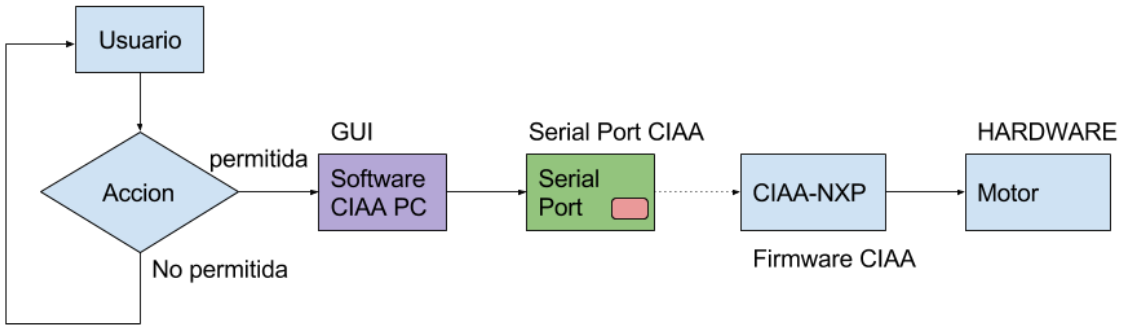


Figura 20: Diagrama de flujo , uso del software de PC

El usuario intentará realizar una operación, si esta está permitida en el estado actual del Software/Firmware, se le permitirá realizarla. El módulo de la interfaz gráfica tendrá los botones habilitados únicamente de aquellas funciones que están permitidas. Al presionar cualquiera de estas, se realizará la comunicación con el módulo del SeialPortCIAA quien codifica el mensaje de acuerdo con el protocolo y hace el envío de éste a través del serial port. La CIAA-NXP por su parte decodificará el mensaje y realizará las operaciones necesarias para realizar dicha acción.

3.5 Diagrama de Módulos

En el siguiente diagrama se identificarán todas las partes del sistema y las relaciones que existen entre ellas. El objetivo es presentar una perspectiva global de la arquitectura y sus componentes.

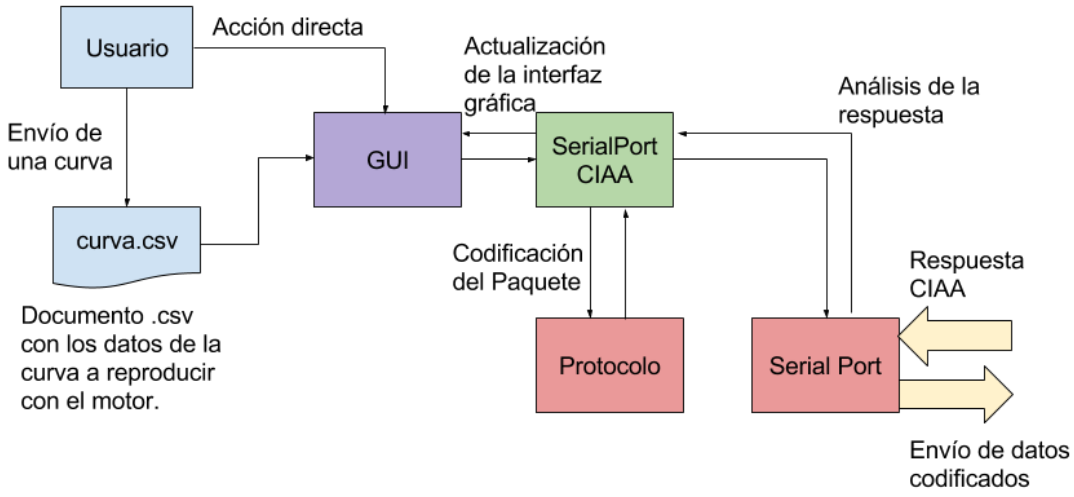


Figura 21: Diagrama de flujo, acción directa del usuario

3.6 Comunicación con la CIAA

La comunicación con la CIAA-NXP, como ya se especificó en anteriores oportunidades, se realizó a través del Serial Port. Los mensajes transferidos, ya sea en dirección a la CIAA o hacia la PC, fueron codificados a partir del protocolo implementando.
Uno de los puntos de mayor interés, se refiere al envío de la curva patrón, con los movimientos necesarios que deberá describir el émbolo. Estos datos son cargados en el software a través de un archivo .csv, el cual posee las frecuencias que se deberá configurar en el pwm del motor en cada milisegundo transcurrido.
La curva es visualizada en un gráfico permitiendo asegurarnos de que la curva a transferir es correcta.
Luego de visualizar la curva, esta puede enviarse hacia la CIAA. El encapsulado de cada dato nuevamente es realizado con el protocolo implementado y luego enviado a través del serial port.
Como el máximo tiempo de la curva es cuatro segundos, esto determina el máximo número de datos que puede tener el archivo .csv (4000 puntos).

3.6.1 Módulo de comunicación RS232

El módulo de comunicación RS232 se codificó utilizando el framework de windows, estando completamente resuelto el envío de datos.

La clase SerialPortCIAA, tiene declarado un objeto SerialPort, el cual se encarga de recibir y enviar los paquetes. Este objeto nos permite armar y decodificar los mensajes de manera sencilla.

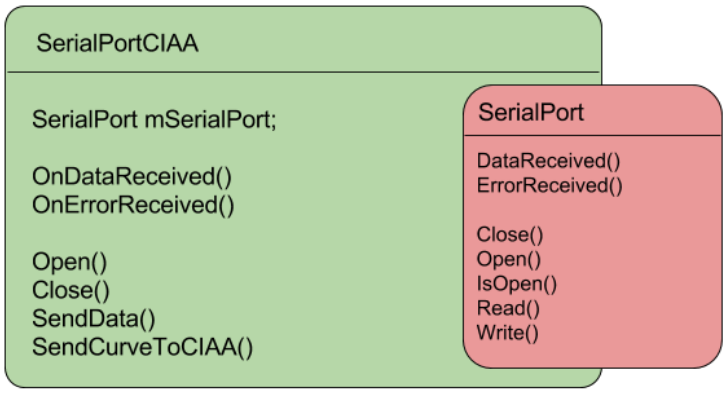


Figura 22: Funciones básicas, módulo de comunicación

El objeto SerialPort genera dos eventos, uno de ellos se invoca cuando tiene datos en la cola para ser leídos, y el otro cuando hubo un error en la línea de comunicación.

3.6.2 Lectura de paquetes

Cuando el SerialPort tiene datos listos para ser leídos, estos se guardan en un buffer y son procesados por el protocolo, de no haberse recibido todos los datos de un mismo paquete, se espera hasta que se haya finalizado de recibir todos los bytes necesarios.

El análisis del paquete lo hace el objeto CIAAProtocol, quien implementa precisamente el protocolo.

Una vez recibidos todos los datos se procesa el paquete. Se ejecutan las acciones necesarias y se envía una respuesta de ser necesario.

3.6.3 Escritura de paquetes

Cuando se deben enviar paquetes a la CIAA, se utiliza el comando write del SerialPort. Previamente el paquete es codificado por el objeto CIAAProtocol, quien implementa el protocolo. De ser necesario se espera por una respuesta antes de enviar el siguiente paquete.

4. Conclusiones

Siendo el objetivo de este trabajo, el desarrollo de un firmware controlador del equipo. En esta versión entregada, si bien es una versión preliminar, cumple con todos los objetivos planteados.
Los puntos de mayor complejidad corresponden a la implementación de un protocolo de comunicación, que fuera extensible a nuevos comandos y a su vez de fácil implementación y uso. Y el envío de una curva patrón, que pueda ser reproducida a través del movimiento del motor.
El resto de los objetivos no fueron de gran complejidad, aunque esto no significa que no demoraron su respectivo tiempo de desarrollo.

La investigación y posterior desarrollo e implementación del protocolo de comunicación, fue en gran medida uno de los pilares en los que se apoyó tanto el firmware como el software de PC. Este protocolo nos asegura que la comunicación entre ambos sistemas es confiable y que los datos recibidos/enviados son correctos y libre de errores.

El segundo punto de mayor complejidad fue la utilización de la curva patrón para el control del motor. Como las interrupciones no son sincrónicas, respecto de la carga y configuración de la curva a valores que sean usables por el PWM implementado. En un principio se comenzaba a usar la curva antes de que esta fuera completamente transformada. Generando errores de lectura y hasta movimientos erráticos en el motor. En una ocasión el valor enviado al motor era de tal magnitud, que impacto con una fuerza importante sobre uno de los finales de carrera, destruyendo dicho sensor. Estos inconvenientes lejos de deprimir obligaron a aumentar las verificaciones y ajustar las variables del firmware que se utilizaban para los controles del motor.

En la forma en que fue desarrollado el firmware, siempre estuvo presente que este proyecto representaba solo una parte inicial del proyecto General. Por tal motivo el software y firmware esta autodocumentado, con variables con nombres bien identificados y con posibilidad de expansión, sin muchos cambios drásticos en el código fuente.

5. Glosario

.NET	Plataforma Microsoft de desarrollo.
ACK	Acrónimo de Acknowledgement (Confirmación).
API	Application Program Interface (Interfaz de programa de aplicación).
Aplicación	Programa utilizado para realizar un determinado tipo de trabajo.
ASCII	American Standard Code of Information Interchange. Esquema estándar de codificación de caracteres de un byte.
ASYNC	Asyncronous (Asíncrono).
Asíncrono	Comunicación realizada entre un emisor y un receptor en tiempos diferentes.
Baudios	Numero de cambios de estado de una señal por segundo.
BaudRate	Velocidad de transmisión.
Binario	Sistema de numeración de base 2 que utiliza los símbolos 0 y 1;
Bit	Binary digit, Dígito binario.
Bloque	Conjunto de caracteres enviados conjuntamente durante una comunicación.
Buffer	Espacio en memoria que se utiliza para almacenar datos.
Bug	Error en un programa.
Byte	Unidad mínima direccionable de datos. 8 bits forman 1 byte.
C#	C Sharp, Lenguaje de Programación nativo de .NET
Canal	Vía de comunicación de datos interna/
Caracter	Unidad mínima de escritura identificada por 1 byte.
Checksum	Suma de comprobación realizada con compuertas XOR.
CIAA-NXP	Computadora Industrial Abierta Argentina.
Controlador	Driver.
CRC	Acrónimo de Cyclic Redundancy Check (Comprobación de redundancia cíclica).
CSV	Comma Separated Values. Archivo de valores separados por coma.
Curva Patrón	Curva de referencia de flujo en función del tiempo.
DB-9	Zócalo y/o enchufe con 9 clavijas que se usa para conectar dispositivos de comunicación.
DCD	Data Carrier Detect.
Driver	Programa pequeño que controla un periférico o algún dispositivo físico.
DTR/DSR	Data Terminal Ready / Data Set Ready
EIA/TIA-232E	Versión del Estándar de comunicación serie RS-232
Estator	Parte fija del motor.
Espirometría	Técnica que mide los flujos y los volúmenes respiratorios útiles para el diagnóstico y el seguimiento de patologías respiratorias.
Espirómetro	Equipo que realiza las mediciones de los flujos y volúmenes respiratorios.
Firmware	Un programa software que no puede ser alterado dinámicamente durante su funcionamiento normal.
Flujo	Caudal de aire.
Framework	Paquete integrado. Consiste en una serie de herramientas y bibliotecas.
GND	Ground. Tierra.
GUI	Graphical User Interface (Interfaz Gráfica de Usuario)
Handshaking	Intercambio de pulsos de sincronización.
IRQ	Solicitud de Interrupción.
ISO	International Standards Organization (Modelo para sistemas de Red)
kbps	Kilobits por segundo

Limit Switch	Finales de carrera.
Match	Coincidir con el valor programado.
MAX232	Circuito integrado de la marca Maxim Integrated Circuit para convertir señales RS-232 a TTL.
Micro	Unidad de proceso contenida en un chip situado en una plaqueta.
Motherboards	Placa Madre de una PC.
ms	MiliSegundo.
OSI	Modelo OSI, Open Systems Interconnection
Paquete	Unidad de transmisión que consta de información binaria que representa datos.
PC	Programa de computadora.
Pin	Contacto de una plaqueta.
Prescalers	Divisores.
Protocolo	Conjunto de reglas establecidas para fijar la forma en que se realizan las transacciones.
Pulso	Duración corta de una onda o señal.
Rotor	Parte móvil del motor.
RS-232	Recommended Standard-232. Estándar de comunicación serie entre el computador y los periféricos.
RTS/CTS	Request to send / clear to send
Rx	Línea de Recepción de datos.
Sensor	Dispositivo cuyo fin es recibir estímulos y transformarlos en información.
SOF	Start of Frame. Caracter de inicio de trama.
Software	Programa, procedimientos y reglas asociados concernientes a la operación de un sistema de PC.
Tick	Unidad de incremento del Timer.
Timer	Temporizador, reloj.
Trama	Enmarca el principio y fin de cada fragmento de datos.
TTL	Niveles de tensión que van dentro de una ventana de 5 voltios.
Tx	Línea de Transmisión de datos.
U(s)ART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus.
USB-To Serial	Adaptador de señal USB a Serial RS-232.
V	Volt.
VCC	Unidad de tensión de corriente continua.
XON/XOFF	Transmisor preparado/Transmisor ocupado.

6. Bibliografía

[1] J. R. M. D. J. V. J.L. Hankinson, «Method to produce American Thoracic Society flow-time,» *Copyright ERS Journals Ltd,* 1997.

[2] I. E. Pernia, «Asignación de Pines, CIAA-NXP,» 05 ABR 2016. [En línea].

[3] J. P. V. Joaquín Rodríguez, «Módulo 3, Creando un nuevo proyecto en la EDU-CIAA,» Buenos Aires, Argentina, 2016.

[4] C. Team, «Schematic CIAA - Computadora Abierta Argentina Version NXP,» Buenos Aires, Argentina, http://www.proyecto-ciaa.com.ar/, 2016.

[5] N. B. 2015, «User manual UM10503, LPC43xx/LPC43Sxx ARM Cortex-M4/M0 multi-core,» 2015.

[6] H. M. C. a. s. Technology, «D2 Drive User's Manual,» 2016.

[7] M. c. a. S. T. HIWIN, «AC Servo Motor & Drive Technical Information,» 2015.

[8] M. c. a. S. T. HIWIN, «AC Servo Motor & Drive Technical Information,» 2016.

[9] M. Integrated, «MAX232, MAX232A,» 2017. [En línea]. Available: https://www.maximintegrated.com/en/products/interface/transceivers/MAX232.html.

[10] RS-232, «RS-232,» 2017. [En línea]. Available: https://es.wikipedia.org/wiki/RS-232.

[11] I. Gottlieb, Electric Motors & Control Techniques (2nd ed.), TAB Books, 1994.

[12] B. G. Liptak, Instrument Engineers' Handbook: Process Control and Optimization., CRC. Tylor & Francis, 2005.

[13] P. P. Acarnley, Stepping motors: a guide to modern theory and practice, P. Peregrinus on behalf of the IEE, 1984.

[14] Y. e. William H. Yeadon and Alan W, Handbook of Small Electric Motors, McGraw-Hill, 2001.

Anexo 1 – WBS

Task Mode	WBS	Nombre de tarea	Duration	Predecessors	Resource Names
Auto Scheduled	**1**	**Gestión de Proyecto**	**56,06 days**		**Natalia;Victor**
Auto Scheduled	1.1	Definición del Alcance y Propósito del proyecto	24 hours		Victor;Natalia
Auto Scheduled	1.2	Supuestos del proyecto	24 hours		Natalia;Victor
Auto Scheduled	1.3	Requerimientos	48 hours		Natalia;Victor
Auto Scheduled	**1.4**	**Definición de los Entregables**	**5 days**		**Natalia;Victor**
Auto Scheduled	1.4.1	Hardware	1 day		Natalia;Victor
Auto Scheduled	1.4.2	Firmware	1 day		Natalia;Victor
Auto Scheduled	1.4.3	Software	1 day		Natalia;Victor
Auto Scheduled	1.5	Desglose en WBS	34,5 hours	4;5	Natalia;Victor
Auto Scheduled	1.6	Diagrama de Gantt	8 hours	9	Natalia;Victor
Auto Scheduled	1.7	Armado de Informes	60 hours		Natalia;Victor
Auto Scheduled	**1.8**	**Reuniones**	**14,19 days**		**Natalia;Victor**
Auto Scheduled	1.8.1	Reunión semanal Definición del proyecto, asignación de roles	1 hour		Natalia;Victor
Auto Scheduled	1.8.2	Presentación CIAA-NXP	6 hours		Natalia
Auto Scheduled	1.8.3	Ajuste de requisitos, Definición del Motor	4 hours		Natalia;Victor
~~Auto Scheduled~~	~~1.8.4~~	~~Reunión para el Análisis de la Regla digital~~	~~3 hours~~		~~Natalia;Victor~~
Manually Scheduled	**1.9**	**Gestión de Riesgos**	**6,06 days**		**Natalia;Victor**
Auto Scheduled	1.9.1	Identificación de los riesgos	1 day		Natalia;Victor
Auto Scheduled	1.9.2	Estimación de ocurrencia	1 day		Natalia;Victor
Auto Scheduled	1.9.3	Estimación de consecuencia	1 day		Natalia;Victor
Manually Scheduled	**1.10**	**Gestión de Calidad**	**6,06 days**		**Natalia;Victor**
Auto Scheduled	1.10.1	Grado de Calidad	1 day		Natalia;Victor
Auto Scheduled	1.10.2	Costos de Conformidad	1 day		Natalia;Victor
Auto Scheduled	**2**	**Documentos**	**10 days**	**4;5;40;72**	**Natalia;Victor**
Auto Scheduled	**2.1**	**Manual de Usuario**	**10 days**		**Natalia;Victor**
Auto Scheduled	2.1.1	Definición de los Requerimientos	4 hours		Natalia;Victor
Auto Scheduled	2.1.2	Diseño	4 hours		Natalia;Victor
Auto Scheduled	2.1.3	Primer Borrador	12 hours		Natalia;Victor
Auto Scheduled	2.1.4	Versión Final	24 hours		Natalia;Victor
~~Auto Scheduled~~	~~2.2~~	~~Manual de Calibración~~	~~10 days~~		~~Natalia;Victor~~
~~Auto Scheduled~~	~~2.2.1~~	~~Definición de los Requerimientos~~	~~4 hours~~		~~Natalia;Victor~~
~~Auto Scheduled~~	~~2.2.2~~	~~Diseño~~	~~4 hours~~		~~Natalia;Victor~~

~~Auto Scheduled~~	~~2.2.3~~	~~Primer Borrador~~	~~12 hours~~		~~Natalia;Victor~~
~~Auto Scheduled~~	~~2.2.4~~	~~Versión Final~~	~~24 hours~~		~~Natalia;Victor~~
Auto Scheduled	**2.3**	**Manual de Service**	**10 days**		**Natalia;Victor**
Auto Scheduled	2.3.1	Definición de los Requerimientos	4 hours		Natalia;Victor
Auto Scheduled	2.3.2	Diseño	4 hours		Natalia;Victor
Auto Scheduled	2.3.3	Primer Borrador	12 hours		Natalia;Victor
Auto Scheduled	2.3.4	Versión Final	24 hours		Natalia;Victor
Auto Scheduled	**3**	**Hardware**	**75,06 days?**	**4;16**	**Victor**
~~Auto Scheduled~~	~~3.1~~	~~Control de posición~~	~~15 days~~		~~Victor~~
~~Auto Scheduled~~	~~3.1.1~~	~~Pruebas de salidas de señal del sensor de posición 16hs~~	~~12 hours~~		~~Victor~~
~~Auto Scheduled~~	~~3.1.2~~	~~Adaptación de señal SIN & COS~~	~~10 days~~		~~Victor~~
~~Auto Scheduled~~	~~3.1.2.1~~	~~Investigación sobre AO~~	~~10 hours~~	~~42~~	~~Victor~~
~~Auto Scheduled~~	~~3.1.2.2~~	~~Elección de componentes para prototipo~~	~~2 hours~~	~~44~~	~~Victor~~
~~Auto Scheduled~~	~~3.1.2.3~~	~~Armado de prototipos~~	~~12 hours~~	~~45~~	~~Victor~~
Auto Scheduled	**3.2**	**Motor actuador**	**14,13 days**	**41**	**Victor**
Auto Scheduled	3.2.1	Definición de motor a utilizar	12 hours		Victor
Auto Scheduled	**3.2.2**	**Desarrollo de salidas de potencia**	**5 days**		**Victor**
Auto Scheduled	3.2.2.1	Elección de componentes para prototipo	2 hours	48	Victor
Auto Scheduled	3.2.2.2	Armado de prototipos	12 hours	50	Victor
Auto Scheduled	3.2.3	Pruebas de fuerza y velocidad	1 day	51	Victor
Auto Scheduled	**3.3**	**Sensor de detección de final de carrera**	**10,88 days**	**47**	**Victor**
Auto Scheduled	3.3.1	Investigación sobre sensado mecánico y óptico	12 hours		Victor
Auto Scheduled	**3.3.2**	**Adaptación de señales**	**5,94 days**		**Victor**
Auto Scheduled	3.3.2.1	Elección de componentes para prototipo	2 hours	54	Victor
Auto Scheduled	3.3.2.2	Armado de prototipos	12 hours	56	Victor
Auto Scheduled	3.3.3	Montaje de sensores	4 hours	57	Victor
Auto Scheduled	**3.4**	**Control de seguridad**	**23,13 days?**	**53**	**Victor**
Auto Scheduled	3.4.1	Análisis de factores de riesgo	18 hours		Victor
Auto Scheduled	3.4.2	Protección electrónica por sobre corrientes y tensiones	18 hours	60	Victor
Auto Scheduled	3.4.3	Integración con el sistema de final de carrera	8 hours	61	Victor
Auto Scheduled	3.4.4	Corte de energía de los motores	1 day?	62	Victor
Auto Scheduled	3.4.5	Pruebas de seguridad	4 hours	63	Victor
Auto Scheduled	**3.5**	**Sistema de control de temperatura**	**17,06 days**	**59**	**Victor**
Auto Scheduled	3.5.1	Definición de los sensores a utilizar	18 hours		Victor
Auto Scheduled	3.5.2	Diseño y montaje de sistema “standalone” para control	18 hours	66	Victor
Auto Scheduled	3.5.3	Montaje de los sensores	4 hours	67	Victor

Auto Scheduled	3.5.4	Armado de prototipo	6 hours	68	Victor
Auto Scheduled	**3.6**	**Prototipado Final en PCB**	**9,06 days**	**65**	**Victor**
Auto Scheduled	3.6.1	Armado y pruebas	24 hours		Victor
Auto Scheduled	**4**	**Firmware CIAA**	**65 days**		**Natalia**
~~Auto Scheduled~~	~~4.1~~	~~Control de posición~~	~~23 days~~		~~Natalia~~
~~Auto Scheduled~~	~~4.1.1~~	~~Estudio de conexión con la CIAA a nivel Hardware~~	~~8 days~~	~~16~~	~~Natalia~~
~~Auto Scheduled~~	~~4.1.3~~	~~Desarrollo de software contador de pulsos~~	~~6 hours~~	~~41;77;74~~	~~Natalia~~
~~Auto Scheduled~~	~~4.1.4~~	~~Uso del DSP para la generación del diente de sierra~~	~~8 hours~~	~~78~~	~~Natalia~~
~~Auto Scheduled~~	~~4.1.5~~	~~Generación de curvas Patrón en Posición~~	~~10 hours~~	~~79~~	~~Natalia~~
~~Auto Scheduled~~	~~4.1.6~~	~~Censado del desplazamiento~~	~~4 hours~~	~~80~~	~~Natalia~~
~~Auto Scheduled~~	~~4.1.7~~	~~Contrastado con tablas de posición~~	~~6 hours~~	~~81~~	~~Natalia~~
Auto Scheduled	**4.2**	**Control del actuador del motor**	**16,13 days**	**47**	**Natalia**
Auto Scheduled	4.2.1	Investigación sobre el hardware a utilizar	6 hours		Natalia
Auto Scheduled	4.2.2	Desarrollo del software controlador del motor	6 hours	84	Natalia
Auto Scheduled	4.2.3	Relación pasos del motor vs. pasos tabla tiempo	18 hours	85	Natalia
Auto Scheduled	4.2.4	Carga de tablas pasos según tiempo	6 hours	86	Natalia
Auto Scheduled	4.2.5	Corrección de los pasos	6 hours	87	Natalia
Auto Scheduled	**4.3**	**Control del lazo de realimentación8 days**	**7,06 days**	**73;83**	**Natalia**
~~Auto Scheduled~~	~~4.3.1~~	~~Verificación de la posición de la regla vs, velocidad/posición del motor~~	~~2 days~~		~~Natalia~~
Auto Scheduled	4.3.2	Corrección de la posición/velocidad del motor	16 hours	90	Natalia
Auto Scheduled	**4.4**	**Detección de Fin de carrera**	**3,06 days**	**53;89**	**Natalia**
Auto Scheduled	4.4.1	Investigación sobre el hardware a utilizar	4 hours		Natalia
Auto Scheduled	4.4.2	Deteccion de ocurrencia y notificacion	6 hours	93	Natalia
Auto Scheduled	**4.5**	**Control de seguridad del sistema**	**10 days**	**59;92**	**Natalia**
Auto Scheduled	4.5.1	Interrupción en CIAA (Alta prioridad)	10 hours		Natalia
Auto Scheduled	4.5.2	Tarea de comunicación a alto nivel	10 hours	96	Natalia
Auto Scheduled	4.5.3	Función de RESET del sistema	6 hours	97	Natalia
~~Auto Scheduled~~	~~4.6~~	~~Control de Temperatura~~	~~10 days~~	~~65;95~~	~~Natalia~~
~~Auto Scheduled~~	~~4.6.1~~	~~Investigación sobre forma de censado~~	~~4 hours~~		~~Natalia~~
~~Auto Scheduled~~	~~4.6.2~~	~~Rutina de control de temperatura~~	~~10 hours~~	~~100~~	~~Natalia~~
~~Auto Scheduled~~	~~4.6.3~~	~~Alarmas por alta/Baja Temperatura~~	~~10 hours~~	~~101~~	~~Natalia~~
Auto Scheduled	**4.7**	**Protocolo de Comunicacion**	**20 days**	**99**	**Natalia**
Auto Scheduled	4.7.1	Recepción de comandos (funcionalidades)	1 day		Natalia
Auto Scheduled	**4.7.2**	**Comunicación con PC**	**16,88 days**		**Natalia**
Auto Scheduled	4.7.2.1	Rutina para enviar a la PC las lecturas de la regla correspondientes a la última curva	10 hours	104	Natalia

Auto Scheduled	4.7.2.2	Rutina para enviar a la PC las posiciones del motor de la última curva	10 hours	106	Natalia
Auto Scheduled	4.7.2.3	Rutina para enviar a la PC las temperaturas censadas en la última curva	10 hours	107	Natalia
Auto Scheduled	4.7.2.4	Salvado de curvas patrón en la EEPROM	12 hours	108	Natalia
Auto Scheduled	**5**	**Software Visualizador para PC**	**11,13 days?**	**72**	**Natalia**
Auto Scheduled	5.1	Requerimientos	1 day?		
Auto Scheduled	5.2	Diseño	1 day?		
Auto Scheduled	5.3	Aplicación GUI en C#	6 hours		Natalia
Auto Scheduled	**5.4**	**Comunicación con la CIAA**	**10,06 days**		**Natalia**
Auto Scheduled	5.4.1	Modulo de comunicación RS232 o similar	6 hours		Natalia
Auto Scheduled	5.4.2	Rutinas de Lectura de datos	8 hours		Natalia
Auto Scheduled	5.4.3	Formateo de datos para pos proceso (si vienen en array, uno detrás de otro, cargar distintas estructuras)	12 hours		Natalia
Auto Scheduled	5.4.4	Envío de comandos a la CIAA	6 hours		Natalia
Auto Scheduled	5.4.5	Carga de curvas Patrón en la CIAA	6 hours		Natalia
Auto Scheduled	5.5	modulo para Graficar puntos	10 hours		Natalia
Auto Scheduled	5.6	Visualización de curvas patrón	6 hours		Natalia
~~Auto Scheduled~~	~~5.7~~	~~Creación de informes~~	~~10 hours~~		~~Natalia~~
Auto Scheduled	**6**	**Integración**	**12,19 days**	**40;72**	**Natalia;Victor**
Auto Scheduled	6.1	Verificación de la compatibilidad de las partes	1 day		Natalia;Victor
Auto Scheduled	6.2	Ensamble de las partes mecánicas	1 day	124	Natalia;Victor
Auto Scheduled	6.3	Ensamble de las partes electrónicas	1 day	125	Natalia;Victor
Auto Scheduled	6.4	Ensamble del controlador con el pistón	1 day	126	Natalia;Victor
Auto Scheduled	**7**	**Verificación y Validación**	**9,06 days**	**123**	**Natalia;Victor**
Auto Scheduled	7.1	Test de componentes	1 day		Natalia;Victor
~~Auto Scheduled~~	~~7.2~~	~~Desarrollo de ensayos~~	~~1 day~~	~~129~~	~~Natalia;Victor~~
~~Auto Scheduled~~	~~7.3~~	~~Ensayos con curva Patrón~~	~~1 day~~	~~130~~	~~Natalia;Victor~~
Auto Scheduled	**7.4**	**Validación de la documentación**	**7,06 days**		**Natalia;Victor**
Auto Scheduled	7.4.1	Manuales de Uso	1 day	131	Natalia;Victor
Auto Scheduled	7.4.2	Manuales de Service	1 day	133	Natalia;Victor
~~Auto Scheduled~~	~~8~~	~~Calibración~~	~~9,06 days~~	~~128~~	~~Natalia;Victor~~
~~Auto Scheduled~~	~~8.1~~	~~validación de flujo y volumen~~	~~1 day~~		~~Natalia;Victor~~
~~Auto Scheduled~~	~~8.2~~	~~validación de desplazamiento y velocidad~~	~~1 day~~	~~136~~	~~Natalia;Victor~~
~~Auto Scheduled~~	~~8.3~~	~~validación y calibración de sensores~~	~~1 day~~	~~137~~	~~Natalia;Victor~~
~~Manually Scheduled~~	~~8.4~~		~~1 day?~~		

Tabla 1: Tabla del Plan WBS del proyecto, Gantt

Anexo 2 - Tabla de Riesgos asociados al proyecto.

Identificación y Evaluación Cualitativa de Riesgos

Nombre del Proyecto	Patrón de flujo y volúmen espiratorio para la calibración de instrumentos de valoración de la función pulmonar

Probabilidad	Valor Numérico	Impacto	Valor Numérico
Muy Improbable	0,10	Muy Bajo	0,05
Relativamente Probable	0,30	Bajo	0,10
Probable	0,50	Moderado	0,20
Muy Probable	0,70	Alto	0,40
Casi Certeza	0,90	Muy Alto	0,80
None	0	None	0

Tipo de Riesgo	Probabilidad x Impacto
Bajo	< 0.05
Moderado	0.06> & <0.18
Alto	>0.18

Casi Certeza	0,90	0,045	0,09	0,18	0,36	0,72
Muy Probable	0,70	0,035	0,07	0,14	0,28	0,56
Probable	0,50	0,025	0,05	0,1	0,2	0,4
Relativamente	0,30	0,015	0,03	0,06	0,12	0,24
Muy Improbable	0,10	0,005	0,01	0,02	0,04	0,08
		0,05	0,10	0,20	0,40	0,80
		Muy Bajo	Bajo	Moderado	Alto	Muy Alto

Código del Riesgo	Descripción del Riesgo	Causa	Disparador del Riesgo	Afectaciones al proyecto		Probabilidad de Ocurrencia	Objetivo Afectado	Estimación de Impacto		Tipo de Riesgo		Estrategia a seguir
R001	Disponibilidad de los recursos Humanos	Sobrecarga de trabajo del personal. Asignación del mismo personal a varios proyectos.	Nuevas funciones del personal.	Los tiempos del proyecto se veran afectados, y puede alargarse los tiempos de entrega	0,30	Relativamente Probable	Alcance		0	Moderado	0,06	Estudio de posibles candidatos para reemplazar aquellos recursos que pudieran dejar el proyecto
							Tiempo	Moderado	0,20			
							Costo		0			
							Calidad		0			
R002	Imposibilidad de cumplir con los requerimientos	No se dispone de los componentes para desarrollar el módulo	Demora en la compra. Falta de horarios para usarlos	Pueden alargarse los tiempos de finalizacion, al no poder utilizar el hardware necesario	0,70	Muy Probable	Alcance	Moderado	0,20	Alto	0,28	
							Tiempo	Moderado	0,20			
							Costo		0			
							Calidad		0			

R003	Perdida de interes en el proyecto	Personal de UNSAM ya no tiene interés en continuar con el proyecto	Falta de recursos económicos, salida al mercado de un equipo mas rentable	Suspención temporaria o definitiva (parcialmente o totalmente) del proyecto.	0,10	Muy Improbable	Alcance	Moderado	0,20	Moderado	0,06	Recorte de funcionalidades, que permita terminar un prototipo para la aprobacion del proyecto final de carrera
							Tiempo	Alto	0,40			
							Costo		0			
							Calidad		0			
R004	Renuncia de personal Clave	Motivos personales. Problemas por no conformidad con las autoridades	Falta de motivaciones. Falta de información	El proyecto puede verse suspendido por tiempo indeterminado o discontinuado	0,30	Probable	Alcance	Alto	0,40	Alto	0,24	Continuacion del proyecto de forma autonoma, con recorte de funcionalidades
							Tiempo	Alto	0,40			
							Costo		0			
							Calidad		0			
R005	Variación de los costos del mercado	Inflación de precios. Cierre de importaciones	Nuevas políticas gubernamentales	Aumento de los costos	0,30	Relativamente Probable	Alcance		0	Moderado	0,075	Busqueda de componentes alternativos que permitan el reemplazo de los de costos elevados, de no encontrarse, pueden suceder dos escenarios, que se pida un aumento de presupuesto y se conceda tal aumento, o que el proyecto deje de ser solventable y entonces se decida recortar la funcionalidad
							Tiempo		0			
							Costo	Moderado	0,20			
							Calidad	Muy Bajo	0,05			
R006	Cambios en los requisitos del proyecto.	Existian mas requerimientos ocultos que no fueron notificados	Necesidad de cumplir cierto requerimiento	Reestructuracion del proyecto, alargue de tiempos de entrega	0,10	Muy Improbable	Alcance	Muy Bajo	0,05	Bajo	0,025	Determinar exaustivamenete lo que el cliente quiere, lo que el equipo necesita realizar y las necesidades fundamentales a cumplir
							Tiempo	Moderado	0,20			
							Costo		0			
							Calidad		0			
R007	Los clientes no entienden la usabilidad del producto	Mal diseño de la GUI del proyecto	Entrega del equipo a un potencial cliente	Rediseño de la Interfaz Grafica	0,10	Muy Improbable	Alcance		0	Bajo	0,025	Realizar un estudio previo a la etapa de diseño que permita definir una GUI amigable al usuario
							Tiempo	Moderado	0,20			
							Costo	Bajo	0,05			
							Calidad		0			
R008	Subestimacion de los tiempos de desarrollo	No se conocia a fondo el alcance de cada requerimiento	Estudio para implenetacion	Alargue de tiempos	0,30	Probable	Alcance		0	Alto	0,18	Pedir asesoramiento, o utilizar metodos para determinar los tiempos mas adecuadamente
							Tiempo	Moderado	0,20			
							Costo		0			
							Calidad	Alto	0,40			

R009	Mala Planificacion del proyecto	No se tomo el tiempo necesario para la realizacion de una planificacion a fondo	Implementacion	Alargue de tiempo, necesidad de hardware adicional, o recursos adicionales	0,30	Relativamente Probable	Alcance	Moderado	0,20	Alto	0,18 Realizar un cronograma de actividades, bien especifico y fragmentado
							Tiempo	Alto	0,40		
							Costo		0		
							Calidad		0		

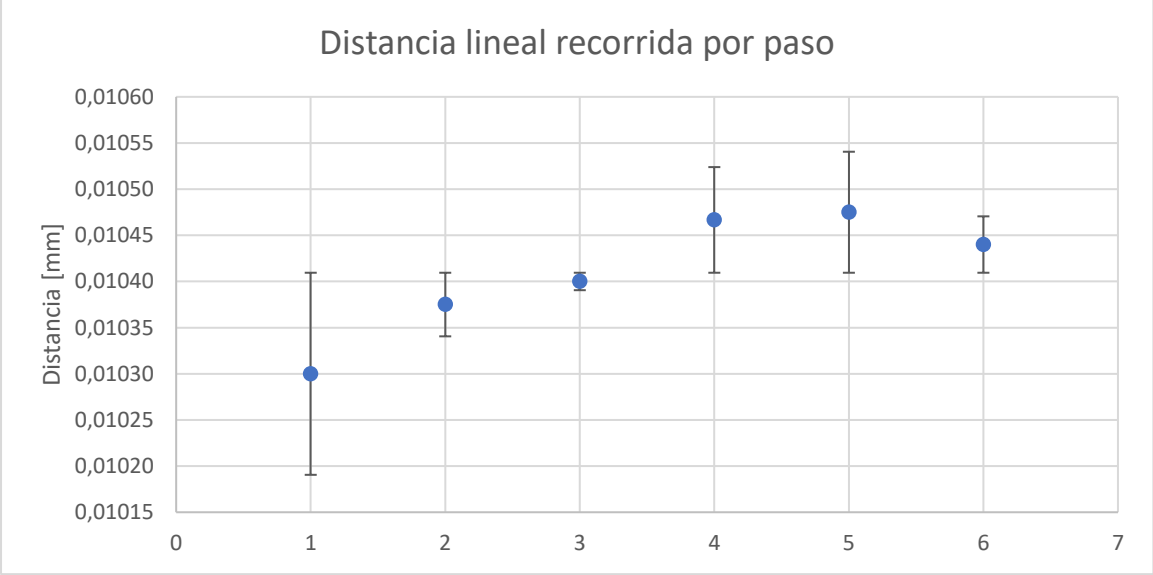
Tabla 2: Tabla de Riesgos asociados al proyecto.

Anexo 3 – Estudio Estadístico

Se realizo el cálculo de la distancia lineal recorrida, por paso efectuado del motor. Con la siguiente tabla se pudo verificar la repetibilidad del movimiento y confirmar el paso lineal calculado, al momento de construir el engranaje y polea adosado al motor.
Se inicio el movimiento siempre desde la misma posición, y a la misma velocidad (5000Hz)

PASOS CONFIGURADOS	DISTANCIA TOTAL RECORRIDA [MM]	DISTANCIA RECORRIDA POR PASO [MM]	ERROR [MM]
5000	51,5	0,01030	0,0001094
8000	83	0,01038	0,0000344
10000	104	0,01040	0,0000094
15000	157	0,01047	0,0000572
20000	209,5	0,01048	0,0000656
25000	261	0,01044	0,0000306

Promedio de la distancia lineal por paso: (0,01041 ± 0,0000511) mm
Error porcentual cometido por paso, e = 0,5%



Anexo 4 – Diagrama de Flujo

Firmware - Diagrama de flujo de la función principal del programa.

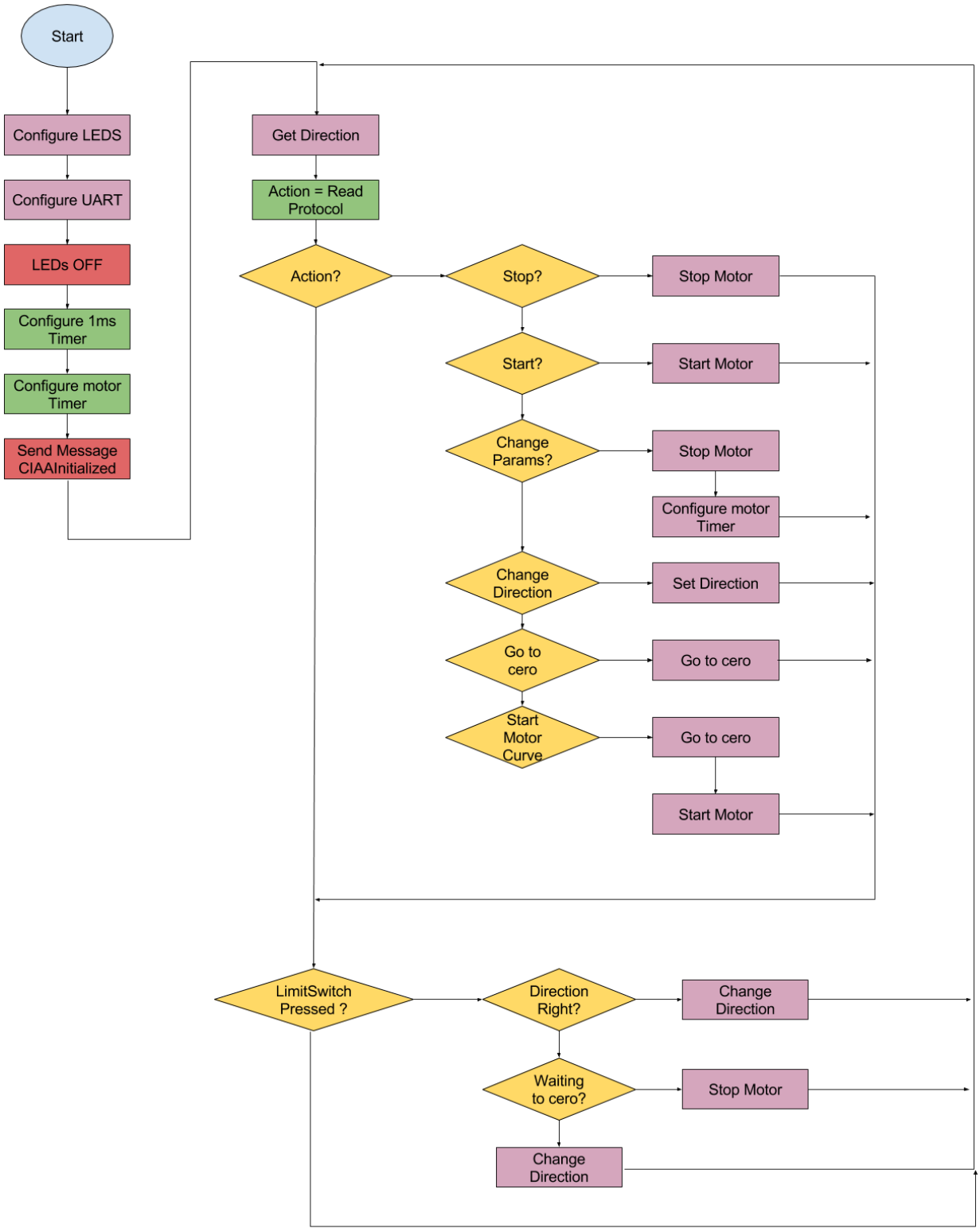


Figura 23: Diagrama de flujo de la función Main del Firmware implementado.

Anexo 5 – Diagrama de clases