
CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

# Controlador CAN de servomotores

**Autor:**
**Ing. Alejandro Virgillo**

Director:
Esp. Ing. Gabriel Gavinowich (FIUBA)

Jurados:
Nombre del jurado 1 (pertenencia)
Nombre del jurado 2 (pertenencia)
Nombre del jurado 3 (pertenencia)

*Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires, entre Octubre de 2021 y Abril de 2023.*

# *Resumen*

En el presente trabajo se aborda el proceso de planificación, diseño y fabricación de un dispositivo que actúe de interfaz para que un operario de planta pueda monitorear y programar motores paso a paso que usan una plaqueta de control a medida conectados a través de un bus CAN (Controller Area Network). El desarrollo se plantea para uso industrial y se ensaya en la planta de la empresa argentina Cambre ICyFSA.

El documento detalla las herramientas de planificación y gestión empleadas, la selección de componentes electrónicos, la descripción de los protocolos de comuncación empleados, el desarrollo del software embebido, el diseño y fabricación del hardware y su gabinete y las consideraciones de manufactura que se tomaron. Todo esto acompañado con su documentación apropiada y con las pruebas de validación realizadas.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción general

En esta sección se da una explicación general de los temas sobre los cuales se basa este trabajo. Se dará una introducción al protocolo CAN - *Controller Area Network* y a qué es un servomotor. También se hablará sobre el proyecto SN-17 y la motivación y alcance del trabajo, así como el estado del arte de esta tecnología.

## 1.1. Controller Area Network - CAN

*Controller Area Network* o CAN es un protocolo de comunicación desarrollado por Bosch [**wikipedia_CAN**] orientado originalmente a la industria automotriz, y hoy en día es empleado en muchas otras aplicaciones. En el año 1991 Bosch publicó la especificación CAN 2.0 donde se diferencian aspectos de identificación de dispositivos conectados en la red y, en el año 1993, se estandarizó el protocolo bajo la norma internacional ISO 11898 [**web_ISO_CAN**].

CAN se caracteriza por su robustez y bajos requerimientos de cableado. En su concepción, se pensó para proveer comunicaciones deterministicas en sistemas complejos, teniendo las siguientes cualidades [**Understanding_CAN**]:

- Prioridad de mensajes y latencia máxima asegurada.
- Comunicaciones a varios dispositivos al mismo tiempo.
- Bus multi-maestro.
- Detección de errores en nodos y mensajes.
- Protocolo asincrónico - sin línea de clock.

Para realizar la transferencia de información CAN emplea líneas de transmisión, comunmente llamados CAN High y CAN Low, y estos son los únicos enlaces requeridos. Al comienzo de cada mensaje, se emplea un identificador, que indica la prioridad del mensaje, así como a quien está dirigido. Usando esta información cada nodo de la red determina que hacer. En la Figura 1.1, se puede ver un esquema básico de una red CAN.

## 1.2. Servomotores

Un servomotor[**Industrial_Automation_Hands_On**] es un tipo de motor eléctrico que tiene la capacidad de controlar la posición angular del eje, así como la

[1] https://www.seeedstudio.com/blog/2019/11/27/introduction-to-can-bus-and-how-to-use-it-with-arduino/

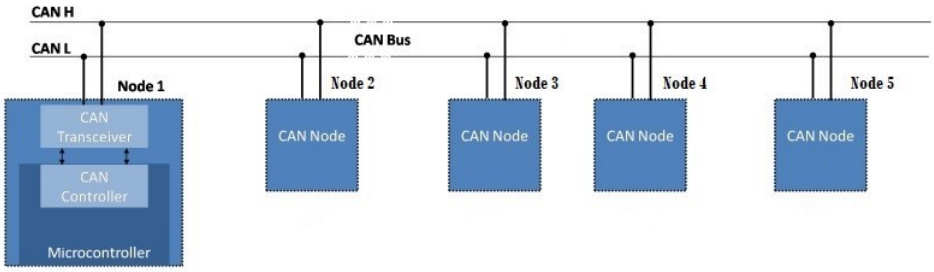


FIGURA 1.1. Esquema de red CAN[1]

velocidad de rotación y el torque. En general, el tipo de motor eléctrico que se emplee no determina si es o no un servomotor, sino que cuente las cualidades de control mencionadas. Para lograr esto, suele emplearse un sistema de lazo cerrado que se retroalimenta con la información proporcionada por un sensor que mide la posición angular del eje, llamado encoder. Esto es procesado por un controlador, quien se encarga de proporcionar la corriente adecuada a las bobinas del motor para que este se mueva de la forma indicada. En la Figura 1.2 se puede ver un servomotor con sus distintas partes.


FIGURA 1.2. Esquema un servomotor[2]

## 1.3. Proyecto SN-17 - Servo Nema 17

El proyecto SN-17 es un sistema desarrollado por la organización A3 Engineering que permite convertir motores eléctricos del tipo paso a paso o *steppers* en servomotores. Este tipo de motores tienen, similar a un servomotor, la capacidad de controlar su posición y velocidad, pero con un lazo de control abierto. En caso de que haya perturbaciones al funcionamiento normal, el motor pierde el control de la posición y debe realizar una rutina de *homing* para recuperarlo. Esto puede traer problemas en ciertas aplicaciones de precisión. Además, son incapaces de entregar torque constante a la salida, por el mismo motivo. El sistema SN-17

[2] https://www.logicbus.com.mx/blog/que-es-un-servo-motor/

agrega un encoder y un controlador al motor, generando un lazo cerrado y logrando las funcionalidades de un servomotor. En la Figura 1.3 se puede observar una placa de control SN-17. Esta se coloca en la parte trasera de un motor paso a paso y se lo conecta directamente.

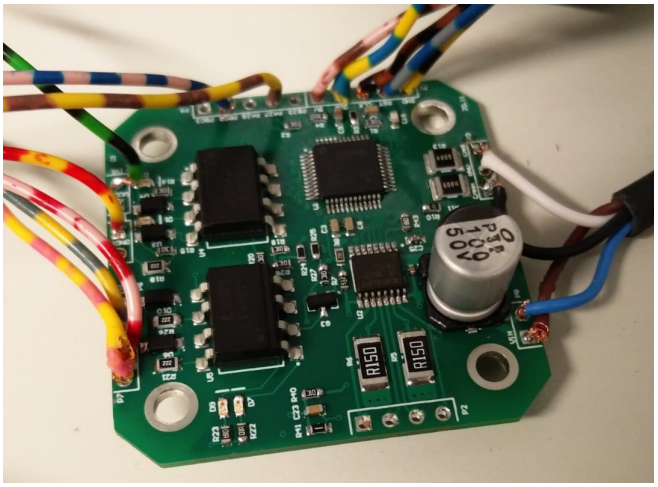

FIGURA 1.3. Plaqueta SN-17

Además de generar las funcionalidades mencionadas, el sistema SN-17 tiene señales discretas industriales que le permiten comunicarse a través de ese medio con controladores industriales (*Programmable Logic Controllers* o PLCs) o pueden controlar un pequeño proceso por su cuenta.

Actualmente, se está trabajando en implementar estos sistemas en la planta de la empresa Cambre ICyFSA, donde se están construyendo distintos dispositivos para aplicaciones industriales con estos motores. En la Figura 1.4 se puede observar un dibujo CAD de un actuador lineal con un sistema SN-17 implementado. Este funciona como un prensador en un proceso industrial.

## 1.4. Motivación

En la actualidad, el ámbito de actuadores industriales está principalmente dominado por aquellos de carácter neumático. Sin embargo, en los últimos años, el uso de actuadores eléctricos ha empezado a ser más significativo debido a las mayores aptitudes de control que estos poseen. Estos suelen emplear un servomotor en su interior, y junto con un mecanismo generan el movimiento deseado con una precisión superior. Aún así, este tipo de actuadores tienen un problema en su elevado costo, que termina limitando su uso. En este contexto, la organización A3 Engineering desarrolló el sistema SN-17 en un intento de disminuir los costos de los servomotores de aplicaciones pequeñas y, en un futuro, de los actuadores eléctricos.

El sistema SN-17 cuenta con un problema a la hora de su utilización: la programación de los motores es compleja y se requiere de altos conocimientos técnicos para realizar modificaciones. Esto se debe a que, para cambiar las instrucciones del programa, es necesario modificar el firmware del sistema. Cuando se implementó inicialmente en planta, el sistema funcionó de forma exitosa, pero poco flexible.

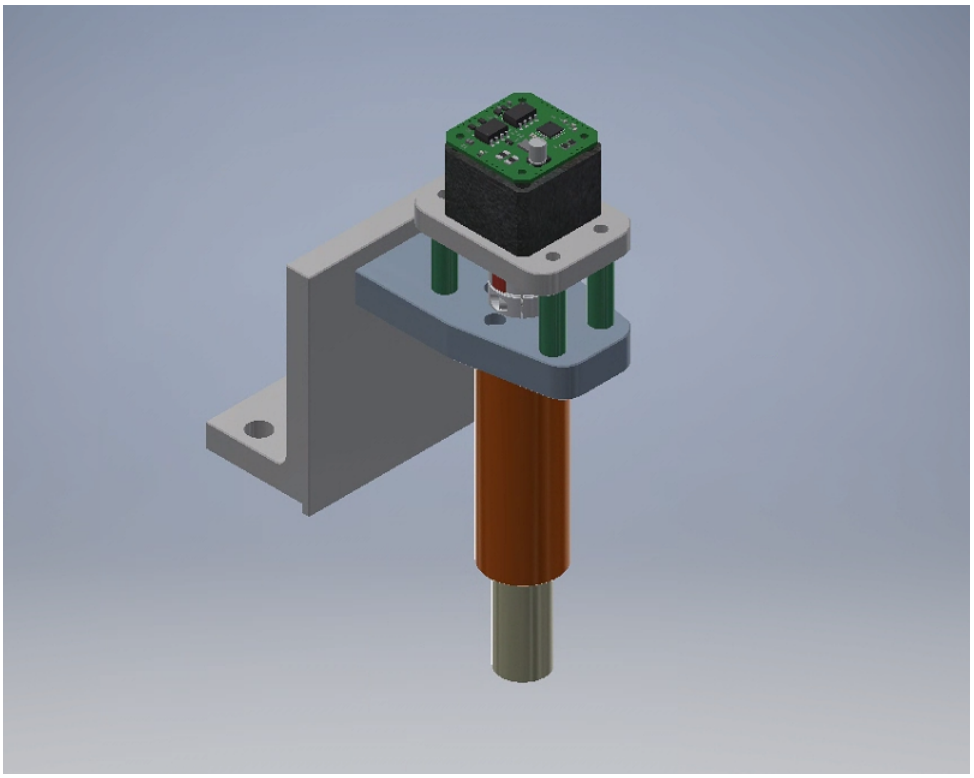

FIGURA 1.4. Actuador lineal con SN-17

Con esta situación, se planteó la necesidad de que un técnico de planta sin mucho entrenamiento pudiera cambiar la programación y parámetros de funcionamiento de los motores. También, se consideró que sería conveniente, para maquinarias con varios motores conectados, poder hacer que todos estén dentro de una misma red, permitiendo así controles sobre conjuntos de motores.

## 1.5. Estado del arte

Actualmente existe una gran oferta de servomotores y motores paso a paso. En general, lo que en el ámbito industrial se refiere a servomotores**[Industrial_Automation_Hands_On]** son motores del tipo DC *brushless* o AC de inducción (monofásicos y trifásicos). Estos tienen un valor significativamente mayor que los *steppers* y suelen requerir de *drivers* externos y, en caso de motores AC, variadores de frecuencia, convirtiendolos en soluciones de alto costo. En los casos que se requiere alta potencia o muy alto rendimiento, estos son la mejor opción. Suelen requerir de técnicos muy capacitados para su programación y suelen ofrecer interfaces de usuario avanzadas.

Por otro lado, en lo referido a motores *steppers*, la oferta también es variada. Estos pueden adquirirse a precios muy accesibles y en distintos tamaños con lazo abierto. Requieren el uso de un *driver* externo para su operación, y su programación suele hacerse con programas de terceros. Dependiendo la aplicación, pueden requerir de conocimientos técnicos elevados. También existen variantes que agregan un encoder de posición angular, comúnmente llamados *closedloop steppers*. Estos solventan uno de los mayores problemas de estos motores, relacionado con la pérdida de posición en caso de una perturbación, pero no permiten entregar

torque constante. También requieren de *drivers* externos y, según el fabricante, las interfaces de programación pueden ser complejas.

Finalmente, existen placas de control que, además de funcionar como un *closed loop stepper*, tienen un *driver* incorporado que permite alimentar las bobinas de una forma diferente, logrando entregar torques constantes. El problema principal de estas soluciones es que suelen tener características de *hobbista* y no son aplicables en ámbitos industriales y, además, suelen requerir mucho conocimiento técnico para su programación. En la Figura 1.5 se puede ver un ejemplo de una de estas placas llamado *Mechaduino* [**web_mechaduino**].

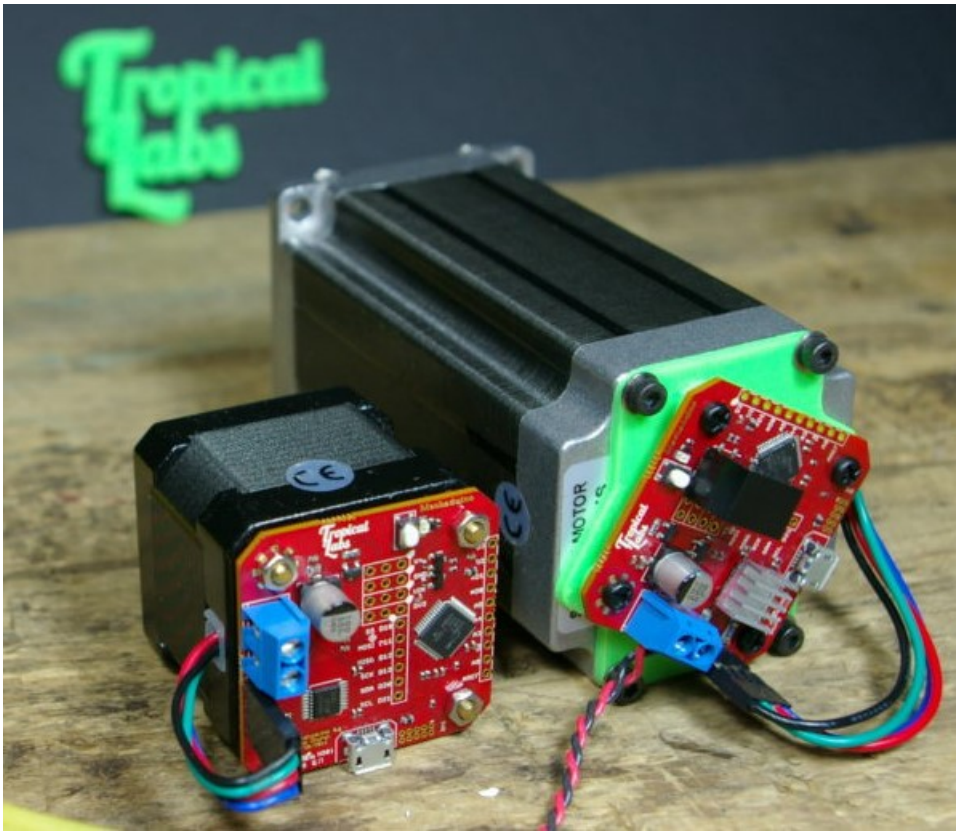


FIGURA 1.5. Proyecto Mechaduino[3]

En la Tabla 1.1 se resume la información explicada en esta sección.

TABLA 1.1. Resumen de características de servomotores

| Motor | Control | Driver | Programación | Industrial | Precio (USD) |
|---|---|---|---|---|---|
| Servomotor | Lazo cerrado | Externo | Complejo | Sí | >$ 1.500 |
| *Stepper open loop* | Lazo abierto | Externo | Intermedio | Sí | $ 300 |
| *Stepper closed loop* | L. Cerrado s/torque | Externo | Intermedio | Sí | $ 400 |
| *Mechaduino* | Lazo cerrado | Interno | Complejo | No | $ 200 |
| SN-17 + interfaz | Lazo cerrado | Interno | Simple | Sí | $ 300 |

[3] https://tropical-labs.com/mechaduino/

## 1.6. Objetivos y alcance

El objetivo de este trabajo es proporcionar una interfaz de usuario para el sistema SN-17 que permita que un operario, con poco entrenamiento, pueda modificar los programas de los disntintos motores conectados a través de una red CAN con el sistema desarrollado. A su vez, debe poder emplearse para monitorear el estado los motores conectados cuando están operativos. Por lo tanto, el proyecto incluye:

- Una interfaz de usuario que permita configurar y supervisar los servomotores conectados.
- La construcción e implementación de la estructura de mensajes CAN.
- La configuración de la red CAN.
- El desarrollo y fabricación de una plaqueta con el sistema embebido.
- El desarrollo del firmware para este sistema y la adaptación del firmware del sistema SN-17.

# Capítulo 2

# Introducción específica

Este capítulo aborda una explicación más detallada de algunos de los puntos explicados en el capítulo anterior, como el sistema SN-17 y el protocolo CAN. También, se presentan algunas características de las entradas y salidas de los controladores industriales PLCs y se explican los distintos componentes seleccionados para el trabajo y su funcionamiento.

## 2.1. Especificaciones SN-17

El sistema SN-17 se desarrolló para controlar motores *stepper* pequeños del tipo NEMA 17. Este es un estándar ampliamente utilizado para este tipo de motores, que indica las dimensiones que deben tener. Mediante una adaptación mecánica, también se puede emplear la plaqueta para controlar motores de menor o mayor tamaño, siempre y cuando sean del tipo *stepper* y no requieran corrientes mayores a 2 A.

El sistema se alimenta comúnmente con 24 V, con lo que se alimentan las bobinas del motor y el regulador de tensión de 5 V con el cuál se alimenta la electrónica de la plaqueta. Esto está compuesto por un encoder magnético [**web_AS5047D**], que permite sensar la posición angular del eje del motor; un *driver* del tipo doble puente H[**web_A4954**], que controla la corriente que circula por las bobinas del motor; un *transciever* CAN[**web_transciever_CAN**], que convierte y recibe las señales eléctricas del bus CAN; y un microcontrolador ATMSAMC21[**web_ATSAMC21G18A**], encargado de controlar el proceso. En lo que respecta al microcontrolador, este cuenta con una arquitectura M0+ y tiene, entre sus periféricos, un controlador CAN incorporado.

La plaqueta cuenta con un circuito de entradas y salidas del tipo PNP que se encuentran eléctricamente aisladas del microcontrolador usando optoacopladores[**web_optoacoplador**
Estas pueden alimentarse a tensiones distintas que el resto del sistema y se usan para interactuar con un controlador industrial del tipo PLC.

El sistema tiene un ciclo de control de lazo cerrado del tipo PID[**paper_PID_steppers**] al cual se le indica un valor deseado (*set point*) de posición, velocidad o torque. Con esta información, se la compara con los valores obtenidos del encoder y se determina cómo deben ser alimentadas las bobinas del motor para alcanzar el *set point*. Para el funcionamiento de esto, el encoder debe ser calibrado junto con el motor, mediante una rutina en la que se arma una tabla de referencia que luego es usada en operación.

Sobre el control mencionado, corre una aplicación en la que se cargan programas que el motor debe realizar. Estos programas estan compuestos de instrucciones configurables, que permiten establecer los modos de control deseados (posición, velocidad o torque), los *set points* y errores admisibles para estos (*thresholds*), una limitación del torque para esa instrucción, los tiempos que deben cumplirse para considerar correcta la instrucción (*hold time*) y los tiempos para que se considere que el sistema entró en estado de error (*timeout*). Otras instrucciones tienen que ver con flujo de programa, interacción con las entradas y salidas y comunicaciones a través del puerto CAN.

Existen también configuraciones que se le pueden aplicar al motor, estas son:

- Constantes PID del lazo de control
- Tipo de rutina de *homing* del motor
- Funcionamiento de entradas y salidas
- Guardado de posiciones de eje en memoria

El sistema también puede recibir comandos de forma manual:

- Calibración de encoder y motor
- Activación o apagado de motor
- Cerado de motor
- Activación de salidas

En la Tabla 2.1 se resume la información sobre las distintas funciones que la aplicación del sistema SN-17 puede realizar.

TABLA 2.1. Funciones de SN-17

| **Señal** | **Tipo** | **Descripción** |
|---|---|---|
| Tipo de instrucción | Instrucción | Define la instrucción |
| Límite de torque | Instrucción | Torque máximo de instrucción |
| Modo de control | Instrucción | Posición, velocidad, torque |
| *Set point* | Instrucción | Valor de lazo de control |
| *Threshold* | Instrucción | Valor de error admisible |
| *Hold time* | Instrucción | Tiempo de cumplimiento |
| *Timeout* | Instrucción | Tiempo de no cumplimiento |
| Guardar posición | Configuración | Guarda posición actual |
| Constantes PID | Configuración | Constantes lazo de control |
| Entradas y salidas | Configuración | Funcionamiento de las IO |
| *Homing* | Configuración | Establece rutina de cerado |
| Calibración | Comando | Inicia la rutina de calibración |
| Activar motor | Comando | Enciende/apaga el motor |
| *Go Home* | Comando | Inicia la rutina de cerado |
| Activar salidas | Comando | Enciende/apaga salidas |

## 2.2. Características del protocolo CAN

Como se trató en el capítulo 1, el protocolo CAN está estandarizado en la norma internacional ISO 11898[**web_ISO_CAN**]. El estándar abarca solamente las capas física y de enlace de datos, es decir, las 2 capas más bajas en un modelo OSI - *Open System Interconection*, como el que se puede ver en la Figura 2.1. Para las capas superiores, existen otros estándares, como CANOpen[1], del cual se tomaron ideas para este trabajo, pero no se implementa en si.

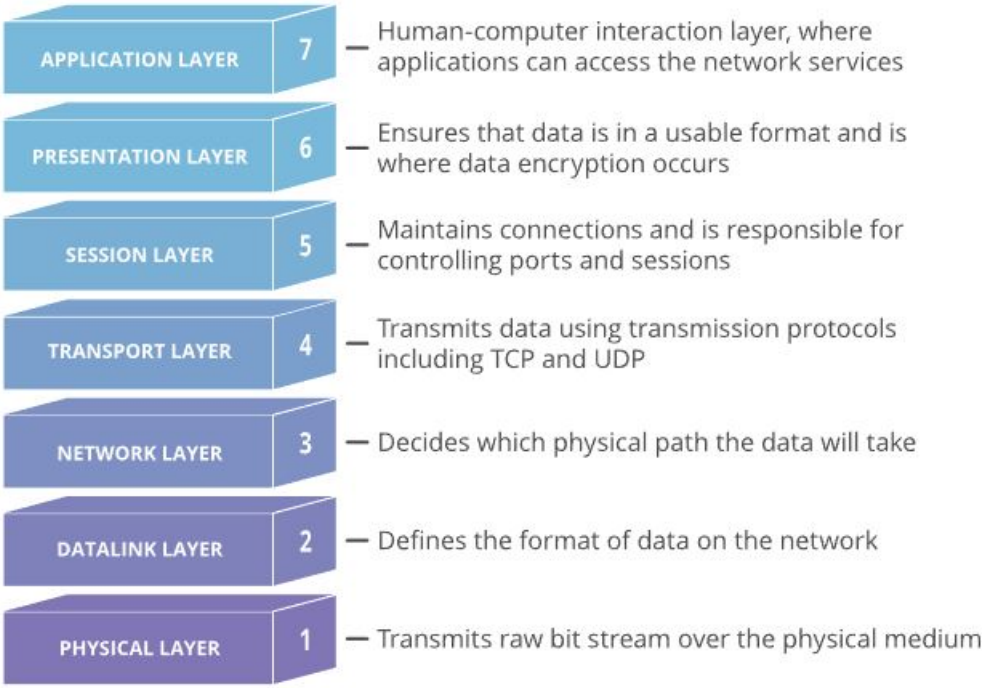


FIGURA 2.1. Modelo de capas OSI[2]

Otras características que tiene la red están relacionadas con el armado del circuito de los nodos, el cual suele estar especificado en las hojas de datos de los *transcievers*, así como la necesidad de colocar una resistencia de terminación en los extremos de la red, como puede visualizarse en la Figura **??**. El valor de esta resistencia depende de muchas variables, como el largo de la red y la velocidad de transmisión y suelen usarse guías para seleccionarlas. Los medios físicos de transmisión, así como los conectores no estás especificados por la norma, aunque existen recomendaciones [**Embedded_Networking_CAN**].

Las señales usadas son del tipo diferencial y se clasifican en dominantes y recesivas. Una señal dominante en el bus tiene prioridad por sobre una señal recesiva, lo que permite que varios dispositivos estén conectados y puedan hablar al mismo tiempo y que se puedan detectar colisiones[**Embedded_Networking_CAN**]. En un estado recesivo, las líneas CAN-H y CAN-L están al mismo nivel de tensión, generalmente 2.5 V en sistemas embebidos, aunque el estándar indica otros

[2]https://www.can-cia.org/canopen/

[2]https://www.cloudflare.com/es-es/learning/ddos/glossary/open-systems-interconnection-model-osi/

[3]https://www.seeedstudio.com/blog/2019/11/27/introduction-to-can-bus-and-how-to-use-it-with-arduino/

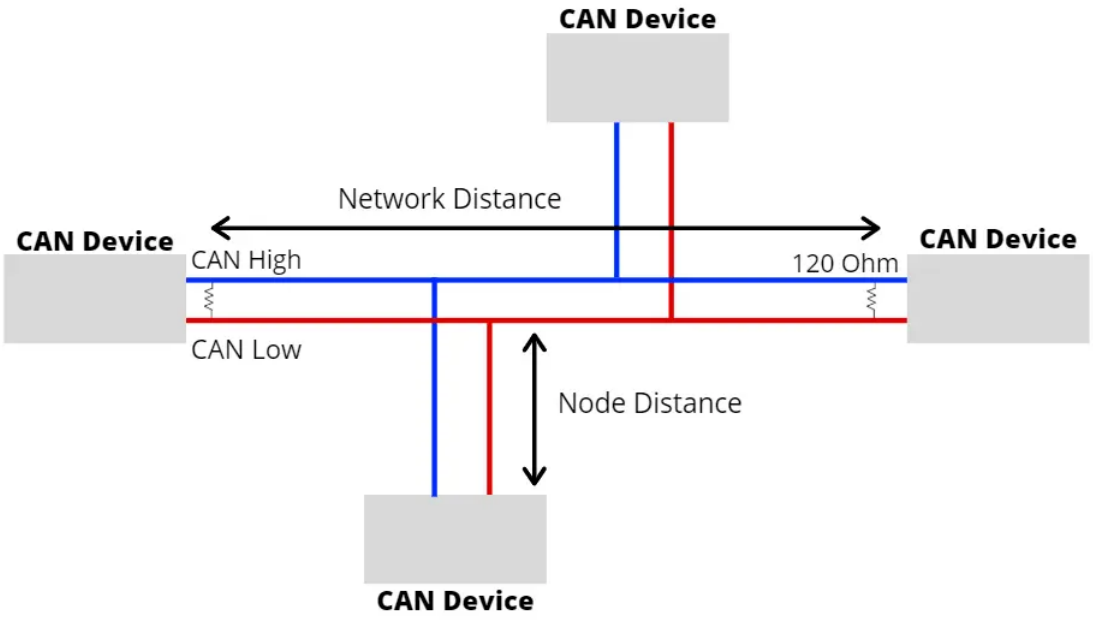


FIGURA 2.2. Esquema de red CAN con resistores de terminación[3]

valores adicionales de tensión admitidos. En un estado dominante, CAN-L baja, al menos 1 V, mientras que CAN-H sube, al menos, 1 V generando una diferencia entre ambos.

El bus CAN se caracteriza por ser asincrónico, es decir, no tener una línea de reloj. Para sincronización, se usa un método conocido como *Non Return To Zero* o NRZ. Este consiste en determinar una cantidad máxima de bits que pueden ser transmitidos en el bus sin una transición entre estados recesivos y dominantes. En cada transición, los nodos se resincronizan, evitando largas cadenas de mensajes no coordinadas. Entonces, para lograr esto, CAN emplea el *Bit-Stuffing* que fuerza el cambio de estado en la línea cada 5 bits iguales.

El estándar también subdivide el tiempo de un bit en distintos segmentos: Sincronización, propagación, fase 1 y fase 2. Estos se describen en una unidad de tiempo menor, referida como *time quanta*. Para un buen funcionamiento de la red, todos estos parámetros deben ser correctamente configurados en los distintos nodos CAN.

En lo que respecta a las tramas CAN, existen de 2 tipos: estándar y extendida. En este trabajo el enfoque está en las tramas estándar que se caracterizan por tener un identificador de 11 bits, donde se encodea información del receptor del mensaje. En la Figura 2.3 se puede ver un esquema detallando la trama CAN estándar. La trama se separa en:

- Comienzo de trama - **SOF**
- Arbitraje
- Control
- Data a envíar
- Verificación

- Reconocimiento - *Acknowledge*
- Fin de trama - **EOF**

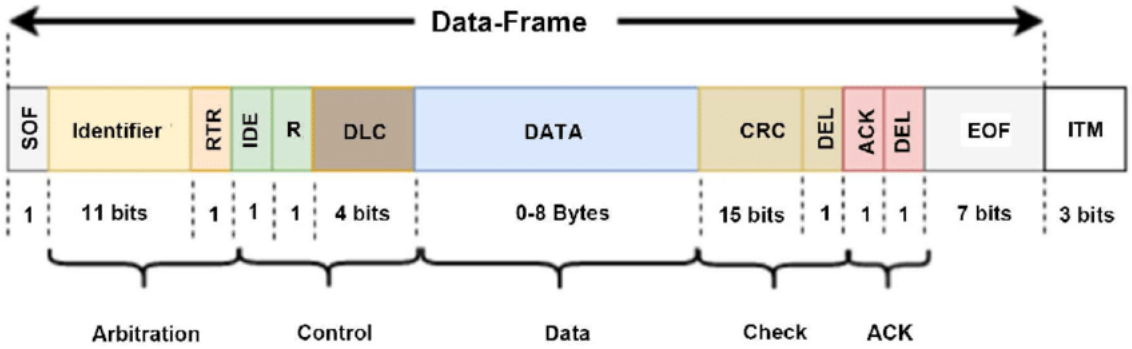


FIGURA 2.3. Trama CAN estándar[4]

La sección de arbitrake está compuesto por el identificador y el bit RTR (*Remote Transmission Request*), que se utiliza para pedir a la red que se transmita cierto mensaje. El identificador es procesado por cada nodo de la red y, mediante el uso conjunto de máscaras y filtros, determinan si el mensaje corresponde al nodo o no. También, en caso de que 2 o más nodos quieran transmitir un mensaje al mismo tiempo, el que lo haga con el identificador más bajo será el que tendrá prioridad y tomará el control de la red. Esta funcionalidad se consigue gracias a la lógica AND del circuito del bus. Los nodos que no logran transmitir su mensaje, debido a la prioridad inferior, pueden identificar esto y retransmitir cuando el bus esté nuevamente desocupado.

El campo de control del frame tiene al bit IDE, que identifica el tipo de trama entre estándar y extendida y el DLC, donde se indica la cantidad de bytes que tendrá el mensaje. Esta puede ser entre 0 y 8 bytes de largo.

Luego de el envío de la data, se encuentran los campos de verificación y reconocimiento. La verificación está compuesta por el campo CRC, que permite comprobar que la información transmitida es igual a la información recibida, es decir, que no haya habido errores de transmisión. Luego de esto, viene el sector de reconocimiento, donde lo receptores reconocen que el mensaje se ha recibido correctamente enviando un bit dominante. La trama termina con una sección de fin de trama.

En la actualidad, se comercializan una gran cantidad de controladores CAN. La gran mayoría ofrece todas las funcionalidades descriptas, que son lo requerido por el estándar. Según la aplicación, se pueden elegir controladores que tengan distinto número de filtros y máscaras para identificación de mensajes, tamaño de buffer de mensajes, tanto de recepción como de transmisión, velocidad de operación, entre muchas otras. También, muchos microcontroladores incluyen un periférico de CAN, que suele ser una solución conveniente para implementar redes de este tipo. Una vez elegido el controlador, la hoja de datos indica cómo deben configurarse los distintos parámetros. Esto, como se explicó, debe hacerse de forma correcta para asegurar un buen funcionamiento de la red.

[4] https://www.researchgate.net/publication/328607559_Classification_Approach_for_Intrusion_Detection_in_Vehicle_Systems

## 2.3. Entradas y salidas de controladores industriales

Un PLC - *Programmable Logic Controller* es un tipo de controlador ampliamente utilizado para manejar procesos industriales. Se caracterizan por su robustez y su forma de programación, muy similar a la lógica de relé.

En general, los PLC requieren interactuar con gran número de sensores y actuadores presentes en un proceso, por lo que cuentan con amplios módulos de entradas y salidas digitales. Estos módulos suelen adaptar el nível de tensión de las señales que reciben y transmiten y suelen proteger los circuitos internos del PLC aislando eléctricamente las señales[**Introduction_Industrial_Automation**]. Normalmente, el estándar industrial es trabajar a 24 V de corriente contínua, aunque también es común usar 48 V DC o corrientes AC de línea y, en casos poco frecuentes, pueden aparecer otros valores. Los controladores industriales suelen estar preparados para trabajar con la mayoría de estas condiciones.

Uno de los circuitos de acondicionamiento de salidas de PLC se puede observar en la Figura 2.4. En este caso es del tipo NPN, que hace referencia al transistor empleado y el tipo de conexión externa que requiere para su línea común. Notar el uso de un opto acoplador para separar eléctricamente los circuitos y la cantidad de elementos de protección que se agregan para dar robustez. Existen también circuitos del tipo PNP, que cambian el transistor que emplean y el conexionado, y también hay con relés.

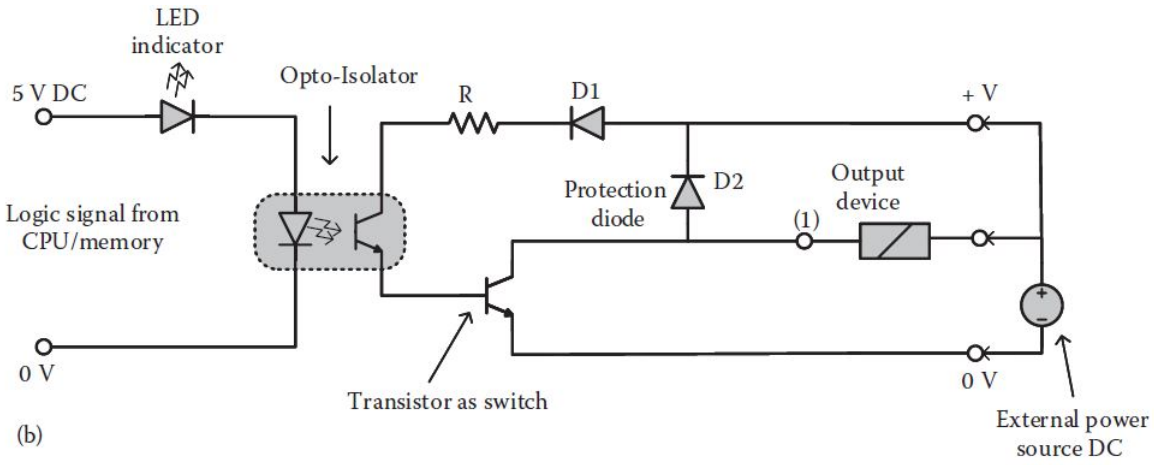


FIGURA 2.4. Circuito NPN[**Introduction_Industrial_Automation**]

Para especificar un módulo de entradas o de salidas, normalmente se deben especificar los rangos de voltaje y corriente de operación, el consumo máximo de corriente, los valores de tensión lógicos y la velocidad y frecuencia de conmutación.

En el mercado existen gran variedad de controladores industriales, cada uno con distintos módulos de entradas y salidas y distintos periféricos disponibles, según la aplicación.

## 2.4. Características de componentes electrónicos empleados

En esta sección se dará una explicación de los distintos componentes electrónicos usados en este trabajo. Se tratará sobre su funcionamiento y sobre los recursos disponibles para trabajar con ellos.

### 2.4.1. Pantallas LED y conversores I2C

Las pantallas LCD ofrecen una forma conveniente y económica para generar una interfaz de usuario. Para los proyectos de electrónica, uno de los modelos más populares es el panel de texto basado en el Hitachi HD44780[**Arduino_Cookbook**]. En la Figura **??** se muestra uno de estos displays, en particular uno de 4 líneas y 20 caracteres por línea. Estos dispositivos solo pueden representar caracteres, no pueden hacer dibujos ni gráficos, por lo tanto las interfaces que se obtienen son sencillas.

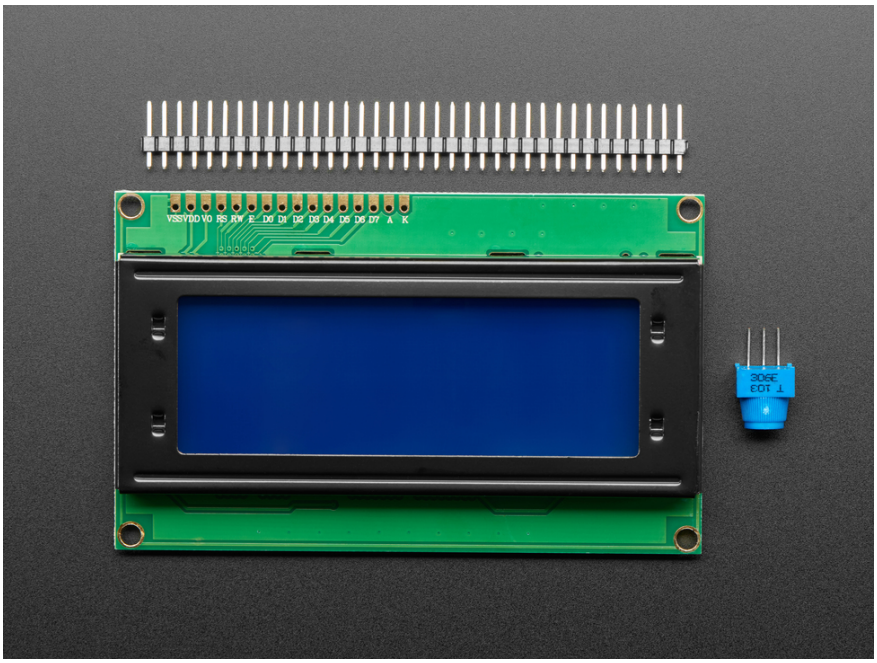

FIGURA 2.5. Pantalla LCD 20x4[5]

Existen modelos de LCD que incluyen una interfaz I2C (*Inter-Integrated Circuit*), un protocolo de comunicación simple, muy utilizado en sistemas embebidos. Esta interfaz facilita la interacción con el display, que normalmente requiere una serie de conexiones discretas para controlarlo. De esta manera, se pueden envíar comandos para modificar la configuración del dispositivo o mostrar texto, según se requiera.

La mayoría de microcontroladores que se emplean en la actualidad tienen un periférico de I2C e incluyen drivers de este, lo que también promueve la implementación de las interfaces. El protocolo es del tipo serial y sincrónico, requiriendo 2 líneas de información, SDA y SCL, por donde se transmite la data y el reloj,

[5] https://www.adafruit.com/product/198

respectivamente. En un bus I2C, los distintos dispositivos conectados tienen un código identificador que se emplea para controlar las comunicaciones.

A la hora de implementar una solución con este tipo de pantallas, puede tomarse de referencia alguna de las librerías de código abierto que se encuentran disponibles en la web. Esto puede ayudar a reducir significativamente los tiempos de desarrollo y es el camino que se tomó en este trabajo.

### 2.4.2. Matriz de botones

Las matrices de botones consisten en un conjunto de contactos normalmente abiertos que conectan una fila con una columna cuando se presionan. La metodología de uso consiste en conectar las filas a pines de entrada con resistores de pull-up en el microcontrolador y las columnas a pines de salida. La secuencia que se realiza es poner el nivel de tensión de una columna por vez en bajo (mientras las otras se mantienen en un nivel alto). En ese momento, se leen las entradas de las filas, si alguna está en un nivel bajo es indicación de que el botón de esa fila y columna fue presionado (ya que normalmente estarían en un nivel alto por las resistencias de pull-up)[**Arduino_Cookbook**]. En la Figura 2.6 se puede visualizar un esquema de conexionado de una matriz de botones 4x3 a un microcontrolador. Notar también que se muestran los conexionados internos de la matriz, de las filas y las columnas.

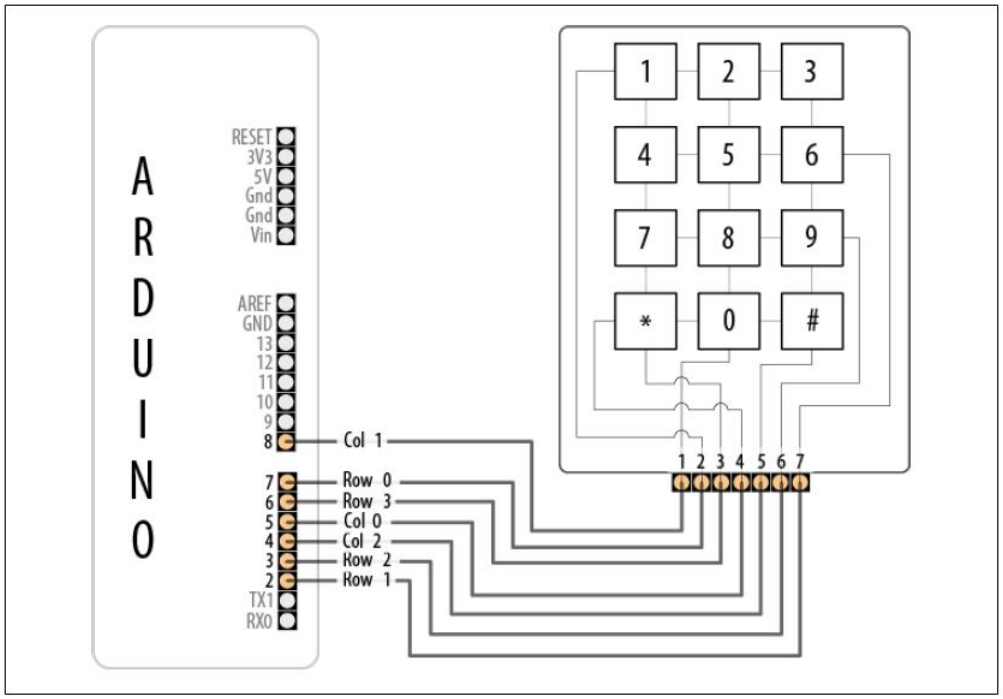


FIGURA 2.6. Conexión de matriz de botones 4x3[**Arduino_Cookbook**]

Como en el caso de las pantallas LCD, existen muchas librerías de código abierto que implementan soluciones de matrices de botones. Se considera conveniente consultarlas si se desea utilizar uno de estos dispositivos, y es lo que se hizo en este trabajo.

### 2.4.3. Conversores UART-USB

El *Universal Asynchronous Receive Transmit* o UART, es un protocolo serial de comunicación ampliamente utilizado en sistemas embebidos. Se caracteriza, principalmente, por su sencillez. Como su nombre indica, es un protocolo asincrónico (sin línea de clock), que requiere solamente 1 línea para envíar data, o 2 para envíar y recibir.

La gran mayoría de los microcontroladores que se utilizan actualmente cuentan con un periférico de UART y se suelen incluír drivers de este. Esto reduce los tiempos a la hora de desarrollar una implementación.

Otro protocolo ampliamente usado es el USB, y suele ser el preferido para interactuar con una PC. Existen en el mercado una gran cantidad de conversores que transforman un mensaje UART a uno USB, permitiendo la interacción entre una PC y un sistema embebido. En general, estos conversores cuentan con drivers para los distintos sistemas operativos y son reconocidos por estos, por lo cual solo es necesario conectarlos a través del puerto USB. Desde la PC, cualquier programa de monitoreo de puertos seriales puede usarse para envíar y recibir mensajes con el sistema embebido.

Este tipo de implementaciones suele ser muy útil para realizar operaciones de búsqueda de errores o para armar interfaces gráficas con una PC, facilitando la operatoria de un usuario final. En este trabajo se utiliza uno de estos módulos para esta finalidad.

# Capítulo 3

# Diseño e implementación

## 3.1. Arquitectura del sistema embebido

## 3.2. Selección de componentes

## 3.3. Desarrollo de software

INDICAR QUE EL SOFTWARE SE DESARROLLO PRIMERO EN PLACA DE DESARROLLO Y DESPUES SE IMPLEMENTO

### 3.3.1. Driver CAN

### 3.3.2. Interfaz HMI

### 3.3.3. Interfaz UART-USB

## 3.4. Modificaciones firmware SN-17

## 3.5. Desarrollo de Hardware

Para el desarrollo del hardware se utilizó el software de diseño Altium[1] y se eligió como fabricante a PCBWING[2] que es una empresa con la que trabaja comúnmente Cambre ICyFSA. Se tomó, como punto de partida, las capacidades técnicas de este fabricante.

Se decidió hacer una placa de 4 capas, con dimensiones menores a 100 x 100 mm. Esto se debe a que el fabricante PCBWING ofrece precios más económicos para estas condiciones de fabricación y se consideró que no imponen restricciones importantes el diseño requerido.

## 3.6. Desarrollo de gabinete

Para el desarrollo del gabinete se usó el software de diseño Autodesk Inventor[3]. Se determinó armar un ensamble en que incluya todos los componentes y se lo pensó para poder luego ser impreso en 3D.

[1] https://www.altium.com/

[2] https://www.pcbwing.com/

[3] https://www.autodesk.com/products/inventor/overview?term=1-YEAR&tab=subscription

Para los componentes comprados, como la pantalla LCD y la matriz de botones, se tomaron los modelos 3D libres conseguidos a través de la plataforma GrabCAD[4]. Estos se verificaron para asegurarse que sus medidas correspondieran con el dispositivo físico. Por ultimo, el modelo de la plaqueta desarrollada se exportó desde el software Altium, donde se diseñó.

También, se priorizó el acceso a los conectores de la plaqueta, para permitir realizar cambios de cableados con poco esfuerzo, pero manteniendo los circuitos protegidos. Esto se puede ver en la Figura 3.1 donde se muestra el ensamble completo en Inventor. Notar que las piezas superiores, donde están la pantalla y el teclado, se muestran transparentes, para ayudar con la visualización.

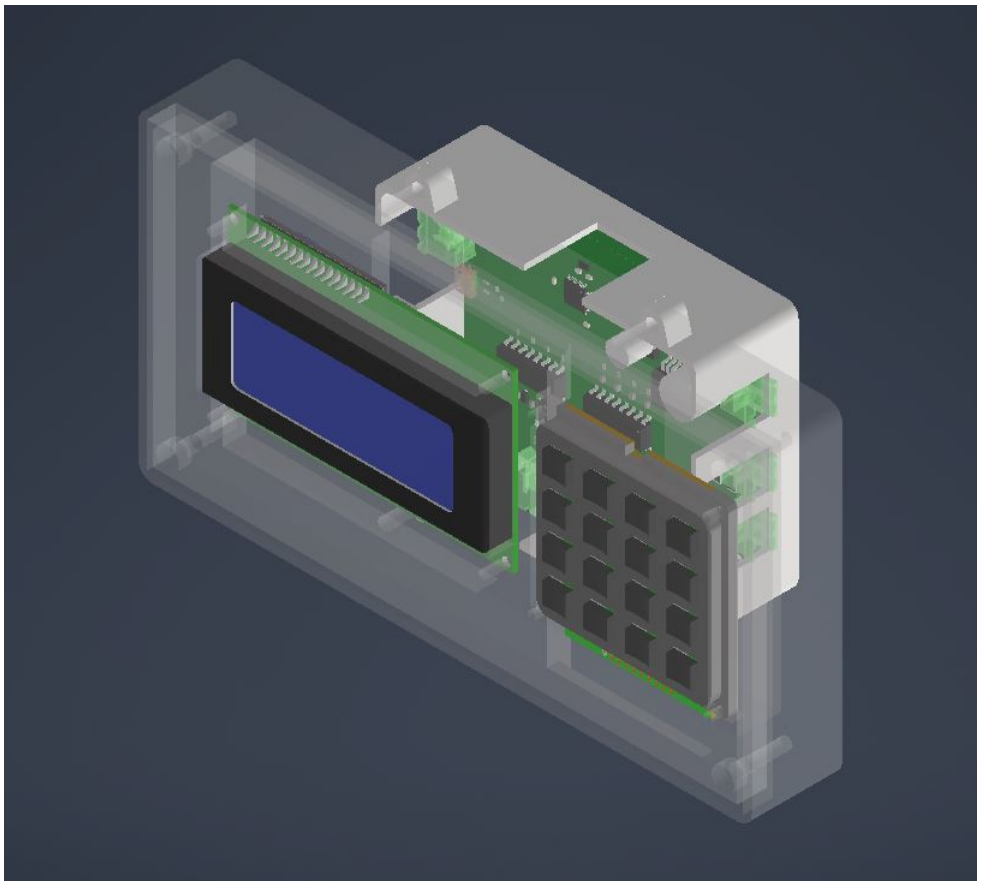

FIGURA 3.1. Ensamble 3D de gabinete

Como se puede ver, el ensamble consta de 3 componentes: una tapa delantera donde apoyan el display y el teclado, una caja intermedia que encierra a estos y una caja trasera donde se coloca la plaqueta de control. Se decidió por esta forma para permitir una presentación cómoda de los elementos de interfaz, para que el cableado interno quede protegido y para que se tenga fácil acceso a los conectores de la plaqueta, como se mecionó previamente.

Una vez cerrado el diseño, se exportó el archivo al programa UltiMaker Cura[5], donde se generaron los archivos de fabricación para impresión 3D. Se decidió

[4] https://grabcad.com/

[5] https://ultimaker.com/software/ultimaker-cura

utilizar PLA, que es uno de los plásticos más económicos y simples de manipular, y que cuenta con las características mecánicas apropiadas para la aplicación. Luego de fabricado, se forman las roscas en las distintas piezas y se ensambla el conjunto.

# Capítulo 4

# Ensayos y resultados

En este capítulo se detallan los ensayos y mediciones realizados sobre el sistema, para validar su funcionamiento y el cumplimiento de los requerimientos del trabajo. Se explicará cómo está compuesto el banco de pruebas utilizado y los instrumentos empleados, cómo se comprobó el envío de mensajes CAN a través de la red, qué ensayos eléctricos se realizaron sobre el sistema y cómo se verificó el envío de datos a través de UART y USB. Finalmente, se hablará de los ensayos que se realizaron en la planta de Cambre ICyFSA.

## 4.1. Banco de pruebas

Para validar el correcto funcionamiento del sistema, se ensambló un conjunto, se le cargó el firmware y se armó una red CAN con algunos motores paso a paso con plaquetas SN-17 conectadas. Dependiendo el ensayo a realizar, se conectaron 1 o más motores a la red, junto con un osciloscopio INSERTAR MARCA DEL OSCILOSCOPIO). Los dispositivos se alimentan con una fuente DC regulable YIHUA 305D[1] trabajando a 24 V.

En la Figura 4.1 puede verse un esquema de la composición del banco de pruebas y los conexionados. En la PC se corre un programa de monitor serial llamado PuTTy[2] que se emplea para visualizar de forma simplificada el mensaje que el sistema intenta envíar a la red CAN.

## 4.2. Ensayo de mensajes CAN

En la Figura 4.2 se puede ver una de las tramas CAN tomadas desde el osciloscopio. En esta, se pueden ver claramente las señales CAN-H y CAN-L en un estado recesivo a 2.5 V y separándose un poco más de 1 V en ambos sentidos en un estado dominante, este es el comportamiento esperado. Notar también la falta de ruido en la señal, lo que indica la correcta operación de los resistores de terminación

AGREGAR MEDICION DE TIEMPO

Para la verificación de los mensajes CAN se probaron cada uno de los mensajes posibles. Se revisó que el identificador y la data fueran correctas y que los sistemas conectados realizaran las acciones correspondientes.

[1] http://yihuasoldering.com/product-4-2-30v-dc-power-supply/160008/

[2] https://www.putty.org/

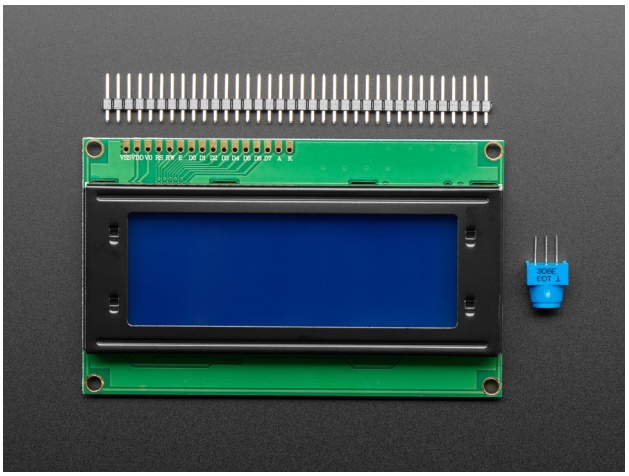

FIGURA 4.1. Banco de pruebas utilizado para verificaciones

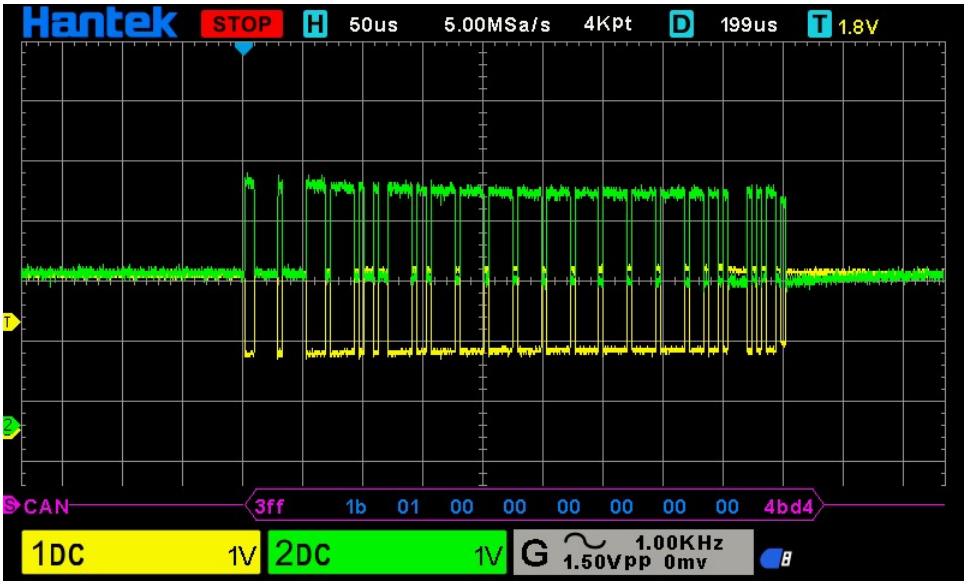


FIGURA 4.2. Níveles de señal CAN en osciloscopio

En la Figura 4.3 se puede ver una imagen tomada del osciloscopio para la instrucción de calibrar motor (código: 0x0c). En la parte iferior puede verse el decodificador CAN que el osciloscopio trae incorporado, marcando correctamente la instrucción. También, se puede ver el identificador del mensaje (código: 0x0b) compuesto por el identificador del motor (0x05) y el bit final indicando que es un mensaje desde el dispositivo controlador. El resto de los bytes de data se marcan en 0, ya que no es necesario envíar más información para este tipo de mensaje, los últimos bytes pertenecen al CRC.

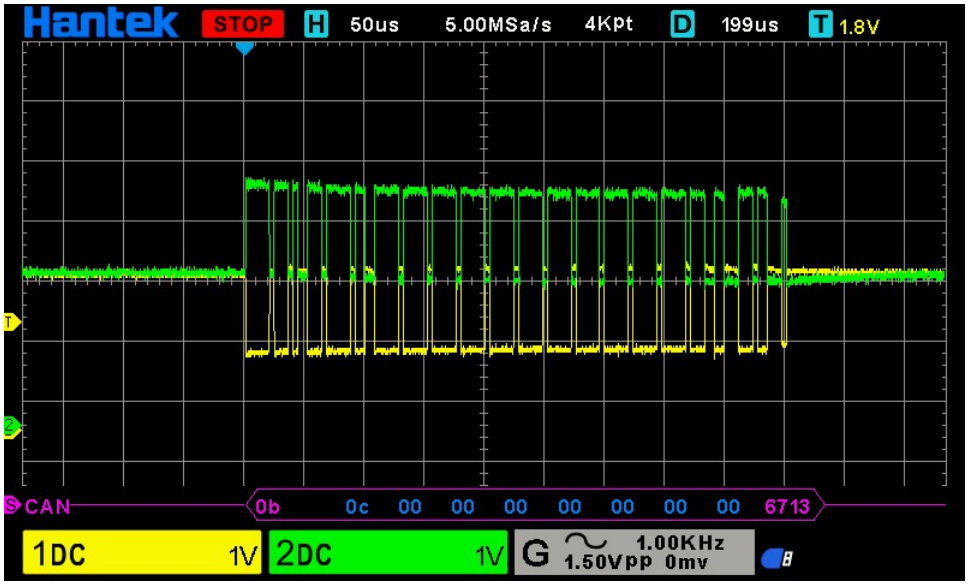


FIGURA 4.3. Instrucción calibrar motor

Otro ejemplo de mensaje puede visualizarse en la Figura 4.4 en la que se envía al motor un comando manual de movimiento angular (código: 0x10), en dirección en contra de las agujas del reloj (código 0x01), un ángulo de 360 grados (códigos: 0x01 y 0x68). En este caso, como este ángulo es un número que requiere más de un byte para su representación, aparece descompuesto en el mensaje. Si se hace la conversión del número 168, de hexadecimal a decimal, se comprueba que es efectivamente 360. Al recibir el mensaje, el motor correctamente gira lo estipulado.

## 4.3. Ensayos eléctricos

## 4.4. Ensayos de mensajes UART

## 4.5. Pruebas de funcionamiento en planta

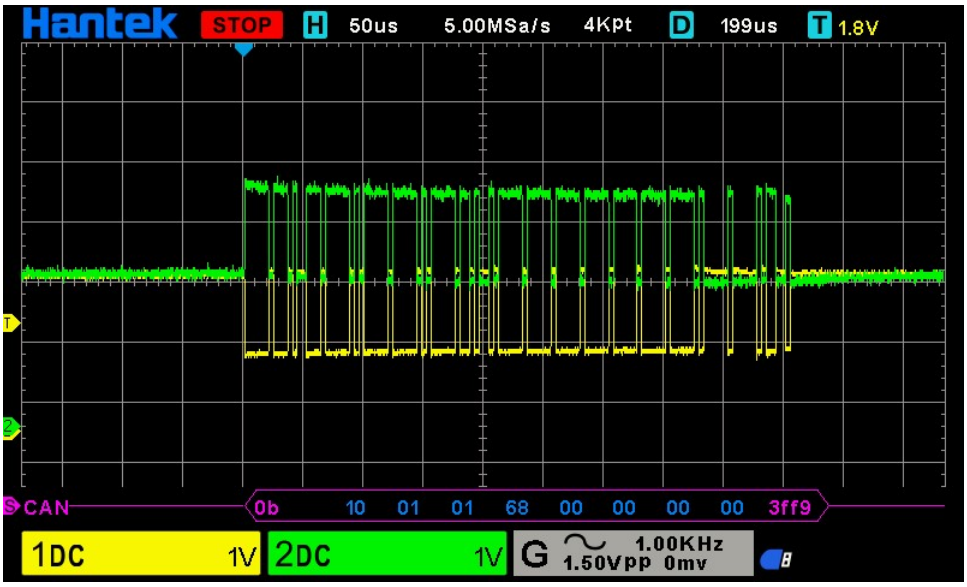


FIGURA 4.4. Instrucción manual mover motor

# Capítulo 5

# Conclusiones

## 5.1. Conclusiones generales

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puedo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

## 5.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.