HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

----- ○ 🕮 ○ -----

PROJECT I

DESIGNING CURRENT SOURCE USING DP DRIVE AND CANOPEN COMMUNICATION PROTOCOL

Instructor: PhD. Hoàng Mạnh Cường

Student: Phan Công Hoàng Quân
Student ID: 20222754

Hà Nội, 2025

Table of Contents

PREFACE

Nowadays, modern technology appears in every aspect of our life, and medical science is not an exception. The application of automation and artificial intelligence has created an extraordinary leap in the development of medical technology. The robotic dual-electromagnet actuation system for a magnetic capsule endoscope is a technology used in digestive organ diagnoses. To be specific, the capsule, equipped with permanent magnets, can be controlled under the effect of magnetic field which creates safe and deep penetration into human tissue. In this project, I am sincere to introduce about the topic "Designing Current Source using DP Drive", a small effort to create an electromagnet to control the WCE (wireless capsules endoscope), a part of the DEMA system. Besides, I also did a research about CANopen communication, which is used in controlling the drive.

This project is not only a combination of solid knowledge from theory to practice but also a reflection of the passion and relentless efforts of all team members.

We sincerely appreciate your attention and support throughout the implementation of this project.

We would like to express our heartfelt gratitude to Mr. Hoang Manh Cuong and the members of LIDP for accompanying and supporting us during the execution of this project.

CHAPTER 1: ABOUT CANOPEN COMMUNICATION PROTOCOL

1.1. Introduction

1.1.1. CAN (Controller Area Network) Protocol

The Controller Area Network (CAN) is a serial communication bus designed for robust and flexible performance in harsh environments, and particularly for industrial and automotive applications.

CAN defines the data link and physical layer of the Open Systems Interconnection (OSI) model, providing a low-level networking solution for high-speed in-vehicle communications. CAN was developed to reduce cable wiring, so the separate electronic control units (ECUs) inside a vehicle could communicate with only a single pair of wires.

1.1.2. CANopen Protocol

CANopen is a high-level communication protocol and device profile specification that is based on the CAN protocol. The protocol was developed for embedded networking applications, such as in-vehicle networks. The CANopen umbrella covers a network programming framework, device descriptions, interface definitions and application profiles. CANopen provides a protocol which standardizes communication between devices and applications from different manufacturers. It has been used in a wide range of industries, with highlights in automation and motion applications.

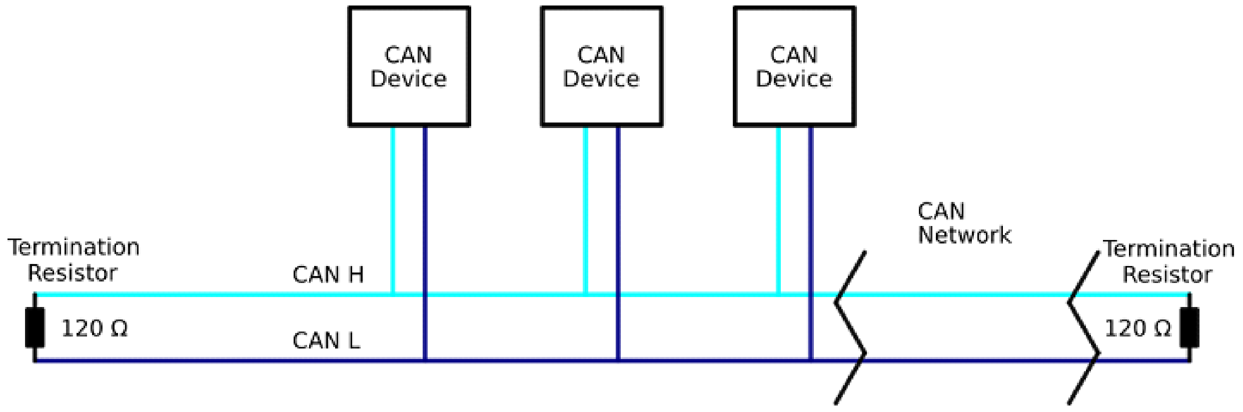


Figure 1. CANopen network model

1.2. OSI communication systems model

In terms of the OSI communication systems model, CAN layered architecture consists of the first two levels: the physical layer and the data link layer:

- Physical layer defines the lines used, voltages, high-speed nature, etc. The physical layer is responsible for the transmission of raw data.
- The data link layer includes the fact that CAN is a frame-based (messages) protocol. This layer is responsible for node-to-node data transfer. It allows you to establish and terminate the connection. It is also responsible for detecting and correcting the errors that may occur on the physical layer.

CANopen covers the top five layers: network (addressing, routing), transport (end-to-end reliability), session (synchronization), presentation (data encoded in standard way, data representation) and application. The application layer describes how to configure, transfer and synchronize CANopen devices.

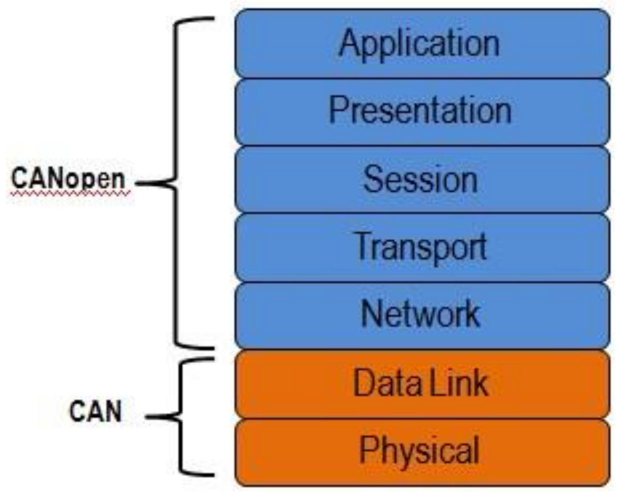


Figure 2. CAN and CANopen in the OSI Model

1.3. CAN layered architecture

1.3.1. CAN Framing

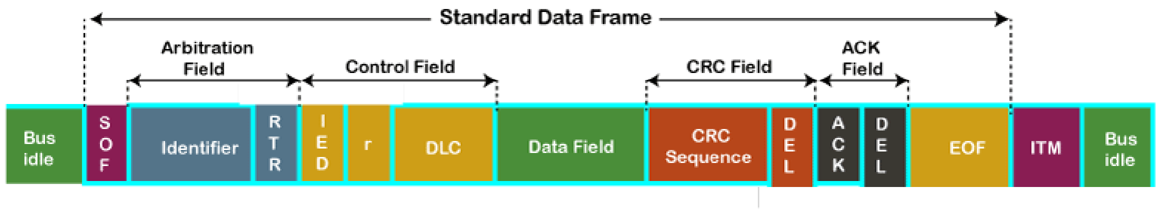


Figure 3. CAN frame

- **SOF:** SOF stands for the start of frame, which indicates that the new frame is entered in a network. It is of 1 bit.
- **Identifier:** A standard data format defined under CAN 2.0 A specification uses an 11-bit message identifier for arbitration. Basically, this message identifier sets the priority of the data frame.
- **RTR:** RTR stands for Remote Transmission Request, which defines the frame type, whether it is a data frame or a remote frame. It is of 1-bit.
- **Control field:** It has user-defined functions.

- **IDE:** An IDE bit in a control field stands for identifier extension. A dominant IDE bit defines the 11-bit standard identifier, whereas recessive IDE bit defines the 29-bit extended identifier.
- **DLC:** DLC stands for Data Length Code, which defines the data length in a data field. It is of 4 bits.

- **Data field:** The data field can contain up to 8 bytes.

- **CRC field:** The data frame also contains a cyclic redundancy check field of 15 bit, which is used to detect corruption if it occurs during the transmission time. The sender will compute the CRC before sending the data frame, and the receiver also computes the CRC and then compares the computed CRC with the CRC received from the sender. If the CRC does not match, then the receiver will generate the error.

- **ACK field:** This is the receiver's acknowledgment. In other protocols, a separate packet for an acknowledgment is sent after receiving all the packets, but in case of CAN protocol, no separate packet is sent for an acknowledgment.

- **EOF:** EOF stands for end of frame. It contains 7 consecutive recessive bits known End of frame.

1.3.2. CAN Communication Principle

- As we know, CAN messages are sent based on the priority set in the arbitration field: The smaller the message identifier, the higher the message priority would be.
- CAN messages are transmitted in these steps:
- Step 1: The sender wants to send the message and waits for the CAN bus to become idle. If the CAN bus is idle, then the sender sends the SOF or the dominant bit for the bus access
- Step 2: Then, it sends the message identifier bit in the most significant bit. If the node detects the dominant bit on the bus while it has transmitted the recessive bit, it means that the node has lost its arbitration and stops transmitting further bits.
- Step 3: The sender will wait and resend the message once the bus is free.

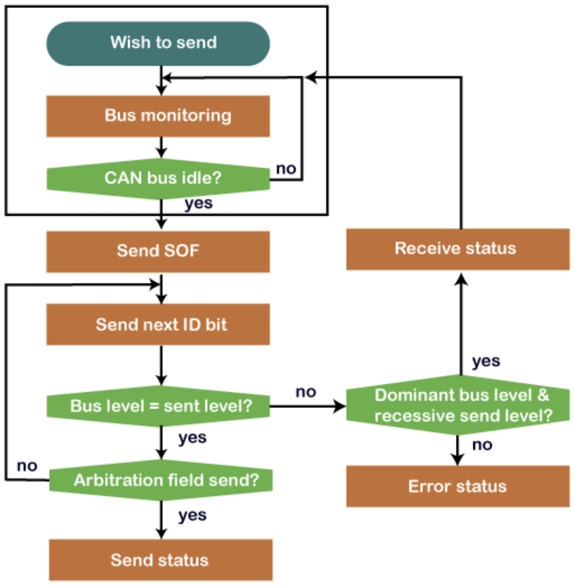


Figure 4. Flowchart of how arbitration works

1.3.3. CAN Network Characteristic

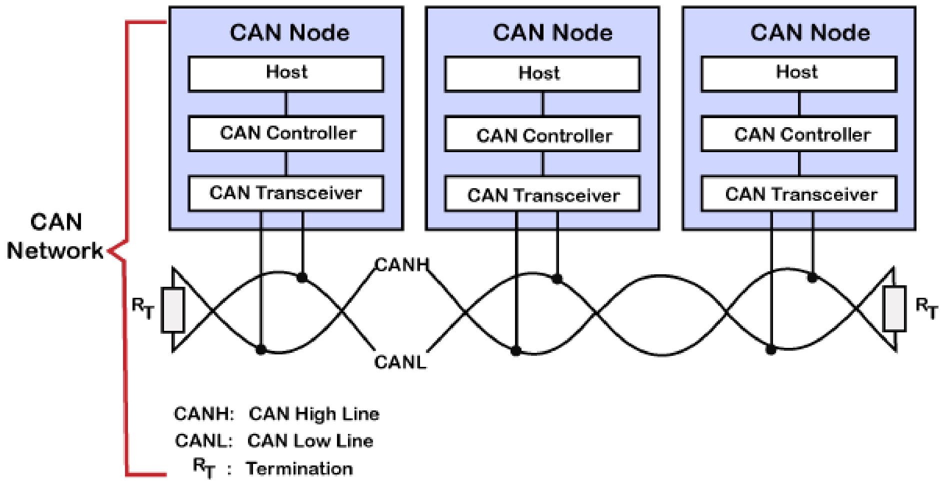


Figure 5. CAN Network

- A CAN Network consists of multiple nodes. Each node contains three main elements which are:
- Host: A host is a microcontroller or microprocessor which decides what the received message means and what message it should send next.
- CAN Controller: CAN controller deals with the communication functions described by the CAN protocol. It also triggers the transmission, or the reception of the CAN messages.
- CAN Transceiver: CAN transceiver is responsible for the transmission or reception of the data on the CAN bus. It converts the data signal into the stream of data collected from the CAN bus that the CAN controller can understand.
- Besides, a CAN bus consists of two lines, i.e., CAN LOW and CAN HIGH line. The transmission occurs due to the differential voltage applied to these lines. And unshielded twisted pair cables are used to transmit or receive data in the network. The CAN bus must be terminated at both ends by a 120-ohm resistor placed across CAN_H and CAN_L so that reflections of signals are avoided.

1.4. The Basics of the CANopen Application Layer

1.4.1. Object Dictionary

Object Dictionary (OD) is a standardized structure that stores configuration and process data. An object is roughly equivalent to a memory that holds a value. It is a requirement for all CANopen devices to implement an OD.

Each object is accessible with a 16-bit address called the object index. Some objects contain subcomponents with 8-bit addresses called sub-indices. The CANopen standard defines that certain addresses and address ranges must contain specific parameters.

OD is the method by which a CANopen device can be communicated through CANopen Messages. The basic data types included in the object dictionary are boolean, void (placeholder), unsigned integer, signed integer, floating point and character. More complex data types, such as strings, date and time can be constructed from the basic data types.

1.4.2. CANopen Message

a. CANopen Message Format

The message format for a CANopen frame is based on the CAN frame format where there are only two parts of the CAN frame the user needs to access, namely the Arbitration field, and the Data fields. All other fields are automatically configured by the CAN hardware.

- The Arbitration field:

+ COB-ID: Every message has a unique COB-ID that identifies the message type and in case of node specific messages, the node number. These COB-IDs begin with a base number and the addition of the NODE-ID completes the COB-ID.

+ RTR Bit: The remote transmission request bit is used in some specific cases when the host would like to request information from a node.

+ Node-ID: Every node on the CANopen network must have a unique node-ID, between 1 and 127.

- The Data field: The content of the Data field depends on the CANopen message type. Numerical data larger than 1 byte must be organized into “Little Endian” format.

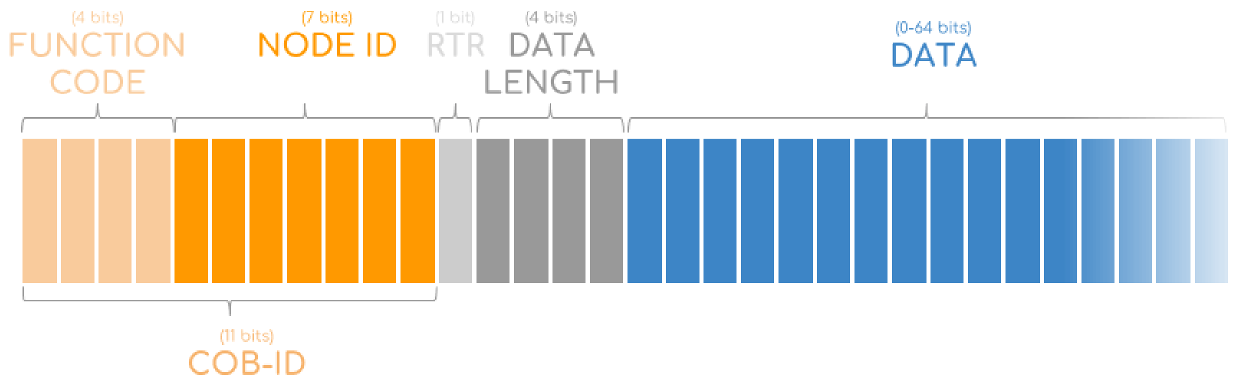


Figure 6. CANopen frame format

b. Network Management Message

Network management services include the ability to change the state of a slave between initializing, pre-operational, operational and stopped. The NMT protocol allows for the CANopen network to control the communication state of individual nodes. The pre-operational state is mainly used for the configuration of CANopen devices. As such, PDO communication is not permitted in the pre-operational state. PDO communication becomes possible in the operational state. In the stopped state, a node can only do node guarding or heartbeats but cannot receive or transmit messages. Certain types of CANopen communication are allowed in different states. For example, SDOs are allowed in the preoperational state, but PDOs are not. This is because SDOs are often used to initialize object dictionary parameters, whereas PDOs are often used to transfer continually updating data.

c. Guarding and Heartbeats Message

The CANopen specification requires that nodes must use some method to check whether a node is "alive" or not. The two methods available are: node guarding and heartbeats, with the latter being the preferred method.

In the heartbeat protocol, a CANopen node periodically sends out a heartbeat message which lets the CANopen master or the heartbeat consumer, know that the node is still alive. If a heartbeat message does not arrive within a certain period, the master can take specific action. Such an action might be to reset the node or to report an error to an operator.

In the node guarding protocol, the CANopen master polls the slave nodes for their current state information. If the node does not respond in a specific period, the master assumes the node is dead and will take an action.

d. Emergency messages

Each node in a CANopen network is assigned a single emergency (EMCY) message that communicates the node's status. Note that the heartbeats and node guarding protocol are intended to be used to convey communication failures, whereas emergency messages are used to convey errors within the node (i.e. sensor failure). An EMCY message is identified by a COB-ID of 80h + Node ID. The data portion of an EMCY message contains information about the error that occurred.

e. Service Data Object (SDO) messages

Definition: The CANopen protocol also specifies that each node on the network must implement a server that handles read/write requests to its object dictionary. This allows for a CANopen master to act as a client to that server. The mechanism for direct access (read/write) to the server's object dictionary is the **Service Data Object (SDO)**. The node whose object dictionary is accessed is referred to as the SDO server, and the node grabbing the data is referred to as the SDO client.

Operating principle: Typically, the master CANopen node will send a request to the network, and the node of interest will respond with the data requested. CANopen uses reserved message IDs to facilitate this communication. When the SDO client wants to request information from the server, it sends an SDO request using a CAN-ID of 600h + Node ID. The server will then respond using a CAN-ID of 580h + Node ID.

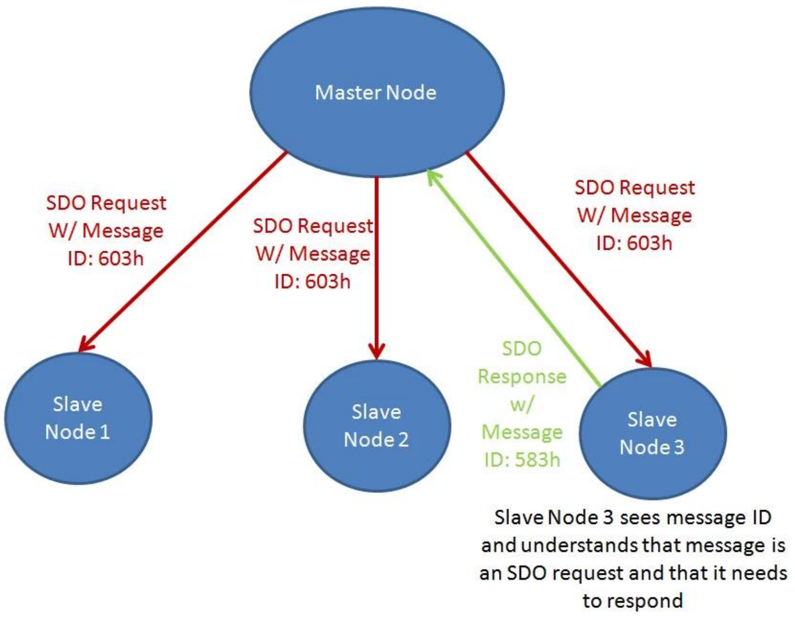

Master Node
SDO Request W/ Message ID: 603h
SDO Request W/ Message ID: 603h
SDO Request W/ Message ID: 603h
SDO Response w/ Message ID: 583h
Slave Node 1
Slave Node 2
Slave Node 3
Slave Node 3 sees message ID and understands that message is an SDO request and that it needs to respond

Figure 7. Example of SDO operation

f. Process Data Object (PDO) messages

Definition: Process Data Objects (PDO) is a different method used to transfer process data when data can be changing in time. Since SDO communication only allows access to one OD index at a time which leads to overhead for accessing continually changing data, PDO method is used.

Classification:

- Transfer PDOs (TPDOs): The data coming from the node
- Receive PDOs (RPDOs): The data coming to the node

Besides, there are two types of parameters in PDOs:

- The configuration parameters specify the COB-ID, the transmission type, inhibit time and the event timer. A PDO transfer can be initiated with different methods.

Type of transmission	*Description*
Event driven transmission	Initiated when the process data changes
Time driven transmission	Initiated at a fixed time interval
Individual polling	Initiated using remote request (not commonly used)
Synchronized polling	Initiated using SYNC signal (frequently used as a global timer)

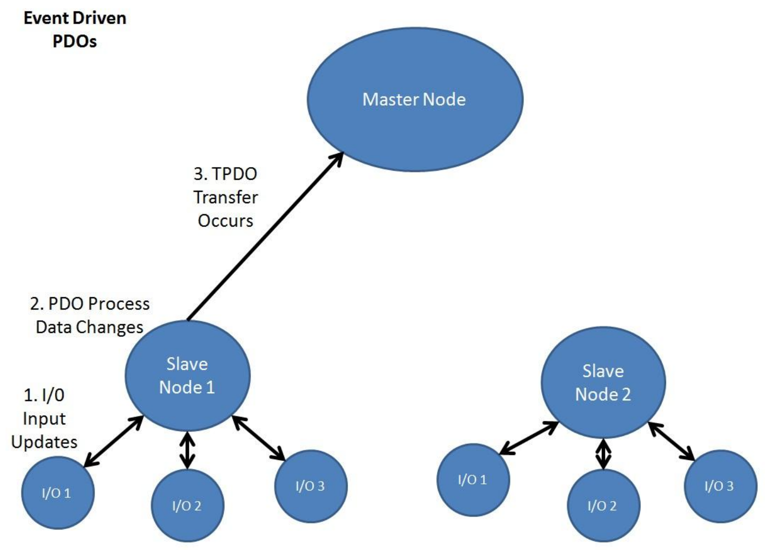
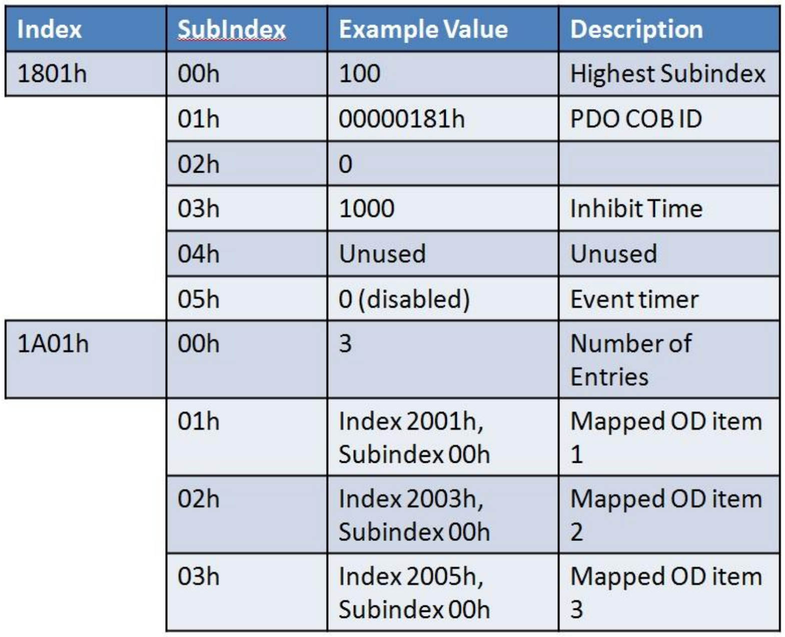


Figure 8. Example of event driven transmission

- The mapping parameters specify the object dictionary values sent by a single PDO message as the message may contain data from many object indices.



Index	SubIndex	Example Value	Description
1801h	00h	100	Highest Subindex
	01h	00000181h	PDO COB ID
	02h	0	
	03h	1000	Inhibit Time
	04h	Unused	Unused
	05h	0 (disabled)	Event timer
1A01h	00h	3	Number of Entries
	01h	Index 2001h, Subindex 00h	Mapped OD item 1
	02h	Index 2003h, Subindex 00h	Mapped OD item 2
	03h	Index 2005h, Subindex 00h	Mapped OD item 3

Figure 9. Example of TPDO Object Dictionary

CHAPTER 2: ABOUT DPCANIE-030A800 DRIVE

2.1. Introduction

DPCANIE-030A800 is in the DigiFlex Performance Series digital servo designed and produced by Advanced Motion Controls to drive brushed and brushless servo motors, stepper motors, and AC induction motors. The drive can be configured for a variety of external command signals. Commands can also be configured using the drive's built-in Motion Engine, an internal motion controller used with distributed motion applications. In addition to motor control, these drives feature dedicated and programmable digital and analog inputs and outputs to enhance interfacing with external controllers and devices.

This DP Series drive features a CANopen interface for networking and a RS-232 interface for drive configuration and setup.

In this project, I will use this drive as a current source to generate a desired stable DC current into the induction coil so that I can create an electromagnet for the DEMA system.

Figure 10. DPCANIE -030A800

2.2. Block diagram of the drive

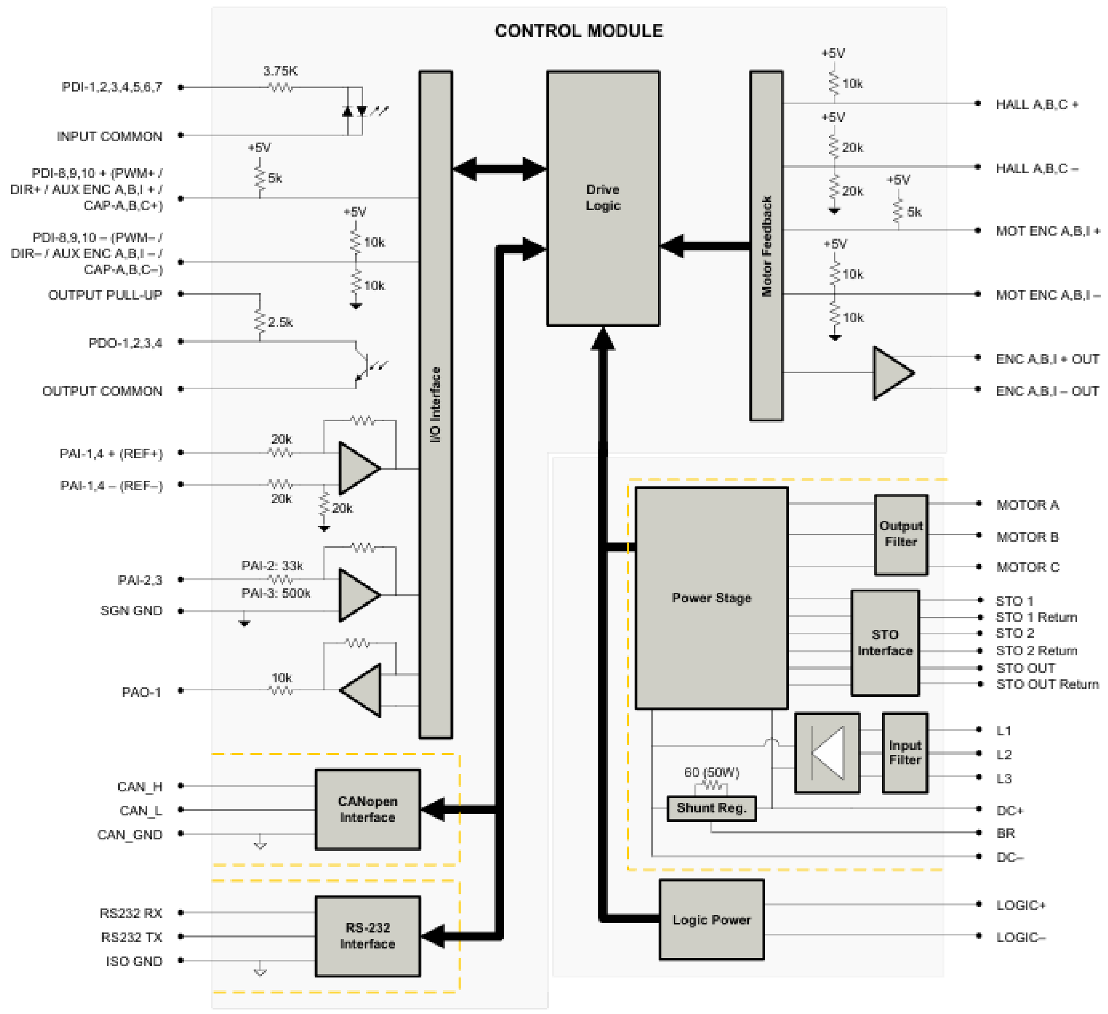


Figure 11. Block diagram of DPCANIE-030A800 drive

2.3. Features of the drive

• **Modes of Operation:**	• **Command Source:**	• **Feedback Supported:**	• **Power Specification**
- Profile Modes - Cyclic Synchronous Modes - Current - Velocity - Position	- 10V Analog - PWM and Direction - Encoder Following - Over the Network - Sequencing - Indexing	- 10VDC Position - Halls - Incremental Encoder - Auxiliary Incremental Encoder	- Rated Voltage: 480VAC - AC supply voltage range : 200-480VAC - AC Input phases : 3 - AC supply frequency : 50-60Hz - Logic Supply voltage : 20-30VDC (850 mA)

- Interpolated Position Mode (PVT)	- Jogging	- Tachometer (10VDC)	

2.4. Communication Protocol

2.4.1. CANopen

- DP series drive features a CANopen interface for networking. DPCANIE-030A800 drives use the CAN Physical Layer as defined by the CAN in Automation (CIA) standards DS-102 V2.0.
- **Features of CANopen:**

- 3-wire bus is all that is needed to connect drives together (CAN_H, CAN_L and GND).
- Differential transmission for noise immunity.
- Up to 1Mbit/sec speeds possible.
- Up to 128 nodes per CAN network.
- Robust message arbitration with collision detection/prevention built into the physical layer.
- Bi-directional (non-polled) communication possible.
- The CAN bus must be terminated at both ends by a 120-ohm resistor placed across CAN_H and CAN_L so that reflections of signals are avoided.

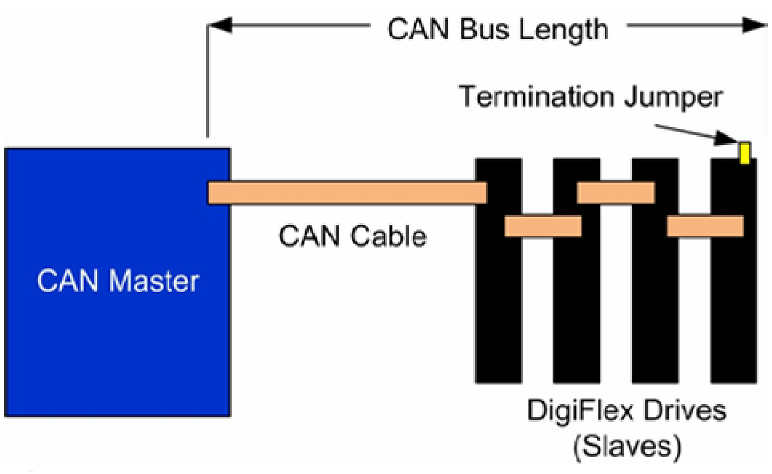


Figure 12. Drives network through CANopen

2.4.2. RS (Recommended Standard) 232

- DP CANopen drives also support RS232 as a secondary communication channel. RS232 interface is used for drive configuration and setup. The DriveWare software can run over the RS232 channel during operation to monitor quantities in real time making system design and commissioning fast.
- **Definition:** RS232 is a standard protocol used for serial communication. It defines serial communication using DTE (Data Terminal Equipment) and DCE (Data Communication Equipment) signals.
- **Features:**

- Inexpensive hardware.
- Simple 3 wire bus (TX, RX, and GND).
- Speeds up to 155.2K baud are possible.
- It allows simultaneous two-way communication.

- **How it works:**

- Data Bits: The data is broken down into individual bits (binary digits: 0s and 1s).
- Start Bit: The transmission begins with a start bit (usually 0), signaling the beginning of a new data packet.
- Data Bits Transmission: The data bits are sent one after another in a specific order, typically from the least significant bit (LSB) to the most significant bit (MSB).
- Parity Bit (Optional): An optional parity bit can be included for error checking.
- Stop Bits: One or more stop bits (usually 1) signal the end of the data packet.

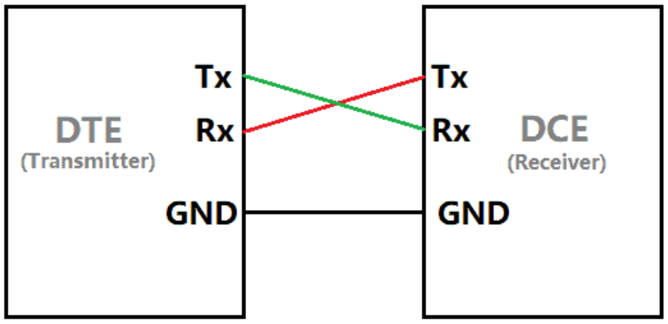


Figure 13. RS232 model

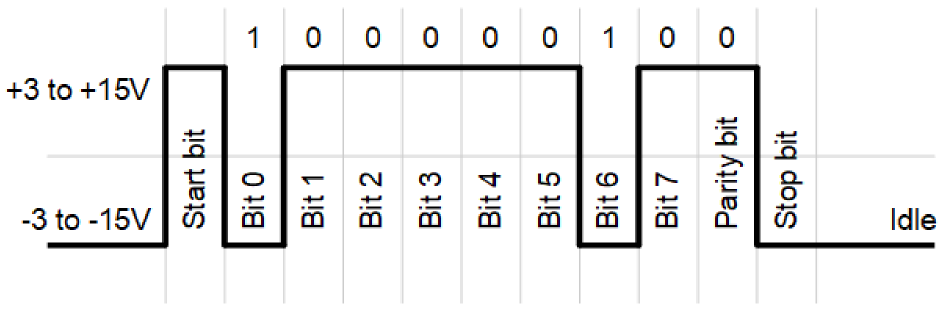


Figure 14. RS232 message frame format

2.5. Mode of operation

- In this DEMA system, I control the drive in Profile Current Mode to generate a desired stable DC current.
- Profile Current Mode allows users to:
- Directly command the motor current (or torque, which is proportional to current).
- Bypass higher-level control loops like velocity or position control.
- Achieve precise torque regulation by directly controlling the electrical current in the motor windings because electric torque is merely a constant Kt multiplied by a magnitude of current.
- How Profile Current Mode works:
- Setpoint Input: The system receives a current setpoint (in mA, A, or a proportional value) via the **CANopen network**.
- Current Regulation:
  + The servo drive uses its internal **PI controller** to regulate the actual motor current to match the commanded setpoint.
  + Feedback from motor current sensors (e.g., shunt resistors, Hall sensors) is used for accurate control.

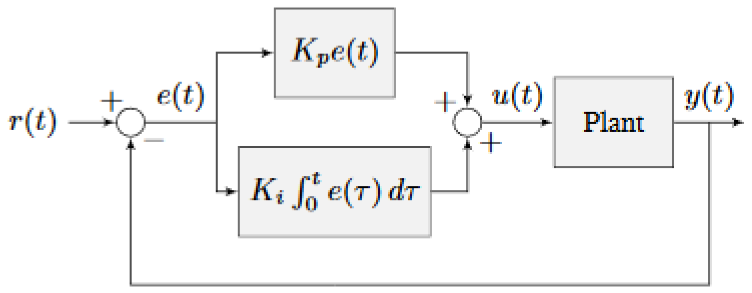


Figure 15. PI controller

2.6. DriveWare

- DriveWare 7, also developed by Advanced Motion Controls, is free software, which is used to commission, configure, troubleshoot and integrate all AMC DP servo drives.

- All drive limits, control loops (current, velocity, and position) and event handling can be configured in DriveWare
- Features:
- Easy-to-use interface common to everyday applications
- Up to 16 unique Indexes - relative or absolute motion profiles
- Combine Indexes with Homing routines and other control functions to create up to 16 different Sequences
- Built-in oscilloscope and waveform generator
- Real-time gain changes to optimize tuning and achieve the highest performance
- On-the-fly Mode Switching
- On-the-fly Gain Set Switching

CHAPTER 3: HARDWARE DESIGNING

3.1. Block diagram

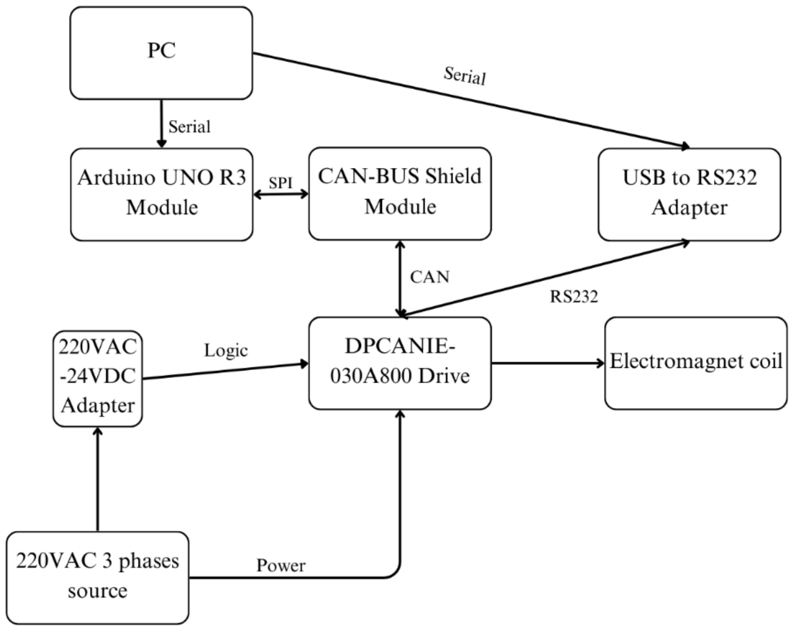


Figure 16. Block diagram of the system

To control the drive to generate a constant desired current, we must design a system to send CAN messages to operate the exact function of the drive. Also, the system should be controlled through an optimized and friendly interface.

- **Power source:** Supply the power source to the drive.
- **Logic source:** Supply the logic power to the drive.
- **Arduino UNO R6 Module:** Work as a “brain” of the system with the main mission is to send the messages to the drive.
- **CAN-BUS Shield Module:** Work as a transceiver, receive SPI messages from the MCU module and transmit CAN message to the drive.
- **USB to RS232 Adapter:** Connect the drive to the PC to configure the function through DriveWare.
- **Coil:** Work as an electromagnet with constant DC current generated by the drive.

3.2. Details of the components

3.2.1. Arduino UNO R3 Module

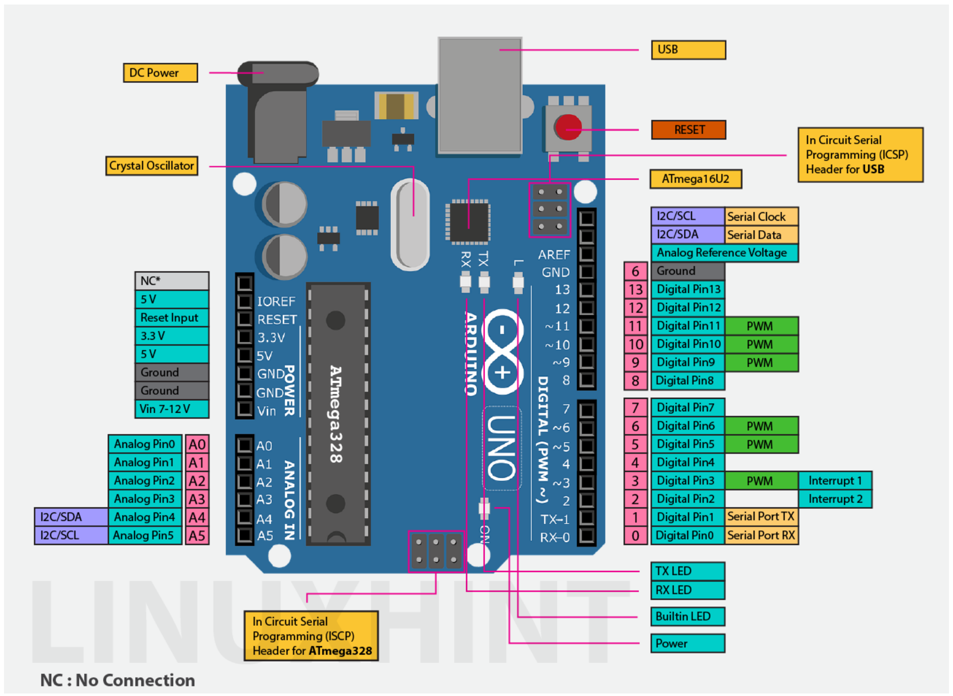


Figure 17. Arduino UNO R3 Module

- **CPU:**

- CPU: ATmega328P microprocessor
- High performance, low power AVR 8-bit microcontroller
- EEPROM: 1KB
- SRAM: 2KB
- Clock speed: 16MHz

• **Peripheral features:**

- Two 8-bit Timer/Counters with separate prescaler and compare mode
- One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
- Real time counter with separate oscillator
- Six PWM channels
- Master/slave SPI serial interface
- Byte-oriented 2-wire serial interface (Phillips I2C compatible)
- Programmable watchdog timer with separate on-chip oscillator
- On-chip analog comparator
- Interrupt and wake up on pin change

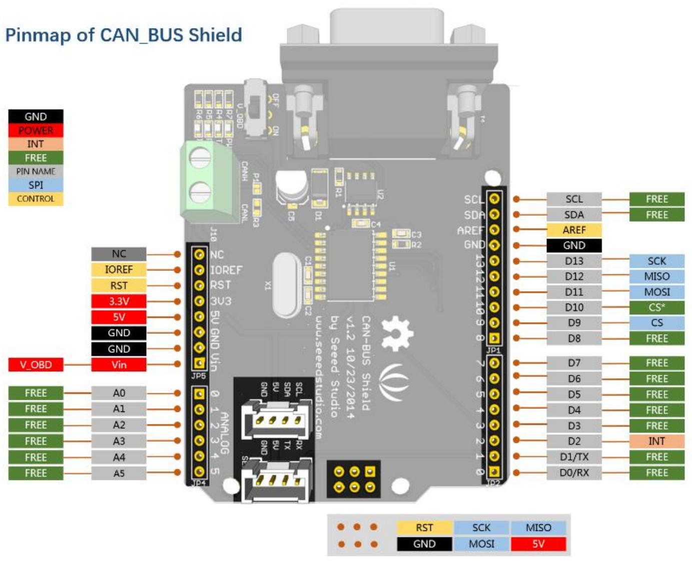
• **Operating voltage:** 2.7V to 5V

• **Temperature range:** –40°C to +125°C

• **Speed grade:**

- 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
- 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)

3.2.2. CAN-BUS Shield Module



- This CAN-BUS Shield adopts **MCP2515** CAN Bus controller with SPI interface and **MCP2551** CAN transceiver to give Arduino/Seeeduino CAN-BUS capability.
- The MCP2551 is a high-speed CAN transceiver, fault-tolerant device that serves as the interface between a CAN protocol controller and the physical bus. MCP2551 provides differential transmit and receive capability for the CAN protocol controller and is fully compatible with the ISO-11898 standard, including 24V requirements. It will operate at speeds of up to 1 Mb/s.
- **Features of MCP2551:**

- Slope control input
- Supports 1 Mb/s operation
- Implements ISO-11898 standard physical layer requirements
- Suitable for 12V and 24V systems
- Externally controlled slope for reduced RFI emissions
- Permanent dominants detect
- Low current standby operation
- High noise immunity due to differential bus implementation

- **Features of the module:**

- Implements CAN V2.0B speed up to 1 Mb/s
- SPI Interface speed up to 10 MHz
- Standard (11 bit) and extended (29 bit) data and remote frames
- Two receive buffers with prioritized message storage
- Industrial standard DB-9 connector
- LED indicators

3.2.3. USB to RS232 adapter

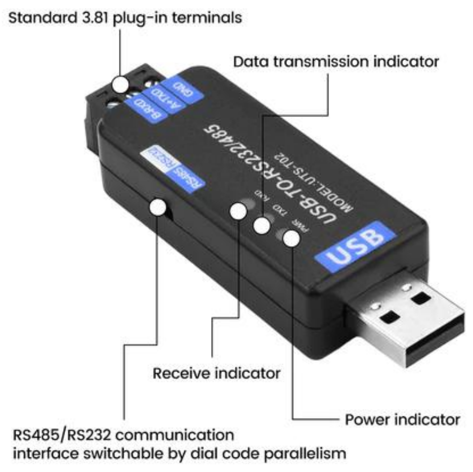

Standard 3.81 plug-in terminals
Data transmission indicator
GND
A+TXD
B-RXD
USB-TO-RS232/485
MODEL-UTS-T02
RXD
TXD
PWR
USB
Receive indicator
Power indicator
RS485/RS232 communication interface switchable by dial code parallelism

Figure 18. USB to RS232 adapter

- The DPCANIE-030A800 Drive is configured through RS232 interface by using DriveWare software. Hence, the USB to RS232 adapter is used to connect the PC and the drive.
- The adapter utilizes the CH343G Chip which has plug-and-play compatibility with Windows, macOS, Linux, ...
- **Features:**

- Communication Speed: 50 bps to 1,000,000 bps.
- Transmission mode : point to multipoint, point to point.
- Interface protection :
  + 600W surge protection.
  +15kV ESD protection.
  +Built-in 120Ω termination resistor for balanced resistance.
- Transmission distance: Up to 10m at low speed

- **USB to RS485/RS232 Interface Description:**

- GND: Connects to external GND.
- A+ (TXD): RS485 connects to A, RS232 connects to RXD.
- B- (RXD): RS485 connects to B, RS232 connects to TXD.

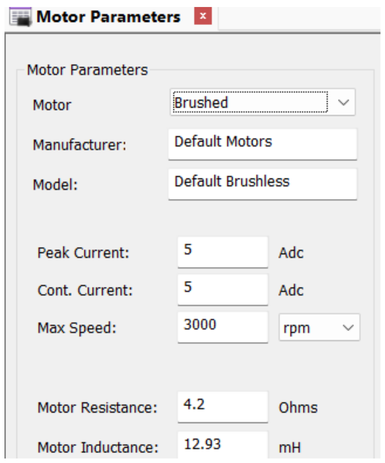
CHAPTER 4: FIRMWARE AND CONTROL INTERFACE DESIGNING

4.1. Controlling output current using CANopen messages

4.1.1. Drive configuring using DriveWare

- **Command source:**
  Interface Input: Used for RS-485/RS-232 as the command source.

- **Coil parameter:**

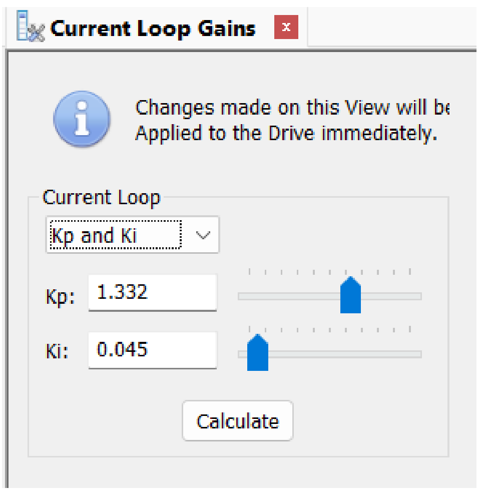

Motor Parameters

Motor Parameters

Motor	Brushed	
Manufacturer:	Default Motors	
Model:	Default Brushless	
Peak Current:	5	Adc
Cont. Current:	5	Adc
Max Speed:	3000	rpm
Motor Resistance:	4.2	Ohms
Motor Inductance:	12.93	mH

Figure 19.Coil parameters

- **Current loop gains**: the gain PI parameters are calculated using the parameters of the coil so that the desired current can be generated in the coil.


Current Loop Gains

Changes made on this View will be Applied to the Drive immediately.

Current Loop

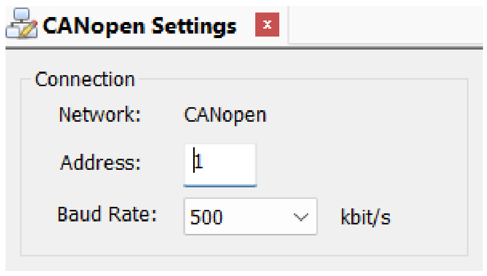
Kp and Ki

Kp: 1.332

Ki: 0.045

Figure 20. Calculated PI parameters

- **Network settings:**


CANopen Settings

Connection

Network: CANopen

Address: 1

Baud Rate: 500 kbit/s

Figure 21. CANopen settings

4.1.2. Controlling by sending CAN messages

a) Flowchart of message sending

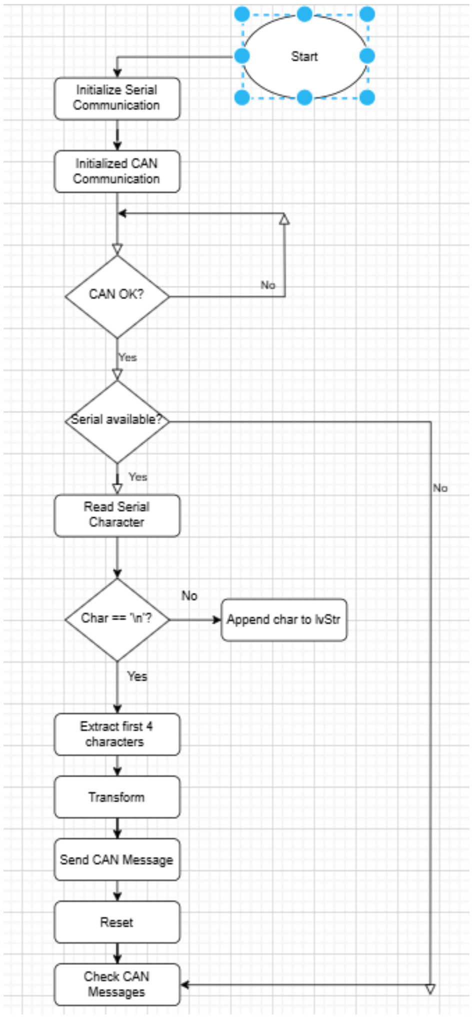


Figure 22. Message sending flowchart

b) How CAN messages control output current

- The current is obtained following these steps:

- Convert Amps to drive units:

+ Use DC2 scale factor for Interface Input:

DC2 = (2^15) / (Kp) = (2^15)/ (30) = 1092.267

+ (Desired Current) * (Scale Factor) = (integer value)

(x Amp (Desired current)) * 1092.267

+ Round to nearest int and convert to hex

+ Divide the hex value in to High Byte and Low Byte

- Then the message is sent in Little Endian Format:

COB-ID (600h+ Node-ID)	Host Initiates Write SDO (4 or less data bytes)	Object Index (LSB)	Object Index (MSB)	Sub-Index	Data	Data	Data	Data
601h	22h	45	20	01	Low Byte	High Byte	00	00

4.2. Introduction of Human-Machine Interface (HMI)

- Human-Machine Interface (HMI) is a user interface or dashboard connecting a person to a machine, system. HMI is most used in the context of an industrial process.
- In industrial settings, HMIs can be used to visually display data; track production time trends; monitor machine inputs and outputs.
- Commonly, HMI communicates with Programmable Logic Controllers (PLCs) and input/output sensors to get and display information. HMI screens can be used for a single function, i.e. monitoring, tracking and performing other complicated operations.

4.3. Introduction to LabVIEW

- **Definition:** LabVIEW is a graphical programming environment that provides unique productivity accelerators for test system development, such as an intuitive approach to programming, connectivity to any instrument, and fully integrated user interfaces.
- **Feature:**

- Ability to connect to any instrument, regardless of vendor
- Contain a native user interface for controlling and monitoring test
- Consist of thousands of engineering analysis functions
- Compatible with popular programming languages such as Python, C, …

- **Applications:**

- Acquiring data and control instruments
- Monitoring and interacting with experiment test
- Operating communication with industry protocols
- Developing with graphical programming
- Gaining insights from data
- Adding code from other programming languages

4.4. LabVIEW interface in the project

4.4.1. Functions of the LabVIEW control interface

- **User Input:** Allows users to enter controls (current value).
- **Data Display:** Visual representation of measured output current value.
- **Control Buttons:** commands for sending control signals to generate the desired current.
- **Hardware Communication:** Using Serial to receive data (encoded current value).

4.4.2. Interface design

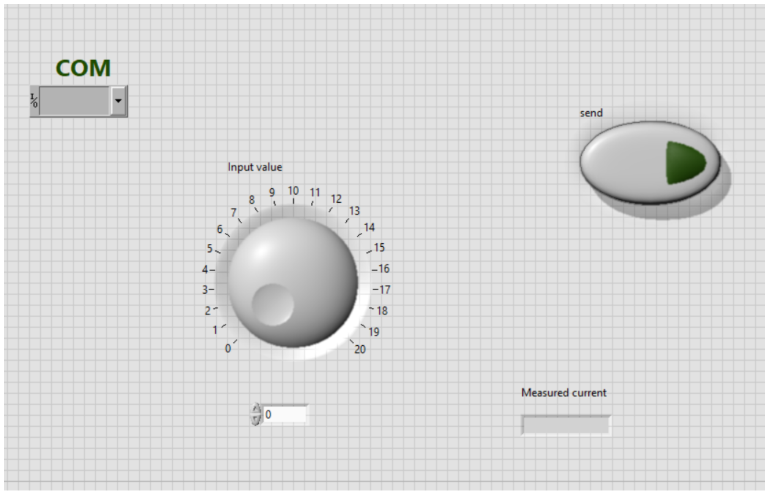


Figure 23. Front Panel

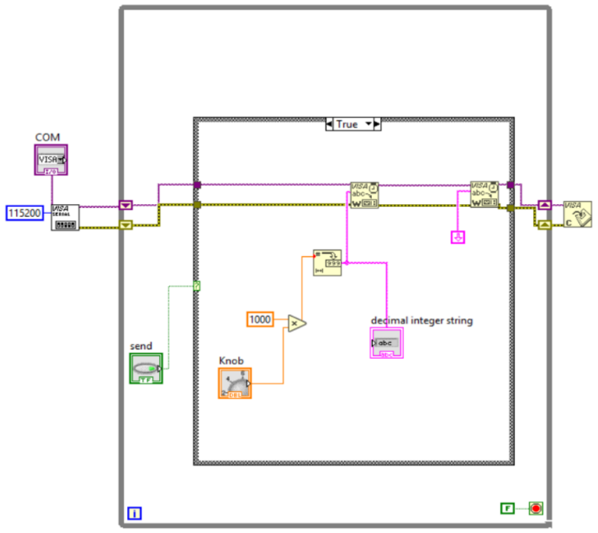


Figure 24. Block Diagram

CHAPTER 5: EXPERIMENT AND EVALUATION

5.1. Experimenting on the final system

5.1.1. Operating the experiment

- The experiment is operated successfully that different input values are applied to check if the output values are exact as desired. Digital multimeter is used to measure the current in the coil.
- Besides, an MCB is used in the input power of the drive to ensure safety to the system and the users.

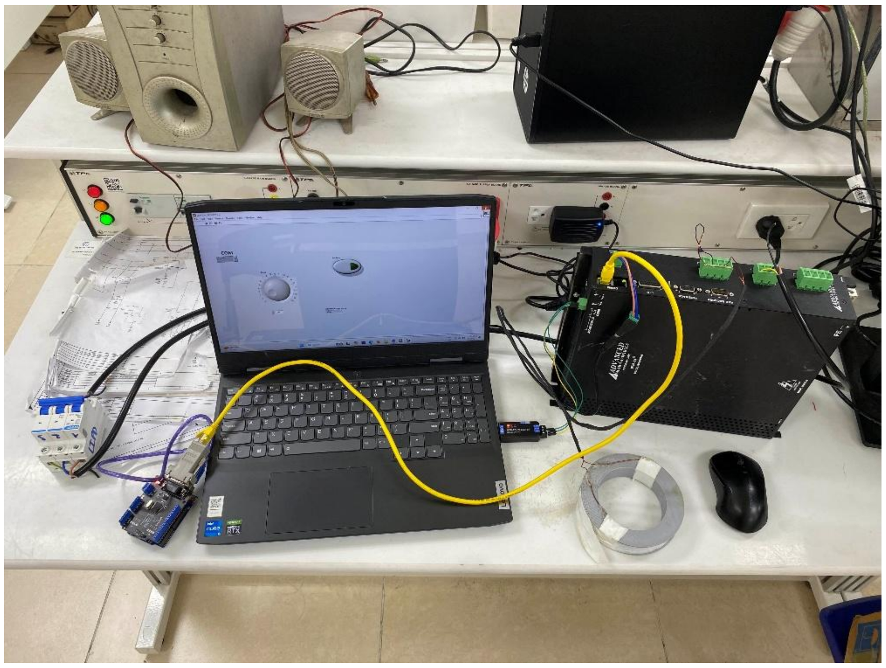


Figure 25. Experiment setup

5.1.2. Experiment results

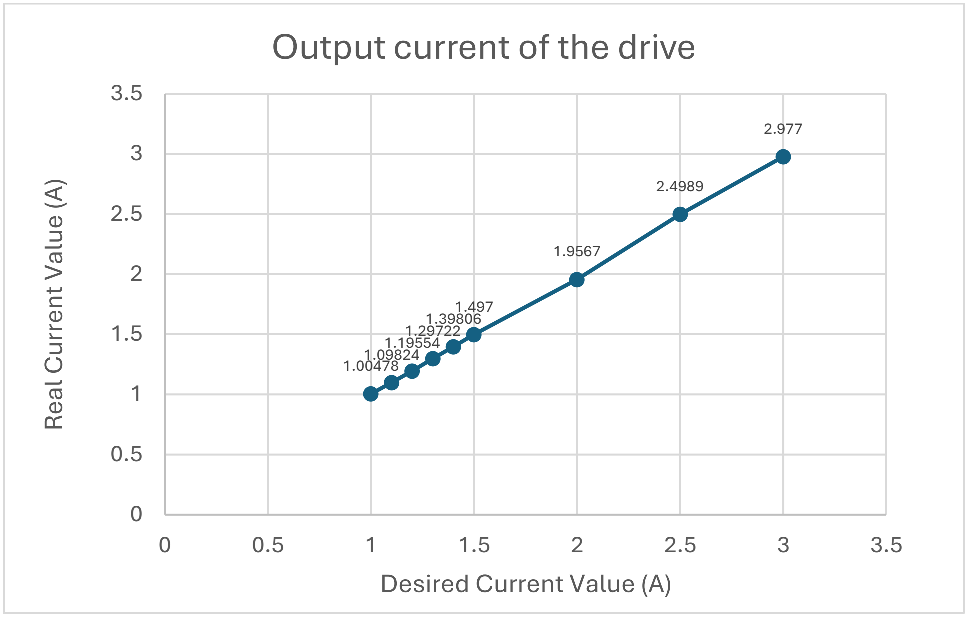


Figure 26. Result graph

5.2. Results evaluation

- Overall, the system has reached the fundamental requirements of the DEMA system that the output current is exactly as desired with an accuracy of 3 decimal places.
- However, there are still some drawbacks of the system that the response speed is not optimized due to the design of the system, especially with the MCU and the CAN transceiver.
- Besides, the LabVIEW interface is also not totally user-friendly with the lack of essential functions and dashboards.

5.3. Development orientation for the project.

- Self-design a PCB with different microcontrollers to optimize CAN communication with the drive.
- Complete the better version of LabVIEW control interface
- Including more drives into the system to create a CAN multi-device system.

REFERENCES

- CAN and CANopen protocols:

The basics of CANOpen. (2013, August 21). NI. https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-basics-of-canopen.html

CAN (Controller Area Network) protocol - javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/can-protocol

ADVANCED Motion Controls. (2025, January 30). *DPCANIE-030A800 CANopen servo drive*. https://www.a-m-c.com/product/dpcanie-030a800/

- DEMA system:

Hoang, M. C., Kim, J., Park, J., & Kim, C. (2022). Six-DOF Localization using Magnetic Induction Effect for Automated Locomotion of an Active Capsule Endoscope. *2022 9th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, 1–6. https://doi.org/10.1109/biorob52689.2022.9925464

- System design:

CAN-BUS Shield v2.0 | Seeed Studio Wiki. (2023, January 10). https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/

Arduino UNO - JavaTPoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/arduino-uno

- LabVIEW :

GeeksforGeeks. (2022, November 9). *What is LabVIEW?* GeeksforGeeks.

https://www.geeksforgeeks.org/what-is-labview/