VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Yen Dang

# DESIGN AND IMPLEMENTATION OF DIGITAL CURRENT LOOP DEVICE FOR FREQUENCY CONVERTER

School of Technology
2024

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

## ABSTRACT

| | |
|---|---|
| Author | Yen Dang |
| Title | Design and Implementation of Digital Current Loop Device for Frequency Converter |
| Year | 2024 |
| Language | English |
| Pages | 49 |
| Name of Supervisor | Jani Ahvonen |

This thesis focuses on the design and implementation of a digital current loop device for a frequency converter to serve as an intermediate device between the motor and the frequency converter. In addition, this current loop device has an advanced feature of fast and slow increments or decrements, which is very convenient and user-friendly for users.

The work methods for this project were split into three main stages: circuit design and prototyping, Arduino software development and manufacturing the physical Printed Circuit Boards (PCB) for the digital current loop device. The expected inputs such as torque (Nm) and speed (rpm) values were fed into a rotary encoder and were displayed on an LCD I2C screen simultaneously. Then, the digital inputs for the torque (0 – 50 Nm) and for the speed (0 – 3000 rpm) were converted into the 0 – 20 mA current loops through DACs connected to the Arduino Nano board. In addition, the prototype measurement was divided into four main parts: Filter, Digital Signal Processing, Digital Analog Converter (DAC) and Analog to Digital Converter (ADC). These individual circuits on breadboards were verified with the software program simultaneously to achieve the accurate outcomes before integrating them together.

The results from the integrated PCBs were consistent with the individual circuit initially tested on the breadboards. Consequently, the desired torque and speed values showed positive correlations with the corresponding current values ranging from 0 – 20 mA. The finished device was assembled by placing PCBs into two metal boxes connected via CAT6 cable. To ensure accuracy and eliminate the capacitive crosstalk issue and frequency interference, signal wires were paired and proper grounding and shielding techniques were implemented. These measures were crucial for maintaining signal stability in potentially noisy environments, ultimately enhancing the accuracy of the 0 – 20 mA current loop outputs.

Keywords Encoder, LCD I2C, Arduino Nano, DAC, current loop, ADC, digital current loop device, frequency converter

## CONTENTS

**LIST OF TABLES**



**LIST OF FIGURES**

# 1 INTRODUCTION

The objective of the thesis is to design and implement a digital 0 – 20 mA device for frequency converter. Users can adjust the expected values such as torque and speed values on an LCD by rotating an encoder. The design of this device is illustrated through hardware and software diagrams.

## 1.1 Block Hardware Diagram

The hardware block diagram of the device is designed as shown below.

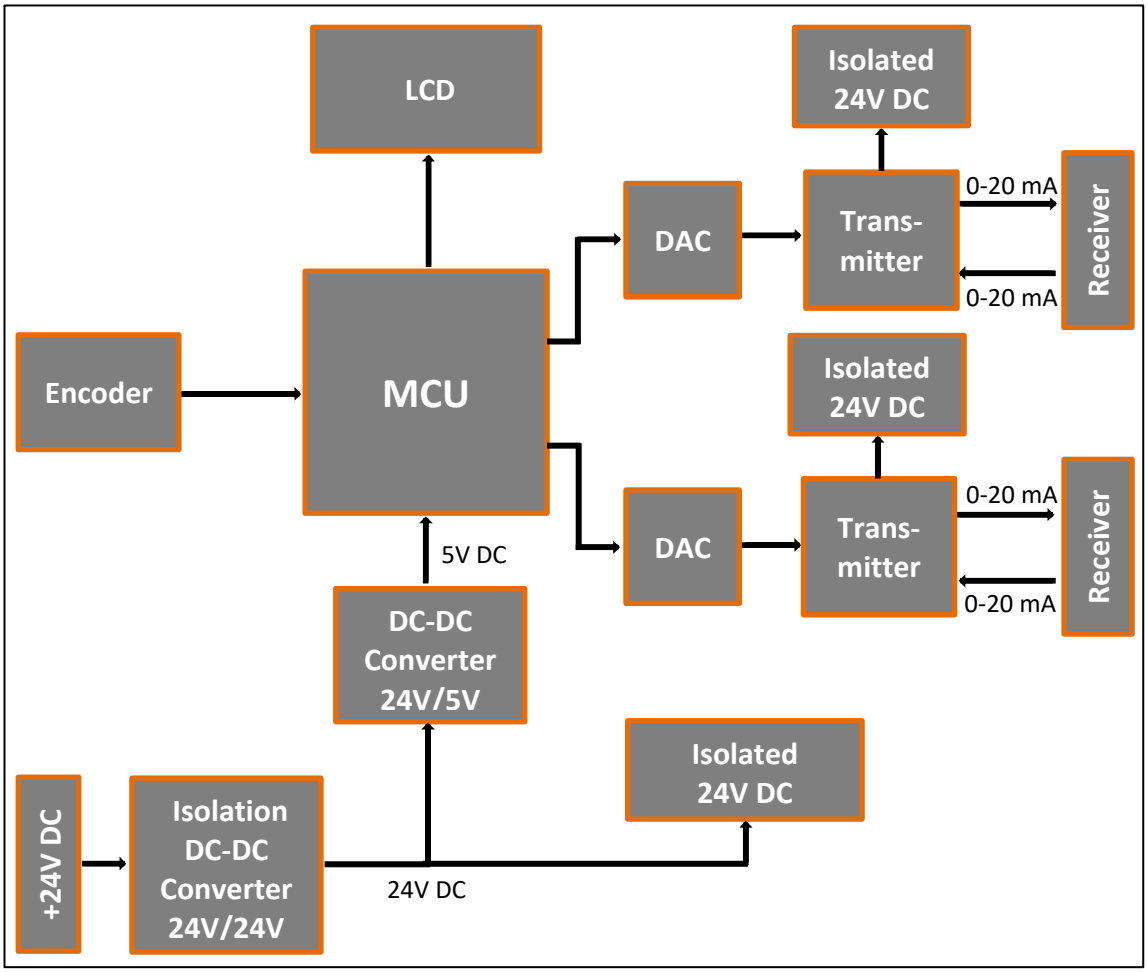


**Figure 1.** The hardware block diagram

Shown in the diagram, the encoder would be used to set the desired current value by users. The desired digital value would be fed into MCU. Then, MCU would process the data by displaying it on the LCD screen. MCU would send the desired digital value to a DAC. The DAC would convert the digital data back into the analog

format. The analog output from DAC would control the current levels in the current loops (transmitter and receiver).

### 1.2 Flowchart Software Diagram

The program for the device is presented in Figure 2.

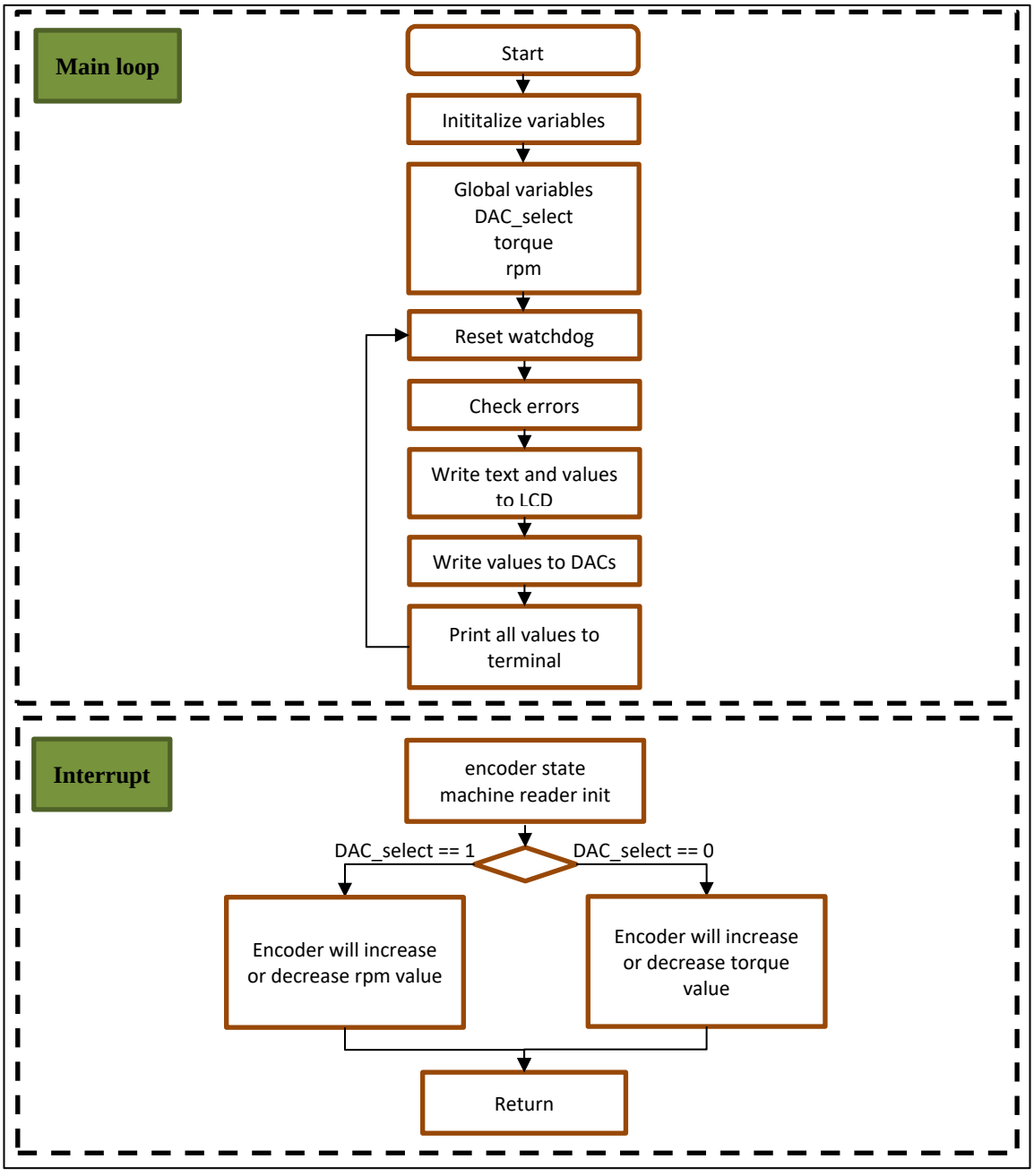


**Figure 2.** The flowchart software diagram.

The main loop comprises five functions such as reset watchdog, check errors, write text and values to LCD, write values to DACs and print all values to terminal. Each of these fuctions has the specific internal behaviour within the system. The reset watchdog serves as a critical safety mechanism by periodically resetting the

watchdog timer to prevent the system from freezes or hangs, thus ensuring continuous operation. It acts as a fail-safe to automatically restart the system if it becomes unresponsive. Next, the check errors function continuously monitors the system for any potential errors by comparing the desired torque and rpm values with their corresponding ADC feedback. It returns a combined error code: 1 for a torque channel error, 2 for an rpm channel error, or 3 if both channels have errors. Another crucial function is the write text and values to LCD, which updates the user interface by displaying relevant errors, torque, and rpm values on the LCD screen. It provides users with real-time information about the system's status and operation. Furthermore, the write values to DACs function converts digital signals to analog outputs through DACs. It calculates new DAC values when the changes occur, then updates the appropriate channels based on the values of the global DAC_select variable. Lastly, the print all values to terminal function outputs all relevant system values and information to a terminal interface. It is essential for debugging and monitoring purposes, allowing users or operators to access detailed system data in real-time.

# 2 REQUIREMENT SPECIFICATION

## 2.1 Functional Requirements

For the current control, the desired current level is adjusted within the range of 0 – 20 mA by turning the knob of the encoder.

For LCD screen, it should display the desired current level adjusted by users.

For the response time, the device should have a fast response time to quickly and accurately adjust the current level.

## 2.2 Requirements for Interfaces

As for the user interface, the device should contain the LCD display and the rotary encoder, which can be used to display and turn a knob to achieve the expected digital values, respectively.

For hardware interface, the device should be able to rotate the encoder manually to achieve the desired values displayed on the LCD display. Additionally, the device contains two current loop outputs that can be directly connected to the analog inputs of a frequency converter. The use of current loop outputs enables for precise control and communication between the device and the converter, allowing real-time data transmission for adjusting accurate parameters such as torque and speed.

For software interface, the device should allow users to control torque and speed value setpoints. Moreover, the device should have a function to detect errors when any open loops occur.

In terms of communication, the device could be programmed via serial communication through MCU's USB port. This allows the MCU to transfer data to the computer. The serial communication port is fixed, so there is no need to modify the setting.

## 2.3 Other Requirements

### 2.3.1 Performance Requirements

There are some requirements for the device's performance, as follows:

- Resolution of measurement: 0.02 mA
- Range of measurement: 0 – 20 mA

### 2.3.2 Usability Requirements

The device should have a user-friendly design, without the requirements for any specialized skills.

### 2.3.3 Safety Requirements

The device must be protected with a fuse to prevent damage from overcurrent or electrical faults.

### 2.3.4 Maintenance Requirements

The maintenance requirements include a more robust and user-friendly digital 0 – 20 mA device that can be monitored remotely and facilitates efficient trouble-shooting and repairs if needed. This approach would help users to ensure its long-term system performance and reliability, reduce the risk of unexpected outputs as well.

### 2.3.5 Portability and Compatibility Requirements

Frequency converters play a crucial role in a variety of fields in daily life such as control systems, power electronics, robotics, and automation, so the digital current loop device would be used with various types of frequency converters. Therefore, both portability and compatibility characteristics should be considered.

- For the portability factor, the designed device should be lightweight and small for ease of installation.

- For the compatibility element, the digital current loop device could be able to communicate with different types of communication interfaces when integrated with a wide range of frequency converters.

### 2.3.6 Environmental Requirements

The device must operate within the temperature range from −30°C to +30°C to ensure the desired functionality.

Furthermore, the device must operate its own environment as designed. Moreover, the device must comply with proper EMC regulations to minimize any potential interferences and noise.

## 2.4 Restrictions of Design

### 2.4.1 Restrictions caused by previously used technologies

The device must be very simple, so it can use MCU Arduino Nano A000005 board schematic as a basic of design /3/.

The digital 0 – 20 mA device requires the existing industrial systems and control devices to have the 0 – 20 mA input terminals. This is because in the real world, the 0 – 20 mA or 4 – 20 mA current loops have commonly used as the standard for signal transmission and electronic control in many industries /16/.

### 2.4.2 Hardware Limitations

The physical device should be as small as possible.

With a 0 mA current signal, it might be difficult to detect the system failures /14/.

Moreover, the wire loop distance between the transmitter and the receiver should be within 500 meters.

### 2.4.3 Software Restrictions

The software for this project would be written in the C language, so only the GCC C-compilers are permitted for developing this program.

### 2.4.4 Other Limitations

The power consumption should be minimized as much as possible while ensuring the whole system has sufficient power to operate smoothly.

In addition, ground loops might be an unpredictable problem to crash the system. This is because when many grounded devices placed in different locations are integrated with each other, which might have different voltage potentials, the noise injection generated from their ground connection might affect grounded devices. Therefore, we assumed all grounded components would be placed at the same ground point to prevent the ground loops /15/.

# 3 OVERVIEW OF FREQUENCY CONVERTERS

A frequency converter is known as a variable frequency drive or adjustable speed drive, which is used for modifying and controlling the speed and torque of electric motors. Initially, this device converts the alternating current (AC) voltage into direct current (DC) voltage. Then, it reconverts this DC back into AC with adjustable frequency and the levels of voltage. This process allows for precise control over the motor's performance, enabling it to operate at the desired torque and rotational speed required for the motor.

In addition, frequency converters can reduce motor power consumption by up to 50%, leading to significant energy savings. Moreover, they can collect motor data such as temperature and vibrations via sensors, enabling condition monitoring and preventive maintenance. This allows for remote operation monitoring and troubleshooting through cloud solutions. Therefore, these advantages make frequency converters essential devices in modern industrial applications, offering improved performance, efficiency and control across a wide range of sectors.

However, frequency converters can create different kinds of interference when assembled directly to motors or connected via cables. This includes acoustic noise, harmonic reactions on the network, and electromagnetic compatibility (EMC) noise, which can affect radio devices and data transfers. Therefore, the installation of frequency converters and motors should address and manage noise and interference by complying with EMC requirements /12/.

# 4 BACKGROUND THEORY

## 4.1 Digital Signal Processing

In the implementation of the digital current loop device, we manipulate the digital desired inputs from the rotary encoder and then convert them to analog signals for transmission to the frequency converter. To ensure accurate tracking and facilitate real-time adjustments, we have integrated an I2C LCD 1602 display in this project. This display serves as a visual interface, presenting the desired digital input signals in a clear and easily readable format, allowing for quick identification of any discrepancies or issues that may arise during operation. Therefore, this feature is useful for users to monitor the system's performance continuously when there are any changes in the wanted digital inputs from the encoder.

### 4.1.1 Rotary Encoder

In the project, the rotary encoder is an incremental encoder, which is PEC11R-4220F-S0024. This encoder converts angular motion or position of a shaft into an analog or digital code to determine position or motion /8/. This encoder creates a specified number of pulses per rotation. There are two output channels of pulses such as Channel A and Channel B, which are offset from each other to indicate the direction of rotation. The phase relationship between these channels is known as quadrature. When the encoder is rotated in a clockwise direction (CW), Channel A is leading Channel B in the output signal sequence. In contrast, during counter-clockwise (CCW) rotation, the signal sequence reverses, meaning Channel B is leading Channel A. The quadrature output table is represented as below /6/.

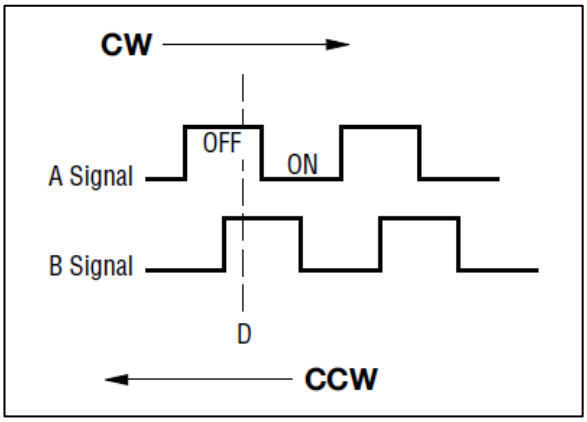


**Figure 3.** Quadrature output table.

Based on the quadrature output table, we could derive a encoder logic state table, which is illustrated as Table 1.

**Table 1.** Encoder logic state.

| Clockwise Direction | | |
|---|---|---|
| **Position** | **Channel A** | **Channel B** |
| 1 | 0 | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 1 |
| 4 | 0 | 1 |

In addition, this encoder is integrated with an additional feature, which is a push switch option. The push button is often used to trigger of specific actions or toggle between different operational modes.

### 4.1.2 I2C LCD1602

An I2C LCD1602 is a display device that combines a standard LCD (Liquid Crystal Display) and an I2C (Inter-Integrated Circuit) module attached to its backside. This device could show text and characters on 2 rows with 16 columns. One of the key advantages of the I2C LCD1602 lies in its simplified connection interface. Unlike traditional LCDs that often require multiple pins for operation, this device utilizes only four pins: GND (ground), VCC (power supply), SDA (Serial Data), and SCL (Serial Clock). This configuration simplifies the wiring process /5/.

## 4.2 Digital-Analog Converter

The Digital-Analog Converter (DAC) is a device that converts a digital input signal into an analog output signal. Specifically, the DAC transforms the binary signal in the form of bits into an analog voltage or current. The DACs find widespread applications in various digital signal processing fields, including audio amplifier, video encoder, display electronics, data acquisition systems and motor control /7/.

In this thesis, we utilized two DFRobot Gravity 12-Bit I2C DAC modules equipped with EEPROM (Electrically Erasable Programmable Read-Only Memory) to facilitate the conversion of digital input signals from the encoder into voltage or current signals suitable for the frequency converter. These DAC modules incorporate a 12-bit DAC MCP4725 chip, support a wide input voltage range of 3.3 – 5 V and include an I2C address selection switch. The integration of EEPROM in these DAC modules significantly enhances data retention and system reliability. This non-volatile memory component could retain the DAC input values while the power is turned-off and resume the DAC output upon power resumption /7/.

Moreover, the input voltage for DAC modules was chosen at 5 V. Instead of using the internal voltage of Arduino Nano, we chose for an external voltage circuit, aiming for more stable and precise values for the whole system. Therefore, we used the reference chip of TL431, namely Adjustable precision Zener shunt regulator, to generate a constant output voltage at 5 V, a reference voltage for both DACs and Arduino Nano. The circuit of the external voltage was designed in LTspice, as shown in Figure 4.

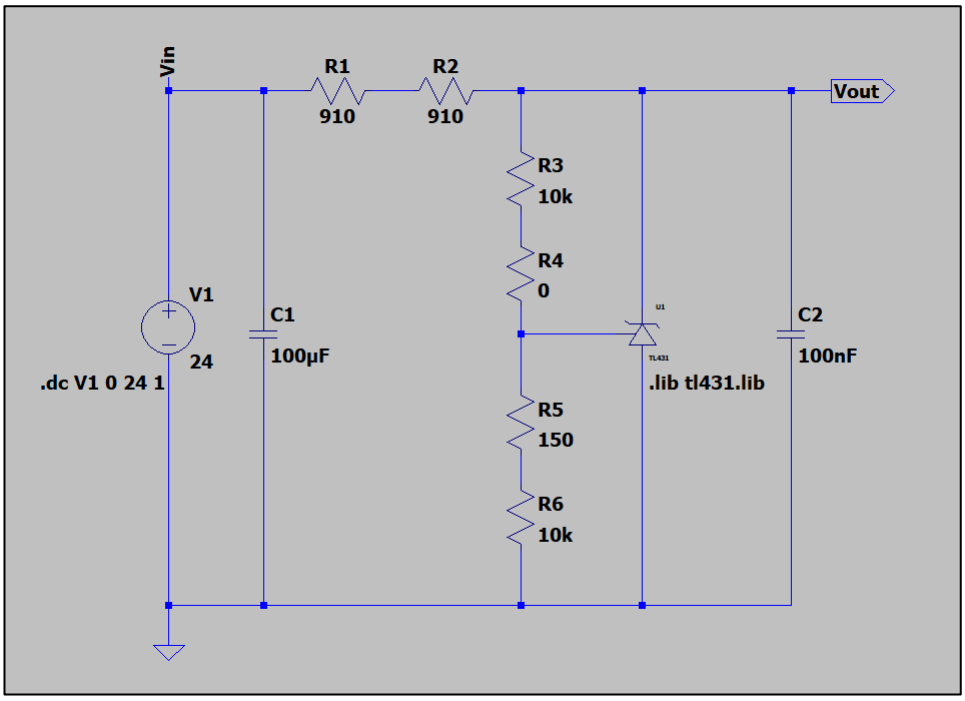


**Figure 4.** The external voltage circuit drawn in LTspice.

The circuit was then simulated as shown in the picture below. The results indicated that the input voltage of 24 V, serving as the voltage source, was successfully converted to the constant output voltage of 5 V.

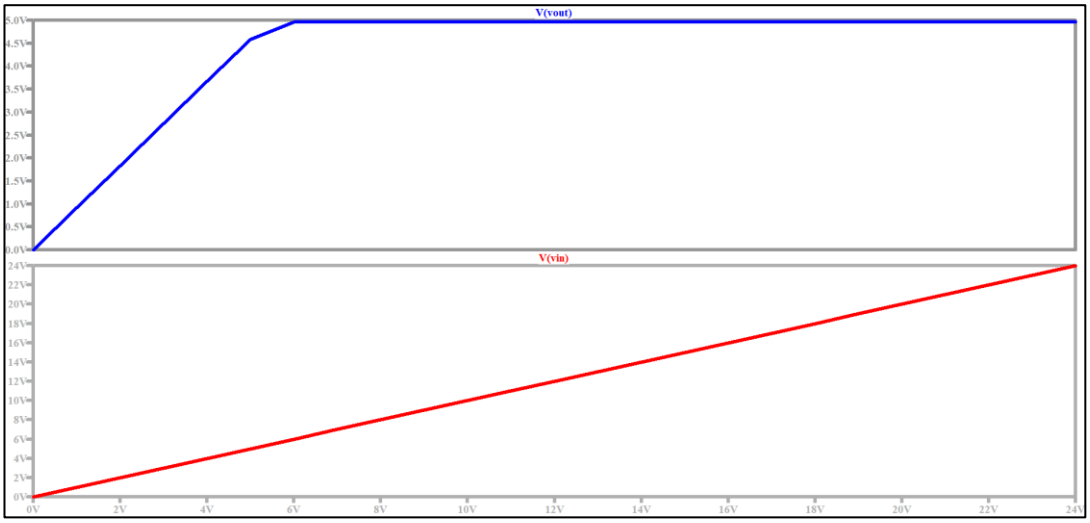


**Figure 5.** Simulation of the external voltage from the TL431 chip.

## 4.3 Current Loop

### 4.3.1 Transmitter Circuit

After converting the digital input signals to the analog signals, we connected the analog output to a transmitter circuit for converting the input voltage into an output current. The transmitter circuit incorporates two key components: an operational amplifier (LM324) and a transistor (BC547).

The LM324 operational amplifier was utilized for maintaining signal integrity and ensuring that even small voltage changes are accurately represented in the output current. The LM324 series feature cost-effective quad operational amplifiers with true differential inputs, designed for single supply operation within a 3.0 – 32 V range. The amplifier offers low quiescent currents and numerous advantages such as short-circuit protection, true differential input stages. Moreover, the LM324 series also provides low input bias currents, internal compensation, and a common mode range extending to the negative supply. With industry-standard pinouts and ESD (Electrostatic Discharge) protection, these versatile amplifiers suit various applications requiring reliable operation /13/.

Following the amplifying stage, we used the BC547 Bipolar Junction Transistor (BJT), which is used for amplifying current or switching. The transistor has three regions: emitter, base, and collector. By controlling the base current, the transistor can be turned on or off, allowing a larger current to flow between the collector and emitter. This property makes it useful for amplifying small signals or acting as a switch. The BC547 is popular due to its affordability, availability, and ease of use /10/. Hence, this BC547 was used to generate the 0 – 20 mA output current in our device.

The simplified transmitter circuit applied in this project in Figure 6.

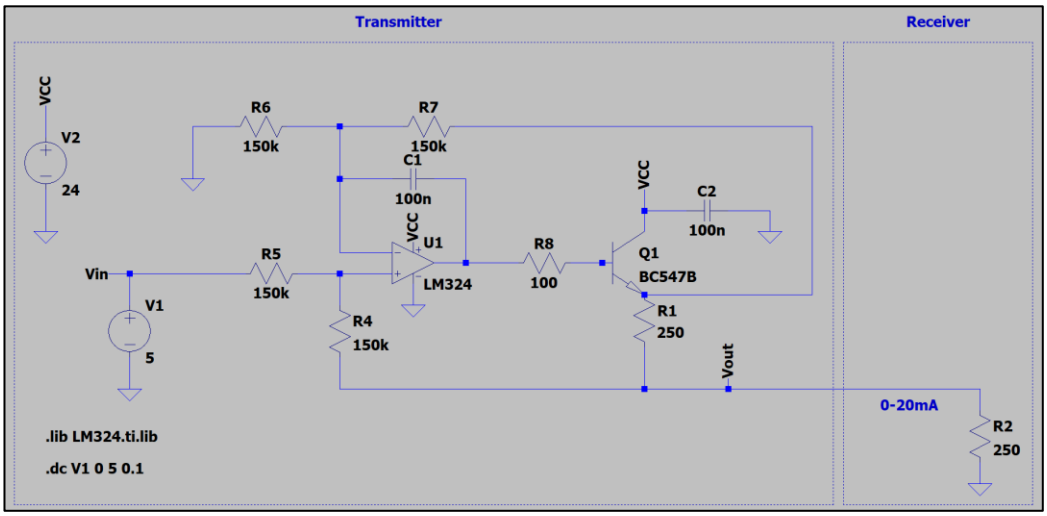


**Figure 6.** The simplified transmitter circuit drawn in LTspice.

In addition, we also simulated the transmitter circuit above to measure the output current of transmitter and the output voltage of receiver. After simulating, within the input voltage range of 0 – 5 V, the transmitter output current was from 0 mA to 20 mA and the receiver output voltage was from 0 V to 5 V. These results were shown in Figure 7 below.

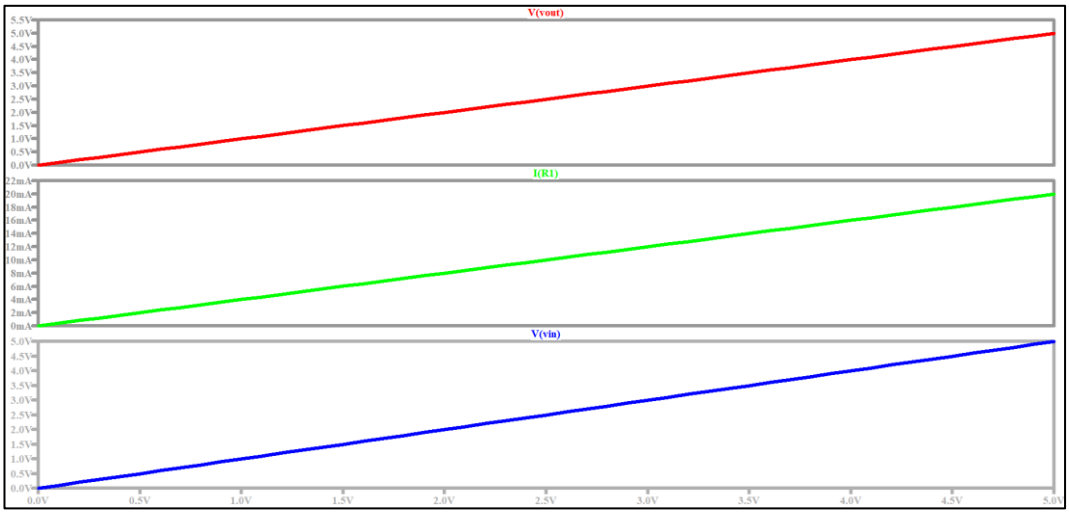


**Figure 7**. Simulation of the simplified transmitter circuit.

### 4.3.2 Feedback Loop Circuit

To complete the current loop, we implemented a feedback loop circuit designed to capture and process the transmitted current signal. This circuit performs the essential function of converting the received current back into its original voltage form, enabling further processing or utilization of the signal. This circuit consisted

of two main parts such as a resistor of 250 Ohm simulating frequency converter or receivers and the feedback loop. To be specific, the feedback loop incorporated three main parts: the low pass filter cut-off frequency, two diodes of 1N4148 for ESD protection and a resistor of 250 Ohm.

Firstly, the low pass filter part consisted of a resistor of 10 kOhm, and a capacitor of 100 nF for filtering out frequencies above the cut-off point. The cut-off frequency was calculated following the below formula:

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi * 10k\Omega * 100nF} = 159.1\ Hz\ \approx 0.16\ kHz$$

This calculated cut-off frequency of approximately 0.16 kHz ensured that the circuit effectively attenuates signals above this threshold, resulting in a more stable output signal.

Secondly, these diodes of 1N4148 acted as a safeguard, protecting the sensitive components of the circuit from potentially damaging voltage spikes that might occur due to static electricity or other sources.

Lastly, the receiver circuit included the resistor of 250 Ohm to control and maintain the targeted current output signals, preserving the 0 – 20 mA current flow transmitted from the transmitter circuit and the receiver.

Here is the simulating receiver (the resistor of 250 Ohm) and feedback loop circuit shown as below.

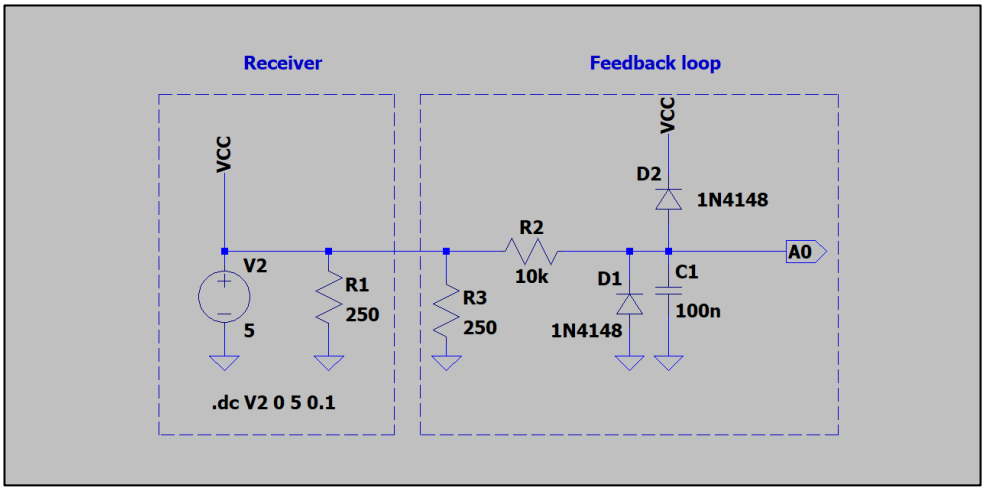


**Figure 8.** The simulating receiver and feedback loop circuits drawn in LTspice.

The simulation results from this simulating receiver and feedback loop circuits were shown as Figure 9 below. The figure illustrated that the current value going through R1 represented as the receiver or the frequency converter was from 0 – 20 mA. In addition, the 0 – 20 mA current flow was transmitted to the feedback loop. Then, this current flow was converted into the output voltage of 5 V.

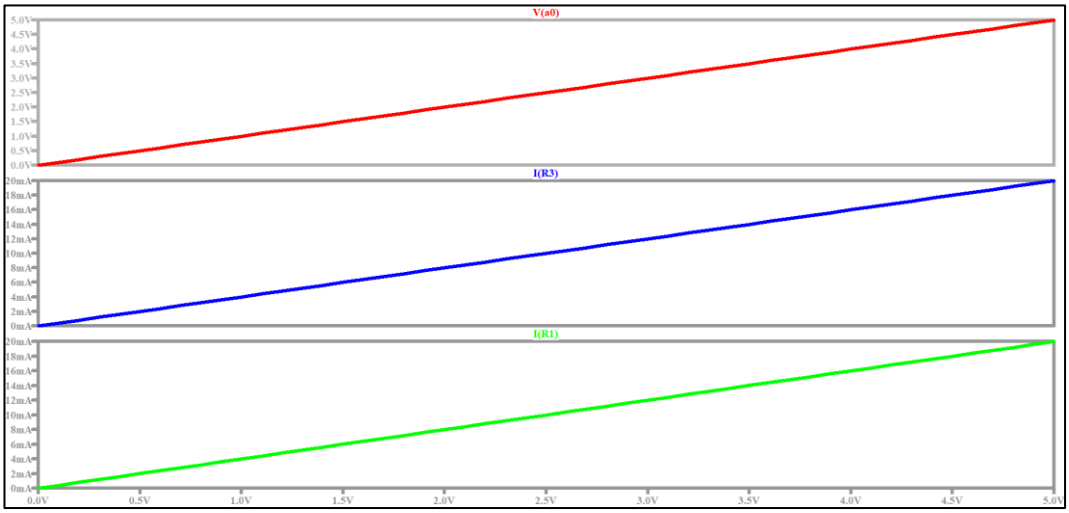


**Figure 9.** Simulation of the simulating receiver and feedback loop circuits.

# 5 SCHEMATICS AND IMPLEMENTATION

The project's schematics were divided into two main parts, including a main board schematic and a digital signal processing schematic. The digital signal processing circuit included the encoder and LCD circuits. The remaining circuits including a filter, external reference voltage, DACs, transmitter and feedback loop circuits connected to Arduino Nano belonged to the main board schematic.

## 5.1 Schematics

The filter circuit's schematic was designed as below:

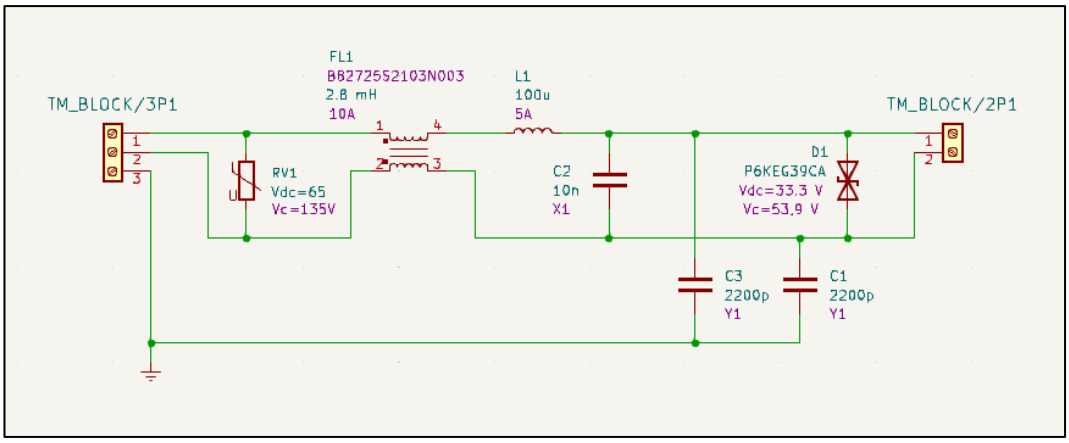


**Figure 10.** The filter circuit's schematic.

The main board circuit's schematic was designed as below:

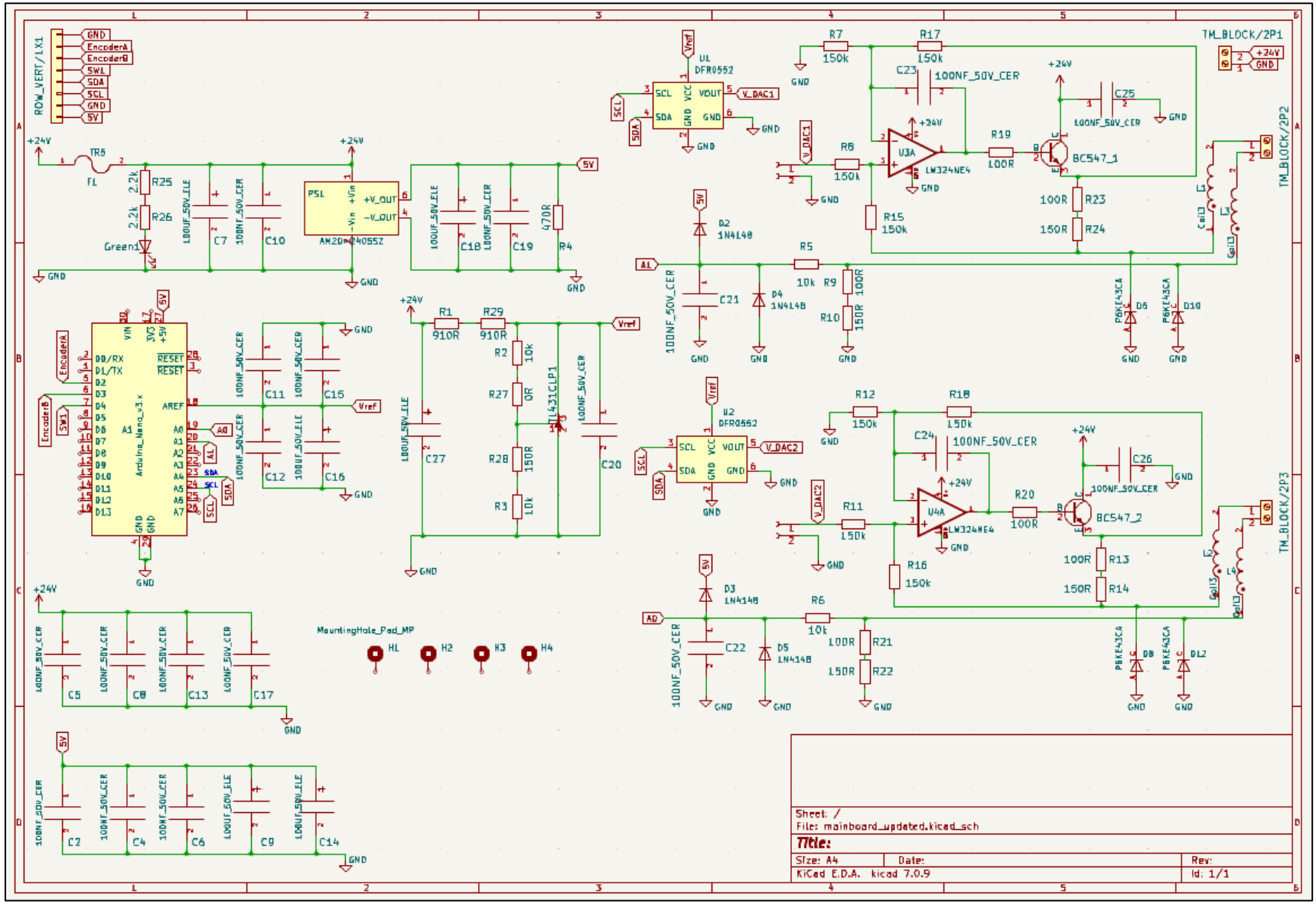


**Figure 11.** The main board circuit's schematic.

The encoderLCD circuit's schematic was designed as below:

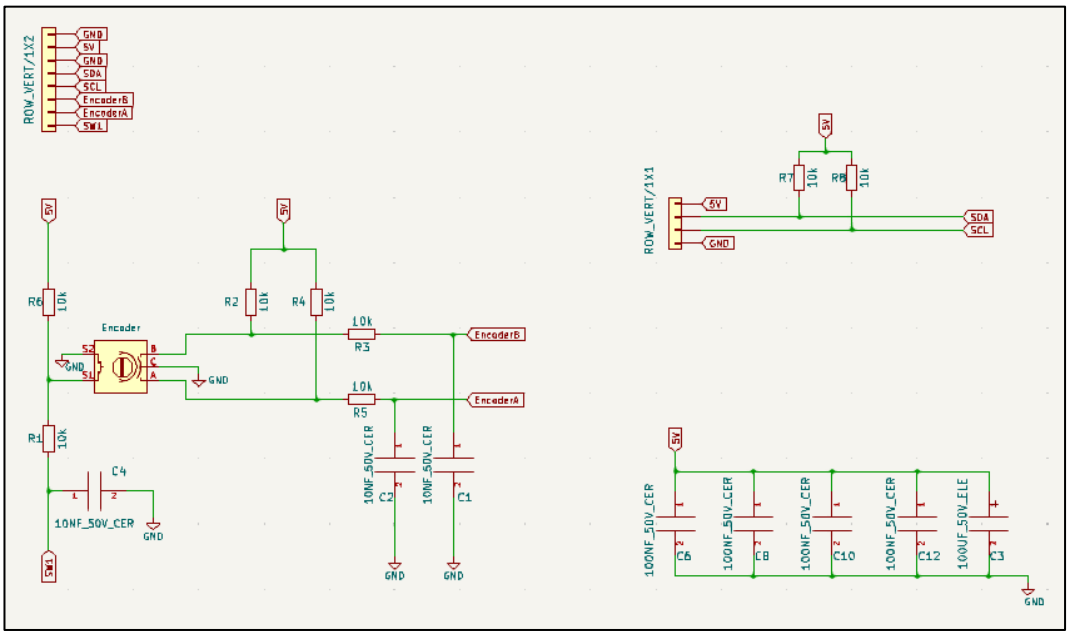


**Figure 12**. The encoderLCD circuit's schematic.

## 5.2 Software Implementation

This project built up the device with two channels of 0 – 20 mA current loop using Arduino software program. The implementation involved setting up and controlling a system with the LCD display, two DACs, the rotary encoder including two main external interrupts, feedback loops and ADC. Additionally, a watchdog timer is implemented to reset the system if it becomes unresponsive. Initially, we needed to install the appropriate libraries used in this project.

```
#include <Wire.h>
#include "DFRobot_MCP4725.h"
#include <LiquidCrystal_I2C.h>
#include <avr/wdt.h> //Watchdog Timer library
```

**Figure 13.** The appropriate libraries.

To address inaccuracies in the DAC's reference voltage, we set a calibration value to achieve a more precise output voltage. Our target was 5 V for each DAC when achieving the current value at 20 mA, resulting in a calibration value of 5000 millivolts:

**Calibration of DAC = 5*1000 = 5000**

In addition, the calibration ensures precise conversion between analog and digital values throughout the system. Due to Arduino Nano offers a 10-bit resolution ADC for analog pins, the calibration of ADC could be calculated as below.

**Calibration of ADC =** $2^{10} - 1 = 1023$

These calibrations were defined in the software as follows:

```
#define calibration_value_for_ADC_ch0 1023
#define calibration_value_for_ADC_ch1 1023
#define calibration_value_for_DAC_ch0 5000
#define calibration_value_for_DAC_ch1 5000
```

**Figure 14.** The calibrations for ADC and DAC for both channels.

In this program, every change of the torque and speed values by using the encoder are updated through the function of **void write_dac(float torgue, int rpm)**. In other words, this function calculates new DAC values when there are any changes occuring, then updates the appropriate channels.

```
void write_dac(float torgue, int rpm)
{
  static int rpm_old=0;
  static float torgue_old=0;

  if(rpm!=rpm_old) //if different than old value
  {
    //for example calibrated value 4087 and wanted 3000 rpm so 4087/3000*3000=4087
    int dac_rpm=(float)calibration_value_for_DAC_ch1/maximum_value_for_rpm*rpm;
    //here real DAC write
    DAC2.outputVoltage(dac_rpm);
  }

  if(torgue!=torgue_old) //if different than old value
  {
    //for example calibrated value 4081 and wanted 50.00 torgue so 4081/50*50=4081
    int dac_torgue=(float)calibration_value_for_DAC_ch0/maximum_value_for_torgue*torgue;
    //here real DAC write
    DAC1.outputVoltage(dac_torgue);
  }
  torgue_old=torgue;
  rpm_old=rpm;
}
```

**Figure 15.** Updating the torque and speed changes through the function of void write_dac(float torque, int rpm).

The encoder has two functions such as push-button and changing encoder states when controlling the desired input values. In this project, the **void read_push_button_state(void)** function manages the state of a push-button

from the encoder. It uses a static variable 'timex' for 200 ms debouncing to avoid multiple triggers from one press. When the button is pressed (indicated by digitalRead(SW_PIN) = = 0), and the debounce time has passed, it toggles the DAC_select variable between true and false, switching between DAC1 and DAC2. The timex variable is then updated to the current time using millis() for accurate debouncing in subsequent presses.

```
void read_push_button_state(void) //if push-button pressed on encoder
{
  static long timex=0;

  if(digitalRead(SW_PIN)==0&&millis()>timex+200)
  {
      DAC_select=!DAC_select;
      timex=millis();
  }
}
```

**Figure 16.** The state of a push-button.

An external interrup is the **void ISR_encoderChange(void)** function, which is responsible for processing the changes of the encoder's state to modify the torque and rpm values. Here is a part of this function shown in Figure 17. In this function, we used a lookup table of the encoder state to determine the rotation direction such as a clokcwise direction (CW) or a counterclockwise direction (CCW).

```
void ISR_encoderChange(void)
{
  static unsigned long _lastIncReadTime = micros();
  static unsigned long _lastDecReadTime = micros();
  int _pauseLength = 25000;

  // Encoder interrupt routine for both pins. Updates counter
  // if they are valid and have rotated a full indent

  static uint8_t old_AB = 3;  // Lookup table index
  static int8_t encval = 0;   // Encoder value
  static const int8_t enc_states[]  = {0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0}; // Lookup table

  old_AB <<=2;  // Remember previous state

  if (digitalRead(CLK_PIN)) old_AB |= 0x02; // Add current state of pin A
  if (digitalRead(DT_PIN)) old_AB |= 0x01; // Add current state of pin B

  encval += enc_states[( old_AB & 0x0f )];

  // Update counter if encoder has rotated a full indent, that is at least 4 steps
```

**Figure 17.** The state of the encoder changes as an external interrupt.

Due to the encoder has two outputs, generating four possible states: 00, 01, 10 and 11, the lookup table could be represented as below:

**Table 2.** Lookup table of the encoder changes.

| Previous State | Current State | Index (Binary) | Index (Decimal) | Change |
|---|---|---|---|---|
| 00 | 00 | 0000 | 0 | 0 |
| 00 | 01 | 0001 | 1 | -1 |
| 00 | 10 | 0010 | 2 | 1 |
| 00 | 11 | 0011 | 3 | 0 |
| 01 | 00 | 0100 | 4 | 1 |
| 01 | 01 | 0101 | 5 | 0 |
| 01 | 10 | 0110 | 6 | 0 |
| 01 | 11 | 0111 | 7 | -1 |
| 10 | 00 | 1000 | 8 | -1 |
| 10 | 01 | 1001 | 9 | 0 |
| 10 | 10 | 1010 | 10 | 0 |
| 10 | 11 | 1011 | 11 | 1 |
| 11 | 00 | 1100 | 12 | 0 |
| 11 | 01 | 1101 | 13 | 1 |
| 11 | 10 | 1110 | 14 | -1 |
| 11 | 11 | 1111 | 15 | 0 |

Furthermore, this function implementes a feature of fast or slow increments or decrements based on the time elapsed since the last change, ensuring smooth and responsive adjustments. The Figure 18 illustrates an example of the feature of fast or slow increments for four step forward.

```
if( encval > 3 )
{       // Four steps forward

  if((micros() - _lastIncReadTime) < _pauseLength)
  {
    if(DAC_select==1&&rpm<=(maximum_value_for_rpm-step_for_rpm))
      rpm=rpm+step_for_rpm;
    else if(DAC_select==1&&rpm>(maximum_value_for_rpm-step_for_rpm))
      rpm=maximum_value_for_rpm;

    if(DAC_select==0&&torgue<=(maximum_value_for_torgue-step_for_torgue))
      torgue=torgue+step_for_torgue;
    else if(DAC_select==0&&torgue>(maximum_value_for_torgue-step_for_torgue))
      torgue=maximum_value_for_torgue;
  }
  else
  {
    if(DAC_select==1&&rpm<=(maximum_value_for_rpm-minimum_step_for_rpm))
      rpm=rpm+minimum_step_for_rpm;

    if(DAC_select==0&&torgue<=(maximum_value_for_torgue-minimum_step_for_torgue))
      torgue=torgue+minimum_step_for_torgue;
  }
  _lastIncReadTime = micros();

  encval = 0;
}
```

**Figure 18.** Fast or slow increments featuring for four steps forward.

Additionally, the software program is also included safety checks to keep the torque and rpm values within their defined limits.

```
if(torgue>=maximum_value_for_torgue) //for safety
  torgue=maximum_value_for_torgue;
else if(torgue<0)
  torgue=0;

if(rpm>=maximum_value_for_rpm) //for safety
  rpm=maximum_value_for_rpm;
else if(rpm<0)
  rpm=0;
```

**Figure 19.** Safety checks within defined limits.

Besides, we also added one function for detecting possible errors in a control system by comparing the desired torque and rpm values with their corresponding ADC feedbacks, which was **int open_loop_error_check(void)** as follows. For torque values, it flags an error if the desired value exceeds approximately 100 while the feedback is below 20. A similar check is performed for rpm. The function returns a combined error code: 1 for a torque channel error, 2 for an rpm channel error, or 3 if both channels have errors. This error checking mechanism helps identify potential issues in the open loop control system, such as disconnections.

```
int open_loop_error_check(void)
{
  //calculates feedback value from ADC (torgue and rpm) and scales to 0-4095
  int feedback_adc_value_for_torgue=(float)calibration_value_for_DAC_ch0/calibration_value_for_ADC_ch0*analogRead(A0);
  int feedback_adc_value_for_rpm=(float)calibration_value_for_DAC_ch1/calibration_value_for_ADC_ch1*analogRead(A1);
  int error_ch0=0;
  int error_ch1=0;

  //calculates comparable DAC value from wanted torgue and scales to 0-4095
  int dac_value_for_torgue=(float)calibration_value_for_DAC_ch0/maximum_value_for_torgue*torgue;

  //calculates comparable DAC value from wanted rpm and scales to 0-4095
  int dac_value_for_rpm=(float)calibration_value_for_DAC_ch1/maximum_value_for_rpm*rpm;

  //if wanted torgue > 100/4095*5V=0,122V and feeback less than 20/4095*5V=0,024V gives alarm
  if(dac_value_for_torgue>100&&feedback_adc_value_for_torgue<20)
    error_ch0=1; //maybe open loop error in CH0

  //if wanted rpm > 100/4095*5V=0,122V and feeback less than 20/4095*5V=0,024V gives alarm
  if(dac_value_for_rpm>100&&feedback_adc_value_for_rpm<20)
    error_ch1=2; //open loop error in CH1

  /*
  error = 1 torgue channel error
  error = 2 rpm channel error
  eroor = 3 torgue and rpm channel error
  */

  return (error_ch0+error_ch1);
}
```

**Figure 20.** Open loop error check featuring.

### 5.3 I2C Communication Protocol

For data communication and transfer in this project, we leveraged the I2C (Inter-Integrated Circuit) protocol. This choice was particularly apt as key components of our system - namely the DAC DFR0552, I2C LCD, and Arduino Nano - all support I2C addressing, allowing for integration.

The I2C protocol is a serial communication protocol that uses two wires: SDA (Serial Data) and SCL (Serial Clock), simplifying the wiring and reducing the number of pins required. This protocol is used for communication between a master or controller (or multiple masters) and a single or multiple slave devices. The SDA line carries the actual data being transmitted, while the SCL line provides the clock signal that synchronizes the data transfer. Here is an example of I2C network in Figure 21 /17/.

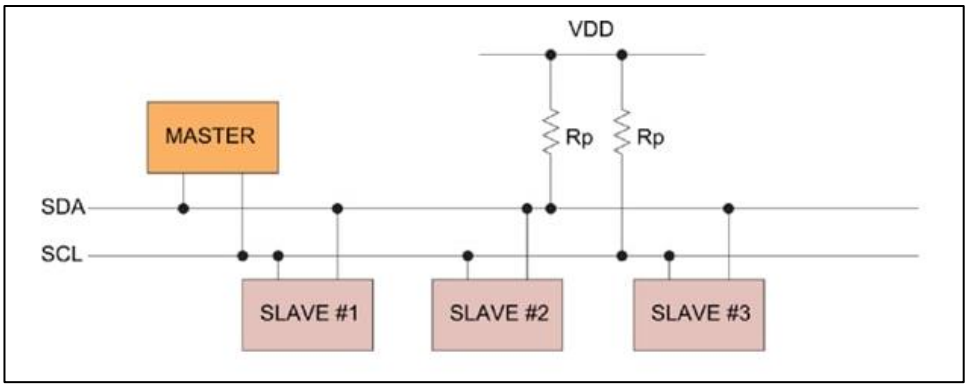


**Figure 21.** I2C network.

# 6 TESTING AND TROUBLESHOOTING THE PROTOTYPE ON BREADBOARDS

As our project evolved, we continually refined and enhanced our software to address increasingly sophisticated requirements dictated by the evolving prototype design. In the initial stages of development, our primary focus was on creating software capable of generating a precise output current of 20 mA and a voltage of 5 V. These parameters formed the foundation of our early prototyping efforts and served as the baseline for our initial testing phase. Consequently, when we conducted our preliminary evaluations using breadboards, we utilized this initial version of the software to power and control the prototype.

To ensure the reliability and accuracy of our design, we embarked on a thorough testing phase. This involved constructing and evaluating the critical components of both the main board and the encoderLCD circuits. We utilized breadboards for this purpose, as they provided an ideal platform for prototype testing. Our main goal was to verify if the actual output results matched our expected outcomes and simulation results in LTspice. We also aimed to ensure these results were accurately displayed on the LCD interface, which was crucial for user interaction and system monitoring. Here are the main components of the prototype tested on breadboards.

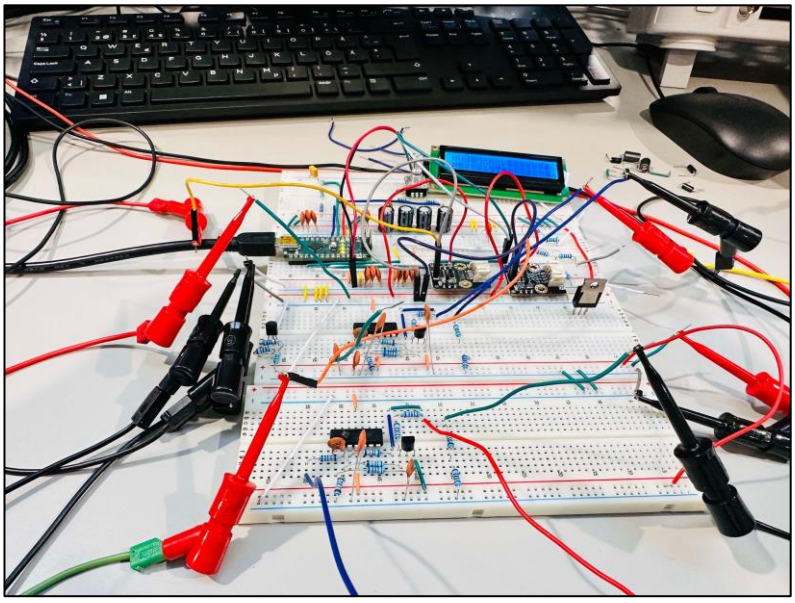

**Figure 22.** The main components of the prototype testing on breadboards.

During the testing phase, we encountered several significant issues that required careful attention and resolution. One of the primary challenges we faced was related to ground connections. We discovered it was crucial to connect all ground points together to effectively eliminate ground noise and achieve more accurate outputs. This realization came after a costly learning experience in our initial testing rounds. After three DACs were broken or burned, we found that interference from ground points made the input voltages of the DACs abnormal. This occurred because the output grounds of the DACs were not connected to other ground points, and the grounds of each breadboard were not linked together. Consequently, the DACs' output voltages were out of the operational range required in the datasheet. As a result of this incident, we properly connected all ground points in the prototype to create a stable reference point for our circuits.

After addressing this issue, the final output values were still higher than our calculated values, and another DAC was broken. We found that noise generated within the breadboards was the main cause of this problem. When we tested the prototype with the reference voltage created by the TL431 chip, the output values were much higher than expected, contradicting the simulation results. However, by using a 5 V external power supply as the reference voltage, the output results were closer to the expected values.

Consequently, after conducting thorough testing and troubleshooting procedures as above, we recognized the need for a more stable and reliable platform for our prototype. This prompted us to initiate the process of designing PCB layouts.

# 7 DESIGN AND MANUFACTURE OF PRINTED CIRCUIT BOARDS

Our transition to PCBs was driven by the need for a more stable, noise-resistant platform than the breadboards we had been using. This crucial step in our development process ensured the reliability of the entire system design.

In this prototype, we designed and manufactured three distinct printed circuit boards (PCBs) based on the three comprehensive schematics mentioned earlier. This approach optimized the layout and functionality of each circuit, maximizing efficiency and performance in a finished product.

## 7.1 Design PCB layouts

### 7.1.1 The Filter's PCB

The filter's PCB was designed as below:

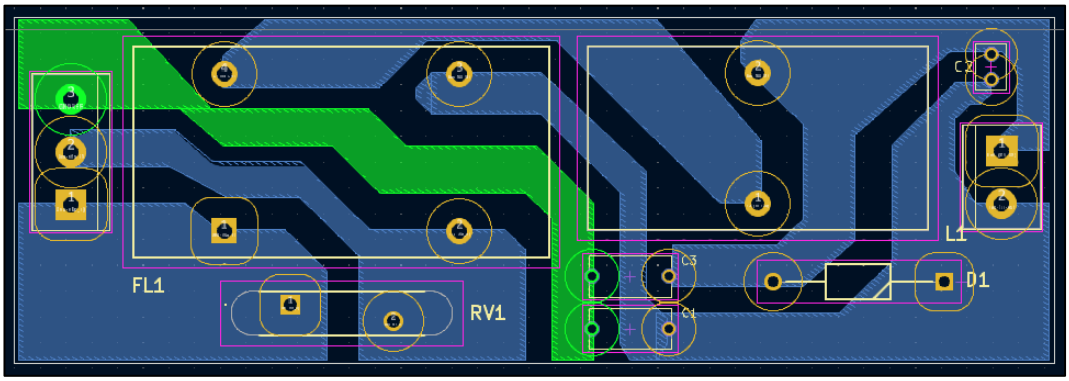


**Figure 23.** The filter's PCB.

### 7.1.2 The Main Board's PCB

We designed a PCB for the main board part in two layers by using both SMD and through hold components.

A critical aspect of our design process involved thorough thermal analysis. To mitigate the risk of component overheating, we conducted detailed power consumption calculations for resistors. Our benchmark for thermal safety was set at a conservative one-fifth of the maximum rated power consumption for some resistors. In this device, through-hole resistors have 600 mW and SMD resistors have 250

mW according to the power consumption indicated by the manufacturers. Hence, we could calculate the maximum rated power consumption for through-hole and SMD resistors:

The maximum rated power consumption for through-hole resistor:

$$P_1 = \frac{600}{5} = 120\ \text{mW}$$

The maximum rated power consumption for SMD resistor:

$$P_2 = \frac{250}{5} = 50\ \text{mW}$$

If the real power consumption is smaller or equal to $P_1$ and bigger than $P_2$, the resistor should be through-hole.

Here is the PCB layout of the main board.

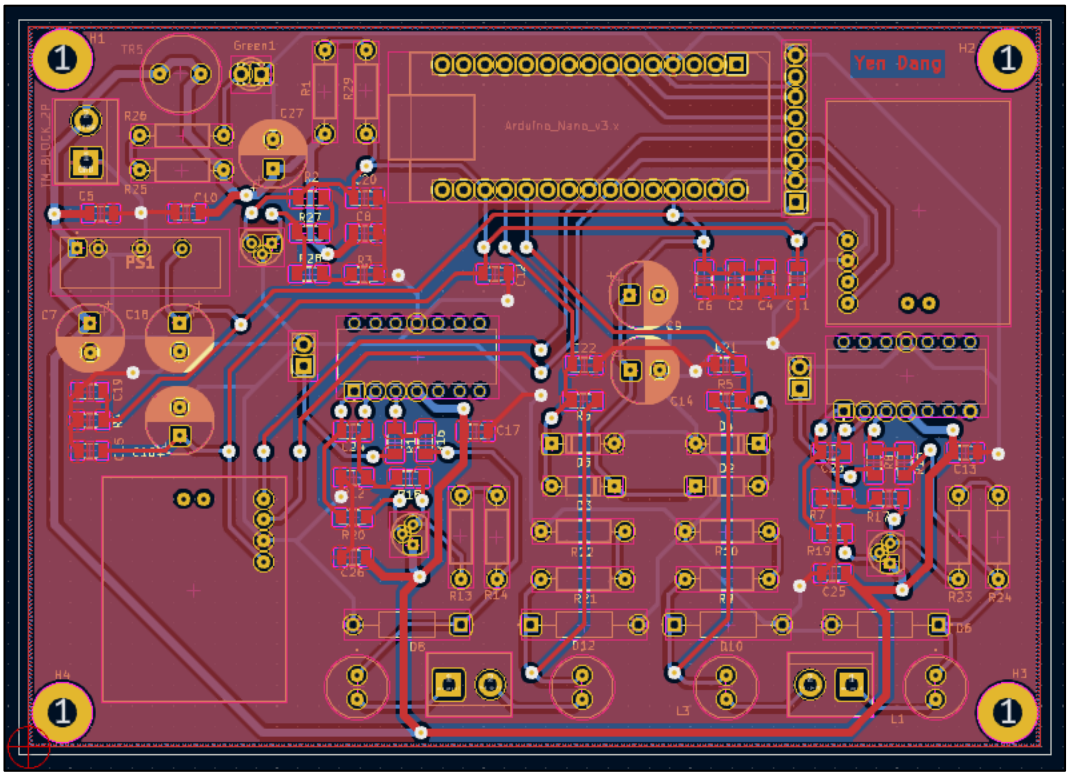


**Figure 24.** The mainboard's PCB.

### 7.1.3 The EncoderLCD's PCB

The PCB of the encoderLCD circuit was designed in one layer.

Here is the PCB layout of this part.

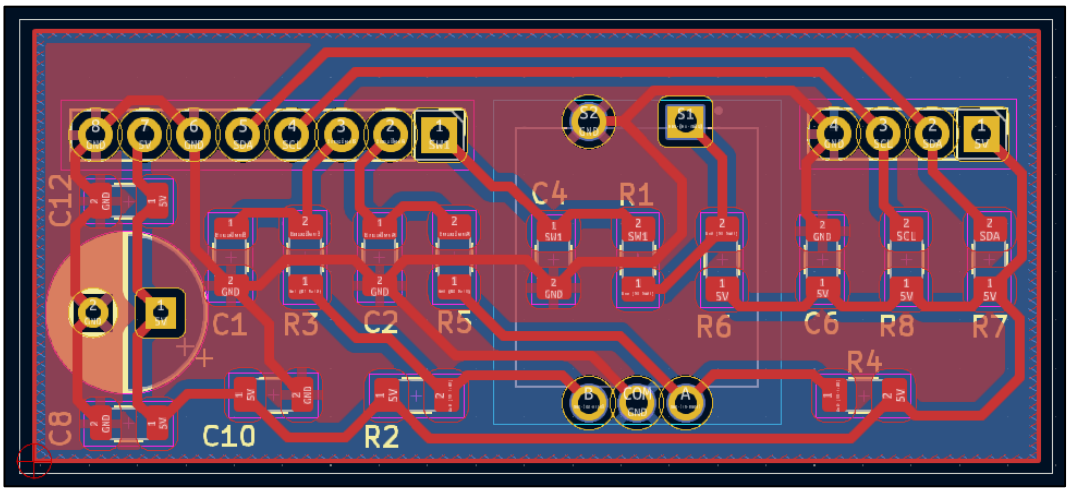


**Figure 25.** The encoderLCD's PCB.

### 7.2 PCBs Fabrication and Assembly

Our next step involved the manual fabrication of PCBs for the main board, encoderLCD and filter circuits. Following this, we assembled and soldered the electronic components onto these PCBs. The resulting physical PCBs are illustrated in Figure 26.

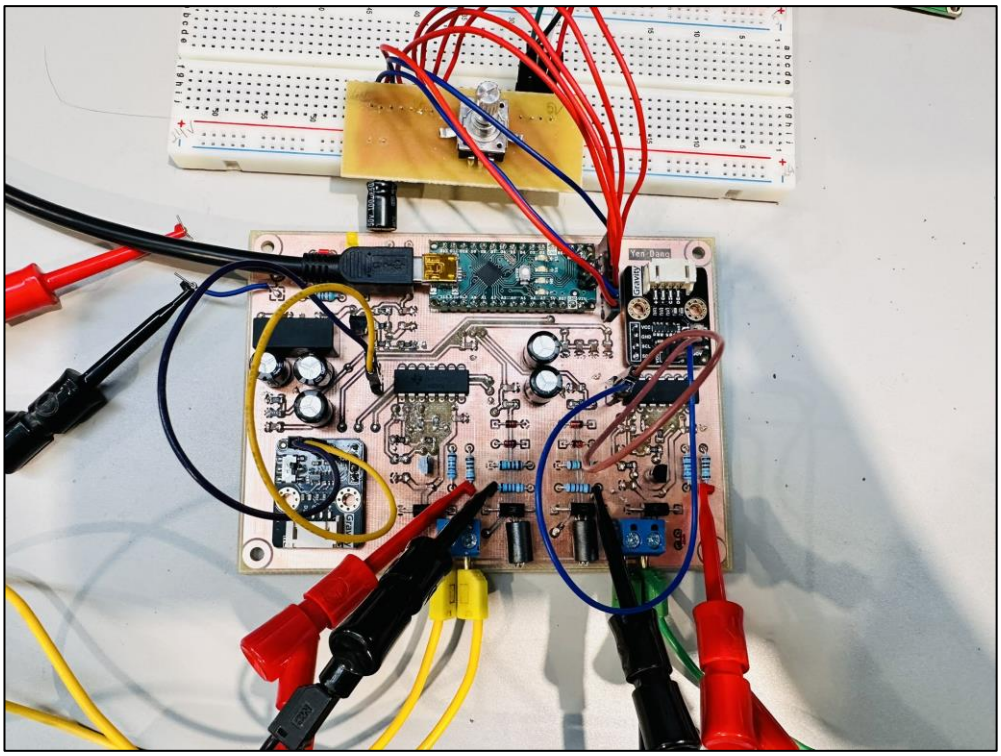

**Figure 26**. The assembled physical PCBs.

Upon testing these PCBs, we observed a significant improvement in the performance of our prototype. The output results demonstrated a marked increase in precision and accuracy, a clear indication of the benefits of moving from breadboards to PCBs. Especially, the values we obtained aligned much more closely with our calculated theoretical values, providing a strong validation of our design approach. This confirmed the validity of our previous testing and troubleshooting stages.

Furthermore, the wanted input values displayed on the LCD interface showed a high degree of consistency with the actual output results. This correlation between the displayed and measured values was particularly encouraging, as it confirmed the effectiveness of our user interface and the overall system integration. The improved accuracy and reliability achieved through the use of PCBs represented a significant milestone in our project's development, bringing us closer to a finalized, production-ready design. In fact, we placed an order of physical PCBs for ten devices from a professional manufacturer. This step aimed to validate our design's reproducibility and enhance the overall reliability and robustness of our project.

Then, we assembled these PCBs into two metal boxes to make a finished device as shown below:

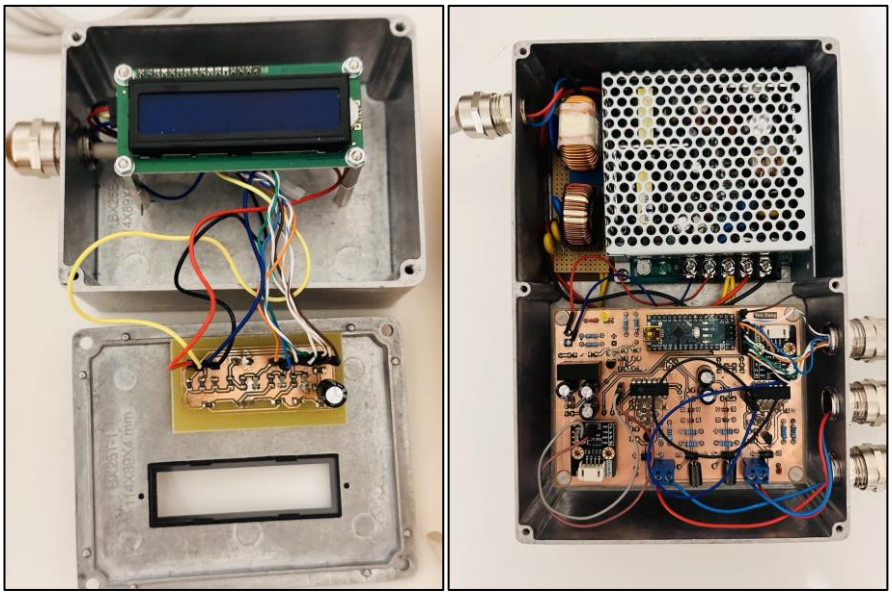

**Figure 27.** The finished current loop device.

However, during the development of our finished device, we faced a significant challenge related to signal integrity, specifically a crosstalk issue arising from the cable connection between two metal boxes. We used a CAT6 cable to connect the main board and encoder-LCD circuits, which included four pairs of wires. Initially, we randomly connected wires to signal pins. Consequently, without adjusting the encoder, some values on the LCD fluctuated randomly, indicating interference and noise. To solve this issue, we carefully separated different signals across the available wire pairs, ensuring that related signals were grouped together within the twisted pairs. In this device, we assigned dedicated pairs for critical signal combinations such as SDA and SCL for I2C communication, two ground (GND) connections for proper referencing, encoderA and encoderB for rotary encoder functionality, and a pair for the switch signal and 5 V power supply. This approach helped to minimize the impact of capacitive coupling between different signal types, thereby preserving the integrity of our data transmission. As a result, the finished device worked properly.


**Figure 28**. Testing results of the finished current loop device.

We observed that when the desired torque and speed values were set to 50 Nm and 3000 rpm respectively, the corresponding current and voltage values measured approximately 20 mA and 5 V for both channels. These measurements aligned with the simulation results of the receiver circuit.

# 8 TESTING AND COMPRATIVE ANALYSIS

This chapter describes the output results of ten devices featuring professionally manufactured PCBs, and compares them to our main prototype.

## 8.1 Testing Finished Current Loop Device with a Real Motor

After the finished current loop device generated stable and accuracy output values, we started testing it with a real motor by connecting it via a frequency converter. The setup for testing the finished device with the frequency converter is shown in Figure 29.

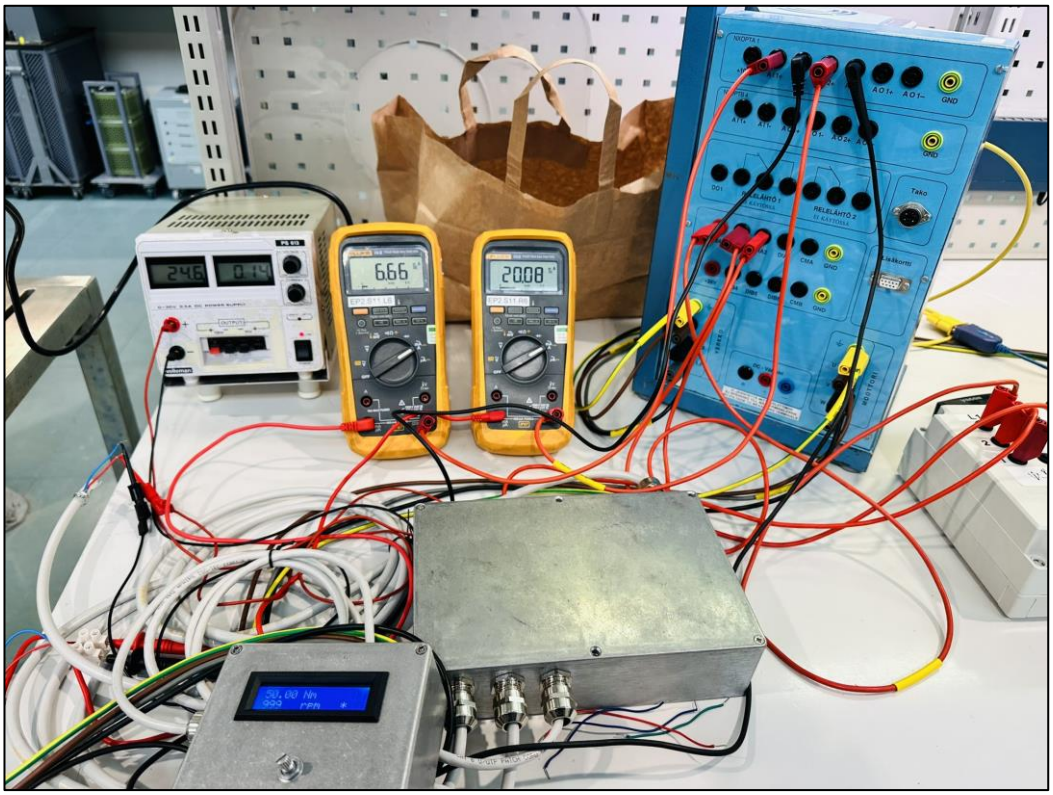


**Figure 29.** Setup for testing the finished device with the frequency converter.

The real-time outputs were also visible in a NCDrive software, as shown in Figure 30 below.

**Figure 30.** Real-time outputs displayed in the NCDrive software.

Despite the initial success, we faced challenges with the output stability of the prototype due to frequency interference. This led to errors or abnormal characters on the LCD display. Upon investigation, we identified the root causes: unshielded CAT6 cable and EMC conductors lacking proper grounding. These issues allowed electromagnetic interference to affect the signal integrity, compromising the reliability of our output results. To eliminate the frequency interference, we implemented EMI shielding tapes around the CAT6 cable and connected all EMC conductors of both metal boxes to the ground. As a result, the device operated with significantly improved stability.


**Figure 31**. EMI shielding tapes applied to the CAT6 cable and EMC conductors grounded.

Moreover, the current and voltage diagrams were measured in an oscilloscope as shown in Figure 32 below.

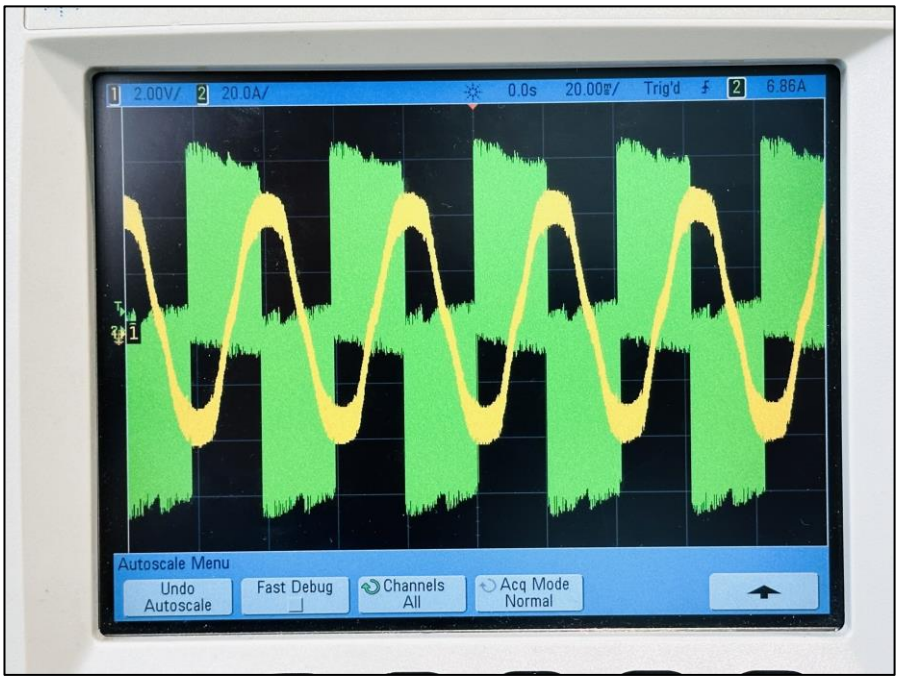


**Figure 32.** Current and voltage diagram in the Oscilloscope.

## 8.2 Comparative Analysis of Output Results of Ten Devices and Main Prototype

In this project, we used Fluke 179 multimeters to measure output values such as current. According to the Fluke 179 datasheet (page 11), the accuracy is calculated using the following formula:

**Accuracy = ± ([% of Reading] + [Counts]) = ± (1.0% + 3)**

Using this formula, we calculated the total possible measurement error of 0.26 mA and then converted to an actual current values range of 19.74 mA to 20.26 mA as shown in Table 3. These calculations provided a framework for interpreting our results, accounting for equipment limitations when assessing the accuracy and reliability of the digital current loop device.

**Table 3.** Error measurement range for actual current values.

| Description | Current value | Unit |
|---|---|---|
| Range of currents | 20.00 | mA |
| Percentage error | 0.20 | mA |
| Count error | 0.06 | mA |
| **Total possible error** | **0.26** | **mA** |
| **Actual current values** | **19.74** | **mA** |
| | **20.26** | **mA** |

We documented torque, speed, and current values for the main prototype across 200 steps, with the full dataset in Appendix 1. Two graphs illustrated our findings: Figure 33 shows the relationship between desired torque and actual current output, while Figure 34 depicts the desired speed versus real current output.

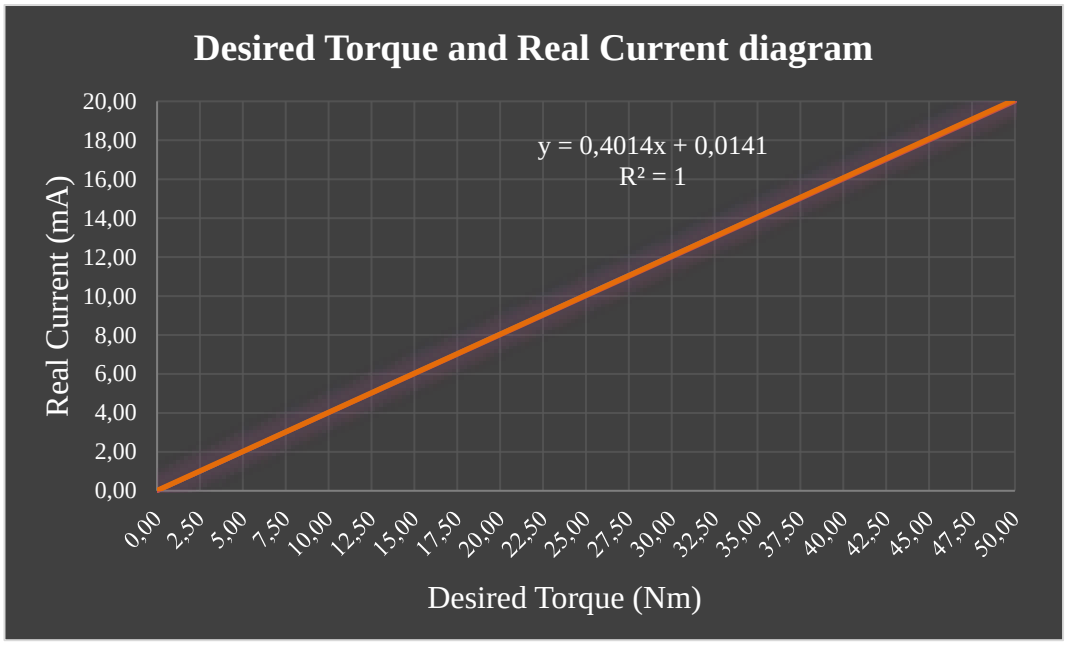


**Figure 33.** Desired torque and real current diagram.

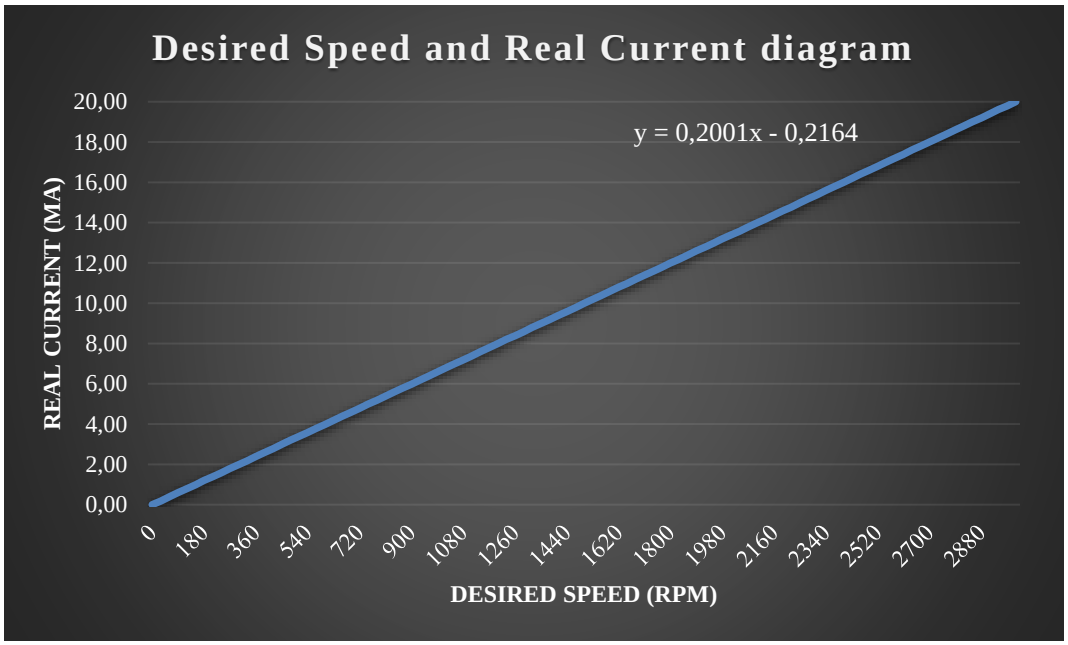


**Figure 34.** Desired speed and real current diagram.

Additionally, we recorded torque, speed, and current values for ten devices, comparing them to the results of the main prototype at 200 steps. The full dataset was presented in Appendix 2. To illustrate our findings, we created two graphs. Figure 35 shows the relationship between desired torque and actual current output for all devices, while Figure 36 depicts desired speed versus real current output. These diagrams were useful to identify performance patterns and deviations among the manufactured devices.

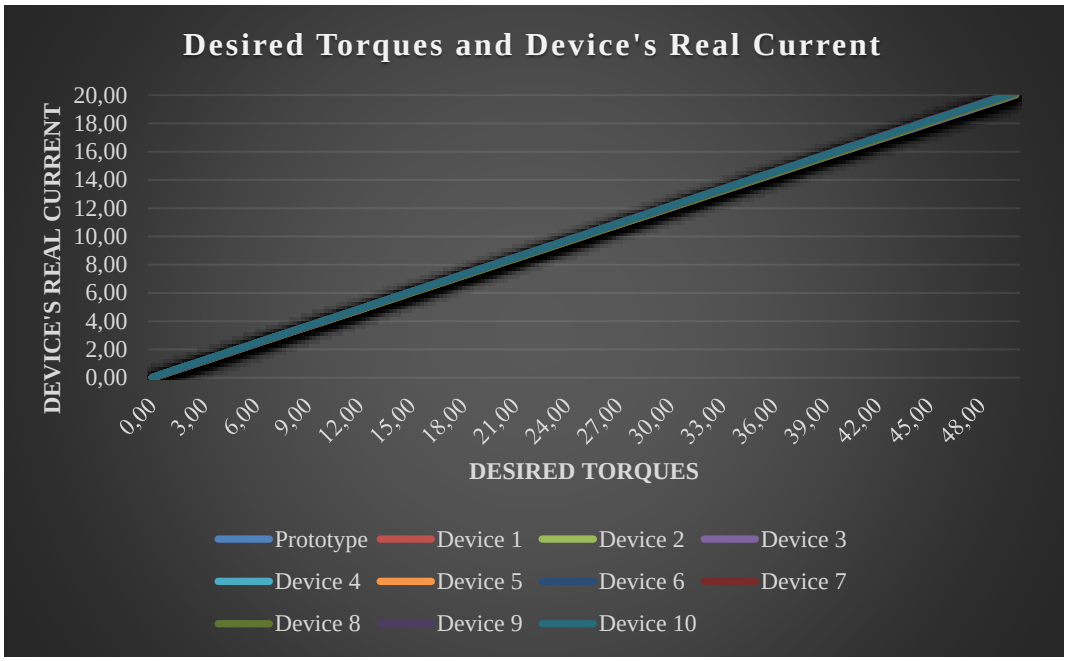


**Figure 35.** Desired torque and actual current output across all devices.

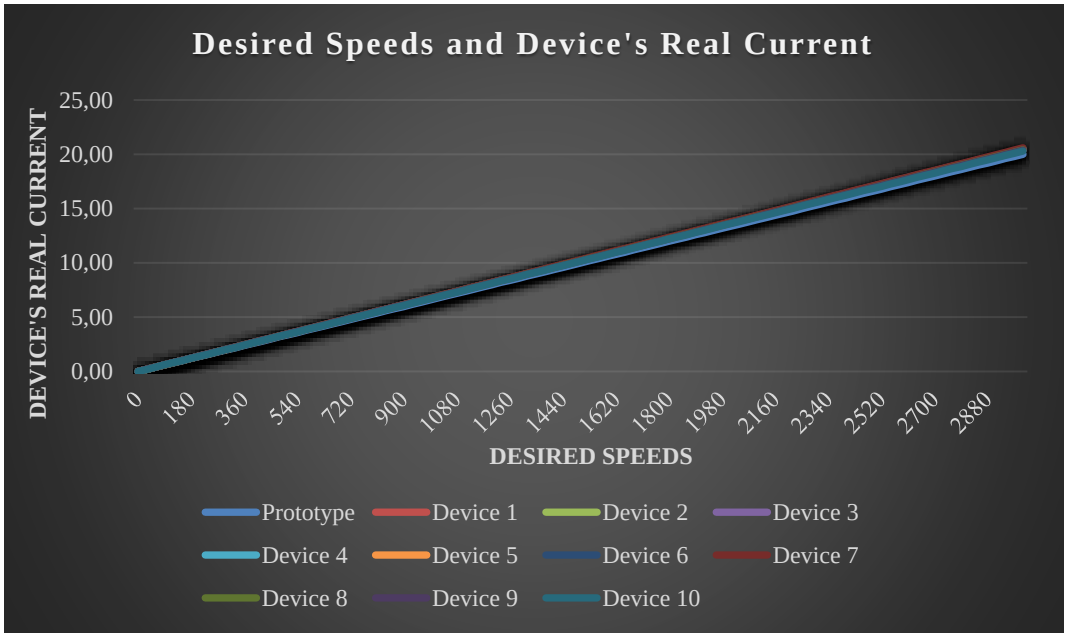


**Figure 36.** Desired speeds and actual current output across all devices.

As graphs above illustrate, the output torque and speed values of the ten professionally manufactured devices closely aligned with those of the main prototype. Minor variations were observed, potentially due to manufacturing tolerances, component differences, or assembly variations. This close alignment between the prototype and manufactured units demonstrate a high level of the robustness and reproducibility of our design.

# 9 CONCLUSION

The purpose of this project was to design and implement a digital 0 – 20 mA current loop device for frequency converters in motor technology applications. The development process involved three main stages: circuit design, software programming, and PCB design and manufacturing. The finished current loop device can read the desired values input by controlling the rotary encoder and convert them to analog signal values, creating the 0 – 20 mA current loop transmitting to Arduino Nano. When connecting this device to both the actual frequency converter and simulated environments using a 250 Ω resistor for testing purposes, we observed positive relationships between the desired torque (Nm) and speed (rpm) values and the current values measured by Fluke multimeters. Furthermore, we noted the same positive correlations with current values for the additional ten devices featuring professionally manufactured PCBs. The following data illustrates these correlations in detail:

- The desired torque values range from 0 – 50 Nm, with corresponding current values measured from 0 – 20 mA.
- The desired speed values range from 0 – 3000 rpm, with corresponding current values measured from 0 – 20 mA.

The project faced several challenges, including ground noise, poor quality of breadboards, capacitive cross-talk issue and frequency interference. These were addressed through the manufacture of integrated PCBs and the implementation of proper shielding and grounding techniques. The use of CAT6 cables with twisted pair wiring helped minimize capacitive cross-talk and external interferences. After eliminating these problems, the finished current loop device worked properly.

In summary, the project successfully achieved its goal of generating a reliable digital current loop device. However, for further enhancement on the performance of the device, we would strongly recommend that EMC testing would be implemented on this device. This crucial phase would serve to verify the long-term operational stability of the device and control potential issues that may arise during extended periods of use. In addition, since the DC power supply converter may

generate noise, we also recommend EMC testing on the filter circuit in future developments. This focused approach would effectively eliminate interference and noise, further enhancing the accuracy and precision of the device in real-world applications.

## LIST OF REFERENCES


/1/ 1N4148 Datasheet.

/2/ AM2D-2405SZ Datasheet.

/3/ Arduino Nano Datasheet.

/4/ COIL3-100uH Datasheet.

/5/ Datasheet and Manual of I2C LCD1602.

/6/ Datasheet of PEC11R-4220F-S0024 Incremental Encoder.

/7/ DFRobot Gravity 12-Bit I2C DAC Datasheet.

/8/ Dynapar. (n.d.). What is an incremental encoder. https://www.dynapar.com/technology/encoder_basics/incremental_encoder/#:~:text=What%20is%20an%20Incremental%20Encoder,most%20commonly%20used%20rotary%20encoders.

/9/ Elprocus. (n.d.). About Digital to Analog Converter (DAC) and Its Applications. https://www.elprocus.com/digital-to-analog-converter-dac-applications/

/10/ Elprocus. (n.d.). What is BC547 Transistor Working and Its Applications. https://www.elprocus.com/bc547-transistor-working-and-its-applications/

/11/ Filter B82725S2103N003 Datasheet.

/12/ Hoyer. (24 April 2023). Full control and optimised operation with frequency converters. https://hoyermotors.com/about-hoyer/information/product-and-market-news/full-control-and-optimised-operation-with-frequency-converters/

/13/ LM324 Datasheet.

/14/ Mauricio, D. (n.d.). Why a 0 mA Signal is Not Practical. https://www.predig.com/whitepaper/why-0-ma-signal-not-practical

/15/ Paonessa, S. (n.d.). Ground loops & Non-Isolated Commons. https://www.predig.com/indicatorpage/ground-loops-non-isolated-commons

/16/ Paonessa, S. and McDuffee, B. (n.d.). Back to Basics: The Fundamentals of 4 – 20 mA Current Loops. https://www.predig.com/indicatorpage/back-basics-fundamentals-4-20-ma-current-loops

/17/ Souza, R. (n.d.) I2C Protocol. https://www.prodigytechno.com/i2c-protocol

/18/ Specification for Approval of Coil3.

/19/ TL431 Datasheet.

/20/ Toroidal Inductor (Vertical, 100uH) Datasheet.

/21/ TR5-K Datasheet.

/22/ TVS Diodes Datasheet (P6KE Series).

/23/ TVS Varistor Datasheet.