Mechanical Engineering

[Team #: 20F09]
[Team ExoActuator]

To: [Dr. Oman]

From: Chancelor Cuddeback, Callum Fisher, Alex Frieden, Abdulrahman Alshammari, and Joshua Davidson

Date: [2/26/21]

Subject: [Implementation Memo]

Our team was tasked with the challenge of creating a testing bench to test the robot actuators. To test the actuators, the team must learn how to communicate with the actuator and its CAN bus protocol. This is done by using Serial UART and CAN 2.0 protocols in Arduino and CAN. The team uses a Nano 33 BLE Arduino board to communicate between the two protocols so the team can make a code to make the AK80-9KV100 motor rotate.

The actuator has an integrated MIT Mini Cheetah controller and the team's goal is to make it move. After that is achieved then the team will make a controller for the mini actuators that the client has in his lab. These mini actuators do not have controllers built into them so the team will have to code a current controller into them themselves. If the team meets these requirements, then additional tasks will be added on as per client requests.

1 Customer Requirements (CRs)

Customer Requirements:

- Make a testing bench.
- Construct a code that will test the AK80-9KV100 motor.
- Stay within $3,000

This project has changed a couple times since the beginning of the project. The project is still to test the AK80-9KV100 motor, but it is to just test the motor and not add sensors and brakes like it was initially introduced to the team from Dr. Trevas. The client has made it clear that making the motors spin is the main priority and if there is enough then more tasks will be added on for the team to do.

2 Engineering Requirements (ERs)

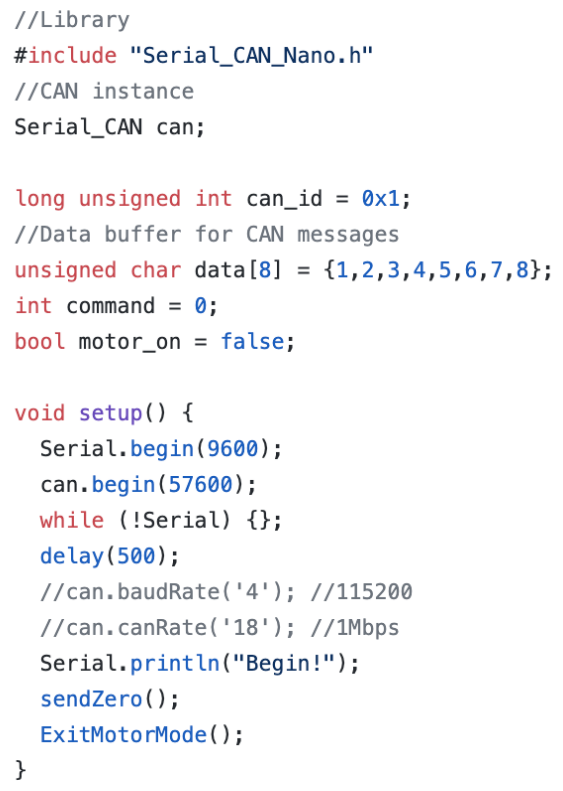
Engineering requirements:

1. AK80-9KV100 24V Nom 12A (Max 24A).
2. Controller should use CAN 2.0 and Serial UART protocols
3. Use a nano 33 ble module
4. Mounted to frame

2.1 ER #1: CAN Code

2.1.1 ER #1: Basic Control

The firmware of the motor controller is open source and has been used to aid in programming the communication specifics. The team must write a program to control the actuator using a serial to CAN module with an Arduino nano 33 ble. This program has so far demonstrated basic position control, but further work must be done to complete the torque and angular velocity control. As a side note, the program has yet to properly receive messages from the motor during operation. This must also be fixed. The program uses a modified version of the Seria_CAN_Aduino library from SeeedStudios, found at https://github.com/Longan-Labs/Serial_CAN_Arduino. A picture of setup() and some initialization instructions may be found to the left of this text. The actuator is configured using TeraTerm and this is where the CAN and Serial transmission Rates are set (1Mbps and 57600 baud respectively). Currently the program is being controlled by user input collected from the serial monitor, this will likely remain the same with some altered character bindings. The motor specific functions, which may be found in the source on Github, were modified from Skyentific's code at https://www.youtube.com/watch?v=HzY9vzgPZkA.



```
//Library
#include "Serial_CAN_Nano.h"
//CAN instance
Serial_CAN can;

long unsigned int can_id = 0x1;
//Data buffer for CAN messages
unsigned char data[8] = {1,2,3,4,5,6,7,8};
int command = 0;
bool motor_on = false;

void setup() {
  Serial.begin(9600);
  can.begin(57600);
  while (!Serial) {};
  delay(500);
  //can.baudRate('4'); //115200
  //can.canRate('18'); //1Mbps
  Serial.println("Begin!");
  sendZero();
  ExitMotorMode();
}
```

Figure 1: Basic Program Snippet

2.2 ER #2 (changed from fall): 3D Printed Mounts

2.2.1 ER #3: Mount Details

The mount for the motor was iterated several times in solidworks using the dimensions of the motor given on their website. The mount was printed with a low infill, and then adjustments were made to fit the mount to the motor and the 80/20 aluminium frame. M3 screws were used to attach the motor to the mount, and t-bolts were built into the side to allow the mount to sit at a right angle to the frame. The design was then changed to remove the t-bolts and replace them with a hole for a screw to attach the mount to the frame. This would allow for the mount to be fastened in place instead of sliding freely along the frame.

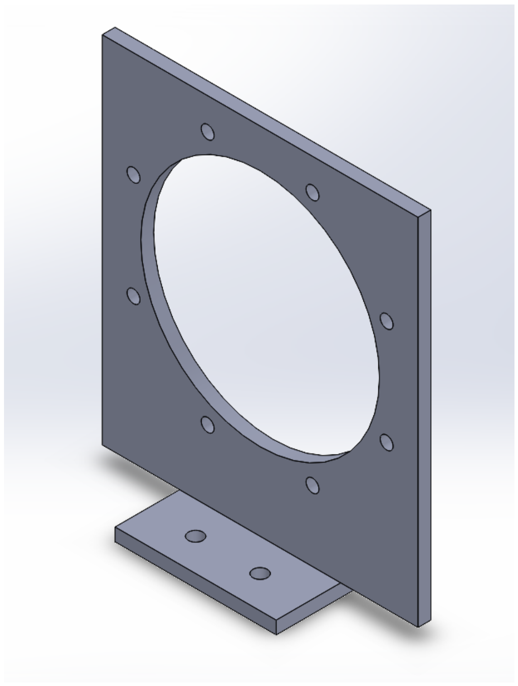


Figure 2: Motor Mount CAD Model

2.3 ER #3 :AK80-9 CAN Controller

2.3.1 ER #3: CAN Controller

2.4 ER#4 Testing Frame

2.4.1 ER #4: Frame assembly

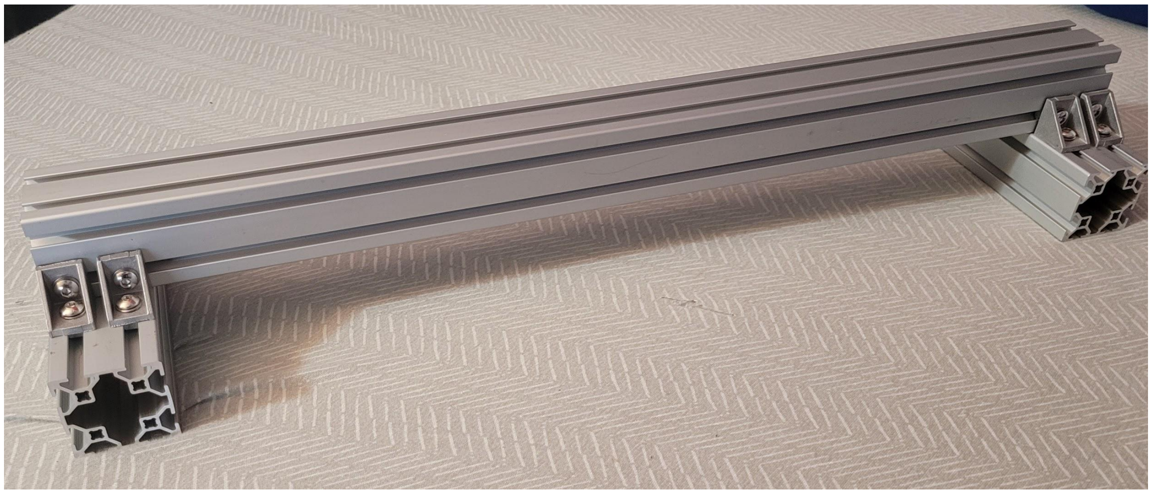


Figure 3: Aluminum 80/20 Extrusions

The frame the motor will be mounted has been fully assembled using aluminum extrusion from 80/20 Inc as of the writing of this document. The frame uses two 5-inch pieces of the 4040 profile as feet, and one 20-inch piece of 4040 profile across as the main structure of the frame. The frame was assembled using

T-Slot nuts on 90-degree brackets to attach the 20-inch member to the two 5-inch members.

3 Design Changes

3.1 Design Iteration 1: Change in [subsystem/component] discussion

The original design for the frame was to have the mount mounted parallel to the 80/20 frame. The extrusion was then changed to have the mount be perpendicular to the 80/20 as to save on material and shorten the time for printing. This also makes the print easier to make small fixes to the print with a file making the mount fit better in the frame.

Changes made to the code over time has been to refine the position, acceleration, and torque commands for the CAN code. The team was able to convert open source code found on github, the document shared to the team by the employer, and from videos that talk about the AK80 motor and how to code it using the CAN code built into the motor. After collecting all the open source codes, the team compiled them into one code. This code has been used so far to make the motor rotate which is what the client wanted.

4 Future Work

4.1 Further Design

The teams goals for the next few weeks is to work on refining the code and adding some more features so the client can do more with the motors. While refining the code, the team will need to pick a suitable battery to power the system. Some batteries have been looked into but one has not been finalized just yet. Further features will also be added according to client specification. Features such as: Torque control, integrating a PID controller, and angular velocity control.

4.2 Schedule Breakdown

The team has several active schedules in a shared drive. The schedules have been segmented into engineering deliverables and course deliverables. These schedules are very broad and are only meant to generally guide the team. Smaller action items are created informally amongst the team, and larger course deliverables have separate schedules and task assignments. The tables of the current schedules may be found below.

Engineering Plan of Action			
Task	**Due**	**Lead**	**Notes**
Send in revised purchase order	Jan 22	Alex/Joshua	We'll refine it with Dr.Lerner
Select Battery	Mar 5	Abdulrahman	24V, BMS, Light, 60A max, 20-30 minute runtime
Develop Procedure to find motor CAN IDs	Feb 5	Chance	Terraterm?, FTDI breakout
Create simple hardware abstracted CAN program	Feb 5	Callum/Chance	Arduino, SEED library, various control modes
Solder power source connections	Feb 10	Alex/Joshua	May have to purchase more
Print motor mounts	Feb 15	Alex	Use PLA printer until nice filament arrives
Validate various control modes	Mar 12	Chance	Angular velocity, torque, postion if time permits
Re-create motor Driver PCB	Mar 22	Abdulrahman	This should be multiple subtasks, up to you
Validate motor mounts	Mar 12	Alex	
Validate battery Selection	Mar 15	Chance	

Table 1: Engineering Deliverables

Capstone Deliverables		
Assignment	**Due**	**Portions**
Self Learnings	Jan 22	Individual
Hardware review	Feb. 8	--
Peer Eval 1	Feb 8	Individual
Website Check	Feb 15	Joshua
Implementation Memo	Feb 22	--
Midpoint Presentation	Mar 1	--
Individual Analysis II	Mar 8	Individual
Hardware review(Meeting + Memo)	Mar 15	--
Peer eval 2	Mar 15	Individual
Website Check	Mar 22	Joshua
Draft of Poster	Mar 29	--
Final Presentation	April 5	--

Final Product, Operation/Assembly Manual	April 12	--
Final Report, Final Poster	April 19	--
Client Project Handoff	April 26	--
Peer Eval 3	April 26	Individual
CAD Package, Website Check	April 26	--, Joshua

Table 2: Course Deliverables

Appendix

Current Program: https://github.com/ChanceCuddeback/CAN_TMotor
Some screenshots of the program: