42: Multi-Axis Cobot For Factory Automation

Adrian Guzman, Emily Hamsa, Ethan Woods, Jaishil Shah

CONCEPT OF OPERATIONS

REVISION – 2
5 December 2024

CONCEPT OF OPERATIONS
FOR
42: Multi-Axis Cobot For Factory Automation

TEAM <42>

APPROVED BY:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	9/12/2024	All Members	All Members	Draft Release - Revision 0
1	9/26/2024	All Members	All Members	Changed MCU selection in Section 3.6
2	12/5/2024	All Members	All Members	Updated team name to 42

1.1.

1.2. Table of Contents

1.3. List of Tables

1.4. List of Figures

2. Executive Summary

In many manufacturing industries, safety and efficiency are critical to ensure a smooth operation within a fabrication facility (fabs) or working environment. Fabs have heavy machinery and moving parts, making it difficult for humans to access certain places. To aid the workforce, collaborative robots (cobots) are deployed within factories allowing human control from safe distances. Cobots range in size, and application, and help streamline processes within manufacturing. This solution will protect human lives and reduce inefficiencies in labor by eliminating the process of entering a hostile or hazardous work environment.

This project aims to develop a multi-axis collaborative robot (cobot) that replaces manual human intervention with remote control, reducing the risk of injury and fatigue while improving efficiency in factory settings. The cobot will assist with light tasks such as lifting small loads (1-2 lbs), transporting items between locations, and providing precise movement control through a wireless app. Key software tools for the design and development include Altium, CCStudio, AutoCAD, and VSCode (with Swift). The cobot's hardware will feature a C2000x MCU and multiple B161x motor drivers to ensure reliable operation.

Versatility is a big focus of the cobot’s design, and thus we are including the ability to lift a small object or lightweight box using the pincher arm. The cobot will be battery-powered with rechargeable batteries, with a voltage supply range of 24-48V. Additionally, a wireless app will be developed for precise control of the cobot’s movements. To maximize its range of motion, the cobot will contain 5 axes of rotation supporting both 180° and 360°. Internally, the cobot will be driven through TI’s best-in-class C2000™ MCUs, power ICs, and motor drivers, ensuring precise motor control.

3. Introduction

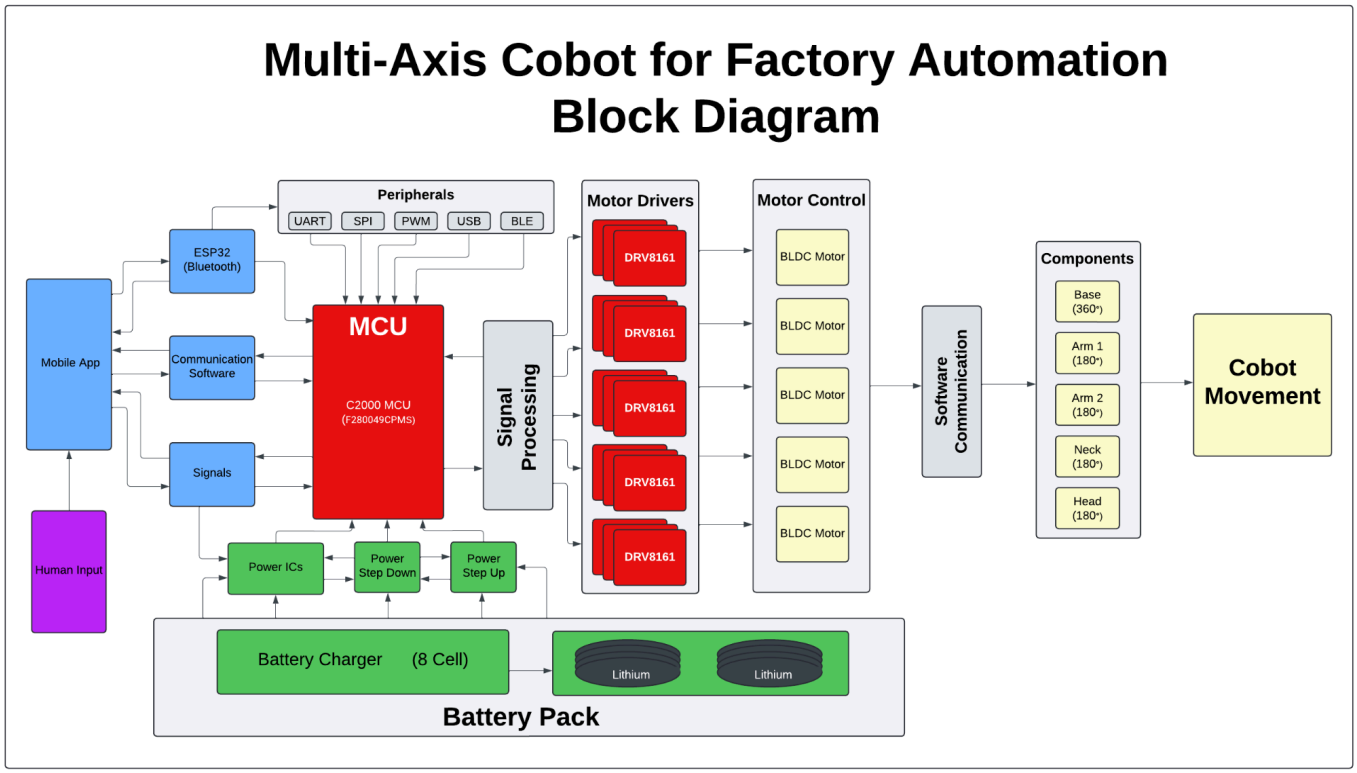
Factories and commercial industries use cobots to perform tasks that are unsafe and require numerous movements that would be considered strenuous for humans. Thus, the goal of this cobot is to make the production process safer and more efficient. For this project, we are developing a multi-axis cobot that will facilitate the movement of different loads across a factory or commercial setting. The cobot will be wirelessly connected using a mobile device, battery-powered, and support loads of 1-2 lbs. Considering that many warehouses have safety regulations and guidelines to follow, this cobot is designed with many of these in mind. Thus, this project’s main focus is to develop an operational cobot that could be placed in a factory setting to complete tasks safely and in a time-oriented manner.

3.1. Background

Collaborative robots, also known as cobots, are a crucial component in factory settings where safety is at the utmost priority and manual labor is considered too dangerous. By having the cobot function through remote control, we can increase the safety and efficiency of workers in these industrial settings, where small tasks could be performed by these cobots rather than human workers. This project aims to develop a multi-axial cobot that can lift a load of 1-2 lbs through a wireless connection.

3.2. Overview

1. Designing a block diagram and flow chart including the key operating specifications.
2. Develop the motor driver, microprocessor, power supply, and battery management control with safety systems, and respective PCBs for each.
3. Develop a user interface to control the cobot in the form of a wireless app.
4. Develop the physical design of the cobot, including the machine and any 3D-printed parts.



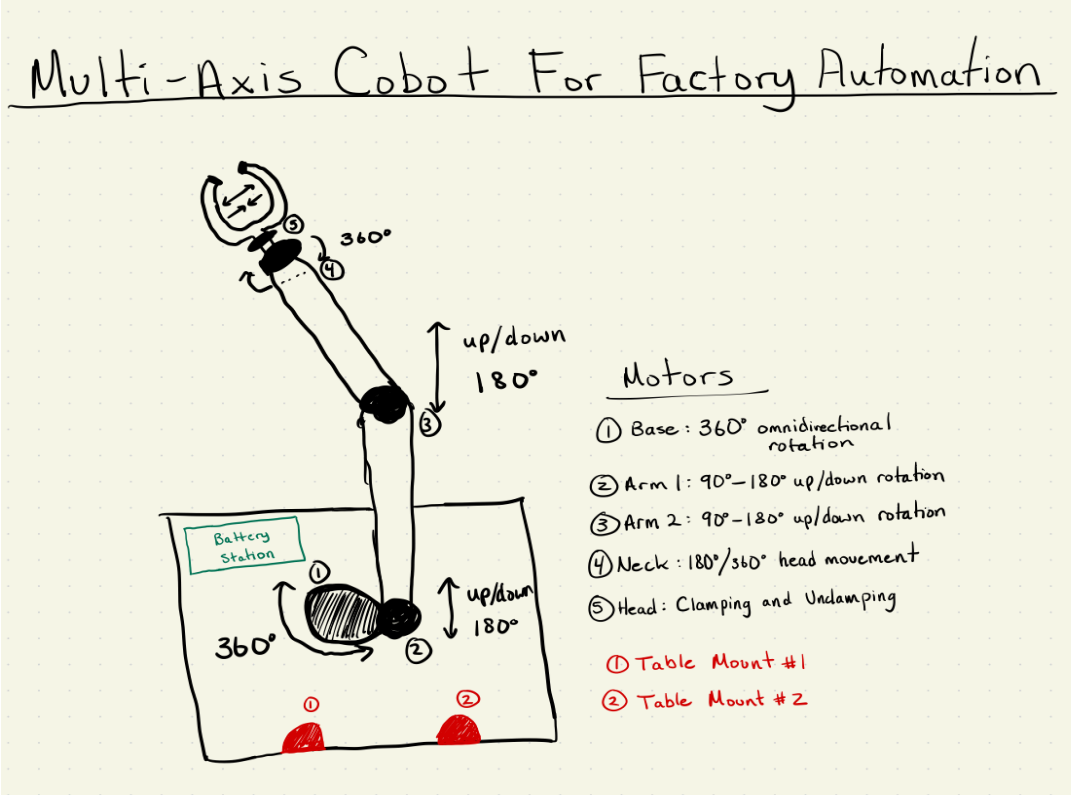
3.3. Figure 1. Multi-Axis Cobot Block Diagram

Segment Piece	Degree of Rotation
Base	360°
Arm 1	180°
Arm 2	180°
Neck	360° or 180°
Head	180°

3.4.

3.5.

3.6. Table 1. Cobot Segment Degrees of Rotation



3.7. Figure 2. Cobot Prototype Diagram

3.8. Referenced Documents and Standards

Project Outline: Project Description

MCU Documentation

1. C2000 MCU Documentation: F280049CPZS

2. C2000 Evaluation Module: LAUNCHXL-F280049C - EVM (C2000 Evaluation Module)

Motor Driver Documentation
DRV816x Motor Driver Documentation: DRV816x 100V Half-Bridge Smart Gate Driver
DRV8161 Evaluation Module: DRV8161 EVM

Wireless Control Documentation
ESP 32 Documentation**:** ESP 32 Technical Reference Manual
Swift Programming Course: Swift Programming Fundamentals

Motor Documentation - TBD
Battery Documentation - TBD
Physical Design Documentation - TBD
Standards Documentation - TBD

4. Operating Concept

4.1. Scope

This project involves the design and fabrication of a human-controlled, multi-axis collaborative robot (cobot) for use in a variety of manufacturing and commercial environments. The cobot will assist employees by improving efficiency in both strenuous tasks, such as moving boxes, and precise tasks, like handling small loads and assembling items. Operated remotely and equipped with a pincher-style head, it is designed to enhance productivity in diverse settings, performing repetitive and labor-intensive duties that benefit from human oversight and precision.

4.2. Operational Description and Constraints

The cobot will work wirelessly in commercial industries as well as factory settings performing tasks such as moving boxes and small loads, and the operator will use an app to control the cobot. The claw attachment will be used for precise tasks such as moving small objects and loading boxes, as well as moving small boxes between locations. It will move on multiple axes, ranging from 180 degrees to 360 degrees of rotation at each joint, and will be powered by a rechargeable battery.

One constraint of this project is the weight constraints on the load the cobot can lift. The design is only specified to operate in lifting loads up to 2 pounds. The radius of its reach is another constraint, as it will only be able to reach 2 feet from its base. Design

constraints will also need to be accounted for. There is a limited budget of $400, and a limited time frame to complete the design.

This cobot has applications in many industries, including the semiconductor, automotive, medical industries. This cobot could be modified to lift and move wafers in a semiconductor clean room setting. Being able to lift boxes would aid employees in the automotive field when building cars, and would be able to precisely move and place lighter metal parts. In the medical industry, the cobot would be able to use its precision for moving small tools such as tweezers or scalpels, assisting doctors during surgery.

Product	Quantity	Cost
EVM Board (MCU)	2	$78.00
Microprocessor Unit	2	$17.89
Bluetooth Driver	1	$19.99
EVM Board (Motor Drivers)	1	$229.00
Motor Driver Chip	5	$5.66
Anti Static Wrist Strap	2	$13.98
Robotic Arm Pincher	1	$26.99
SUM:		**$391.51**

4.3. Table 2. Cost Analysis (To be updated)

4.4. System Description

The system will be split into four subsystems: power and battery management, MCU and processing design, wireless connectivity and interface development, and motor driving system design. The physical construction of the cobot will be divided amongst team members equally in a 25% share.

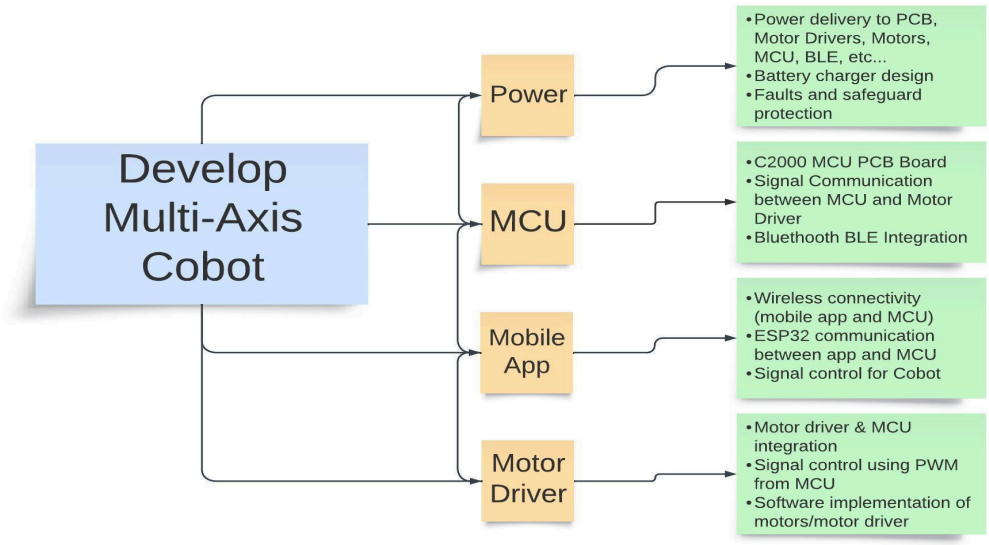
The power and battery management system covers the design of the power PCB which will deliver power to the motor drivers, motors, MCU, Bluetooth systems, and other electrical components. On the battery management side, this system will cover the selection of a battery and the design of a battery charger. This system will also account for possible faults and implement safeguards to protect the system.

The MCU and processing subsystem are responsible for the MCU PCB design, ESP32 Bluetooth integration, and communication between the MCU and motor driver. This subsystem will use Altium to design schematics and build physical hardware using the C2000 MCU. Furthermore, the MCU/Processing & Motor Driving subsystems will work closely to align on signal communication between the MCU and motor driver.

The wireless connectivity and interface system consists of a mobile application and a Bluetooth connection that allows communication between the cobot and the device sending in the inputs. The mobile application will be developed using Swift and will be compatible with any iOS device, including iPad and iPhone. It will allow the user to control all movements seamlessly and wirelessly. The Bluetooth driver, an ESP-32, will connect

directly to the MCU and allow the C2000 to take the inputs from the mobile device and convert them into outputs performed by the cobot. Since we are using a Bluetooth interface, the multi-device connection is compatible and allows any device to connect.

The motor driving design system will create the motor and motor driver PCB. This system will also create the signals needed to effectively operate the motors, which will be Brushless DC (BLDC) motors. This system will achieve the operation of two motors, a 360° and 180° motor controlled by the 8161x TI Motor Drivers. The motor drivers will read a signal from the C2000 MCU and spin/rotate as needed based on the signal. This will all be driven through the B161EVM before integration.



4.5. Figure 3. Subsystem Flow Chart

4.6. Modes of Operations

The cobot operates in three distinct modes. In the first mode, it uses its pincher to handle and pick up smaller objects in areas where employees may be present, though their presence is not required as the cobot is remotely controlled. The second mode involves lifting, moving, and placing down objects or boxes, as well as tasks such as using tools or adding items to boxes. The third mode is the charging state, where the cobot recharges its power.

4.7. Users

The target user for this cobot is a factory employee who performs redundant, strenuous tasks such as moving boxes from one place to another, transferring small objects, working with small tools, and loading boxes. These employees will benefit from using the cobot as they will be able to stay outside of dangerous areas to operate the cobot, removing the need to follow protocols to enter. Users will also be able to operate this cobot in areas amongst other employees to complete tasks like moving small loads and boxes.

Training will be required to use this cobot. Basic UI training will need to be completed to understand the functionality of the app, as well as training on using the Bluetooth features. Safety training will also be required to ensure standards are met for using a cobot in a factory setting. Understanding the cobot's thresholds, such as weight limits, is another important piece of training that will be needed before the operator can begin using the cobot. If the operator plans to use the cobot in areas with other employees, additional training regarding the safety of operation amongst humans may be required to ensure safety on the factory floor.

4.8. Support

Operators will be given a "crash course" on using the mobile application. In-person training on safety standards and operating conditions will also be provided, which will include weight management, operating radius, axis movement, claw operation, and other operating constraints. Operation in areas with other employees will also require additional training. Training on how to repair any damage to the equipment will be provided, as well.

5. Scenario(s)

5.1. Transporting and Using Small Objects or Tools

In this scenario, the cobot will be used in an environment where it will have the ability to operate in freedom alongside other human workers, with the goal of moving a small, lightweight object from one location to another. As the cobot is human operated, the primary objective is to transport the item without damaging or breaking it. The human operator must also be aware of the surroundings within the working environment in the case of external human intervention.

The cobot will be equipped with a set of pinchers or clamps with rubber pads attached to the end. It will be able to handle these objects with precision and full range of motion as it lifts the item up. The items the cobot can handle range from 1-2 pounds, such as light tools, food items, loose parts, and small plastic pieces.

The cobot has a hard time picking up flat items using its clamps, so the items would already be upright and ready to grab by the operator. Once picked up, the operator would be able to freely move the object with 360° of rotation and place it down or use the item at a secondary location, within the range of the cobot. This could be as easy as moving an object from one place to another to placing small objects inside bigger boxes.

From an operational point of view, a human operator will control the robot via remote control, responsible for all aspects of its movement and functionality. This includes tasks such as placing and charging the battery, as well as ensuring precise control during the cobot's movements. As the cobot is not autonomous, it requires a skilled human operator to handle the objects with the necessary precision and accuracy they need, alongside constant monitoring of the surroundings around the cobot, as to not injure or interfere with other workers.

5.2. Unpredictable Factory Environment

In this scenario, the cobot will be used in an environment where working conditions are dangerous for humans, such as hazardous and high-risk spaces. The cobot is human-operated, allowing for the pickup and placement of small, lightweight objects in areas where humans cannot operate safely.

The cobot will be equipped with pinchers or clamps that have rubber pads attached to the ends, allowing it to handle items with precision and care. These objects, weighing between 1-2 pounds, could include small objects that are potentially hazardous to humans. The cobot would struggle with flat items on a surface, so ideally the items are positioned upright already.

Once the object is picked up, the cobot can be rotated 360°, giving the operator full control to transport the item to another location or use it within the workspace. Tasks may range from moving items from one place to another, placing them in larger containers, or completing small assembly tasks. The human operator is responsible for controlling all aspects of the cobot’s movement via remote control, ensuring precise handling of objects and monitoring the surroundings to avoid accidents or collisions.

The cobot is not autonomous and relies entirely on a skilled human operator who ensures that tasks are performed safely and efficiently. This operator also manages the cobot's battery charging, positioning, and any other maintenance required to keep it operational. The use of the cobot in this environment significantly reduces risks for human workers while maintaining the efficiency needed for object handling tasks in hazardous conditions.

6. Analysis

6.1. Summary of Proposed Improvements

The multi-axis collaborative robot (cobot) introduces significant enhancements in safety, efficiency, and cost-effectiveness within warehouse and factory environments. By integrating a wireless user interface, the system minimizes direct human involvement, prioritizing worker safety and reducing potential hazards. The cobot’s ability to operate with five degrees of rotation enables it to handle and move objects weighing up to 2 lbs with high precision and efficiency. This capability streamlines repetitive tasks, reduces manual labor, and enhances overall operational productivity. Additionally, the cobot’s design contributes to cost savings by optimizing task performance and reducing the need for manual intervention in potentially risky environments.

6.2. Disadvantages and Limitations

Some limitations of our design include a carrying capacity of 1-2 lbs and an operational range of 2 feet from the base of the cobot. Additionally, due to the size and shape of our pincher mechanism, objects that are flat with little height off the ground will not be supported by the cobot. Since we are catering to industries which may have limited factory space, the range of motion must be precise in order to prevent damage to warehouse equipment and the cobot. One disadvantage of our design is the power supplied to the cobot. Since we are not connecting this machine to direct power but rather connecting a battery, the usage of the cobot will be limited to the duration of the 8-cell battery and the extra rechargeable batteries. To utilize the machine after a full day of use, the batteries must be recharged and plugged back into the machine.

6.3. Alternatives

Some alternate solutions we proposed include a physical controller and a combined cobot head that incorporates a hook and a pinching mechanism. With the physical controller, the benefit is that it would be easier to use, however, the drawbacks are lengthy enough to conclude that it is an inefficient proposition. With a dedicated controller with physical analog inputs, there would be a level of management of the signal interference that is avoided with the mobile application. Additionally, the mobile application allows many devices to communicate with the cobot, as well as being able to connect to different cobots. The final drawback of the physical controller is that if the device gets damaged, replacing or fixing it would be very costly. With a mobile application, releasing a software patch to fix any bugs lowers cost as well as time to resume functionality. The other alternative would be creating one cobot head with the hook and pincher attached. The issue with this solution is that it is not as professional in design, and does not display the versatility of the cobot in a consumer-friendly manner. Additionally, the hook is capable of damaging delicate parts that a dedicated rubber-headed pincher is capable of safely transporting.

6.4. Impact

The multi-axis collaborative robot (cobot) is designed to enhance operational efficiency and safety within manufacturing and commercial environments. By allowing human operators to control the cobot remotely, it reduces the physical strain and risk associated with repetitive or demanding tasks, such as moving objects and loading items. This shift not only minimizes the risk of worker fatigue and injury but also optimizes task precision and productivity.

Moreover, the cobot contributes to a safer work environment by performing tasks that could otherwise expose employees to hazardous conditions. Its presence in factory and warehouse settings helps maintain a safer operational environment by enabling workers to manage and control the cobot from a distance. This approach supports overall improvements in workplace ergonomics and efficiency, making it a valuable asset for modern manufacturing and commercial operations.

42: Multi-Axis Cobot For Factory Automation

Adrian Guzman, Emily Hamsa, Ethan Woods, Jaishil Shah

INTERFACE CONTROL DOCUMENT

REVISION – 1
5 December 2024

Interface Control Document
FOR
42: Multi-Axis Cobot For Factory Automation

Prepared by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	9/26/2024	All Members	All Members	Draft Release - Revision 0
1	12/5/2024	All Members	All Members	Updated error in team name, team 11 to team 42

Table of Contents

List of Tables

List of Figures

1. Overview

This document provides an overview of the integration between the different subsystems. The four subsystems are split into power, MCU/processing, wireless, and motor driving. The power subsystem will deliver accurate power to components, MCU/processing will ensure signals are sent from MCU, wireless will communicate with the cobot via Bluetooth, and motor driving will create motor movement. In this document, you will find the necessary system requirements to achieve this integration for the cobot. This document includes necessary references and definitions as well as physical, thermal, and electrical interfaces.

2. References and Definitions

2.1. References

Document Number	Revision/Release Date	Document Title
IEEE 802.15	June 2005	IEEE Standard for Information Technology
TMS320F28004x	January 2023	TMS320F28004x Real-Time Microcontrollers
DRV816x	July 2024	DRV816x 100V Half-Bridge Smart Gate Driver with Integrated Protection and Current Sense Amplifier
ESP-32	September 2024	ESP-32 Series

Table 1: References

2.2. Definitions

Cobot	Collaborative Robot
MCU	Microcontroller Unit
mA	Milliamp
mW	Milliwatt
in	Inch
lb	Pound
PCB	Printed Circuit Board
PMS	Power Management System
TBD	To Be Determined
V	Volt

3. Physical Interface

3.1. Weight

3.1.1. Weight of Cobot Apparatus

This section covers the weight of physical components and the design of the cobot. Encompassed in this section are the motor driving subsystem, MCU and processing subsystem, power management subsystem, and wireless communication subsystem. While we have not confirmed exact weights, we roughly estimate the cobot apparatus will weigh 10 pounds, not including the physical design and base.

Component	Weight	Number of Items	Total Weight
Motors	~ .4lb - 2.6lb	5	TBD
Motor Drivers	~ 0.22lb	15	TBD
Pincher	0.121lb	1	0.121lb
ESP-32	0.121lb	1	0.121lb
Physical Design and Base	TBD	1	TBD
Associated PCB’s	TBD	TBD	TBD

Table 2: Weight of Cobot Components

3.1.2. Weight of Battery Management

The battery management system is weighed separately because, in the final design, the battery and its charging system will be removable. While we have not confirmed exact weights, we roughly estimate the battery management system will weigh 4 pounds.

Component	Weight	Number of Items	Total Weight
Battery	~4lb	1	~4lb
Associated PCB’s	TBD	TBD	TBD

Table 3: Weight of Battery Management Components

3.2. Dimensions

3.2.1. Dimension of Motor Driving Subsystem

Each motor will have different specifications. For each motor to function, it will have a PCB with 3 DRV8161 Motor Drivers. We estimate the PCB alongside each motor will be at least 1.18 inches by 1.18 inches.

Component	**Length**	**Width**	**Height**
Motors	~ 1.57 - 3.15in	~ 1.18 - 3.15in	~ 1.57 - 3.94in
Motor Drivers	~ 0.2in	~ 0.12in	TBD
Associated PCB's	~ 1.18in	~ 1.18in	TBD

Table 4: Motor Driving Subsystem Dimensions

3.2.2. Dimension of MCU/Processing Subsystem

The MCU subsystem will consist of a PCB that holds the F280049CPZS microcontroller which has dimensions 0.63 inches by 0.63 inches. Furthermore, with other ICs and passive components integrated into the PCB, we estimate it will be at least 3.94 inches by 3.94 inches.

Component	**Length**	**Width**	**Height**
F280049CPZS	~ 0.63in	~ 0.63in	TBD
Associated PCB's	~ 3.94in	~ 3.94in	TBD

Table 5: MCU and Signals Processing Subsystem Dimensions

3.2.3. Dimension of Power/Battery Management Subsystem

The power and battery management subsystem envelops the battery, which we estimate will be 48 volts, alongside the PCBs that are needed to run the PMS. We estimate the battery will be around 10.63 inches by 3.15 inches by 2.76 inches and the PCBs needed will be around 3.34 inches by 3.34 inches.

Component	**Length**	**Width**	**Height**
Battery	~ 10.63in	~ 3.15in	~ 2.76in
Associated PCB's	~ 3.34in	~ 3.34in	TBD

Table 6: Power and Battery Management Subsystem Dimensions

3.2.4. Dimension of Wireless Communication Subsystem

The wireless communication subsystem is built around an ESP-32 board. The ESP-32 typically comes in size 0.71 inches by 1.00 inches by 0.12 inches. Including the size of the ESP-32 board and associated ICs, we estimate the dimensions of the PCB will be 2.76 inches by 2.76 inches.

Component	Length	Width	Height
ESP-32	~ 0.71in	~ 1.00in	~ 0.12in
Associated PCB's	~ 2.76in	~ 2.76in	TBD

Table 7: Wireless Communication Subsystem Dimensions

3.3. Mounting Locations

3.3.1. Motor Mounting

Each motor will need to be mounted and secured to the robotic arm apparatus. To prevent sag and weight issues, the mountings will have to be strong enough to support the weight of the motor under maximum load.

3.3.2. Cobot to Base Mounting

To ensure a strong center of gravity when lifting and moving a payload, the bottom (Base) of the cobot will be mounted to a baseboard or metal sheet. This allows for the cobot to have a designated location on a square surface to support the weight of the robotic arm with and without a load. By securing the cobot to a base plate, we can better control the lean and influence the movement of the cobot on its center of gravity.

3.3.3. Base to Table Mounting

To ensure the cobot does not tip over if its center of gravity changes, the base plate the cobot is mounted to will also be mounted to a table. The base plate can clamp onto the outside lip of the table it is placed on, which ideally means the table size is the size of the base plate. By clamping the base plate to the table, we eliminate the center of gravity changing due to the base plate lifting and leaning during movement.

4. Thermal Interface

Our project involves working with high voltage and current, which means cooling will be essential for the PCBs, motors, and other components to prevent overheating. Depending on the power draw and consumption, many of these parts will require adequate thermal management. Since the cobot is not designed for continuous operation over extended periods, a heatsink in the form of a metal heat dissipation plate, along with proper airflow, should be sufficient to maintain safe operating temperatures.

While a cold wall isn't necessary due to the intermittent operation of the system, heatsinks will still be required for most, if not all, of the motors in the robotic arm. These motors, subjected to high voltage and current, will generate heat more rapidly than lower-power alternatives. However, given the cobot's non-continuous, human-triggered operation, we believe that proper ventilation and airflow should be enough to manage any potential heat issues, even with the higher power motors.

In summary, the use of simple metal heat dissipation plates combined with adequate ventilation should address all thermal concerns without the need for more complex cooling solutions like a cold wall.

5. Electrical Interface

Provide details on the electrical interface. Examples are:

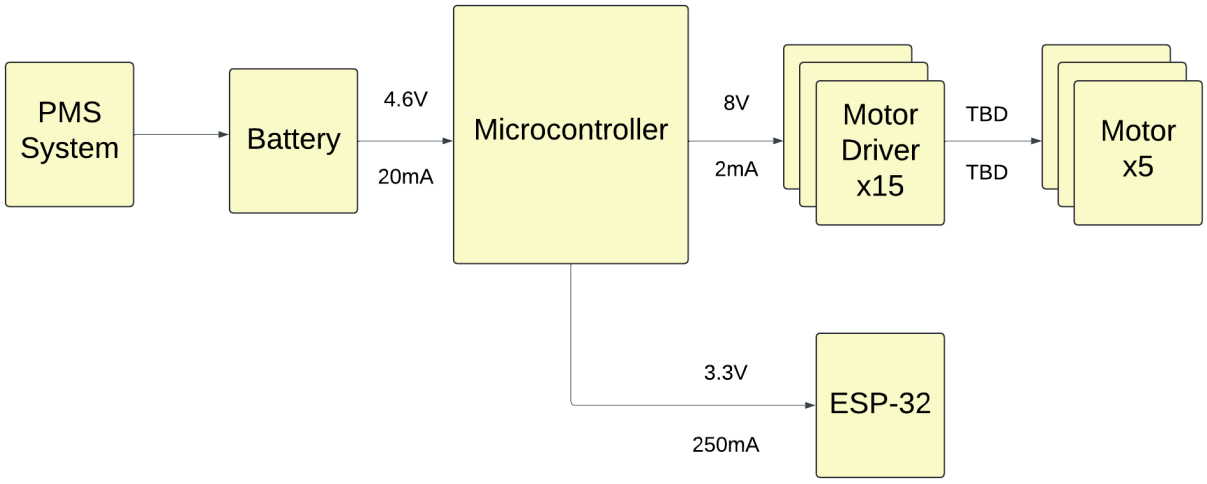


Figure 1: Electrical Interface Diagram

5.1. Primary Input Power

5.1.1. Primary Power Source for Cobot

All power for the cobot will be supplied by an external, rechargeable battery. The voltage rating for the battery is 48V, and the amp-hour specification is currently TBD as we are finalizing parts selection. Our battery will be connected to a battery management system, which will include regulators and buck converters to deliver power to all subsystems in the required amounts.

5.1.2. Battery Charging

Our team will also be developing a charging unit for the battery. Because we are using a large, multi-cell battery, this system will include safeguards and associated components to ensure when the battery is charging, cells are charged in a balanced way, and potential faults are detected.

5.1.3. Microcontroller

The cobot will be using the F280049CPZS microcontroller from the C2000 family at Texas Instruments. This MCU has a 32-bit CPU at 100 Mhz and includes 356 KB of on chip memory. There is no BLE module, hence an ESP-32 will be required externally.

5.2. Signal Interfaces

5.2.1. DRV8161 Pulse Width Modulation Signals

The communication between the motor drivers and motors will be facilitated through 3-phase Pulse Width Modulation (PWM), utilizing a C2000 MCU to generate the necessary control signals. This setup ensures precise and efficient control over the motor phases.

5.2.2. Digital Signals Between Mobile App and Cobot

The cobot will use an ESP-32 to connect the user's mobile app to the cobot through Bluetooth. The ESP-32 for use with Bluetooth is compliant with Bluetooth v4.2 BR/EDR and Bluetooth LE specifications, and has +9 dBm transmitting power.

5.3. User Control Interface

Users will be able to operate the cobot using a wireless mobile application that will be developed using FlutterFlow. This mobile app will include dialogs and menus which will allow the user to complete training videos and modules, connect to the cobot via Bluetooth, and receive error messages from the cobot.

5.4. Voltage and Current Levels

5.4.1. Maximum Voltage and Current

Component	Voltage [V]	Current [mA]	Power [mW]
Motor Driver	20	2	40
MCU	3.3	20	92
ESP-32	3.6	250	900
Motors	TBD	TBD	TBD

Table 8. Maximum Voltage, Current, and Power Levels

The table above outlines the maximum ratings for voltage and current consumption of each component. Operating under maximum conditions will not be done frequently, however, it is important to take into consideration when designing safety and sizing our battery.

5.4.2. Typical Voltage and Current

Component	Voltage [V]	Current [mA]	Power [mW]
Motor Driver	8	2	16
MCU	3.3	20	66
ESP-32	3.3	250	825
Motors	TBD	TBD	TBD

Table 9. Typical Operating Voltage, Current, and Power

The table above shows a more accurate representation of the power needs of each component during operation. While most of the components are relatively low-power, we anticipate our motors will have a much larger power requirement.

6. Communications / Device Interface Protocols

6.1. Wireless Communications (Bluetooth)

6.1.1. Bluetooth

The cobot uses Bluetooth v4.2 for wireless communication between the mobile app and the cobot's microcontroller unit (MCU). Since the C2000 MCU does not have a built-in Bluetooth module, an ESP-32 is employed for this purpose. The ESP-32 complies with the IEEE 802.15.1 standard, which ensures low-power, short-range wireless communication.

6.2. Host Device

6.2.1. C2000 Microcontroller Unit (MCU)

The C2000 serves as the primary Host Device for the cobot, as it is responsible for real-time processing and motor control. The C2000 will receive signals from the ESP-32, which communicates with the user’s controller. Then, the C2000 will generate PWM signals to send to the Motor Drivers for precise Motor control.

6.3. Device Peripheral Interface

6.3.1. ESP-32 Connection with User Interface

The ESP-32 serves as the wireless communication interface between the User Interface and the C2000. Using Bluetooth, the ESP-32 transits signals and commands from the Interface to the C2000. The communication is handled through UART.

6.3.2. Motor Driver and Motor Communication

The communication between the C2000 MCU and the motor drivers is managed through 3-phase Pulse Width Modulation (PWM) signals. These PWM signals control the motor phases to regulate speed, torque, and positioning. The C2000 MCU outputs these PWM signals directly to the motor drivers, ensuring precise control.

6.3.3. Micro-USB

A Micro-USB port will be needed to interface between the computer and the MCU. The MCU offers I2C, CAN, SPI, UART, and FSI. The communication protocol to use for micro-USB is still to be determined.

42: Multi-Axis Cobot For Factory Automation

Adrian Guzman, Emily Hamsa, Ethan Woods, Jaishil Shah

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – 1
5 December 2024

Functional System Requirements
FOR
42: Multi-Axis Cobot For Factory Automation

Prepared by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	9/24/2024	All Members	All Members	Draft Release - Revision 0
1	12/5/2024	All Members	All Members	Updated error in team name, team 11 to team 42, updated characteristics section

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Purpose and Scope

This specification defines the technical requirements for the development items and support subsystems delivered to the client for the project. Figure 1 shows a representative integration of the project in the proposed CONOPS. The verification requirements for the project are contained in a separate Execution and Validation Plan.

Warehouse and factory settings are dangerous environments in which to work. Workers are subject to injury, burnout, and fatigue with repetitive tasks. Instead of exposing humans to these risks, we aim to provide a collaborative robot, or “cobot”, solution to improve warehouse safety and efficiency. Our cobot shall be able to move objects (1-2 lbs) via human control from one location to another. A cobot solution will perform better than a human in tedious repetitive workloads as there is a reduced chance of error. Furthermore, cobots minimize the amount of risk humans are exposed to in a factory setting. Placing a cobot in dangerous environments eliminates the chance of a human worker getting struck by heavy machinery, falling objects, or other loose objects. Given that the cobot is wirelessly controlled, human intervention is still required, however, our solution will minimize the required manpower needed to achieve a task. We hope to minimize risk within warehouse operations and improve human worker longevity.

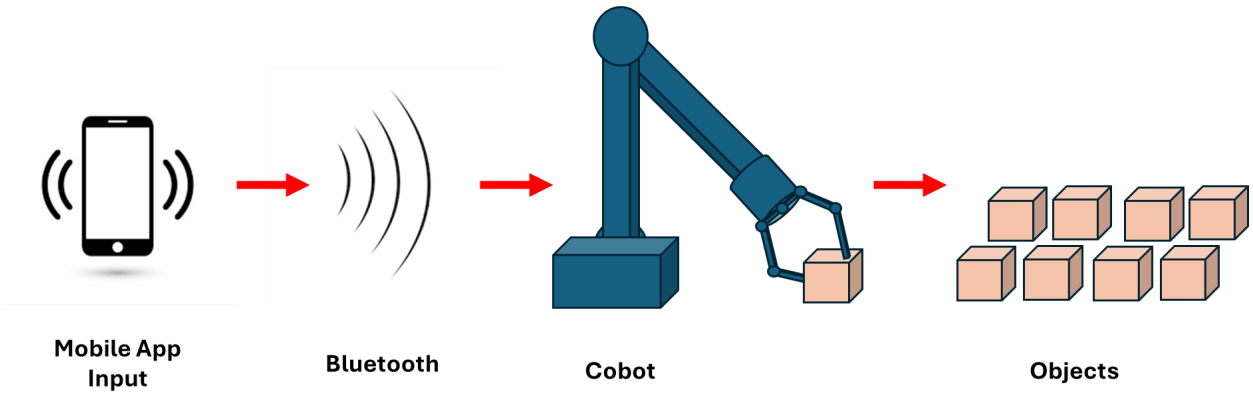


Figure 1. Conceptual Image of Multi-Axis Cobot

1.2. Responsibility and Change Authority

Every member is equally responsible for ensuring the requirements are met and will do so by reviewing deliverables when fellow team members signify their tasks have been completed. Changes proposed by any team members must be discussed with each other and the client, Joshua Maize. Responsibility will be further broken down at the subsystem level, which is outlined below in Table 1.

Subsystem	**Responsibility**
Power and Battery Management	Emily Hamsa
Microcontroller	Adrian Guzman
Wireless Connectivity	Jaishil Shah
Motors and Motor Drivers	Ethan Woods
Physical Design	Full Team Effort

Table 1. Subsystem Leads

2. Applicable and Reference Documents

2.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	**Revision/Release Date**	**Document Title**
IEEE 802.15.1.	June 2005	IEEE Standard for Information technology

Table 2. Applicable Documents

2.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
SLVSGZ1A	July 2024	DRV816x 100V Half-Bridge Smart Gate Driver with Integrated Protection and Current Sense Amplifier
SPRS945G	January 2023	TMS320F28004x Real-Time Microcontrollers
ESP-32	September 2024	ESP32 Series

Table 3. Reference Documents

2.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings, or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

This section defines the minimum requirements that the development items must meet. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered.

3.1. System Definition

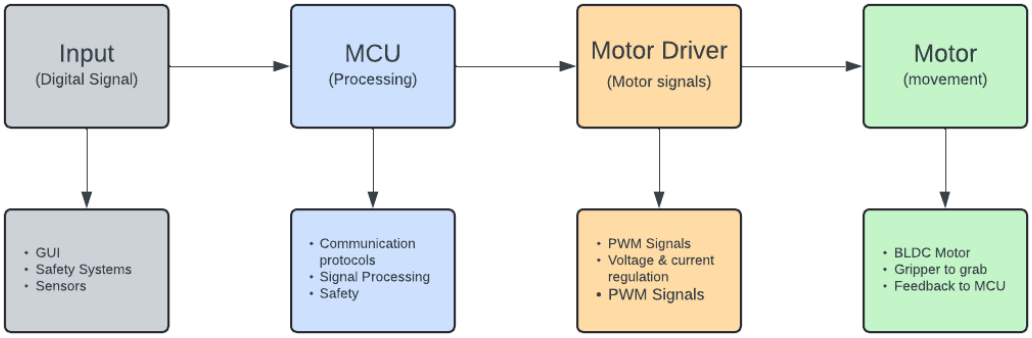


Figure 2. Multi-Axis Cobot Flowchart

The multi-axis collaborative robot used for factory automation is a solution to dangerous working environments for factory employees and the repetitive fatigue of monotonous tasks. This cobot will be operated via a wireless mobile app and will be powered by a rechargeable battery, allowing users flexibility to place the robot in a desired location and perform specific tasks. There will be four subsystems: power and battery management, microcontroller design, motor and motor driver design, and mobile application development.

When in use, the operator will be able to drive the motion of the cobot using the mobile application, which will be used on an iOS device. Signals sent from the app will be delivered to the cobot via Bluetooth, and we will use an ESP-32 for this functionality. The microcontroller will convert these signals into directions for the motor drivers, which will in turn deliver instructions to the motors. Using software communication, the motors will move each correlated component of the cobot. The cobot will be powered with a battery, and through the use of step-down converters, power will be delivered at the appropriate voltage and current rating to each component in the cobot. When charging, the battery management system will recharge the battery, and will include multi-cell balancing components to ensure each cell is charged to the maximum and not overloaded.

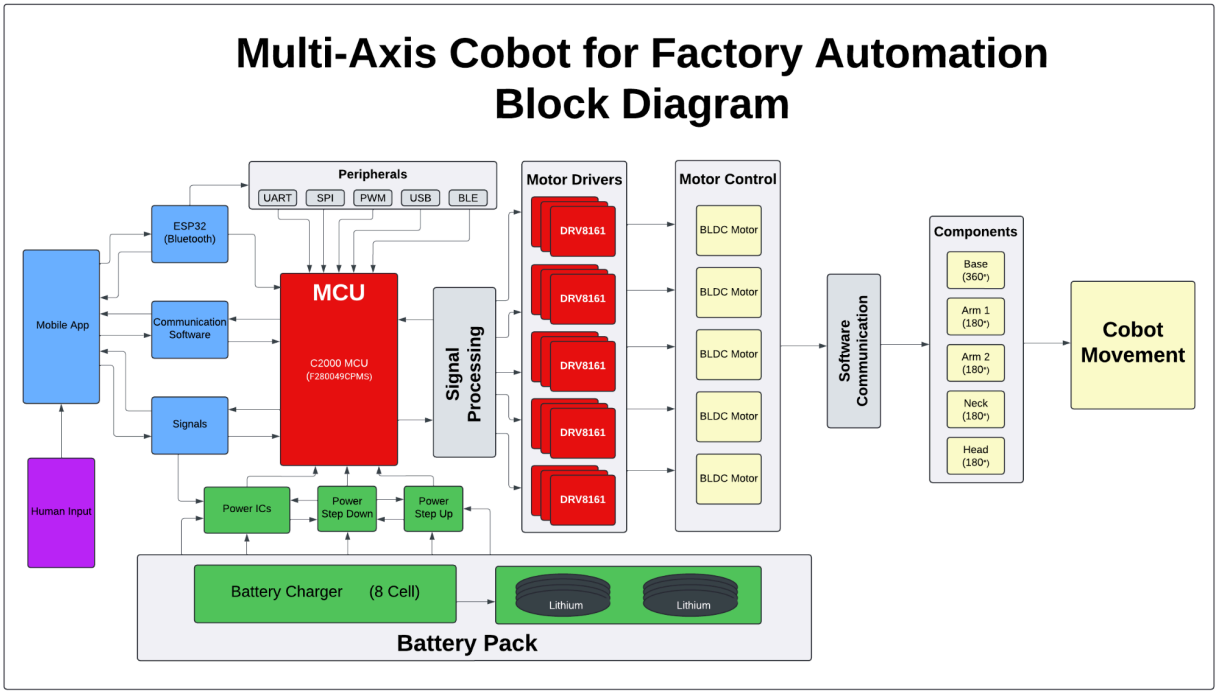


Figure 3. Multi-Axis Cobot Block Diagram

The cobot will also be able to send signals back to the mobile app to indicate any system errors. These errors include object detection, physical errors to motor drivers, and connection errors. Object detection will send warnings back to the user if there is an object interfering with the cobot's movement, such as a wall. Any physical damage or operation errors pertaining to the motor drivers will be sent to the user as well, indicating that servicing the cobot may be needed. If the mobile application is unable to connect to the cobot, error messages will be sent to the user as well.

3.2. Characteristics

3.2.1. Functional / Performance Requirements

3.2.1.1 C2000 EVM Wakeup

Launch EVM and connection is successful.

Rationale: In order to program MCU, understand the EVM by using example code on CCS and other TI-provided documents.

3.2.1.2 MCU & Motor Driver

MCU is correctly created to connect with motor drivers in the schematic.

Rationale: For motor motion, MCU and motor drivers have to work together to create signals with PWM modules.

3.2.1.3 JTAG Programming of MCU

The MCU is able to be programmed through JTAG. This will allow flashing onto the chip.

Rationale: The C2000 MCU needs to be capable of programming through JTAG. To enable this, signals such as TDI, TDO, TCK, TMS need to be properly routed.

3.2.1.3.1 Schematic & PCB Creation for MCU/Connectivity Subsystem

Schematic and PCB footprinting integrating MCU, motor drivers, and ESP32.

Rationale: Schematic and PCB design has no errors in validation. To ensure proper design of PCB the design rule check (DRC) shall come back with zero errors.

3.2.1.3.2 Schematic & PCB Creation for Motor Driver Subsystem

Creating schematics for PCB boards for individual motors, 5 in total.

Rationale: To ensure proper motor control, the MCU and motor driver subsystem need to be aligned on the proper signals required for motor motion.

3.2.1.4 Precise Motor Control

Through encoders, motors will move a few degrees at a time.

Rationale: Motors need to operate with different degrees of rotation. To support the different movements, various measurements of angles each motor can support will be performed.

3.2.1.5 All Motor Movement

All motors can move through communication with the MCU and motor drivers.

Rationale: Motor functionality is an integral part of ensuring the cobot moves accurately. The motors will need to be tested various times to ensure accurate representation of output with given input via wireless app.

3.2.1.6 Schematic & PCB Creation for Power Management

Power management system PCB schematics and footprinting are fully designed in Altium.

Rationale: Power will be imputed at 48V, not all components in the cobot will be at 48V. It is important that the power management subsystem can step down to different voltages that are required at different parts of the process. This will need a PCB with different power rails.

3.2.2. Physical Characteristics

3.2.2.1 Mounting

When the system is lifting maximum payload, it stays mounted on its surface without tipping.

Rationale: The cobot will only be able to support a specific load range. More than is allowed will make the cobot fall over. To find this range, the cobot will be tested by attaching weights to the arms of the cobot, measuring the maximum payload before the system tips over.

3.2.2.2 System Precision

Physical movement and load capacity functionality can move objects precisely and accurately.

Rationale: The cobot should not move an object differently than is intended as it will lead to damage to the item and or cobot. Ensure that the system axes can move at a minimum of 2 degrees, and move objects 2 inches on the table.

3.2.2.3 Structure Integrity

All components and joints are structurally sound.

Rationale: So that the cobot does not collapse when under load, the maximum payload needs to be tested. The cobot should support lifting a maximum payload, the physical design can support both the weight of itself and the load.

3.2.3. Electrical Requirements

3.2.3.1. Inputs

3.2.3.1.1 Input Voltage

The battery will supply a constant 48V to the system at the input

Rationale: The battery shall give 48V to the system. By using a multimeter, test to ensure the system holds at 48 volts while the battery is connected. This is required to ensure no voltage drop-off that could impact the functionality of the cobot.

3.2.3.1.2 Battery Sizing

Choose a battery size that will adequately power the cobot based on the load list.

Rationale: In order to power the cobot properly, battery sizing using formulas and a load list to determine appropriate voltage and amp-hour sizing is required.

3.2.3.1.3 Voltage Step-Down

Power PCB will step-down voltage to the subsequent system requirements

Rationale: Different systems of the cobot will require different input and output voltages. By using a multimeter, test each voltage rail and ensure it is holding the correct voltage.

3.2.3.1.4 MCU & Bluetooth

MCU schematic is created with ESP-32 for Bluetooth connectivity

Rationale: In order for Bluetooth to work via the wireless app, the mobile app needs to be able to communicate with the MCU. The MCU and wireless subsystems shall align on requirements, all necessary signals are accounted for and communication between Bluetooth and MCU can be displayed.

3.2.3.1.5 Motor Movement

Motor Drivers can send signals to move the Motor through the MCU

Rationale: For accurate and rapid motion of the motor, the MCU and motor drivers need to exchange signals. To achieve this, the two subsystems must align on required signals for control, and display movement through MCU control with motor drivers and motor.

3.2.3.1.6 GUI Operational

User Interface (application) is deployed to mobile device

Rationale: To control the cobot, the wireless GUI has to be operational and send out signals via Bluetooth. Once the connection is stable, input lag is minimal, and outputs are properly received/translated.

3.2.3.1.7 C2000 MCU Input from ESP-32

The C2000 is able to receive input from ESP-32 via toggling of high/low GPIO pins.

Rationale: To control the cobot, the ESP-32 will need to send high/low signals to the C2000. The C2000 will then receive these signals to output the required PWM waveform.

3.2.3.1.8 Button Functionality

The button will respond to human input and send digital output

Rationale: The cobot will be controlled via a wireless app. The app needs to be able to accept human input so that signals can be sent to the cobot and output can be visualized. This will lead to human interaction that works without fault.

3.2.3.2. Outputs

3.2.3.2.1 LED Program

After PCB assembly, flash the MCU and perform the LED light blinking program to ensure it is working.

Rationale: To demonstrate understanding of C2000 controls, software, and signals the C2000 EVM will be used. The CCS GUI with TI-provided software will help develop code that functions on the EVM. Once the EVM development is successful, this code can be replicated onto the created PCB for the MCU subsystem. Programming the MCU is an important part of achieving cobot functionality. In the end, an LED blinking on the board will confirm functionality.

3.2.3.2.2 Wireless Connectivity and Connection to MCU

Mobile application connects to the ESP-32 ad C2000 MCU.

Rationale: The mobile app is the main form of communication with the cobot. It is important to not lose connection and therefore we must test that the connection does not drop out more than 1 time in 5 minutes while also connecting within 30 seconds.

3.2.3.2.3 PWM Phase Output from Motor Drivers

The Motor Drivers can output Phase Signals which will be used to drive the Motors

Rationale: To be able to actually move a 3-Phase BLDC Motor, there must be 3 separate PWM Phases, each generated by 1 Motor Driver. By showing we can output one Phase Signal from a motor driver, we can say we can theoretically output 3 as well.

3.2.3.2.4 MCU to Motor Driver Output

MCU is correctly created to connect to the motor drivers in the schematic.

> *Rationale: To ensure proper communication between the motor driver and MCU, the schematic on Altium must be created with zero DRC errors. Furthermore, the two subsystems must Align to ensure signals are being sent properly to show motor movements.*

3.2.3.2.5. Output DC Voltage

The power management and supply PCB will output a DC voltage with minimal noise.

> *Rationale: To ensure the cobot components are working correctly, it is important that they are receiving the correct and expected voltage, which is a DC voltage value.*

3.2.3.2.6 MCU Generates PWM Signals

MCU can generate PWM signals from GPIO Pins, these will be sent to the motor driver.

> *Rationale: For motor motion, the MCU needs to be capable of generating PWM signals.*

3.2.4. Environmental Characteristics

3.2.4.1 Warehouse and Factory Safety Considerations

Ensure the surface that the cobot is stationed on is able to support at least 30 pounds so that it can sufficiently hold the cobot's weight, and that the surroundings are clear of items that could potentially be damaged by the cobot's movement.

> *Rationale: The cobot should not damage the environment it is placed in. Test the weight limit of the surface using weights that total 30 pounds and use a ruler to measure the radius of movement of the cobot, checking if there are damageable items in its path.*

3.2.4.1 Delicate Material (Handle with Caution)

The cobot is able to lift and handle delicate items without causing them any physical damage.

> *Rationale: The cobot shall not break anything it picks up. If objects are broken when being lifted, the purpose of the cobot is defeated. Cobot shall pick up a delicate item, such as a cracker or an object that has surfaces that could be easily punctured, and move from one location to another. Ensure there is no physical damage to the delicate item.*

3.2.5. Failure Propagation

3.2.5.1 Object Detection in Factory Setting

If part of the cobot's arm, head, or other components is turning or moving into a surface, such as a wall, that is restricting the movement of the cobot, an error message will be sent to the user's mobile application.

> *Rationale: In a factory setting, many objects are moving around. The cobot should not damage any other objects and will send warning messages to the user if movement can not be continued. Using the controller, turn the cobot into a wall so that it is unable to continue turning, and ensure an error message is sent to the user through the mobile application.*

3.2.5.2 Physical Errors with Motor Drivers

If there is physical damage detected to the motor drivers that is inhibiting the cobot to function correctly, an error will be sent to the user's mobile application.

Rationale: To work the user of potential faults, the motor driver will give the user a warning if an error has been detected. Such errors can include physical damage, object obstruction, or low battery.

3.2.5.3 Connection Errors

If the mobile app is not able to connect to the cobot, or the mobile app loses connection to the cobot, an error message will be sent to the user's mobile application.

Rationale: Connection should be constant from wireless app to cobot. If connection drops, danger can occur and humans no longer have control of the cobot. To prevent this, error messages will be sent to the user via mobile app.

3.2.5.4 C2000 Error Detection System Functionality

Any errors that the cobot or MCU detects will send a signal to the ESP-32 for the mobile application to receive.

Rationale: Signal detection using the MCU and ESP-32 is implemented to give human warning. The error message will display on the mobile application so that the user is made aware and the error or faulty use can be corrected.

4. Support Requirements

4.1 iOS Device Compatibility

The device is required to run the latest version of iOS to run this application. (9/24/24: iOS 18.0) Make sure that bluetooth is on and other devices are disconnected prior to device setup.

Rationale: The mobile app will be placed on the app store for user download. The user must have the latest iOS versions to comply with mobile applications. It is recommended to have the battery on the phone at full charge so that the connection will not drop if the phone dies.

4.2 Surface Preparation

Before placement of the device onto the workstation, make sure to clear the area and make sure the clamp is secure onto the surface.

Rationale: The device should be placed on a clean flat surface. If this is not done, risk of short circuit or other damage can be caused. Ensure that all liquids, hazardous material, and

sharp or dangerous objects are either relocated or placed in an area that is outside the range of the cobot.

4.3 Device Warranty and Replacement

This device contains parts manufactured by Texas Instruments and individually designed components.

Rationale: As the sponsor of the project, Texas Instruments devices should go into this cobot. To replace TI parts, directly contact Texas Instruments. In order to replace any individually designed parts, contact the Multi-Axis Cobot Design Team.

4.4 Training Requirement

Please ensure all technicians complete training and have been given warehouse/factory instruction on what the machine is being utilized for.

Rationale: In order to access the device connection and control on the mobile application, the user must complete the training and watch the safety demonstration.

4.5 Electrical Safety

Make sure all batteries, wires, and electrical components are secure and not in the range of the cobot's movement.

Rationale: The Multi-Axial Cobot Design Team is not responsible for any damage, harm, or injury caused in the workplace. Please understand that all responsibility falls upon the user.

4.6 Human Safety

Do not operate the device if humans are within 5 feet of the cobot.

Rationale: As previously stated, any harm caused in the workplace due to the cobot falls on the user of the device. Ensure all technicians understand their responsibility and are aware of the preventative measures.

Appendix A: Acronyms and Abbreviations

TI	Texas Instruments
IEEE	Institute of Electrical and Electronics Engineers
iOS	iPhone Operating System
CCS	Code Composer Studio
LED	Light-Emitting Diode
PCB	Printed Circuit Board
DRC	Design Rule Check
MCU	Micro-Controller
ESP-32	Espressif32 Bluetooth Driver
C2000	(TI) Microcontroller
GUI	Graphical User Interface
DRC	Design Rule Checking
EVM	Evaluation Module
PWM	Pulse Width Modulation
BLDC	Brushless Direct Current

ICD	Interface Control Document
UART	Universal Asynchronous Receiver Transmitter
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
BLE	Bluetooth Low Energy

Appendix B: Definition of Terms

42: Multi-Axis Cobot For Factory Automation

Adrian Guzman, Emily Hamsa, Ethan Woods, Jaishil Shah

SUBSYSTEM REPORTS

REVISION – 0
5 December 2024

SUBSYSTEM REPORT
FOR
42: Multi-Axis Cobot For Factory Automation

PREPARED BY:

_____
Author Date

APPROVED BY:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
0	12/5/2024	All Members	All Members	Draft Release - Revision 0

Table of Contents

List of Tables

List of Figures

1. Overview

The multi-axis cobot for factory automation will be able to lift and move objects wirelessly to improve productivity, minimize worker injury, and work in dangerous environments. The cobot will take user input through a mobile app and then move according to the input. The system is broken down into four subsystems, motor/motor driving, MCU/processing, power/battery management, and user interface/wireless connectivity. Each team member has researched, developed, and tested their individual subsystem. The validation and execution plans have been followed closely to ensure an easy path to integration in the following semester.

2. Motor/Motor Driver Subsystem - Ethan Woods

2.1. Subsystem Introduction

The Motor/Motor Driver Subsystem is responsible for controlling the movement of the Collaborative Robot (Cobot) through the selection and control of the motors. This subsystem focuses on identifying the motors capable of handling the calculated torque requirements for the different joints in the Cobot. These motors are driven using Texas Instruments' DRV8161 Motor Drivers, which controls the 3-Phase Brushless DC (BLDC) motors using generated PWM signals.

To achieve motor movement, custom PCBs were designed and fabricated with the primary goal of spinning a 3-phase BLDC motor. These boards integrate key functionalities, including voltage step-down circuits, Motor Driver circuits, and connections for phase signal generation, inspired by the Texas Instruments' DRV8161 Evaluation Module. The PCB system consists of a main board and two supporting boards, collectively enabling control and high-performance operation of the motor. The design has compatibility with the LAUNCHXL-F280049C, a Texas Instruments' C2000 Launchpad Development Kit, and supports the high torque, voltage, and current requirements of the motors needed.

2.2. Subsystem Details

During this semester, the subsystem had two main goals: **Motor Selection** and **Motor Movement**. The first goal was to calculate the necessary torque needed for each motor in the Cobot to accommodate its maximum payload, desired dimensions, and material weight. Based on these calculations, five motors were selected—one for each joint: Base, Arm 1, Arm 2, Neck, and Head. All motors except for the Head are 3-phase BLDC motors, chosen for their precision and compatibility with the DRV8161 Motor Drivers.

The second goal was to move and spin one of the selected motors. To achieve this, the DRV8161 Motor Drivers were used to generate the three phases required for the operation of a 3-phase BLDC motor. This involved designing and creating a system of PCBs that worked together to generate and route the necessary signals for motor control. The motor was connected to a main board, which ensured it received all the signals needed for movement and allowed control through an external device.

2.2.1. Torque Calculations and Motor Selection

At the start of the semester, the team met to conceptualize the operation and movement of the Cobot and establish its primary constraints. As a Collaborative Robot, the design emphasized smooth and unrestricted movement within its range of motion. This discussion led to the decision to implement a 5-Degree of Freedom (DoF) design, requiring five motors—one for each joint. At this stage, the focus was solely on the degrees of freedom and the maximum payload the Cobot should handle, without addressing details about its material composition or dimensions.

With this conceptual framework in place, the next step involved analyzing the conditions under which the motors would operate. Using free-body diagrams and calculations, the assumptions and constraints necessary to determine the torque range required for motor selection were defined. Below is the process followed to determine the torque range required for each motor. Calculations are not provided here but are available in the document linked below.

These calculations and results are presented in the document: *Motor Calculations*
If the link does not work, send concerns to ewoods738@tamu.edu or (210)774-0465

To calculate the required torques for each of the five motors in the conceptual Cobot, the design was translated into a 5-jointed robotic arm representation. This step enabled visualization of the system and the creation of the free-body diagram (FBD) shown below, which depicts the robotic arm in its worst-case scenario. By analyzing this worst-case configuration, the torque calculations are intentionally overestimated, ensuring a safety margin in the design rather than risking underestimations. Below is the conceptual 5-DoF design and the corresponding free-body diagram used as the basis for the torque calculations.

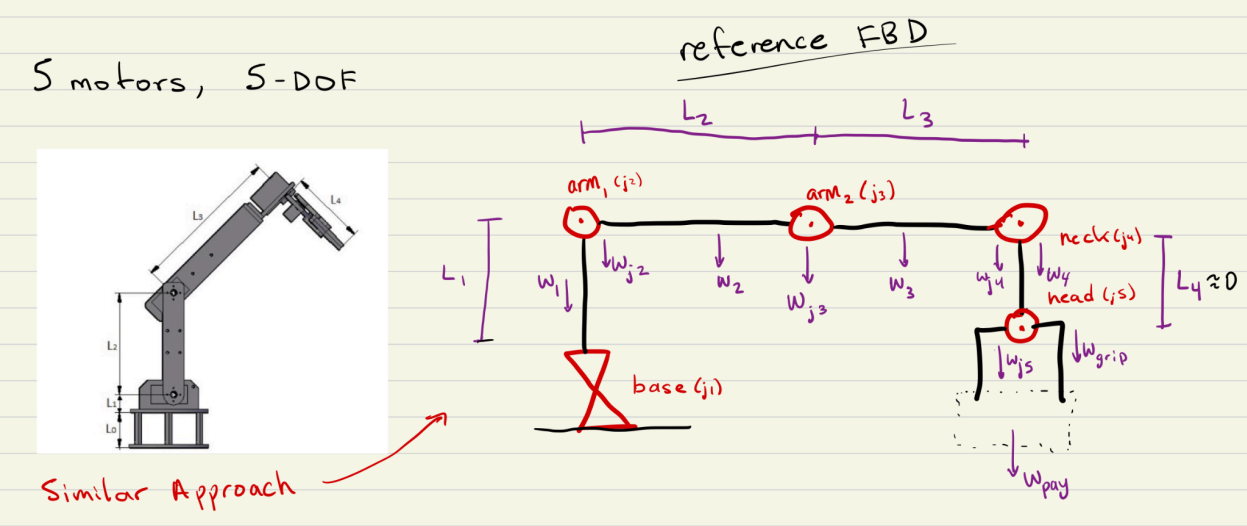


Figure 1: Conceptual Design and Free-Body Diagram

With the creation of the free-body diagram (FBD), the forces acting on each joint were determined, enabling the calculation of the gravitational or angular torque required for each motor. Motors responsible for lifting (Arm 1 and Arm 2) require gravitational torque calculations, while those responsible for rotation (Base and Neck) require angular torque calculations. To account for variability in the Cobot's design, the torque requirements were classified into three ranges: **Low Torque**, **Medium Torque**, and **High Torque**.

- **Low Torque** represents a smaller and lighter-than-expected design.
- **Medium Torque** reflects the expected design based on our initial assumptions.
- **High Torque** assumes a heavier, overestimated design to ensure safety margins in the motor selection.

Torque Categories: Differences and Similarities

The differences between the torque ranges are based on varying constraints such as arm length, material density, and acceleration. However, certain assumptions, such as the payload and the gripper motor's specifications, remained consistent across all categories. The table below summarizes these differences and similarities:

Category	Base and Neck Acceleration	Weight of Arms (Material Density,Cross-Section)	Length of Arms	Payload Weight	Gripper Motor Weight/ Length	Joint Motor Weight
Low Torque	Reduced	Lower density, smaller cross-section	Shorter	Fixed (2 lb / 0.91 kg)	Fixed* (Amazon Servo Motor)	Fixed/ Estimated**
Medium Torque	Average/ Expected	Average density and cross-section	Average/ Expected	Fixed	Fixed*	Fixed/ Estimated**
High Torque	Increased	Higher density, and larger cross-section	Longer	Fixed	Fixed*	Fixed/ Estimated**

Table 1: Differences and Similarities between Torque Categories

* The motor and claw specifications are on Amazon. These were used during calculations.
** Weights were estimated based on the average weights of different motors that could be used during the project. Ended up being 1.4 kg for Joint 2, 0.8 kg for Joint 3, and 0.25 kg for Joint 4.

With the completion of the torque calculations, the estimated torque requirements for each motor in the Cobot have been determined across three categories: Low Torque, Medium Torque, and High Torque. These calculations provide a clear range of torque values for each joint—Base, Arm 1, Arm 2, Neck, and Head—ensuring the selected motors can handle both expected and extreme conditions. The table below summarizes the calculated torque values for each motor in all three categories, serving as the foundation for motor selection.

Motor	Low Range Torque	Mid Range Torque	High Range Torque
Base Motor	.1428 N/m**	.5339 N/m	1.993 N/m
Arm 1 Motor	563 oz-in***	1197 oz-in	2763 oz-in
Arm 2 Motor	181.72 oz-in	354 oz-in	745 oz-in
Neck Motor	.00649 N/m	.00973 N/m	.0129 N/m
Gripper Motor*	11 kg/cm	11 kg/cm	11 kg/cm

Table 2: Motor Torque for Each Category

* Gripper Motor is bought off Amazon. Cannot change the value of the torque produced or required for the Gripper Motor as 11 kg/cm should be more than enough for 2 lbs.
** N/m is the measurement for angular torque, as it is the amount of torque needed to produce the acceleration necessary for the Base and Neck motors
*** oz-in is a measurement of torque in relation to gravity. It shows the lifting power needed for that motor to lift a payload of 2 lbs when considering the entire design of the Cobot.

Motor Selection Based off Estimated Torque Range

With the established torque ranges for each joint, we could now select motors that meet the performance requirements while aligning with the constraints of the Cobot's design. The motors chosen are high-power, 3-phase BLDC motors, capable of handling the calculated torque for their respective joints. The table below highlights the selected motors, along with their key specifications such as voltage, current, torque output, and cost, ensuring they are well-suited for their designated roles in the Cobot.

Motor	Name	Link	Torque	Rated Voltage	Rated Current	Encoder Type	Weight	Price
Base	57BSA	link	0.6 N/m	48 V	5.4 A	Hall	1.35 kg	$59
Arm 1	NEMA 34 Hudson	link	1223 oz-in	48 V	~10 A max: 20 A	Differential/ Hall	2.8 kg	~$360
Arm 2	NEMA 23 Hudson	link	223 oz-in	48 V	~ 10 A max: 30 A	Differential/ Hall	0.6 kg	~$210
Neck	42BYA	link	0.04 N/m	24 V	1 A	Hall	0.3 kg	$20
Head	MG996R	link	11 kg/cm	5 V	0.17 A	N/A	0.095 kg	$27

Table 3: Selected Motors for 5-DoF Collaborative Robot

With the torque calculations complete and motors selected, my focus for the remainder of the semester shifted to achieving motor movement. The NEMA 23 Hudson motor, paired with a 48V 25A power supply, was chosen as the test motor for this phase. Using custom-designed PCBs, the goal is to successfully drive and spin the chosen motor. The process of how this was accomplished is outlined in Section 2.2.2 below.

2.2.2. Motor Movement

To enable movement in the selected 3-Phase BLDC motor, the focus shifts to the second critical component of this subsystem: Texas Instruments' DRV8161 Motor Driver. This motor driver serves as the foundation for controlling all motor operations in this project by generating the phase signals required for motion. Given that our above motor operates with three phases, three DRV8161 Motor Drivers are necessary to facilitate its movement. The following sections will outline the design and implementation of a system of PCBs used to produce the necessary control signals, leading to the successful operation of the 3-Phase BLDC motor.

2.2.2.1. DRV8161 Motor Driver

The **DRV8161** is a high-performance, 100V half-bridge smart gate driver designed to enable precise and reliable control of high-voltage, high-current motor systems. As part of Texas Instruments' latest DRV816x family, released in early 2024, this driver integrates advanced features such as a low-offset current sense amplifier, robust protection mechanisms, and versatile PWM control options. The DRV8161 supports a wide range of features that make it ideal for high-performance applications, such as an industrial Collaborative Robot. It includes built-in protection systems to guard against undervoltage, overcurrent, and overheating, ensuring safe operation under various conditions. With a Pulse-Width Modulation (PWM) signal input, the chip produces phase signals needed to control motor movement accurately.

Figure 2 illustrates a simplified schematic of how three DRV8161 Motor Drivers are used to generate the three-phase signals required to control a BLDC motor. The figure also includes the 20-pin pinout of the DRV8161, highlighting the essential connections and signals needed for operation.

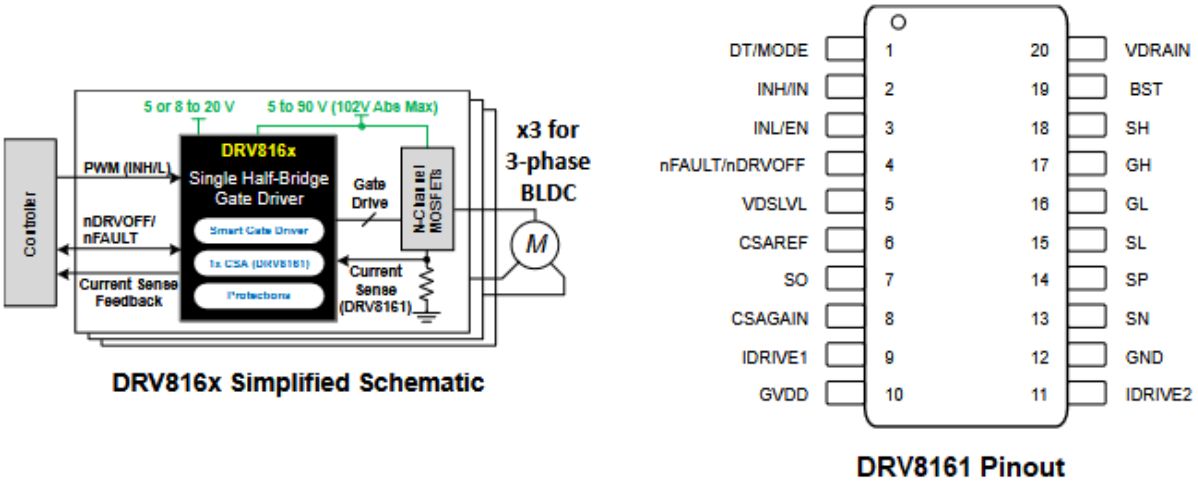


Figure 2: DRV8161 Simplified Schematic and Pinout

To understand how the DRV8161 operates, the functionality can be separated into two parts: **Phase Output** and **Hardware Selection and Fault Detection.**

Phase Output

The first key aspect of the DRV8161 is the actual generation of the Phase Signal through the various internal and external circuits of our motor driver. The DRV8161 motor driver operates by receiving a 12V supply voltage (GVDD) to power the chip. A PWM input signal is provided, where the high and low signals (INH/INL) control the gate driver. The input PWM signal interacts with the Dead Time/Mode (DT/MODE) configuration, determining the specific mode of operation for the driver. Based on this, the chip generates the necessary output signals: Bootstrap (BST), High-Side Gate (GH), High-Side Source (SH), Low-Side Gate (GL), and Low-Side Source (SL). These signals are then fed into the N-Channel MOSFET circuits, which are responsible for switching and ultimately creating the motor's phase signals.

Additionally, the DRV8161 includes an integrated current sense amplifier to monitor the current flow through the MOSFETs. This feature provides an added layer of protection, helping to prevent overcurrent situations that could damage the MOSFETs or other components in the system. Through this sequence, the DRV8161 efficiently drives the motor while ensuring safe operation under varying load conditions.

Hardware Selection and Fault Detection

The second key aspect of the DRV8161's operation is its integrated protection and hardware selection features. The chip allows users to configure various hardware settings through external resistors connected to specific pins. These pins include IDRIVE1, IDRIVE2, VDSLVL, and DT/MODE, each offering different levels of control for fault detection and functionality. The desired settings are selected by adjusting the resistors in the circuit around the DRV8161.

In addition to these hardware selection options, the DRV8161 is equipped with several internal fault detection mechanisms, which are communicated through the nFAULT pin. This pin is pulled low during a fault condition, indicating issues such as GVDD undervoltage, VDS overcurrent, or thermal shutdown. With the help of a microcontroller monitoring these fault signals, the system can protect itself under these conditions. When a fault is detected, the chip disables the gate driver outputs, turning the external MOSFETs off to prevent further damage. Normal operation resumes only once the fault condition is cleared.

To better understand how the DRV8161 integrates into a system, let's now explore the PCB design process required to drive a 3-phase BLDC motor and enable its operation.

2.2.2.2. PCB Design and Creation

Understanding the Signals and Circuits Needed for Movement

To design the necessary PCBs for driving the motor, it is crucial to first understand the signals required for operating the 3-phase BLDC motor. The initial step involves identifying the motor's pinout, which defines the signals needed for proper motor control. In this case, the focus is on the NEMA 23 Hudston Servo Motor and its corresponding Molex Mini-Fit Jr. connector. This pinout provides key signal connections, such as phase signals and communication pins, that are essential for motor movement.

Pin #	AWG	Signal Name	Pin #	AWG	Signal Name
1	16	P DRAIN	2	-	NO CONNECT
3	26	COMM S-T	4	26	COMM R-S
5	26	COMM T-R	6	26	E DRAIN
7	26	GND	8	26	ENC A~
9	**16**	PHASE R	**10**	**16**	PHASE S
11	**16**	PHASE T	12	26	+5VDC IN
13	26	ENC I	14	26	ENC B
15	26	ENC A	16	26	ENC B~

Table 4:Pinout for the Molex Mini-Fit Jr. Pinout (Motor Pinout)

Once the motor's pinout is understood, the next step is to examine the motor driver's role in generating the required control signals. This process involves utilizing the DRV8161 Evaluation Module (EVM) paired with the LAUNCHXL-F280049C development kit. This hardware combination, along with an external GUI on a laptop, facilitates the generation of PWM signals to drive the motor while also providing fault detection and diagnostics via built-in LEDs. Figure 3 below highlights the key areas involved in the operation of the EVM alongside the GUI.

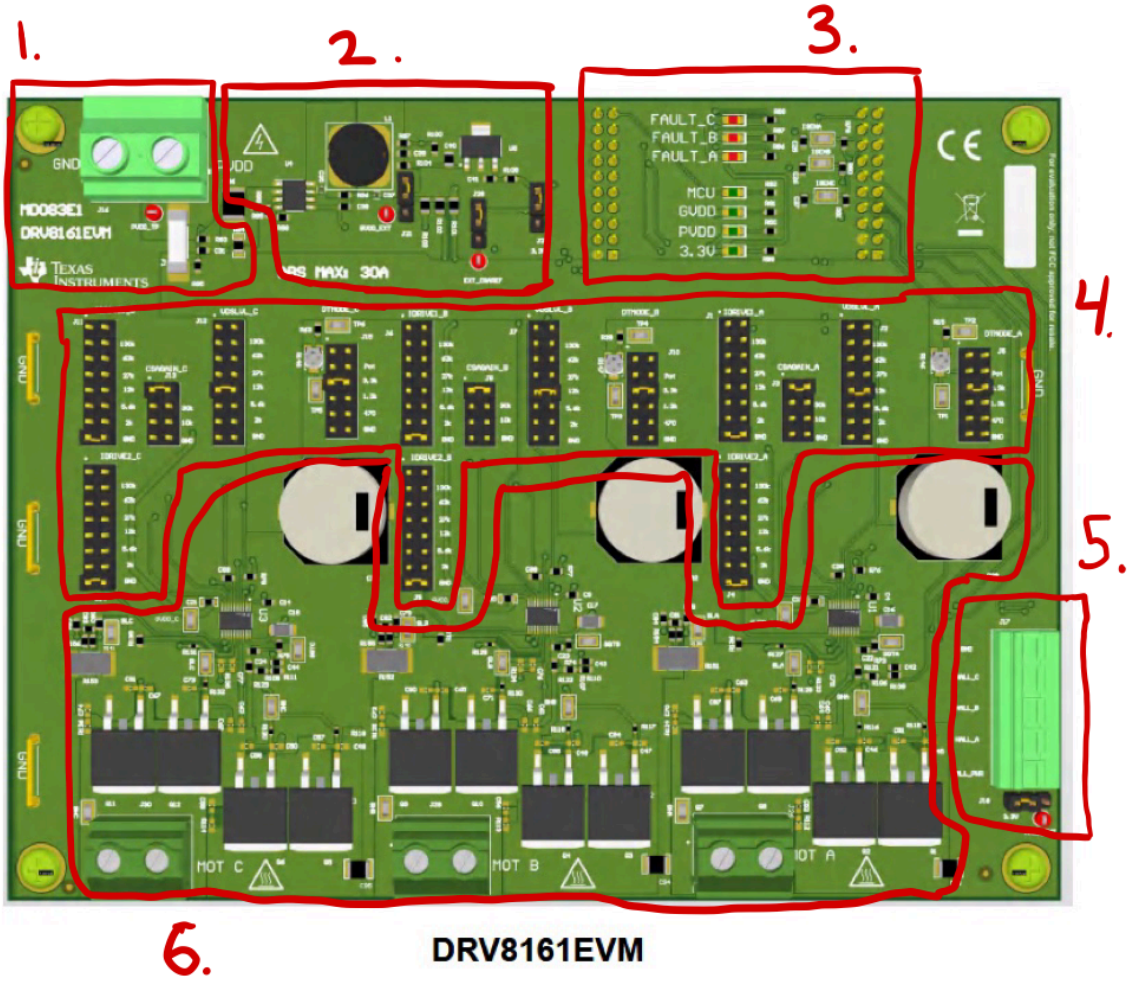


Figure 3: DRV8161 Evaluation Module's Necessary Circuits and Signals

1. **PVDD/GND Input**: The power supply input and ground for the entire board, with an input voltage range of 4.5V to 48V and a maximum current rating of 30A.
2. **12V and 3.3V Step-Down Circuits**: These circuits step down the input voltage to provide 12V GVDD for the motor driver and 3.3V for the CSAREF and MCU voltage requirements.
3. **Launchpad Integration and Fault/Status LEDs**: The LAUNCHXL-F280049C is mounted beneath the EVM and connected to a laptop via MicroUSB. The laptop interfaces with the GUI, which controls the MCU/EVM code and provides real-time fault and status monitoring via built-in LEDs.

4. **Hardware Configuration Jumpers**: Resistor jumpers allow for customization of the IDRIVE, CSA gain, VDSLVL, and dead time mode settings for each of the three DRV8161 motor drivers.
5. **Hall Sensor Inputs**: This feature takes input from an external Hall sensor in the motor, adjusting the phase outputs accordingly. These adjustments are made through the GUI for precise control.
6. **Motor Driver Circuits**: The EVM contains three identical motor driver circuits, each driving the phase signals generated by the DRV8161. Large terminals are provided at the bottom for routing the phase signal outputs.

Finally, we can categorize the critical and non-critical signals and circuits involved in driving the NEMA 23 motor using the DRV8161. **Critical signals and circuits** are essential for generating the phase outputs required by the motor drivers, such as the input voltages and Launchpad connections. **Non-critical signals and circuits**, on the other hand, are not strictly necessary for phase signal generation and can be configured to a fixed value, allowing for greater flexibility in PCB design. For example, the hardware selection signals can be set to a single, fixed value and do not need to be dynamically adjustable.

Critical Signals and Circuits:

- **PVDD/GND Input:** Supply power and GND to board. Used by almost everything on the board.
- **Launchpad Connection:** Launchpad connection jumpers to route signals generated by the GUI and the Launchpad to/from the Motor Drivers.
- **Hall Sensor Inputs:** Critical feedback mechanisms from the motor, allowing for accurate phase output adjustments.
- **Motor Phase Outputs:** The control phase signals generated by the DRV8161 Motor Driver, which control the rotation of the motor.
- **Step-Down Circuits:** 12V, 5V, and 3.3V step-down circuits are needed because of different components on the board.
- **Encoder Inputs:** Another critical feedback mechanism from the motor, allowing for precise control of movement from the motor.

Non-Critical Signals and Circuits:

- **Hardware Selection Jumpers (IDRIVE, CSA Gain, VDSLVL, DT Mode):** These signals are configurable through resistor combinations, but can be set to a fixed mode, eliminating the need for a dynamic adjustment jumper circuit.
- **Fault and Status LEDs:** Although non-critical, it is always helpful to have Fault and Status LEDs to help diagnose problems. Provides visual feedback from the MCU/Motor Driver to see faults.

With a clear understanding of the critical and non-critical signals required to drive the NEMA 23 motor using the DRV8161, we can now transition to the implementation of the system of three PCBs that were made to generate motor movement.

Integrated PCB Design for Motor Operation

The design and development of the PCBs were heavily inspired by the DRV8161 Evaluation Module, which provided a strong foundation for implementing the motor driver circuits. Many schematics, layout designs, and component footprint libraries required for the motor driver were readily available through Texas Instruments, streamlining the design process. Additionally, the voltage step-down circuits, an essential part of the PCBs, were adapted from schematics provided by Emily Hamsa, who is responsible for the Power Subsystem. Key elements such as the MOSFET layout and driver circuit designs drew direct inspiration from the DRV8161 EVM, ensuring both functionality and reliability. Below are some examples of the referenced schematics and layouts that influenced the PCB design.

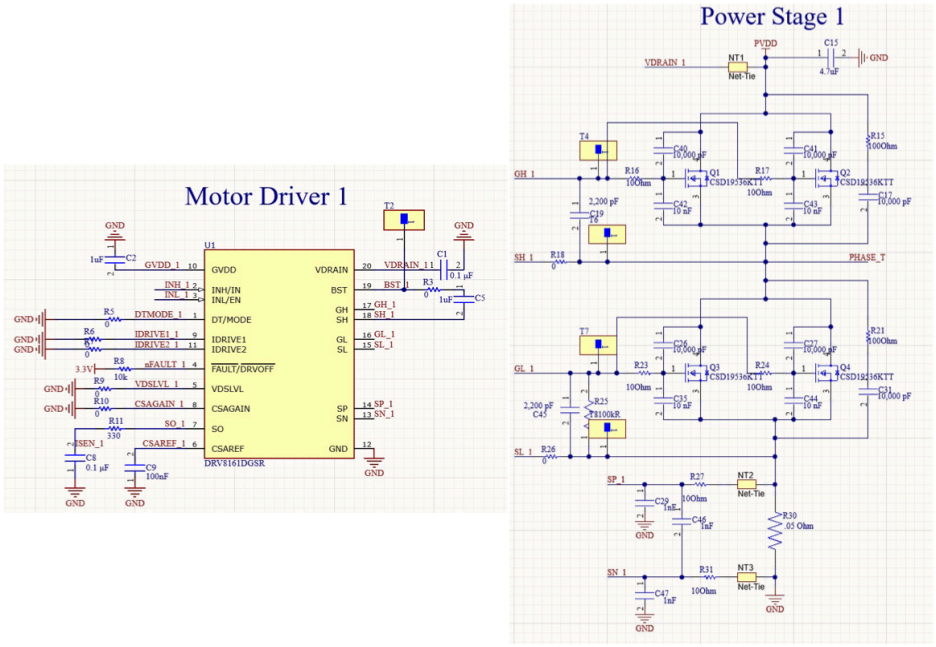


Figure 4: Motor Driver Schematics (Chip and Power Stage (Phase))

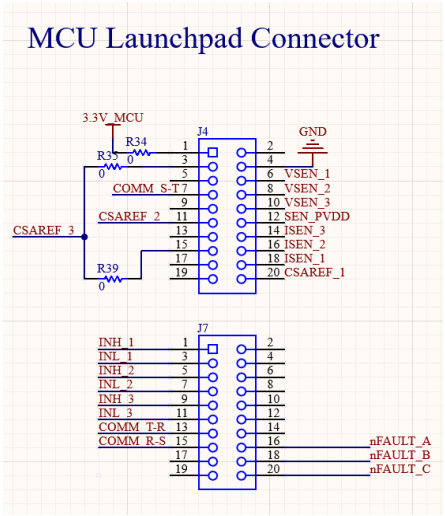


Figure 5: MCU Launchpad Connector Schematic

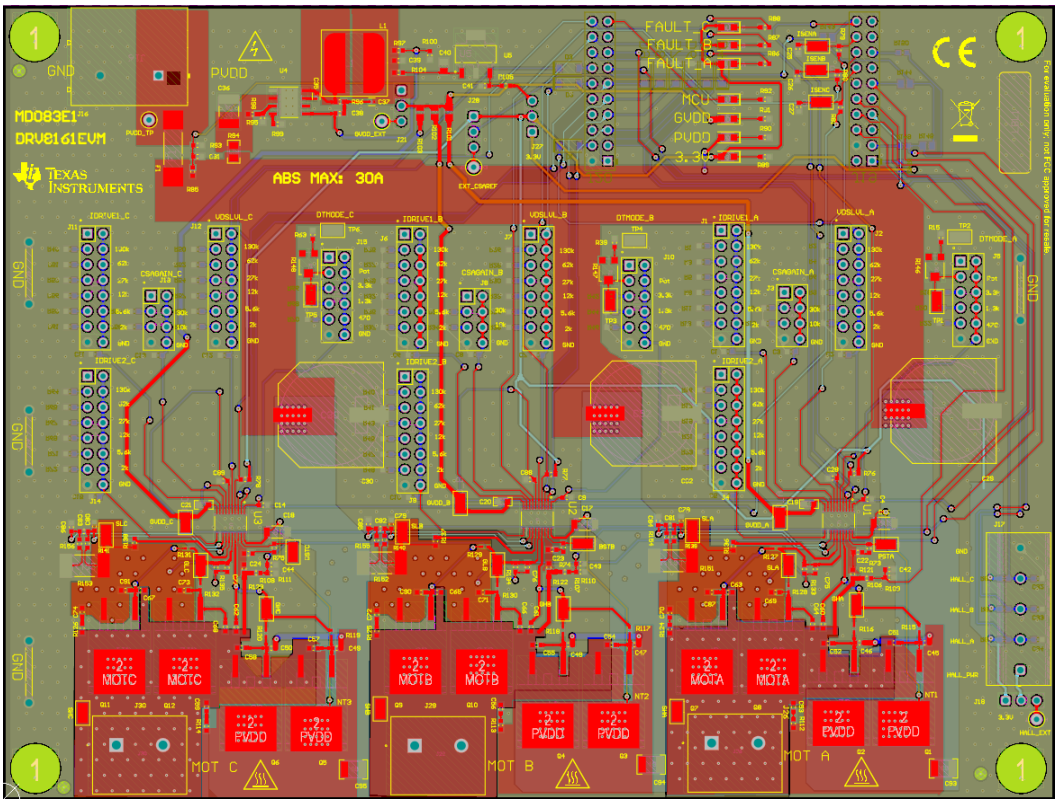


Figure 6: Reference DRV8161 EVM PCB Layout

Above in Figure 3 and Figure 6, you can see the Hardware Selection jumpers for the different logic levels in the DRV8161 Motor Driver. Since this is a non-critical circuit in the creation of the phase signal, our boards were able to be simplified by setting each selection to its ‘Level 0’ logic level by having a 0Ω resistor for each selection. By setting the selection levels for each signal to its base value, we ensured that there would be no unexpected complications during operation and could omit the large circuitry needed for the resistance jumper selection system.

To finally generate motor movement using three DRV8161 Motor Drivers, a system of three PCB boards was developed: **Board A, Board B, and Board C**. Each board corresponds to one of the three phase signals required by the motor and produced by the motor drivers. Among these, Board A serves as the ‘Main Board,’ as it houses additional circuits necessary for the motor's operation. The other two boards, Board B and Board C, focus primarily on driving their respective phases. Below is a breakdown of each board, highlighting its key functionalities and the circuits it contains.

Board A

Board A serves as the ‘Main Board’ since it will be the main powerhouse of the three PCBs. It will serve as the board that the Launchpad and the motor will plug into and run off of. It will have output terminals for each of the step-down voltage circuits on the board, terminals for motor feedback functions, and terminals to input Motor Driver phase outputs from Boards B and C. Below are the key functions that are on the board:

Key Functions: **(Numbered in Figure 7 Below)**

- Take PVDD/GND Input (48V, 30A Input) **(1)**
- Motor Driver Circuit (Phase Generation) **(2)**
- Voltage Step-Down Circuits (Schematics from Emily Hamsa)
  - 48V PVDD -> 12V GVDD **(3a)**
  - 48V PVDD -> 5V Hall/Encoder **(3b)**
  - 48V PVDD -> 3.3V MCU **(3c)**
- Input Terminals for Phase B & C **(4)**
- LAUNCHXL-F280049C Jumper Connector **(5)**
- Input/Output Terminals for Encoder/Hall Sensor **(6)**
- Output Terminals for 12V, 5V, and 3.3V **(7)**
- Motor Connector Mating Terminal **(8)**

Board A was designed with the goal of isolating the ability to produce a single phase signal, making it straightforward to replicate this functionality across two additional boards—Board B and Board C. This modular approach simplifies the design process and ensures consistency between phases. Below is the completed build of Board A, labeled with its corresponding key functions as outlined above.

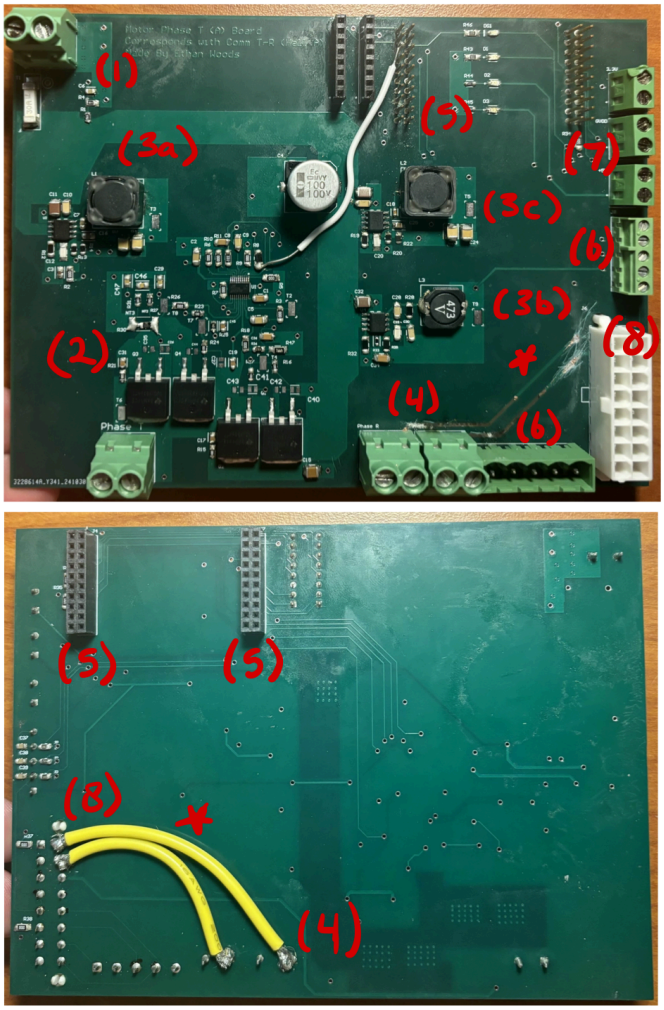


Figure 7: Front and Back Implementation of the 'Main Board', Board A

Boards B & C

Boards B and C complete the system by generating the final phase signals required to drive motor movement. These boards were intentionally designed for simplicity, relying solely on the signals provided by Board A to produce their respective phase outputs. Connections between Board A and Boards B and C are made via jumper wires, ensuring seamless communication. Once all necessary inputs are received, Boards B and C generate their corresponding phase signals, which are then routed back to Board A. Below are the key functions that are on the boards:

Key Functions: **(Numbered in Figure 8 and 9 Below)**

- PVDD/GND Input (48V, 30A Input) **(1)**
- Input Terminals for 12V and 3.3V **(2)**
- Jumper Terminals for Signals from Board A **(3)**
- Motor Driver Circuit (Phase Generation) **(4)**
- Phase Output Terminal **(5)**

Boards B and C are dedicated to generating their respective phase signals, which are then routed back to Board A for integration. This modular design allows each phase signal to be created independently, simplifying the overall system and ensuring flexibility in the design. Below are the finalized builds of Board B and Board C, with their key functions labeled as outlined above.

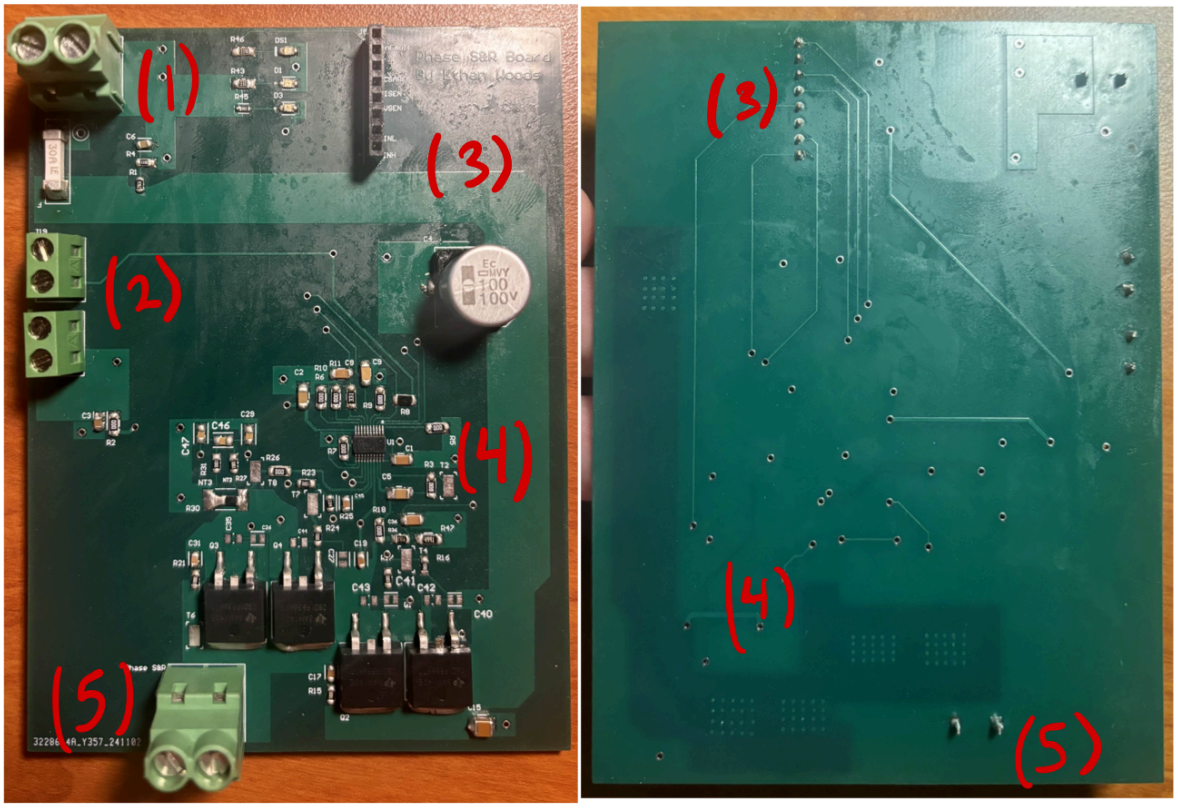


Figure 8: Front and Back Implementation of Board B

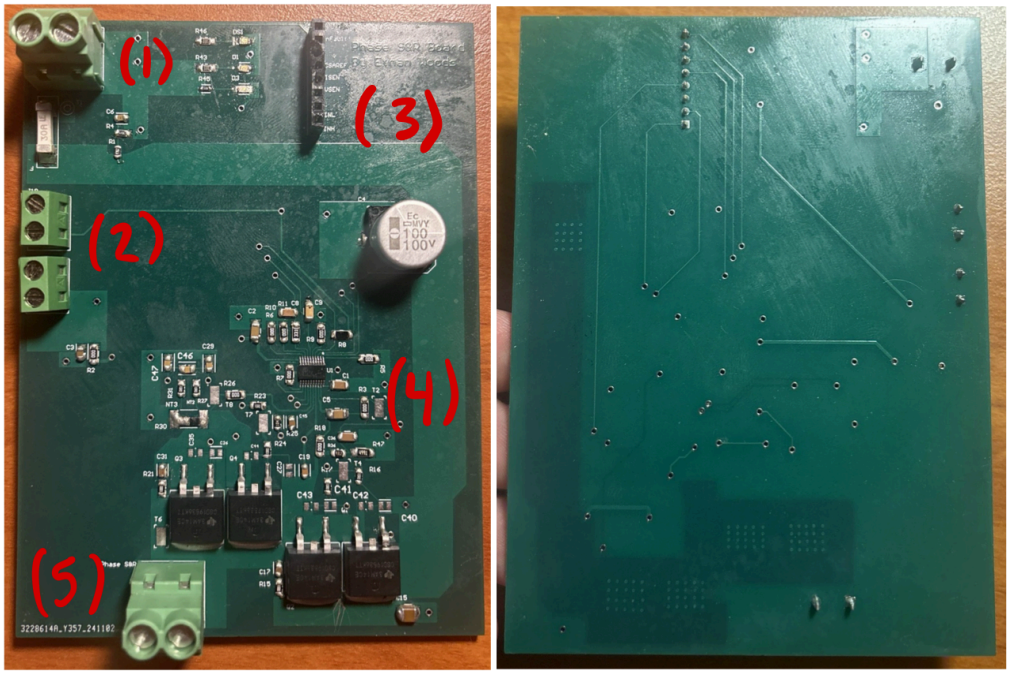


Figure 9: Front and Back Implementation of Board C

With all three boards assembled and interconnected, the system is now ready to drive the motor. By connecting the Launchpad to the designated connector on Board A and linking it to a laptop, we can utilize the GUI software provided by Texas Instruments to control the setup. This GUI mirrors the functionality of the DRV8161 EVM, enabling the user to adjust parameters such as duty cycle and acceleration delay with ease. Through the GUI, the output of our PCB system can be enabled, allowing the generation of the phase signals required to drive the motor. Figure 10 shows the GUI and its configuration.

With the hardware and control system in place, we now move into **Testing and Results**, where we evaluate the performance and functionality of the PCBs in driving the motor.

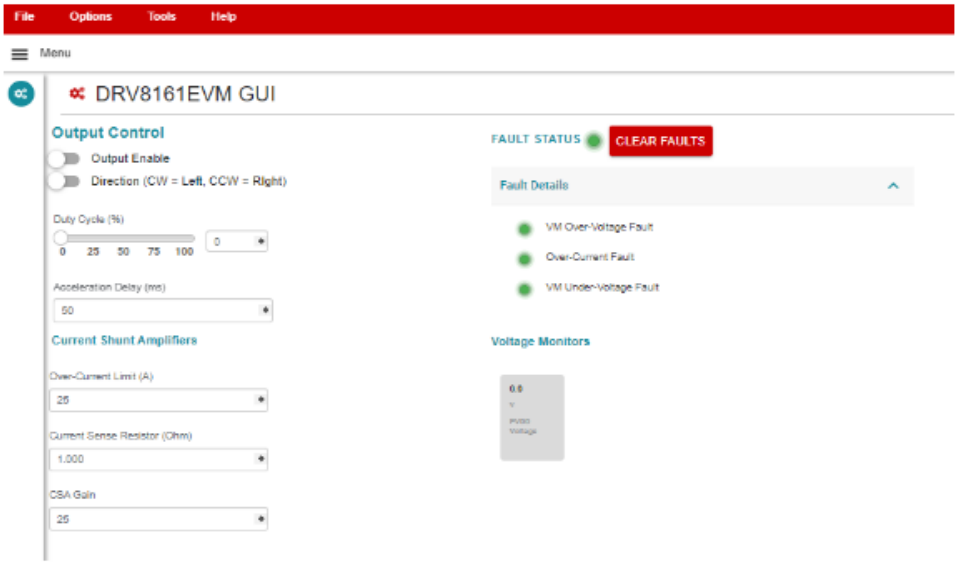


Figure 10: Texas Instruments DRV8161 Evaluation Module GUI

2.2.2.3. Testing and Results

To begin testing each board, they were powered using a 48V 25A power supply. On power-up, the corresponding LEDs on Board A illuminated as expected, confirming proper functionality. However, on Boards B and C, some LEDs were found to be installed backward, preventing them from lighting even when power was applied. Despite this, terminal testing with a multimeter confirmed that all PVDD and GND connections were functioning correctly, ensuring the boards were receiving power.

With power delivery confirmed, the next step was to validate the functionality of the DRV8161 EVM as a baseline for comparison. The EVM was powered using the same 48V 25A power supply and connected to the GUI through the LAUNCHXL-F280049C Launchpad. This setup allowed us to simulate identical circuit conditions to those in our PCB system and measure the phase output voltages generated by the EVM.

To verify the performance of our boards, the phase output voltages from the DRV8161 EVM were measured and used as reference values. By comparing these outputs with the corresponding phase signals from our boards, we could validate whether the PCB system was producing accurate signals needed for motor operation.

Below is the process for connecting and measuring the outputs from the DRV8161 EVM, followed by images of the measured phase signals for comparison.

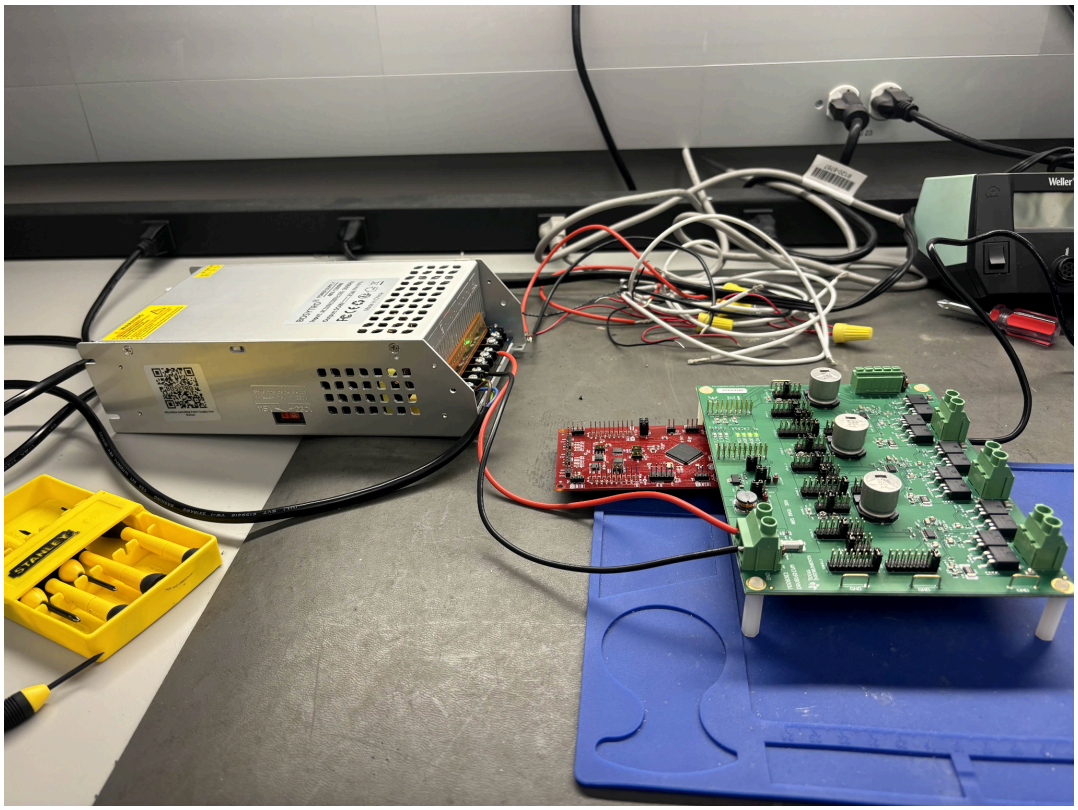


Figure 11: DRV8161 EVM Connected to Power Supply and Launchpad

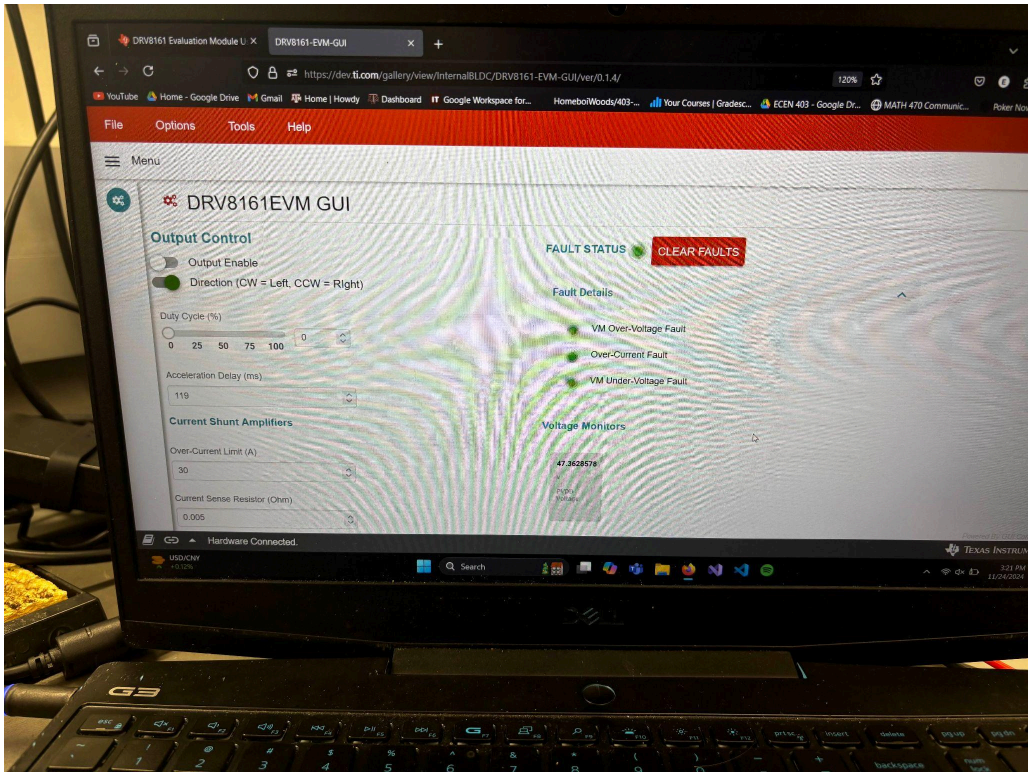


Figure 12: GUI Interface Showing Connection to 48V PVDD with Output Disabled

Once connected, the DRV8161 EVM is successfully linked to the Launchpad and ready to enable its output to produce the three-phase signals required for motor operation. However, while the EVM can generate these signals, we are unable to use it to spin a motor directly due to the design of Board A.

Although Board A includes terminals capable of routing all three-phase signals to the motor connector, this setup is incompatible with the motor driver already present on Board A. Since the motor driver circuit is powered through the board's power supply, it interferes with the output at one of the terminals used to route a phase signal. As a result, the motor cannot be properly connected or controlled via Board A when paired with the EVM.

Instead, validation is performed by comparing the phase output voltages at the EVM's phase terminals with those generated by our PCBs. By measuring and analyzing these outputs, we can confirm the functionality of our design. Below are the results of measuring the phase output terminals for each respective phase.

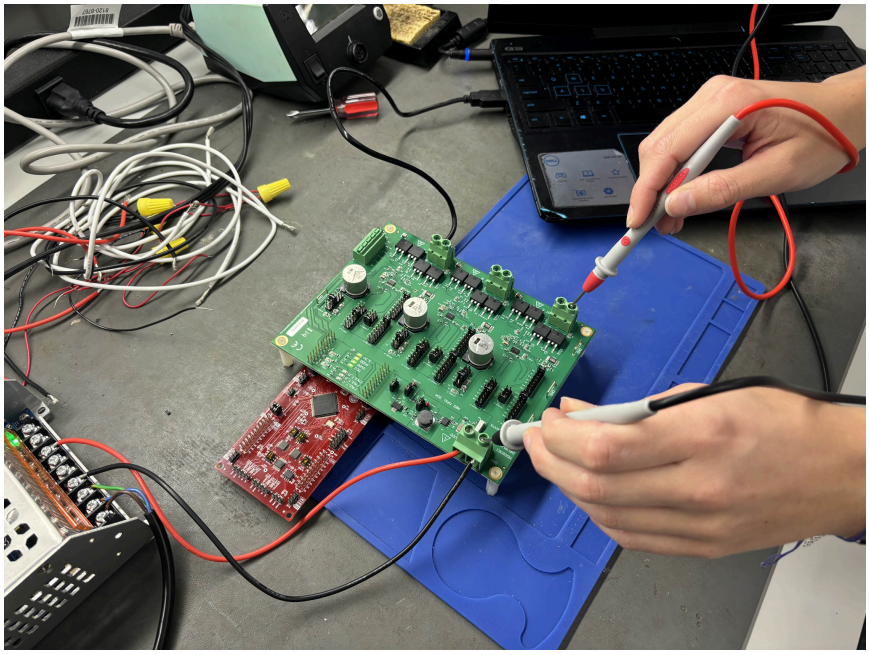
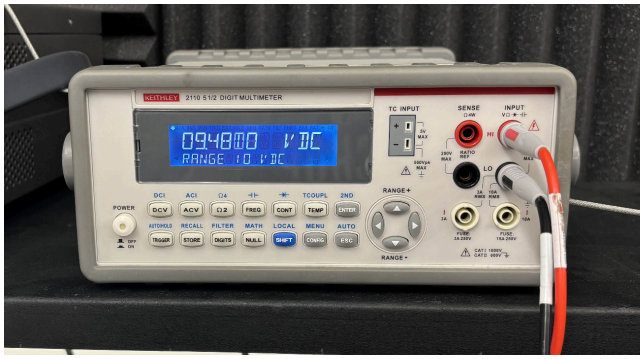
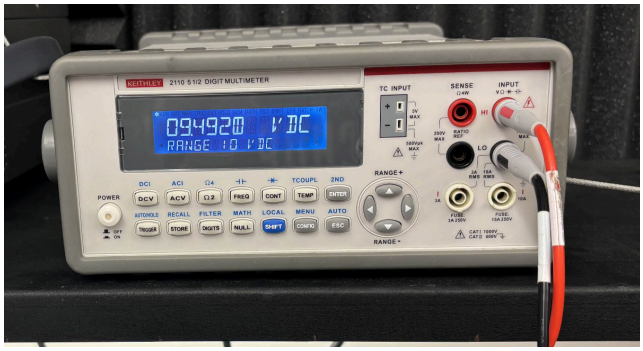


Figure 13: Measuring the Phase Output Terminal on the DRV8161 EVM





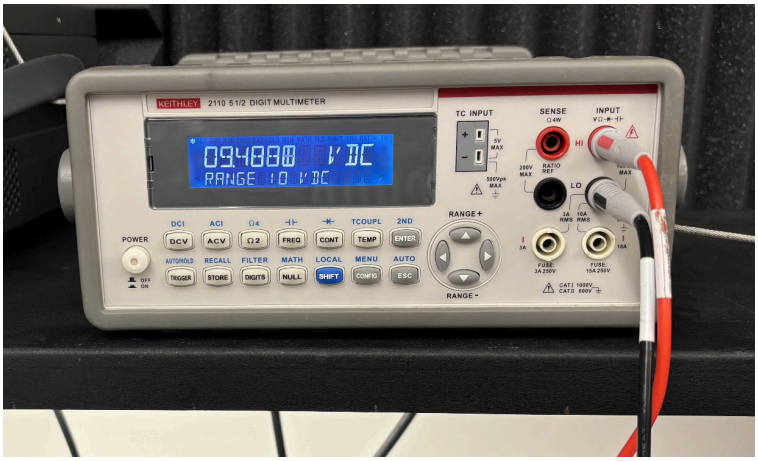


Figure 14: Phase Voltage Outputs on Terminals A, B, and C (Respectively)

As seen in Figure 14, the three-phase output terminals of the DRV8161 EVM measure approximately 9.48 Volts. This voltage serves as the benchmark for validating the outputs of our PCB system. To begin testing the three PCBs, the boards are connected as shown in Figure 15 below.

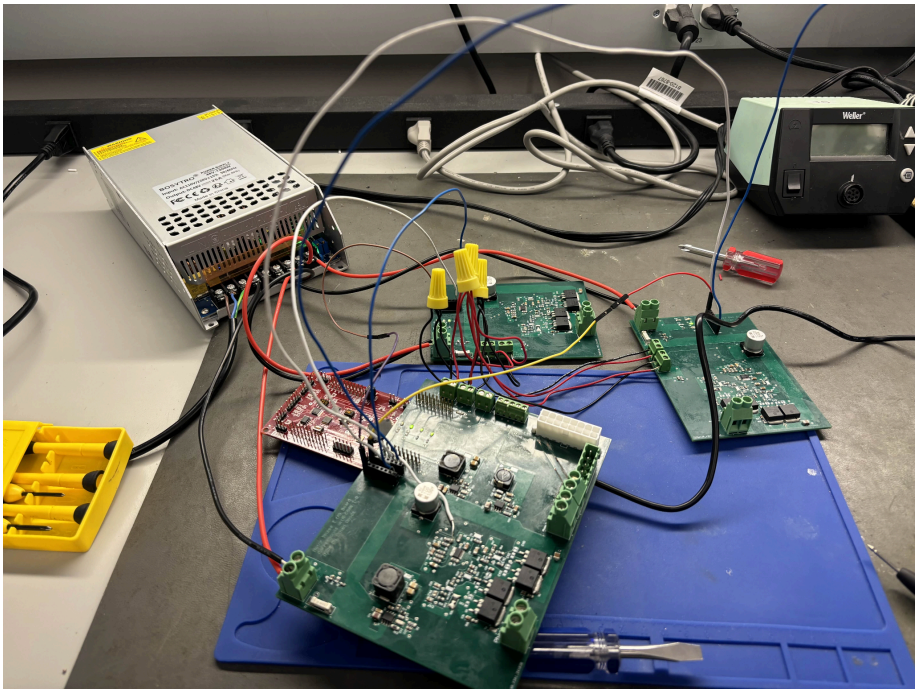


Figure 15: Created PCBs All Wired Together Without Motor and Phases Connected

In this setup, all the boards are powered by the 48V power supply, and the necessary input signals for generating the phases are routed accordingly. However, as depicted in Figure 15, the motor and phase signal connections to Board A are not included in this particular configuration.

Earlier in the day, during an initial test with the motor and phase wires connected, enabling the output via the GUI resulted in a trace on Board A burning out instantly. Despite debugging and reattempting, the same issue occurred a second time. The damaged trace was subsequently repaired with a replacement wire, as shown in the previously referenced Figure 7 (denoted by the star).

Given this outcome, testing now focuses on measuring the voltage outputs at the phase terminals and verifying the functionality of the voltage step-down circuits. The results of these measurements are detailed in the following figures:

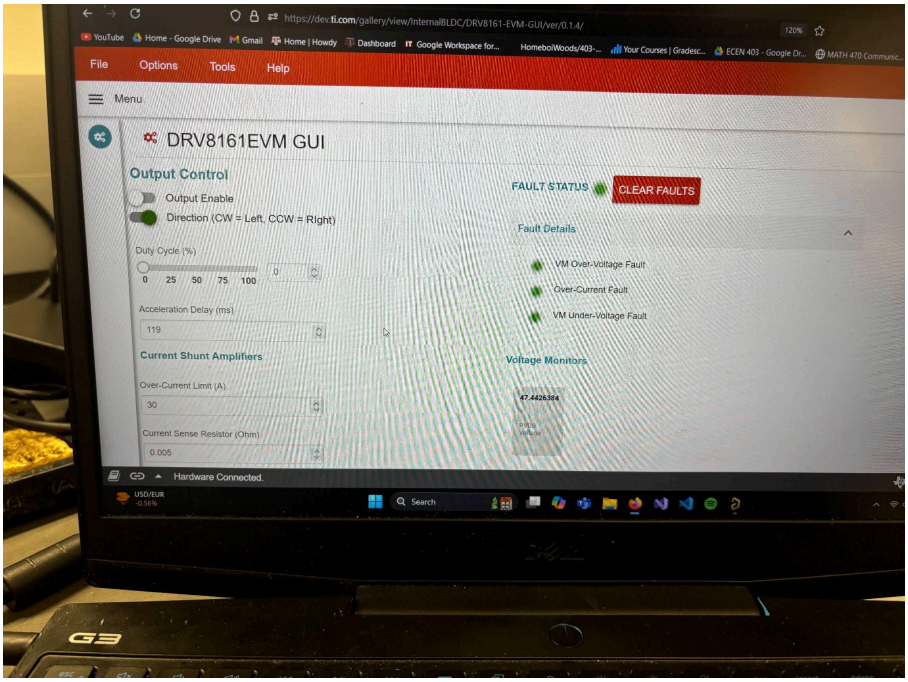


Figure 16: Successful Connection to the GUI Showing 48V PVDD

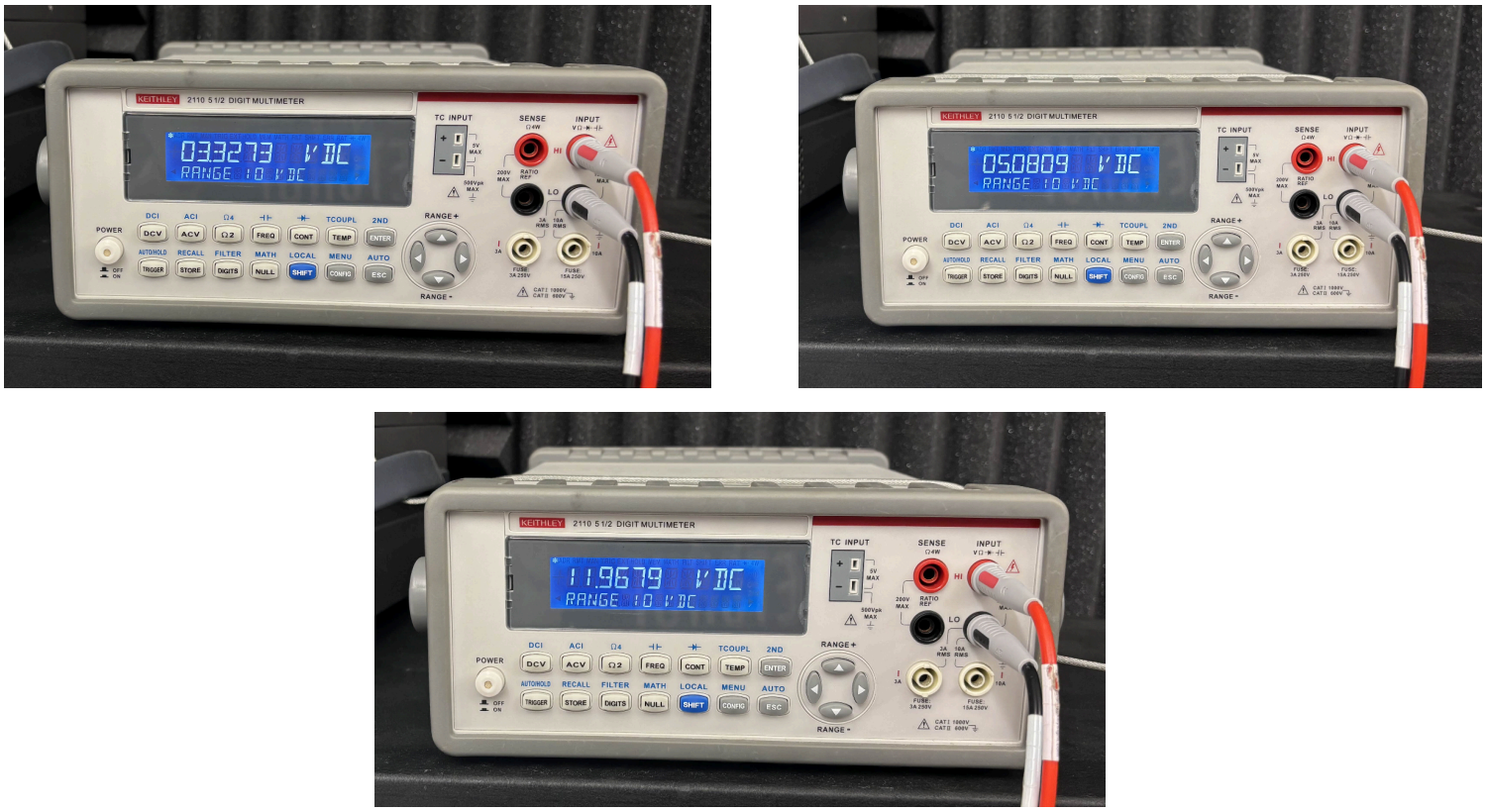


Figure 17: Output of Each Voltage Step-Down Circuit (3.3V, 5V, and 12V Respectively)

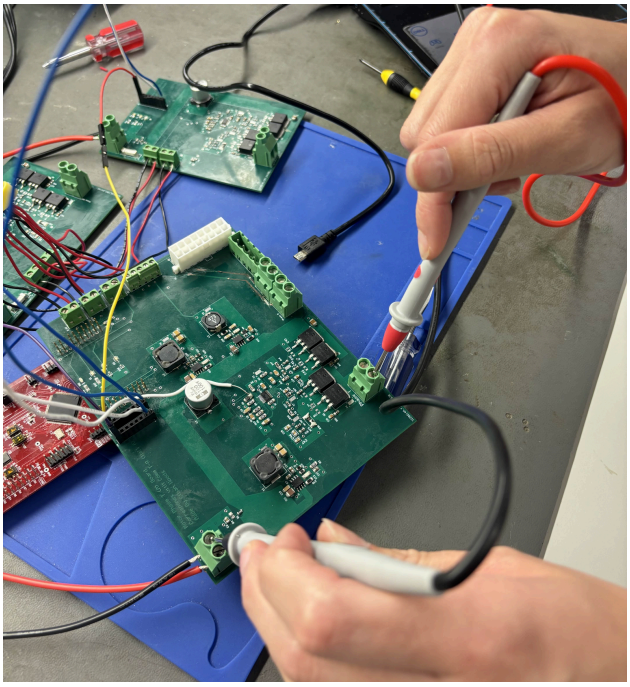


Figure 18: Measuring the Phase Output Terminal on Phase A Board (Same for B and C)

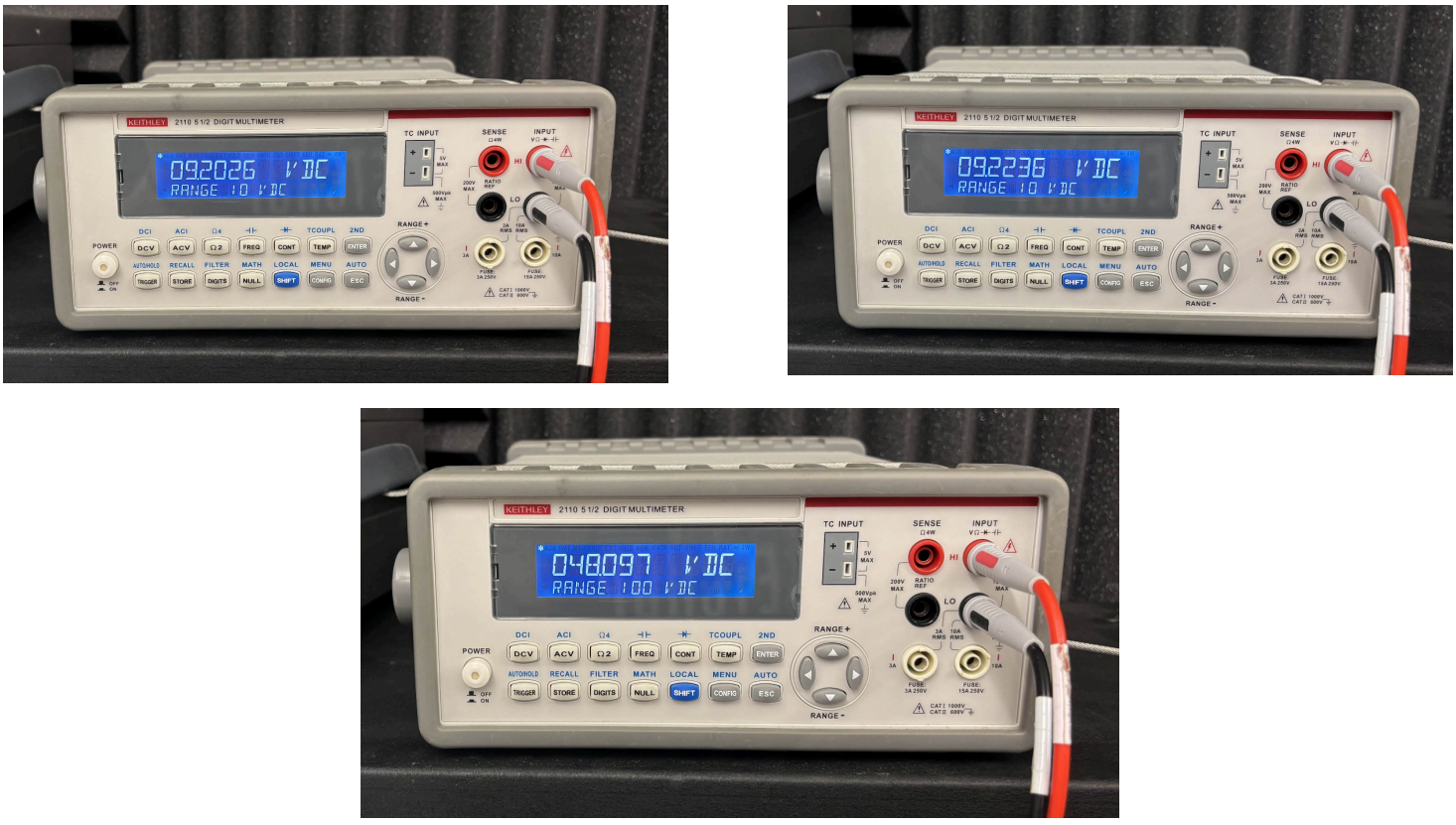


Figure 19: Phase Voltage Output on Each Board (A, B, C) Respectively

In conclusion, while the testing of the PCB system did not achieve the desired outcome of motor movement, significant progress was made in understanding and isolating the issues. The measurement results confirmed that two of the phase terminals (A and B) were functioning as expected, producing voltages near 9.48V. However, the issue with Board C, where one of the terminals was producing an unexpected 48V, highlighted a short to the input voltage. This has been traced back to a potential fault with the DRV8161 chip, which will be further investigated.

Given the importance of having all three motor driver circuits fully functional, Board C's failure prevents the successful operation of the 3-phase BLDC motor. However, with plans for continued work over the winter break, efforts will focus on reworking Board C and potentially creating an additional board to ensure proper motor operation. Though this semester's testing did not achieve the final goal, it laid the critical groundwork needed for future progress.

Looking ahead, further testing and refinement will be necessary to achieve full motor movement and integration with other subsystems in ECEN 404. The experience gained this semester will prove invaluable as we continue to improve the design, and the validation of the system's objectives will be a crucial next step in confirming the success of the overall project.

2.3. Subsystem Validation

The ultimate goal of validating this subsystem was to achieve movement in a 3-Phase BLDC Motor using a system of custom PCBs connected to the C2000 Launchpad. While motor movement was not achieved this semester due to a short on one of the boards, the validation focuses on proving that generating one correct phase signal demonstrates the feasibility of producing the other two in the same manner. This reasoning underscores the importance of achieving even partial functionality, as it lays the groundwork for future success.

Given the complexity of this subsystem, involving over 200 components (primarily 0805-sized) across multiple PCBs, there were many opportunities for small errors to propagate and affect overall performance. Each board required meticulous attention to detail, from schematic design to fabrication and testing, to ensure proper operation. Therefore, the validation approach breaks down the subsystem into measurable, incremental goals to assess the progress made and identify areas for improvement.

Below is the validation plan, highlighting the specific criteria used to evaluate each aspect of the subsystem and how it aligns with the project's objectives:

Test Deliverable	Achieved Goal?	Why/How?	Supporting Figure(s)
Schematic and PCB Design	Yes	Created system of 3 PCBs to work to spin a motor	Figures 7, 8, 9
PWM Phase Output from Motor Drivers	Yes	Achieved an output voltage similar to our benchmark voltage on the EVM	Figure 19
Motor Movement	No	Short on Board C prevented complete integration and motor movement	N/A
Precise Motor Control	No	With no motor movement, no control could be done	N/A
All Motor Movement	No	No motor movement generated	N/A

Table 5: Motor Driver Subsystem Validation Plan Goals

2.4. Subsystem Conclusion

While motor movement remains unachieved this semester, significant groundwork has been laid to make it an attainable goal in the near future. This is a critical step for the Collaborative Robot, as motor-driven movement is central to its functionality. Despite this gap, the process and implementation described show that motor movement is within reach. The priority for the winter break will be to repair or produce a new board to replace the malfunctioning Board C. With a functioning Board C—or an alternative board designed to fulfill its role—motor movement should be straightforward, requiring only connection to the GUI and Launchpad.

Beyond this immediate task, the Motor/Motor Movement subsystem is well-positioned to transition into the next phase of integration. The current approach, using three motor drivers to spin a 3-Phase BLDC motor, will remain the foundation moving forward. However, next semester's design will consolidate the motor driver circuits onto a single main board, which will simplify wiring and reduce the overall complexity of the system. This main board will interface with a dedicated voltage step-down board from the Power Subsystem, streamlining power distribution. Custom connectors for each motor will also ensure proper compatibility and ease of use.

This brings us to a key consideration: complexity. Creating and testing these initial boards, which only include the minimum functionality to generate phase signals, required designing over 200 components. Scaling this approach to accommodate four additional motors would demand an immense amount of time and effort, which is further compounded by the limited resources and lack of mechanical engineering expertise within the team. As a result, there is growing support for simplifying the collaborative robot into a crane-like design. This shift would reduce the motor count from four to two (plus a claw), cutting the number of required boards in half and allowing the team to focus on optimizing the system's core functions.

Despite these challenges, the subsystem's achievements this semester provide a solid foundation for future work. With phase signals successfully generated and the boards communicating effectively with the MCU subsystem, integration is already underway. Next semester, the focus will include refining the PWM signals needed to drive the motor driver circuits, collaborating with the Power Subsystem to route external voltage levels directly onto the main board, and enhancing interaction with the Wireless Connectivity Subsystem for mobile application control. This forward-looking plan ensures that, with continued effort, the Motor/Motor Movement subsystem will achieve full functionality and play a pivotal role in the Collaborative Robot's success.

3. MCU/Processing Subsystem - Adrian Guzman

3.1. Subsystem Introduction

The MCU/Processing subsystem is designed to receive input signals from the mobile application subsystem and generate PWM waveforms that will be used by the motor driver subsystem to control a motor. This subsystem plays a major role in supporting the goal of achieving a fully functional Collaborative Robot (cobot). The subsystem utilizes the TMS320F280049C microcontroller from Texas Instruments' C2000 family of real-time control MCUs. The power subsystem supplies the necessary power to the MCU to facilitate the completion of tasks. The MCU subsystem has been designed to incorporate compatibility and integration to other subsystems. The required deliverables of this subsystem have been accomplished by adhering to the execution and validation plan. Furthermore, the subsystem was thoroughly tested in the lab to ensure the necessary signals can be received and generated to achieve the end project goal. The following documentation provides a detailed breakdown as to what has been accomplished.

3.2. Subsystem Details

3.2.1 Subsystem Block Diagram

The following figure presents a high level block diagram for the MCU/Processing Subsystem, which is capable of receiving signals from the ESP-32 and generating PWM signals for motor control.

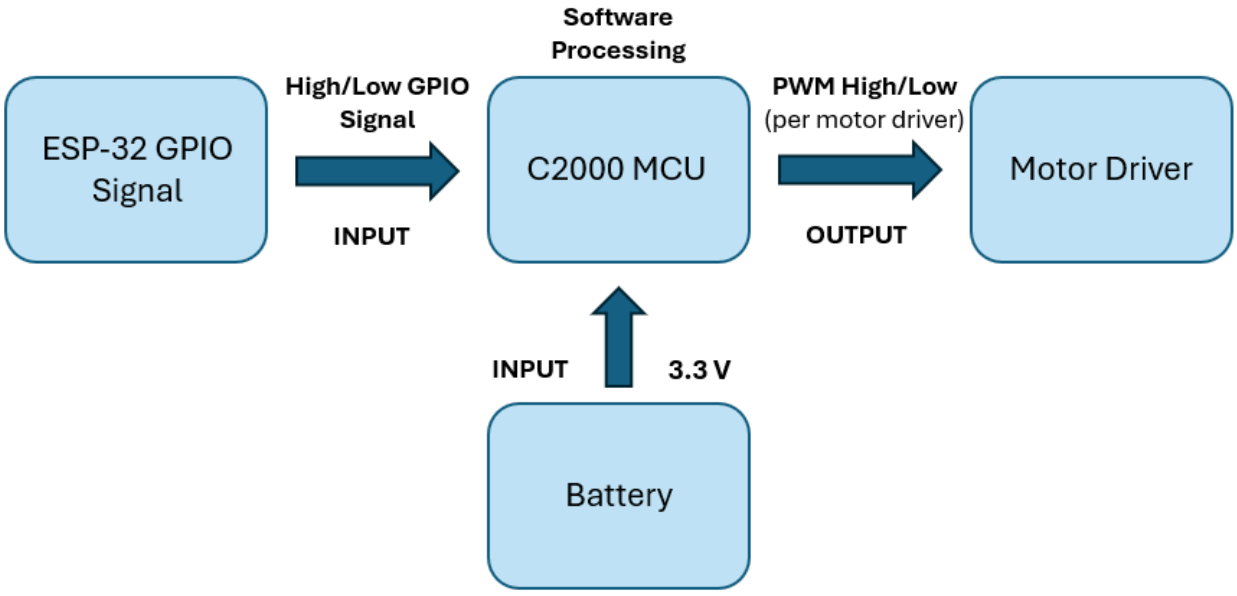


Figure 20: MCU/Processing Block Diagram

The MCU/Processing subsystem utilizes the TMS320F280049C MCU, which features a variety of peripherals such as pulse-width modulation (PWM) for motor control and a large quantity of General-Purpose Input/Output (GPIO) pins. The GPIO pins will support the communication between the C2000 and the ESP-32 within the mobile application subsystem.

The mobile application subsystem includes a mobile app with a game style like controller. When a button is pressed, a GPIO pin on the ESP-32 shall change from low to high. The C2000 will detect this input and generate the necessary PWM signals. These PWM signals will be sent to the DRV816x motor drivers to rotate a Brushless DC motor (BLDC).

3.2.2. PCB Schematic Design

The primary objective for the MCU/Processing subsystem was to design a PCB that enables functionality of the MCU, including power to the MCU, programming via JTAG, generating PWM signals, establishing communication with external devices like the ESP-32, and verifying GPIO functionality. Key features of the MCU include its 3.3V operating voltage, 16 ePWM channels, and 40 GPIO pins.

The PCB was built with reference from the C2000 Launchpad development board, an evaluation board that TI created to incorporate the TMS320F280049C MCU for developmental purposes. The following sections of the document detail the design of the PCB which focuses on GPIO allocation, Power, Programming, and Miscellaneous parts.

3.2.2.1. General-Purpose Input/Output (GPIO)

The figure below details the GPIO allocation within the PCB schematic.

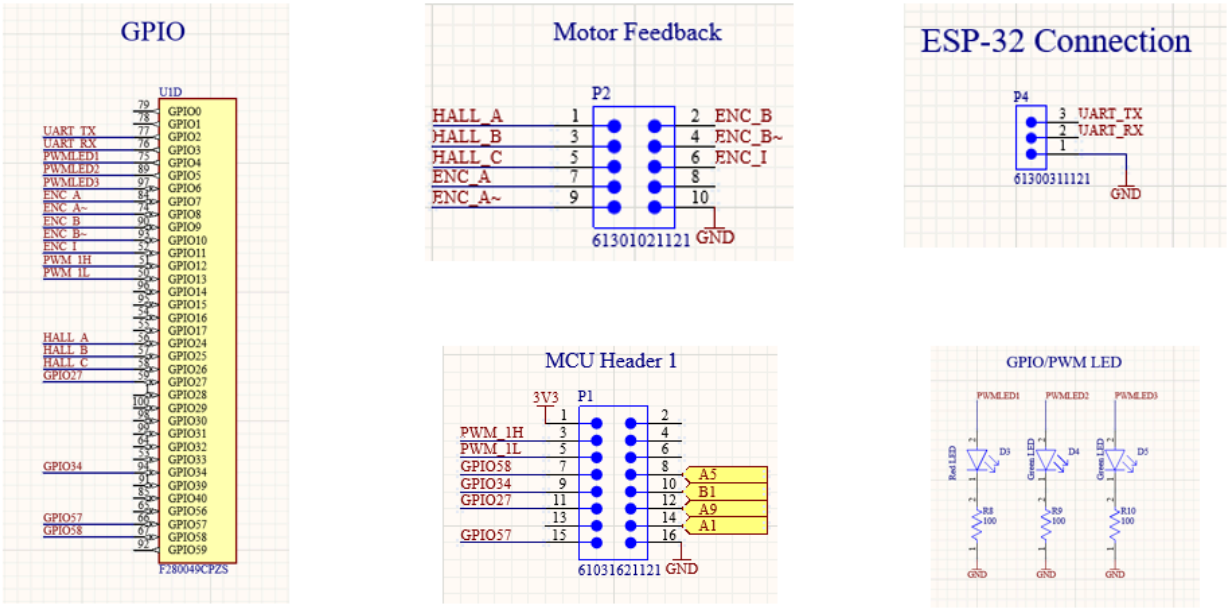


Figure 21: GPIO PCB Schematic

The GPIO pins contain specific features, pins with the ability to generate PWM signals and additional GPIO functionality were considered when deciding what pins to implement to headers. In the next PCB revision, all the GPIO pins from the MCU will be routed to 2.54mm x 2.54mm headers. The first 16x pins (GPIO0-GPIO15) include the 16x ePWM modules and will be reserved solely for PWM generation. The MCU contains 35 accessible GPIO pins, 19 of which are used in the current PCB design. These pins allow testing of the ePWM modules and generic input/output writing to GPIOs.

In the above figure, GPIO pins 2 - 13 were used to develop PWM signals for the motor driver subsystem. Additionally, these pins were used to toggle LEDs, showcasing GPIO functionality and timing, as well as rotate servo and stepper motors. The additional GPIO pins were used to connect the ESP-32 and C2000 to communicate through high/low signal generation.

One BLDC motor requires three motor drivers, one motor driver for each phase of the motor. Each motor driver will receive a high and low PWM signal, a fault check signal, and reference voltage. The below table showcases the necessary signals to be sent from the MCU to three motor drivers. The signals include PWM generation, fault check, and reference voltage.

Pin Number	Signal Name	Description	Purpose
79	GPIO0	ePWM-1 Output A	High PWM
78	GPIO1	ePWM-1 Output B	Low PWM
77	GPIO2	ePWM-2 Output A	High PWM
76	GPIO3	ePWM-2 Output B	Low PWM
75	GPIO4	ePWM-3 Output A	High PWM
89	GPIO5	ePWM-3 Output B	Low PWM
98	GPIO30	General I/O 30	Fault Check
99	GPIO31	General I/O 31	Fault Check
64	GPIO32	General I/O 32	Fault Check
23	A0	DAC-A Output	Reference Voltage
-	3V3	Reference Voltage	Reference Voltage

Table 6: C2000 Pinout (Signals sent to 3x Motor Drivers)

The below table dictates the pinout between the C2000 and ESP-32. Although testing was not performed with these exact GPIO pins, other general-purpose pins were used. This table is to map out the future of what will be required. The purpose of these connections is to establish communication between the ESP-32 and C2000. The ESP-32 will toggle pins high or low based on user input to the mobile application. The C2000 will read input on the GPIO pins below to determine the desired movement from the user and generate the subsequent PWM signals.

Pin Number	Signal Name	Description	Purpose
56	GPIO24	General-Purpose Input Output 24	UP
57	GPIO25	General-Purpose Input Output 25	DOWN
58	GPIO26	General-Purpose Input Output 26	LEFT
59	GPIO27	General-Purpose Input Output 27	RIGHT
1	GPIO28	General-Purpose Input Output 28	OPEN
100	GPIO29	General-Purpose Input Output 29	CLOSE

Table 7: C2000 Pinout for ESP-32 Connectivity

3.2.2.2. Power

This section details the power design for the MCU. The MCU operates at 3.3V, which is supplied through the terminal block and will allow integration with the power subsystem. To maintain stable voltage, decoupling capacitors on the VDD, VDDA, and VDDIO pins were placed with values of 0.1uF, 10uF, and 2.2uF. Additionally, power indicator LEDs were included to display when the system is powered.

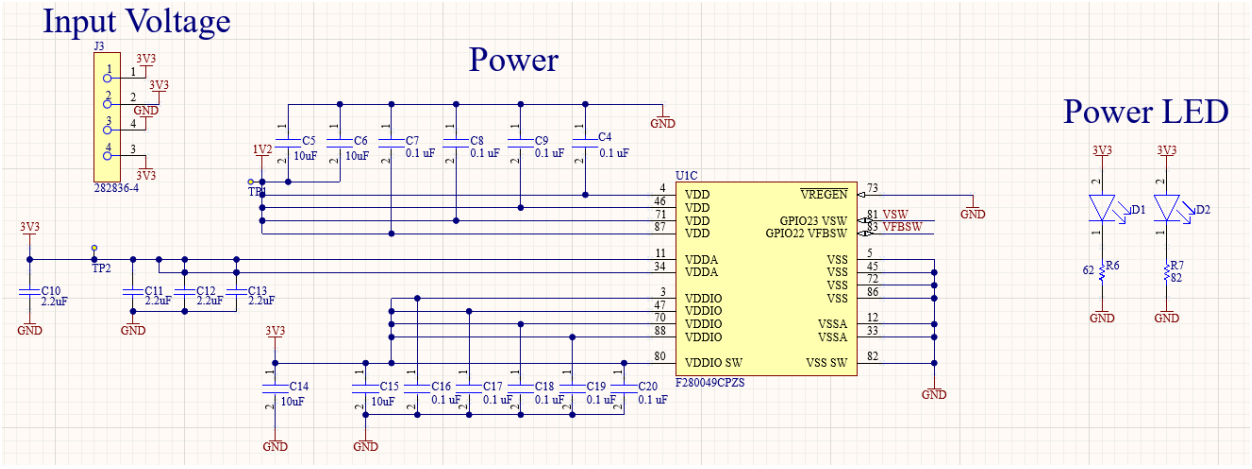


Figure 22: Power PCB Schematic

3.2.2.3. Programming (JTAG)

This section details the schematic design for programming via JTAG. The programming process requires the TMS, TDI, TDO, and TCK signals. Due to sponsor uncertainty and concerns with connectivity to the debug probe, two headers were incorporated into the design: J2 (6 pin 2.54mm x 2.54mm) and J1 (14 pin 1.27mm x 1.27mm). However, J1 was ultimately unused due to connection issues with the XDS-110 debug probe.

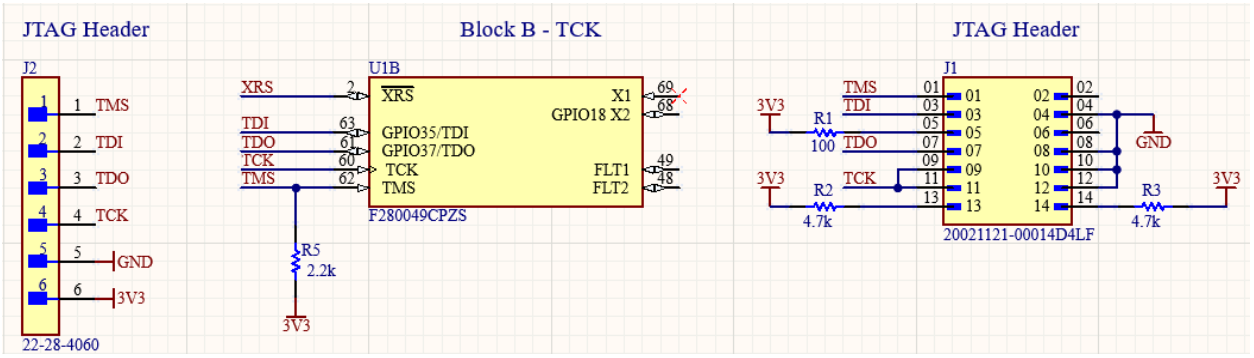


Figure 23: JTAG Programming PCB Schematic

3.2.2.4. Miscellaneous

This section outlines miscellaneous components of the PCB that were ultimately unused. The figure below highlights the ADC block, reset button, launchpad input, VSW input, and launchpad LED input. Throughout the semester, it became clear that these components were unnecessary and will be removed for the next generation design planned for next semester.

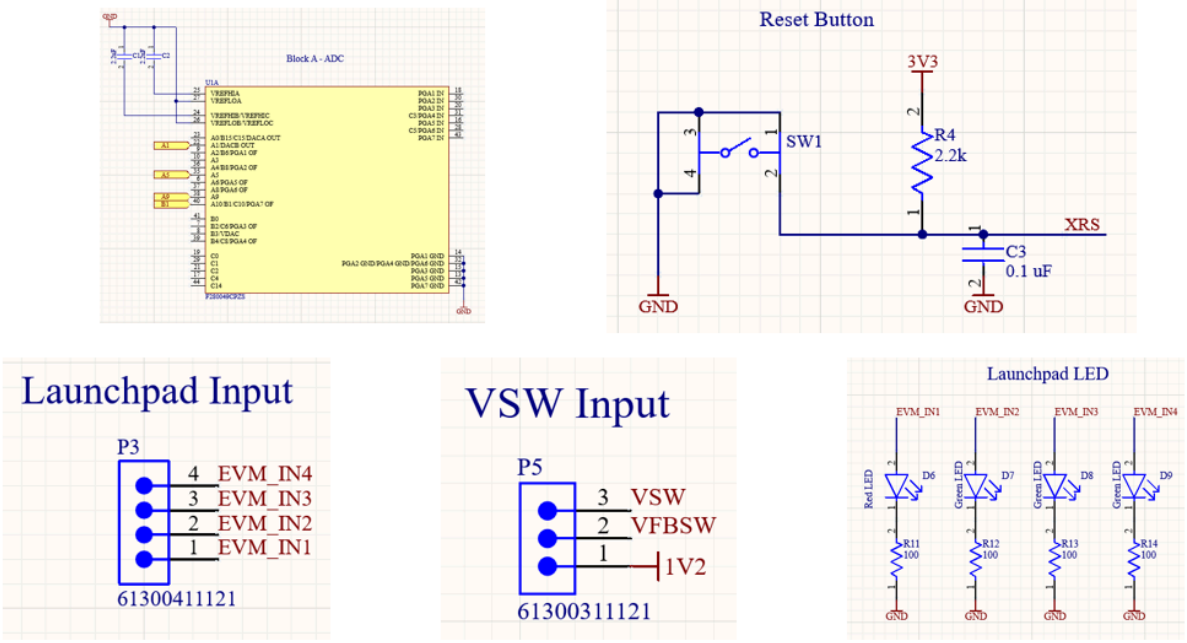


Figure 24: Miscellaneous PCB Schematic

3.2.3. C2000 Launchpad Development

To become familiar with programming and developing on the TMS320F280049C MCU, the evaluation board was ordered from Texas Instruments. This evaluation board is developed to allow testing with all features of the MCU. Additionally, TI provides a software development kit (SDK) known as C2000Ware that includes a variety of example codes to begin development on the launchpad.

To program the launchpad, TI's choice of IDE, Code Composer Studio (CCS) was used. After installation of CCS and C2000Ware, setup of the launchpad was completed by ensuring the correct target device, debug probe, and file configuration was selected.

Using C2000Ware, examples such as "led_ex1_blinky" and "epwm_ex13_up_aq" were used to begin development. LED Blinky allowed control of a GPIO pin to toggle an LED on and off. Additionally, within this source file, additional software was developed to further understand the C2000 MCU. This included a program to spin a MicroServo 9g SG90, a program to spin a step motor 28BYJ-48 with a ULN2003 motor driver, and a program to facilitate communication between an ESP-32 and the C2000 MCU. The ePWM example generated square waveforms for viewing on an oscilloscope. This was successfully outputted for viewing and later on, development of different PWM generation code was written.

The figure below shows the LED Blinky example that was performed on the launchpad. The software writes to GPIO34, sets it high for 1000ms (ON), then sets it low for 1000ms (OFF). The script is in an infinite loop and repeats forever. This example was repeated on the created PCB and demonstrated on demo day.

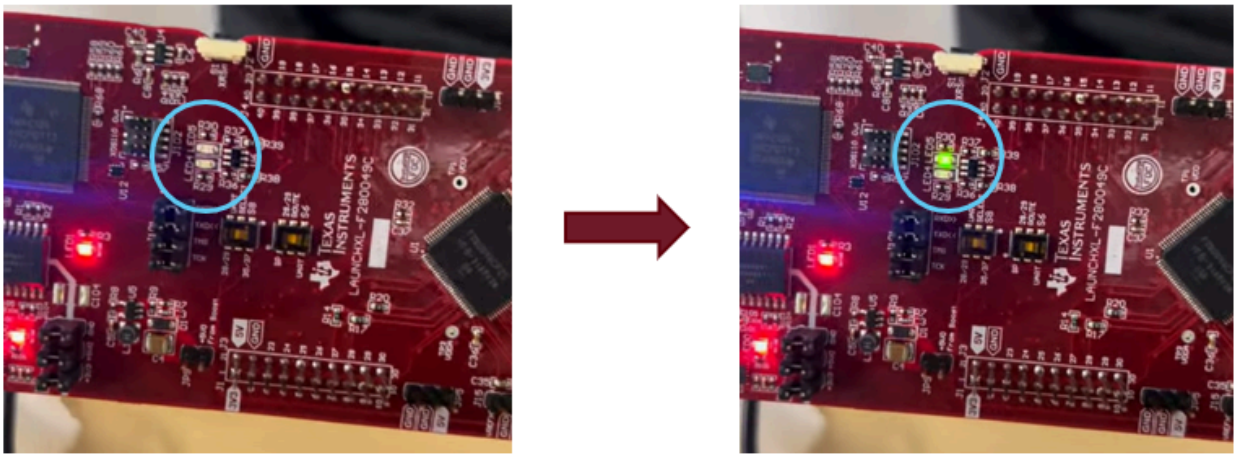


Figure 25: LED Blinky

The figure below details another test performed on the launchpad. A MicroServo was used to understand development with GPIO pins. The MicroServo uses 3 pins: GND, 5V, CONTROL. Using GPIO10 as the control, PWM signals were generated ranging 1000µs to 2000µs to rotate the motor from 0° to 180°.

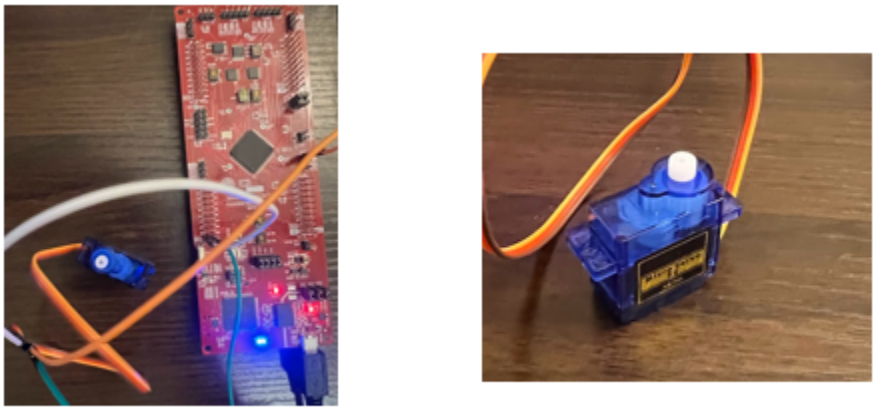


Figure 26: MicroServo 9g SG90

The figure below details a rotating stepper motor example performed on the launchpad. The stepper motor requires four control signals IN1, IN2, IN3, IN4, and 5V. The motor driver supplies IN1, IN2, IN3, IN4, GND, and 5V. The program rotates the stepper motor continuously in one direction through a sequence of GPIO High/Low activations. There are a total of 2038 steps in one full rotation, 90° rotation is shown in the image above. This example was performed on the created PCB and demonstrated on demo day.

Figure 27: Stepper Motor w/ ULN2003 Motor Driver

The figure below demonstrates how the launchpad was able to output PWM signals using ePWM1, ePWM2, ePWM3. The oscilloscope is probing GPIO0 which corresponds to ePWM1A. The above image shows the duty cycle changing from 97% to 23%, this is a gradual change in duty cycle to show the difference between the waveforms over time. This was replicated on the PCB and demonstrated on demo day as well.

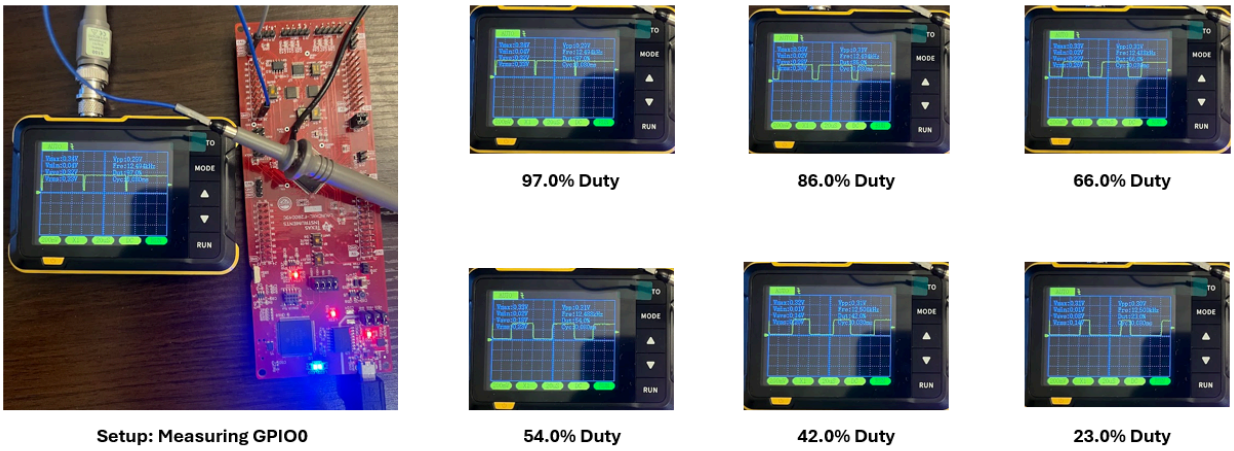


Figure 28: PWM Signal Modulation

Overall, the launchpad was able to provide a foundation of understanding for developing on a C2000 MCU. All the examples performed on the launchpad were replicated on the PCB board that was developed, with most of them being performed at demo.

3.2.4. PCB Testing and Results

The PCB board shown below was created, soldered, and assembled. Throughout the assembly process, the board was debugged to ensure traces were properly routed and correct voltages were delivered. The switch in the top right corner was removed and a trace between GND and SW1 cut as the button was shorting to ground due to an error in schematic. Before this, the device was in a constant state of reset through the XRS pin. Furthermore, the first MCU placed on the PCB had to be removed and replaced as human error led to voltage probing the wrong pin, which destroyed the initial IC.

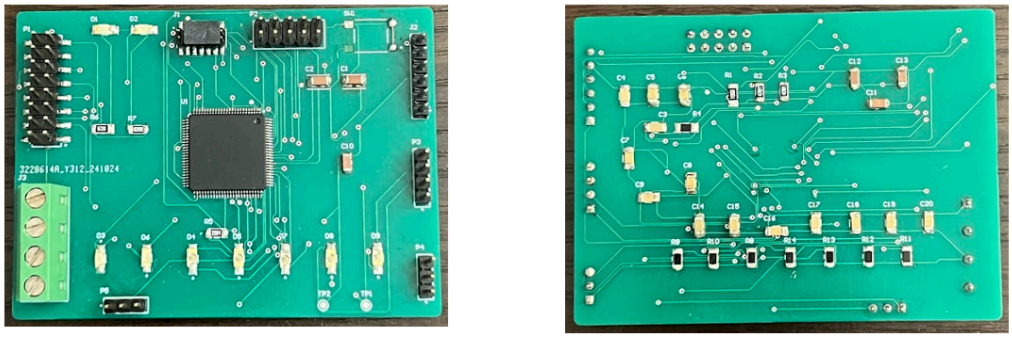


Figure 29: Front & Back of PCB

3.2.4.1. Power

The PCB requires a 3.3V supply to provide the necessary operating voltage for the MCU. In the lab, this was tested by connecting a 3.3V input to the terminal block on the PCB. To ensure stable operation and no voltage drop off, the power supply was set to 3.3V at 100mA. The amperage was gradually increased on the power supply until no voltage drop off was observed. This ensures integration with the power subsystem, which provides a 3.3V power rail with up to 1A.

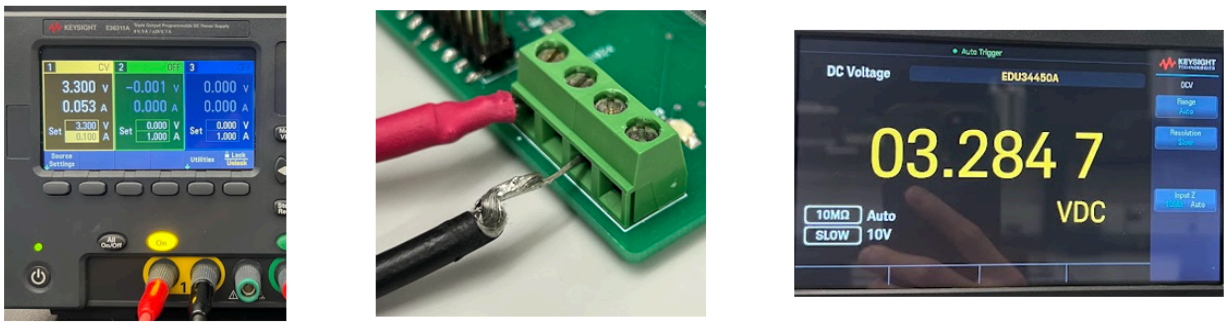


Figure 30: 3.3V Input to Terminal Block

3.2.4.2. JTAG Flashing & Programming

The PCB was able to be programmed through the JTAG protocol. The necessary signals (TDI, TDO, TMS, and TCK) were correctly wired to a header on the PCB that connects to the JTAG header on the launchpad. The launchpad includes a built-in XDS-110 debug probe which is used to interface between the software written on CCS and then build and compile the software onto the target device (TMS320F280049C).

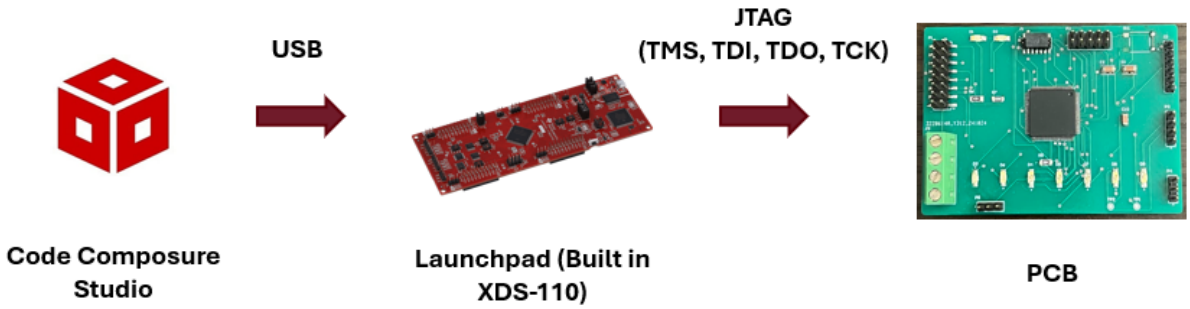


Figure 31: Successful JTAG Programming Method

Before the MCU was able to be programmed through the launchpad, there were a series of issues encountered, the following text explains the debugging process to obtain the solution. Originally, the sponsor recommended using an external XDS-110 debug probe. This debug probe would interface between the PC via USB and the PCB via JTAG. This method of programming did not work as the MCU could not detect the probe and a series of errors would occur.

This issue was overcome by using the onboard XDS110 that is built into the LAUNCHXL-F280049C (evaluation board). By removing the jumpers off of J101 on the evaluation board, the TDI, TDO, TMS, and TCK signals become available. By connecting the JTAG header from the PCB to the JTAG header on the launchpad, the MCU on the PCB can be programmed through the evaluation board. After doing so, programs are built, compiled, and successfully flashed onto the target MCU.

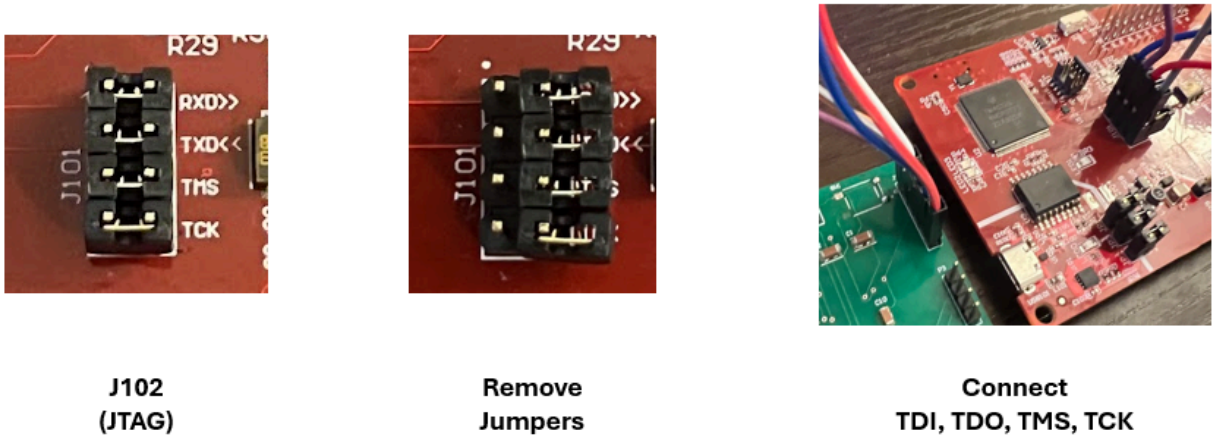


Figure 32: Flashing through Launchpad

The figure below demonstrates the console output after a program is successfully built, compiled, and flashed onto the MCU.

```
C28xx_CPU1: GEL Output:
Memory Map Initialization Complete
C28xx_CPU1: GEL Output: ... DCSM Initialization Start ...
C28xx_CPU1: GEL Output: ... DCSM Initialization Done ...
C28xx_CPU1: GEL Output: ... DCSM Initialization Start ...
C28xx_CPU1: GEL Output: ... DCSM Initialization Done ...
```

Figure 33: Successful Connection

The below figure shows the software for a Demo that was performed during demo day. In this demo, the PCB LEDs would toggle ON/OFF based on the high/low input received by the ESP-32. The PCB also would spin a stepper motor to show functionality of the GPIOs.

```c
//Code for Demo
//ESP->C2000->LED Blink on High and Low
//Motor Spinning -> Stepper motor programming


#include "F28x_Project.h"

// GPIO definitions for motor driver (IN1, IN2, IN3, IN4)
#define GPIO7 7   // IN1
#define GPIO8 8   // IN2
#define GPIO9 9   // IN3
#define GPIO10 10 // IN4

// GPIO definitions for LEDs and ESP32 input
#define DEVICE_GPIO_PIN_INPUT 2   // GPIO2 (connected to ESP32)
#define DEVICE_GPIO_PIN_LED1 4    // GPIO4 (LED1)
#define DEVICE_GPIO_PIN_LED2 5    // GPIO5 (LED2)
#define DEVICE_GPIO_PIN_LED3 6    // GPIO6 (LED3)

// Function prototypes
void delay_ms(unsigned int ms);
void stepMotorWithLED(int stepCount);
void initializeGPIO(void);
void initializeClock(void);
void updateLEDs(void);

void main(void) {
    // Step 1: Initialize clock and GPIOs
    initializeClock();
    initializeGPIO();

    // Debugging: Verify GPIO2 toggles correctly
    GPIO_SetupPinMux(3, GPIO_MUX_CPU1, 0);  // Configure GPIO3 for debugging
    GPIO_SetupPinOptions(3, GPIO_OUTPUT, GPIO_PUSHPULL);

    // Step 2: Main loop for motor and LED control
    while (1) {
        stepMotorWithLED(2038);  // Rotate motor and update LEDs based on GPIO2
        delay_ms(1000);          // 1-second delay before the next cycle
    }
}

// Function to initialize clock
void initializeClock(void) {
    EALLOW;
    ClkCfgRegs.CLKSRCCTL1.bit.OSCCLKSRCSEL = 0;  // Use INTOSC2
    ClkCfgRegs.SYSPLLCTL1.bit.PLLEN = 1;         // Enable PLL
    ClkCfgRegs.SYSPLLMULT.bit.IMULT = 10;        // Multiply INTOSC2 (10 MHz) to 100 MHz
    while (ClkCfgRegs.SYSPLLSTS.bit.LOCKS != 1); // Wait for PLL to lock
    ClkCfgRegs.SYSCLKDIVSEL.bit.PLLSYSCLKDIV = 0; // No clock divider (100 MHz)
    EDIS;
}
```

Initialization

```c

// Function to initialize GPIOs
void initializeGPIO(void) {
    InitGpio();

    // Configure motor driver GPIOs (GPIO7, GPIO8, GPIO9, GPIO10)
    EALLOW;
    GPIO_SetupPinMux(GPIO7, GPIO_MUX_CPU1, 0);
    GPIO_SetupPinMux(GPIO8, GPIO_MUX_CPU1, 0);
    GPIO_SetupPinMux(GPIO9, GPIO_MUX_CPU1, 0);
    GPIO_SetupPinMux(GPIO10, GPIO_MUX_CPU1, 0);

    GPIO_SetupPinOptions(GPIO7, GPIO_OUTPUT, GPIO_PUSHPULL);
    GPIO_SetupPinOptions(GPIO8, GPIO_OUTPUT, GPIO_PUSHPULL);
    GPIO_SetupPinOptions(GPIO9, GPIO_OUTPUT, GPIO_PUSHPULL);
    GPIO_SetupPinOptions(GPIO10, GPIO_OUTPUT, GPIO_PUSHPULL);

    // Configure ESP32 input (GPIO2)
    GPIO_SetupPinMux(DEVICE_GPIO_PIN_INPUT, GPIO_MUX_CPU1, 0);
    GPIO_SetupPinOptions(DEVICE_GPIO_PIN_INPUT, GPIO_INPUT, GPIO_PULLUP);

    // Configure LEDs (GPIO4, GPIO5, GPIO6)
    GPIO_SetupPinMux(DEVICE_GPIO_PIN_LED1, GPIO_MUX_CPU1, 0);
    GPIO_SetupPinMux(DEVICE_GPIO_PIN_LED2, GPIO_MUX_CPU1, 0);
    GPIO_SetupPinMux(DEVICE_GPIO_PIN_LED3, GPIO_MUX_CPU1, 0);

    GPIO_SetupPinOptions(DEVICE_GPIO_PIN_LED1, GPIO_OUTPUT, GPIO_PUSHPULL);
    GPIO_SetupPinOptions(DEVICE_GPIO_PIN_LED2, GPIO_OUTPUT, GPIO_PUSHPULL);
    GPIO_SetupPinOptions(DEVICE_GPIO_PIN_LED3, GPIO_OUTPUT, GPIO_PUSHPULL);

    // Turn OFF all LEDs initially
    GPIO_WritePin(DEVICE_GPIO_PIN_LED1, 0);
    GPIO_WritePin(DEVICE_GPIO_PIN_LED2, 0);
    GPIO_WritePin(DEVICE_GPIO_PIN_LED3, 0);
    EDIS;
}
```

GPIO LED Toggle

```c
// Function to step the motor while checking LEDs
void stepMotorWithLED(int stepCount) {
    int i;

    for (i = 0; i < stepCount; i++) {
        // Step 1
        GPIO_WritePin(GPIO7, 1);
        GPIO_WritePin(GPIO8, 0);
        GPIO_WritePin(GPIO9, 0);
        GPIO_WritePin(GPIO10, 1);
        updateLEDs();
        delay_ms(1);

        // Step 2
        GPIO_WritePin(GPIO7, 1);
        GPIO_WritePin(GPIO8, 1);
        GPIO_WritePin(GPIO9, 0);
        GPIO_WritePin(GPIO10, 0);
        updateLEDs();
        delay_ms(1);

        // Step 3
        GPIO_WritePin(GPIO7, 0);
        GPIO_WritePin(GPIO8, 1);
        GPIO_WritePin(GPIO9, 1);
        GPIO_WritePin(GPIO10, 0);
        updateLEDs();
        delay_ms(1);

        // Step 4
        GPIO_WritePin(GPIO7, 0);
        GPIO_WritePin(GPIO8, 0);
        GPIO_WritePin(GPIO9, 1);
        GPIO_WritePin(GPIO10, 1);
        updateLEDs();
        delay_ms(1);
    }
}

// Function to update LEDs based on GPIO2
void updateLEDs(void) {
    Uint16 inputState = GPIO_ReadPin(DEVICE_GPIO_PIN_INPUT);

    // Debugging: Toggle GPIO3 based on GPIO2
    GPIO_WritePin(3, inputState);

    if (inputState == 1) {  // GPIO2 is HIGH
        GPIO_WritePin(DEVICE_GPIO_PIN_LED1, 1);
        GPIO_WritePin(DEVICE_GPIO_PIN_LED2, 1);
        GPIO_WritePin(DEVICE_GPIO_PIN_LED3, 1);
    } else {  // GPIO2 is LOW
        GPIO_WritePin(DEVICE_GPIO_PIN_LED1, 0);
        GPIO_WritePin(DEVICE_GPIO_PIN_LED2, 0);
        GPIO_WritePin(DEVICE_GPIO_PIN_LED3, 0);
    }
}

// Millisecond delay function
void delay_ms(unsigned int ms) {
    unsigned int j;
    for (j = 0; j < ms; j++) {
        DELAY_US(1000); // 1 ms delay
    }
}
```

Motor Spin & ESP-32 Communication

Figure 34: C2000 Software for Demo

The figure below shows the software written on Arduino IDE to toggle a GPIO pin on the ESP-32 to high and low every second. This signal was received by the C2000.

```c
#define TXD_PIN 1  // Default UART0 TX pin (GPIO1)

void setup() {
    pinMode(TXD_PIN, OUTPUT);  // Configure TX pin as
}

void loop() {
    digitalWrite(TXD_PIN, HIGH);  // Set TX pin HIGH
    delay(1000);                  // Wait 1 second
    digitalWrite(TXD_PIN, LOW);   // Set TX pin LOW
    delay(1000);                  // Wait 1 second
}
```

Figure 35: ESP-32 Arduino Software (High/Low Toggle)

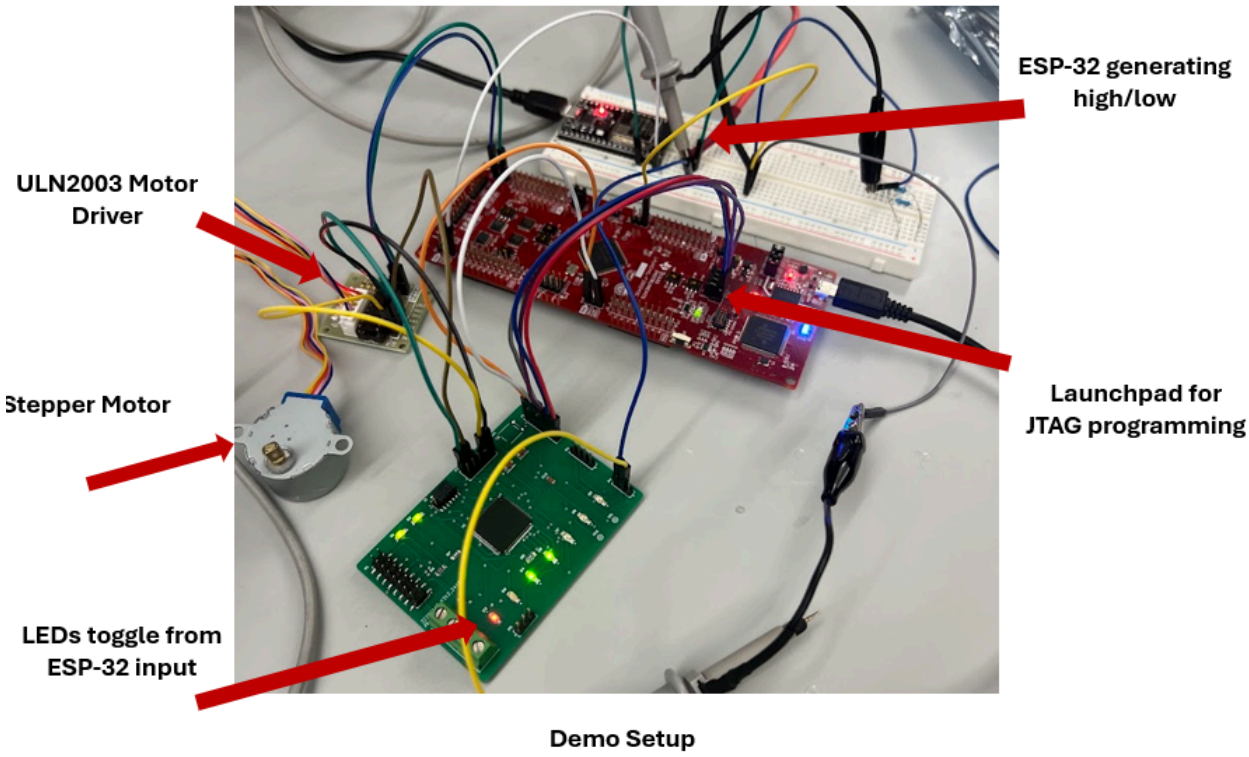


Figure 36: Demo Setup:

3.2.4.3. PWM Waveform Generation

The PCB was successfully able to generate PWM waveforms that are to be used to drive the high and low gate driver inputs (INH/IN) (INL/EN) on the DRV816x motor driver. The motor driver subsystem will use 3x motor drivers per BLDC motor. Each motor driver will require 2x PWM signals from the MCU. Therefore, per BLDC motor, the MCU subsystem needs to provide 6x PWM signals. The MCU/Processing subsystem can successfully generate the required signals through the use of GPIO0 - GPIO15 with every two GPIOs in this range corresponding to ePWMA and ePWMB for a total of 16x possible PWM signals from one MCU.

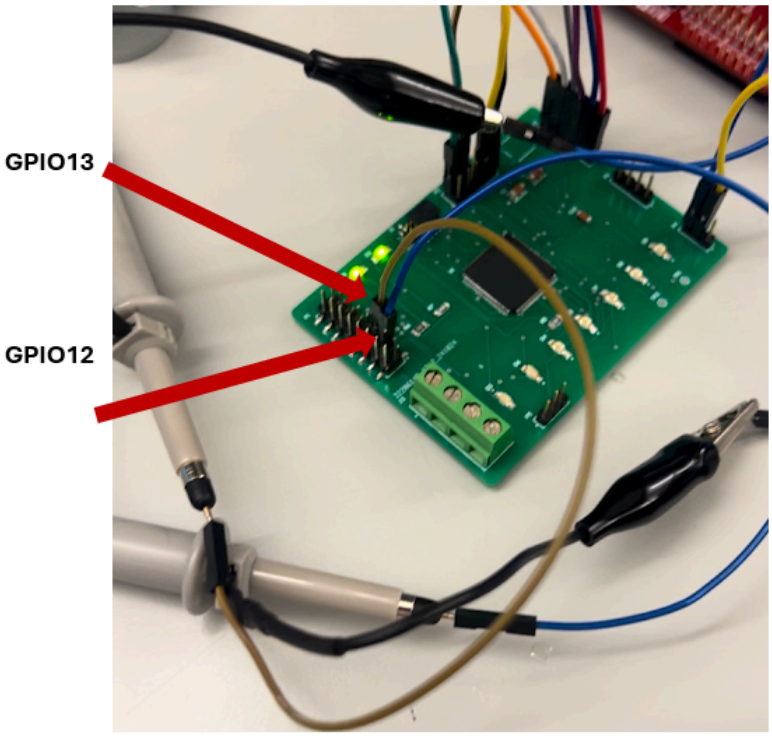


Figure 37: Probing PWM Output Pins

The figure below shows an oscilloscope screenshot of two PWM waveforms being formed through EPWM7A & EPWM7B at 50kHz. This was generated on the created PCB.

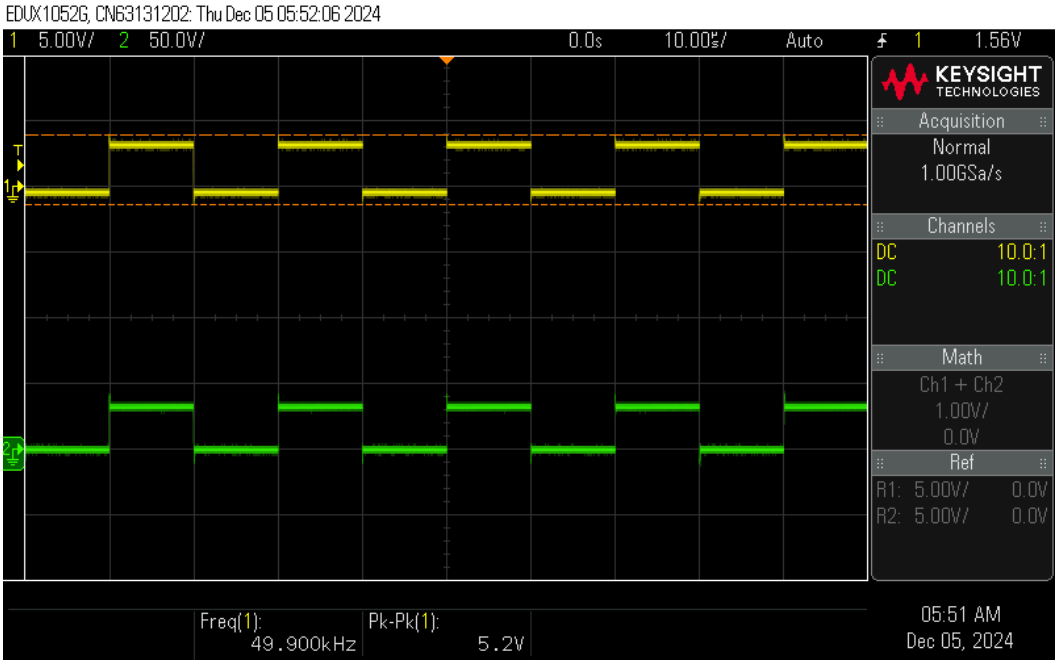


Figure 38: PWM Generation EPWM7A & EPWM7B (GPIO 12 & GPIO13)

The figure below shows two PWM waveforms being generated by the “epwm_ex13_up_aq” example code. These waveforms are being generated on the created PCB.

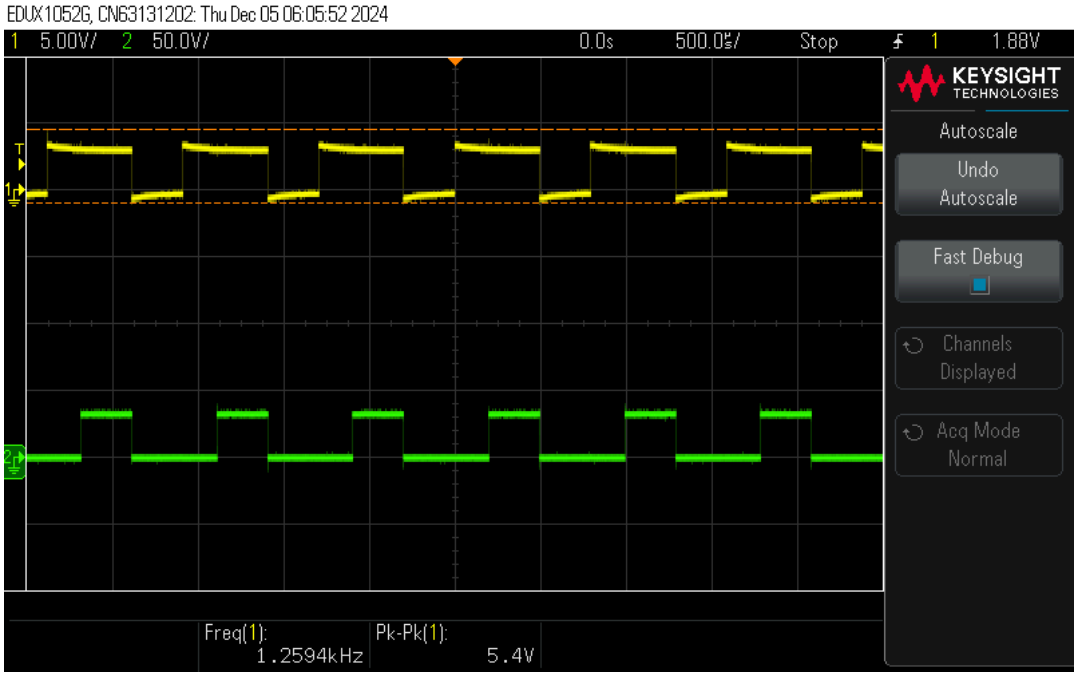


Figure 39: PWM High/Low Generation

The two previous figures demonstrate that the PCB is capable of generating the required PWM waveforms. The MCU/Processing subsystem will be able to send these PWM signals to the motor driver. The subsystem is ready for integration to the motor driver subsystem. After integration, the exact waveform requirement to rotate a BLDC motor is pending confirmation from the motor driving subsystem and is the next step to spin a motor.

3.2.4.4. Communication with ESP-32

Communication between the ESP-32 and C2000 MCU was successful. The wireless connectivity subsystem will use an ESP-32 to interface with the mobile application. To test connectivity between the ESP-32 and C2000, an ESP-32 was purchased by the MCU/Processing subsystem and development began.

A program was written to toggle GPIO1 to high for 1 second and low for 1 second on the ESP-32. This program ran infinitely to constantly generate a high and low signal. The goal was for the C2000 to recognize this change of input from the ESP-32 and toggle an LED based on if the input is high or low to show that communication between these two devices was successful. To do so, GPIO1 from the ESP32 was connected to GPIO2 on the C2000 MCU. The software developed within CCS used the GPIO_ReadPin function to detect if the input state of GPIO2 was high or low. Based on this, the C2000 would use the GPIO_WritePin function to toggle the GPIOs connected to three LEDs (GPIO4, GPIO5, GPIO6) seen in the figure below.

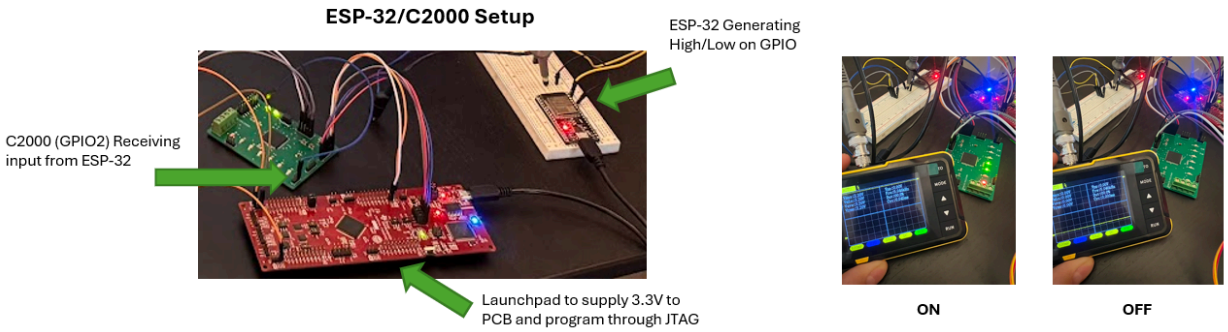


Figure 40: ESP-32/C2000 Setup

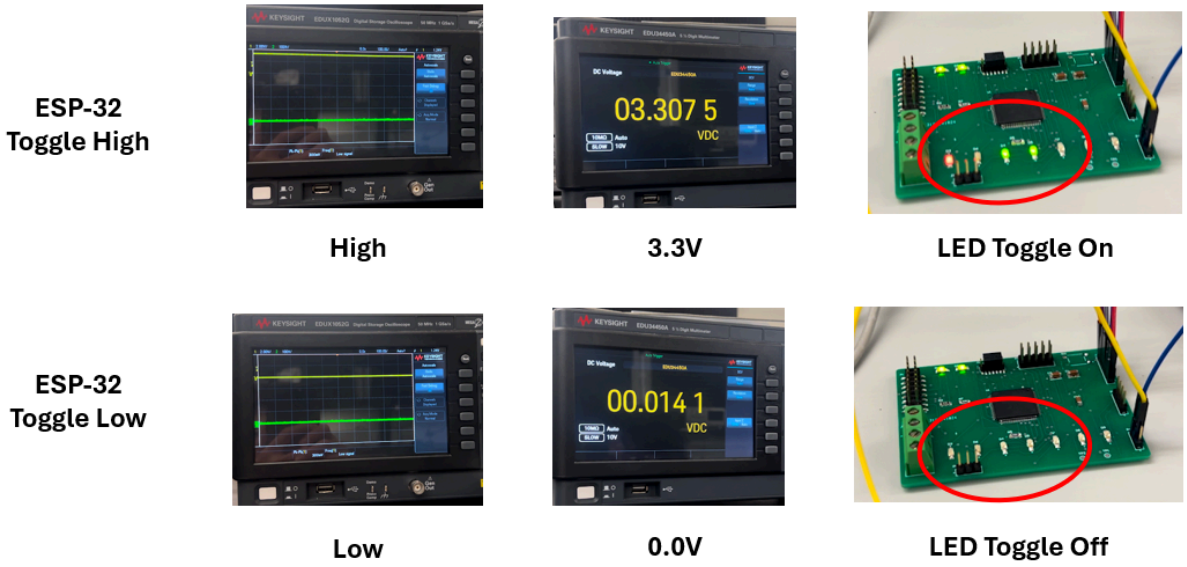


Figure 41: ESP-32 to C2000 Communication (LED Toggle)

The figure above demonstrates that communication between the ESP-32 and the C2000 was successful by using high/low signal generation on the GPIOs. A similar concept shall be used to integrate with the wireless subsystem next semester. When the user selects movement on the mobile application, (up, down, left, right, open, close) the ESP-32 will toggle a specific pin to high, the C2000 will then detect the input and through a series of if statements, output the desired PWM waveform to send to the motor driver to achieve the desired movement.

3.3. Subsystem Validation

The requirements of the MCU/Processing subsystem were validated in accordance with the validation and execution plan. The initial design of the PCB schematic was reviewed thoroughly before export to manufacturing, the development on the C2000 evaluation board built a foundation of programming knowledge, and the testing and validation of the PCB was accomplished to show the required deliverables of the MCU/Processing subsystem. The main objectives such as supplying power to board, programming through JTAG, generating PWM waveforms, and receiving signals from the ESP-32 were achieved.

The previous documentation provided in section 3.2.4. “PCB Testing and Results” documents in detail the steps taken to produce the required deliverables of the subsystem. Images to support the validation of the subsystem are included in section 3.2.4. The table below lists the required validation tasks for the subsystem, if and how they were achieved, and the supporting figure to ensure successful completion of validation.

Deliverable	**Achieved (Yes/No)**	**How**	**Supporting Figure**
3.3V power to PCB	Yes	Terminal block input with 3.3v at 500mA minimum to ensure no voltage drop off.	Figure 30
JTAG Programming	Yes	JTAG programming through onboard XDS-110 debugger on the launchpad.	Figure 31, 32, 33
PWM Waveform Generation	Yes	Software written using ePWM modules to generate waveforms on GPIO pins .	Figure 37, 38, 39
ESP-32/C2000 Communication	Yes	Software written to receive input from ESP-32 through GPIO pins on C2000. Toggle instructions through high/low input.	Figure 40, 41

Table 8: MCU/Processing Validated Deliverables

3.4. Subsystem Conclusion

To conclude, the MCU/Processing subsystem was validated to achieve the milestones listed in the validation plan. The execution plan had milestones achieved, with it nearing completion as soon as subsystems are integrated in the coming semester.

The subsystem performs as expected and is capable of generating and receiving the required signals from the different subsystems. This includes receiving input from the wireless subsystems ESP-32, receiving 3.3v from the power subsystem, and generating the required PWM signals for the motor driver subsystem. The validation and execution plan aided in keeping the MCU/Processing subsystem deliverables on time and aligned with the team members. Furthermore, constant communication between team members ensured compatibility between systems to allow integration following the conclusion of the semester.

Although the subsystem performed as expected, work will be done during winter break to allow a smooth transition into next semester. The goals for winter break include developing a new PCB board, writing new firmware for the MCU, and strategizing with team members over physical design of the cobot. The new PCB board shall include a larger quantity of headers, allowing access to the necessary pins from the MCU. The firmware development will refine the PWM signal generation and communication with the ESP-32. Additionally, the approach to develop a physical cobot will be researched, with team support growing for a crane like structure.

The subsystem is now ready for integration with the motor driver, power, and wireless subsystems. Collaboration between the MCU and motor driver subsystems will continue to ensure proper development of firmware and signal generation, supporting BLDC motor motion. This momentum will carry into the next semester as the team transitions to a more team-oriented approach to accomplish the goal of delivering a fully functional collaborative robot (cobot).

4. Power & Battery Management Subsystem - Emily Hamsa

4.1. Subsystem Introduction

The power and battery management subsystem is responsible for providing power to the cobot and its associated components. The subsystem uses a PCB to convert the input voltage to five different output voltages, with an additional output voltage that is connected directly to the input. To ensure seamless integration with the other subsystems, this PCB has been designed according to a load list that was developed as a part of planning for this subsystem, which breaks down each component requiring power and records the voltage and current specified by their data sheets. As the cobot is wireless, the input will be a 48V, 20Ah battery, which was selected also based on the load list and for a runtime of 30 minutes. Validation of this subsystem was completed based upon the validation plan to further ensure completeness and success during integration stages.

4.2. Subsystem Details

4.2.1. Planning and Selection

Initial planning for this subsystem included the development of a load list, which was used to select a battery size and to determine the necessary output voltages, as well as assist the PCB design stage to ensure trace widths would be able to withstand higher current demands of some components. Below is the completed load list.

Component Name	Component Type	High/ Low Voltage	Amps	A	Voltage Input	V	Supply Voltage Min	Supply Voltage Max	Power
PDRV8161DGSR	motor driver	high	2mA IGVDD	0.002	12V	12	8V	20V	0.024
									0
https://www.omc-stepperonline	BLDC motor 1	high	5.4A	5.4	48V	48		48V	259.2
https://teknic.com/hudson-mod	BLDC motor 2	high	5.1A	5.1	48V	48		48V	244.8
https://teknic.com/hudson-mod	BLDC motor 3	high	7.7A	7.7	48V	48		48V	369.6
https://www.omc-stepperonline	BLDC motor 4	high	1A	1	24V	24		48V	24
Robotic Arm Pincher	pincher	high	0.9A	0.9	6V	6	4.8V	7.2V	5.4
									0
	Motor Encoder Boards			0.36		5			1.8
TMS320F28004	MCU	low	90 mA Max, 61 mA Typ	0.061	3.3V VDDIO and VDDA, 1.2V VDD	3.3	-0.3 V	4.6 V	0.2013
									0
ESP32	bluetooth components	low	up to 250 mA	0.25		3.3	2.2V	3.6V	0.825

Capacity Calculations		
Total Power	905.8503	Watts, sum of all component's power
Duration	0.5	Hours, desired runtime for the robot
Energy Consumption	452.92515	Wh, Watts * Duration
Account for Safety/ Efficiency	1132.312875	Wh, Energy consumption * 2 (assuming 50% depth of discharge) * 1.25 (accounting for efficiency losses)
Battery Capacity	23.58985156	Ah, Wh / Voltage of Battery

Table 9: Cobot Load List

Also included in the load list are calculations for the necessary battery size based on power demands of the system. Assuming a runtime of 30 minutes and a voltage size of 48 volts, a 48 volt 20 amp-hour battery was selected.

Grouping components with the same output voltage together to determine the total current demanded by the voltage rail, TI WeBench was then used to select TI-branded buck converters that had appropriate voltage output capabilities and current ratings. The output voltage rails, their current ratings and the selected converters are as followed:

- LM5163DDAR: 3.3V, 0.5V
- LMR36510FADDAR: 5V, 1A
- LM5012DDAR: 6V, 2.5A
- LMR36510FADDAR: 12V, 1A
- LM5164DDAR: 24V, 1A

4.2.2. Schematic and PCB Design

4.2.2.1. Schematic Design

Using TI WeBench and the load list described above, individual circuits were generated based on input voltage, desired output voltage, and the current rating needed for the circuit. After compiling the five selected circuits, Altium was used to design the individual schematics for each output voltage rail and connect them together through the power and ground nets to be used on one PCB. During the schematic design stage, terminal blocks were also selected, ensuring each component needing power would have access to a power and ground connection on the PCB.

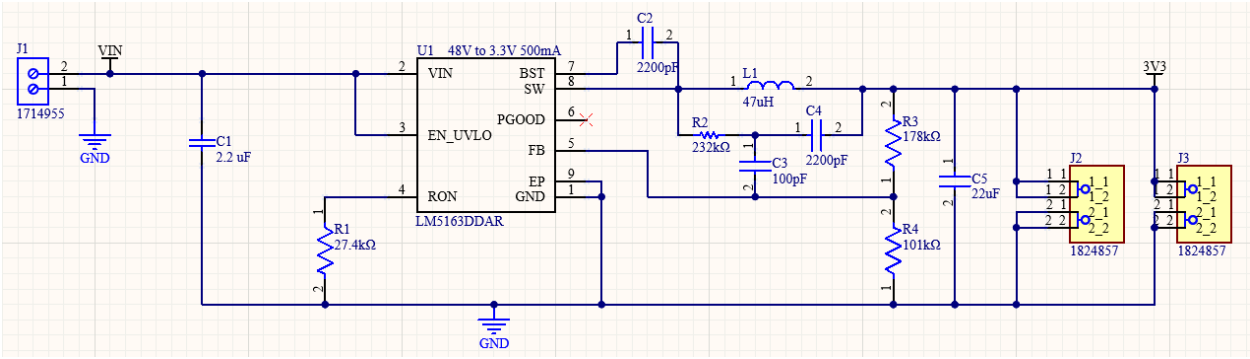


Figure 42: 3.3V, 0.5A Step-down Circuit Schematic

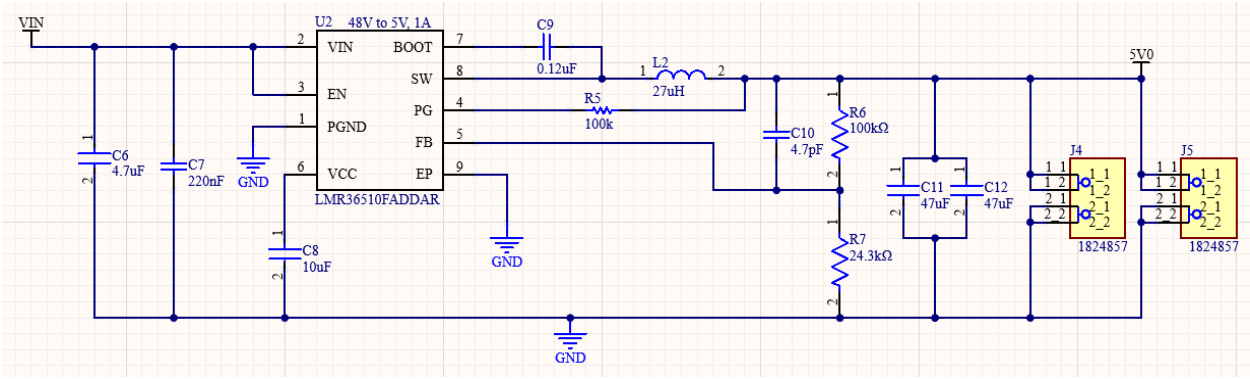


Figure 43: 5V, 1A Step-down Circuit Schematic

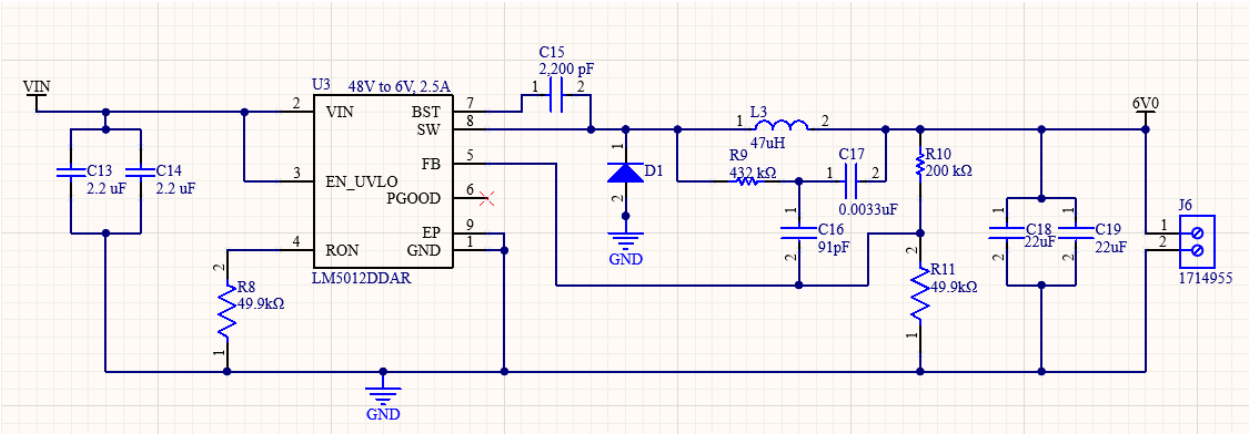


Figure 44: 6V, 2.5A Step-down Circuit Schematic

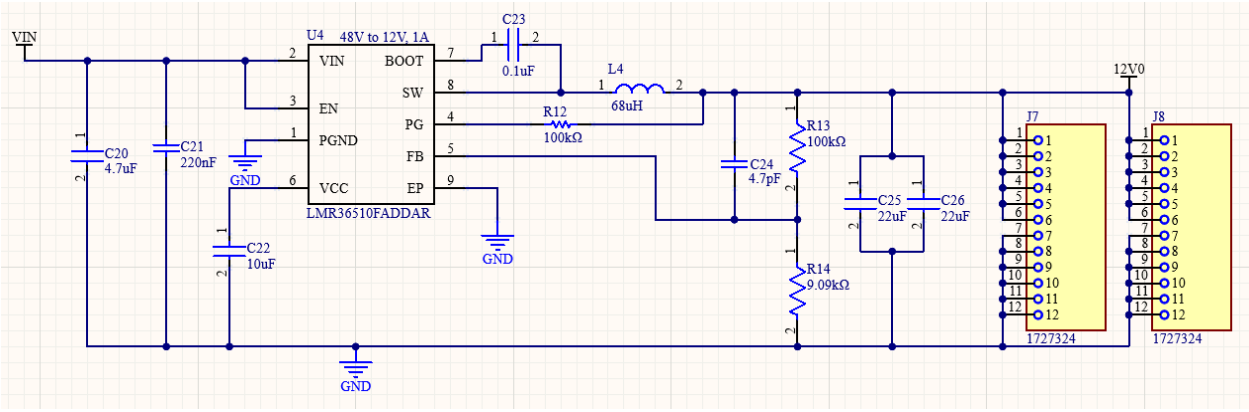


Figure 45: 12V, 1A Step-down Circuit Schematic

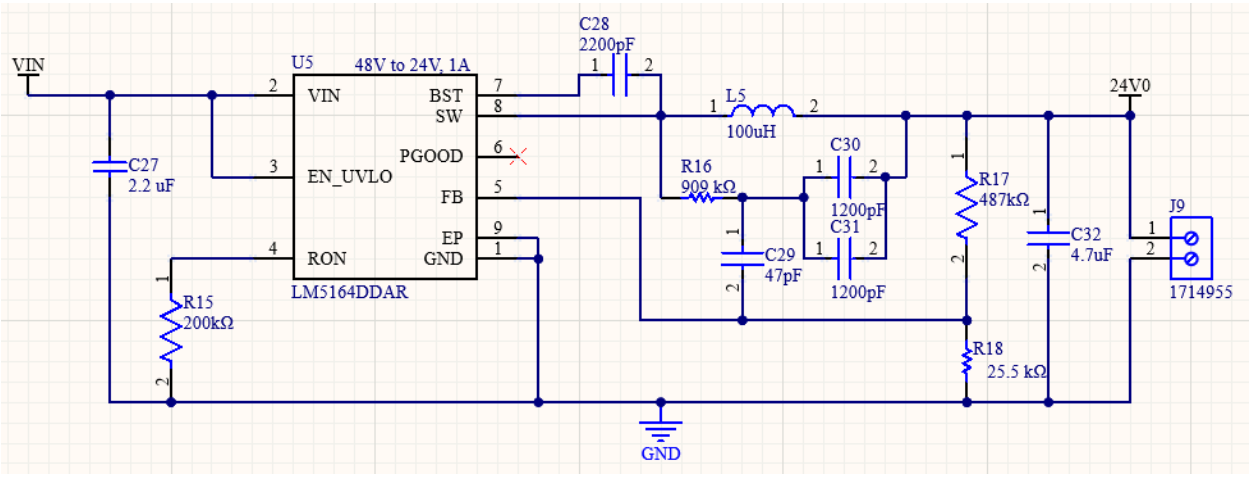


Figure 46: 24V, 1A Step-down Circuit Schematic

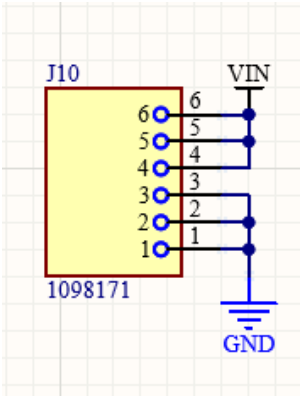


Figure 47: Connection from Vin Directly to Output Schematic

4.2.2.2. PCB Design and Fabrication

Layout and routing of the PCB was based on the schematics and the load list. Each step-down circuit was laid out isolated from the others and close to the terminal block providing the output from the converter. Routing and trace widths were determined based on the load list, ensuring that each trace was wide enough to sustain any large amounts of current expected for the load and preparing for integration.

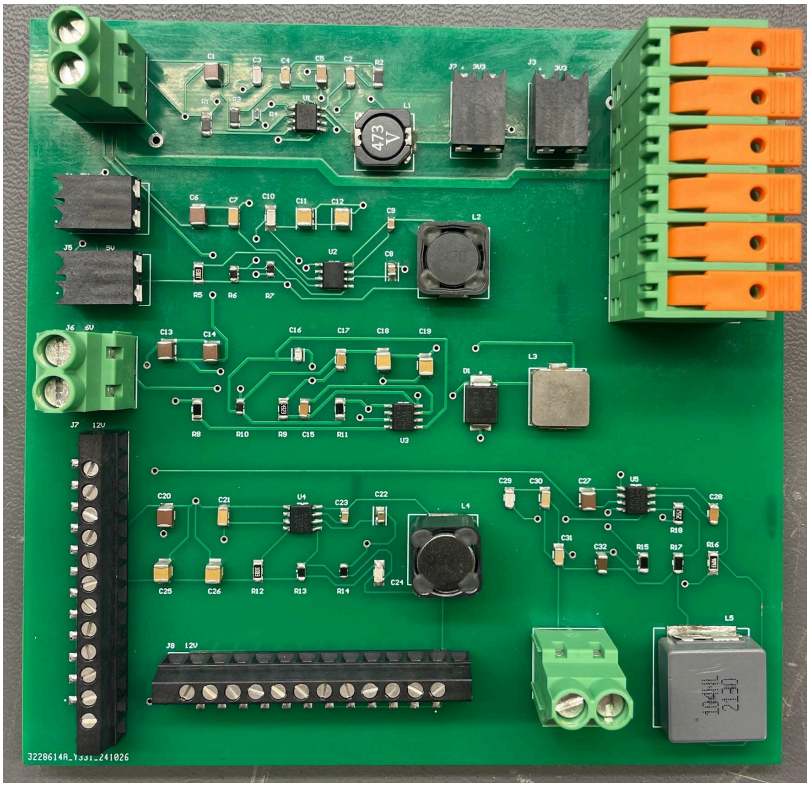


Figure 48: Power Supply PCB

4.3. Subsystem Validation

Validation for this subsystem was done based on the validation plan developed at the beginning of the semester. Within the validation plan there are tests to ensure that the output voltage is reading the correct voltage, and the output voltage is steady and not noisy.

4.3.1. Output Voltage Verification

To verify that the output voltage is reading correctly, a digital multimeter was used. The first output voltage to be tested was the 3.3V terminal. When testing this under 25V input and intermediate values leading up to 25V, the output was successful.

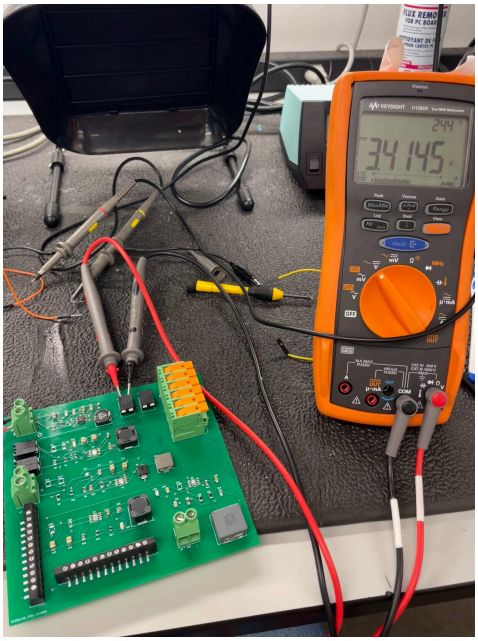


Figure 49: 3.3V Output When Tested with 25V Input

Testing the 6V, 12V, 24V, and terminal connected directly to the input were also successful when tested with intermediate voltages up to and including 30V.

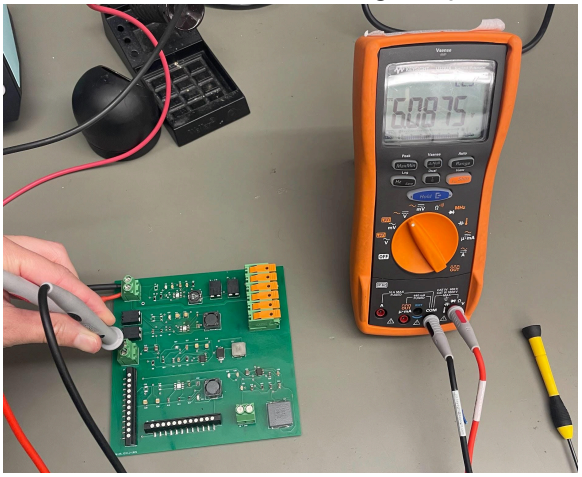


Figure 50: 6V Output When Tested with 30V Input

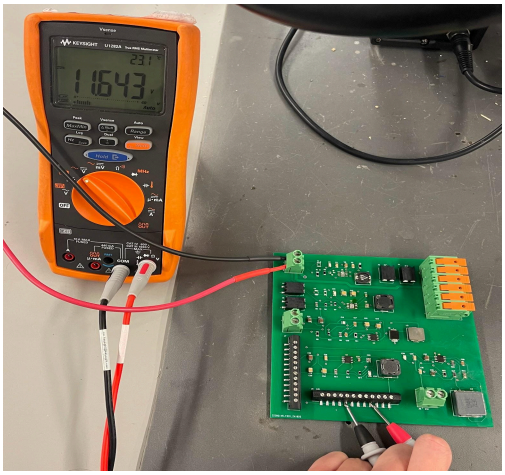


Figure 51: 12V Output When Tested with 30V Input

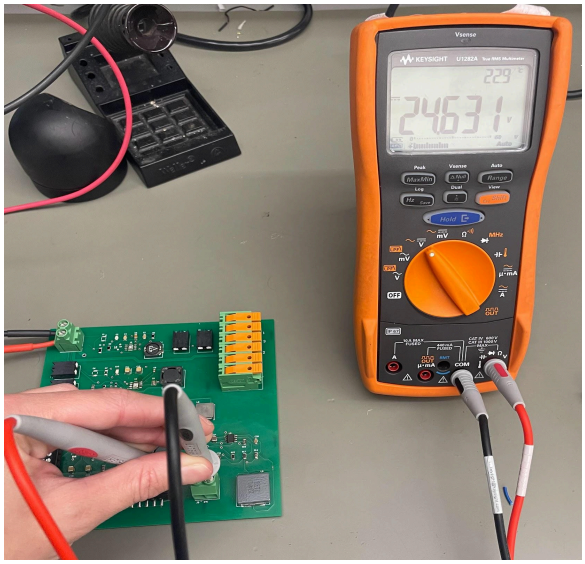


Figure 52: 24V Output When Tested with 30V Input

There were some complications during this testing. When connecting the components to activate the 24V step-down circuit, the converter used in the 3.3V converter circuit shorted between power and ground. Additionally, on the power subsystem PCB, the output of the 5V step-down circuit outputs less than one volt. Debugging for both of these issues included testing for continuity throughout the circuit, changing the converters, verifying the designs against TI WeBench, and verifying the components on the board are correct. None of these attempts fixed these issues, and so it is suspected that this is caused by a PCB board layout and routing issue. Further evidence for this is found in the motor and motor driver subsystems's PCB, as the 3.3V and 5V step down schematics developed and designed in this subsystem were both as is on the PCB, and are operating as expected, with the only change being the physical layout and routing on the PCB.

While these voltage outputs have not been verified for the full expected input of 48V, once the entirety of the PCB is completed and verified at an intermediate voltage of 30V, there is little concern or doubt that continuing to raise the input voltage to 48V will cause the board to fail a multimeter test.

4.3.2. Input and Output Voltage Noise Verification

To ensure that the input and output terminals have limited noise and are truly delivering DC voltage, an oscilloscope was used. When testing the 3.3V and 6V are connected to a 25V input, there is little to no noise in the output or input terminals.

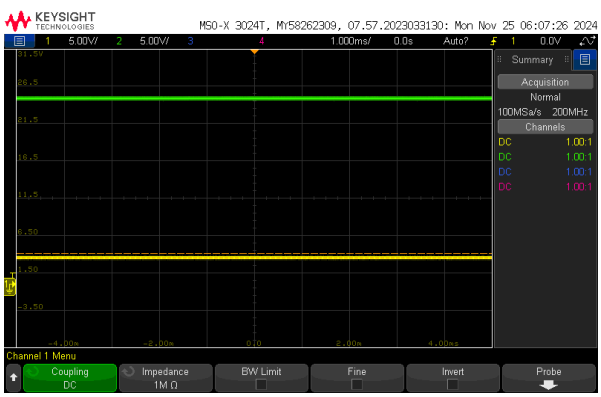


Figure 53: 3.3V Noise Output When Tested with 25V Input

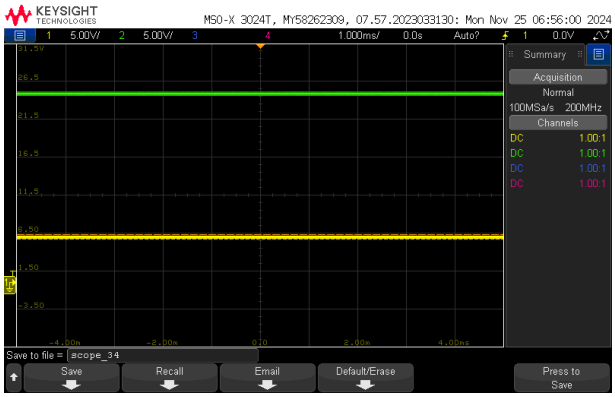


Figure 54: 6V Noise Output When Tested with 25V Input

When testing the 24V output with a 30V input, there is also very little noise in the system.

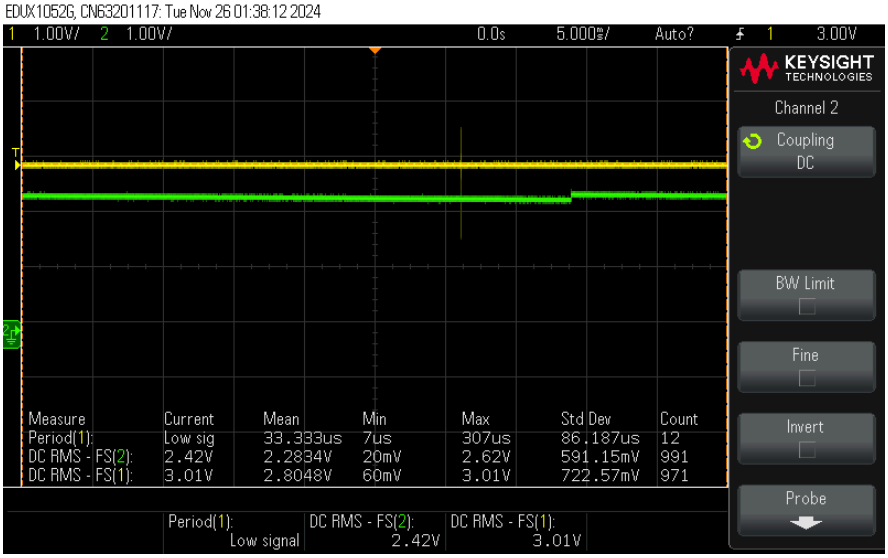


Figure 55: 24V Noise Output When Tested with 30V Input

Once adding the 12V output converter circuit and testing with a 25V input, noise in the system increased.

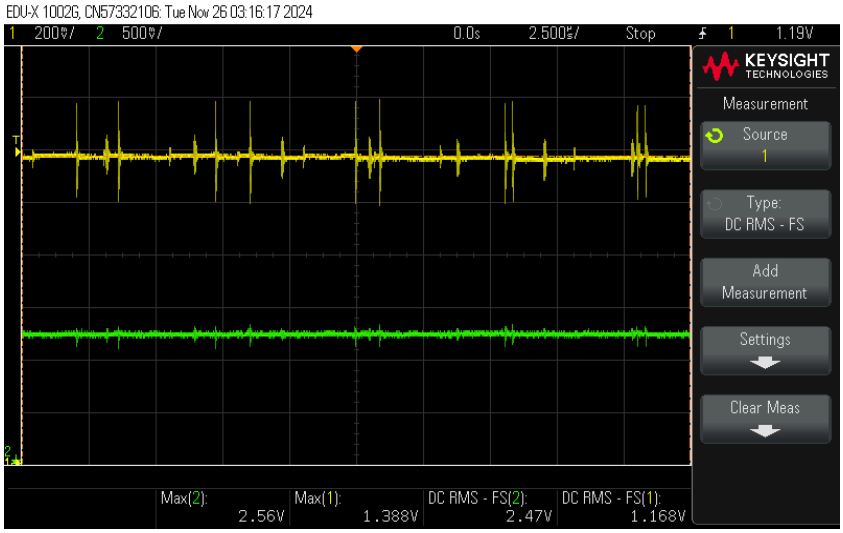


Figure 56: 12V Noise Output When Tested with 30V Input

This is suspected to be from an inductor or capacitor that is a part of the 12V step-down converter circuit. Debugging to resolve this was attempted by replacing the inductor in the circuit to ensure the component was not damaged, and this did not resolve the excess noise. Further debugging to resolve this issue will include research into the capacitors and inductors and verification of their ratings.

Deliverable	**Achieved (Yes/No)**	**Criteria**	**Supporting Figure**
3.3V DC Output	Yes	When testing with a multimeter, the voltage is 3.3V, and the oscilloscope shows minimal noise.	Figure 49, Figure 53
5V DC Output	No	When testing with a multimeter, the voltage is 5V, and the oscilloscope shows minimal noise.	N/A
6V DC Output	Yes	When testing with a multimeter, the voltage is 6V, and the oscilloscope shows minimal noise.	Figure 50, Figure 54
12V DC Output	No	When testing with a multimeter, the voltage is 12V, and the oscilloscope shows minimal noise.	Figure 51, Figure 56
24V DC Output	Yes	When testing with a multimeter, the voltage is 24V, and the oscilloscope shows minimal noise.	Figure 52, Figure 55

Table 10: Validated Deliverables: Power Supply PCB

4.4. Subsystem Conclusion

Looking forward to integration, once a few tasks are completed, there should be no major roadblocks. Beginning with the 3.3V and 5V step-down converter circuits, a comparison between the layout and routing in the motor and motor driver subsystem's PCB and this subsystem's PCB will be completed, and a new board will be developed and fabricated to address any discrepancies between the layout and routing of the two. To fix the excess noise throughout the system, research into each component used in the 12V step-down circuit will be completed to determine which capacitor or inductor is causing noise, and selection of a new component which will be better suited for the application in this PCB.

After verification of the entire board is completed at an intermediate value of 30V, including a test with the digital multimeter to ensure voltages are as expected and with the oscilloscope to ensure the noise is no longer in excess, tests at the full expected input of 48V will be completed. In addition to these tests, testing with an E-load will be completed to verify that under the expected current demand from the cobot, the PCB is still able to deliver reliable DC voltage and withstand the load.

These updates and testing will be completed over winter break, prior to the start of the spring semester and integration with the rest of the subsystems. After the verification is complete, integration and movement towards the collective goal of cobot operation will be mostly straightforward for this subsystem.

5. User Interface & Wireless Connectivity - Jaishil Shah

5.1. Subsystem Introduction

The User Interface & Wireless Connectivity Subsystem allows seamless communication between an iOS mobile device using a Flutterflow-developed application and a C2000 MCU via an ESP-32 Bluetooth module. This subsystem is split into two major components, the User Interface (UI), and the Wireless Communication. The UI is responsible for being the interface between the user and the device, allowing signals to be sent back and forth. The wireless communication aspect is where the ESP-32 comes into play and is responsible for bidirectional data flow. The ESP-32 is capable of receiving data from the mobile device and from there sending it over to the MCU via high/low GPIO pins to perform cobot movement. Since the goal of the cobot is to create a safe and seamless procedure where the user does not have to physically interact with the robotic device, wireless communication & a virtual interface are crucial in fulfilling that goal.

5.2. Subsystem Details

This section will break down the details and design of the subsystem and how they play a pivotal part in ensuring proper connection/communication.

5.2.1. Subsystem Block Diagram

The figure below depicts a generalized flow of how and where signals will be sent via wireless communication. The heart of the subsystem lies within the ESP-32, as it functions as the middleman between the mobile application and the MCU.
(The ESP-32 is displayed in turquoise)

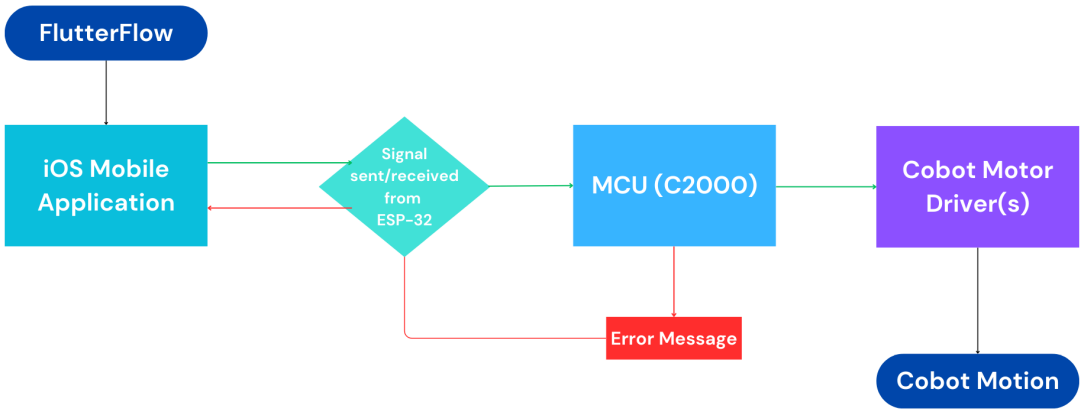


Figure 57: UI & Wireless Connectivity Subsystem Block Diagram

5.2.2.1. Mobile Application (Flutterflow)

The Mobile Application that was developed on Flutterflow had 4 main objectives: An interface that the user can easily understand and interact with, the ability to go back and forth between pages, a proper Bluetooth connection, and the ability to send data to the ESP-32. The first one was the least time-consuming, as it consisted of creating the app layout using Flutterflow's design tools. In total, there are 6 pages to the app, where going back and forth between pages is no concern.

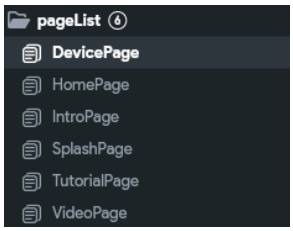
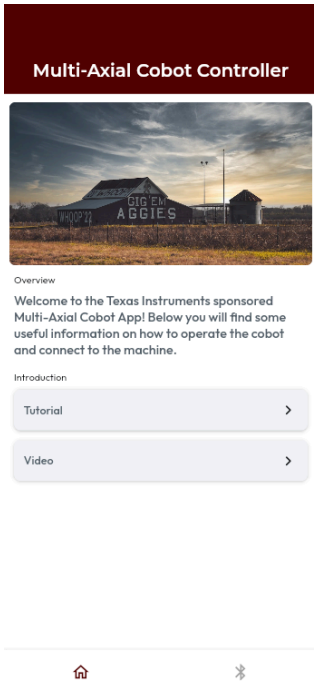
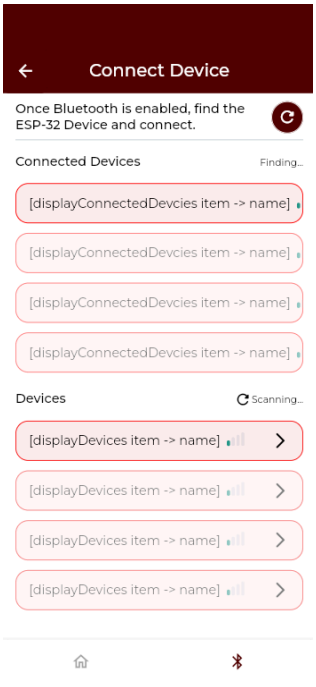
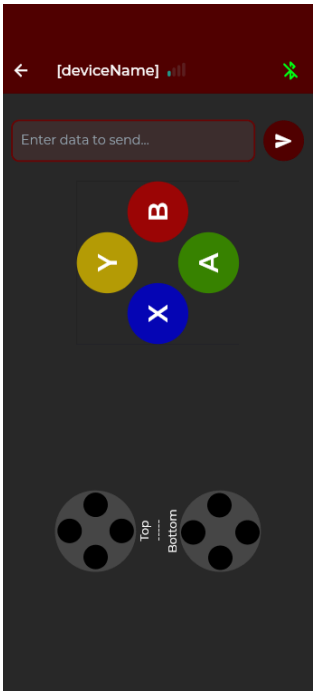


Figure 58: Application Page List

The next three criteria of the app were completed by splitting the tasks into their respective pages. The Intro Page is the first page the user will see when the application is opened, as it is where navigation to all other necessary pages is possible. The Tutorial and Video Pages are where users will be able to access all important information on how to use the cobot and the application (will be completed in 404 after Cobot Assembly). The Home Page is where the Bluetooth connection will be established. The User will enable Bluetooth, search devices, and connect all on this page. Finally is the Device Page, where the User Interface for the Cobot is located. Data can be sent from this page in either button input or string format.

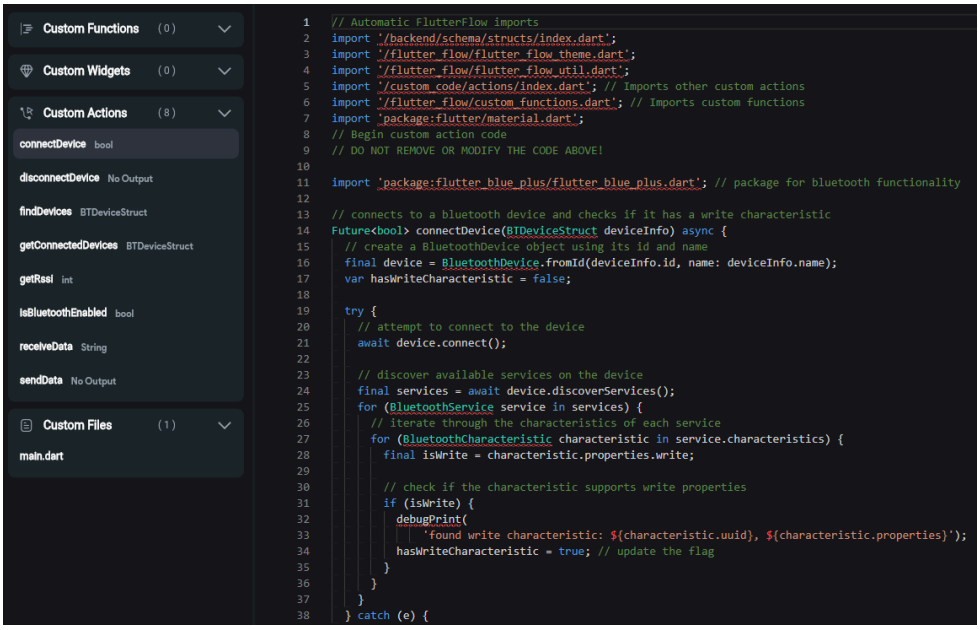






Figures 59,60,61: Intro Page, Home Page, and Device Page (respectively)

Additionally, the Bluetooth aspect required numerous steps to ensure the ability to enable Bluetooth, search for devices, connect to a device, and send data. This was done by creating 8 separate functions within Flutterflow that each tackles a different component of stable and secure Bluetooth connectivity. Figure 62 shows what the Flutterflow functions look like, and in this case, the connectDevices function. Figure 63 shows how some of the actions interact with each other to get the result that is intended. Table 11 lists the functions and a summary of what each of them accomplishes.



Figures 62, 63: Bluetooth Function UI, Block Diagram of Function Overlap

Function	Description
connectDevice	Establishes a connection with a Bluetooth device
disconnectDevice	Disconnects the currently connected device
findDevices	Scans for nearby Bluetooth devices
getConnectedDevices	Creates a list of currently connected devices
getRssi	Returns the signal strength of the Bluetooth connection
isBluetoothEnabled	Check if Bluetooth is enabled on the device
receiveData	Receives data from the device
sendData	Sends data to the device

Table 11: Bluetooth Functions & their Functionality

5.2.2.2. Mobile Application (App Store / Mobile Application Deployment)

Once the application was developed and full functionality was ensured, getting the application onto the mobile device was the next step. This was accomplished by creating an Apple Developer Account and registering to be affiliated with Apple's App Store policies. Once that was completed, the application was sent into Beta deployment through Apple's "Test-Flight" program, which allows the App to be distributed and downloaded on mobile devices linked to anyone who has access to the application. Below is the deployment of the application through the web interface, as well as directly on the mobile device. Additionally, Figure 65 shows how the application was successfully installed on two different devices through the TestFlight program: an iPad Air 4th Gen & an iPhone 16 Pro running iOS 18.0.1.

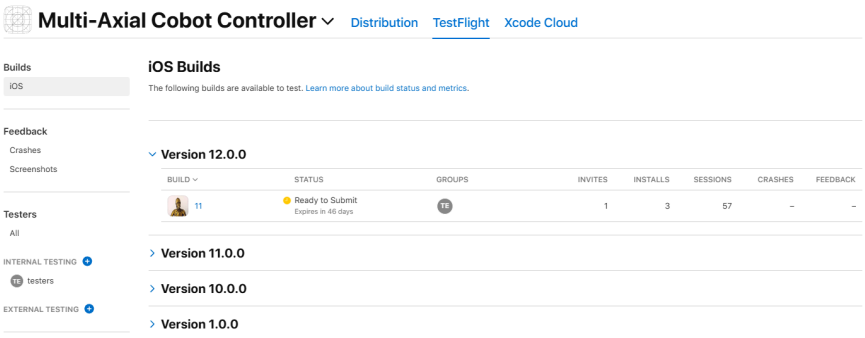


Figure 64: Mobile Application Deployment via TestFlight

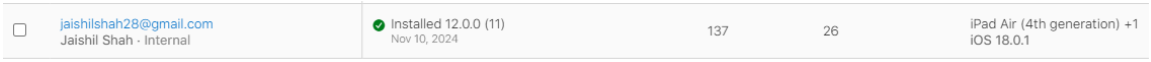


Figure 65: Mobile Application Deployment to iPad and iPhone

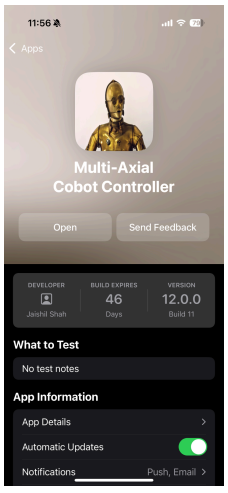


Figure 66: Mobile Application on iPhone

5.2.3. PCB Layout Breakdown

This section is the first of many that detail the second half of this subsystem. The previous sections explained the User interface, while the following sections will go over the Wireless Connection itself.

5.2.3.1. ESP-32 Chip

The largest and most important component in the PCB schematic is the ESP-32 chip. Bluetooth communication resides within the chip and is essential in making the PCB function as intended. This module contains pull-up and current-limiting resistors, capacitors in charge of power stabilization, and interfaces that enable FTDI communication. Some additional functionalities that this module houses include USB data transfer, GPIO operations, serial communication, and debugging/programming communication that is in charge of setting up the ESP-32.

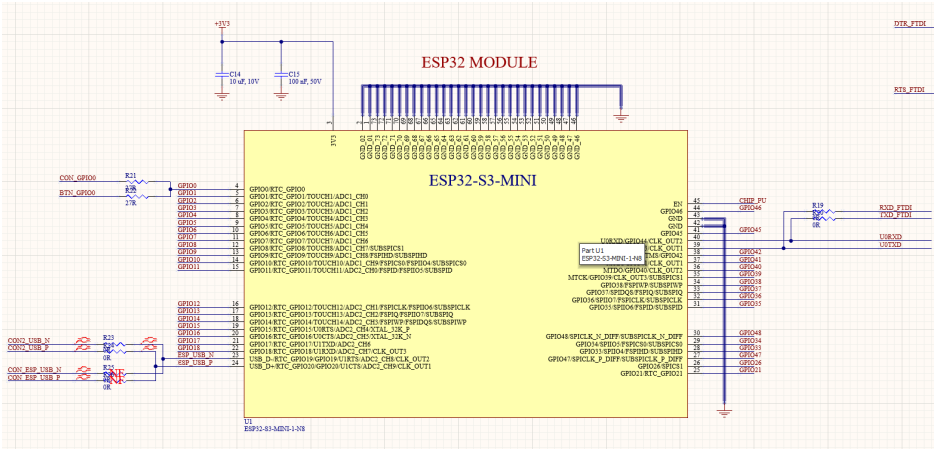


Figure 67: ESP-32-S3-MINI Module Schematic

5.2.3.2. GPIO Headers

This set of headers provides access to the GPIO pins and thus is connected to the ESP-32 module. Each header contains access the the pins, power lines (+3.3V and +5V), and essential ground connections. Additionally, decoupling capacitors of 100nF have been added to ensure power stability and minimize the signal noise along the power supply lines. These headers allow for the interfacing of external components, making this design both adaptable and versatile as it can work with other devices.

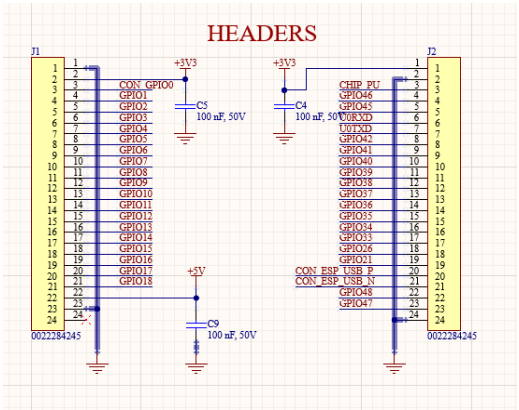


Figure 68: GPIO PIN Headers Schematic

5.2.3.3. Power Selection & Voltage Regulation

This schematic details the circuitry responsible for power and voltage regulation. There is a 5V jumper connection that can select between two 5V sources. A 5V to 3.3V regulator based on the TLV1117-33 device is responsible for converting a 5V input to 3.3 V, with their respective decoupling capacitors. This allows stability and noise reduction. Additionally, there is a 3.3V power selection that allows the selection of a 3.3V output for more downstream components. This part of the schematic is essential for proper voltage levels and flexibility based on the operation of the ESP-32.

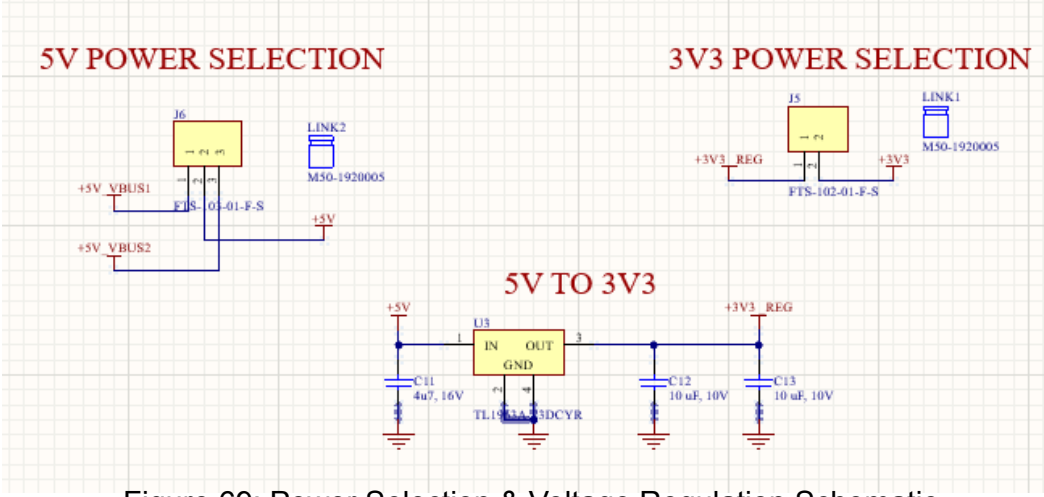


Figure 69: Power Selection & Voltage Regulation Schematic

5.2.3.4. Boot/User & Reset Buttons

The boot/user button allows new firmware to update the board, and the reset button allows power to temporarily be cut off from the board. This allows the board to be reset during debugging and to program the board efficiently.

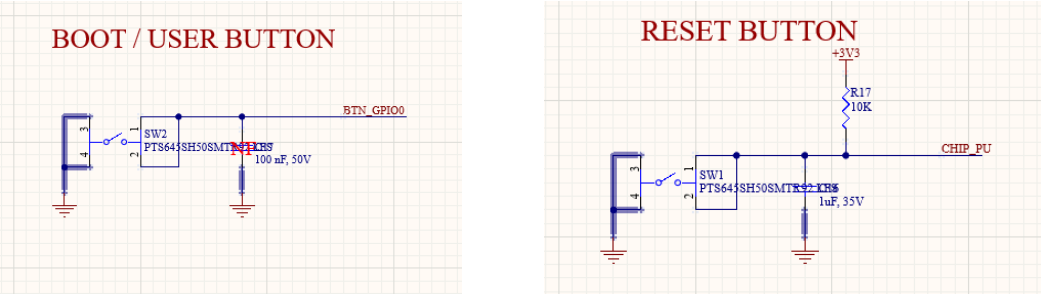


Figure 70: Boot/User & Reset Button Schematics

5.2.3.5. USB Connectors

This schematic displays the USB-C connectivity as well as the USB-UART conversion circuit for this PCB design. Two USB-C connectors handle power and data signals with their respective decoupling capacitors. The USB TO UART module is based on the FT2312X chip and facilitates the communication between USB and microcontroller via UART. It contains power regulation and level shifting, as well as VBUS sensing if it is ever needed. The bottom USB connector is what allows a device to connect directly to the ESP-32 Chip for debugging and programming. It also can function as a power connection for the board. Both of these connectors allow flexible communication and allow connection to and from external devices.

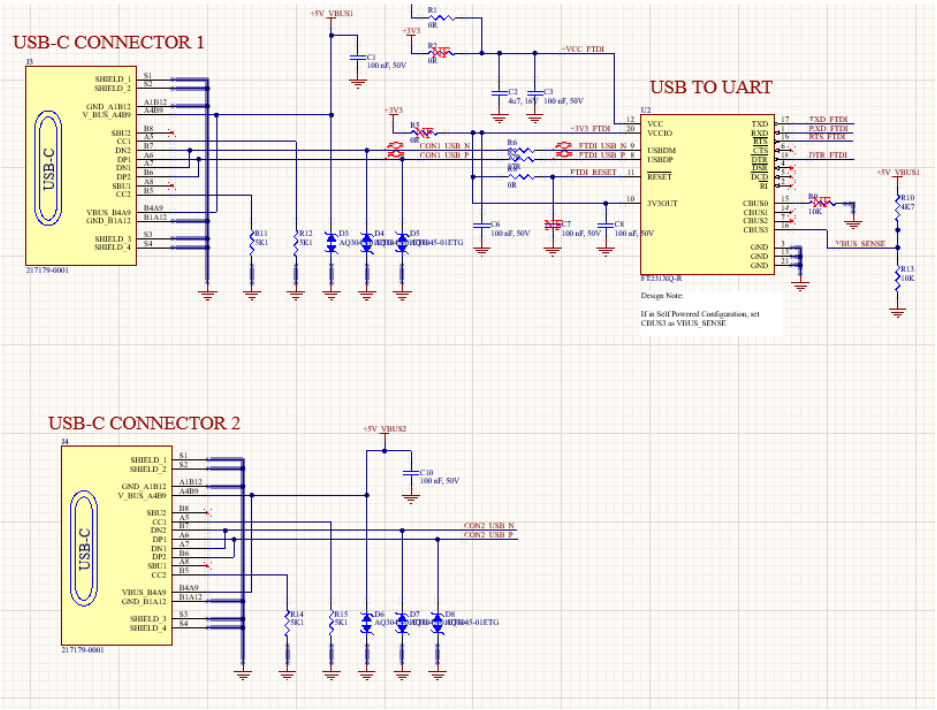


Figure 71: USB Connector Schematics

5.2.3.6. Miscellaneous Components

The left schematic shows the two LEDs that can display a proper connection as well as changes that are made or a successful Bluetooth pairing. The schematic on the right is in charge of differential pair routing, impedance matching, and controlled trace length. To summarize, this schematic will ensure signal integrity and avoid issues regarding noise or signal loss.

Figure 72: Miscellaneous Schematics

5.2.4. GPIO Pin Layout

The table below shows the total number of pins (73) and what each one corresponds to. Notable pins include 39 and 40, which are the TX and RX pins respectively. These are the pins I used to communicate between two ESP-32 devices and what will eventually be used for communication to the C2000. There is compatibility with numerous different devices, such as screens, external LEDs, and many more. For this project, a majority of these pins will not be used, but are great additions because it allows for ideas to be expanded on and is adaptive to any changes that may be present in the future.

1	GND_01	38	MTMS/GPIO42
2	GND_02	39	U0TXD/GPIO43/CLK_OUT1
3	3V3	40	U0RXD/GPIO44/CLK_OUT2
4	GPIO0/RTC_GPIO0	41	GPIO45
5	GPIO1/RTC_GPIO1/TOUCH1/ADC1_CH0	42	GND
6	GPIO2/RTC_GPIO2/TOUCH2/ADC1_CH1	43	GND
7	GPIO3/RTC_GPIO3/TOUCH3/ADC1_CH2	44	GPIO46
8	GPIO4/RTC_GPIO4/TOUCH4/ADC1_CH3	45	EN
9	GPIO5/RTC_GPIO5/TOUCH5/ADC1_CH4	46	GND_46
10	GPIO6/RTC_GPIO6/TOUCH6/ADC1_CH5	47	GND_47
11	GPIO7/RTC_GPIO7/TOUCH7/ADC1_CH6	48	GND_48
12	GPIO8/RTC_GPIO8/TOUCH8/ADC1_CH7/SUBSPICS1	49	GND_49
13	GPIO9/RTC_GPIO9/TOUCH9/ADC1_CH8/FSPIHD/SUBSPIHD	50	GND_50
14	GPIO10/RTC_GPIO10/TOUCH10/ADC1_CH9/FSPICS0/FSPIIO4/SUBSPICS0	51	GND_51
15	GPIO11/RTC_GPIO11/TOUCH11/ADC2_CH0/FSPID/FSPIIO5/SUBSPID	52	GND_52
16	GPIO12/RTC_GPIO12/TOUCH12/ADC2_CH1/FSPICLK/FSPIIO6/SUBSPICLK	53	GND_53
17	GPIO13/RTC_GPIO13/TOUCH13/ADC2_CH2/FSPIQ/FSPIIO7/SUBSPIQ	54	GND_54
18	GPIO14/RTC_GPIO14/TOUCH14/ADC2_CH3/FSPIWP/FSPIDQS/SUBSPIWP	55	GND_55
19	GPIO15/RTC_GPIO15/U0RTS/ADC2_CH4/XTAL_32K_P	56	GND_56
20	GPIO16/RTC_GPIO16/U0CTS/ADC2_CH5/XTAL_32K_N	57	GND_57
21	GPIO17/RTC_GPIO17/U1TXD/ADC2_CH6	58	GND_58
22	GPIO18/RTC_GPIO18/U1RXD/ADC2_CH7/CLK_OUT3	59	GND_59
23	USB_D-/RTC_GPIO19/GPIO19/U1RTS/ADC2_CH8/CLK_OUT2	60	GND_60
24	USB_D+/RTC_GPIO20/GPIO20/U1CTS/ADC2_CH9/CLK_OUT1	61	GND_61
25	GPIO21/RTC_GPIO21	62	GND_62
26	GPIO26/SPICS1	63	GND_63
27	GPIO47/SPICLK_P_DIFF/SUBSPICLK_P_DIFF	64	GND_64
28	GPIO33/SPIIO4/FSPIHD/SUBSPIHD	65	GND_65
29	GPIO34/SPIIO5/FSPICS0/SUBSPICS0	66	GND_66
30	GPIO48/SPICLK_N_DIFF/SUBSPICLK_N_DIFF	67	GND_67
31	GPIO35/SPIIO6/FSPID/SUBSPID	68	GND_68
32	GPIO36/SPIIO7/FSPICLK/SUBSPICLK	69	GND_69
33	GPIO37/SPIDQS/FSPIQ/SUBSPIQ	70	GND_70
34	GPIO38/FSPIWP/SUBSPIWP	71	GND_71
35	MTCK/GPIO39/CLK_OUT3/SUBSPICS1	72	GND_72
36	MTDO/GPIO40/CLK_OUT2	73	GND_73
37	MTDI/GPIO41/CLK_OUT1		

Figure 73: GPIO Pin-Out Sheet

5.2.5. Arduino IDE

Writing the Arduino code was done before the PCB board had been shipped and produced as tinkering with a prebuilt ESP-32 was done in the meantime. Arduino IDE allowed connection to the ESP-32, programming the device, and keep the connection secure through USB. The program that was written allows a user to connect their mobile device to the ESP-32 and send signals from the virtual controller screen. The ESP-32 was able to receive these inputs as shown in the serial monitor located in Arduino IDE. Additionally, a screen was connected to the ESP-32 and was able to show what inputs were sent and received in real-time.

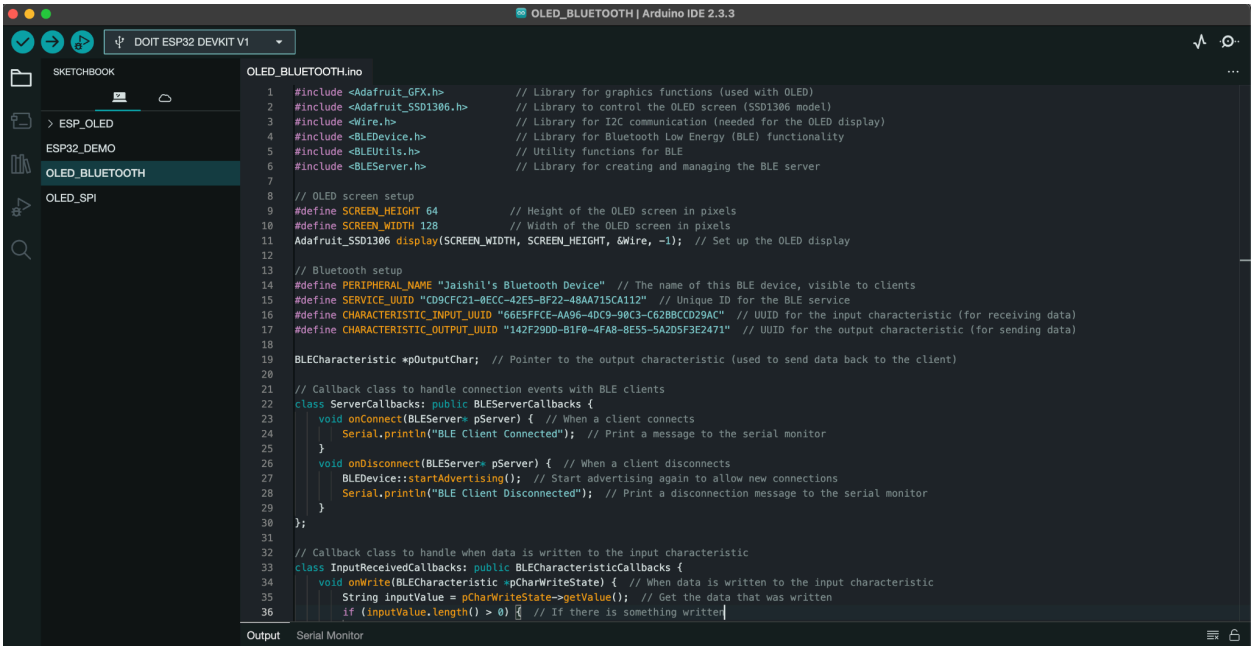


Figure 74: Arduino IDE interface

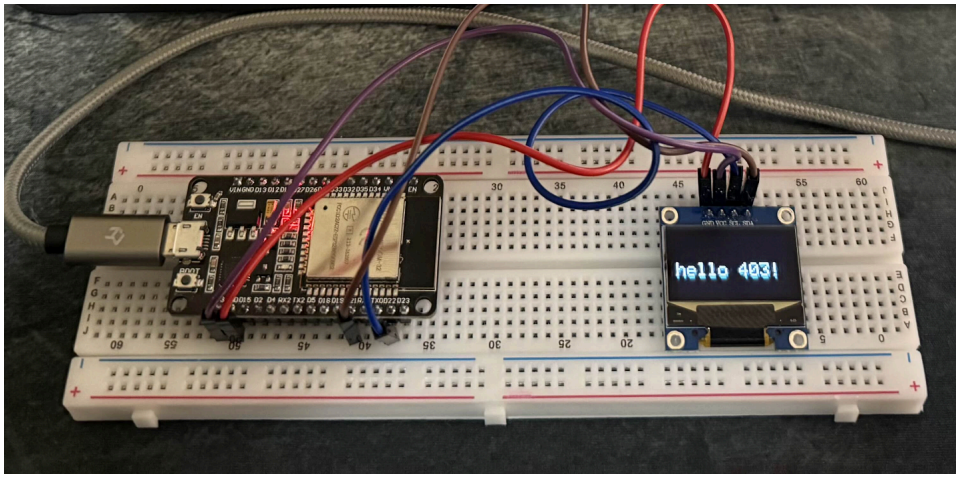


Figure 75: Mobile Input → ESP-32 → OLED screen

5.2.6. Communicating via high/low GPIO Pins & Physical Board Validation

With all the previous sections within this subsystem in mind, it is time to put them together and create a system where all these parts are able to work together.

Below is a four-part system that is able to take an input, communicate via two separate devices, and display the input on a screen. Here is a step-by-step process on how this works and how this implementation will work in ECEN 404. First is the mobile input. Once the Bluetooth device is connected on the mobile application, it is then connected to the student-developed PCB (green board on the image below). That PCB is connected to the black PCB (amazon-ordered) via TX/RX pins which can be depicted by the yellow and green jumper cables. Both PCBs are connected to separate computers, both running and communicating through Arduino IDE. Once the black ESP-32 has received the data from the mobile device, it takes that and transmits a signal to the OLED screen.

This mimics the intended result for cobot design, as instead of the black PCB, there will be a C2000 MCU. Since TX/RX is utilized here, the goal is to use the same process for the MCU. UART was unfortunately not functioning as intended, and thus on this green PCB, that connector is not even soldered on. In 404, tinkering will take place to see if UART is still a feasible solution, but for now, TX/RX pins are confirmed to work.

In order to simplify the process when combining the subsystems, the inputs will be simplified into 6-10 specific inputs that can be read, and be assigned their own GPIO pin. Since this PCB was designed with adaptability in mind, these extra GPIO pins will come in handy.

Additionally, the breadboard and OLED screen were temporary and only there for this semester to display what a MCU will receive when the cobot is designed and created. The screen will be replaced with the motor & motor drivers so that in 404, the process goes: Mobile Device → ESP-32 → MCU → Motor Drivers.

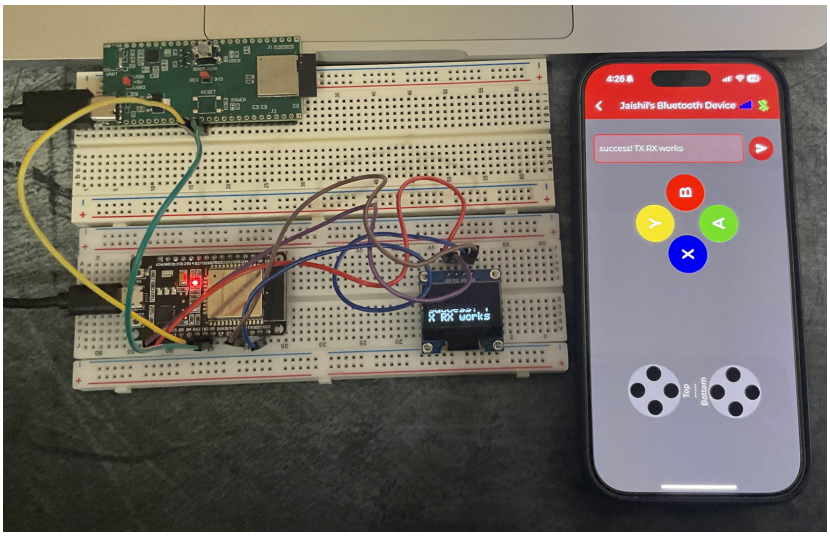


Figure 76: Full Subsystem Functionality Test

Below is the 3D render of the PCB, as well as the necessary components solder onto the physical board. The Reset Button was not soldered on as it was not needed for this semester's functionality, and the UART connector was not installed as it did not work on another board that was previously soldered. (This board below is the third of five boards sent over from JLCPCB. The first was damaged as a trace ripped out, and the second had too much solder that touched where it should not have. The second was the one in which UART was attempted, and it did not function properly. Thus, it was ignored for this rendition of the PCB).

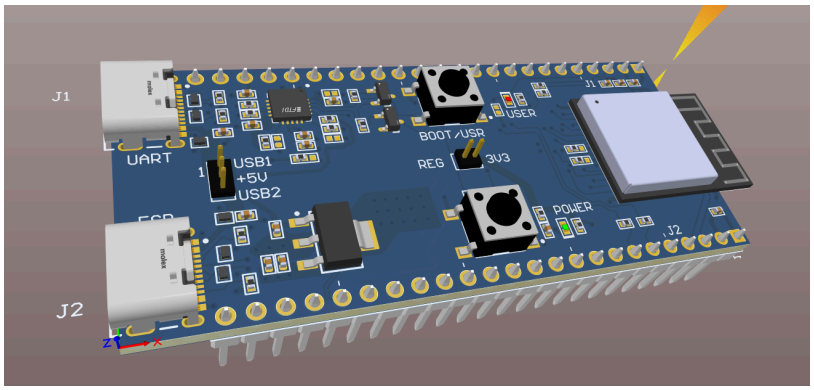


Figure 77: 3D render of ESP-32 PCB

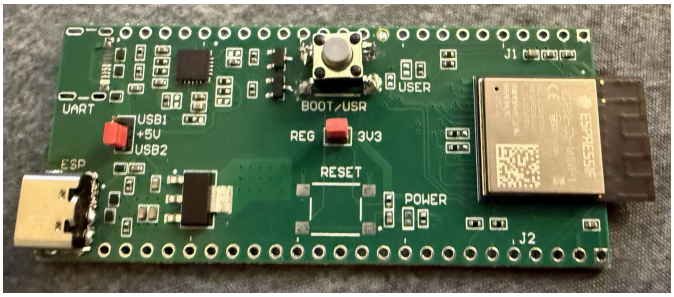


Figure 78: ESP-32 PCB used in four-part system

```
Received Value: success!
Received Value: success! TX RX works!
Received Value: success! TX RX works
Received Value: success! TX RX works
```

Figure 79: Arduino Serial Monitor displaying Signals Received

5.3. Subsystem Validation

Test Name	Successful	Criteria	Proof of Functionality
Wireless Connectivity	Yes	Mobile app. stays connected for a minimum of 5 minutes and connects seamlessly.	Figure 75,79
GUI Operational	Yes	Connection is stable, input lag is minimal, and outputs are properly received.	Figure 76,79
Functionality Messages	Yes	Functionality messages can be sent from the ESP-32 via GPIO Pins	Figure 79
Button Functionality	Yes	All User inputs and interactions work as intended without fault.	Figure 76
Application Device Deployment	Yes	The application is deployed onto the device and is easily accessible to multiple users. Is also multi-device compatible	Figure 65,75

Table 12: Subsystem Validation - Jaishil Shah

5.4. Subsystem Conclusion

Although a proper connection to the ESP-32 PCB that was designed was not established during the final demo, successful connection was established afterward and all criteria in the validation are met. In conclusion, this subsystem can successfully perform all required tasks and ensure that proper communication with other hardware and software will be successful during next semester's assembly. All validation tests passed and all deliverables included in the execution plan were completed on time, excluding the PCB design not functioning properly. Once all components are fixed and started working as intended, this subsystem can be deemed ready for ECEN 404 integration and seamless cooperation with the other three subsystems within the project. The next steps include communicating to the MCU C2000 and from there communicating to the motors & their drivers, as well as making sure the 3V3 & 5V power can enable the ESP-32's operations. In terms of direct work that needs to be accomplished next semester, a proper fabrication of the enclosures and cobot design needs to be completed, as well as complexity and finishing touches to the mobile application.

11: Multi-Axis Cobot For Factory Automation

Adrian Guzman, Emily Hamsa, Ethan Woods, Jaishil Shah

EXECUTION PLAN

Final Report
11: Multi-Axis Cobot For Factory Automation

(zoom in to view)

	Titles	8/27/24	9/3/24	9/10/24	9/17/24	9/24/24	10/1/24	10/8/24	10/15/24	10/22/24	10/29/24	11/5/24	11/12/24	11/19/24	11/26/24	12/3/24	Date Completed
Key:	**Defining Project Scope (8/27)**																8/27/24
Black - ALL	**Subsystem Division / Brainstorming (9/3)**													Not Started			9/3/24
Green - Adrian Guzman	**Creating a Budget Parts List (9/10)**													In Progress			9/12/24
Purple - Jaishil Shah	**Subsystem Software Initialization (9/10/24)**													Completed			9/17/24
Blue - Emily Hamsa	**CONOPS Report (9/15)**													Behind Sch.			9/12/24
Red - Ethan Woods	**Creating a (TI) Parts List (9/17)**																9/17/24
	Swift Compatability (9/17)																9/17/24
	Motor Selection Finalization (9/20)																10/8
	Complete Subsystem Introduction Project (9/24)																9/24
	Subsystem Introduction Project (Emily) (9/27)																9/30
	Project Selection Presentation (9/23)																9/23
	FSR, IDC, Milestones and Validation plan (9/26)																6/26
	Parts Selection Finalization and Battery Sizing (9/27)																10/13
	Competed PMS PCB Design and Ordering (10/11)																10/23
	Mid-Term Presentation (10/14)																10/14
	PCB Design with MCU (10/18)																10/22
	Order PCB Board & Components (10/18)																10/22
	Motor/Motor Driver Testing with EVM Board (10/18)																10/8
	Apple Developer Status / Flutterflow Crash Course (10/20)																10/15
	Subsystem Introduction Project (Jaishil) (10/20)																10/17
	Subsystem Introduction Project (Ethan) (10/20)																10/15
	Software on eval board (LED Blink & Motor Spin) (10/29)																10/25
	Bluetooth Integration & ESP-32 Communication (11/5)																11/5
	PCB Assembly and Soldering (11/08)																11/08
	PCB Testing and Debugging (11/22)																11/25
	PCB Assembly & Soldering (11/4)																11/05
	PCB Design and Ordering (11/4)																11/1
	ESP-32 PCB Design and Ordering (11/5)																11/5
	Final Presentation (11/11)																11/11
	Verify signals on PCB board																11/08
	JTAG debugger connection to MCU																11/12
	LED Blink/Stepper Motor on PCB																11/19
	Soldered and Functional ESP-32 (11/22)																11/22
	Solder Phase A, Phase B, and Phase C Boards (11/19)																11/12
	PWM Signal Generation & ESP-32 Communication																11/24
	Generate Phase Signals on Different Boards (11/25)																11/18
	Subsystem Demonstration (Jaishil) (11/26)																11/26
	Subsystem Demonstration (Ethan) (11/26)																11/26
	Subsystem Demonstration (Emily) (11/26)																11/26
	Subsystem Demonstration (Adrian) (11/26)																11/26
	Final Report (12/5)																12/5

11: Multi-Axis Cobot For Factory Automation

Adrian Guzman, Emily Hamsa, Ethan Woods, Jaishil Shah

VALIDATION PLAN

Final Report
11: Multi-Axis Cobot For Factory Automation

(zoom in to view)

Status Key:	Paragraph Number	Test Name	System Criteria	Criteria for Success	Status	Responsible Engineer(s)
In Progress		**Joint Effort**				
Completed	3.2.2.1	Mounting	When system is lifting the maximum payload, it stays mounted to its surface without tipping.	Attatch weights to arm of Cobot, measure the maximum payload before system tips over.		Full Team
Late	3.2.1.1	C2000 EVM Wakeup	Launch EVM and connection is successful	EVM turns on, CCS software identifies device, simple command run.		Full Team
TBD	3.2.2.2	System Precision	Physical movement and load capacity functionality is able to move objects precisely and accurately.	System axes can move at minimum 2 degrees, and move objects 2 inches on the table.		Full Team
	3.2.2.3	Structure Integrity	All components and joints are structrually sound	When cobot is lifting maximum weight, the physical design is able to support the weight of both itself and the load.		Full Team
		Power Subsystem				
	3.2.1.6	Schematic & PCB Creation	Power management system PCB is fully designed in Altium	Completion of all schematics in Altium and is validated, checked for validation errors, and DRC has no errors.		Emily Hamsa
	3.2.3.1.2	Battery Sizing	Choose battery size that will adequetly power cobot based on load list	Completion of battery sizing using formulas and load list to determine appropriate voltage and amp-hour sizing		Emily Hamsa
	3.2.3.1.1	Input Voltage	The battery will supply a constant DC 48V to the system at the input	Using a oscilloscope, test to ensure system input holds at 48 volts while battery is connected and measure noise		Emily Hamsa
	3.2.3.1.3	Voltage Step-Down	Power PCB will step-down voltage to the subsequent systems requirements	Using a multimeter, test each voltage rail and ensure it is holding correct voltage		Emily Hamsa
	3.2.3.2.5.	Output DC Voltage	The output terminals will hold a true DC output voltage.	Using a oscilloscope, test the output terminals and ensure that the output is truly DC		Emily Hamsa
		MCU/Connectivity Subsystem				
	3.2.1.3.1	Schematic & PCB Creation	Schematic and PCB footprint integrating MCU, motor drivers, and ESP32	Individual sheets within altium are completed and reviewed for accuracy, there are no errors in validation, and DRC comes back with zero errors.		Adrian Guzman
	3.2.1.2	MCU & Motor Driver	Test MCU and motor driver communication	Align with motor driving subsystem to ensure signals are accounted for. Integrate and show motor movement		Adrian Guzman, Ethan Woods
	3.2.3.2.1	LED Program	Flash the evaluation board MCU and get LED program working / GPIO (PWM) generation	Use the C2000 EVM and CCS software developer to create code that functions on EVM. This code will verify understanding of signals. (Ex. LED blinking & PWM generation)		Adrian Guzman
	3.2.3.1.4	MCU & Bluetooth	Test MCU and bluetooth (ESP32) communication	Align with wireless subsystem requirements, all necessary signals are accounted for and communication betweeen bluetooth can be displayed		Adrian Guzman, Jaishil Shah
	3.2.3.2.6	MCU Generates PWM Signals	The oscilloscope outputs PWM waveforms	Probe GPIO pins to visualize PWM output on the oscilloscope		Adrian Guzman
	3.2.3.1.7	C2000 MCU Input from ESP-32	The C2000 MCU will be capable of receiving input from ESP-32	Develop a test where an ESP-32 pin is set to high/low continusoutly. The C2000 will receive the input of this pin and toggle LED ON/OFF based on the ESP-32 input		Adrian Guzman
	3.2.1.3	JTAG Programming of MCU	The MCUs programming signals (TDI, TDO, TCK, TMS) are properly routed	The XDS-110 debugger, built into the launchpad will be used to program the MCU on the PCB. This is done by connecting the required signals (TDI, TDO, TCK, TMS) from the PCB to the launchpad. A program will be written in CCS.		Adrian Guzman

Status Key:						
		Motor Driver Subsystem				
In Progress	3.2.1.3.2	Schemating and PCB Creation	Creating Schematics and PCB boards for Individual Motors (5 Total)	All schematic sheets are created, validated, and DRC has zero errors.		Ethan Woods
Completed	3.2.3.2.2	PWM Phase Output from Motor Drivers	When the Motor Driver PCB is provided power and C2000 Launchpad connection, generate a Corresponding PWM Phase Signal	A generated PWM signal is readable and observable on oscilloscope or other measuring tool. One board can generate one signal.		Ethan Woods
Late	3.2.3.1.5	Motor Movement	Motor Drivers can send signals to move Motor through MCU	Align with MCU subsystem on requried signals for control, display movement through MCU control with motor drivers and motor.		Ethan Woods, Adrian Guzman
TBD	3.2.1.4	Precise Motor Control	Through encoders, Motors will move a few degrees at a time	Perform angular measurements on motor when spinning, record data.		Ethan Woods
	3.2.1.5	All Motor Movement	All motors can move through communication with the MCU and Motor Drivers	Motor functionality is precise and all movements correspond with the intended output		Ethan Woods
		Wireless Subsystem				
	3.2.3.2.2	Wireless Connectivity	Mobile app connection and communication between two devices via RX TX	Mobile app connection does not drop out more than 1 time in 5 minutes, and connects within 30 seconds. ESP-32 can communicate to external devices.		Jaishil Shah, Adrian Guzman
	3.2.3.1.6	GUI Operational	User Interface (application) is deployed to mobile device	Connection is stable, input lag is minimal, and outputs are properly received/translated		Jaishi Shah
	3.2.5	Application Device Deployment	More than one device can connect & Available to all iOS devices	The application is deployed onto the device & easily accessible to multiple users		Jaishil Shah
	3.2.3.1.7	Button Functionality	Button Input Functionality	Human interaction is received and works without fault		Jaishil Shah