Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

VFD Motor Control Introduction

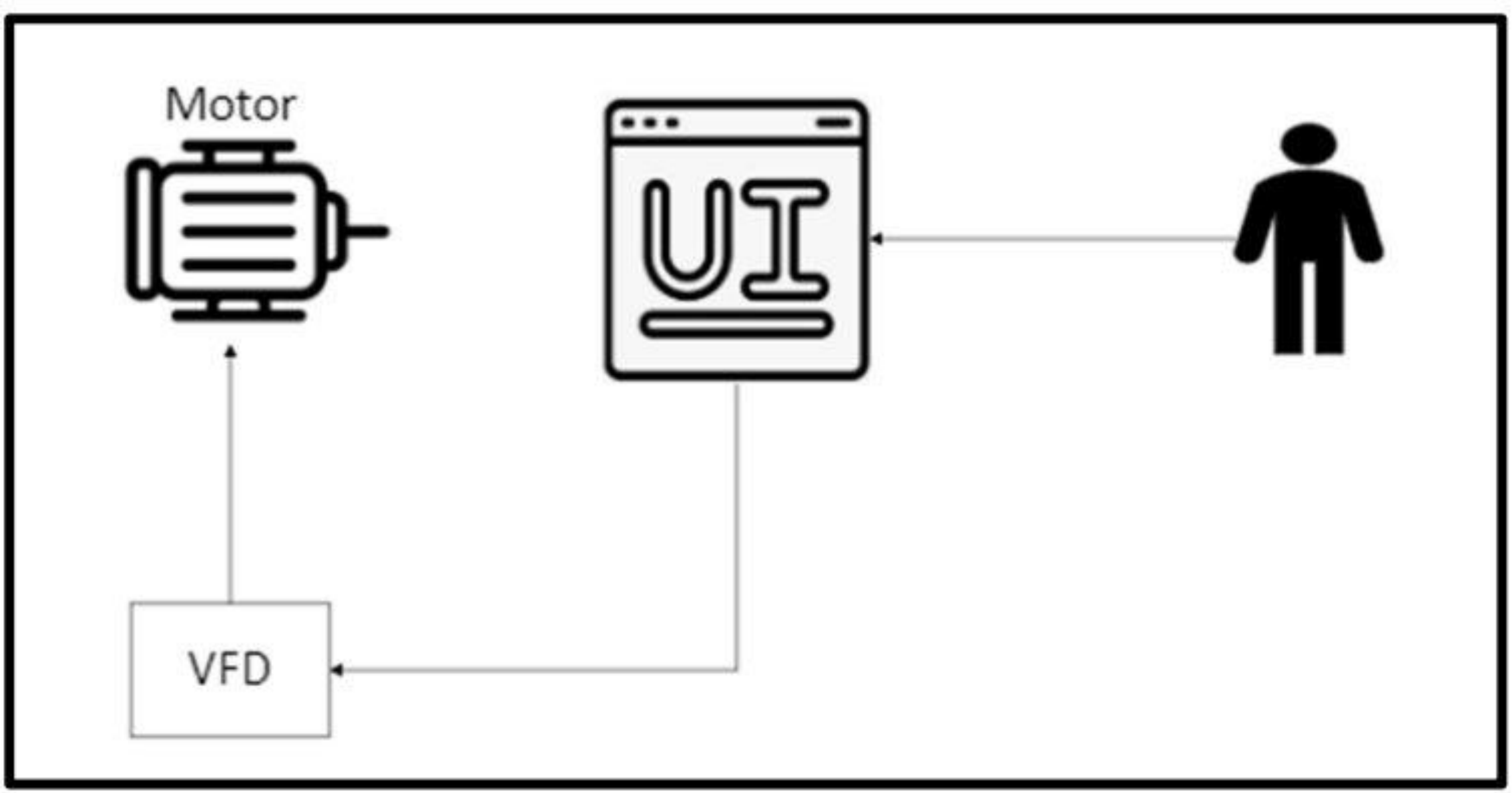
Mackenzie Miller
Andrew Nguyen
Aidan Rader
Ryan Regan

Overview

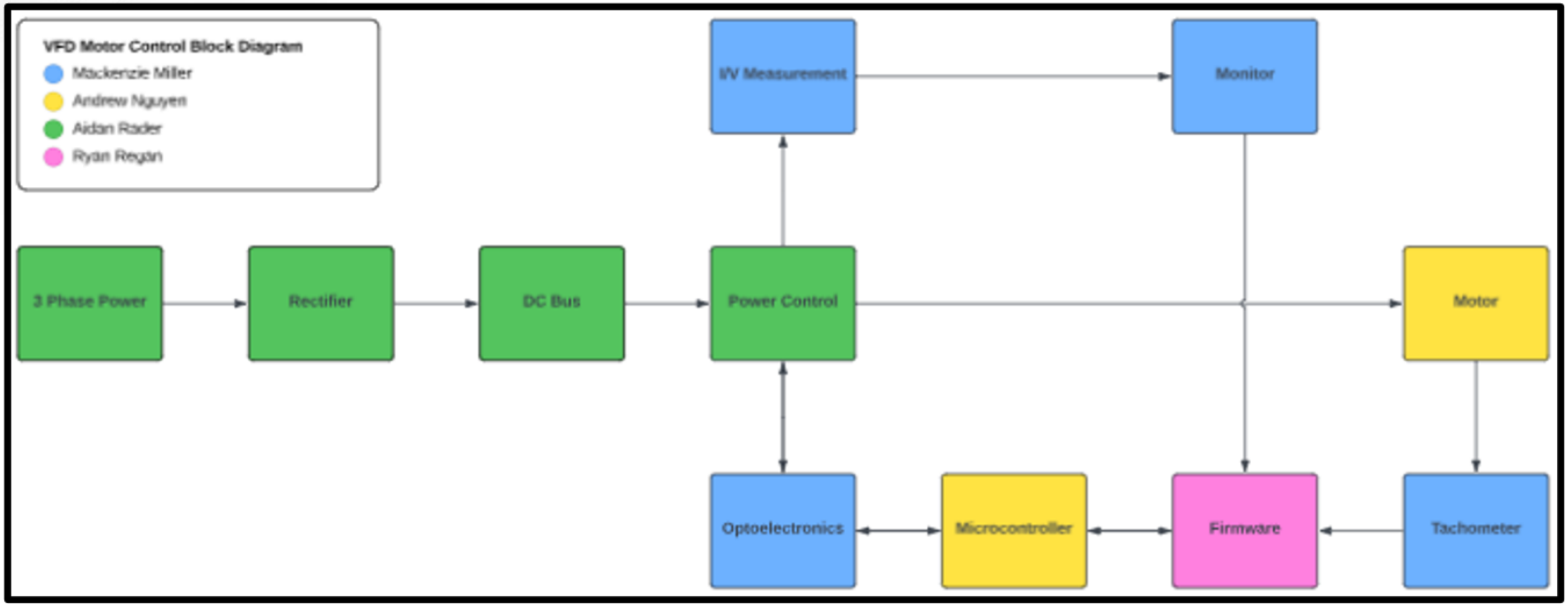
- Problem Statement: A motor control system is needed for an AC induction motor. Traditional motor control systems cannot adjust to varying load demands, resulting in poor energy efficiency, excessive heat generation, and premature component failure.

- Solution Proposal: Develop a Variable Frequency Drive (VFD) motor control system to adjust frequency and voltage to load demands, resulting in improved motor controllability, optimized energy efficiency, enhanced safety, and extended component lifespan.

System Overview



System Overview



VFD Motor Control Block Diagram

- 🔵 Mackenzie Miller
- 🟡 Andrew Nguyen
- 🟢 Aidan Rader
- 🟣 Ryan Regan

I/V Measurement

Monitor

3 Phase Power

Rectifier

DC Bus

Power Control

Motor

Optoelectronics

Microcontroller

Firmware

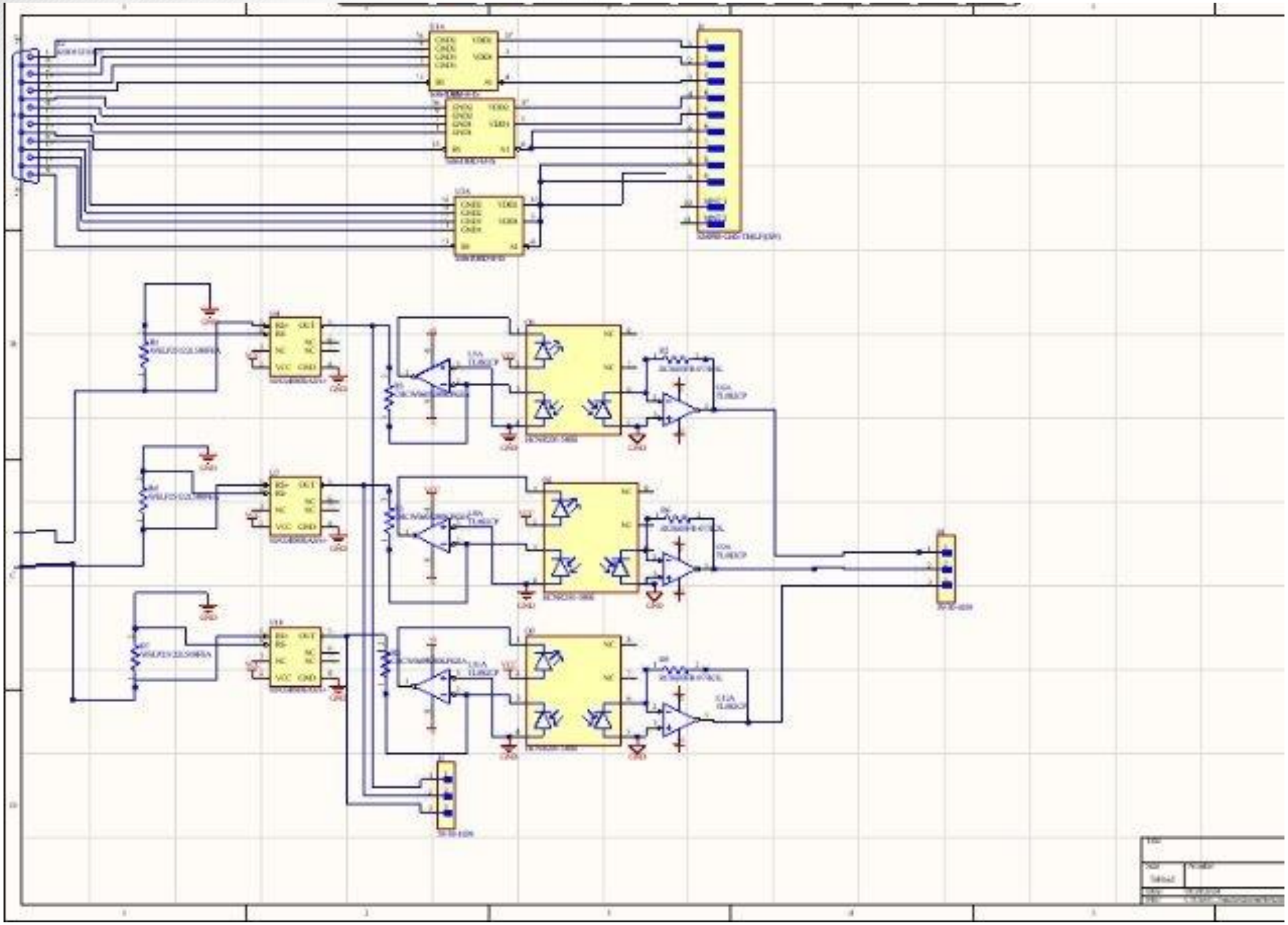
Tachometer

Optoelectronics

Mackenzie Miller

Accomplishments since the last presentation <12> hrs	Ongoing progress/problems and plans until the next presentation
Solidified part numbers and added to part order sheet Ordered first round of parts Finished PCB schematic Started PCB layout	Confirm with groupmates which connectors we want to use Finish PCB layout Start to route Validate PCB and resolve the errors Order PCB

Optoelectronics circuit



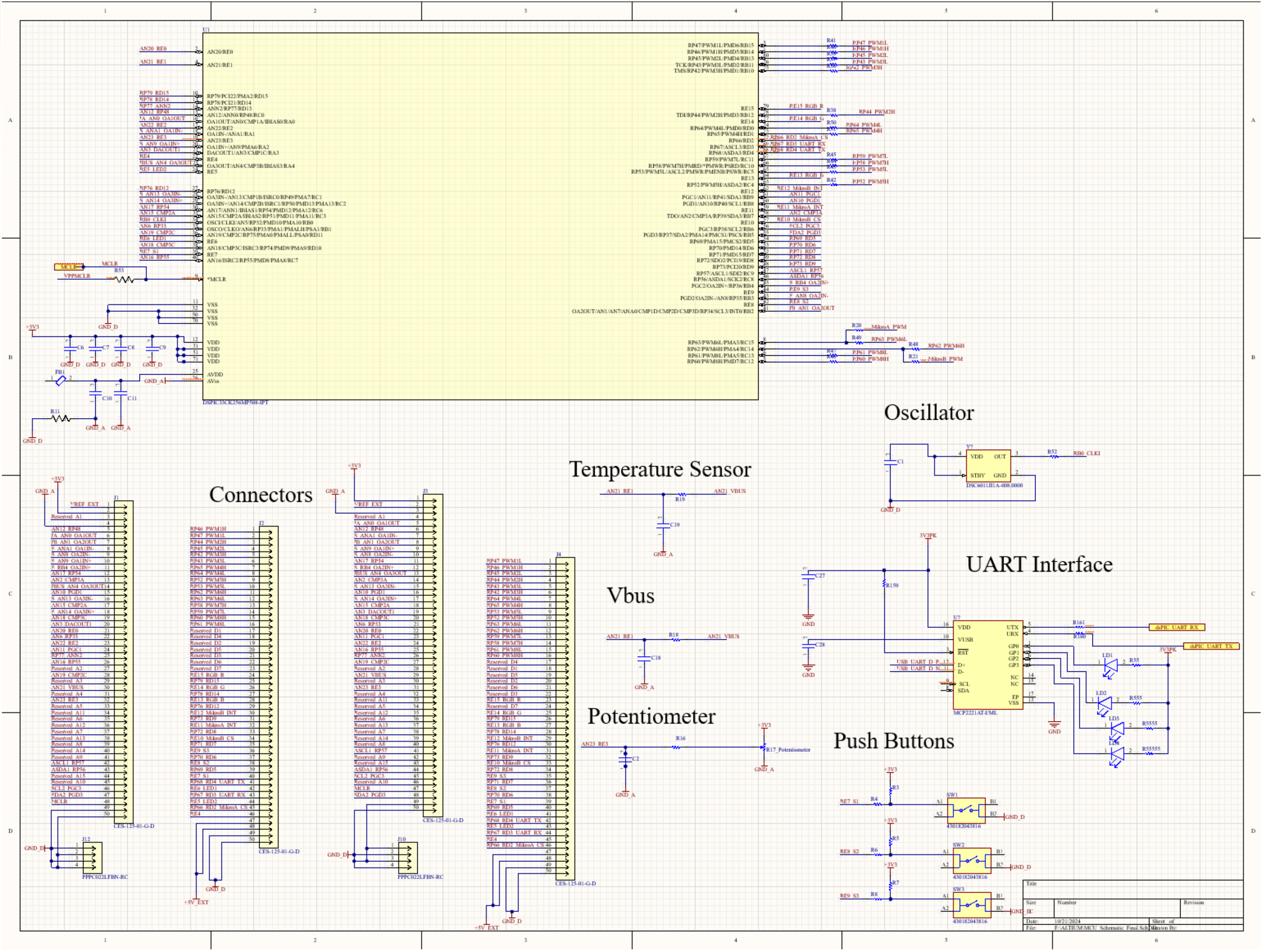
Microcontroller

Andrew Nguyen

Accomplishments since the last presentation <30> hrs	Ongoing progress/problems and plans until the next presentation
Updated parts order spreadsheet Put proper symbol/footprints on schematic Created schematics for MCU, USB to UART interface, potentiometer, push buttons, LEDs, connectors/headers, temperature sensor, oscillator On final part of schematics/validating schematic for PCB routing and ensuring net labels are correct	Finalize USB to UART schematic. Validate schematic and resolve all errors. Route and order PCB and all necessary parts

Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

Current MCU Schematic



Power

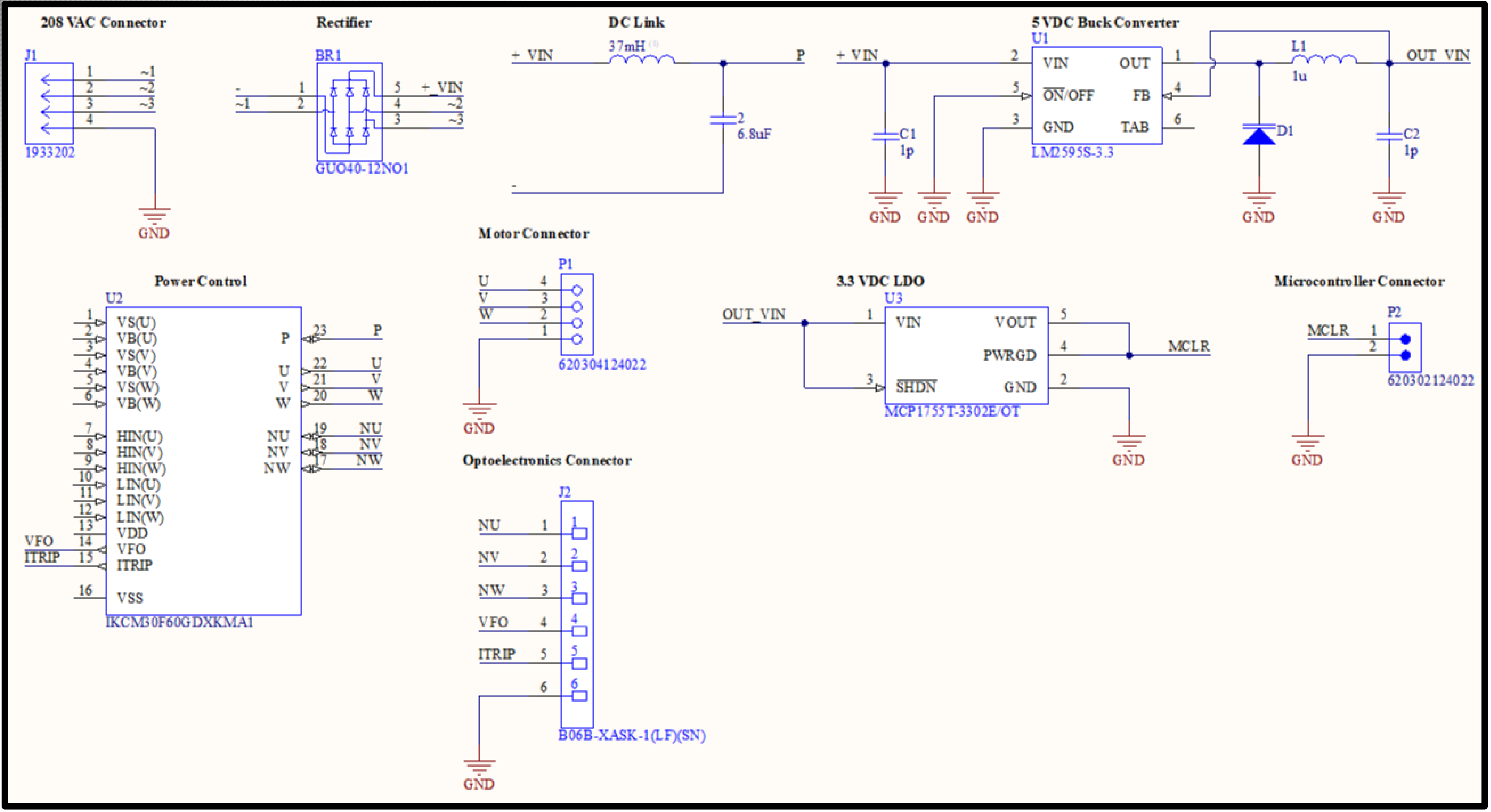
Aidan Rader

Accomplishments since the last presentation 28 hrs	Ongoing progress/problems and plans until the next presentation
-Completed subsystem introduction project (9 hrs) -Ordered parts (1.5 hrs) -Completed rectifier schematic (6.5 hrs) -Completed DC link calculations & schematic (3 hrs) -Completed power control schematic (3 hrs) -Started MCU power supply schematic (2 hrs) -Completed Project Update Presentation (3 hrs)	Ongoing: -MCU power supply schematic -Connector selection -Order more parts -PCB layout Plans: -Complete PCB layout -Complete PCB order -Complete PCB assembly

Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

Power

Aidan Rader

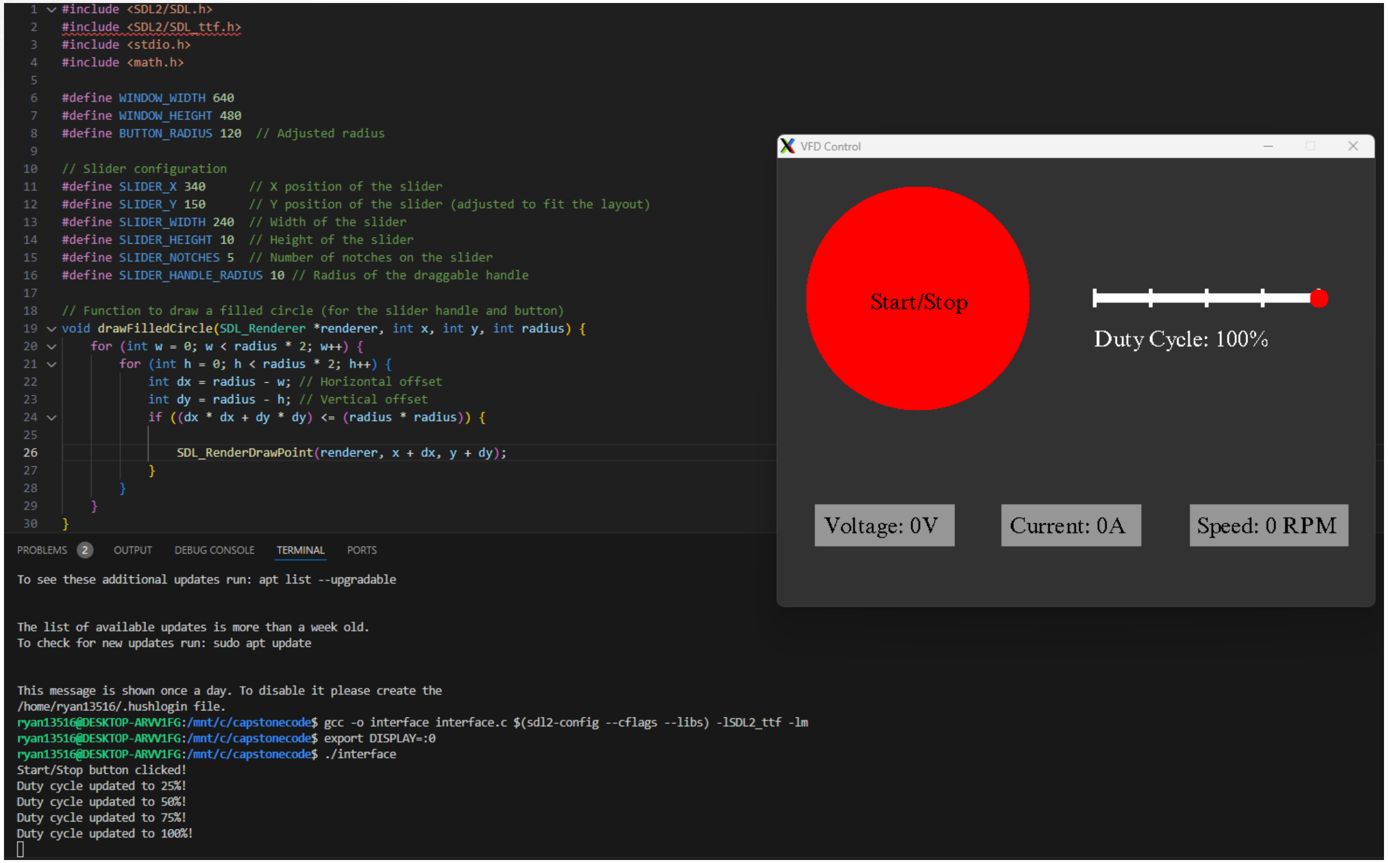


Firmware

Ryan Regan

Accomplishments since the last presentation 2 0 hours	Ongoing progress/problems and plans until the next presentation
- Working GUI demo - Rough outline of final project code - Implementation of buttons and LEDs on Development Board as well as print statements used for debugging - Progress on setting up ADC (Analog-DC) and PWM (Pulse-Width Modulation) modules for use in programming the potentiometer	- Research and finish programming the modules and writing the commands necessary to demonstrate the potentiometer through dimming the RGB LEGS - Determine a better toggling method of using the buttons to reduce potential user error if pressed too fast or too slow - Research on whether implementation of the GUI in MPLab is possible and/or necessary

VSCode GUI Demonstration



```c
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <stdio.h>
#include <math.h>

#define WINDOW_WIDTH 640
#define WINDOW_HEIGHT 480
#define BUTTON_RADIUS 120  // Adjusted radius

// Slider configuration
#define SLIDER_X 340      // X position of the slider
#define SLIDER_Y 150      // Y position of the slider (adjusted to fit the layout)
#define SLIDER_WIDTH 240  // Width of the slider
#define SLIDER_HEIGHT 10  // Height of the slider
#define SLIDER_NOTCHES 5  // Number of notches on the slider
#define SLIDER_HANDLE_RADIUS 10 // Radius of the draggable handle

// Function to draw a filled circle (for the slider handle and button)
void drawFilledCircle(SDL_Renderer *renderer, int x, int y, int radius) {
    for (int w = 0; w < radius * 2; w++) {
        for (int h = 0; h < radius * 2; h++) {
            int dx = radius - w; // Horizontal offset
            int dy = radius - h; // Vertical offset
            if ((dx * dx + dy * dy) <= (radius * radius)) {

                SDL_RenderDrawPoint(renderer, x + dx, y + dy);
            }
        }
    }
}
```

```
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


This message is shown once a day. To disable it please create the
/home/ryan13516/.hushlogin file.
ryan13516@DESKTOP-ARVV1FG:/mnt/c/capstonecode$ gcc -o interface interface.c $(sdl2-config --cflags --libs) -lSDL2_ttf -lm
ryan13516@DESKTOP-ARVV1FG:/mnt/c/capstonecode$ export DISPLAY=:0
ryan13516@DESKTOP-ARVV1FG:/mnt/c/capstonecode$ ./interface
Start/Stop button clicked!
Duty cycle updated to 25%!
Duty cycle updated to 50%!
Duty cycle updated to 75%!
Duty cycle updated to 100%!
```

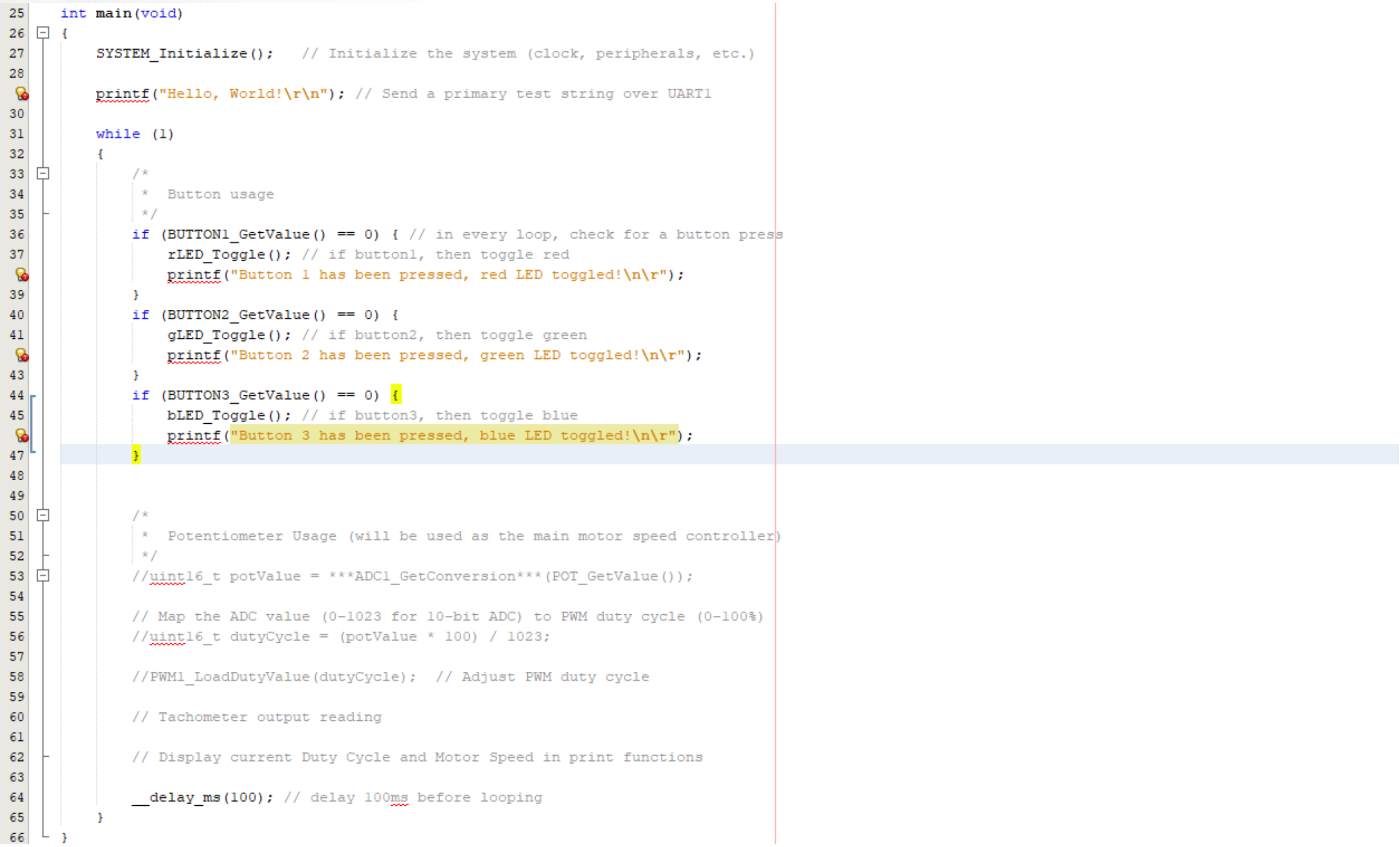
Duty Cycle: 100%

Voltage: 0V

Current: 0A

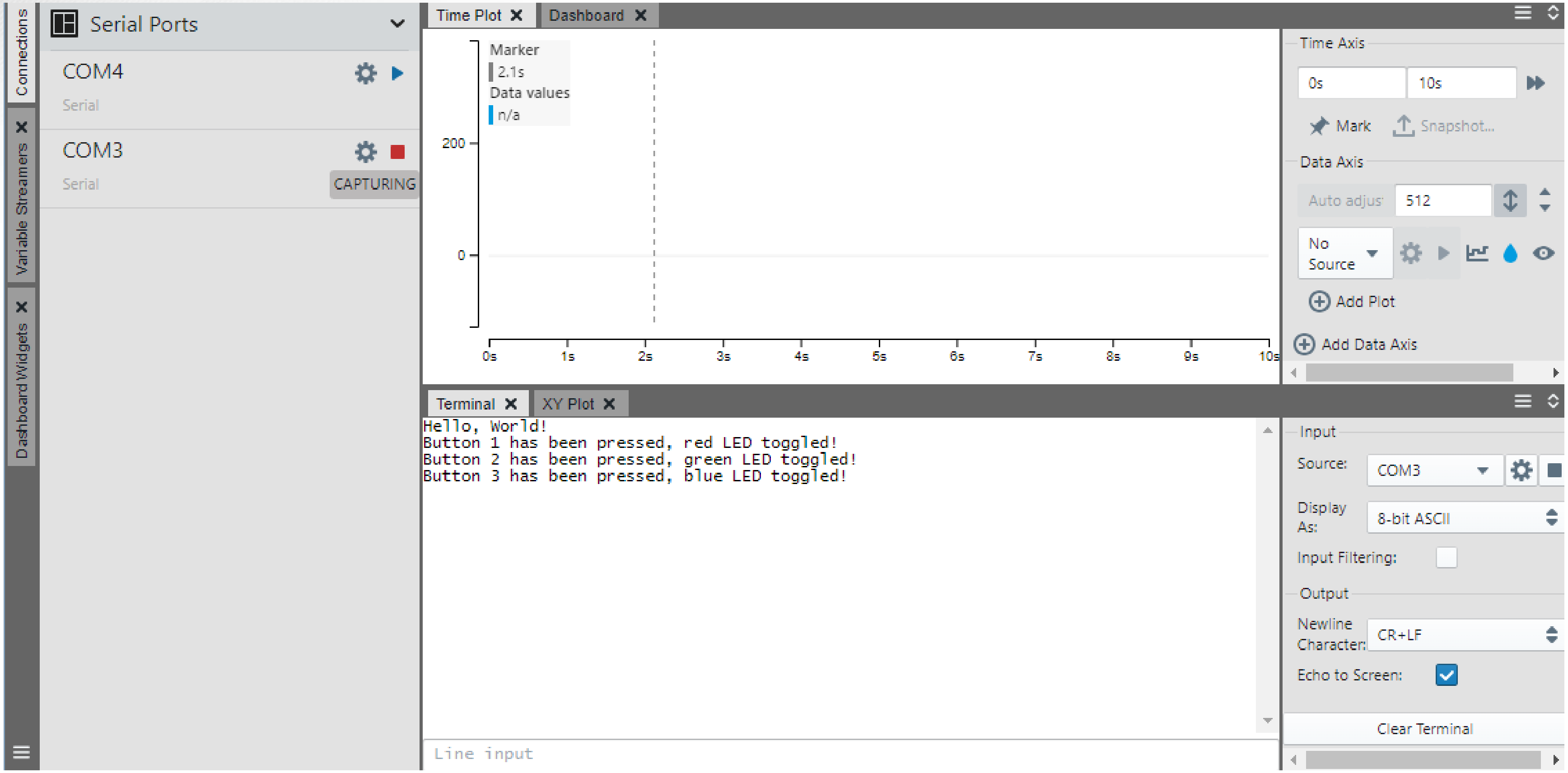
Speed: 0 RPM

Screenshot of Current Firmware



```c
int main(void)
{
    SYSTEM_Initialize();   // Initialize the system (clock, peripherals, etc.)

    printf("Hello, World!\r\n"); // Send a primary test string over UART1

    while (1)
    {
        /*
         *  Button usage
         */
        if (BUTTON1_GetValue() == 0) { // in every loop, check for a button press
            rLED_Toggle(); // if button1, then toggle red
            printf("Button 1 has been pressed, red LED toggled!\n\r");
        }
        if (BUTTON2_GetValue() == 0) {
            gLED_Toggle(); // if button2, then toggle green
            printf("Button 2 has been pressed, green LED toggled!\n\r");
        }
        if (BUTTON3_GetValue() == 0) {
            bLED_Toggle(); // if button3, then toggle blue
            printf("Button 3 has been pressed, blue LED toggled!\n\r");
        }


        /*
         *  Potentiometer Usage (will be used as the main motor speed controller)
         */
        //uint16_t potValue = ***ADC1_GetConversion***(POT_GetValue());

        // Map the ADC value (0-1023 for 10-bit ADC) to PWM duty cycle (0-100%)
        //uint16_t dutyCycle = (potValue * 100) / 1023;

        //PWM1_LoadDutyValue(dutyCycle);  // Adjust PWM duty cycle

        // Tachometer output reading

        // Display current Duty Cycle and Motor Speed in print functions

        __delay_ms(100); // delay 100ms before looping
    }
}
```

Example Output of Firmware



COM4
Serial

COM3
Serial
CAPTURING

Time Axis: 0s – 10s

Data Axis: 512

```
Hello, World!
Button 1 has been pressed, red LED toggled!
Button 2 has been pressed, green LED toggled!
Button 3 has been pressed, blue LED toggled!
```

Input

Source: COM3

Display As: 8-bit ASCII

Input Filtering: □

Output

Newline Character: CR+LF

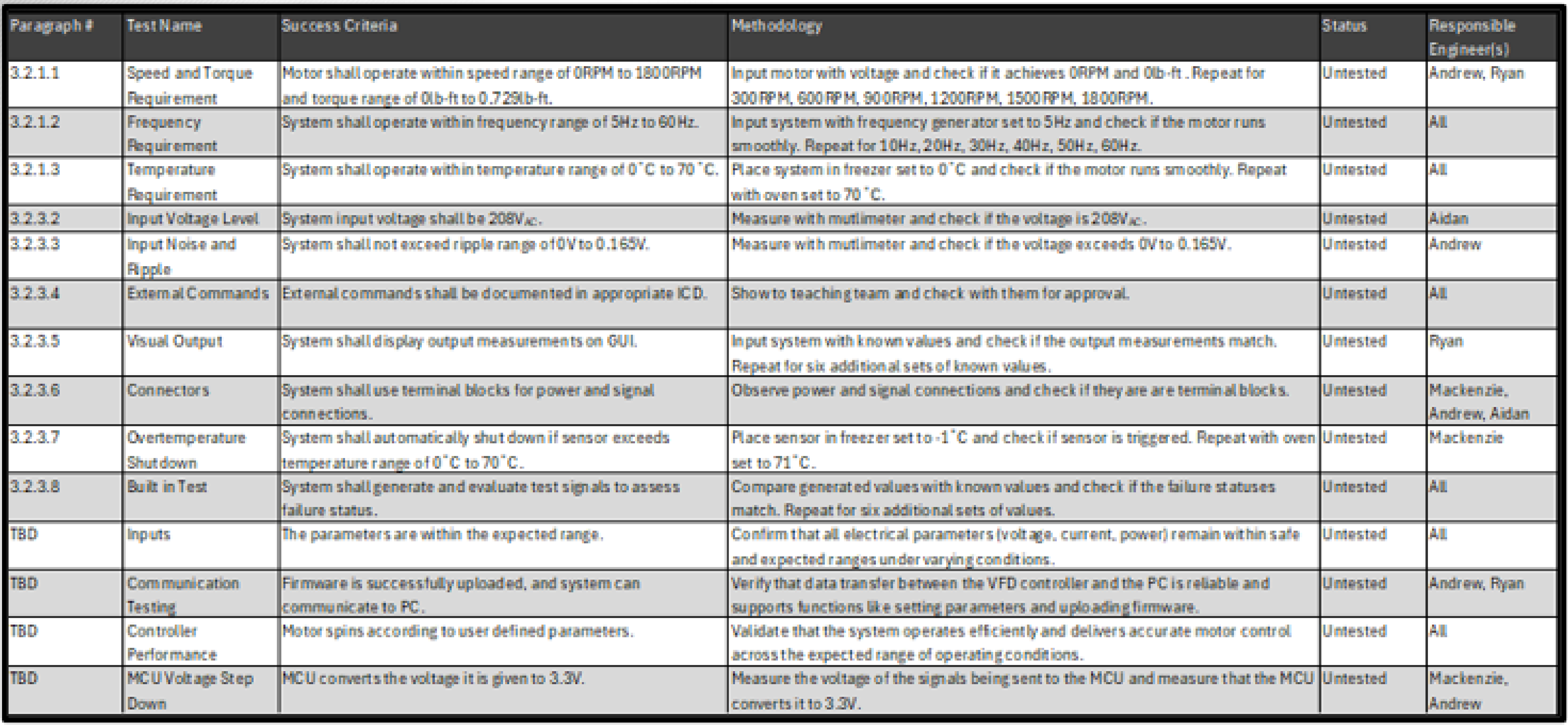
Echo to Screen: ☑

Exexcution Plan 8/20/2024-12/5/2024

	8/20	8/27	9/3	9/10	9/17	9/24	10/1	10/8	10/15	10/22	10/29	11/5	11/12	11/19	11/26	12/3	Date
CONOPS Report	🟩	🟩	🟩	🟩													9/15
FSR, ICD, Milestones, & Validation Plan				🟩	🟩	🟩											9/26
Firmware: GUI Development/Testing				🟩	🟩	🟩											9/24
Project Introduction Presentation						🟩											9/30
Optoelectronics: Subsystem Introduction Project				🟩	🟩	🟩	🟩										10/7
Microcontroller: Subsystem Introduction Project				🟩	🟩	🟩	🟩										10/7
Parts Order 1								🟩									10/8
Optoelectronics: Schematic Layout					🟩	🟩	🟩	🟩									10/14
Power: Subsystem Introduction Project				🟩	🟩	🟩	🟩	🟩									10/14
Firmware: Subsystem Introduction Project				🟩	🟩	🟩	🟩	🟩	🟩								10/15
Firmware: Develop Outline								🟩	🟩								10/15
Microcontroller: Schematic Layout					🟩	🟩	🟩	🟩	🟩								10/19
Power: DC Link Schematic Layout					🟩	🟩	🟩	🟩	🟩								10/19
Power: Rectifier Schematic Layout					🟩	🟩	🟩	🟩	🟩								10/20
Power: Power Control Schematic Layout					🟩	🟩	🟩	🟩	🟩								10/21
Project Update Presentation									🟩								10/21
Power: MCU Power Supply Schematic Layout									🟥								10/22
Parts Order 2									🟨	▨							10/22
Fimware: Add Debug Print Statements to Outline									🟨	▨							10/22
Firmware: Write to Demo Each Component Needed									🟨	▨							10/22
Optoelectronics: PCB Layout								🟨	🟨	▨							10/28
Microcontroller: PCB Layout									🟨	▨							10/28
Power: PCB Layout									🟨	▨							10/28
Optoelectronics: Order PCB										▨							10/28
Microcontroller: Order PCB										▨							10/28
Power: Order PCB										▨							10/28
Firmware: Speed Control Logic Implementation										▨	▨						10/29
Firmware: Validation/Debugging w/ Dev Board										▨	▨	▨					11/5
Optoelectronics: PCB Assembly											▨	▨	▨				11/18
Microcontroller: PCB Assembly											▨	▨	▨				11/18
Power: PCB Assembly											▨	▨	▨				11/18
Final Presentation												▨	▨				11/18
Optoelectronics: Validation/Debugging													▨	▨	▨		11/26
Microcontroller: Validation/Debugging													▨	▨	▨		11/26
Power: Validation/Debugging													▨	▨	▨		11/26
Firmware: Validation/Debugging													▨	▨	▨		11/26
Project Subsystem Demonstration															▨		11/26
Final Report															▨	▨	12/5

▨	Not Started
🟨	In Progress
🟩	Completed
🟥	Behind Schedule

Validation plan



Paragraph #	Test Name	Success Criteria	Methodology	Status	Responsible Engineer(s)
3.2.1.1	Speed and Torque Requirement	Motor shall operate within speed range of 0RPM to 1800RPM and torque range of 0lb-ft to 0.729lb-ft.	Input motor with voltage and check if it achieves 0RPM and 0lb-ft . Repeat for 300RPM, 600RPM, 900RPM, 1200RPM, 1500RPM, 1800RPM.	Untested	Andrew, Ryan
3.2.1.2	Frequency Requirement	System shall operate within frequency range of 5Hz to 60Hz.	Input system with frequency generator set to 5Hz and check if the motor runs smoothly. Repeat for 10Hz, 20Hz, 30Hz, 40Hz, 50Hz, 60Hz.	Untested	All
3.2.1.3	Temperature Requirement	System shall operate within temperature range of 0˚C to 70˚C.	Place system in freezer set to 0˚C and check if the motor runs smoothly. Repeat with oven set to 70˚C.	Untested	All
3.2.3.2	Input Voltage Level	System input voltage shall be 208V_{AC}.	Measure with mutlimeter and check if the voltage is 208V_{AC}.	Untested	Aidan
3.2.3.3	Input Noise and Ripple	System shall not exceed ripple range of 0V to 0.165V.	Measure with mutlimeter and check if the voltage exceeds 0V to 0.165V.	Untested	Andrew
3.2.3.4	External Commands	External commands shall be documented in appropriate ICD.	Show to teaching team and check with them for approval.	Untested	All
3.2.3.5	Visual Output	System shall display output measurements on GUI.	Input system with known values and check if the output measurements match. Repeat for six additional sets of known values.	Untested	Ryan
3.2.3.6	Connectors	System shall use terminal blocks for power and signal connections.	Observe power and signal connections and check if they are are terminal blocks.	Untested	Mackenzie, Andrew, Aidan
3.2.3.7	Overtemperature Shutdown	System shall automatically shut down if sensor exceeds temperature range of 0˚C to 70˚C.	Place sensor in freezer set to -1˚C and check if sensor is triggered. Repeat with oven set to 71˚C.	Untested	Mackenzie
3.2.3.8	Built in Test	System shall generate and evaluate test signals to assess failure status.	Compare generated values with known values and check if the failure statuses match. Repeat for six additional sets of values.	Untested	All
TBD	Inputs	The parameters are within the expected range.	Confirm that all electrical parameters (voltage, current, power) remain within safe and expected ranges under varying conditions.	Untested	All
TBD	Communication Testing	Firmware is successfully uploaded, and system can communicate to PC.	Verify that data transfer between the VFD controller and the PC is reliable and supports functions like setting parameters and uploading firmware.	Untested	Andrew, Ryan
TBD	Controller Performance	Motor spins according to user defined parameters.	Validate that the system operates efficiently and delivers accurate motor control across the expected range of operating conditions.	Untested	All
TBD	MCU Voltage Step Down	MCU converts the voltage it is given to 3.3V.	Measure the voltage of the signals being sent to the MCU and measure that the MCU converts it to 3.3V.	Untested	Mackenzie, Andrew