Winter 3-1-2019

10_FDR_Written_Project Central Vehicle Controller for Hybrid Racecar

Elias D. Bello
Elias.D.Bello.TH@Dartmouth.edu

Trammell J. Saltzgaber
Trammell.J.Saltzgaber.TH@Dartmouth.edu

Leina C. McDermott
Madeleine.C.McDermott.19@Dartmouth.edu

Alex S. Newman
Alexander.S.Newman.19@Dartmouth.edu

Jennifer Jain
Jennifer.Jain.TH@Dartmouth.edu

Cook Engineering Design Center

Final Design Review

Submitted in partial fulfillment of the requirements for
ENGS 90: Engineering Design Methodology and Project Initiation

Central Vehicle Controller for Hybrid Racecar

March 1, 2019

Sponsored by
John Miramonti

Project Team 10
Elias Bello
Jennifer Jain
Leina McDermott
Alexander Newman
Trammell Saltzgaber

Faculty Adviser
Eric Hansen

1867 THAYER SCHOOL OF ENGINEERING AT DARTMOUTH

Executive Summary

The Dartmouth Formula Racing (DFR) team, the College's Formula Hybrid team, requires a Central Vehicle Controller (CVC) for its hybrid race car. The CVC is a central computer that primarily determines the vehicle's hybrid powertrain output and monitors its complex safety-critical systems. DFR's existing CVC lacks reusable hardware and does not support new feature development, preventing the team from advancing the sophistication and dynamic performance of the vehicle.

The goal of this project is to provide DFR with a new CVC that can be used for several years into the future. The new CVC will have well-documented, reusable hardware and updatable firmware. It will also support advanced feature development by logging vehicle data and having customizable hardware and firmware configurations, including infrastructure for specific features including throttle-by-wire, regenerative braking, electronic shifting, and traction control. The project deliverables are a robust Printed Circuit Board (PCB), a mechanical PCB enclosure with automotive-grade connectors that integrate with DFR's low-voltage system, a well-documented firmware package, and comprehensive design documentation on the full system for DFR.

In order to tackle the challenge of developing such a complex controller, the team pursued a parallel-path development strategy that allowed both the firmware and the hardware to be developed fairly independently of one another. This strategy helped ensure that bottlenecks on one path did not cross over to the other path and propagate through our timeline. However, since firmware and hardware were developed by two different teams, effective and coordinated project management and planning was essential to our team's success. In order to ensure that the firmware being written would be able to run on the team's custom PCB, hardware components were selected for the PCB design that could be found on prototyping development boards. This allowed the firmware team to create a prototype of the hardware that would be on the PCB months before the board had arrived. In doing so, the firmware team was able to test code, validate hardware components, and develop the unit tests that would be used to test the team's custom PCB.

Once the PCB arrived, the system integration process began. The PCB was put through extensive hardware testing in order to ensure that the components and circuitry met the hardware design specifications. Once the board could be powered, the MCU was flashed with the firmware unit tests that had been developed on the hardware prototype. These unit tests revealed smaller hardware bugs that the team was able to resolve to ensure consistent behavior between the hardware prototype and the final PCB. Once all unit tests had been run, the firmware capable of running the car was flashed to the PCB and tested using the hardware prototype to mimic the DFR car's CAN and I/O behavior. Once this last test was conducted, the final PCB was mounted in its enclosure and connected to the DFR car's GLVS. The team was able to not only drive the car, but also to log 13 distinct signals to the PCB's onboard SD card.

The team was ultimately able to follow through on its promised deliverables of a new CVC and its accompanying documentation that will support DFR's central vehicle controller needs for years to come.

Table of Contents

1. Overview

1.1. Background

Since 2006, the Dartmouth Formula Racing Team (DFR) has designed and engineered hybrid and electric open-wheel race cars to compete in the annual Formula Hybrid (FH) competition. During the 2018 FH competition, DFR won the hybrid category in all dynamic events. An essential component of the DFR vehicle is the central vehicle controller (CVC), an electronic controller that coordinates the vehicle's subsystems. The controller implements the vehicle's hybrid control strategy, which determines the torque output of the electric motor relative to the internal combustion engine (ICE). The CVC also ensures the safe start-up and shut-down of the vehicle, and monitors safety-critical subsystems.

DFR's existing CVC implements a very simple control strategy that commands torque to the electric motor proportional to the position of the engine throttle. The CVC does not have control over the engine throttle, which is directly linked to the accelerator pedal. This strategy is sub-optimal in terms of both energy efficiency and acceleration performance. The DFR team would like to program the CVC to execute a more sophisticated control strategy in order to improve the car's performance in the FH dynamic events. However, the existing CVC's firmware is undocumented and lacks the infrastructure necessary to change or add controller features. Additionally, DFR has no method for recording the vehicle data necessary to develop and test controller features. DFR would also like to reuse the CVC on future DFR vehicles, but the existing CVC hardware is neither automotive grade nor reusable. The hardware is made up of off-the-shelf development boards that are solder-connected to the vehicle's low-voltage system, making the CVC very difficult to reuse.

1.2. Problem Statement

The DFR team lacks the tools necessary to develop and implement improved hybrid control strategies for their vehicle. The existing central vehicle controller lacks reusable hardware and its design does not support the features necessary to implement advanced control strategies. *Therefore, DFR has a need for a new central vehicle controller that is reusable and capable of supporting advanced control strategy development and implementation.*

1.3. Goal and Scope

The goal of this project is to provide DFR with a platform to develop and implement improved control strategies in 2019 and beyond. To meet this goal, we will provide the team with a reusable control unit that is easy to update and test, and capable of reliably logging vehicle data.

1.4. Deliverables

In order to address our problem statement, we have agreed to provide the following deliverables:

- A functional central vehicle controller PC board that can interface with the DFR car's electrical system
- Firmware that is loaded onto the controller board that can safely control the torque output of the electric motor, monitor sensors on the car, log vehicle data, and safely shut the car down in the case of a safety fault
- An automotive-grade enclosure to house CVC PC board and connectors
- Design files and documentation for our PCB, enclosure, and firmware
- Code documentation including a high-level architecture map and a CAN message table

1.5. Objectives and Requirements

The CVC's driving requirements include the functions required for the CVC to safely operate the vehicle, as well as the added data logging function. The CVC is also required to support the development of four additional controller features: throttle-by-wire, regenerative braking, traction control, and assisted or electronic shifting. These features were selected based on conversations with the DFR team and a review of state-of-the art hybrid control strategies among Formula Hybrid, Formula SAE, and Formula Electric Teams.

Type	Objective	Requirement	Specification	Test	Result
Functions	Control output of hybrid powertrain	Calculate torque output based on pedal position	Receive TPS CAN message from engine ECU at 100 Hz	Count number of TPS messages received over fixed time period	Passed—500 messages received in 5 seconds
		Command torque to electric motor	Send torque commands to motor controller over CAN at >= 20 Hz	Command torque to electric motor using controller	Drove vehicle using electric motor for 5 minutes
	Allow vehicle subsystems to function properly	Enable pre-charge of tractive system	Close second AIR <= 3 seconds after motor controller bus voltage reaches 90% of battery voltage	Log voltages and AIR open/close status during pre-charge routine at 50 Hz	AIR closes in 2.6 seconds (with built-in 2-second delay)
		Monitor subsystems for safety errors	Warning if engine temp >= 80C, Fault if engine temp >= 90C	Set lower temperature thresholds and ensure CVC enters appropriate state	Set thresholds to 20C, CVC entered fault/warning state in ambient temperature
			Fault if >= 10 consecutive missed CAN messages	Send CAN messages from development board, ensure vehicle enters fault state when sending stops	Sending messages at 100 hz, CVC immediately enters fault state when messages stop
	Safety	Safely shut down car when safety error detected	Set 12V safety output pin low <= 0.5 seconds after error	Simulate safety errors; record time between error and shut-down	Shut down in < 50 ms
	Record vehicle data	Log vehicle data to internal storage device	Continuously logs >=10 data points at >=20 hz	Log 10 data points at 20 Hz for 5 minutes	100% of 13 data points logged at 50 Hz for 5 minutes
Means	Support addition of new controller features	Support traction control, electronic throttle control, regenerative braking, electronic gear shifting	>= 4 timer capture input pins for wheel speed sensors	Prove 4 input pins function as timer capture inputs using unit test	Tested and passed on hardware prototype, not yet tested on PCB
			>= 6 Analog inputs (12V or 5V)	Prove 6 input pins function as analog inputs using unit test	Tested and passed on hardware prototype, not yet tested on PCB
			>= 2 PWM outputs	Prove 2 output pins function as PWM outputs using unit tests	Tested and passed on hardware prototype, not yet tested on PCB
	Automotive Grade Design	Functions in high and low temperatures	Functional at temperatures between -40C and 85 C	All hardware components rated for -40 to 125 C temperatures	Passed
Constraints	Support existing vehicle interfaces	Connects to vehicle CAN bus	CAN bus running at 500k Baud Rate	Connect CVC to vehicle CAN bus, ensure all messages are received	Able to read all relevant messages on vehicle CAN bus
		Supports 12V Inputs	>= 9 Discrete 12V Inputs	Test each input using power supply and oscilloscope	Passed
		Supports 12V Outputs	>= 8 Discrete 12V Outputs	Test each output using power supply and oscilloscope	Passed

Figure 1 Project Requirements and Design Specification

2. Design Methodology

2.1. Design Specification Development

We developed our design specification based on the specifications of the CVC's existing interfaces with the vehicle, as well as the additional interfaces required to support future control feature development. The CVC interfaces with the vehicle's Grounded Low-Voltage System (GLVS), electric motor controller, engine electronic control unit (ECU), battery management system, and dashboard. The CVC is powered by 12V from the GLVS, and communicates with most subsystems over the Controller Area Network (CAN) communication bus. The CVC also exchanges information with the vehicle through discrete 12V inputs and outputs.

To determine the specifications for additional controller feature support, we conducted a literature review of the sensor data required for throttle-by-wire, regenerative braking, traction control, and assisted/electronic shifting. The results of this trade study are in Appendix D1. The existing and anticipated CVC interfaces and functional requirements of the vehicle's subsystems informed the development of a detailed design specification. A detailed analysis of the required CVC interfaces is in Appendix E1.

2.2. Design Process

Our design process for this project followed three major parallel tracks: hardware, firmware, and system integration. After a conducting trade studies to design the overall architecture of the system, we designed and tested the hardware and firmware in parallel. The system integration track coordinated the hardware and firmware integration and full-system testing.

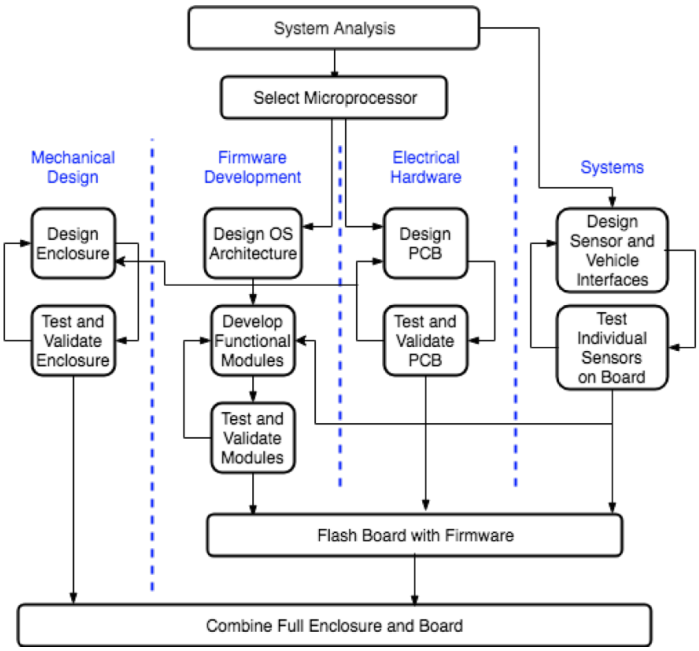


Figure 2 Design Process Flowchart

2.3. Architectural Design Decisions

An important aspect of our design approach was the decision to base our hardware design on open-source development boards. The reasoning for this was both to provide the hardware team with a starting point for their design, and to provide the firmware team with a method for testing code before the PCB design was complete. Before diverging into our parallel development tracks, we selected the development boards that would determine the overall system architecture.

2.3.1. Microcontroller Selection

The microcontroller (MCU), contains the processor on which the firmware runs, as well as the peripherals that allow it to communicate with other hardware components. The previous CVC used an STMicroelectronics MCU with an ARM M4 processor. As power consumption and physical size were not constraints, we only considered MCUs with ARM M7 processors, ARM's highest performance processor designed for embedded systems. We selected the STM32F7 based

its extensive peripheral support, the quality of ST's NUCLEO-F767ZI development board, and our familiarity with ST MCUs. This selection process is detailed in Appendix E2.

2.3.2. CAN Integration

Communicating with the rest of the car necessitated the inclusion of CAN circuitry on-board. Our selected MCU is CAN-compatible, so the board includes circuitry from the Waveshare CAN shield to provide full CAN compatibility on-board. This shield was used by the previous CVC as well, which ensured minimal difficulties during implementation.

2.3.3. PLC Board

To support safety features and other 12V inputs included in the original design of the car, our design includes the circuitry from the X-NUCLEO-PLC01A1 board. This NUCLEO shield contains level shifting circuitry that brings 12V digital inputs down to 3.3V so they can be communicated to the MCU via SPI. In addition, watchdog protection circuitry and temperature monitoring chips add hardware safety tools that complement the existing firmware safety features.

		Ease of Hardware	Justification	**Ease of Firmware**	Justification	**Max Frequency**	Value (MHz)	**~Data Bandwidth**	Value (MB/s)	**Total**
Weight		2		2		3		5		
Data Transfer Option	**SPI**	**5**	Needed hardware is provided by the Nucleo board	**5**	Firmware team has experience with SPI	**3**	27	**2**	3.218650818	**39**
	SDIO (1-bit)	**5**	Needed hardware is provided by the Nucleo board	**4**	Ample resources but firmware team doesn't have experience with SD transfer mode	**5**	50	**3**	5.960464478	**48**
	SDIO (4-bit)	**5**	Needed hardware is provided by the Nucleo board	**4**	Ample resources but firmware team doesn't have experience with SD transfer mode	**5**	50	**5**	5.960464478	**58**

Figure 3 Data Transfer Alternatives Matrix

2.3.4. Data Logging Design

In order to support data-logging functionality, our design required an internal data storage method, as well as a way for DFR members to download and review the data. After analyzing our options, we decided to write data to an onboard SD card via the SDIO protocol for its superior data transmission speed. To download this data, we considered three options-- USB On-the-Go, Ethernet, and Bluetooth. The hardware for each option was included in the first board revision, to give the firmware team more time to test each before selecting which to use for the final design. Finally, we chose to include an accelerometer onboard for additional convenience and future vehicle development. All decision matrices used in this hardware selection (as well as the selection of the MCU) can be found in Appendix E.

3. Hardware Development

3.1. Hardware Design

3.1.1. Power Architecture Design

The CVC interfaces with the car through the GLVS (ground low-voltage system), which is powered by a 12V car battery. To ensure that the components on our board were properly powered, we conducted a power budget, determining what the max current draw from each voltage source would be, and selected power supplies capable of supplying this current. This power budget can be seen in the table below:

Table 1 Power Budget

Part Number	Description	Current Draw (mA)	Net Name
LSM9DS1	Accelerometer	0.6	3.3V
USBLC6-4SC6	USB ESD Protection IC	0.15	3.3V
KMS-1102NL	Ethernet Transformer	0	3.3V
LAN8742A-CZ-TR	Ethernet Transceiver	281.6	3.3V
MCU_LQFP144	MCU	420	3.3V
CLT01-38SQ7-TR	PLC Digital Current Limiter	47	12V
VNI8200XP	PLC Solid State Relay	10	5V
STMPS2151STR	USB Power Distribution Switch	500	5V
SD_CARD_SOCKET	SD Card Socket	500	3.3V
32KB NRF51822	Bluetooth Module	11	3.3V
74LVC2G34DBVR	Bluetooth Level Shifter	50	3.3V
SN65HVD233HD	CAN Transciever	512 (total for 2)	CAN Supply
Si8662AB-B-IS1	Digital isolator IC	26.7	3.3V

Totals by Voltage Net

Net	Current Draw (mA)
3.3V	1337.05
5V	510
CAN Supply	512
12V	1847.05

Given these power totals, we selected the CC6-1205SF-E DC-DC converter to supply our 5V rail. Its 5A current rating is more than sufficient for the 500mA max current draw we calculated. For the 3V power rail, we supplied a 12V to 3.3V Buck converter rated for 2A output current, well above the 1.3A max draw. Given that the CAN power supplies can draw up to 500mA in a fault scenario, we included separate linear regulator for our CAN transceivers: the TI LM317DCY. These are rated for 1.5A and thus are sufficiently rated for the CAN chips they will be supplying.

3.1.2. Protection Circuitry Design

Since the MCU also takes in multiple external sensor inputs directly, we developed protection circuitry that would protect the MCU in the case of an overvoltage scenario at these pins. This circuitry is centered around a positive temperature coefficient (PTC) resistor and a TransZorb diode. The PTC resistor has a resistance value that scales with temperature. In the case of an overvoltage scenario, the TransZorb diode handles the current by dissipating a lot of current. As the current flows through the PTC, it heats up and becomes a larger resistor, limiting the amount of current flowing until it has reached a safe value for the MCU. This circuitry can be seen in the schematic in the datasheet in Appendix K.

3.1.3. Schematic Design

Having selected the components that we needed to include in our design, we planned how we could best allocate the MCU pin resources and our power supply circuitry to ensure reliable function of the system. By utilizing alternate functions of the MCU pins, we devised a configuration that allowed for each of the peripherals we selected to be connected to the MCU at the same time while also providing analog, timer, and digital GPIOs to the user. With the connections fleshed out, we constructed our full schematic in Altium while making edits to pre-existing development boards to remove redundant and unnecessary components from our design. To minimize the number of errors in the schematic, the team had several small, informal design reviews with our project sponsor John Miramonti before conducting a final, formal design review with John and three of his coworkers at SignalQuest in Lebanon, NH.

3.1.4. Layout Design

Using measurements taken from the car, we determined the maximum size of the board—6 inches by 4.6 inches—leaving sufficient room for a water-tight enclosure in which it would eventually be placed. We placed each component on the board and routed the connections between them, utilizing our current budget and an online trace width calculator to ensure that traces were properly sized for the amount of current they would be carrying in a worst-case scenario. We also included mounting holes that grounded the board to the car's chassis and ensured that it could be placed in the enclosure. Just as with the schematic, the team conducted several design reviews of the layout—both internally and with outside advisors John Miramonti, Prof. Jason Stauth, and Thayer '14 Eric Din—to minimize the amount of errors before sending the board out.

3.2. Hardware Validation

We developed a test plan, shown in Appendix F, outlining the steps that we would take to ensure the board's functionality. The test plan was broken down into four stages: pre-board testing, pre-power testing, post-power testing, and post-flash testing. By testing at each of these separate stages, we could begin the next planned phase of testing with a minimal chance of doing damage to the board.

3.2.1. Pre-Board Arrival Testing

Pre-board testing primarily consisted of evaluating the circuitry that regulates voltage levels around important components. This testing was done by soldering relevant ICs to a solder board and powering the ICs with a power supply, then measuring the outputs. The testing procedures that make up the pre-board testing are shown in Appendix F1.

3.2.2. Pre-Power Testing

Pre-power testing comprised preparing the board for power-up by soldering any unmounted components to the board and subsequently checking connections and impedance values of our circuits against our design. Once these checks were complete, we could safely power up our board.

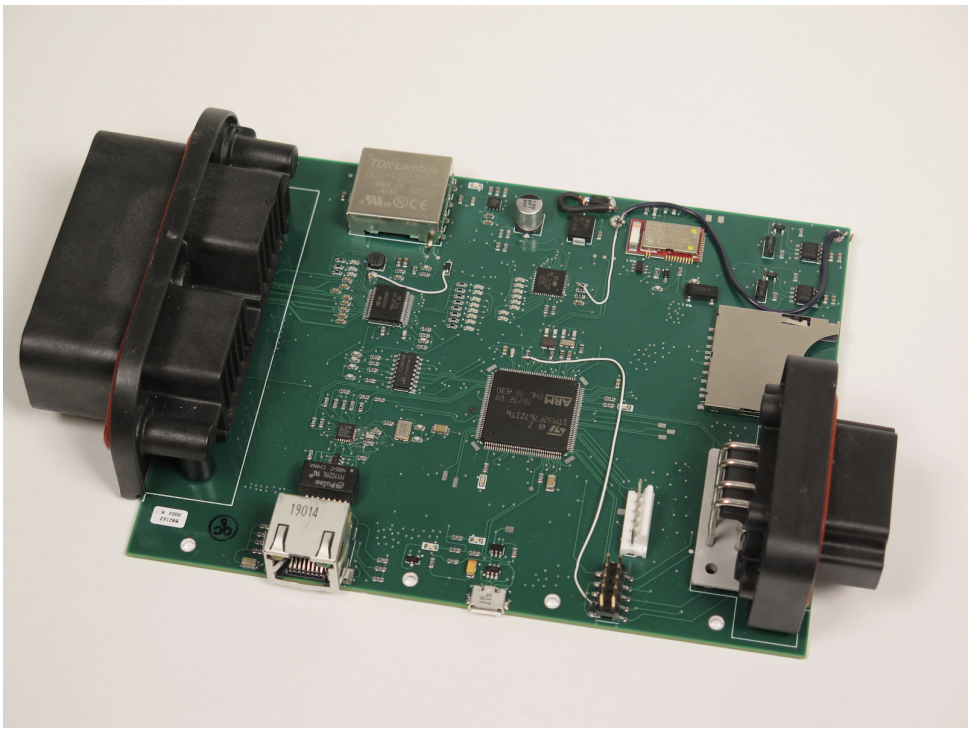


Figure 4 CVC PCB with Corrections

3.2.3. Post-Power Testing

Post power-up testing involved measuring the voltages on each IC, ensuring the power LEDs were on, and monitoring the current draw. The tests conducted at this stage in the hardware testing are listed in Appendix F3.

3.2.4. Firmware Testing

After powering up the board, firmware unit tests were used to validate the PCB hardware. Using feedback from the firmware testing, the team tackled hardware bugs to get the board functioning to specification. An issue with resetting the MCU was fixed by shorting the corresponding SWD connector pin to a different pin on the MCU. A problem with the initialization of the external oscillator was overcome by preparing a second board, on which it was able to initialize. The PLC circuitry was fixed after extensive testing and review of the schematic by the ultimate realization that the level-shifting IC before the SPI communication chip was a slightly different part number - after swapping it out, it performed correctly. In the process of discovering this issue, the hardware team measured voltages and grounds on all of the pins on the MCU and all of the outputs of the VNI, which is the chip used for SPI.

3.3. Mechanical Enclosure Design

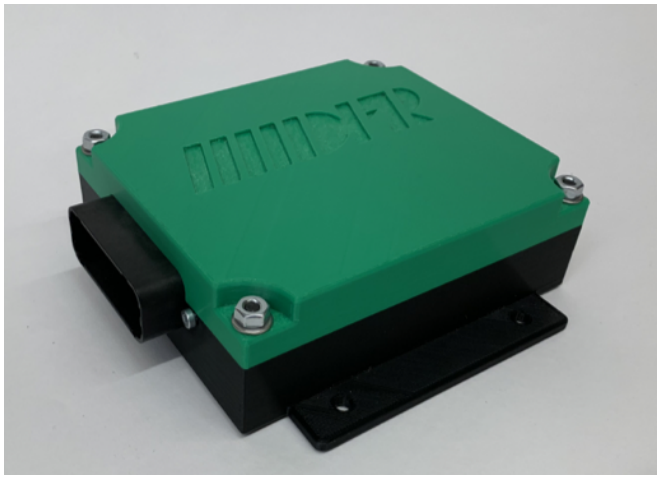


Figure 5 Mechanical PCB Enclosure

In order to meet our requirement of providing an automotive-grade CVC that interfaced with the current and all future DFR cars, the team had to design an enclosure that could safely house the CVC PCB. This enclosure had to be designed to fit within the envelope specified by DFR (CAD shown in Appendix F6), and protect the PCB from vibration, dust and rain. To ensure water-tight seals, the box was designed with a lip in which a rubber gasket may be installed, and was designed to seal to the water-proof TE Deutsch connectors that provide the interface between the CVC and the DFR car's electrical system. The enclosure was designed specifically to fit the unique dimensions of the PCB and provides embedded standoffs that tailor to the mounting holes on the PCB, avoiding large components and traces. The box can be mounted to the car using four ¼-20 rubber stud standoffs arranged according to the dimensions specified in the enclosure drawings (CAD shown in Appendices F7 and F8). In order to prototype quickly, the enclosure was 3D printed with ABS plastic. However, future versions could be injection-molded from ABS instead, especially if large quantities were to be produced.

4. Firmware Development

4.1. Functional Overview

The CVC firmware is the C program that runs on the STM32F7 microcontroller (MCU), which interfaces with the other hardware on the board to provide the desired functionality of the CVC (see Figure 1). The program functions as a state machine that executes the vehicle's start-up sequence and fault handling. The MCU's SPI and CAN hardware drivers allow the program to send and receive data to/from the vehicle. and logs data to an SD card through the SDIO peripheral. The program uses FreeRTOS, an open-source real-time operating system kernel, to schedule tasks and meet its strict timing requirements.

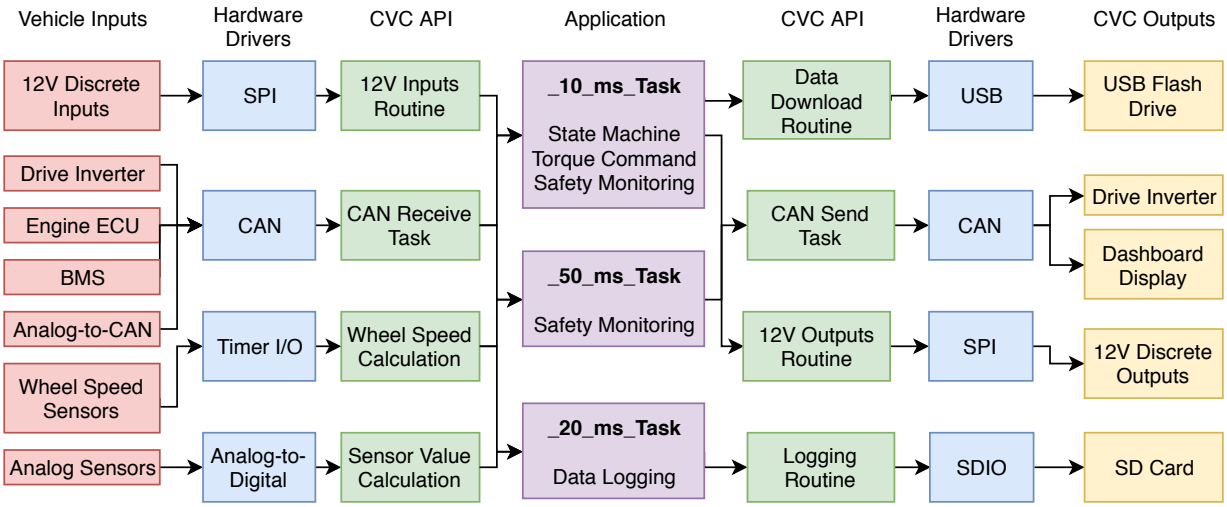


Figure 6 High-level Firmware Block Diagram

4.2. Architecture

The firmware functions on several levels to operate the CVC. At the lowest level, hardware drivers send and receive data to/from the vehicle through the MCU's peripherals (SPI, CAN, SDIO, I2C, ADC, Timer I/Os). The next level is the Middleware layer, which implements the FreeRTOS operating system, FatFs file system, and USB on-the-go protocol. The third layer is the CVC API, which implements CVC-specific functions and provides a layer of abstraction between the driver and middleware layers and the application layer. The top-level application layer implements user-specific code that may be modified to add or change features. The application layer includes a configuration file that allows the user to easily configure existing CVC features.

4.3. Driver Layer

The CVC requires drivers for the MCU's SPI, CAN, SDIO, Analog-to-Digital Converter, and timer I/O peripherals.

Table 2 CVC Peripheral Descriptions

Peripheral	Purpose
SPI	Communication between MCU and 12V discrete I/Os
CAN	Communicate over CAN to send torque commands and monitor data from drive inverter, BMS, engine ECU, and vehicle sensors
SDIO	Write data to SD card for data logging
I2C	Communicate with on-board accelerometer
Analog-to-Digital converter	Receive analog inputs from vehicle sensors
Timer I/O	Accept pulse-train inputs from wheel-speed sensors; generate PWM output to control servo motor for engine throttle control

ST provides drivers for the STM32F7 MCU in the Hardware Abstraction Layer (HAL) and Low Layer (LL) APIs of the STM32 Cube library. The first step of our firmware development was to write unit tests for each driver and test them using development boards. Unit testing details can be found in Appendix G, Figure G4. These unit tests validated both the selected hardware components and the driver code, and provided a framework for driver implementation that we used to build the CVC API.

4.4. Middleware Layer

4.4.1. FreeRTOS Operating System

The CVC firmware uses FreeRTOS to implement a real-time operating system (RTOS). We selected our operating system for its ability to meet our system's **strict timing requirements**, its extensive support documentation, and its scalability. Using FreeRTOS, the program is structured as multiple concurrent tasks. Each task is a separate thread of execution, and the scheduling of tasks is managed by FreeRTOS. Tasks may be set to execute at specified frequencies, or may be triggered by events (such as a message being added to a queue). The FreeRTOS API makes the creation and control of tasks simple, allowing for the easy addition of new features to the controller. However, the decision to use FreeRTOS increased the difficulty of our program design, as additional precautions had to be taken in order to ensure thread safety. In the early stages of firmware development, we created a demo program using FreeRTOS to run a simple LED-blinking task. This program served as the basis for our FreeRTOS implementation, which currently includes the following tasks:

Table 3 CVC Tasks

Task	Function	Execution
Init_Task	Initialize peripherals and logging; create all other tasks	One-time execution at start of program
CAN_Rx_Task	Parse incoming CAN messages and add to CAN inputs table	Executes when message is added to Rx queue
CAN_Tx_Task	Send outgoing CAN messages	Executes when message is added to Tx queue
PLC_Routine_Task	Initiates SPI transmission to send/receive discrete 12V I/Os	Executes at 100 Hz
_20_ms_Task	Logs data to file on SD card	Executes at 50 Hz
_10_ms_Task	High-level state machine, torque commands, and high-speed safety monitoring	Executes at 100 Hz
_50_ms_Task	Low-speed safety monitoring	Executes at 20 Hz

4.4.2. FatFs

FatFs is a platform-independent FAT File System module that allows our program to create and modify files. The program uses the FatFs API to log vehicle data to .csv files on the SD card, and to copy these files to a USB drive when the user requests a data download. To implement this, we created two intermediate diskio drivers linking the FatFs module to the ST-provided SDIO driver and USB Host Mass Storage Class (Appendix G, Figure G2 for FatFs operation diagram).

4.4.3. USB Host Mass Storage Class

Our program uses the USB Host Mass Storage Class (MSC), provided in the STM32 Cube Library, to transfer log files from the on-board SD card to an external USB flash drive. When the user connects a USB drive to the CVC using a micro-USB adapter, the CVC automatically copies all files from the SD card to the USB drive. We chose USB as our data download method over Ethernet because of its compatibility with FatFs, and the extensive support documentation ST provides for USB compared to Ethernet. Additionally, USB draws less power and requires less hardware than Ethernet, simplifying the PCB design.

4.5. CVC API Layer

The CVC API provides a level of abstraction on top of the hardware drivers and middlewares that gives the application layer access to data in the form of abstract variables. In other words, the CVC API uses the hardware drivers and middlewares to implement CVC-specific functions.

These functions are currently organized into four modules: the 12V I/O Interface (cvc_spi), CAN interface (cvc_can), Logging interface (cvc_logging), and FreeRTOS tasks (cvc_tasks).

4.5.1. 12V I/O Module

The 12V I/O module contains the necessary functions to read 12V inputs and set 12V outputs by communicating with the CLT and VNI chips over SPI. The module includes the PLC_Routine_Task, which initiates a SPI transmission at a frequency of 100 Hz, populating the SPI Inputs Vector and transmitting the information in the SPI Outputs Vector (Appendix G, Figure G6).

4.5.2. CAN Interface Module

The CAN interface module contains all the necessary functions to send and receive CAN messages to and from the vehicle's subsystems. The module implements two tasks, CAN_Rx_Task and CAN_Tx_Task, which execute when a CAN message is placed in the CAN Rx and Tx queues respectively. Messages are added to the Rx queue by the CAN driver as they appear on the CAN bus. Messages are added to the Tx queue by application code through the CAN_Send API function. The use of queues prevents messages from being lost if multiple occur in quick succession. The CAN Rx task parses relevant incoming CAN messages and places the parsed data in the CAN_Inputs vector. The module relies on a dictionary of CAN messages to recognize and parse relevant CAN information (Appendix G, Figure G3).

4.5.3. Logging Module

The Logging interface module provides the functions necessary for a user to log desired data to the onboard SD card. The module contains a logging function that writes instantaneous data values to a .csv file. The user may configure which values are logged, and may call the log function from a synchronous task of the desired frequency. The logging module uses FatFs to open and write to a .csv file on the SD card. The logging task also contains a function to copy files from one disk to another, allowing it to copy files from the SD card to a USB flash drive when a download is requested.

4.5.4. Tasks Module

The tasks module manages the FreeRTOS tasks that execute most of the program's functions. These tasks are described in Table 3. The module provides a task structure that allows the user to easily add new tasks to the program.

4.6. Application Layer

The application layer of the firmware is the portion of code that handles all of the major, high-level functions of the car. The synchronous module provides tasks that run at specified frequencies for the user to add application functions to. Currently, the module includes a fast task that runs at 100 Hz, a medium task that runs at 50 Hz, and a slow task that runs at 20 Hz. We have implemented modules with state machine, torque command, and safety monitoring functions that run within these synchronous tasks.

4.6.1. State Machine Module

The state machine module contains the main state machine that governs the car's operation. It runs within the 10 ms (100 Hz) task. The function of the state machine is described in Figure 7 and Table 4.

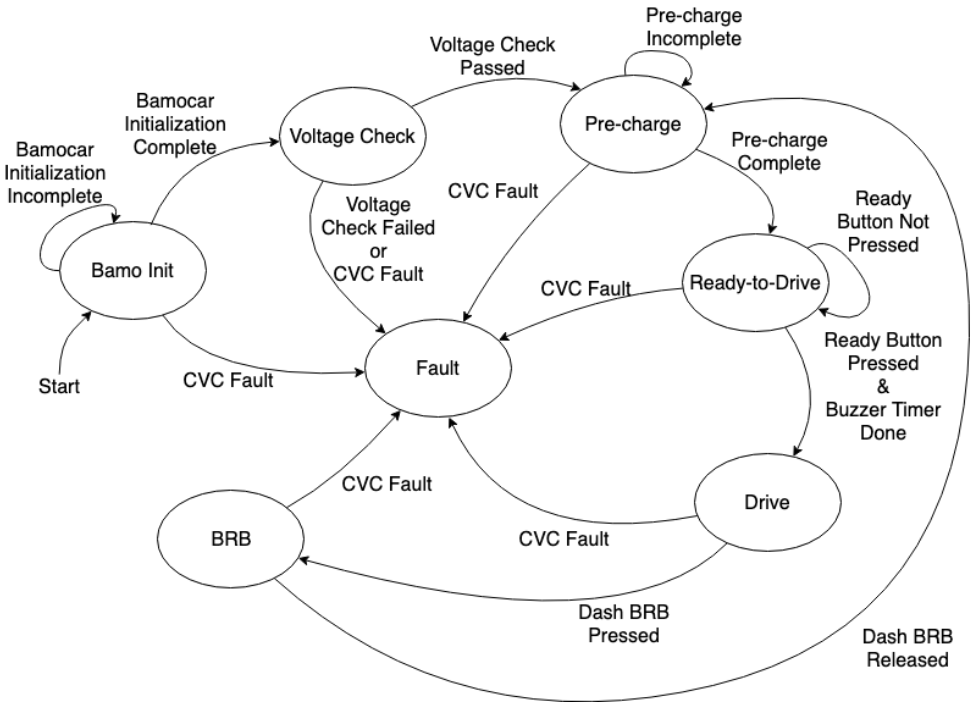


Figure 7 High-level state machine

Table 4 High-level state machine state descriptions

State	Description	Output
Bamo Init	Sends reset and initialization messages to Bamocar	safety = 0, ready to drive = 0, motor enable = 0
Voltage Check	Checks that bus voltage and battery voltage are within expected ranges	safety = 0, ready to drive = 0, motor enable = 0
Pre-charge	Checks that bus voltage is at least 90% of battery voltage, then closes second AIR	safety = 1, ready to drive = 0, motor enable = 0
Ready-to-Drive	Waits for driver to press ready to drive button	safety = 1, ready to drive = 1, motor enable = 0
Drive	Commands torque to electric motor based on engine throttle position	safety = 1, ready to drive = 1, motor enable = 1
BRB	Waits for driver to release BRB (allowing the state to return to pre-charge)	safety = 1, ready to drive = 0, motor enable = 0
Fault	Sets all outputs low	safety = 0, ready to drive = 0, motor enable = 0

4.6.2. Torque Command Module

The torque command module is responsible for calculating and commanding torque outputs to the electric motor. In the future, this module will also command torque to the engine. For now, the torque command is determined by the throttle position of the engine.

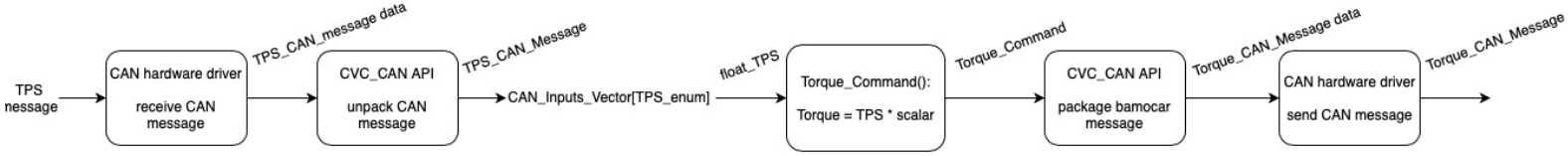


Figure 8 Torque Command Data Flow

This throttle position value is read from the engine ECU over CAN. The value is then used to scale the torque command to the electric motor between zero and the maximum torque of the motor.

4.6.3. Safety Monitoring

Safety monitoring is accomplished both actively, by checking to ensure that certain input vector values fall within a user-specified range, and passively, by calling an error handler when unexpected behavior is observed. Each component, from the car's motor controller to its engine has a unique safety monitoring function that resides in its own module. The error handler on the other hand, is called due to unexpected internal behavior of the CVC and is called with a certain "fault status" and "error code" (Appendix G, Figure G5).

4.7. Firmware Testing and Validation

Firmware testing was conducted using the development boards for the MCU, 12V I/Os, CAN, and SD Card selected at the beginning of the design process. As a testing tool and risk mitigation strategy, we assembled a hardware prototype from these boards that has most of the functionality of the PCB (Appendix G, Figure G6). A Git repository was used for version control and file sharing between the firmware team members. Our modular firmware design allowed us to independently test each module using the hardware prototype before integrating it into the firmware repository.

We began by unit testing hardware-specific firmware for the SPI, SD, CAN, Analog-to-Digital, I2C, and Timer I/O peripherals over the winter break. We then brought up our implementation of the FreeRTOS operating system and FatFs file system. Next, we created the CVC API, the code that implements CVC-specific functions in FreeRTOS using the hardware drivers and middlewares. Finally, we wrote the application code to safely start-up the vehicle, command torque, monitor subsystems, and handle faults. After thoroughly testing each code module using development boards, we conducted bench tests of our start-up and messaging code on the DFR vehicle using the hardware prototype. This integrated testing and development process allowed us to identify problematic code early-on, preventing error propagation throughout the program. A detailed explanation of firmware testing can be found in Appendix G, Figure G4. The unit tests we developed were also valuable tools when validating the PCB hardware.

5. System Integration and Final Deliverables

5.1. Test Drive

The culmination of the team's hardware and firmware validation were two driving tests. The CVC was mounted on top of the box it will replace and wired into the car's GLVS (see Figure 9). The first test drive consisted of one lap of the circle behind the Thayer School of engineering using only the electric motor for propulsion. The test continued with two laps of the circle using only the gas engine. While it was only about 5 minutes of driving, this first test allowed us to validate the CVC's data-logging, safety functionality, torque commands, and robustness to vibration and driving conditions.

The second driving test of the car consisted of a test of the car's full-hybrid powertrain. Both the engine and motor were used for propulsion during the four laps of driving. During this test, the team logged the same 13 data points that were logged in the first test. Using a MATLAB script, the team analyzed the data and created some basic plots (see Appendix H2 for MATLAB code). This script can be used to analyze data from future DFR tests in order to analyze everything from

thermal properties of the powertrain and cooling system, to braking and acceleration performance.

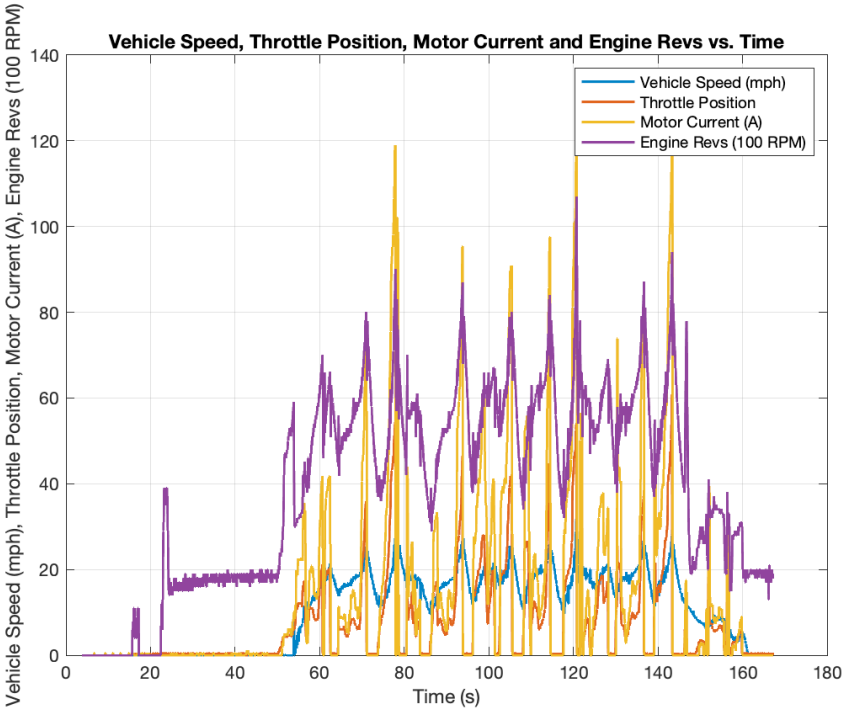


Figure 9 Test Driving Data

5.2. Evaluation of Deliverables

To meet the needs of DFR, the team promised to deliver a functional central vehicle controller on a PC board capable of interfacing with the DFR car's electrical system. Furthermore, the team committed to providing firmware that could monitor sensors, log vehicle data, and safely control the torque output of the DFR car's electric motor. Last, the team committed to housing the hardware in an automotive-grade enclosure and provide a full-suite of documentation for both the hardware and firmware.

The team was ultimately successful in providing DFR with every specified deliverable. The team delivered a functioning controller PC board that has been used to run the car in electric, gas, and hybrid modes. The firmware flashed to the board safely operates the vehicle and accurately logs vehicle data. Finally, the design files and documentation for the CVC has been compiled into a datasheet that will give DFR easy access to any information the team might need. The Altium design files and well as the Github repository that holds the firmware will also be made available to DFR.

6. Economic Analysis

6.1. Project Costs

The total project cost was primarily driven by the PCB order, which cost the team $5,755.33 for 3 assembled and 6 unassembled boards, largely due to the expedited turnaround time the group required in order to complete the project on schedule. The breakdown of this cost is shown in Appendix I1. Firmware costs were primarily driven by money spent on development boards for

prototyping. These costs are shown in Appendix I2. The PCB enclosure was outsourced to 3D HUBS, to ensure a professional-quality product, for a cost of $231.64.

6.2. Market Factors

If the CVC were produced in bulk, the costs would be substantially lower per unit. Advanced Assembly quoted a cost of $570 per board for 100 or more units, and 3D HUBS would price the enclosures at $17.28 per unit for 1,000 or more units. Therefore, when mass-produced, our cost would be $587 per unit. See Appendix I4 for breakdowns of these costs.

The CVC could be marketed to collegiate hybrid and electric teams, automotive aftermarket hobbyists, and small electric or hybrid conversion shops. The CVC offers a unique value proposition for this market because of its versatile functions (such as logging data and controlling motors) and adaptable interfaces. The MoTeC CDL3 is a similar product that offers less data storage capability but doesn't offer the option to implement control strategies which costs around $1,700.

The approximately 50 collegiate formula teams around the country have a budget of around $2.5 million (based on DFR's budget of $50,000). Our other target segment, the automotive aftermarket, is estimated at a size of $2.49 billion. Assuming that 2% of this total is put towards controllers and loggers, a number based on DFR's past spending, the market size for the CVC is around $50 million. If the team priced the CVC at $1,700 and were to capture 1% of this market the first year and an additional 1% every year thereafter for the first five years, there would be a profit of $327,422 in the first year and a total profit of $4.9 million after five years. These calculations are detailed in Appendix I7.

7. Recommendations for Future Work

Though the team is quite happy with the product that we were able to deliver to DFR, we have several recommendations for future work on the CVC. The first recommendation concerns the temporary fixes that the team had to make in order to get the PCB to function properly. Because of the scope of this project and the tight timeline of Engs 90, the team was unable to order a second revision of the PCB for DFR. However, we have compiled a list of Engineering Change Orders (ECOs), included in Appendix J1, that can be used to permanently fix the few hardware issues we have found throughout our testing of the PCB. Furthermore, since the first PCB was designed with testing in mind, many traces and much of the spacing is not fully optimized. Thus, the next PCB ordered can also be optimized for packaging and overall efficiency by eliminating unnecessary jumpers and components. Our expectation is that a second revision of the CVC PCB can be ordered before the beginning of the Spring term, in order to allow ample time for testing and integration of the new hardware.

We also recommend that DFR make the most of the built-in data-logging and easy-to-use application layer of the CVC firmware to fully instrument the car with sensors (wheel speed, steering angle, etc.) to collect data and implement new features (such as throttle-by-wire and traction control). The CVC may already be capable of controlling the DFR car, but its greatest value to DFR should be the foundation it provides for building new, optimized control strategies.

Appendix A

Deliverables Agreement

We will provide the following deliverables to John Miramonti and the DFR team on March 1st, 2019:

- A functional central vehicle controller PC board that can interface with the DFR car's electrical system
- Firmware that is loaded onto the controller board that can safely control the torque output of the engine and electric motor, monitor all sensors on the car, log vehicle data to an internal storage drive, and safely shut the car down in the case of a safety fault
- An automotive-grade enclosure to house CVC PC board and connectors
- Design files and documentation for our PCB, enclosure, and firmware
- Code documentation including a high-level architecture map and a CAN message table

I hereby agree to the contract outlined above:

Name: *John Miramonti* Date 11/9/2018

Name: *Alexander Newman* Date 11/9/2018

Name: *Elias Bello* Date 11/9/2018

Name: *Jennifer Jain* Date 11/9/2018

Name: *Leina McDermott* Date 11/9/2018

Name: *Trammell Saltzgaber* Date 11/9/2018

Appendix B

Rule Number	Rule	Relevance
IC1.6.2 Accelerator Actuation - General	All systems that transmit the driver's control of the speed of the vehicle, commonly called "Accelerator systems", must be designed and constructed as "fail safe" systems, so that the failure of any one component, be it mechanical, electrical or electronic, will not result in an uncontrolled acceleration of the vehicle. This applies to both IC engines and to electric motors that power the vehicle. The Accelerator control may be actuated mechanically, electrically or electronically, i.e. electrical Accelerator control (ETC) or "drive-by-wire" is acceptable.	Throttle Control Implmentation
IC1.6.4 Electrical Accelerator Actuation	When electrical or electronic throttle actuation is used, the throttle actuation system must be of a fail-safe design to assure that any single failure in the mechanical or electrical components of the Accelerator actuation system will result in the engine returning to idle (IC engine) or having zero torque output (electric motor). Teams are strongly encouraged to use commercially available electrical Accelerator actuation systems. The methodology used to ensure fail-safe operation must be included as a required appendix to the Design Report.	
EV2.10.1 Precharge	The AIR contacts must be protected by a circuit that will pre-charge the intermediate circuit to at least 90% of the rated accumulator voltage before completing the intermediate circuit by closing the second AIR. The pre-charge circuit must be disabled if the shutdown circuit is deactivated; see EV7.1. I.e. the pre-charge circuit must not be able to pre-charge the system if the shutdown circuit is open.	Start-up Sequence
EV2.10.3	The pre-charge circuit must operate regardless of the sequence of operations used to energize the vehicle, including, for example, restarting after being automatically shut down by a safety circuit.	
EV5.5.4	Teams must be prepared to demonstrate spacings on team-built equipment. Information on this must be included in the ESF (EV13.1). (a) Teams must supply high resolution (min. 300 dpi at 1:1) digital photographs of team- designed boards showing: (i) All layers of unpopulated boards (inner layers or top/bottom layers that don't photograph well can be provided as copies of artwork files.) (ii) Both top and bottom of fully populated and soldered boards. If dimensional information is not obvious (i.e. 0.1 in x 0.1 in spacing) then a dimensional reference must be included in the photo. (b) Spare boards should be made available for inspection. Teams should also be prepared to remove boards for direct inspection if asked to do so during the technical inspection.	PCB Design
EV7.1.5	In the event of an AMS, IMD or Brake over-travel fault, it must not be possible for the driver to re-activate the tractive system from within the cockpit. This includes "cycling power" through the use of the cockpit shutdown button. Note: Resetting or re-activating the tractive system by operating controls which cannot be reached by the driver26 is considered to be working on the car.	Safety State Machine
EV7.6.4	The cockpit shutdown button must be driver resettable. I.e. if the driver disables the system by pressing the cockpit-mounted shutdown button, the driver must then be able to restore system operation by pulling the button back out. Note: There must still be one additional action by the driver after pulling the button back out to reactivate the motor controller per EV7.7.2.	
EV7.7.2	After enabling the shutdown circuit, at least one action, such as pressing a "start" button must be performed by the driver before the vehicle is "ready to drive". I.e. it will respond to any accelerator input.	
EV7.7.3	The "start" action must be configured such that it cannot inadvertently be left in the "on" position after system shutdown.	
EV10.5.1	Vehicles that pass the rain test will receive a "Rain Certified" sticker and may be operated in damp or wet conditions. See: ARTICLE D3. If a vehicle does not pass the rain test, or if the team chooses to forego the rain test, then the vehicle is not rain certified and will not be allowed to operate in damp or wet conditions.	Enclosure design
T7.1.1	The car must be equipped with a braking system that acts on all four wheels and is operated by a single control.	Brake Control Implementation

Figure B1: Formula Hybrid Relevant Rules

Appendix C

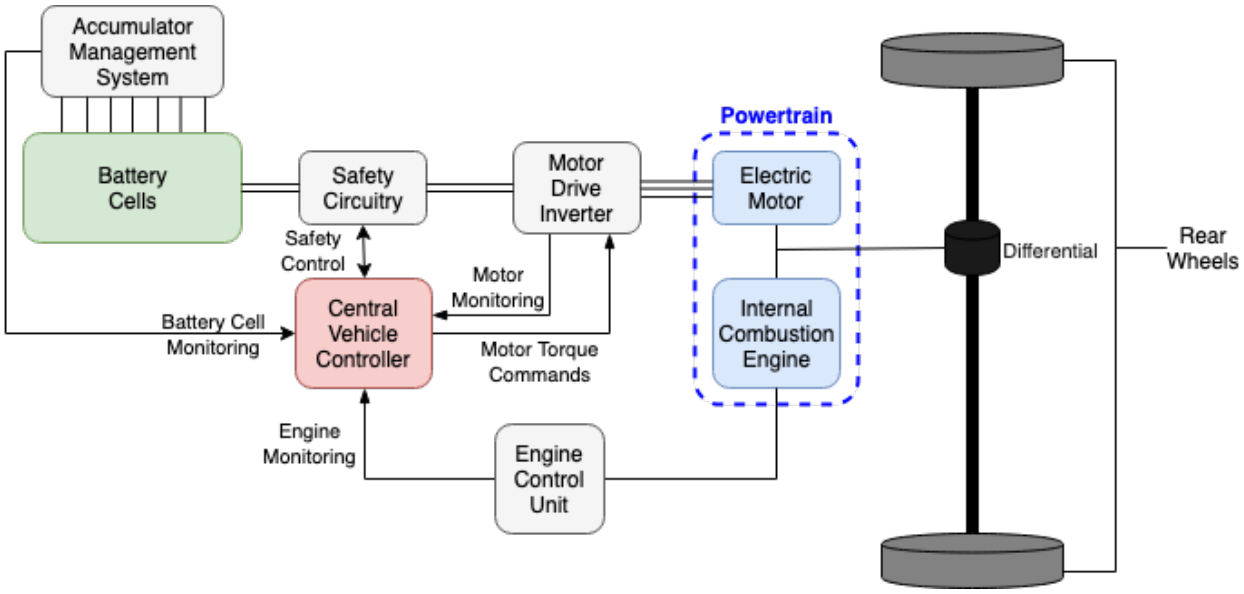


Figure C1: DFR System-Level Block Diagram

Appendix D

Recommended Data	Sensors Needed	Sensor Type	Throttle-by-wire Justification	Traction Control Justification	Regenerative Braking Justification	Automatic Gear Shifting Justification
Braking Pedal Force	Brakeline Pressure Sensors	Transducer (analog)	N/A	N/A	Driver applying force to brake pedal could be used to enable regen	N/A
Battery Pack Current	From Battery Management System (CAN)	N/A	N/A	N/A	Ensure that current sent back to battery pack does not exceed cell charge limit to avoid cell degradation	N/A
Engine RPM	From Engine ECU (CAN)	N/A	N/A	N/A	Ensure that brake regen is disengaged when low in the RPM range to avoid stalling the car	Gear shifts as a function of engine RPM and vehicle speed
Wheel Speeds	Wheel Speed Sensors	Pulse Train Output	N/A	Differences in wheel speeds are used to determine if the wheels have lost traction	Vehicle speed required to calculate torque and generated electricity fed into batteries	Gear shifts as a function of engine RPM and vehicle speed
Steering Wheel Angle	Steering Wheel Angle Position Sensor	Potentiometer (analog)	N/A	Represents cornering request from driver	N/A	N/A
Accelerator Pedal Position	Accelerator Pedal Position Sensor	Potentiometer (analog)	Represents torque request from driver, informs torque commands to engine and electric motor	Indicates requested torque from driver, important for understanding desired torque output	N/A	N/A
Engine Throttle Position	From Engine ECU (CAN) or potentiometer	Analog	Necessary to compare actual throttle position to requested throttle position	N/A	Driver's release of throttle could be used to enable regen	N/A
Motor RPM	From Drive Inverter (CAN)	N/A	Ensure motor does not exceed maximum RPM	N/A	N/A	N/A
Fuel Level	Fluid level sensor	Analog	Inform logic for torque split between engine and motor	N/A	N/A	N/A

Figure D1: Sensor Trade Study

Appendix E

	Signal	Type	Purpose	Justification
Required Inputs	Up-shift	12V Discrete Input	Electronic Shifting	Additional Feature Support
	Down-shift		Electronic Shifting	
	ICE Switch		Driving mode setting	DFR System Requirement
	Electric Motor Switch		Driving mode setting	
	Ready-to-drive input		FH Rules Requirement	
	Dash BRB press		FH Rules Requirement	
	IMD safety circuit fault		Detect IMD safety fault	
	BMS Safety Circuit Fault		Detect BMS safety fault	
	Bamocar Safety Circuit Fault		Detect Bamocar safety fault	
	Accelerator Pedal Position	Analog	Throttle By Wire, Traction Control	
	Rear master cylinder pressure		Regenerative braking	
	Front master cylinder pressure		Regenerative braking	
	Fuel level		Throttle By Wire	
	Engine Throttle Position		Throttle By Wire, Regenerative Braking	
	Steering wheel angle		Traction Control	
	FL wheel speed	PWM input	Traction Control, Regen Braking, Auto Gear Shifting	Additional Feature Support
	FR wheel speed			
	RL wheel speed			
	RR Wheel speed			
Required Outputs	Downshift solenoid	12V Discrete Output	Auto Gear Shifting	Additional Feature Support
	Upshift solenoid		Auto Gear Shifting	
	Ignition-kill		Auto Gear Shifting, Safety	
	Safety pin		FH Rules Requirement	DFR System Requirement
	Ready-to-drive out		FH Rules Requirement	
	Motor Enable		Enable motor output	
	Brake light		FH Rules Requirement	
	IC Fan On/Off		Engine Cooling	
	Motor torque control	Analog	Optional analog motor torque control	Additional Feature Support
	IC throttle control	PWM	Throttle By Wire	DFR System Requirement
	HV Cooling Pump		HV System Cooling	

Type	Quantity
12V Discrete Input	9
Analog Input	6
PWM Input	4
12V Discrete Output	8
Analog Output	1
PWM Output	2

Figure E1: CVC Required Inputs and Outputs

		Max Operating Speed	Value (MHz)	Flash	Size	SRAM	Size	ADCs	max # channels, # bits	SPI Availability	# of SPI	I2C Availability	# of I2C
	Weight	5		4		4		5		5		2	
MCU	STM32 F7 Series	5	216	5	512 KB - 2 MB	5	320 - 512 KB	4	24, 12	5	6	5	4
MCU	SAM E70	5	300	5	Up to 2MB	4	Up to 384 KB	5	32, 12	4	5	4	3
MCU	Kinetis KV-5x	5	240	4	Up to 1MB	3	Upt to 256KB	5	32, 12	3	3	3	2

		CAN Availability	# of CAN	USB Availability	# of USB	UART Availability	# of UART	GPIO Availabliity	# of GPIO	Availability of PCB Files	Open Source Altium Files Available	Total
	Weight	4		3		2		4		5		
MCU	STM32 F7 Series	5	3	5	1 / 2.0	3	4	5	168	5	Yes	206
MCU	SAM E70	3	2	5	1 / 2.0	5	8	3	114	0	No	163
MCU	Kinetis KV-5x	5	3	0	0	4	6	3	111	0	No	139

Figure E2: MCU Decision Matrix

		Ease of Hardware	Justification	Ease of Firmware	Justification	Max Frequency	Value (MHz)	Approx. Data Bandwidth	Value (MB/s)	Total
Weight		2		2		3		5		
Data Transfer Option	SPI	**5**	Needed hardware is provided by the Nucleo board	**5**	Firmware team has experience with SPI	**3**	27	**2**	3.219	**39**
Data Transfer Option	SDIO (1-bit)	**5**	Needed hardware is provided by the Nucleo board	**4**	Ample resources but firmware team doesn't have experience with SD transfer mode	**5**	50	**3**	5.96	**48**
Data Transfer Option	SDIO (4-bit)	**5**	Needed hardware is provided by the Nucleo board	**4**	Ample resources but firmware team doesn't have experience with SD transfer mode	**5**	50	**5**	5.96	**58**

Figure E3: Data Transfer Method Decision Matrix

		Data Transfer Speed	Value (Mbps)	Ease of Use	Justification	Difficulty to Implement	Justification	Range	Value (m)	Total
Weight		1		2		1		1		
Data Download Method	6LoWPAN	1	0.02 to 0.25	5	Wireless	3	Add-on board, SPI connection	4	10 to 30	18
	Ethernet	5	125	4	Quick plug/unplug	5	Already on board	3	Wired	21
	Bluetooth (v4.0)	4	24	5	Wireless, use phone or PC	3	Add-on board, SPI connection	4	10	21
	USB on the Go	5	480	4	Quick plug/unplug	5	Already on board	3	Wired	21
	WiFi	4	54	2	Wireless, requires internet connection	4	Add-on board, UART connection	4	20	16

Figure E4: Data Download Method Decision Matrix

		Variety of Communication Methods	Methods Available	Max Data Output Speed	Rate Value	Supports + 3g Motion	Value	Resolution	Value	Features	Justification	Total
Weight		4		4		5		4		5		
Accelerometer Chip	MPU 6050	3	I2C	5	8 kHz	5	Yes	5	0.122 mg/LSB (at +/- 4g), 8.75 mdps/LSB (at +/-245 dps)	5	Accelerometer, gyroscope	102
	LSM9DS1	5	SPI, I2C	3	80 Hz	5	Yes	5	0.122 mg/LSB (at +/- 4g)	5	Accelerometer, gyroscope, and magnetometer	102
	LIS331HH (A) & L3GD20	3	SPI, !2C (2x)	4	1 kHz (A), 760 Hz (G)	5	Yes	3	3 mg/LSB (at +/- 6g), 8.75 mdps/LSB (at +/- 245 dps)	5	Accelerometer, gyroscope	90

Figure E5: Accelerometer Selection Decision Matrix

Appendix F

Test power and safety circuitry off-board:	See below for components:								
LD39050PU33R	5V to 3.3V Linear Regulator	5V IN	3.3V	3.32V	Expected output				
LTC3624HMSE	12V to 3.3V Buck Converter	12V IN	3.3V	3.28V	Expected output				
5V Protection	5V Protection Circuitry	5V IN	3.3V	3.14V	Expected output	12V IN	3.3V	7.63V	Too high (fail)
12V Protection	12V Protection Circuitry	12V IN	3.3V	2.516V	Okay (within threshold)	24V IN	3.3V	2.99V	Expected output
DC-DC	12V to 5V DC-DC Converter	12V IN	5V	5.01V	Expected output				
LM317DCY	12V to 3.3V Linear Regulator	12V IN	3.3V	3.16V	Expected output				

Figure F1: Pre-Board Arrival Test Results

Pre-Power	**Description**	**Checked**
Solder on DNM Components	See below for components:	x
L1	Bead inductor	x
L2	Bead inductor	x
DC-DC converter	12V to 5V DC-DC Converter	x
Bluetooth Module	Bluetooth Low Energy Module	Waiting on delivery
Test for shorts on all ICs	See below for checked ICs:	x
LD39050 (U5)	5V to 3.3V Linear Regulator	x
LTC3624 (U19)	12V to 3.3V Buck Converter	x
MCU (U6)	STM32 Microcontroller	x
STMPS (U9)	Mosfet Switch	x
H1102NL (U3)	Transformer	x
LAN (U4)	Ethernet Transceiver	x
USBLC6 (U2)	ESD Protection	x
LSM9DS1 (U1)	Accelerometer	x
SD Card Socket (U13)	SD Card Socket	x
Bluetooth Module (U14)	Bluetooth Low Energy Module	Can't yet
74LVC2G34 (IC1)	Dual Buffer Gate	x
LM317DCY (U17, U18)	Voltage Regulator	x
SN65HVD (B1, B3)	CAN Transceiver	x
CLT01 (U7)	SPI Transmitter	x
Si8662 (B7)	Level Shifter	x
VNI8200XP (U8)	SPI Receiver	x
CC6 (DC-DC1)	12V to 5V DC-DC Converter	x
Make sure power traces reach the proper pins	Continuity check MCU pins and connection points	x
Check all component values from BOM	Measure component values	x
Ensure everything has been soldered correctly	Visually inspect solder job	x
Set current limits on power supply (1.8A)	Use current-limiting power supply	x
Check for any soldering irregularities	Visually inspect solder job	x
Solder PGND to GND	Ensure all components on same ground plane	x

Figure F2: Pre-Power Test List

Post-Power	Description	Checked	Voltage Reading 1	Voltage Reading 2
Power the PLC to ensure no errors	Check for error light:	Voltage high		
Check power LEDs	Power good LED should be on, error light off	PLC error LED on		
Power remainder of board and check voltages on ICs	See below for checked ICs:			
LD39050 (U5)	5V to 3.3V Linear Regulator	checked	5.01V in	3.31V out
LTC3624	12V to 3.3V Buck Converter	checked	12.02V in	3.37V out
MCU (U6)	STM32 Microcontroller	checked	3.36V	
STMPS (U9)	Mosfet Switch	checked	5.01V	
H1102NL (U3)	Transformer	checked	3.36V	
LAN (U4)	Ethernet Transceiver	checked	3.36V	
USBLC6 (U2)	ESD Protection	checked	3.37V	
LSM9DS1 (U1)	Accelerometer	checked	3.31V	
SD Card PRT-12769 (U13)	SD Card Socket	checked		
MDBT40-256RV3 (U14)	Bluetooth Low Energy Module	No BT		
SN74LVC2G34DBVR (IC1)	Dual Buffer Gate	No BT		
LM317DCY (U17, U18)	Voltage Regulator	checked	12.05V in	3.26V out
SN65HVD (B1, B3)	CAN Transceiver	checked	3.31V	
CLT01-38SQ7-TR (U7)	SPI Transmitter	checked	3.31V	
SI8662AB-B-IS1 (B7)	Level Shifter	checked	3.31V	6.62V (5V expected)
VNI8200XP (U8)	SPI Receiver	checked	6.62V (5V expected)	
CC6-1205SF-E (DC-DC1)	12V to 5V DC-DC Converter	checked	12.02V in	5.01V out
Make sure that supply isn't hitting current limit	Woest-case scenario: 1.8 amps	checked	140mA	
Ensure validity of connections for connectors	Continuity checks - see below:	checked		
J1	Jumper 1 - Continuity Check	checked		
J2	Jumper 2 - Continuity Check	checked		
CN3	USART to USB Connector - Continuity Check	checked		
CN*1	40-Pin Connector - Continuity Check	checked		

Figure F3: Post-Power Test Results

Voltage Pin	Checked	Ground Pin	Checked
6	x	16	x
17	x	38	x
30	x	51	x
32	x	61	x
33	x	83	x
39	x	94	x
52	x	107	x
62	x	120	x
72	x	130	x
84	x		
95	x		
108	x		
121	x		
131	x		
143	x		
144	x		

Figure F4: MCU Power and Ground Check Data

VNI Circuitry	**Value**	**On-board measurement**	**Off-board measurement**
C81	22nF	26nF	On-board was good
L3	100uH	575uH	570uH
C92	10nF	14nF	10.4nF
R92	2k	.79k	2.005k
C94+C95	5.17uF	5.2uF	On-board was good
R94	3.3k	0.136k	3.25k6
R93	10k	.794k	9.962k
C96	100pF	5.3uF	101.5pF

Figure F5: VNI Circuitry Component Value Checks

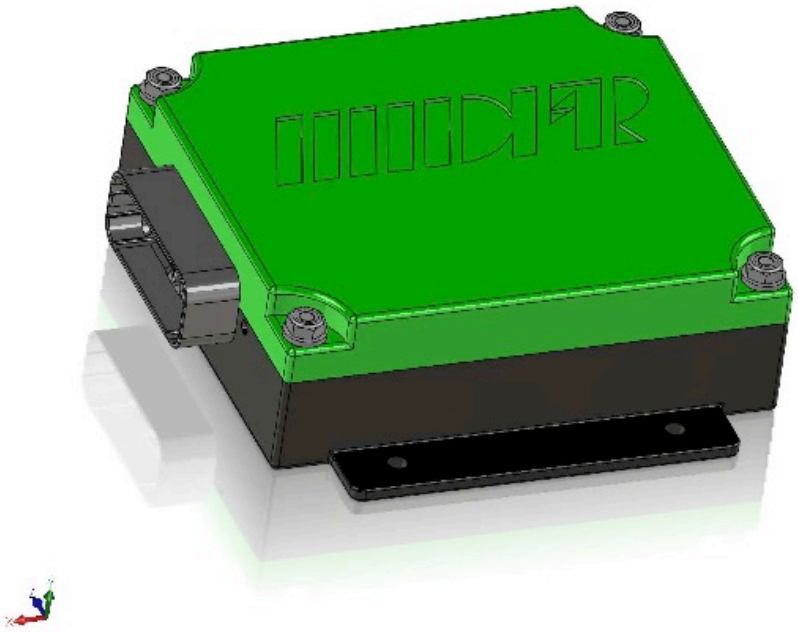


Figure F6: Final Enclosure CAD Assembly

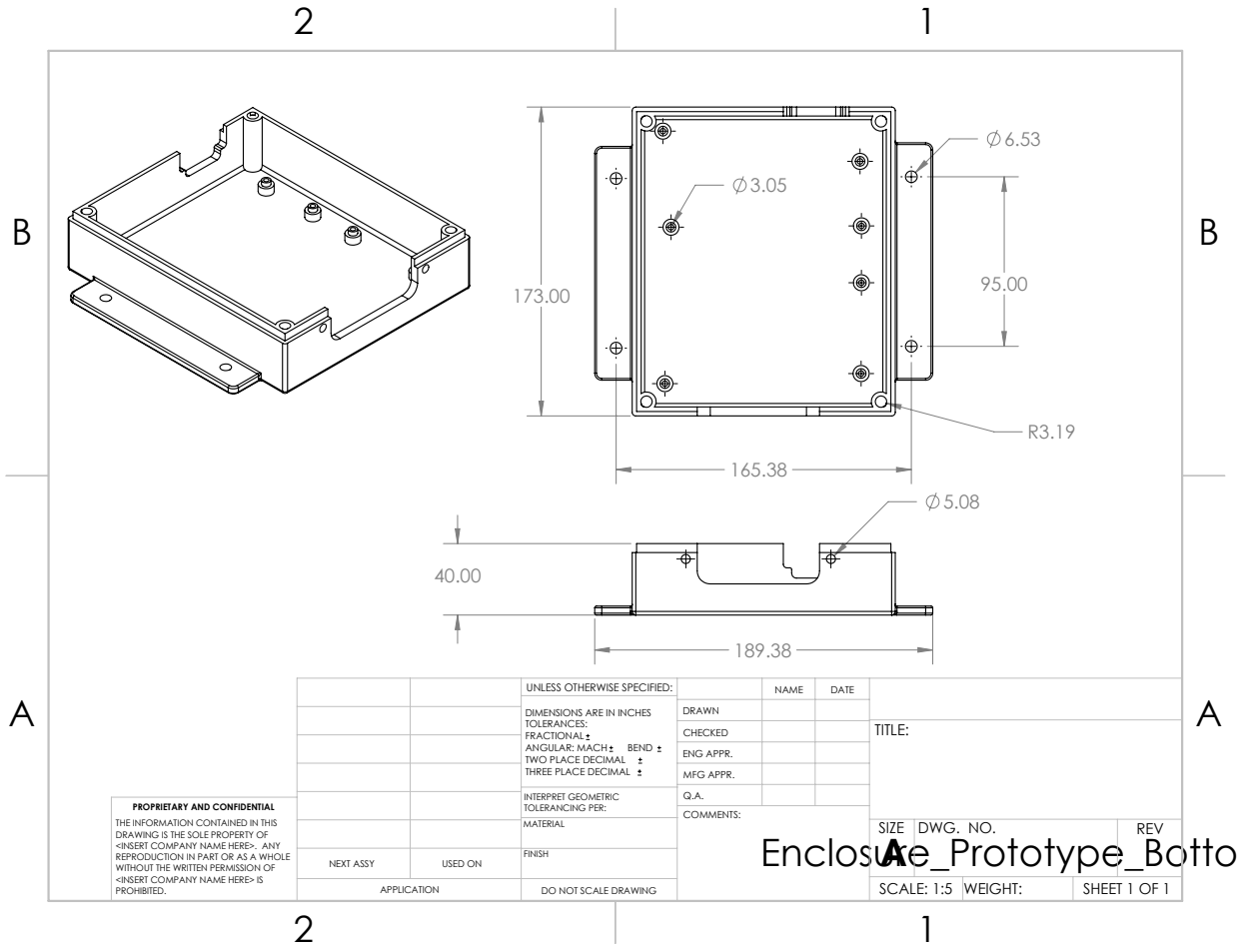


Figure F7: Bottom of Enclosure CAD Drawing

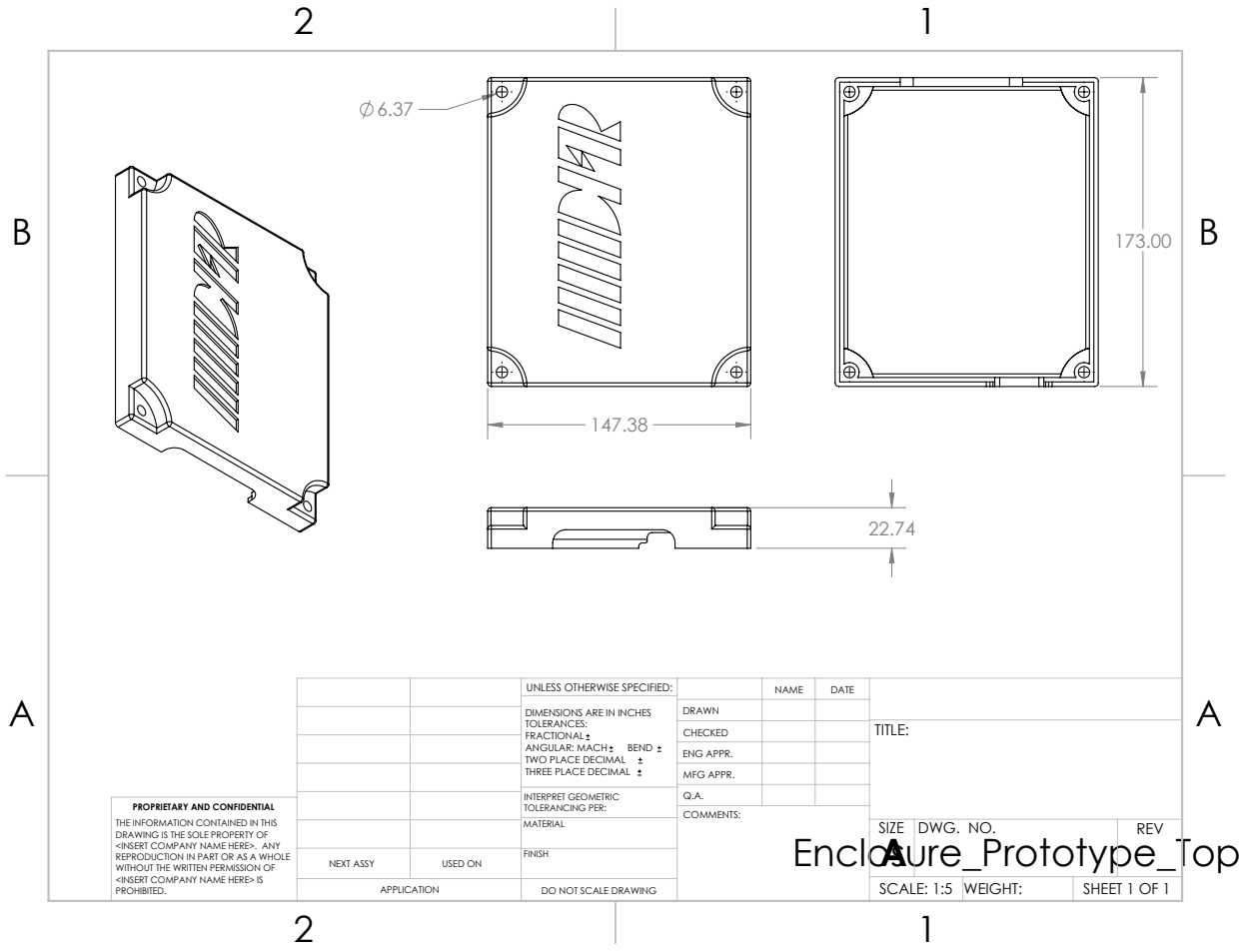


Figure F8: Top of Enclosure CAD Drawing

Appendix G

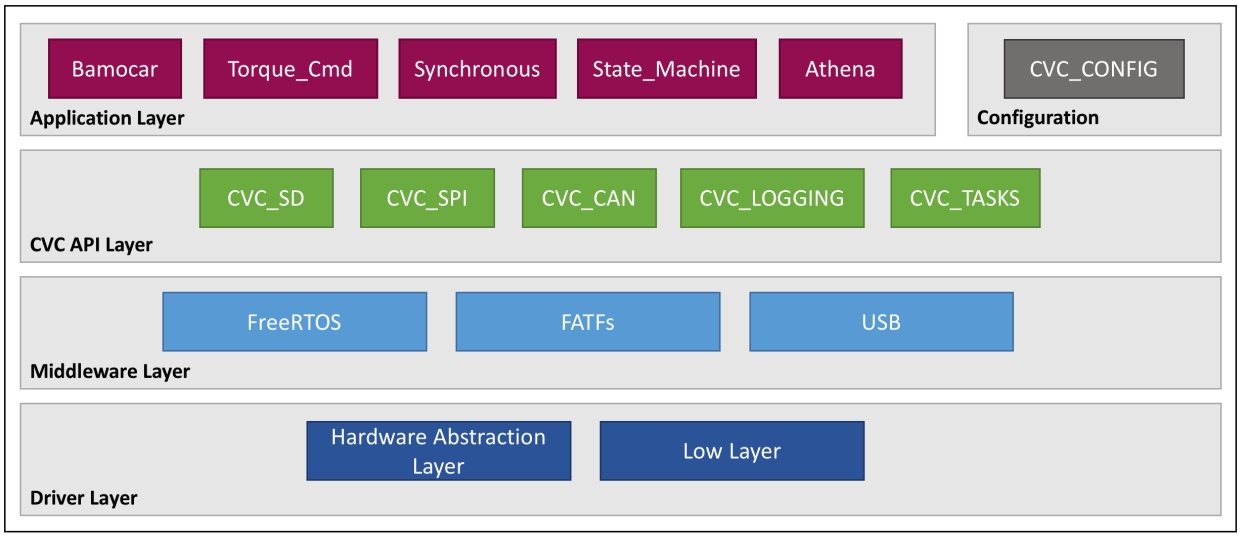


Figure G1: Firmware Architecture Map

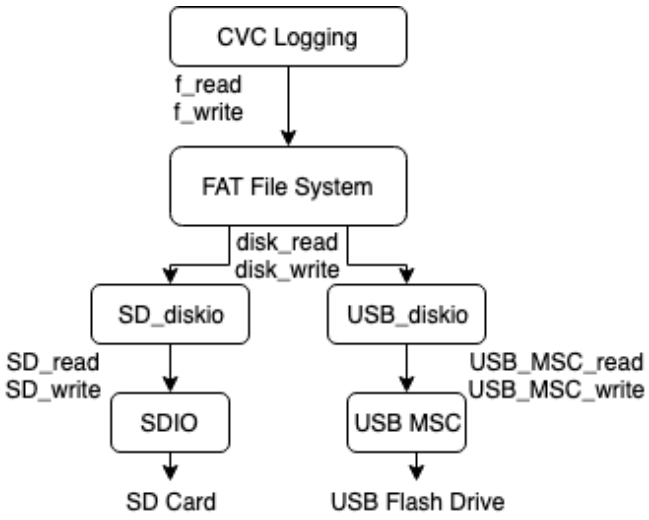


Figure G2: FAT File System Block Diagram

Subsystem	CAN Message ID	Reg ID (Bamocar)	Message	Transmission Frequency
Bamocar	0x180		Status	10 Hz
Bamocar	0x180	0x30	Motor RPM (MSB)	10 Hz
Bamocar	0x180	0x5F	Motor Current	10 Hz
Bamocar	0x180	0xA0	Motor Torque	10 Hz
Bamocar	0x180	0x8A	Motor Voltage	10 Hz
Bamocar	0x180	0x49	Motor Temp	10 Hz
Bamocar	0x180	0x8F	Bamocar Fault	10 Hz
Bamocar	0x180	0xEB	Bamocar Bus Voltage	10 Hz
Bamocar	0x180	0xE0	Bamocar D_1_OUT_1	10 Hz
Bamocar	0x180	0xE1	Bamocar D_1_OUT_2_	10 Hz
Athena ECU	0x200	-	Engine Revolution Counter	100 Hz
0x200	-	RPM on cycle	2	
0x200	-	Throttle Position %	4	
0x200	-	Manifold Air Pressure	6	
Athena ECU	0x310	-	Engine Temperature	100 Hz
0x310	-	Air Temperature	1	
0x310	-	Oil Temperature	2	
0x310	-	KL15 Battery Voltage	3	
0x310	-	KL30 Battery Voltage	4	
0x310	-	Barometric Pressure	6	
Athena ECU	0x312	-	Fan Overheat Flag	100 Hz
-	Active Map or Driving Mode	5	0	
EMUS BMS	0x1B6	-	Min Cell Voltage	100 Hz
-	Max Cell Voltage	1	0	
-	Average Cell Voltage	2	0	
-	TOTAL VOLTAGE (2nd byte)	3	0	
-	TOTAL VOLTAGE (LSB)	4	0	
-	TOTAL VOLTAGE (MSB)	5	0	
-	TOTAL VOLTAGE (3rd byte)	6	0	
EMUS BMS	0x1B7	-	Min Cell Temp	
-	Max Cell Temp	1	0	
-	Average Cell Temp	2	0	
EMUS BMS	0x1BA	-	CURRENT (MSB)	
-	CURRENT (LSB)	1	0	
-	ESTIMATED CHARGE (MSB)	2	0	
-	ESTIMATED CHARGE (LSB)	3	0	
-	ESTIMATED STATE OF CHARGE	6	0	

Figure G3: CAN Message Dictionary

Branch	Module Purpose	Test
SPI_driver	Test function of SPI hardware driver	Transmit data from a master board to a slave board and compare data received with data sent to ensure no data is lost
CAN_driver	Test function of CAN hardware driver	Transmit data between two boards that coordinates LEDs, incrementing through the colored LEDs on user button press
ADC_driver	Test function of the ADCs and ADC driver	Attempt to read data from a GPIO pin using a DMA buffer, used a current-limited power supply to feed signal
I2C_driver	Test function of the I2C driver	Transmit data from master to slave and back to master, LEDs illuminate if successful or unsuccessful
TIM_INPUTCAPTURE_driver	Test fucntion of timer-capture and pwm pins	Use timer to output pwm signal from one timer pin and use timer-capture input pin to read signal and compare to signal sent
PLC_rtos	Test PLC SPI communication functions in freertos	Read 12V inputs and output matching 12V pattern
CAN_protection	Test semaphore-protected implementation of CAN functions	Read CAN messages from another board, cross-check data read with fake data sent
PLC_protection	Test semaphore-protected implementation of PLC functions	Actuate 12V outputs to match 12V inputs
CAN_parser_validation	Test CAN parsing functions in FreeRTOS	If SENDER macro defined, send CAN messages with recognizable identifiers and known data values (flash to nucleo test bench). Flash test hardware without SENDER defined, use debugger to check CAN_inputs for expected values
SD_driver_validation	Test SD driver in bare metal (NOT FreeRTOS)	Use FatFs-provided driver test function to test basic SD_diskio driver. Green and Blue LEDs light up if test successful, red LED if test fails
SD_driver_rtos	Test SD driver in FreeRTOS	Use FatFs-provided driver test function to test FreeRTOS-compatible SD_diskio driver. Green and Blue LEDs light up if test successful, red LED if test fails
SD_rtos	Test writing to SD using FatFs API in FreeRTOS	Simple task opens file, writes using f_write, closes file. Check file written to SD using laptop.
startup	Test vehicle startup sequence, no data logging	On vehicle, test startup sequence; ensure no fault occurs, ensure vehicle enters each state properly

Figure G4: Firmware Testing Summary

Figure G5: Safety Hierarchy Explanation

The fault status is an enumerated type that consists of the following values: CVC_OK = 0, CVC_WARNING = 1, CVC_RST_FAULT = 2, CVC_HARD_FAULT = 3. CVC_OK indicates that the car is working as expected, and the error handler need not be called. CVC_WARNING indicates a minor anomaly (such as a slightly high engine coolant temperature) and will send a warning to the driver, without shutting down the car fully. A CVC_RST_FAULT will shut down the car, but allow the driver to restart the car without outside assistance (assuming that the fault has cleared). A CVC_HARD_FAULT indicates a serious system failure (such as an engine coolant temperature near boiling, 90 C). This fault shuts the car down completely and does not allow the driver to restart until the external GLVS button has been pressed.

The cvc error code is another enumerated type that assists in debugging faults. This enumerated type holds different types of errors (QUEUE_ERR = 1, CAN_ERR = 2, VOLTAGE_ERR = 3, ENGINE_ERR = 4, LOGGING_ERR = 5). Currently these values can only be read through an external debugger, however, in the future these error codes can be displayed on the dashboard to further ease the debugging process.

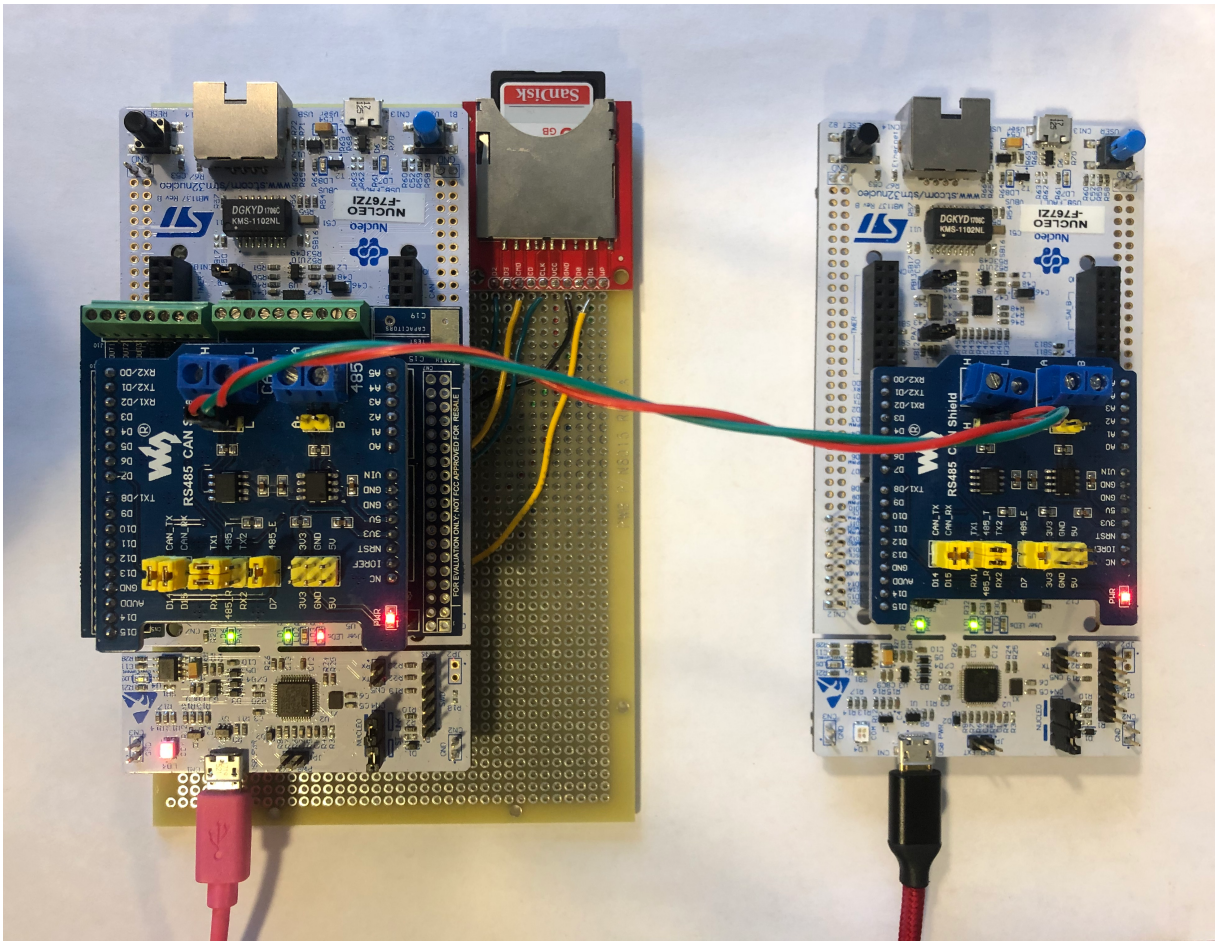


Figure G6: CAN Testing using hardware prototype (left) and Nucleo-CAN Shield Test Bench

SPI Outputs Vector	
OUT1	Empty
OUT2	Empty
OUT3	Downshift
OUT4	Upshift
OUT5	safety
OUT6	RTD
OUT7	BamoRFG
OUT8	Ignition_kill

SPI Inputs Vector	
IN1	Empty
IN2	Empty
IN3	ICE_EN
IN4	Motor_EN
IN5	RTD_in_button
IN6	Dash_BRB
IN7	Empty
IN8	Empty

Figure G7: SPI Inputs and Outputs Vectors

Link to GitHub Repository: https://github.com/alexnewman23/DFR_CVC

Appendix H

Figure H1: Enclosure Mounted on the DFR Car

```
% Alex Newman
% Engs 90
% Driving Data Analysis

clear all;
close all;
clc;
```

Conversion Constants

```
bamocar_Nmax_setting = 6000;
bamocar_rpm_scale = 32767;

bamocar_current_scale = 32767;
bamocar_current_conversion = 282/(500);

bamocar_voltage_scale = 32767;
bamocar_voltage_conversion = 460/32767;

final_drive_reduction = 11/34;
rolling_radius = 0.254;                    % m

engine_tps_max_val = 232;

mps_to_mph = 2.2369363;

battery_voltage_multiplier = 0.01;
battery_current_multiplier = 0.1;
```

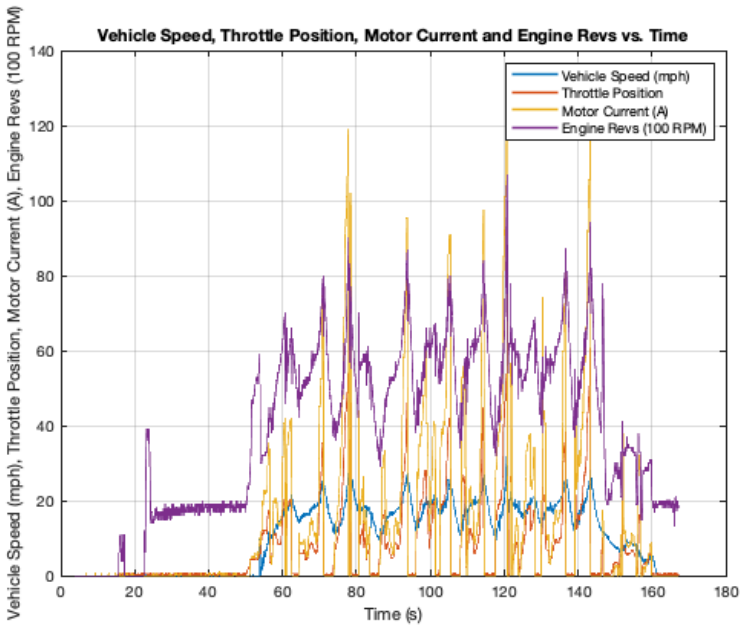
Main

```
%Data = readtable('LOG_FILE_2-27-19_DRIVING_RAW.csv');
Data = readtable('LOG_FILE_3-1-19_DRIVING_RAW.csv');
Ticks = Data.TICKS;

Bamocar_Bus_voltage = Data.BAMO_BUS_VOLTAGE;
Motor_RPM = Data.MOTOR_RPM;
Motor_Current = Data.MOTOR_CURRENT;
Motor_Torque = Data.MOTOR_TORQUE;
Motor_Voltage = Data.MOTOR_VOLTAGE;
Motor_Temperature = Data.MOTOR_TEMP;

Engine_RPM = Data.ENG_RPM;
Engine_TPS = Data.ENG_TPS;
Engine_Temperature = Data.ENG_TEMP;
Air_Temperature = Data.AIR_TEMP;

Battery_voltage = Data.BATT_VOLTAGE;
Battery_current = Data.BATT_CURRENT;

true_engine_rpm = swapbytes(uint16(Engine_RPM));
```

```
time = Ticks*0.001;     % seconds

% clean Motor_RPM, Motor_Current and Motor_Voltage data
size = length(Motor_RPM);
for i=1:1:size

    if (Motor_RPM(i) > bamocar_rpm_scale)
        Motor_RPM(i) = 0;
    else
        Motor_RPM(i) = Motor_RPM(i)*bamocar_Nmax_setting/
bamocar_rpm_scale;
    end

    if (Motor_Current(i) > bamocar_current_scale)
        Motor_Current(i) = 0;
    else
        Motor_Current(i) =
 Motor_Current(i)*bamocar_current_conversion;
    end

    if (Motor_Voltage(i) > bamocar_voltage_scale)
        Motor_Voltage(i) = 0;
    else
        Motor_Voltage(i) =
 Motor_Voltage(i)*bamocar_voltage_conversion;
    end

end

vehicle_speed = Motor_RPM * (1/60) * final_drive_reduction * 2 * pi...
                * rolling_radius;                   % m/s
vehicle_speed_mph = mps_to_mph*vehicle_speed;       % mph


% clean TPS data
for i=1:1:size
    if Engine_TPS(i) > engine_tps_max_val
        Engine_TPS(i) = engine_tps_max_val;
    end
end


% clean engine temperature data
for i=1:1:size
    if Engine_Temperature(i) ~= 0
        Engine_Temperature(i) = Engine_Temperature(i)-40;
    end

    if Air_Temperature(i) ~= 0
        Air_Temperature(i) = Air_Temperature(i)-40;
    end
end
```

```
% convert and clean battery voltage
Battery_voltage = Battery_voltage*battery_voltage_multiplier;
for i=1:1:size
    if Battery_voltage(i) > 1000
        Battery_voltage(i) = median(Battery_voltage);
    end

end


scaled_TPS = Engine_TPS*(100/engine_tps_max_val);


figure(1);
plot(time, vehicle_speed_mph, 'Linewidth', 1.5);
xlabel('Time (s)');
ylabel('Vehicle Speed (mph), Throttle Position, Motor Current (A),
 Engine Revs (100 RPM)');
title('Vehicle Speed, Throttle Position, Motor Current and Engine Revs
 vs. Time');

hold on;
plot(time, scaled_TPS, 'Linewidth', 1.5);
plot(time, Motor_Current, 'Linewidth', 1.5);
plot(time, true_engine_rpm/100, 'Linewidth', 1.5);
legend('Vehicle Speed (mph)', 'Throttle Position', 'Motor Current
 (A)', 'Engine Revs (100 RPM)');
grid on;
hold off;


figure(2);
plot(time, Engine_Temperature, 'Linewidth', 1.5);
xlabel('Time (s)');
ylabel('Engine Temperature (C)');
title('Temperatures vs. Time');

hold on;
plot(time, Air_Temperature, 'Linewidth', 1.5);
legend('Engine Temperature', 'Air Temperature', 'Location', 'east');
hold off;


figure(3);
plot(time, Battery_voltage, 'Linewidth', 1.5);
title('Throttle Position, Battery Voltage and Engine Revs vs. Time');
xlabel('Time (s)');
ylabel('Battery Voltage (V), Engine revs (RPM), Throttle Position');

hold on;
plot(time, true_engine_rpm/100, 'Linewidth', 1.5);
plot(time, scaled_TPS, 'Linewidth', 1.5);
legend('Battery Voltage (V)', 'Engine Revs (100 RPM)', 'Throttle
 Position', 'Location', 'east');
```

```
hold off;
```


Vehicle Speed, Throttle Position, Motor Current and Engine Revs vs. Time
Vehicle Speed (mph)
Throttle Position
Motor Current (A)
Engine Revs (100 RPM)
Vehicle Speed (mph), Throttle Position, Motor Current (A), Engine Revs (100 RPM)
Time (s)
0
20
40
60
80
100
120
140
160
180

KK

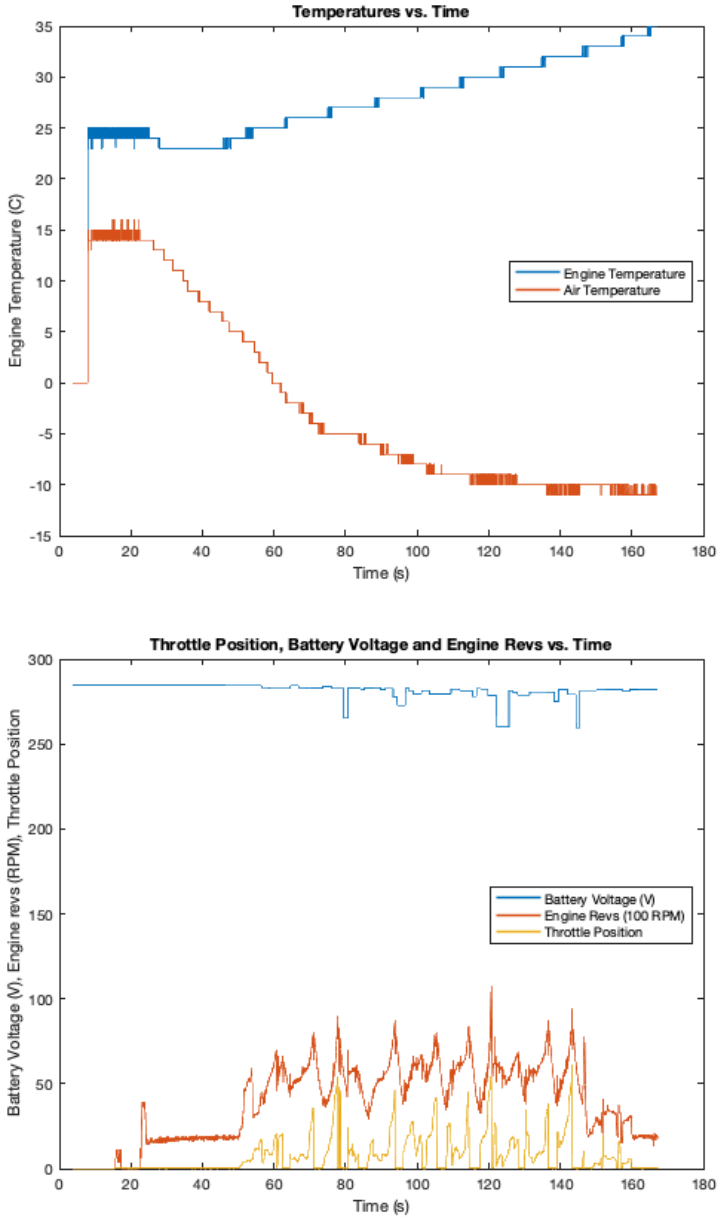


Figure H2: MATLAB Script Used to Process Driving Data

Appendix I

Procedure	Quantity	Turn Around Time (days)	Cost
Assembly	3	3	$694.34
PCB Fabrication	5	1	$2,015
Parts (including extra parts)	3	3	$2,581.19
Non-recurring expenses (machine configuration)			$465
Total	3	7	$5,755.53

Figure I1: PCB Fabrication and Assembly Costs

Board Name	Supplier	Manufacturer	Part Number	Unit Price
NUCLEO F767ZI	Digi-Key	STMicroelectronics	497-16525-ND	$27.67
Adafruit Bluetooth SPI Friend	Adafruit	Adafruit	2633	$17.50
SparkFun SD/MMC Card Breakout	SparkFun	4UCON	12941	$9.95
RS485 CAN Shield	Waveshare	Waveshare	rs485-can-shield	$11.99
X-NUCLEO-PLC01A1	Digi-Key	STMicroelectronics	497-16001-ND	$10
SparkFun 9DoF IMU Breakout	SparkFun	STMicroelectronics	13284	$15.95
Total				$93.06

Figure I2: Firmware Prototype Costs

3D HUBS

3D Hubs Manufacturing LLC
228 East 45th Street Suite 9E
10017, New York, NY

Quote: 1LH2RKT2

Bill to

Thayer School of Engineering
Trammell Saltzgaber
14 Engineering Drive
Hanover, NH 03755
United States
alexandernewman23@gmail.com
+1 646-306-2032

Ship to

Thayer School of Engineering
Alexander Newman
14 Engineering Drive
Hanover, NH 03755
United States
alexandernewman23@gmail.com
+1 646-306-2032

Quote Details

Quote #	1LH2RKT2
Lead Time	4 Business Days
Quote Date	2019-02-14
Expiry Date	2019-03-16

Secure Payment Link Powered by stripe
https://www.3dhubs.com/manufacture/payment/quote/1793d1f3-fa98-497c-96fa-6fe9b3ca2453

	Description		Qty	Unit Price	Price
1	**Enclosure_Prototype_Bottom.SLDPRT** 189.4x40.0x173.0 mm	3D printing / FDM / Standard ABS / Black / 100µm / 100% infill	1	$129.71	$129.71
2	**Enclosure_Prototype_Top.SLDPRT** 147.4x22.7x173.0 mm	3D printing / FDM / Standard ABS / Green / 100µm / 100% infill	1	$101.93	$101.93
Shipping					$0.00

Subtotal	$231.64
Sales tax	$0.00
Total	USD $231.64

Signature:

Trammell Saltzgaber

By signing or submitting a payment, customer agrees to the specifications of the quote (#1LH2RKT2) and the attached Terms & Conditions.

Payment Details

Secure Payment Link https://www.3dhubs.com/manufacture/payment/quote/1793d1f3-fa98-497c-96fa-6fe9b3ca2453 Powered by stripe

Figure I3: Quote for 3D Printing of Enclosure

Number of PCBs	Assembly Turn Around Time	PCB Day Turn Around Time	Cost
10	5	10	$11,643.36
50	5	10	$33,098.12
100	5	10	$57,041.00

Figure I4: Cost of Mass-Producing PCB

Number of Units	Unit Cost of Top	Unit Cost of Bottom	Total Cost
1	$103.28	$134.90	$238.18
2	$91.19	$121.92	$426.22
5	$78.96	$112.37	$956.65
10	$72.71	$106.02	$1,787.30
25	$67.50	$100.08	$4,189.50
50	$65.14	$94.88	$8,001.00
100	$62.18	$91.00	$15,318.00
200	$62.14	$90.95	$30,618.00
300	$62.14	$90.94	$45,924.00
400	$62.13	$90.94	$61,228.00
500	$62.13	$90.94	$76,535.00

Figure I5: Cost of Mass-Producing Enclosure without Injection Molding

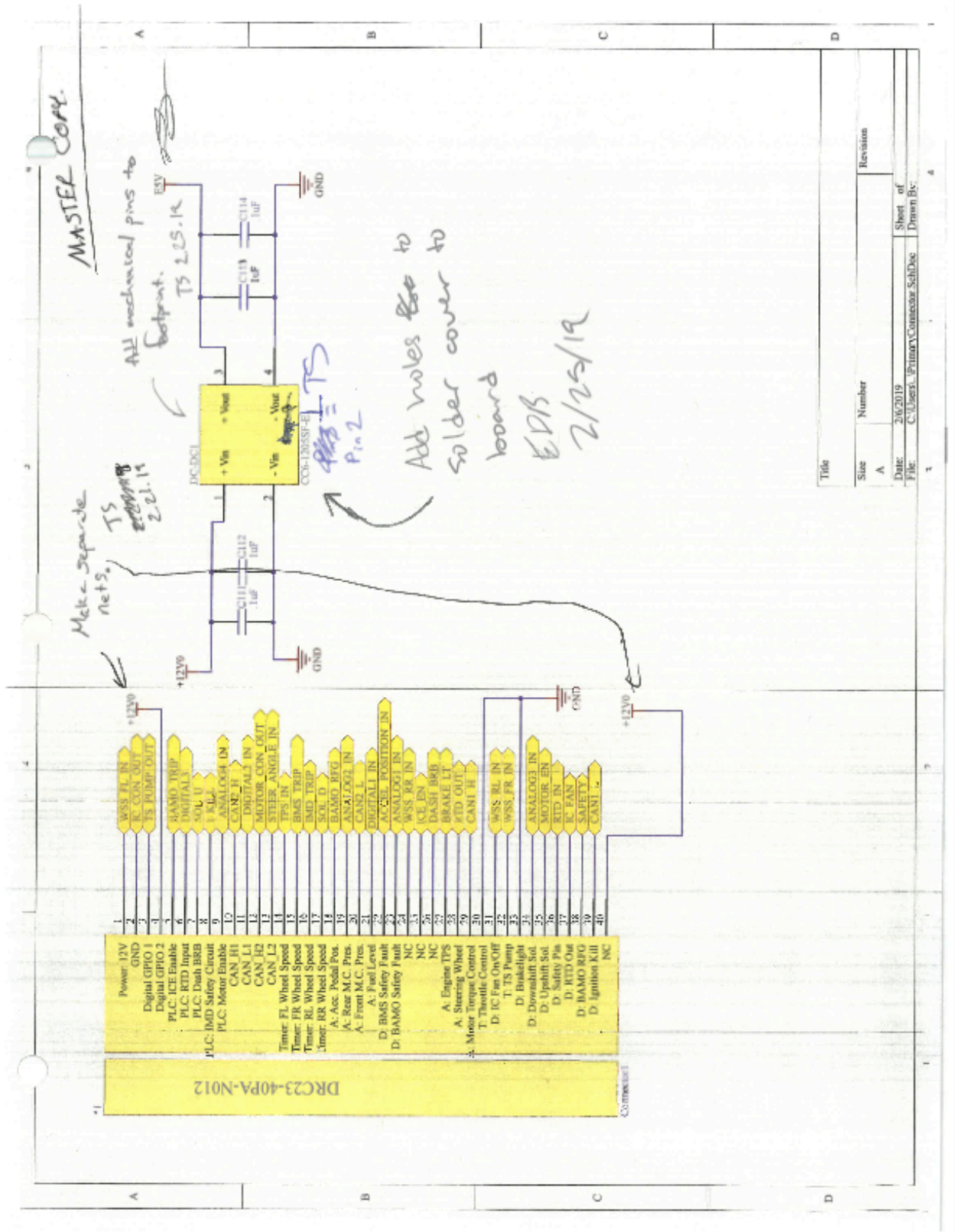
Number of Units	Tooling Cost for Top	Tooling Cost for Bottom	Unit Cost of Top	Unit Cost of Bottom	Total Cost
500	$6,496.65	$8,557.42	$0.91	$1.51	$16,264.07
1000			$0.84	$1.38	$17,274.07

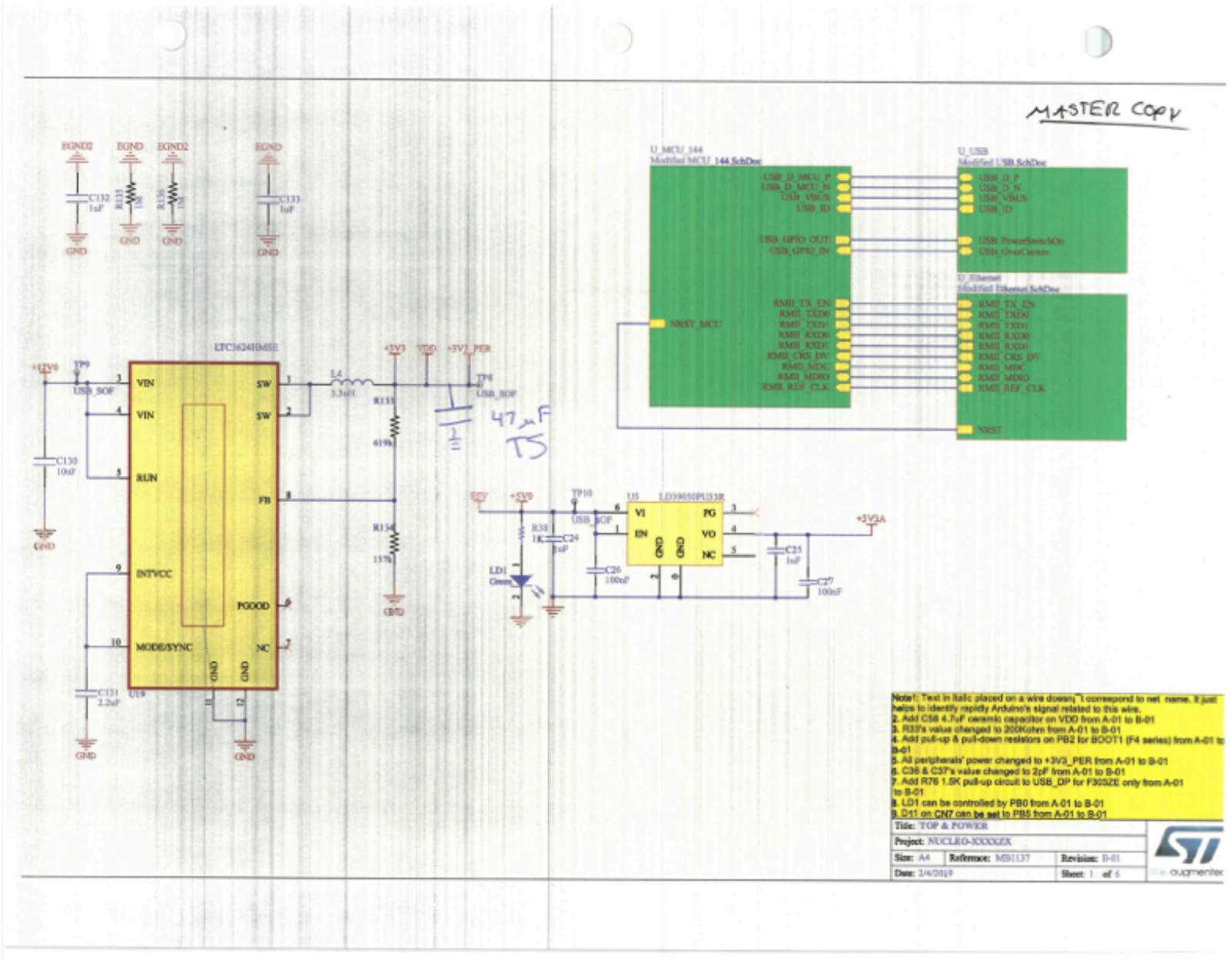
Figure I6: Cost of Mass-Producing Enclosure with Injection Molding

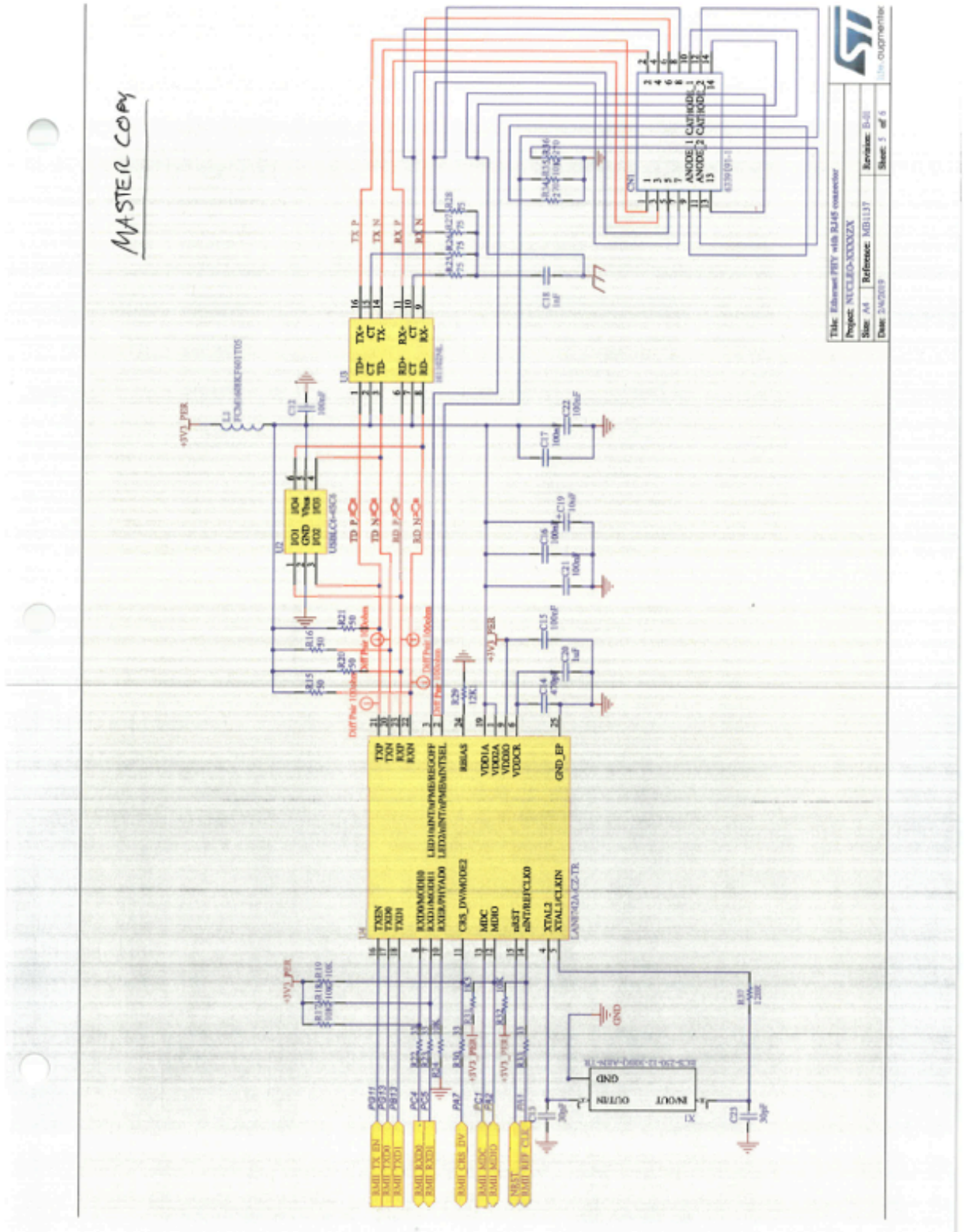
Year	% Market	Revenue	# units sold	Profit	Total Profit	Market size = 50,000,000
1	1	500000	294	327422	327422	
2	2	1000000	588	654844	982266	
3	3	1500000	882	982266	1964532	
4	4	2000000	1176	1309688	3274220	
5	5	2500000	1471	1636523	4910743	

Figure I7: Market Analysis and Profit Estimates

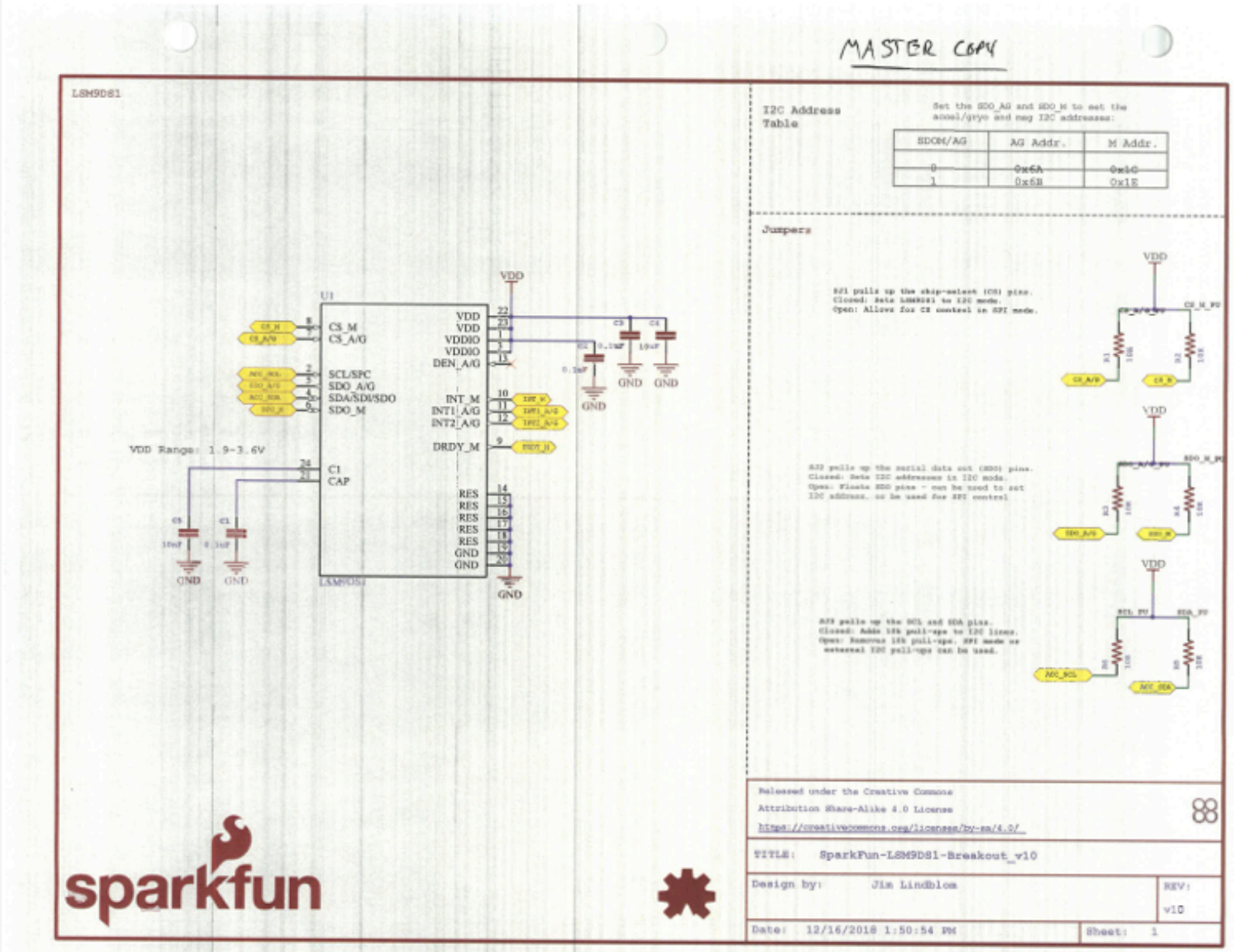
Appendix J


MASTER COPY
All mechanical pins to footprint.
TS 2.25.19
Make separate nets.
TS 2.21.19
Add holes here to solder cover to board
EDS 2/25/19
DC-DC1
CC6-1205SF-E
+ Vin
- Vin
+ Vout
- Vout
C111 1uF
C112 1uF
C113 1uF
C114 1uF
+12V0
5V
GND
DRC23-40PA-N012
Power 12V
GND
Digital GPIO 1
Digital GPIO 2
PLC: ICE Enable
PLC: RTD Input
PLC: Dash BRB
PLC: IMD Safety Circuit
PLC: Motor Enable
CAN_H1
CAN_L1
CAN_H2
CAN_L2
Timer: FL Wheel Speed
Timer: FR Wheel Speed
Timer: RL Wheel Speed
Timer: RR Wheel Speed
A: Accel. Pedal Pos.
A: Rear M.C. Pres.
A: Front M.C. Pres.
A: Fuel Level
D: BMS Safety Fault
D: BAMO Safety Fault
NC
NC
NC
A: Engine TPS
A: Steering Wheel
T: Motor Torque Control
T: Throttle Control
D: IC Fan On/Off
T: TS Pump
D: Brakelight
D: Downshift Sol.
D: Upshift Sol.
D: Safety Pin
D: RTD Out
D: BAMO RFG
D: Ignition Kill
NC
WSS_FL_IN
IC_CON_OUT
TS_PUMP_OUT
BAMO_TRIP
DIGITAL5
SOL_U
I_KILL
ANALOG4_IN
CAN2_H
DIGITAL2_IN
MOTOR_CON_OUT
STEER_ANGLE_IN
TPS_IN
BMS_TRIP
IMD_TRIP
SOL_D
BAMO_RFG
ANALOG2_IN
CAN2_L
DIGITAL1_IN
ACCEL_POSITION_IN
ANALOG1_IN
WSS_RR_IN
ICE_EN
DASH_BRB
BRAKE_LT
RTD_OUT
CAN1_H
WSS_RL_IN
WSS_FR_IN
ANALOG3_IN
MOTOR_EN
RTD_IN
IC_FAN
SAFETY
CAN1_L
Title
Size
A
Number
Revision
Date: 2/6/2019
File: C:\Users\...\Primary\Connector SchDoc
Sheet of
Drawn By:

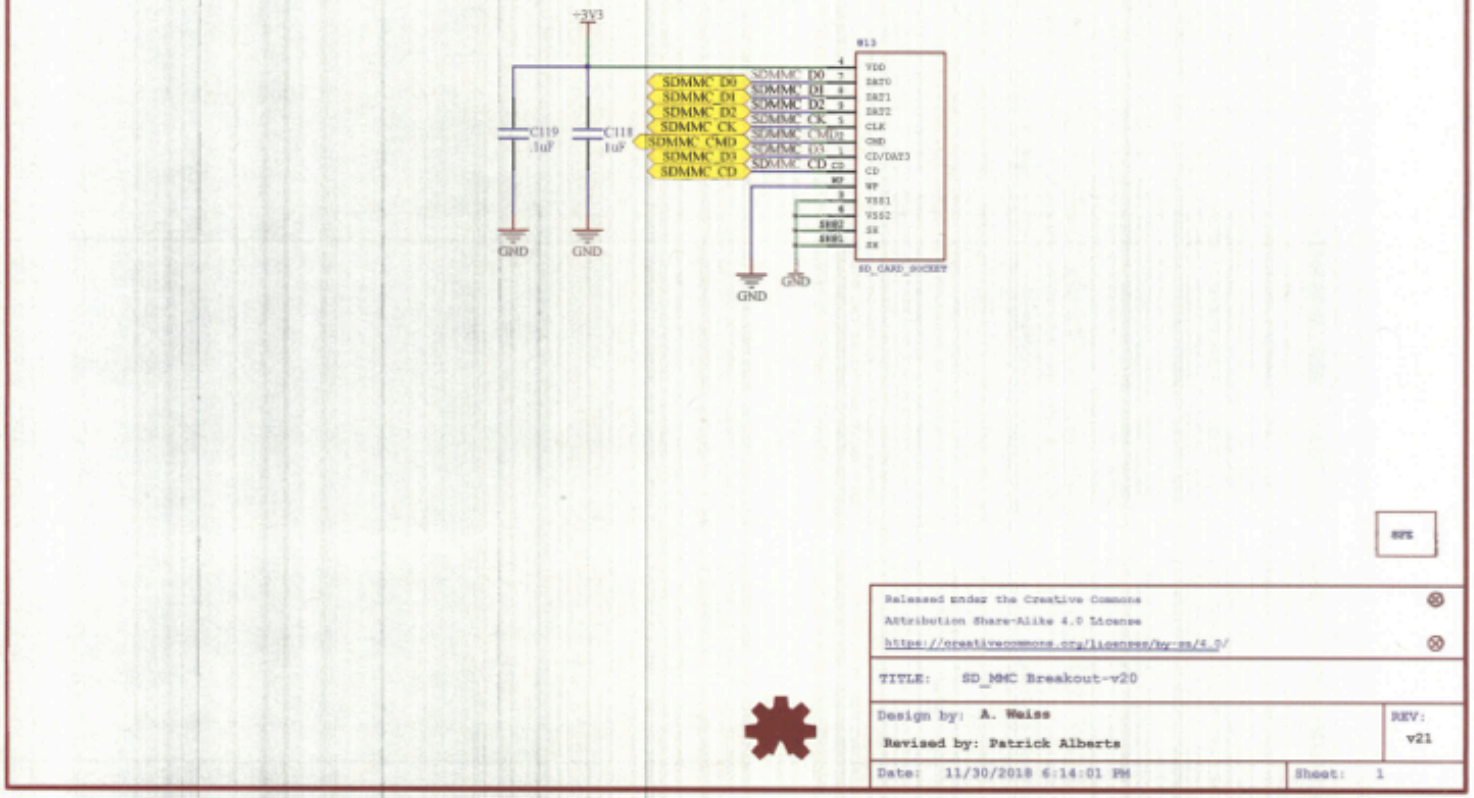

MASTER COPY
U_MCU_144
Modified MCU_144.SchDoc
U_USB
Modified USB.SchDoc
U_Ethernet
Modified Ethernet.SchDoc
NRST_MCU
NRST
LTC3624IMSE
VIN
SW
RUN
FB
INTVCC
PGOOD
MODE/SYNC
NC
GND
47uF
LD1
Green
LD39050PU33R
VI
VO
PG
EN
NC
+12V0
+3V3
+3V3A
+5V0
USB_SOF
Note1: Text in Italic placed on a wire doesn't correspond to net name. It just helps to identify rapidly Arduino's signal related to this wire.
2. Add C58 4.7uF ceramic capacitor on VDD from A-01 to B-01
3. R33's value changed to 200Kohm from A-01 to B-01
4. Add pull-up & pull-down resistors on PB2 for BOOT1 (F4 series) from A-01 to B-01
5. All peripherals' power changed to +3V3_PER from A-01 to B-01
6. C36 & C37's value changed to 2pF from A-01 to B-01
7. Add R76 1.5K pull-up circuit to USB_DP for F303ZE only from A-01 to B-01
8. LD1 can be controlled by PB0 from A-01 to B-01
9. D11 on CN7 can be set to PB5 from A-01 to B-01
Title: TOP & POWER
Project: NUCLEO-XXXXZX
Size: A4
Reference: MB1137
Revision: B-01
Date: 2/4/2019
Sheet: 1 of 5

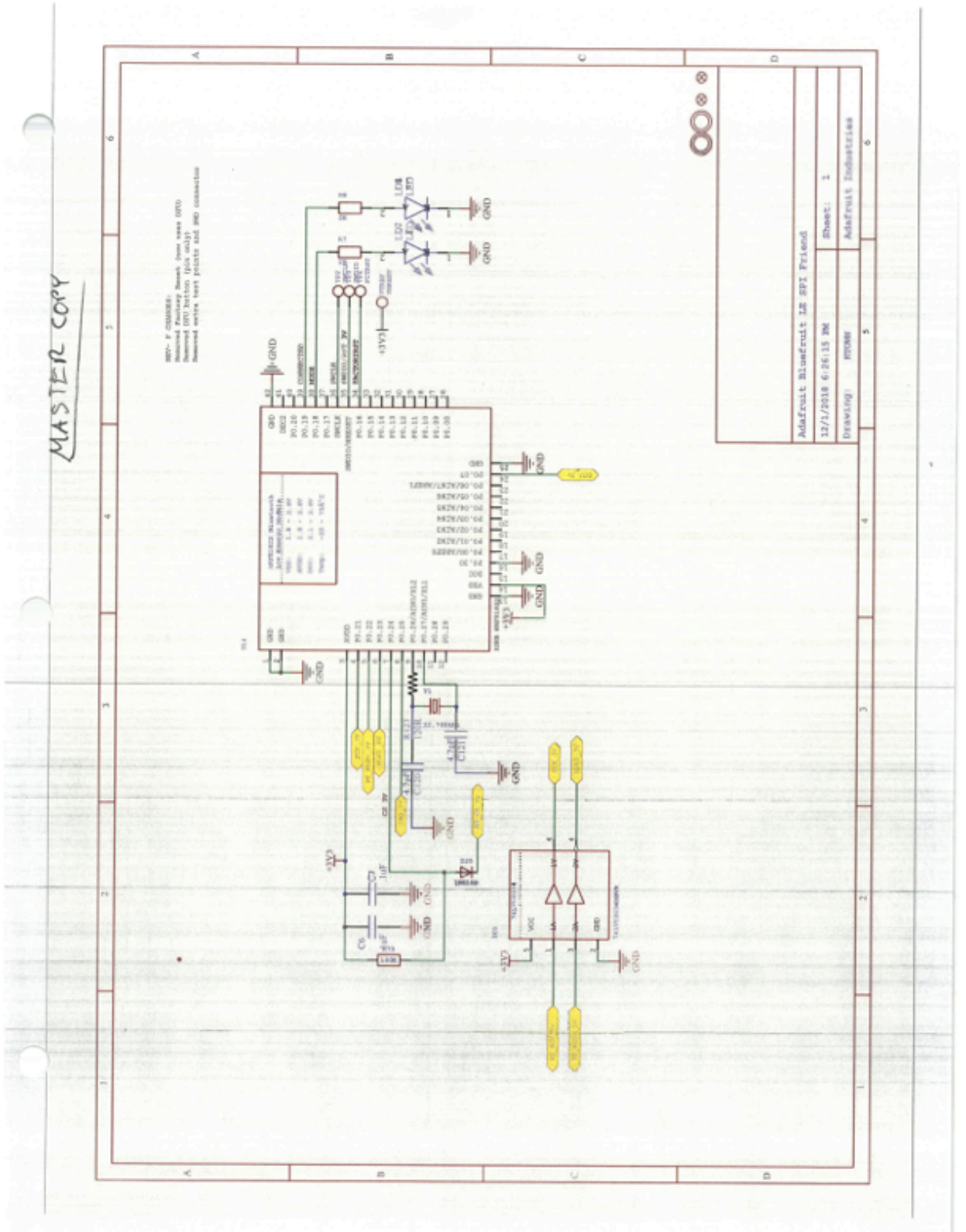

MASTER COPY

SS

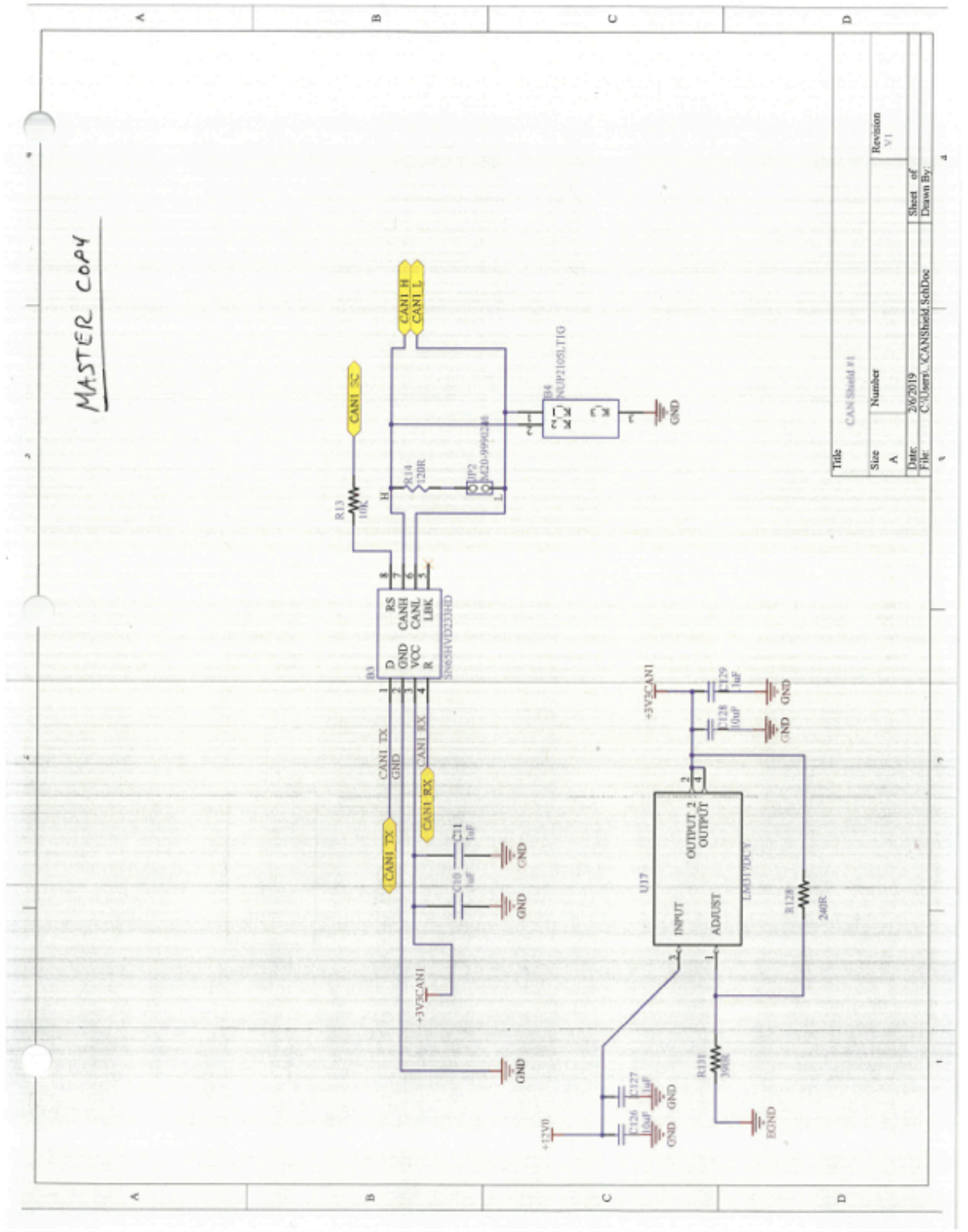

MASTER COPY
LSM9DS1
I2C Address Table
Set the SDO_AG and SDO_M to set the accel/gyro and mag I2C addresses:
SDOM/AG
AG Addr.
M Addr.
0
0x6A
0x1C
1
0x6B
0x1E
Jumpers
VDD
U1
CS_M
CS_A/G
SCL/SPC
SDO_A/G
SDA/SDI/SDO
SDO_M
INT_M
INT1_A/G
INT2_A/G
DRDY_M
C1
CAP
VDD
VDD
VDDIO
VDDIO
DEN_A/G
RES
RES
RES
RES
RES
RES
GND
GND
VDD Range: 1.9-3.6V
GND
GND
GND
GND
GND
Released under the Creative Commons
Attribution Share-Alike 4.0 License
https://creativecommons.org/licenses/by-sa/4.0/
TITLE: SparkFun-LSM9DS1-Breakout_v10
Design by: Jim Lindblom
REV: v10
Date: 12/16/2018 1:50:54 PM
Sheet: 1
sparkfun

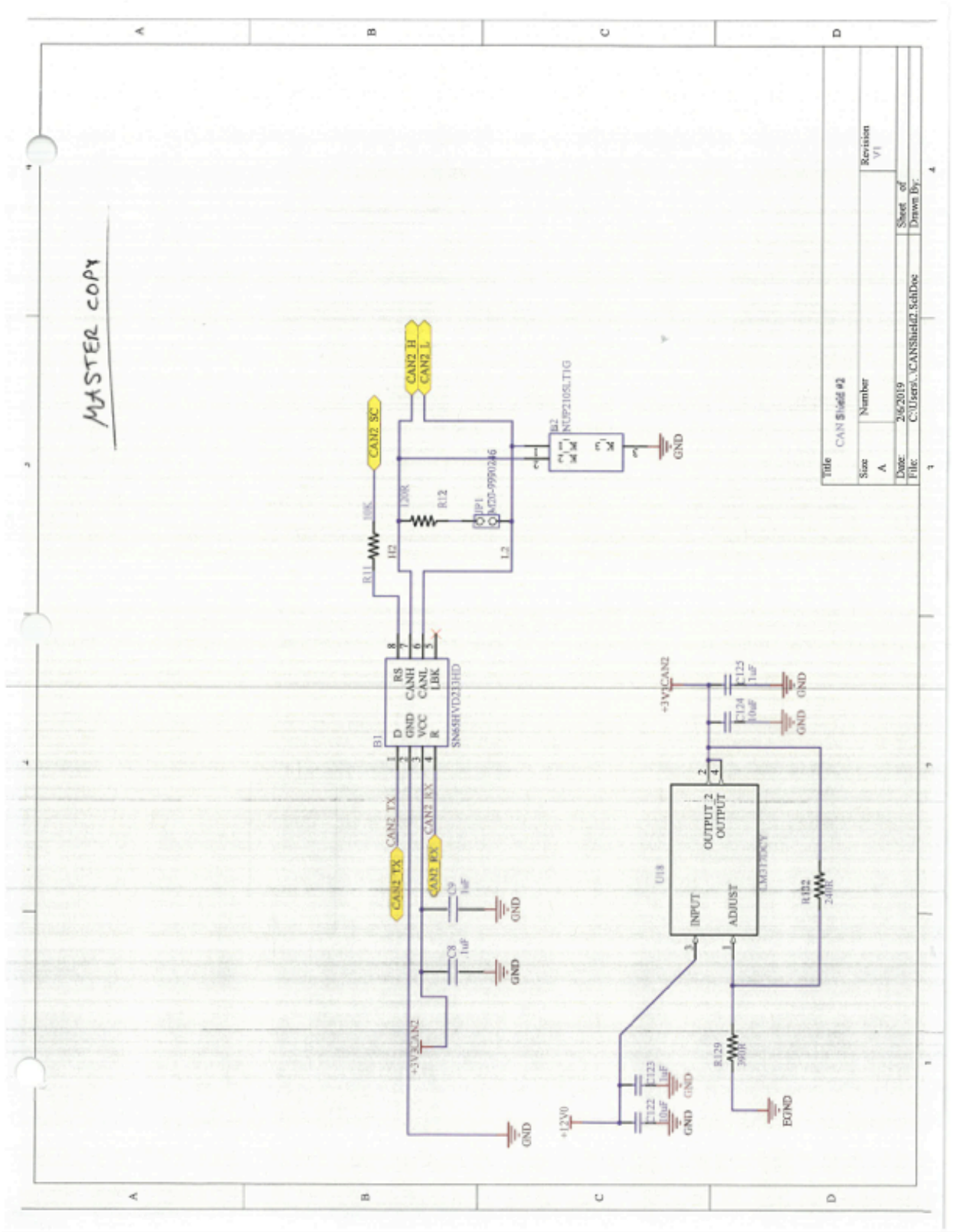
MASTER COPY

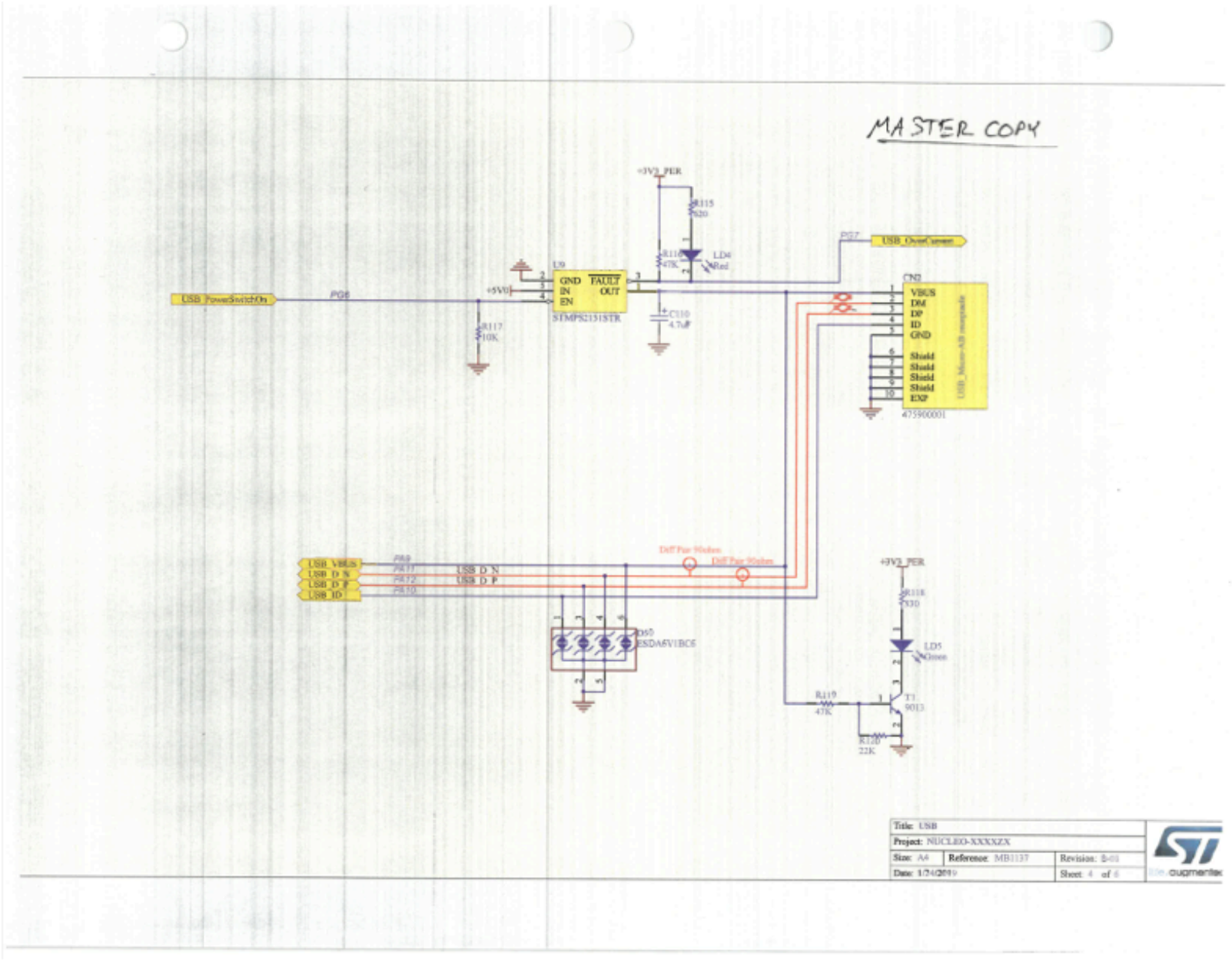

+3V3
C119
.1uF
C118
1uF
SDMMC_D0
SDMMC_D1
SDMMC_D2
SDMMC_CK
SDMMC_CMD
SDMMC_D3
SDMMC_CD
GND
GND
GND
GND
SD_CARD_SOCKET
Released under the Creative Commons
Attribution Share-Alike 4.0 License
https://creativecommons.org/licenses/by-sa/4.0/
TITLE: SD_MMC Breakout-v20
Design by: A. Weiss
Revised by: Patrick Alberts
REV: v21
Date: 11/30/2018 6:14:01 PM
Sheet: 1


MASTER COPY
Adafruit Bluefruit LE SPI Friend
12/1/2018 6:26:15 PM
Sheet: 1
Drawing: FRIEND
AdaFruit Industries

UU

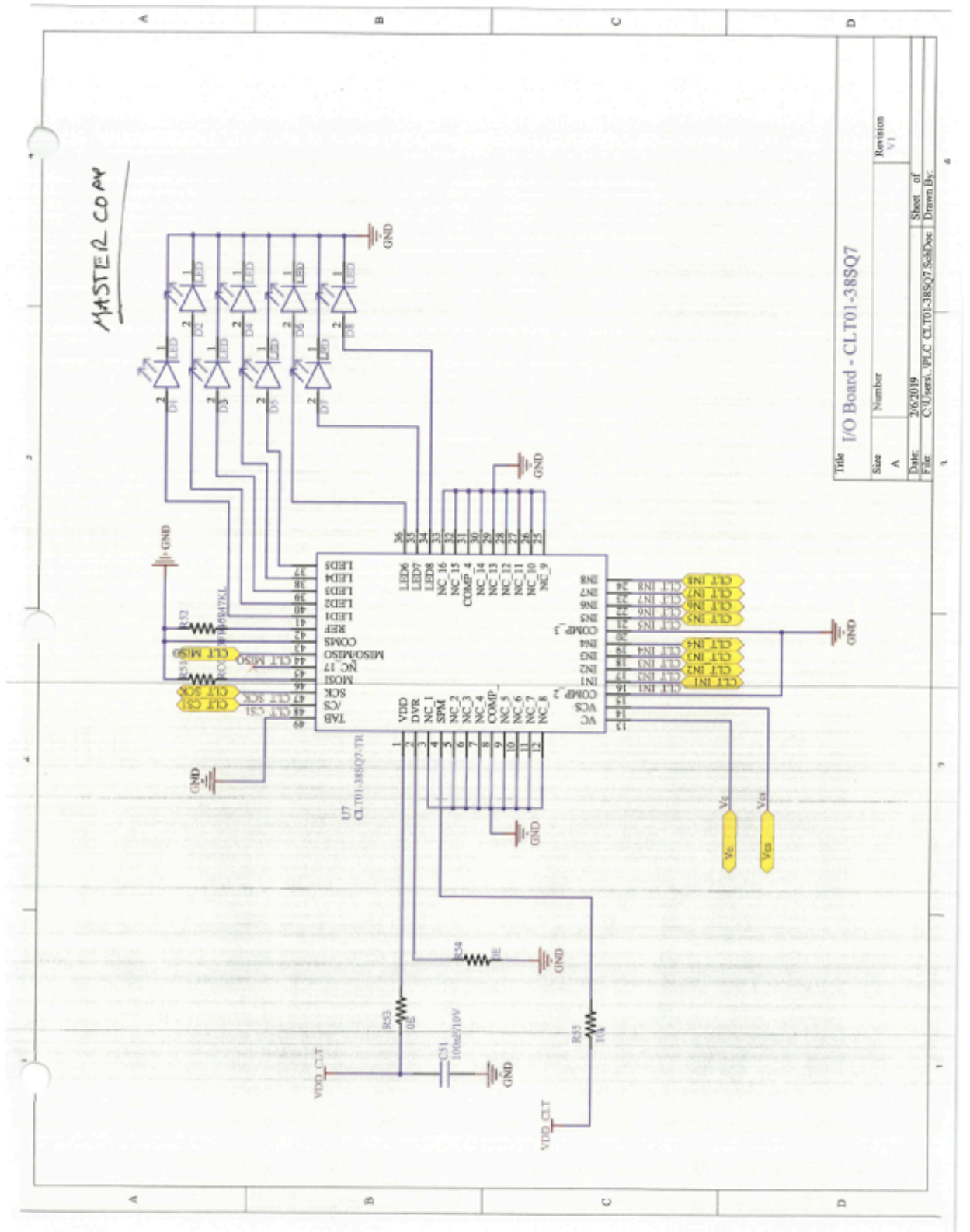

MASTER COPY
CAN1 TX
CAN1 RX
CAN1 H
CAN1 L
+3V3CAN1
+12V0
GND
U17
LM317DCY
INPUT
ADJUST
OUTPUT 2
OUTPUT
SN65HVD233HD
NUP2105LT1G
Title
CAN Shield #1
Size
A
Number
Revision
V1
Date:
2/6/2019
File:
C:\Users\...\CANShield.SchDoc
Sheet of
Drawn By:


MASTER COPY
CAN2_H
CAN2_L
CAN2_SC
R11
10K
U2
R12
120R
JP1
M20-9990246
L2
B2
NUP2105LT1G
GND
B1
SN65HVD23DHD
RS
CANH
CANL
LBK
D
GND
VCC
R
CAN2_TX
CAN2_RX
C9
1uF
C8
1uF
+3V3CAN2
U18
OUTPUT_2
OUTPUT
INPUT
ADJUST
LM317DCY
C125
1uF
C124
10uF
R132
240R
R129
390R
C123
1uF
C122
10uF
+12V0
EGND
Title
CAN Shield #2
Size
A
Number
Revision
V1
Date:
2/6/2019
Sheet of
File:
C:\Users\..\CANShield2.SchDoc
Drawn By:

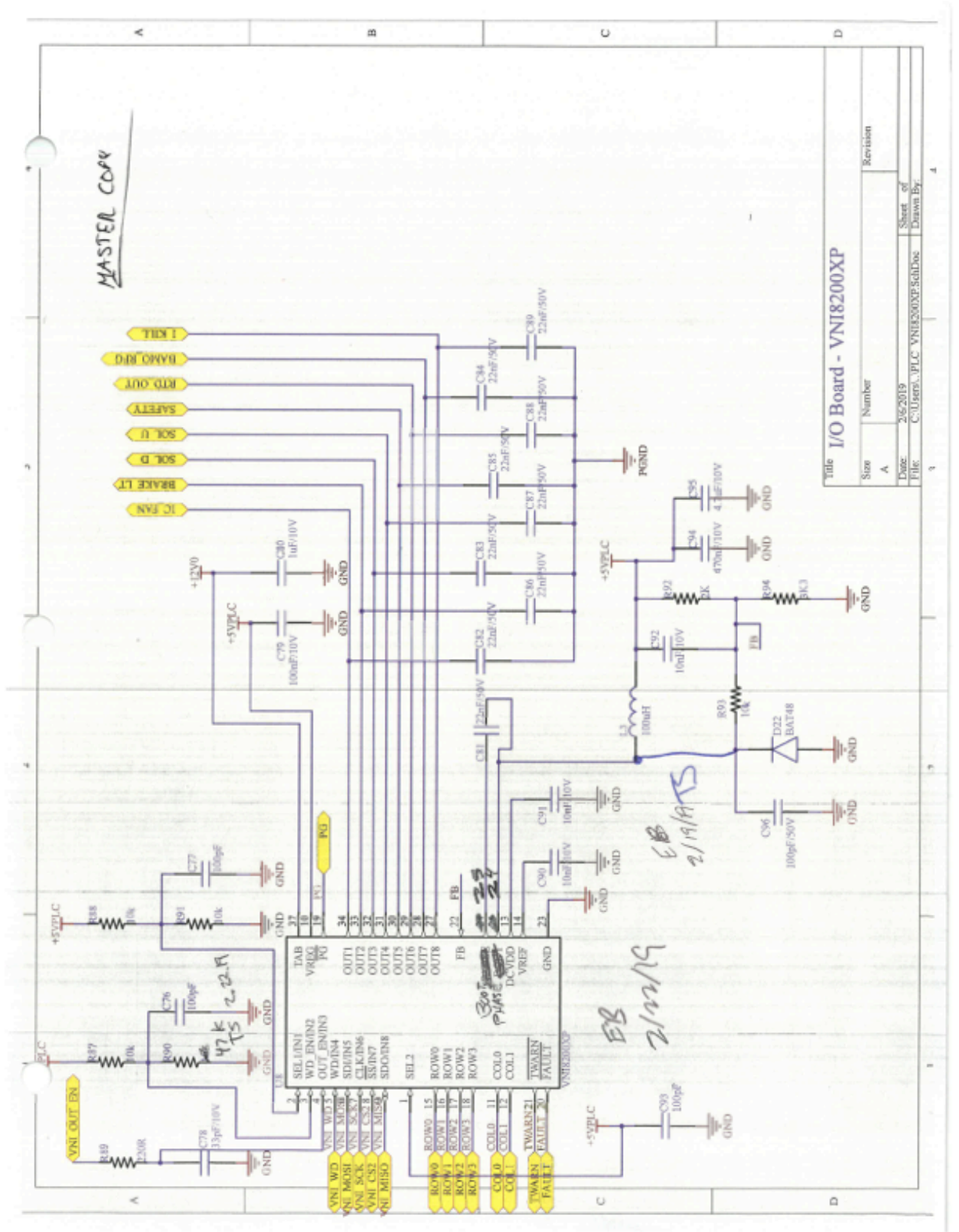

MASTER COPY
+3V3_PER
R115
620
LD4
Red
R116
47K
U9
GND
IN
EN
FAULT
OUT
STMPS2151STR
+5V0
USB_PowerSwitchOn
PG8
PG7
USB_OverCurrent
R117
10K
C110
4.7uF
CN2
VBUS
DM
DP
ID
GND
Shield
Shield
Shield
Shield
EXP
USB_Micro-AB receptacle
475900001
USB_VBUS
USB_D_N
USB_D_P
USB_ID
PA9
PA11
PA12
PA10
USB_D_N
USB_D_P
Diff Pair Modem
Diff Pair Modem
D5
ESDA6V1BC6
+3V3_PER
R118
620
LD5
Green
R119
47K
T1
9013
R120
22K
Title: USB
Project: NUCLEO-XXXXZX
Size: A4
Reference: MB1137
Revision: B-01
Date: 1/24/2019
Sheet: 4 of 6
st.com/stm32nucleo

XX

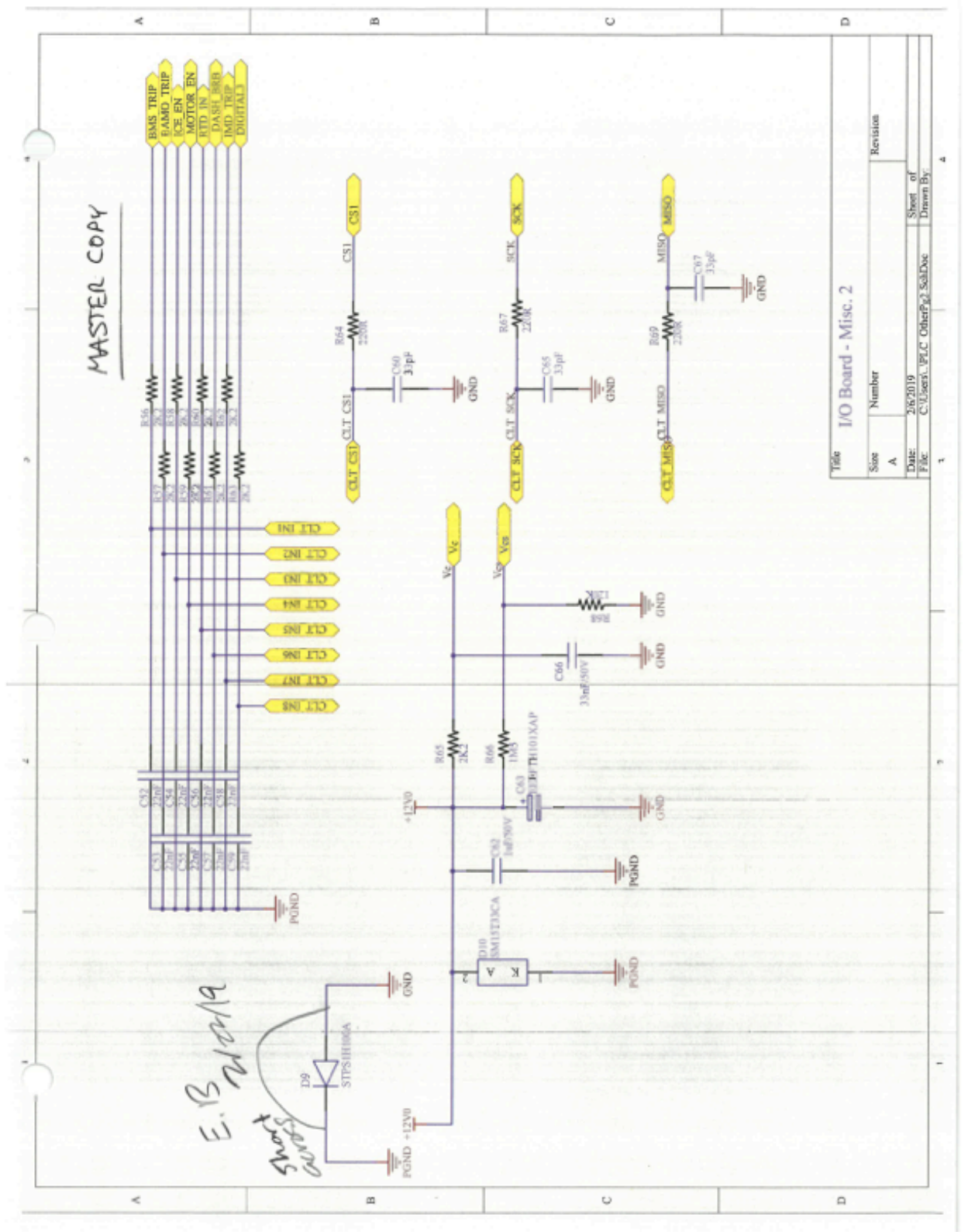
MASTER COPY


MASTER COPY
I/O Board - CLT01-38SQ7
Size
A
Number
Revision
V1
Date:
2/6/2019
Sheet of
File:
C:\Users\..\PLC CLT01-38SQ7.SchDoc
Drawn By:
U7
CLT01-38SQ7-TR
GND
VDD_CLT
R52
R53
R54
R55
C51
100nF/10V
CLT_MISO
CLT_SCK
CLT_CS1
CLT_IN1
CLT_IN2
CLT_IN3
CLT_IN4
CLT_IN5
CLT_IN6
CLT_IN7
CLT_IN8
Vc
Vca

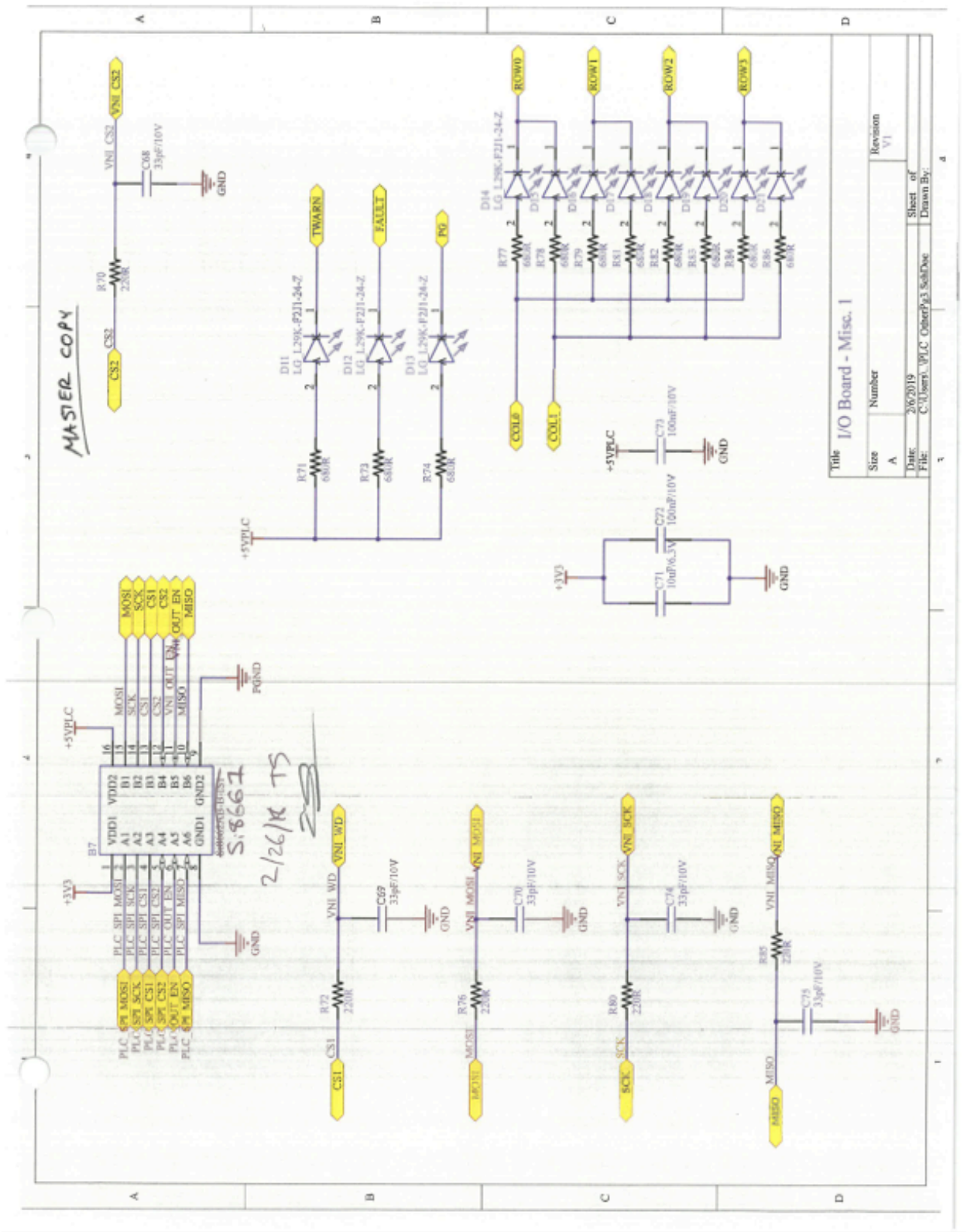
ZZ


MASTER COPY
I/O Board - VNI8200XP
Title
Size
A
Number
Revision
Date: 2/6/2019
File: C:\Users\...\PLC VNI8200XP.SchDoc
Sheet of
Drawn By:
VNI8200XP
+12V0
+5VPLC
PGND
GND
I KILL
BRAKE LT
SOL D
SOL U
SAFETY
RTD OUT
BRAKE RTD
IC FAN
VNI OUT EN
VNI WD
VNI MOSI
VNI SCK
VNI CS2
VNI MISO
ROW0
ROW1
ROW2
ROW3
COL0
COL1
TWARN
FAULT
PLC
PG
FB
SEL1/IN1
WD_EN/IN2
OUT_EN/IN3
WDI/IN4
SDI/IN5
CLK/IN6
SS/IN7
SDO/IN8
SEL2
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
TAB
VREG
VS
DCVDD
VREF
R87 10k
R88 10k
R89 2.49R
R90 10k
R91 10k
R92 2K
R93 10k
R94 8K3
C76 100pF
C77 100pF
C78 33pF/10V
C79 100nF/10V
C80 1uF/10V
C81 22nF/50V
C82 22nF/50V
C83 22nF/50V
C84 22nF/50V
C85 22nF/50V
C86 22nF/50V
C87 22nF/50V
C88 22nF/50V
C89 22nF/50V
C90 10nF/10V
C91 10nF/10V
C92 10nF/10V
C93 100pF
C94 470nF/10V
C95 4.7uF/10V
C96 100pF/50V
L3 100uH
D22 BAT48

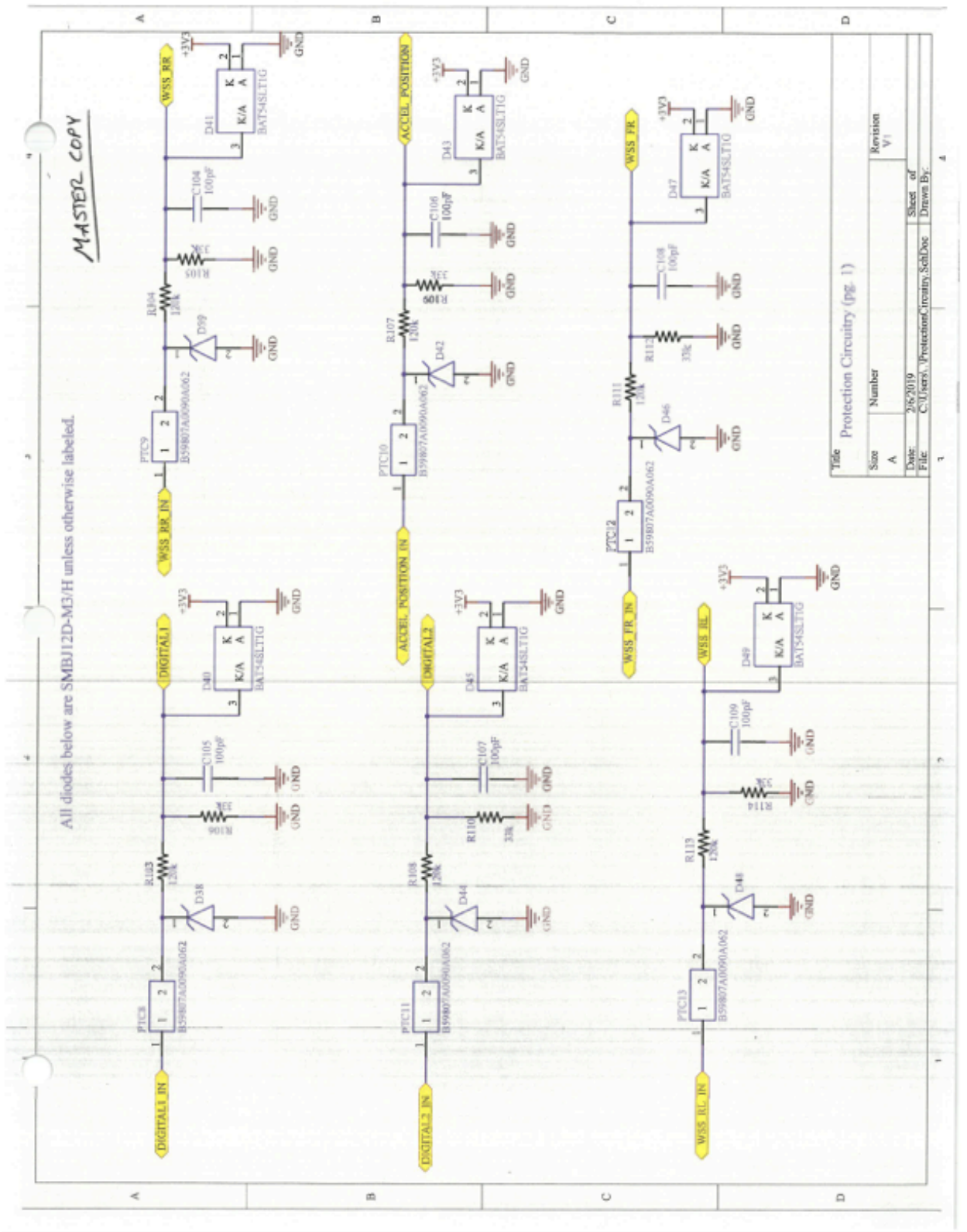
AAA


MASTER COPY
I/O Board - Misc. 2
BMS_TRIP
BAMO_TRIP
ICE_EN
MOTOR_EN
RTD_IN
DASH_BRB
IMD_TRIP
DIGITAL3
CS1
SCK
MISO
CLT_CS1
CLT_SCK
CLT_MISO
CLT_IN1
CLT_IN2
CLT_IN3
CLT_IN4
CLT_IN5
CLT_IN6
CLT_IN7
CLT_IN8
Vc
Vss
+12V0
PGND
GND
STPS1H100A
SM15T33CA
Size
A
Number
Revision
Date: 2/6/2019
File: C:\Users\...\PLC Other\Pcb2.SchDoc
Sheet of
Drawn By:

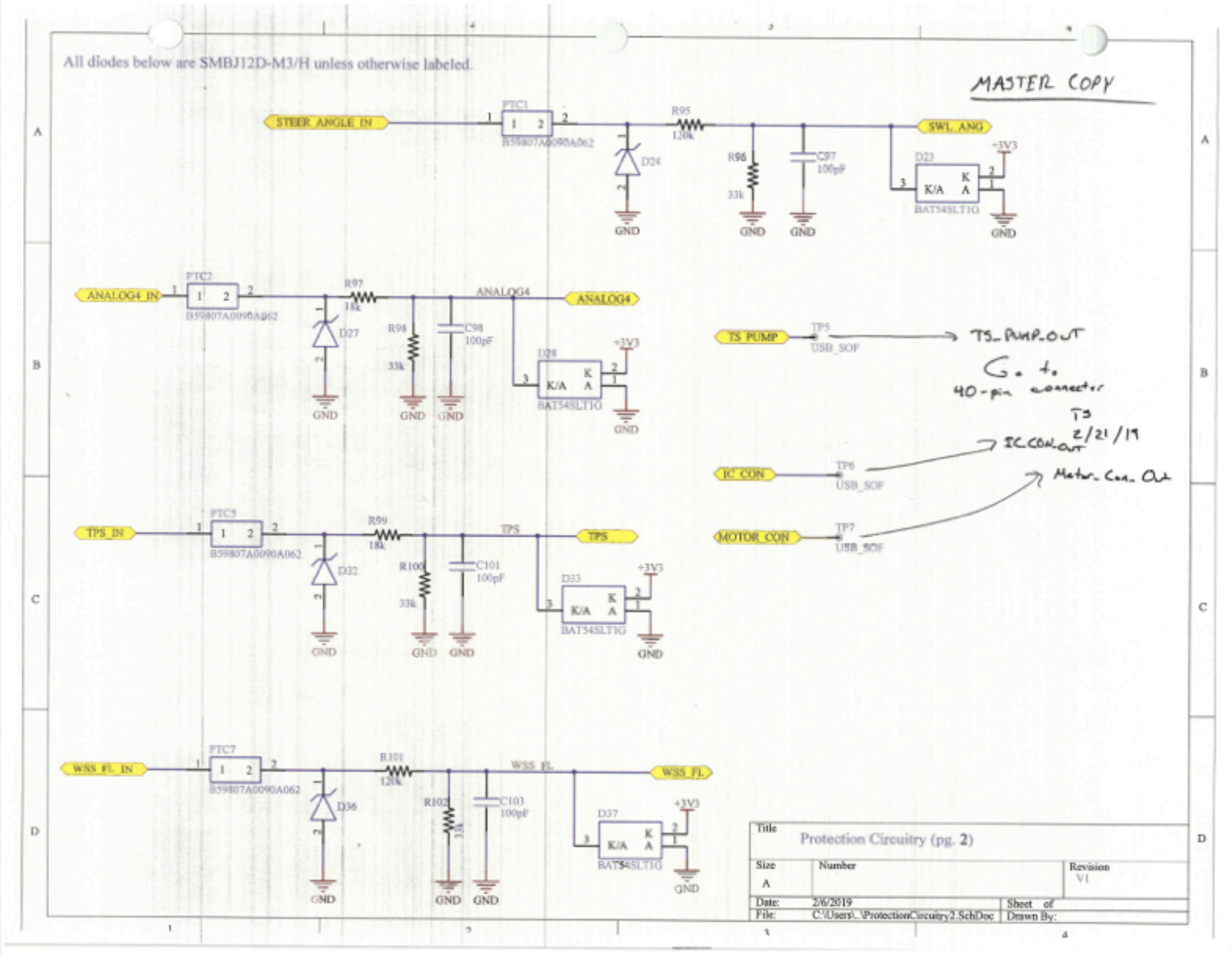
BBB


MASTER COPY
I/O Board - Misc. 1
Si8661
2/26/K TS
+5VPLC
+3V3
PGND
GND
TWARN
FAULT
PG
ROW0
ROW1
ROW2
ROW3
COL0
COL1
VNI_CS2
VNI_WD
VNI_MOSI
VNI_SCK
VNI_MISO
PLC_SPI_MOSI
PLC_SPI_SCK
PLC_SPI_CS1
PLC_SPI_CS2
PLC_OUT_EN
PLC_SPI_MISO

CCC


MASTER COPY
All diodes below are SMBJ12D-M3/H unless otherwise labeled.
Protection Circuitry (pg. 1)
WSS_RR
WSS_RR_IN
DIGITAL1
DIGITAL1_IN
ACCEL_POSITION
ACCEL_POSITION_IN
DIGITAL2
DIGITAL2_IN
WSS_FR
WSS_FR_IN
WSS_RL
WSS_RL_IN
+3V3
GND
BAT54SLT1G
B59807A0090A062

DDD


All diodes below are SMBJ12D-M3/H unless otherwise labeled.
MASTER COPY
STEER_ANGLE_IN
PTC1
B59807A0090A062
R95
120k
SWL_ANG
D24
R96
33k
C97
100pF
D23
BAT54SLT1G
+3V3
GND
ANALOG4_IN
PTC2
R97
18k
ANALOG4
D27
R98
33k
C98
100pF
D28
TS_PUMP
TP5
USB_SOF
TS_PUMP_OUT
Go to
40-pin connector
T3
2/21/19
IC_CON
TP6
IC_CON_OUT
MOTOR_CON
TP7
Motor_Con_Out
TPS_IN
PTC5
R99
18k
TPS
D32
R100
33k
C101
100pF
D33
WSS_FL_IN
PTC7
R101
120k
WSS_FL
D36
R102
33k
C103
100pF
D37
Title
Protection Circuitry (pg. 2)
Size
A
Number
Revision
V1
Date:
2/6/2019
Sheet of
File:
C:\Users\...\ProtectionCircuitry2.SchDoc
Drawn By:

EEE

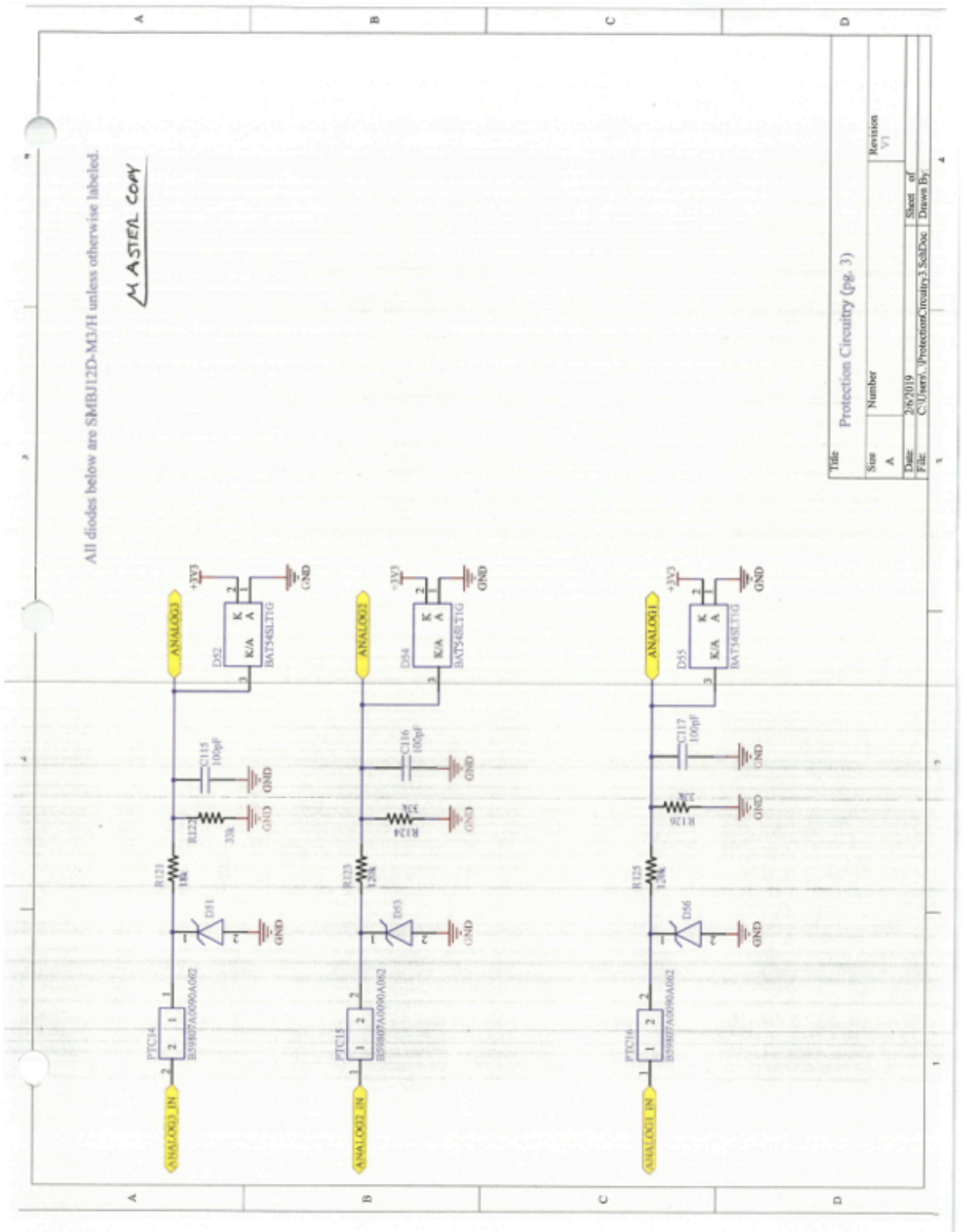


Figure J1: ECO List for Schematic

Appendix K

Datasheet begins on next page.

Central Vehicle Controller Data Sheet

DARTMOUTH FORMULA RACING TEAM

Table of Contents

1. Introduction

The Central Vehicle Controller (CVC) is a printed circuit board (PCB) that serves as the controller for the Dartmouth Formula Racing (DFR) car. It interfaces with the rest of the car through a forty-pin automotive-grade connector. This connector allows for the use of several pre-determined sensors, while providing additional flexibility in the form of multiple analog and digital GPIO pins. These inputs are sent through protective circuitry that prevents overloading of sensitive components on the board, such as the MCU, in the case of a high-voltage short. In addition to the input-specific protection circuitry, hardware from the X-NUCLEO-PLC01A1 board provides an additional layer of protection. In the case that the board gets too hot or the input voltage drops too low, on-board circuitry will cause it to shut down. The board also includes an accelerometer. The data from all these sensors can be stored on an SD card in a .csv format. To download this data, users have three options provided on-board: Bluetooth, Ethernet, and USB On-The-Go.

2. Features

The CVC offers the following features:

- STM32 microcontroller in LQFP144 package with overvoltage-protected GPIOs
- Ethernet compliant with IEEE-802.3-2002 (depending on STM32 support)
- 26 status indicator LEDs
- HSE crystal:
    - 32.768 kHz crystal oscillator
- Board connectors:
    - Ethernet RJ45
    - USB with Micro-AB
    - SparkFun SD/MMC socket
    - TE Deutsch 40-pin connector
    - Molex 8-pin SWD connector
    - TE AMP 6-pin connector
    - TE Deutsch 8-pin debug connector
- SPI communication protocol ready
- CAN communication protocol ready
- LSM9DS1 3D accelerometer, gyroscope, and magnetometer
- Power-supply: 12-volt external supply
- On-board debugger/programmer with SWD connector:
- USB re-enumeration capability: virtual COM port, mass storage, debug port
- Comprehensive free software libraries and examples available with the STM32Cube package
- Supported by wide choice of Integrated Development Environments (IDEs) including IAR™, Keil®, GCC-based IDEs, Arm® Mbed™
- ARM® Mbed Enabled™ (see http://mbed.org)

3. Hardware Layout and Configuration

3.1. Block Diagram

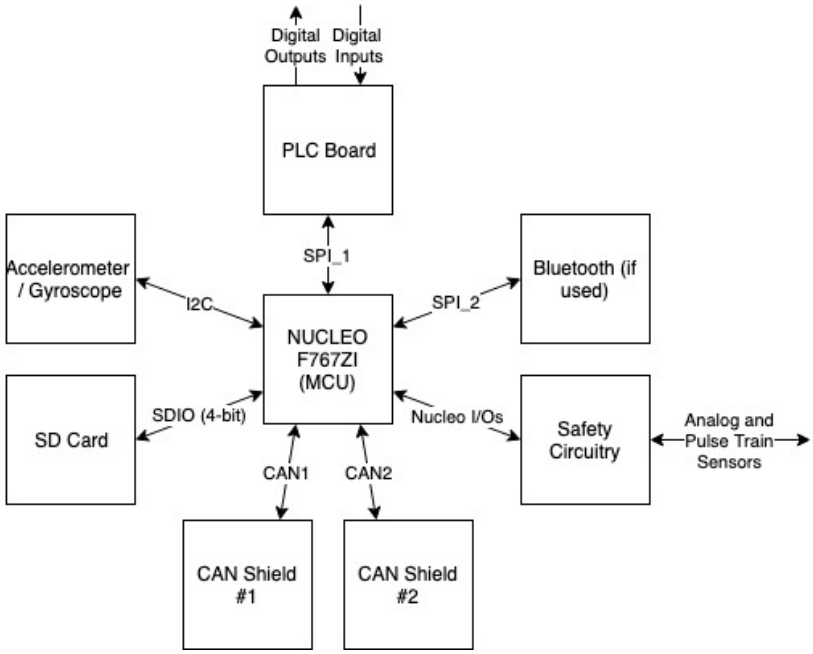


Figure 1: Hardware Block Diagram

3.2. PCB Layout

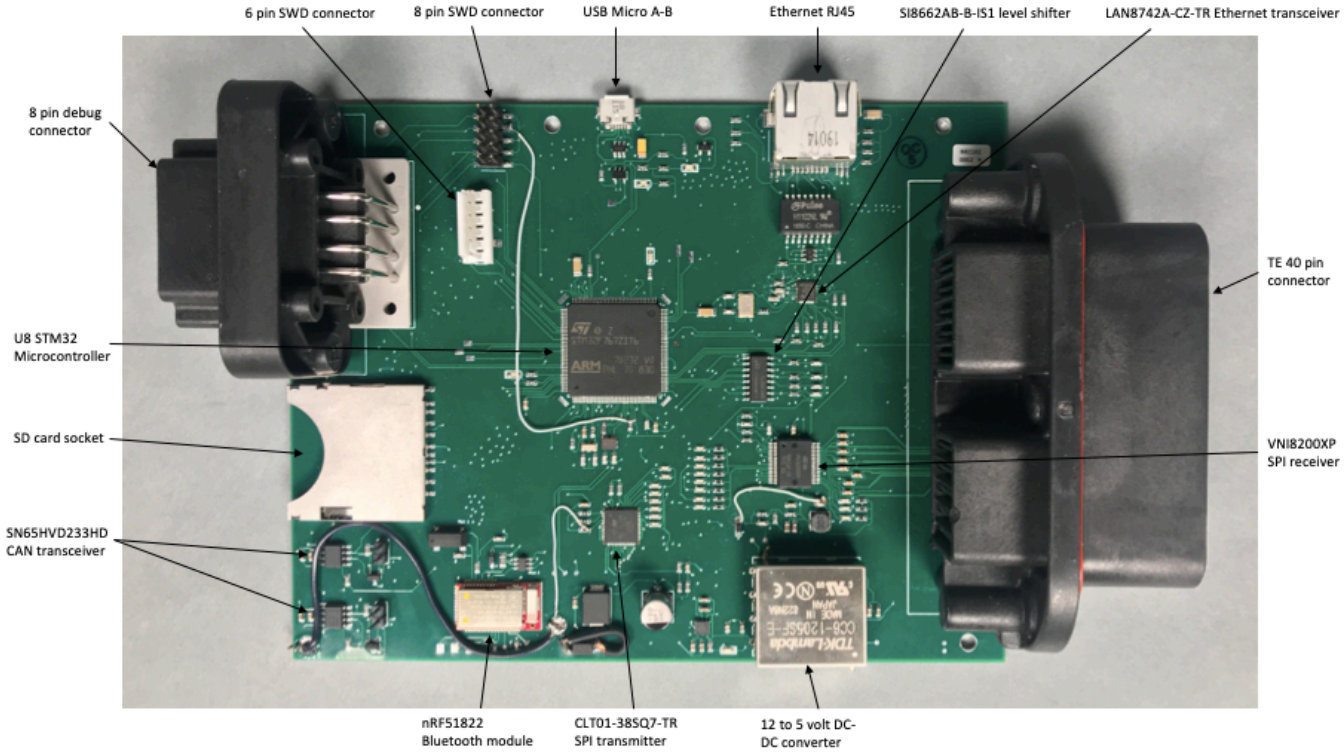


Figure 2: PCB Top Layout

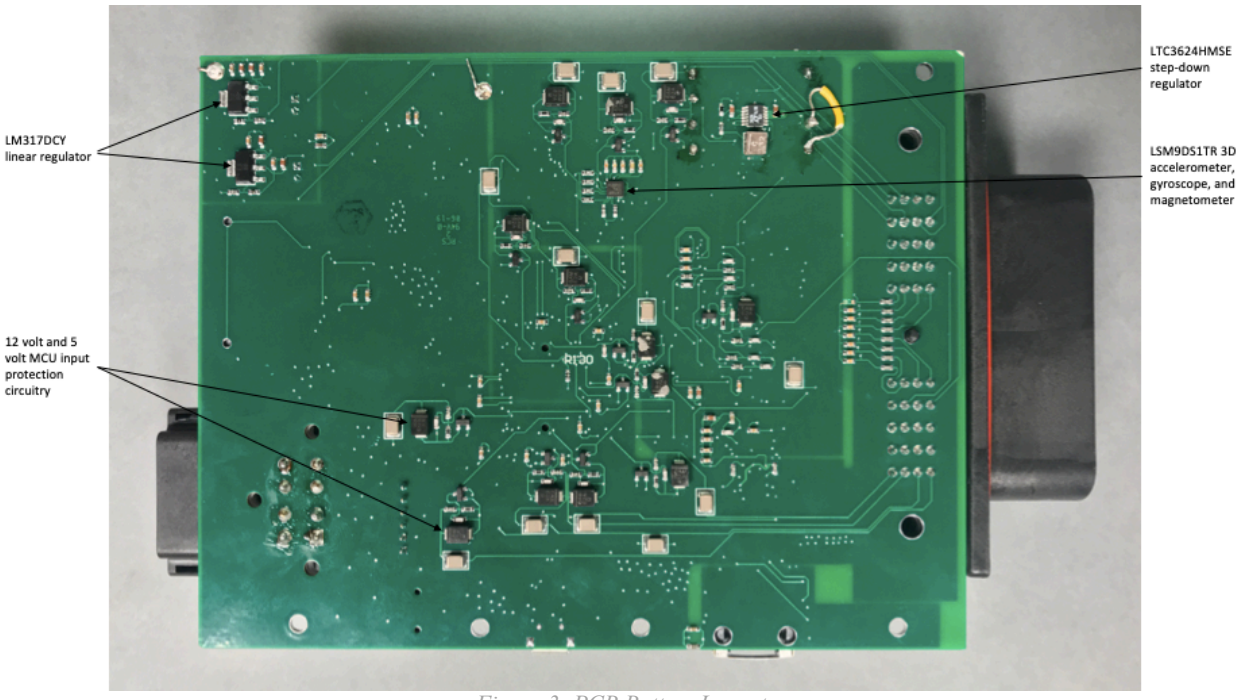


Figure 3: PCB Bottom Layout

3.3. Mechanical Drawing and Mounting Guidelines

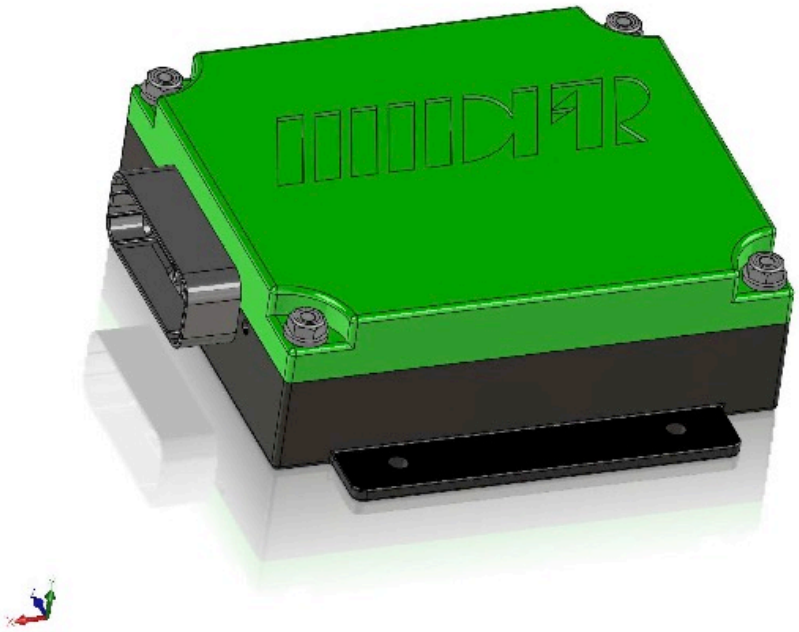


Figure 4: Full Enclosure Model

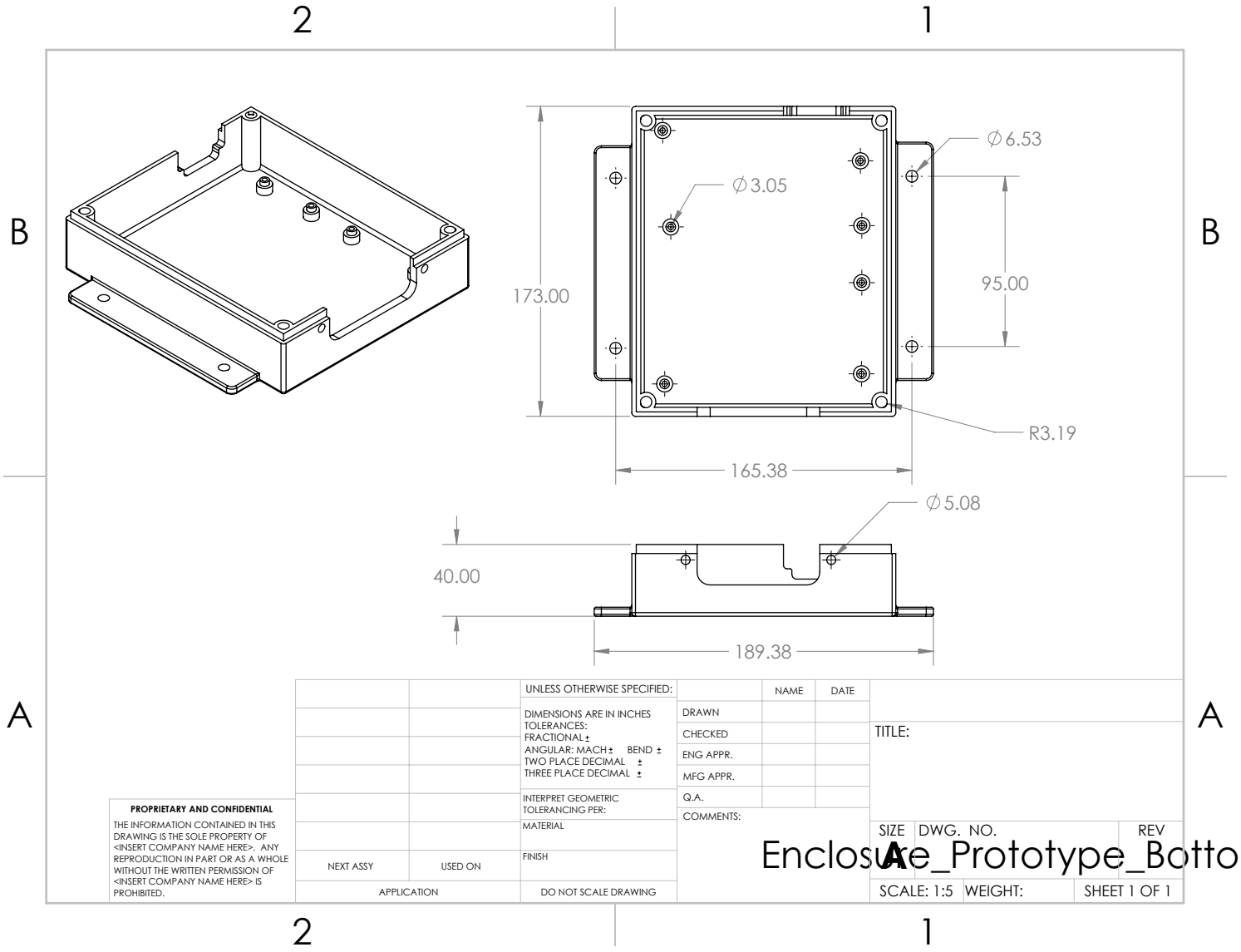


Figure 5: Enclosure Bottom Technical Drawing

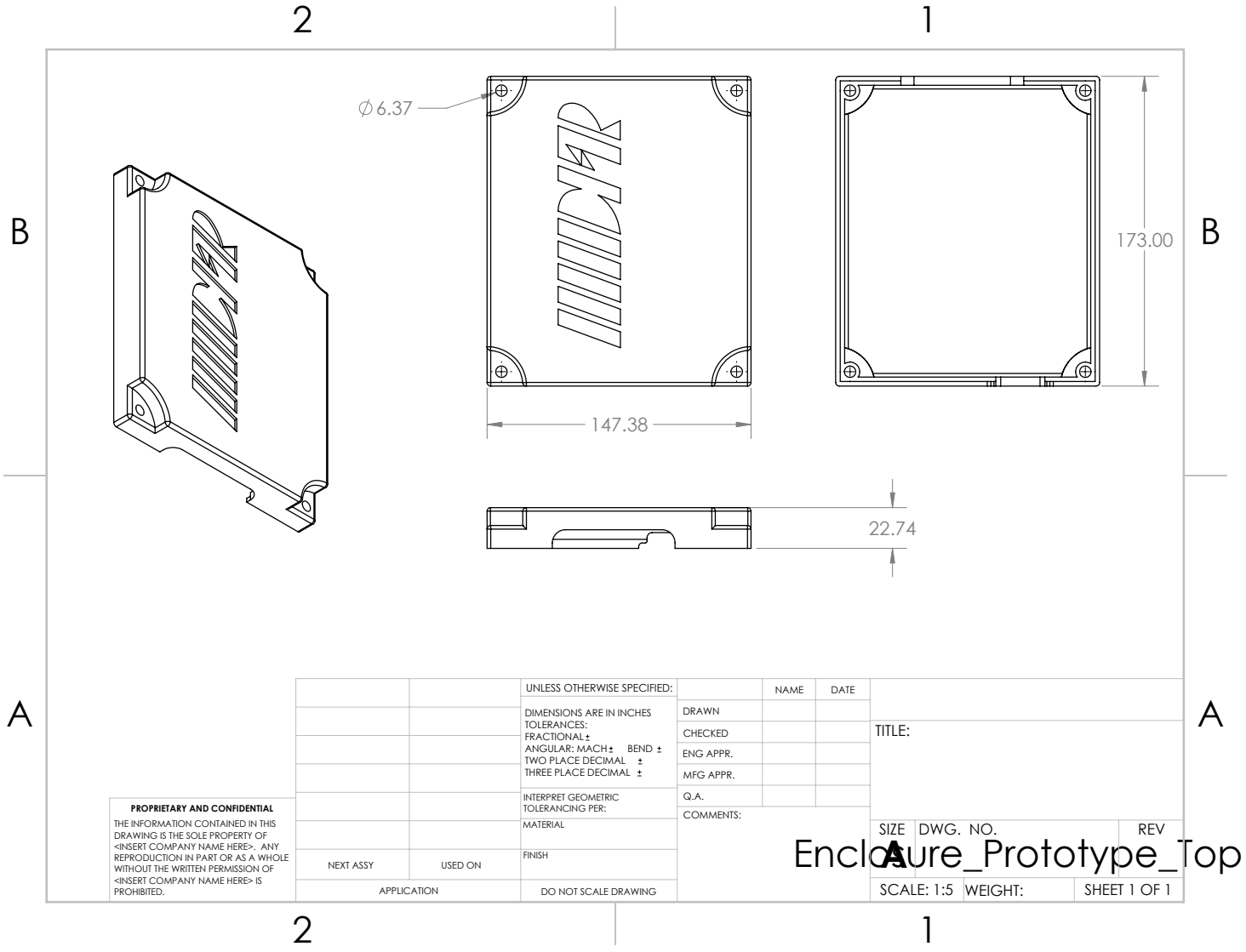


Figure 6: Enclosure Top Technical Drawing

3.4. 40-Pin Connector

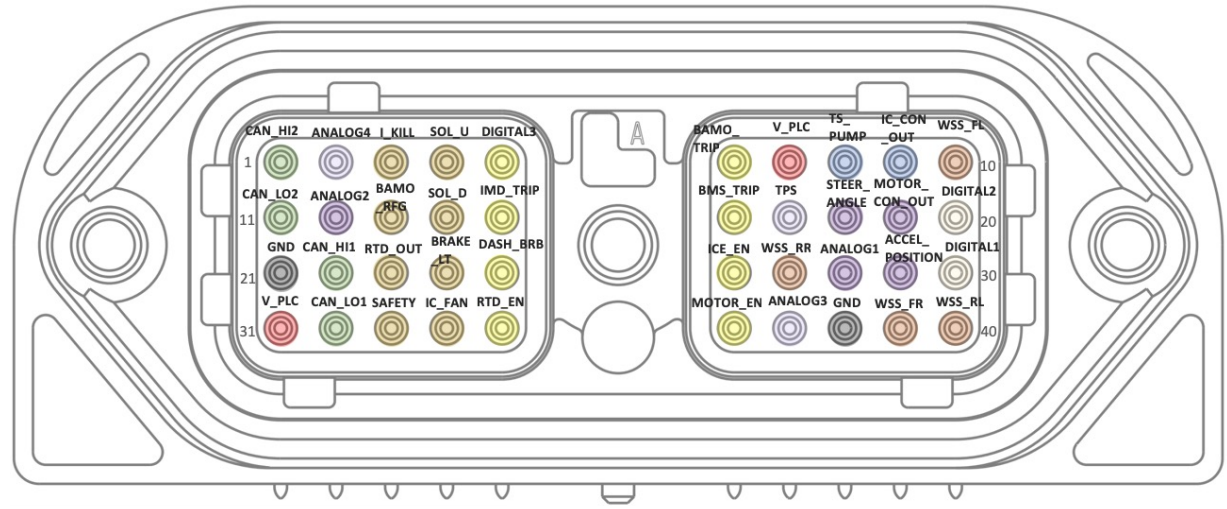


Figure 7: Annotated 40-pin Deutsch Connector

Table 1: Pin Assignments for 40-pin Deutsch Connector

Pin Number	Pin Type	Pin Name	Function
1	timer input capture	WSS_FL	Front-left wheel speed
2	PWM output	IC_CON_OUT	Internal combustion engine throttle control
3	PWM output	TS_PUMP	Tractive system pump
4	power	V_PLC	Programming logic controller board power
5	digital input	BAMO_TRIP	Bamocar safety circuit fault
6	digital input	DIGITAL3	Digital general purpose input output
7	digital output	SOL_D	Downshift solenoid
8	digital output	I_KILL	Ignition kill
9	analog	ANALOG4	Analog general purpose input output
10	CAN	CAN_HI2	CAN high
11	digital input	DIGITAL2	Down-shift
12	analog output	MOTOR_CON_OUT	Motor torque control
13	analog input	STEER_ANGLE	Steering wheel angle
14	analog input	TPS	Engine throttle position
15	digital input	BMS_TRIP	Battery management system safety circuit fault
16	digital input	IMD_TRIP	Insulation monitoring device safety circuit fault
17	digital output	SOL_D	Downshift solenoid
18	digital output	BAMO_RFG	Bamocar RFG
19	analog	ANALOG2	Front master cylinder pressure
20	CAN	CAN_LO2	CAN low
21	digital input	DIGITAL1	Up-shift
22	analog input	ACCEL_POSITION	Accelerator pedal position
23	analog input	ANALOG1	Rear master cylinder pressure
24	timer input capture	WSS_RR	Rear-right wheel speed
25	digital input	ICE_EN	Internal combustion engine enable
26	digital input	DASH_BRB	Dash big red button press
27	digital output	BRAKE_LT	Brake light
28	digital output	RTD_OUT	Ready-to-drive out
29	CAN	CAN_HI1	CAN high
30	ground	GND	Common ground
31	timer input capture	WSS_RL	Rear-left wheel speed
32	timer input capture	WSS_FR	Front-right wheel speed
33	ground	GND	Common ground
34	analog input	ANALOG3	Fuel level
35	digital input	MOTOR_EN	Motor enable
36	digital input	RTD_IN	Ready-to-drive input
37	digital output	IC_FAN	Internal combustion engine fan on/off
38	digital output	SAFETY	Safety pin
39	CAN	CAN_LO1	CAN low
40	power	V_PLC	Programming logic controller board power

3.5. 8-Pin Debug Connector

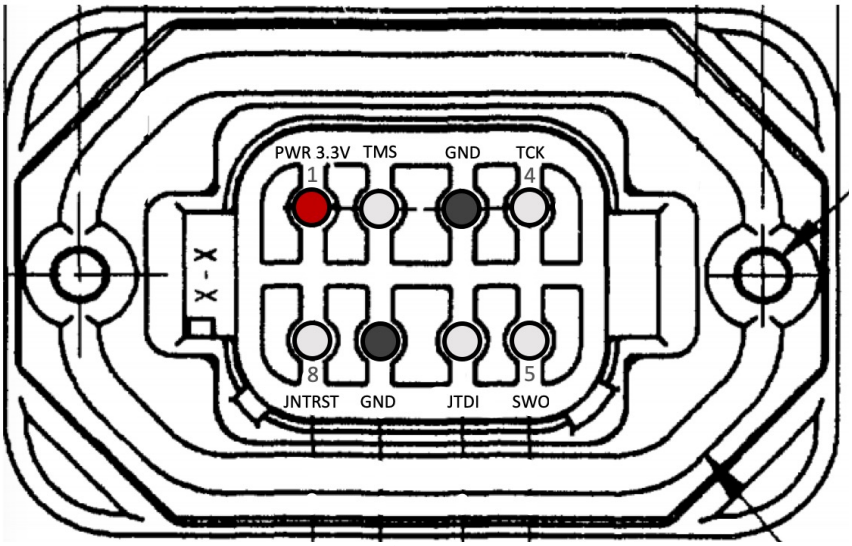


Figure8: Annotated 8-pin Deutsch Connector

Table 2: Pin Assignments for 8-pin Deutsch Connector

Pin Number	Pin Type	Pin Name	Function
1	power	PWR 3.3V	MCU reference voltage
2	serial wire debug I/O	TMS	SW IO (serial wire debug in/out)
3	ground	GND	Common ground
4	clock	TCK	SW CLK (clock into the core)
5	serial-wire output	SWO	SWO (serial-wire output)
6	test data output	JTDI	JTAG TDO (test data output)
7	ground	GND	Common ground
8	reset	JNTRST	JTAG TRST (power up test reset signal)

4. Electrical Characteristics

4.1. Power Supply

The CVC's power is provided by an external source: V_PLC (12.9 V – 16 V), via pin 31 or 7 on the 40-pin Deutsch connector. Ground should be attached to GND on either pin 21 or 38 on the 40-pin Deutsch connector.

The SWD connector can be used to input 3.3 volts to the board and power just the MCU. This allows for flashing code to the board or resetting the MCU without powering the entire system.

4.2. Maximum Minimum Ratings

The CVC operates on a voltage range of 12.9 volts to 16 volts. Higher operating voltages are not recommended as the onboard DC-DC converter and buck converter may start to fail.

The absolute maximum current drawn by the CVC is 1.8 amps, if all PLC inputs and outputs and CAN transceivers are being used. If a current draw greater than this is seen, it is recommended to power off the CVC and inspect components for damage. The CVC will typically draw around 100 to 200 mA of current, under normal operating conditions, depending on the amount of activity.

4.3. Safety Circuitry

Safety circuitry designed to protect the MCU pins from overvoltage and current saturation is placed in front of the MCU pins that are wired directly to the 40-pin connector. This circuitry consists of a TransZorb transient voltage suppression (TVS) diode, a PTC fuse resistor, a Schottky

diode, two resistors forming a voltage divider, and a stabilizing capacitor. The circuit is designed to keep the current at safe input levels, via the PTC resistor vastly increasing in resistance as current increases, in the event of a short to a higher voltage source.

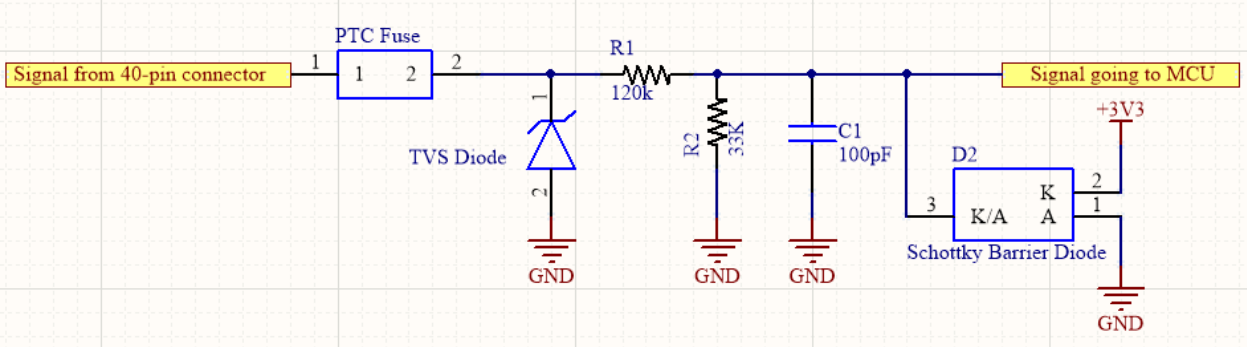


Figure 9: MCU Protection Circuitry

4.4. OSC Clock Supply

HSE on-board oscillator from X3 crystal: The X3 crystal has the following characteristics: 8MHz, 8pF, 20ppm. C37 and C38 must be soldered with 4.3pF capacitors.

HSI internal oscillator: It is possible to provide a clock from the HSI's internal oscillator. This provides a clock signal at the same base frequency as the mounted HSE oscillator with less accuracy (+/- up to 1%).

5. Operating Conditions

5.1. Voltage

The CVC is designed to operate at voltages between 12.9 and 16 volts. The safety circuitry is designed to protect the MCU from shorts with voltage sources of up to 24 volts, although components on the board may still be damaged at this level.

5.2. Temperature

The most sensitive parts on the CVC are rated for -35 to +85 degrees Celsius. The board should be operated within this range of temperatures for optimal performance and to avoid damaging the ICs onboard.

5.3. Drivers

Before connecting the CVC to a Windows 7, Windows 8 or Windows XP PC via USB, a driver for ST-LINK/V2-1 must be installed. It can be downloaded from the www.st.com website.

In case the CVC board is connected to the PC before installing the driver, the PC device manager may report some interfaces as "Unknown".

To recover from this situation, after installing the dedicated driver, the association of "Unknown" USB devices found on the CVC to this dedicated driver, must be updated in the device manager manually.

6. Bill of Materials

Table 3: Bill of Materials for PCB

Description	Part Number	Designator	Quantity on Board	Vendor	Unit Cost
40-Pin Connector	DRC23-40PA-N012	*1	1	Digi-Key	$53.64
High Temperature 3.3 V CAN Transceiver	SN65HVD233HD	B1, B3	2	Digi-Key	$68.8100
Dual Line CAN Bus Protector	NUP2105LT1G	B2, B4	2	Arrow	$0.0844
Silicon Labs PCB SMT 6-Channel Digital Isolator, 2500 Vrms, 16-Pin SOIC	SI8662AB-B-IS1	B7	1	Arrow	$1.9340
0.1µF ±10% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	CC0603KRX7R9BB104	C1, C2, C3	3	Digi-Key	$0.0840
4.7µF Molded Tantalum Capacitors 16V 1206 (3216 Metric) 4 Ohm	TAJA475K016RNJ	C110	1	Arrow	$0.2794
4.7pF ±0.25pF 50V Ceramic Capacitor C0G, NP0 0603 (1608 Metric)	CC0603CRNPO9BN4R7	C120, C121	2	Digi-Key	$0.0610
30pF ±5% 50V Ceramic Capacitor C0G, NP0 0603 (1608 Metric)	CC0603JRNPO9BN300	C13, C23	2	Arrow	$0.0505
2.2µF ±10% 25V Ceramic Capacitor X7S 0603 (1608 Metric)	GRM188C71E225KE11D	C131	1	Digi-Key	$0.1900
470pF ±5% 100V Ceramic Capacitor C0G, NP0 0603 (1608 Metric)	AC0603JRNPO0BN471	C14	1	Digi-Key	$0.1670
1000pF ±10% 2000V (2kV) Ceramic Capacitor X7R 1206 (3216 Metric)	CC1206KKX7RDBB102	C18	1	Digi-Key	$0.2630
Multilayer Ceramic Capacitors MLCC - SMD/SMT 3000V 2pF C0G 1808 10% HI VOLT	1808HA220KAT2A	C29, C30	2	Mouser	$0.5190
4.3pF ±0.5pF 50V Ceramic Capacitor C0G, NP0 0603 (1608 Metric)	C0603C439D5GAC7867	C31, C32	2	Digi-Key	$0.2720
2.2µF ±10% 16V Ceramic Capacitor X7R 1206 (3216 Metric)	CC1206KKX7R7BB225	C34, C35	2	Arrow	$0.2423
10000pF ±10% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	CC0603KRX7R9BB103	C4, C19, C122, C124, C126, C128, C130	7	Digi-Key	$0.0570
10000pF ±10% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	CC0603KRX7R9BB103	C5	1	Digi-Key	$0.0570
4.7µF ±10% 10V Ceramic Capacitor X7S 0603 (1608 Metric)	C1608X7S1A475K080AC	C50,C95	2	Digi-Key	$0.2070
1µF ±20% 16V Ceramic Capacitor X7R 0603 (1608 Metric)	CL10B105MO8NNWC	C6, C9, C11, C20, C24, C25, C36, C112, C113, C118, C132, C133	12	Digi-Key	$0.0420
1µF ±10% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	CC0603KRX7R9BB105	C62	1	Mouser	$0.5720
100µF 50V Aluminum Electrolytic Capacitors Radial, Can - SMD 340 mOhm @ 100kHz 2000 Hrs @ 105°C	EEE-FTH101XAP	C63	1	Digi-Key	$0.5630
0.033µF ±5% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	06035C333JAT2A	C66	1	Digi-Key	$0.1520
33pF ±5% 50V Ceramic Capacitor C0G, NP0 0603 (1608 Metric)	C0603C330J5GACTU	C68, C69, C70, C74, C75, C78,'C60, C65, C67	9	Digi-Key	$0.1370
0.1µF ±10% 16V Ceramic Capacitor X7R 0603 (1608 Metric)	CC0603KRX7R7BB104	C7, C8, C10, C111, C114, C119, C123, C125, C127, C129,'C12, C15, C16, C17, C21, C22, C26, C27, C37, C38, C39, C40, C41, C42, C43, C44, C45, C46, C47, C48, C49'C51, C72, C73, C79	35	Digi-Key	$0.0395
10000pF ±10% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	CC0603KRX7R9BB103	C71	1	Digi-Key	$0.0570
1µF ±10% 10V Ceramic Capacitor X7R 0603 (1608 Metric)	CC0603KRX7R6BB105	C80	1	Digi-Key	$0.1430
0.022µF ±10% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	C0603C223K5RACTU	C81, C82, C83, C84, C85, C86, C87, C88, C89,C52, C53, C54, C55, C56, C57, C58, C59	17	Digi-Key	$0.0730
10000pF ±10% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	CC0603KRX7R9BB103	C90, C91, C92	3	Digi-Key	$0.0570
0.047µF ±10% 10V Ceramic Capacitor X7R 0603 (1608 Metric)	C0603C473K8RAC7867	C94	1	Digi-Key	$0.2330
100pF ±5% 50V Ceramic Capacitor C0G, NP0 0603 (1608 Metric)	CC0603JRNPO9BN101	C96,C76, C77, C93, C97, C98, C101, C103, C104, C105, C106, C107, C108, C109, C115, C116, C117	17	Digi-Key	$0.0610
Jack Modular Connector 8p8c (RJ45, Ethernet) 90° Angle (Right) Shielded, EMI Finger Cat5	6339191-1	CN1	1	Arrow	$4.9187
USB on the go Connector	47590-0001	CN2	1	Digi-Key	$0.6820
Connector Header Through Hole 6 position 0.100" (2.54mm) for USART to USB	640456-6	CN3	1	Digi-Key	$0.2450
Green 570nm LED Indication - Discrete 1.7V 0603 (1608 Metric)	LG L29K-G2J1-24	D1, D2, D3, D4, D5, D6, D7, D8	8	Mouser	$0.4000
TVS Diode Bi-Directional SM15T33CA 59V, 1500W, SMC 2-Pin	SM15T33CA	D10	1	Digi-Key	$0.6480
Green 570nm LED Indication - Discrete 1.7V 0603 (1608 Metric)	LG L29K-G2J1-24	D11, D12, D13, D14, D15, D16, D17, D18, D19, D20, D21	11	Mouser	$0.3290
Schottky Diode, 0.35A max, 40V, 2-Pin, DO-35	BAT48JFILM	D22	1	Digi-Key	$0.3140
Schottky Diode Array, 0.2A, 30V, 3-Pin SOT-23	SBAT54SLT1G	D23, D28, D33, D37, D40, D41, D43, D45, D47, D49, D52, D54, D55	13	Digi-Key	$0.3020
TranZorb Diode 12V 19.6V	SMBJ12D-M3/H	D24, D27, D32, D36, D38, D39, D42, D44, D46, D48, D51, D53, D56	13	Digi-Key	$0.2744
Diode Schottky 10V 3A (DC) Surface Mount SOD-323	BAT60JFILM	D25	1	Digi-Key	$0.2980
TVS Diode	ESDA6V1BC6	D50	1	Digi-Key	$0.3240
Diode Schottky 100V 1A 2-pin Surface Mount SMA (DO-214AC)	STPS1H100A	D9	1	Digi-Key	$0.3650
DC/DC Converter 1 Output 5V 1.2A 9V - 18V Input	CC6-1205SF-E	DC-DC1	1	Digi-Key	$15.9600
Buffer, Non-Inverting 2 Element 1 Bit per Element Push-Pull Output SOT-23-6	SN74LVC2G34DBVR	IC1	1	Digi-Key	$0.4670
8-Pin Connector	DT15-08PB	J1	1	Digi-Key	$8.9200
Connector Header Surface Mount 10 position 0.100" (2.54mm)	M20-8750542	J2	1	Digi-Key	$0.6900

Connector Header Through Hole 2 position 0.100" (2.54mm)	M20-9990246	JP1, JP2	2	Digi-Key	$0.0990
Bead Inductor	FCM1608KF-601T05	L1, L2	2	Sierra	$1.9200
100µH Semi-Shielded Wirewound Inductor 420mA 1.596 Ohm Max Nonstandard	SRN4018-101M	L3	1	Digi-Key	$0.4030
Fixed Inductors 3.3uH 20% 6.6A 28.6mOhms AEC-Q200	XAL4030-332MEB	L4	1	Mouser	$1.8000
Yellow-Green 573nm LED Indication - Discrete 2V 0603 (1608 Metric)	19-21SYGC/S530-E2/TR8	LD1, LD5	2	Digi-Key	$0.1832
Blue LED Indication - Discrete 2.8V 0603 (1608 Metric)	EAST16084BA8	LD2	1	Digi-Key	$0.2280
Red LED Indication - Discrete 1.95V 0603 (1608 Metric)	19-217/R6C-AL1M2VY/3T	LD3, LD4	2	Digi-Key	$0.1680
Green 570nm LED Indication - Discrete 1.7V 0603 (1608 Metric)	LG L29K-G2J1-24-Z	LD7, LD8	2	Digi-Key	$0.3292
Ceramic PTC Thermistor 265V 15mA (3225 Metric)	B59807A0090A062	PTC1, PTC2, PTC5, PTC7, PTC8, PTC9, PTC10, PTC11, PTC12, PTC13, PTC14, PTC15, PTC16	13	Digi-Key	$0.9820
10 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-0710KL	R1, R2, R3, R4, R5, R6, R11, R13, R17, R18, R19, R24, R32, R35, R40, R50, R55, R87, R88, R90, R91, R93, R117	23	Digi-Key	$0.0097
620 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-07620RL	R115	1	Arrow	$0.0016
47 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-0747KL	R116, R119	2	Digi-Key	$0.0240
330 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-07330RL	R118	1	Arrow	$0.0018
120 Ohms ±0.1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thin Film	ERA-3AEB121V	R12, R14, R37, R49, R127	5	Arrow	$0.1975
22 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-0722KL	R120	1	Digi-Key	$0.0240
240 Ohms ±5% 0.25W, 1/4W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200, Pulse Withstanding Thick Film	ESR03EZPJ241	R128	1	Arrow	$0.0674
390 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200, Moisture Resistant Thick Film	CRGCQ0603F390R	R129, R131	2	Arrow	$0.0030
0 Ohms Jumper 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thick Film	RMCF0603ZT0R00	R130	1	Digi-Key	$0.0160
240 Ohms ±5% 0.25W, 1/4W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200, Pulse Withstanding Thick Film	ESR03EZPJ241	R132	1	Arrow	$0.0674
619 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-07619KL	R133	1	Arrow	$0.0015
137 kOhms ±0.1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thin Film	ERA-3AEB1373V	R134	1	Arrow	$0.2366
1 MOhms ±0.1% 0.063W, 1/16W Chip Resistor 0603 (1608 Metric) Thin Film	CPF0603B1M0E	R135, R136	2	Arrow	$0.1154
49.9 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-0749R9L	R15, R16, R20, R21	4	Digi-Key	$0.0240
33 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-0733RL	R22, R23, R30, R33	4	Arrow	$0.0016
75 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-0775RL	R25, R26, R27, R28	4	Arrow	$0.0017
12.1 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-0712K1L	R29	1	Arrow	$0.0015
1.5 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-071K5L	R31	1	Digi-Key	$0.0240
270 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-07270RL	R34, R36	2	Digi-Key	$0.0240
1 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-071KL	R38, R41	2	Arrow	$0.0014
680 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-07680RL	R39	1	Arrow	$0.0014
0 Ohms Jumper 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603JR-070RL	R44, R45	2	Digi-Key	$0.0152
200 Ohms ±0.5% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Thin Film	RT0603DRD07200RL	R48	1	Digi-Key	$0.1740
47 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-0747KL	R51	1	Digi-Key	$0.0240
15 kOhms ±0.1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thin Film	ERA-3AEB153V	R52	1	Arrow	$0.2821
0 Ohms Jumper 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thick Film	RMCF0603ZT0R00	R53, R54	2	Digi-Key	$0.0116
2.2 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-072K2L	R56, R57, R58, R59, R60, R61, R62, R65'R63	9	Digi-Key	$0.0240
220 Ohms ±5% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200, Moisture Resistant Thick Film	CRGCQ0603J220R	R64, R67, R69, R70, R72, R76, R80, R85, R89	9	Arrow	$0.0023

1.5 MOhms ±5% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thick Film	ERJ-3GEYJ155V	R66	1	Digi-Key	$0.0430
120 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200, Moisture Resistant Thick Film	AC0603FR-07120KL	R68, R95, R101, R103, R104, R107, R108, R111, R113, R123, R125	11	Arrow	$0.0016
2 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moisture Resistant Thick Film	RC0603FR-072KL	R7, R8, R92	3	Digi-Key	$0.0240
680 Ohms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thick Film	ERJ-3EKF6800V	R71, R73, R74, R77, R78, R79, R81, R82, R83, R84, R86	11	Digi-Key	$0.0650
100 kOhms ±0.1% 0.125W, 1/8W Chip Resistor 0805 (2012 Metric) Automotive AEC-Q200 Thin Film	ERA-6AEB104V	R75	1	Digi-Key	$0.3020
3.3 kOhms ±5% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200, Moisture Resistant Thick Film	CRGCQ0603J3K3	R94	1	Arrow	$0.0034
33 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thick Film	ERJ-3EKF3302V	R96, R98, R100, R102, R105, R106, R109, R110, R112, R114, R122, R124, R126	13	Arrow	$0.0055
18 kOhms ±5% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Automotive AEC-Q200 Thick Film	RMCF0603JT18K0	R97, R99, R121	3	Arrow	$0.0010
Bipolar (BJT) Transistor NPN 25V 500mA 100MHz 225mW Surface Mount SOT-23-3 (TO-236)	BC818-40LT1G	T1	1	Arrow	$0.1216
Accelerometer, Gyroscope, Magnetometer, Temperature, 3 Axis Sensor I²C, SPI Output	LSM9DS1TR	U1	1	Arrow	$5.6250
SD Memory Card Connector, push-push normal type, CD enable, WP enable	PRT-12769	U13	1	Sparkfun	$1.9500
nRF51822 Bluetooth Low Energy Module	MDBT40-256RV3	U14	1	Seed	$7.5000
Linear Voltage Regulators 3 T 1.5A Adj Pos V	LM317DCY	U17, U18	2	Arrow	$0.4999
Buck Switching Regulator IC Positive Adjustable 0.6V 1 Output 2A 12-TSSOP (0.118", 3.00mm Width) Exposed Pad	LTC3624HMSE#PBF	U19	1	Arrow	$5.0000
TVS DIODE 5.25V 17V SOT23-6	USBLC6-4SC6	U2	1	Arrow	$0.3781
Audio Transformers / Signal Transformers 100BaseTX SMD NonPoE 350uH .65Ohms 1-Port	H1102NL	U3	1	Arrow	$1.5512
1/1 Transceiver Full Ethernet 24-SQFN (4x4)	LAN8742AI-CZ-TR	U4	1	Arrow	$1.4363
Linear Voltage Regulator IC Positive Fixed 1 Output 500mA 6-DFN (3x3)	LD39050PU33R	U5	1	Arrow	$0.8481
ARM® Cortex®-M7 STM32F7 Microcontroller IC 32-Bit 216MHz 2MB (2M x 8) FLASH 144-LQFP (20x20)	STM32F767ZIT6	U6	1	Digi-Key	$15.9800
8-Line Digital Current Limiter SPI QFN48, 8 Channel Protector, 15 35 V, 48-Pin	CLT01-38SQ7-TR	U7	1	Digi-Key	$2.3930
Octal high side smart power solid state relay with serial/parallel selectable interface on chip	VNI8200XP	U8	1	Verical	$6.0490
IC PWR SWITCH 1CH 500MA SOT23-5	STMPS2151STR	U9	1	Future Electronics	$0.2400
25MHz ±25ppm Crystal 12pF 30 Ohms 4-SMD, No Lead	ECS-250-12-30BQ-AEN-TR	X1	1	Digi-Key	$0.7590
32.768kHz ±20ppm Crystal 12.5pF 70 kOhms 2-SMD, No Lead	NX3215SA-32.768KHZ-EXS00A-MU00525	X2	1	Digi-Key	$0.6020
8MHz ±50ppm Crystal 8pF 500 Ohms 2-SMD, No Lead	NX3225GD-8MHZ-STD-CRA-3	X3	1	Digi-Key	$0.8350
32.768kHz ±20ppm Crystal 12.5pF 50 kOhms 4-SOJ, 5.50mm pitch	ABS25-32.768KHZ-T	Y1	1	Arrow	$0.3405
				Total	$330.41

7. PCB Schematics

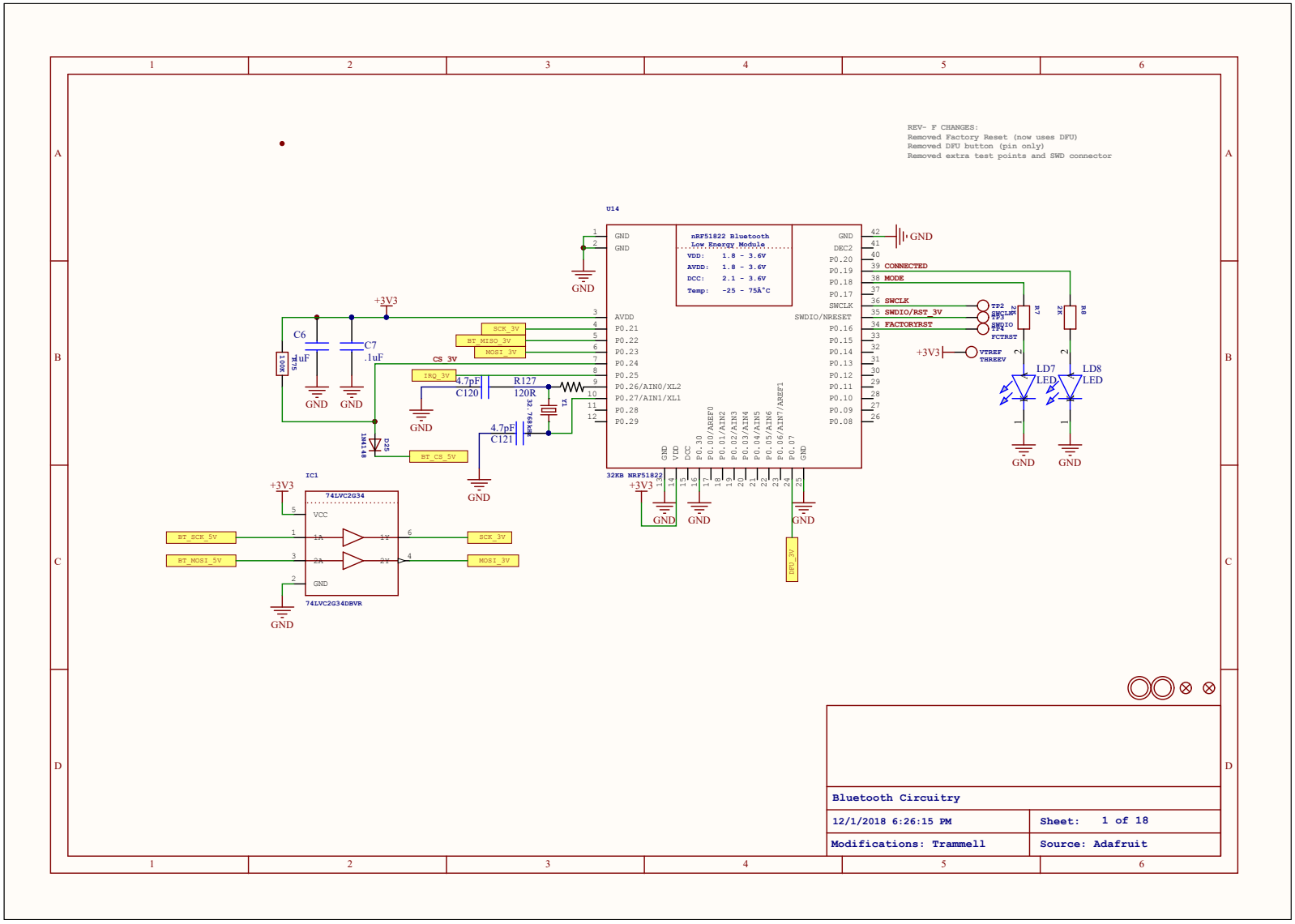


Figure 10: Bluetooth Circuitry Schematic

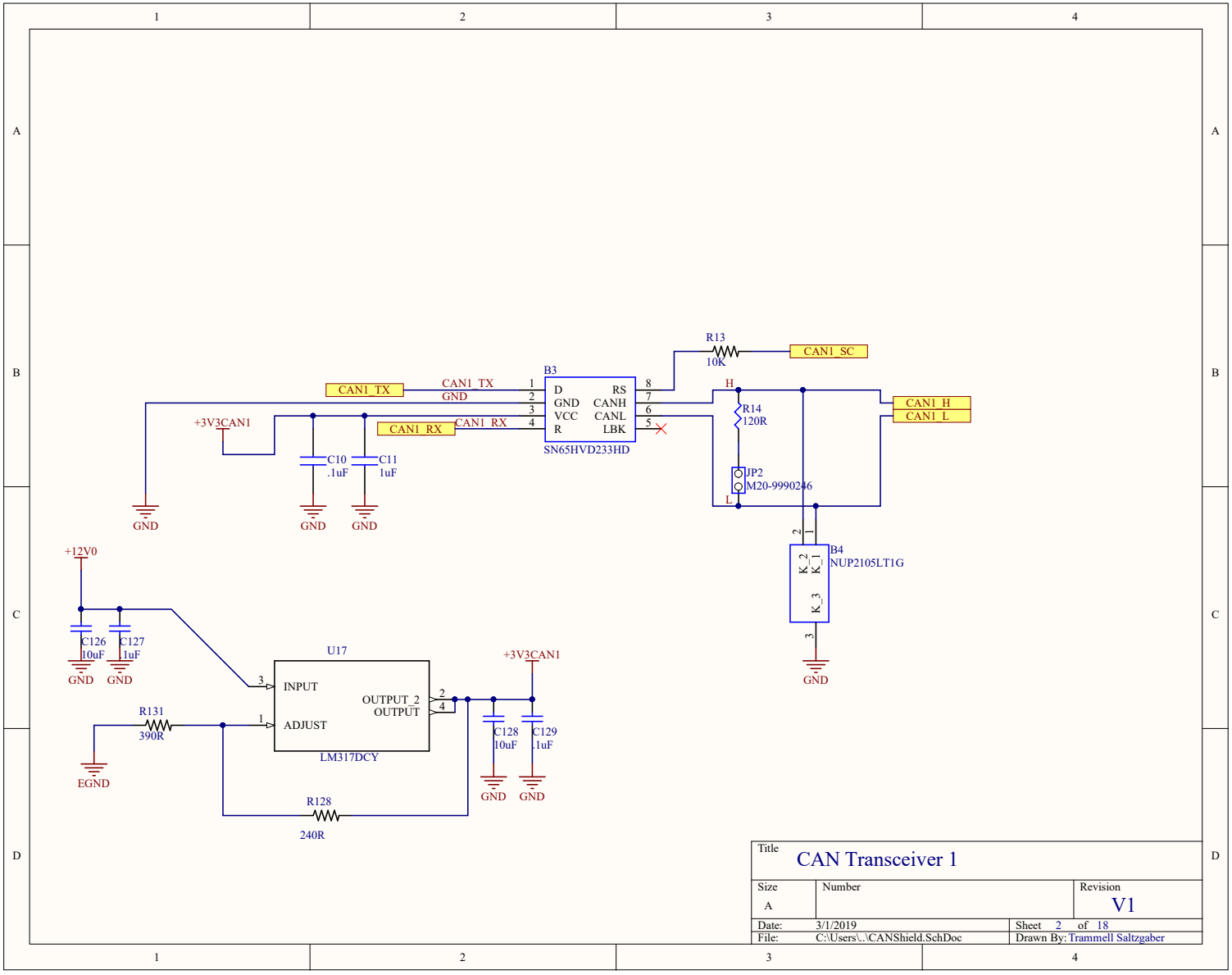


Figure 11: First CAN Transceiver and Power Supply Circuitry

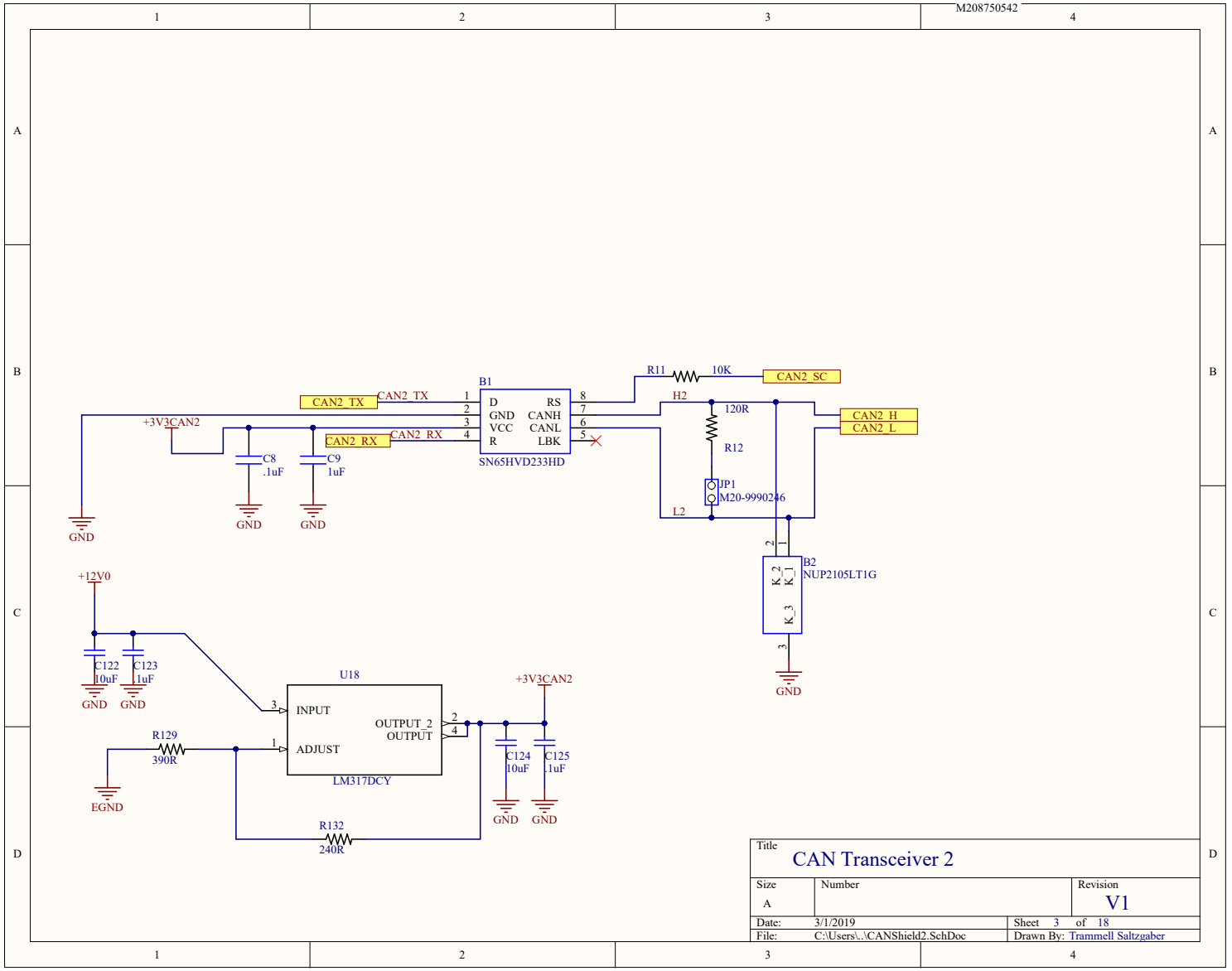


Figure 12: Second CAN Transceiver and Power Supply Circuitry

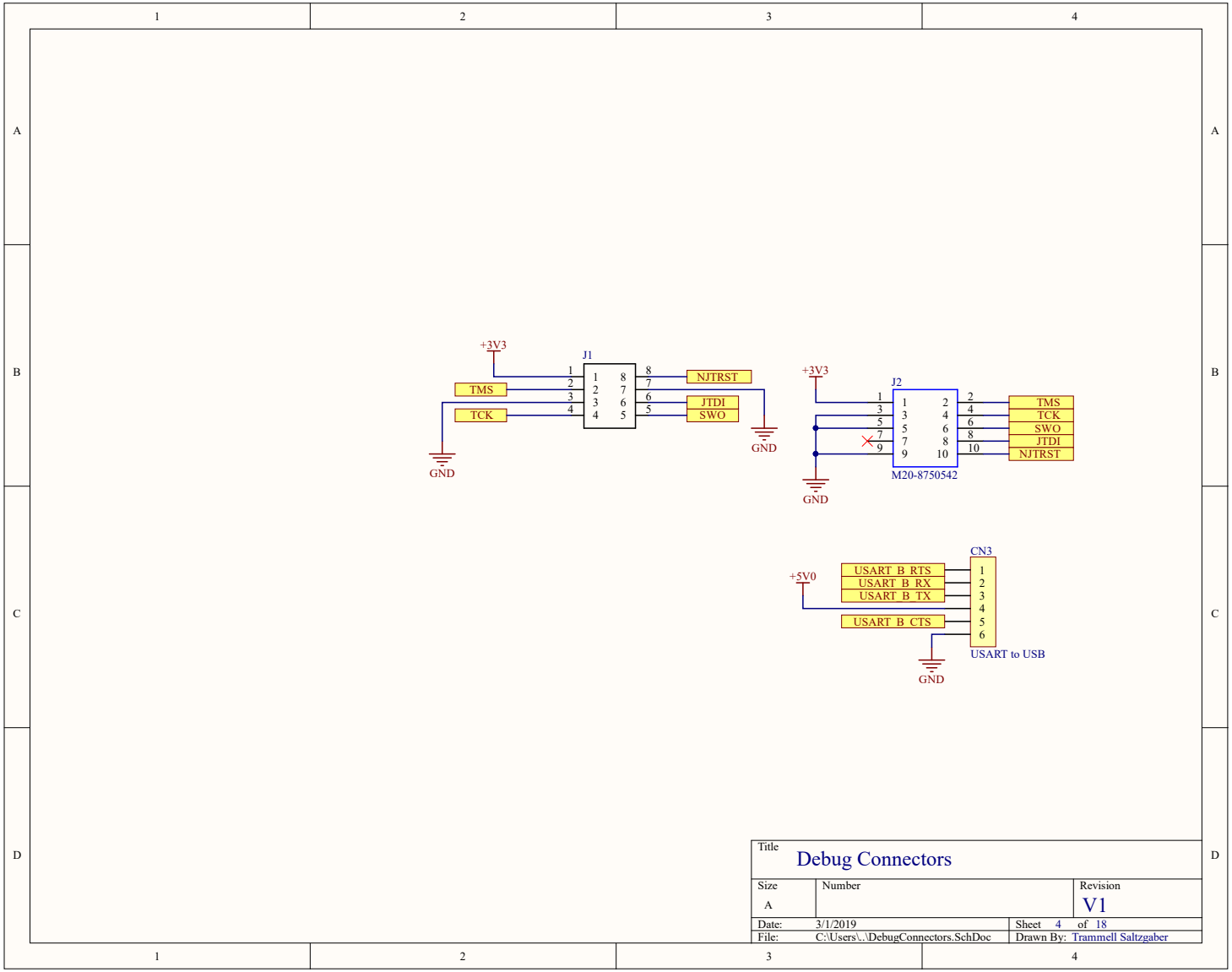


Figure 13: Debug Connector Circuitry

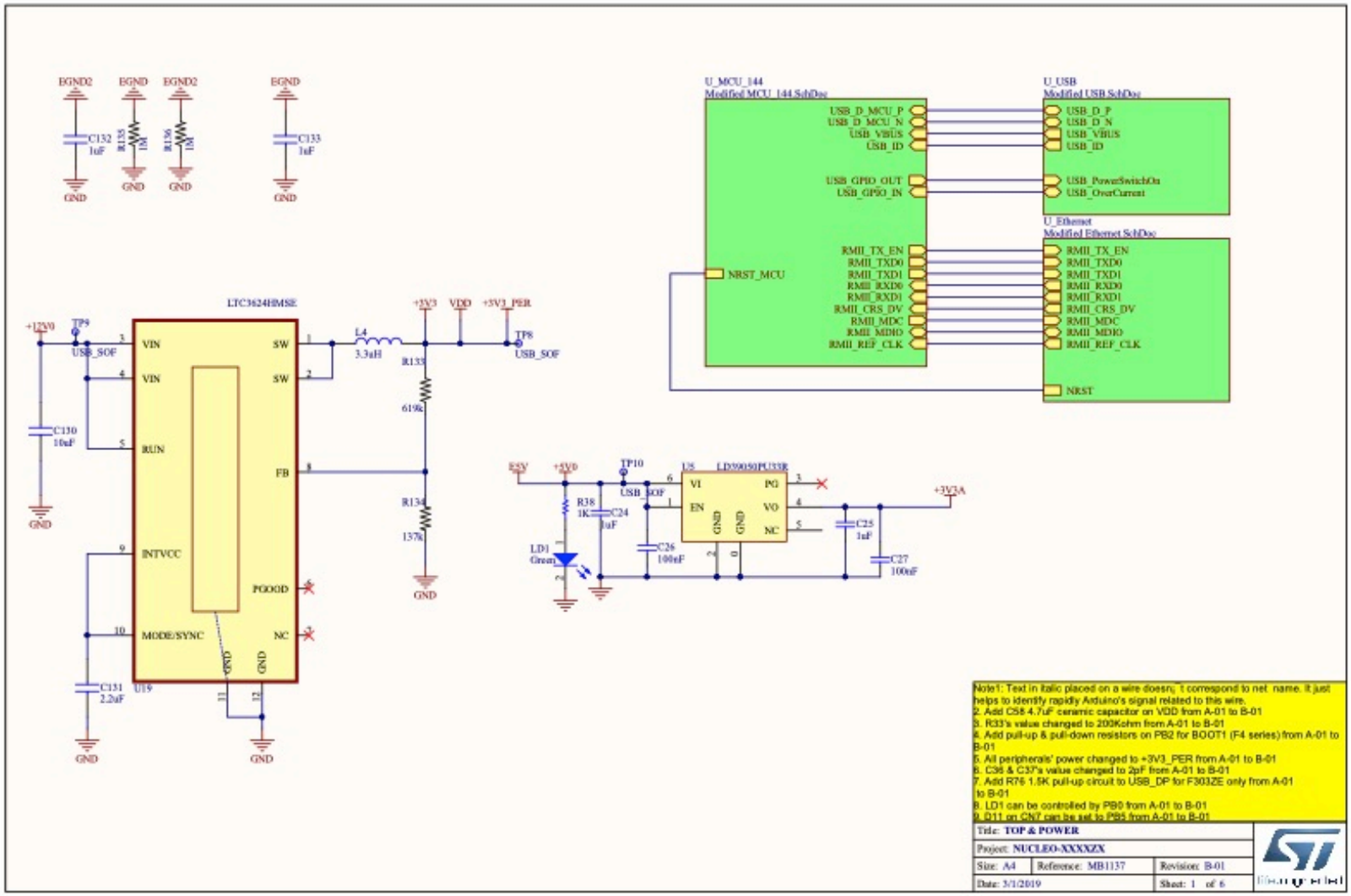


Figure 14: High-Level MCU Schematic with Power Circuitry

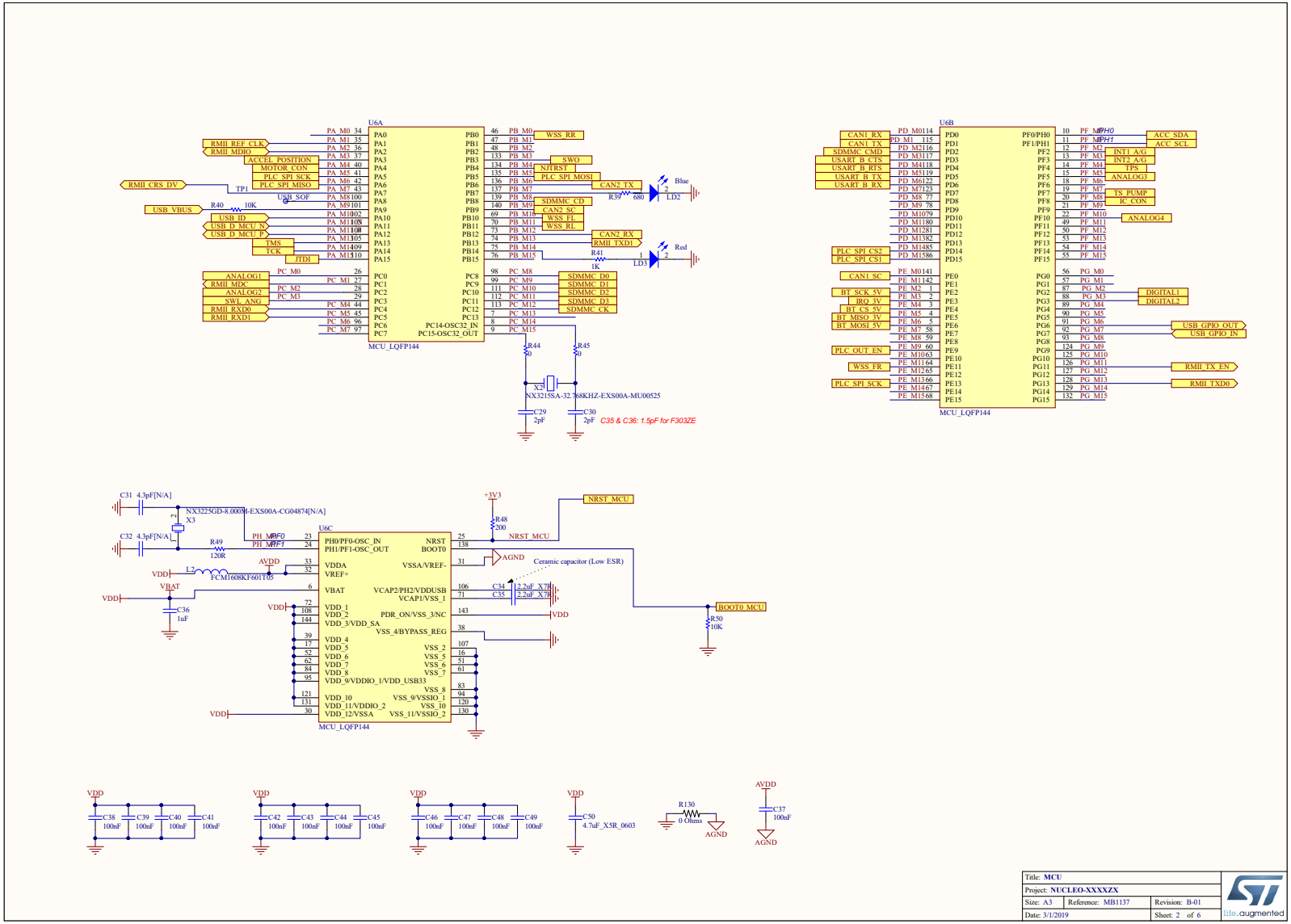


Figure 15: MCU Schematic

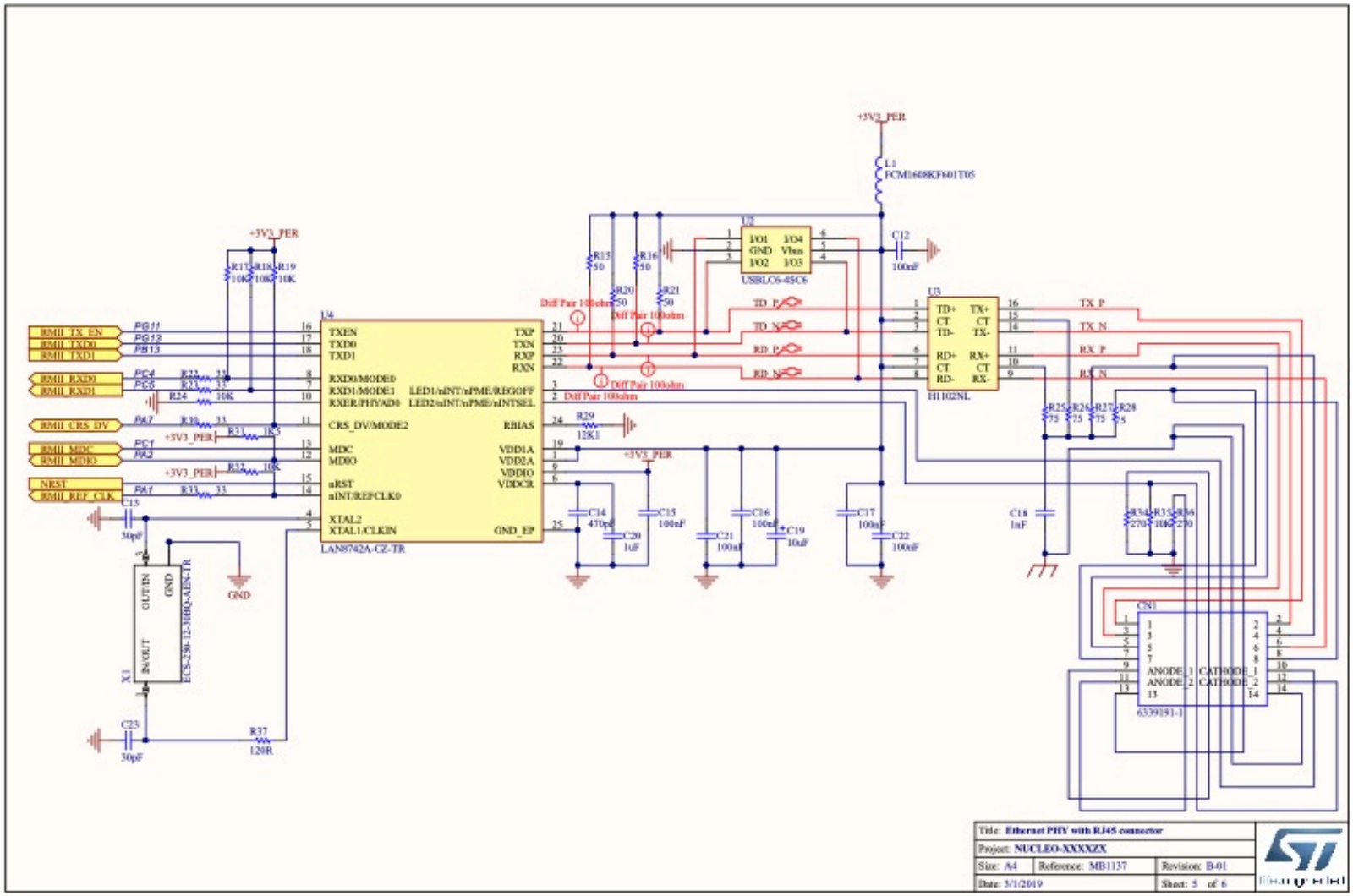


Figure 16: MCU Ethernet Schematic

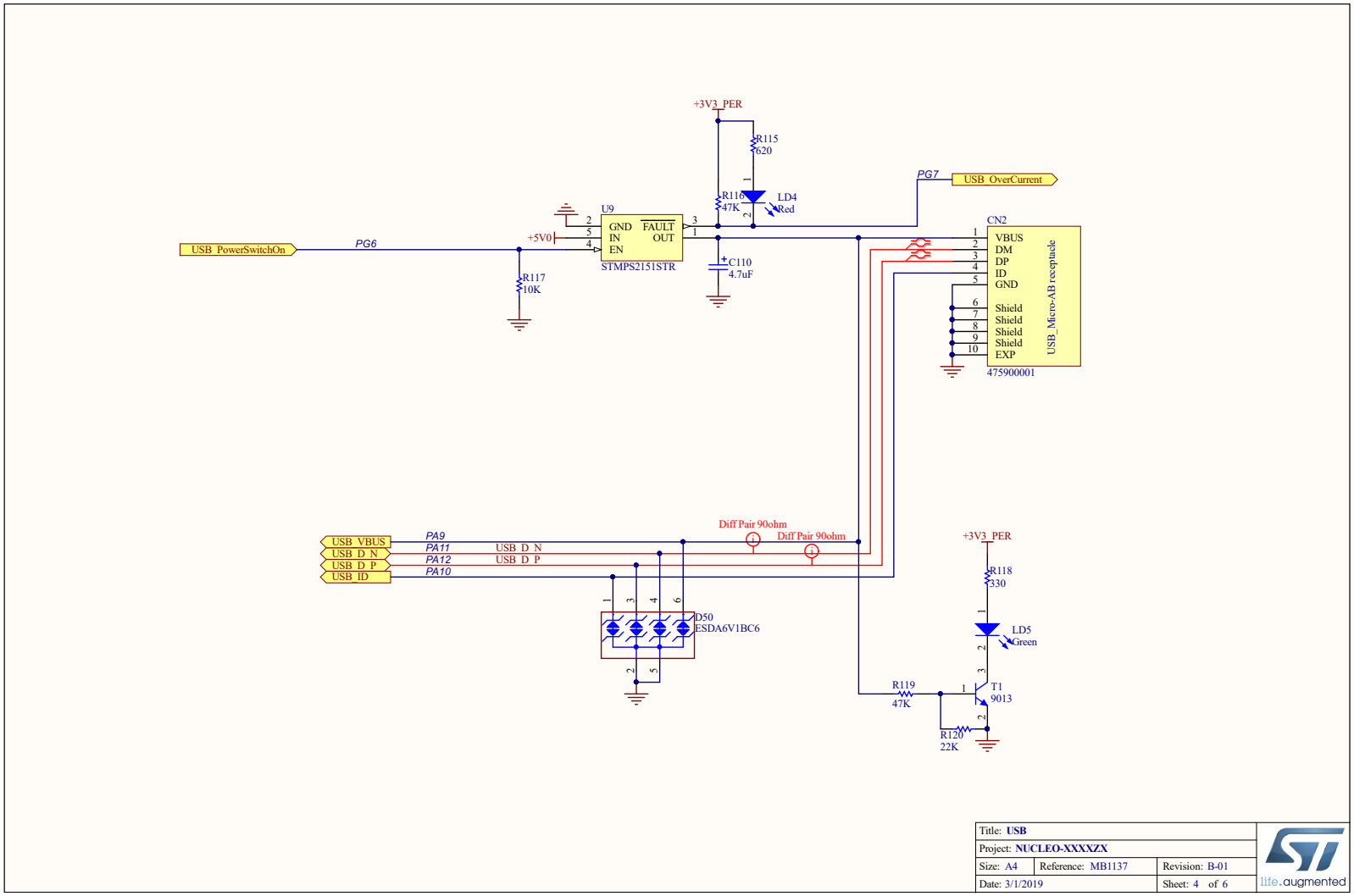


Figure 17: MCU USB Schematic

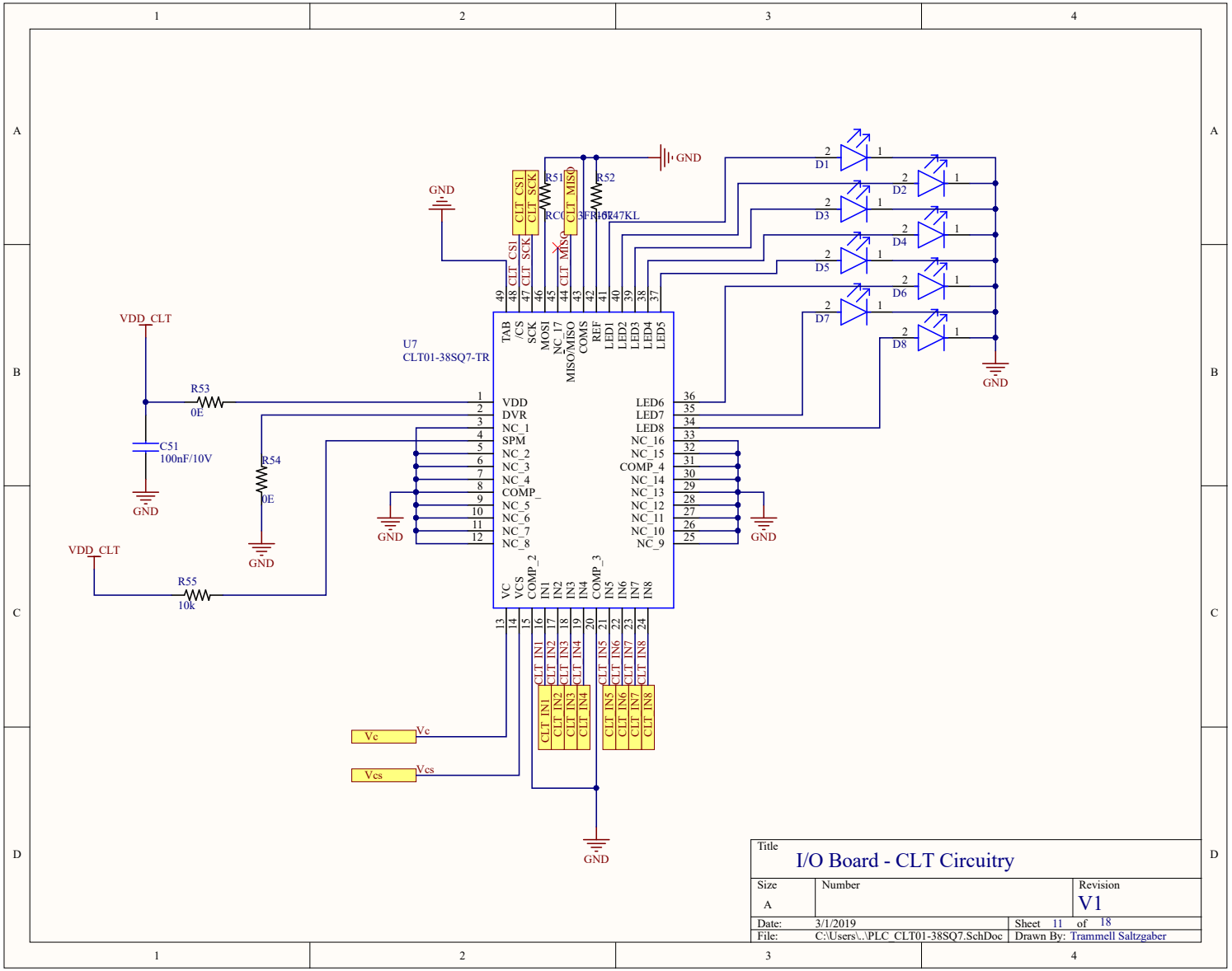


Figure 18: PLC Board CLT01-38SQ7 Circuitry

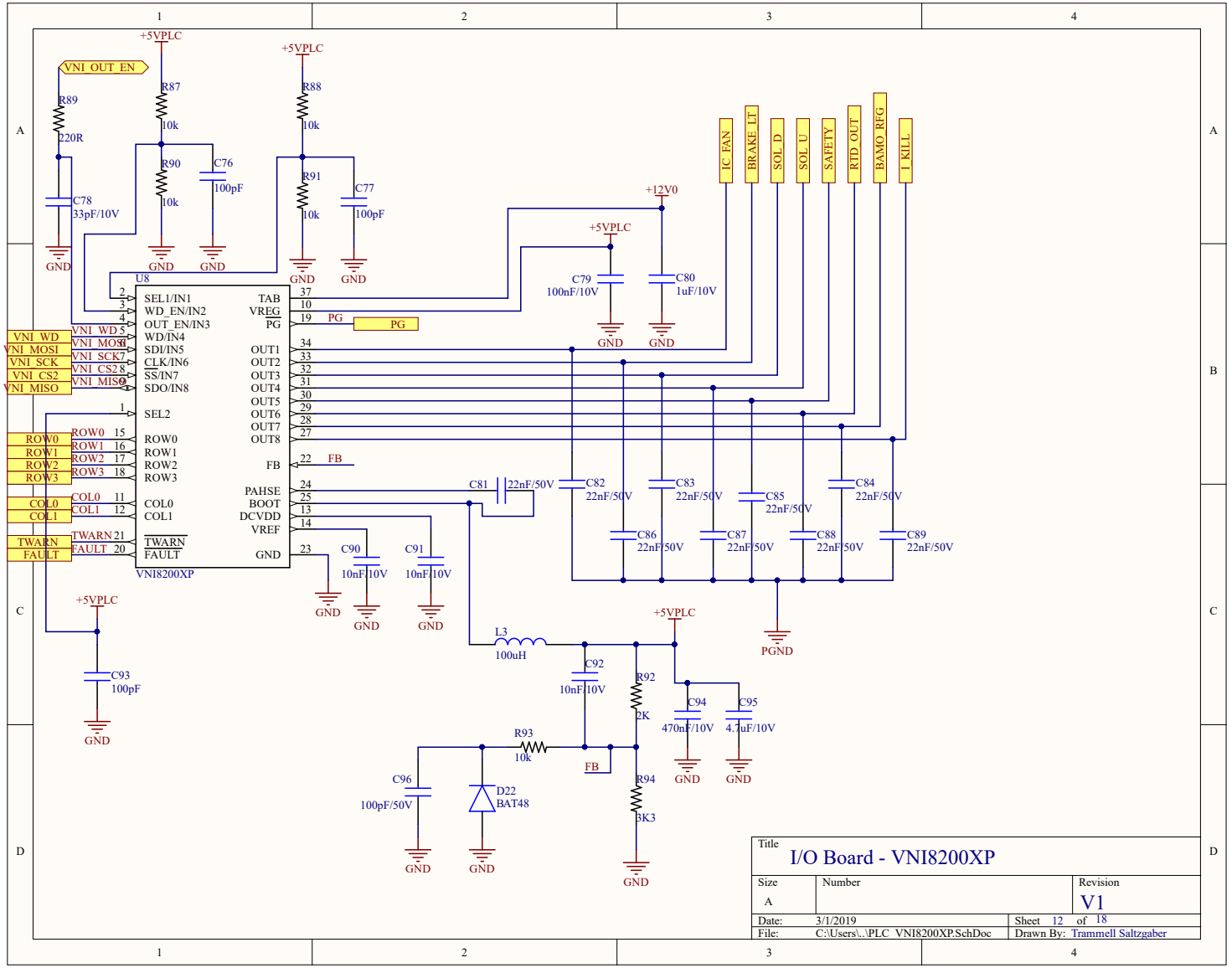


Figure 19: PLC Board VNI8200XP Circuitry

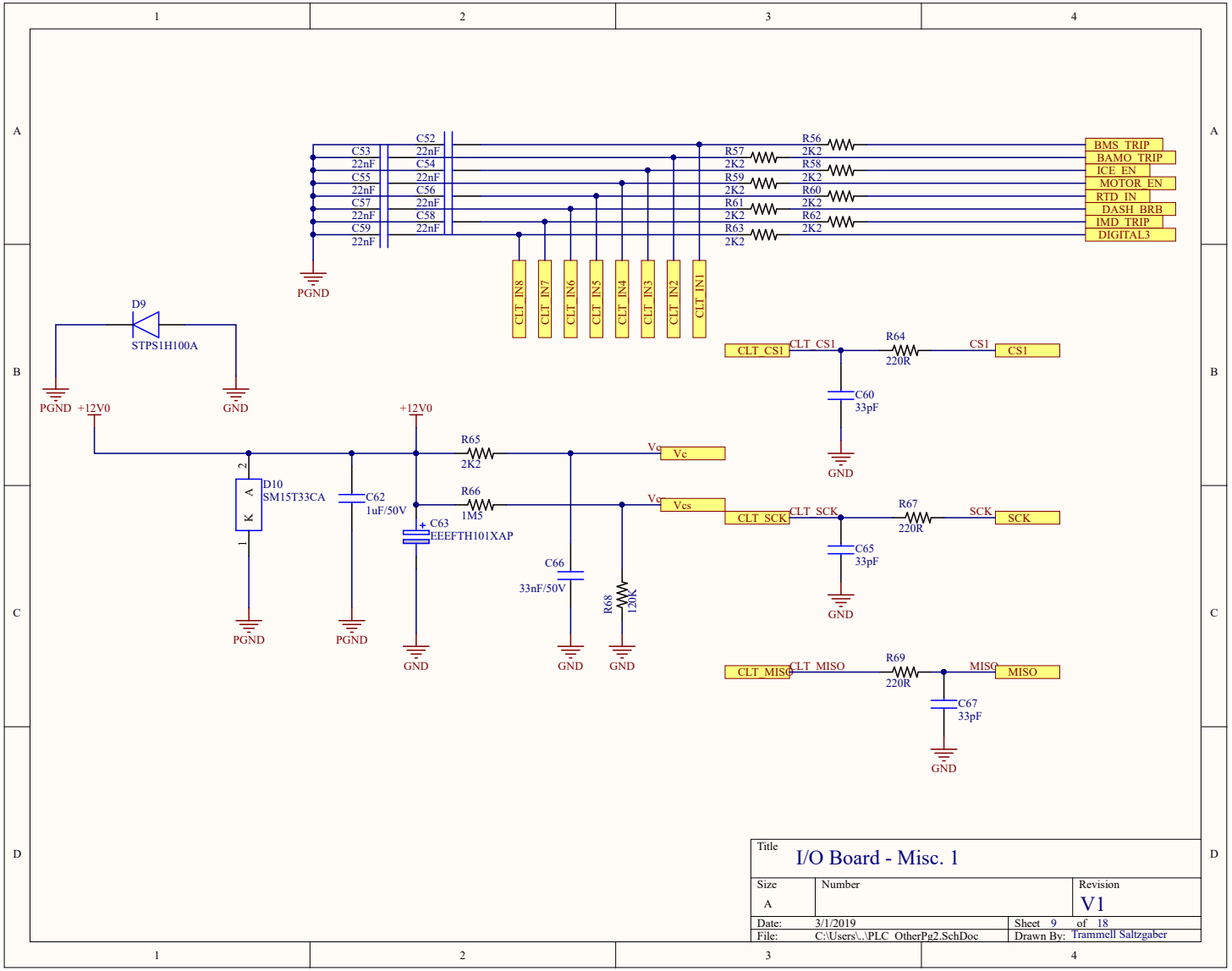


Figure 19: PLC Misc. Circuitry Schematic

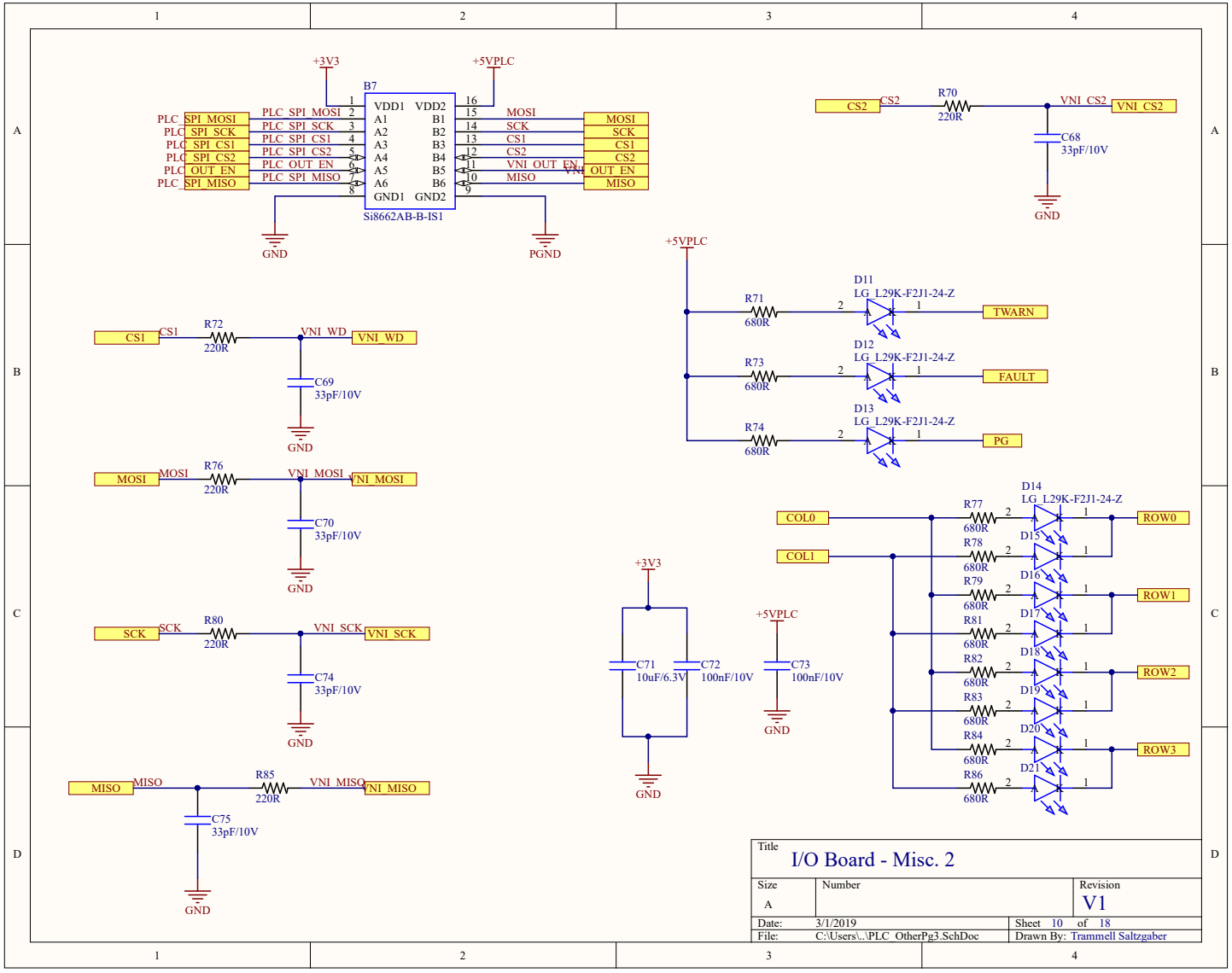


Figure 20: Second PLC Misc. Circuitry Schematic

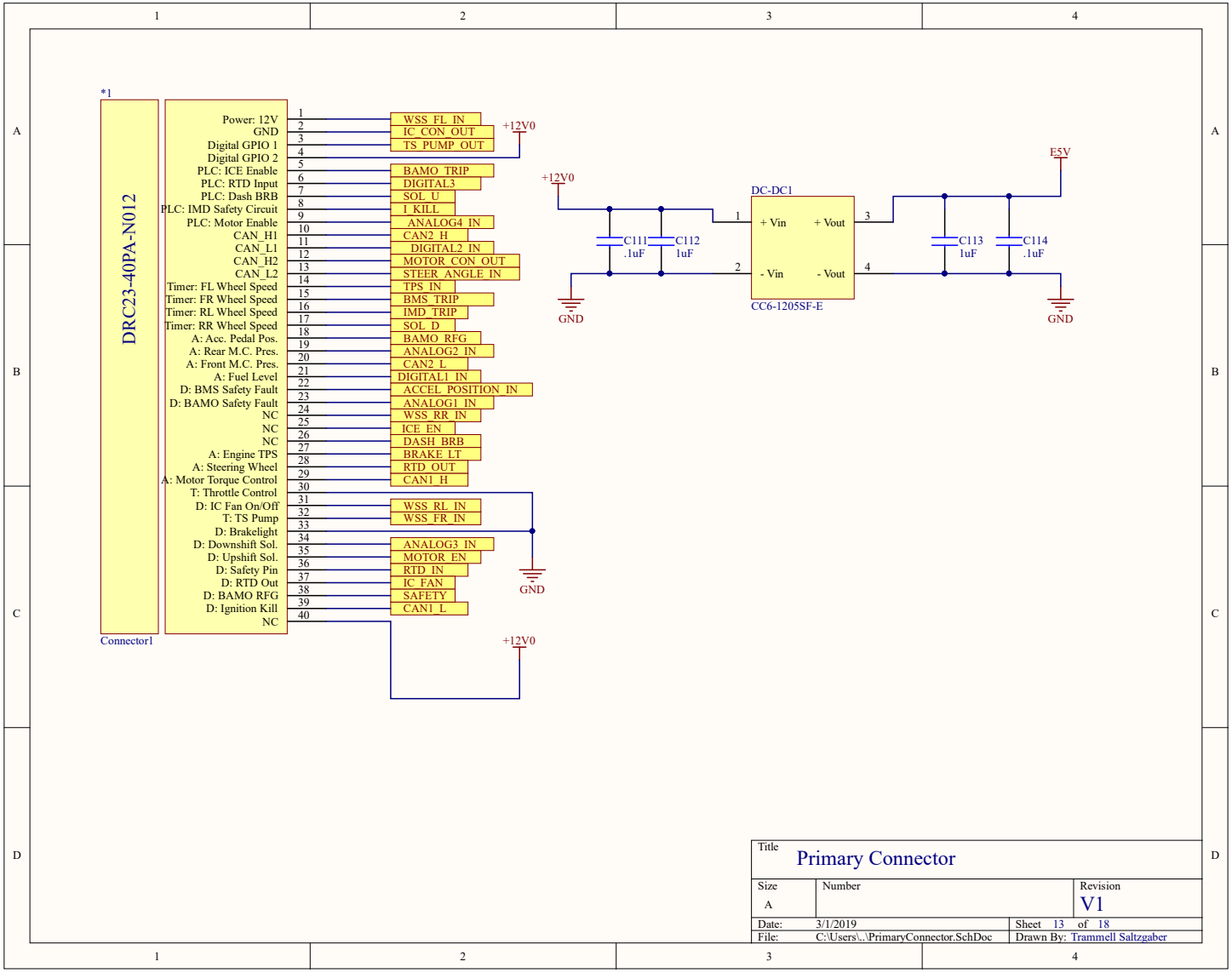


Figure 21: Primary Connector and 12V to 5V DC-DC Converter Schematic

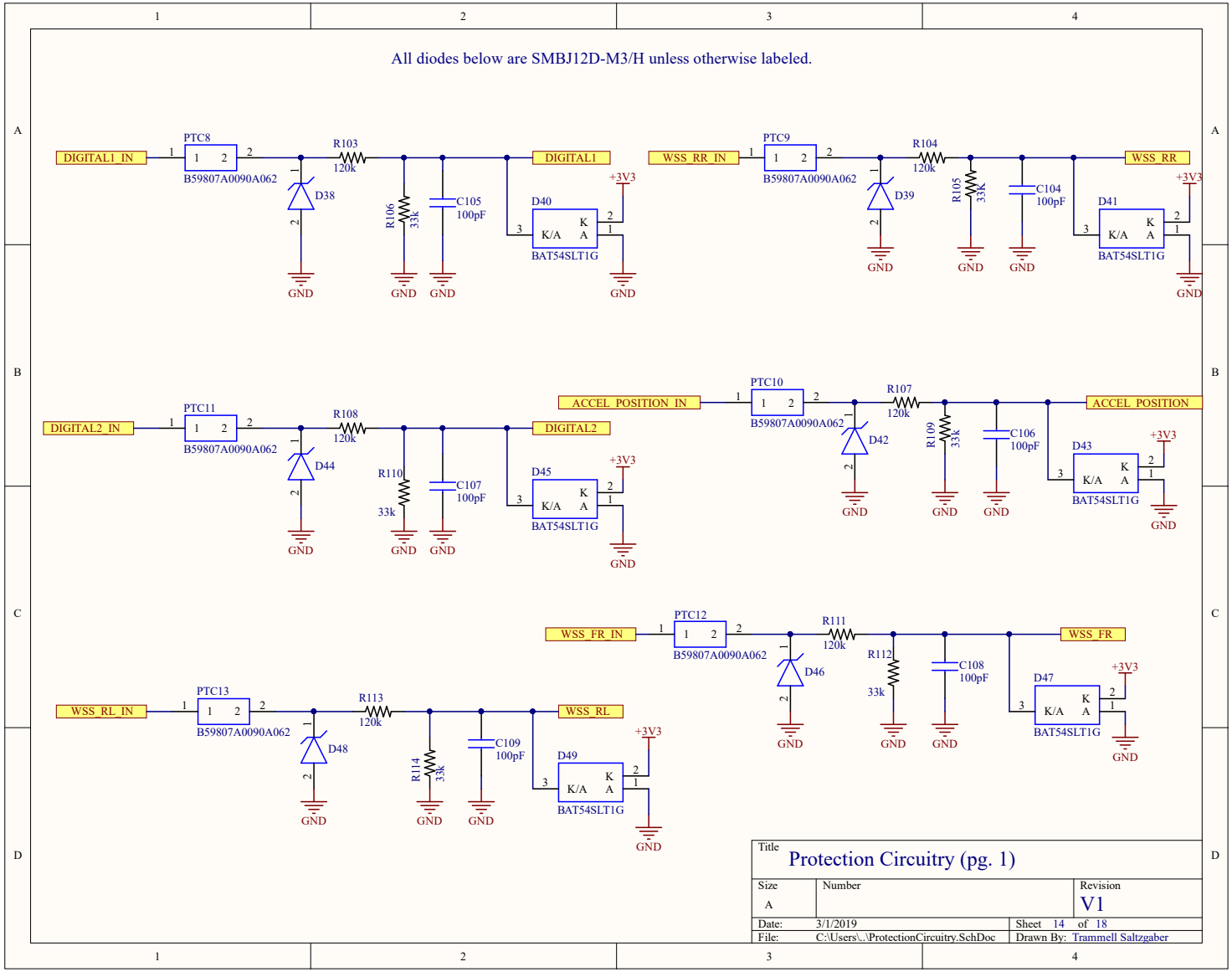


Figure 22: First Protection Circuitry Schematic

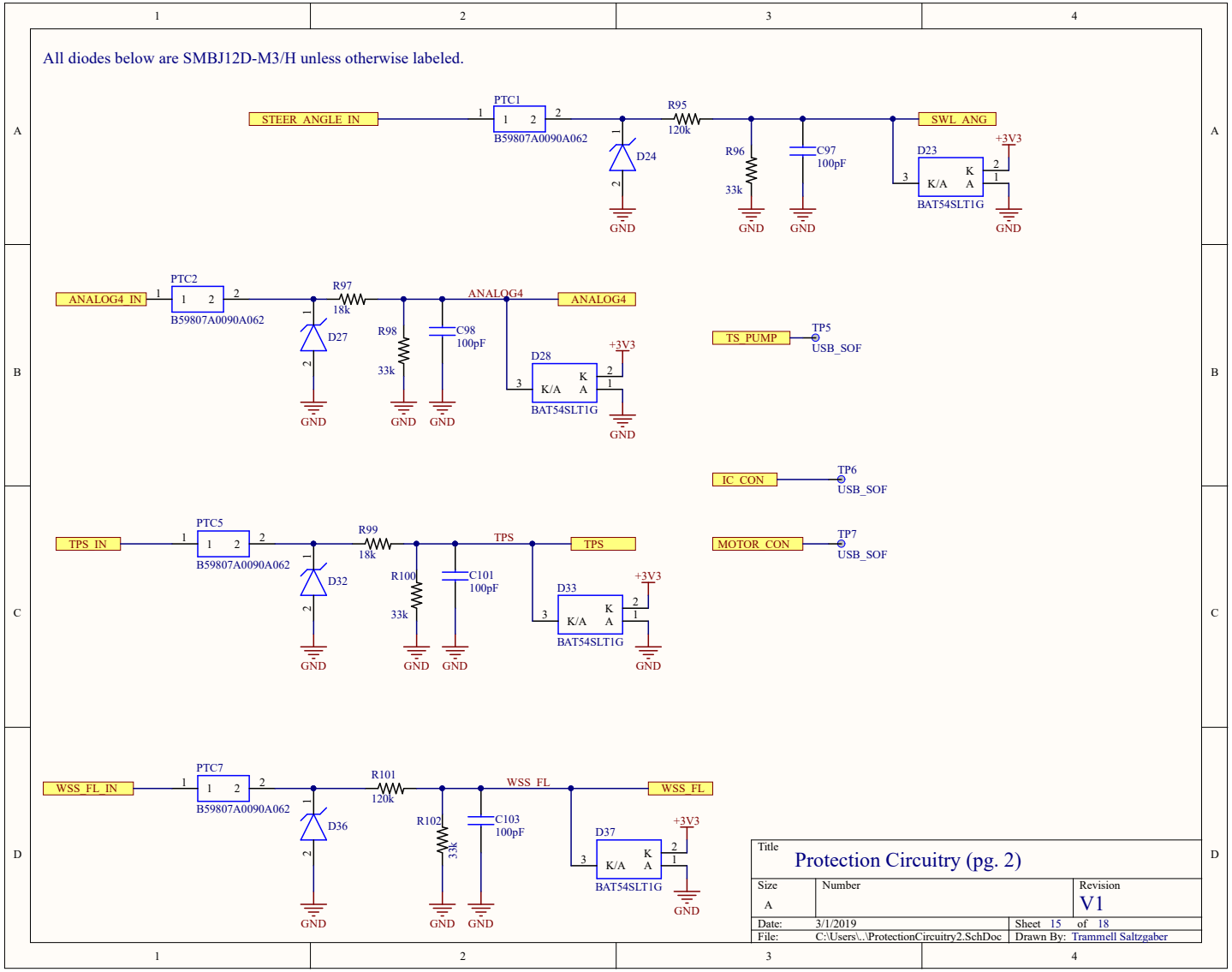


Figure 23: Second Protection Circuitry Schematic

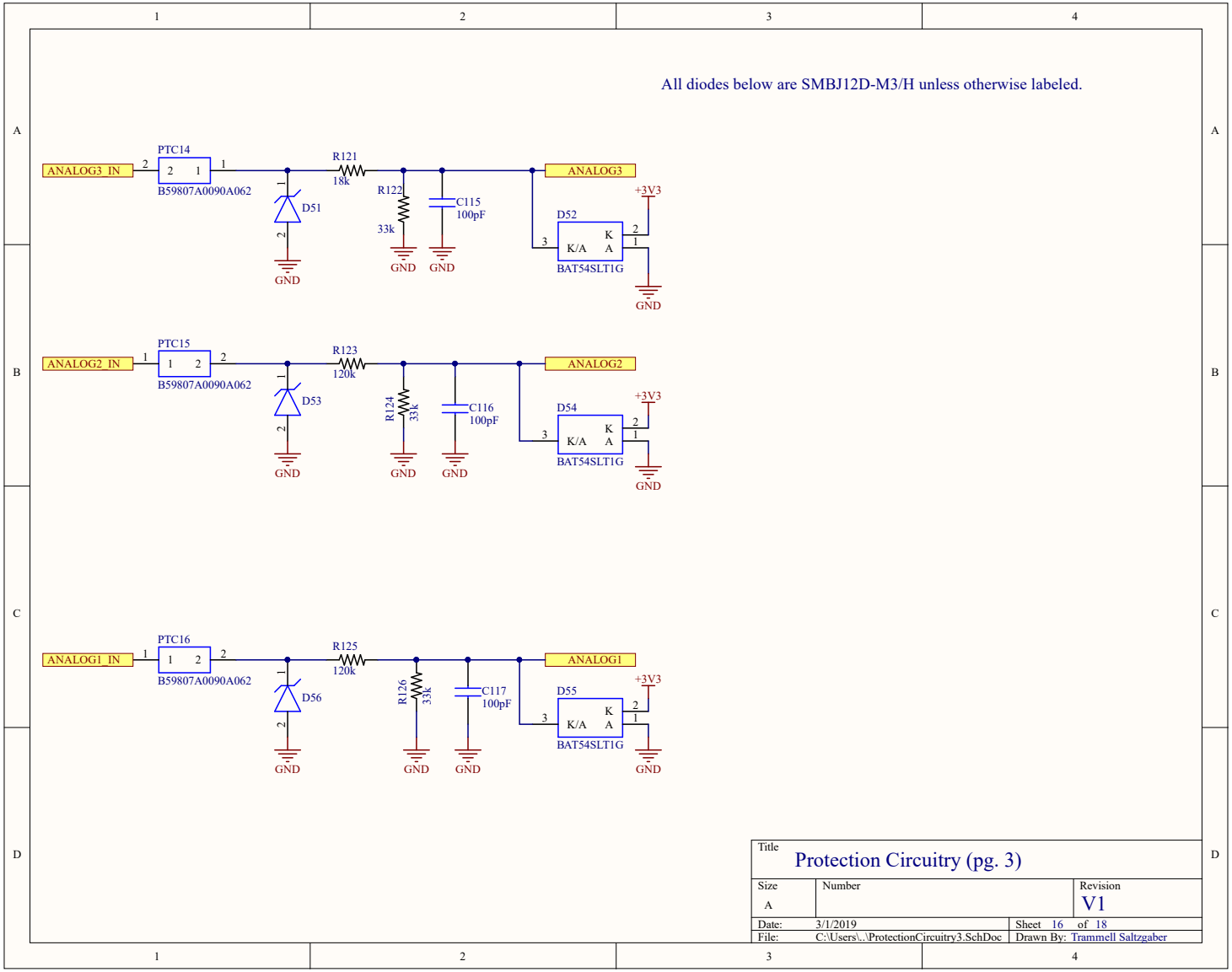


Figure 24: Third Protection Circuitry Schematic

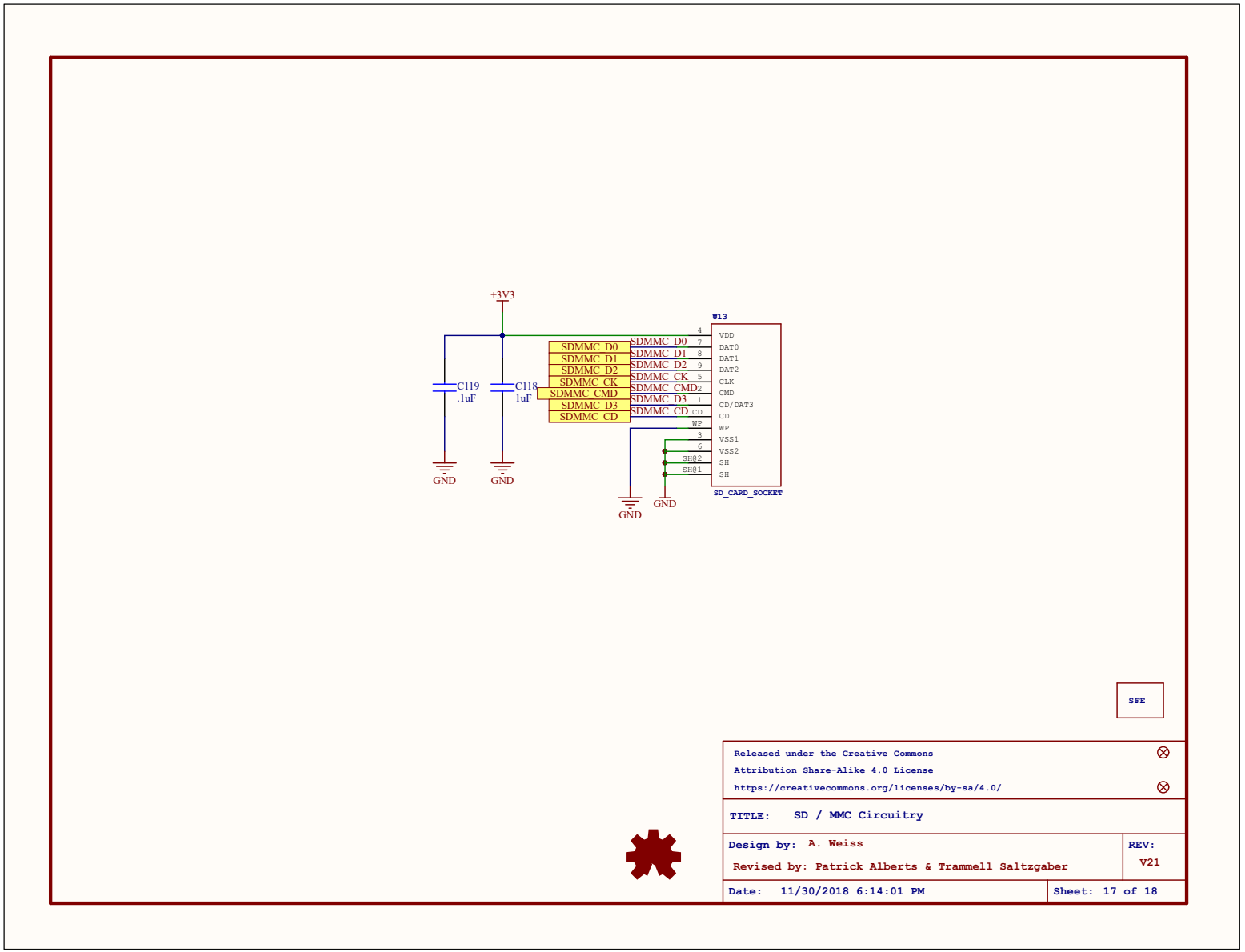


Figure 25: SD Card Slot Schematic

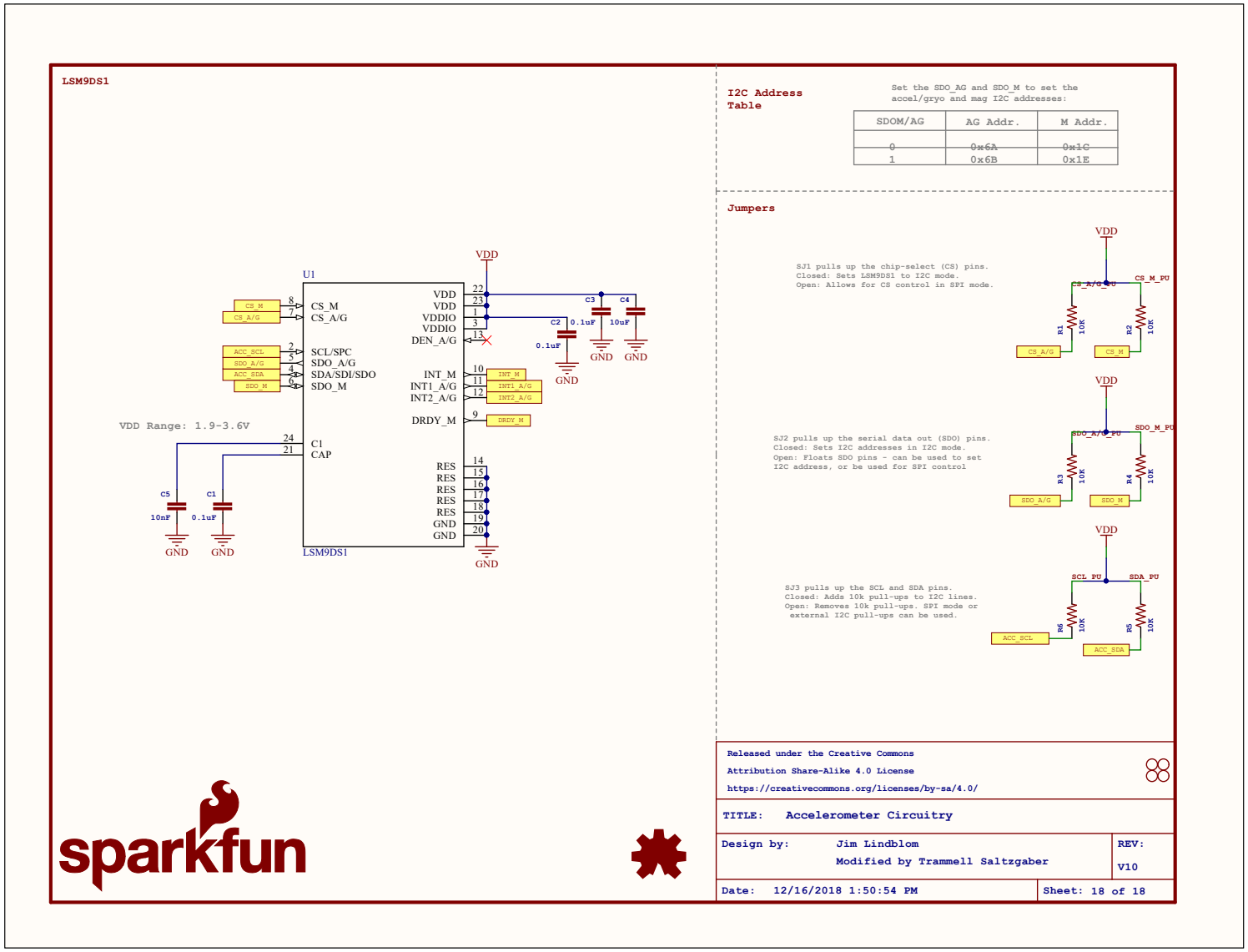


Figure 26: Accelerometer Schematic

8. PCB Layout

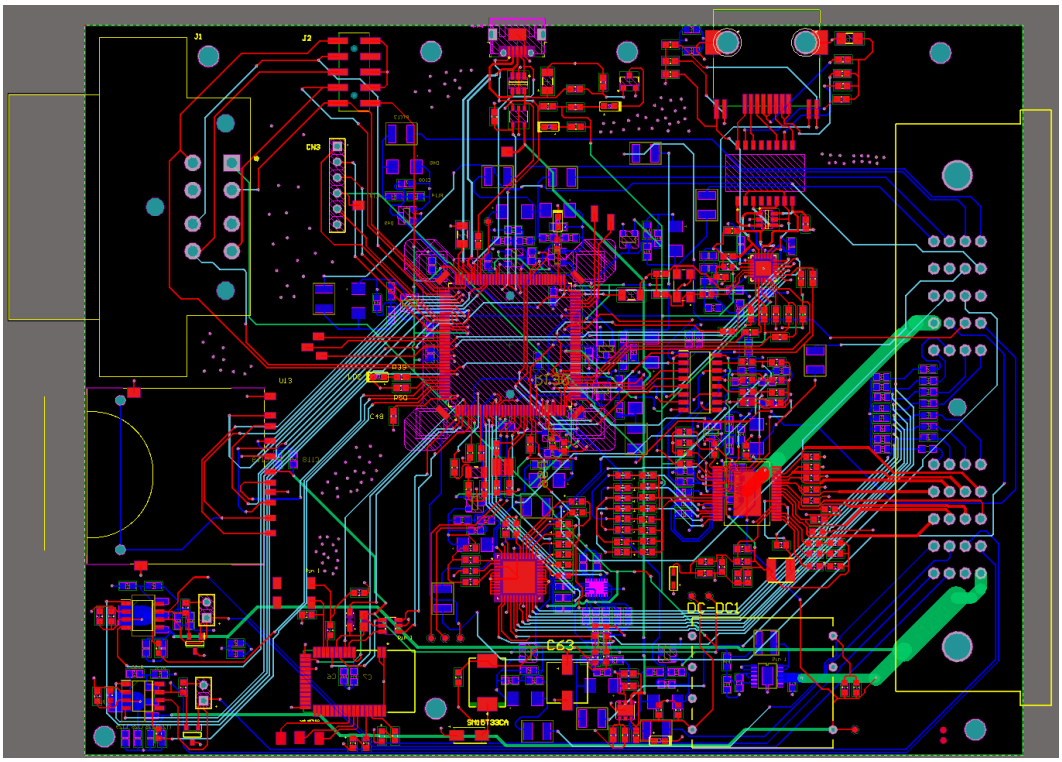


Figure 27: Full PCB Layout

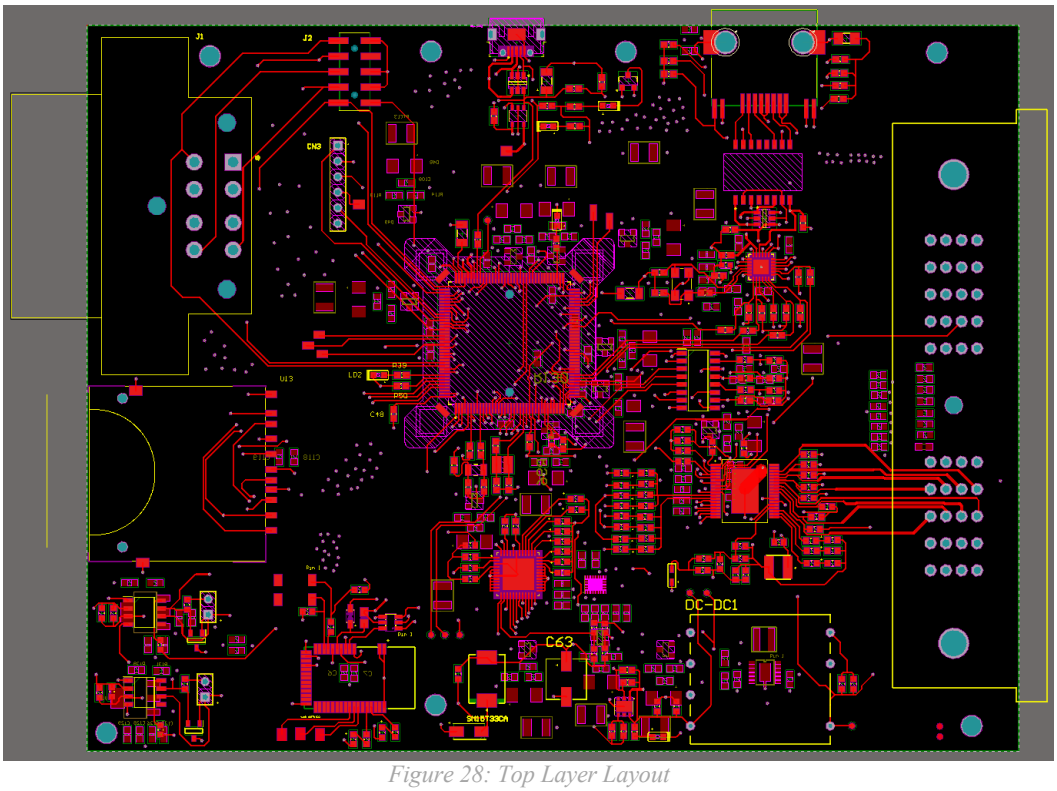


Figure 28: Top Layer Layout

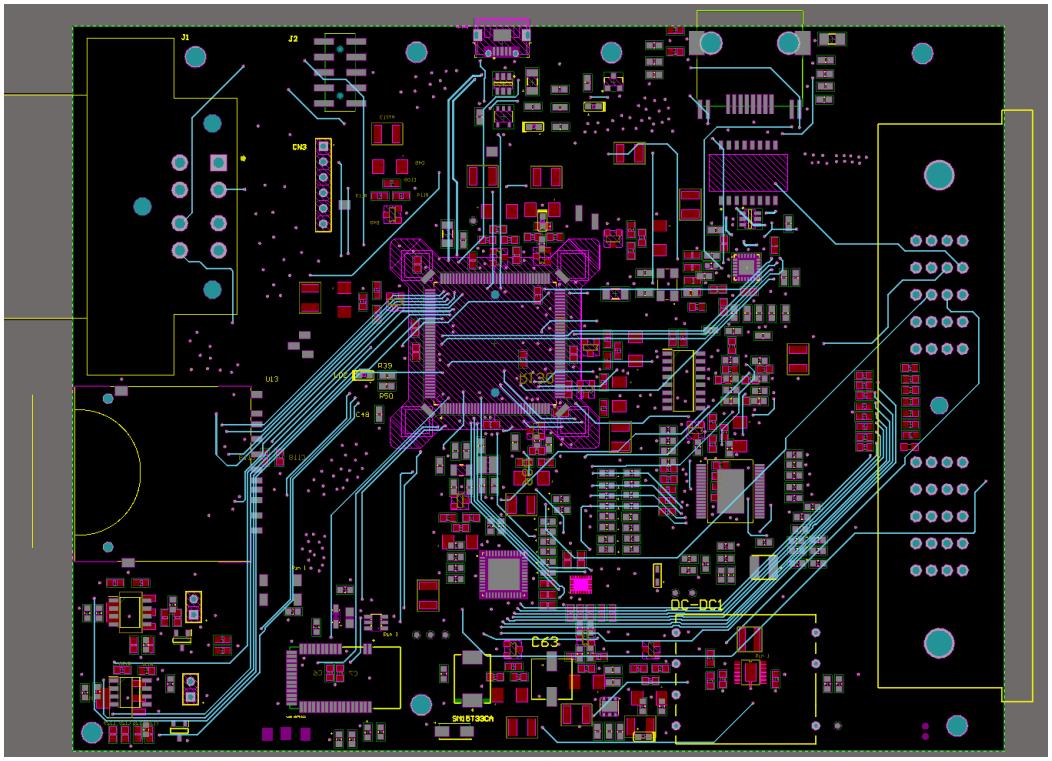


Figure 29: Signal Layer Layout

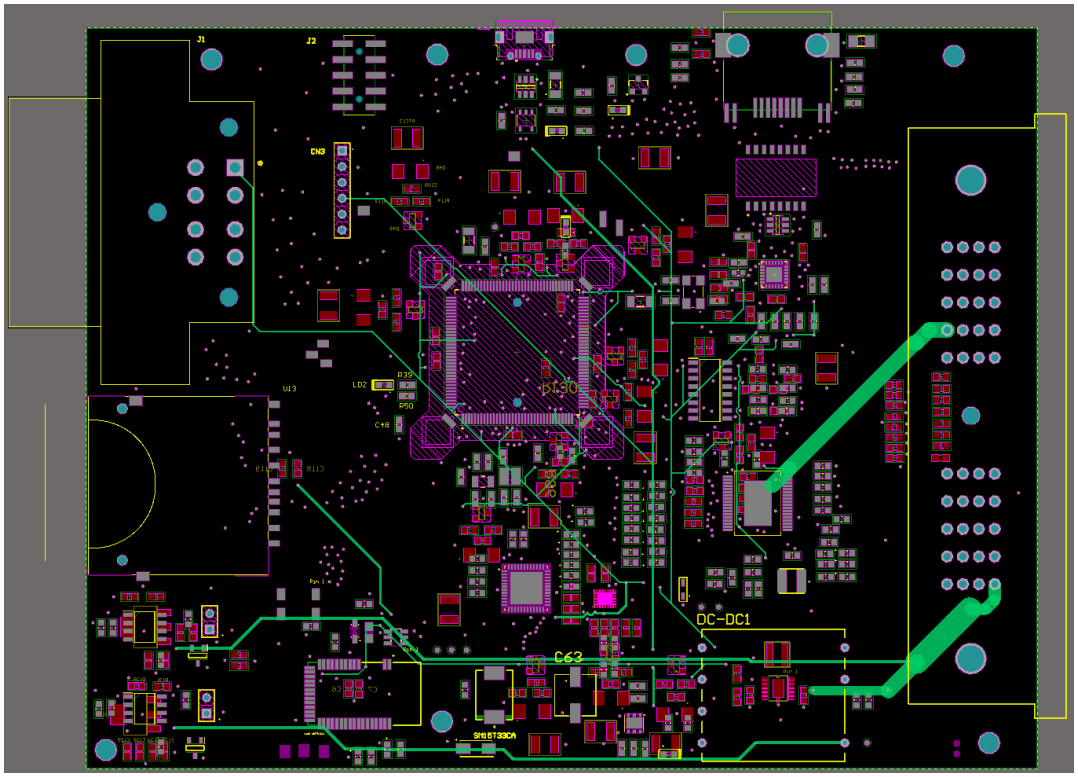


Figure 30: Power Layer Layout

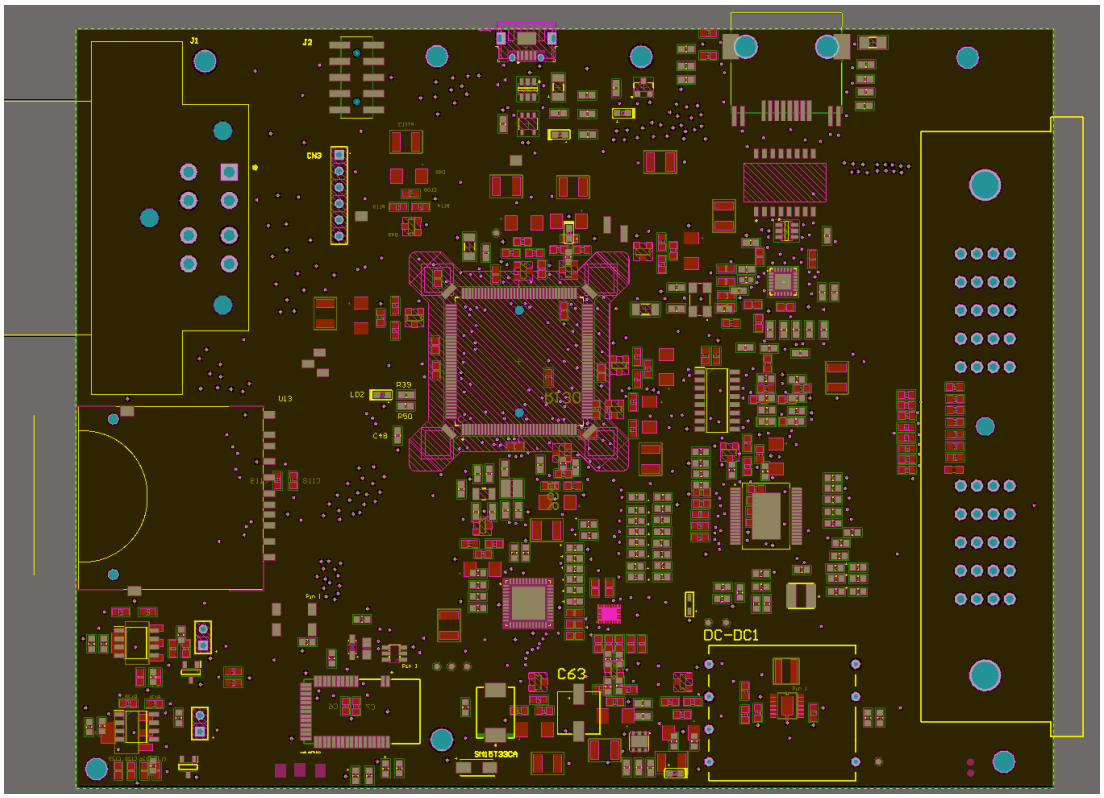


Figure 31: Ground Plate Layer Layout

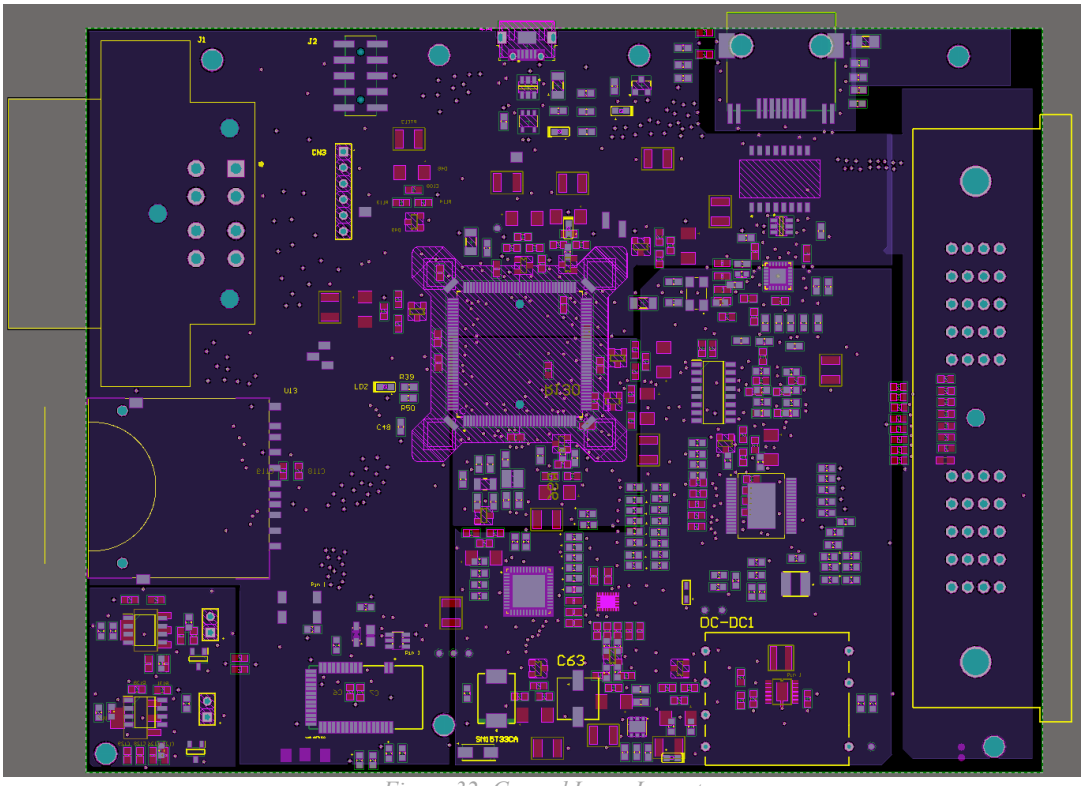


Figure 32: Ground Layer Layout

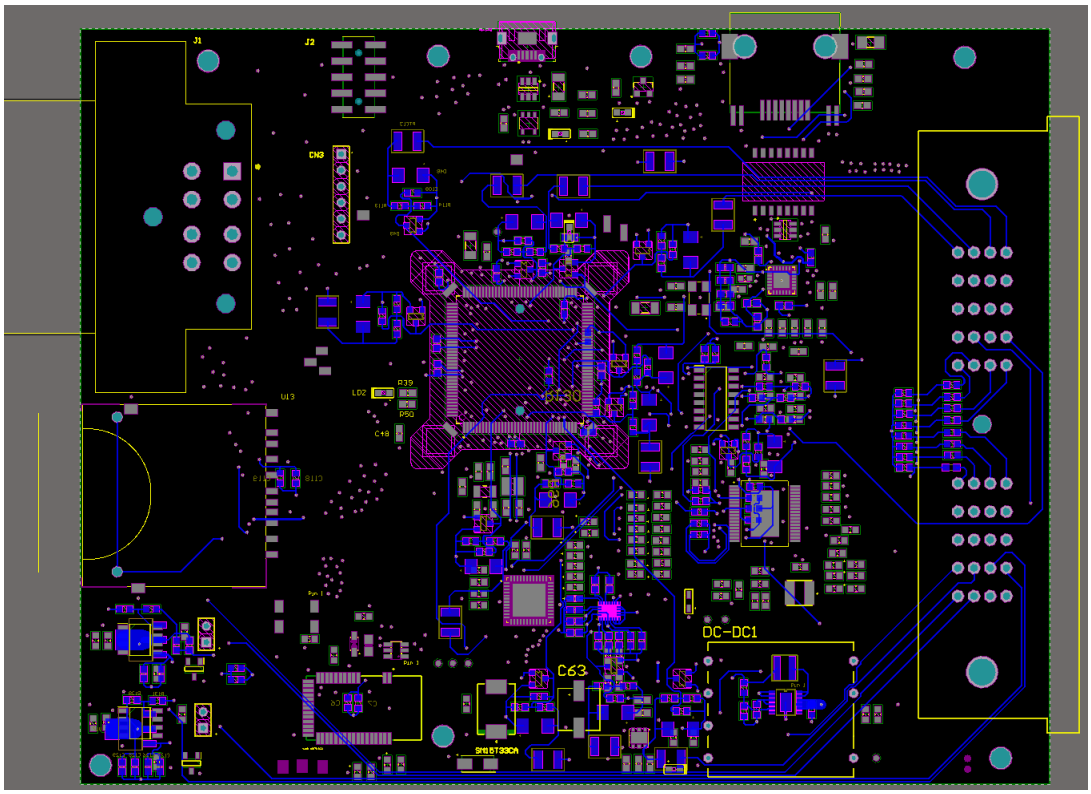


Figure 33: Bottom Layer Layout

References

[1] Rinker, Dylan. “Design Improvements and Performance Validation of a Competitive Hybrid FSAE Vehicle.” *University of Idaho*, 2014, www.uidaho.edu/engr/research/niatt/tranlive/database/dtrt12gutc17-ui-klk908.

[2] “Uvic Formula Hybrid.” *Racecar Engineering*, 15 May 2016, www.racecar-engineering.com/cars/uvic-formula-hybrid/.

[3] Caratti, Andrea, et al. “Development of a Predictive Model for Regenerative Braking System.” *2013 IEEE Transportation Electrification Conference and Expo (ITEC)*, 2013, doi:10.1109/itec.2013.6573497.

[4] Shoubo, Li, et al. “Traction Control of Hybrid Electric Vehicle.” *2009 IEEE Vehicle Power and Propulsion Conference*, 2009, doi:10.1109/vppc.2009.5289563.

[5] McFarland, Zak, et al. “Formula Hybrid Drivetrain Design.” *Digital Commons - CalPoly*, June 2012, digitalcommons.calpoly.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1131&context=mesp.

[6] Gatta, Nick, et al. “Formula SAE Electric Drive Control.” *The University of Akron*, 2012.

[7] Pettersson, M., and L. Nielsen. “Gear Shifting by Engine Control.” *IEEE Transactions on Control Systems Technology*, vol. 8, no. 3, 2000, pp. 495–507., doi:10.1109/87.845880.

[8] “Uvic Formula Hybrid.” *Racecar Engineering*, 15 May 2016, www.racecar-engineering.com/cars/uvic-formula-hybrid/.

[9] Ivanov, Valentin, et al. “Wheel Slip Control for All-Wheel Drive Electric Vehicle with Compensation of Road Disturbances.” 2014, pp. 1–10., doi:10.13140/RG.2.1.3645.5680/1.

[10] Parsania, Pratik, and Ketan Saradava. “Drive-By-Wire Systems In Automobiles.” *Journal of Systematic Computing, At VVP Engineering College*, vol. 6, Dec. 2012.

[11] Borrelli, Francesco, et al. “A Hybrid Approach to Traction Control.” *Hybrid Systems: Computation and Control Lecture Notes in Computer Science*, 2001, pp. 162–174., doi:10.1007/3-540-45351-2_16.

[12] Jalali, Kiumars, et al. “Development of a Fuzzy Slip Control System for Electric Vehicles with In-Wheel Motors.” *SAE International Journal of Alternative Powertrains*, Apr. 2012, pp. 46–64., doi:10.4271/2012-01-0248.

[13] Lewton, Dylan Lewis. “Application of Floating Pedal Regenerative Braking for a Rear-Wheel-Drive Parallel-Series Plug-In Hybrid Electric Vehicle with an Automatic Transmission.”*Dissertations and Theses*, p. 307.

[14] Singh, G., et al. “Novel Automated Manual Transmission Gear-Shift Map Modelling Based on Throttle Position.” *International Journal Of Automotive And Mechanical Engineering*, vol. 15, no. 1, 2018, pp. 5053–5073., doi:10.15282/ijame.15.1.2018.12.0391.

[15] Shen, Wenchen, et al. “Optimization of Shift Schedule for Hybrid Electric Vehicle with Automated Manual Transmission.” *Energies*, vol. 9, no. 220, 2016, doi:10.3390/en9030220.

[1] M. K. Yoong *et al*., "Studies of regenerative braking in electric vehicle," *IEEE Conference on Sustainable Utilization and Development in Engineering and Technology*, Petaling Jaya, 2010, pp. 40-45.

[16] “Arm Cortex KV5x.” NXP, 2018, www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/kinetis-cortex-m-mcus/v-seriesreal-time-ctlm0-plus-m4-m7/kinetis-kv5x-240-mhz-motor-control-and-power-conversion-ethernet-mcus-based-on-arm-cortex-m7:KV5x.

[17] “SAM E70.” Digikey, 2018, www.digikey.com/en/product-highlight/a/atmel/sam-e70-microcontrollers.

[18] "Nucleo F767ZI.” ST, 2018, www.st.com/en/evaluation-tools/nucleo-f767zi.html

[19] “SparkFun SD/MMC Card Breakout.” *COM-14646 - SparkFun Electronics*, www.sparkfun.com/products/12941.

[20] “NUCLEO F767ZI Datasheet.” ST, www.st.com/resource/en/data_brief/nucleo-f767zi.pdf.

[21] “X-NUCLEO-IDB05A1 .” *ST*, 2018, www.st.com/en/ecosystems/x-nucleo-idb05a1.html.

[22] *Getting Started with the FP-NET-6LPETH1 Software Package Connecting 6LoWPAN IoT Nodes to the Internet via Ethernet Networks*. ST, 2017.

[23] “STMicroelectronics WiFi Expansion Board for SPWF04SA for STM32 Nucleo.” *RS Components*, uk.rs-online.com/web/p/radio-frequency-development-kits/1438875/.

[24] “I-NUCLEO-MANTIS .” *ST*, 2018, www.st.com/en/evaluation-tools/i-nucleo-mantis.html.

[25] “X-NUCLEO-IDW01M1 .” *ST*, 2018, www.st.com/en/ecosystems/x-nucleo-idw01m1.html.

[26] “USB CONNECTOR GUIDE — GUIDE TO USB CABLES.” *C2G*, 2018, www.cablestogo.com/learning/connector-guides/usb.

[27] “X-NUCLEO-IDB04A1 .” *ST*, 2018, www.st.com/en/ecosystems/x-nucleo-idb04a1.html.

[28] “SPIRIT1 .” *ST*, 2018, www.st.com/en/wireless-connectivity/spirit1.html.

[29] “L3GD20 Datasheet.” ST, 2013, www.st.com/resource/en/datasheet/l3gd20.pdf.

[30] “MPU-6000 and MPU-6500 Datasheet.” InvenSense, 2013, store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3 4.pdf.

[31] “LSM9DS1 Datasheet.” ST, 2015, www.st.com/resource/en/datasheet/DM00103319.pdf.

[32] "LIS331HH Datasheet." ST, 2009, www.st.com/resource/en/datasheet/lis331hh.pdf.

[33] “SparkFun Triple Axis Accelerometer and Gyro Breakout - MPU-6050.” *COM-14646 - SparkFun Electronics*, www.sparkfun.com/products/11028.

[34] Walls, Colin. “Selecting an Operating System for an Embedded Application.” Embedded, www.embedded.com/design/operating-systems/4436454/Selecting-an-operating-system-for-an-embedded-application-.
[35] “List of Open Source Real-Time Operating Systems.” Open Source RTOS, www.osrtos.com/.
[36] Holt A., Huang CY. (2014) Introduction. In: Embedded Operating Systems. Undergraduate Topics in Computer Science. Springer, London