Low-Cost CAN Logger

Ryan Dillard, Maxim Feoktistov, Anton Liakhovitch, Ashley Reid

Table of Contents

1.0 Overview

1.1 Executive Summary-

The Low Cost Data Logger is a low cost automotive data collection device designed for Hyster-Yale Materials Handling, inc. The data logger attaches to the diagnostics port of a vehicle and records messages sent through the Controller Area Network (CAN) bus - the machine's internal communication system [1]. The recorded data includes metrics like sensor measurements, error codes, and machine state. Engineers can test vehicles while running the data logger, then download and analyze the data to determine the causes of issues. Such a device allows for rapid identification of vehicle design issues, improving product quality and expediting the testing process.

Hyster-Yale already employs an existing third-party CAN logging solution. However, the current solution is expensive - incurring both high initial fees for the hardware and high recurring fees for software licenses. This project aims to provide an inexpensive alternative, minimizing costs while striving to match the same standard of quality as the existing solution. The new solution is fast, accurate, and reliable. Compared to the existing solution, this design improves on repairability, maintainability, and extensibility.

1.2 Team Communication Protocols and Standards

Maxim Feoktistov	feoktism@oregonstate.edu	System integration & PCB design
Anton Liakhovitch	liakhova@oregonstate.edu	Firmware
Ashley Reid	reidash@oregonstate.edu	CAD and PCB soldering
Ryan Dillard	dillardr@oregonstate.edu	Code and Circuitry wiring

Table1: Team Protocols and Standards

Topic	**Protocol**	**Standard**
Task Management	Team will use Trello for task assignment and record of completion.	During team meetings, the team will review tasks to be completed and assign out cards that represent these tasks in Trello. When a task is complete, individuals responsible will move it to the “completed” stack.
On-time Deliverables	Each team member shall complete their weekly tasks by the deadline.	It is the responsibility of each team member to ask for help if they are not on track and be honest with the team about their progress.
Work Quality	Each team member is expected to produce professional quality work.	If a task is completed in poor quality(does not meet most requirements), the team must vote if they should revise that task.
Communication	Team will use Discord to communicate. There will be weekly in person meetings.	It is the responsibility of each team member to attend every meeting and actively participate in chat.If a team member cannot attend a meeting, they should notify the team before the start of the meeting
Team Integrity	Teammates are honest with their individual progress.	Everytime the team asks for updates, each teammate will honestly discuss their progress so far, whether or not they are behind. This is to prevent any further miscommunication.
Work Load	Each team member will be assigned an equal workload	When dividing work, the team will estimate how much time each member's portion will take. The team will make sure that these time estimates are comparable, with no more than 20% deviation.
Collaboration	Team members help each other in a productive manner, without overstepping boundaries.	When providing assistance on another team member's task, team members will either wait until asked for assistance or will politely offer help. Help can be refused at any time.

Communication Analysis

Project Partner Information and Project expectations:

- *Project Partners' interest in the project and main role(s):*
  - Obtain a more efficient CAN logger to save time and money.

- *Project Partners' profession/company:*
  - Electrical engineer in Hyster-Yale Materials Handling.

- *Main types of information the Project Partners will want to know and why:*
  - Weekly updates about the project to make sure progress is being made.

- *Project Partners' level of technical knowledge and terminology related to your project:*
  - New to the project but has experience with the product we want to replicate at a lower cost.

- *Preferred format and frequency for communication of various types of information:*
  - Have weekly meetings as needed, over Zoom. Communication preferred through email.

1.3 Gap Analysis

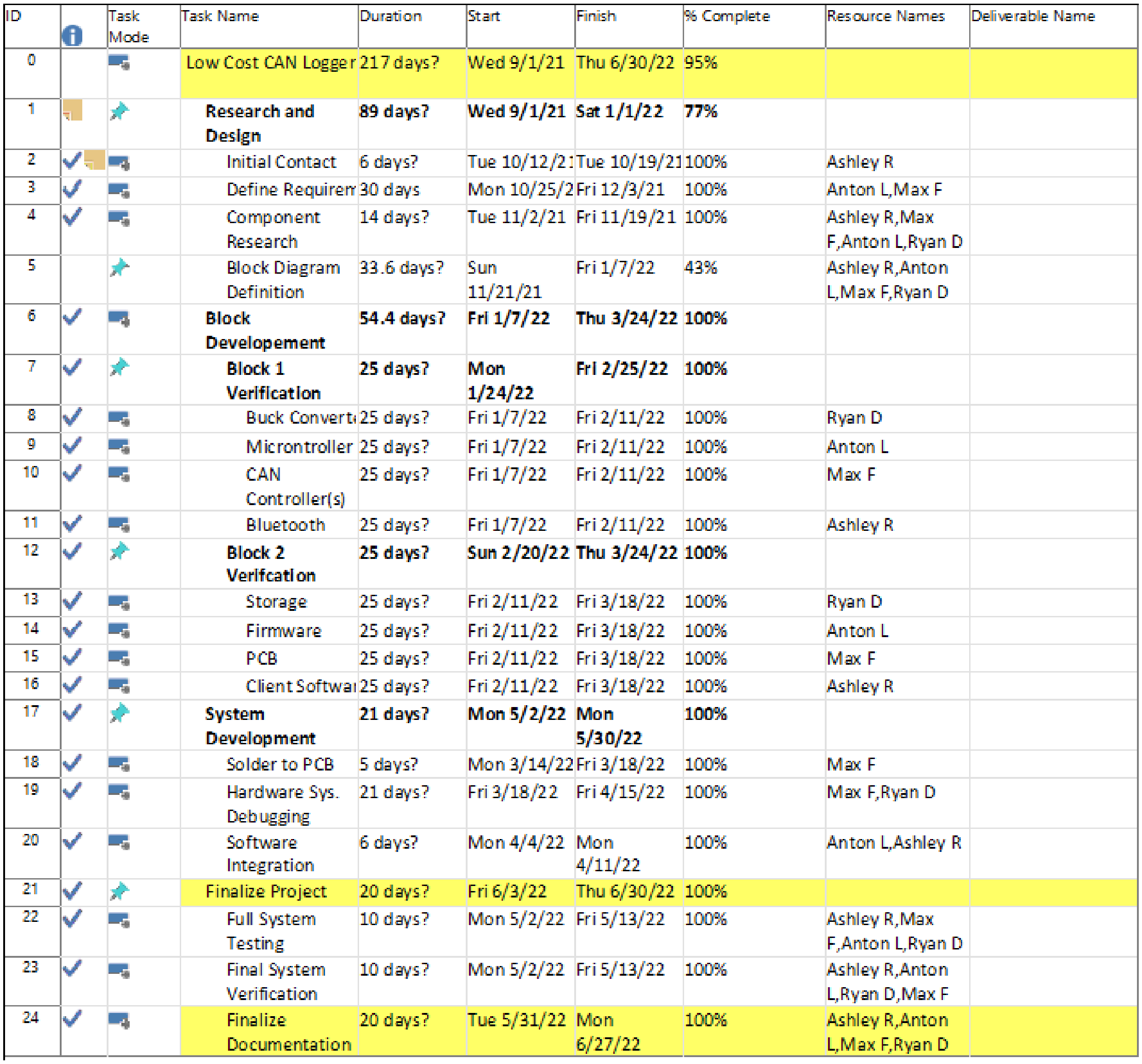
CAN loggers are used by companies to record messages sent on the CAN bus of a vehicle or lift truck. They are a common tool in industrial and design environments. Engineers often need to analyze machine behavior after logging information from a lift truck with the logger attached [1]. Hyster-Yale's current solution costs more than $2500 per unit and requires expensive licenses for software tools.

In conversations with the project partner, issues with Hyster-Yale's current solution were highlighted. The difficulty of retrieving data was a particularly pertinent problem. The project partner highlighted the high cost to acquire the system, and continued costs to use it. Another issue with the system was the limited ability to interact with and fix the device. These tools are proprietary, making the system a black box to the user. This creates an opening for a low-cost alternative that has similar functionality, while giving more freedom to the users and better fitting their needs. The Low Cost Can Logger is estimated to cost under $100 per unit.

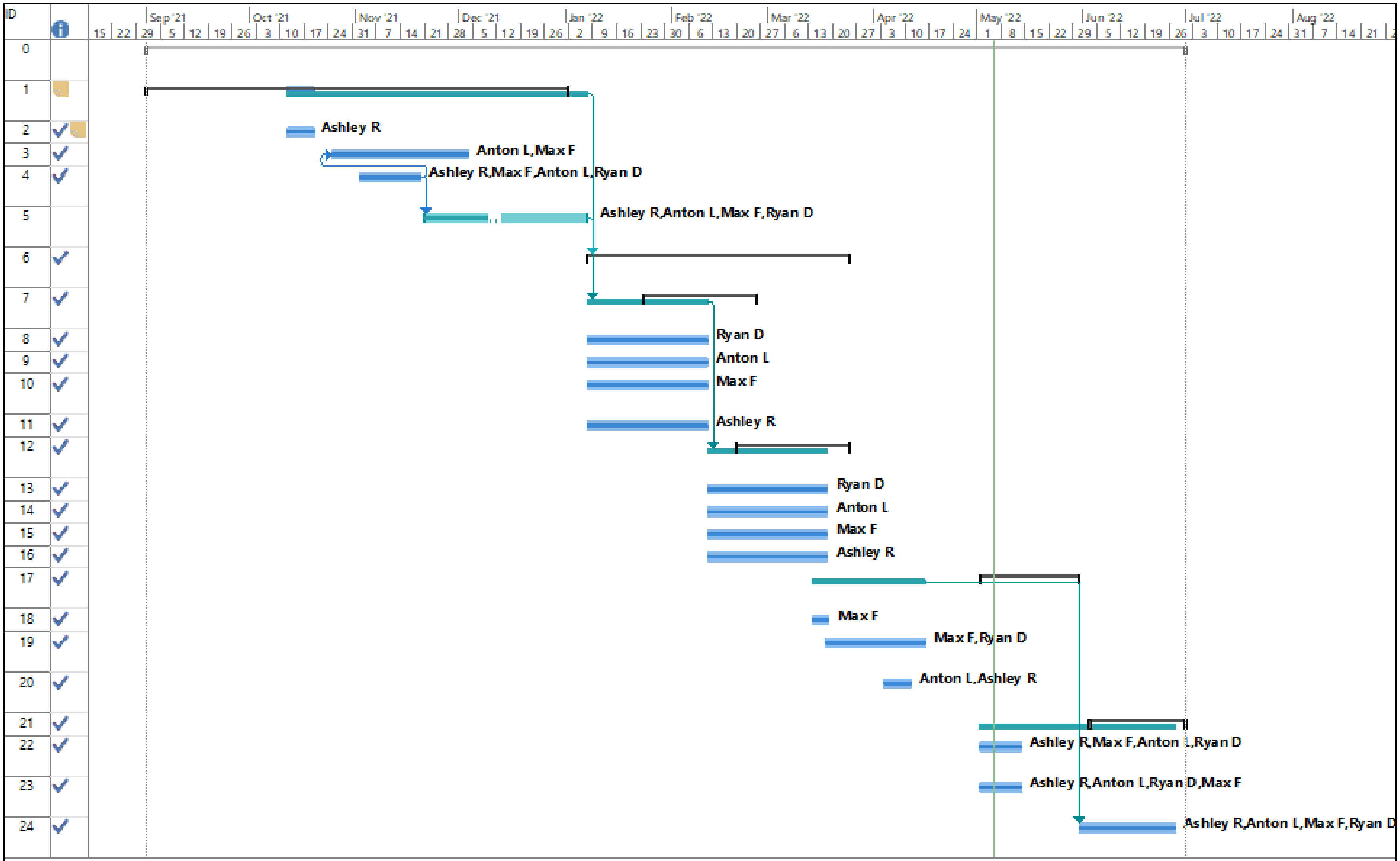
The primary end users are engineers and technicians at Hyster-Yale. This group of end users determine the needs which influence the direction of the project. For instance, the project must be able to reliably record data over extended periods of time. Engineers and technicians use this data for testing and diagnostics, requiring the CAN logging solution to be reliable. To meet reliability standards, it needs to work in rugged conditions around water and dirt, and handle a fluctuating power supply. In addition, it is currently time consuming for Hyster-Yale technicians to retrieve and access data from the existing CAN logging solution. The proposed solution thus includes wireless data offload support. Since Hyster-Yale's requirements for a CAN logging system are reasonable for any automotive manufacturer, the Low Cost Can Logger may also attract other customers in the industry.

1.4 Project Timeline

The table below displays the data used to create the timeline. The highlighted bars in the table display the critical path seen in the timeline visual show after. The deliverables in the spreadsheet coordinate with a bar on the chart as well.



ID	ⓘ	Task Mode	Task Name	Duration	Start	Finish	% Complete	Resource Names	Deliverable Name
0			Low Cost CAN Logger	217 days?	Wed 9/1/21	Thu 6/30/22	95%		
1			**Research and Design**	**89 days?**	**Wed 9/1/21**	**Sat 1/1/22**	**77%**		
2	✓		Initial Contact	6 days?	Tue 10/12/2:	Tue 10/19/21	100%	Ashley R	
3	✓		Define Requirem	30 days	Mon 10/25/2	Fri 12/3/21	100%	Anton L,Max F	
4	✓		Component Research	14 days?	Tue 11/2/21	Fri 11/19/21	100%	Ashley R,Max F,Anton L,Ryan D	
5			Block Diagram Definition	33.6 days?	Sun 11/21/21	Fri 1/7/22	43%	Ashley R,Anton L,Max F,Ryan D	
6	✓		**Block Developement**	**54.4 days?**	**Fri 1/7/22**	**Thu 3/24/22**	**100%**		
7	✓		**Block 1 Verification**	**25 days?**	**Mon 1/24/22**	**Fri 2/25/22**	**100%**		
8	✓		Buck Convert	25 days?	Fri 1/7/22	Fri 2/11/22	100%	Ryan D	
9	✓		Microntroller	25 days?	Fri 1/7/22	Fri 2/11/22	100%	Anton L	
10	✓		CAN Controller(s)	25 days?	Fri 1/7/22	Fri 2/11/22	100%	Max F	
11	✓		Bluetooth	25 days?	Fri 1/7/22	Fri 2/11/22	100%	Ashley R	
12	✓		**Block 2 Verifcation**	**25 days?**	**Sun 2/20/22**	**Thu 3/24/22**	**100%**		
13	✓		Storage	25 days?	Fri 2/11/22	Fri 3/18/22	100%	Ryan D	
14	✓		Firmware	25 days?	Fri 2/11/22	Fri 3/18/22	100%	Anton L	
15	✓		PCB	25 days?	Fri 2/11/22	Fri 3/18/22	100%	Max F	
16	✓		Client Softwai	25 days?	Fri 2/11/22	Fri 3/18/22	100%	Ashley R	
17	✓		**System Development**	**21 days?**	**Mon 5/2/22**	**Mon 5/30/22**	**100%**		
18	✓		Solder to PCB	5 days?	Mon 3/14/22	Fri 3/18/22	100%	Max F	
19	✓		Hardware Sys. Debugging	21 days?	Fri 3/18/22	Fri 4/15/22	100%	Max F,Ryan D	
20	✓		Software Integration	6 days?	Mon 4/4/22	Mon 4/11/22	100%	Anton L,Ashley R	
21	✓		Finalize Project	20 days?	Fri 6/3/22	Thu 6/30/22	100%		
22	✓		Full System Testing	10 days?	Mon 5/2/22	Fri 5/13/22	100%	Ashley R,Max F,Anton L,Ryan D	
23	✓		Final System Verification	10 days?	Mon 5/2/22	Fri 5/13/22	100%	Ashley R,Anton L,Ryan D,Max F	
24	✓		Finalize Documentation	20 days?	Tue 5/31/22	Mon 6/27/22	100%	Ashley R,Anton L,Max F,Ryan D	



Project: Low Cost CAN Logger
Date: Fri 5/6/22

Task		Inactive Milestone	◇	Start-only	[	Path Predecessor Summary Task	
Split		Inactive Summary		Finish-only	]	Path Predecessor Normal Task	
Milestone	◆	Manual Task		External Tasks		Critical	
Summary		Duration-only		External Milestone	◇	Critical Split	
Project Summary		Manual Summary Rollup		Deadline	⬇	Progress	
Inactive Task		Manual Summary		Path Predecessor Milestone Task	◆	Manual Progress	

1.5 References and File Links

1.5.1 References

[1] Steve Corrigan, “Introduction to Controller Area Network (CAN),” *Texas Instruments*, Aug 2009. [Online]. Available: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1635394468633&ref_url=https%253A%252F%252Fwww.google.com%252F

1.5.2 File Links

1.6 Revision Table

Date	Name	Section Revised	What was revised?
10/20/21	Ryan D	1.3 &1.6	Added revision table and started Gap Analysis
10/20/21	Maxim F	1.2 &1.6	Revised revision table and started section 1.2. **Section 1.2 requires Team revision.**
10/21/21	Ashley R	1.4 &1.6	Added Initial Project Timeline
10/28/21	Maxim F	1.2	Revised section headers
11/8/21	Ryan D	1.3	Revised section using instructor, and peer feedback.
11/11/21	Maxim F	1.2 & 1.5	Revised some wording on 1.2 and formatted reference in section 1.5
11/9/21	Ashley R	1.4	Revised Timeline and added more description
11/19/21	Ashely R	1.4	Overhauled timeline layout
5/5/22	Anton L	1.1-1.6	Revised for tense, grammar

2.0 Requirements, Impacts, and Risks

2.1 Requirements

Verification prerequisite:
The verification procedures for several requirements shall depend on a "test bench" device. The test bench shall consist of a microcontroller-based system, capable of sending test data over two CAN channels. The test bench device shall support all protocols and speeds supported by the CAN Logger, unless otherwise stated. The test bench shall be capable of sending any specific, arbitrary set of CAN data.

The requirements are as follows:

2.1.1 Interface Bus

PPR*:* Interface with 2 or more CAN buses.

ER: *The system will support at least two CAN channels*.

Verification procedure:

1. Connect both CAN channels of the CAN Logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously.
3. Run the test bench and record data with the CAN Logger.
4. Verify that the data received by the CAN Logger is the same data sent by the test bench.

Test pass condition: The received data must be identical to the sent data.

2.1.2 Interface Type

PPR: Interface with both J1939 and CANopen.

ER: *The system will log both J1939 and CANopen.*

Verification procedure:

1. Connect both channels of the CAN logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously. The data sequences must use both standard and extended identifiers as required by CANopen and J1939 respectively.
3. Run the test bench and record data with the CAN Logger.
4. Verify that the data received by the CAN Logger is the same data sent by the test bench.

Test pass condition: The received data must be identical to the sent data.

2.1.3 Device Storage

PPR: *Store captured data on an SD card.*

EP: *The system will record data to two separate files.*

Verification procedure*:*

1. Connect both CAN channels of the CAN Logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously.
3. Run the test bench and record data with the CAN Logger.
4. Check the data of the SD card, there should be two files saved on the SD card.

Test pass condition: The SD card must contain two files of CAN data.

2.1.4 Firmware accessibility

PPR: *The user should be able to update the firmware*

EP: *There will be a user guide that will provide information on how to update the system firmware.*

Verification procedure:

1. Share the user guide with the project partner.
2. Get feedback from the project partner.
3. Repeat steps 1 and 2 until the project partner approves the final edition.

Test pass condition: Project partner approves user guide.

2.1.5 Power Supply

PPR: *The device should operate on a voltage range of 8-32v and draw a maximum of 0.375A*

EP: *The system will operate within the following power supply requirements:*
Vmax: 35V
Vmin: 8V
Inominal: 55mA
Ipeak: 500mA

Voltage verification procedure:

1. Use a power supply to power the CAN logger.
2. Set the power supply to 8V and use the test bench to send data to the CAN logger to check for proper operation (the definition of "proper operation" depends on the currently loaded firmware).

3. Set the power supply to 32V and use the test bench to send data to the CAN logger to check the power supply.

Test pass condition: The received data must be identical to the data sent by the test bench.

Current verification procedure:

1. Connect the CAN logger to a power supply.
2. Power the CAN logger from an 8V power supply. Assuming that the device draws a relatively constant amount of power, it will draw the largest current when it is running at the lowest rated voltage.
3. Set the test bench to continuously send random data on both CAN channels. This data will not need to be verified.
4. Run the logger for 30 seconds, taking current draw measurements once every second.

Test pass condition: The average measured current does not exceed Ipeak, and does not deviate from Inominal by more than 10mA for longer than five seconds for the entire test duration.

2.1.6 Client Software

PPR: *Wirelessly transfer collected data to a computer.*

ER: *The system will output a CSV file on the client computer.*

Verification procedure:

1. Connect both CAN channels of the CAN Logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously.
3. Run the test bench and record data with the CAN Logger.
4. Use the CAN Logger PC software solution to download the data from the device over bluetooth. Use either a Windows or Linux system for this test.
5. Via inspection, verify that the data is in proper CSV format.
6. Verify that the received data is the same as the data sent by the test bench.

Test pass condition: The received data must be identical to the sent data when using both Windows and Linux systems.

2.1.7 Invalid Data

PPR: *The device should handle invalid packets without shutting down.*

EP: *The system will handle invalid packets and continue to operate normally.*

Verification procedure:

1. Connect both CAN channels of the CAN Logger to the test bench.

2. Set the test bench to send a sequence containing at least one type of invalid data packets. Invalid packets may include:
    a. Packets that are too short.
    b. Packets that are too long.
    c. Packets sent at the wrong baud rate.
3. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously.
4. Verify that the CAN logger has logged or ignored the invalid data, and continued to log the subsequent valid data.

Test pass condition: The received data must be identical to the sent data, except for the erroneous packets. Erroneous packets must be ignored or marked by an appropriate error message in the CSV file.

2.1.8 Data Accuracy

PPR*: The device should not miss packets.*

ER: *The system will process input data at a rate of at least 1 Mb/s.*

Note: 1 Mb/s refers to the baud rate, not the data rate.

Verification procedure:

1. Connect both CAN channels of the CAN Logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously. The data shall be sent at the maximum possible rate, and shall be of the highest possible complexity. The sequence shall be at least 1MB in size.
3. Run the test bench and record data with the CAN Logger.
4. Verify that the received data is the same as the data sent by the test bench.

Test pass condition: At least 99.9% of received data must be identical to the data sent by the test bench.

2.2 Design Impact Statement

2.2.1 Introduction

This assessment serves to explore the different negative impacts that a CAN Logger could possibly introduce or magnify. As the future electrical engineers, there is a responsibility to consider these impacts as our future designs will have real world impacts. This school year, the team's goal is to produce a lower cost CAN logger that still has the power to perform the same amount of accurate data collection. The team collectively works with Hyster-Yale company to produce their possible best version. The CAN Logger is a universal tool used in many industries, but in Hyster-Yale's case, the CAN logger is used to monitor forklifts. Typically, the actual CAN Logger has more passive actions, as it assists in machine maintenance and provides insight to multiple systems running. The CAN logger is used as a tool for a company's actual products or large devices. However, exploring more into the uses and creation of the CAN logger, the realization was set that it was a pathway for the CAN bus to take over many industries. Researching into the CAN loggers, it's an incredibly diverse tool used into multiple industries. It doesn't stay constrained to just forklift companies. The original design impacts multiple professional sectors. It also never has many insights into how its end of life uses are done. After the initial exploration, the team learned that the CAN logger does have potential negative impacts in the following sectors: public health, safety, and welfare impacts; cultural and social impacts; and lastly, environmental impacts. The rest of the paper will follow this format touching on all of these subjects respectively. The paper will also contain positive impacts that deem important to mention.

2.2.2 Public Health, Safety, and Welfare Impacts

To make a product that is more affordable to produce and maintain, which is the team's main goal, there can be potential sacrifices to it's quality. For example, the team is working to produce a CAN logger at the refraction of the price, including all research and development costs. To cut down this price, it is not possible to pick the components with the absolute highest safety rating. This comes at a price because, as people interact with it everyday to retrieve data, there is a higher risk with more maintenance and everyday usage[8]. Some of these potential risks include: unreliable voltage control, lower quality internal components, and potentially affect other wireless connections. For instance, if there is an edge case that was not tested for a large current pull, there will be a much higher risk of imploding the systems. A lot of internal components,when made at a lower rate, introduce different varieties of problem, the biggest being that it contains lead. Despite low level traces, there are studies that continue to prove the high risks of using leaded components[3].

Moreover, wireless connections, when done improperly, pose a risk to those that require internal technology. Indstrusty technology has shown long term side effects with public citizens who have pacemakers or other internal medical equipment, as these can interfere with the signals being produced by a couple beats. With minimal exposure, this can be a rare off beat signal[9] However, if there user required daily interactions with a device like these, there have been occasions where the internal equipments missed multiples beats during the day or there was long term decrease in pulse accuracy, "especially if it was poorly constructed electrical device" [4] One of the biggest conversations my group has had was on how the system was to transmit data and how much in the budget can be allotted to a higher quality wireless module.

Lastly, when the data loggers are created with a lower cost, shorter time, and confined testing time, there is no way to verify its accuracy over a long period of time. The unknown potential for its reducing accuracy over time is something to consider. Most systems products that require a CAN logger, are cumbersome, such as forklifts, sem trucks, autonomous vehicles. If their lower cost CAN Logger was not accurate after a year and this was not caught by anyone working with these large mechanisms, the safety repression could be disastrous[8]. Despite this limited testing time during the school year, the team has planned to take the difference in accuracy between test periods and to make these long term predictions as accurate as possible. Even though it isn't perfect, any attempt helps try to prevent these accidents. This is a practice many companies attempt in order to create accurate safety readings and warnings. When the project is officially finished by the team, this planned design will continue to be mitigated by the suggestion of our team.

2.2.3 Cultural and Social Impacts

The biggest reason behind this senior design project is the price. The market CAN loggers currently runs for 2500, and the licensed software that interacts with loggers costs at least a thousand dollars more. While this tool is incredible for large companies and ensures the viability of large machinery, it bottlenecks this product away from the rest of the public. The team's current design would reduce the cost of a CAN logger to 50 dollars. Taking into account the company's exact needs and programming non-licensed code. More affordable and accessible options allows smaller companies to reach their goals and have the ability to create and process larger systems for a company of their own size. However, the cheaper CAN loggers would have less reliability, life span, and efficiency, which directly leads to project failure. There was a study that found that collectly, employee's mental health declines proportionally to the success or failure of their work[10]. CAN Loggers have the responsibility of being a successful tool to carry out, and work towards these goals. Despite this, CAN loggers being just a tool compared to an actual product or invention, it directly impacts someone's life. CAN Loggers and their counterpart, can buses have now been so streamlined that they have impacted communities by replacing maintenance jobs and replacing labor.

It could have great impacts as now this type of technology with open source code. If there is a possibility that we can create our technology with open source code instead of restricting it to our company due to the NDA, it could cut down exponentially on usage costs. This would also add to the movement starting to be picked up that all software could be open source, allowing humanity to move forward faster and encourage all of us to work towards the same goal, the opportunity to create a better future. To have the ability to create from open source, it drives down the cost to allow anyone who wants to learn or create. It breaks down social barriers and blurs the lines between different economic status[2]. The team must sign NDAs to develop this project in partnership with the project partner, however something we have had multiple discussions over is the code to extract data from the system. Keeping this open source would not reveal any trade secrets, yet would allow access to this material that could drive down the cost of this tool. It would decrease the gap in who could use this product. Taking it to a bigger scope, it could provide everyday users the tools to better their own lives and the lives of those around them. It shifts the culture of giving these big companies the power, creating more access and practicality. Despite this project having a smaller scope, this could be part of normalizing open-source code, if the project partner agrees.

2.2.4 Environmental Impacts

Reflecting back to the usability of the CAN logger, let's magnify on the lifespan and end of life cycle of data logger products. It is good practice to keep in mind how long these products should last relative to negative impacts these materials of the logger will have . According to a study, this kind of technology should last at least 10 to 15 years[1]. If it was achievable within budget and timeframe, the team could meet this goal. However in reality, when making a cheaper product in a short amount of time, this is not always feasible. The products' microcontroller will stop having updated support within 10 years. One of the team's goals is to be able to make a product that is modularized enough to be able to switch out components as time goes on, and technology improves. It is difficult to prepare for that type of situation, so one of the main goals will be to ensure the code is maintainable and written in a clear and concise style that is easy to understand. So if a product needs to be switched out, only that component piece needs to be thrown away, but the rest of the system can continue to run with a new module.

Besides reducing the amount of waste by modularizing the code, another possible solution would be to at least ensure that whenever the parts do need to be thrown away, the materials it is made with, have a less negative impact on the environment. In the electronics industry, it is extremely hard to create biodegradable waste. This currently can't be a oal, but the waste we do produce is even more harmful to our earth than average .Currently, even though electronic waste only takes responsibility for 2-3% of our landfills[7]. It is responsible for up to 70% of the world's hazardous waste. This is now not just an environmental issue but also a safety issue. Reading the IEE vows, there is a responsibility, as engineers, to prevent as much toxic waste as possible entering our landfills, and then minimizing how much actual waste is created with what engineer's invent. For the team's project, specifically, it is difficult to mitigate the specific materials, as there are current constraints to budget, timeline, and availability for small orders. However, the enclosure for our design will have slimmer design, and be built hopefully with wood so it is made with biodegradable material. On a larger scale, research shows that there is no alternative to actual components, however, companies are practicing a similar technique. Packaging and enclosures are trying to use less material, parking with cardboard instead of plastic. It is the best situation right now in terms of mitigation.

2.2.5 Economic Impacts

These past two years have induced a lot of supply chain issues and cost of living around the world. Reflecting on it, this could be an amazing opportunity to assist this issue. If the team is able to make a successful low cost CAN logger, there would be ways that could allow people using forklifts in warehouses, semi truck drivers, and the maintenance employees working on them all day, to have a much bigger sense of relief and more time to actually get their work done. This could create a positive domino effect.

Also wanting to point back to how a lower cost CAN logger can reduce the gap between two people with much different socioeconomic backgrounds[5]. It allows more and more people to buy these for potentially their own personal products, especially with the cost of living rightnow going for goods. Even though it can be a little bit of a stretch as more things that require CAN loggers are bigger systems done by company, people would have to spend through but rather have a positive impact on the enocy by allowing individuals themselves to purchase it rather than just a company. Going off of that, it has a positive impact on the supply chain issue around the world by making people's labor more efficient and easier.

One big potential downside to the lower cost CAN logger is that, if it makes people's jobs much easier, could it potentially replace the positions of the employees. There would be less need for jobs for hands on tracking of the CAN busses if a CAN data logger can fulfill these roles independently and transmit their data into a database to be handled by another algorithm. There is potential for this to happen as we have seen it in years past. Looking at past solutions, this has happened with big system tractors. Decades ago, there were thousands of people working in the farming industry. Nowadays, technology in tractors has improved so much it has only a fraction of the labor that there once was[4]. This is actually a product that CAN buses and CAN data loggers have had a direct impact in, underlining the absolute possibility. It is saddening but an accurate CAN Logger with a CAN bus being implemented everywhere on the market could drive the forces of technology, especially in autonomous vehicles and autonomous trucks, that the semi truck industry is actually looking to replace some of its labors as long as the autonomous vehicles are accurate enough. It is still in the beginning stages but it could definitely contribute to this fear. New technology is exciting but there is the risk that what if people can find another job that meets their skills but instead take away practically a whole sector of labor.

2.2.6 Conclusion

Reflecting on the possibilities, I think as engineering majors, there is the power to define how much our inventions, breakthroughs, and new technology will impact the earth. Going over our the possible negative design implications, the team should take into these design ideas when building and investing into the low Cost CAN logger:

- Research components and ensure they are sourced properly
- Ensure that design is modularized so items can individually be replaced instead of system
- Continue to ensure the cost to build the system is practical
- Look for ways to lower to hazard waster the system will produce, such as a biodegradable enclosure
- Ensure that the code used for this system is open source
- Make sure that the product is user friendly for the every consumer

It is difficult to imagine something as relatively insignificant a tool could affect our day to day lives, but looking at how tractors took over a whole industry, there are micro plastics invading our oceans, chemical ingested from tainted water due to careless acts done back in the 70's, anything and everything can lead to either a better of world or a worse world. This tool could have the greatest economic impact seen with the example of advanced transportation. Discussing this with my team, it is agreed that this is something that matters, as we just want to make the world a more efficient, better place to live in.

2.2.7 References

[1]A. Brent and C. Labuschagne, "Social Indicators for Sustainable Project and Technology Life Cycle Management in the process industry (13 pp + 4)," *The International Journal of Life Cycle Assessment*, vol. 11, no. 1, pp. 3–15, Jan. 2006. [Online] Available: https://link.springer.com/article/10.1065/lca2006.01.233

[2] B. Fitzgerald, “The transformation of Open source software,” *MIS Quarterly*, vol. 30, no. 3, p. 587, 2006. [Online] Available:https://www.jstor.org/stable/25148740

[3]C. M. L. Wu, D. Q. Yu, C. M. T. Law, and L. Wang, “Properties of lead-free solder alloys with Rare Earth element additions,” Materials Science and Engineering: R: Reports, vol. 44, no. 1, pp. 1–44, 2004.[Online] Available: https://www.sciencedirect.com/science/article/abs/pii/S0927796X04000105

[4]G. L. Smirnov, “Patterns of growth of the working class and change in its composition by trade and Skill,” *Soviet Sociology*, vol. 6, no. 1-2, pp. 25–33, Dec. 2014. [Online] Available: https://doi.org/10.1016/j.mser.2004.01.001

[5]M. Warschauer and T. Matuchniak, “New Technology and Digital Worlds: Analyzing Evidence of equity in access, use, and outcomes,” *Review of Research in Education*, vol.34, no. 1, pp. 179–225, Mar. 2010.[Online] Available: https://www.semanticscholar.org/paper/New-Technology-and-Digital-Worlds%3A-Analyzing-of-in-Warschauer-Matuchniak/afcee00ca804a27e0abc4263fb526532c2b71c22

[6]P. Chowdhury, S. K. Paul, S. Kaisar, and M. A. Moktadir, “Covid-19 pandemic related supply chain studies: A systematic review,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 148, p. 102271, Apr. 2021.[Online] Available: https://pubmed.ncbi.nlm.nih.gov/33613082/

[7]S. Needhidasan, M. Samuel, and R. Chidambaram, “Electronic waste – an emerging. threat to the environment of Urban India,” *Journal of Environmental Health Science and Engineering*, vol. 12, no. 1, 2014.[Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3908467/

[8]T. Kern and L. Willcocks, “Exploring Information Technology Outsourcing Relationships: Theory and practice,” The Journal of Strategic Information Systems, vol. 9, no. 4, pp. 321–350, Dec. 2000.[Online] Available: https://doi.org/10.1016/S0963-8687(00)00048-2

[9]Ø. Michelsen, “Use of reliability technology in the process industry,” Reliability Engineering & System Safety, vol. 60, no. 2, pp. 179–181, May 1998.[Online] Available: https://www.researchgate.net/publication/341606900_Reliability_Analysis_in_Process_Industries-An_Overview

[10]“Devices that may interfere with ICDS and pacemakers,” *www.heart.org*, 30-Sep-2016. [Online].https://www.heart.org/en/health-topics/arrhythmia/prevention--treatment-of-arrhythmia. Accessed: 06-Dec-2021. [Online] Available: https://www.heart.org/en/health-topics/arrhythmia/prevention--treatment-of-arrhythmia/devices-that-may-interfere-with-icds-and-pacemakers

[11]M. Russo, L. Guo, and Y. Baruch, “Work attitudes, career success and health: Evidence from China,” Journal of Vocational Behavior, vol. 84, no. 3, pp. 248–258, 2014.[Online] Available: https://doi.org/10.1016/j.jvb.2014.01.009Get

2.3 Risks

Table 2.1. Risk Assessment and Action Plans

Risk ID	Risk Description	Risk Category	Risk Probability	Risk Impact	Performance indicator	Responsible party	Action Plan
R1	Part availability [1]	Technical Schedule	50%	High	Seller stock levels	Anton Liakhovitch	Retain a list of alternatives part stockers
R2	Covid	Public health Schedule	30%	Medium	Infection rates	Ryan Dillard	Retain alternative meeting plans.
R3	Project partner availability	Organization Political	10%	High	Decreased contact	Ashley Reid	Transfer responsibility. Inform Instructors or get alternate contact
R4	Project falling behind schedule	Organizational Scheduling	50%	Medium	Timelines are starting to slip	Max Feoktistov	Retain times for emergency meetings.
R5	Teammate Fails	Organizational Scheduling	12.5%	High	Grades	Ryan Dillard	Transfer responsibilities
R6	Going Over Budget	Organizational	10%	Low	Budget	Max Feoktistov	Retain an accurate Bill of Materials
R7	Customer Require-ments Change	Organization Scheduling	10%	Medium	Customer Requirements	Ashley Reid	Reduce changes to engineering requirements
R8	Lack of access to tools and information	Technical Political	10%	Low	Ability to test and build system	Anton Liakhovitch	Avoid reliance on one testing method. Retain communication with Project Partner and Instructors

2.4 References and File Links

2.4.1 References

[1] M. Ludwikowshi, W. Sjoberg, “Semiconductor shortage and the U.S. auto industry.” *Reuters*, para. 1, June 22, 2021. [Online]. Available: https://www.reuters.com/legal/legalindustry/semiconductor-shortage-us-auto-industry-2021-06-22/

2.4.2 File Links

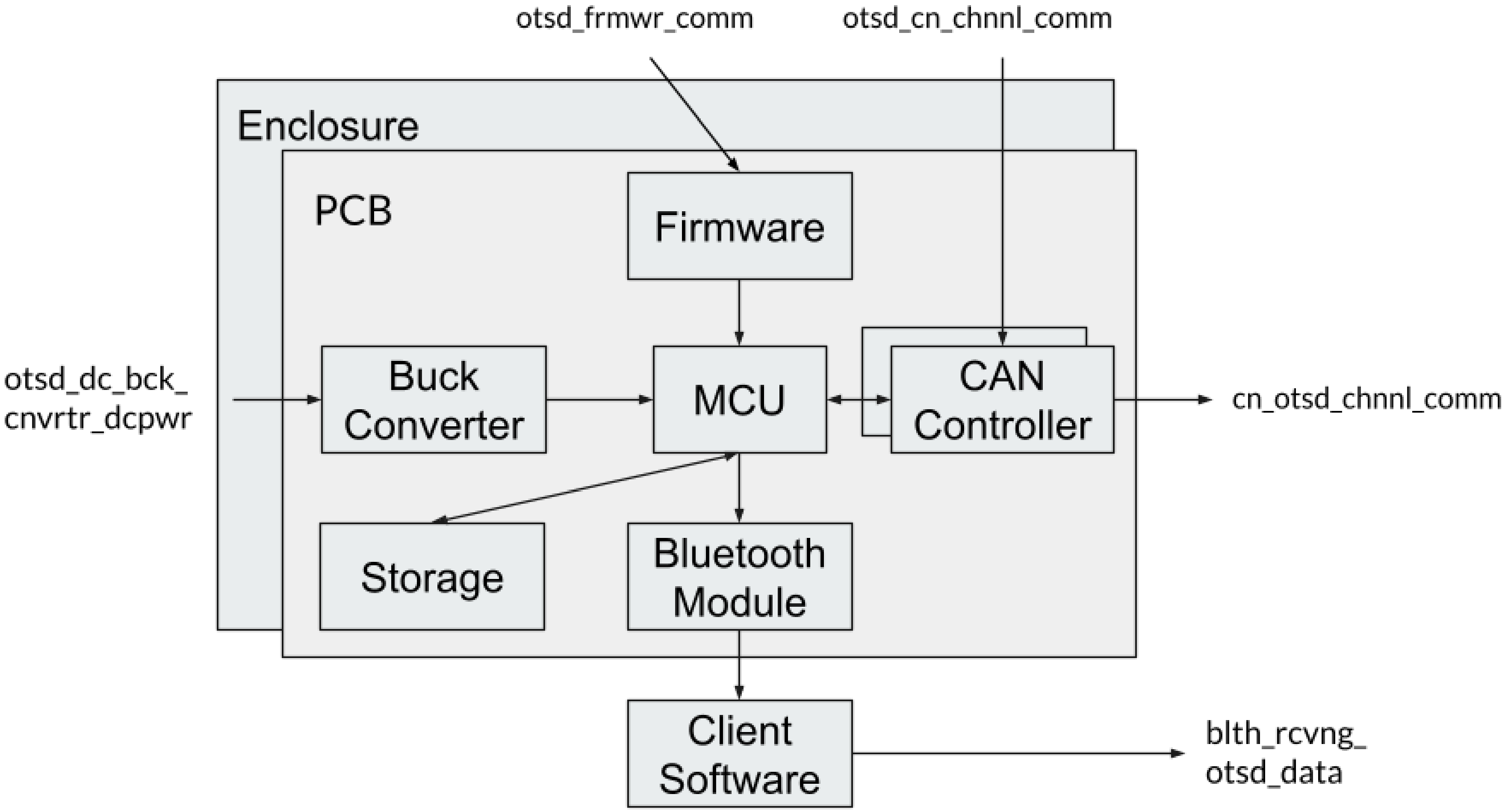
2.5 Revision Table

Date	Name	Section Revised	What was revised?
10/25/21	Ryan D	2.1-2.5	Formatted Section 2,created the revision table, and risk table
10/28/21	Max F	2.1	Created Requirements section, defined requirements.
10/28/21	Anton L	2.1	Edited and defined the requirements
10/28/21	Ryan D	2.3	Filled out the risks table
11/11/21	Ryan D	2.3	Revised and reformatted risks table
11/11/21	Ashley R	2.3	Revised and reformatted risks table
11/11/21	Maxim F	2.1	Revised and reformatted the requirements section
11/18//21	Maxim F	2.1	Revised requirements based on feedback
11/18/21	Anton L	2.1	Revised requirements based on feedback
12/02/21	Anton L	2.1	Replaced "bluetooth" requirement with "firmware updatability" requirement
10/29/21	Ashley Reid	2.2	Initial Draft
11/13/21	Ashley Reid	2.2	Added two additional resources
12/1/21	Ashley Reid	2.2	Revised on specific concerns from Rachel
12/4/21	Ashley Reid	2.2	Overall Document Revision
12/5/21	Ashley Reid	2.2	Final Format and Spellcheck
5/5/22	Ashley Reid	2.2	Extra Revisions after grading

*Please **only** include notable revisions to Section 2 and any tasks that need to be done to the Section

3.0 Top-Level Architecture

3.1 Block Diagram



3.2 Block Descriptions

3.2.1 Buck Converter

The MCU and other modules in the project require a lower voltage than what the vehicle supplies to the system. The Logger accepts an input voltage between 8-32V. The buck converter regulates the voltage down to two voltages: 5V and 3.3V. This powers the MCU and other modules. Ryan Dillard is in charge of developing this block.

3.2.2 MCU

The microcontroller is an STMicroelectronics STM32f411CEU6. In order to perform all necessary functions in the CAN logger system, an MCU must meet a specific set of requirements. It must have at least three SPI interfaces - one for the storage module and one for each CAN transceiver. It must have a maximum clock speed of at least 4MHz, so that it can process data at maximum speed from both CAN interfaces and send that data to storage in real time. It must have a UART for interfacing with the Bluetooth module. Finally, the MCU must accept an input voltage of either 5V or 3.3V. The STM32F411CEU6 meets all of these requirements. Anton Liakhovitch is responsible for this block.

3.2.3 MCU Firmware

The MCU firmware consists of two parts. The first is a critical section which collects CAN data from the CAN controller and records it on the storage medium. This section is optimized as much as possible for stable real-time performance. The other part of the firmware handles data upload via Bluetooth. This section runs while the critical section is stopped, and has no hard performance requirements. Anton Liakhovitch is responsible for this block.

3.2.4 CAN Controller/Transceiver

The CAN Controller/Transceiver block is responsible for Receiving and Transmitting CAN signals and communicating with the MCU. The MCP2515 CAN controller is being used to interpret the CAN messages and send the packages to the MCU in a package that the MCU can interpret. Maxim Feoktistov is responsible for the CAN Controller/Transceiver block.

3.2.5 Storage

Messages sent over the CAN bus must be stored to use in analysis later. The MCU transmits data to the storage medium via SPI. The data is stored on an SD card at 25MHz with the SPI connection. Data is stored as text in the csv file format. Ryan Dillard is in charge of developing this block.

3.2.6 Bluetooth Module

The customer requires the capability to wirelessly transfer data from the CAN logger to a computer. A bluetooth module allows a user to collect the data from a system without worrying about security issues. This project uses the HC-06. Ashley Reid is in charge of developing this block.

3.2.7 Client Software

Although the Bluetooth interface works standalone with standard serial terminal applications, a graphical client program makes the Bluetooth interface easier to use. The client software allows the user to download data from the CAN Logger and set various parameters. Ashley Reid is in charge of this block.

3.2.8 PCB

The system contains a single PCB that integrates all the components. EAGLE is used to create the schematic containing all the components of the system and design a PCB board.The PCB design is then ordered at OSHPark.com. Maxim Feoktistov will be responsible for the PCB block.

3.2.9 Enclosure

This system will be used in a rugged environment and building an enclosure ensures that it has a form of protection. This can potentially increase the longevity of the system. Additionally, an enclosure increases the professionalism of the project when presenting to our project partners. It is manufactured with 3D-printed PLA. Ashley Reid is in charge of this block.

3.3 Interface Definitions

Interface Name	Specifics
otsd_bck_cnvrtr_dcpwr	● Vin_max: 35V ● Vin_min: 8V ● I_peak: 500mA ● I_nominal: 375 mA ● Other: Barrel Jack
otsd_cn_cntrllrtrnscvr_comm	● Baud Rate= 250K/500K ● CAN Protocol [1] ● Logic Level High: 3.75 V ● Logic Level Low: 1.25 V
bck_cnvrtr_bltth_dcpwr	● Ipeak: 100mA ● I_nominal: 30mA ● Vmax: 3.5V ● Vmin: 3.0V
bck_cnvrtr_cn_cntrllrtrnscvr_dcpwr	● Ipeak: 100mA ● Inominal: 65mA ● Vmax: 3.5V ● Vmin: 3.0V
bck_cnvrtr_strg_dcpwr	● Ipeak: 100mA ● Inominal: 30mA ● Vmax: 3.5V ● Vmin: 3.0V
bck_cnvrtr_mc_dcpwr	● Ipeak: 100mA ● Inominal: 10mA ● Vmax: 3.5V ● Vmin: 3.0V[2]
frmwr_mc_data	● C Language ● Architecture: ARM Cortex M4 with DSP extensions ● Code Size: 512kB max ● Configures: USART & SPI
otsd_frmwr_comm	● Serial Wire Debug interface ● Logic Level: 0V Low ● Logic Level: 3.3V High ● Maximum CLK: 50MHz
bltth_clnt_sftwr_data	● Communication with Bluetooth driver ● Virtual RS-232 interface ● Data: ASCII data in CSV format

	● Command set: AT-style commands from client
bltth_mc_comm	● RS-232 serial protocol ● Baud rate = 9600 ● Logic Level: 0V Low ● Logic Level: 3.3V High
cn_cntrllrtrnscvr_otsd_comm	● Baud Rate=250K/500K ● CAN Protocol [1] ● Logic Level High: 3.75 V ● Logic Level Low: 1.25 V
cn_cntrllrtrnscvr_mc_comm	● SPI ● Data is received by reading 8 byte registers ● 10Mb/s maximum speed ● Logic Level: 0V Low ● Logic Level: 3.3V High
strg_mc_data	● SD over SPI ● Logic Level: 0V low ● Logic Level: 3.3V High ● 10Mb/s maximum speed
clnt_sftwr_otsd_data	● Output: text file written to client filesystem ● Text file in CSV format ● LF line endings ● UTF-8 encoding ● File saved with RW permissions for current user and group
mc_bltth_comm	● RS-232 serial protocol ● Baud rate = 9600 ● Logic Level: 0V Low ● Logic Level: 3.3V High
mc_cn_cntrllrtrnscvr_comm	● SPI ● Logic Level: 0V Low ● Logic Level: 3.3 High ● 10Mb/s maximum speed ● Data is received by reading 8 byte registers
mc_strg_data	● SD over SPI ● Logic Level: 0V low ● Logic Level: 3.3V High ● 10Mb/s maximum speed

3.4 References and File Links

3.4.1 References

[1]Steve Corrigan, “Introduction to Controller Area Network (CAN),” *Texas Instruments*, Aug 2009. [Online]. Available: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1635394468633&ref_url=https%253A%252F%252Fwww.google.com%252F

[2] “STM32F411xC STM32F411xE,” *ST life.augmented,* Dec 2017. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f411re.pdf

CAN controller transceiver datasheet
https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf

3.4.2 File Links

3.5 Revision Table

Date	Name	Section Revised	What was revised?
11/15/21	Ryan D	3.1-3.5	Formatted Section 3,created the revision table
11/18/21	Max F	3.2-3.3	Created and Filled out Interface Table, added block description
11/18/20	Ashley R	3.2	Added personal block description
11/18/21	Ryan D	3.3	Added interface properties
11/18/21	Anton Liakhovitch	3.2-3.3	Added info on part descriptions, added interface properties
12/2/21	Anton Liakhovitch	3.1 & 3.3	Revised interface properties, changed block diagram formatting
12/2/21	Ryan D	3.3	Revised interface properties
12/2/21	Max F	3.3	Revised interface properties
5/5/22	Anton L	3.1-3.5	Edited for tense and grammar

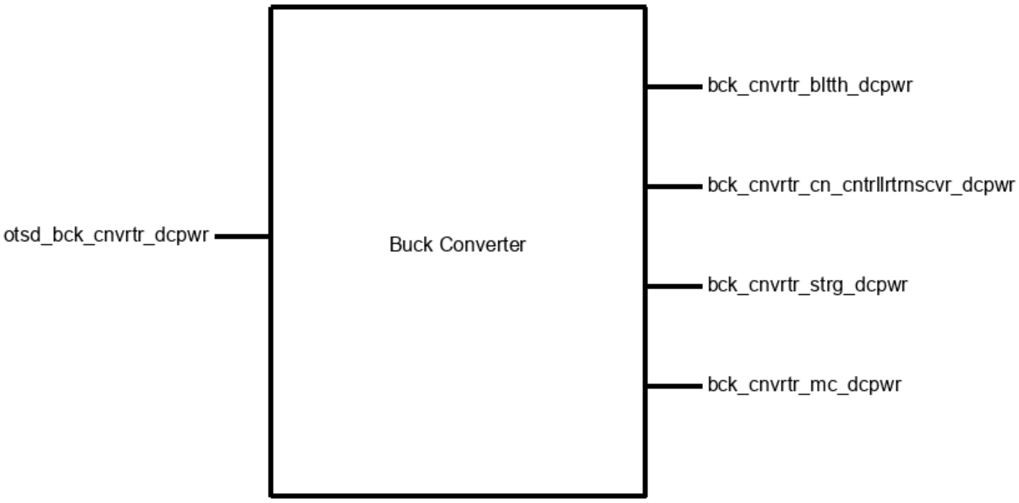
4.0 Block Validations

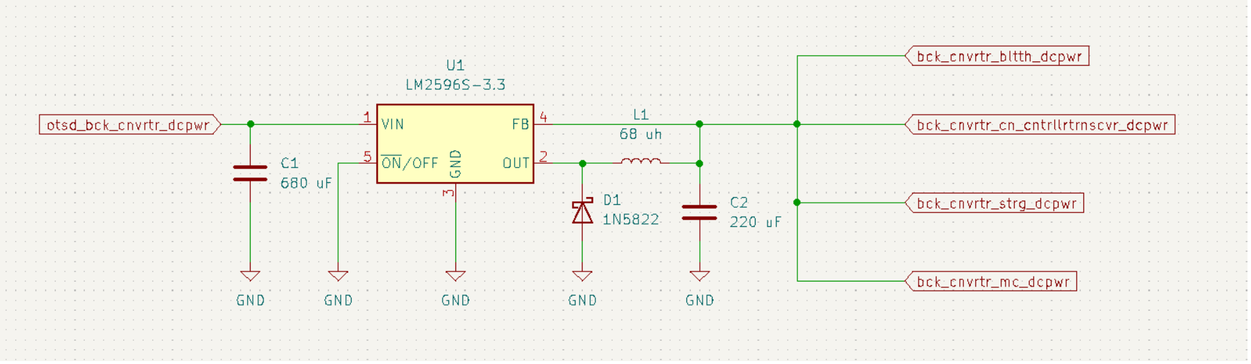
4.1 Buck Converter

4.1.1 Description

The MCU and other modules in the project require a lower voltage than will be supplied to the system. The Logger accepts a DC voltage between 8-32V from the vehicle. The buck converter regulates the voltage down to 3.3V to power the MCU and other modules. Ryan Dillard championed the development of this block.

4.1.2 Design





1. C1 needs to be an electrolytic capacitor able to handle up to 40V for the system to work.

4.1.3 General Validation

The design of this block is guided by the system power requirements. This block is the power regulator for the system. It supplies 3.3V to all other components in the system.

Additionally it will be able to take an input voltage ranging from 8V up to 35V. This power supply will be at least 65% efficient if you calculate the Watts Out vs the Watts into the system.

To achieve this, the design uses the LM2596 simple switcher step-down voltage regulator. This meets the main system power requirement by supporting input voltage ranges from 4.75V to 40V with up to 3A of load. Additionally, the 3.3V version handedly meets the efficiency requirements with 73% efficiency if it's under a 3A load with an input voltage of 12V.

4.1.4 Interface Validation

Table 1: Interface Property Validation otsd_bck_cnvrtr_dcpwr

Interface Property	Why the value was used	Details that support this value
Inominal: 375mA	This current was chosen using the power limit of 3W for the system. This would be the current highest regular current.	For the LM2596 in the TO-263 package: ● Component is rated to 3A (Electrical Characteristics – 3.3-V Version, pg. 6) [1] ●
Ipeak: 500mA	During peak usage if the system becomes a 4W load this would be the highest current it would need to handle.	For the LM2596 in the TO-263 package: ● Component is rated to 3A (Electrical Characteristics – 3.3-V Version, pg. 6) [1] ●
Vmax: 35V	This voltage was chosen as being able to run off 35V maximum was a system requirement.	For the LM2596 in the TO-263 package: ● Max voltage input is 40V (Electrical Characteristics – 3.3-V Version, pg. 6) [1] ●
Vmin: 8V	This voltage was chosen as being able to run off 8V minimum was a system requirement.	For the LM2596 in the TO-263 package: ● Min voltage input is 4.75V (Electrical Characteristics – 3.3-V Version, pg. 6) [1] ●

Table 2: Interface Property Validation bck_cnvrtr_bltth_dcpwr

Interface Property	Why the value was used	Details that support this value
Inominal: 30mA	This value was chosen based on the	For the HC-05: ● The operating current of the device is 30mA

	bluetooth module device operating current.	
Ipeak: 100mA	This was chosen by adding a safety margin to the current and then doubling the current. We don't expect the bluetooth module to ever need this much current, but we should have the ability to supply it.	For the HC-05: ● The operating current of the device is 30mA
Vmax: 3.5V	This was chosen based on the potential variation of the Buck Converter plus 100mV.	For the LM2596 in the TO-263 package: ● Max output voltage is 3.432V (Electrical Characteristics – 3.3-V Version, pg. 6) [1]
Vmin: 3.0V	This was chosen based on the potential variation of the Buck Converter minus 100mV.	For the LM2596 in the TO-263 package: ● Min output voltage is 3.168V (Electrical Characteristics – 3.3-V Version, pg. 6) [1]

Table 3: Interface Property Validation bck_cnvrtr_cn_cntrllrtrnscvr_dcpwr

Interface Property	**Why the value was used**	**Details that support this value**
Inominal: 60mA	This value was chosen based on the operating current of the 2 CAN controllers, 2 CAN transceivers and a voltage boost regulator	For the MCP2515: ● The operating current is 10mA [2] For the MCP2551: ● The operating current is 10mA [4]
Ipeak: 100mA	This value was chosen based on the operating current of the 2 CAN controllers, 2 CAN transceivers and a voltage boost	For the MCP2515: ● The operating current is 10mA [2] For the MCP2551: ● The operating current is 10mA [4]

	regulator.We don’t expect the module to ever need this much current, but the block should have the ability to supply it.	
Vmax: 3.5V	This was chosen based on the potential variation of the Buck Converter plus 100mV.	For the LM2596 in the TO-263 package: ● Max output voltage is 3.432V (Electrical Characteristics – 3.3-V Version, pg. 6) [1]
Vmin: 3.0V	This was chosen based on the potential variation of the Buck Converter minus 100mV.	For the LM2596 in the TO-263 package: ● Min output voltage is 3.168V (Electrical Characteristics – 3.3-V Version, pg. 6) [1]

Table 4: Interface Property Validation bck_cnvrtr_strg_dcpwr

Interface Property	Why the value was used	Details that support this value
Inominal: 30mA	Based on 10Mb/s write speed	From SD standard ● At Max write SD card would use 0.33W at 50Mb/s
Ipeak: 100mA	Specified as max by SD standard	For the STM32f411CEU6 in the LQFP64 package: ● Max clock speed supports 50Mb/s [1] From SD standard ● At Max write SD card would use 0.33W at 50Mb/s
Vmax: 3.5V	This was chosen based on the potential variation of the Buck Converter plus 100mV.	For the LM2596 in the TO-263 package: ● Max output voltage is 3.432V (Electrical Characteristics – 3.3-V Version, pg. 6) [1]
Vmin: 3.0V	This was chosen based on the potential variation of the Buck Converter minus 100mV.	For the LM2596 in the TO-263 package: ● Min output voltage is 3.168V (Electrical Characteristics – 3.3-V Version, pg. 6) [1]

Table 5: Interface Property Validation bck_cnvrtr_mc_dcpwr

Interface Property	Why the value was used	Details that support this value
Inominal: 10mA	Based on the expected current needs of the MCU	For the LM2596 in the TO-263 package: ● Component is rated to 3A (Electrical Characteristics – 3.3-V Version, pg. 6)[1] For the STM32f411CEU6 in the LQFP64 package: ● Runs at 9.4mA at 50MHz [3]
Ipeak: 100mA	Based on maximum running power of MCU with extra features in used multiplied by 2	For the LM2596 in the TO-263 package: ● Component is rated to 3A (Electrical Characteristics – 3.3-V Version, pg. 6)[1]
Vmax: 3.5V	This was chosen based on the potential variation of the Buck Converter plus 100mV.	For the STM32f411CEU6 in the LQFP64 package: ● The MCU can work on 1.7 to 3.6V (pg. 1)[3] For the LM2596 in the TO-263 package: ● Max output voltage is 3.432V (Electrical Characteristics – 3.3-V Version, pg. 6)[1]
Vmin: 3.1V	This was chosen based on the potential variation of the Buck Converter minus 100mV.	For the STM32f411CEU6 in the LQFP64 package: ● The MCU can work on 1.7 to 3.6V (pg. 1)[3] For the LM2596 in the TO-263 package: ● Min output voltage is 3.168V (Electrical Characteristics – 3.3-V Version, pg. 6)[1]

4.1.5 Verification Process

For input interfaces:

1. Set up the circuit for the buck converter
2. Connect the circuit to a power supply, and to a variable load generator at 200mA load.
3. Apply a load to the voltage regulator, and sweep the voltage from the power supply from 8V to 35V.
4. Use a multimeter to measure the output voltage.
5. The otsd_bck_cnvrtr_dcpwr interface passes the output and stays within the 3.0V to 3.5V range, and the input current doesn’t exceed nominal.
6. Adjust the load so that the block will operate for extended periods with 375mA input current, and 500mA peak input current.

For output interfaces:

1. Set up the circuit for the buck converter
2. Connect the circuit to a power supply, and to a variable load generator.
3. Adjust load to match the sum of nominal currents from bck_cnvrtr_mc_dcpwr, bck_cnvrtr_strg_dcpwr, bck_cnvrtr_cn_cntrllrtrnscvr_dcpwr, and bck_cnvrtr_bltth_dcpwr at 190mA.
4. Sweep the input voltage from 8V to 35V.
5. Measure the output voltage to see if it stays in the 3.0V to 3.5V range.

6. If the output voltage can stay in the 3.0V to 3.5V range and maintain nominal current output the interfaces pass.
7. Adjust input voltage to 12V
8. Adjust the load to the sum of peak currents from bck_cnvrtr_mc_dcpwr, bck_cnvrtr_strg_dcpwr, bck_cnvrtr_cn_cntrllrtrnscvr_dcpwr, and bck_cnvrtr_bltth_dcpwr at 500mA.
9. Measure output voltage to see if it stays in the 3.0V to 3.5V range for 1 second.
7. If the output voltage can stay in the 3.0V to 3.5V range and maintain peak current output the interfaces pass.

For power efficiency:

1. Set the circuit up for the buck converter.
2. Connect the circuit to a power supply and to a variable load generator.
3. Set input voltage to 8V
4. Set load to 200mA
5. Measure the input current and calculate the input power.
6. Measure the output voltage and calculate the output power.
7. Divide the input power by the output power to get a rough efficiency.
8. Repeat with input voltage at 35V
9. This passes the power constraint if efficiency is above 65% for both min and max voltages.

4.1.6 References and File links

4.1.6.1 References

[1] “LM2596 Simple Switcher,” *Texas Instruments*, November 1999, [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2596.pdf?ts=1642779956402&ref_url=https%253A%252F%252Fwww.bing.com%252F

[2] “Stand-Alone CAN Controller with SPI Interface,” *Microchip*, [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf

[3] “STM32F411xC,” *STm*, December 2017, [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f411re.pdf

[4] “High-Speed CAN Transceiver,” *Microchip*, [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/20001667G.pdf

4.1.6.2 File Links

4.1.7 Revision Table

Date	Name	Section	Revision

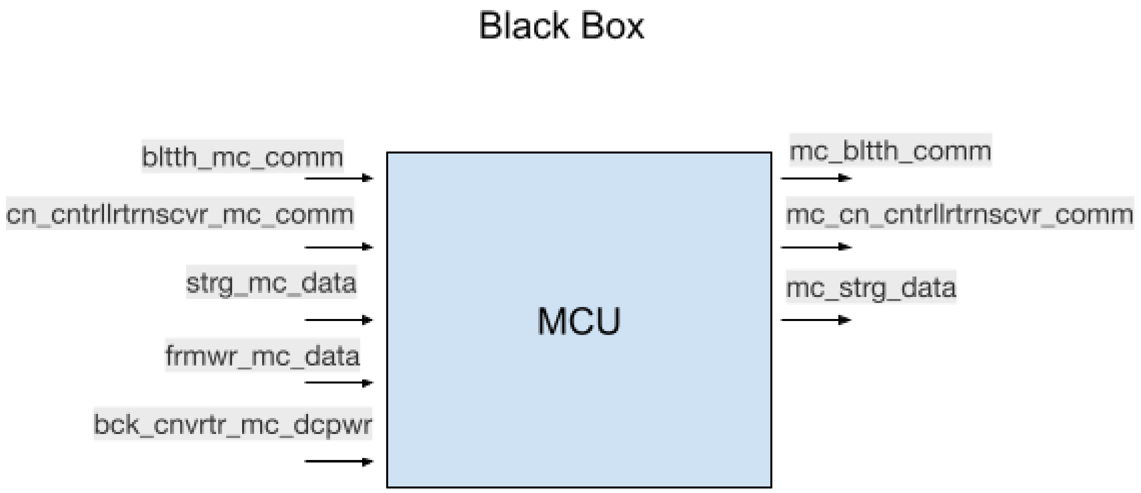
1/7/2022	Ryan Dillard	4.1.1-4.1.7	Created document and wrote rough content for each section
1/19/2022	Ryan Dillard	4.1.1	Edited section to match current needs of the system, and removed old information
1/20/2022	Ryan Dillard	4.1.6	Fixed references and file links to be in the correct positions
1/21/2022	Ryan Dillard	4.1.4	Edited interface properties after communication with the team about updating the needs.
1/21/2022	Ryan Dillard	4.1.5	Clarified methods to show that currents were being verified with a variable load, and the system is expecting all blocks to interfaces to be operating at once.

4.2 MCU

4.2.1 Description

The MCU (microcontroller unit) processes incoming data, stores it, and communicates with client computers. The part chosen for this design is an STMicroelectronics STM32f411CEU6. In order to perform all necessary functions in the CAN logger system, an MCU must implement several specific features. It must have at least three SPI interfaces - one for the storage module and one for each CAN transceiver. It must have a maximum clock speed of at least 4MHz, so that it can process data at maximum speed from both CAN interfaces and send that data to storage in real time. It must have a UART for interfacing with the Bluetooth module. Finally, the MCU must accept an input voltage of either 5V or 3.3V. The STM32F411CEU6 meets all of these requirements. Anton Liakhovitch is responsible for this block.

4.2.2 Design

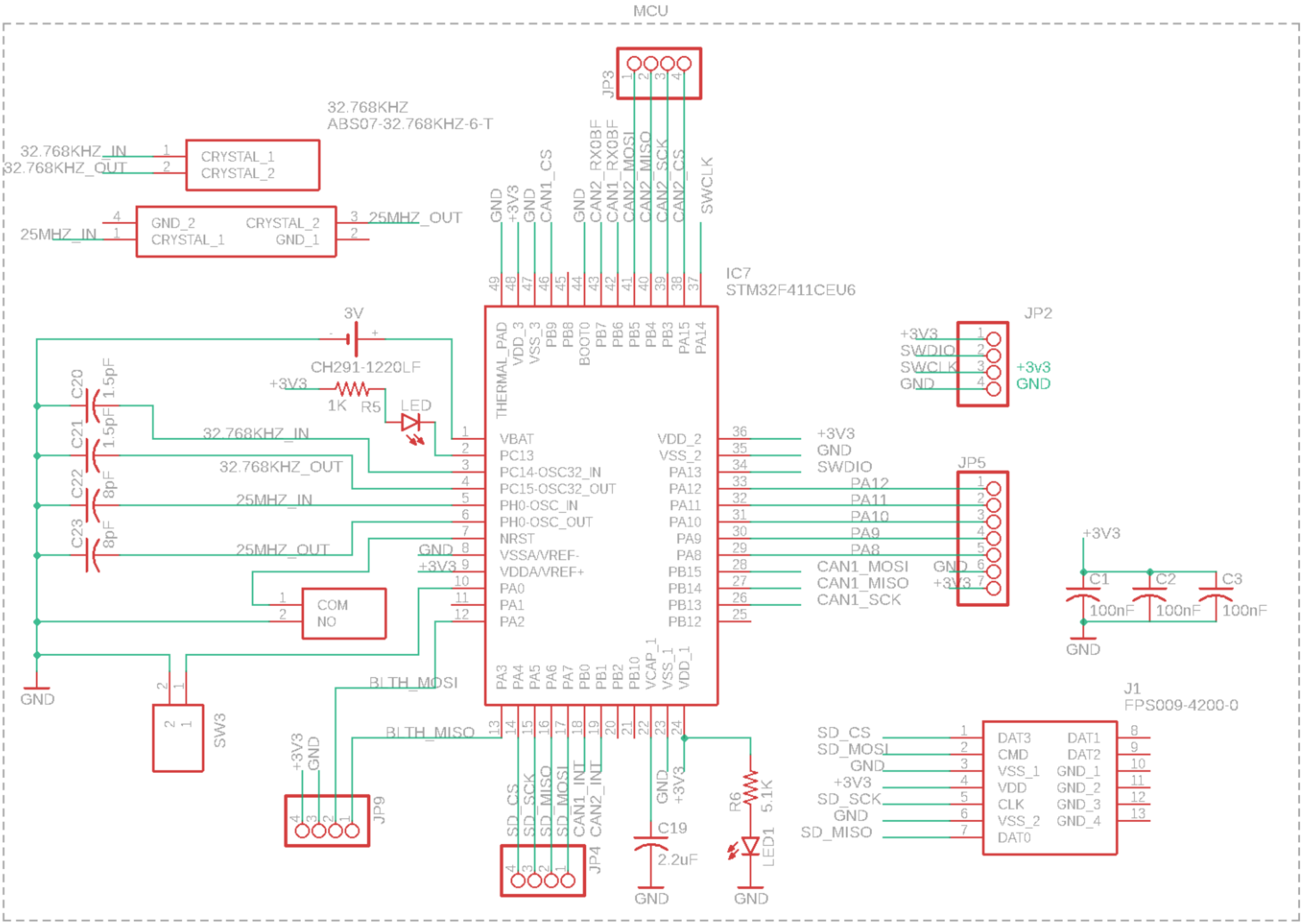


The MCU hardware block design is heavily based on a development board by WeAct Studio. Unneeded parts, such as the analog voltage reference circuitry, are removed.

Schematic (next page):

Ground connections between all blocks are assumed.

Additionally, the following inter-block connections are made. Note that connections are labeled by connecting block, not interface - this is because some connections (for instance, SCK) may belong to multiple interfaces.


MCU
JP3
1 2 3 4
32.768KHZ
ABS07-32.768KHZ-6-T
32.768KHZ_IN
32.768KHZ_OUT
CRYSTAL_1
CRYSTAL_2
GND_2
CRYSTAL_1
CRYSTAL_2
GND_1
25MHZ_IN
25MHZ_OUT
GND
+3V3
GND
CAN1_CS
GND
CAN2_RX0BF
CAN1_RX0BF
CAN2_MOSI
CAN2_MISO
CAN2_SCK
CAN2_CS
SWCLK
IC7
STM32F411CEU6
THERMAL_PAD
VDD_3
VSS_3
PB9
PB8
BOOT0
PB7
PB6
PB5
PB4
PB3
PA15
PA14
VBAT
PC13
PC14-OSC32_IN
PC15-OSC32_OUT
PH0-OSC_IN
PH0-OSC_OUT
NRST
VSSA/VREF-
VDDA/VREF+
PA0
PA1
PA2
VDD_2
VSS_2
PA13
PA12
PA11
PA10
PA9
PA8
PB15
PB14
PB13
PB12
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB10
VCAP_1
VSS_1
VDD_1
3V
CH291-1220LF
+3V3
1K
R5
LED
C20
1.5pF
C21
1.5pF
C22
8pF
C23
8pF
GND
COM
NO
SW3
BLTH_MOSI
BLTH_MISO
+3V3
GND
JP9
SD_CS
SD_SCK
SD_MISO
SD_MOSI
CAN1_INT
CAN2_INT
JP4
GND
+3V3
C19
2.2uF
GND
R6
5.1K
LED1
GND
JP2
+3V3
SWDIO
SWCLK
GND
+3v3
GND
+3V3
GND
SWDIO
JP5
PA12
PA11
PA10
PA9
PA8
CAN1_MOSI
CAN1_MISO
CAN1_SCK
GND
+3V3
+3V3
C1
100nF
C2
100nF
C3
100nF
GND
J1
FPS009-4200-0
SD_CS
SD_MOSI
GND
+3V3
SD_SCK
GND
SD_MISO
DAT3
CMD
VSS_1
VDD
CLK
VSS_2
DAT0
DAT1
DAT2
GND_1
GND_2
GND_3
GND_4

Connecting Block	**Connection**	**MCU Pin**	**Peripheral**
Buck Converter	VCC	VCC	
Bluetooth Module	RX (MCU)	PA3	UART 2
Bluetooth Module	TX (MCU)	PA2	UART 2
Can Controller 1	MISO	PB14	SPI 2
Can Controller 1	MOSI	PB15	SPI 2
Can Controller 1	SCK	PB13	SPI 2
Can Controller 1	CS	PB9	SPI 2
Can Controller 1	INT	PB0	
Can Controller 1	RX0BF	PB6	
Can Controller 2	MISO	PB4	SPI 3
Can Controller 2	MOSI	PB5	SPI 3
Can Controller 2	SCK	PB3	SPI 3
Can Controller 2	CS	PA15	SPI 3
Can Controller 2	INT	PB1	
Can Controller 2	RX1BF	PB7	
SD Card	MISO	PA6	SPI 1
SD Card	MOSI	PA7	SPI 1
SD Card	SCK	PA5	SPI 1
SD Card	CS	PA4	SPI 1
Firmware Prog	SWDIO	PA13	SWD
Firmware Prog	SWCLK	PA14	SWD
MCU	Key (Active low)	PA0	
MCU	LED (Active low)	PC13	
MCU	GND	BOOT0	

4.2.3 General Validation

Microcontroller integration generally consists of two parts - designing microcontroller support components, and choosing pins for inter-block interfaces. All decisions were made with interface properties in mind.

The support component design is taken from an existing open-hardware development board, developed by WeAct Studio. The schematic is stripped of parts which are unnecessary for this block (USB, power supply, ADC voltage reference), but otherwise copied exactly. This allows for prototyping and block verification procedures to use the existing development board, without any risk of integration problems arising from a switch to a custom design late in the project. Additionally, this simplifies the design process and removes opportunities for human error.

Pin assignments are somewhat arbitrary, as the chosen MCU has an overabundance of I/O for the given task. However, consideration was given to ensure that the chosen pins did not interfere with any peripherals which may be added in the future. For instance, UART1 and USB peripherals might be used for debugging purposes later in the project.

4.2.4 Interface Validation

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
bck_cnvrtr_mc_dcpwr : Input		
Inominal: 10mA	Determined by budgeting power supply current output among components. Note: this property means that the power supply must be capable of supplying *at least* 10mA of constant current to the MCU, and the MCU must draw *no more than* 10mA on average. Either side may exceed the spec.	According to the MCU datasheet, the MCU draws a nominal current of 9.4mA when running at 50MHz with all peripherals enabled. In practice, the MCU runs with some peripherals disabled.
Ipeak: 100mA	Determined by budgeting power supply current output among components.	According to the MCU datasheet, the MCU draws a maximum of 10.1mA when running at 50MHz with all peripherals enabled. In

		practice, the MCU runs at a lower speed with some peripherals disabled.
Vmax: 3.5V	Determined from power supply characteristics.	The MCU datasheet asserts that the device will operate normally if the supply voltage does not exceed 3.6V.
Vmin: 3.0V	Determined from power supply characteristics.	The MCU datasheet asserts that the device will operate normally if the supply voltage is at or above 1.8V.

frmwr_mc_data : Input

Messages: C language	The C language was chosen due to its efficiency and high compatibility with embedded systems.	ST Microelectronics provides a compiler toolchain capable of compiling C for the stm32f411.
Other: Architecture: ARM Cortex M4 with DSP extensions	Specified by MCU datasheet.	ST Microelectronics provides a compiler toolchain capable of compiling C for the stm32f411.
Other: Code size: 512kB maximum	Specified by MCU datasheet.	The datasheet asserts that the device will operate normally as long as code does not take up more than 512kB.
Protocol: SWD	While it is possible to program the STM32F411CEU6 via various interfaces through bootloaders, SWD provides the most reliable and easy to use interface.	The datasheet asserts that the MCU supports programming via SWD.

bltth_mc_comm : Input

Datarate: 9600 Baud	The baud rate was arbitrarily chosen to be 9600 baud. This speed is high enough for a responsive user interface, but low enough to ensure reliability.	The USART peripheral supports a range of baud rates, including 9600.

Other: Logic Level Voltage: 3.3V	GPIO logic levels specified by bluetooth module datasheet.	MCU GPIO pins are capable of producing or detecting the requisite logic levels, given a 3.3V supply voltage.
Protocol: RS-232	Specified by bluetooth module datasheet.	The MCU USART peripheral is capable of RS-232 communication.

cn_cntrllrtrnscvr_mc_comm : Input

Other: SPI Clock Speed at least 8MHz	Specified by CAN controller datasheet.	The MCU datasheet specifies that SPI peripherals operate at up to 50MHz. As SPI sends one bit per clock pulse, this equates to a maximum data rate of 50Mb/s - exceeding the specification.
Messages: Data is received by reading 8 byte registers.	Specified by CAN controller datasheet.	Higher-level communication protocols on top of SPI may be implemented in software on the MCU.
Other: Logic Level Voltage: 3.3V	Specified by CAN controller datasheet.	MCU GPIO pins are capable of producing or detecting the requisite logic levels, given a 3.3V supply voltage.
Protocol: SPI	Specified by CAN controller datasheet.	The MCU includes enough SPI peripherals for both CAN controllers.

strg_mc_data : Input

Other: SPI Clock Speed at least 8MHz	The data input rate was chosen arbitrarily, so as to be fast enough to provide a smooth user interface when downloading data over Bluetooth.	The MCU datasheet specifies a maximum SPI speed of 50Mb/s
Other: Logic Level Voltage: 3.3V	Specified by SD standard.	MCU GPIO pins are capable of producing or detecting the requisite logic levels, given a 3.3V supply voltage.
Protocol: SPI	Specified by SD standard.	The MCU includes SPI peripherals.

mc_bltth_comm : Output

Messages: Baud Rate: 9600	The baud rate was arbitrarily chosen to be 9600 baud. This speed is high enough for a responsive user interface, but low enough to ensure reliability.	The USART peripheral supports a range of baud rates, including 9600.
Protocol: RS-232 serial	Specified by bluetooth module datasheet.	The MCU USART peripheral is capable of RS-232 communication.
Other: Logic Level Voltage: 3.3V	GPIO logic levels specified by bluetooth module datasheet.	MCU GPIO pins are capable of producing or detecting the requisite logic levels, given a 3.3V supply voltage.

mc_cn_cntrllrtrnscvr_comm : Output

Other: SPI Clock Speed at least 4MHz	Specified by CAN controller datasheet.	The MCU datasheet specifies that SPI peripherals operate at up to 50MHz.
Messages: Data is received by reading 8 byte registers.	Specified by CAN controller datasheet.	Higher-level communication protocols on top of SPI may be implemented in software on the MCU.
Other: Logic Level Voltage: 3.3V	Specified by CAN controller datasheet.	MCU GPIO pins are capable of producing or detecting the requisite logic levels, given a 3.3V supply voltage.
Protocol: SPI	Specified by CAN controller datasheet.	The MCU includes enough SPI peripherals for both CAN controllers.

mc_strg_data : Output

Other: SPI Clock Speed at least 4MHz	According to the MCP2515 CAN controller datasheet, data can enter the system at a maximum rate of 1Mb/s. As there are two CAN controllers, the maximum data rate becomes 2Mb/s. Storing the binary data as hexadecimal	The MCU datasheet specifies a maximum SPI speed of 50Mb/s

	numbers in ASCII doubles the throughput requirement to 4Mb/s. An extra 6Mb/s are added to account for extra metadata and for inefficiencies in processing.	
Other: Logic Level Voltage: 3.3V	Specified by SD standard.	MCU GPIO pins are capable of producing or detecting the requisite logic levels, given a 3.3V supply voltage.
Protocol: SPI	Specified by SD standard.	The MCU includes SPI peripherals.

4.2.5 Verification Plan

All Logic Level Properties:

When performing each of the following tests, additionally perform the following procedure to test logic levels:

1. Attach an oscilloscope to the data bus. Set the oscilloscope to one-shot mode, triggered by either edge.
2. Perform the other test.

Pass condition: Oscilloscope shows logic high values of 3.3v and logic low values of 0v.

All Power Supply Properties:

During any of the following tests, additionally perform the following procedure to test power supply properties:

1. Attach the DUT to a bench power supply. Attach a multimeter to measure current draw.
2. Power the DUT with Vmin volts.
3. Run the test.
4. Power the DUT with Vmax volts.
5. Run the test.

Pass condition: The device performs normally when powered from either voltage, and current draw never exceeds I_{max}.

$I_{nominal}$ may be found in the STM33F411CEU6 datasheet.

Firmware:

The firmware loading interface is verified by loading the test program for any of the following tests, and verifying that the device functions correctly.

All RS-232 or SPI Interfaces:

Construct a test rig, consisting of an STM32F411CEU6 evaluation board connected to the DUT via the interface under test.

Write a "sender" program, which sends a set sequence of data over the interface under test.

Write a "receiver" program, which receives data over the interface under test. The receiver may either output the data to a computer for analysis, or verify the data itself.

To test output interfaces:

1. Load the "sender" program onto the DUT.
2. Load the "receiver" program onto the test bench.
3. Connect a logic analyzer to the relevant data bus, and start capture.
4. Run the "sender", and record results from both devices.
5. Determine data clock speed with the logic analyzer.

Pass condition: Data received is identical to data sent. Measured data clock must be within 2% of the specification.

4.2.6 References

[1] STM32F411CEU6 datasheet:
https://www.st.com/resource/en/datasheet/stm32f411re.pdf

[2] HC-05 Bluetooth module datasheet:
https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf

[3] MCP2515 CAN controller datasheet:
https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf

4.2.7 Revision Table

Date	Name	Section Revised	What was revised?
01/06/22	Anton L	All sections	First draft created
01/21/22	Anton L	All sections	Incorporated instructor feedback

4.3 MCU Firmware

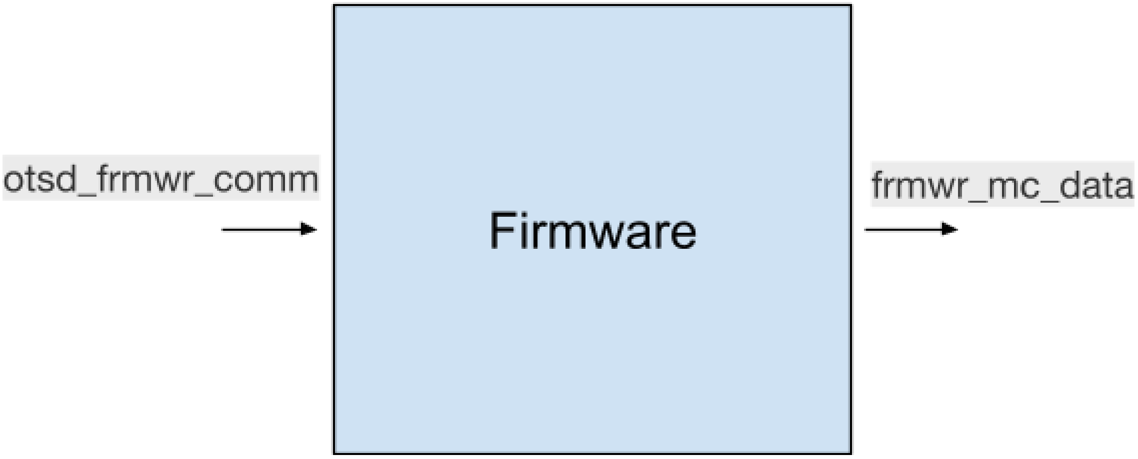
4.3.1 Description

The MCU firmware block encompasses the data stored in program memory on the MCU. Interfaces to the firmware block consist of the hardware interface for loading firmware onto the MCU, as well as the software interface between the STM32's ARM Cortex M4 processor and the firmware binary.

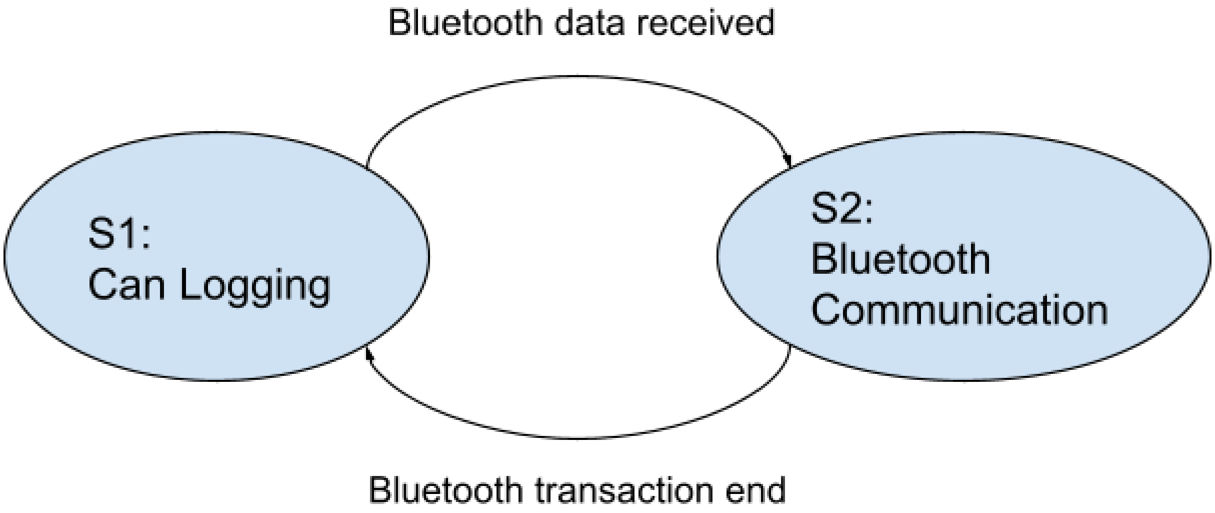
The MCU firmware consists of three parts. The first is a critical section which collects CAN data from the CAN controller and records it in main memory. This section is interrupt-driven, and is optimized as much as possible for stable real-time performance. The second part is the main loop, which pulls CAN data from main memory and stores it on the SD card. A circular buffer producer-consumer queue allows the CAN handler to safely pass packets to the main loop. Another part of the firmware handles data upload via Bluetooth. This section runs while the critical section is stopped, and has no hard performance requirements. Anton Liakhovitch is responsible for this block.

4.3.2 Design

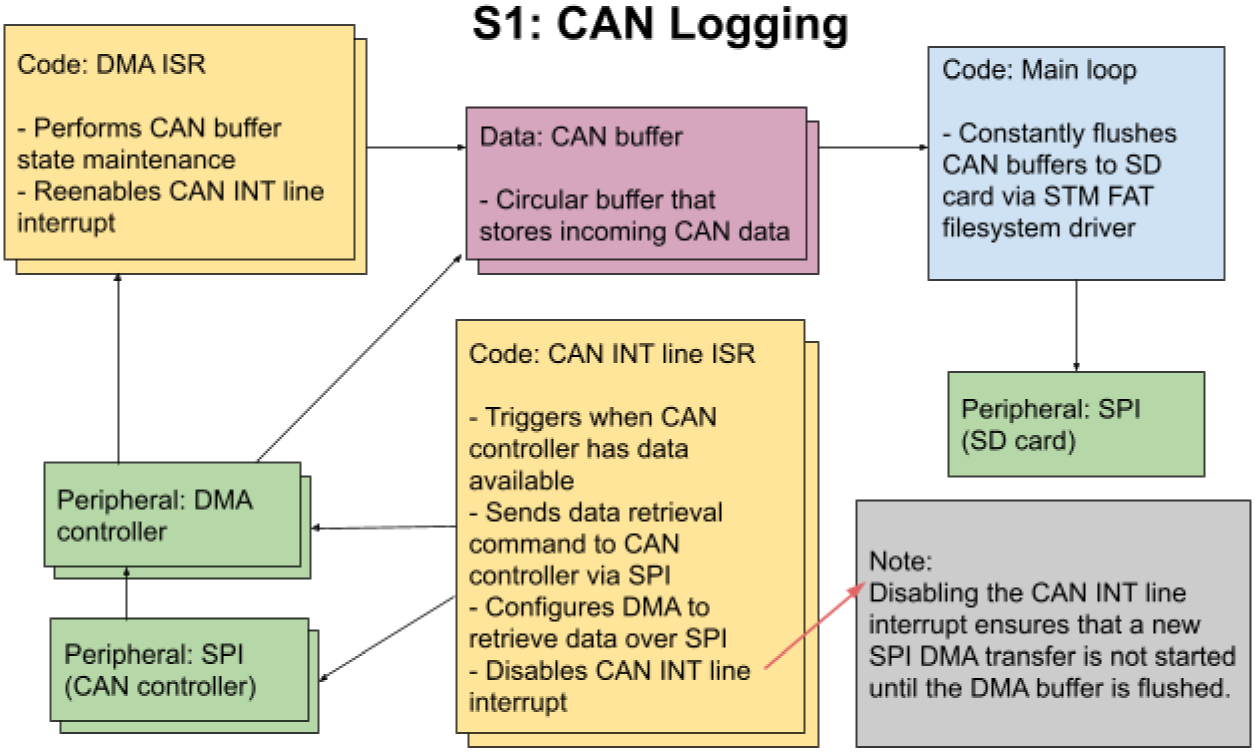
Black Box



State Machine Diagram

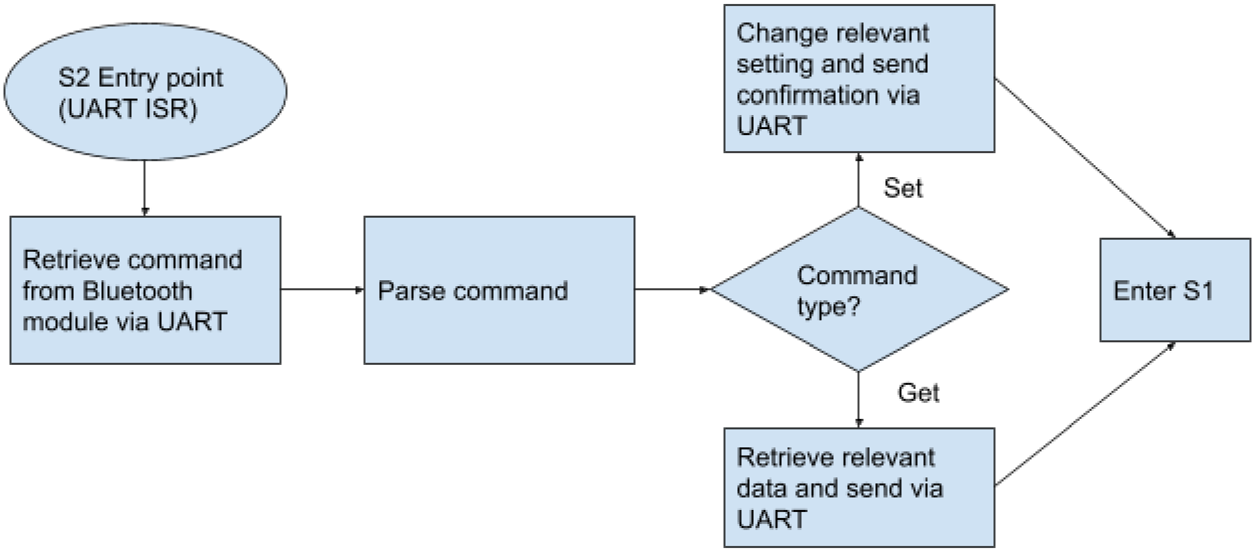


S1: CAN Logging


Code: DMA ISR
- Performs CAN buffer state maintenance
- Reenables CAN INT line interrupt
Data: CAN buffer
- Circular buffer that stores incoming CAN data
Code: Main loop
- Constantly flushes CAN buffers to SD card via STM FAT filesystem driver
Code: CAN INT line ISR
- Triggers when CAN controller has data available
- Sends data retrieval command to CAN controller via SPI
- Configures DMA to retrieve data over SPI
- Disables CAN INT line interrupt
Peripheral: SPI (SD card)
Peripheral: DMA controller
Peripheral: SPI (CAN controller)
Note:
Disabling the CAN INT line interrupt ensures that a new SPI DMA transfer is not started until the DMA buffer is flushed.

Note: This is not a traditional program flowchart. Blocks represent memory, memory-mapped I/O, or code. Arrows represent the flow of data.

S2: Bluetooth Communication


S2 Entry point (UART ISR)
Retrieve command from Bluetooth module via UART
Parse command
Command type?
Set
Get
Change relevant setting and send confirmation via UART
Retrieve relevant data and send via UART
Enter S1

4.3.3 General Validation

The firmware interface properties exist to ensure that any firmware written for the CAN Logger project successfully downloads and runs on the microcontroller. In order to ensure maximum reliability in this regard, the project uses the official ST-link V2 hardware debugger and the official STM32CubeMX software development kit. STMicroelectronics designed these resources to be compatible with its microcontrollers, so there are no foreseeable reasons why the firmware may not download to the MCU or run on the MCU.

Alternatively, it may be possible to use the Arduino toolchain. However, the Arduino environment results in slow compilation and poor runtime performance, has no support for debugging, and generally has poor support for STM32 microcontrollers. This should be used only as a last resort.

The firmware itself is designed to maximize CAN data input speed at all costs, while also prioritizing flushing of data to the SD card. This is achieved through extensive use of interrupt service routines and the two DMA (Direct Memory Access) controllers on the MCU. Interrupts ensure that the microcontroller spends resources on an SPI transaction as soon as it is needed, and only when it is needed. This both improves SPI response time and maximizes CPU time available to other tasks (such as writing to the SD card). The DMA controller allows the bulk of every SPI transaction to happen in the background without any involvement from the CPU, once again saving CPU time. Maximizing input speed is necessary to ensure that the Logger does not miss any data sent in fast bursts over the CAN bus. This is done at the expense of buffer size and total average throughput, which are less important as long as the CAN bus runs at less than 100% utilization.

4.3.4 Interface Validation

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
otsd_frmwr_comm : Input		
Other: Logic Level: 3.3V	Specified by MCU datasheet.	MCU GPIO pins are capable of producing or detecting the requisite logic levels, given a 3.3V supply voltage.
Other: Clock Rate: Automatically negotiated	Specified by MCU datasheet.	The ST-link V2 programmer is capable of automatically negotiating an interface clock speed compatible with the

		transmission medium.
Protocol: Serial Wire Debug Interface	While it is possible to program the STM32F411CEU6 via various interfaces through bootloaders, SWD provides the most reliable and easy to use interface.	The STM32F411CEU6 datasheet asserts that the MCU supports programming via SWD.

frmwr_mc_data : Output

Messages: C language	The official STM32 SDK best supports the C language. While there are other SDKs and languages available (Arduino, Rust), the STM32 SDK provides the best balance between software support and runtime performance.	The firmware is developed in c99-compliant C.
Other: Architecture: ARM Cortex M4 with DSP extensions	Specified by the STM32f411CEu6 datasheet.	The GCC compiler is configured with the ARM Cortex M4 target.
Other: Code size: 512kB maximum	Specified by the STM32f411CEu6 datasheet.	Early I/O test firmwares are significantly under the size limit, so it is highly unlikely that the final binary will exceed the limit.

4.3.5 Verification Plan

All interface properties may be verified via a single test.

1. Write a test program in C (verifies "C language" property). This program can do anything whatsoever as long as its behavior is verifiable. For example, it may blink an LED.
2. Connect an STLink V2 debugger to the computer and attach it to the SWD interface of the microcontroller (verifies the "Protocol: Serial Wire Debug" property).
3. Attach an oscilloscope to the SWCLK and SWDIO lines of the SWD interface, set it to one-shot mode, and set it to trigger on any edge.
4. Program the microcontroller via OpenOCD. Verify that the clock speed reported by OpenOCD is under the 50MHz maximum (Verifies the Clock Rate: 50MHz Max property). Verify that OpenOCD did not report a program size error (verifies the "Code size" property).
5. Verify that the oscilloscope shows a 3.3V logic level (Verifies the Logic Level: 3.3V property)

6. Verify that the test program behaves as it should when running on the microcontroller. This verifies the architecture property, as a program binary compiled for the wrong architecture will not run at all.

4.3.6 References

[1] STM32F411CEU6 datasheet:
https://www.st.com/resource/en/datasheet/stm32f411re.pdf

4.3.7 Revision Table

Date	Name	Section Revised	What was revised?
02/04/22	Anton L	All sections	First draft created
02/18/22	Anton L	All sections	Revised for submission
5/5/22	Anton L	All sections	Revised for tense, grammar

4.4 CAN Controller/Transceiver

4.4.1 Description

The CAN Controller/Transceiver block is responsible for Receiving and Transmitting CAN signals and communicating with the MCU. The MCP2515 CAN controller is being used to interpret the CAN messages and send the packages to the MCU in a package that the MCU can interpret. Maxim Feoktistov is responsible for the CAN Controller/Transceiver block.

4.4.2 Design

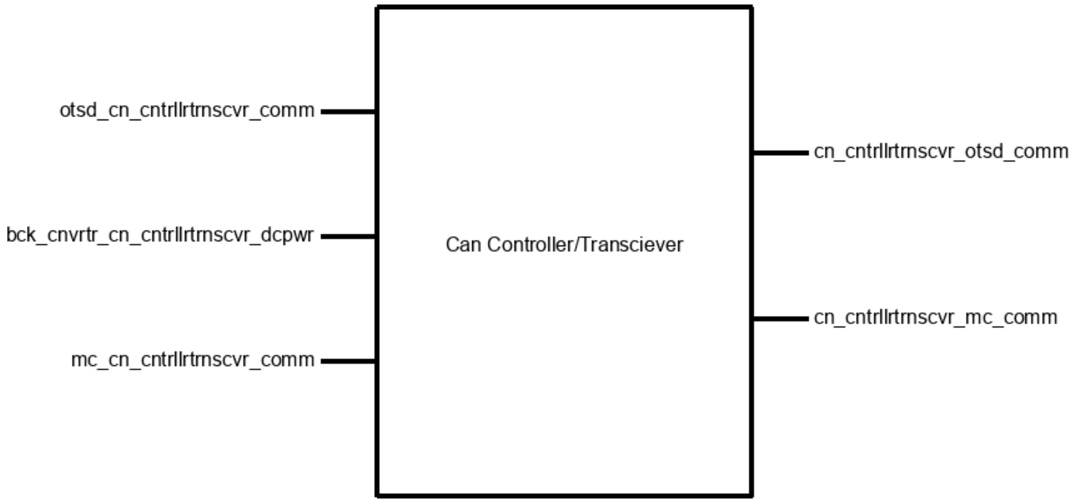

otsd_cn_cntrllrtrnscvr_comm
bck_cnvrtr_cn_cntrllrtrnscvr_dcpwr
mc_cn_cntrllrtrnscvr_comm
Can Controller/Transciever
cn_cntrllrtrnscvr_otsd_comm
cn_cntrllrtrnscvr_mc_comm

Figure 1: CAN Controller Block with Interfaces

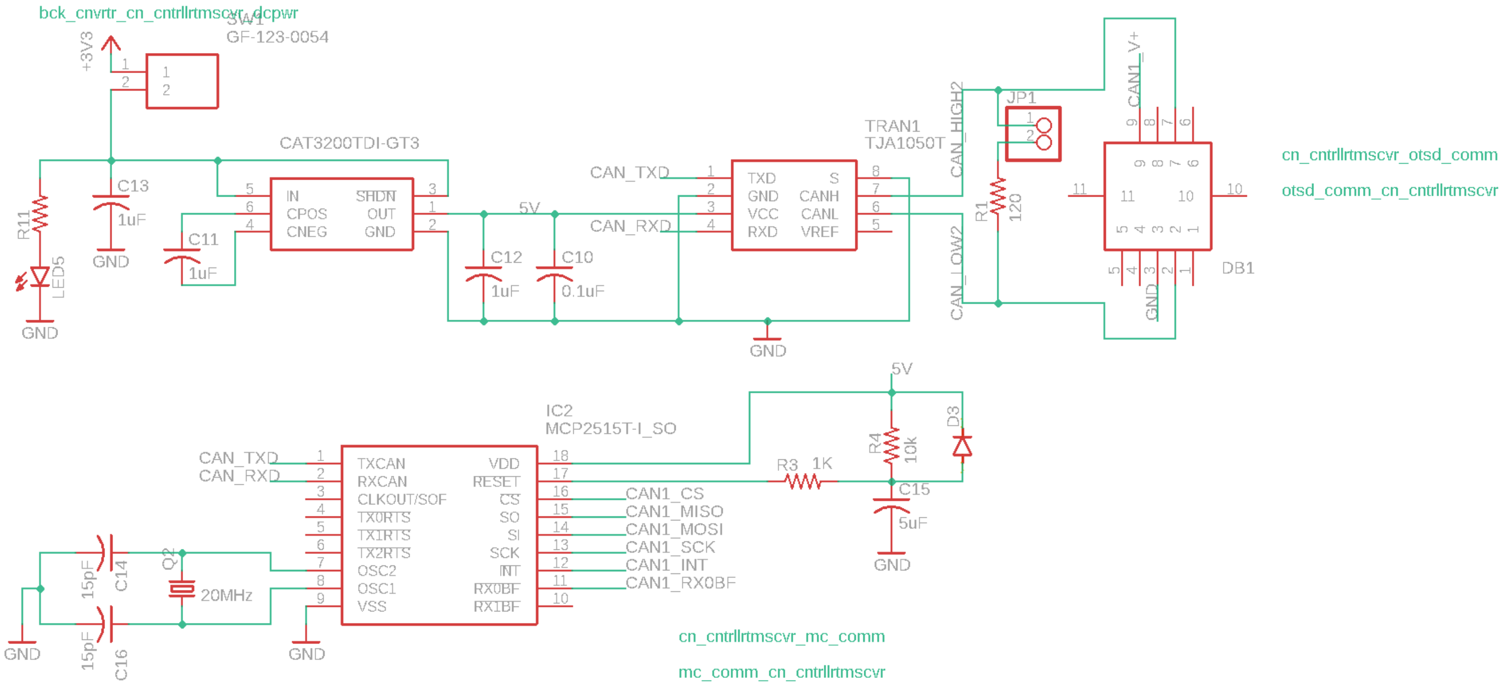


Figure 2: CAN Controller Schematic

4.4.3 General Validation

The block is designed to communicate with a CAN bus and consists of a transceiver, converter, and a controller. The design uses a db9 connector to connect the CAN bus to the design as per customer request. The connector is widely used in the automotive industry and is abundant and cheap. A CAN transceiver (TJA1050) is used to take in the CAN high and CAN low signals. The general application of the transceiver was referenced when designing the block [1]. The CAN controller (MCP2515) communicates with the CAN transceiver and can communicate with an MCU, sending 8 byte registers through SPI protocol.The CAN controller datasheet also contained the general application of the controller [2]. Since the block needs a 5V power supply and the system operates at 3.3V, the CAN transceiver and controller will need a booster that converts 3.3V to 5V. The CAT3200−5 DC-DC converter is a cheap and abundant chip that outputs a fixed 5V supply. The DC-DC converter datasheet contained an example of how to use the converter [3].

4.4.4 Interface Validation

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**

Input : otsd_comm_cn_cntrllrtrnscvr
Output: cn_cntrllrtrnscvr_otsd_comm

Baud Rate: 250K/500K	The two Baud Rates that the CAN bus can be operating at.	TJA1050 datasheet, page 2: High s **to 1**peed transceiver of **up Mbaud**.
CAN Protocol	The device must take in a CAN bus signal	TJA1050 datasheet, page 2: TJA1050 is the interface between the CAN protocol controller and the physical CAN bus.
DB9 Connector to connect CAN bus with CAN controller/ transceiver Block	Standard CAN bus connection is through a DB9 connector	CAN Protocol Standards [5]: Industrial standard to use a DB9 connector for CAN bus with CAN Low at pin 2 and CAN High at Pin 7.

Input: mc_comm_cn_cntrllrtrnscvr
Output: cn_cntrllrtrnscvr_mc_comm

SPI Protocol	SPI provides high speed communication between MCU and CAN controller	MCP2515 Datasheet, page 1: The MCP2515 interfaces with MCUs via an industry standard SPI
Data is received by reading 8 byte registers	CAN bus messages consist of up to 8 bytes of data.	MCP2515 Datasheet, page 9: Figure 2-1: Specifies data field of up to 8 bytes
SPI Clock Speed at least 4MHz	The CAN controller must be able to operate at speeds of at least 4MHz for efficient communication	MCP2515 Datasheet, page 77: Peak Clock frequency at 10MHz
Logic Level Voltage: 3.3V	The system will operate at 3.3V.	MCP2515 Datasheet, page 74: Logic Level High range:

		2.31V - 4.3V.

bck_cnvrtr_cn_cntrllrtrnscvr_dcpwr

Ipeak:100mA	Peak current allocated to this block to prevent drawing too much power.	MCP2515 Datasheet, page 74: Peak current = 10mA TJA1050 datasheet, page 6: Peak current = 75mA Total peak current is 85mA
Inominal: 60mA	Current allocated to this block to fulfill the limited power dissipation requirement.	MCP2515 Datasheet, page 1: Typical operating current = 5mA TJA1050 datasheet, page 2: Typical operating current = 50mA Total operating current = 55mA
Vmax: 3.5V	The power supply (at 3.3V) of the system varies by about 200mV	MCP2515 Datasheet, page 74: Max voltage = 5.5V cat3200-5 Datasheet - page 3 Max voltage = 4.5V
Vmin: 3.0V	The power supply (at 3.3V) of the system varies by about 200mV	MCP2515 Datasheet, page 74: Max voltage = 2.7V cat3200-5 Datasheet - page 3 Max voltage = 2.7V

4.4.5 Verification Process

1. Put together all the components specified in the schematic (Figure 2) together.
2. Use a DC power supply to supply the system with 3.3V.
3. Connect the CAN bus to the system via a DB9 connector.
4. Connect a SPI compatible MCU to the CAN Controller SPI pins.

For the **bck_cnvrtr_cn_cntrllrtrnscvr_dcpwr** interface:

5. While the system is operating, change the voltage on the power supply to 3.5V and 3.0V and inspect if the system is operating normally.

6. While the system is operating, inspect the current draw on the power supply to find the nominal current of the system.
7. Load the system by having the system operate at top speed and inspect the current spike on the power supply to find the peak current.

For the **cn_cntrllrtrnscvr_otsd_comm** and **otsd_comm_cn_cntrllrtrnscvr** interfaces:

8. Ensure that the CAN bus is connected to the CAN transceiver via a DB9 connector and inspect if the MCU is receiving data from the CAN bus.
9. Change the baud rate of the CAN bus to 500K and 250K
10. Change the baud rate setting on the MCU and ensure that the system is able to read the messages from the CAN bus on both baud rates.

For the **cn_cntrllrtrnscvr_mc_comm** and **mc_comm_cn_cntrllrtrnscvr** interfaces:

11. While the system is operating, measure the voltage of the CAN_TXD and CAN_RXD, to find the logic level voltage of those pins
12. Inspect the CAN messages that are received by the MCU and determine the number of bytes per message.
13. Use the MCU to determine the maximum speed of the system.

4.4.6 References and File links

4.4.6.1 References

[1] “TJA1050 High speed CAN transceiver,” *Philips Semiconductors*, May 2002, [Online]. Available:https://www.nxp.com/docs/en/data-sheet/TJA1050.pdf

[2] “MCP2515 Stand-Alone CAN Controller with SPI Interface Data Sheet,” *Microchip*, Jan 2019. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf

[3] “cat3200-d.pdf,”*Onsemi*, Apr 2014, [Online]. Available: https://www.onsemi.com/pdf/datasheet/cat3200-d.pdf

[5] “CAN Bus Connectors”, *KVASER*, [Online]. Available: https://www.kvaser.com/about-can/the-can-protocol/can-connectors/

4.4.6.2 File Links

4.4.7 Revision Table

Date	Section Revised	What was revised?
1/5/22	4.4.1-4.4.3	The Initial Creation of the Sections

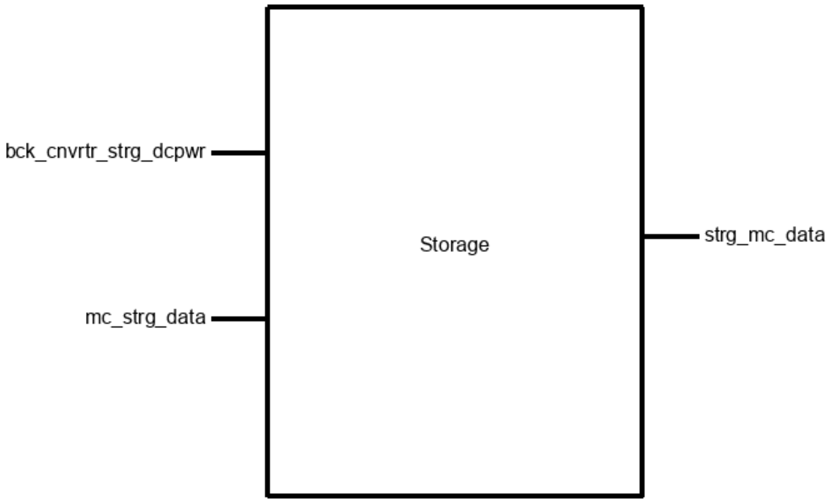
1/7/22	4.4.4-4.4.7	The Initial Creation of the Sections
1/17/22	4.4.2,4.4.3,4.4.6,4.4.7	Changed Schematic Layout and Revised some of the wording. Changed References to IEEE format and edited Revision Table
1/21/22	4.4.2-4.4.5	Updated Schematic, Revised wording of the the validation paragraph, updated some interface properties and revised the testing process

4.5 Storage

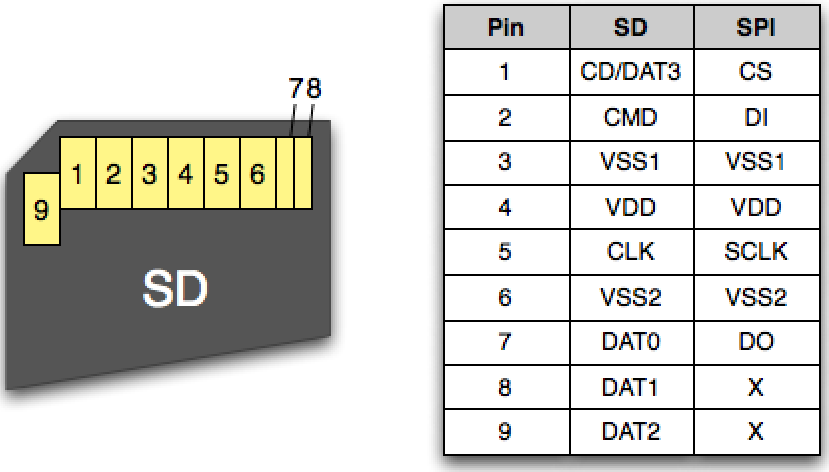
4.5.1 Description

The CAN frames logged by the system from the CAN bus must be stored so it can be used for analysis. The data will be stored on an SD card, connected to the PCB with an SD card socket. It will use a SPI connection to communicate with the MCU. The SPI connection will operate with a clock signal of at least 8MHz. The MCU will be reading, and storing data onto the SD card. The CAN frames will be stored as text in a csv file format. Ryan Dillard will be in charge of developing this block.

4.5.2 Design

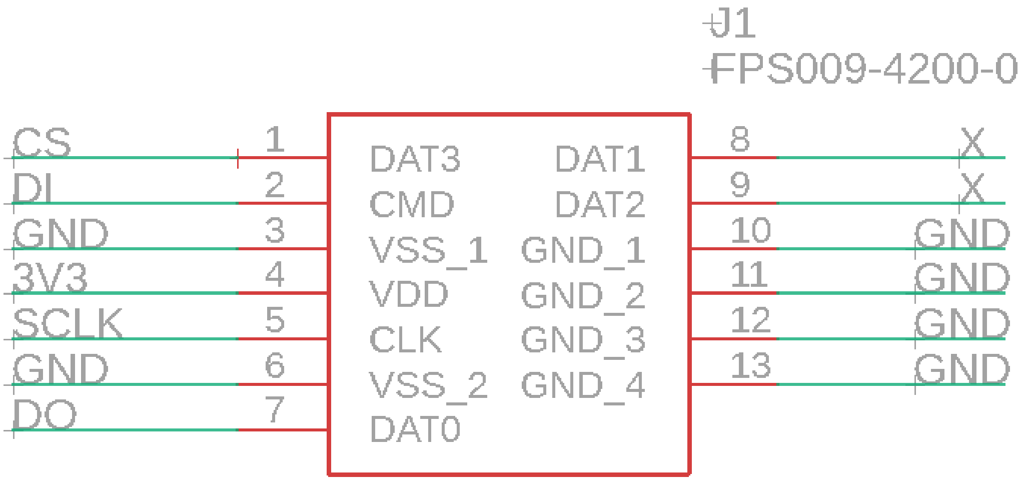


Black Box diagram of the block



Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	X
9	DAT2	X

SD card pinouts



Socket for an SD card for a PCB

Pins Corresponding to Block Interfaces		
mc_strg_data	**stg_mc_data**	**bck_cnvtr_strg_dcpwr**
CS	**CS**	**3V3**

DI	**DI**	**GND**
SLK	**SLK**	
DO	**DO**	

4.5.3 General Validation

This block serves two functions for the system. The first is that it stores the configuration files for the device. The second reason is to store a file of CAN frames logged from the CAN bus. This block uses a SPI interface to communicate with the MCU and receive CAN data. The SPI connection allows for a high data rate to meet the data rate needed for the system. Additionally SD cards natively work at 3.3V which is the same as the MCU making the connection process simpler.

The components of this device are simple: it is a SD card socket, pull up resistors, and a SD card. Storage could’ve been handled by using a MCU with a large internal flash, or external flash. We avoided using either of these solutions to maintain the ease of replacing the storage. Using an SD card allows the users to upgrade the storage size if needed.

4.5.4 Interface Validation

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**

bck_cnvrtr_strg_dcpwr : Input

Inominal: 30mA	Based on 10Mb/s write speed	From SD standard ● At Max write SD card would use 0.33W at 50Mb/s
Ipeak: 200mA	This value was chosen as a worst case scenario. The storage block should likely never reach this current, but the Power supply would be able to supply it. This does vary by the card, but at default the current shouldn’t	For the STM32f411CEU6 in the LQFP64 package: ● Max clock speed supports 50Mb/s [1] From SD standard ● At Max write SD card would use 0.33W at 50Mb/s

	exceed this.	
Vmax: 3.5V	This value was chosen to give a little room on the SD card operating voltage specifications. Additionally it slightly exceeds the potential variation of the power supply.	Based on the operating voltage of an SD card. - A 3.3V SD card can operate from 2.7V to 3.6V
Vmin: 3.0V	This value was chosen to give a little room on the SD card operating voltage specifications. Additionally it slightly exceeds the potential variation of the power supply.	Based on the operating voltage of an SD card. - A 3.3V SD card can operate from 2.7V to 3.6V

strg_mc_data : Output

Other: SPI Clock Speed at least 8MHz	This was chosen as to meet system requirements this is the minimum speed we will need for the SD card	-The MCU datasheet specifies that SPI peripherals operate at up to 50MHz. As SPI sends one bit per clock pulse, this equates to a maximum data rate of 50Mb/s - exceeding the specification.[1] -SD cards operate at two speed modes. The default mode clock speed is 25MHz, and a high speed mode at 50MHz.[2]
Other: Logic Level Voltage: 3.3V	Specified by SD card standard. We are also using a MCU that operates of 3.3V so we can communicate at 3.3V	Based on the operating voltage of an SD card. - A 3.3V SD card can operate from 2.7V to 3.6V
Protocol: SPI	Specified by SD card standard	There are two bus protocols for SD cards.[3] - Native SD mode which uses a 4 bit bus - SPI mode using a 1 bit bus

mc_strg_data : Input

Other: Logic Level Voltage: 3.3V	Specified by SD card standard. We are also using a MCU that operates of 3.3V so we can	Based on the operating voltage of an SD card. - A 3.3V SD card can

	communicate at 3.3V	operate from 2.7V to 3.6V
Other: SPI Clock Speed at least 8MHz	This was chosen as to meet system requirements this is the minimum speed we will need for the SD card	-The MCU datasheet specifies that SPI peripherals operate at up to 50MHz. As SPI sends one bit per clock pulse, this equates to a maximum data rate of 50Mb/s - exceeding the specification.[1] - SD cards operate at two speed modes. The default mode clock speed is 25MHz, and a high speed mode at 50MHz.[3]
Protocol: SPI	Specified by SD standard	There are two bus protocols for SD cards.[3] - Native SD mode which uses a 4 bit bus - SPI mode using a 1 bit bus

4.5.5 Verification Process

Bck_cnvrtr_strg_dcpwr Verification:

1. Attach the storage device to a bench power supply.
2. Power the storage device with Vmin volts.
3. Send a message to the storage device over SPI
4. Check storage device contents for the message
5. Power the storage device with Vmax volts.
6. Send a message to the storage device over SPI
7. Check the storage device contents for the message

If the stored message matches the sent message in both sends and the device performs without exceeding peak current it passes.

mc_strg_data, and stg_mc_data verification:

1. Send a message over SPI to the storage device.
2. Using an Oscope to capture message one-shot mode attached to the data bus.
3. Verify with the Oscope frame that the data is clocked at at least 8Mhz.
4. Verify with the Oscope frame that the data logic levels match.
5. Repeat this process but measure the frame from the transmit of the storage.

Pass condition: If the message is stored on the storage device accurately, and a new message can be read from the storage device, additionally if the measurements from the Oscope match the properties then mc_strg_data and strg_mc_data pass.

4.5.6 References and File links

4.5.6.1 References

[1] STM32F411CEU6 datasheet:
https://www.st.com/resource/en/datasheet/stm32f411re.pdf

[2] https://www.sdcard.org/developers/sd-standard-overview/

[3]
https://www.sdcard.org/downloads/pls/pdf/?p=Part1_Physical_Layer_Simplified_Specification_Ver8.00.jpg&f=Part1_Physical_Layer_Simplified_Specification_Ver8.00.pdf&e=EN_SS1_8

4.5.6.2 File Links

4.5.7 Revision Table

Date	Name	Section	Revision
12/3/2022	Ryan Dillard	4.5.1-4.5.7	Created document and wrote rough content for each section
12/4/2022	Ryan Dillard	4.5.4	Added detail to interface properties, and to verification plan.
12/17/2022	Ryan Dillard	4.5.1	Revised the design, and added additional detail to clarify the purpose of this block, and what it is.
12/18/2022	Ryan Dillard	4.5.5 & 4.5.4	Added additional detail to the verification plan, and filled out a few properties that weren’t completely justified.

4.6 Bluetooth Module

4.6.1 Description

One of the customer's requirements is to wirelessly transfer collected data to a computer. In order to be able to output the files onto a CSV file onto a file, the team has opted to use a bluetooth module to receive all the data collected by the CAN buses. This module can receive values from both channels and update its output to the terminal in real time. This will allow the user some flexibility and efficiency when using the CAN logger. Ashley Reid is the champion of this block.

4.6.2 Design

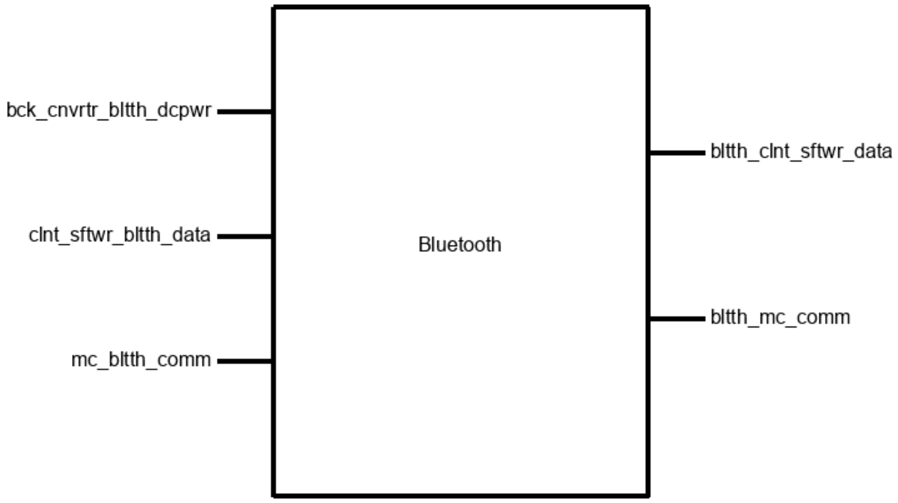


Figure 1

Figure one displays the black box for the Bluetooth module. Outputs on the left and inputs on the right.

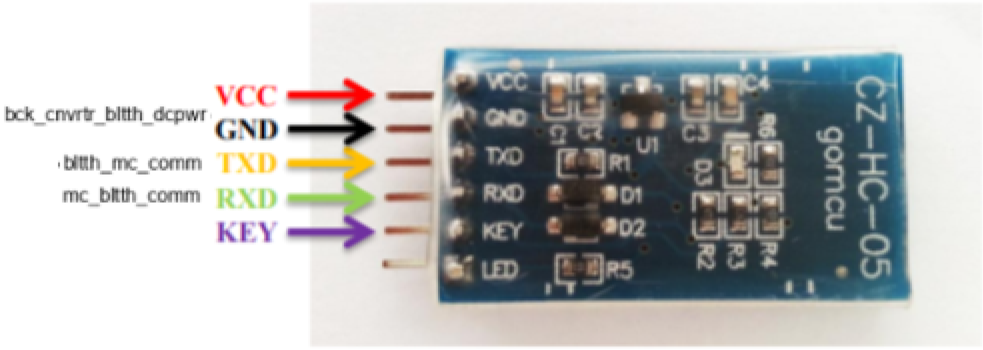


Figure 2

The image above is a picture of the physical modeling to be used on the PCB. Interfaces that are wired are labeled accordingly. TXD writes the texts to the MCU and RXD reads text from the MCU.

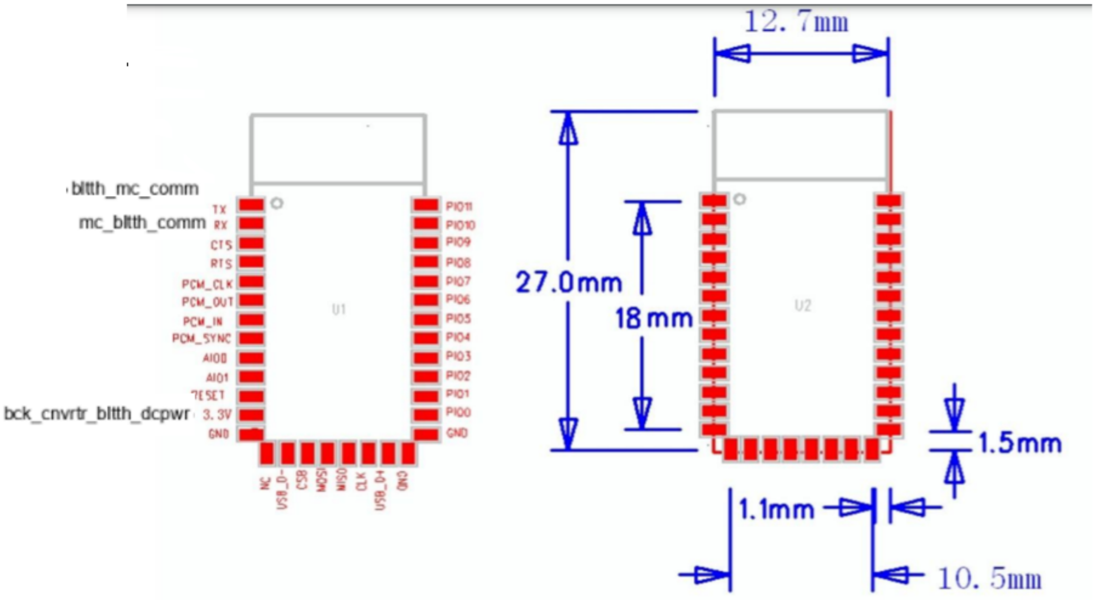


Figure 3
Figure 3 has the wired interfaces labeled on the actual chip itself.

4.6.3 General Validation

The module picked out, the HC-06 was picked for our system design, as it provides a simple integration with the main body of the system and any laptop. Because the system will only be used within the Hyster-Yale site, as specified by the project partner, there is no need to keep this connection exclusive to the site's wifi. The HC-06 has a PCB board that already interacts with the bluetooth chip and only needs specific wiring with simplified wiring outputs which include the following: VCC, GND, RXD, and TXD. Here the RXD shows the receiving end, the TXD is the transmitting data end.

Using this module assists the cost of parts, as if the team were to go to the Wifi router, it was concluded that the system would need a whole router box because of the clearances it would need. It was also decided, because the specified module that will be used, the HC-06, is widely available on Amazon. It also has the standard use of 3.3 V as its logic level, allowing for more consistency and ease for the rest of the system, as almost all parts of the system will be operating at 3.3 V.

Using a whole module instead of designing another schematic eliminates the risks of errors with the actual schematic and cuts down on ordering time.

4.6.4 Interface Validation

mc_bltth_comm : Input

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Messages: Baud Rate: 9600	This was chosen as a baud rate essentially as it was a happy medium for its high responsiveness but also reliability.	USART peripheral can support a 9600 baud rate. It is also specifically stated it is supported on bluetooth data sheets.
Protocol: RS-232 serial	The Bluetooth Data sheet indicated that this is the communication type.	USART peripheral can support the RS-232 communication .
Other: Logic Level Voltage: 3.3V	Logic Level specified by bluetooth data sheet. Specified by bluetooth data sheet.	USART peripheral can support the RS-232

bltth_mc_comm : Output

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Data rate: 9600 Baud	This was chosen as a baud rate essentially as it was a happy medium for its high responsiveness but also reliability.	USART peripheral can support a 9600 baud rate. It is also specifically stated it is supported on bluetooth data sheets.[data sheet #1, page 1]
Other: Logic Level Voltage: 3.3V	Logic Level specified by bluetooth data sheet.	Stated by bluetooth data sheet. [data sheet #1, page 3]
Protocol: RS-232	The Bluetooth Data sheet	USART peripheral can

	indicated that this is the communication type.	support the RS-232 communication

Clnt_sftwr_bltth_data : Input

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Data rate: 9600 Baud	This was chosen as a baud rate essentially as it was a happy medium for its high responsiveness but also reliability.	USART peripheral can support a 9600 baud rate. It is also specifically stated it is supported on bluetooth data sheets. [data sheet #1, page 1]
Other: Logic Level Voltage: 3.3V	Logic Level specified by bluetooth data sheet.	Stated by bluetooth data sheet. [data sheet #1, page 3]
Protocol: RS-232	Logic Level specified by bluetooth data sheet. Specified by bluetooth data sheet.	USART peripheral can support the RS-232 communication

Bck_cnvrtr_bltth_dcpwr : Input

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Inominal 20 mA	From the data sheets, it's the baseline operating current when not paired, so decide to use this as the nominal value.	This value was found in the this webpage dedicated to the HC-05 module [1]
Ipeak: 100 mA	From the bluetooth data sheet, it can handle this current, but as an absolute peak	This value was found in the data sheet for the HC-05 module. [1]
Vmax 3.5V	According to the data sheet, it	This value was found in the

	can handle 3.6 V at most, but and shorting the window to avoid ruining the module.	data sheet for the HC-05 module. [data sheet #1, page 3]
Vmin : 3.1 V	According to data, this is the absolute lowest it device can operate at.	This value was found in the data sheet for the HC-05 module. [data sheet #1, page 3]

Bltth_clnt_sftwr_data : Output

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Messages: ASCII data in CSV format	ASCII values are almost universal. It also allows for a range of values that was shown necessary by sample data.	Shown in user manual, the bluetooth module is able to handle ASCII values and contain a delimiter
Other: Communication with Bluetooth driver	It needs to be able to connect to any laptop with a bluetooth driver to transmit messages	Specified by bluetooth data sheet. [data sheet #1, page 3]
Protocol: Virtual RS-232 interface	This is the specified interface from the manual.	Specified by bluetooth data sheet. [data sheet #1, page 3]

4.6.5 Verification Process

1) Setup bluetooth module on breadboard and attach to power.
2) Use a serial port to connect the BluetoothModule to the laptop.
3) Use DC power power generator to test current and voltages at the nominal and peak values.
4) Use Oscope to properly check the logic level threshold.
5) Connect BluetoothModule to computer wireless bluetooth connection.
6) Open one terminal and send a message via the serial port to bluetoothModule. Open terminal with serial port connection and look for sent values.
7) Expecting it will be received, send this message back to the laptop via wireless bluetooth connection
8) Check if it is in the correct ASCII format and if they match.

9) Change the original message on bluetooth on the first terminal and check if the second terminal matches the message shortly after.
10) Change the message on the second terminal and check to see if it successfully changes on the 1st terminal.

4.6.6 References and File links

4.6.6.1 References

[1] "HC-05 - Bluetooth module," *Components101*, 16-Jul-2021. [Online]. Available: https://components101.com/wireless/hc-05-bluetooth-module. [Accessed: 22-Jan-2022].

4.6.6.2 File Links

Main Data Sheet Reference
Physical Images of the Module and Further Description

4.6.7 Revision Table

Date	Name	Section	Revision
1/31/2022	Ashley R	4.6.4 & 4.6.5	Modified Interfaces and verification plan to match the updated version due to testing obstacles
1/21/2022	Ashley R	4.6.4 & 4.6.5	Changed Interface definitions and property descriptions.Switched up whole verification plan so MCU is no longer needed.
1/20/2022	Ashley R	4.6.2 & 4.6.6	Added labels to figures, edited links and references
1/7/2022	Ashley R	4.6.4-4.6.7	Wrote a rough draft of each section.
5/5/22	Ashley R	4.6.1 &4.6.7	Adjusted wording to align with real world application

4.7 Client Software

4.7.1 Description

One of the customer's requirements is to wirelessly transfer collected data to a computer. To make this easier for the customer's use, this client software has a command line and graphical user interface. The features of this client software include: ability to download data from the main system via bluetooth, stores the data as different CSV's file respective to their can bus, change the baud rate, and delete CAN data files. Ashley Reid is the champion of this block.

4.7.2 Design

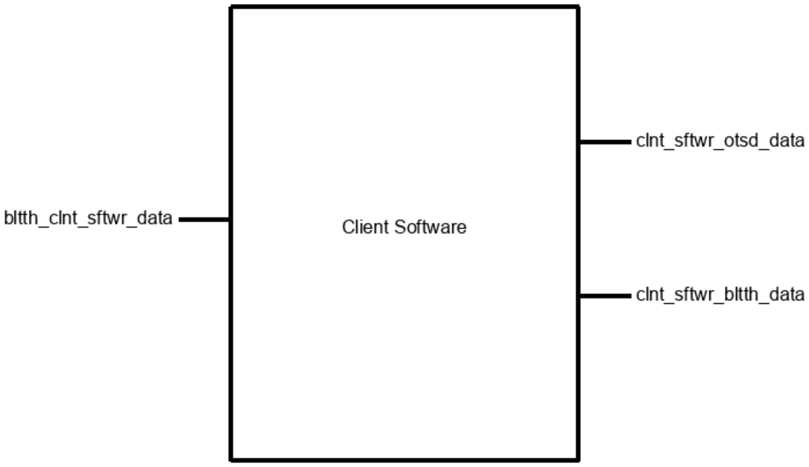


Figure 1

Figure one displays the black box for the Client Software module. Outputs on the left and inputs on the right.

4.7.3 General Validation

The goal of the client software is to prepare the data for interpretation by the company. It allows for the company to successfully save the data without having to second guess where the files need to be saved and formatted. The data sent over with this preserved and ignores any bad packets that may have been sent over. The files are automatically created and formatted correctly. This code also lets the user send commands back to the STM32 controller, like request the data CAN files, and delete these files per request as well. The requests will be performed and receive success messages back from the MCU.

The client software also has an optional GUI made for extra ease of use.

4.7.4 Interface Validation

Bltth_clnt_sftwr_data : input

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
ASCII data	ASCII values are almost universal. It also allows for a range of values that was shown necessary by sample data.	Shown in user manual, the bluetooth module is able to handle ASCII values and contain a delimiter
Communication with Bluetooth driver	It needs to be able to connect to the Bluetooth module in the system in order interact with its data	Specified by bluetooth data sheet. [Bluetooth Data Sheet, page 3]
Virtual RS-232 interface	This is the universal serial interface so this is just ensuring that it works with the bluetooth module even more	USART peripheral can support the RS-232

Clnt_sftwr_otsd_data : Output

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
File saved with full RW permissions for current user and group	This ensures that the data won't be accidentally changed but still be able to used and imported	USART peripheral can support a 9600 baud rate. It is also specifically stated it is supported on bluetooth data sheets.[Bluetooth data sheet, page 1]
UTF-8 encoding	Because it's how we can universally save these values.	UTF_8 encoding is common for saving and sending files in linux and windows.
CSV text file saved to local filesystem	CSV will allow the data received to be manipulated easily by the project partner.	The python package csv converts text to csv if original format is correct
LF line endings	To indicate when the data has	Terminal supports use of

	been fully received	newline and so does windows and linux

Clnt_sftwr_bltth_data : Output

Interface Property	**Why is this interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
ASCII data	ASCII values are almost universal. It also allows for a range of values that was shown necessary by sample data.	Shown in user manual, the bluetooth module is able to handle ASCII values and contain a delimiter
Communication with Bluetooth driver	It needs to be able to connect to the Bluetooth module in the system in order interact with its data	Specified by bluetooth data sheet. [Bluetooth Data Sheet, page 3]
Virtual RS-232 interface	This is the universal serial interface so this is just ensuring that it works with the bluetooth module even more	USART peripheral can support the RS-232

4.7.5 Verification Process

1) Connect bluetooth module with laptop and power supply. Connect physical serial port to laptop as a stand in for the STM32 microcontroller.
2) Open the serial port in Arduino IDE as this will send messages with Linefeed.
3) Open up Windows Powershell and navigate to python script and run python script
4) Check COM6 port for messages sent from Client Software. After integration, these will be commands sent to the STM32 for different lines of data between the two transceivers
5) In place of sending data over, I will be sending data from the serial port over the bluetooth module. To prove this is successful, the client software will print it to the command line. The delimiter of these values will be interpreted with newline characters during this action.
6) The client software then saves these values in a CSV. When finished, close the powershell and open the CSV file. Show the messages sent over were successful and you can write in the files as well.

4.7.6 References and File links

4.7.6.1 References

[1] “Tkinter - Python interface to TCL/TK¶,” ***tkinter - Python interface to Tcl/Tk - Python 3.10.2 documentation*****. [Online]. Available: https://docs.python.org/3/library/tkinter.html. [Accessed: 05-Feb-2022].**

4.7.6.2 File Links

Bluetooth Data Sheet **-**
CSV Library Documentation

4.7.7 Revision Table

Date	Name	Section	Revision
2/28/2022	Ashley R.	4.7.5 & 4.7.3	Due to testing, these two sections have been overhauled to reflect the GUI is not part of the verification plan
2/10/2022	Ashley R.	4.7.5	Added details to verification plan
2/10/2022	Ashley R	4.7.3	Added better justification
2/4/2022	Ashley R	4.7.1-4.7.7	Initial rough draft of each section
5/5/2022	Ashley R	4.7.3 & 4.7.1	Updated description and validation better reflects the clients software role after real world application

4.8 PCB

4.8.1 Description

The system contains a single PCB that integrates all the components within the Buck Converter, Storage, MCU, and CAN Controller/Transceiver blocks. EAGLE is being used to create the schematic containing all the components of those blocks and is used to design a PCB board. The PCB design is then ordered at OSHPark.com. Maxim Feoktistov is responsible for the PCB block.

4.8.2 Design

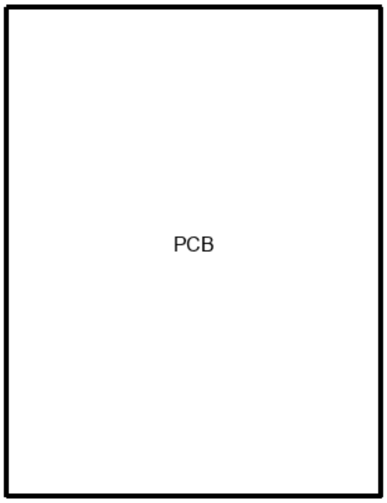


Figure 1: PCB Block

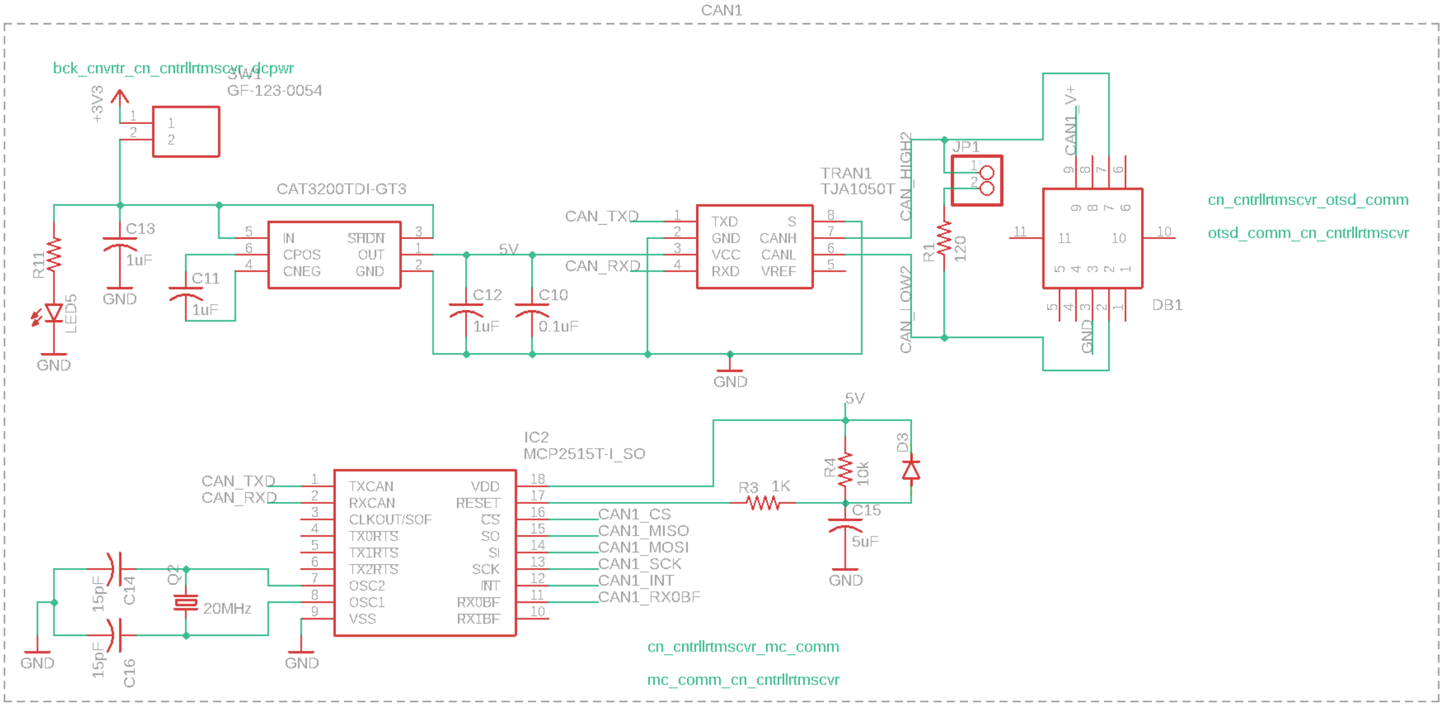


Figure 2: CAN 1 Schematic

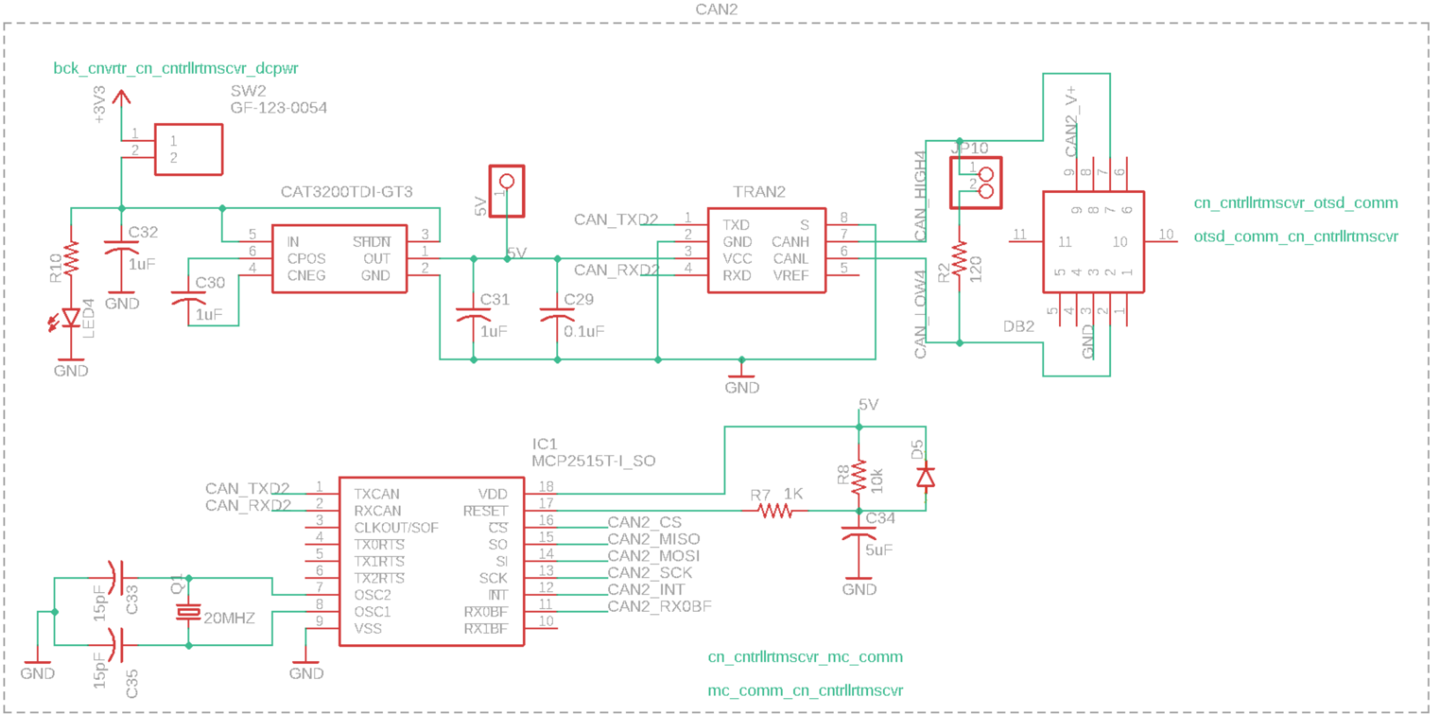


Figure 3: CAN 2 Schematic

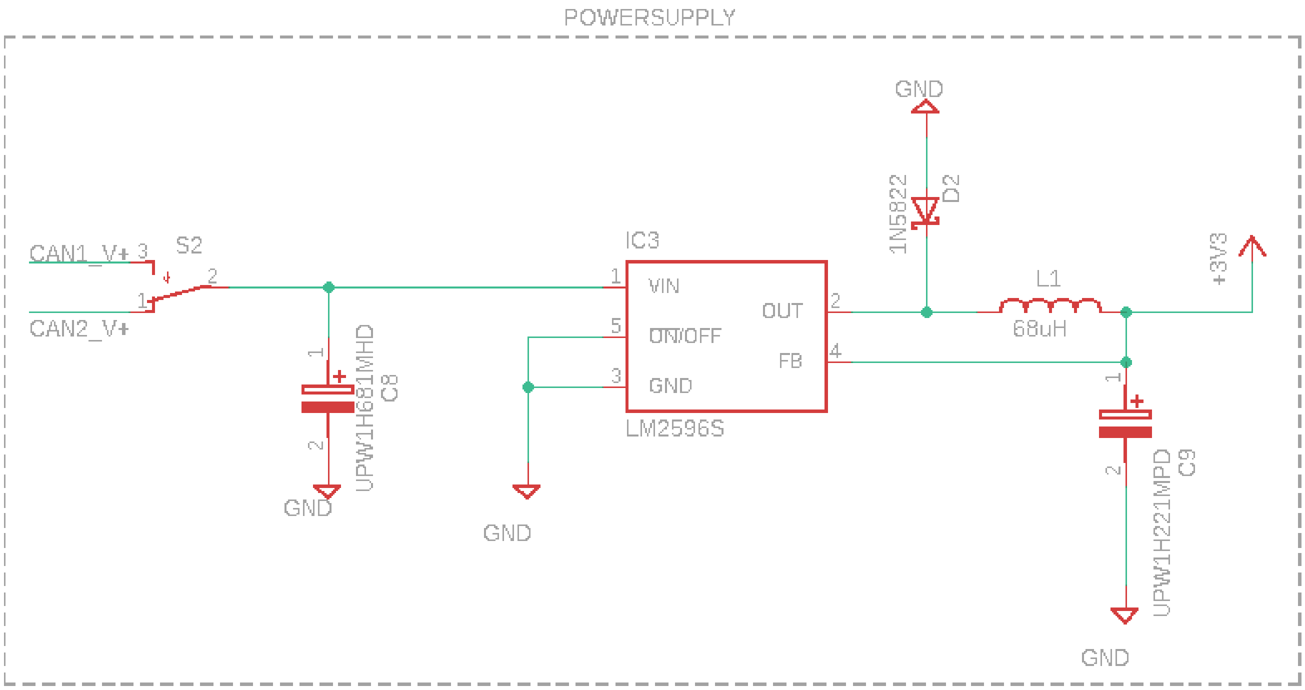


Figure 4: Power Supply Schematic

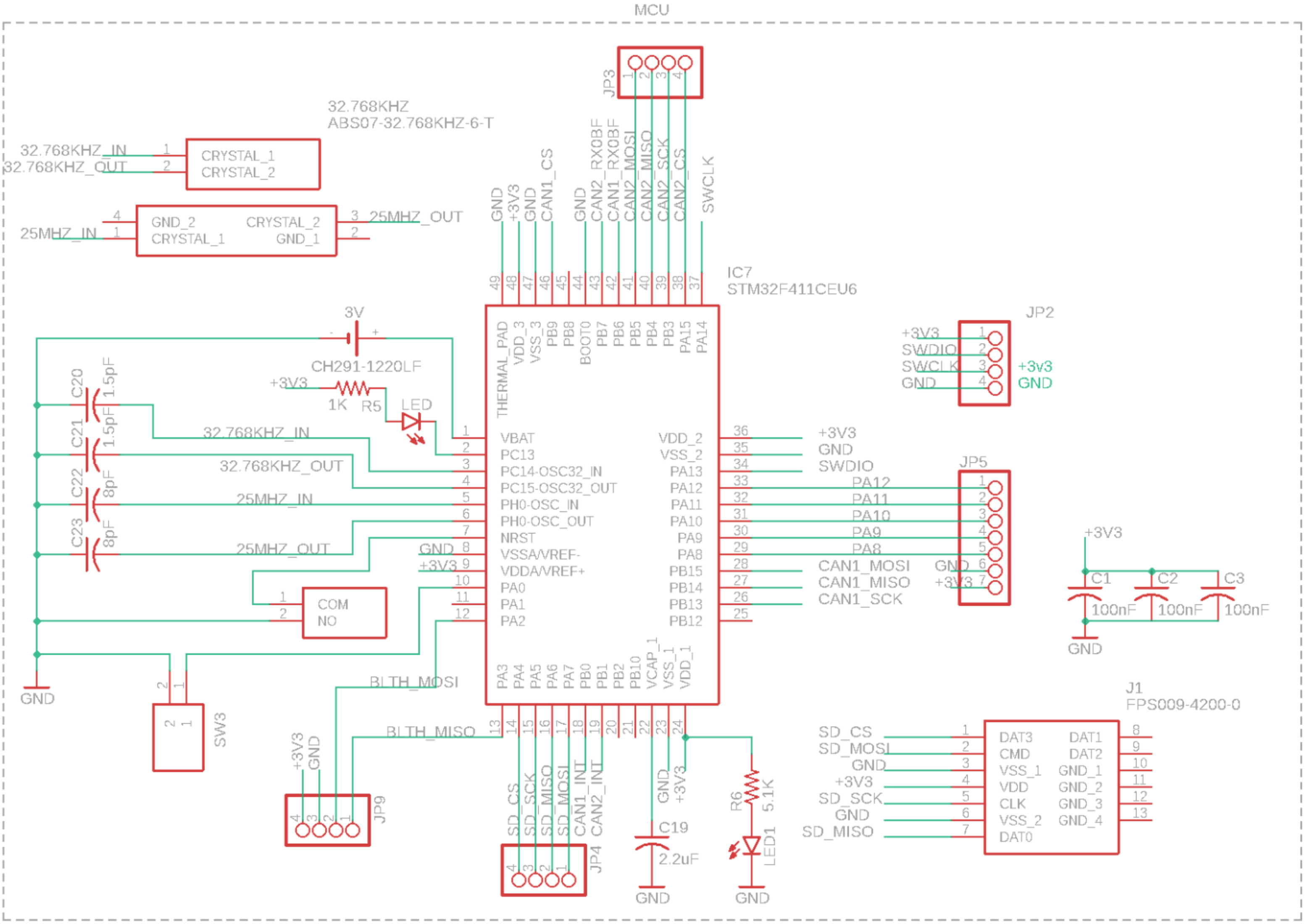


Figure 5: MCU and SD Card Schematic

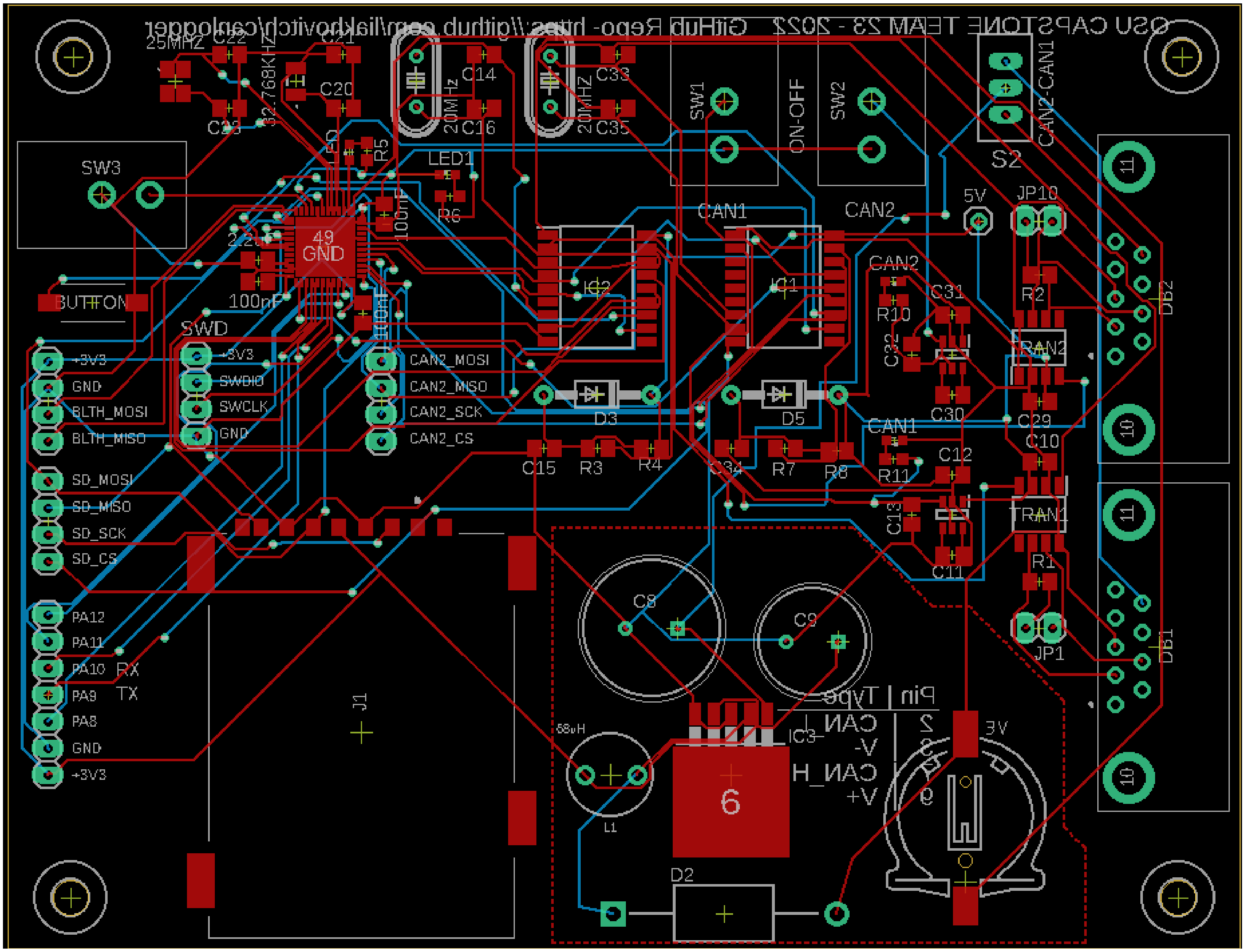
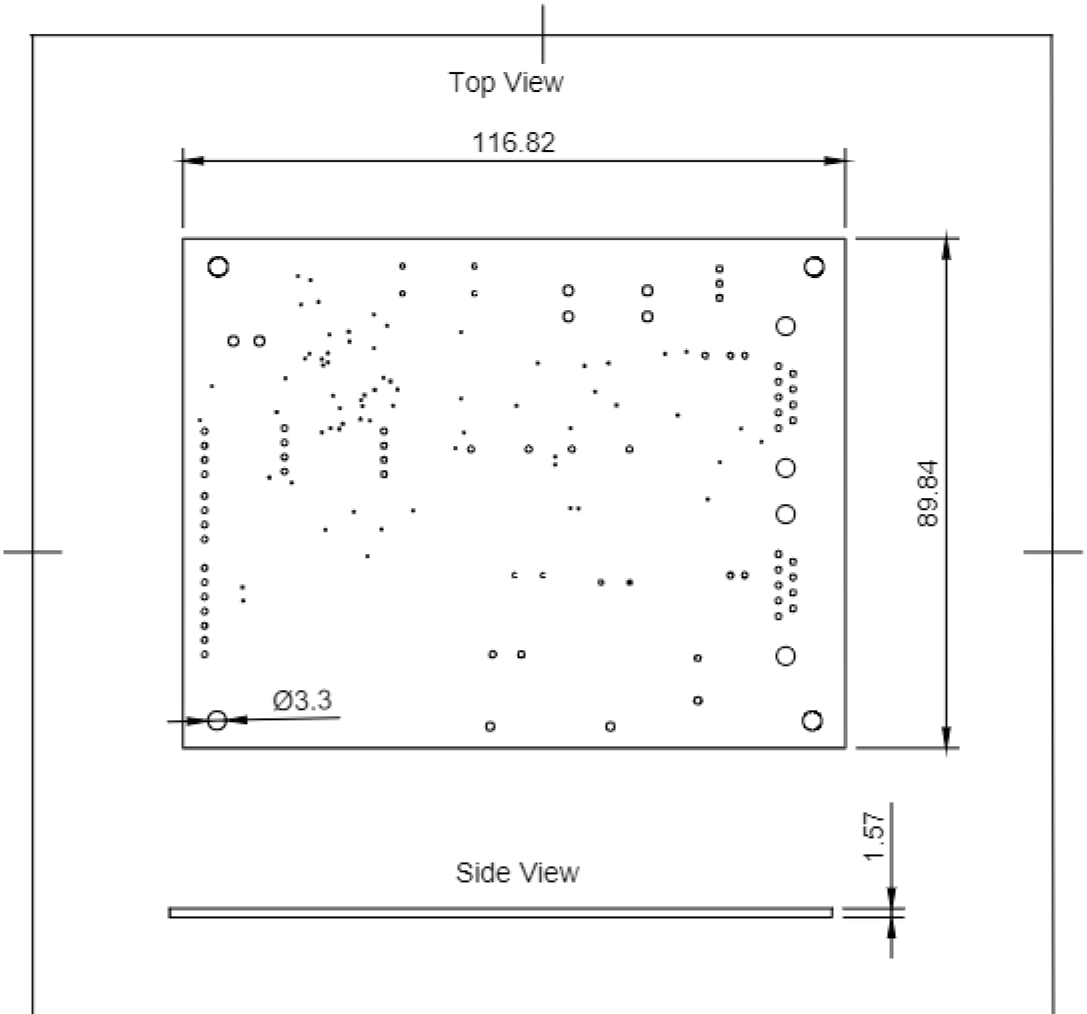


Figure 6: PCB Layout and Design



*Note: dimensions in mm

Dept.	Technical reference	Created by Maxim Feoktistov	Approved by
		Document type	Document status
		Title Project PCB	DWG No.
			Rev. Date of issue 2/18/2022 Sheet 1/1

Figure 7: PCB Dimensions

4.8.3 General Validation

This block consisted of combining already existing schematics into one big schematics. The design schematics used in the PCB block have been validated and verified to work properly and meet the interface property requirements.

The thickness of the traces are 1 oz/ft^2 and the width of the traces vary based on voltage and current. A trace width calculator was used to calculate the necessary width of the traces based on current, voltage, temperature, and trace length. Specifically the Digikey calculator was used [1]. The current and voltage levels are known from previous testing of the independent blocks and the temperature variations are given the datasheets of the components. Heat Sinks are used in areas where there is high current like within the power supply block.

The most popular footprints were picked when designing the PCB board. The site called “Component Search Engine” was used to verify the risk level of choosing a component and the footprint [2]. Low risk level footprints were used for components such as capacitors, resistors, and oscillating crystals. The risk level is determined by multiple factors including part availability, product lifecycle, and price.

The PCB is to be ordered through OSHPark.com. If for some reason OSHPark.com will not be available, the alternative PCB manufacturer that will be used is called JLCPCB. JLCPCB creates quality PCBs with a quick turnaround time but are located in China, so shipping times might take longer compared to OSHPark which is located in the United States.

4.8.4 Interface Validation

*The block does not contain any Interfaces

4.8.5 Verification Process

1. Solder all the components specified in the schematics.
2. Connect the CAN bus to the system via a DB9 connector.
3. Upload necessary Code to the MCU to receive/transmit CAN messages.
4. While data is being transmitted by the CAN bus, record and store the data into the SD card.
5. Verify that data has been saved into the SD card by pulling out the SD card from the PCB and checking data on the SD card via a computer.

4.8.6 References and File links

4.8.6.1 References

[1] “PCB Trace Width Calculator,” *Digikey Electronics*, [Online]. Available:https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width

[2] “Electronic Component Search Engine,” *Component Search Engine*, [Online]. Available:https://componentsearchengine.com/

4.8.6.2 File Links

4.8.7 Revision Table

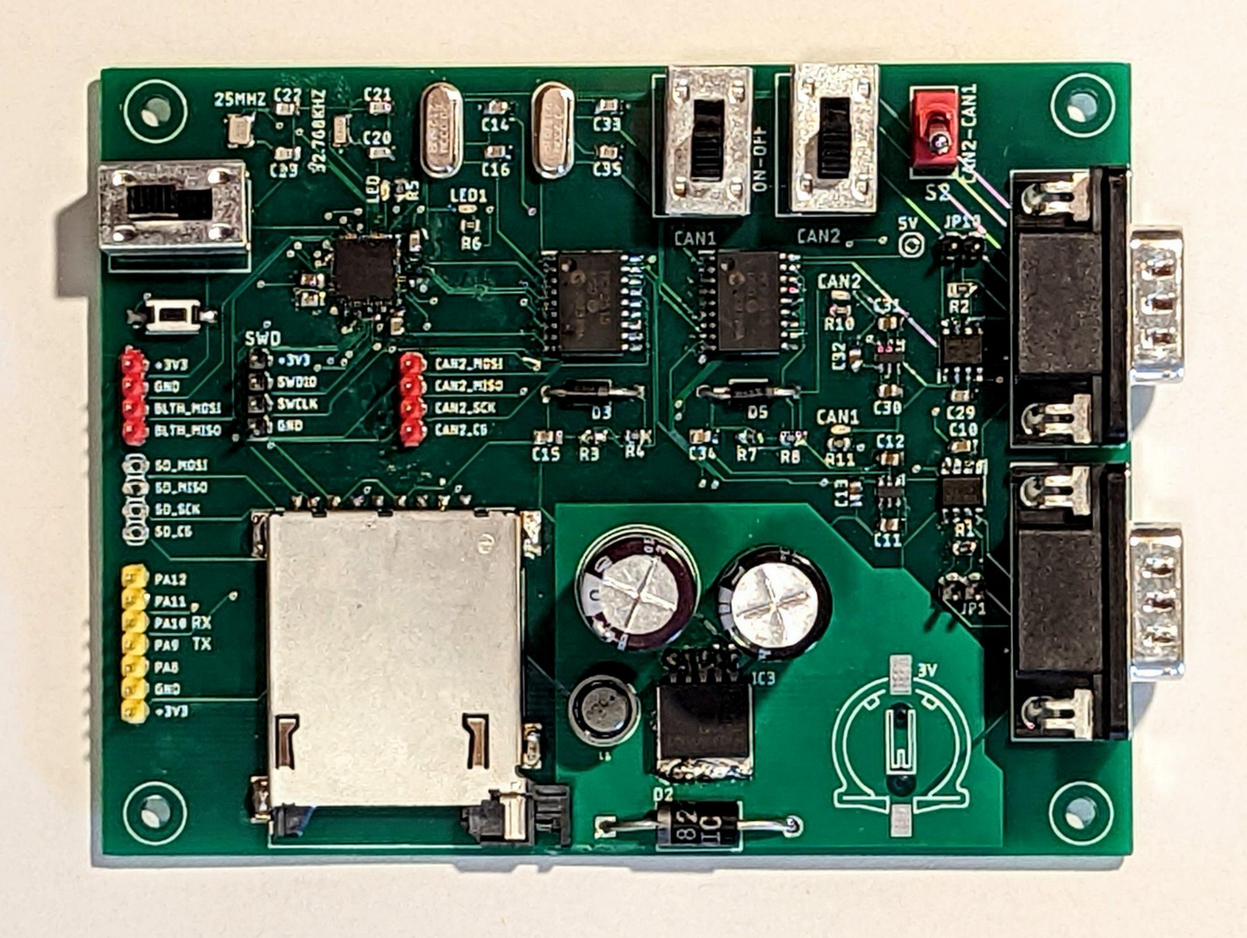
Date	Section Revised	What was revised?
2/1/22	4.8.1	Created document
2/4/22	4.8.1-4.8.7	Created schematics and added information to the document
2/15/22	4.8.2	Created PCB Design and Drawing
2/16/22	4.8.2 & 4.8.5	Revised Schematics and Testing process
2/18/22	4.8.2-4.8.6	Incorporated feedback and revised wording in the sections.
5/6/22	4.8	Updated text and figures

5.0 System Verification Evidence

5.1 Universal Constraints

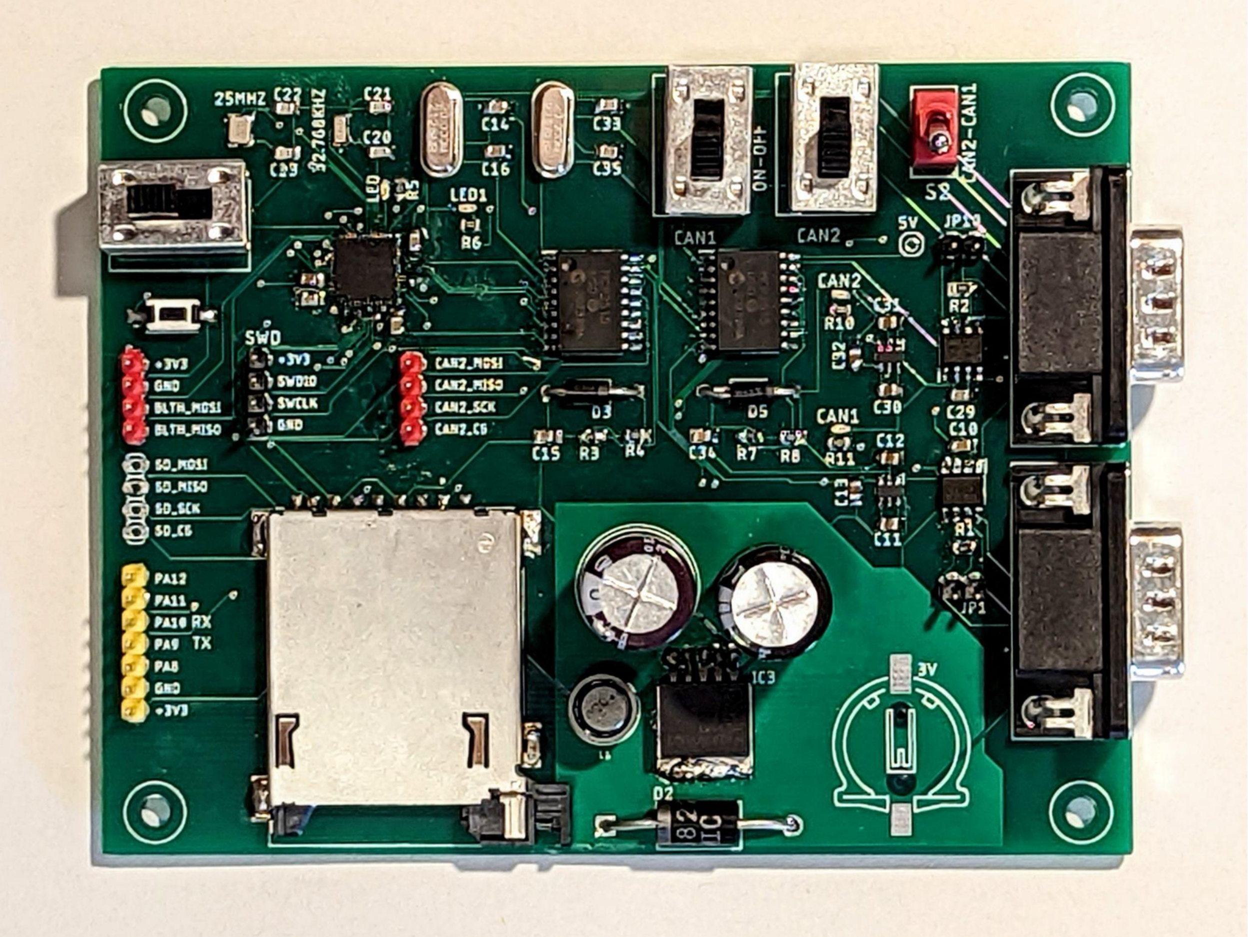
5.1.1 The system may not include a breadboard

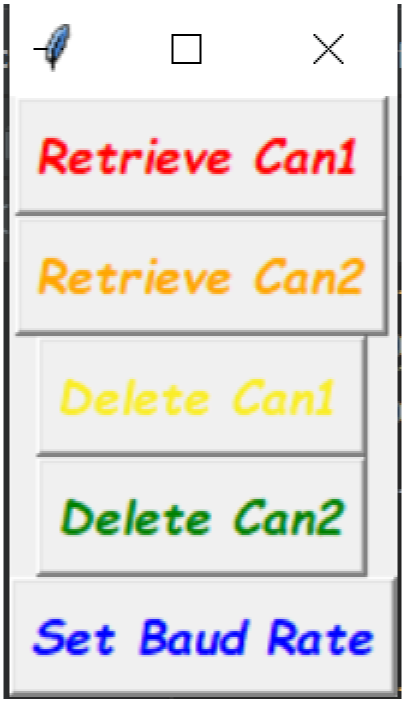
The system uses two PCBs for all components. One PCB is the bluetooth module. The other PCB contains the Buck Converter, Storage, MCU, and CAN Controller/Transceiver Blocks. With all electrical components on these two PCBs there is no need for a breadboard.



5.1.2 The final system contains a student designed PCB, and custom application

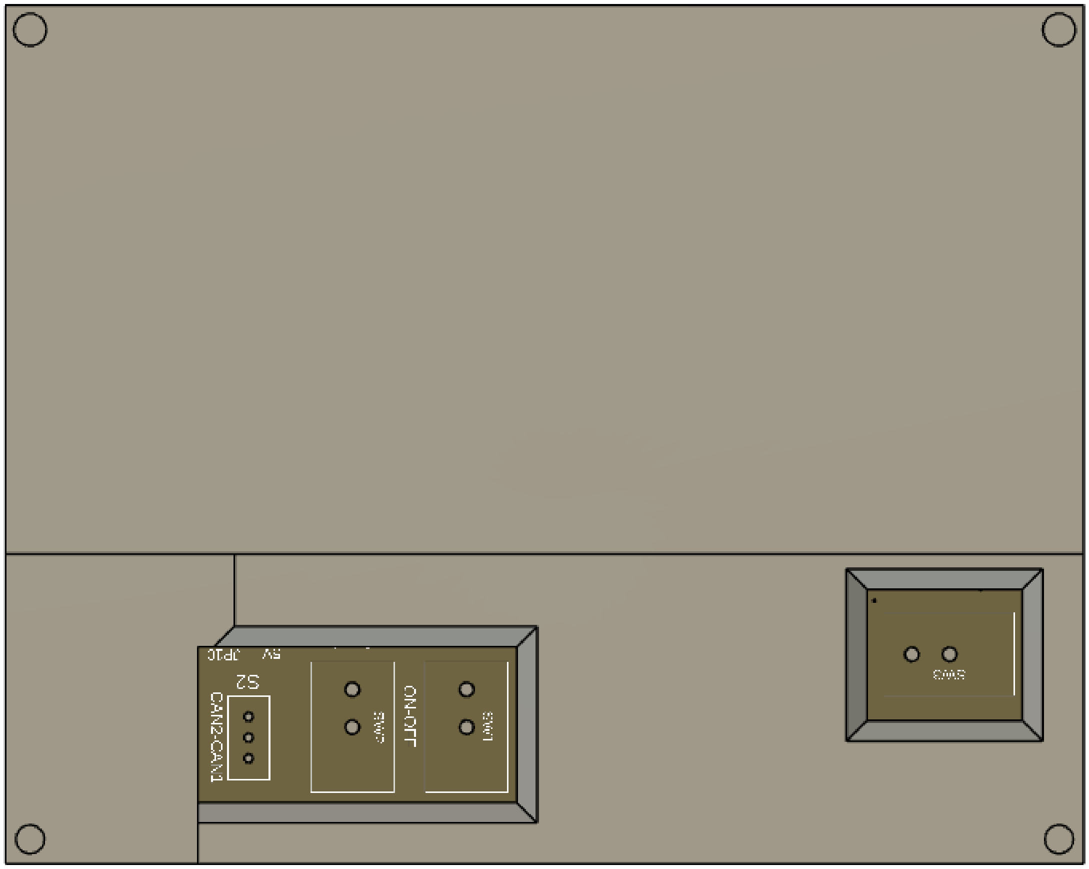
One of the blocks in this system is the PCB, custom-built to hold the Buck Converter, Storage, MCU, and CAN Controller/Transceiver Blocks. Additionally, the system includes Client software which connects to the bluetooth module. This software handles the process of reading data from the system and adjusting the system configuration.

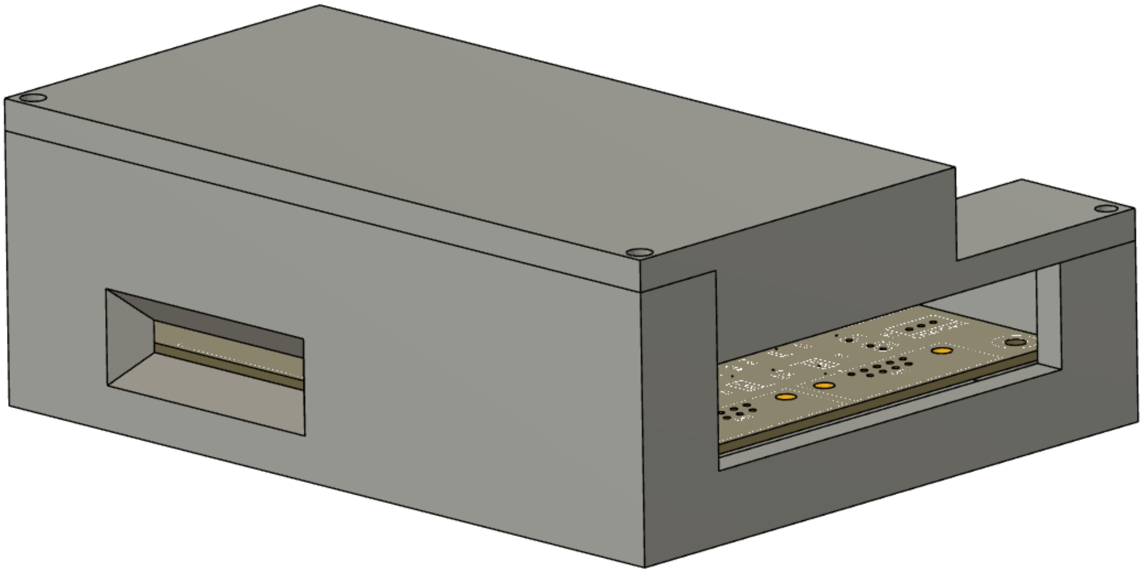

25MHZ
C22
C21
32.768KHZ
C20
C23
C14
C33
C16
C35
ON-OFF
CAN2-CAN1
S2
LED1
R6
CAN1
CAN2
5V
JP1
SWD
+3V3
SWDIO
SWCLK
GND
CAN2_MOSI
CAN2_MISO
CAN2_SCK
CAN2_CS
+3V3
GND
BLTH_MOSI
BLTH_MISO
SD_MOSI
SD_MISO
SD_SCK
SD_CS
PA12
PA11
PA10 RX
PA9 TX
PA8
GND
+3V3
D3
D5
C15
R3
R4
C34
R7
R8
CAN2
R10
C31
R2
C32
C30
CAN1
R11
C29
C10
C12
C13
C11
R1
JP1
IC3
3V
D2

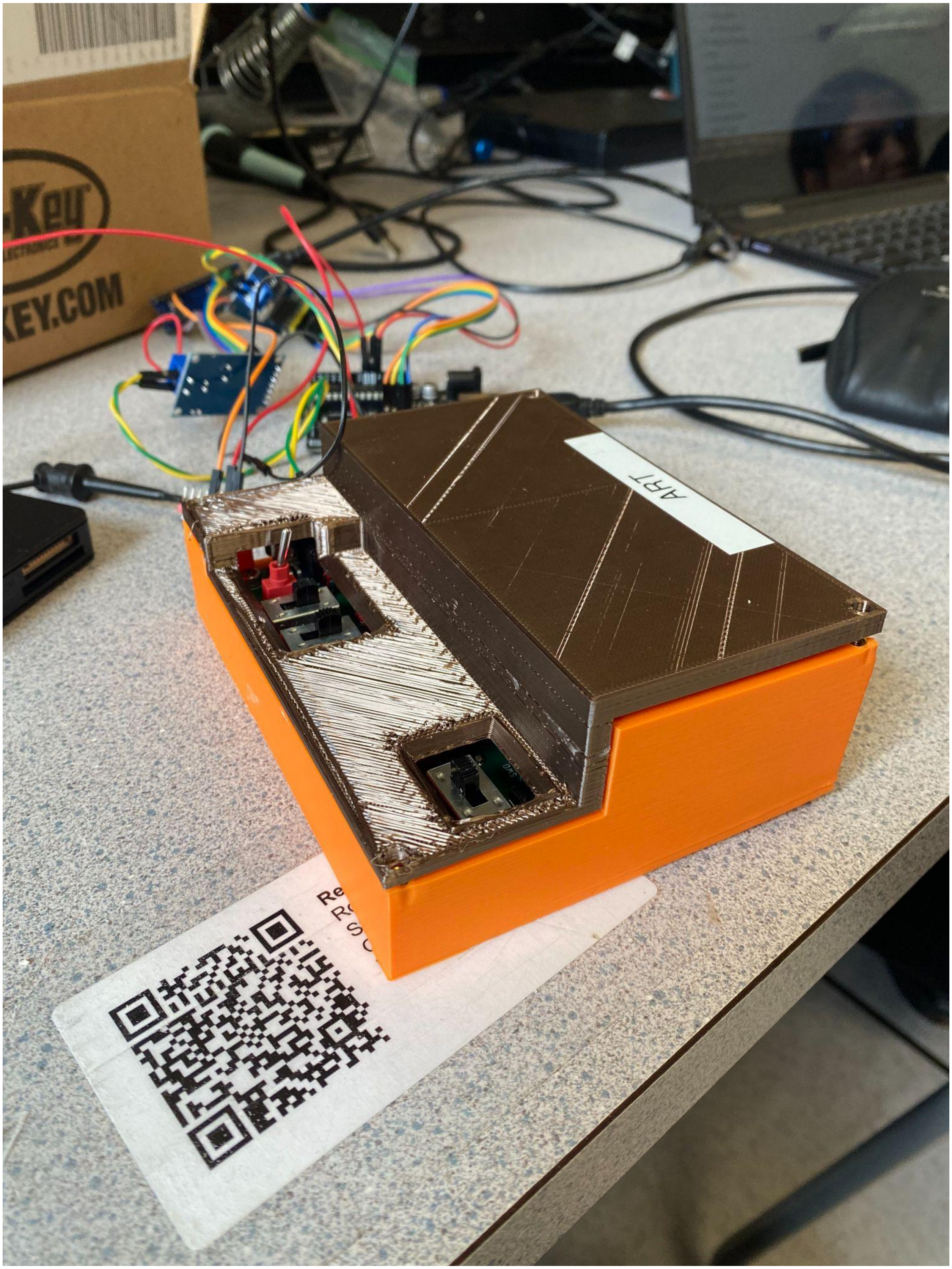

Retrieve Can1
Retrieve Can2
Delete Can1
Delete Can2
Set Baud Rate

5.1.3 If present, the enclosure ruggedly encloses the contents

This system has an enclosure mounting the PCB and bluetooth module. There are openings for the DB-9 CAN connectors and the SDcard. Additionally, there are openings to access the buttons and switches on the PCB.

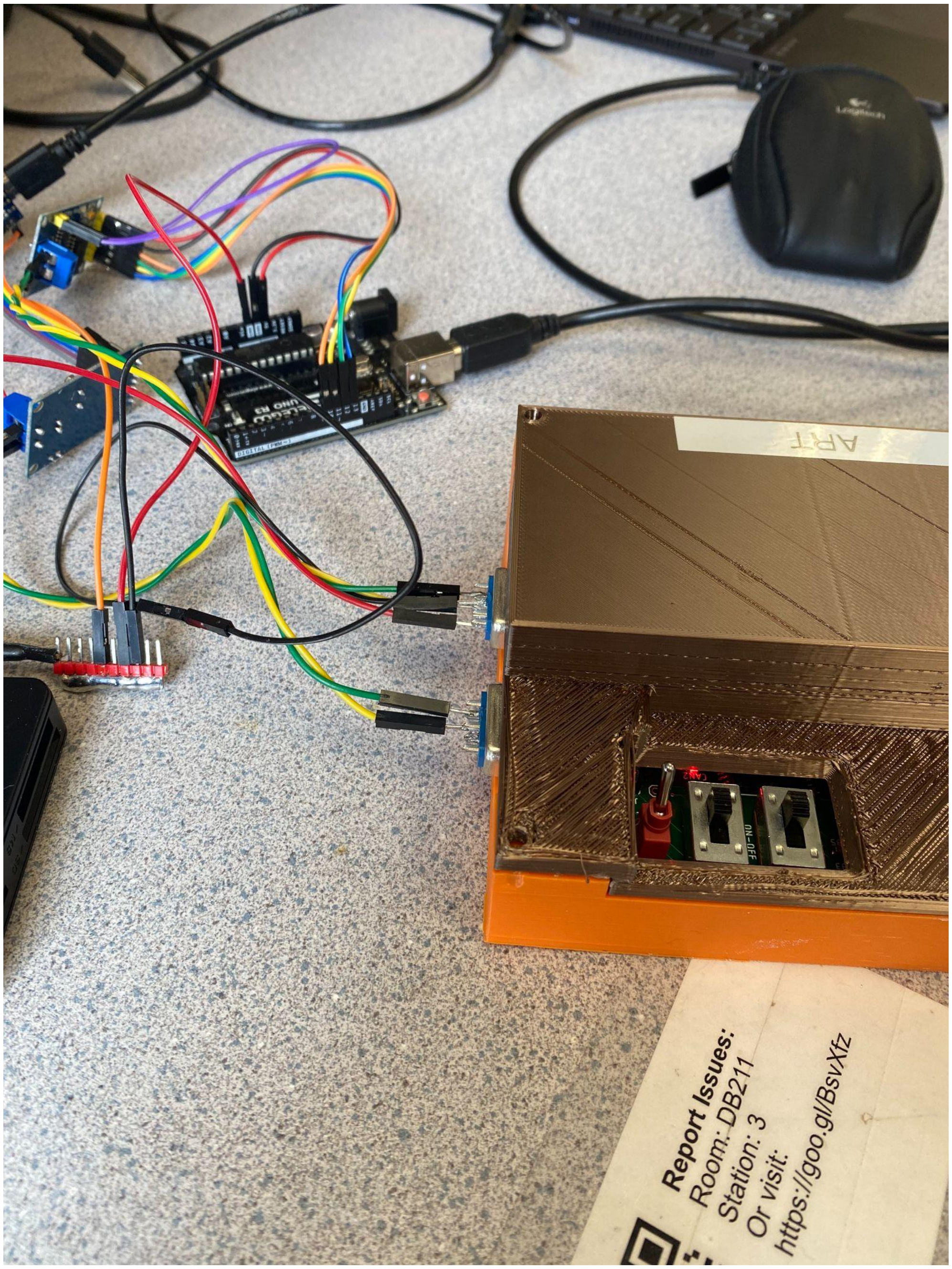

5V JP10
S2
CAN2-CAN1
SW2
ON-OFF
SW1
SW3




ART

5.1.4 All wire connections to the PCB must use connectors

The PCB has two types of connections available: standard header pins for system debug purposes, and two DB-9 connectors for CAN input.



5.1.5 All power supplies in the system must be 65% efficient

The system uses a LM2596 3.3V simple switcher as the power supply. The documentation for the version we used lists a typical efficiency of 73% at 12V input and 3A load[1]. Tests of the power supply for use in our system resulted in a 70% efficiency at 8V input, and 200mA load.

5.1.6 The system may be no more than 50% purchased modules

The system contains a custom PCB which includes 4 out of the 5 physical blocks that the project contains. Therefore, the system consists of 20% of purchased modules.

5.1 Interface Bus

5.1.1 Requirement

PPR: Interface with 2 or more CAN buses.

ER: *The system will support at least two CAN channels*.

5.1.2 Testing Processes

Verification procedure:

1. Connect both CAN channels of the CAN Logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously.
3. Run the test bench and record data with the CAN Logger.
4. Verify that the data received by the CAN Logger is the same data sent by the test bench.

Test pass condition: The received data must be identical to the sent data.

5.1.3 Testing Evidence

https://youtu.be/ApjvbllP6sc

This video shows us going through the process of connecting two CAN channels to the system. Then we sent two different verifiable sets of CAN data to the system over the channels. It compares the received data to the master data.

5.2 Interface Type

5.2.1 Requirement

PPR: Interface with both J1939 and CANopen.

ER: *The system will log both J1939 and CANopen.*

5.2.2 Testing Processes

Verification procedure:

1. Connect both channels of the CAN logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously. The data sequences must use both standard and extended identifiers as required by CANopen and J1939 respectively.
3. Run the test bench and record data with the CAN Logger.
4. Verify that the data received by the CAN Logger is the same data sent by the test bench.

Test pass condition: The received data must be identical to the sent data.

5.2.3 Testing Evidence

https://youtu.be/ApjvbllP6sc
In this video we connect the CAN logger to the test bench. Both testers send distinct CAN frames. One tester sends CAN data using the standard size frame used with CANopen, the other uses the extended size frame for J1939. Looking at the data in the end you can see that both types are received and logged correctly.

5.3 Device Storage

5.3.1 Requirement

PPR: *Store captured data on an SD card.*

EP: *The system will record data to two separate files.*

5.3.2 Testing Processes

Verification procedure:

1. Connect both CAN channels of the CAN Logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously.
3. Run the test bench and record data with the CAN Logger.
4. Check the data of the SD card, there should be two files saved on the SD card.

Test pass condition: The SD card must contain two files of CAN data.

5.3.3 Testing Evidence

https://youtu.be/ApjvbllP6sc

This video shows both can loggers connected to the system. They send two distinct verifiable data sequences on both channels. After running the test bench it shows the data stored on the SD card. It contains two files with CAN data that matches the master data.

5.4 Firmware Accessibility

5.4.1 Requirement

PPR: *The user should be able to update the firmware*

EP: *There will be a user guide that will provide information on how to update the system firmware.*

5.4.2 Testing Processes

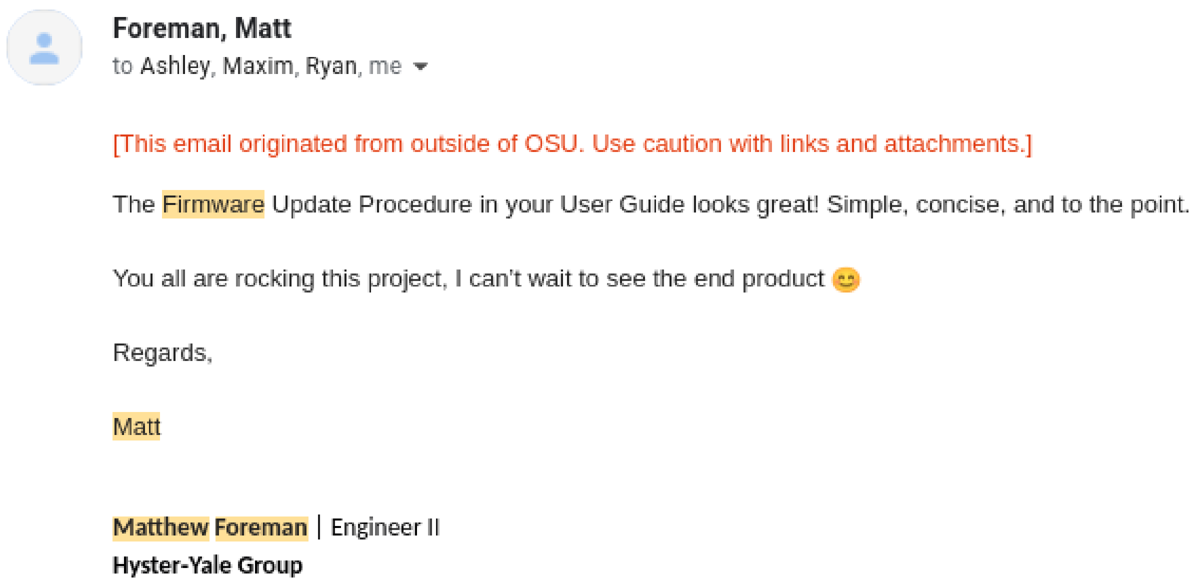
Verification procedure:

1. Share the user guide with the project partner.
2. Get feedback from the project partner.
3. Repeat steps 1 and 2 until the project partner approves the final edition.

Test pass condition: Project partner approves user guide.

5.4.3 Testing Evidence

https://drive.google.com/file/d/1FIcX40qcXUKBXNWCmRV8rS2qejBXqxGv/view?usp=sharing



Foreman, Matt
to Ashley, Maxim, Ryan, me ▾

[This email originated from outside of OSU. Use caution with links and attachments.]

The Firmware Update Procedure in your User Guide looks great! Simple, concise, and to the point.

You all are rocking this project, I can't wait to see the end product 😊

Regards,

Matt

Matthew Foreman | Engineer II
Hyster-Yale Group

5.5 Power Supply

5.5.1 Requirement

PPR: *The device should operate on a voltage range of 8-32v and draw a maximum of 0.375A*

EP: *The system will operate within the following power supply requirements:*
Vmax: 35V
Vmin: 8V
Inominal: 55mA
Ipeak: 500mA

5.5.2 Testing processes

Voltage verification procedure:

1. Use a power supply to power the CAN logger.
2. Set the power supply to 8V and use the test bench to send data to the CAN logger to check for proper operation (the definition of "proper operation" depends on the currently loaded firmware).
3. Set the power supply to 32V and use the test bench to send data to the CAN logger to check the power supply.

Test pass condition: The received data must be identical to the data sent by the test bench.

Current verification procedure:

1. Connect the CAN logger to a power supply.
2. Power the CAN logger from a 14V power supply. Assuming that the device draws a relatively constant amount of power. This is the expected voltage on the vehicles.
3. Set the test bench to continuously send random data on both CAN channels. This data will not need to be verified.
4. Run the logger for 30 seconds, taking current draw measurements once every second.

Test pass condition: The average measured current does not exceed Ipeak, and does not deviate from Inominal by more than 10mA for longer than five seconds for the entire test duration.

5.5.3 Testing Evidence

Interface Property	Why the value was used	Details that support this value
Inominal: 375mA	This current was chosen using the power limit of 3W for the system. This	For the LM2596 in the TO-263 package: ● Component is rated to 3A (Electrical Characteristics – 3.3-V Version, pg. 6) [1]

	would be the current highest regular current.	
Ipeak: 500mA	During peak usage if the system becomes a 4W load this would be the highest current it would need to handle.	For the LM2596 in the TO-263 package: ● Component is rated to 3A (Electrical Characteristics – 3.3-V Version, pg. 6) [1]
Vmax: 35V	This voltage was chosen as being able to run off 35V maximum was a system requirement.	For the LM2596 in the TO-263 package: ● Max voltage input is 40V (Electrical Characteristics – 3.3-V Version, pg. 6) [1]
Vmin: 8V	This voltage was chosen as being able to run off 8V minimum was a system requirement.	For the LM2596 in the TO-263 package: ● Min voltage input is 4.75V (Electrical Characteristics – 3.3-V Version, pg. 6) [1]

*Referenced Section 4.1.4

5.6 Client Software

5.6.1 Requirement

PPR: *Wirelessly transfer collected data to a computer.*

ER: *The system will output a CSV file on the client computer.*

5.6.2 Testing Processes

Verification procedure:

1. Connect both CAN channels of the CAN Logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously.
3. Run the test bench and record data with the CAN Logger.
4. Use the CAN Logger PC software solution to download the data from the device over bluetooth. Use either a Windows or Linux system for this test.
5. Via inspection, verify that the data is in proper CSV format.
6. Verify that the received data is the same as the data sent by the test bench.

Test pass condition: The received data must be identical to the sent data when using both Windows and Linux systems.

5.6.3 Testing Evidence

https://youtu.be/ApjvbllP6sc
In this video we run the test bench sending data to the system. The Logger records and stores the data. With the PC software we connect to the system over bluetooth. Here we download the csv files, and compare them to the master files. They match so this requirement is met.

5.7 Invalid Data

5.7.1 Requirement

PPR: *The device should handle invalid packets without shutting down.*

EP: *The system will handle invalid packets and continue to operate normally.*

5.7.2 Testing Processes

Verification procedure:

5. Connect both CAN channels of the CAN Logger to the test bench.
6. Set the test bench to send a sequence containing at least one type of invalid data packets. Invalid packets may include:
    a. Packets that are too short.
    b. Packets that are too long.
    c. Packets sent at the wrong baud rate.
7. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously.
8. Verify that the CAN logger has logged or ignored the invalid data, and continued to log the subsequent valid data.

Test pass condition: The received data must be identical to the sent data, except for the erroneous packets. Erroneous packets must be ignored or marked by an appropriate error message in the CSV file.

5.7.3 Testing Evidence

https://youtu.be/sfDMEtvizoI
Setting the system to communicate at a different baud rate as the test benches. Send a test sequence to the system. We checked that the system didn't receive any packets. We then set the system to the same baud rate as the test benches. We send a second test sequence to the system. The system manages to store data after a barrage of invalid packets.

5.8 Data Accuracy

5.8.1 Requirement

***PPR**: The device should not miss packets.*

***ER:** The system will process input data at a rate of at least 1 Mb/s.*

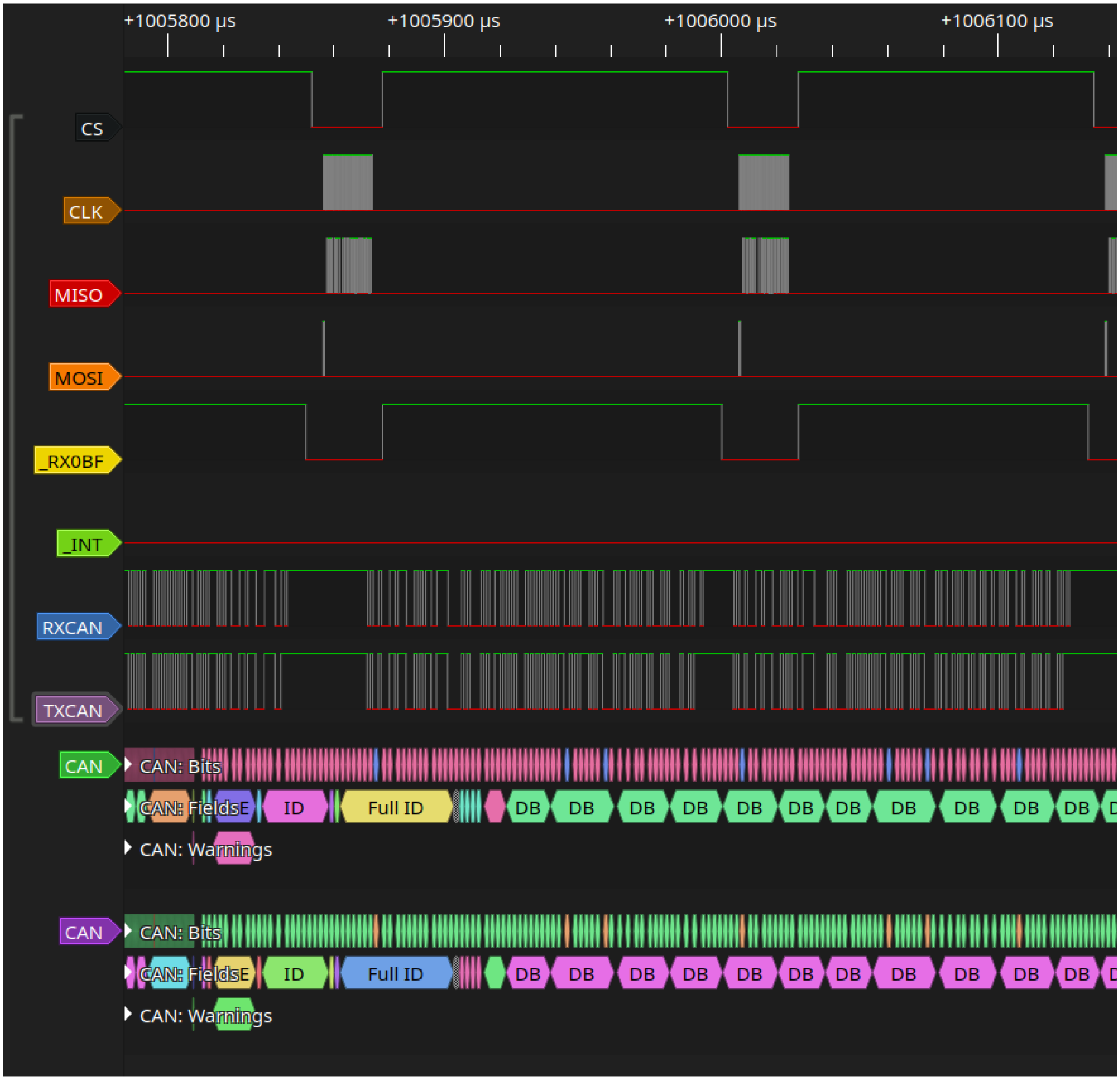
5.8.2 Testing Processes

Note: 1 Mb/s refers to the baud rate, not the data rate.
Verification procedure:

1. Connect both CAN channels of the CAN Logger to the test bench.
2. Set the test bench to simultaneously send two distinct, arbitrary and verifiable data sequences on both channels simultaneously. The data shall be sent at the maximum possible baud rate (1 megabaud/s). Run the test bench and record data with the CAN Logger.
3. Verify that the received data is the same as the data sent by the test bench.

Test pass condition: At least 99.9% of received data must be identical to the data sent by the test bench.

5.8.3 Testing Evidence



5.X References and File Links

5.X.1 References

[1] “LM2596 Simple Switcher,” *Texas Instruments*, November 1999, [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2596.pdf?ts=1642779956402&ref_url=https%253A%252F%252Fwww.bing.com%252F

5.X.2 File Links

5.5 Revision Table

Date	Name	Section Revised	What was revised?
3/5/2022	Ryan Dillard	5	Created section, Initial formatting, and responses to Universal Constraints
3/5/2022	Maxim Feoktistov	5.2-5.3	Created the Sections
3/5/2022	Anton Liakhovitch	5.3	Added testing evidence
3/14/2022	Anton Liakhovitch	5	Changed power testing procedure
4/21/2022	Ryan Dillard	5	Added Sections for all requirements, and placed old processes
5/2/2022	Ryan Dillard	5	Added Test Evidence to Requirements
5/2/2022	Ashley Reid	5.7	Revised wording
5/5/22	Anton Liakhovitch	5.1	Edited for clarity

6.0 Project Closure

6.1 Future Recommendation

6.1.1 Technical Recommendation

1. Minor PCB issues:
    a. Flip the TX/RX pins for the Bluetooth module on Project PCB.
    b. Add more ground pins to PCB for testing purposes
    c. Separate 5V line between CAN channels so that each CAN channel can be individually powered and controlled by a (on/off) switch
2. Current design does not implement any security to the PCB in case of power loss which can negatively affect the SD card. This issue can be fixed by adding an uninterrupted power supply so that the system can safely unmount the SD card on power loss.
3. The SD card pins are directly connected to the MCU and contain floating values during power on. MCU pull up resistors are not set up until the MCU initializes them. This is long after the SD card powers on and as a result becomes unresponsive. It is recommended to add pull-up resistors to the SD card inputs to eliminate any floating nodes.
4. Implement time stamps and RTC (realtime clock). The system should periodically include timestamps with the data, so that engineers can determine when a particular CAN message was sent. While it is impractical to include a timestamp with every message, it is possible to add a timestamp every couple of seconds. The system already includes the necessary RTC hardware, so the feature can be implemented via a firmware update.

6.1.2 Global Impact Recommendation

1. Make PCB smaller to decrease e-waste.
    1.1. The current PCB design is focused around development and system debug, with extra pinouts, and switches. This has resulted in a much larger PCB than necessary. As a result the system generates much more e-waste than needed. The system design could be optimized and the PCB could easily be half the size. To do so we recommend using both sides of the PCB in addition to the removal of most header pins on the PCB.
2. Use unleaded solder.
    2.1. Lead is known to have long-lasting negative effects on the human body. Using unleaded solder to assemble the system can drastically reduce the negative impact the system will have on the environment at the end of its life cycle.
3. Bluetooth Security.
    3.1. The current solution uses a bluetooth connection that anyone can connect to, and uses limited access to the system, and area around the system as the main

method of security. If this device is to be used in an open access location it would allow malicious actors to take advantage of the system. If this occurs the best case scenario would give the malicious actors access to proprietary information, or the ability to prevent the tool from logging. If sending remote frames is set up on the system it would be used to damage equipment in the worst case. This would be solved by implementing authentication on connection to the device.

6.1.3 Teamwork Recommendation

1. Have a project manager to enforce usage of correct tools.
   1.1. When starting to work together the team setup several tools to help organize works, tasks, and communicate. Trello quickly was thrown to the side and GitHub wasn't enforced until late into the project. Without Trello many tasks were solved last minute, and responsibilities weren't clear at times.
2. Set up everyone's GitHub during the same meeting.
   2.1. During this project, each team member had different pieces of code to write for the system. Teams that members of different programming backgrounds can have varying levels of Git experience. Due to version control being extremely important with these long term projects, setting up the Git Repo with everyone in a real time meeting decreases the amount of confusion and more effective collaboration. It will also decrease the amount of time spent on block integration.
3. Weekly team meetings discussing current progress
   3.1. Sometimes during the project, the team didn't have weekly meetings to discuss current progress of the project and how everyone in the team is doing. As a result, there were times where the team members were not aware of what progress the team was making and if the team was meeting deadlines. Therefore, it is recommended to have weekly meetings where each team member can talk about their progress and any questions or concerns they may have.
4. The team should review the designed PCB in detail before ordering the PCB.
   4.1. The initial version of the PCB has had some bugs and issues that could have been prevented with more research and team collaboration. It is recommended to go over each component as a team to prevent having issues with the Project PCB.

6.2 Project Artifact Summary with Links

Artifacts Availables in GitHub Repo Include:

- PCB Schematic and Design
- Enclosure Design

- Firmware

Github Repository - https://github.com/liakhovitch/canlogger

- Currently unavailable pending an operational security review by Hyster-Yale.

6.3 Presentation Materials

Expo Poster:
https://docs.google.com/presentation/d/e/2PACX-1vSz-24G3YwC5kZ3OaclfZyMUAA1-Ca7dknDKNdYjyoMdBoFacnaVEY-ay7nTlVYJw/pub?start=false&loop=false&delayms=3000

Presentation:
https://docs.google.com/presentation/d/e/2PACX-1vQDI0ajRa5YeW5vHCVNoMRtvEUJESIgYYAtoO8_Il0t7wnIarvxjMIH9nnjCi8__476ojC_0D0DCrK5/pub?start=false&loop=false&delayms=3000

Revision Table

Date	Name	Section Revised	What was revised?
05/02/2022	Max	6	Created Section
05/04/2022	Max	6.1	Added more detailed explanations
5/5/2022	Ryan	6.1	Explained some recommendations
05/05/2022	Max, Anton, Ashley, Ryan	6.1	Added more detailed explanations