PROJECT CANDL

Final Report

November 23, 2020

Jack Burrows, Chris Hill, Asa McVay,
Ethan Mitchell, Desmond Stevens,
Kitae Woo, Mohammad Zuaiter

Executive Summary

Club Car manufactures light personal transportation vehicles for the golf, utility, and consumer industries. Currently, the Test Engineering department at Club Car relies on third-party data loggers for field testing. The Controller Area Network (CAN) bus on the cars transmit important vehicle information, such as battery pack voltage and throttle input. This information is currently sent to a remote server over a 3G cellular network, where Club Car engineers can pull the data for analytics and design decisions. The current loggers cost roughly $800 each, preventing more widespread field-testing data.

Club Car approached the Auburn capstone design team with the idea to create an in-house solution for a data logger. This logger should serve as a replacement for the third-party ones used in the field today, with a target cost per unit of $50, not including housing. The project would be able to connect to any 4G LTE network, as well as Wi-Fi for improved connectivity. The device would be able to read the vehicle CAN bus and store then transmit messages to a server. Once on the server, the data should be processed and analyzed. The finished device would then be used on field cars for testing, and with a reduced price (from $800 to $50), more cars could be outfitted with loggers. With these requirements, the project team began Project CANDL – CAN Data Logger.

Project CANDL aims to design a data logger that is capable of reading Club Car vehicle information and transmitting that information remotely to a server where the data can be processed. Relevant information to be collected includes messages from the CAN bus, external sensor data, and location data. This project will produce a compact, affordable data logger with the specifications detailed in this document. The design will consist of multiple wireless radio modules, a microcontroller, and other essential embedded circuit design components. Custom software will be written for this device utilizing open-source tools. Back-end software is in scope, with the goal of taking the transmitted data and processing and reporting it in an organized fashion. The final solution then will be a production-ready PCB for the data logger device, the embedded software for the device, and the back-end software and server set-up required for reporting on the data collected. Other deliverables include the development boards used during design, schematic and printed circuit board files, and all documentation relating to design.

Table of Contents

Introduction

Club Car currently uses the HEM Data OBD Logger. This device has three CAN ports, 4GB of storage, 3G cellular support, Wi-Fi 802.11 b/g/n, optional GPS, and an accelerometer. The HEM logger operates from a 4V to 36V supply and features a sleep mode. The product sells for about $800 and includes some software, but the large entry price prevents Club Car from installing more data loggers on field cars. Project CANDL offers a replacement to the HEM data logger, one that is designed in-house from scratch, providing significant upgrades and cost savings.

The data logger being designed for Project CANDL will target $50 per device, a steep decrease in price from the HEM logger's $800. This will allow more data loggers to be installed on test vehicles and is low enough that Club Car could consider adding these devices to production cars in the future, without increasing the retail price. Project CANDL will provide research, component selections, schematic and PCB designs, as well as software for the HEM data logger replacement. Future improvements will be recommended, as the project's scope is to produce a working proof of concept. Two hardware designs will be suggested: a full design with all components, and a test board consisting off all components except the wireless modules. The second design will be intended for testing with cellular development kits in the event suppliers may change or other options are considered in the future.

Hardware Design

Objectives

Ethan Mitchell, Kitae Woo, Desmond Stevens

For Project CANDL, the data logger will only be physically connected to a Club Car vehicle. This means all power must either come from a battery or from the car. In the field, a logger may be left in vehicle for weeks at a time, so connecting the vehicle for power was preferred. Every electric Club Car vehicle has a 12V power line which was used for the input of the device described in this report. 12V is too high for most chip-level components, so various voltage supplies will be necessary. The current supply of each line is an important specification that should be considered when selecting specific components. These lines are detailed more in the "Power Supplies" section.

The microcontroller is the heart of this project. It should have the ability to run many tasks. There was a lot of communication with the software team to make sure the microcontroller could deliver in terms of performance. Therefore, it required many functions and abilities. Enough clocks, capable processing speed, ability to read and deliver data, and abundant interfaces. These requirements are detailed more in the "Microcontroller" section. Since only a CAN controller was built into the microcontroller, a CAN transceiver was also required. It did not require many specs since it only needed to read data. To save some budget, a level shifter was separately purchased.

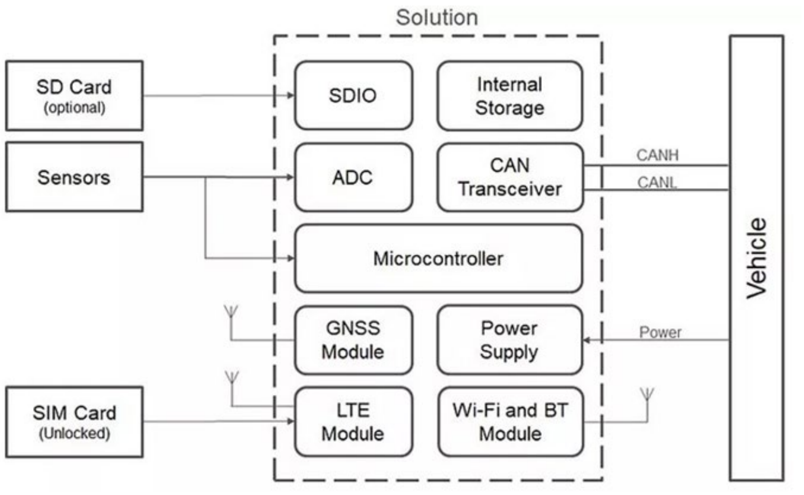


Figure 1: System Block Diagram

Design Process

Power Supplies

Ethan Mitchell

For Project CANDL, there are several power domains that are required. The microcontroller, CAN transceiver, level shifter, and Wi-Fi module all require a 3.3V supply. The cellular module needs 3.8V, and the peak input current could be as high as 2A. A 5V supply is necessary for the CAN transceiver, as is a 1.8V line for both the level shifter and Wi-Fi module. The input for the data logger is 12V from Lithium-powered vehicles. Table 1 shows power requirements for all components, taking into consideration maximum current draws. This initial gathering of power requirements was extremely helpful when selecting components.

Table 1: Power Requirements

Name	Voltage	Typ. Current	Max. Current	Comments
STM32L433RC	3.3V	18.4 mA	100 mA	Max on supply pin, Typ. is Run Mode with Flash at 25°
MCP2542FD	5V	2.5 / 55 mA	5 / 70 mA	Main supply, recessive / dominant
	3.3V	7 / 200 mA	20 / 400 mA	IO supply
TXS0108EPWR	3.3V		6 µA	
	1.8V		2 µA	
EC21-V	3.8V	1.8A	2A	Peak supply during data transmission
FC20	3.3V	372 mA		Main supply
	1.8V	Not specified		IO supply

The 3.3V and 3.8V lines needed to be able to supply at most 872mA and 2A, respectively, given a 12V input. With these specifications, the Analog Devices LTC3636 was chosen. Multiple options were considered, but due to previous experience, Analog Devices chips were preferred. The LTC3636 is a dual-channel switch mode power supply that allows for an 8V to 20V input. Each output rail can supply up to 6A, which is well above the maximum current expected in this application. This makes the LTC3636 a suitable choice for this application. The chip is $4.45 at 1000 units. The additional components needed to produce a 3.3V and 3.8V output were next calculated. The calculations are detailed in the "Additional Figures and Suppoting Materials" section in the appendices.

The circuit was constructed in LTspice for simulation and verification of component selection. The schematic is shown in Figure 2, using recommended components[1] for parts not calculated above. After running a transient simulation with an input of 12V, the outputs were 3.297V and 3.791V. The transient simulation for both outputs supplying a 2.5A draw is shown in Figure 17 in the appendices. The LTspice simulation verifies the LTC3636 design.

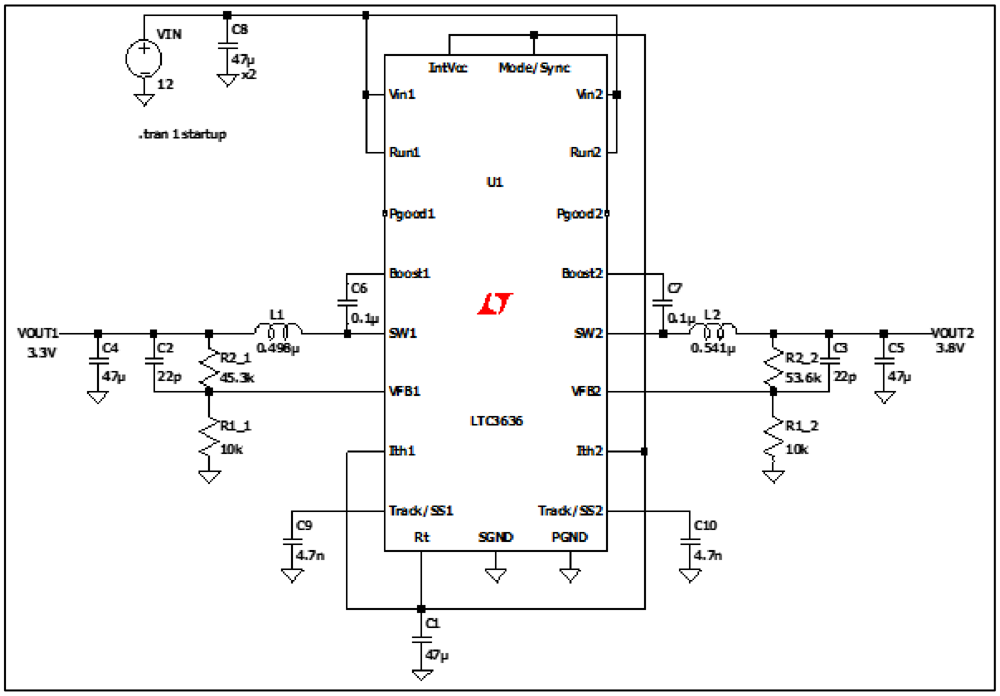


Figure 2: LT3636 Schematic in LTSPice, 3.3V and 3.8V

Since maximum current draw was lower for the 1.8V and 5V lines, a linear dropout regulator was chosen for each. The Analog Devices LT3080[2] is an adjustable LDO that allows up to 1.1A on the output. Various LDOs could have been chosen, but since only typical, not maximum, currents for the 1.8V and 5V devices were specified, an LDO capable of supplying around 1A was selected to ensure compatibility. On the LT3080, the voltage output is determined by a single resistor, simplifying design. The LT3080 was selected for both the 1.8V and 5V supplies due to its simplicity and low cost of $1.08 at 1000 units.

The resistor that sets the output voltage is determined by the following equation: $R_{SET} = \frac{V_{OUT}}{10\mu A}$. Figure 3 shows the schematic for the 1.8V supply, and Figure 4 shows the 5V supply.

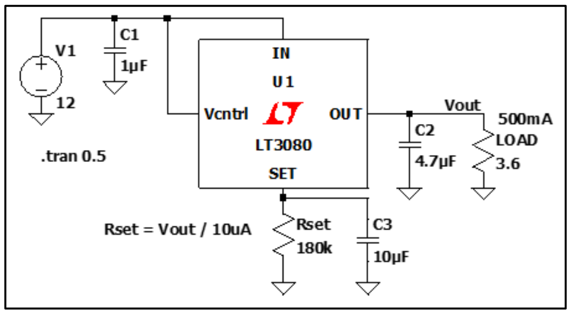


Figure 4: LT3080 SChematic in LTSPice, 1.8V

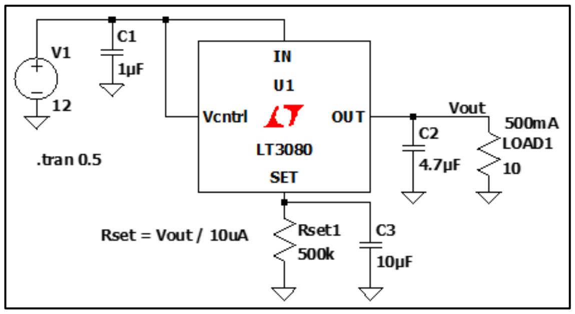


Figure 3: LT3080 Schematic in LTSpice, 5V

There were a few considerations made during the design process. The overall cost of each data logger should be no more than \$50. The LTC3636 requires two inductors which can be expensive depending on specifications. For the schematic in Figure 2, for example, the Bourns SRN8040TA-R50Y was chosen for L1 due to its low cost of \$0.242 at 2000 units, while still meeting design specifications. The inductor does have a 30 percent tolerance though, so if that needs to be smaller, then price will go up. It is always important to keep pricing in mind when selecting components for the final board, as many parts add up quickly.

An area of concern during design was with the LDO supplies. The LT3080 requires a minimum of 500µA on the output to function properly. If the components on the 1.8V and 5V lines use less current for periods of time, the voltage output may drop which could cause unintended operation. After discussing options with Club Car engineers, it was decided that a resistor could be placed on the output lines to force the minimum current requirement. This insures that at no point will the LDO stop operating correctly.

With the schematics verified and parts selected (see "Bill of Materials" in the appendix), the power supply designs were constructed in Altium Designer. Two separate schematic pages were created: one for the 3.3V and 3.8V dual-channel supply, and one for the two LDO supplies for 1.8V and 5V. In the appendices under the section "Additional Figures and Supporting Material," Figure 19 shows the completed LTC3636 schematic, and Figure 20 shows the LT3080 designs. Footprints for various components were downloaded from the manufacturer when available, and standard footprints for surface mounted device (SMD) resistors and capacitors were created. Some footprints can be seen in Figure 5

(Murata inductor, LTC3636, LT3080, 1206 resistor) captured from Altium. The printed circuit board design is discussed in the section “Printed Circuit Board.”

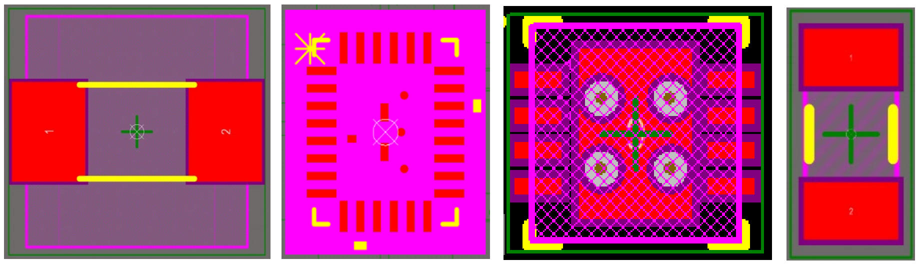
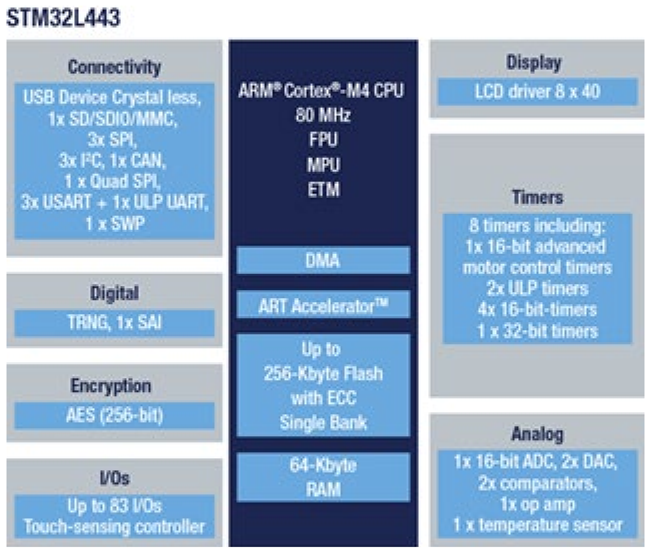


Figure 5: Various Footprints from Components from the Power Supply Section

Microcontroller

The microcontroller and CAN transceiver had to meet several requirements: specifications, price, interfaces, and functionality. For the software team, proper specification of the microcontroller was required to run the final embedded program. If the CPU was too slow or had only one clock, it might not be able to run the required software or run slowly. To avoid this situation, a minimum specification was required: over 10MHz CPU, 32 bit, and at least 2 clocks. To support some software such as LED, reset, and communication, extra pins were required. For the hardware interfaces, an SD card and CAN transceiver were required for this project to save and read data. To satisfy the budget, price was also important. After this consideration, the ST Micro STM32L433RCI6 was selected for the microcontroller and the Microchip MCP2542 for the CAN transceiver.


STM32L443
Connectivity
USB Device Crystal less, 1x SD/SDIO/MMC, 3x SPI, 3x I²C, 1x CAN, 1 x Quad SPI, 3x USART + 1x ULP UART, 1 x SWP
Digital
TRNG, 1x SAI
Encryption
AES (256-bit)
I/Os
Up to 83 I/Os Touch-sensing controller
ARM® Cortex®-M4 CPU 80 MHz FPU MPU ETM
DMA
ART Accelerator™
Up to 256-Kbyte Flash with ECC Single Bank
64-Kbyte RAM
Display
LCD driver 8 x 40
Timers
8 timers including: 1x 16-bit advanced motor control timers 2x ULP timers 4x 16-bit-timers 1 x 32-bit timers
Analog
1x 16-bit ADC, 2x DAC, 2x comparators, 1x op amp 1 x temperature sensor
Figure 6: STM32L433RCI6 Block Diagram

This microcontroller had enough speed (80MHz), more than two clocks, SPI, CAN, and USART. Serial Peripheral Interface (SPI) allows us to use an SD card. Universal Synchronous Asynchronous Receiver Transmitter (USART) allows us to communicate with the wireless modules.

Also, this microcontroller had a built-in CAN controller. The price was $3.27 which satisfied budget targets. To connect power and other parts, a schematic was made. Altium Designer was used as the schematic and PCB tool. This program offered both a schematic design and PCB design and it allowed converting schematic to PCB. Since the voltage level for data transmission on the microcontroller and wireless modules were different, a level shifter was also used to shift the voltage from 3.3V to 1.8V. After component selection and schematic creation were completed, PCB design was started. During the PCB design, it was found that the STM32L433RCI6 had less than 1mm pads since it was a Ball Grid Array (BGA) package. Because of that, Altium did not allow routing the STM32L433RCI6. Therefore, the microcontroller was changed to the STM32L433RCT6. The RCI6 is UFBGA packaging, but the RCT6 option used LQFP-64, a typical pin setup. Physical pin placements were different but the function was identical, it was cheaper, it had same number of pins, and it allowed routing in the PCB editor software.

After meeting with the software team, several pins were decided to run what functions. USART2 for LTE(PA0-PA6), CANRX and CANTX for CAN transceiver (PB8-9), and SDMMC1 (PC9-12, PD2,PB7) for SD card. To remove and reduce noise, a decoupling capacitor was used for Vin. Decoupling capacitors are usually used to filter undesired noise from power supplies. Since the CPU clock was 80 MHz, 1k impedance, 10uF capacitor, and 0.1uF capacitors were used. 0.1uF capacitors were ceramic to reduce budget but 10uF was non-ceramic and tantalum. The minimum value for the tantalum capacitor was 4.7uF. But to maximize the reducing noise effect, 10uF was selected. Also, decoupling capacitor was needed to be placed closer to IC for maximizing the reducing noise effect.

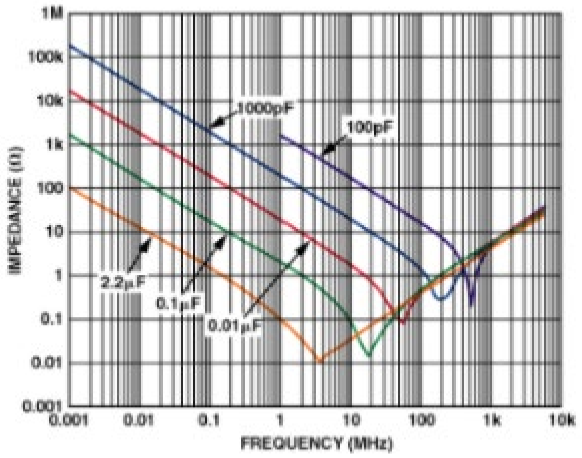


Figure 7: Self-Resonance in Capacitors

A switch was added to the NRST line to enable reset. Then a level shifter, SD card, and CAN transceiver were connected to pins where software team decided to use. After the schematic design, the PCB design was started. Unlike schematic design, PCB design requires more detail about components such as height and length for creating footprints. Figure 8 shows Altium's footprint creator tool. The final schematic for the microcontroller can be seen in the appendices in Figure 21.

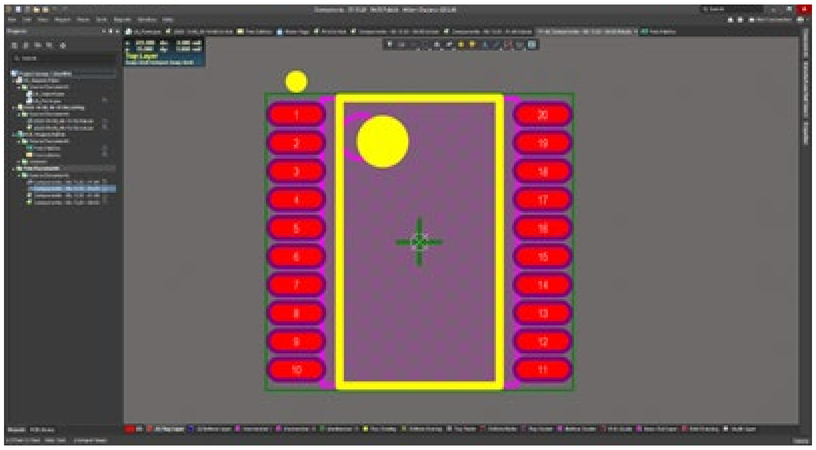


Figure 8: Altium Footprint Creator Tool

Interfaces – CAN, Level Shifting, SDIO

The existence of a CAN system was considered during the microcontroller selection. Controller Network Area (CAN) was at the core of this project. A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer. In a car, a CAN bus system is used to controller air conditioning, Anti-Thief systems, etc. In this project, the CAN bus is used to read vehicle information, then save it to an SD card.

Two types of CAN buses exist: CANFD and CAN2.0. CANFD is the latest version of CAN bus. However, Project CANDL did not require the higher spec CAN bus. To save budget, CANFD was rejected from the candidates.

Even though a CAN controller was built-in to the chosen microcontroller, a CAN transceiver was not built in. A CAN transceiver is a circuit that can communicate with the phyisical CAN bus and extract data from it. The CAN transceiver itself cannot communicate the data, so it must be connected to the controller on the microcontroller.

For Project CANDL, the MCP2542FD CAN transceiver was chosen. The pinout can be seen in Figure x in the appendices. It is product from the company MICROCHIP. It has only one simple function which was receiving data, but it was cheap ($0.60). More expensive models had built in level shifters but it was over $10. For the design, both ends of the pair of signal wires (CAN_H and CAN_L) must be terminated. This is because communication flows both ways on the CAN bus. The termination resistor should be matched with the nominal impedance. Therefore, a 120Ω resistor was used for terminating.

The TXS0108EPWR from Texas Instruments was chosen for the level shifters. It had supply voltage 1.2V - 5.5V. And this level shifter allowed us shifting the voltages needed for data transmission from 3.3V to 1.8V. The cellular and Wi-Fi modules required 1.8V, while the microcontroller operated at 3.3V. The schematic for the CAN transceiver and level shifters can be seen in Figure 22 in the appendices under the "Additional Figures and Supporting Materials" section.

Wireless Modules – LTE and Wi-Fi

Desmond Stevens

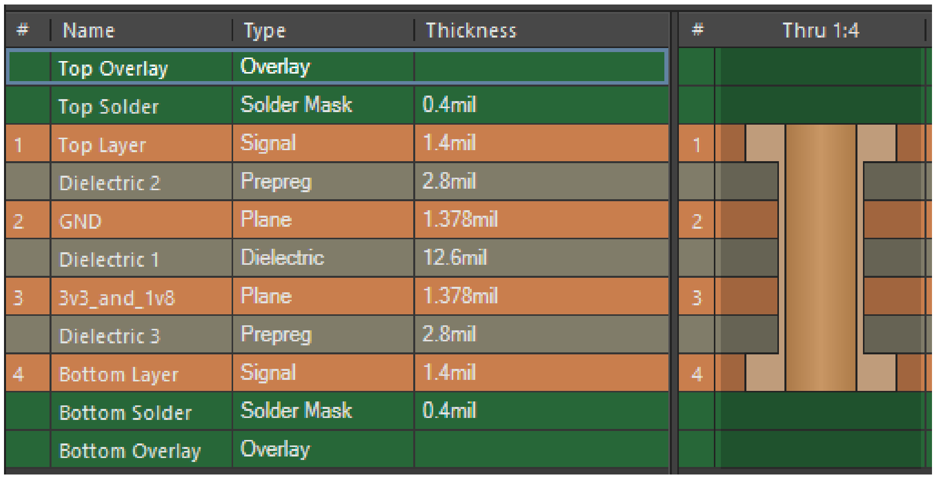
For the project, the cellular module gives assistance to the server by being able to transmit data wirelessly. Initially, cellular data will be the secondary method of communication to Wi-Fi. When Wi-Fi is not present, cellular data will then become the prioritized method of communication. Originally for this project the Quectel EG21-G LTE was chosen. The Quectel EG21-G LTE offers LTE Cat 1, maximum data rates up to 10 Mbps downlink and 5 Mbps uplink and offers an embedded GNSS module. The GNSS module is used to offer GPS locating. After ordering dev kits with the EG21-G chip, news was received from Quectel that the LTE chip that had been chosen for this project was not compatible with the Wi-Fi chip that had been selected. This news came after previously reaching out to Quectel in the Spring semester asking about the chips and their compatibility. In the Spring semester, it was relayed that the two chips were compatible. After receiving this unfortunate news so late into the game, the process of finding a new chip had to be done expeditiously. The Quectel EC21-V was then chosen due to its pin-to-pin compatibility to the EG21-G. This chip was also compatible to the Wi-Fi chip that was already selected. The only major difference in this chip was that it did not offer a global variant.

For this project, the Wi-Fi module will be used as the preferred means of communication. Its purpose will also be to transmit data to a server wirelessly. The Quectel FC20 Wi-Fi module was chosen for this project. The FC20 Wi-Fi module offers dual-band Wi-Fi, BT v4.2, and is compatible with the EC21-V LTE module. The FC20 provides a low-power SDIO 3.0 interface for WLAN and UART&PCM interfaces for BT function, and supports LTE/WLAN coexistence. The FC20 series hardware can withstand operation temperatures ranging from -35°C to +75°C and storage temperatures ranging from -40°C to +90°C. The schematics for both the cellular and Wi-Fi modules can be found in the appendices, in Figures 23 and 24, respectively.

Printed Circuit Board

Ethan Mitchell, Kitae Woo, Desmond Stevens

Two printed circuit board (PCB) designs were considered for Project CANDL. The first is the complete board. This is the production board, containing all components: power supplies, microcontroller, CAN transceiver, level shifters, LTE module, and Wi-Fi module. The full board can be seen in the appendices under “Additional Figures and Supporting Material” in Figure 25. For the PCB, it was important to keep the total layer number low to reduce manufacturing costs. Four layers were defined for the complete board as follows, from top to bottom: Signal 1, Ground, Split Power, Signal 2. The top layer, Signal 1, is for most of the routing between components. The 3.8V and 5V power supply lines are also located on this layer. Since these lines are supplies and not also in the domains for data transmission, they could be left near other signals. The next layer was ground, which is useful as a plane for electromagnetic interference reasons as well as convenience for using vias to connect to the plane. This avoids having to run a trace across the entire board to make a connection. For the third layer, a power plane was created and then split between two domains: 3.3V and 1.8V. This was chosen since those two domains are used for data transmission between the microcontroller and cellular and Wi-Fi chips. Having a supply line at 3.3V sitting next to a data line waiting for a 3.3V data signal is not good practice, as the supply could interfere with the signal. The final layer is the bottom of the board. This was defined as another signal layer, to make routing easier. If there was a trace that ran into another trace on the top layer, a via could be made that connected to the bottom of the board where the trace could then be routed on that layer instead. Figure 9 shows the defined layers for Project CANDL’s complete board.



#	Name	Type	Thickness	#
	Top Overlay	Overlay		
	Top Solder	Solder Mask	0.4mil	
1	Top Layer	Signal	1.4mil	1
	Dielectric 2	Prepreg	2.8mil	
2	GND	Plane	1.378mil	2
	Dielectric 1	Dielectric	12.6mil	
3	3v3_and_1v8	Plane	1.378mil	3
	Dielectric 3	Prepreg	2.8mil	
4	Bottom Layer	Signal	1.4mil	4
	Bottom Solder	Solder Mask	0.4mil	
	Bottom Overlay	Overlay		

Figure 9: Layer Stackup for the Complete Board Design

The other board made for Project CANDL is the test board. This contains core functionality, while removing the wireless modules. This was suggested by the Club Car team to be able to test the main components of the design, while not having to worry about RF issues. The full test board can be seen in the appendices, Figure 26. Headers were added to the test board to allow for easy connection and testing once printed. These headers were connected to the necessary microcontroller pins for controlling the Quectel development kit. The test board can easily be connected to test points on the Quectel board for verification purposes, or with any development kit if software on the microcontroller is adjusted. As seen in Figure x, there is a header for communication lines at 3.3V default levels, as well as at shifted 1.8V levels for the Quectel board. Both were added so there was more flexibility for testing. A third header was added, one with the following signal: 3.3V, 3.8V, 1.8V, 5V, and ground. This allowed for simple power supply testing, or to use for powering external devices.

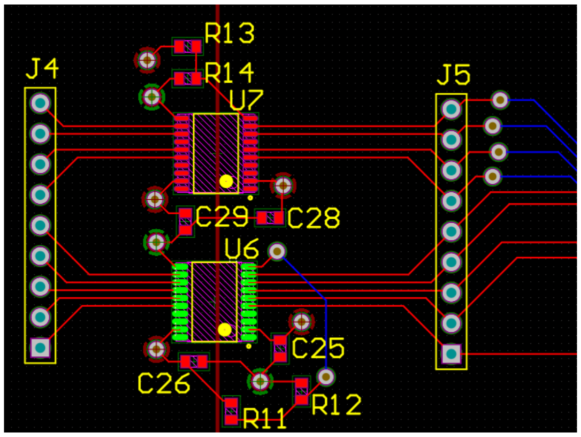


Figure 10: Test Board Headers

Performance

Ethan Mitchell, Kitae Woo, Desmond Stevens

The hardware design consisted of schematic and PCB design. The PCB was not ordered, as more time would be necessary to properly review the entire board. In addition, Gerber, drill, and other relevant board files would be required to send to a manufacturing house. Turnaround for a board would be a few days plus shipping, and by the time the two PCBs were completed, it was late in the semester. After consulting with Club Car, it was decided that it would be best to focus on making sure all design files are correct, rather than rushing to get an actual board made and shipped.

LTSpice was used in simulation for each power supply: the LTC3636 and the two LT3080 chips. These designs were verified before creating Altium schematics and eventually placing and routing the circuits on a PCB. Test pins were added to the PCB to ensure these power supplies could be tested when a board is ordered in the future.

For testing, an SD card adapter was not included. Therefore, a micro SD card connector was purchased separately and connected to SDMMC pin (PD2, PC7-12). USART is a type of a serial interface device that can be programmed to communicate asynchronously or synchronously. In this project, USART1 was used for the JTCK. UART is a type of a serial interface device that can be only communicate asynchronously. UART2 was used to operate the Quectel EC-21(Cellular Module) and UART3 was used to debug the EC-21. This microcontroller had its own CAN controller pins (PB8-9), CAN_RX and CAN_TX, which were connected to an CAN transceiver development board for testing.

Embedded Software Design

Objectives

Asa McVay, Mohammad Zuaiter, Jack Burrows

In designing the software for the microcontroller, the goal was to create a framework from which the engineers at Club Car could build a finished product, with as many features implemented as time allows during this semester. A proof of concept with CAN collection and communication between the microcontroller and the server was also set as a goal in order to demonstrate proper functionality at presentation time. The desired features were to:

1. Be able to collect particular kinds of CAN messages from the vehicle.
2. Store those messages on an SD card.
3. Upload the data to a server over WiFi or LTE.

The implementation of this involves using a real time operating system (RTOS) to control two threads that would 1) collect and store the CAN data and 2) control internet access and data upload (Figure 27 in the "Additional Figures and Supporting Material" section in the appendices). Interrupts for CAN and UART are also to be implemented in order to allow for immediate handling of incoming messages on those interfaces.

For internet connectivity, the connection is primarily handled by the Quectel EC21 LTE chip which also controls the WiFi connection. A mode for provisioning the WiFi is included in the design to allow the user to input credentials for their router to the unit and reduce traffic across LTE to save on data costs. Furthermore, MQTT is used as the messaging protocol by which the messages would be sent up to the server. An MQTT client is included on the LTE chip, so implementation of this feature involves sending AT commands to the LTE chip instead of creating a client on the microcontroller itself, speeding up development time significantly.

Design Process

RTOS

Asa McVay

In an embedded system, the RTOS is the real time operating system on top of which all of the other code is written. The primary uses of the RTOS are task and memory management to help simplify complex designs by breaking different jobs into tasks (described in this report previously as threads). These tasks also have their own memory usage that the OS initializes and deinitializes with the task.

For the RTOS implementation in this project, Amazon's FreeRTOS was chosen due to its superb documentation and support. This makes integration with other software components much simpler because of the abundance of examples and free software that support this system. For example, the Paho MQTT project has an implementation of MQTT written specifically for FreeRTOS, so should we have chosen to use that client instead of the one included on the LTE chip, design would have been much simpler than with other RTOSs.

The reason for using an RTOS in this project was specifically to allow for multi-threading. This allows segmentation of the project into multiple parts as well as simultaneous operation of different features. The hope was that this would help simplify our design process by letting us hand off a thread to one developer, and another thread to a second developer with minimal communication between the two, and therefore less time waiting for responses. The actual implementation of FreeRTOS was trivial due to its support by STMicroelectronics in their CubeMX software. This meant that implementation was just a matter of checking a box, adding the tasks that we wanted to the project, and allocating memory for each task.

MQTT Communication

Asa McVay

The MQTT protocol is a lightweight communication system widely used in IOT development. It uses a publish and subscribe method of communication where various "topics" are created, and any client that is subscribed to the topic will receive messages that are published to it by other clients. It was chosen due to its widespread use and the familiarity that some team members have with it.

The implementation of MQTT on the microcontroller consistes of creating topic names and then subscribing and publishing to those topics using AT commands on the LTE chip. The topic names are of the following format: "UID/topic". The UID is the unique ID of the microcontroller that is used in the topic name for the purpose of letting every controller have its own set of topics without there being a possibility of interfering with other controller's operation. The topics allowed for each device are CAN, GPS, command, and error. The message format currently being used applies only to the CAN topic, but is formatted as follows:

{“msg_id”: “<insert id from CAN message>”, “data”: “<insert CAN data>”}

Wireless Communications

Mohammad Zuaiter

Wireless communication is based on UART communication. This is implemented using an interrupt routine for receive and currently blocking code for transmitting. A basic set of functions were built to manage the transmit and receive of AT commands and responses with a basic set of commands as part of initialization.

The functions are currently used by the MQTT implementation for transmission. The interrupt routine has a set number of states that are set with a state for searching for message codes, one for text response, another for text response and message code, a mode for fixed receive buffer, and a mode that incorporates variable length response with a fixed size prefixed with a response text and length header. Figure 28 in the appendices shows the main UART2 interrupt routine states.

Data Collection – CAN and SDIO

Jack Burrows

Data collection code is based on interrupt-driven CAN bus reception. Every CAN message has a unique ID. A lower ID means a higher priority. In the case of this project, we accept all messages and parse them into ID and data. Filters can be set up in the future to request only messages with specific IDs. First, CAN must be initialized through a CAN_init function. This is where all CAN bus abilities, interrupts and filters can be set up. We then must overwrite the receive interrupt callback function using FIFO0. FIFO allows for messages to be stored in a “first in first out” queue to be received.

The main function then can call the data collection function. Once storeData() is called, data collection begins. We set up microSD storage for the future using FATFS. This allows the user to create a FAT file system on the logical drive and write the CAN bus data to a text file for exportation.

Performance

Asa McVay, Mohammad Zuaiter, Jack Burrows

The features completed in this project are the FreeRTOS and peripheral setup, the AT messages to send to the LTE chip over UART, UART send and receive operation, and the JSON message forming for CAN messages. Features that are nearly working are CAN receive and SD card reading and writing.

Troubleshooting has proved to be difficult and as a consequence the interrupt being used to drive CAN is currently non-operational. While there is code that is meant to implement it, further work must be done for it to be operational. It is currently unknown if the issue is due to wiring problems or if it is software related.

Back-End Software Design

Objectives

Chris Hill

The objectives of the back-end design are to receive the MQTT messages from the data logger, parse and decode the data, and store the data in a SQL database. Another goal was to use an analytical software, Tableau, to graphically represent the stored data.

To accomplish these objectives, a Linux virtual machine (VM) was created. On this VM, an MQTT broker server and a MySQL database server were hosted. The scripting language Python was used to receive the MQTT messages, decode the CAN data, and insert the data into SQL tables.

Design Process

Chris Hill

The first step to accomplishing the design goals was creating a Linux VM. This was accomplished using Oracle VirtualBox. A Linux Ubuntu virtual disk was downloaded and used as the disk for the VM.

Next, the servers needed to be created and configured. MySQL and Mosquitto MQTT broker were downloaded through the Linux terminal. The Mosquitto broker automatically runs in the background after downloading. To view connections to the broker, the broker service was run in debug mode through a Linux terminal window. MySQL was started and a new database called "candl_data" was created. Within this database, three tables were created. For this project, only a few tables were needed to show that the decoding and data insertion process were viable. In an actual, full use case, there will be a table for each CAN message ID. For security, a user was created on the SQL server with restricted access to the single, candl_data database.

After the servers were configured, several Python scripts were created to define necessary functions to receive CAN data, decode the data, and insert the data into the database. One script focused on the connection to the SQL database. Within this script, a class was created to manage the database connection. Creating a class makes the connection process simple as all that is needed to connecting is creating an instance of the class. Other functions were defined within this script to handle the formatting of SQL commands to insert data to and read data from the appropriate tables. The second script was used as the MQTT listener client. This script uses the Eclipse Paho MQTT Python module to create an MQTT client that can connect to the broker and subscribe to topics. Using this client, the data sent by the microcontroller can be received and decoded. Functions were written to properly decode the data sent in JSON format from the microcontroller. Using a CAN module for Python, the CAN data, message ID and sensor readings, was translated from hexadecimal bytes to scaled decimal values. The CAN module uses a CAN DBC file, provided by Club Car, to decode the values properly based on the message and signal definitions withing the DBC file. After decoding, the data is passed to an insert function defined in the SQL handling script, and the decoded data is inserted into the proper table. One other python script was used, but no functions were defined in it. Instead, constants, such as the MQTT broker IP address, SQL user information, and other server connection constants, were defined in this script. This file was imported into the other scripts. Doing this allows constant values to be changed in one place, rather than in several. This will allow for easier configuration when the scripts are given to Club Car.

Performance

Chris Hill

After writing the scripts, the functionality was tested by creating dummy CAN data and publishing the data to the MQTT broker. The listener, which was subscribed to all topics, received any MQTT message sent to the broker, and decoded the data.

Original testing consisted of generating data on the VM and publishing the data locally. This was done to test decoding and the SQL insert functions. These tests verified that the Python scripts properly decoded the published data and could format and insert the data into the proper SQL tables.

Later testing more closely resembled the final design. Another team member published data from the LTE module to the broker. Not only did the connection to the broker succeed, the data was properly received and decoded. The data was also inserted properly into the table, verifying the back-end software functioned as expected.

One objective that was not met due to lack of time and experience with the software was the graphical analysis of the data. The LTE module testing occurred later than planned due to delays in software completion and an unexpected change in LTE modules due to misinformation by Quectel. Because of the delays, there was not sufficient time to set up Tableau and graph the data stored in the database. Another limitation was the lack of access to Tableau. As Tableau requires a software license to use, Club Car had to send a company laptop that was intended to be used; however, because the database lived on a VM on a separate laptop, the connection to Tableau became more complicated and fell out of scope with the goals of the project. However, this lack of graphical analysis does not take away from the main goal of decoding and storing historical data.

Summary and Further Improvements

Hardware

Schematic

Additional components / connections, Modifications

The power supply designs were verified with LTSpice simulation, but this is a theoretical approximation of how the circuits will work. Hardware testing should be performed by using the test PCB, where multiple test points are available on header pins. Additionally, for the LTC3636, ripple currents may be too high, in which case different inductors will need to be selected. A slower switching speed could also be implemented, but this will increase the ripple current. For the LT3080 circuits, the maximum output current is 1.1A. From the initial power requirements gathering in Table 1, this is acceptable, but testing should be done on a board to verify. An alternate option to the LDOs is to use another LTC3636 chip for the 1.8V and 5V lines. Switch-mode power supplies (SMPS) are great for consistent power output, especially in cases with large current draws. The only components that would need to be changed are the inductors and feedback resistors to change the output levels.

Microcontroller's PCB design was not seriously considered at the beginning of the project. Therefore, the microcontroller had to be changed almost at the end of the project because route was too small. For the future improvements, combability with other PCB components/board should be considered. On the PCB board, 10uF capacitor should have been placed furthest from the power like schematic. Because it helps to smooth out any low-frequency changes in an input voltage. To maximize decoupling capacitor's effect, PCB placement also should be considered.

Future improvements for the wireless modules would consists of possibly enabling the Bluetooth features between the two chips. This can be done by connecting the Bluetooth and COEX lines between the LTE and Wi-Fi chips. Another future improvement would possibly be exploring other LTE and Wi-Fi chips. The chip selected for this project does not offer a global variant. There are other chips on the market that do offer this capability. (Stevens)

PCB

A design-for-test approach was taken with the Test Board, where signals of interest were connected to easy-to-access headers. This allows the test board to be made and correct operation verified. The test board does not have the wireless modules on it, so it can be connected to the Quectel development kit for testing. However, the connections on the headers could be connected to any development kit, meaning any supplier could be evaluated with the core design of Project CANDL.

Some areas that are unknown with the complete board design involve radio frequency (RF) problems. The current routings of the antenna connections cross a large portion of the board. This poses greater risk for interference from other signals on the PCB. After talking with engineers at Club Car, these risks can be mitigated by using ground pours on the entire top layer, in addition to running ground traces next to the antenna lines. This will guide most noise to ground. Another way to combat noise on the board is to reduce the noise from the sources. The biggest producers of noise on a PCB are switching signals. This includes UART and serial communication lines. Another switching noise source is power supplies. On the current board designs, the LTC3636 is a switching supply, which is inherently noisy. This can be designed against by installing shielding on the final board to prevent electrical noise from escaping the power supply regions on the PCB. More layers can also be added, to ensure RF signals have their own routing plane, rather than being next to other signals on the top or bottom layers.

Software

Embedded

For the embedded side of the project, many features originally planned to be included in the project at the beginning were made into future improvements since the scope of the project was so large. Each section has their own list of improvements that must be included before it would be viable for production. These are roughly the same as how the sections were broken up for design.

RTOS and Firmware

Asa McVay

In the underlying software of the system, some desired features were dropped fairy early on in development due to the complexity perceived by Club Car. In particular, the ability to update the firmware over the cloud was made as an item to be added later since creating a bootloader is a complex, time consuming task.

MQTT and Messaging

Asa McVay

There are several features included in the original design having to do with messaging to the server that have not been completed and are thus being made an item for future improvement. These features are command receiving and execution, error reporting on the “error” topic, and GPS data uploads.

Commands were a desired feature so that aspects of the devices could be changed after deployment. This would include commands like firmware updates and upload frequency. Error reporting would be helpful to allow for diagnostic reporting for customers or field tests. GPS was a feature desired by Club Car as part of the data they wanted to store on their servers, but while the hardware is in place to use it, the software has yet to be implemented due to time constraints.

Wireless Communication

Mohammed Zuaiter

Future improvements include the extension of the receive interrupt routine and library to accommodate all relevant AT command responses and the development of a function library with relevant AT commands fixed into them. This will be extended with a TCP/TLS library for TCP communications across Wi-Fi and LTE. Additional matinence of the code will include switching to use enumeration for the interrupt routine as opposed to using char values for the various states and modes.

Additional improvements include the implementation of Wi-Fi code with provisioning mode, and if the FC-20 is used, Bluetooth support for potential wireless sensors.

Implementations of GPS code is an expected area of improvement with decoding of NMEA code on the debug UART and implementation on this interrupt routine to save GPS data for logging and update of the real-time clock from the very accurate and internet independent always available GPS clock.

Additional implementation includes startup control and shutdown of the LTE module through power pins located on the LTE chip to the microcontroller. This also includes status checks and self-diagnostics of internet connectivity.

CAN and SDIO

Jack Burrows

Future improvement for data collection that should be considered is the use of filters to obtain specific messages based on the desire of the user. This can allow the user to filter out unnecessary messages from the CAN bus. If desired, Club Car can set the CAN bus to follow a loopback reception. This will constantly record messages from the CAN bus instead of waiting for an interrupt.

Back-End

Chris Hill

The main functionality of the back-end was completed, but a few additional improvements could be made.

The first improvement is expanding the amount of tables in the SQL database. Since the goal was to properly decode data and show that this data could be stored, only a few tables were created based on consultation with employees at Club Car. The scripts are easily able to handle any new tables. The majority of overhead would be placed on the creation of the tables with the proper naming of tables and columns. Since Club Car will have to create and configure a SQL server within their server infrastructure, the tables were limited to focus work on the data decoding.

Another improvement would be a custom TCP connection module, rather than using MQTT. This improvement was discussed with Club Car, who agreed such a design was not in the project scope, but will most likely occur once documentation is handed over to Club Car. Such a change would require a complete change in the receiving Pyhton script and data decoding, but would allow for lower data costs. Because an LTE connection is used to send data, cost can become a concern when the project is scaled to several hundred vehicles. The current implementation focused on functionality over data costs, and so MQTT was used because of its simplicity.

The final improvement would be the implementation of graphical data analysis, such as Tableau. This was originally planned but fell out of scope as the project deadline neared.

References

1. LTC3636 Datasheet, Analog Devices, https://www.analog.com/media/en/technical-documentation/data-sheets/LTC3636-3636-1.pdf
2. LT3080 Datasheet, Analog Devices, https://www.analog.com/media/en/technical-documentation/data-sheets/3080fc.pdf
3. STM32L433RCxx Datasheet and Refernece Manual, ST Micro, https://www.st.com/en/microcontrollers-microprocessors/stm32l433rc.html#documentation
4. MCP2542FD Datasheet, Microchip, https://www.mouser.com/datasheet/2/268/MCP2542FD_MCP2542WFD_4WFD_Data_Sheet_DS20005514C-1890692.pdf
5. TXS0108EPWR Datasheet, Texas Instruments, https://www.ti.com/lit/ds/symlink/txs0108e.pdf?ts=1606073304606&ref_url=https%253A%252F%252Fwww.google.com%252F
6. EG21-G, Quectel, https://www.quectel.com/product/eg21g.htm
7. EC21, Quectel, https://www.quectel.com/product/ec21.htm
8. FC20, Quectel, https://www.quectel.com/product/fc20.htm
9. STM32 HAL Drivers User Manual, https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf
10. FreeRTOS Documentation, https://www.freertos.org/Documentation/RTOS_book.html

Appendices

Team Members and Design Sections

Ethan Mitchell

Team Lead

System Architecture, Power Supplies, PCB

Jack Burrows

Embedded Software – Data Collection

Chris Hill

Back-End Software – Processing and Analytics

Asa McVay

Embedded Software – RTOS, MQTT

Desmond Stevens

Wireless Communications Modules

Kitae Woo

Microcontroller, CAN Transceiver

Mohammad Zuaiter

Embedded Software – Wireless Configuration

Project Management Information

Ethan Mitchell

The management approach has focused on collaborative design. Multiple tools were selected to increase productivity and support the project team. Slack was chosen as the main messaging service for the Project CANDL team. It is widely used in industry, allowing the creation of channels within a team workspace. For example, there are dedicated channels for discussion about meetings, embedded software, and hardware. In addition, Slack has support for the integration of third-party apps, like Microsoft Office and Zoom. For file hosting, a SharePoint intranet site was set up for team members. A GitHub was also created as a repository for any code written this semester.

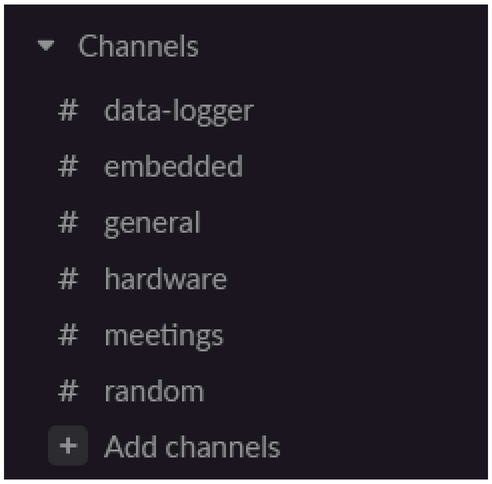
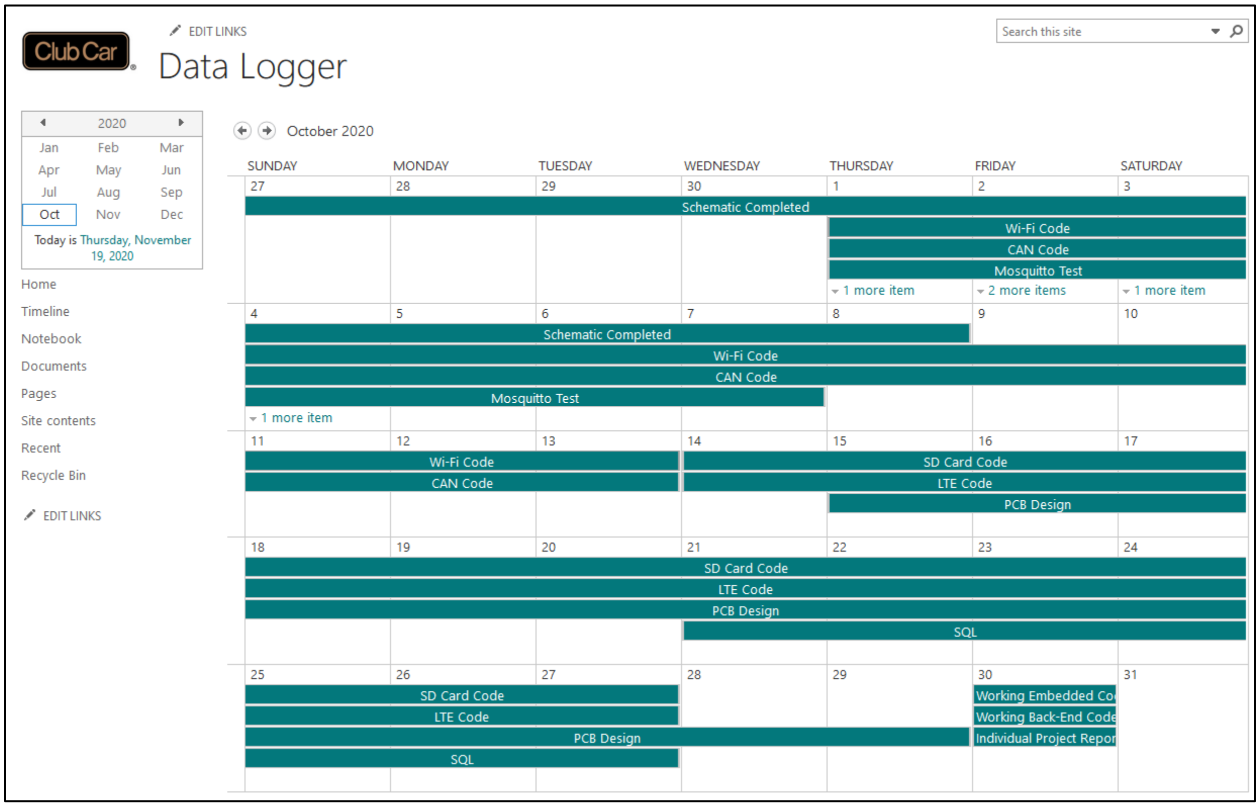


Figure 11: Slack Channels for Project CANDL

Team meetings are held weekly on Wednesdays at 2:00 and serve as a check-in with the team regarding tasks completed in the past week and goals for the next week. These meetings are conducted over Zoom. Each week, there is also a meeting with Club Car engineers to discuss progress. These occur on Thursdays at 5:00 to ensure class and work conflicts are avoided. The meetings are through Teams and allow the Project CANDL team to share design progress and ask questions with the team at Club Car. There has been some great feedback throughout the semester and having weekly communication has been a great asset.

To track progress throughout the semester, a timeline in SharePoint was created. The timeline feature allows tasks and goals to be created with start and end dates and has the option of tracking percentage complete on any given task. These tasks and goals can be assigned to the team or to individuals, keeping the design process moving. Figure x displays an example list of tasks from SharePoint.

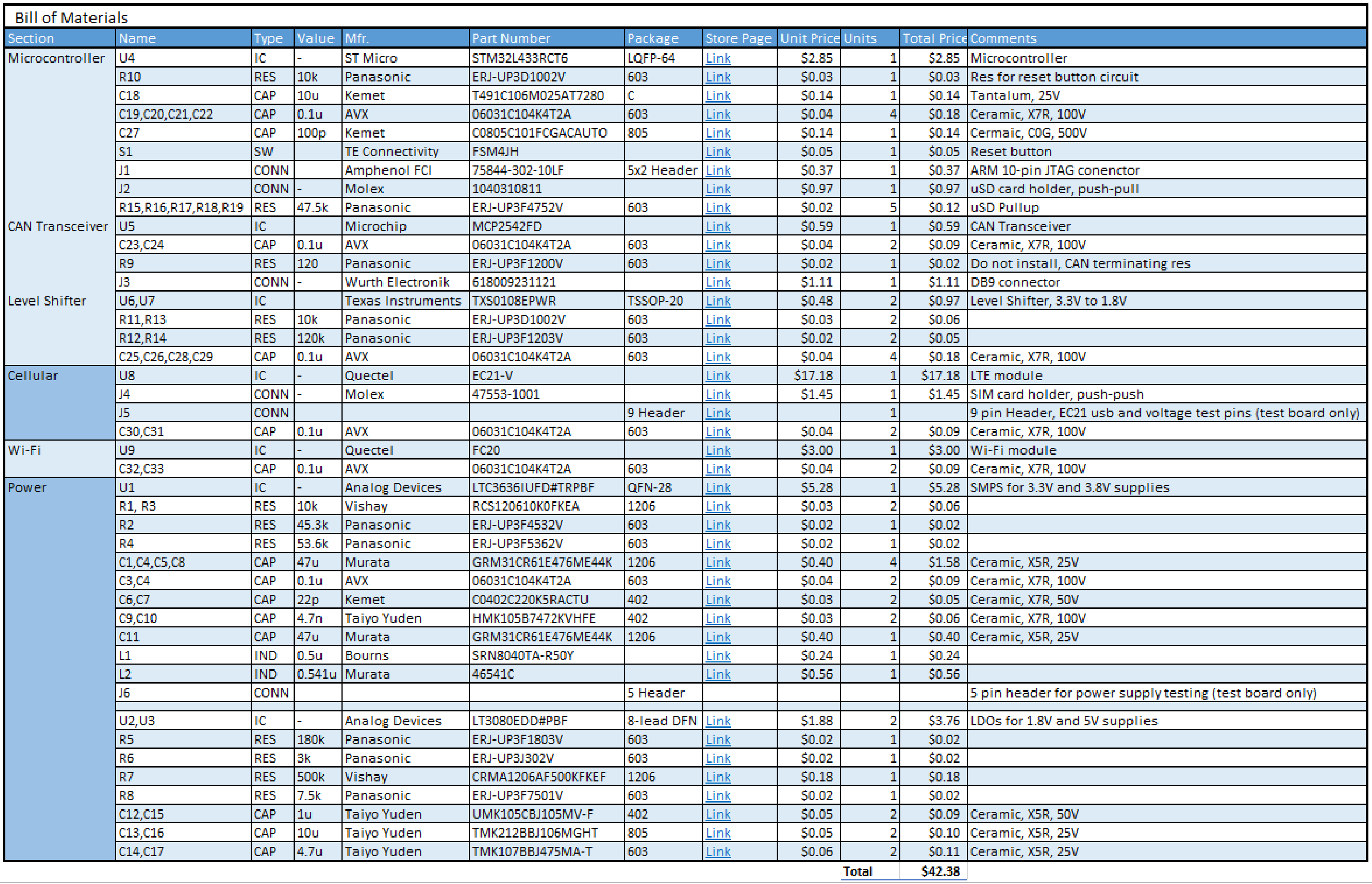


SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
27	28	29	30	1	2	3
Schematic Completed						
				Wi-Fi Code		
				CAN Code		
				Mosquitto Test		
				1 more item	2 more items	1 more item
4	5	6	7	8	9	10
Schematic Completed						
Wi-Fi Code						
CAN Code						
Mosquitto Test						
1 more item						
11	12	13	14	15	16	17
Wi-Fi Code			SD Card Code			
CAN Code			LTE Code			
				PCB Design		
18	19	20	21	22	23	24
SD Card Code						
LTE Code						
PCB Design						
			SQL			
25	26	27	28	29	30	31
SD Card Code					Working Embedded Co	
LTE Code					Working Back-End Code	
PCB Design					Individual Project Repor	
SQL						

Figure 12: SharePoint Calendar Timeline, October Tasks

Bill of Materials

A Bill of Materials (BOM) file was created to track individual component prices for the final board design. This was important, as a target price of $50 was a requirement. In addition to prices, the BOM contains valuable information such as part number, manufacturer, schematic reference name, and link to the Mouser store page for the component. When all part costs were added, the total cost was $42.38, which was well under the $50 requirement, while still meeting all design specifications. Figure x on the following page shows the BOM.



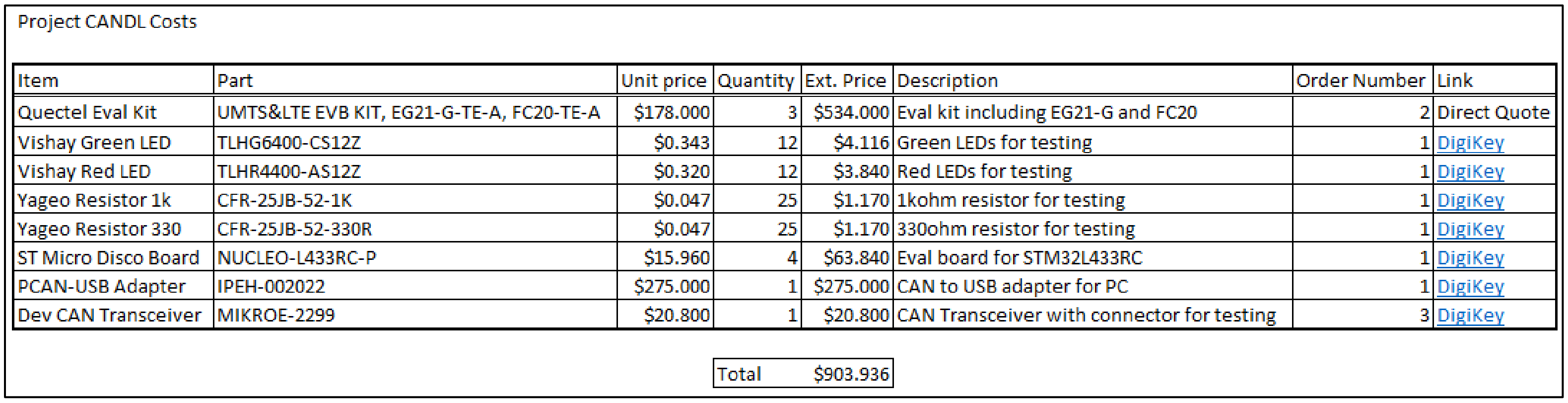
Bill of Materials

Section	Name	Type	Value	Mfr.	Part Number	Package	Store Page	Unit Price	Units	Total Price	Comments
Microcontroller	U4	IC	-	ST Micro	STM32L433RCT6	LQFP-64	Link	$2.85	1	$2.85	Microcontroller
	R10	RES	10k	Panasonic	ERJ-UP3D1002V	603	Link	$0.03	1	$0.03	Res for reset button circuit
	C18	CAP	10u	Kemet	T491C106M025AT7280	C	Link	$0.14	1	$0.14	Tantalum, 25V
	C19,C20,C21,C22	CAP	0.1u	AVX	06031C104K4T2A	603	Link	$0.04	4	$0.18	Ceramic, X7R, 100V
	C27	CAP	100p	Kemet	C0805C101FCGACAUTO	805	Link	$0.14	1	$0.14	Cermaic, C0G, 500V
	S1	SW		TE Connectivity	FSM4JH		Link	$0.05	1	$0.05	Reset button
	J1	CONN		Amphenol FCI	75844-302-10LF	5x2 Header	Link	$0.37	1	$0.37	ARM 10-pin JTAG conenctor
	J2	CONN	-	Molex	1040310811		Link	$0.97	1	$0.97	uSD card holder, push-pull
	R15,R16,R17,R18,R19	RES	47.5k	Panasonic	ERJ-UP3F4752V	603	Link	$0.02	5	$0.12	uSD Pullup
CAN Transceiver	U5	IC		Microchip	MCP2542FD		Link	$0.59	1	$0.59	CAN Transceiver
	C23,C24	CAP	0.1u	AVX	06031C104K4T2A	603	Link	$0.04	2	$0.09	Ceramic, X7R, 100V
	R9	RES	120	Panasonic	ERJ-UP3F1200V	603	Link	$0.02	1	$0.02	Do not install, CAN terminating res
	J3	CONN	-	Wurth Electronik	618009231121		Link	$1.11	1	$1.11	DB9 connector
Level Shifter	U6,U7	IC		Texas Instruments	TXS0108EPWR	TSSOP-20	Link	$0.48	2	$0.97	Level Shifter, 3.3V to 1.8V
	R11,R13	RES	10k	Panasonic	ERJ-UP3D1002V	603	Link	$0.03	2	$0.06	
	R12,R14	RES	120k	Panasonic	ERJ-UP3F1203V	603	Link	$0.02	2	$0.05	
	C25,C26,C28,C29	CAP	0.1u	AVX	06031C104K4T2A	603	Link	$0.04	4	$0.18	Ceramic, X7R, 100V
Cellular	U8	IC	-	Quectel	EC21-V		Link	$17.18	1	$17.18	LTE module
	J4	CONN	-	Molex	47553-1001		Link	$1.45	1	$1.45	SIM card holder, push-push
	J5	CONN				9 Header	Link		1		9 pin Header, EC21 usb and voltage test pins (test board only)
	C30,C31	CAP	0.1u	AVX	06031C104K4T2A	603	Link	$0.04	2	$0.09	Ceramic, X7R, 100V
Wi-Fi	U9	IC	-	Quectel	FC20		Link	$3.00	1	$3.00	Wi-Fi module
	C32,C33	CAP	0.1u	AVX	06031C104K4T2A	603	Link	$0.04	2	$0.09	Ceramic, X7R, 100V
Power	U1	IC	-	Analog Devices	LTC3636IUFD#TRPBF	QFN-28	Link	$5.28	1	$5.28	SMPS for 3.3V and 3.8V supplies
	R1, R3	RES	10k	Vishay	RCS120610K0FKEA	1206	Link	$0.03	2	$0.06	
	R2	RES	45.3k	Panasonic	ERJ-UP3F4532V	603	Link	$0.02	1	$0.02	
	R4	RES	53.6k	Panasonic	ERJ-UP3F5362V	603	Link	$0.02	1	$0.02	
	C1,C4,C5,C8	CAP	47u	Murata	GRM31CR61E476ME44K	1206	Link	$0.40	4	$1.58	Ceramic, X5R, 25V
	C3,C4	CAP	0.1u	AVX	06031C104K4T2A	603	Link	$0.04	2	$0.09	Ceramic, X7R, 100V
	C6,C7	CAP	22p	Kemet	C0402C220K5RACTU	402	Link	$0.03	2	$0.05	Ceramic, X7R, 50V
	C9,C10	CAP	4.7n	Taiyo Yuden	HMK105B7472KVHFE	402	Link	$0.03	2	$0.06	Ceramic, X7R, 100V
	C11	CAP	47u	Murata	GRM31CR61E476ME44K	1206	Link	$0.40	1	$0.40	Ceramic, X5R, 25V
	L1	IND	0.5u	Bourns	SRN8040TA-R50Y		Link	$0.24	1	$0.24	
	L2	IND	0.541u	Murata	46541C		Link	$0.56	1	$0.56	
	J6	CONN				5 Header					5 pin header for power supply testing (test board only)
	U2,U3	IC	-	Analog Devices	LT3080EDD#PBF	8-lead DFN	Link	$1.88	2	$3.76	LDOs for 1.8V and 5V supplies
	R5	RES	180k	Panasonic	ERJ-UP3F1803V	603	Link	$0.02	1	$0.02	
	R6	RES	3k	Panasonic	ERJ-UP3J302V	603	Link	$0.02	1	$0.02	
	R7	RES	500k	Vishay	CRMA1206AF500KFKEF	1206	Link	$0.18	1	$0.18	
	R8	RES	7.5k	Panasonic	ERJ-UP3F7501V	603	Link	$0.02	1	$0.02	
	C12,C15	CAP	1u	Taiyo Yuden	UMK105CBJ105MV-F	402	Link	$0.05	2	$0.09	Ceramic, X5R, 50V
	C13,C16	CAP	10u	Taiyo Yuden	TMK212BBJ106MGHT	805	Link	$0.05	2	$0.10	Ceramic, X5R, 25V
	C14,C17	CAP	4.7u	Taiyo Yuden	TMK107BBJ475MA-T	603	Link	$0.06	2	$0.11	Ceramic, X5R, 25V
									Total	**$42.38**	

Figure 13:Bill of Materials

Order Tracking File

A file was created to track all orders placed during the semester. This file kept a running list of purchases and their A file was created to track all orders placed during the semester. This file kept a running list of purchases and their costs and subtracted from the overall budget of $1500.00. The total development cost this semester was $903.94, leaving $596.06 left in the budget. Figure x shows orders placed.



Project CANDL Costs

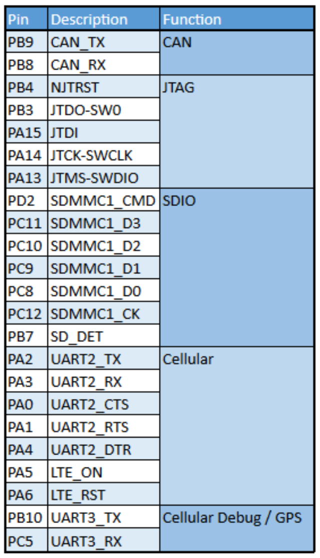
Item	Part	Unit price	Quantity	Ext. Price	Description	Order Number	Link
Quectel Eval Kit	UMTS&LTE EVB KIT, EG21-G-TE-A, FC20-TE-A	$178.000	3	$534.000	Eval kit including EG21-G and FC20	2	Direct Quote
Vishay Green LED	TLHG6400-CS12Z	$0.343	12	$4.116	Green LEDs for testing	1	DigiKey
Vishay Red LED	TLHR4400-AS12Z	$0.320	12	$3.840	Red LEDs for testing	1	DigiKey
Yageo Resistor 1k	CFR-25JB-52-1K	$0.047	25	$1.170	1kohm resistor for testing	1	DigiKey
Yageo Resistor 330	CFR-25JB-52-330R	$0.047	25	$1.170	330ohm resistor for testing	1	DigiKey
ST Micro Disco Board	NUCLEO-L433RC-P	$15.960	4	$63.840	Eval board for STM32L433RC	1	DigiKey
PCAN-USB Adapter	IPEH-002022	$275.000	1	$275.000	CAN to USB adapter for PC	1	DigiKey
Dev CAN Transceiver	MIKROE-2299	$20.800	1	$20.800	CAN Transceiver with connector for testing	3	DigiKey

Total	$903.936

Figure 14: Order Tracking File, Development Costs

Master Pinout

The Master Pinout contains the pinout for every component used in Project CANDL. The Excel The Master Pinout contains the pinout for every component used in Project CANDL. The Excel file contains a separate worksheet for each component and is a reference tool for quickly looking up pin descriptions and connections. This file contains the BOM as a worksheet, as well as the power requirements. The Master Pinout gathers all relevant hardware information into one place. The pin assignments sheet is shown in Figure x, and an example pinout is shown in Figure x.



Pin	Description	Function
PB9	CAN_TX	CAN
PB8	CAN_RX	
PB4	NJTRST	JTAG
PB3	JTDO-SWO	
PA15	JTDI	
PA14	JTCK-SWCLK	
PA13	JTMS-SWDIO	
PD2	SDMMC1_CMD	SDIO
PC11	SDMMC1_D3	
PC10	SDMMC1_D2	
PC9	SDMMC1_D1	
PC8	SDMMC1_D0	
PC12	SDMMC1_CK	
PB7	SD_DET	
PA2	UART2_TX	Cellular
PA3	UART2_RX	
PA0	UART2_CTS	
PA1	UART2_RTS	
PA4	UART2_DTR	
PA5	LTE_ON	
PA6	LTE_RST	
PB10	UART3_TX	Cellular Debug / GPS
PC5	UART3_RX	

PA0, PA1, PA4 not used in current code revision

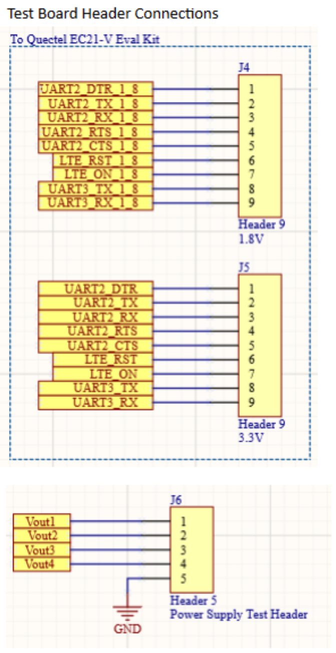
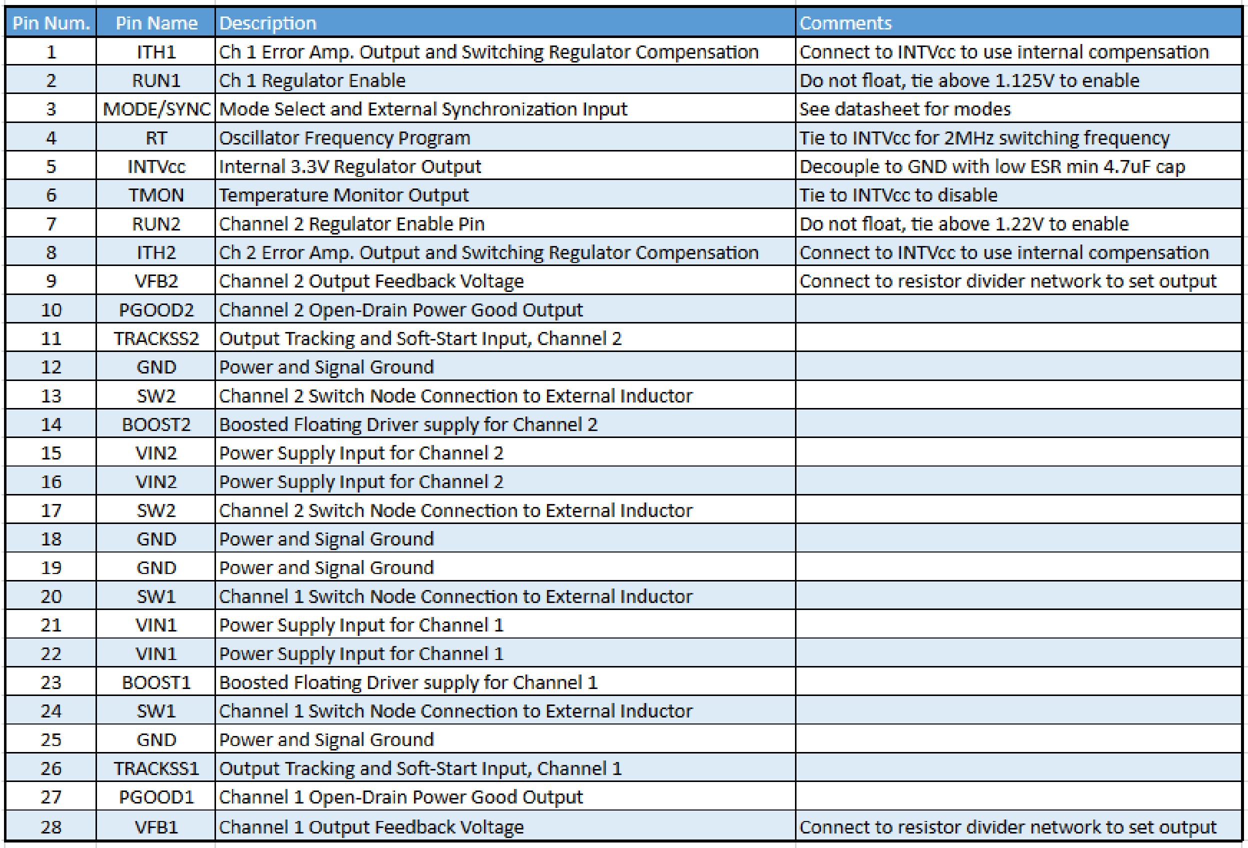


Figure 15: Pin Assignments in the Master Pinout File



Pin Num.	Pin Name	Description	Comments
1	ITH1	Ch 1 Error Amp. Output and Switching Regulator Compensation	Connect to INTVcc to use internal compensation
2	RUN1	Ch 1 Regulator Enable	Do not float, tie above 1.125V to enable
3	MODE/SYNC	Mode Select and External Synchronization Input	See datasheet for modes
4	RT	Oscillator Frequency Program	Tie to INTVcc for 2MHz switching frequency
5	INTVcc	Internal 3.3V Regulator Output	Decouple to GND with low ESR min 4.7uF cap
6	TMON	Temperature Monitor Output	Tie to INTVcc to disable
7	RUN2	Channel 2 Regulator Enable Pin	Do not float, tie above 1.22V to enable
8	ITH2	Ch 2 Error Amp. Output and Switching Regulator Compensation	Connect to INTVcc to use internal compensation
9	VFB2	Channel 2 Output Feedback Voltage	Connect to resistor divider network to set output
10	PGOOD2	Channel 2 Open-Drain Power Good Output	
11	TRACKSS2	Output Tracking and Soft-Start Input, Channel 2	
12	GND	Power and Signal Ground	
13	SW2	Channel 2 Switch Node Connection to External Inductor	
14	BOOST2	Boosted Floating Driver supply for Channel 2	
15	VIN2	Power Supply Input for Channel 2	
16	VIN2	Power Supply Input for Channel 2	
17	SW2	Channel 2 Switch Node Connection to External Inductor	
18	GND	Power and Signal Ground	
19	GND	Power and Signal Ground	
20	SW1	Channel 1 Switch Node Connection to External Inductor	
21	VIN1	Power Supply Input for Channel 1	
22	VIN1	Power Supply Input for Channel 1	
23	BOOST1	Boosted Floating Driver supply for Channel 1	
24	SW1	Channel 1 Switch Node Connection to External Inductor	
25	GND	Power and Signal Ground	
26	TRACKSS1	Output Tracking and Soft-Start Input, Channel 1	
27	PGOOD1	Channel 1 Open-Drain Power Good Output	
28	VFB1	Channel 1 Output Feedback Voltage	Connect to resistor divider network to set output

Figure 16: Pinout for the LTC3636 in the Master Pinout File

Additional Figures and Supporting Material

LTC3636 Component Calculations and Simulation Results

The switching frequency of the SMPS should not be close to any frequency of wireless modules on the final data logger PCB design to avoid noise issues. The lowest LTE bands are 410 MHz, Wi-Fi bands are 2.4 GHz and 5 GHz, and the GNSS carrier frequency is 1575.42 MHz. Connecting R_T (the resistor responsible for switching frequency) to $INTV_{CC}$ selects 2 MHz as the switching frequency via an internal resistor, which would not interfere with the radio modules on the final board. To select the inductor values with a switching frequency of 2 MHz, where the ripple current $\Delta I_{L(MAX)}$ is set to 40 percent of the max output 6A as recommended by the datasheet[1]:

$$L = \left(\frac{V_{OUT}}{f * \Delta I_{L(MAX)}}\right)\left(1 - \frac{V_{OUT}}{V_{IN(MAX)}}\right)$$

$$L1 = \left(\frac{3.3V}{2MHz * 2.4A}\right)\left(1 - \frac{3.3V}{12V}\right) = 0.498\ \mu H$$

$$L2 = \left(\frac{3.8V}{2MHz * 2.4A}\right)\left(1 - \frac{3.8V}{12V}\right) = 0.541\ \mu H$$

With the inductor values known, the feedback resistors were next chosen. Using the below equation, R1 and R2 were selected for each output voltage. R1 refers to the bottom resistor and R2 is the top. R1 was set to 10 kΩ for both sides. For the 3.3V side (with L1), R2 was 45 kΩ. For the 3.8V side (with L2), R2 was 53.3 kΩ. Picking from standard 1 percent resistor sizes, the corresponding values are 45.3 kΩ and 53.6 kΩ.

$$V_{OUT} = 0.6\left(1 + \frac{R2}{R1}\right)$$

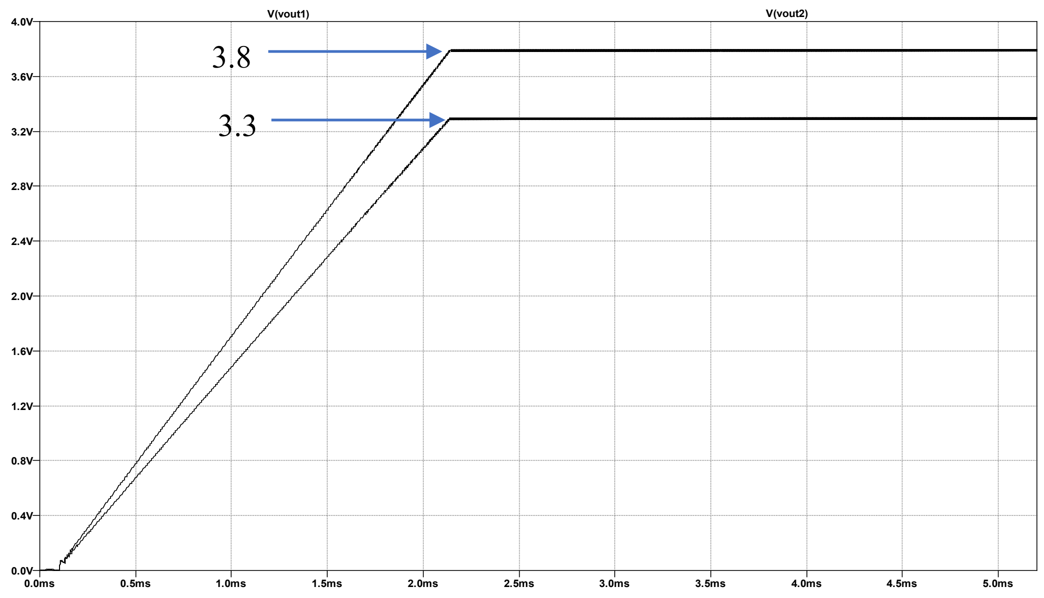


Figure 17: LTSPice Transient Simulation for the LTC3636 Design

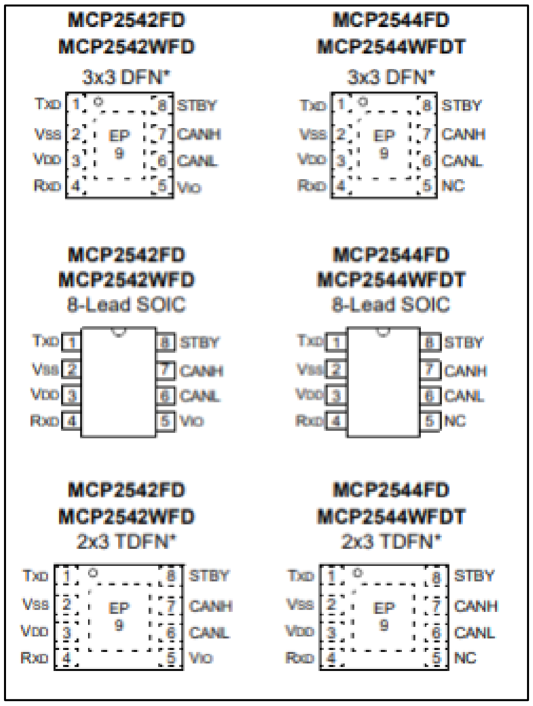


Figure 18: MCP2542FD Pinout

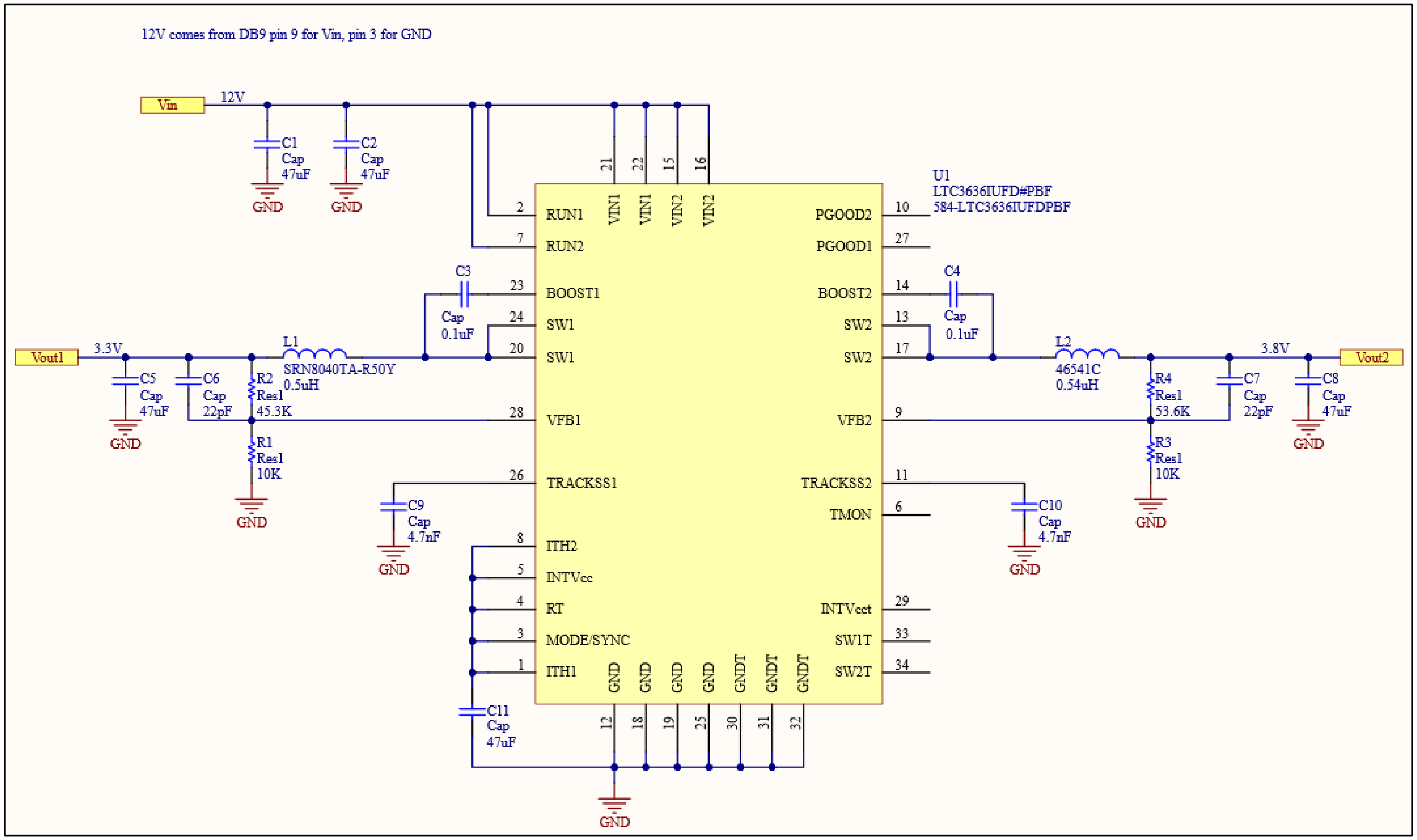


Figure 19: Schematic for 3.3V and 3.8V Dual-Channel Power Supply, LTC3636

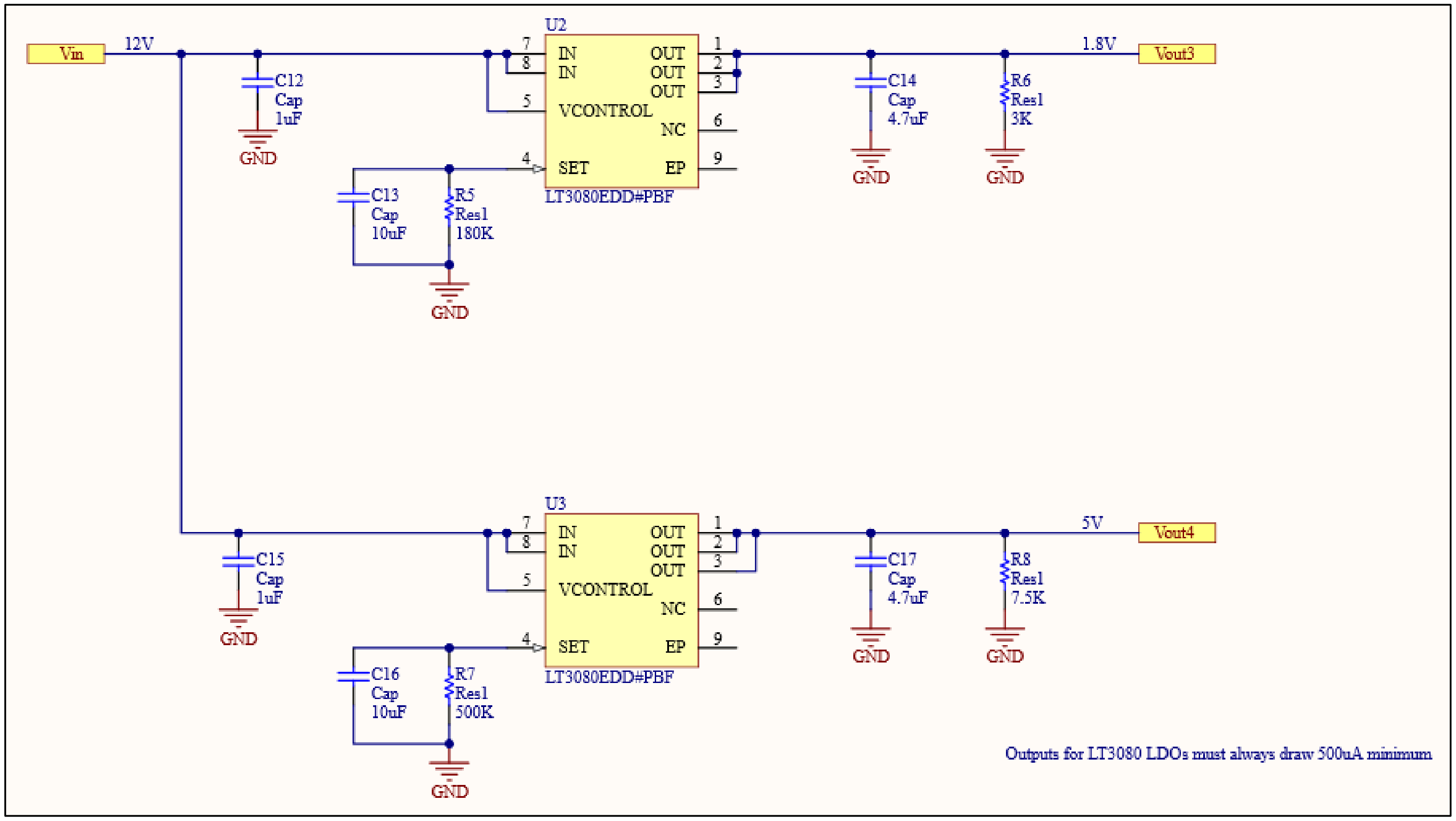


Figure 20: Sschematic for 1.8V and 5V Power Supplies, LT3080

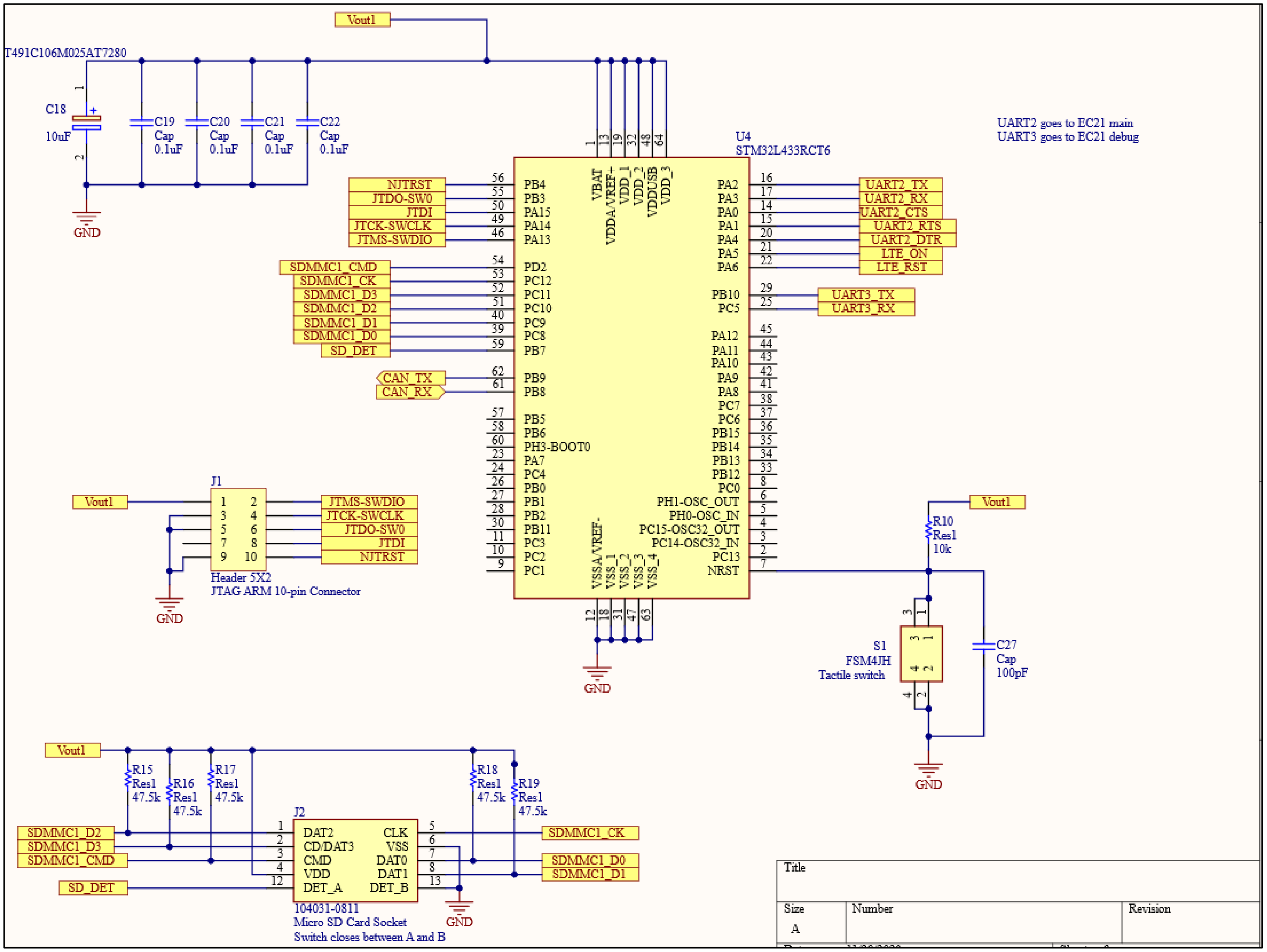


Figure 21: Schematic for the Microcontroller, JTAG Connector, and SD Card Socket

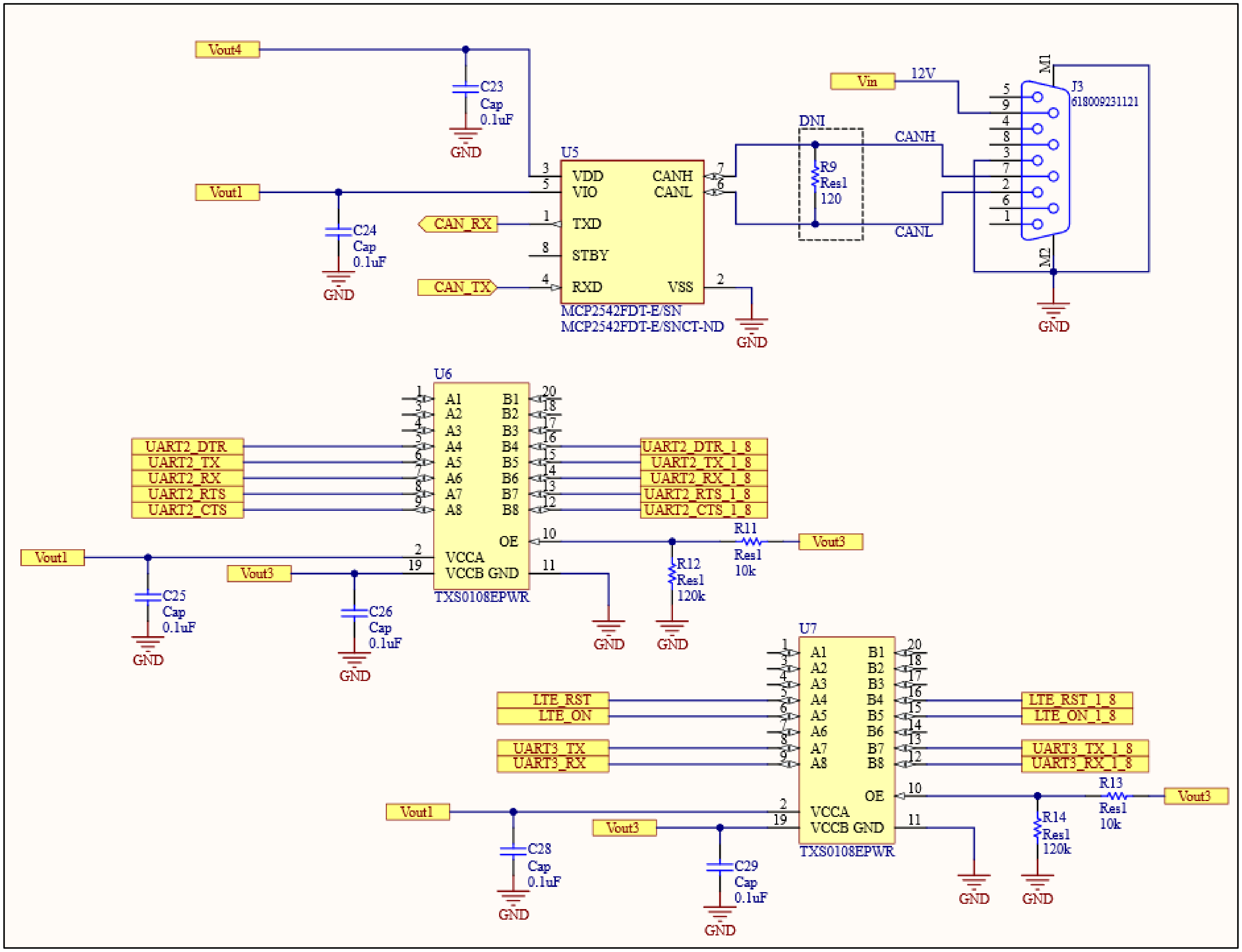


Figure 22: Schematic for the CAN Transceiver and Level Shifters

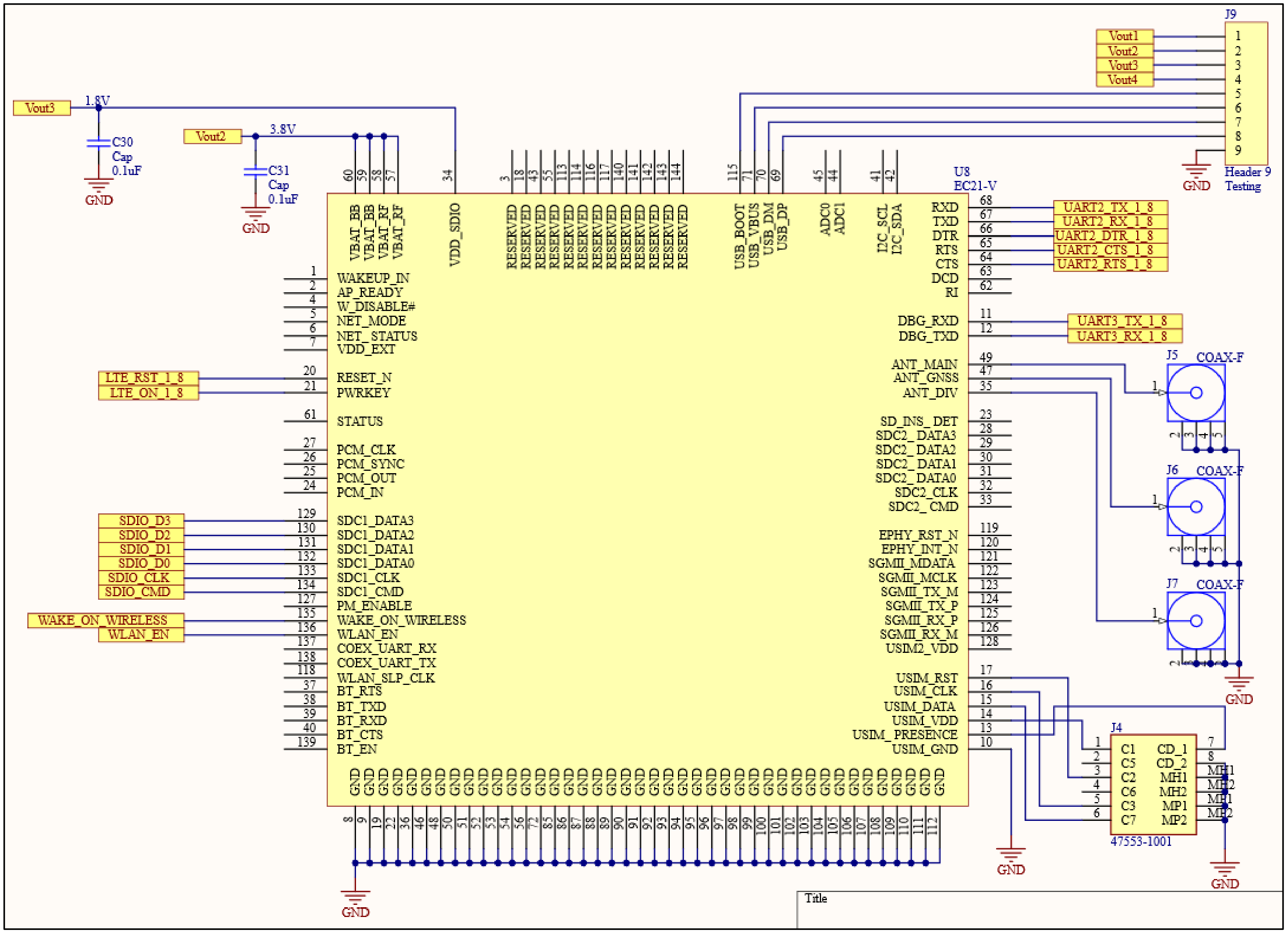


Figure 23: Schematic for the EC-21V Cellular Module and SIM Card Socket

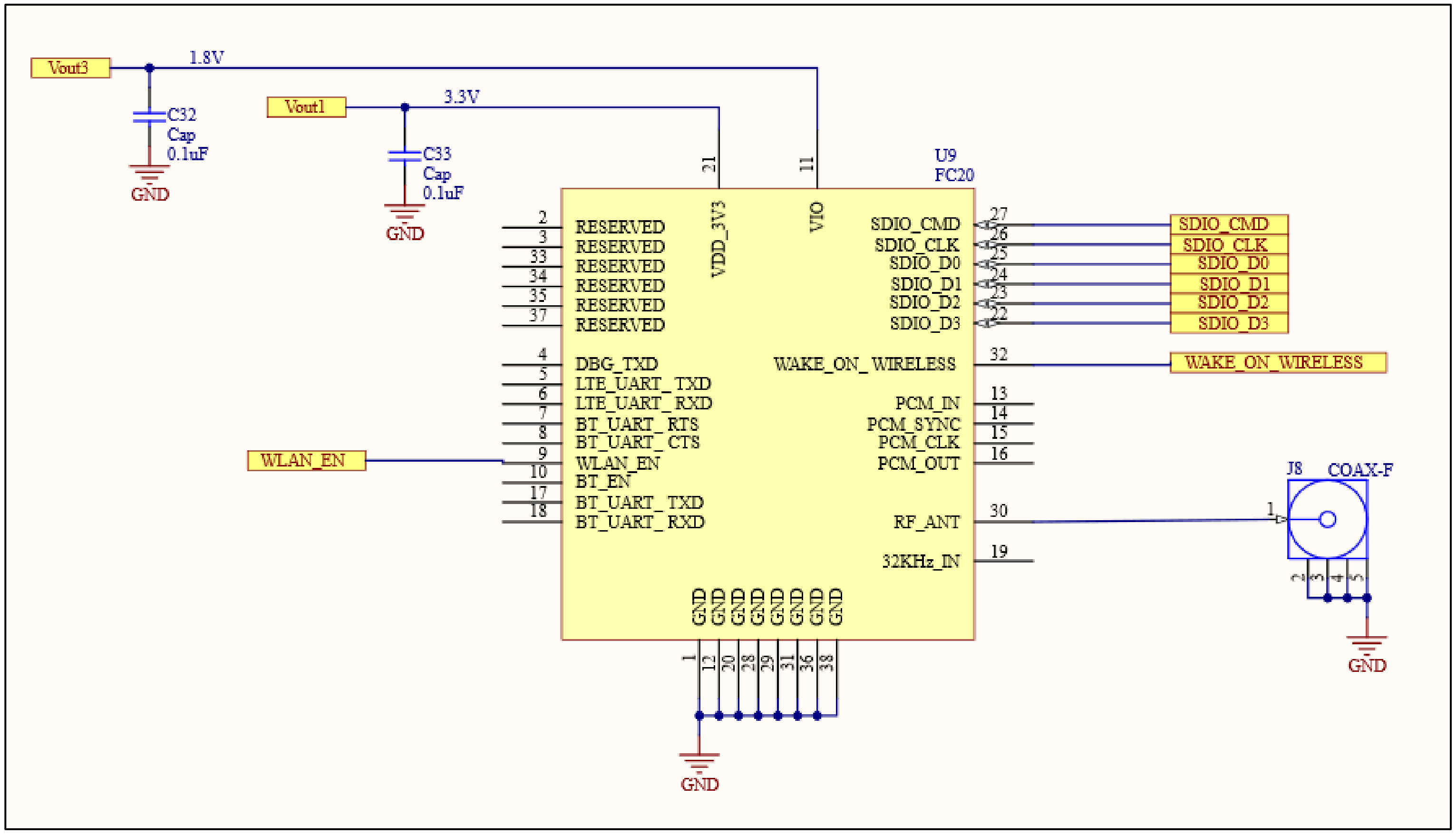


Figure 24: Schematic for the FC-20 Wi-Fi Module

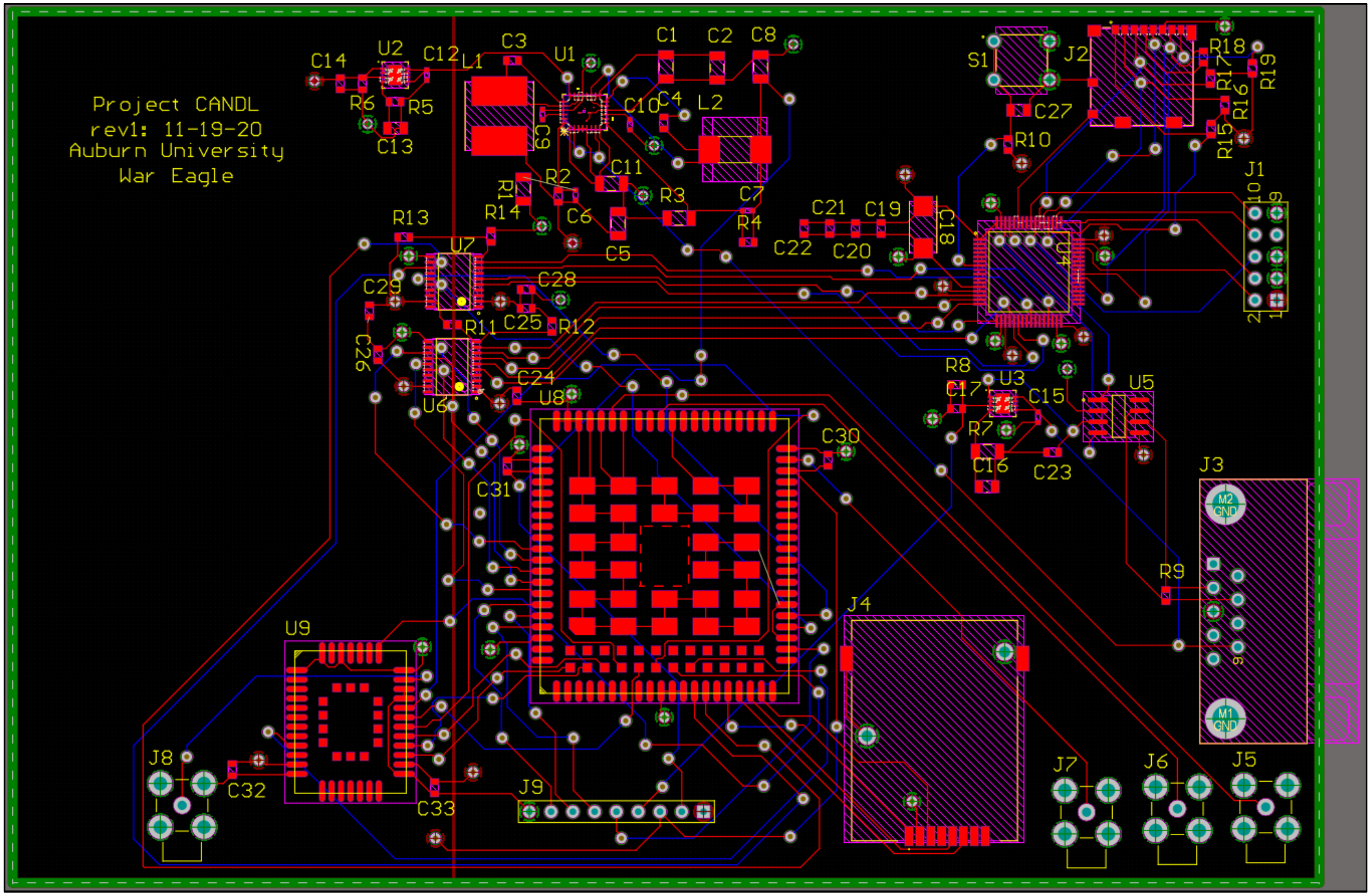

Project CANDL
rev1: 11-19-20
Auburn University
War Eagle
C14 U2 C12 L1 C3 U1 C1 C2 C8 S1 J2 R18 R19 R17 R16 R15
R6 R5 C13 C9 C10 C4 L2 C27 R10 J1
R1 R2 C11 R3 C7 R4 C21 C19 C18 C22 C20 U4
R13 R14 C6 C5 U7 C28 C29 C25 R11 R12 C26 C24 U6 U8
R8 U3 C17 C15 U5 R7 C16 C23 C30 C31 J3 M2 GND M1 GND R9
J4 U9 J8 C32 C33 J9 J7 J6 J5

Figure 25: PCB Design for Project CANDL Complete Board

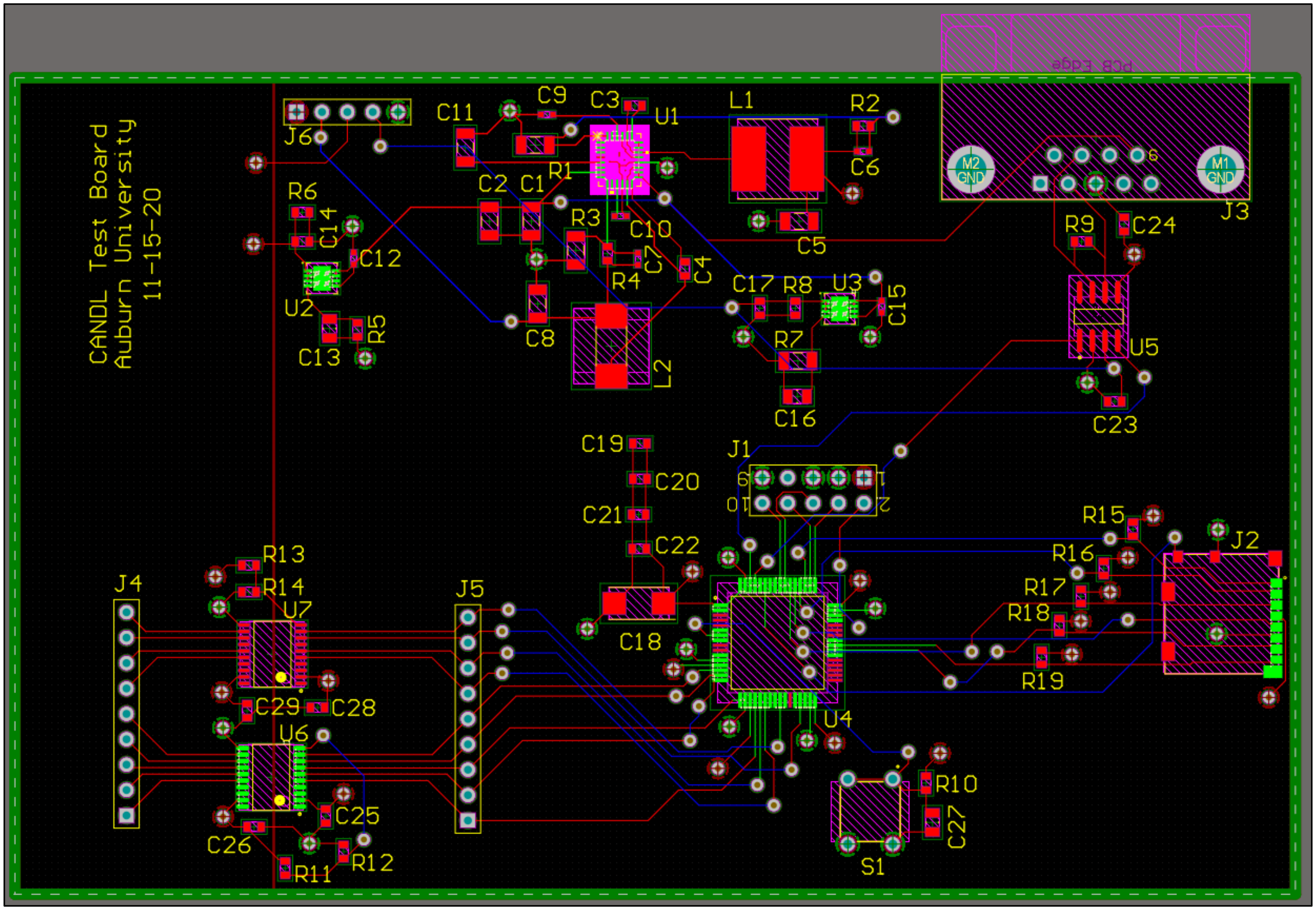


Figure 26: PCB Design for Project CANDL Test Board

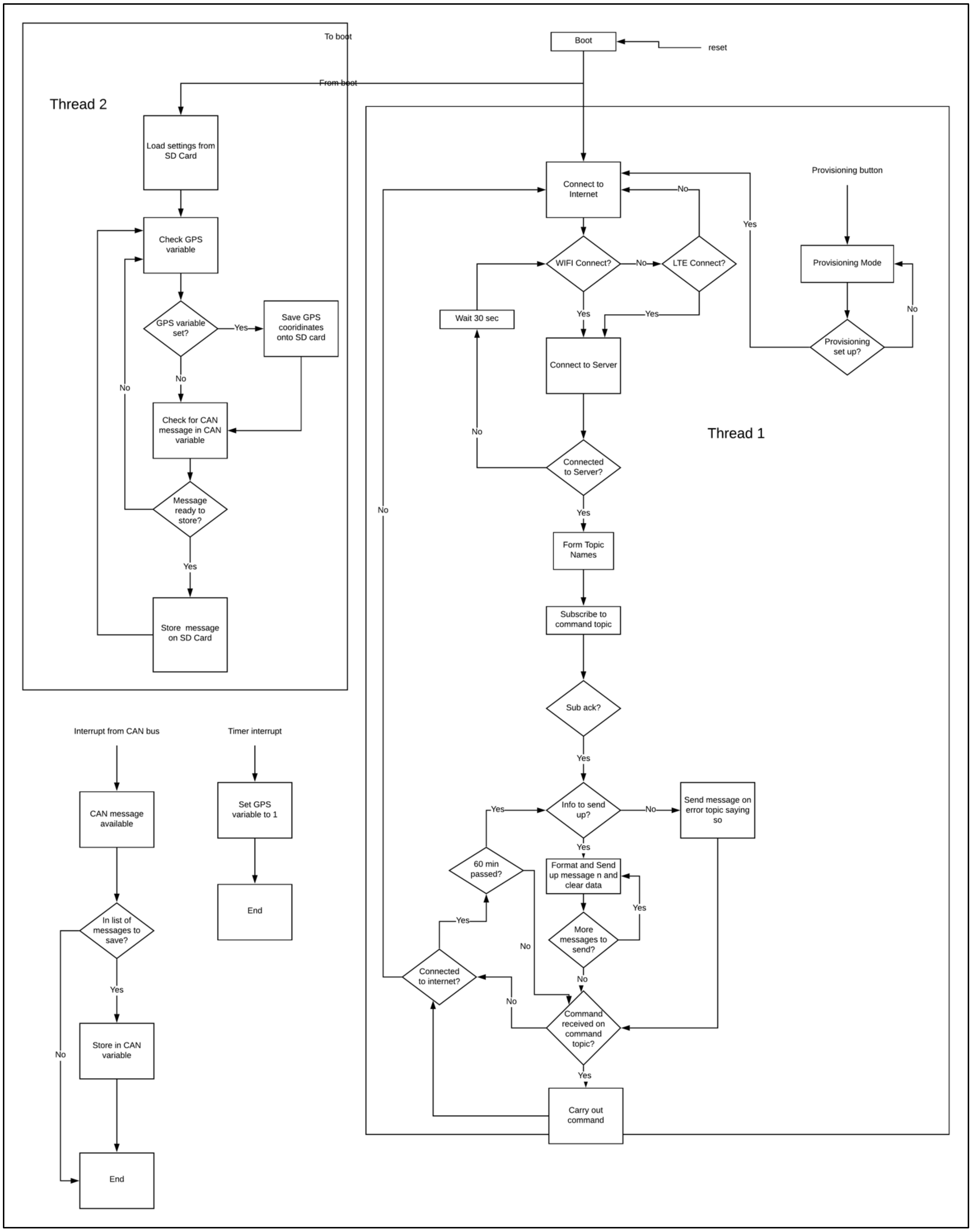


Figure 27: RTOS Flowchart, Operation of Two Threads

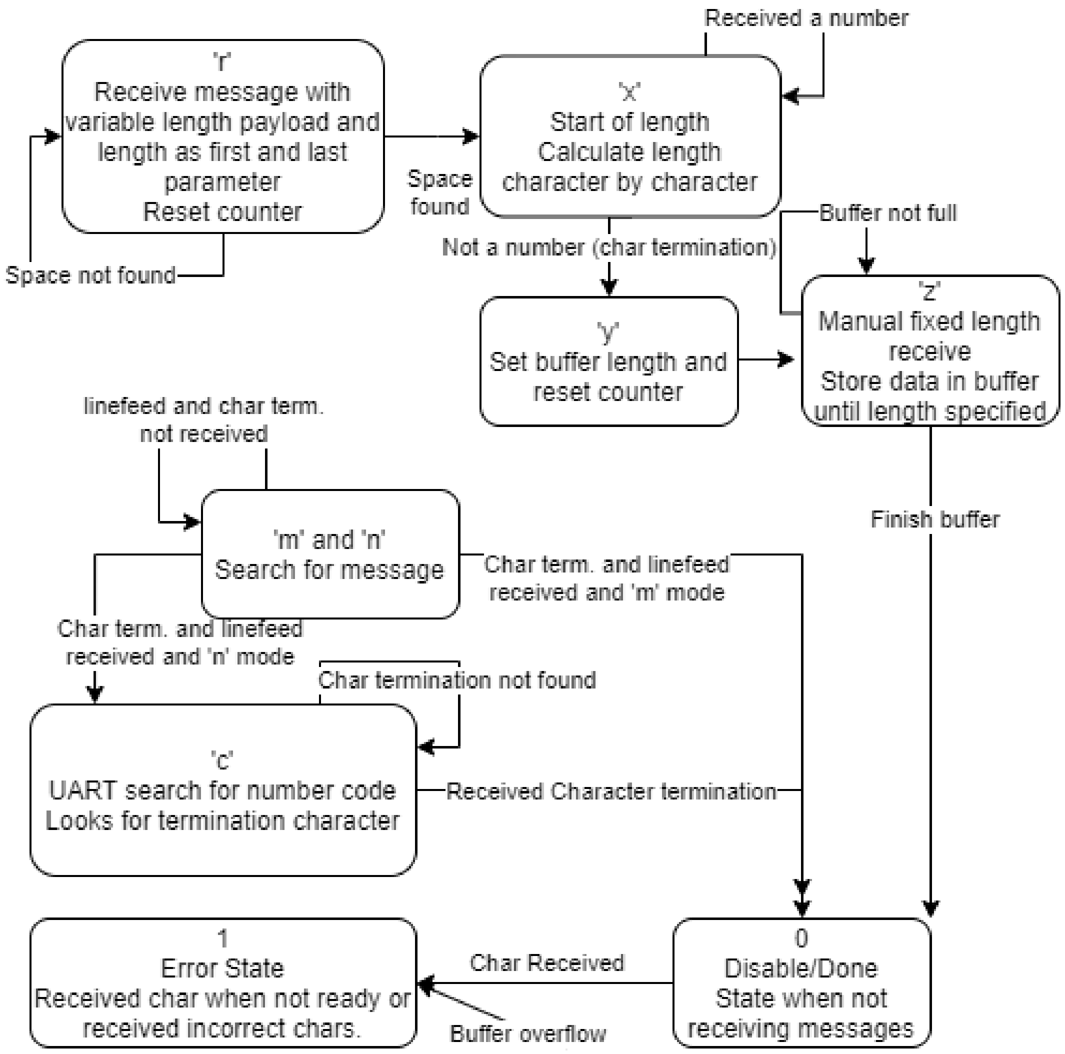
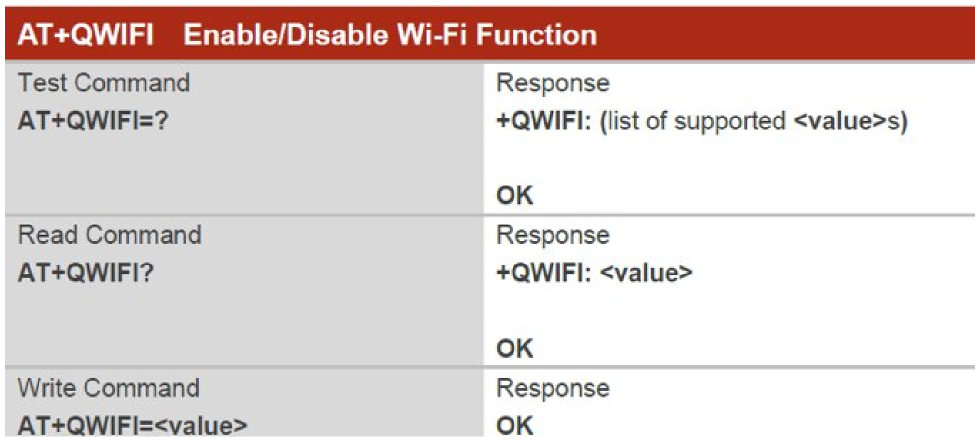

Received a number
'r'
Receive message with variable length payload and length as first and last parameter
Reset counter
Space found
'x'
Start of length
Calculate length character by character
Buffer not full
Space not found
Not a number (char termination)
'y'
Set buffer length and reset counter
'z'
Manual fixed length receive
Store data in buffer until length specified
linefeed and char term. not received
'm' and 'n'
Search for message
Char term. and linefeed received and 'm' mode
Finish buffer
Char term. and linefeed received and 'n' mode
Char termination not found
'c'
UART search for number code
Looks for termination character
Received Character termination
1
Error State
Received char when not ready or received incorrect chars.
Char Received
Buffer overflow
0
Disable/Done
State when not receiving messages

Figure 28: AT Main UART2 Interrupt Routine States



AT+QWIFI Enable/Disable Wi-Fi Function	
Test Command **AT+QWIFI=?**	Response **+QWIFI:** (list of supported **<value>**s) **OK**
Read Command **AT+QWIFI?**	Response **+QWIFI: <value>** **OK**
Write Command **AT+QWIFI=<value>**	Response **OK**

Figure 29: Example AT Command for the Quectel EC-21