MICROMOUSR

By

Andrew Betbadal

Nicholas Radler

Katelyn Schoedl

Final Report for ECE 445, Senior Design, Spring 2019

TA: Amr Martini

1 May 2019

Project No. 77

Abstract

The Petronics team proposed Micromosur as a proof of concept for a scaled down, less expensive model of the Mousr device. The final Micromousr device included much of the same Mousr functionality, including both manual control through Bluetooth and an autonomous game mode. The internal components were stripped to the essentials and printed on one board. Upon completion the device was able to maintain a closed-loop system and drive the appropriate PWM signals as needed by the different modes.

The results in this report have demonstrated that Micromousr is indeed capable of operation despite being stripped down to the bare essentials. All components of the system have been separately integrated with the main processor to communicate signals with each sensor, however we were unable to integrate all components together in one system. Motor function can be controlled through the existing app via Bluetooth, achieved in this project using a Bluetooth antenna module, or by the autonomous mode. The physical components were printed on one PCB and the price of electronic components fell to less than $20, significantly lower than that of Mousr.

Contents

1. Introduction

As technology has evolved to become more capable and affordable, the world has seen a significant change in the way we entertain ourselves. While human entertainment has become more technologically advanced, the way we play with our pets has basically stayed the same. Existing products include frisbees, treats, balls, and even puzzle games for pets, but the repetitive simplicity of these classic toys leave plenty of room for improvement.

Petronics is a Champaign startup founded in 2014 with the goal of creating the best automated cat toy in the world. They accomplished this with Mousr, a robotic mouse toy for cats that can operate autonomously or interactively through a user based app. The toy can not only entertain cats for half an hour on every charge cycle, but it also provides a fun and interactive bonding experience for cat owners. Mousr functions through a closed-loop feedback system, and the design includes a speaker system, three LED indicators, a proximity sensor, and multiple autonomous game modes. In comparison to traditional cat toys priced closer to $20 and under, this advanced product comes at a commercial price point of $150. Such a gap narrows the market for Mousr due to the high cost of the toy. MicroMousr is built at a low cost allowing the device to retail at less than half the cost of Mousr.

By condensing the internal components of MicroMousr into less parts, we have proven that this robotic cat toy is capable of providing almost the same basic functionality as Mousr at a price point much more attractive to consumers. We have proven modularly that the internal components are capable of existing on one single printed circuit board (PCB) that is housed in a body of similar size to the original Mousr. Encoders are no longer included in motor control feedback, relying solely on the other sensors in the device. The new product still allows for Autonomous and Manual control modes through the existing app. The Manual mode works similarly to a standard RC car, taking user input from Bluetooth and reflecting that in the mouse motion through pulse width modulation (PWM). In the Autonomous mode, the mouse can detect nearby objects and successfully maneuver the room, playing with reactions from the cat for about 30 minutes.

1.1 Objectives

For the MicroMousr proof of concept, we broke down our functional requirements into three high-level objectives:

1. Sensor components must exist on one board and be capable of fitting in the physical body of the Mousr product (3.4 x 2.2 x 1.4 inches).
2. Electronic components must cost less than the current Mousr model.
3. Must be able to demonstrate basic movements and control while in Autonomous Mode, and respond to user control when in Manual mode.

During the design process, we realized we needed to more specifically define requirements for the controls portion of the project. We spoke with Petronics regarding our demonstration of basic MicroMousr movements, splitting this requirement into two specific demos:

A. A remote control driving demo would rely on two inputs from the app: speed set point and yaw, which will be received by MicroMousr as BLE messages.
B. An autonomous driving demo would utilize the ToF and IMU to perform uncontrolled behaviors. These behaviors can be very simple, but chiefly emphasize a functional integration of sensors, actuators, and microcontrollers. For example, an 'auto-braking demo' could show the MicroMousr driving straight toward an object, stopping before hitting it, reversing away, and then driving back at the wall to stop again. The idea would be that this could run in a loop indefinitely.

We were also able to record specific measurements on the size of the existing Mousr components, and we realized the PCB itself would need to be considerably smaller than the measurements of the actual body. Mainly, the width of the PCB should be no more than 30 mm to fit in the provided gearbox. This is discussed in the physical design.

1.2 System Overview

The system-level block diagram (Figure 1) outlines the integrated operations of MicroMousr. We accounted for the power connections by checking maximum and minimum values for each component, and planned for I2C connections through available GPIO pins. I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers and other peripherals in embedded systems.

The final design did not deviate much from the original high-level plan, but we did need to adjust some of the pin connections between components (Appendix A). The basic I/O displayed in Figure 1 also included our push button and LED circuits, and we also chose to include a charge protection circuit within the charging connection. SPI was cross listed with I2C in the block diagram, but SPI was ultimately not involved with the final design.

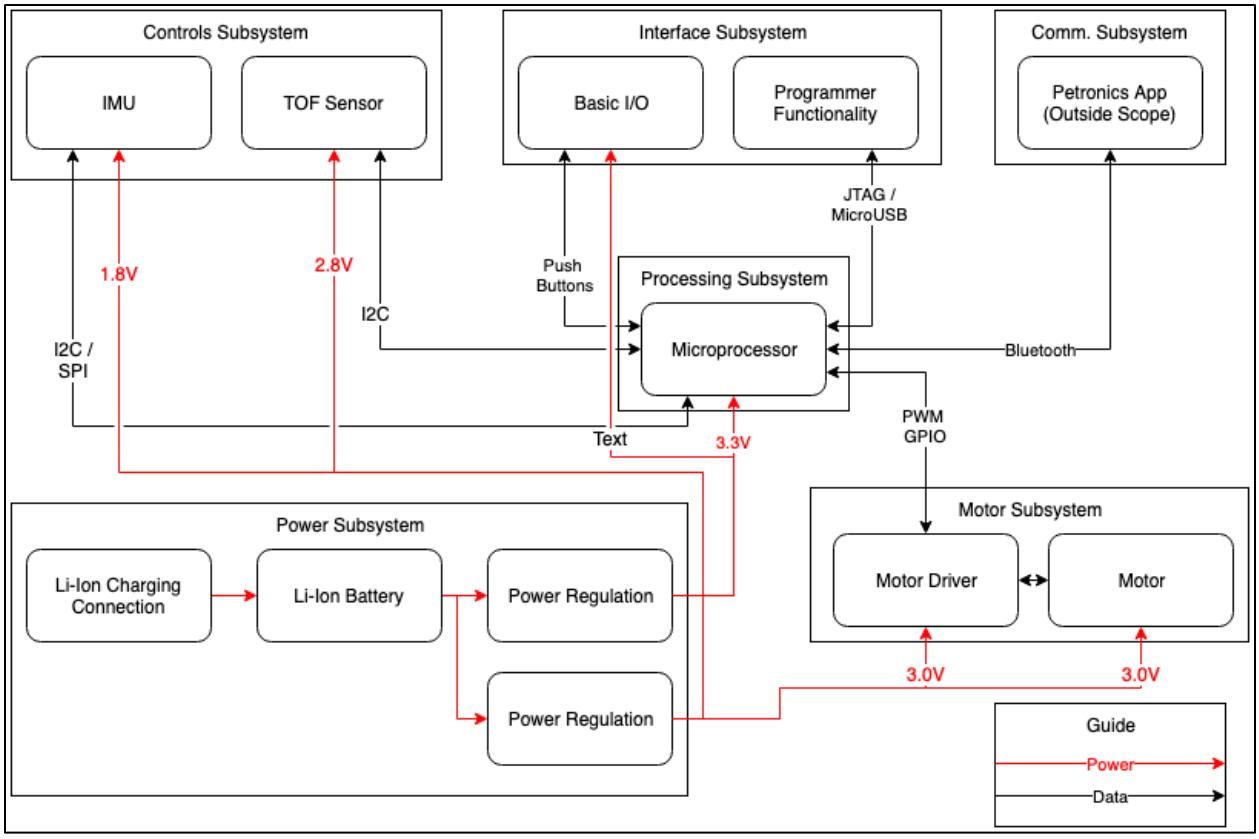


Figure 1 High-Level MicroMousr Block Diagram

In our group of 3 team members, quantity of work was evenly distributed across different portions of this project (Appendix B – Labor Distribution). Objectives 1 and 2 required verification of each pin connection, building a PCB with the appropriate layout and spacing, designing test boards for each subsystem, and ordering the compatible parts for the boards. The focus of objective 3 was the in-depth control algorithms and communication. This portion is ultimately required for the hardware connections to communicate with the microprocessor, allowing the motors to rotate according to sensor feedback. We first focused on completing the hardware portion of the project to comply with order timelines and to allow time for testing the incoming components, but in retrospect, we should have worked through the hardware in parallel with software (Appendix C – Schedule).

2 Design and Verification

2.1 Mechanical Design

In designing the hardware portion of our project, we needed to consider the size constraints of the existing Mousr body. The provided gearbox, consisting of two motors and a set of wheels, was measured to be 30mm wide at the points between the wheels. This would impact the design layout of the PCB. We needed to allow for a USB charging connection and an attachment to the Time of Flight sensor perched at a specific angle on the head of the PCB. The best way to account for these design components was to 3D print a base for holding the PCB and external components (Appendix D).

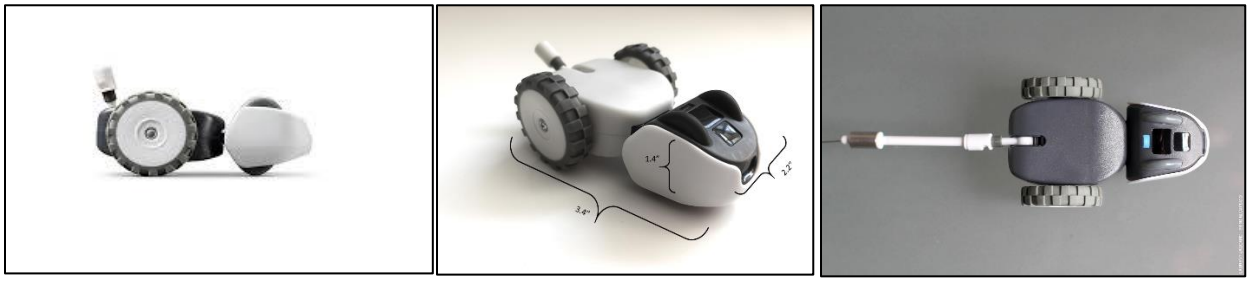


Figure 2 Mousr design (source: Petronics.io) 3.4 x 2.2 x 1.4 inches with body and wheels

2.2 Circuit Design Procedure

To design the system schematic, we not only needed to understand the pinouts for each chip but also the communication required for each connection to the microprocessor. The clock (SCA) and data (SDL) lines each required pullup resistors to enable communication. We chose 10k Ohm pullup resistors for each line:

$$Rp(min) = \frac{(Vcc - Vol(max))}{Iol} \quad (1)$$

Based on this calculation and the datasheet standard for this chip, we chose a value of 10k Ohms for each line. In the final design, the BMD-300 microprocessor antenna module actually included these pullup resistors, along with the passive requirements, so for this design the exact values were not a concern.

Although the BMD-300 module included passive components, we still had to apply the application schematic for the remaining chips. These were provided on the datasheets for each device. This depicts the connections necessary for communication with the chip. Additional considerations are explained in each section below.

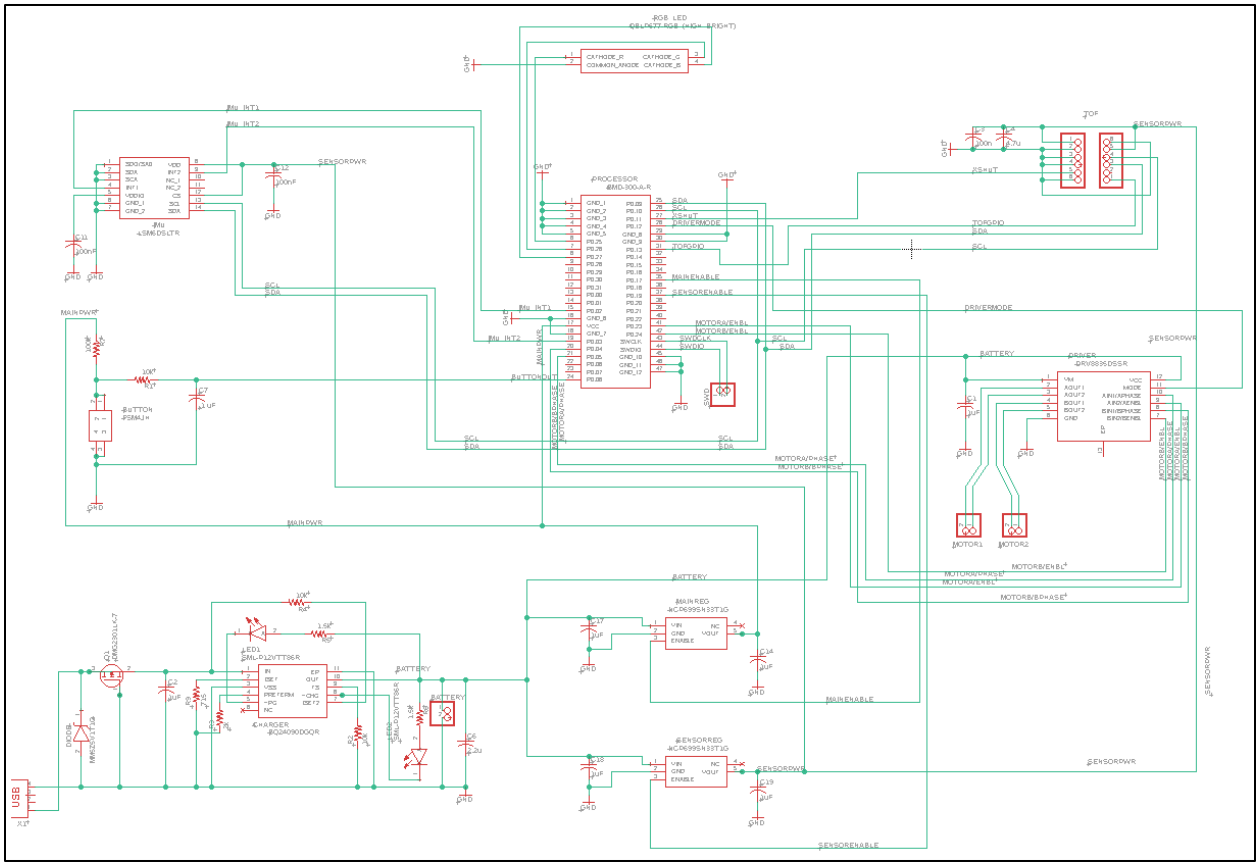


Figure 3 MicroMousr schematic in EAGLE

We also listed a set of requirements for each individual part of the circuit so we could modularly verify that each section was working early on. These tables can be found in Appendix E.

2.2.1 Power Supply and Charging Circuit

Component: BQ24090
Inputs: 120 V, I Set, Temperature Set
Outputs: 5V DC, ISET/100/500 mA

Li-Ion Battery: 4.2 V Rechargeable Li-Ion Battery
Inputs: Charge regulated current from MicroUSB
Outputs: 4.2 V + 2%

The charging circuit requirements were actually combined with the battery requirements in our final design. The battery is expected to be at industry standard so there was no need for us to charge and discharge the battery for such a long period of time. We made sure to read through the Li-Ion Battery Guidelines (Section 4.3 - Ethics and Safety) before integrating the battery as a load in our test circuits, and we could easily measure the voltage of the battery using a multimeter along with the current being drawn through the battery load of the charging circuit. This voltage was confirmed at 3.9 V (we had a 4.2 V Li-Ion battery with a 3.7 V nominal voltage) and powered the motors during our demo. The 4.2 V and 3.7 V refer to the maximum voltage of the cell and the average voltage of the cell, respectively, as the

battery begins charged at full value then begins to discharge. This voltage was capable of powering all chips on the circuit (power consumption table - Appendix F).

Since we did not get our charging circuit working, we were unable to to perform the verifications we originally planned. However, several hours of use throughout the time we were testing the motors and motor drivers did not have a noticeable change on the battery voltage, and because of this we concluded that the battery requirements are satisfied based on the calculations done for the power system in our design document.

Rather than using the exact application schematic for the BQ24090, Petronics recommended that we should include some sort of circuit protection using MOSFETs to control power flowing into the load and to prevent negative voltage input. We chose to use a P-Channel enhancement-type (normally off) MOSFET as a voltage controlled current source to VIN and ISET2 on the BQ24090 charging chip. After more research we ultimately decided to include two Zener diodes as transient voltage suppressors in the circuit. These are commonly used when protecting electrical nodes whose signals are bidirectional or can have voltage levels both above and below the reference voltage.

A single-layer copper test PCB was printed to test the values we calculated for this circuit before placing it on the board. Upon applying 4.2 V to the input and -3 V to the output with a larger 100k resistor acting as load, we read the current output of the BQ24090 as 0.4 mA and even by following safety guidelines the chip actually caught on fire. We redesigned the circuit and used an uncharged battery as a load, and did not apply a negative voltage since our assumption was that the negative voltage caused more current to be drawn than the chip could handle. This time we had an ammeter in series with the battery to monitor current. The chip did not heat up during this testing, but we were unable to get a current to flow and charge the battery. We have a few ideas on why this would happen. Our main guesses are that the diodes used for voltage protection may have prevented the flow of current into the proper inputs on the chip. Rather than solder the charging circuit to the final PCB, we decided to directly attach the battery to the LDO system to avoid any dangerous situations.

2.2.2 Voltage Regulators

Component: TI NCP699SN14T1G (x2)
Inputs: 4.2V +/- 2%
Outputs: +3.3 V to ToF, IMU, Motor Drive, LED +3.3 V to microprocessor, button

In this case the PCB contains two regulator chips, one providing 3.3 V to the main processing system and the other providing 3.3 V to the peripherals. We first performed this verification with the NCP69918 that we accidentally ordered instead of the NCP69933 - the difference between these two chips is that the former has an output of 1.8 V while the latter has an output of 3.3 V. This test is demonstrated in Appendix G, which shows both the debouncing circuit/push button connected with the LDO. You can see that in this test the chip worked properly, jumping from 0 to 1.8 V as soon as the button is pressed. This test was replicated with the 3.3 V NCP69933 however the output was not saved. Using the same plot process, it is obvious that the power jumps almost immediately, in less than 100 ms. Because we used the development board instead of the PCB for our final demo, we simulated the power plane shutdown

by triggering a GPIO low output from the development vboard when the button was pressed down. This output went to the voltage regulator enable pin, shutting off the output of the voltage regulator.

2.2.3 Microprocessor

Component: Cortex M4 Processor
Inputs: 2.0V-5.5V power, IMU and ToF sensor readouts via I2C, push button state, serial Bluetooth data
Outputs: PWM via GPIO to Motor Drive, serial Bluetooth data to Petronics App

Our original design was to use an nRF52832-QFAA as the microprocessor. We chose this since it is based around a Cortex M4, has Bluetooth low energy capabilities, and would integrate with the software already written by Petronics. We decided to switch to the BMD-300 for our final design since it is built around an nRF52 but already has an antenna built in. We made this design change because designing and building an antenna is outside our scope but Bluetooth capabilities are absolutely integral to the overall project.

Because the microprocessor was such a core part of our demonstrated functionality, much of the requirements were verified through the end operation of our project. As demonstrated in lab, we drove 3 PWM signals from our microprocessor : 2 signals to the motor driver and 1 signal to the LED circuit. As demonstrated, our final product interfaced with all 3 of the Petronics App driving modes : Manual Mode Simple (singular throttle value and 360 degrees of direction), Manual Mode Complex (variable throttle value and 360 degrees of direction), and autonomous mode. The PWM duty cycle to the motors was dependent on a combination of throttle and turn angle. In forward mode, the duty cycle was just the percentage of throttle passed in via Bluetooth, and in turns the PWM duty cycle of a single wheel was set to reflect the magnitude of the turn angle. In autonomous mode, we took input from the TOF sensor via the I2C bus and at 70 mm sent the car into a spin, rotating the left wheel backwards and the right wheel forwards, until it had at least 800 mm of space in front of the device to go straight again. Seen below, we connected the microprocessor GPIO output to the enable signal of the 3.3 V LDO, and with push button GPIO input toggled the enable signal. During the demonstration, at microprocessor boot up we immediately transitioned from off mode to to active mode with multi-thread initialization. As an additional consideration, we made sure the I2C traces were of similar lengths on the PCB design with minimal sharp edges so the signals could travel in phase with each other. Differences in the length of the ToF and motor attachments was not significant to change these values.

2.2.4 Time of Flight

Component: VL53L0X ST Time of Flight Sensor
Inputs: +1.8V Supply (2.6V-3.5V Operating Range), I2C Serial Clock, I2C Data line, XSHUT signal
Outputs: I2C Serial Data

The Time of Flight sensor was used as part of our design so that we could include obstacle avoidance as part of our autonomous mode. This chip provides a very low amount of noise in the data, in a tiny package, at a very low cost. This helps with fitting everything on one board, helps keep our total product cost low, and is integral to our autonomous mode making it influential for all of the high level requirements.

All of the ToF requirements were demonstrated in our final design with the custom ToF board mounted via soldered headers. Because the PCB on the vehicle chassis we used sits several centimeters above the ground, we did not get signal interference from ground recognition and the ToF positioned at the front with flexible wires gave clear field of vision. The demo vehicle was soldered in one position, but also had a clear field of vision. In our autonomous mode demonstration, the vehicle enters a spin once an object is detected 70 mm away. In order to achieve this, we pulled the raw ToF data in mm and enumerated an action space based on the current ToF data and the current robot action state (spinning, not spinning, etc.). Our final product drove the ToF Enable signal high once the microprocessor was booted up, and switching to autonomous mode immediately triggered ToF reading.

2.2.5 Inertial Measurement Unit

Component: ST LSM6DSLTR Accelerometer/Gyroscope
Inputs: 1.8 V Supply (2.6 V-3.5 V Operating Range), I2C Serial Clock, I2C Data line, CS enable
Outputs: I2C SDA data, SDO address bit

The IMU is part of the design so that we would be able to have more sophisticated control in autonomous mode. It is composed of an accelerometer and a gyroscope which measure device movement about the x, y, and z axes and rotation rate around the axes. This particular chip, like the Time of Flight sensor, was chosen for its size, cost, and low level of noise.

We were unable to integrate the IMU in the final demo. During debugging, we realized that the I2C communication was never initializing for the IMU, but was for the TOF. By probing the chip development board we realized that the LSM6DSL had been fried through an applied over voltage. By the time we realized this, we had already begun developing the high level application file using just the TOF and user input. In order to integrate the IMU, we would have had to change our polling process for TOF data, as the IMU is set up to poll continuously until an interrupt is received from the TOF to prevent overwriting data. Additionally, the IMU would have been primarily useful for a more advanced high level game control, which we had not written, so integrating the chip at the time did not seem to be a worthwhile endeavor. If we were to repeat this design process, we would have more carefully tested the development board and prepared compatible polling for both ToF and IMU sensors from the beginning.

2.2.6 Motor Driver

Component: DRV8835DSSR
Inputs: PWM signal to A/B Phase pins, Logical High to A/B Phase pins and Mode pin, VM (motor voltage), VCC (Chip power)
Outputs: A/B OUT1 and A/B OUT2 which connect to the motors.

This motor driver was chosen due to the fact that is a dual channel H-bridge IC. This was important since it can drive both motors from a relatively small footprint. It takes an input for direction and a PWM for each motor, and based on the duty cycle of the PWM allows the motors to pull current directly from the battery. This chip was also chosen since the motors pull a maximum of 1.2 A and the chip is capable of handling 1.5 A.

In RC mode, we are not only able to drive the PWM duty cycles using Bluetooth input from the Petronics app, but we are also able to rotate forwards and backwards in both directions. We utilized the phase/enable mode of the DRV8835 in our firmware, such that for each motor we had an input controlling the direction and the PWM duty cycle. When the GPIO input from the push button is toggled, we send the duty cycles for the motors to zero. In autonomous mode, the motors both drive straight forward and any input from the app is replaced with input from ToF sensor (as described in the Microprocessor requirements).

2.2.7 RGB LED

Component: QBLP677-RGB
Inputs: 3.1 V PWM
Outputs: RGB color spectrum

Our LED is controlled by PWM duty cycles. In our demo, we only integrated the red cathode as we only needed to use one color for proof of concept of our immediate tests. The light was kept on, and when the system was placed in reverse mode, the duty cycle was increased causing the light to get brighter. This also proved the microprocessor could integrate with various systems. If we were to repeat this process, we could have used the geen and blue cathodes to distinguish between modes.

2.2.8 Push Button

Component: FSM4JH
Inputs: 3 V
Outputs: Signal High

We utilized the push button in our final demonstration. After building the debouncing circuit, we first connected the output to the oscilloscope, driving the input at 5 Volts. We confirmed that the circuit had no bounce, and then integrated the circuit by configuring a GPIO input to transmit an ORB message on the GPIO toggle. We drove the PWM motor 1 and 2 duty cycles to zero on the push button signal, and re enabled the duty cycles again on the next GPIO input toggle. In our demonstration circuit, we configured pin 31 as the push button input. The output of the debounce circuit is shown in Appendix G - Figure 5.

2.3 Software Design Procedure

2.3.1 System Initialization Procedure

During initialization, our system kicks off the individual thread tasks. As referenced in Appendix H -Figure 6, we first kick off peripheral tasks. Bluetooth, push button, and the Time Of Flight sensors are first started and verified before beginning motor drive and high level control tasks. Our high level controller subscribes to messages from both the TOF and Bluetooth, so those systems must be initialized prior to beginning the high level controller. Additionally, the motor driver subscribes to messages from the high level controller, so it must be initialized after peripherals. We chose to initialize the motor driver before the high level controller, however, because we poll the high level controller in a thread blocking statement, so the motor driver will not continue it's task until a message has been received from the high level controller.

2.3.2 High Level Multithreaded System Description

Our project's firmware is divided amongst several threads that run independently but communicate via the ORB message passing protocol, as referenced in Section 2.3.4. Because our system receives various peripheral GPIO and BLE inputs, we needed to ensure the latency of one signal would not affect the reception of another. As seen in Appendix H - Figure 7, our high level controller was the focal point for our multithreaded system, which will be discussed in more detail in Section 2.3.3.

2.3.3 ORB Message Passing Protocol

The ORB Messaging protocol was proprietary firmware that Petronics uses in their current system. In order to communicate via ORB, we first had to create topics. These were struct definitions that explicitly described what the contents of a specific message were. In our project, we created two unique topics which can be seen in Appendix H - Figure 8. The first, BSPSignal, contained only a boolean value that was toggled when the button was pressed. Additionally, rcPWM, contained two float values and two boolean values. Once these topics were declared, we were able to set up advertisers and subscribers. Once a thread task declared an advertiser for a topic, we continually published that topic's subscribers. Each topic could have multiple subscribers and advertisers.

In order to subscribe to a topic, there were two functionalities we took advantage of. The first, poll, was a blocking statement, so that tasks were not able to continue until a message was returned to the polling function. This ensured that tasks that relied on messages from other threads would not continue until those messages were received. This is especially critical in our motor driver task, as we would not assign duty cycle values or motor directions until we had received direction from the high level controller.

The other function we used was checkAndCopy. This was a non-blocking statement, allowing threads to continue and returning a boolean true if a message had been received. We primarily used this in our high level controller when dealing with BLE messages. A user will not be interacting with our application all the time, so we did not want to hold the rest of our controller function while waiting for new messages from the Bluetooth module.

2.3.4 High Level Controller Design

Our high level controller was the central decision maker in our project. The flowchart diagram for our high level control can be seen in Appendix H - Figure 9. With every iteration of our controller task, we first polled for Bluetooth messages. We wanted the user to be able to override current system state at any point. As mentioned in the previous section, this was a non-blocking query, enabling the system to continue without constant user input.

If there were an incoming BLE move command, we enumerated on the command's angle. As visualized in Appendix H - Figure 10 , we had 6 unique cases based on the command's angle, and set motor 1 and 2 direction accordingly. The command's throttle value was used to set the intensity of each wheel, but the angle determined the percentage of the full throttle that would be distributed to the wheel's duty cycle. As an example, if the angle is 89 degrees, indicating an aggressive right turn, the right and left wheel's direction was set to forward, but the right wheel's duty cycle percentage was set to almost 0. We

assigned the ORB message values r_PWM, l_PWM, spin, and reverse accordingly and published it to the motor driver.

If there were not a Bluetooth command ready and we were in autonomous mode, we polled the TOF sensor. This was polled using the blocking query, as the TOF input was crucial for deciding our state in autonomous mode. We enumerated on this TOF data as visualized in Appendix H - Figure 11. Our primary task in autonomous mode was obstacle avoidance, so MicroMousr stopped and spun until there was no longer an object close to it according to the TOF input. Similarly to RC mode, we assigned r_PWM, l_PWM, spin, and reverse accordingly and published to the motor driver.

2.3.5 Driver Design

The nrf52 PWM driver base allows PWM sequence creation with specification of values, channels, and clock frequencies. We created three separate channels, one for the right and left motor PWM and the LED PWM signal. Because the nrf52 driver base does not allow mid-sequence duty cycle changes, we set the PWM sequence to repeat and created a PWM event handler. As seen in Appendix H - Figure 12, this event handler constantly listens for the PWM events to finish. We were able to modify the duty cycle by creating global variables to represent the percentages we wanted for each PWM. Once a PWM event finished, we multiplied these percentages by the max duty cycle frequency and assigned them to their respective PWM channels.

Within our PWM event, we polled for the rcPWM signal passed from the high level controller using a blocking statement. Once we the signal was received, we enumerated on the boolean Spin and Reverse first, as these two booleans set the direction for the motors first. Once we had the direction, we assigned the r_PWM and l_PWM float value in the received signal to our global variables. These global variables would be used in the PWM event handler.

3. Costs

3.1 Parts

The cost of Mousr components cannot be disclosed due to our agreement with Petronics. However, the cost was more than halved (Appendix I). The most expensive component of the board, aside from the actual PCB manufacturing and shipment, was the BMD-300 microprocessor antenna module at $11.27 for a single chip. If we eventually switch back to the NRF52832 without an antenna module, the price goes down to $5.73. This is something to consider for the future.

3.2 Labor and Prototyping

All lab machinery was used within the ECE building at the University of Illinois. See Appendix I for a cost breakdown and more details. We were able to utilize the ECE Machine Shop to create basic test circuits with a fast turnaround time. We also visited MakerLab for the 3D printed chasse.

4. Conclusion

4.1 Issues and Uncertainties

We encountered several unexpected issues as we began full integration of MicroMousr. During firmware development, the LSM6DSL component we ordered came to us fried. Everytime we attempted to establish I2C communication, the TOF would acknowledge the slave-master connection and the IMU would not. We continued development, fully integrating the TOF, before we soldered a new IMU component onto the dev board we were utilizing. At this point, however, our firmware had already been written for a single peripheral on the I2C bus, so integrating the IMU would have required developing an interrupt driven system, so that the I2C bus data would never be overwritten. This was too close to the demonstration date, however, and we had already built our autonomous control to operate solely on the TOF data.

Another issue we encountered was transitioning from the nrf52832 development board to the BMD-300 SMD chip. While we were able to establish communication and debugging through the exposed SWDIO and SWDCLK pins on the chip, our flashed code was unable to run properly. After debugging and research, we realized that the BMD-300, while an nrf52 based chip, has its own low level driver libraries. In order to integrate our application on the SMD chip, we would have had to rewrite the drivers we wrote and the drivers Petronics gave us, unable to use all the libraries already integrated into the code base.

4.2 Accomplishments

Overall we were able to prove that Petronics will be able to bring a more affordable version of Mousr to the market. There were many small accomplishments along the way that made this possible. On the hardware side, designing the PCB such that all of our components were on one board and in locations that make sense in the placement in the body of the device was integral. This was especially important when it came to the Time of Flight sensor which had to be attached at the front of the board, but angled so that it had the proper orientation compared to the ground and device. The I2C lines had to be the same length and without sharp angles. Being able to have a board that worked considering all the restrictions was a big milestone.

From the software side, a big accomplishment was getting the I2C protocol working and establishing communication between the sensors and the microprocessor. On top of this, we were able to get drivers for the motors, PWM, and push button working. We were able to get all of these threads working together through a publish/subscribe based communication system. From a high level perspective, we were also able to get manual and autonomous modes working which pull from bluetooth and the time of flight and update PWM cycles to the motors and LED in response.

4.3 Ethics and Safety

This project was pursued in good faith of ourselves, the UIUC electrical engineering department, our sponsor Petronics, and their customers. As stated in the IEEE Code of Ethics Item 1, we must “hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or

the environment." In light of this, there were several potential safety hazards involved with designing and prototyping the MicroMousr system. We prototyped our device only in approved spaces with adequate equipment and supervision. We exercised caution while soldering and working with components of the battery, as emphasized in the Safety Quiz and safety guidelines of this course. ECE 445 safety guidelines state that designs utilizing a lithium ion battery must work with extra caution. The Safe Battery Practices document states that the battery must always be stored in a secure location with the terminals covered by insulating material (Section III). Additionally, we were "REQUIRED to find the Material Safety Data Sheet (MSDS) and data sheet" and we kept this documentation on hand at all times in the lab (Section IV). We were sure to NEVER over charge, over discharge, over heat, or short circuit the battery. We were also sure that the final product was safe, with no risk of the human or the cat exposing internal electronics or otherwise misusing the product in a way that could produce electrical harm. To do this, we considered all possible environments the product may be used in. The parts we chose are able to withstand environments such as extreme temperature or humidity with functional temperature ranges from -20°C ~ 70°C. Should the parts malfunction, nobody should be harmed at a maximum charge of only 6 volts. We built test boards for each subcircuit to test before integrating them on the main board. This was especially useful when we had issues with current draw from our battery circuit test board causing the charging chip to overheat. Because of this, we ultimately did not include the charging circuit on the final PCB.

In addition to prioritizing physical design safety concerns, there are ethical considerations to be made when working closely alongside an established company. The ACM Code of Ethics Item 1.2 states "examples of harm include unjustified physical or mental injury, unjustified destruction or disclosure of information, and unjustified damage to property, reputation, and the environment. This list is not exhaustive." To adhere to these guidelines, we keep proprietary information out of reach of unauthorized parties and respect the donation of time and supplies from Petronics and the engineering department. We respect the reputation of Petronics and were frugal with the resources they provided for us. Misuse of available resources would have led us to directly break IEEE Code of Ethics Item 9, "to avoid injuring others, their property, reputation, or employment by false or malicious action." We must adhere to ACM Code of Ethics Item 1.7, "honor confidentiality." Throughout the development phase, our project team worked together "to assist colleagues and co-workers in their professional development and to support them in following this code of ethics" (IEEE Code of Ethics Item 10). We supported one another in our own development and strove to make the design process as seamless as possible. We did this as we strove "to achieve high quality in both the processes and products of professional work" (ACM Code of Ethics Item 2.1).

4.4 Future Work

In the future, we'd like to use the NRF52832 with an antenna trace built onto the board rather than the built-in antenna module. This will require more work in changing the PCB layout to account for the extra space and passive components, but would eliminate all driver issues that disconnected hardware and software. In addition, the charging circuit uses two indicator LEDs to display status. The PG and CHG pins could instead be connected to extra GPIO pins on the microprocessor. In this situation, the chip could immediately end all running systems when connected to the charger, or stop charging for certain

signals. When the MicroMousr is operational, we'd like to incorporate more games nearing the complexity of Mousr.

4.5 Acknowledgements

We would like to thank the University of Illinois Electrical and Computer Engineering department, ECE 445 course staff, our teaching assistant Amr Martini, and most importantly, Petronics, for their endless support for this project. The MicroMousr project would not be the same without the guidance provided by every supporting member of our larger team.

References

[1] ACM. "Code of Ethics and Professional Conduct." [Online], June 22, 2018. Available: https://www.acm.org/code-of-ethics

[2] Amazon.com. "Petronics Mousr Interactive Robotic Cat Toy." Product, 2018. Available: https://www.amazon.com/Petronics-Interactive-Robotic-Automatically-Smartphone/dp/B07HBC5M6Q/ref=cm_cr_arp_d_product_top?ie=UTF8

[3] Renewable Energy Innovation. (2014). *Charge Controller Project – Power Switching* [Online]. Available: https://www.re-innovation.co.uk/docs/open-charge-regulator/charge-controller-project-power-switching/

[4] R. Quan. (2017, Oct. 13). *A Guide to Using FETS for Voltage Controlled Circuits, Part 1* [Online]. Available: https://www.edn.com/design/analog/4458955/A-guide-to-using-FETs-for-voltage-controlled-circuits--Part-1

[5] STMicroelectronics, "World's Smallest Time-of-Flight ranging and gesture detection sensor", VL53L0X datasheet, May 2016 [Revised Apr. 2018]. Available: https://www.st.com/resource/en/datasheet/vl53l0x.pdf

[6] Texas Instruments, "bq2409x 1A, Single-Input, Single Cell Li-Ion and Li-Pol Battery Charger", BQ24090 datasheet, Jan 2010 [Revised Aug. 2015]. Available: http://www.ti.com/lit/ds/symlink/bq24092.pdf

[7] Nordic Semiconductor, "nrf52832 Product Specification v1.3", nrf52832 datasheet, Jul. 2016 [Revised Feb. 2017]. Available: https://www.mouser.com/datasheet/2/297/nRF52832_PS_v1.3-1117956.pdf

[8] STMicroelectronics, "iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope", LSM6DSL datasheet, May 2017 [Revised Sept. 2017]. Available: https://www.st.com/resource/en/datasheet/lsm6dsl.pdf

[9] IEEE. Policies, Section 7 - Professional Activities. "Code of Ethics." [Online], February 2006. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

Appendix A Pin Connections

Table 1 Pinouts

Part	Pin	Connection	Function
Microprocessor	2	ToF, IMU I2C SCL Connection	I2C SCL
	3	ToF, IMU I2C SDA Connection	I2C SDA
	4	LSM6DSL SD0	Defining IMU I2C address LSB
	14	Voltage Regulator 1 Enable	Enable / Disable Voltage Regulator
	15	Voltage Regulator 2 Enable	Enable / Disable Voltage Regulator
	16	TTDM on bq24090	Timed charging disable / enable
	32, 33	Decoupling circuit	Required for radio power supply decoupling
	41	DRV8835DSSR (MotorDrive) AENBL	PWM output for H-Bridge A
	21	DRV8835DSSR (MotorDrive) APHASE	Direction control for H-Bridge A
	42	DRV8835DSSR (MotorDrive) BENBL	PWM Output for H-Bridge B
	20	DRV8835DSSR (MotorDrive) BPHASE	Direction control for H-Bridge B
	41	LED Cathode	PWM Output for RGB LED
	42	FSM4JH	Input for push button
	13, 36, 38	To LDO Power Supply	VDD

	Die Pad	VSS	Required for chip operation

Table X. Microprocessor Pin Connections

Part	Pin	Connection	Function
Time of Flight	1, 11	To LDO power supply	AVDD for chip, AVDDVCSEL for laser
	2, 3, 4, 6, 12	To ground	Ground for chip, ground for laser
	9	To NRF Chip I2C SDA	SDA, I2C slave data transfer, no internal pull up
	10	To NRF chip I2C SCL	SCL, I2C slave clock, no internal pull up
	5	AVDD	Signals wake up period (no HW standby)
	7	Unconnected	No slave driven interrupts allowed

Table 14. ToF Pin Connections

Part	Pin	Connection	Function
IMU	1	Microprocessor GPIO pin 4	Responsible for defining last bit of I2C address (dual functionality unused)
	5, 8	Regulated LDO voltage	VDD and VDD_io. Same voltage range so we will apply the same voltage
	6, 7	GND	Ground for chip
	4, 9	GND	INT 1 and INT 2 are forced to ground. Slave functionality so no interrupts to master
	2, 3	VDDIO	Unused due to I2C slave configuration. Might change depending on internal pull up (need to ask about this)
	12	GND	Value of 0 sets I2C comm enabled and disable SPI
	13	To NRF chip I2C	SCL, I2C slave clock, no internal pull up

		SCL	
	14	To NRF chip I2C SDA	SDA, I2C slave data transfer, no internal pull up

Table 15. IMU Pin Connections

Part	Pin	Connection	Function
Regulator 1	1	Battery High	Vin
	2	GND	GND
	3	Processor GPIO Pin 14	Enable
	4	N/C	
	5	Vout	VDD of Microprocessor, Push Button

Table 16. Regulator 1 Pin Connections

Part	Pin	Connection	Function
Regulator 2	1	Battery High	Vin
	2	GND	GND
	3	Processor GPIO Pin 15	Enable
	4	N/C	
	5	Vout	VM/VCC of Motor, Motor Driver, Tof, IMU

Table 17. Regulator 2 Pin Connections

Part	Pin	Connection	Function
Charging Chip	1	VBUS	Takes in voltage from the MicroUSB for charging
	2	GND	Defines fast charge current setting by resistance

	3	GND	VSS GND for the chip
	Thermal Pad (under)	GND	Must be connected to GND at all times
	8	Pin 10 after resistor and diode	Low(FET on) indicates Charging, high (FET off) indicates no charging / charging done
	5	Pin 10 after resistor and diode	Low (FET on) indicates input voltage is above UVLO and OUT voltage
	10	Battery, voltage regulator Vin	Power output. If plugged in, load powered by charging connection
	9	Host GPIO	If disabled, prevents timed charging
	4	GND	Determines termination current level (i.e. max current allowable)

Table 18. Charging Pin Connections

Part	Pin	Function	Connection
Motor Drive	1	VM (Motor Supply)	Bypass to GND w/ 0.1 μF ceramic capacitor
	2	AOUT1 (Bridge A output 1)	Connect to motor A
	3	AOUT2 (Bridge A output 2)	
	4	BOUT1 (Bridge B output 1)	Connect to motor B
	5	BOUT2 (Bridge B output 2)	
	6	GND	Connect to ground
	7	BENBL (Bridge B ENABLE input)	Internal pulldown resistor, supply with PWM signal
	8	BPHASE (Bridge B PHASE input)	Internal pulldown resistor, Sets direction of H-bridge B
	9	AENBL (Bridge A ENABLE input)	Internal pulldown resistor, supplied with PWM signal
	10	APHASE (Bridge A PHASE input)	Internal pulldown resistor, Sets direction of H-bridge A

	11	MODE (Input mode select)	Internal pulldown resistor, Set to Logic high for PH/EN mode
	12	VCC (Device Supply)	Bypass to GND w/ 0.1 μF ceramic capacitor

Table 19. Motor Drive Pin Connections

Part	Pin	Function	Connection
LED	1	Cathode end of LED	PWM output of Microprocessor
	2, 3, 4	Anode end of LED	GND

Table 20. LED Pin Connections

Appendix B Labor Distribution

Table X Labor Costs

Name	Hourly Rate	Hours	Total	Total x 2.5
Andrew Betbadal	$30	80	$2400	$6000
Nicholas Radler	$30	80	$2400	$6000
Katelyn Schoedl	$30	80	$2400	$6000
Machine Shop	$30	3	$90	$225
Petronics	$40	8	$320	$2460
Total	**$160**	**251**	**$7610**	**$20685**

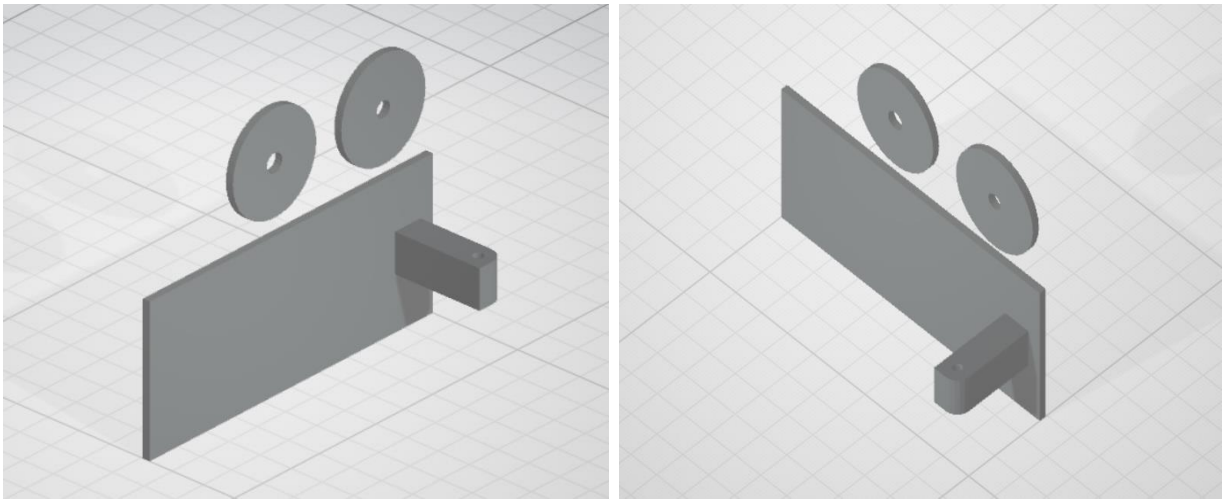
Appendix C Schedule

Week	Tasks and Responsibilities
2/18	Complete Design Document - All Set meeting with Petronics for following week - All Sign up for Design Document Review - All
2/25	Design Document Review - All Meet with Petronics to go over design and order parts - All
3/4	Teamwork Evaluation - All Soldering Assignment - All Set meeting with Petronics for following week - All Breadboard circuit and work on firmware development - Nick Update PCB design - Katelyn Finalize control algorithm design - Andrew
3/11	Meet with Petronics and set meeting for after break- All Order PCB - Katelyn Test firmware - Andrew Work on control algorithm implementation - Nick
3/18	Spring Break
3/25	Individual Progress Reports - All Meet with Petronics and set meeting for next week - All Finalize and order PCB - Katelyn Finalize Firmware - Andrew Finalize control algorithm implementation - Nick
4/1	Work on Final Paper - All Create mechanical body - Katelyn Meet with Petronics to combine our work with their physical/mechanical systems - All Test product and make any final adjustments - All
4/8	Work on Final Paper - All
4/15	Mock Demonstration - All Demonstration Sign Up - All Mock Presentation Sign Up - All
4/22	Demonstration - All Mock Presentation - All Presentation Sign Up - All
4/29	Presentation - All Poster Session - All Turn in Final Paper - All Turn in Lab Notebook - All

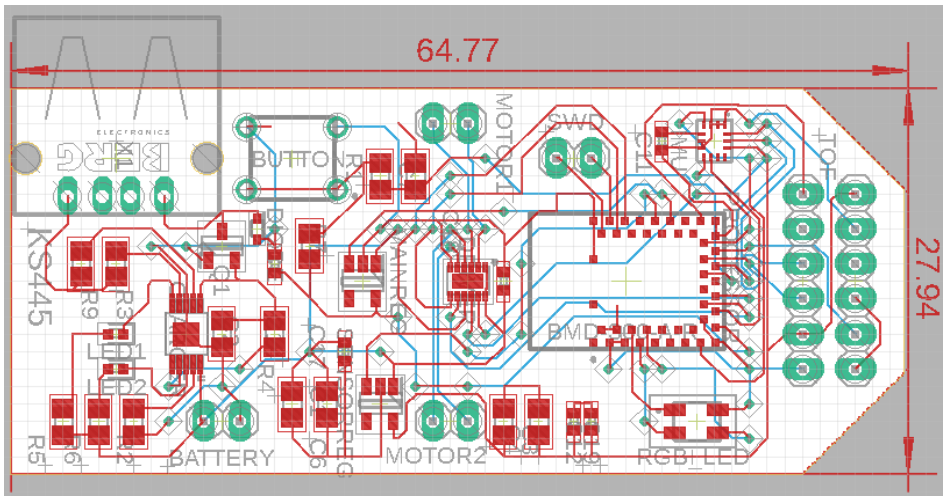
	Teamwork Evaluation - All Lab Checkout - All Award Ceremony - All

Table 12. Spring 2019 Design Schedule

Appendix D Physical Design

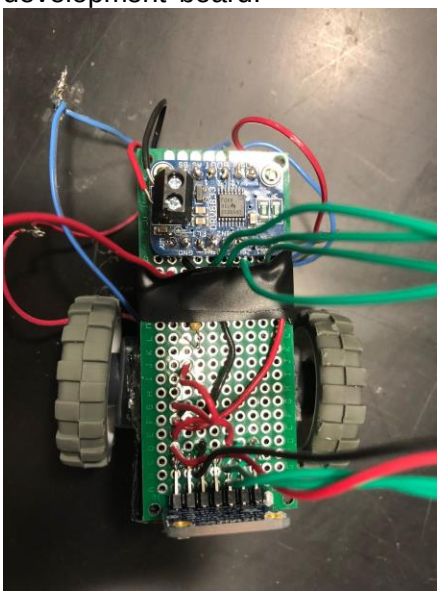


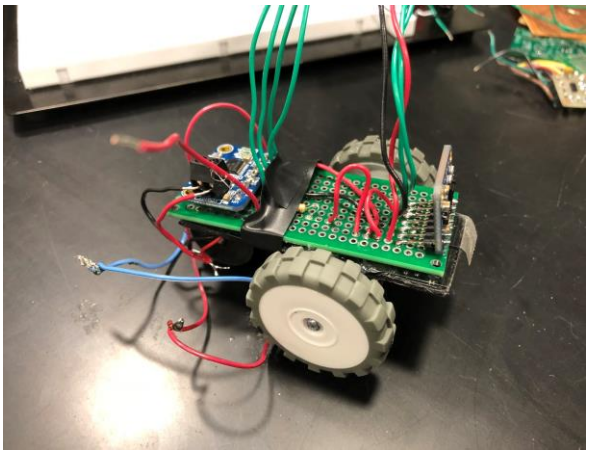
Pictured is the Inventor model of PCB mount in *.stl* file format. The approximate measurements of this part (including the "wheels") were 50mm x 70mm x 21mm, before shrinkage. The "wheels" do not rotate but merely were attached to drag along with the motorized wheels, similar to the "ears" on the head of Mousr. The flat part of the rectangular mount was printed flat on the 3D print surface to save time. This was designed in Inventor and printed in Makerlab at UIUC.

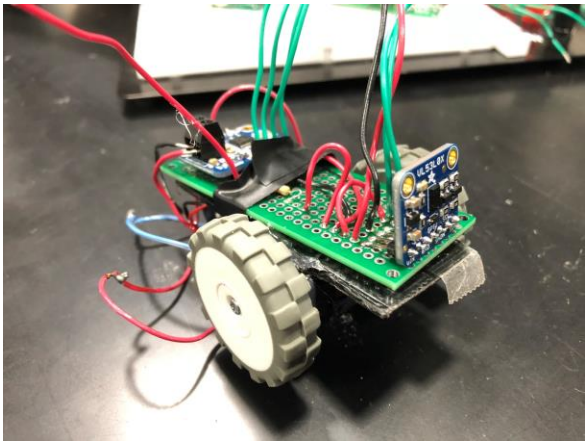


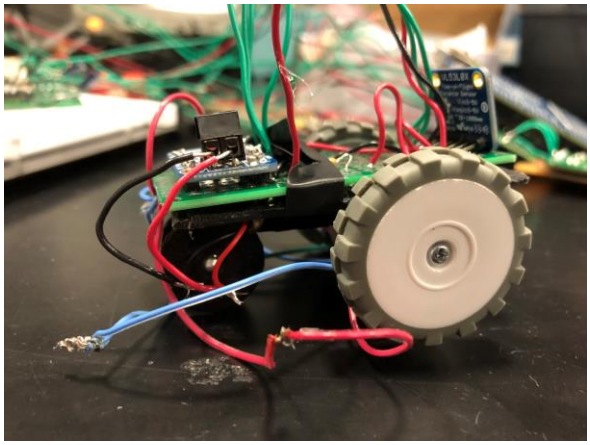
Top and side views of Micromousr showing motor driver development board (left side of the perf board, back end of the robot) and ToF development board (right side of the perf board, front part

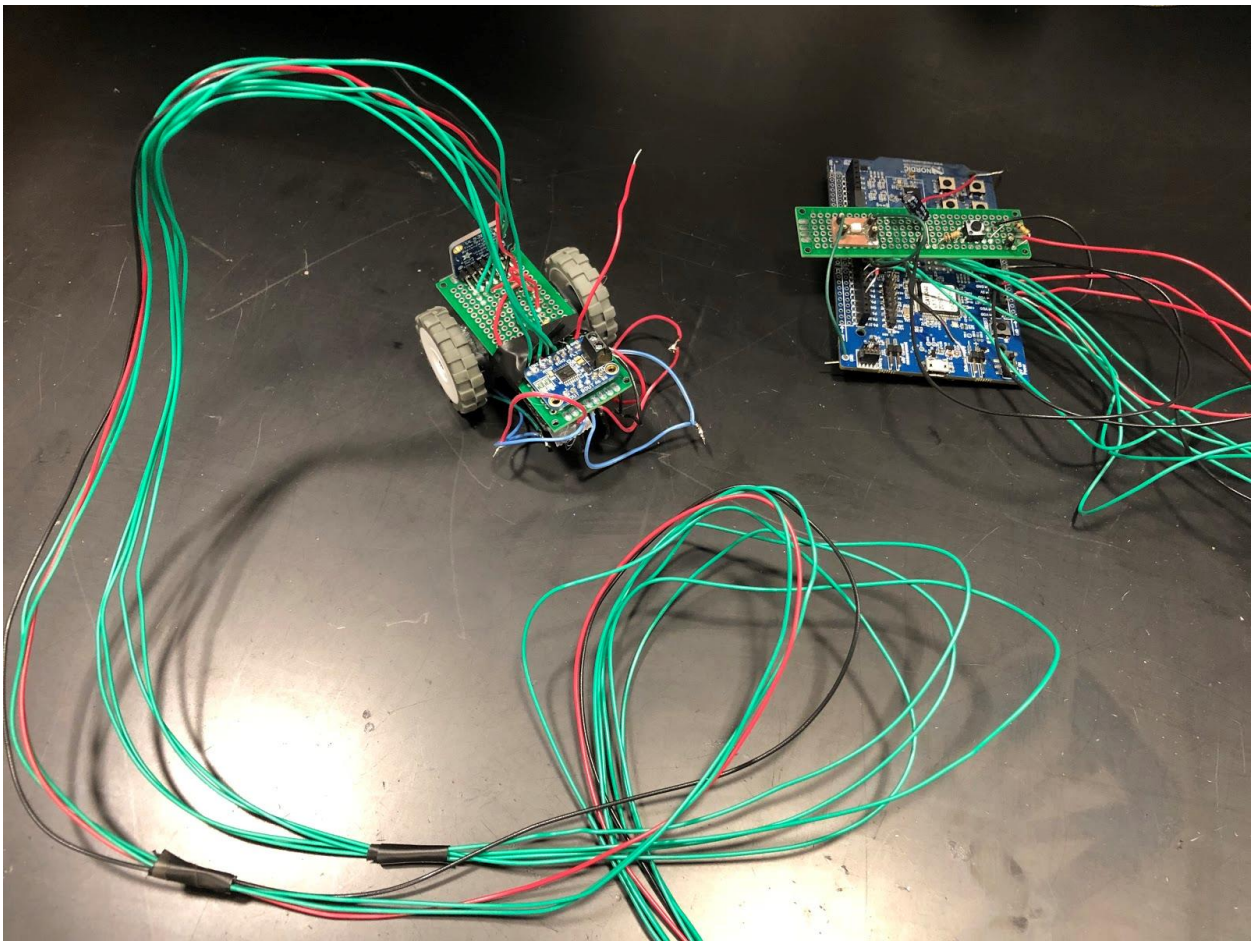
of the robot) on custom 3D printed chassis. Motor driver powers two motors built into the gearbox below this board. GPIO connections (green wires) attach to microprocessor development board.











Appendix E Requirements and Verification Table

Requirement	Verification	Verification status (Y or N)
1. **Microprocessor Requirements** a. Transmit enable/disable signals to power planes b. Drive PWM signal from decision algorithm and user input via Bluetooth c. Switch between low energy and normal operation d. Drive I2C clock at 64MHz	1. **Microprocessor Verifications** a. Drive on/off signal using push button and confirm enable / disable signal sent appropriately using Arduino. b. Connect M4 to Arduino and supply digital input from Arduino to M4. Drive PWM at different duty cycles based on digital input from GPIO on M4. Write test software to connect and receive bluetooth data. Using M4 in debugging mode, print bluetooth input to computer monitor with varying user input. c. Transmit low-power signal to GPIO using signal generator. Transmit reset signal to GPIO and read GPIO pin out. Confirm reset confirmation latency less than 1500 ms. d. Set up I2C communication with debugging Arduino in slave mode. Connect oscilloscope to SCL line and confirm I2C drive at 64 MHz.	Y Y Y Y
2. **Optical (ToF) Requirements** a. Maintain obstacle recognition at a distance of > .7 meters before hitting max value. b. Sensor is mounted in a location on board allowing a full 25 degree field of view of the environment. c. Switch between low energy and normal operation	2. **Optical (ToF) Verifications** a. Attach debugging Arduino to SDA line and confirm non-max values from at least 0 - .7 meters before dropout (i.e. before non-recognition). Place object at max range of ToF in high power mode (2 meters). Confirm object detection accuracy within + .2% (i.e. recognition noise occurs within + 2% of samples). b. Measure 25 degree range from sensor to ensure functional recognition test region from Step 1 spans entire field of vision. Confirm detection as above at the center and the edge of this 25 degree range. c. Leave object in low power state then drive high power signal via activated power plane. Confirm object detection begins after .26us (predetermined start setup time).	Y Y Y

3. **Inertial (IMU) Requirements** a. The IMU must be able to communicate with the M4 processor via I2C interface. b. The IMU must accurately detect when the MicroMousr is lifted, tilted, or rotated. c. Transitions from low-power to normal operation with minimal boot up waste (1500 ms).	3. **Inertial (IMU) Verifications** a. Connect SDA/SCL pin to debugging Arduino. Confirm slave address of 1101010 transmitted properly. b. I2C connected to debugging Arduino, set base orientation at ground on ECEB SDL floor. c. Leave object in low power state then drive high power signal via activated power plane. Confirm object detection begins after .25us (predetermined start setup time).	N N N
4. **Voltage Regulation Requirements** a. Output voltage in the range 3.3 with max current up to 0.9A per chip in normal power. b. Begin power distribution within 100 ms of enable.	4. **Voltage Regulation Verifications** a. Connect regulator to power supply providing 4.2 V. Connect regulator to a resistive load such that the voltage output reads 3.3V and ensure the current through the load is below 0.9A using multimeter at output pin. b. Set up with 4.2 V supply as above but keep the power off. Connect an oscilloscope across the resistive load, and put it single trigger mode. Turn on the power then use the oscilloscope to verify that the time for the voltage rise is less than 100ms.	Y Y

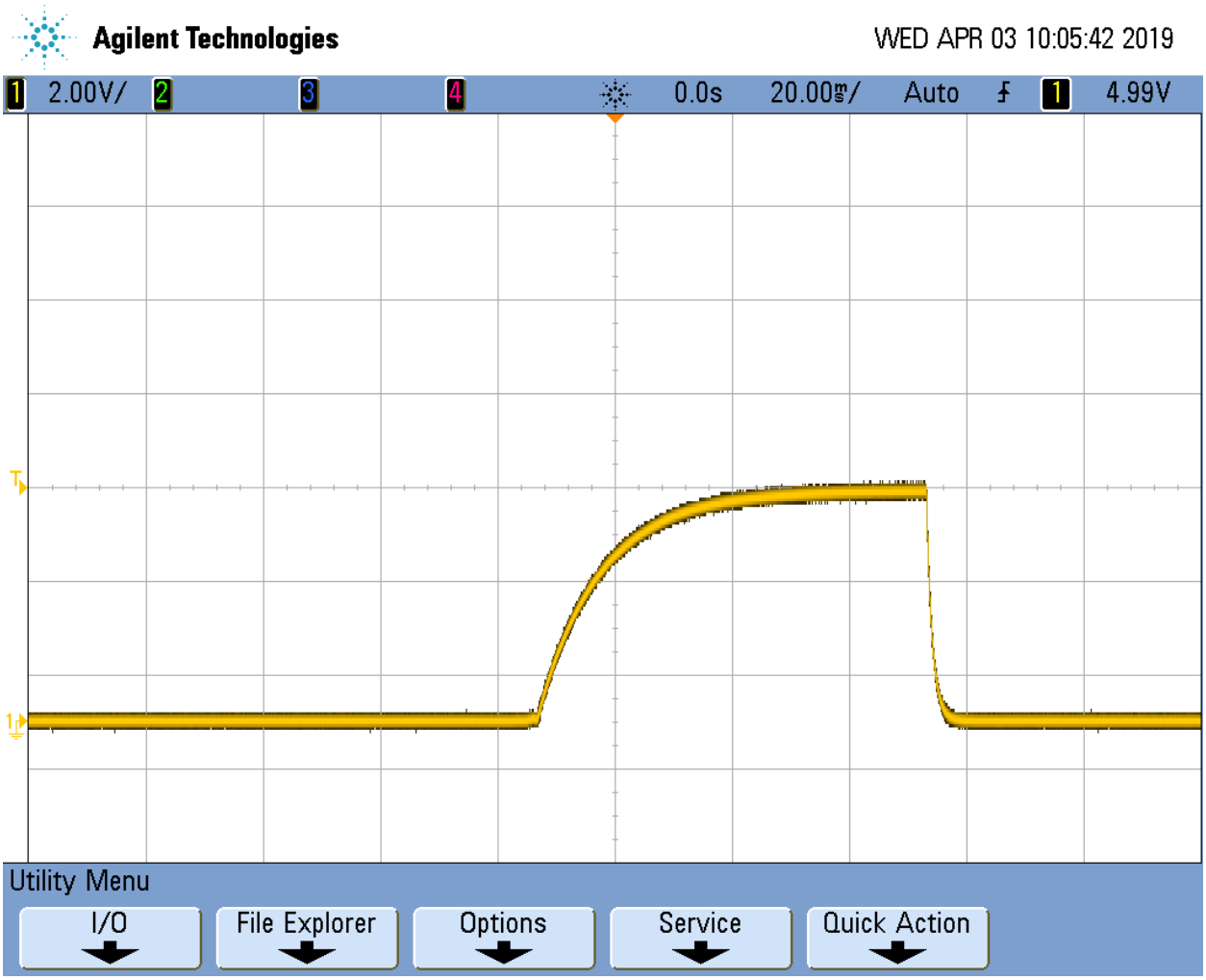
5. **Battery Requirements** a. At full charge, support system in ON mode for at least 30 minutes. b. At full charge, support system in OFF mode for at least 1.5 months.	5. **Battery Verifications** a. Connect Li-Ion battery to charging circuit and charge fully. Connect to voltage load and pull max load (Table 21). Report time until discharge, and repeat. b. Charge as above, connect to voltage load and pull off load (Table 21). Apply load for 5 hours, and test check remaining charge using multimeter.	N/A
6. **Charging Circuit Requirements** a. Be able to charge 4.2 V battery	6. **Charging Circuit Verifications** a. Discharge a 4.2 V li-ion battery. Connect battery to Output of BQ24090DGQR, which is connected to a voltage supply of 5 V. Once output of pin 8 on the BQ24090DGQR goes high, use voltmeter to make sure the battery outputs a voltage of 4.2 V +/- 1%	N
7. **Motor Drive Requirements** a. Must be able to to drive motors in any combination of forward rotation, backward rotation, or no rotation.	7. **Motor Drive Verifications** a. Since the two H-Bridges on the DRV8835DSSR are independent, we can test both at the same time. i. Apply a DC voltage of 3 V at pins 1, 12 and connect pin 6 to ground. ii. Set pin 11 to logical high and apply logical low to pins 8, 10. iii. Have a resistor in series with an ammeter in between pins 2 and 3, and another resistor in series in between pins 4 and 5. iv. Apply PWM signals at 50% to pins 7 and 9 and check the ammeters to make sure current is positive. v. Apply logical high to pins 8, 10 and check that the current flowing through the ammeters is negative.	Y
8. **Push Button Requirements** a. Push button circuit has zero voltage bounce.	8. **Push Button Verifications** a. Attach debounce circuit to power supply and oscilloscope. Confirm no voltage bounce.	Y

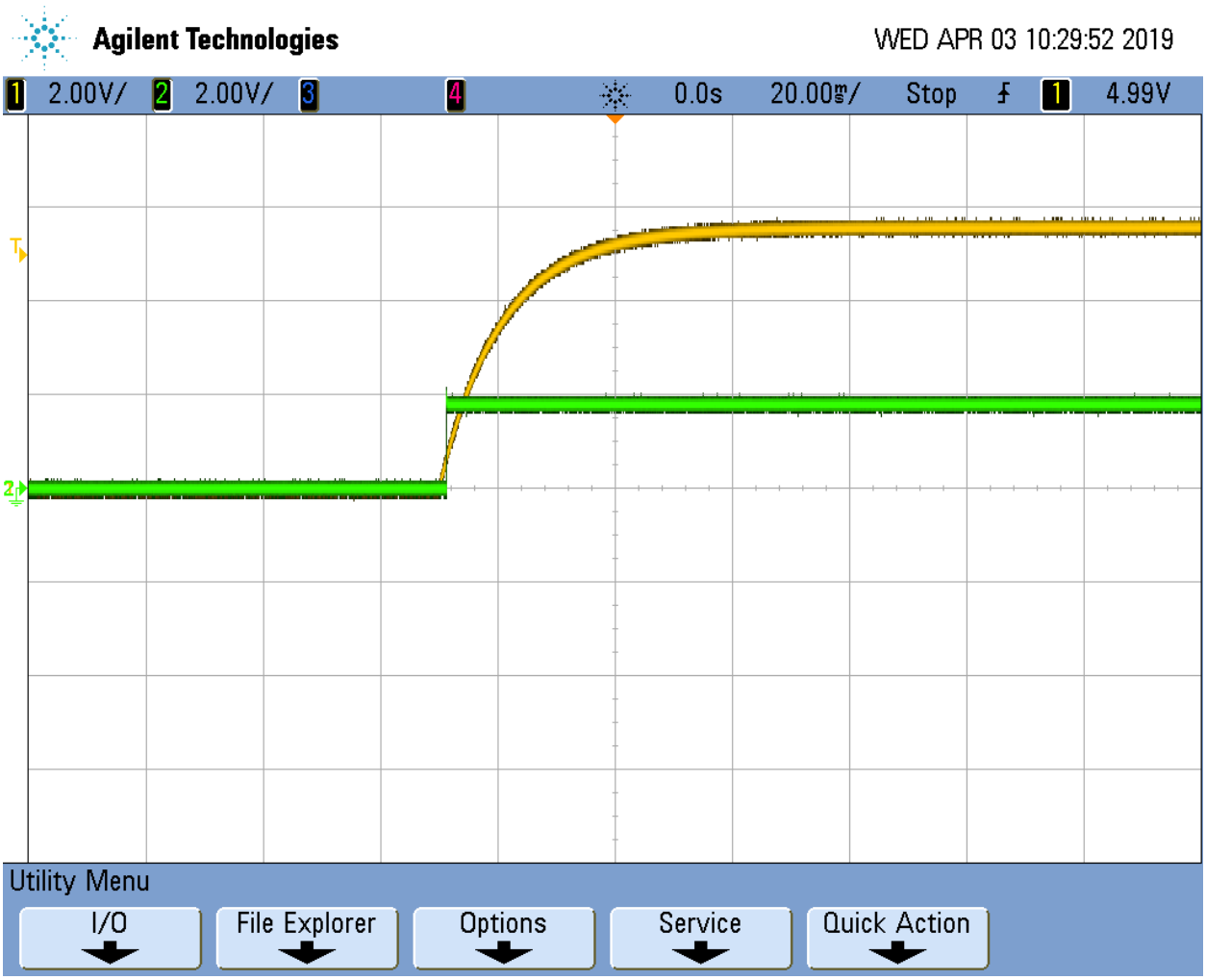
9. **LED Requirements** a. Display R G and B colors from Microprocessor.	9. **LED Verifications** a. Attach cathode and anode connections to positive and negative terminal of signal generator. Drive PWM signal at 15%, 50%, 85%.	Y

Appendix F Power Consumption Table

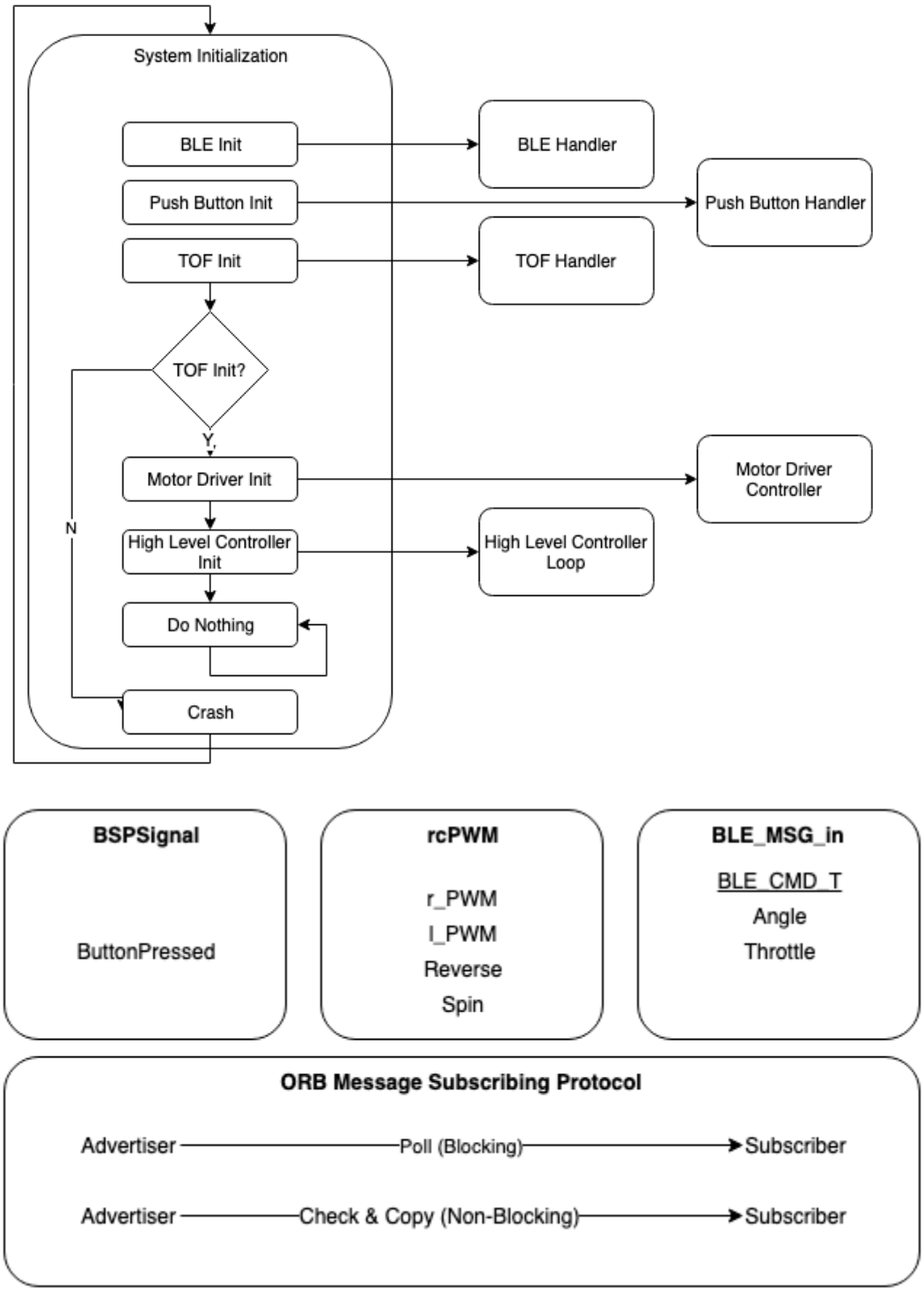
Load	Voltage (Typical)	Voltage Range	Current (Active)	Current (OFF / SLEEP)	Power Consumption (Active)	Power Consumption (OFF / SLEEP)
Processor	3.0	1.7-3.9	5.4mA	.7µA	16.2mW	.0021mW
ToF	+2.8V	2.6V-3.5V	19mA	0A	20mW	0W
IMU	1.8V	1.71V-3.6V	.45mA	0A	0.81mW	0W
Motor Driver	[Reference **Calculation 2**]				769.2mW	0W
LED	2V	2V-5V	20mA	0A	75mW	0W
Push Button	3V	1V-24V	15µA	15µA	.045mW	.045mW
Voltage Regulator (x2)	4.2V	2.1V-6V	40µA	0A	.168mW	0W
Total Power Consumption					821.186mW	0.05799 mW

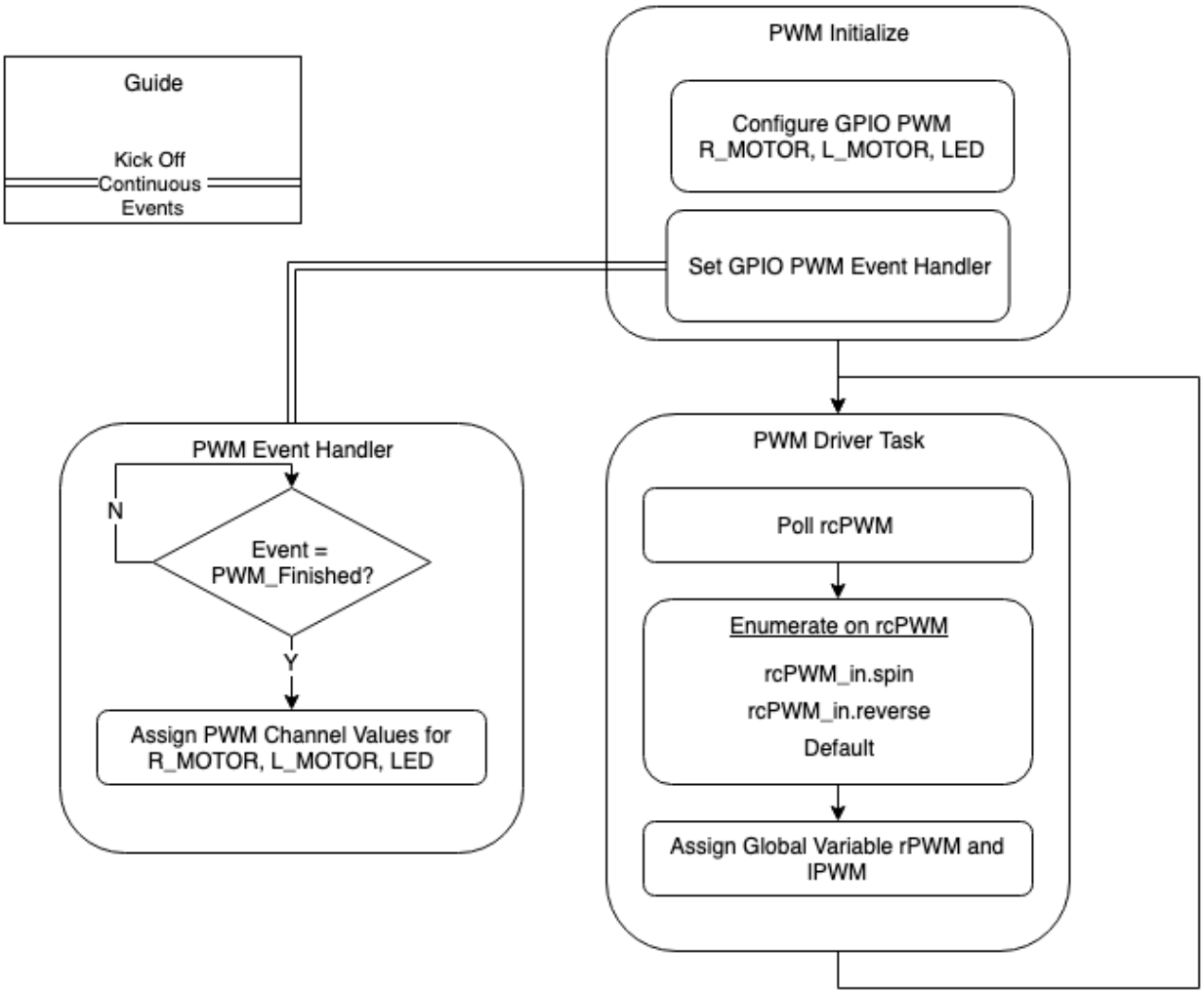
Appendix G Oscilloscope Outputs

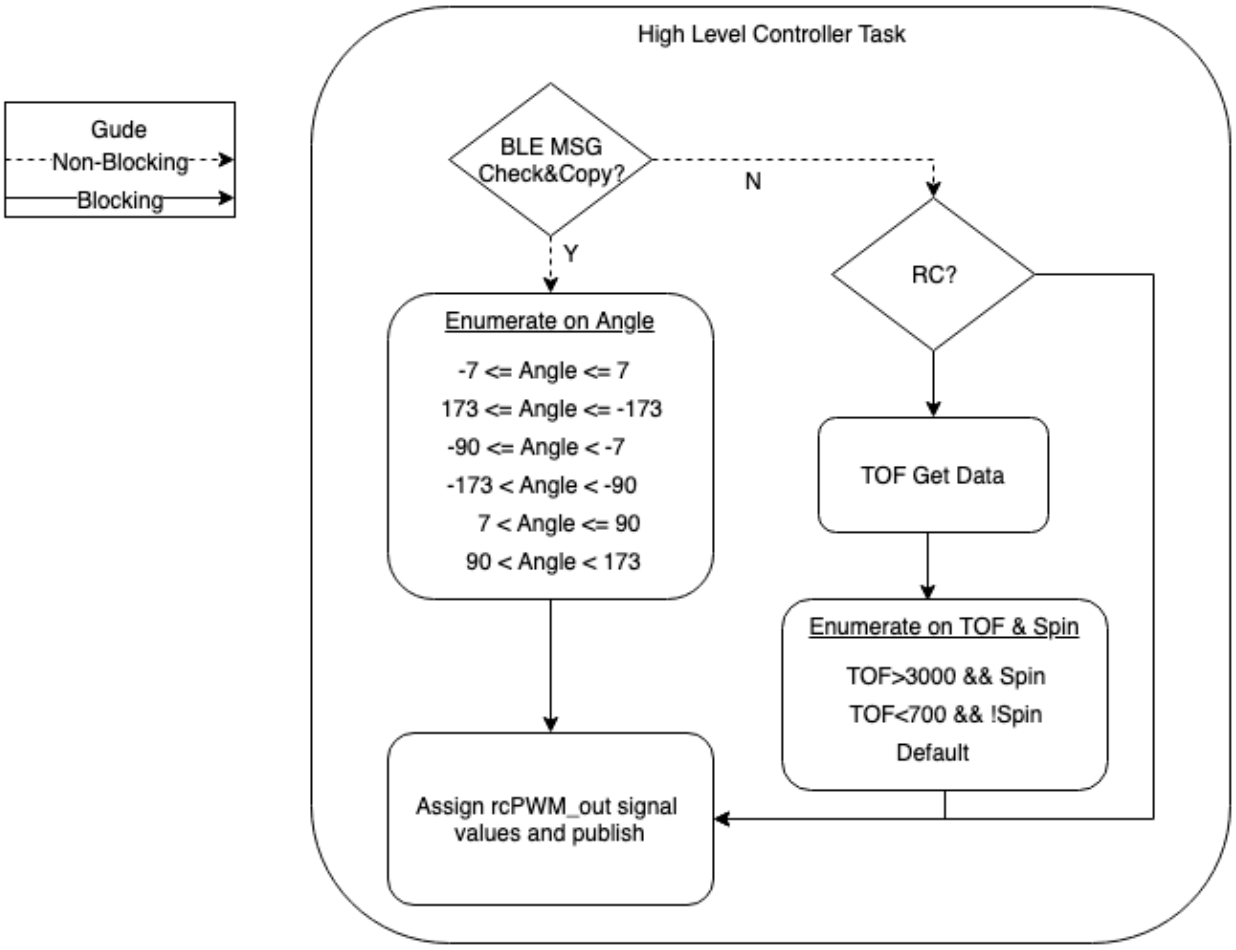


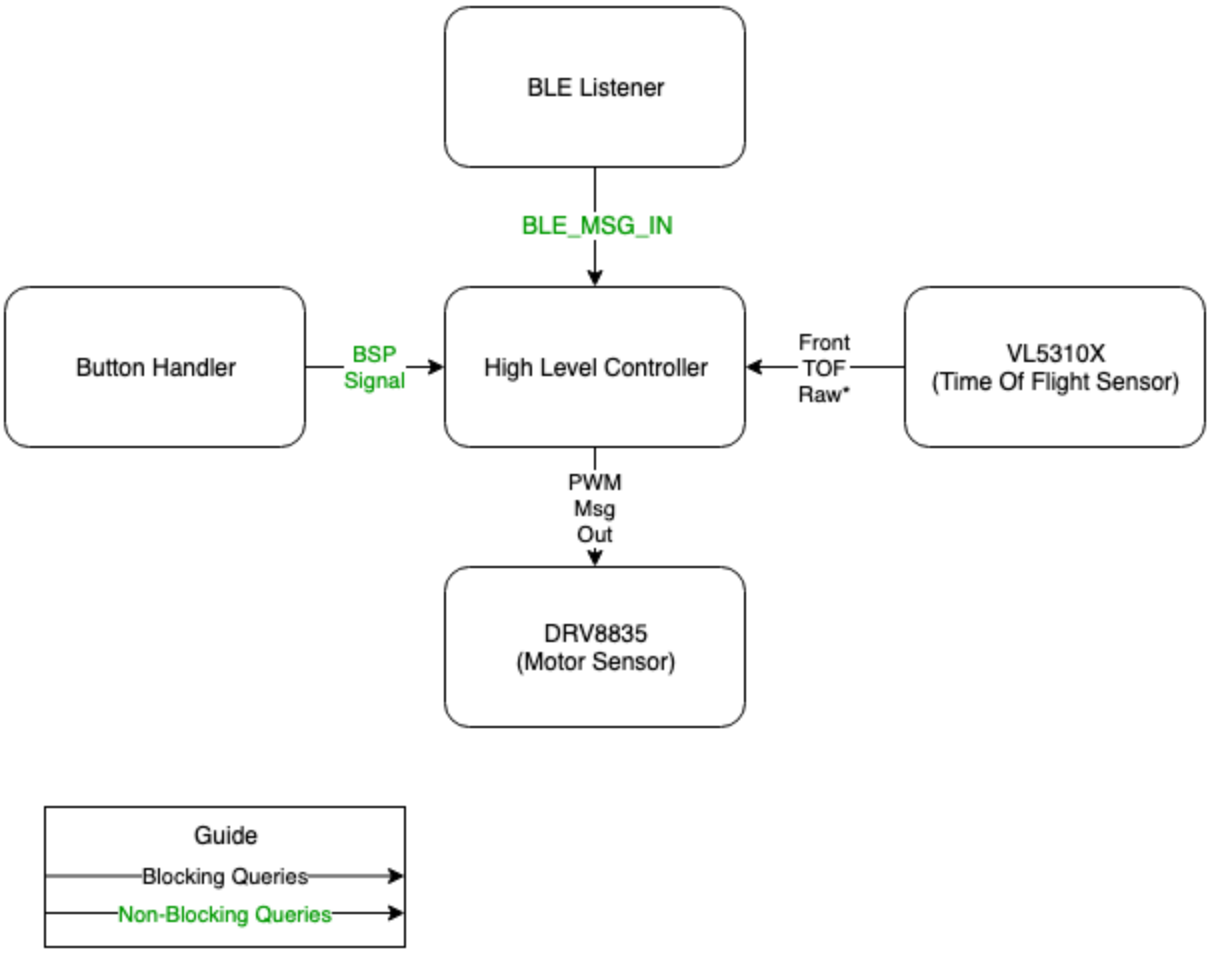

Agilent Technologies
WED APR 03 10:29:52 2019
1 2.00V/ 2 2.00V/ 3 4 0.0s 20.00ms/ Stop 1 4.99V
Utility Menu
I/O
File Explorer
Options
Service
Quick Action

Appendix H Software Flowcharts




Guide
Kick Off
Continuous
Events
PWM Initialize
Configure GPIO PWM
R_MOTOR, L_MOTOR, LED
Set GPIO PWM Event Handler
PWM Event Handler
N
Event =
PWM_Finished?
Y
Assign PWM Channel Values for
R_MOTOR, L_MOTOR, LED
PWM Driver Task
Poll rcPWM
Enumerate on rcPWM
rcPWM_in.spin
rcPWM_in.reverse
Default
Assign Global Variable rPWM and
lPWM


Gude
Non-Blocking
Blocking
High Level Controller Task
BLE MSG
Check&Copy?
N
Y
RC?
Enumerate on Angle
-7 <= Angle <= 7
173 <= Angle <= -173
-90 <= Angle < -7
-173 < Angle < -90
7 < Angle <= 90
90 < Angle < 173
TOF Get Data
Enumerate on TOF & Spin
TOF>3000 && Spin
TOF<700 && !Spin
Default
Assign rcPWM_out signal
values and publish



Appendix I Costs

Table X Parts Costs

Device	Amt.	Value	Cost	Bulk Cost (500)	Product	Footprint	Product Link
Battery	1	3.7 V Nominal	$3.95	$3.95	ASR00003	-	
Push Button	1		$0.11	$0.08	FSM4JH	FSM4JH	
Charging	1		$1.30	$0.76	BQ24090DGQR		
Diode	2	5.1 V Zener	$0.25	$0.07	MM5Z5V1T1G	SODFL1608X70N	
Motor Driver	1		$1.67	$0.99	DRV8835DSSR	SON50P200X300X80-13N	

IMU	1		$4.09	$2.56	LSM6DSLTR	LGA-14(2.5X3X0.86)	
Indicator LED	2				SML-P12VTT86R		
RGB LED	1		$0.89	$0.38	QBLP677-RGB	QBLP677RGBHIGHBRIGHT	
TOF	1		$5.97	$3.07	VL53L0CXV0DH/1	-	
Voltage Regulator	2		$0.47	$0.20	NCP699SN33T1G	SOT95P275X110-5N	
Motor	2				-	-	
Processor Module	1		$11.31	$7.54	BMD-300-A-R	BMD-300	
C1	1	1uF	$0.03	$0.03	VJ0805Y105KXXTW1BC	C0805	
C2	1	1uF	$0.03	$0.03	VJ0805Y105KXXTW1BC	C0805	
C3	1	100n	$0.25	$0.07	08053C104M4T4A	C0805	
C4	1	4.7u				C0805	
C6	1	2.2u				C0805	
C7	1	.1 uF				C0805	
C11	1	100nF	$0.12	$0.03	TMK105B7104KV-FR	C0402	
C12	1	100nF	$0.12	$0.03	TMK105B7104KV-FR	C0402	
C14	1	1uF				C0402	
C17	1	1uF				C0402	
C18	1	1uF				C0402	
C19	1	1uF				C0402	
Q1	1	DMG2301LK-7				SOT96P240X110-3N	
R1	1	10k				R0805	

R2	1	10k				R0805	
R3	1	2k				R0805	
R4	1	10k				R0805	
R5	1	1.5k				R0805	
R6	1	1.5k				R0805	
R7	1	100k				R0805	
R9	1	715				R0805	
Total Cost			**$31.27**	**$20.05**			