Modular Electric Skateboard

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree in Bachelor of Science

By

Nicholas Fajardo

Phillip Konyeaso

Zane Weissman

Date: 4/26/19

Project Advisors:

Professor Susan Jarvis

Professor Craig Putnam

Abstract

The Modular Electric Skateboard (MESB) allows the user to attach components tailored to specific demands. The MESB is equipped with two motors controlled by a handheld RC controller and is capable of achieving speeds as high as 20 mph. The deck can be disassembled and reassembled by hand to swap out major mechanical components of the board. The board's electrical system includes a housing for mounting hot-swappable accessories, which can communicate to a database. A front end application was implemented that reads and writes to the aforementioned database, allowing visualization of aggregate data.

Contents

List of Figures

List of Tables

1 Background

There are many means of transportation available in cities; some are more time consuming than others. Though cars are nearly ubiquitous in America, bicycles, buses, and trains remain popular alternatives. Recently, electric skateboards have filled a particular niche for some travelers. They have the advantages of decent speed (often above 20 mph) and good portability but are often held back by low range, making them ideal for short rides and trips that include trains or buses as well. Electric skateboards and similar devices can take many forms; the market is still new, but most are recognizable as a skateboard or longboard with a motor and drive train, an enclosure for batteries and electronics, and a handheld speed controller which may be wired or wireless.

1.1 Skateboard Parts and Terminology

Skateboarding has been around since the mid 1900s, and has changed drastically since its inception. Figures 1a and 1b show the different parts that make a modern complete skateboard, and the list below defines each part in detail.

- Deck - The surface on which the rider stands and to which the trucks and griptape are attached. It is essential for any deck to be sturdy, but also have some amount of flex for comfort. Decks vary in size and material, but they are typically wooden and a few feet long.

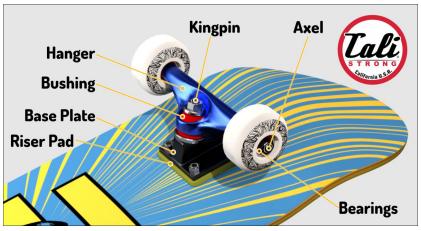


Figure 1a: Skateboard Parts

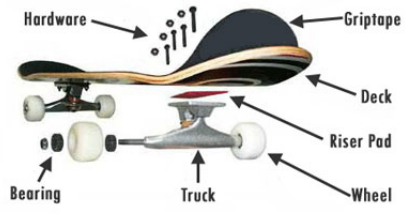


Figure 1b: Skateboard Parts

  - Griptape - An adhesive surface for the top of the deck. One side is a very strong tape; the other side is a rough, sandpaper-like surface designed for high friction with the soles of the rider's shoes. Griptape comes in different grits (measure of "roughness") and colors, but it is most often black.
  - Nose - Front of the board. A typical skateboard has a nose that is angled up.
  - Tail - Rear of the board. A typical skateboard has a tail that is angled up.

- Trucks - The assemblies that connect the wheels to the deck.
  - Hanger - The main, T-shaped portion of the truck.
  - Riser Pad - A pad used to add distance between the wheel and the deck. Often made of a material conducive to dampening vibration, such as rubber or cork.
  - Baseplate - The metal piece of the truck that is bolted to the deck. The rest of the truck sits on a threaded rod that extends

downwards from the baseplate.

  - Bushings - Spacers between the kingpin, the hanger, and the base plate. Bushings can be replaced to adjust the force required to turn while riding; i.e., a bushing that is harder to compress will require the user to lean to a direction more in order to perform the same degree of turn as a softer to compress bushing. Often made of some sort of rubber polymer.
  - Kingpin - A nut, usually a locknut, used to cap the hanger and bushing assembly. It can be loosened and tightened to fine tune the hardness of its bushing.
  - Axle - The rod used to attach the wheels, typically integral to the hanger. Axle nuts screw onto the axle's threaded ends to hold the wheels in place.
  - Bearings - Assemblies that sit between the axles and wheels to ensure smooth rotation.
  - Wheels - Skateboard wheels vary in size, shape, color, and material (though typically they are some type of relatively hard plastic); most are interchangeable without changing any other parts. Size and material can have significant effects on shock absorption and the frictional forces between the wheels and the road. A new set of wheels can drastically change the way that a skateboard handles.

- Hardware - All the nuts and bolts needed to attach the trucks to the

deck.

- Longboard - Type of skateboard more suited for traveling long distances. Longboard decks are elongated and often somewhat flexible or bouncy; longboard trucks are typically looser (i.e. requiring less force to turn).

1.1.1 Board Material

When looking at skateboards/longboards they are usually made out of a variety of materials. They are often separated into three main categories based on main material used for composition. The list below explains the differences between each board type.

- Canadian Maple veneer - this material is the industry standard for most skateboards/longboards. It is slightly flexible, with medium resistance to pressure, and very durable[8]. Most people are used to riding this type of board. (This is the material we will be using for the project.)

- Bamboo - this material is not used for skateboards usually, and can be found in longboards. It is very flexible, with high resistance to pressure, however as a result it does not hold shape while pressure is applied (large bend). Some people are accustomed to this board type.

- Baltic Birch plywood - this material is a happy medium between the prior two materials, but it is more likely to splinter.

1.2 Board Shape

Board shape is an integral part of any skateboarding system, affecting multiple things during use such as the response to rough terrain, the portability of the device, and the turning behavior. Board shape itself can be split into multiple categories. The two categories that this project focuses on is the deck shape and the concavity of the board. Taking both into consideration allowed the group to pick a strong candidate board to begin with for testing.

1.2.1 Deck Shape

Deck shape is important, because when longboarding the deck shape largely determines whether the board is meant for cruising or downhill. The two standard types are shown in the images below.

Figure 2a: Cruiser

Figure 2b: Downhill

- Cruisers - This is the most popular longboard style, and the deck size allows for greater weight distribution. It is reliable, however they usually aren't stiff and have shorter wheelbases compared to downhill decks. Their main purpose is riding on flat ground comfortably and quickly.

- Downhill - As the name implies, this board type is for riding downhill quickly. It is important that deck doesn't flex much, and has a wide

wheelbase otherwise the rider could lose control easily and become injured. It is usually advised to wear protective gear while riding these boards. When reaching high speeds (60-80 mph), going downhill is very dangerous, and crashes at that speed can be lethal.

1.2.2 Concavity

Below are a types of concavity on the board. For the most part it is up to the user, and how he/she prefers the deck to feel under his/her feet.

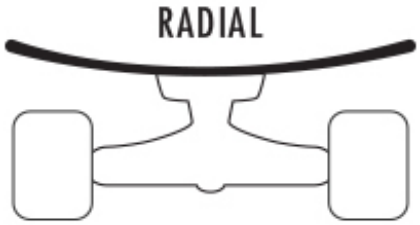


Figure 3a: Radial: most common design, widely used. Conventional wisdom holds that the deeper the curve, the better the feel of the board.

Figure 3b: Progressive: similar to radial decks, but some say it provides a more locked-in feel with more secure footing.

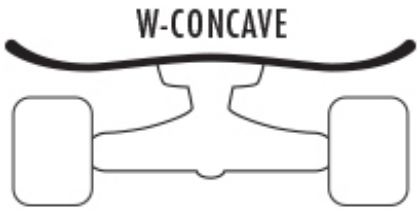


Figure 3c: W-Concave: easier shifting of weight from foot to heel, highly precise and responsive deck.

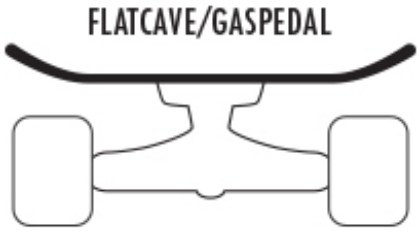


Figure 3d: Flatcave: provides a more relaxed ride, but because of curved sides can apply some sudden shifts in weight.

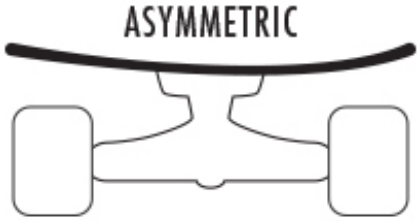


Figure 3e: Asymmetric: this board is good for riders who heavily rely on their heels to shift weight.

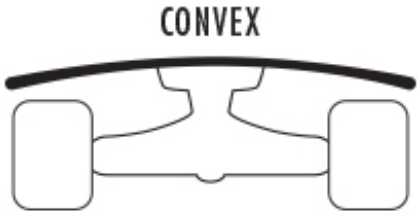


Figure 3f: Convex: provides natural foot placement, but not as easy to control or as safe.

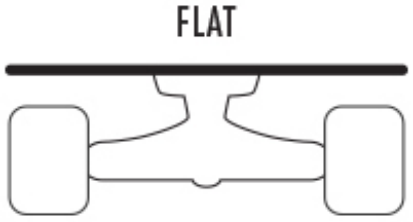


Figure 3g: Flat: allows for plenty of space for feet, and makes possible interesting tricks like boardwalking, which cannot be done easily on other decks.

1.3 Electric Skateboards

An electric skateboard (ESB) is a skateboard with one or more electric motors and drive trains attached to one or more wheels. The battery that powers the motor(s) is generally fixed to the bottom of the deck in a housing that protects it and accompanying electronics from damage. The motor is often controlled with a handheld controller which may be wireless or wired. Alternatively, some ESBs include pressure or strain sensors in the deck and use the rider's balance to control the motor. ESBs are usually longboards, as they tend to provide smoother rides at the higher speeds often achieved with the electric motor.

One example of a relatively standard ESB is the Boosted Board brand "Dual+ XR" pictured in Figure 4, which happens to also be one of the most popular off the shelf ESBs on the market today. It retails for $1,599

Figure 4: Boosted Board Dual+ XR

and claims a top speed of 22 mph and a range of 14 miles with its 2000W

motor and regenerative braking. Some more examples of popular ESBs can be found in Table 1.

Table 1: Popular ESBs on the Market

Board Name	Top Speed (mph)	Battery Range (miles)	Weight (lbs)	Price (Dollars)
Inboard M1	22	7	14.5	1,000
Boosted Plus/Stealth	22/24	14	17	1,399/1.599
Blink Qu4tro	23	22	24	1,699
Boosted Mini S/X	18/20	7/14	15/16.8	749/999
Onewheel +/+XR	19	5-7/12-18	25	1,399/1,799

2 System Overview

The modular electric skateboard (MESB) is an electric skateboard with a deck comprised of multiple sections which can be interchanged and replaced, a handheld remote controller for controlling the electric motor, a rail/data bus system attached to the underside of the deck which is capable of housing a number of accessories, and a small computer system which manages those accessories via a simple, but flexible, API.

2.1 Mechanical Systems

2.1.1 Board and Linkages

The most obvious modular feature of the MESB is that the deck itself can be quickly disassembled and reassembled to replace the entire tail and tip. Since the trucks, wheels, and motors are all mounted on the tail and tip, this allows the user to quickly make major changes to the way that the board drives.

The board has two V-shaped cuts separating the tips from the center piece as shown in Figure 7. Because of this feature, supporting linkages were needed in order to connect the pieces and improve stability. The final result we decided on was a double-V link, shown in 6 that faces the opposite direction of the cut. An example of an alternate shape that was considered, a single-V link can be seen in Figure 5 in section 3.4.4.

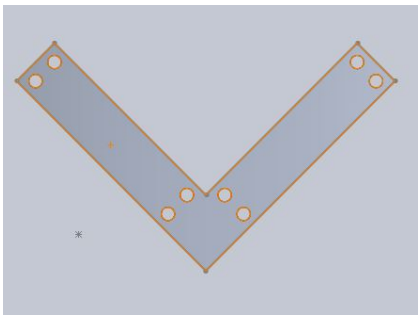


Figure 5: Single-V

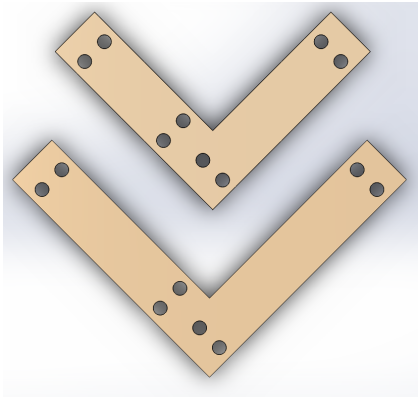


Figure 6: Double-V

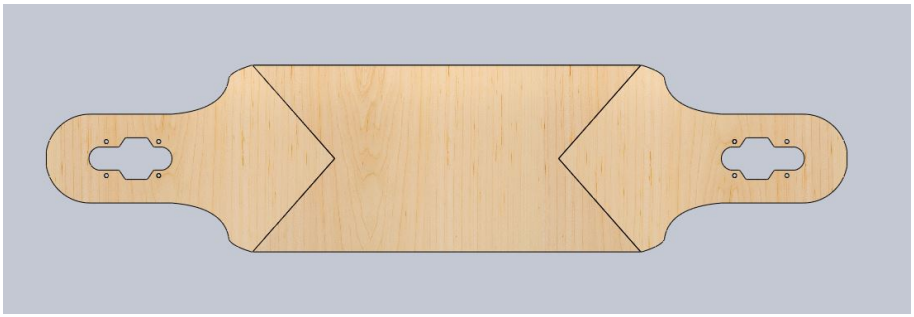


Figure 7: Board with V-shaped cuts at tail and tip

2.2 Shock Absorption

To prevent splintering of the board at the nose and tail end of the board, shock absorbing material has been added in between the junctions of the board. Shock absorption provides comfort to the user and durability to the board; furthermore, this shock absorbing layer can aid in preventing water damage by keeping water out of the junction. After thorough research and comparison, sorbothane, shown in Figure 8, was determined to be one of the best options for what the board required. It is highly flexible, elastic, water resistant, and compressible, and it is extremely affordable.

Figure 8: Sorbothane

2.3 Control Systems and Control Processing

The control system is comprised of three components: a standard RC transmitter, a RC receiver (both shown in Figure 9) and a Turnigy SK8-ESC V4.12 motor controller (Figure 10).

Figure 9: Standard RC transmitter and receiver

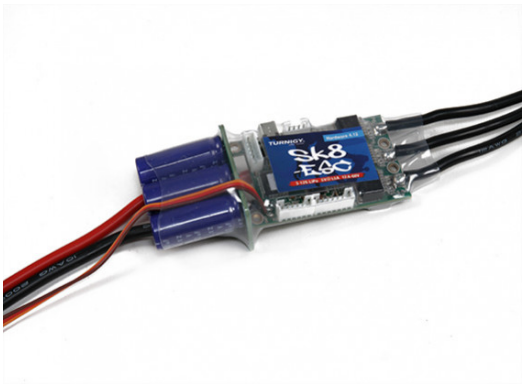


Figure 10: Turnigy SK8-ESC V4.12

The RC transmitter has some basic control options like maximum throttle settings and trim, but overall the transmitter-receiver system can more or less be treated as a black box. The input is the displacement of the throttle trigger on the transmitter and the output is a pulse position modulation (PPM) signal output from the receiver which encodes the displacement of the trigger based on the throttle settings set on the back panel of the transmitter.

The PPM signal can be fed directly to the SK8-ESC. The SK8-ESC runs re-configurable firmware called VESC which can be set to use the incoming PPM signal as the input to its control scheme. Inside the controller there is a transfer function that handles converting the ppm signal to a desired set-point speed, utilizing a simple linear relationship.

2.3.1 Control Loop Design

When designing the response for the MESB system, multiple factors were proposed that could affect the overall experience of the user. After consideration of the user's ride, pace, and speed desires, a proportional-integral-derivative (PID) controlled loop with a static gain was chosen due to the flexibility of the closed control loop.

Most electronic skateboard systems utilize an open control loop, however this is a rudimentary technique that should not be employed when sensors could be used to measure the speed of the system and the torque applied by a given motor; this additional data can be used to improve the precision of the control loop. Since access is available to such sensors, the closed control

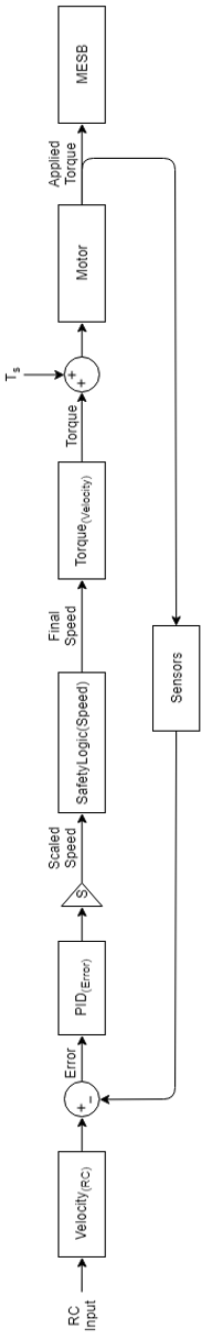


Figure 11: Closed PID control loop with static gain

loop Figure 11) is the better option to achieve the desired response discussed earlier.

Specifically, this loop will derive the desired (set point) speed from the RC input using a linear relationship, and then will subtract the error (difference between the set point and actual speed) which will be fed into a proportional-integral-derivative (PID) control. These PID parameters can be arranged to a specific user's needs, which affects the lag and speed parameters of the system under operation. After the error is fed into the PID loop, it is then scaled by a gain in order to create a lag for high errors. After the speed is calculated, it is then converted to torque using an instantaneous constant power relation from the motor

$$\tau = \frac{Power}{Speed} \tag{1}$$

relating the desired speed error with the physical properties of the board under zero incline. This is then added to a feed forward torque that calculates the needed torque to keep the board static under a specific incline and weight. This total torque is then converted into a current which can be directly fed into the drive system.

The controls are implemented with VESC, an open source motor controller firmware designed for use in electric skateboards and similar vehicles, running on Turnigy SK8-ESC motor controllers. This control loop was intended to be implemented on the MSP430 series microcontroller, which

performs initial filtering and failsafes that VESC does not support, and the motor controller, which would perform the final stages of the control loop and regulate the actual current to the motors. However, it was determined that there was significant lag when passing PWM signals through the MSP430 and attempts to identify the source of this lag or reduce it were unsuccessful, so only the SK8-ESC motor controllers were used for the control scheme. In a production version of the MESB, it would be ideal to have all of the controls done on one microprocessor running one piece of software; a fork of VESC, which is open source, could likely provide all of the features the control scheme needs.

2.4 Electrical and Power Systems

2.4.1 High Voltage Power System

The high voltage power system is exclusively for powering the motors, which require significantly higher voltage and current than any of the other electronics on the MESB. It consists of an array of Samsung 25R 18650 2500mAH 20A lithium ion cells, arranged as two parallel series of ten, providing roughly 30V with a capacity of 5000 mAH. The power switch for the main power system is a 60A breaker, providing electromechanical protection against excessive current draw which could be caused by shorts or other electrical issues.

2.4.2 Low Voltage Power

The low voltage power system powers the MSP430F5529 microcontroller, the Rock64 mini-PC, and any electronic accessories with 5V DC. The MSP430 is on a Launchpad development board which includes a 5V to 3.3V regulator, allowing it to use this voltage. A regulator circuit was designed (see Section 3.4.2) to provide sufficient power to the MSP430 and Rock64 as well as up to 2A at 5V to each accessory. However, this circuit was never assembled or installed due to time constraints and the fact that the SK8-ESC motor controllers each have a 5V 1.5A regulator for up to 3A together, providing enough power to test all low power electronics.

2.5 Computer and Accessory System

Accessories are held in a dust and water resistant housing, mounted to the underside of the deck, which also contains other electronic systems. Figure 12 shows the housing mounted on the deck, while Figure 13 shows an exploded view of the housing itself.

Accessories slot into the base of the housing, and then a horizontal support structure is bolted on top of them to secure them during use. On the end opposite the beam is the accessory's Mini-DIN plug (for more detail, see section 3.3.3), used for power and data, which sockets into a port mounted to a PCB mounted to the housing.

The accessories are controlled by a Rock64 mini-PC[4] running a minimal

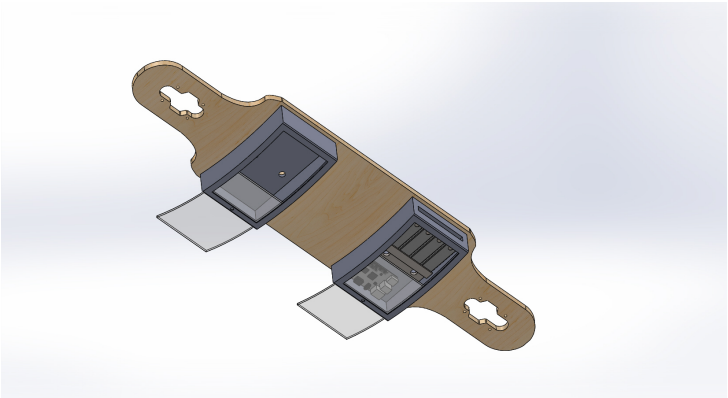


Figure 12: Electronics housings

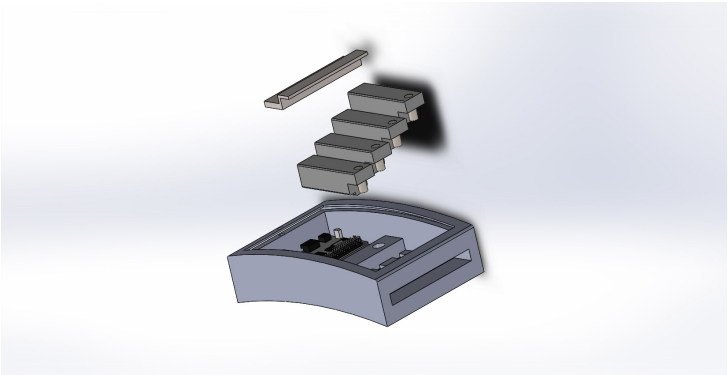


Figure 13: Exploded view of accessory system in housing

Linux installation. The Rock64 acts as a server to the client accessories using a simple application programming interface (API) implemented with I^2C as the physical layer. The Rock64 also hosts a web server intended to be accessed by a smartphone or other portable computer. The web server provides a simple graphical interface for managing connected accessories and the data they collect, as shown in Figure14.

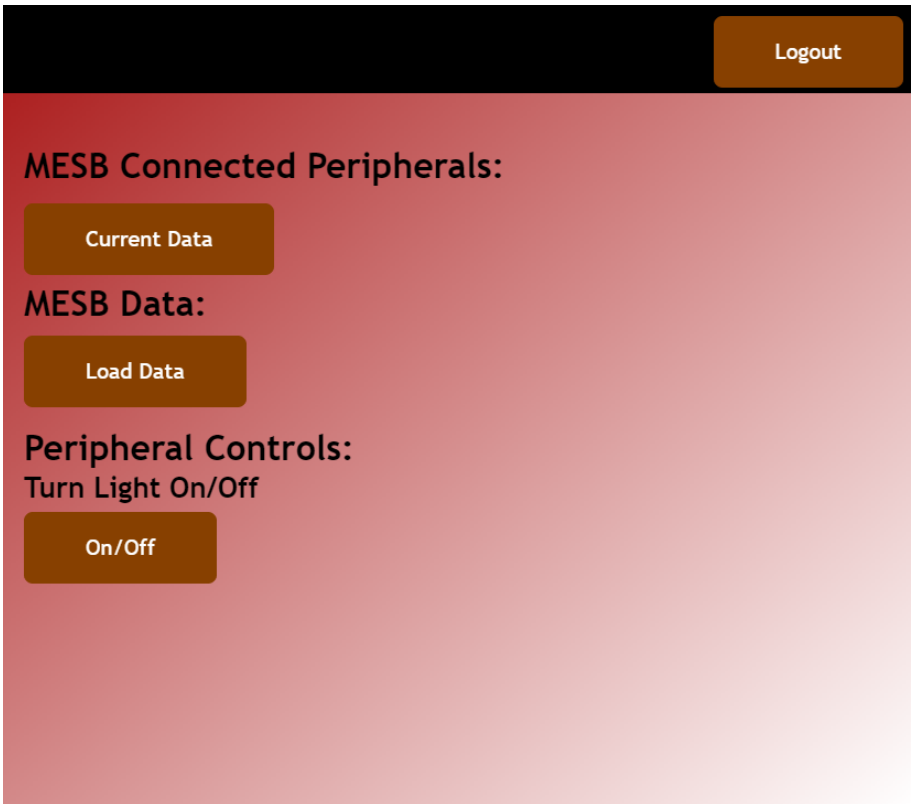


Figure 14: Homepage of website

3 Design Analysis

3.1 Mechanical System Analysis

Representing the various elements of the Modular Electric Skateboard system requires a study on a variety of sections of the physical properties of the board. This includes calculating for the torque required to move the skateboard given its position, calculating the wheel contributions to a system's speed when a rotation around the board axis is induced, and using said calculations to derive the motors and gear ratios to use for the project.

3.1.1 General Torque Calculations

The system, in general, has a variety of physical properties that will not change during use.

Table 2: Physical System Properties

Board Length	39”
Board Width	9.06”
Truck Width	8.04”
High Friction Wheel Diameter	2.625”
Low Friction Wheel Diameter	2.48”

It is also important to note that the following assumptions are made during this analysis:

- Wheel is inducing point contact on surface

- Rider and system mass are joined together, represented as a center of mass somewhere on the system
- There is no slippage
- Since we assume point contact, there will be no kinetic friction during movement

Using these properties, and a simple free body diagram of the system, we can derive a general equation to map the motor torque and the acceleration on the given system, show in Equation 3.

$$\theta = Incline, a_x = Acceleration, R = WheelRadius, M = SystemMass \tag{2}$$

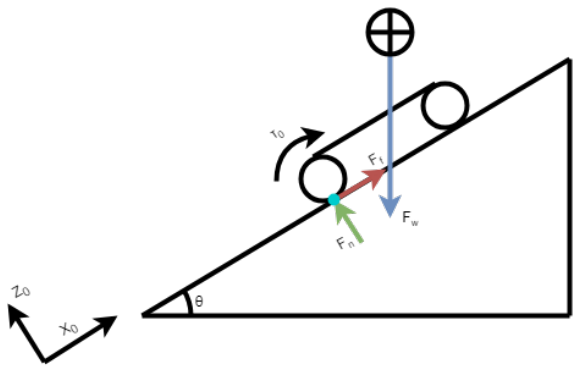


Figure 15: Free Body Diagram of System

$$
\begin{aligned}
\sum F_{x_0} &= ma_x = F_f - \tau_0 * R - F_w sin(\theta) \\
\sum F_{x_0} &= ma_x = F_n \mu_s - \tau_0 * R - F_w sin(\theta) \\
\sum F_{x_0} &= ma_x = F_w cos(\theta) \mu_s - \tau_0 * R - F_w sin(\theta) \\
\tau_0 &= \frac{a_x - gcos(\theta)\mu_s + gsin(\theta)}{-R\frac{1}{m}}
\end{aligned}
\tag{3}
$$

The torque that is required has a relation to the acceleration of the system in general at any given moment. We can convert this in terms of output motor torque:

$$\tau_0 = \tau_{motor} \mu_t T$$

$$\mu_t = transmission\ efficiency$$

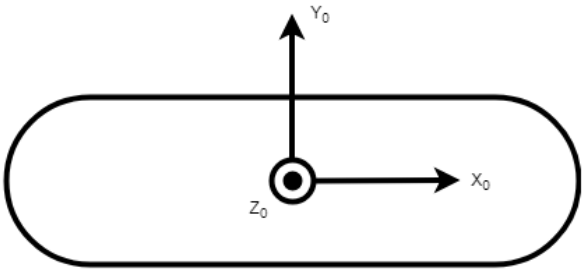
$$T = transmission\ ratio$$

$$\tau_{motor} = \frac{a_x - gcos(\theta)\mu_s + gsin(\theta)}{-\mu_t T R\frac{1}{m}} \tag{4}$$

Using 4 we can map the motor torque to the acceleration of the body, which will help us calculate the necessary torque the motor needs to apply to the system given some sensor data and user input on requested speed.

3.1.2 Wheel Contribution Calculations

Equation 4 found a mapping between the board acceleration and motor torque, given some environment and system parameters. However, this case will only be accurate whenever the board torque is applied when a rotation around the board axis (X_0) is not present.



In cases when the user induces a turn on the axis, each wheel will contribute differently to the total board velocity, which is something that needs to be mathematically modeled in order to adjust speeds when inducing a rotation γ around the X_0 axis[1]. Please note that the following statements are valid and exploited in this derivation of wheel velocities:

- Any γ around X_0 induces a rotation angle around the front and rear trucks, and the angle on each truck is equal in magnitude, but opposite in direction.
- All wheels have the same properties
- We know the physical properties of the board, as specified in Table 2

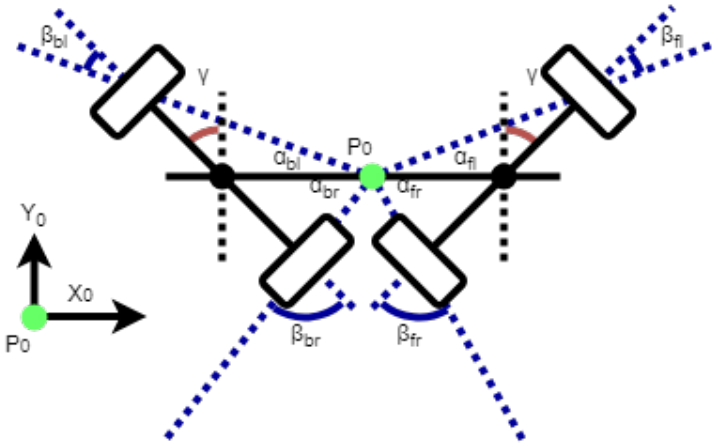


Figure 16: Position state by a induced rotation(γ) on the board axis

Assuming γ is not zero, we can use the following relations to derive each wheel contribution:

$$L = \textit{distance from point } P \textit{ to center of wheel}$$

$$\vec{V} = \textit{Velocity of total body (relative to point } P\textit{, in body frame)}$$

$$\textit{Rolling Constraint} = [sin(\alpha+\beta), -cos(\alpha+\beta), -Lcos(\beta)]\vec{V} - r\dot{\delta} = 0 \quad (5)$$

$$\textit{Sliding Constraint} = [cos(\alpha+\beta), sin(\alpha+\beta), Lsin(\beta)]\vec{V} - r\dot{\delta} = 0 \quad (6)$$

These relations are based off of alpha and beta, which can be found for each wheel using the known parameters

[1] An induced rotation means force applied to some displacement off the x, but still lying in the X_0 and Y_0 plane.

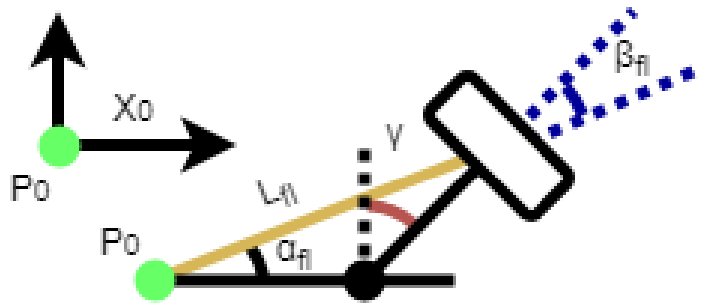


Figure 17: Position of Front Left Wheel with relation to the board axis

$$\alpha_{fl} = cos^{-1}(\frac{\frac{T}{2} - L_{fl}^2 - \frac{d}{2}^2)}{-2L_{fl}\frac{d}{2}}) \tag{7}$$

$$\beta_{fl} = cos^{-1}(\frac{\frac{d}{2} - L_{fl}^2 - \frac{T}{2}^2)}{-2L_{fl}\frac{T}{2}}) \tag{8}$$

Similar calculations can be performed on each wheel, which in the end will yield functions that map our board rotation and wheel speeds to the given board speed. We will summarize the process of combining the wheel velocity equations gathered from the process above and solving for the board velocity $\vec{V}$ inside a function:

$$\vec{V} = mapRotationToSpeed(BoardRotation) = mapRV(\gamma) \tag{9}$$

This will allow us to more accurately produce a body velocity to feed into

a control system to then tell the board what torque to produce.

3.1.3 Motor Torque Calculations

The physical analysis described above produced the required torque that is needed to drive the board. Using this calculated torque, the desired motor configuration and transmission ratio can be calculated. Motor configurations always consists of a DC motor connected to a wheel. This connection is either directly connected [11] or belt driven [10], and can utilize one or two motors.

In order to model this physical system, MATLAB objects and scripts were created representing the system itself. Figure 18 shows the UML diagram of the implemented MATLAB model. The functions and properties all have public access, but properties are set as immutable, which is for safe manipulation during run time. These objects were then utilized in evaluation scripts that comparatively analyzed various motors, drive trains, and motor configurations. It is important to note that we limited our analysis to belt driven systems and a static drive train ratio and efficiency due to financial and time constraints.

The Turnigy Aerodrive SK3 - 6374-192KV Brushless Outrunner Motor seemed to provide the best results. However, as shown in Table 3, this is not the most cost effective solution. After running these calculations, further discussion outlined the need for a motor with a sensor (Which happened to be none of the motors listed below). Some research was conducted to find a motor similar to the best performing motor with a integrated encoder/hall

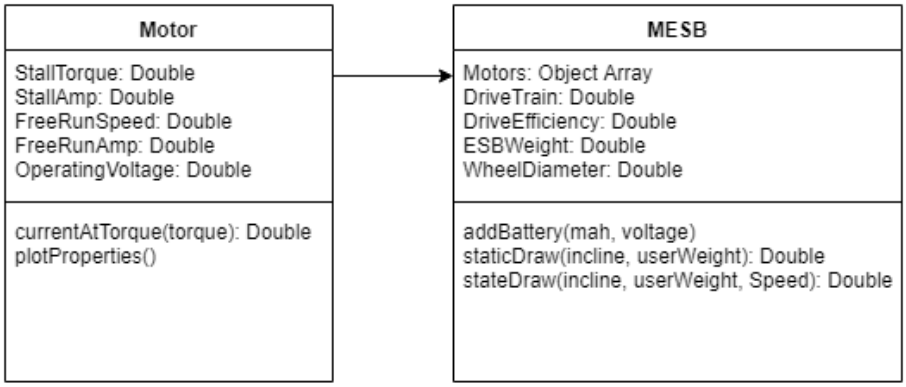


Figure 18: MATLAB System Models (UML)

effect sensor. The result of this research was the selection of a motor that can deliver more torque (3.9 Nm), and has a built in hall effect and temperature sensor, the Turnigy SK8 6354-140KV Sensored Brushless Motor (Figure 19). Two of these motors were ordered for a dual motor drive configuration.

Name	**Max Torque (Nm)**	**Price (Dollars)**
Turnigy Aerodrive 190kv	3.420	90
Turnigy Aerodrive 245kv	2.868	76
ProDrive v2 5060 270kv	2.923	60
Tacon 110 295kv	1.177	60
C5055	1.989	25

Table 3: Motor Evaluation Script Results

Figure 19: Sensored Brushless Motor

3.2 Modular Platform

Initially, Picatinny rails (Figure 20) seemed to be a good way to mount accessory modules onto the board. These rail systems allow any unit with a matching coupling device to be mounted to the rail system, and the device can vary in size without affecting the quality of the coupling system when under load.

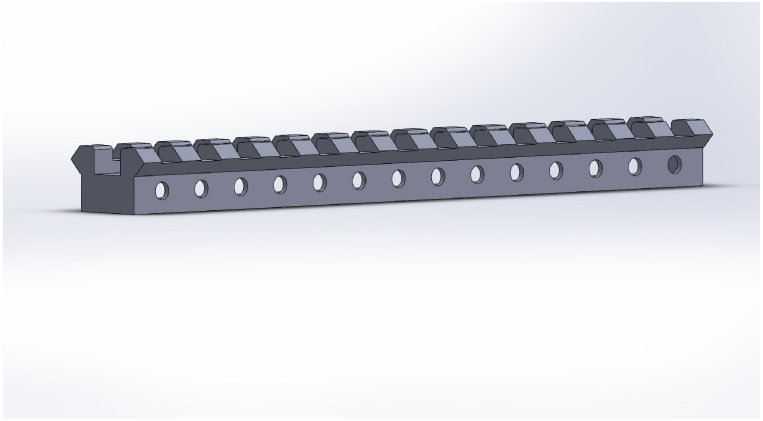


Figure 20: Picatinny Rail

After some thought and discussions with external resources, we started to

conceive of various modular systems that would be better from a manufacturing perspective. Specifically, the objective was to make the components a standard size, and develop some sort of shrouded housing where the modular units could be attached or detached at will.

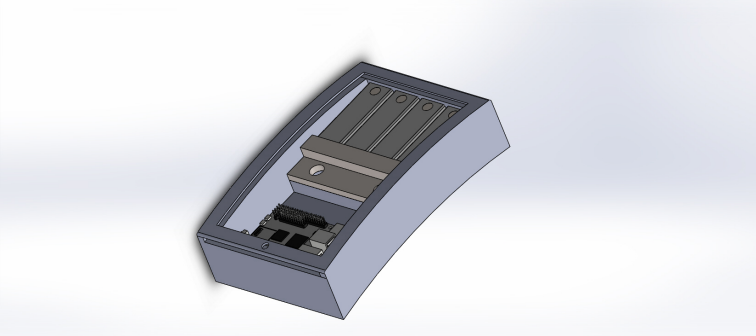


Figure 21: Housing for electronics and accessories

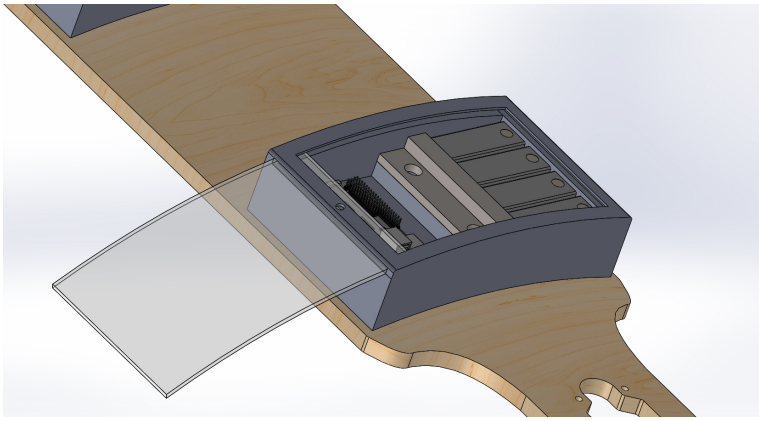


Figure 22: Housing on board

3.3 Trade Studies

This section breaks down in detail several trade-offs of components, protocols, et cetera which had to be considered once the system level design was finished. Each column compares the qualities of a certain attribute of the systems under study, and lists the point value score assigned to each quality. The maximum score in each category is listed in the header of each category with a "/" preceding it; the maximum achievable total score is listed in the header of the leftmost column listing the names of the systems being compared.

3.3.1 Real Time Microcontroller

An essential part of the system is a microcontroller to run the real time control loop. Before choosing a specific processor or development board, several microcontroller (MCU) platforms were compared. At least some variants of each of the microcontrollers in Table 4 would meet the basic requirements of the MCU for the real time control system on the MESB, such as hardware interrupts and an appropriate number of analog to digital converters, timers, etc. Though all of these platforms had their advantages and disadvantages,

Table 4: Real Time Controls Microcontroller Trade Study[2, 3, 5]

Platform	**/20**	**Price**	/5	**Architecture**	/5	**Core Clock**	/2	**Current Draw**	/2	**Familiarity & Ease of Use**	/6
MSP430	**16**	$10-18	4	16 bit RISC	3	8-24 MHz	1	5 mA	2	High	6
STM32 Cortex	**13**	$10-25	4	32 bit ARM Cortex-m	4	32-400MHz	2	100-500 uA/MHz	0	Moderate	3
ATMega328P	**15**	$3-10	5	8 bit AVR RISC	1	up to 20 MHz	1	5 mA	2	High	6

the MSP430 was chosen for its capable and flexible 16-bit architecture, its

low cost and power consumption, and for the developers' familiarity with the ins and outs of its hardware and development tools.

Table 5 compares various MSP430 development boards. The names are truncated to avoid redundancy and save space; the development boards being compared are the MSP-EXP430FR2355, MSP-EXP430FR2433, etc., using the MSP430FR2355, MSP430FR2433, etc. microprocessors, respectively.

The MSP430F5529 offers above-average MSP430 performance for below-

Table 5: MSP430 Trade Study[3]

MCU	**/15**	**Price**	/8	**Core Clock**	/2	**RAM + Storage**	/5	**Current Draw**	/3	**ADCs**	/2
FR2355	**12**	$13	5	24 MHz	2	32KB	2	142 uA/MHz,~1 uA standby	1	1x12b	2
FR2433	**13**	$10	8	16 MHz	1	15.5KB	1	125 uA/MHz; <1 uA standby	2	8x10b	1
FR5994	**11**	$17	1	16 MHz	1	264KB	5	118 uA/MHz; 500 nA standby	2	1x12b	2
FR4133	**10**	$14	4	16 MHz	1	15.5KB	2	126 uA/MHz; <1 uA standby	2	10x10b	1
FR6989	**10**	$18	0	16 MHz	1	128KB	4	100 uA/MHz; 350 nA standby	3	16x12b	2
FR5969	**11**	$16	2	16 MHz	1	62KB	3	100 uA/MHz; .4 uA standby	3	16x12b	2
F5529	**16**	$13	5	25 MHz	2	136KB	4	150 uA/MHz; 2 uA standby	3	1x12b	2

average price. While its higher clock rate may not be necessary, 128KB of flash memory provides a comfortable amount of program storage space.

3.3.2 Accessory Bus Serial Protocol

Table 6 compares several serial protocols which were considered for the main accessory bus: I^2C was chosen for its two-wire communication, ease of use, and more than sufficient speed. Some other serial protocols shared some but not all of these desirable attributes, but I^2C scored nearly perfectly overall.

Table 6: Accessory Bus Serial Protocol Trade Study

Name	/27	Wires, Shared	/3	Add'l Wires per Device	/10	Speed	/4	Ease of Use	/10
I²C	**25**	2	2	0	10	100k-3.4Mbps	3	easy	10
CAN	**17**	2	2	0	10	1Mbps	3	hard	2
SPI	**15**	3	1	1	3	10Mbps	4	medium	7
UART	**15**	0	3	2	0	~10-100kbps	2	easy	10
LIN	**17**	1	3	0	10	20kbps	1	hard	3

3.3.3 Data Connector Plug and Socket

Table 7 compares several types of connectors which were considered for the data/power connections to accessories: Mini-DIN was chosen as the connector standard for the accessory system for its durability, voltage and current capacity, and reasonable price.

Table 7: Data Connector Trade Study

Name	/21	Data Lines	/2	Water Resistance	/2	Locking	/2	Terminal Area	/3	Terminal Length	/3	Price/ Pair	/3	Strength	/2	Current (A)	/2	Voltage (V)	/2
TRRS 3.5mm	**15**	2	2	No	1	No	1	Small	3	Small	3	$2-5	2	Yes	2	1	1	5	1
Mini-DIN	**19**	1-7	2	Yes	2	No	1	Small	3	Small	3	$2-5	2	Yes	2	2-7	2	30	2
Micro USB-B	**11**	3	2	Yes	2	No	1	Small	3	Small	3	$3-7	1	No	1	2	2	5	1
Mini USB-B	**19**	3	2	Yes	2	No	1	Small	3	Small	3	$1	3	Yes	2	1	1	2-5	1
4P4C	**18**	2	2	Yes (IP67)	2	Yes	2	Small	3	Small	3	$1	3	Yes	2	1	1	2-5	1
8P8C	**19**	6	2	Yes (IP67)	2	Yes	2	Small	3	Small	3	$1	3	Yes	2	2	2	2-5	1
DE-9	**18**	7	2	Yes	2	Yes	2	Large	1	Medium	2	$1-2	2	Yes	2	5	2	100	2

3.3.4 Shock Absorption

Table 8 compares several types of shock absorbing materials which were considered to put place in between the joints of the of the tips and midsection: The results show that sorbothane was the best choice for the shock absorbing material of those considered. Its important properties include water resis-

Table 8: Shock Absorbing Material Trade Study

Name	/19	Impact Resistance	/5	Water Resistance	/5	Flexibility	/1	Durability	/5	Cost	/3
Sorbothane	**17**	Very High	5	Very High	5	Yes	1	Very High	5	$30	1
Rubber	**16.5**	High	4	Very High	5	Yes	1	High	4	$15	2.5
Silicone	**15**	Med	3	Very High	5	Yes	1	High	4	$20	2
Springs	**13.5**	Med High	3.5	Med	3	Yes	1	High	4	$20	2
Oil	**9**	Very Low	1	Low	2	No	0	Med	3	$10	3
Foam	**11.5**	Low	2	Very High	5	No	0	Med Low	2.5	$20	2

tance, giving it longevity in rough weather conditions as well as the ability to protect portions of the board against water damage, and strong absorption of impact shock due to its plasticity, providing comfort to the rider and protection to the structural elements of the board when it hits a bump, crack, or curb.

3.4 Testing

3.4.1 Road Roughness and Ride Quality

International standards for road roughness have existed for a few decades. Most commonly used is the international roughness index (IRI), a measure of the motion of a vehicle's suspension as it travels a certain distance over a road, reported in units of meters per kilometer or equivalently, millimeters per m. However, the assumed "standard vehicle" is a car, which of course has very different suspension characteristics from a skateboard. Furthermore, most cities do not regularly measure the IRI of their roads, nor are IRI measuring devices cheap or readily accessible, so measuring the IRI of a road or finding a road with a known IRI would be rather impractical.

Instead, roads in the neighborhood around Worcester Polytechnic Institute were categorized by simple subjective experience of riding a regular, unpowered longboard with the same deck which would then be used for the MESB. This provided, at the very least, a subjective baseline for the comfort and ease of riding on different types of roads, and a basic way to categorize roads based on the features that impact the feeling of the ride. The categories, with example streets, are as follows:

- Freshly paved (Sever St): Top layer of asphalt still intact, covering the stones beneath. Ride is extremely smooth. Almost no vibration felt, no cracks or rough patches to avoid.

- Smooth (Somerset St, sidewalk around Elm Park): Asphalt in great condition, but the top layer has worn away and the small stones in the asphalt are visible. Noticeable vibration from the asphalt. Very few cracks or other issues.

- Medium (Dover, William): Asphalt is significantly rougher but not destroyed. Vibration is more substantial; there are many small cracks that can be felt when riding over them but that are not serious obstacles. Very few or no large cracks or potholes that would need to be avoided.

- Rough (Nicholas Fajardo's driveway): Asphalt is rough to ride on with significant damage in some areas, including some that are impassable. Not possible to traverse at high speeds.

The MESB should be capable of traversing all of these types of terrain, even if it requires some caution, and it should be comparably comfortable to the stock longboard with the same deck.

3.4.2 Low Voltage Power

A potential low voltage power system, for the microcontroller, CPU, and accessories, was designed in OrCAD Capture for simulation in PSpice. Figure 23 shows the circuit.

There was no PSpice model available for the battery, so a model using experimentally determined voltages, resistances, and capacitance's was implemented. The left-hand side of the circuit is an array of different battery models for the same lithium cell at different charges, which are shown on the top left. Pairs of identical models are connected in series because the circuit requires two lithium cells. The right-hand side is the regulator circuit, composed of a simple LM7805 5V voltage regulator and several bypass power transistors to drive additional current. This circuit was ultimately never assembled due to time constraints and since voltage regulators on the motor controllers proved sufficient for testing purposes.

3.4.3 Linkage Analysis

The are many metals that can be used to support structures. So it is important to look at the different strengths of each metal/alloy. When researching it was important to look at certain qualities. The following categories were

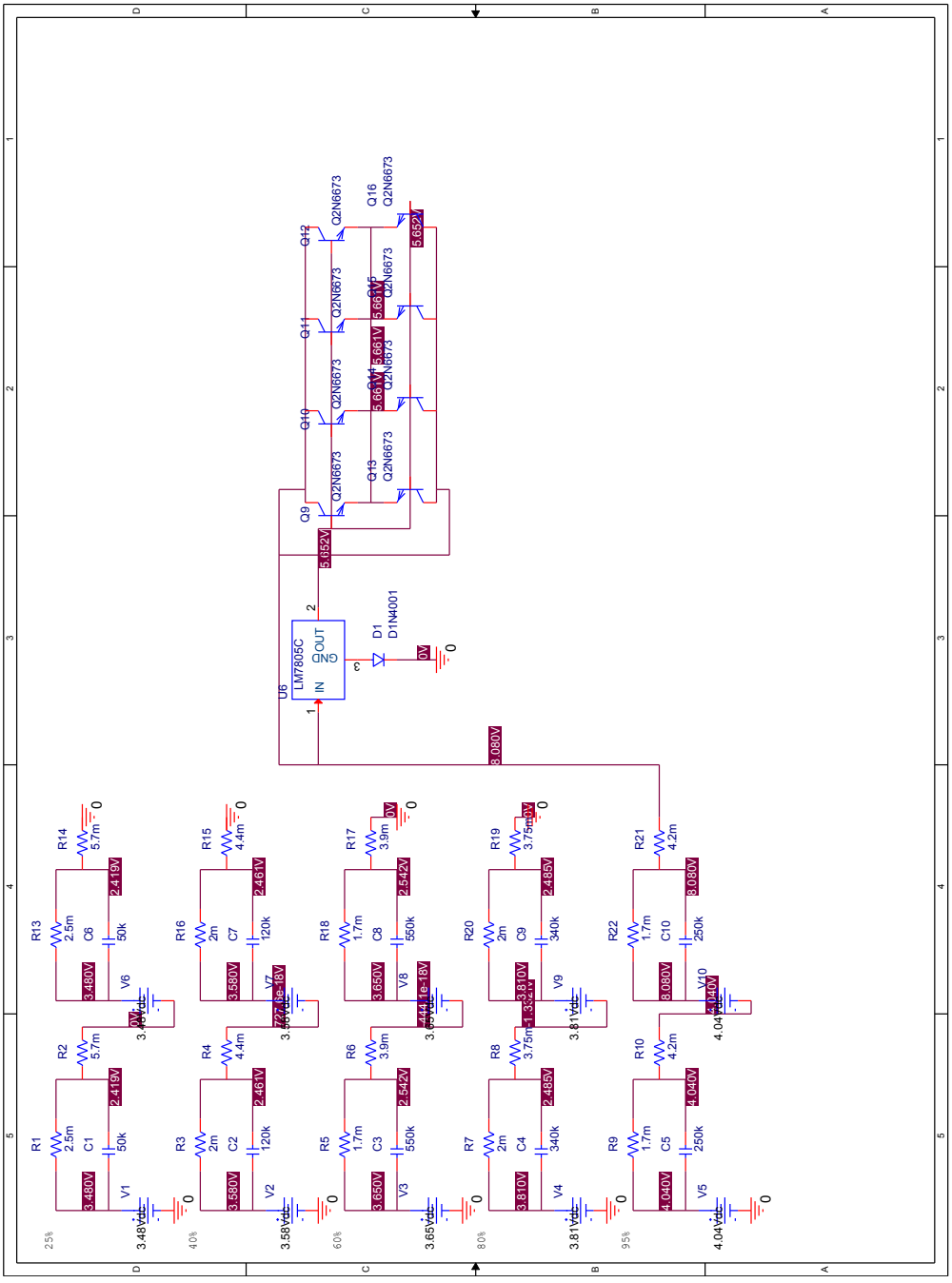


Figure 23: Low voltage power circuit schematic

important when searching for the correct metal to attach the tip and tail.

- Water Resistance - the ability to not corrode easily when water is applied to object.
- Yield Strength - the stress at which a specific amount of plastic deformation is produced, usually taken as 0.2 percent of the unstressed length.
- Compressive Strength - the resistance of a material to breaking under compression.
- Tensile Strength - the resistance of a material to breaking under tension.
- Impact Strength - the capability of the material to withstand a suddenly applied load and is expressed in terms of energy.
- Cost - how much for the total amount of metal we would need.
- Machineability - how easy is it to work with the material

Table 9 shows a trade study of materials considered for the linkages.

Table 9: Linkage Material Trade Study[6, 7, 1]

Material	/19	Water Resistance	/2	Yield Strength	/3	Compressive Strength	/3	Tensile Strength	/3	Impact Strength	/3	Machineability	/2	Cost ($)	/3
Galvanized steel	**14.5**	Yes	2.0	Medium	2	High	3	Low/Medium	1.5	Low/Medium	1.5	yes	2	5.98	2.5
Chromium	**5**	No	0	Low	1	Low	1	Medium	2	Low	1	no	0	(can't purchase easily)	0
Titanium	**11.5**	Yes	2	Medium/High	2.5	Medium	2	Medium	2	Medium	2	no	0	54.5	1
Stainless Steel	**14**	Yes	2	Medium/High	2.5	Medium	2	Medium/High	2.5	Medium	2	yes	1.0	19.27	2
Steel	**10.5**	No	0	Medium	2	Low	1	Medium	2	Low	1	yes	2	10.48	2.5
Aluminum Alloy 7075	**15**	Slightly	1	Medium/High	2.5	Medium/High	2.5	Medium	2.5	Medium/High	2.5	yes	2	10-30	2

The results clearly show that Aluminum 7075 (T6) is the best choice of the materials considered to make up the board linkages.

3.4.4 Statics Analysis

When producing a board, it is important to make sure that it has enough safety and security for consumers to ride without fear of the board breaking. To make the board more modular, the tips were separated from the centerpiece, so that meant a supporting link(s) was needed to keep the board together. Using SolidWorks, the group 3D-modeled the board and possible linkage types. The average pressure generated from a person is about 16 psi, so when running simulations, the assembly was tested at 18 psi to guarantee that it could support the vast majority of people. Through this testing it was apparent which linkages were more effective than others.

Below are some Linkage types that were tested. There were also configurations using double the number of links.

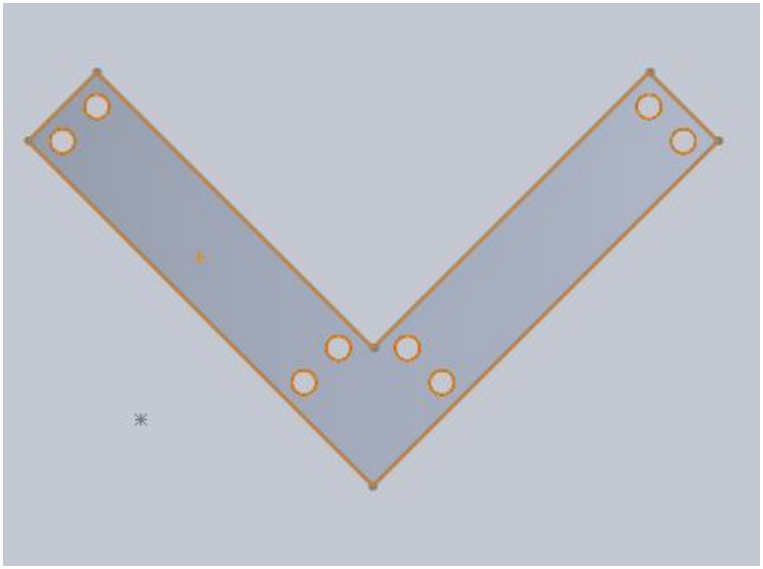


Figure 24: Single-V linkage

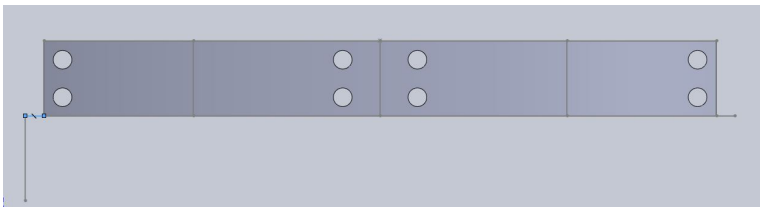


Figure 25: Single bar linkage

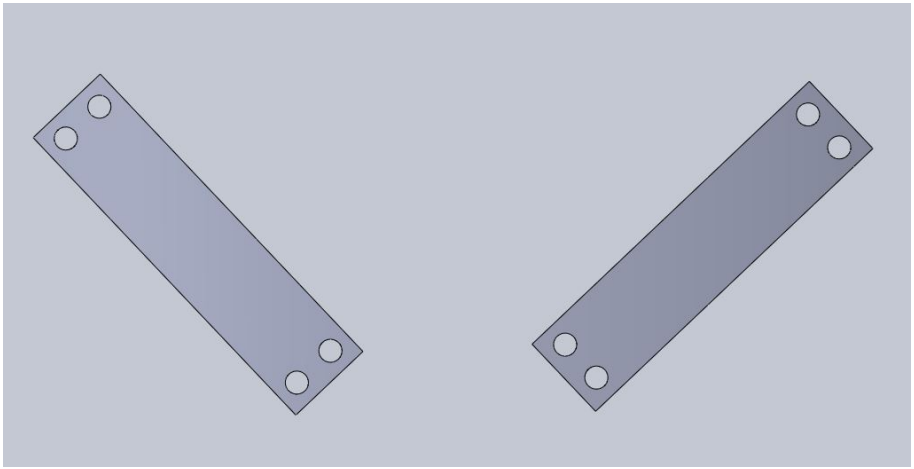


Figure 26: Single accent linkage

The result of all the testing indicated that the Double V structure was the most structurally sound. With the board assembly and links applied static tests were done on the whole system. The test done was a linear static analysis in which the stresses and deformations are displayed on the board based on the forces applied. In this case a simulated 250 pound load was placed on the board to simulate a person's weight when standing on the board. The red arrows, seen in Figure 27 represent the force being applied to the board uniformly.

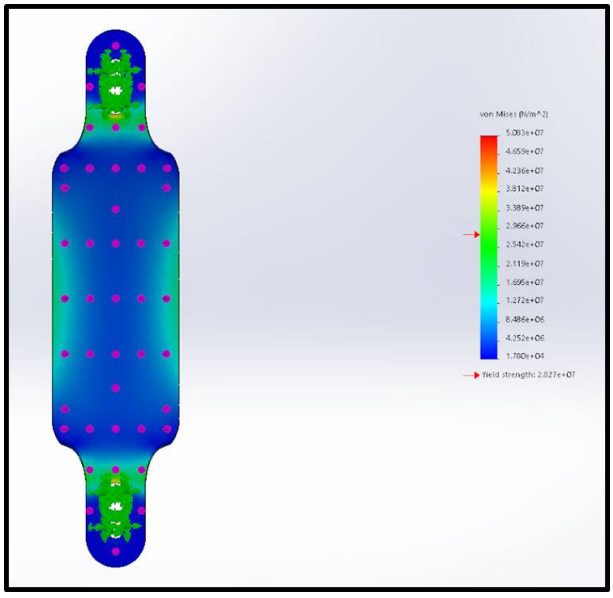


Figure 27: Successful statics test displaying effects of person standing on board

When looking at the result, as long as the coloration on the board does not change to red, the board will not break or splinter under the load applied.

3.4.5 Dynamics Analysis

Similar to the statics analysis, dynamics tests were done on the board with links using SolidWorks. To simulate dynamics testing, a linear dynamics test was used. A simulated person of 250 pounds was still used for the testing baseline. However, while riding a longboard or skateboard it is important to note that turning requires a shifting of weight. As such, for this test a short simulation where the weight distribution,went from 2/3 of a person's weight on the left side and 1/3 on the right to the inverse was tested. This transition would determine whether or not the board could be used for more aggressive turns than often required. Below is the completed dynamics test.

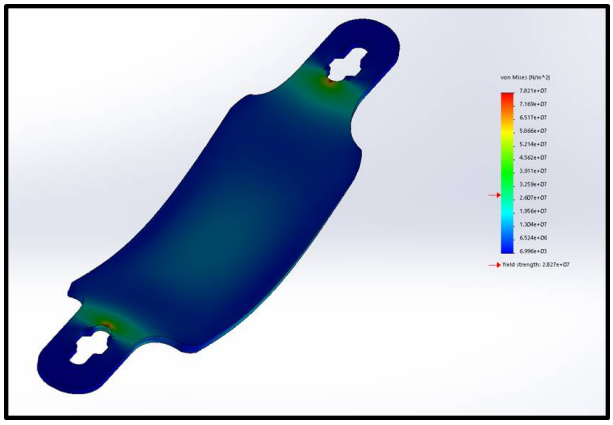


Figure 28: Successful dynamics test of person shifting weight on board

Results show that as long as the coloration on the board does not change to red, the board will not break/splinter under the load applied and moving. The reason there is a large dip in the board is due to the angle of the image. The other reason is because without simulating the trucks with the board, it

cannot show the natural curvature the board would have when say turning. This fact was cross-checked with other references of people who have modeled longboards and skateboards for confirmation.

4 Software

4.1 Real Time System

The real time system, as discussed before, is centered around a Texas Instruments MSP430F5529 micro-controller. Software for the real time system was compiled and debugged with Texas Instruments' development environment, Code Composer Studio v8.2.0, and the included compilers, debuggers, and libraries. The TI v18.1.4.LTS MSP430 compiler was used with the `msp430f5529.h` C header file and the `lnk_msp430f5529.cmd` linker command file, both freely available from Texas Instruments. As discussed in section 2.3, the MSP430 was initially intended to be used inline with the pulse position modulation (PPM) signal, receiving a signal from the controller, filtering it, and then generating an output signal. However, after the latency was found to be unreasonably high and could not be decreased, and since the VESC motor controller firmware allows for sophisticated control options, the MSP430 ultimately was used only to collect PPM values and report them to the main computer, the Rock64, for storage in the database. All code written for the MSP430 as well as necessary Code Composer Studio project files can be found in the git repository at `https://github.com/zane-weissman/msp-ccs`; the

code for a simple PPM passthrough demo utilizing both read and write can be found in the directory `ppm_through`; the directory `dual_ppm_measure` contains the code for the final implementation which reads two PPM signals and reports them over I^2C. The MSP430x5xx and MSP430x6xx Family User's Guide[9] was referenced extensively in the development of this software and may be a useful reference to a reader who wishes to familiarize themself with the MSP430 or understand the operation of this software in more detail.

4.1.1 Reading PPM Signals

The functions necessary for reading the PPM signals generated by the RC receiver are defined in the C header file `dual_ppm_mesasure/ppm.h` and corresponding source file `dual_ppm_measure/ppm.c`. The setup function `ppm_setup`, which must be called before interrupts are enabled globally, configures the P1.2 and P1.3 pins in function mode so that they can be used to trigger timer capture, and then configures the TimerA0 timer. This is a 16-bit timer, configured for "continuous mode," meaning that it counts from zero to $2^{16} - 1$ and then rolls over back to zero. The clock on this timer is driven by the internal 32768 Hz clock. Capture settings are set so that the timer's value is stored and an interrupt is triggered on any rising or falling edge of the signal at P1.2. The capture signal is synchronized with the timer clock to avoid race conditions.

The other function defined in `dual_ppm_measure/ppm.c` is `PPM_ISR`, the interrupt service routine (ISR) that handles the interrupts generated by the

timer module when a rising or falling edge of the signal on P1.2 or P1.3 (corresponding to capture signals 1 and 2, respectively) is detected. First, this ISR reads the interrupt vector `TA0IV` (automatically clearing it) which indicates which pin read the signal that triggered the interrupt. This allows the program to perform subsequent operations on the appropriate hardware and use the appropriate variables. After this, the ISR captures the value of the control register `TA0CCTL1` or `TA0CCTL2`, depending on which capture signal triggered the interrupt. This register contains the volatile `CCI` bit, which contains the current input value of the capture signal that triggered the interrupt. The ISR then checks if that bit is high or low; if it is high, the signal has just finished rising, so a rising edge triggered the interrupt; if it is low, the signal has just finished falling, so a falling edge triggered the interrupt. On the rising edge, the ISR saves the captured value of the timer, stored automatically in the register `TA0CCR1` or `TA0CCR2`, into the global variable `rising_count_1` or `rising_count_2`. On the falling edge, the ISR subtracts the previously stored rising count from the new captured value to find the number of timer counts since the last rising edge. This difference is stored in a global array which is circularly indexed for speed. After the difference is stored the circular index is incremented.

4.1.2 Generating PPM Signals

The functions used to generate PPM signals can be found in `ppm_through/ppm_set.c`. The function `ppm_set_setup` configures the hardware and starts

producing a PPM signal with period and duty width specified by its two arguments, respectively. PPM generation uses TimerA1 in "up" mode, meaning that the timer counter counts up to a set value before resetting to 0 and continuing to count up. The timers in the MSP430 are attached to comparators which can generate output signals based on the value of the timer; in this case a comparator is attached to the timer in "reset/set" mode: when the counter rolls over from its maximum value to zero, the output bit is set to 1, and when the counter reaches a predefined threshold, it is set to zero. With this configuration, generating a PPM signal of a specific period and duty is as simple as setting the value at which the timer rolls over (stored in the register `TA1CCR0`) to the period in clock cycles minus one and the threshold at which the output bit resets (stored in `TA1CCR1`) to the duty width in clock cycles minus one. Changing the duty width is as simple as updating the value in `TA1CCR1`.

4.1.3 Writing I^{2}C Messages

The C source file `dual_ppm_measure/i2c.c` defines the functions implementing access to the I^{2}C hardware available on the MSP430. It includes the function `UCB0_i2c_setup`, which configures the two I/O pins required for I^{2}C and the serial controller UCB0. The other function intended to be called directly is `UCB0_i2c_send`, which takes two arguments: `buffer`, a pointer to an array of unsigned char containing data to be transmitted, and `length`, the number of elements in the buffer. The function first waits for the I^{2}C controller to

report a stop condition, i.e. that the bus is not busy, to avoid conflicts. Then it copies the pointer `buffer` to a global variable `I2C_TX_data` and the value of `length` to a global variable `I2C_TX_byte_count`. Finally the function sets the start condition, indicating to the serial controller that transmission can begin. This prompts the interrupt `USCI_B0_ISR` to run. This ISR checks the `UCB0IV` interrupt vector, and if it indicates that the controller is ready for transmission and `I2C_TX_byte_count` is nonzero, it copies the data pointed to by `I2C_TX_data` into the actual transmission buffer `UCB0TXBUF` and increments the pointer, and then decrements the counter in `I2C_TX_byte_count`. At this point the ISR exits, but it is quickly called again once the byte has been transmitted and the serial controller is ready to transmit again. This repeats until `I2C_TX_byte_count` has reached zero, at which point the stop condition is set, ending the I^2C transmission, and the interrupt flag is cleared.

4.1.4 The `main` Function

The MSP430 uses the interfaces described in the preceding sections to perform its tasks: reading two PPM signals (one, the input to the motor controller from the RC receiver; the other, an output from the motor controller) and reporting the values read to the Rock64. The MSP430 functions as an MESB accessory in accordance with the accessory API described in section 4.2. Only three of the available API messages are used for this accessory: the "Device" message, declaring a new accessory; the "NewVar" message, declaring a new variable; and the "VarStore" message, reporting the value of

a variable. Each of these message types has a buffer allocated for it and filled with as much data as possible at the beginning of the program to streamline their usage later. During the setup portion of the program, the setup functions for PPM and I^2C are called before anything else. Then interrupts are enabled so that the I^2C interface can be used. Now the first three API messages are sent, one declaring the device with the name "Controls," and two declaring variables named "In" and "Out." Next, the program enters its idle loop, where it checks if the buffers used to store the PPM values have been updated. If they have, the next available value is read from the PPM buffer and then written over I^2C to the Rock64.

4.2 Accessory API

The MSP430's I^2C interface exists to communicate with the main computer of the MESB, the Rock64. The Rock64 interacts with the MSP430 and any other accessories via an API layer implemented on top of I^2C . The API is intended to be as flexible and simple as possible, as it is intended to streamline development of accessories to make the accessory system to hobbyists wishing to make their own accessories. There are five types of messages specified by the API: Device, NewVar, NewFun, VarStore, and FunCall. The fields contained in each of these messages are enumerated in Figure 29.

When an accessory boots up, it should first send a Device message specifying its I^2C slave address and a user-friendly name. Then it should send

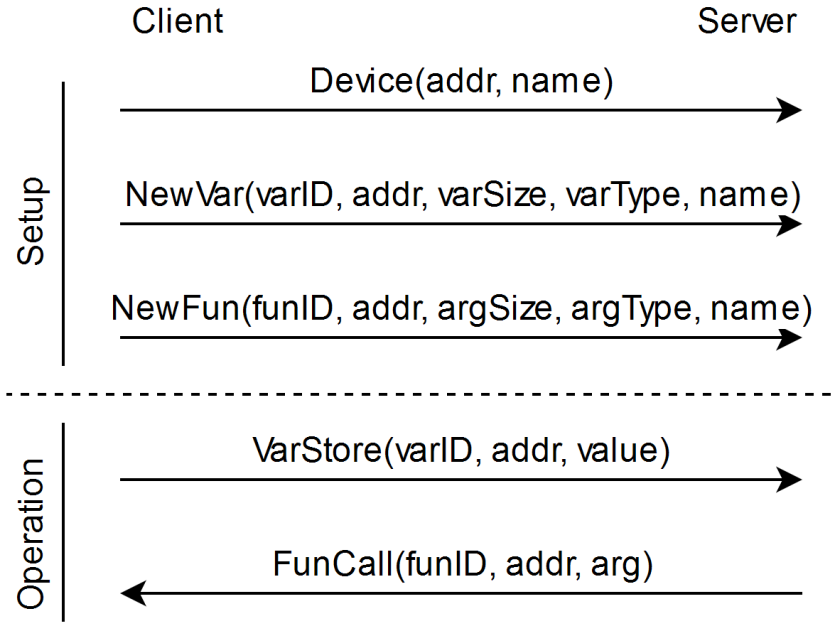


Figure 29: Overview of API messages

as many NewVar and NewFun messages as necessary to specify the variables and functions it will use. Variables are the only way for accessories to access the main computer; they allow accessories to store values that are automatically timestamped and placed into an array by the main computer. Functions are the only way for the main computer to access accessories; they allow the main computer to request that an accessory perform some operation and specify an argument. Functions do not explicitly allow a return value, but an accessory may use VarStore in response to a function call to report a result to the main computer.

The following sections break down the exact specifications of each message type. In general, the first byte's three least significant bits will always hold

the control code specifying the type. For the messages referencing variables or functions, the other five bits of the first byte make up the variable or function ID. These IDs are the shorthand used to refer to specific variables or functions on a per-device basis. The second byte always contains the I^2C address of the accessory. The remaining bytes make up the body of the message and vary more between message types.

4.2.1 Device

The Device message declares a device's name and address. The address is simply the I^2C address, a 7-bit identifier (aligned to the least significant bits within its byte) which should be unique, as it is used internally to identify all information associated with a device. The name is a user friendly string which does not have to be unique. This message should be sent before any other. The fields of the message are enumerated in Table 10.

Table 10: Fields of the Device message

Byte 0	3 LSB	`0b000`
	5 MSB	ignored
Byte 1		I^2C address
Bytes 2-n		name in UTF-8

4.2.2 NewVar

The NewVar message declares a variable assigned to a device. The device is selected by its I^2C address; the variable is identified by its unique 5-bit identifer VarID, but also has a user friendly name which is a string that does not need to be unique. The variable also has a type, which can be integer (encoded by `0b00`), floating point (`0b01`), or string (`0b10`), and a size in bytes. Integers and strings of any size are supported; floating point values should use a size of 2, 4, or 8 corresponding to half, single, and double precision IEEE floating point encodings. Behavior is not defined for floating point variables using other sizes. Table 11 shows all of the fields in the message byte by byte.

Table 11: Fields of the NewVar message

Byte 0	3 LSB	`0b001`
	5 MSB	VarID
Byte 1		I^2C address
Byte 2	2 LSB	type
	6 MSB	size
Bytes 3-n		name in UTF-8

4.2.3 NewFun

NewFun declares a function assigned to a device. The format is very similar to that of NewVar, including a function ID rather than variable ID, and argument type and size rather than variable type and size. Table 12 demonstrates

the specifics of the NewFun message.

Table 12: Fields of the NewFun message

Byte 0	3 LSB	`0b010`
	5 MSB	FunID
Byte 1		I^2C address
Byte 2	2 LSB	type
	6 MSB	size
Bytes 3-n		name in UTF-8

4.2.4 VarStore

The VarStore message is used to store the value of a variable. Variables are not overwritten as they would be in a normal programming context; rather, they are timestamped when they are retrieved and logged with all prior values of the variable. The VarStore message simply includes the device address, variable ID, and value to be stored, as shown in Table 13e

Table 13: Fields of the VarStore message

Byte 0	3 LSB	`0b011`
	5 MSB	VarID
Byte 1		I^2C address
Bytes 2-n		value

4.2.5 FunCall

The FunCall message is the only message intended to be received by devices. It could be sent from the main computer or from another device. When a FunCall is received, the device may do anything or nothing; a function is simply an abstraction used to indicate that something should happen or to pass data (through the argument). The format of FunCall, presented in Table 14 is similar to that of VarStore, just as the format of NewFun is similar to that of NewVar.

Table 14: Fields of the FunCall message

Byte 0	3 LSB	`0b011`
	5 MSB	FunID
Byte 1		I^2C address
Bytes 2-n		argument

4.2.6 Example API Operation

For example, consider an accessory which implements controls for a headlight attached to the front of the board. The operation of such an accessory might be as shown in Figure 30. In this example a device using I^2C address `0x24` declares itself with the name "Headlight" and a single function, "Switch," which takes a single byte integer argument and has function ID 0. The user calls this function by its ID, 0, and the address of its accessory, `0x24`, providing the argument 1 indicating that the headlight should turn on. The

user then calls the same function with the argument 0, indicating that the headlight should turn off. It is the responsibility of the designer of the accessory to ensure that the light itself is turned on and off when the appropriate function calls are received.

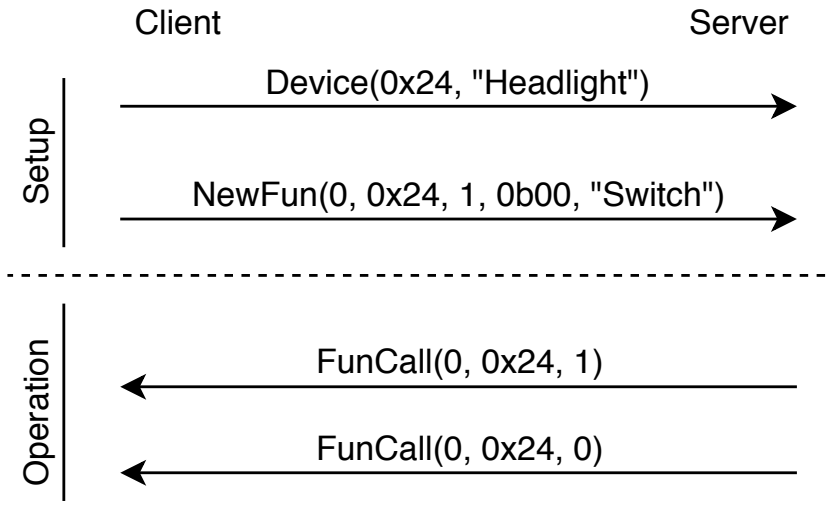


Figure 30: Example of API Operation

4.3 Database Connectivity

The primary database used for this MQP was Firebase. It has the capability to be updated in real-time, and was compatible with both the front-end and back-end which used a combination of Python, HTML, and CSS. The I^2C daemon, a simple Python application, reads incoming I^2C messages and translates them to database requests in real time, while the web server, another Python application, uses the Django web framework to build the front-end web application from the database on demand. Additionally, the web

server can call a simple Python script that sends accessory API messages to accessories. Figure 31 displays how the peripherals and the web application connect to the database.

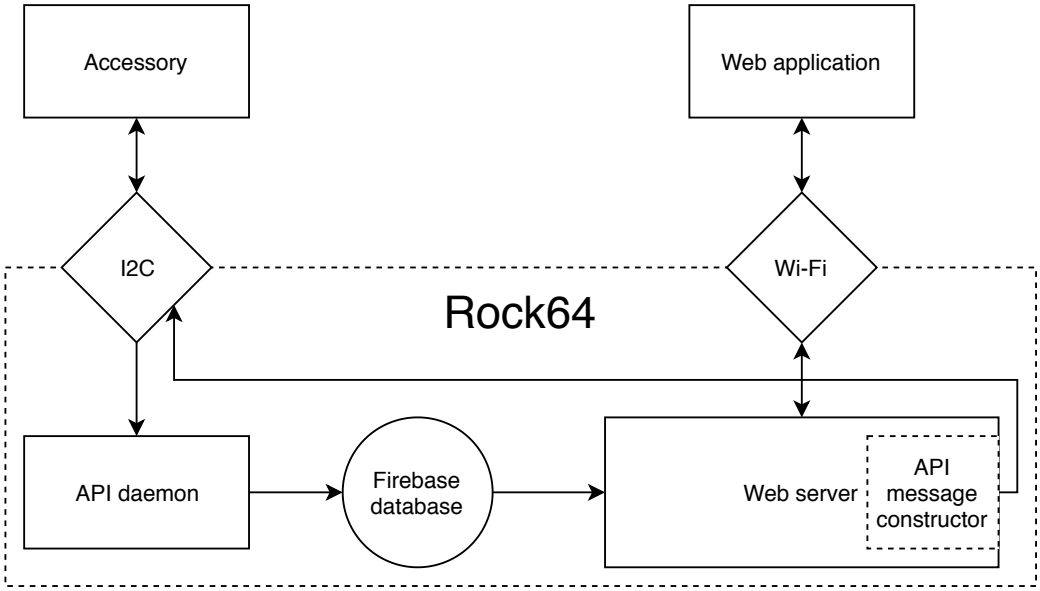


Figure 31: Peripherals to Database Connections

4.3.1 Database

Figure 32 shows some sample data in the database, illustrating its organization. When the system is turned on, a new "run," labeled with a timestamp, is created under the top level directory `runs`. The run contains a directory `peripherals`, which is automatically populated when peripherals are connected to the system and begin reporting their properties. Peripherals are listed in this folder first by their I^2C address. They each have a property `name` which is a user-friendly UTF-8-encoded string, and can have a subdi-

rectory for variables `vars` and one for functions `funcs`. Within these, each variable or function is listed by its unique identifier, used to reference that variable or function in the I^2C API as described in section 4.2. Variables and functions each have a `size` and `type`; for variables these describe the size and type of the variable; for functions these describe the size and type of the argument to the function. More detail on the encoding of the `size` and `type` fields can be found in section 4.2.

4.3.2 Web Application

To connect the firebase database to the website a combination of Python, HTML, and CSS is used. The supporting libraries need were Django for the web based access and controls, and firebase sdk was used to access the firebase database. The project the uses views.py to connect the url links to the html and CSS visuals. There is a Login screen, in which users can type their information to connect to the database (shown in Figure 14). Upon login a home screen is shown with the option to see only the current run or all prior runs. After selecting one of the options, it displays a list of run(s) (shown in Figure 33). Each run contains all of the data for that run displayed in the order of peripherals (both name and ID), their associated vars with ID's, names, sizes, counts, and the values related to those vars which can be displayed by clicking on the hyperlinks. All of the code for the website can be viewed at `https://github.com/Modular-Electic-Skateboard-MQP/WebApp`.

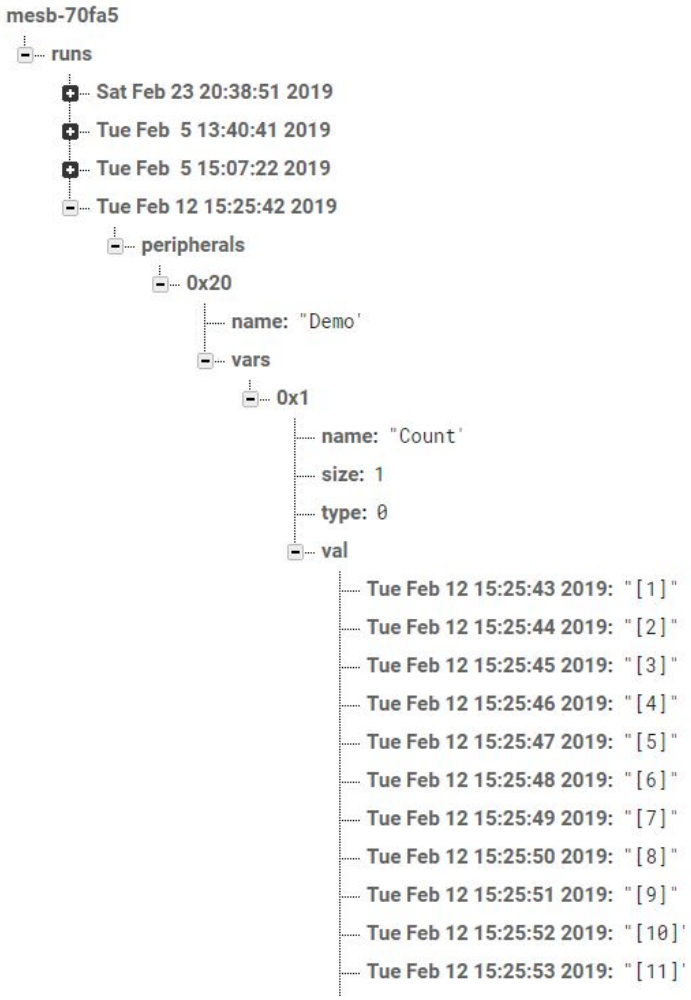


Figure 32: Firebase Data

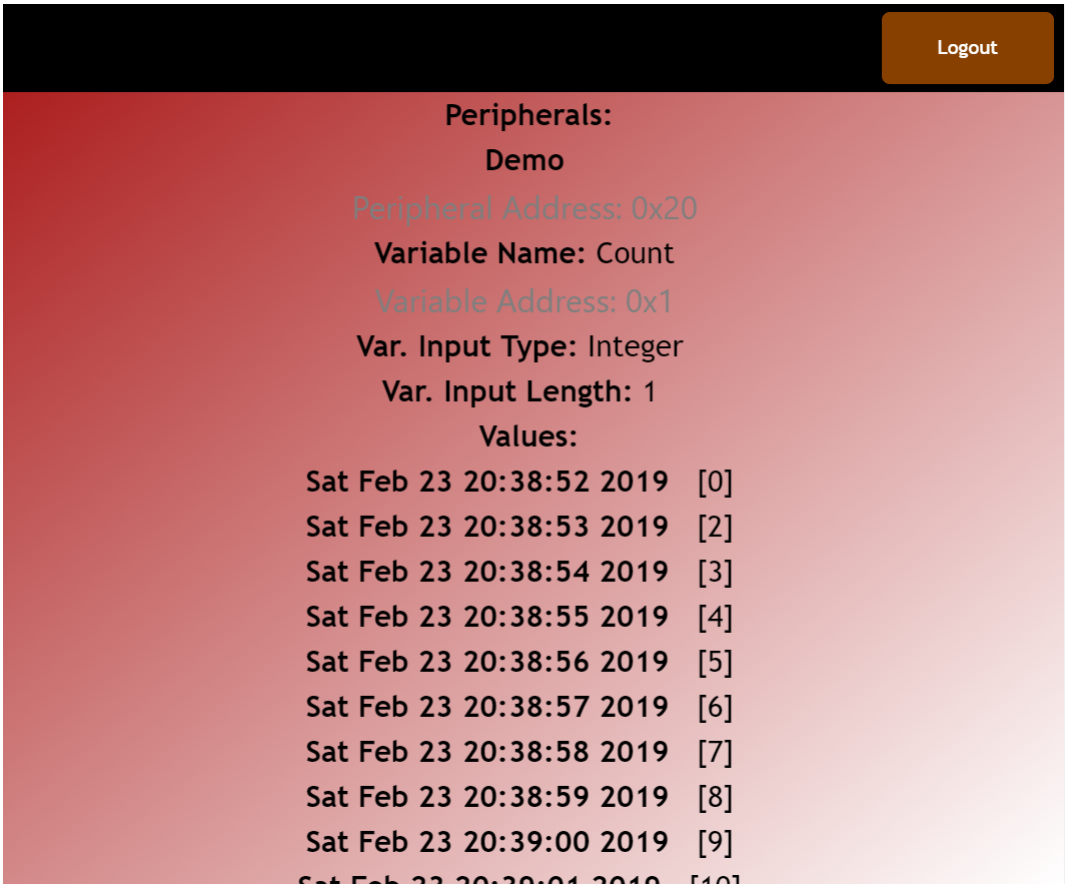


Figure 33: Display of one peripheral, showing a variable with several recorded values

5 Manufacturing

5.1 Deck Fabrication

One of the crucial features of the modular electronic skateboard is the modularity of the board itself. The following sections describe the process of modifying a standard longboard deck to create a deck with interchangeable nose and tail.

5.1.1 Board Separation

The first step in manufacturing this system was to partition the board into three pieces, separating the nose, tail, and middle segment of the board. The cuts were marked with a flexible straightedge that would lie along the curvature of the board, and a bandsaw was utilized to perform the cuts (as you can see in Figure: 34).

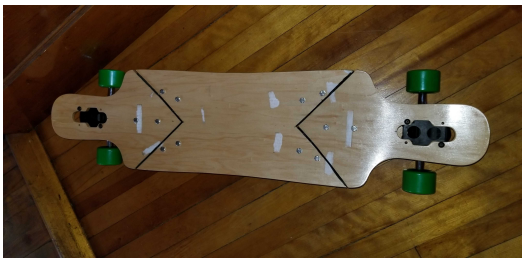


Figure 34: Board from a Top Down View

5.1.2 Link Creation

The links were made from a T7075 aluminum bar (1/2" x 1"). First, the bar stock was shaped by putting it through a roller at a slight angle to help match the curvature of the board. It was then split into four 4" links and four 5" links. Then a 1" by 1" square was milled out of the end of each link to a depth of 1/4", half of the thickness of piece. This was done on concave surface for half of them and convex surface for the other half, so that when placed together the two links can lay flush with each other and with the curvature of the board. One hole was drilled in the center of each square cutout; another hole was drilled at the other end of each link. The holes are not threaded and are meant simply for bolts to pass through and be tightened with a nut on the other side. Figure 35 shows a closeup of the links. This production was all done in the Washburn shop on campus at Worcester Polytechnic Institute.

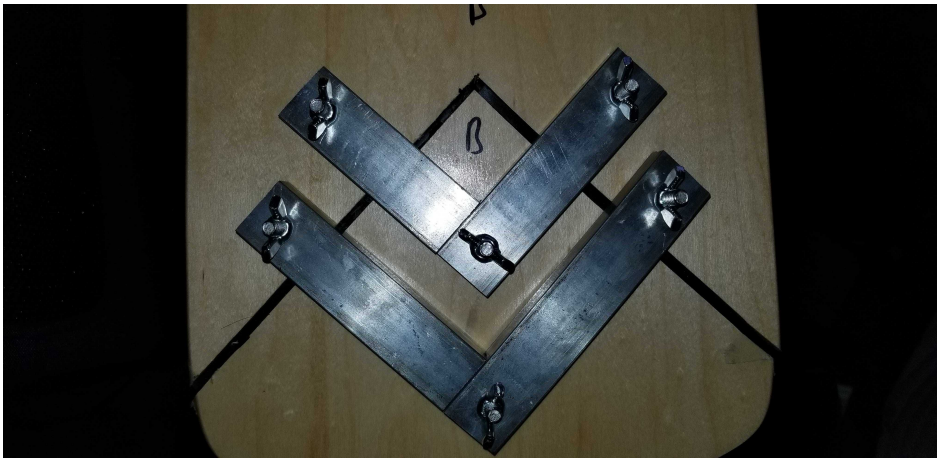


Figure 35: Closeup of Links

5.1.3 Board Assembly

First, in order to assemble the board, the cushioning sorbothane was cut into small strips roughly the thickness of the board. These strips were placed along the inside of the junctions between the nose and tail and the center section. Then the nose and tail were clamped in place to hold the junctions steady. The metal supports were then put into place and clamped as well, and then a hand drill was used to drill holes in the board aligned with the holes in the supports. Finally, the clamps were removed and nuts and bolts were used to fix the supports to the board, as seen in Figure 36.

Figure 36: Completed Board (Underside)

5.2 Drive System

As discussed in 3.1.3, the drive system is comprised of two high powered motors, each of which are paired with a set of sprockets to fit a v-belt. Upon further consideration, two avenues presented themselves for mounting the motor; either fabricate a custom mount, or purchase one available commercially. The latter option was taken due to budget and timing reasons, with the choice of mount being the Turnigy Skateboard Conversion Kit Motor Mount, as shown in Figure 37. In order to comply with size restraints within the board, and considering the size of the motors, the mounts were attached in opposing directions, as visible in Figure 38. After attaching the motors, the motor controllers were placed in any leftover area on the tail. Referencing Figure 39, one can see that one motor controller is mounted on the top, and the other on the bottom. This was due to the size constraints of the leftover surface area, after attaching the motor mounts with motors.

Figure 37: Turnigy Skateboard Conversion Kit Motor Mount

Figure 38: Motor Mounts with Motors on Board

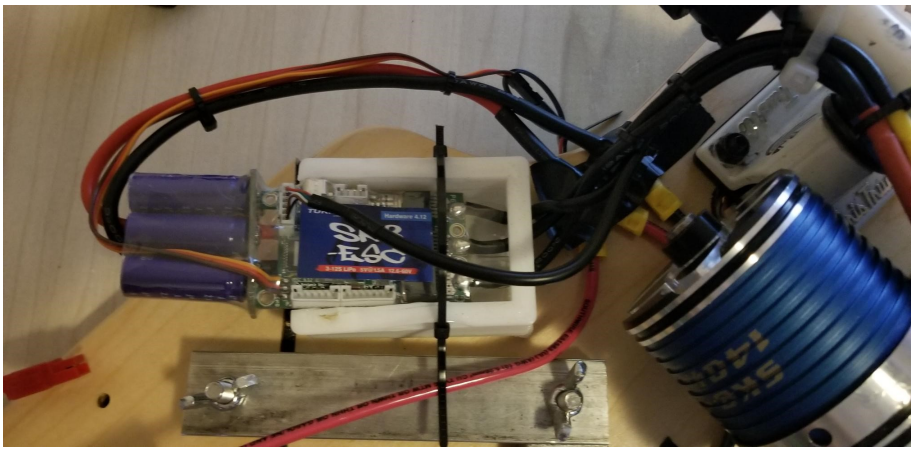


Figure 39: Motor Controllers on Tail

5.3 Electrical System

5.3.1 Power

As discussed in 2.4, the board is using an array of lithium polymer cells to supply power to the various electronic components necessary to drive the skateboard, and power the accessory system. These cells have constraints

that were followed when fabricating the lithium polymer (LiPo) battery packs:

- Cell tabs must be either spot welded or soldered
- Cell casing must be thick enough to ensure proper shielding from the elements, and any contact with metal
- Cells must be easily detachable from the system
- Cells must be arranged in a logical way to maximize space efficiency
- Cells must fit on the board. It was determined that the best way to fit all 20 cells on the board would be to build two packs of 10 cells each, in a 2x5 grid.

These constraints were obeyed by soldering the tab onto the cells in a methodical manner, maintaining the integrity of the solder joints between cell leads and tabs. Once this was done copper wire leads[2] and 6-pin balancing plugs were attached to the cells, and the cells were then wrapped in layers of electrical tape (about 20 layers per pack, approximately 4mm thick).

Once the battery packs were fabricated, they were safely tested for capacity by placing each cell in a LiPo safe bag, which was then paired with a balancing LiPo charger in a plastic reusable container, which was then inserted into a bucket of sand to contain any explosions. These efforts were taken to ensure the safety of all students nearby. Once the LiPo cells were

[2] 10 Gauge, rated for 600 Volts, and 90°C

Figure 40: Battery Pack

charged repeatedly without issue, the bucket of sand was removed from the charging procedure, since the redundancy was no longer needed.

Finally, the battery packs needed to be connected to a switch, in series, in order to complete the battery pack that would power the electronics. A 60 amp AC circuit breaker was obtained to fulfill this purpose. The breaker was modified to switch a DC signal, by soldering the two neutral nodes together. This essentially converts the two phased channels that an AC power signal would use into a single channel for DC current to channel through. It should be noted that the 60 amp breaker was chosen specifically to comply with the maximum calculated current draw. Although the system is designed to not exceed 60 amps, the breaker acts as a redundant, electro-mechanical mechanism to further ensure the rider's safety.

It is important to note that each battery also was charged using a dedicated LiPo charger. The batteries were places inside a plastic bag, which

was then placed inside a bucket of sand to ensure safety while charging. The LiPo charger discharged, and balance charged the cells.

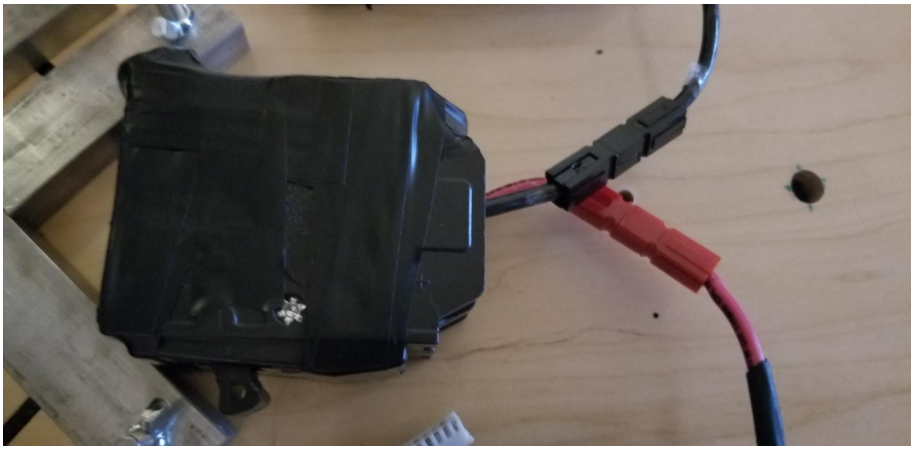


Figure 41: 60 amp breaker attached to LiPo cells

The two battery packs and breaker were mounted on the board using four large zip-ties, and electrical tape for preliminary bench-top testing. Once the rest of the components were successfully mounted onto the board, a custom 3D printed housing was designed to safely contain the batteries and breaker. This housing, as shown in Figure 42, contains the LiPo cells and the breaker safely, while still maintaining a degree of accessibility to each component.

5.3.2 Cable Placement

Since there is very little surface area on the MESB system, it is important to design the layout of the electronic components in a methodical manner. Specifically, when placing the components on the board, there was a set of constraints that were obeyed:

Figure 42: Custom battery mount

- Minimize the amount of wire crossing
- Lay out the motors and motor controllers in a manner where the modularity of the nose and tail replacement is maintained
- Safely route cable connections throughout the board
- Shield connections as much as possible

Obeying these constraints lead to a particular layout, as depicted in Figure43. This layout maintains the modularity of the board by ensuring the grouping of the tail, motors, motor controllers, and cable leads to the motor controllers. In other words, the user can remove the electric part of the board by just unscrewing four wing-nuts.

In order to connect the cables for bench top testing, regular exposed metal terminals were used for rapid prototyping. For environmental testing these were replaced with Anderson connectors, as shown in Figure 44 for secure

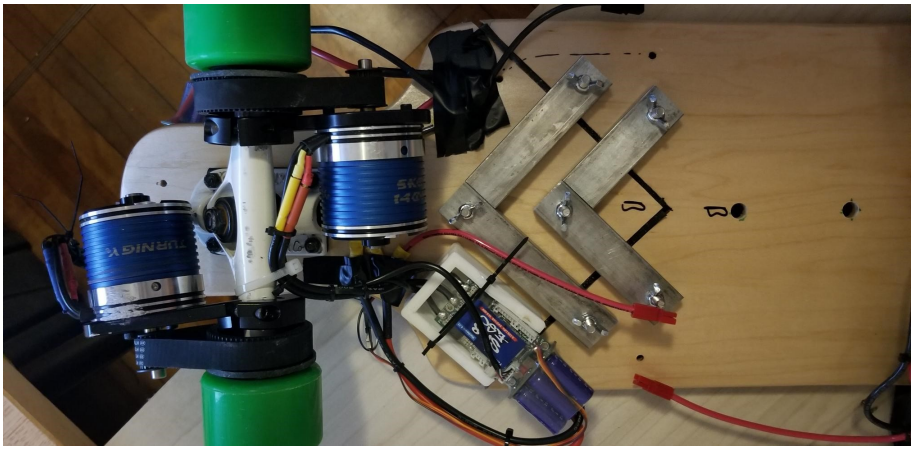


Figure 43: Layout of components and cables on the tail of the board

and insulated connections. These were chosen because of the low cost, high current capacity, and rugged design these connectors employ.

5.4 Modular Aparatus

The housing, previously shown in Figure 22, was the last of the MESB proof of concept components to be installed. Ideally, this piece would be fabricated through a resin or plastic mold. However, due to budget and time constraints, the fabrication was done utilizing a 3D printer. Due to machinery size constraints, the housing was separated in half. These parts, as demonstrated in Figure 45a, were altered in order to ensure repeated accuracy when combining the pieces and mounting the housing to the board. Figure 45b shows the two halves of the housing combined. In addition to the housing, clear Plexiglas was acquired and cut to size to complete the housing.

Figure 44: Anderson connectors, used for most high power electrical junctions (between motor controllers, batteries, and power switch)

Figure 45a: Two halves of accessory housing after 3D printing

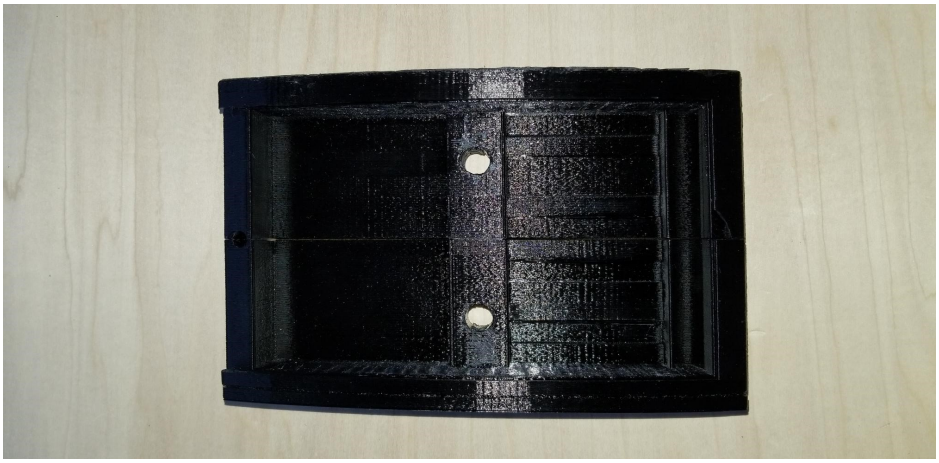


Figure 45b: Halves of accessory housing, assembled

6 Design Validation

6.1 Test Methodology

In order to validate the MESB proof of concept, various tests were designed to evaluate the mechanical and electrical systems.

6.1.1 Static and Dynamic load

The deck, as mentioned in 3.4.4, was designed to handle up to a 250 lb load in both static and dynamic scenarios. In order to test the static load, the team came up with a simple experiment:

1. Place board on a level surface with no load, and measure distance from center of deck to the ground
2. Place pre-weighted bag with 100 lbs of materials on the board, and measure distance from center of deck to the ground
3. Remove bag and measure distance to ground
4. Repeat procedure steps 2 and 3, increasing the load in small increments until it reaches 250 lbs

The dynamic load test, unlike the static load test, is more challenging due to the posed issue of measuring the distance from the ground while the board is under load. It was determined that the most important metric to effectively measure dynamic load was to measure the board deflection

(distance from the center of the board to the ground) before and after a rider with a sufficient load drove the board in flat, acending, and descending roads.

6.1.2 Board Performance

Although it is easy to see if the board will drive under one particular set of circumstances (Refer to video in supplemental material), it is important to ensure that various systems are performing to specifications in a wide variety of scenarios. These tests recorded the performance of the board with respect to current draw and speed. Specifically, in order to evaluate the system design goals (support 250 lbs, at an incline of 20 degrees, sustaining a speed of 20 mph), the test are comprised of the following steps:

1. Tighten any mechanical connections on the board, boot up the electronics and verify successful movement of motors through RC signal.

2. Connect both speed controllers serial data lines to a laptop, enclosed in a backpack to record the data.

3. Have test subject begin to operate the MESB. Make a time stamp on a stopwatch for every time during operation there is a significant change in elevation slope, speed, or other significant events

4. The user, during the ride, will be tasked to ride over steep inclines up to twenty degrees, with varying speeds, in order to identify any stall

configurations for the drive system. Once the aforementioned goals are complete, the user will return to the start.

6.1.3 Static and Dynamic load

The resulting deviations demonstrate that the board's design validates the claim of supporting up to 250 lbs of static weight, and not bearing permanent deviation. We validated this by observing the deflection of the board with various weights, and no permanent board deflection when the weight was removed (supported by Table 6.1.3).

Intuition about the board response might say that there is a strict linear relationship between deviation of the board and applied weight. Fitting a linear relationship between the data points validates this intuition, as shown in Figure 46. The polynomial relationship is shown to prove that higher order polynomials are suspect to accute fitting to the data, hence the polynomial model as shown in the afformentioned figure's behavior.

In terms of dynamics, measurements were taken before and after every ride for the experiments performed for board performance, since the board performance experiment has all the requirements of the dynamics test. The Board Height before every experiment remained the same, at 4.61 inches. After every experiment, there was no difference in deviation.

Table 15: Measured Deflection of Deck under static loads

Statics Test Results		
Distributed Load (pounds)	Board Height (inches)	Deflection Error with removed load (inches)
0	4.61	0
25	4.46	0
50	4.315	0
175	3.68	0
225	3.71	0
250	3.54a	0
200	3.725	0
225	3.55	0
250	3.9	0

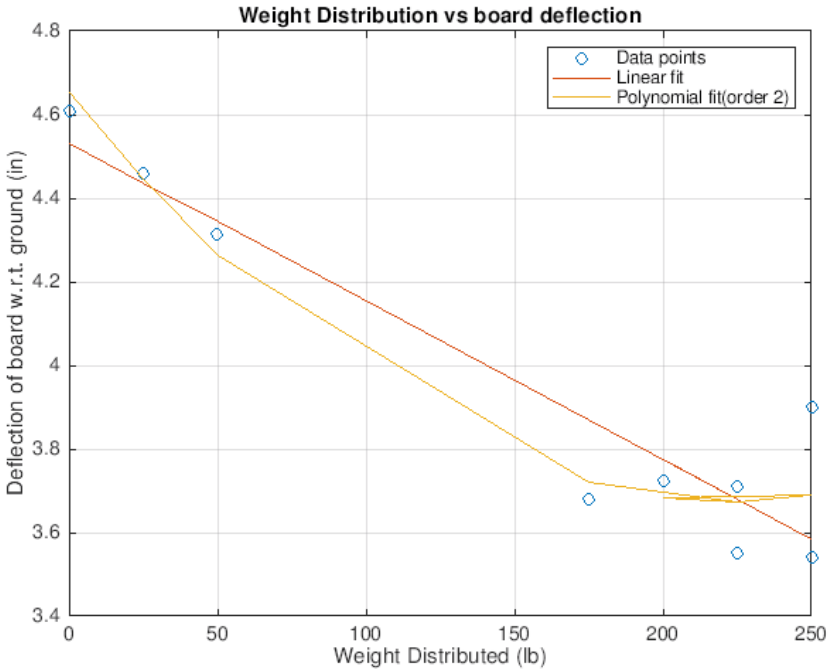


Figure 46: Plot and Linear Model of Static Test Data

6.1.4 Board Performance

The test for Board Performance was conducted on Febuary 25, 2019. This test was conducted in two locations; WPI Gateway parking garage, and WPI "quad." Both are locations within Worcester Polytechnic Institute, Worcester, MA.

The test procedure was followed, as detailed in 6.1.2, with inclines that were estimated to be around 15 percent (regulation for parking garage ramps to not exceed 15 percent). Weather conditions were windy, but not enough to affect the integrity of the data.

Due to the amount of data, only significant time stamps[3] are parsed and shown in Figure 47. This data encompasses the total board performance in terms of the motor draw over the duration of our tests, displaying the motor draw and battery draw at a given state.

Figure 47 shows that the board will draw 50 amps from the drive system at maximum, which is within the expectations of the calculated 60 amps. This lends itself to the motor selection, since a more powerful motor was chosen. The second point that the results show is the battery draw never exceeding 14.82 amps per motor controller (This is possible because of the capacitors present on the motor controllers). It should be noted that the rise time in inclines was significantly low, and the size of the capacitors might have affected the current draw from the battery system.

[3] i.e. moments in time where the board was accelerating, decelerating, under significant load, etc...

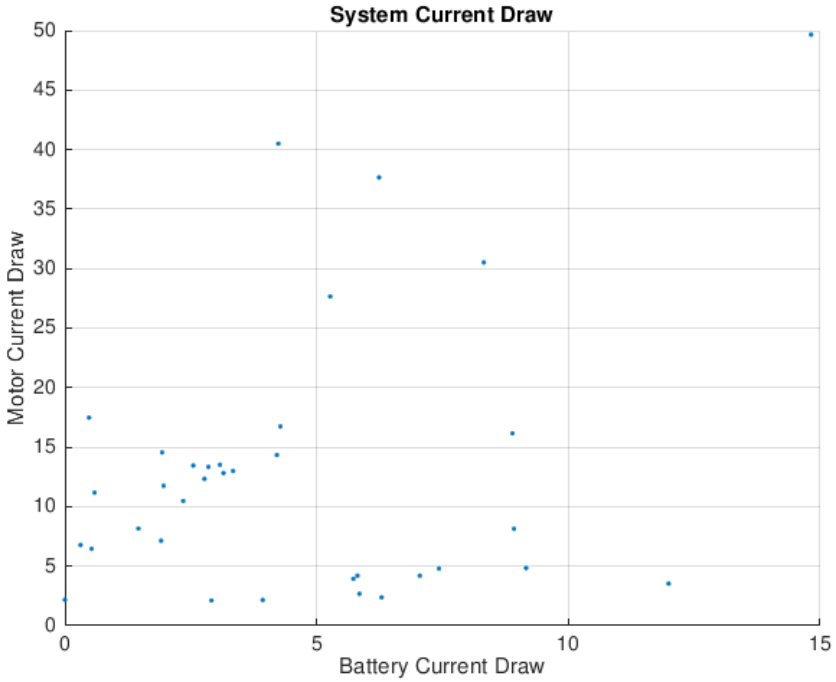


Figure 47: System Current at significant timestamps (Amps)

Overall, the board during the tests was able to achive our design goals. Specifically, be able to handle 250 lbs, be able to accelerate to and maintain a speed of 20 mph, and be able to climb significant inclines.

7 Social Implications

As cities continue to get denser and streets get more crowded, and as climate change becomes a more immediate issue, people will seek practical alternatives to cars in even greater numbers than they do today. Small, cheap, single-user, electric vehicles like the MESB are an economical and

eco-friendly transportation option that is practical to use in combination with public transit. A vehicle like the MESB can significantly expand the radius of practical destinations from a bus stop or subway station.

8 Future Work

There were several aspects of the prototype that could be improved significantly in the future. Most significantly, the MSP430 was not used as part of the motor control loop due to the excessive latency it generated, as discussed in section 2.3, and the motor control features that could be implemented were therefore limited to those available in VESC, the firmware running on the SK8-ESC motor controllers. Ideally, controls would be handled by only one microprocessor, which could run a fork of VESC or some custom software. Additionally, the drive system would benefit from a custom mounting bracket for the motors which would fit the trucks better and could hold the motors at an angle more precisely aligned with the axle, which would allow the the belt system to be more efficient and more reliable. The front end web application could be given an improved graphical user interface, as usability and aesthetics were not heavily prioritized during the design process due to time constraints. And of course, the flexible accessory system of the MESB allows for a wide variety of possible accessories.

References

[1] Aluminum 7075-t6; 7075-t651. http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=ma7075t6.

[2] Atmega328p - 8-bit avr microcontrollers - microcontrollers and processors. https://www.microchip.com/wwwproducts/en/ATmega328p.

[3] Msp430 ultra-low-power mcus — overview — microcontrollers (mcu) — ti.com. http://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/overview.html.

[4] Rock 64. https://www.pine64.org/?page_id=7147.

[5] Stm32 arm cortex microcontrollers - 32-bit mcus - stmicroelectronics. https://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html.

[6] Search matweb for property information. http://www.matweb.com/search/search.aspx, 1996.

[7] Homebuilt aircarft info. https://www.experimentalaircraft.info/articles/aircraft-aluminum.php, 2006.

[8] Hard maple. https://www.wood-database.com/hard-maple/, 2008.

[9] *MSP430x5xx and MSP430x6xx Family User's Guide*, February 2013.

[10] Eric Podwojski. How an electric skateboard works, 2015.

[11] Alex Rowe. Motor control of a hub motor for electric skateboard propulsion. Master's thesis, Massey University, 2016.