Husky Super Brute RV Lift

Nader Abd El Rasol, Lucy Golebiewski, Andrew Melvin, and Hesham Zidan

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450 (12 point Times font)

***Abstract* — The Husky Super Brute stabilizer jack is a practical and useful application of electrical and computer engineering concepts. Recreational vehicles need to be lifted and stabilized to ensure their safety and longevity. The new Husky jack uses a blend of hardware and software to provide a reliable, user-friendly, and innovative way to bring stabilization to customers' vehicles. The PCB was custom designed to withstand high power applications and fit within Husky's existing stabilizer jack. The system is comprised of four main integrated circuits to control the motor with a mobile application, provide security measures, and an extra feature for customers.**

***Index Terms* — DC Motor, Bluetooth Low Energy, RV stabilizer jack, mobile application, accelerometer.**

I. Introduction

Recreational Vehicles, or RVs, need to be stabilized while parked either at home or a campsite. Stabilizing keeps the RV steady when people are walking around inside, or wind is blowing outside. The stabilization provided by the Husky Super Brute jack is important for user safety and the lifespan of the vehicle. Many jacks are manually operated with a hand crank or handheld drill, but this is a fully automated jack controlled with a mobile application. The mobile application connects to the jack via Bluetooth Low Energy. This connection allows the user to lift and lower the stabilizer jack at the touch of a button. There are safety features in place to prevent the user from lifting or lowering the jack if it is already at its maximum or minimum point of operation. The module selected for Bluetooth Low Energy is also the main controller for our system. It controls and receives the data from the motor driver, accelerometer, and hall effect sensor.

The team researched and designed the printed circuit board to fit the requirements given from the sponsor and customer. The PCB needed to fit within the dimensions of the Husky jacks already being sold on the market. This allows the company to easily put the Husky Super Brute board directly into their jacks by plugging their wires into our connectors. The electrical design team needed to conform to these constraints by making sure the size of the board was perfect, the modules were oriented in the correct locations, and the correct connections were implemented. The connectors needed to correspond to the wires within the stabilizer jacks, so the team made sure to accommodate for the manual switches and power from the Husky jack.

One of the most important factors the team needed to consider while designing the circuit board is the high voltage and current supplied to the jack. These brushed DC motors require 12 volts at the input, so we designed the board to withstand extra voltage to a maximum of 20 volts. The rest of the components in this system run off 3.3 volts, so electrical team needed to design the system to convert this input voltage to lower voltages. The current supplied to the motor is between 20 and 30 amps, while the modules require much less current, measured with milliamps and microamps. Working with such high power can be dangerous, so we made sure to take precautions while testing the system. The integration of software and hardware in this project created a beneficial product and provided the team with new skills.

II. System Components

The system is comprised of several different modules that work together to create the final product. This section will discuss the components chosen and their purpose in the design. The following devices were developed by different companies, so the team gained different experiences from each one. An overall system design can be seen in Fig. 1 with the SPI communication and power connections.

A. Cypress Bluetooth Module CYBLE-212006-01

The main controller for this system is the Bluetooth low energy module developed by Cypress, which is now owned by Infineon technologies. This module receives instructions from the user through a mobile application connected through Bluetooth. These signals are then processed in our driver and command the other modules to perform their intended actions. The CYBLE-212006-01 connects to the accelerometer and motor driver through SPI communication lines. The software team developed drivers for each module based on the Cypress SPI protocol. The main program calls the different drivers

to perform actions such as motor control, reading accelerometer data, detecting motion data, and gathering pulse information to calculate motor speed and direction. The Cypress module receives the data and sends some of it back to the phone for the user to view. The main function of the mobile application is to control the motor, but a new lighting function has been implemented in this design. Users can plug LEDs into their stabilizer jack to light up their workspace or just provide light at their campsite. Previous methods of lighting require users to wire the lighting from inside of the RV, which can be very complicated and time consuming. The Husky Super Brute provides an easy connection point for LEDs with a supply voltage of 12 volts. The user can control the colors with their mobile application. The Cypress module driver receives these signals and sends them to the Infineon TLE-92108 driver.

B. Infineon Motor Driver TLE-92108-232QX

The TLE driver is a MOSFET driver that can drive up to eight half-bridges. Half bridges one and two are used to control the motor through extend and retract signals from the 6-position switch connector. Half bridge 3 is used for the brake signal. Half bridges 4-8 are used for controlling the external LEDs. The TLE from Infineon is a new component, so there was not a lot of data and example code to use. This team developed our own driver, discussed in "Software Design."

C. Diodes Incorporated Hall Effect Sensor AH3774

To monitor the speed and direction of the motor, the team chose a hall effect sensor from Diodes Incorporated. The hall effect sensor has a single open-drain output that is switched on and off with a magnet. The south pole of the magnet switches the output on, while the north pole of the magnet switches the output off. Two magnets were placed around the motor for the hall effect sensor to detect their magnetic fields. As the magnets rotate, the sensor sends voltage signals to the Cypress module. The team developed a program to calculate the direction and speed of the motor based on the timing of the pulses. Since the magnets are placed at different intervals, a short pulse indicates the motor is spinning clockwise, while a long pulse indicates the motor is spinning counterclockwise. Using the pulse count and timer within the Cypress module, the team calculated the speed of the motor spinning. This data is useful for implementing safety measures within the system. If the motor is spinning to fast or in the wrong direction, the Cypress will notify the motor driver to stop the motor.

D. Bosch Sensortec Accelerometer BMA456

Along with the hall effect sensor, the team implemented safety features with an accelerometer from Bosch Sensortec. The accelerometer can detect the orientation of the PCB and sends the data to the Cypress module at a rate of 25 Hz. This data is in the form of X Y Z axis coordinates. If the accelerometer detects motion above the set threshold, it sends a motion detection interrupt. This interrupt allows the Cypress module to send a warning notification to the user, stating that the device is experiencing excessive motion. The stabilizing jack should not be tilted out of place while in use and could cause damage if not prevented. To develop the driver for the BMA456, the software team utilized the API provided by Bosch Sensortec for their line of BMA4xx accelerometer devices. While utilizing the API provided much of the code needed for background processes, the main communication and driver needed to be coded by the team. The SPI communication provided some challenges due to the specific requirements in the API, but the team was able to overcome this obstacle. Since the accelerometer and the Bluetooth module were developed by two separate companies, there was not an abundance of resources to refer to while creating the driver. The team was able to communicate with other programmers on community forums provided by Bosch Sensortec and Cypress.

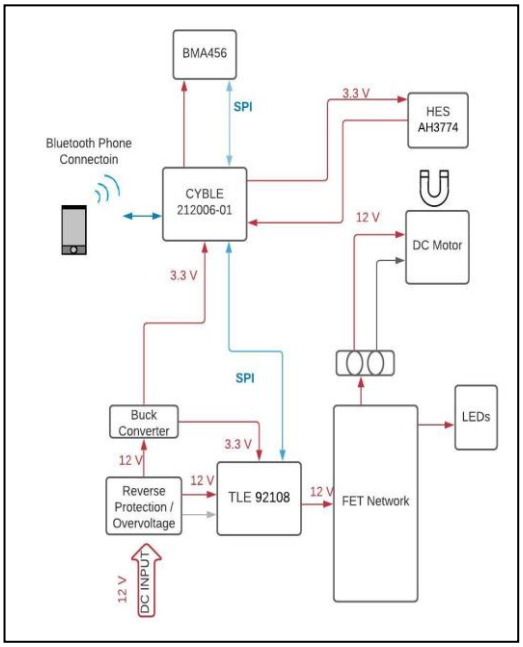

Fig. 1. Block diagram depicting the basic connections between system components. The blue lines represent SPI communication, and the red lines represent voltage.

III. Hardware Design

A. Motor Driver

As mentioned earlier, The TLE driver can drive up to eight half-bridges. Half bridges one and two are used to control the motor through extend and retract signals from the 6-position switch connector. Half bridge 3 is used for the brake signal. Half bridges 4-8 are used for controlling the external LEDs. The driver has three PWM inputs. The three PWM signals are designated for the LEDs. PWM1 for green, PWM2 for blue, and PWM3 for red. Turning the three PWMs together is would be the white LED signal. A different combination of the PWM signals will result in different color. Current from DH (which is the drain high current supply) comes through the first half bridge and passes through the motor and it then passes by the N-channel of the second half-bridge as seen in Fig. 2. A 1 mΩ sense resistor is added to measure the current passing through the motor. Two current sense inputs CSIPX and CSINX from Fig. 2 are inputs to the TLE that are connected to an instrumental amplifier. A shunt circuit is added between the sense resistor and CSIPX and CSINX to protect the TLE from any high power. The current measurement is then given to the microcontroller through a RC low-pass filter. Since the microcontroller sampling rate is around 1000 Hz, we picked R to be 1.5kΩ and C to be 0.1µF.

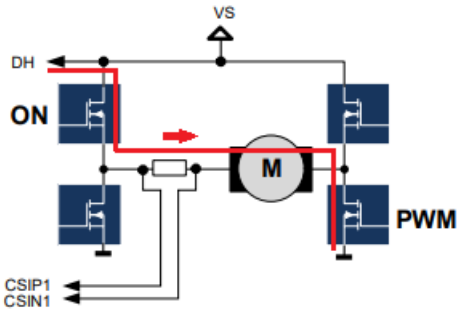

Fig. 2 Current flow through the motor from half-bridge one to half-bridge two.

B. Protection Circuit

Our supply voltage range is between 9 V to 20 V. Our typical supply voltage is 12V. The MAX16141A part is a diode controller that functions as reverse-voltage protection circuit, over-voltage protection circuit and under-voltage protection circuit. If the voltage exceeds our over-voltage threshold, falls below our under-voltage threshold, the part shuts down. The circuit shuts down based on the threshold that we decide. The part has a window-detection threshold comparators, so by using a voltage divider circuit, we can set the values of the resistors, so that when the voltage change beyond the desired thresholds, the N-Channel FETs will turn off. To find the values of the resistors of the voltage-divider circuit, we used equations 1-5.

$$V_{UVTH} = \left(V_{TH} - V_{TH\_HYS}\right)\left[\frac{R_{TOTAL}}{R2+R3}\right], \quad (1)$$

$$V_{OVTH} = \left(V_{TH}\right)\left[\frac{R_{TOTAL}}{R3}\right], \quad (2)$$

$$R3 = \left(\frac{V_{TH} \times R_{TOTAL}}{V_{OVTH}}\right) \quad (3)$$

$$R2 = \left(\frac{\left(V_{TH} - V_{TH\_HYS}\right) \times R_{TOTAL}}{V_{UVTH}}\right) - R3 \quad (4)$$

$$R2 = R_{TOTAL} - R2 - R3 \quad (5)$$

Setting R_total = 100kΩ, get R1 = 95.3kΩ, R2 = 2.8kΩ, and R3 = 2.5kΩ. Low-power mode (SLEEP) is active high. So, we connected SLEEP to a 10kΩ pull-down resistor to ground. In addition, pins RS and OUT are connected to a comparator inside the module. A sense resistor is connected between RS and OUT which compares the current and detects any overcurrent. If any overcurrent is detected, then the module shuts off. The current threshold is calculated using equation 6.

$$I_{OC} = V_{(RS-OUT)}/R_{SENSE} \quad (6)$$

Where V_{RS-OUT} is the voltage of the overcurrent threshold and R_{SENSE} is the sense resistor between pins RS and OUT. I_{OS} = 27.5 A, V_{RS-OUT} is 27.5 mV (which is the maximum V_{RS-OUT} that the module can handle), and the R_{SENSE} = 1 mΩ. Since the motor is rated at around 300 W (12V*25A), then 27.5A is an appropriate overcurrent threshold. For reverse-voltage/reverse-current protection, the module measures the different between V_{IN} and V_{OUT} across the external FETs. If $V_{OUT} - V_{IN}$ = 10mV, then the module activates a fault that turns the gate voltage to zero as well as load current. Fig. 3 is a graph of the effect of a reverse-voltage fault.

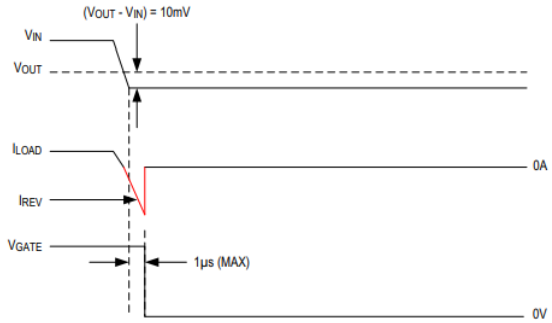

Fig. 3 Reverse voltage fault response.

Connected to the protection circuit is a hold-up circuit which is a circuit of three capacitors in parallel; two of them are ceramic and one is a 100µF electrolytic capacitor that functions as a reservoir. The hold-up circuit is then connected to a 12V-3.3V and a 12V-5V linear voltage regulators. The small signal ICs have a nominal voltage of 3.3V, and the 5V regulator is for the external LEDs. The 5V linear regulator is enabled by the microcontroller through an enable circuit that we designed. The circuit is mainly consisted of a P-channel MOSFET and a N-channel MOSFET which has its source connected to ground. A simulation of the enable circuit for the 5V regulator is shown in Fig. 4.

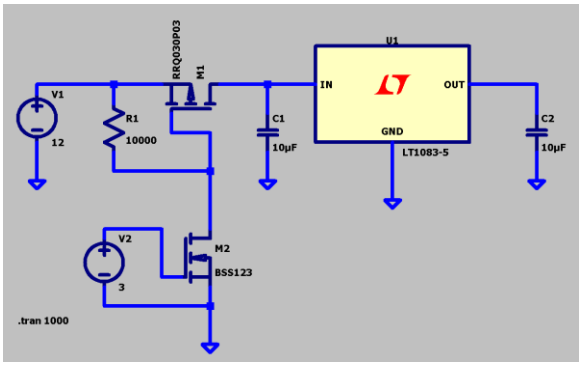

Fig. 4 Enable circuit simulation for the 5V linear regulator.

Simulating the circuit above when V2 (which is the enable signal coming from the microcontroller and the gate-to-source voltage of the N-channel MOSFET) is zero, the voltage at the input of the regulator is 0V as shown in Fig. 5. When V2 is enough of a voltage, the voltage at the input of the regulator becomes close to 12V as shown in Fig. 6. Green is regulator input and blue is the 12V supply.

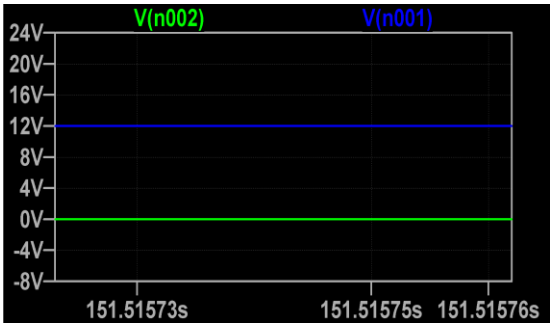

Fig. 5 Enable circuit when V_{GS} of the N-channel = 0V.

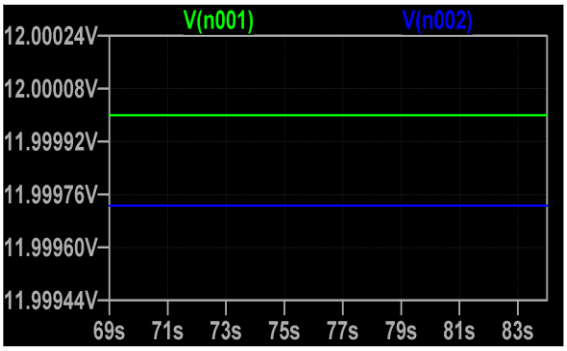

Fig. 6 Enable circuit when V_{GS} of the N-channel = 3V.

C. Connectors.

As for the connectors, we have a 6-position connector for the switches: LIGHT_1, LIGHT_2, EXTEND, RETRACT, and BRAKE. LIGHT_1 and LIGHT_2 are for the LEDs. When the switch is turned on, the signal on the microcontroller is shorted to ground. Since the GPIO for the microcontroller are active low, then the microcontroller gives the signal to turn on the LEDs. We decided to use five signal vertical connectors for external LEDs to be connected: RED, GREEN, BLUE, WHITE, and WORK LIGHT. External LEDs are controlled by the MOSFET driver which gets its signal from the microcontroller. As for the power supply and the motor, we decided to use single horizontal connectors. To each connector, we connected a transient voltage suppressor to prevent any spikes in voltages. We designed a protection circuit to every switch signal so protect the microcontroller. The circuit is consisted of a transient voltage suppressor, a Shockley diode, a pull-up resistor, and a series resistor. Fig. 7 shows the protection circuit for the switch signal EXTEND.

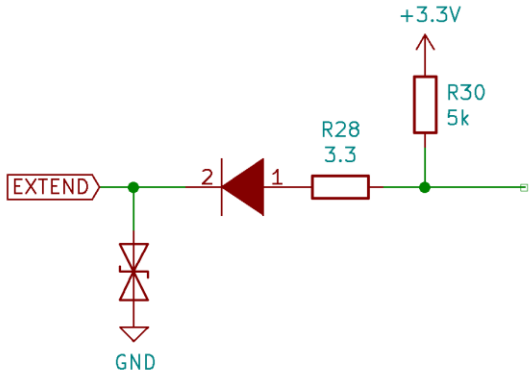

Fig. 7 Protection circuit for switch signals.

D. PCB Design

After measuring the dimensions of the jack and the board-mounting place, we designed the PCB to have a length of 16 cm, and as for the width; the upper part

of the PCB has a width of 7 cm and 6 cm down from the top of the PCB, the width is increased to 8 cm. The PCB is shown in Fig. 8. We decided to design a 4-layer board. The top and bottom layers are for power traces. And the second and third layers are for signal traces. We filled the top and bottom layer with ground. As for the signal ground we designated the bottom left area of the top layer (where the radio is located) to be signal ground. There are three 5-mm screw holes. One at the top left, one is to the right, and one is near the bottom. We designed the board so that the power supply would be connected to the bottom right of the board. There is a drill hole at the bottom right and an edge cut for the zip tie to hold tight the three high current wires, which are SH1, the source current for half-bridge 1, SH2, the source current for half-bridge 2, and the 12V supply. Power traces have a track width of 3 mm. However, we avoided connecting the high-power components with traces; instead, we created different zones for the different high-power components such as the 12V voltage supply, SH_1, and DH_1 which are the drain-high and the source-high inputs for the half-bridge supplying the motor. Each zone on the front copper has its identical sister on the back copper and they are connected through vias. We placed stitching vias all over the zones for thermal relief.

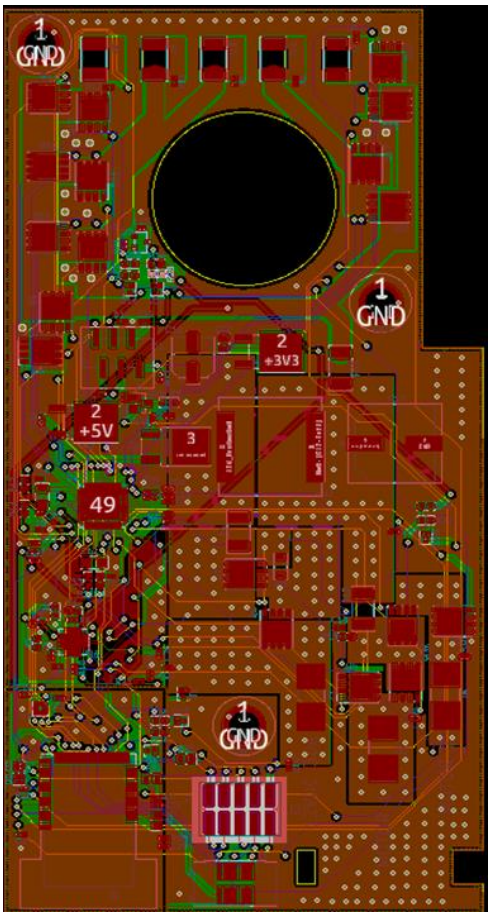

Fig. 8 PCB Design Layout

We tried to place the TLE as close to the middle of the board as possible, so it is close to the FETs at the top and the radio at the bottom. We placed the radio in the very bottom as far from the high current and the motor as possible to avoid radio interference. The MAX part is placed between the TLE and the radio. The six-position connector is connected above the TLE and the LEDs connectors are placed at the top of the board close enough to the FETs that are assigned to control the LEDs. We have the debug connectors placed at the bottom of the board close enough to the radio. We also have the hall effect sensor located as close to the motor as possible.

E. Design Complications

We designed a protection circuit to every switch signal so protect the microcontroller. The circuit is consisted of a transient voltage suppressor, a diode, a pull-up resistor, and a series resistor. When the switch is turned on, we get a voltage of 2.4V at the anode of the diode which is too high to enable the GPIO which is active low. As a result, switches were not working. We removed the 3.3 Ohms series resistor.

The MAX16141A protection circuit always gave a fault when we tried to start the motor. The part activates a fault if V_{OUT} is less than 90% of V_{IN}. We tracked V_{IN}, V_{OUT} Vin – V_{OUT}, and the fault signal
The fault signal is active low, and it went low as V_{IN}– V_{OUT} was more than 10% of $V_{IN.}$ To solve this issue, we jumped a wire over the protection circuit to the regulator.

IV. Software Design

The system is comprised of several different components that communicate through serial peripheral interface (SPI) communication. The software team developed drivers for each device and integrated them together with flags and interrupts. There are different priorities assigned to the different devices. The TLE-92108 motor driver has the highest priority because it is the most important. The extend and retract flags will be raised if the user sends those commands to the CYBLE-212006-01 module. The main loop constantly checks for the flags and calls the corresponding driver based on the flag. The overall software system design can be seen in Fig. 9 with the flag handler within the main loop.

A. Motor Driver Software Design

The TLE-92108 is a new part from Infineon thus it does not yet have software support. This meant that we had write our own driver for it from scratch. The goal was to write a driver that is agnostic enough to be able to work on multiple projects not just ours. This meant that we had to structure the files for the driver from simple defines for the different Hex bytes in one file to functions that take those values and build and send the command packets in another file to functions that state the action that we want to perform and call all the functions necessary in the other files to configure the TLE to make that action happen.

There are two ways to control our Super Brute jack, you can use the physical switches in the jack or you can you the mobile app, thus there are two ways to set the flags that start the different functions of the device. We programmed the CYBLE microcontroller so that it will always give the physical switches priority over the commands issued through the BLE.

One of the seemingly simple parts of the project that turned out much more complicated is controlling the RGB lights through the TLE. The TLE has many safety features that detect overcurrent, overvoltage, drain to source overvoltage, etc. this created a challenge for us as setting the PWM signals that controlled the lights too low cause the TLE to throw a voltage error. This meant that we had to configure some of the TLE's safety feature blanking times to the highest values and turn the PWM channels to stay in a certain range which fixed the issue.

B. Hall Effect Sensor Software Design

The hall effect sensor triggers an interrupt every time the magnet passes by the device. This interrupt sets a flag and adds a count to the pulse counter. The the hall effect sensor's flag check has several different functions within it. When the flag is raised, the direction and speed functions are called, and the data is returned to the main function. There is a pulse timer flag that is set within the watchdog timer's interrupt service routine. This timer starts when the first pulse is received and keeps track of how long the motor has been in motion. The revolution timer is also tied to this timer, but it resets every time there are two pulses. Two pulses signify a full revolution of the motor. By calculating the length of each pulse, the team was able to determine if the motor was spinning clockwise or counterclockwise. Another function determines the speed after each full revolution. If the speed is too fast or the motor is spinning in the wrong direction, an error will alert the user of a problem and possibly stop the motor with the emergency brake.

C. Accelerometer Software Design

The accelerometer triggers two different interrupts in the Cypress Bluetooth module. The new data interrupt triggers every 0.04 seconds to provide a constant supply of data. This triggers the new data flag for the main loop to check. The motion interrupt only triggers when the accelerometer senses motion in the system. When this is triggered, we set a motion

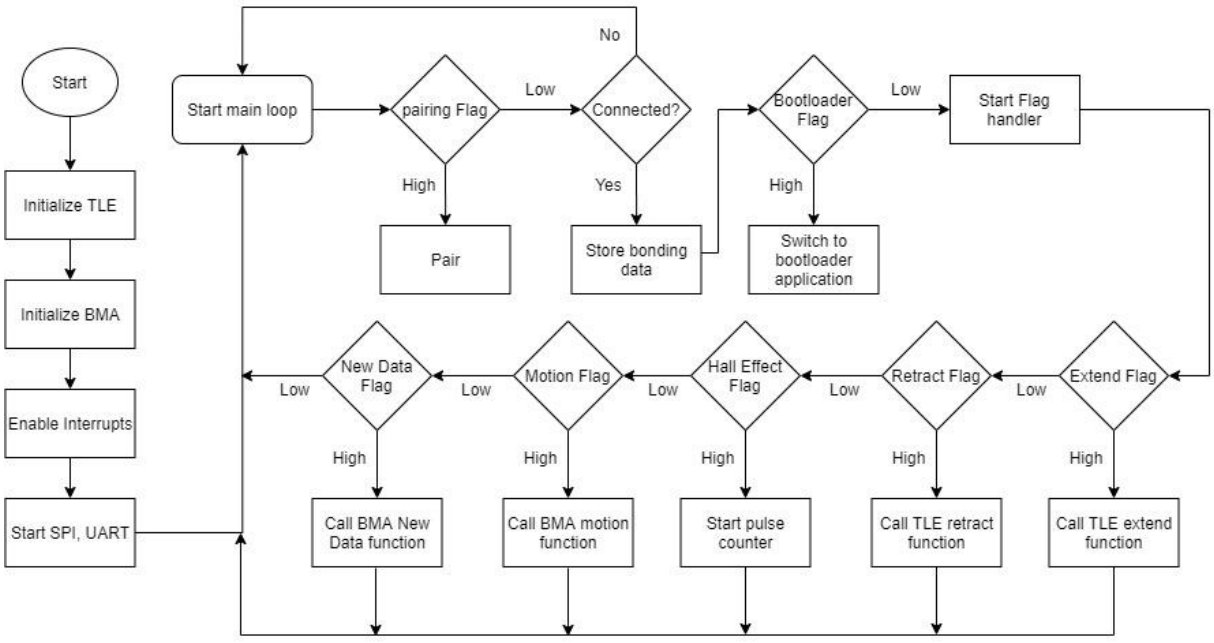

Fig. 9. Flowchart depicting the main flag handler

flag that the main loop checks. The motion function is called from the main loop and determines if the motion is great enough to send an error message to the user or stop the motor completely.

D. Bluetooth Low Energy

Once the software team wrote the drivers and the code worked with the switches, the Bluetooth Low Energy was added to the project. To implement the Bluetooth connection, the team used Cypress's PSOC Creator program because there are tools available to generate Bluetooth low energy code. The GATT protocol was used to send data back and forth between the phone and the PCB. Cypress has an app for Bluetooth design to send commands to the Cypress Bluetooth module. We used this to test our Bluetooth code and make sure that our connection works. Our code is compatible with the Husky Jack mobile application. Commands are sent to the stack from the user, we read these commands and perform the corresponding action. Using Bluetooth, the user can extend, retract, and stop the jack. For additional features, the user can also turn on and off the work light for their workstation and colorful RGB LEDs. The user can send in different color commands to have customized LEDs light up the outside of the RV.

E. Challenges Faced in Software Design

One problem that the software team faced is the SPI clock phase and polarity of the devices. The TLE-92108 requires CPHA to be 1 and CPOL to be 0. The data is driven on the rising edge of the clock and captured on the falling edge of the clock, with SCLK idling at low. On the other hand, the BMA456 accelerometer only works with SPI clock phase and polarity set to 0 and 0 or 1 and 1. Both of these configurations have the data driven on the falling edge and captured on the rising edge. Since these components require opposite SPI configurations, the team had to develop code within the program to change the clock polarity every time communication switches between the TLE-92108 and the BMA456. Due to this inconsistency and the priority of the motor driver, the interrupts sent from the BMA456 cannot always be read. This is not a major problem because the motor does not need constant orientation data. If there is a motion interrupt the motion flag will be raised, the SPI configuration will change, and the orientation data can be read. For our system, the initial SPI configuration is set for the TLE-92108 motor driver and switched in the code when necessary to accommodate the BMA456.

Another challenge faced by the software team was the documentation for the accelerometer. The datasheet did not provide information on certain registers and some values were incompatible with the BMA456. The accelerometer initialization requires a configuration file to be written to certain registers. To do this, paging was used with register 0x5E. In the datasheet, there is no mention of paging and the paging registers 0x5B and 0x5C were not included in the datasheet at all. The software team had to understand all the code in the API to write the driver needed. The API is written for all the BMA4 accelerometers, so some of the code was not needed for our specific accelerometer. Additionally, the datasheet did not correctly document the bits for the interrupt mapping. The interrupts were mapped wrong at first and the team had to use a logic analyzer to determine that the interrupts were not triggering. For weeks, the interrupts were being triggered by noise and the team incorrectly assumed the interrupts were mapped correctly. This taught the team to always use a logic analyzer to make sure that the signals are being sent correctly.

The logic analyzer was also essential for the SPI communication. The API required the team to write SPI bus read and bus write functions that are compatible with their function calls. The original SPI functions worked correctly without the Bosch API, but when the functions were called from the API there were errors. The team worked hard to figure out why, and eventually found out that there were two dummy bytes needed in the BMA SPI protocol. However, the API discards one dummy byte. To solve this problem, the team had to discard one dummy byte and return the data read array with a dummy byte in index position 0. This setback took a long time to figure out, but it provided a great learning experience for embedded coding. The team had previously thought that one dummy byte was used in SPI communication, but we learned that there can be more than one. It is important to understand the different components completely to program them correctly. Throughout this project, we have learned the valuable skill of reading and understanding datasheets. This is very important because every component is different, and we will need to read datasheets for the rest of our careers. The challenges experienced in this project taught us valuable information that we will always remember.

V. Conclusion

The overall design of the system required the student engineers to develop both hardware and software skillsets. The drivers were difficult to complete without a strong programmer on the team, but it provided an opportunity for everyone to learn more about embedded systems. Learning patience and approaching the problem systematically is how we were able to find solutions to the various difficulties in the project. Developing the schematic and PCB layout was very tedious but also a great learning experience for everyone. After the board was developed, a few errors were found. These errors provided the biggest learning experiences. Having these problems showed us how to find errors within a board and how to solve them. This also gave us more soldering experience. Overall, this project was a great way to develop new skills.

Acknowledgement

The authors wish to acknowledge the assistance and support of our sponsor, RV Intelligence, for their continuous help in the production of this product. The authors would also like to acknowledge the customer, Husky Towing Products.

References

Incorporated, Diodes. “AH3774 Datasheet.” High-Voltage High Sensitivity Hall Effect Latch, 2015, www.diodes.com/assets/Datasheets/AH3774.pdf.

Sensortec, Bosch. “BMA456 Datasheet.” BMA Digital, Triaxial, Acceleration Sensor, 2017, www.mouser.com/datasheet/2/783/BST-BMA456-DS000-1509567.pdf.

Technologies, Infineon. “Cypress CYBLE-212006-01.” Cypress Bluetooth Module, Infineon, 2019, www.cypress.com/file/318881/download.

Technologies, Infineon. “TLE-92108 MOSFET Driver.” Multiple MOSFET Driver IC, 2019, www.infineon.com/dgdl/Infineon-TLE92108-232QX-DataSheet-v01_00-EN.pdf?fileId=5546d462749a7c2d01749b3138d607ed.

Biography

Lucy Golebiewski is a senior at the University of Central Florida. She will receive her Bachelor of Science in Computer Engineering with a minor in secure computing and networks in December of 2020. She is currently an intern at Capacitech Energy, researching and manufacturing cable-based capacitors. In the future, she plans to work on renewable energy or remote sensing technology to help benefit the environment.

Hesham Zidan is a senior at the University of Central Florida receiving his Bachelor of Science in Electrical Engineering in the fall of 2020. He has done research in distribution system optimization. Post-graduation, he plans to work as substation designer and system planner for large scale power networks.

Andrew Melvin is a senior at the University of Central Florida. He will be receiving his Bachelor of Science in Electrical Engineering with a double minor in Physics and Robotics at the end of Spring 2021. Through his time in the Air Force as well as working on various side projects, he has accumulated a plethora of experience on mechanical and electrical systems. His ultimate career goal is to work in the robotics field on medical robotics and fully integrated prosthetics.

Nader Abd El Rasol is a senior at the University of Central Florida. He will receive his Bachelor of Science in Computer Engineering with a minor in Intelligent Robotic Systems in December of 2020. He is currently working part-time at wholefoods market winter park, His goal is to use what he learned working on all the different projects he’s worked on over the years to begin a career in robotics and automation system R&D.