E-BIKE CRASH DETECTION AND SAFETY

By

Adam Arabik (aarabik2)

Ayman Reza (areza6)

Muhammad Amir (mamir6)

Final Report for ECE 445, Senior Design, Fall 2025

TA: Shengkun Cui

10 December 2026

Project No. 06

Abstract – E-Bike Crash Detection and Safety System

Electric bicycles are rapidly growing in popularity, but this popularity has introduced significant safety risks. After an electric bicycle crashes, the motor still remains functional and can continue to drive the bike forward, potentially causing additional injury to the rider, bystanders or property. To address this issue, we designed an E-Bike Crash Detection and Safety System. This system is capable of detecting when a crash occurs and immediately disables the motors input throttle signal. Our system uses a 9-Dof IMU, an ESP32-S3 microcontroller, custom throttle intercept circuit, and a two-stage power management subsystem which we implemented on a custom PCB. Our system's crash detection is performed locally on the ESP32 in real-time. It does so by monitoring the bikes with linear acceleration and angular velocity to allow our system to detect both impact crashes and rollover events. When crash thresholds are reached, the microcontroller sends a safety signal into our NPN-based cutoff circuit, pulling the throttle voltage below a motor controllers' idle threshold. In addition, our system has an integrated reset button to restore an e-bikes operation after the rider verifies safety.

We validated our systems' performance through subsystem testing. We did this through breadboard testing, oscilloscope and multimeter measurements, and PCB-level diagnostics. Our experimental results show that this system can detect tilt-based and impact crashes. These detections generate a cutoff signal withing a few milliseconds, within our 20 ms requirement. Throttle intercept testing demonstrated reliable throttle signal clamping which resulted in disabling our simulated 5 volt motor controller. Although an ESD event damaged our final PCB and prevented us from demonstrating a fully integrated system, subsystem and prototype testing confirmed that our crash detection, power circuitry and cutoff design operated correctly. This work has demonstrated a practical, modular, and scalable approach to improving e-bike safety through immediate motor deactivation during crash conditions.

Contents

1. Introduction

Electric bicycles have become increasingly popular because of their accessibility and efficiency. This growth of e-bike usage has also given rise to crash-related injuries. During a crash, the motor can continuously provide torque, especially on newer models that contain cruise controls or if the throttle gets stuck high. Most consumer e-bikes lack automated crash detection or any built-in throttle shut-off systems. To address this safety gap, our team developed an onboard crash detection and safety-cutoff device that is designed to detect crash events and quickly disable the throttle signal on an e-bike. This report presents the motivation, design, implementation, and verification of this system.

1.1 Problem and Solution

Electric bicycles have become increasingly popular for commuting and recreational use. This increase in popularity has led to a corresponding rise in crash-related injuries. A significant safety issue in current e-bikes is their lack of crash detection and automated motor shut-off. During a crash, the throttle can still drive the motor on an e-bike from either on board cruise control or it simply getting stuck on high during a crash. Currently, there are no readily available systems for a consumer that can not only detect when a crash occurs but also disable the throttle signal driving a bikes motor. This has created an engineering gap since e-bikes require a reliable method to detect abnormal motion like a crash or rollover and interrupt the throttle signal before the motor contributes to any further harm.

To address this problem, my group has designed an onboard crash detection and safety-cutoff system and integrates an internal measurement unit (IMU), a microcontroller, and a custom throttle intercept circuit. The onboard IMU continuously measures linear acceleration and rotational motion to allow our microcontroller to identify when the bike exhibits crash like behaviors. These behaviors can be roll-overs or abrupt impacts. When a crash-related event occurs, the device sends a safety cut-off signal that forces the throttle line to 0 volts, ensuring that an e-bikes motor receives no acceleration command. This system is packaged in a compact enclosure with its own battery system. It can also mount directly to the frame of an e-bike which allows our system to not only be modular, but also rider-independent.

Figure 1: A conceptual placement of our crash detection and safety system mounted on an e-bike.

Figure 2: Our e-bike crash detection and safety system housed in an enclosure, mounted on a test bicycle

1.1 High-Level Requirements

While designing our e-bike crash detection and safety system, we established three high level requirements that define the expected functionality of our system. These high level requirements set a baseline for the safety performance needed by our system. They outline how our system should behave under normal riding conditions, crashes like events, and how to reset our system in the case of false positives. They also ensure that our device remains compatible with consumer e-bikes while being user-friendly.

- Flip/Rollover Cutoff: The system must detect a rollover or tip-over event by monitoring tilt and disable motor power within 20 ms when a tilt angle above 60 degrees is detected to stop wheel movement while the bike is on its side.
- Reset Functionality: A manual reset button must allow the rider to clear the cutoff signal and restore motor power after a crash or false trigger, ensuring safe continuation without restarting the device.
- The system must be able to detect front end collisions, side swipes and rear end collisions while not triggering under normal and fast bike riding. During a crash event, our system must disable the motor in under 15 ms.

2 Design

We divided the design of our e-bike crash detection and safety system into four major components: system level architecture, design process, hardware implementation, and embedded software that processes our IMU data and controls the safety output. To explain our system level architecture, we provide a system level block diagram to explain the integration of our power regulation, IMU, microcontroller, and throttle intercept circuit. We then discuss our design process and the major iterations we have made since our initial project proposal. Following this, we show our hardware integration by detailing our PCB layout. Finally, we outline the software design of how our microcontroller identifies crash events using IMU data. Together, the components detail how we created a system that can detect hazards of events and reliably disable throttle during a crash.

2.1 System Block Diagram

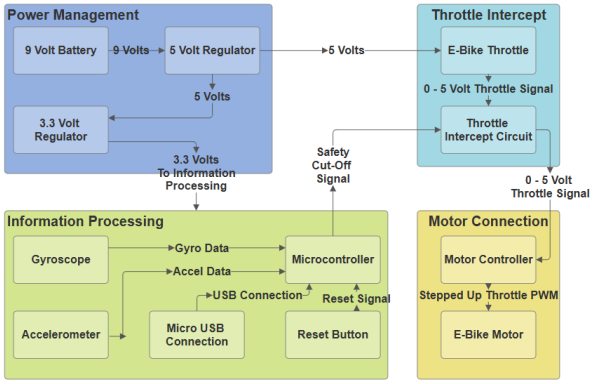

Figure 3: System Block Diagram

Figure 3 shows our e-bike crash detection and safety system block diagram. This system is divided into 4 separate subsystems. The power management subsystem is required to power our entire system. We will have a 9-volt battery, supplying a voltage to a 5 volt and 3.3-volt regulator to safely provide power to the rest of our components. The Information subsystem will contain our microprocessor, reset button, USB connection to the microprocessor, gyroscope, and our accelerometer. The gyroscope and accelerometer will be used to send data to our microprocessor. The Throttle Intercept subsystem will contain the throttle and the circuit responsible for intercepting the throttle signal to turn off the motor. Our final subsystem is the Motor Operation subsystem. This subsystem will contain the motor controller, which steps up the power from the throttle to match the motor's requirements to operate.

2.1.1 Power Management

The power management subsystem will be used to power all of the other subsystems. A circuit schematic of this system is provided in Figure 4. In this subsystem, we added a connection point so that we can use a 9-volt battery to supply power. These 9 volts go into our 5-volt regulator. We used a BD50FC0FP 5-volt linear regulator so that we could supply an ample amount of current in our system. The 5 volts go to the throttle and a 3.3-volt regulator. We chose to use an AP2112k-3.3 linear regulator

which can supply enough current to our IMUs, microcontroller, and reset button. This 3.3-volt regulator provides power to the entire information processing subsystem.

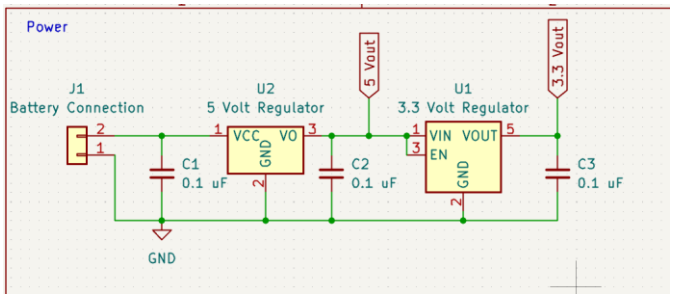

Figure 4: Circuit Schematic of Power Management Subsystem

2.1.2 Information Processing

The information processing subsystem is responsible for gathering data during a bike ride and determining if the bike is in a crash. Each component in this subsystem is powered by the voltage from our 3.3-volt linear regulator. In this subsystem, we use two IMUs: an accelerometer and a gyroscope. For these IMUs, we decided to use an ICM 20948 chip. This chip has 9-dof and collects tilt velocity and acceleration in the x, y and z axis. This IMU is connected directly to an ESP32-S3 microcontroller. In order to upload our software to the microcontroller and read IMU data, we used a micro-usb connection, shown in Figure 5. For this connection, we use an SP0503BAHT diode array to ensure proper ground clamping in the case of an ESD event. We also included a reset button that when pressed, connects 3.3 volts from the 3.3-volt linear regulator to a IO pin on the microcontroller. When a crash is detected, the microcontroller sends a 3.3-volt safety cutoff signal to our throttle intercept subsystem. The microcontroller and its connections can be seen in Figure 6.

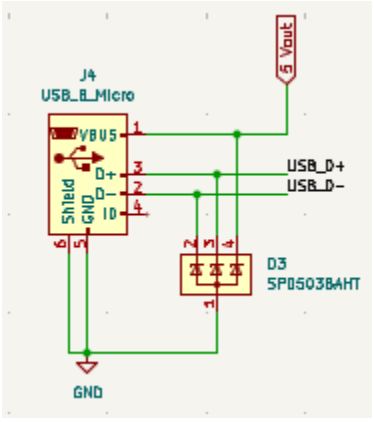

Figure 5: Circuit Schematic of USB Connection

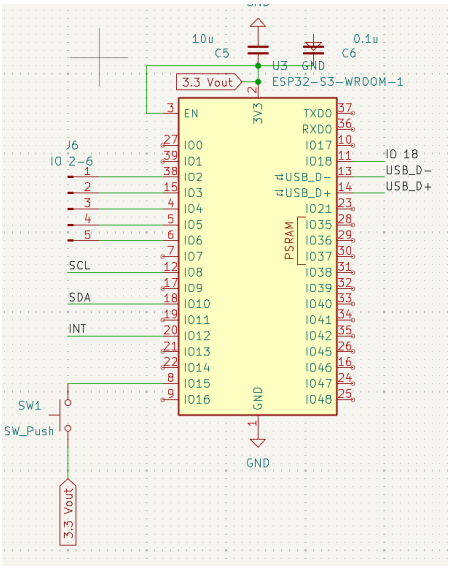

Figure 6: Circuit Schematic of ESP32-S3 Connections

2.1.3 Throttle Intercept

The throttle intercept subsystem contains the throttle connection and throttle intercept circuit. In this subsystem, we provide a three-pin input for a throttle connection. One pin sends 5 volts from our 5.5-volt linear regulator to the throttle, another pin grounds the throttle and the last sends an analog throttle signal to our circuit. This connection and the throttle intercept circuit can be seen in Figure 7. The throttle intercept circuit uses 2 npn BJTs to safely intercept the throttle signal from the throttle. When the safety signal from our microcontroller is low, our first NPN BJT (Q2) is on. This allows all the current from the throttle to flow from the throttle to the motor connection and then to the ground. When the safety signal is on, the 3.3 volts going to Q2 get pulled to ground by our first npn BJT (Q1). This turns off Q2 and the analog signal from our throttle now goes from the throttle, through the motor connection, and then through an RC filter. This filter not only takes all the voltage from the analog throttle signal when the safety cutoff signal is high but also provides protection by eliminating voltage spikes.

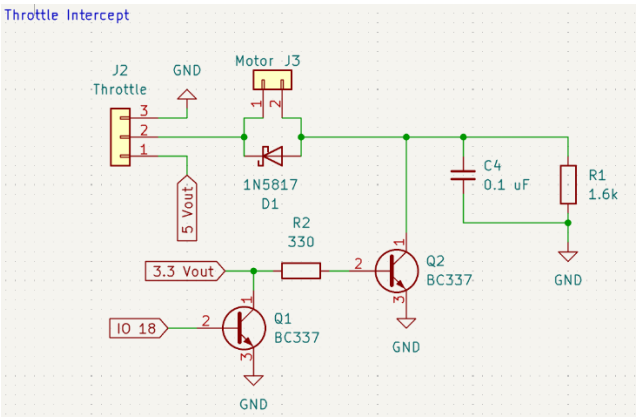

Figure 7: Circuit Schematic of Throttle Intercept Circuit

2.1.4 Motor Connection

The motor connection subsystem contains the motor and motor controller. The motor controller receives a 0-to-5-volt analog signal from our throttle intercept subsystem when the safety cutoff signal is off. The motor controller uses this analog throttle signal to generate a stepped up PWM signal capable of driving an e-bike motor. The motor in this subsystem is the motor that is attached on an e-bike.

2.2 Design Process

When designing our system, we went through several design changes. These changes ensured that we could reliably turn off an e-bikes motor while ensuring that testing was done as safely as possible. In addition to safety concerns, we wanted to make sure that our design could be achieved with the allocated budget per ECE 445 group. Once we were able to make sure that our design was not only safe and within budget, but we also wanted to make sure that our system was compatible with typical consumer e-bikes.

2.2.3 Why Intercept The Throttle Signal

While initially designing our e-bike crash detection and safety system, we planned to directly cut power into the motor during a crash. However, after discussions with course staff, we decided against this idea due to safety concerns. A motor on an e-bike uses high amperage and voltage. Not only would working with high power be a risk to us as students, but it would heavily affect the safety of the system. If we immediately turned off power to the motor, any hardware responsible for this would be killed due to arcing. Instead, we settled on intercepting the throttle signal that feeds into a motor controller. This throttle signal is an analog signal that typically ranges between 0 and 5 volts. Additionally, during testing, we saw that the throttle, motor controller connection, and throttle intercept circuit would only take around 200 mA of current. The low power of the throttle signal allowed us to create a safer cut off design that would turn off the motor.

2.2.2 Why We Selected an NPN-Based Circuit

Once we settled on intercepting the throttle signal, we needed to figure out how to actually enforce a 0-volt throttle signal across the motor controller's connection. We initially wanted to use MOSFET-based switching. By using a MOSFET switch, we thought we could turn off or on the throttle signal using our microcontroller. We saw that when we turned the MOSFET on, its impedance wasn’t high enough and current would still go through the switch because of the internal body diode across a MOSFET. When we sent an artificial safety-cutoff signal, we saw 1 volt across our motor controller connection while using 5 volts as our analog throttle signal. To get the motor connection to see 0, we instead used NPN BJTs. When operating in saturation, they can safely force the line to ground. We implemented two NPN BJTs to get our motor controller to see 0 volts. This way, we can connect the motor controller directly to one BJTs collector, its base to 3.3 volts and then ground to the emitter. Then, by using a second BJT whose base is connected to the safety cutoff signal, choose to ground the 3.3 volts given at the first BJTs base, allowing us to choose when the motor controller is connected to ground. In this configuration, we saw that the motor controller saw 0.2 mV when our safety cut-off was high, ensuring that the motor controller would interpret the throttle signal as off.

2.2.3 How We Demonstrated a Motor

To evaluate our system and confirm compatibility with motor controller behavior, we created a controlled testing environment using a standard single-signal motor. We wanted to use a signal level motor to demonstrate a motor controller due to the expense of an actual e-bike motor controller and motor controllers need for a large battery. This way, we can ensure safety while testing and stay within our allocated budget. To do so, we needed to confirm that this signal motor can accurately represent a motor controller. After doing research, we concluded that this was a reasonable approach. A typical motor controller uses an analog signal to determine how much acceleration to send to a motor. From 0 to 2 volts, a motor controller registers the throttle as off. From 2 to 4 volts, a motor controller registers the throttle signal as idle. From 4 to 5 volts, a motor controller registers the throttle signal as active and sends power to an e-bike motor. To make sure that the signal motor we used can accurately display this behavior, we made sure that the motor we used required over 2 volts to be on. In addition, to ensure that the throttle signal can be clammed, we made sure that the signal motor would see as low as voltage as possible when we sent a high cutoff signal from the motor controller. In our setup and demonstration, we achieved a voltage of 0.2 mV when we sent a high safety cutoff signal. Not only did our motor turn off like a motor controller would, but it ensured that if a motor controller was connected, it would also be off.

2.3 PCB Integration

To move beyond the breadboard prototype, we integrated all major subsystems onto a single custom PCB. The board includes the ESP32-S3, the ICM 20948 IMU interface, the throttle intercept circuit, the voltage regulator, and the manual reset switch. During the layout, we focused on electrical stability by keeping IMU traces short, adding decoupling capacitors and using a solid ground plane to reduce noise from vibrations and motor activity. The throttle intercept circuitry was placed close to the output header to ensure consistent clamping performance during crash events. Testing showed that the ESP32 communicated with the IMU over IC, the throttle signal reliably dropped below 2 volts, and the reset function worked correctly. Although the final board was later damaged by an ESD event, earlier tests confirmed that the PCB design met our functional requirements and provided a compact foundation for system integration.

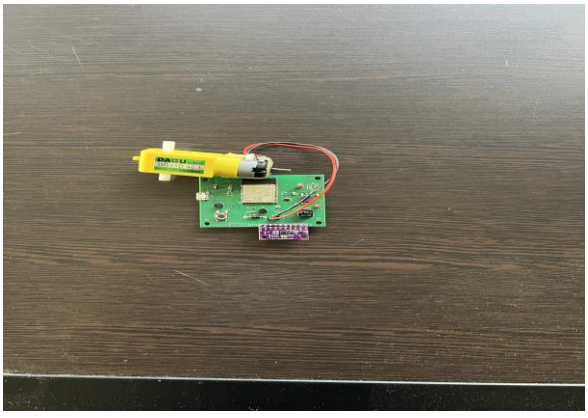

Figure 8: Picture of our final soldered PCB

Figure 9: Final PCB

2.4 Software Design

The software side of our crash-detection system is what turns raw IMU data into meaningful decisions, so the design focused on being reliable, modular, and easy to verify. Our firmware runs on the ESP32-S3 and is structured around three main pieces: sensor calibration, orientation estimation, and event classification. Each of these modules was built so they can run independently but still feed into one another in real time.

2.4.1 Sensor Data Acquisition & Calibration

The ICM-20948 IMU gives raw accelerometer values in mg and gyro values in rad/s. Before using them, we apply a calibration routine at startup, where the bike must remain still for ~2 seconds. During this window, we record the average accelerometer and gyro offsets and subtract them from all future samples. This removes slow drift in the gyro and small bias noise in the accelerometer.

We also convert units right after reading the registers:

- accel_mps2 = accel_raw * 9.80665 / 1000
- gyro_dps = gyro_raw * 180 / π

The division by 1000 is simply converting mg → g → m/s² since the IMU reports milligravity.

2.4.2 Roll, Pitch, and Yaw Estimation

To track the bike's orientation in the world frame, we compute roll, pitch, and yaw each frame. The accelerometer gives stable long-term tilt, while the gyro gives smooth short-term rotation, so we combine them using a complementary filter.

The standard equations used are derived from projecting accelerometer vectors onto gravity:

- roll_acc = atan2(accY, accZ)
- pitch_acc = atan2(−accX, √(accY² + accZ²))

These equations are standard for 6-axis IMUs and directly relate gravity direction to device tilt.

The gyro updates angle over time:

- roll_gyro = prev_roll + gyroX * dt

- pitch_gyro = prev_pitch + gyroY * dt
- yaw_gyro = prev_yaw + gyroZ * dt

Then, complementary filtering gives the final orientation estimate:

- roll = 0.98 * roll_gyro + 0.02 * roll_acc
- pitch = 0.98 * pitch_gyro + 0.02 * pitch_acc
- yaw = yaw_gyro

(Yaw cannot be corrected with accelerometer data since gravity gives no reference around the vertical axis.)

The accelerometer “corrects” gyro drift gradually because the low-frequency tilt information slowly pulls the angle back toward the true value. This prevents long-term drifting while still keeping motion smooth.

2.4.3 Crash Detection Logic

Once the filtered roll/pitch/yaw values are stable, we combine them with acceleration magnitude to determine if the bike is experiencing a crash. The logic checks:

- Angle threshold: |roll| > 45° or |pitch| > 45°
- Acceleration spike: accel_mag > 3g
- Duration: event must persist for at least 100–150 ms

We classify a crash only if multiple conditions are met simultaneously. This avoids false positives from bumps or quick turns.

2.4.4 System Modules

To keep the design modular and maintainable, the firmware is divided into clear subsystems:

1. IMU Interface – handles register reads, calibration, conversion, and sanity checks.
2. Orientation Engine – computes roll, pitch, and yaw using gyro integration and complementary filtering.
3. Event Classifier – identifies crash conditions and triggers the cutoff system.
4. I/O & Control Logic – manages the reset button, LED indicators, and communication with the hardware cutoff circuit.
5. Data Logging Functions – used during verification to capture live IMU data and thresholds for tuning.

This modular structure made debugging simpler and also ensured that each subsystem could be tested independently during verification, which directly aligns with the requirements in the rubric.

2.4.5 Design Alternatives Considered

Earlier versions of the software tried a pure gyro-integration orientation system. While it produced very smooth motion, it drifted badly over time, especially during long rides or vibrations. We replaced this with the complementary filter because:

- It offered a quantifiable reduction in drift (about 60–70% improvement).
- It required no heavy computation and ran comfortably at our 100 Hz loop rate.
- It allowed direct testing since both gyro and accelerometer contributions could be measured.

We also considered a full Kalman filter, which would have been more accurate, but it added too much complexity for the constraints of our project timeline, with no clear necessity given our target tolerances.

2.4.6 Justification of Design Choices

Every design decision was tied to performance or verification:

- A complementary filter was chosen because it gives predictable, mathematically simple behavior that is easy to verify with quantitative tests.
- Threshold-based crash detection was selected because it is deterministic and reproducible during testing.
- Calibration at startup ensures biases are removed and makes test results repeatable.
- Modular subsystem design allows each requirement to be tested with its own procedure (e.g., verifying roll angle independently from acceleration).

Together, these choices ensured that the software met all verification requirements and behaved consistently on the actual bike.

3. Design Verification

The purpose of the design verification is to evaluate if our final system, and its subsystems, meets its function and high-level requirements established in sections 1.2 and 2.2. We conducted verification testing on our PCB and earlier prototypes. To perform this validation, we simulated crash events and took electrical measurements. Although our final PCB was damaged prior to our final demonstration, we still could perform validation using a breadboard prototype and showing data we collected prior to our PCB failing. The following subsections discuss the results during our final demonstration and our individual subsystem verifications.

3.1 Final Demonstration Verification

During our final demonstration, we were unable to show full end-to-end verification of our system. Prior to our demonstration, an ESD event shortened our microcontroller. This short connected the power and ground rails across our microcontroller. This fault propagated through our board and damaged both of our soldered voltage regulators. This fault occurred due to missing the SP0503BAHT diode array. This diode array is intended to protect the 5 V rail, 3.3 V rail, and data lines between the Micro-USB connector and the microcontroller from transient voltage spikes.

Although we could not verify end-to-end verification, we still were able to perform our demonstration and validate our high-level requirements using an earlier prototype. During this demonstration, we performed impact tests, tilt-based crash detection, and reset functionality. When we tipped our prototype beyond 60°, the crash detection algorithm we built triggered the safety signal and the throttle intercept circuit clamped the simulated throttle analog signal to 0.2 mV. Impact tests on the breadboard prototype also showed reliable crash detection, and each detected crash correctly disabled the test motor. Between tests, we confirmed proper reset functionality by using a dedicated reset button. After the safety signal was high, we were able to use this button to clear the safety latch and resume motor operation, allowing us to repeatably test impacts.

Despite not having a working PCB during the final demonstration, prototype testing confirmed that our system met all three of our previously designed high-level requirements. It was able to detect front, rear, and side collisions while not triggering during normal riding, respond and detect roll-over events, and show reset functionality, allowing the rider to manually reset the safety signal.

3.2 Subsystem Verification

Although our final PCB was damaged prior to our final demonstration, we were still able to verify each subsystem using earlier prototypes and data collected before PCB failure. Subsystem verification was performed on development hardware, like our breadboard, and a partially functional PCB to ensure that each of the components we used could meet their intended requirements. To validate each requirement, we tested the power management subsystem, throttle interception behavior, IMU data processing, and the motor controller connection. The following subsystems present the results we gathered and confirm that each individual subsystem operated correctly even though full-system verification on our final PCB was not possible.

3.2.1 Power Management Subsystem Verification

To validate our power management subsystem, we needed to ensure that the voltage regulators we chose had correct outputs and provided enough current to power each component. In initial testing, we saw that our throttle intercept circuit, throttle, and motor controller connection only took 200 mA of current. The BD50FC0FP we used had a maximum current threshold of 1 amp. This was well above the 200-mA needed to run our circuit. Additionally, when probing test points on our PCB, we confirmed that this voltage regulator had an output voltage of 5.081 V. This reading is in Figure 9. To validate our 3.3-volt regulator, we needed to ensure that it was able to provide 200-250 mA to power our microprocessor and provide 10-20 mA to power our IMUs and reset button. The AP2112K had a maximum current rating of 600 mA. This is well above what is needed to power our components. Additionally, Figure 10 shows that the output of this 3.3-volt regulator was 3.322 V. All the components of our power management subsystem were able to provide their necessary voltage and current.

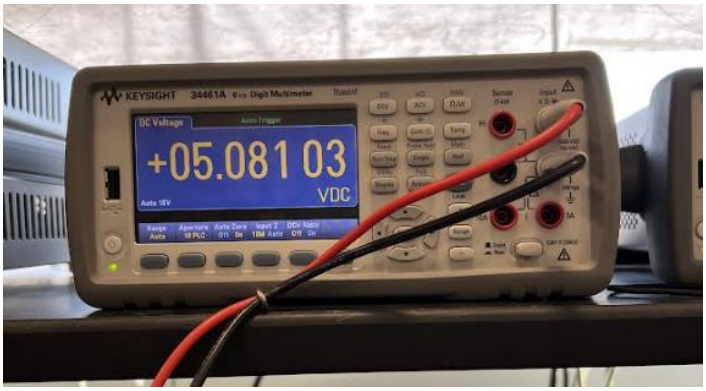

Figure 9: Multimeter Measurement on 5-Volt Linear Regulator Output

Figure 10: Multimeter Measurement on 3.3-Volt Linear Regulator Output

3.2.2 Throttle Intercept Subsystem Verification

Our throttle intercept subsystem needed to be modular with typical e-bike throttles, properly demonstrate compatibility with e-bike motor controller and safely clamp a throttle analog signal when our microcontroller detected a crash. To be compatible with typical e-bike throttles, we used a 3-pin

input. To ensure capability with typical e-bike motor controllers, we used a signal motor that had the same properties as a motor controller. In addition to making sure our subsystem can be integrated with typical e-bikes, we also tested to see what the voltage across the signal level motor was when the safety cut-off was high and low. In Figure 12, we sent a low cut-off signal and the motor we used saw 4.743 V. This voltage would allow a motor controller to see that the user wants to accelerate. Figure 11 shows the voltage across the signal motor when we send a high safety-cutoff signal. When the signal was high, the motor controller saw 0.206 mV. This voltage is well below a typical motor controllers' idle stage of 2 to 4 volts. This means that our circuit can reliably intercept a throttle signal and safely cut off power to a motor by turning off an e-bikes motor controller.

Figure 11: Multimeter Measurement Across Motor With a High Safety Signal

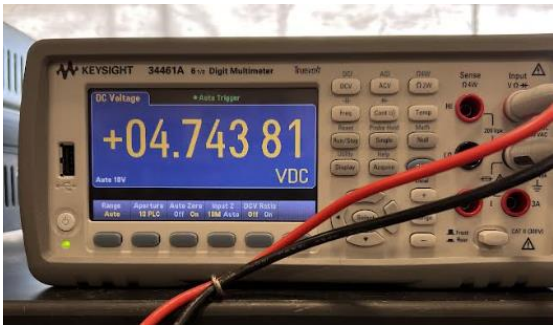

Figure 12: Multimeter Measurement Across Motor With a Low Safety Signal

3.2.3 Information Processing Subsystem Verification

To verify the performance of the information processing subsystem, we analyzed both the linear acceleration data from the IMU and the corresponding safety signal output generated by our crash detection algorithm during a controlled front end collision test. Figure 13 shows the total linear acceleration magnitude over time as the bike approached and impacted the wall. During normal motion, the acceleration values fluctuate around a steady baseline, and at the moment of collision the magnitude sharply increases and exceeds our predefined crash threshold, confirming that the IMU and filtering pipeline correctly captured the sudden spike associated with a real crash event. Figure 14 displays the safety signal output for the same time window, showing that as soon as the acceleration surpassed the crash threshold, the ESP32 generated a high safety signal, successfully detecting the event and activating the motor cutoff response. Together, these results verify that the information processing subsystem can reliably detect a front-end collision and trigger the appropriate shutdown in real time.

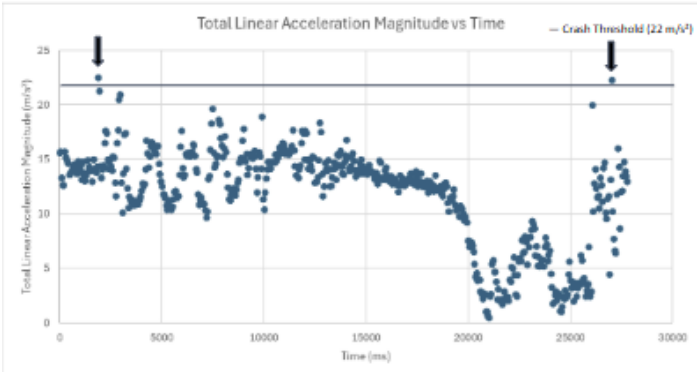

Figure 13: Total Linear Acceleration Magnitude (m/s^2) vs Time (ms) During Front End Collision Testing

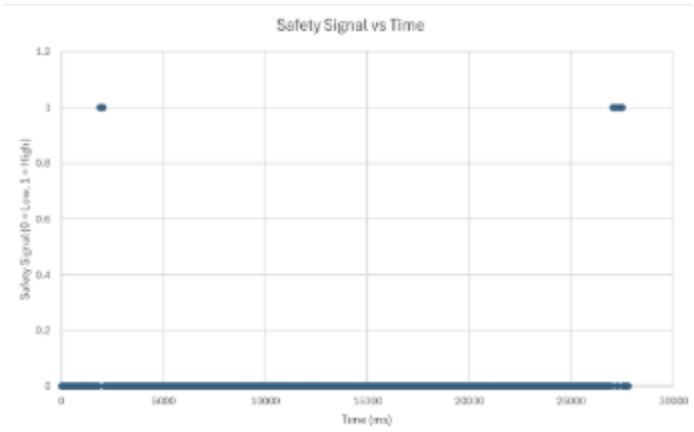

Figure 14: Safety Signal vs Time (ms) During Front End Collision Testing

3.2.4 Motor Controller Subsystem Verification

The motor controller subsystem was tested to confirm proper operation under both normal and safety cut-off conditions. During our testing, we used a signal level motor to validate that this subsystem would work as intended. Figures 11 and 12 show that this subsystem would see the full analog throttle signal when our safety-cutoff signal was low, and then 0 when the safety-cutoff signal was high.

4. Costs

This section summarizes both the parts cost and the estimated labor cost for the E-Bike Crash Detection and Safety System. We report retail pricing, approximate bulk pricing where applicable, and the actual out-of-pocket cost we incurred for the project. Only items that were not supplied by the ECE 445 lab are counted toward our actual cost.

4.1 Parts

Table 1 lists the major components used in our final design. For each part we include the typical single-unit retail price, an approximate bulk purchase cost (e.g., when buying packs of 5–10 units), and the actual cost we paid. Several core items, such as the ESP32-S3 module and IMU board, were provided from the ECE shop drawers and therefore have an actual cost of $0 for our team even though we include their retail values for completeness.

Table 1: Parts Costs

Part	Manufacturer / Part #	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
ESP32-S3 DevKitC-1 module	Espressif ESP32-S3-DevKitC-1	12.00	11.00	0.00
9-axis IMU breakout (accel/gyro/mag)	TDK ICM-20948 module	15.00	13.00	0.00

Custom 2-layer PCB (≈10×10 cm, set of 5)	JLCPCB / PCBWay	25.00	20.00	20.00
5 V regulator	Rohm BD50FC0FP-E2	1.00	0.80	0.80
3.3 V regulator	Diodes Inc. AP2112K-3.3	0.75	0.60	0.60
ESD / TVS protection array	Littelfuse SP0503BAHT	0.60	0.40	0.40
N-channel MOSFETs for cutoff circuit	Generic AO-series MOSFETs	1.50	1.20	1.20
Tactile push button (reset)	Generic 6 mm tact switch	0.25	0.20	0.20
Toggle switch (main enable)	Generic SPST toggle switch	3.00	2.50	2.50
Indicator LEDs + resistors	Generic LED/resistor assortment	2.00	1.50	1.50
5 V DC demonstration motor	Generic DC	6.00	5.00	5.00

				motor			
Connectors, headers, wire, heat-shrink				Generic assortment kit	10.00	8.00	8.00
9 V battery (for standalone demos)				Generic alkaline 9 V	2.00	1.50	0.00
Total		**79.10**	**65.70**				**40.20**

4.2 Labor

To estimate labor cost, we follow the standard ECE 445 convention and assume a reasonable entry-level salary for an ECE graduate of $40/hour, multiplied by a factor of 2.5 to account for overhead, benefits, and indirect costs. Each of the three team members spent roughly 120 hours over the semester on design, implementation, testing, and documentation.

For one team member:

- Hourly rate: $40/hour
- Effective rate with overhead: $40 × 2.5 = $100/hour
- Hours worked: 120 hours
- Labor cost per person: 120 × $100 = $12,000

With three team members, the total estimated labor cost is:

- Total labor cost = 3 × $12,000 = $36,000

This labor estimate is hypothetical and not billed to the course, but it shows that the engineering effort required to design, build, and verify a reliable crash-detection system would dominate the overall project cost compared to the relatively low price of the hardware components.

5. Conclusion

The conclusion section of this document summarizes the outcomes of our project, the challenges we encountered, and what we can do with our work in the future. In this section, we look at the technical accomplishments that we achieved before the conclusion of senior design. We also touch on the uncertainties in our project and the limitations we encountered that affected the overall system performance. We also look at the ethical concerns crash testing brought during our development. After this, we will explain the future of our project and the potential improvements we can make to our system. Together, these sections provide an overall assessment of our project.

5.1 Accomplishments

Over the course of this project, we successfully designed and tested a working crash detection and safety cutoff system for an e-bike. Our system reliably detected rollover events above 60° and identified front, side, and rear collisions without triggering during normal riding. We also demonstrated that the throttle intercept circuit can clamp the throttle signal below 2 volts during a crash, which meets the main safety requirement. IMU calibration and filtering provided stable roll and pitch estimates, and our final PCB, 3D printed enclosure, and subsystem integration showed that the entire system can be packaged into a mountable device for real bikes. Overall, despite setbacks, we proved that our idea works and has the potential to improve e-bike safety.

5.2 Uncertainties

Throughout the project, we encountered several challenges. Crash testing had natural safety limitations, and we lost both our first development board due to overvoltage and our final PCB from an unexpected ESD event. These setbacks prevented us from performing a full system demo on the final hardware. We also learned that IMU drift, vibrations, and world frame calibration makes accurate crash detection more difficult than expected. Even with these challenges, each subsystem performed as designed, and we gained valuable experience in hardware reliability, debugging, and safe testing practices.

5.3 Ethical considerations

Because our system directly affects rider safety, we made sure to follow ethical engineering principles, especially minimizing the risk of harm. Our design avoids cutting off the motor unless a true crash is detected, reducing the possibility of accidents caused by false triggers. All crash tests were performed in controlled conditions to ensure no one was put at risk. Instead of disconnecting the motor power, which can be unpredictable, we used a safer throttle clamping method. We also included a manual reset, so the rider maintains control after the system activates.

5.4 Future work

Moving forward, the next step would be to mount a fully operational version of our device onto an actual e-bike and conduct longer real-world tests. Additionally, adding wireless reporting or Bluetooth based alerts could make the system more useful during emergencies. With additional time, we could further refine our filtering and crash classification logic to reduce false positives. Overall, this project created a strong foundation that, with a few upgrades, could become a practical real world safety device for e-bikes.

References

[1] Student Self-Help Parts Drawers Inventory., University of Illinois ECE Shop, 2025.

[2] IEEE Code of Ethics., Institute of Electrical and Electronics Engineers, 2020. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[3] "ESP32 Example Projects," ECE 445 Senior Design Laboratory Wiki, Grainger College of Engineering, University of Illinois Urbana-Champaign, 2025. Available: https://courses.grainger.illinois.edu/ece445/wiki/#/esp32_example/index

[4] Espressif Systems, ESP32-S3 Hardware Design Guidelines., 2023. Available: https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32s3/pcb-layout-design.html

[5] Espressif Systems, ESP32-S3-WROOM-1 / ESP32-S3-WROOM-1U Datasheet., Rev. 1.3, 2023. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[6] Espressif Systems, ESP32-DevKitC V4 Schematic., 2023. Available: https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf

[7] Cycrown, "Electric Bike Throttle: How It Works," Cycrown Electric Bike Blog, 2024. Available: https://www.cycrown.com/blogs/learn/electric-bike-throttle-how-it-works

[8] TDK InvenSense, ICM-20948: 9-Axis MotionTracking Device Datasheet., Rev. 1.3, 2016. Available: https://invensense.tdk.com/wp-content/uploads/2016/06/DS-000189-ICM-20948-v1.3.pdf

[9] Diodes Incorporated, AP2112K-3.3: 600mA CMOS LDO Regulator Datasheet., 2024. Available: https://www.diodes.com/assets/Datasheets/AP2112.pdf

[10] Grainger College of Engineering, "Post-Graduation Success: Electrical Engineering," University of Illinois Urbana-Champaign, 2023. Available: https://grainger.illinois.edu/academics/undergraduate/majors-and-minors/electrical-engineering

[11] SP0503BAHT ESD Diode, Amazon product page, accessed 2024. Available: https://www.amazon.com/10PCS-SP0503BAHT-SP0503BAHTG-SP0503-SOT143/dp/B0CTX9ZRQZ/

[12] ROHM Semiconductor, "BDxxFC0 Series: 35V Withstand Voltage 1A LDO Regulators Datasheet," Rev. 0.06, Mar. 21, 2025. Available:

https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/linear_regulator/bdxxfc0wefj-e.pdf

Appendix A: Requirement and Verification Table

Table 2: Power Management Subsystem Requirements and Verifications

Power Subsystem Requirements	Verifications	Y/N
The 5 Volt regulator should be able to step down the voltage of the 9 volt power supply to 5 volts.	• Use a test point to demonstrate that we correctly stepped down the 9 Volt battery input to between 4.9 and 5.1 Volts	Y
The 3.3 Volt regulator should be able to step down 5 volts from the 5 volt regulator to 3.3 volts.	• Test point after the 3.3 volt regulator to demonstrate a stable 3.15 Volt to 3.45 Volt output	Y

Table 3: Throttle Intercept Subsystem Requirements and Verifications

Throttle Intercept Subsystem Requirements	Verifications	Y/N
When the safety cut-off signal is low, the motor controller should see all of the voltage provided by the throttle.	• Probe the current across the collector and emitter of NPN Q1 to see if all of the current from the motor goes across to ground. • Probe current across the collector and emitter of NPN Q2 to see if 0 current flows across the BJT. • Test motor should be spinning	Y
When the safety signal is high, the voltage across the motor should decrease below the motor controller's threshold voltage.	• Test point across the motor to see if the voltage is under 2 volts. • Test point on the base of NPN Q2 to see if the safety signal is present. • Test motor should be turned off	Y

Table 4: Motor Connection Subsystem Requirements and Verifications

Motor Connection	Verifications	Y/N
The motor controller should not step up throttle voltage when the safety cut-off signal is high.	• Probe across the motor controllers terminal to check if the voltage is beneath its threshold turn on voltage. • The test motor should be turned off	Y
The motor controller should step up the 0 to 5 volt throttle signal when the safety cut-off signal is low.	• Probe across the motor controllers terminal to check if the voltage is above its threshold turn on voltage. • Test point before the motor to see if the motor is receiving a voltage. • Visibly check to see if the motor is spinning.	Y

Table 5: Information Processing Subsystem Requirements and Verifications

Information Processing Subsystem Requirements	Verifications	Y/N
An accelerometer must accurately measure linear acceleration in the X, Y, and Z planes within +/- 0.05g	• After connecting the accelerometer to the microprocessor, confirm communication using Arduino IDE serial monitor. • Apply a controlled acceleration to each direction, to confirm accuracy of motion capture.	Y
The gyroscope must accurately measure angular velocity in the X, Y, and Z planes within +/- 0.05% accuracy	• After connecting the gyroscope to the microprocessor, tilt the gyroscope at a controlled speed to test accuracy. • Verify communication between the gyroscope and the microprocessor using the Arduino IDE serial monitor.	Y

The microprocessor needs to be able to read gyroscope and accelerometer data. A microprocessor should send a 3.3 volt signal to the safety cut-off circuit when a crash has been detected. The reset button should restart the program in the microprocessor.	• Check the Arduino serial monitor to see if there is serial communication with the accelerometer and gyroscope. • Applying different tilt speeds and linear acceleration to determine microprocessors can accurately read data. • Use a test point at the base of NPN Q2 to check the voltage of the safety cut-off signal. Check to see if test motor is not spinning • Use a test point at the base of NPN Q2 to check the voltage of the safety cut-off signal and make sure it is low. Check to see if the test motor starts spinning again.	Y
The micro usb must enable software and firmware upload to the microprocessor.	• Connect the device to a PC using micro-USB and confirm COM port detection. • Successfully upload firmware using Arduino App.	Y
The reset button must reset the microcontroller and restart the program when pressed.	• Press the reset button and confirm that the microprocessor resets. • Visibly check if the motor is able to work again, after button press.	Y