

Netaji Subhas University of Technology



IoT workshop

EIECC19

Project Report File

IoT based Crash Detection Smart Helmet

Submitted to:-

Prof. Dhananjay V. Gadre

Submitted by:-

Prayush Rai (2020UEI2803)
Prakhar Tripathi (2020UEI2828)

ABSTRACT

The growing number of road accidents involving two-wheeler riders necessitates innovative solutions to improve a rider's safety. This project aims to design and develop an IoT-enabled smart helmet with crash detection capabilities and automatic emergency response feature to enhance road safety for riders. The smart helmet system employs accelerometer sensor, GPS technology, and ESP 32 having a built-in wifi module to monitor the rider's safety, providing real-time updates and potential accident detection.

In the event of an accident, the smart helmet promptly transmits the location coordinates of the incident to designated emergency contacts and relevant authorities. This enables swift response and minimises the time taken for help to reach the affected rider. Rapid assistance is crucial in the golden hour after an accident.

The paper also discusses the project's implications on improving overall road safety, reducing fatalities, and the potential for future advancements and widespread adoption of smart helmets in the motorcycle and scooter rider community.

1 Introduction

1.1 Description

The importance of road safety for riders cannot be overstated, as they are highly vulnerable to severe injuries and fatalities in the event of an accident. With the increasing number of road accidents involving motorcycles and scooters, the need for effective safety measures is more crucial than ever.

Our Smart helmet which we have named as headBud is designed to address this pressing issue by providing real-time monitoring and accident detection through sensors, GPS technology, and ESP 32. These components work together to continuously assess the rider's safety and detect potential crashes, ensuring timely assistance when needed.

Upon detecting an accident, the smart helmet instantly sends location coordinates of the crash site to pre-selected emergency contacts, facilitating prompt response and reducing the time taken for assistance to arrive. This can be crucial in saving lives and preventing long-term injuries.

HeadBud not only emphasises the importance of road safety for two-wheeler riders but also strives to create a tangible impact by reducing accident-related fatalities and injuries. By fostering a safer riding environment, the project aims to contribute significantly to the overall improvement of road safety and the well-being of motorcycles.

1.2 Motivation

The motivation behind the Smart Helmet project stems from a strong desire to improve road safety for two-wheeler riders and reduce the number of accident-related fatalities and injuries. The following pointers outline the key driving factors behind the development of this innovative smart helmet:

- 1. Alarming Accident Statistics :** The high number of road accidents involving motorcycles and scooters highlights the urgency for effective safety measures. The development of Smart Helmet is fueled by

the need to address this alarming trend and provide a practical solution to enhance rider safety.

2. Vulnerability of Riders : Two-wheeler riders are more vulnerable to severe injuries and fatalities compared to other road users, owing to the lack of a protective shell and limited safety features. The Smart Helmet project aims to offer an additional layer of protection to riders by providing a comprehensive safety solution through advanced technology.

3. Rapid Emergency Response : In the event of an accident, timely assistance plays a critical role in saving lives and preventing long-term injuries. The Smart Helmet is designed to facilitate a swift emergency response by automatically sending location coordinates of the crash site to pre-selected contacts.

4. Future Scalability and Expansion : The Smart Helmet project has the potential to integrate additional safety features, such as blind-spot detection, bluetooth communication with the fellow passenger and collision warnings, further enhancing its capabilities. This scalability and potential for expansion make the smart helmet a promising solution for improving road safety for riders

1.3 Objective

The Smart Helmet project has several objectives aimed at enhancing road safety, improving rider experience, and promoting the adoption of advanced safety measures among two-wheeler riders. The following are the key objectives of the project, outlined in a detailed manner:

1. Enhance Road Safety: The primary objective of the Smart Helmet project is to improve road safety for two-wheeler riders by providing a comprehensive safety solution that incorporates crash detection and sends an automatic emergency response

2. Facilitate Rapid Emergency Response: In the event of an accident, the Smart Helmet is designed to generate an emergency response by automatically sending the location coordinates of the crash site to designated contacts. This objective aims to ensure that riders receive prompt

assistance, which can be critical in saving lives and preventing long-term injuries.

3. Expand Helmet Capabilities: The Smart Helmet project has an objective of exploring potential future enhancements, such as blind spot detection, collision warnings, and bluetooth communication with fellow passengers. By integrating additional safety features and assistance tools, the project aims to develop a comprehensive and versatile safety solution that addresses a wide range of riding scenarios and safety concerns.

In summary, the objectives of the Smart Helmet project are centred around enhancing road safety, promoting the adoption of advanced safety measures, and improving the overall riding experience for two-wheeler riders. By achieving these objectives, the project aspires to make a significant impact on rider safety and contribute to the well-being of motorcycle and scooter riders worldwide.

2 Components

2.1 ESP32



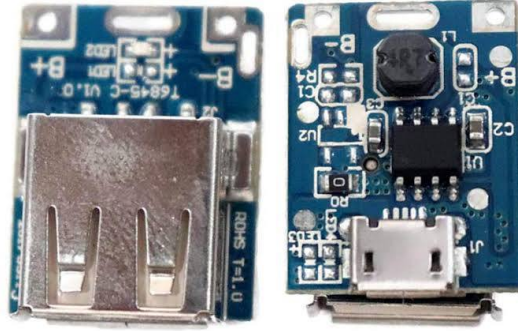
ESP32 is a powerful, low-cost, Wi-Fi and Bluetooth-enabled

microcontroller developed by Espressif Systems. It is designed for use in Internet of Things (IoT) applications and is based on the Xtensa LX6 CPU architecture. The ESP32 includes dual-core processors, a variety of communication interfaces, and an extensive set of peripherals, making it ideal for a wide range of IoT applications. The ESP32 features a rich set of features, including:

- **Dual-core processors:** The ESP32 includes two Tensilica Xtensa LX6 processors, providing high performance and flexibility.
- **Wi-Fi and Bluetooth connectivity:** The ESP32 includes built-in Wi-Fi and Bluetooth modules, providing seamless wireless connectivity.
- **Wide range of communication interfaces:** The ESP32 supports a wide range of communication interfaces, including SPI, I2C, UART, CAN, and Ethernet.
- **Rich set of peripherals:** The ESP32 includes a rich set of peripherals, including ADC, DAC, I2S, and SDIO.
- **Low power consumption:** The ESP32 features a low-power mode that enables it to operate for extended periods on battery power.
- **Support for various operating systems:** The ESP32 supports various operating systems, including FreeRTOS, Zephyr, and MicroPython.

The ESP32 is widely used in IoT applications, such as home automation, industrial automation, smart cities, and wearable devices. Its low cost, small size, and rich set of features make it an ideal choice for a wide range of IoT applications.

2.2 Charging and Booster Circuit (T6845-C)



The ESP32 requires a stable power supply to operate reliably, and a charging and booster circuit can help ensure that the device receives consistent power. A charging and booster circuit typically includes a battery, a charging port for the battery, and a step up voltage regulator. The boost converter is used to increase the voltage of the battery to the required level for the sensors and modules connected to ESP32. The

charging and booster circuit can be designed using off-the-shelf components or purchased as a pre-made module. When designing a charging and booster circuit, it is important to ensure that the components used are compatible with the ESP32's voltage requirements and that the circuit can supply the required current.

It is also important to consider the size and weight of the circuit, as well as the battery capacity and charging time. A larger battery will provide longer operating time, but it will also increase the weight and size of the device. Similarly, a faster charging circuit will charge the battery more quickly, but it may also increase the risk of overheating or overcharging.

Overall, a charging and booster circuit is an essential component for powering an ESP32 in portable or battery-operated applications. By providing a stable power supply, it can help ensure that the device operates reliably and efficiently, improving overall performance and user experience.

2.3 Neo 6M GPS Module



The Neo 6M GPS module is a compact, high-performance, and cost-effective GPS receiver module designed for various applications, including navigation, tracking, and location-based services. It features the u-blox Neo 6M GPS chipset, which is known for its high sensitivity, low power consumption, and rapid position acquisition capabilities.

Features:

- **High Sensitivity:** The Neo 6M GPS module maintains a strong GPS signal under an open sky.
- **Rapid Position Acquisition:** The module provides fast location data with a Cold Start Time to First Fix (TTFF) of 27 seconds, Warm Start TTFF of 27 seconds, and Hot Start TTFF of 1 second.
- **Low Power Consumption:** The Neo 6M GPS module has a low power consumption of 45mA at 3.0V during continuous operation, making it suitable for battery-powered applications.
- **Compact Size and Lightweight:** With dimensions of 25mm x 25mm x 6mm, the module is a compact and lightweight solution for applications with space constraints and weight considerations.
- **UART and I2C Interfaces:** The module supports both UART and I2C communication interfaces, providing flexibility in connecting

and communicating with various microcontrollers and processors.

- **Built-in Antenna:** The module comes with a built-in ceramic patch antenna, ensuring reliable GPS signal reception without the need for an external antenna.

In conclusion, the Neo 6M GPS module is a powerful, compact, and versatile solution for applications requiring accurate and real-time positioning data. Its high sensitivity, rapid position acquisition, and low power consumption make it an ideal choice for the Smart Helmet project, ensuring swift and reliable location coordinates in the event of a crash.

2.4 ADXL345 Accelerometer



The ADXL345 is a compact and low-power 3-axis accelerometer designed for various applications, including motion sensing and orientation detection. It is suitable for the Smart Helmet project to determine rider's acceleration and detect crashes. Key features :

- **High Resolution:** 13-bit accuracy across all three axes (X, Y, and Z).
- **Adjustable Sensitivity:** User-selectable sensitivity ranges of $\pm 2g$, $\pm 4g$, $\pm 8g$, and $\pm 16g$.
- **Low Power Consumption:** $23 \mu A$ in measurement mode and $0.1 \mu A$ in standby mode at 3.3V.

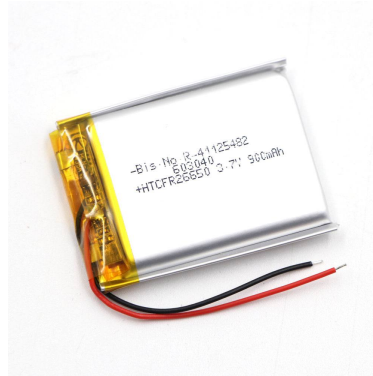
- **Digital Output:** I2C or SPI interface for easy communication with microcontrollers.
- **Small Form Factor:** Dimensions of 3mm x 5mm x 1mm, making it lightweight and easy to integrate. Overall, the ADXL345 accelerometer is an ideal choice for the Smart Helmet project due to its precise measurements, adjustable sensitivity, and energy-efficient operation.

2.5 Rocker Switch



A rocker switch is a simple and versatile electromechanical component commonly used for toggling electrical circuits on and off. Designed with a user-friendly rocker mechanism, it is ideal for a wide range of applications, including controlling the power supply, switching modes, or selecting options. In the Smart Helmet project, a rocker switch can be employed to activate or deactivate the system as needed. A rocker switch is a practical and reliable choice for the Smart Helmet project due to its simple operation, compact design, and high durability. Its versatility and ease of installation make it an ideal component for controlling the activation and deactivation of the helmet's system.

2.6 3.7 v 900 mAh Lithium Polymer Rechargeable Battery



The 3.7V lithium polymer rechargeable battery is a lightweight and high-energy-density power source ideal for portable devices like the Smart Helmet. Key features and benefits include:

- **High Energy Density:** Stores more energy per unit of weight compared to other battery chemistries.
- **Lightweight Design:** Thin and flexible form factor suitable for space-constrained applications.
- **Rechargeable:** Cost-effective and environmentally friendly solution for repeated use.
- **Wide Operating Temperature Range:** Performs reliably in temperatures between -20°C and 60°C .
- **Low Self-Discharge Rate:** Maintains a higher percentage of charge when not in use, extending battery life.
- **Versatile Capacity Options:** Available in various capacities to suit different power requirements.

Overall, the 3.7V lithium polymer battery is an excellent choice for the Smart Helmet project due to its high energy density, lightweight design, and rechargeable nature, ensuring reliable performance and adaptability to project-specific power requirements.

3 IoT Platform Used

3.1 Twilio



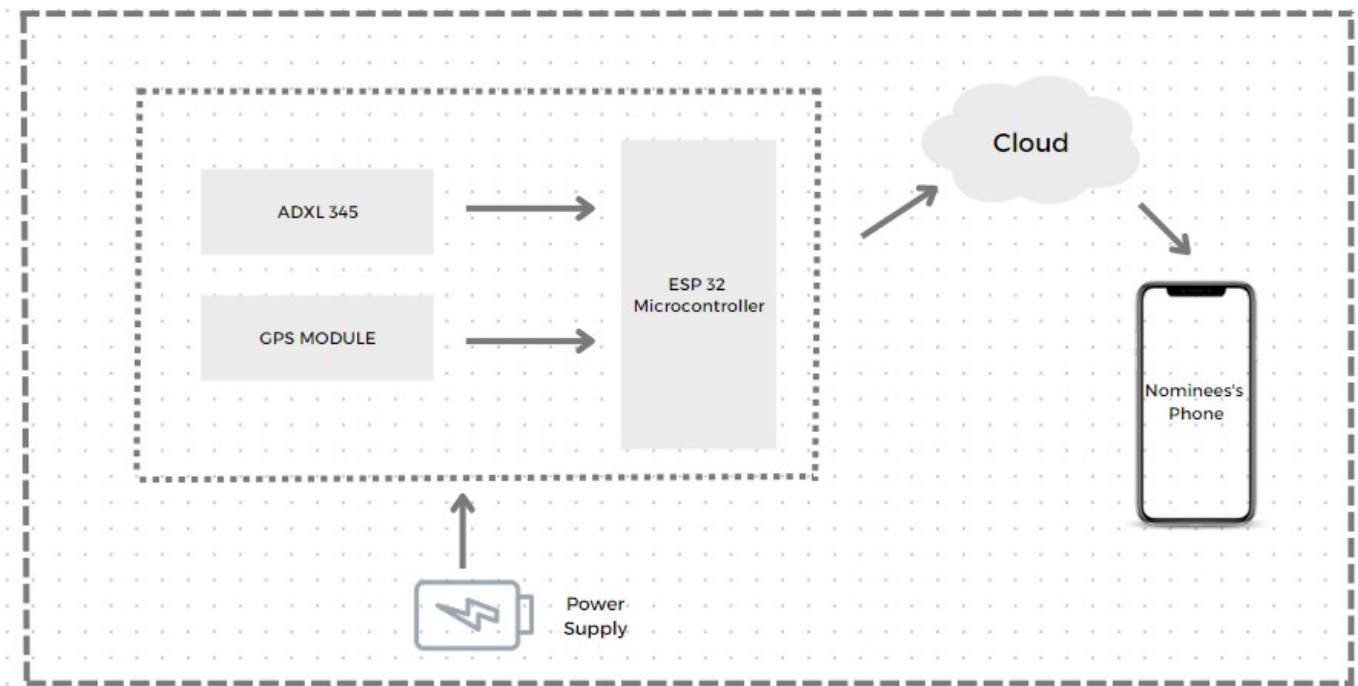
Twilio IoT is a powerful and versatile cloud-based platform designed to help developers build, deploy, and scale connected devices and applications. By leveraging Twilio's robust infrastructure and suite of communication APIs, the Smart Helmet project can easily integrate real-time connectivity, messaging, and location services. The following highlights key features and benefits of the Twilio IoT platform: Key

Features:

- **Global Connectivity:** Twilio IoT offers seamless global connectivity with a vast network of carriers, ensuring reliable communication for connected devices and applications.
- **Scalability:** The platform is designed to handle a wide range of devices and traffic, enabling the Smart Helmet project to grow and adapt to changing demands.
- **Security:** Twilio IoT provides robust end-to-end encryption and advanced security features, protecting data transmission and ensuring the privacy and integrity of the Smart Helmet system.
- **Flexible APIs:** Twilio's suite of communication APIs simplifies the integration of SMS, voice, and data services into the Smart Helmet, enabling efficient communication between the device and emergency contacts.
- **Real-Time Monitoring and Analytics:** The platform provides real-time monitoring and analytics tools, allowing for proactive troubleshooting and optimization of the Smart Helmet's IoT connectivity.

In conclusion, the Twilio IoT platform is an ideal solution for the Smart Helmet project due to its global connectivity, scalability, and robust security features. The platform's flexible APIs and easy integration capabilities enable seamless communication, real-time monitoring, and efficient operation of the Smart Helmet system.

4 Block Diagram



5 Working

The IoT-enabled Smart Safety Helmet is an innovative solution designed to enhance road safety for two-wheeler riders by detecting crashes and sending location coordinates to the rider's nominees. The system leverages a combination of sensors, IoT connectivity, and other components to ensure efficient and reliable operation. Here is a detailed description of the working of the Smart Safety Helmet project:

- **Sensing and Detection:** The ADXL345 accelerometer continuously monitors the rider's acceleration and sends the data to the microcontroller. In the event of an abrupt change in acceleration, the

acceleration crosses a threshold value which is predefined and stored in the microcontroller's memory.

The accelerometer measures the acceleration along the 3 axes, so the net acceleration will be:

$$a_n = (ax^2 + ay^2 + az^2)^{1/2}$$

- **GPS Location Tracking:** The GPS also is continuously monitoring the location coordinates and sending them to the microcontroller. When the acceleration crosses the threshold value, the microcontroller sends these gps coordinates to the cloud. When $a_n \geq threshold$, the microcontroller will send the gps coordinates to the cloud.

We have set the threshold to be 18m/sec^2

- **IoT Connectivity:** The Twilio IoT platform facilitates seamless communication between the Smart Helmet and the cloud. The microcontroller sends the crash event and location data to the cloud using the Twilio IoT platform.
- **Notification System:** Once the crash data is uploaded to the cloud, the Twilio communication APIs trigger the notification process. The rider's nominees receive SMS with the location coordinates and a script notifying about the crash event, allowing them to take immediate action and send help to the crash site.
- **Power Management:** The 3.7 V lithium polymer rechargeable battery provides a reliable power source for the Smart Helmet's components. Its high energy density, lightweight design, and rechargeable nature ensure continuous operation and extended battery life.
- **User Interface:** A rocker switch is integrated into the helmet for easy activation and deactivation of the system. The switch enables riders to conveniently control the helmet's operation, ensuring the system is active when needed.

In summary, the IoT-enabled Smart Safety Helmet works by integrating sensors, IoT connectivity, and a notification system to detect crashes, retrieve location coordinates, and send alerts to the rider's nominees. The system's efficient power management, user-friendly interface, and real-time monitoring capabilities ensure reliable and effective operation, ultimately enhancing road safety for two-wheeler riders.

6 Designing Process

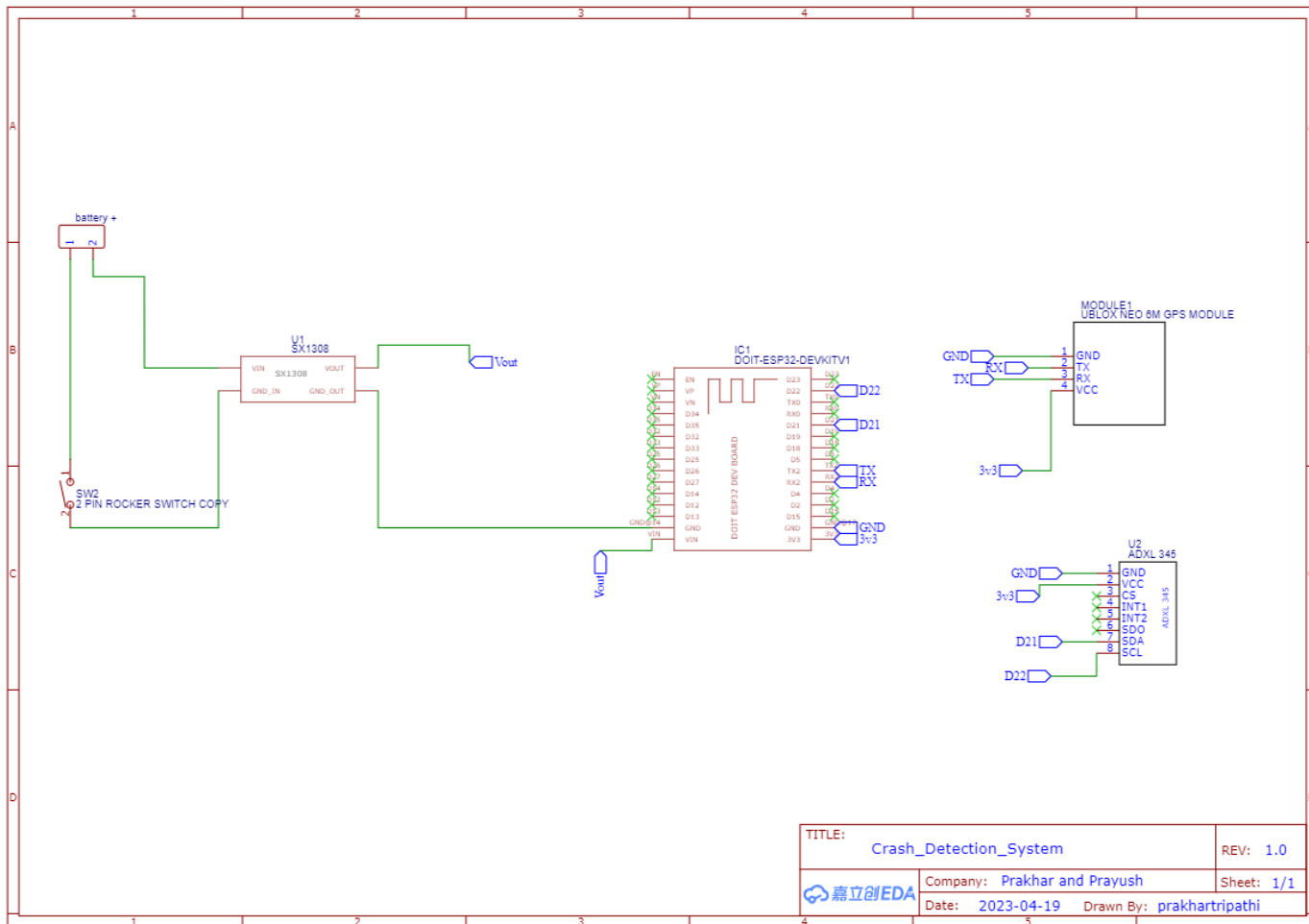
The design process of the Smart Safety Helmet involves planning, execution, and testing across two main domains: hardware and software.

6.1 Hardware Design

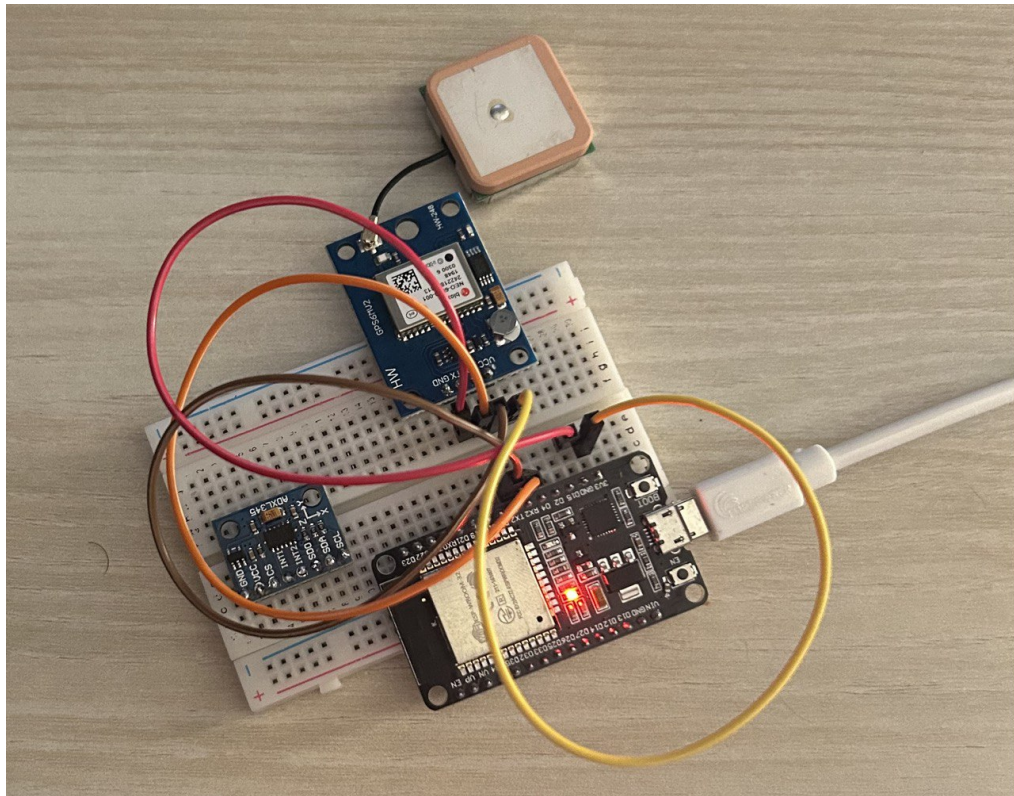
The hardware design process entails the appropriate selection of components, their effective integration, and testing to ensure the helmet's functionality. Here's a detailed explanation:

- **Requirement Analysis:** Before beginning the design process, it is critical to perform a thorough requirement analysis. This process involves understanding the needs of the end-user, the operational environment, and the technical specifications necessary for the project. This analysis will help determine the type of sensors needed, the level of accuracy required, the power supply requirements, and the necessary communication protocols.
- **Component Selection:** The first step involves choosing the right hardware like the ADXL345 accelerometer, Neo 6M GPS module, 3.7V lithium polymer rechargeable battery, rocker switch, and a suitable microcontroller. Make sure that these components align with the project's technical requirements such as power consumption, performance, and size. Verify the specifications of each component and ensure their compatibility with each other and the project requirements.

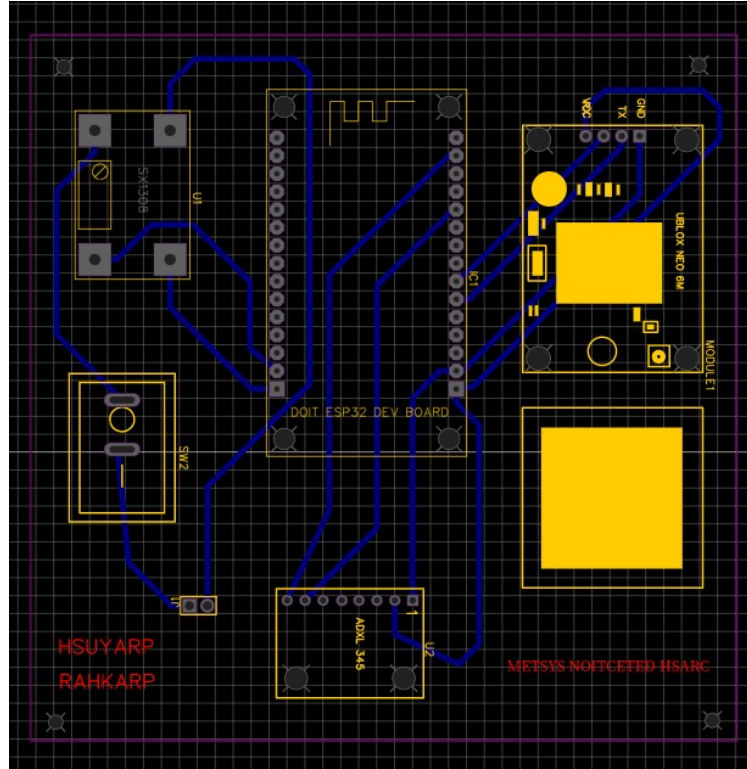
- Circuit Design:** Next, a schematic diagram illustrating the connections between the microcontroller, accelerometer, GPS module, battery, and switch needs to be designed. This design should ensure efficient power management and reliable data transfer. Precaution: Ensure the circuit is designed accurately to avoid short circuits and component failures.



- Prototyping:** After the PCB layout, assemble the components for a prototype. This stage allows validation of the design, testing the system's functionality, and making necessary adjustments. We tested our circuit on a breadboard first.



- **PCB Layout:** The finalised circuit design is then transformed into a Printed Circuit Board (PCB) layout. It should accommodate the physical size of the components and ensure robust electrical connections. Avoid overcrowding of components on the PCB, ensure the clearance and the track width is sufficient. Once the PCB is made, Handle it with care to prevent damage from static electricity or mishandling.



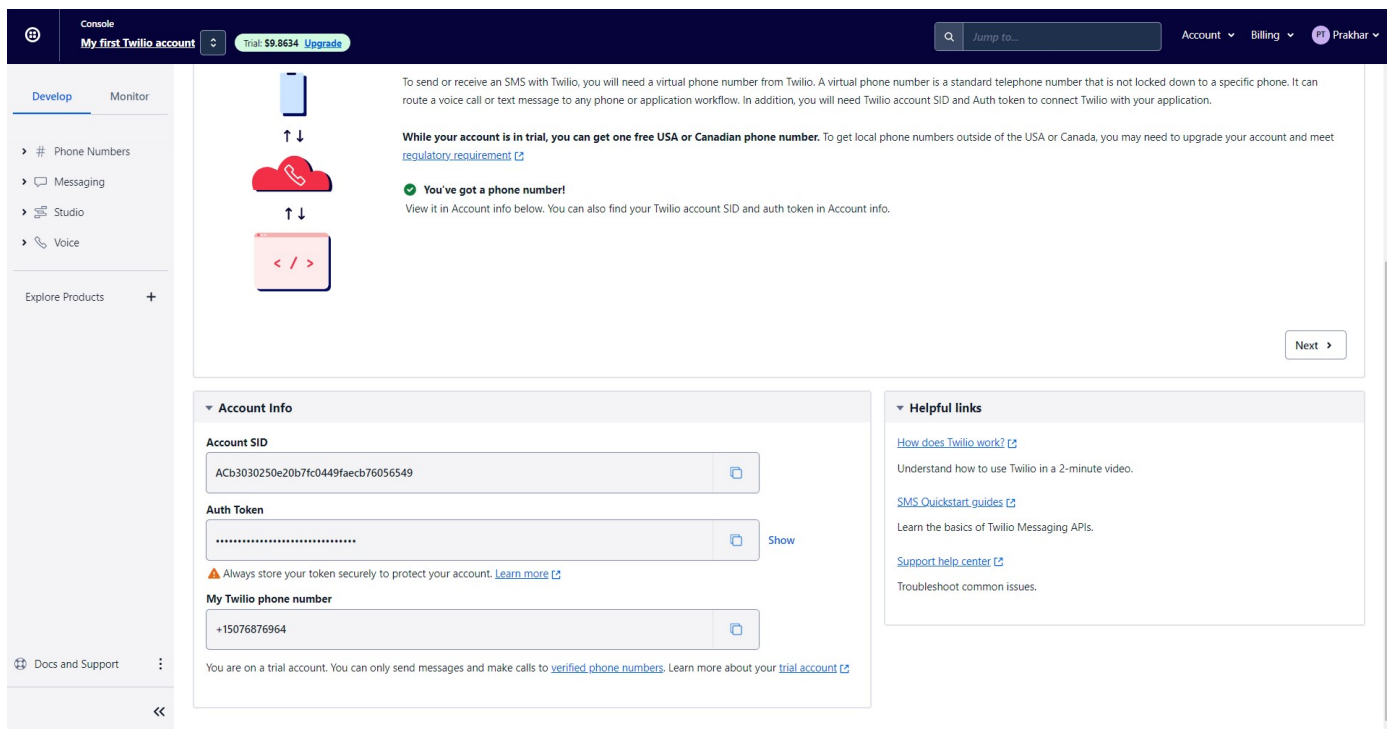
- **Testing:** The prototype undergoes testing under different conditions to ensure its reliability and performance. Problems encountered during this phase should be addressed, followed by iterative prototyping and testing until satisfactory performance is achieved.

6.2 Software Design

The software design process involves developing the firmware for the microcontroller, a cloud-based application for managing the helmet's functionalities, and ensuring seamless communication between them:

- **Firmware Development:** The firmware controls the operations of the accelerometer, GPS module, and manages power consumption. It is responsible for processing accelerometer data, retrieving GPS coordinates, and handling communication with the Twilio IoT platform.
- **Cloud Application Development:** The cloud-based application, developed on the Twilio IoT platform, receives crash

data from the helmet, triggers notifications to the rider's nominees, and provides real-time monitoring and analytics tools. Precaution: Make sure the application adheres to data privacy standards and is capable of handling high data loads.



• **Integration:** After developing the firmware and the cloud application, they are integrated to enable seamless communication. This involves programming the microcontroller to send data to the platform and setting up the Twilio IoT platform to receive the data. Precaution: Ensure secure and reliable data transmission between the helmet and the cloud platform.

6.3 Codes

Accelerometer testing:

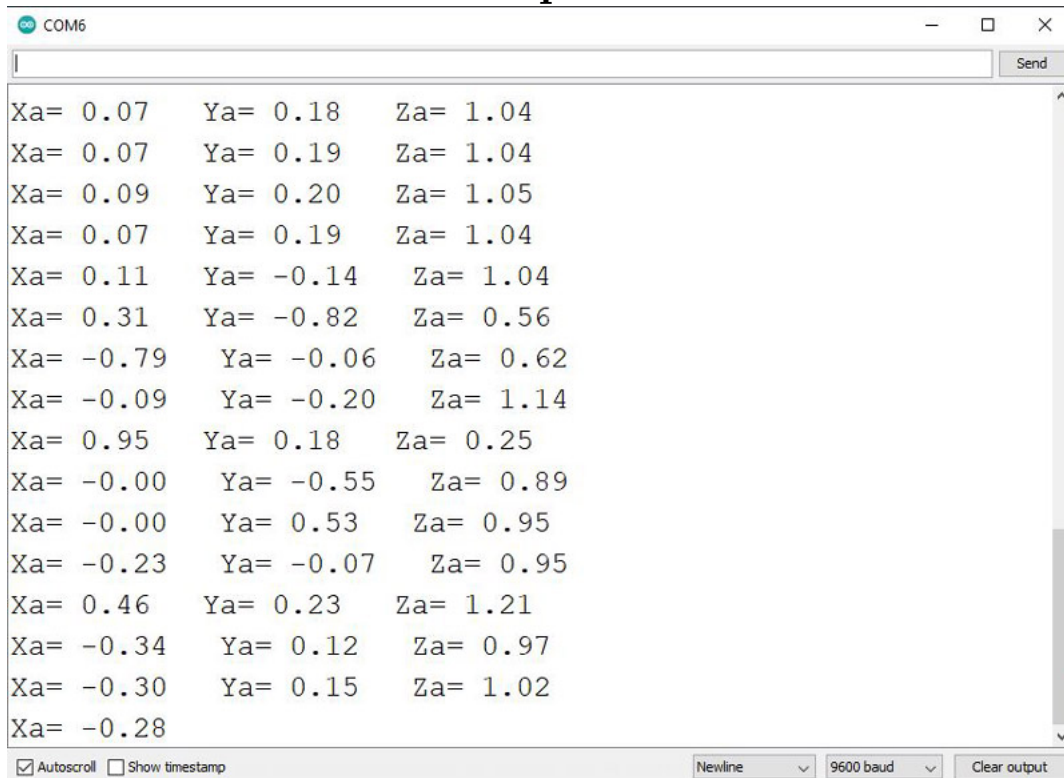
```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

void setup(void)
{
  Serial.begin(9600);
  if(!accel.begin())
  {
    Serial.println("Could not find a valid ADXL345 sensor, check
wiring!");
    while(1);
  }
}

void loop(void)
{
  sensors_event_t event;
  accel.getEvent(&event);
  Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("
");
  Serial.print("Y: "); Serial.print(event.acceleration.y); Serial.print("
");
  Serial.print("Z: "); Serial.print(event.acceleration.z);
Serial.println("  ");
  delay(2000);
}
```

Output:



```
COM6
Xa= 0.07   Ya= 0.18   Za= 1.04
Xa= 0.07   Ya= 0.19   Za= 1.04
Xa= 0.09   Ya= 0.20   Za= 1.05
Xa= 0.07   Ya= 0.19   Za= 1.04
Xa= 0.11   Ya= -0.14   Za= 1.04
Xa= 0.31   Ya= -0.82   Za= 0.56
Xa= -0.79   Ya= -0.06   Za= 0.62
Xa= -0.09   Ya= -0.20   Za= 1.14
Xa= 0.95   Ya= 0.18   Za= 0.25
Xa= -0.00   Ya= -0.55   Za= 0.89
Xa= -0.00   Ya= 0.53   Za= 0.95
Xa= -0.23   Ya= -0.07   Za= 0.95
Xa= 0.46   Ya= 0.23   Za= 1.21
Xa= -0.34   Ya= 0.12   Za= 0.97
Xa= -0.30   Ya= 0.15   Za= 1.02
Xa= -0.28
```

GPS testing:

```
#include <SoftwareSerial.h>
#include <TinyGPS++.h>

SoftwareSerial gpsSerial(16,17); // RX, TX
TinyGPSPlus gps;

void setup() {
  Serial.begin(9600);
  while (!Serial) {}

  gpsSerial.begin(9600);
  gpsSerial.print("hello");
}

void loop() {
  while (gpsSerial.available() > 0) {
    if (gps.encode(gpsSerial.read())) {
      if (gps.location.isValid()) {
        Serial.print("Latitude: ");
        Serial.print(gps.location.lat(), 6);
        Serial.print(", Longitude: ");
        Serial.println(gps.location.lng(), 6);
        delay(2000);
      }
    }
  }
}
```

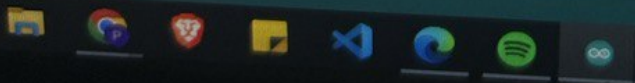
Output:

COM8

Latitude: 26.772733, Longitude: 80.966580
Latitude: 26.772733, Longitude: 80.966580
Latitude: 26.772733, Longitude: 80.966580
Latitude: 26.772733, Longitude: 80.966622
Latitude: 26.772733, Longitude: 80.966622
Latitude: 26.772733, Longitude: 80.966622
Latitude: 26.772733, Longitude: 80.966622
Latitude: 26.772733, Longitude: 80.966622
Latitude: 26.772733, Longitude: 80.966622
Latitude: 26.772733, Longitude: 80.966622
Latitude: 26.772740, Longitude: 80.966649
Latitude: 26.772740, Longitude: 80.966649
Latitude: 26.772740, Longitude: 80.966649
Latitude: 26.772740, Longitude: 80.966649
Latitude: 26.772740, Longitude: 80.966649
Latitude: 26.772740, Longitude: 80.966649
Latitude: 26.772740, Longitude: 80.966649
Latitude: 26.772740, Longitude: 80.966649
Latitude: 26.772737, Longitude: 80.966664
Latitude: 26.772737, Longitude: 80.966664
Latitude: 26.772737, Longitude: 80.966664
Latitude: 26.772737, Longitude: 80.966664
Latitude: 26.772737, Longitude: 80.966664
Latitude: 26.772737, Longitude: 80.966664
Latitude: 26.772737, Longitude: 80.966664
Latitude: 26.772737, Longitude: 80.966664
Latitude: 26.772737, Longitude: 80.966664

☒ Autoscroll ☐ Show timestamp

Newline



29°C Haz

Crash detection test without Iot:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

#define CRASH_THRESHOLD 18 // adjust this value to set the crash detection
threshold

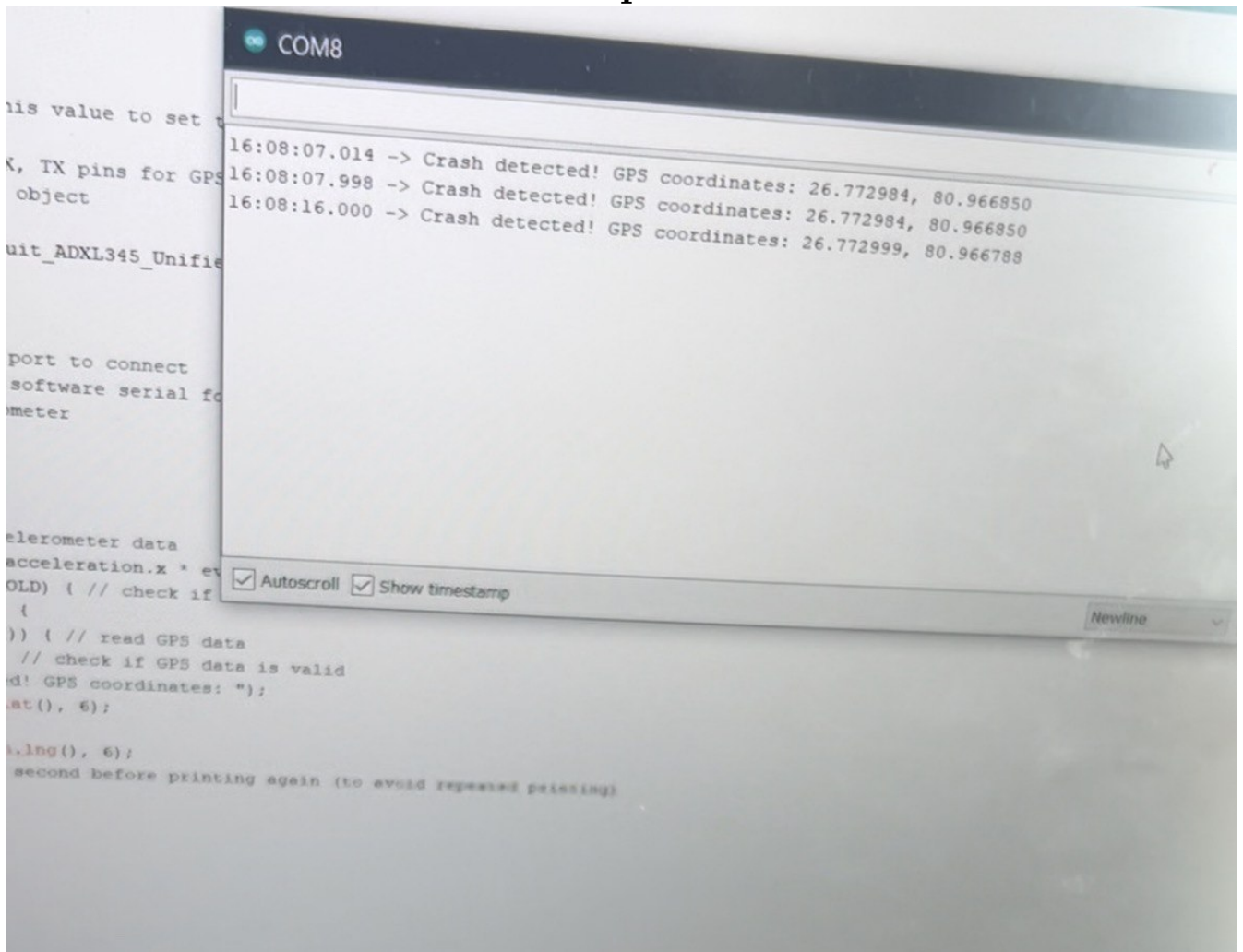
SoftwareSerial gpsSerial(16, 17); // RX, TX pins for GPS module
TinyGPSPlus gps; // create a TinyGPS++ object

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345); // create an
ADXL345 object

void setup() {
  Serial.begin(9600);
  while (!Serial); // wait for serial port to connect
  gpsSerial.begin(9600); // start the software serial for the GPS module
  accel.begin(); // start the accelerometer
}

void loop() {
  sensors_event_t event;
  accel.getEvent(&event); // read accelerometer data
  float accel_magnitude = sqrt(event.acceleration.x * event.acceleration.x +
event.acceleration.y * event.acceleration.y + event.acceleration.z *
event.acceleration.z); // calculate the magnitude of the acceleration vector
  if (accel_magnitude >= CRASH_THRESHOLD) { // check if a crash has been
detected
    while (gpsSerial.available() > 0) {
      if (gps.encode(gpsSerial.read())) { // read GPS data
        if (gps.location.isValid()) { // check if GPS data is valid
          Serial.print("Crash detected! GPS coordinates: ");
          Serial.print(gps.location.lat(), 6);
          Serial.print(", ");
          Serial.println(gps.location.lng(), 6);
          delay(1000); // wait for 1 second before printing again (to avoid
repeated printing)
        }
      }
    }
  }
}
```

Output:



The image shows a serial monitor window titled "COM8" displaying three lines of output. The output indicates that a crash was detected at three different times, each with associated GPS coordinates. The coordinates are 26.772984, 80.966850 for the first two crashes and 26.772999, 80.966788 for the third. The background shows a portion of a C++ code file with comments and function calls related to an accelerometer and GPS module.

```
16:08:07.014 -> Crash detected! GPS coordinates: 26.772984, 80.966850
16:08:07.998 -> Crash detected! GPS coordinates: 26.772984, 80.966850
16:08:16.000 -> Crash detected! GPS coordinates: 26.772999, 80.966788
```

COM8

☒ Autoscroll ☒ Show timestamp

Newline

is value to set t
K, TX pins for GPS
object
uit_ADXL345_Unifie
port to connect
software serial fo
meter
elerometer data
acceleration.x * ev
OLD) { // check if
{
}) { // read GPS data
// check if GPS data is valid
d! GPS coordinates: ");
at(), 6);
l.log(), 6);
second before printing again (to avoid repeated printing)

Crash detection test with Iot:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <base64.h>

// Replace with your network credentials
const char* ssid      = "Infinix ZERO 5G";
const char* password = "87654321";

// Replace with your Twilio account SID and auth token
const char* account_sid = "ACb3030250e20b7fc0449faecb76056549";
const char* auth_token = "b16bad3042efbe70e812f31c2b251da9";

// Replace with your Twilio phone number and recipient phone number
const char* twilio_phone_number = "+15076876964";
const char* recipient_phone_number = "+919654937307";

#define CRASH_THRESHOLD 18 // adjust this value to set the crash
                             detection threshold

SoftwareSerial gpsSerial(16, 17); // RX, TX pins for GPS module
TinyGPSPlus gps; // create a TinyGPS++ object

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345); //
create an ADXL345 object

// Function to send an SMS message with Twilio
void sendSMS(String message) {
  HTTPClient http;
  http.begin("https://api.twilio.com/2010-04-01/Accounts/" +
String(account_sid) + "/Messages.json");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  http.setAuthorization(account_sid, auth_token);
  String postData = "From=" + String(twilio_phone_number) + "&To=" +
String(recipient_phone_number) + "&Body=" + message;
```

```

    int httpResponseCode = http.POST(postData);
    http.end();
}

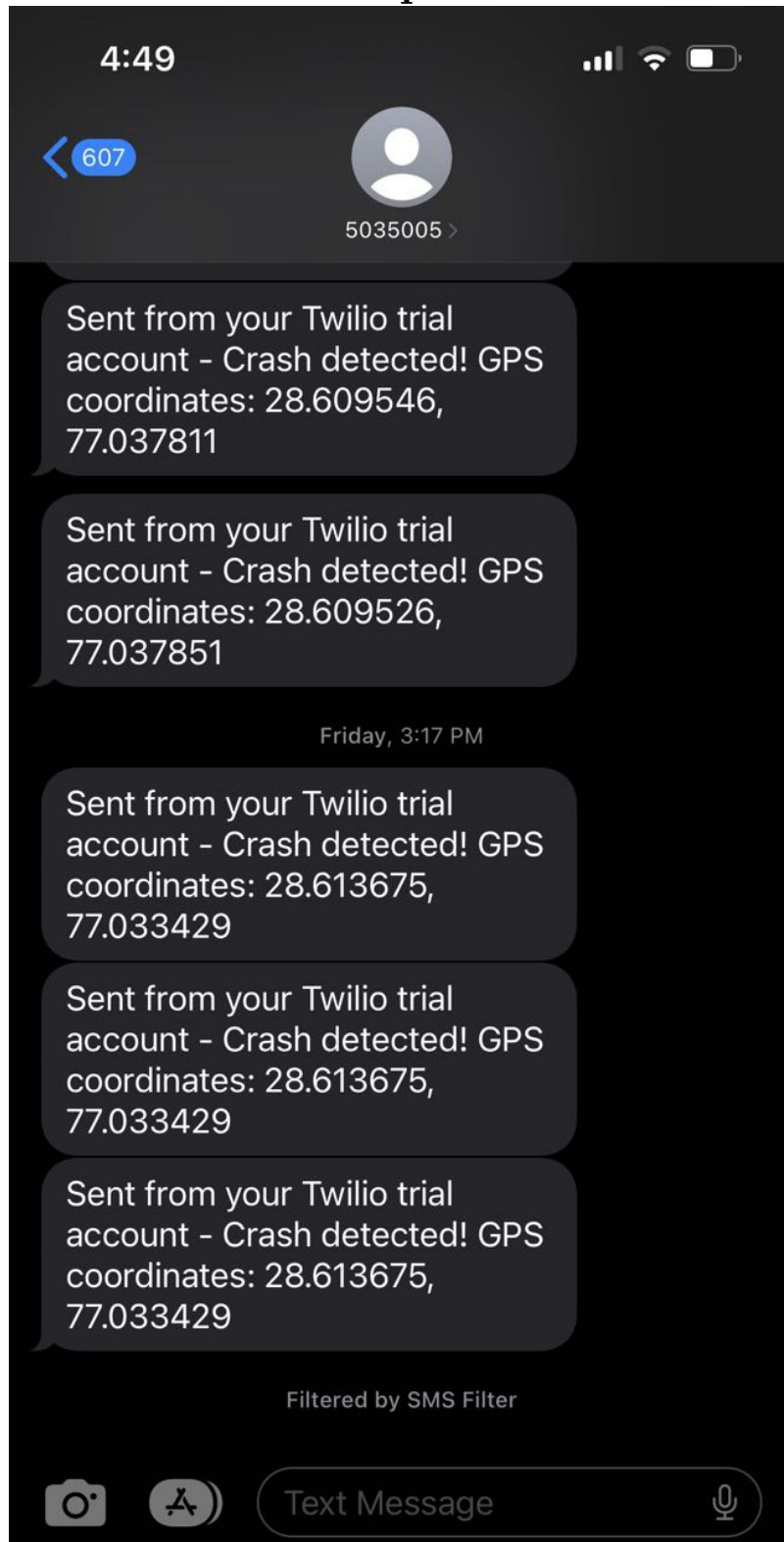
void setup() {
    Serial.begin(9600);
    while (!Serial); // wait for serial port to connect
    gpsSerial.begin(9600); // start the software serial for the GPS module
    accel.begin(); // start the accelerometer

    // Connect to Wi-Fi
    Serial.printf("Connecting to %s ", ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.print(".");
    }
    Serial.println(" connected");
}

void loop() {
    sensors_event_t event;
    accel.getEvent(&event); // read accelerometer data
    float accel_magnitude = sqrt(event.acceleration.x *
event.acceleration.x + event.acceleration.y * event.acceleration.y +
event.acceleration.z * event.acceleration.z); // calculate the magnitude
of the acceleration vector
    if (accel_magnitude >= CRASH_THRESHOLD) { // check if a crash has been
detected
        while (gpsSerial.available() > 0) {
            if (gps.encode(gpsSerial.read())) { // read GPS data
                if (gps.location.isValid()) { // check if GPS data is valid
                    String message = "Crash detected! GPS coordinates: " +
String(gps.location.lat(), 6) + ", " + String(gps.location.lng(), 6);
                    sendSMS(message);
                    Serial.println(message);
                    delay(1000); // wait for 1 second before printing again (to
avoid repeated printing)
                }
            }
        }
    }
}
}

```

Output:



6.4 Setup and Fabrication:

Setup and fabrication form a crucial phase in the creation of the IoT-Enabled Smart Safety Helmet. This process involves several key steps, from board layout design to the creation of the helmet's outer case.

- **PCB Fabrication:**

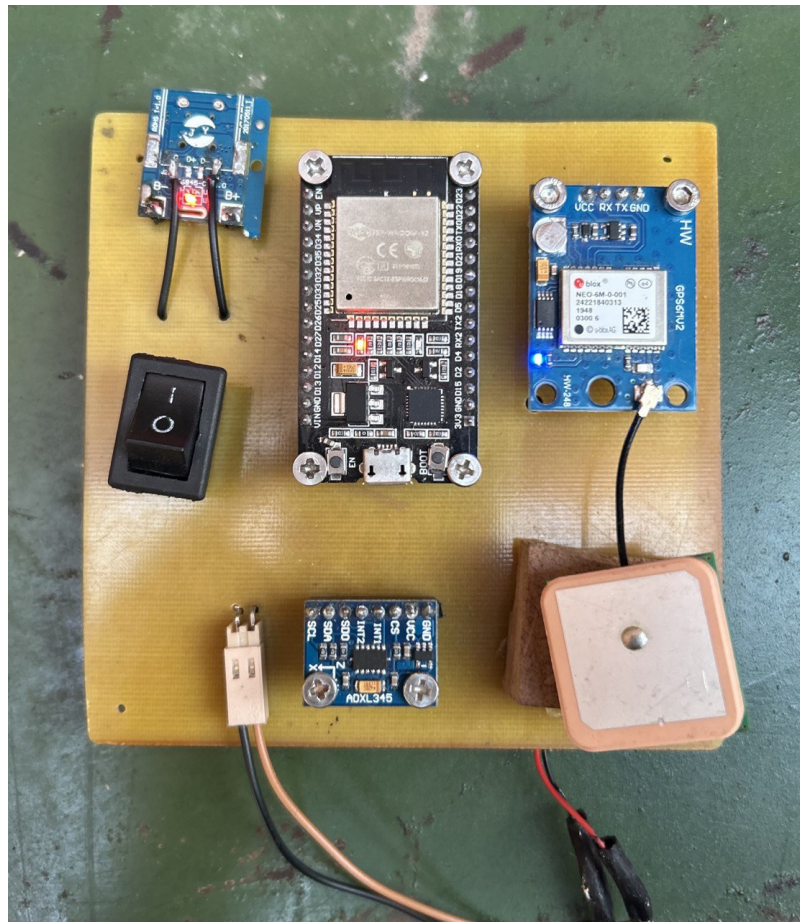
Once the board layout design is finalised, the PCB is fabricated. This process involves Preparing the copper board by cutting to the required dimensions and polishing the copper side. Etching the copper-clad board based on the layout design, applying a protective layer to prevent oxidation and drilling holes for the through-hole components . Our Texas Instruments lab helped us in achieving our PCB.

Precaution: Ensure that the PCB's quality and specifications align with the design requirements. Check for any defects in the PCB before moving to the next stage.



• Connecting the Components:

After the PCB is fabricated, components such as the ADXL345 accelerometer, Neo 6M GPS module, 3.7V lithium polymer rechargeable battery, rocker switch and ESP32 which is the main unit of our system are soldered onto the board. Header pins are first soldered, then the components are simply attached to the pins and tightened on the board with the screws. This process requires a high level of precision to ensure reliable connections and prevent damage to the components or the PCB.



• Outer Case of the Helmet:

The final stage of the fabrication process involves integrating the assembled PCB into the helmet. The design of the outer case should provide space for the PCB and all connected components without compromising the comfort and safety features of the helmet. Depending on the design, this may require modifications to the

helmet's inner shell or padding. The placement should also facilitate easy maintenance and upgrades of the system.

Ensure that the placement of the PCB does not affect the structural integrity or comfort of the helmet. Check that all components are securely fixed and do not move around when the helmet is in use.

In conclusion, the setup and fabrication of the IoT-Enabled Smart Safety Helmet involve careful planning, attention to detail, and testing at each stage to ensure a reliable and comfortable prototype.



(a) front view



(b) side view



(c) side view



(d) Top view

Figure 1: FINAL PRODUCT

7 Conclusion and Future Developments

7.1 Conclusion:

The successful completion of the IoT-Enabled Smart Safety Helmet project marks a vital stride towards amplifying road safety for two-wheeler riders. The helmet, equipped with an ADXL345 accelerometer, Neo 6M GPS module, 3.7V lithium polymer rechargeable battery, and a rocker switch, not only detects crashes but also provides crucial location information to predefined contacts in real-time. The integration of the Twilio IoT platform further boosts its communicative prowess, transforming the helmet into an intelligent, networked device that serves as a proactive safety measure for riders.

7.2 Future Developments:

We plan to continue working on the Crash detection system aka our headbud. There are some potential future developments and suggestions by our Guru Dr D.V Gadre that we will be exploring ahead:

- **Use of NAVIC instead of GPS:** For Indian consumers, substituting the Neo 6M GPS module with a NAVIC receiver could deliver enhanced location accuracy. NAVIC (Navigation with Indian Constellation), developed by ISRO, offers precise real-time positioning and timing services over India and its surrounding region. Achieving this development would involve researching NAVIC-compatible modules, testing their accuracy and reliability, and integrating them into the helmet's design, replacing the current GPS module.
- **Making the system user independent:** The system needs to be connected to the internet so that it can send the gps coordinates. Now presently we are using the user's Phone hotspot for connectivity. We will be removing this dependency by making the system self sufficient for connecting with the internet.

- **Smart Connectivity Features:** Helmet could include smart connectivity features like voice-activated controls or integration with other smart devices. This could provide riders with hands-free control over their smartphones or navigation systems, enhancing both safety and convenience.
- **Real-Time Location updates:** One of the future objectives will be to make a platform where the nominees can see the live location of the user while they are riding the bike. Real time data will be reflected on the screen displaying the location of the rider.
- **Integrating Google Maps:** We want to improve the user's convenience. Instead of the gps coordinates alone, the user should be able to open the link of the location on the google map. This will make it faster for the help to reach.
- **Use of Arduino:** We can explore using Arduino UNO in place of ESP 32 as it takes less power than the ESP.
- **Design Improvements:** Future designs might focus on enhancing the comfort and aesthetic appeal of the helmet, making it more attractive.

In essence, the Smart Safety Helmet project, while already a crucial development in two-wheeler rider safety, holds exciting potential for further evolution. The future prospects promise even greater advancements in safety, convenience, and user experience, leading to a more secure, intelligent, and rider-friendly future.

8 To T.I Lab and Sir

We would like to express our heartfelt gratitude to **D.V. Gadre Sir** and the **T.I. Lab** for their assistance and resources during our project. Though we encountered some challenges along the way, the support we received from the lab was invaluable. The equipment and resources

provided by the lab were crucial in enabling us to complete our work, and we are grateful for the opportunity to have used them. We would especially like to thank D.V. Gadre sir for his guidance and expertise, which proved invaluable at critical moments throughout the project. We feel privileged to have had access to such a knowledgeable and experienced mentor. Once again, thank you to D.V. Gadre sir and the T.I. lab for their assistance and resources during this project.

9 References

- http://dvgadre.blogspot.com/2018/09/so-you-want-to-build-electronics-project_25.html
- https://www.academia.edu/30822896/Smart_Helmet_with_Sensors_for_Accident_Prevention
- https://www.ijert.org/automatic-vehicle-accident-detection-and-messageing-system:_text=A
- <http://troindia.in/jokauurnal/ijcesr/vol5iss5part4/29-35.pdf>
- https://www.youtube.com/watch?v=euJAcVfn0C4ab_channel=Twilio