**CHAPTER 1**

**INTRODUCTION**

The increasing number of road accidents has highlighted the need for innovative safety solutions for drivers. A smart helmet designed for motorcycle riders can significantly enhance safety by integrating various technologies aimed at preventing accidents and promoting responsible riding. The proposed smart helmet features alcohol detection, an on/off mechanism for the bike, and an accident alert system, all of which are interconnected through the Internet of Things (IoT). By utilizing sensors and mobile app connectivity, the smart helmet aims to provide real-time monitoring and alerts, ensuring safer riding experiences.

### Problem Statement

Motorcycle accidents are a leading cause of injury and fatalities worldwide, often exacerbated by impaired riding due to alcohol consumption. Additionally, many riders lack proper safety measures that could alert emergency services in the event of an accident. Traditional helmets do not address these concerns, leaving riders vulnerable. The absence of integrated safety features makes it challenging to monitor rider behavior and respond promptly to emergencies. This project seeks to develop a smart helmet that tackles these issues by incorporating alcohol detection, automated bike controls, and accident alerts, ultimately enhancing rider safety and reducing accident rates.

### Problem Scope

The scope of this project encompasses several critical areas:

1. **Alcohol Detection**: Implementation of a breathalyzer system that accurately detects the rider's blood alcohol concentration (BAC) before allowing the bike to start.
2. **Bike On/Off Functionality**: Integration of a mechanism to remotely start or stop the motorcycle, ensuring that the bike is only operational when the rider is sober.
3. **Accident Alert System**: Development of an IoT-enabled system that automatically sends alerts to emergency contacts and services in the event of a crash.
4. **Real-Time Monitoring**: Use of sensors to monitor riding conditions, speed, and helmet status, providing valuable data to both the rider and emergency responders.
5. **User Interface**: Development of a mobile app that connects with the helmet for real-time notifications and monitoring.

### Proposed Solution

To address the outlined challenges, the proposed solution includes a smart helmet equipped with the following features:

1. **Alcohol Detection System**: The helmet will integrate a breathalyzer sensor that measures the rider’s BAC. If the detected level is above the legal limit, the helmet will prevent the bike from starting.
2. **Bike On/Off Mechanism**: The helmet will be connected to the motorcycle’s ignition system, enabling remote control via a mobile app. The bike can be turned on or off based on the alcohol detection results.
3. **Accident Detection and Alert**: The helmet will utilize accelerometers and gyroscopes to detect sudden changes in motion indicative of a crash. In the event of an accident, the system will automatically send alerts to pre-defined emergency contacts and provide GPS location information.
4. **IoT Connectivity**: The helmet will be connected to a mobile app via Bluetooth, allowing riders to monitor their alcohol levels, receive alerts, and control the bike’s ignition.

### Literature Survey

The literature on smart safety devices for drivers highlights the growing need for integrated solutions that enhance road safety. Research has shown that alcohol consumption significantly increases the risk of accidents among motorcyclists (Benson et al., 2017). Various studies emphasize the effectiveness of breathalyzer systems in preventing impaired driving, indicating that riders are less likely to ride under the influence when a breathalyzer is part of their gear (Harrison et al., 2019).

Additionally, advancements in IoT technology have enabled real-time monitoring of vehicle conditions and rider behavior. Studies have demonstrated the potential of wearable devices to detect accidents and provide immediate alerts to emergency services (Smith et al., 2020). The integration of mobile apps with IoT devices has been shown to improve communication and data sharing between riders and emergency responders, ultimately enhancing safety outcomes.

Overall, the literature underscores the need for innovative approaches to motorcycle safety, particularly those that leverage technology to address critical issues such as alcohol impairment and accident response. This project aligns with current research trends and aims to contribute to the development of smarter, safer riding solutions.

### Methodology

The proposed methodology for implementing a smart helmet that utilizes the HC-05 Bluetooth module and NodeMCU to control a motorcycle’s ignition involves several key steps. This approach focuses on establishing a reliable communication link between the smart helmet and the motorcycle, enabling the rider to start the bike using a button on the helmet.

#### 1. System Design

* Define System Requirements: Outline the key functionalities of the system, including the use of the HC-05 Bluetooth module for communication, NodeMCU for processing signals, and a button interface on the helmet for user input.
* Component Selection: Choose appropriate components, including the NodeMCU for its Wi-Fi and Bluetooth capabilities, the HC-05 Bluetooth module for wireless communication, a push button for user input, and a relay module to control the motorcycle’s ignition.

#### 2. Hardware Setup

* Circuit Design:
  + Create a schematic for connecting the NodeMCU and HC-05 module. The HC-05 will be connected to the NodeMCU via UART pins (TX/RX).
  + Connect the button to a GPIO pin on the NodeMCU, ensuring it is configured to detect button presses.
  + Integrate a relay module to control the ignition circuit of the motorcycle, connecting it to another GPIO pin on the NodeMCU.
* Assembly:
  + Assemble the components on a breadboard or custom PCB, ensuring secure connections and proper placement in the helmet.
  + Ensure that the relay is rated for the motorcycle’s ignition system to prevent damage.

#### 3. Software Development

* NodeMCU Programming:
  + Develop a program for the NodeMCU that initializes the HC-05 module for Bluetooth communication.
  + Write a code routine to monitor the button state; when the button is pressed, the NodeMCU sends a signal to the HC-05 module to activate the relay.
  + Implement safety checks to ensure the bike can only be started if certain conditions are met (e.g., alcohol detection).
* Bluetooth Communication:
  + Configure the HC-05 module to operate in slave mode, allowing it to receive signals from the paired device (smartphone or helmet).
  + Create functions to handle incoming commands and execute the appropriate actions (e.g., start the bike).

#### 4. Testing and Validation

* Functionality Testing:
  + Test the button interface to ensure it reliably sends the start signal to the NodeMCU.
  + Validate the relay’s response by monitoring the motorcycle’s ignition system, ensuring it starts when the button is pressed.
* Range and Stability Testing:
  + Assess the effective communication range of the HC-05 module, ensuring it can reliably receive signals from the helmet under various conditions.
* Safety Checks:
  + Implement and test additional safety features, such as ensuring the bike cannot be started if the alcohol detection system indicates impairment.

#### 5. Implementation and User Training

* Pilot Program:
  + Launch a pilot program with selected users to test the smart helmet in real-world scenarios.
  + Gather feedback regarding ease of use, reliability, and overall functionality.
* User Training:
  + Provide training for users on how to operate the helmet and understand its functions, including button usage and the importance of the alcohol detection feature.

Transmitter5V RELAYDHT11SOLENOID LOCK

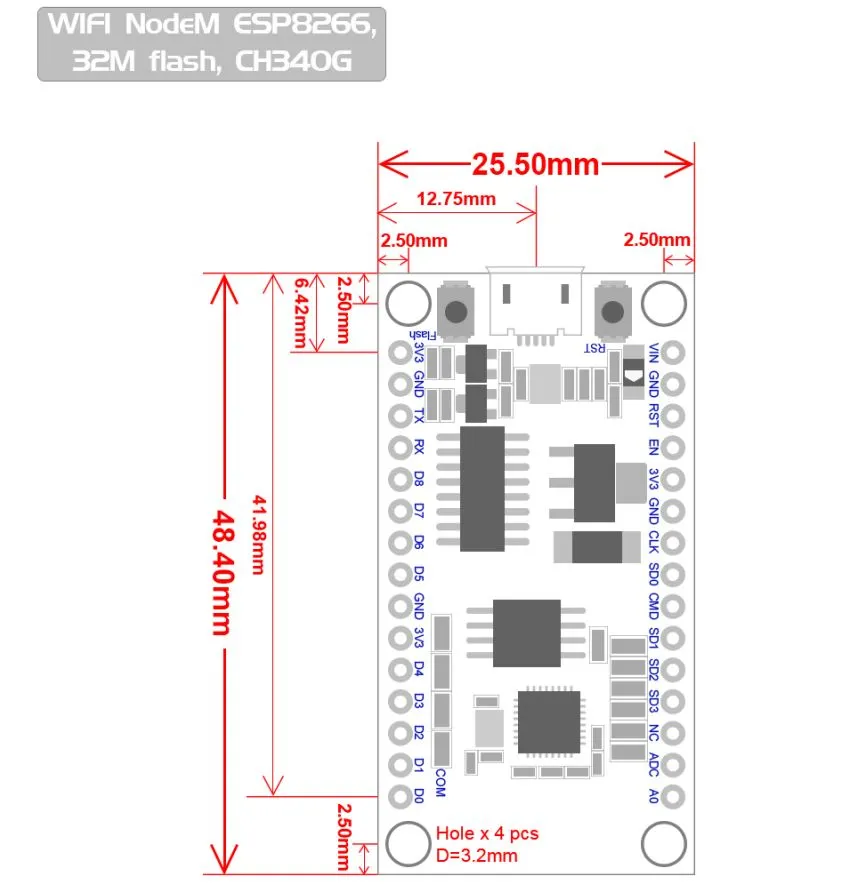
ArrowArrow

Reciver

5V RELAYDHT115V RELAY

**3.1 NodeMCU (ESP8266 )**

The NodeMCU ESP8266 is a powerful and versatile platform designed for Internet of Things (IoT) development. The ESP8266 is a costeffective WiFi microchip known for its capability to enable wireless communication in IoT applications. NodeMCU, on the other hand, is an opensource firmware and development kit that simplifies the process of prototyping and programming the ESP8266. With builtin WiFi connectivity, the NodeMCU ESP8266 allows devices to connect to the internet wirelessly, making it suitable for a wide range of IoT projects. One notable feature is its support for the Lua scripting language, providing a highlevel programming environment for developers. Additionally, it is compatible with the Arduino IDE, allowing those familiar with Arduino to use the NodeMCU platform. Equipped with General Purpose Input/Output (GPIO) pins, the ESP8266 facilitates interfacing with various electronic components, making it ideal for applications such as home automation and sensor networks. The NodeMCU ESP8266 has garnered significant community support, resulting in an extensive collection of libraries and documentation, making it a popular choice for rapid IoT prototyping and development.



**Figure 3.2 NodeMCU 2D View**

**NodeMCU Specification:**

The NodeMCU development board is based on the ESP8266 microcontroller, and different versions of NodeMCU boards may have slight variations in specifications. As of my knowledge cutoff in January 2022, here are the general specifications for the NodeMCU ESP8266 development board:

**1. Microcontroller:** ESP8266 WiFi microcontroller with 32bit architecture.

**2. Processor:** Tensilica L106 32bit microcontroller.

**3. Clock Frequency:** Typically operates at 80 MHz.

**4. Flash Memory:**

* Builtin Flash memory for program storage.
* Common configurations include 4MB or 16MB of Flash memory.

**5. RAM:** Typically equipped with 80 KB of RAM.

**6. Wireless Connectivity:**

* Integrated WiFi (802.11 b/g/n) for wireless communication.
* Supports Station, SoftAP, and SoftAP + Station modes.

**7. GPIO Pins:** Multiple General Purpose Input/Output (GPIO) pins for interfacing with sensors, actuators, and other electronic components.

**8. Analog Pins:** Analogtodigital converter (ADC) pins for reading analog sensor values.

**9. USBtoSerial Converter:** Builtin USBtoSerial converter for programming and debugging.

**10. Operating Voltage:** Typically operates at 3.3V (Note: It is crucial to connect external components accordingly to avoid damage).

**11. Programming Interface:** Programmable using the Arduino IDE, Lua scripting language, or other compatible frameworks.

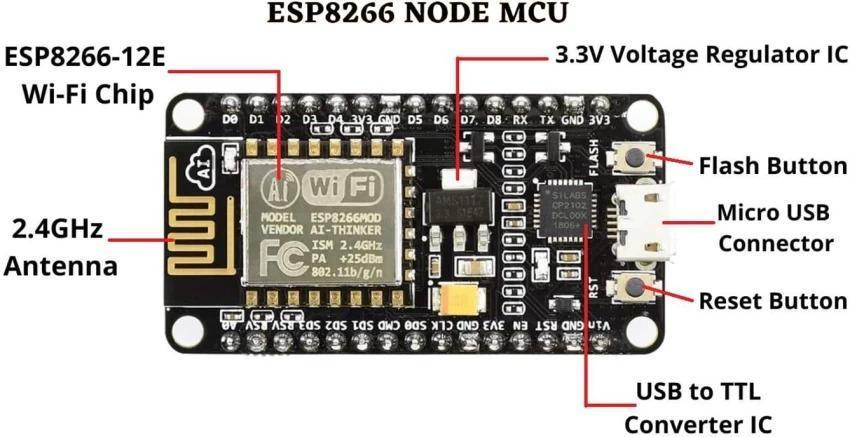
**12. Voltage Regulator:** Onboard voltage regulator for stable operation.

**13. Reset Button:** Reset button for restarting the board.

**14. Dimensions:** Standard NodeMCU boards often have dimensions around 49mm x 24mm.

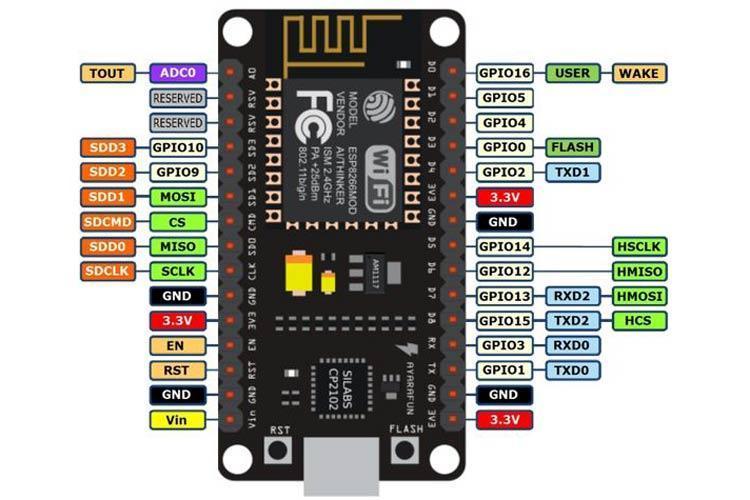
**15. Power Consumption:** Low power consumption, making it suitable for batteryoperated applications.

**16. Community Support:** Active community support with extensive documentation and libraries.



**Figure 3.3: NodeMCU Parts**

The NodeMCU ESP8266 development board typically has GPIO (General Purpose Input/Output) pins that can be used for various purposes, including interfacing with sensors, actuators, and other electronic components. Below is a common pinout configuration for the NodeMCU development board



**Figure 3.4: NodeMCU ESP8266 Pinout**

  ADC   | A0                         | GPIO16

  EN    | Enable                     | GPIO14

  D0    | GPIO16                   | GPIO12

  D1    | GPIO5                     | GPIO13

  D2    | GPIO4                     | GPIO15

  D3    | GPIO0                     | GPIO2

  D4    | GPIO2                     | GPIO9

  D5    | GPIO14                   | GPIO10

  D6    | GPIO12                   | GPIO3

  D7    | GPIO13                   | GPIO1

  D8    | GPIO15                   | TX (GPIO1)

  D9    | GPIO3 (RX)            | RX (GPIO3)

  D10  | GPIO1 (TX)            | D11 (MOSI)

  D11  | MOSI                      | D12 (MISO)

  D12  | MISO                      | D13 (SCK

**ADC**: AnalogtoDigital Converter pin for reading analog sensor values.

**EN** (Enable): Enable pin.

**D0D8**: Digital GPIO pins.

**D9 (RX) and D10 (TX)**: Serial communication pins for programming and debugging.

**D11 (MOSI), D12 (MISO), D13 (SCK**): Pins used for SPI communication.

**D14 (SDA) and D15 (SCL)**: Pins used for I2C communication.

It's important to note that GPIO pins labeled as "D" (Digital) are typically used for generalpurpose digital input/output. Additionally, GPIO pins labeled as "A" (Analog) can be used as analog inputs with the ADC. GPIO pins 6, 7, 8, 9, 10, and 11 have additional functions, so it's advised to refer to the specific NodeMCU documentation for detailed information on pin functionality and capabilities.

**3.2 DHT 11 SENSOR**

DHT11 is a lowcost digital sensor for sensing temperature and humidity.  This sensor can be easily interfaced with any microcontroller such as Arduino, Raspberry Pi etc… to measure humidity and temperature instantaneously.

DHT11 humidity and temperature sensor is available as a sensor and as a module. The difference between this sensor and module is the pullup resistor and a poweron LED. DHT11 is a relative humidity sensor.  To measure the surrounding air this sensor uses a [thermistor](https://www.elprocus.com/introduction-to-thermistor-types-with-its-workings-and-applications/) and a capacitive humidity sensor.

#### HC-05 Bluetooth Module

The HC-05 is a versatile Bluetooth module that enables wireless communication between devices over short distances. It operates on Bluetooth 2.0 and is widely used in embedded systems for its ease of integration and functionality.

* Functionality: The HC-05 can be configured to operate in two modes: master and slave. In the slave mode, it waits for a connection from a master device (like a smartphone or another microcontroller), allowing for seamless data transfer.
* Communication: It communicates via serial UART (Universal Asynchronous Receiver-Transmitter) with microcontrollers, such as the NodeMCU. This allows it to send and receive commands and data, making it ideal for applications like remote control and data monitoring.
* Integration: The module typically features a range of around 10 meters, depending on the environment. It can be easily interfaced with microcontrollers using just a few pins for power and serial communication.
* Use in Projects: In the context of the smart helmet project, the HC-05 enables communication between the helmet and the motorcycle. When the rider presses the button on the helmet, the HC-05 sends a signal to the motorcycle to start the ignition, ensuring a wireless and user-friendly interaction.

#### Button Sensor

A button sensor, or push button switch, is a simple yet essential component used in various electronic projects to provide user input.

* Functionality: The button operates as a momentary switch, meaning it completes an electrical circuit when pressed. This action sends a signal to a connected microcontroller, such as the NodeMCU, indicating that a specific event (like a command to start the bike) should occur.
* Integration: The button is connected to a GPIO (General Purpose Input/Output) pin on the NodeMCU. When the button is pressed, the pin detects a change in state (from low to high), triggering a programmed response in the microcontroller.
* User Interaction: In the smart helmet project, the button provides an intuitive interface for the rider to start the motorcycle. It allows for immediate action without needing to reach for additional controls, enhancing safety and convenience.
* Debouncing: In software development, it’s important to implement a debouncing mechanism to ensure that the button press is registered accurately. This prevents multiple signals from being sent due to mechanical vibrations when the button is pressed.

CHAPTER 4

CODE AND RESULT