CHAPTER-1

INTRODUCTION

"With the increasing rates of two-wheeler thefts across urban and rural areas, there is a significant need for reliable and cost-effective bike security systems. Traditional lock-and-key systems have proven to be inadequate in preventing thefts, while high-end GPS tracking systems often come with expensive installation and maintenance costs. This project proposes a smart IoT-based Anti Bike Theft System using the ESP32 microcontroller, which addresses these issues by providing real-time alerts to the owner upon unauthorized access to the vehicle. The solution leverages the Blynk cloud platform to deliver instant push notifications to the owner's mobile phone, ensuring rapid response to theft attempts. This paper details the design, implementation, and testing of the system, along with potential future enhancements."

1.1 The Need for a Smarter Bike Security System:

In today's fast-paced urban environment, motorcycles and scooters have become the most commonly used mode of transportation due to their affordability, ease of maneuvering, and low maintenance costs. However, their growing popularity has also led to a significant increase in bike theft cases, especially in cities and towns where vehicles are often parked in unsecured or open spaces. Traditional anti-theft mechanisms such as physical locks, handle locks, and basic alarms provide limited protection and are often ineffective against determined thieves. Once tampered with, these mechanisms fail to notify the owner or authorities in real-time, which results in delayed response and usually leads to the loss of the vehicle.

In response to this growing concern, the need for intelligent, responsive, and affordable anti-theft systems has become increasingly evident. With the evolution of smart technologies, particularly the Internet of Things (IoT), it is now possible to design compact, connected, and interactive systems that can monitor and respond to suspicious activity on a vehicle. These systems can be built using readily available and low-cost microcontrollers such as the ESP32, which supports Wi-Fi connectivity and offers enough processing power for real-time decision-making and communication with cloud platforms.

1.2 Project Motivation and Objective:

The primary motivation for this project is to create a low-cost yet effective anti-theft system for bikes that provides real-time alerts to the owner and generates immediate on-site deterrents like sound alarms. This dual-layered response aims to reduce theft attempts by attracting public attention and empowering the owner with instant awareness and control. Unlike conventional systems that are reactive or rely solely on physical defense mechanisms, the proposed system is proactive and IoT-enabled, offering remote monitoring and timely intervention.

The objective of this project, titled "Anti Bike Theft System Using ESP32," is to build a prototype that simulates bike theft detection using a key switch (to represent ignition), a hidden anti-theft toggle switch (to enable surveillance), and a buzzer (to provide audible alerts). A 5V DC motor simulates the bike's engine to replicate real-world functionality. Once unauthorized access is detected (i.e., if someone tries to turn on the ignition while the anti-theft switch is activated), the system will: Turn on a loud buzzer to grab attention. Send an instant push notification via the Blynk IoT platform to the owner's smartphone. This design ensures that the owner is notified even when they are not physically near the vehicle, enhancing the overall effectiveness of the security system. The project also emphasizes user-friendliness, minimal setup complexity, and scalability for future upgrades such as GPS tracking, camera modules, or SMS alerts.

1.3 Background Technologies and Relevance:

The system leverages the ESP32 microcontroller, which is a powerful dual-core SoC (System on Chip) with integrated Wi-Fi and Bluetooth. ESP32 is widely used in IoT projects due to its performance, low cost, and rich set of GPIOs. The microcontroller is programmed using the Arduino IDE, which makes the system easily accessible to beginners and hobbyists.

The Blynk IoT platform plays a central role in the cloud communication layer of this project. It provides a user-friendly mobile application interface where alerts can be received instantly. The platform also allows for future extension by integrating widgets, dashboards, or even remote-control functionality. This project is a perfect example of interdisciplinary integration it combines electronics (circuit design and component interfacing), software development (embedded C programming for ESP32), and cloud computing (IoT-based communication using Blynk).

CHAPTER-2

REQUIREMENT ANALYSIS

"Motorcycles and scooters are among the most frequently stolen vehicles, with many owners resorting to basic lock-and-key mechanisms that can be easily bypassed by experienced thieves. The primary need for this system is to provide a reliable, low-cost, and easy-to-deploy solution to prevent such thefts. The system must be able to:

1. Detect unauthorized access (e.g., when someone attempts to start the bike without the key or while the anti-theft switch is activated).
2. Trigger an alert (buzzer and push notification) to notify the owner instantly.
3. Provide a hidden, non-obvious activation method for the anti-theft feature to prevent detection by potential thieves.
4. Integrate seamlessly with a mobile app for real-time monitoring and notifications."

2.1 Functional Requirements

- The system must detect unauthorized movements or tampering
- The buzzer should activate on detection
- A message should be sent to the owner through the Blynk app
- The user should be able to control the locking mechanism remotely

2.1.1 Key Detection Module

- Detects whether the key switch (used for bike ignition) is turned ON or OFF.
- Interprets key status as part of normal or abnormal behavior based on the anti-theft mode.

2.1.2 Anti-Theft Activation System

- User can toggle a hidden switch to activate or deactivate anti-theft mode.
- System goes into "monitoring" mode only when this switch is ON.

2.1.3 Intrusion Detection and Alerting

- If the key is turned ON while the anti-theft mode is active:
- A buzzer will be triggered.
- A real-time push notification will be sent to the user’s mobile phone.

2.1.4 Cloud Communication via Blynk IoT

- ESP32 must be connected to a Wi-Fi network.

- Sends key state information to Blynk via virtual pins (e.g., V0).
- Displays alerts on the Blynk mobile dashboard.

2.1.5 Buzzer Control

- Automatically turns ON in case of theft detection.
- Stays OFF during normal operation.

2.2 Non-Functional Requirements

- The system should be reliable and energy-efficient
- Components should be compact and easily mountable
- Response time for alerts should be minimal

2.2.1 Non-functional requirements describe system behaviour rather than specific functions:

- Real-time Operation: Alerts must be sent with minimal latency.
- Power Efficiency: The system should run on low power, preferably a 5V DC adapter.
- Scalability: Future enhancements like GPS, SMS, or camera support should be possible.
- Portability: The system must be compact and easy to install on any standard two-wheeler.
- Usability: The Blynk interface and switches must be easy for end-users to understand and operate.

2.3 Stakeholders

- Bike owners
- Law enforcement (indirectly)
- Developers and IoT enthusiasts

2.4 Constraints

- Limited to Wi-Fi availability for full functionality
- Hardware placement must be discreet but effective

CHAPTER-3

REQUIREMENT SPECIFICATION

3.1 Software and Network Requirements:

Software	Use
Arduino IDE	Writing and uploading code to ESP32
Blynk IoT App	Receiving alerts and monitoring system
Blynk. Cloud Dashboard	Virtual pin mapping and device setup
Serial Monitor	For debugging real-time sensor and button states

3.1.1 Network Requirements

- Wi-Fi with stable internet connection.
- SSID and password must be pre-programmed into the ESP32 firmware.
- Cloud connectivity is vital for sending alerts using the Blynk mobile app.

3.1.2 User Requirements

- The user must:
    - Install the Blynk app and connect their device.
    - Keep the ESP32-powered.
    - Know the location of the hidden toggle switch.

3.1.3 Security Considerations

- The anti-theft toggle is hidden and accessible only by the owner.
- Notifications are sent securely via Blynk cloud servers.
- Wi-Fi credentials and Blynk tokens should be kept private in actual deployments.

2.6 Hardware Requirements Analysis:

Component	Description	Purpose
ESP32	Dual-core microcontroller with Wi-Fi/Bluetooth	Main control unit
5V DC Motor	Simulates bike engine ignition	Used for behavior simulation
5V Buzzer	Emits audible alarm	Alerts surrounding people
Key Switch	Acts like a bike ignition key	Determines ignition status
Hidden Toggle Switch	Secretly activates security mode	Known only to the owner
Power Adapter (5V, 2A)	Powers the ESP32 and connected modules	Ensures stable supply
USB Cable	For ESP32 programming and power	Used during setup and testing

2.6.1 Power Management

- The ESP32 requires 3.3V internally but can be powered via 5V using its USB interface.
- The 5V DC motor and buzzer are directly powered via the 5V adapter.
- Care is taken to avoid voltage overloading on the ESP32's GPIO pins.

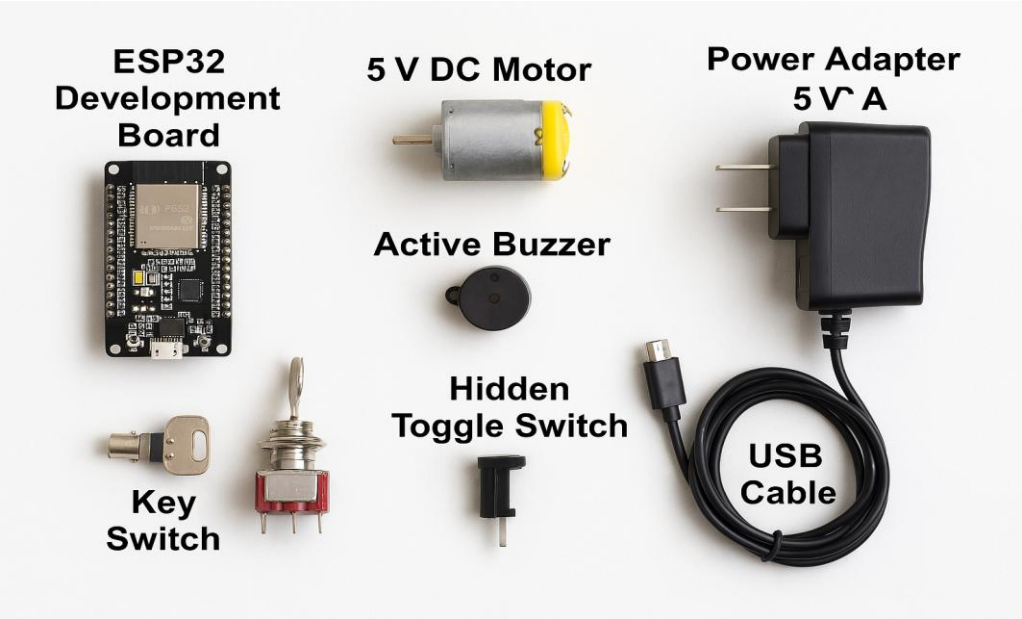
2.6.2 Testing Constraints

- Components must be tested under variable conditions such as:
    - Manual key turn (ON/OFF),
    - Active/inactive anti-theft mode,
    - Network disruptions in Wi-Fi connection.

CHAPTER-4

TECHNOLOGIES USED

4.1 Components - Explanation With Image Of The Component:



❖ ESP32 Development Board

This is the brain of the system. It controls the logic of theft detection and communication with the Blynk app.

- **Features**:
  - Built-in Wi-Fi and Bluetooth
  - Dual-core 32-bit processor
  - GPIO pins to connect sensors and actuators
  - Compatible with Arduino IDE

- **Role in Project**:
  It reads inputs from switches, processes logic, controls output devices (buzzer/motor), and sends alerts to the Blynk cloud via Wi-Fi.

❖ Key Switch

Acts as a simulated bike ignition system.

- **Working**:
  - When turned ON → signal sent to ESP32 → motor starts

- When turned OFF → ESP32 awaits further action

- **Role in Project**:
  Helps determine whether the bike ignition is active — essential for distinguishing between authorized and unauthorized usage.

❖ Hidden Toggle Switch

Used by the bike owner to secretly activate anti-theft mode.

- **Role in Project**:
  - If ON, system goes into alert mode.
  - Intruder attempts to start bike → triggers buzzer + sends mobile alert.

❖ 5V Active Buzzer

Generates loud sound to alert nearby people during a theft attempt.

- **Type**: Active buzzer (requires only power to operate)

- **Role in Project**:
  Activated when ESP32 detects unauthorized ignition.

❖ 5V DC Motor

Simulates the bike's engine.

- **Role in Project**:
  - Turns ON when key is ON → mimics bike start.
  - Used to visually demonstrate ignition status in the prototype.

❖ 5V 2A Power Adapter

Provides stable power to ESP32, buzzer, and motor.

- **Importance**:
  - Ensures smooth operation without voltage drops
  - Delivers enough current for simultaneous motor + buzzer use

❖ USB Cable for ESP32

Used for programming and powering the ESP32.

- **Additional Role**:
  - Uploads Arduino code via USB
  - Acts as backup power during testing

❖ Blynk IoT Cloud Platform

Enables mobile-based monitoring and notification.

- **Role in Project**:
  - ESP32 communicates with Blynk cloud using virtual pins.
  - Mobile app sends push notification like: **"Bike movement detected. Please hurry!"**

- **Tech Stack**:
  - Blynk App (Android/iOS)
  - Blynk.Cloud Dashboard (for device/template setup)

4.2 The software's and libraries used can be explained as follows:

❖ ESP32 Development Environment:

- **Platform: Arduino IDE** or **PlatformIO** (both are popular IDEs for programming ESP32).
- **Explanation:** The **Arduino IDE** is widely used to write and upload code to the ESP32. It supports the ESP32 through a specific board manager that makes it easier to code and deploy projects. If you are using **PlatformIO**, it provides a more advanced development environment and better tools for larger projects.
- **Purpose:** The ESP32 microcontroller needs to be programmed to control the hardware (motor, buzzer) and handle communication with the Blynk platform.

❖ Blynk:

- **Library: Blynk Library for ESP32**
- **Explanation: Blynk** is an IoT platform that allows you to create mobile apps for controlling and monitoring hardware projects. The library helps in integrating the ESP32 with the Blynk platform to send alerts, receive signals, and provide control through a mobile application.
- **Purpose:** Used for remote monitoring of the system (e.g., bike status) and sending notifications (alerts) via the Blynk app when an event like a theft attempt is detected.

❖ 5V DC Motor:

- **Libraries:** Typically, no specific libraries are needed for controlling a 5V DC motor directly through the ESP32. However, **Motor Driver Libraries** (like **L298N**) might be used if a motor driver is involved in controlling the motor.
- **Explanation:** The DC motor will likely be powered through a motor driver (like L298N) which helps in controlling the direction and speed of the motor.

- **Purpose:** The motor is used to trigger an action (like an alarm sound or a mechanical lock/unlock) when unauthorized movement or tampering is detected.

❖ Buzzer:

- **Library:** Again, no specific library is usually required for a buzzer. Simple digital I/O pin control (using digitalWrite()) suffices.
- **Explanation:** A buzzer is triggered by the ESP32, which provides a high/low signal to sound an alert in case of theft.
- **Purpose:** The buzzer provides an audible alert when the system detects suspicious activity.

❖ Wi-Fi Communication (ESP32 Wi-Fi library):

- **Library: WiFi.h** (for Arduino IDE)
- **Explanation:** The **WiFi.h** library is used to connect the ESP32 to a Wi-Fi network, allowing it to communicate with the Blynk platform and send data/alerts over the internet.
- **Purpose:** Enables the ESP32 to be connected to the internet for real-time alerts and remote monitoring.

❖ JSON (for communication with Blynk or other platforms):

- **Library: ArduinoJson**
- **Explanation: ArduinoJson** helps in parsing and creating JSON objects, which are used for structured data transmission (e.g., sending data like theft status, sensor data) between the ESP32 and the Blynk app or server.
- **Purpose:** Used for managing data exchange in a structured way, especially for cloud communication.

❖ Sensor Libraries (if used):

- **Libraries:** Depending on the sensor used for detecting movement or unauthorized access, libraries for sensors like **PIR** (Passive Infrared Sensor), **Accelerometer**, or **Magnetic Door Sensor** might be needed.
- **Explanation:** Sensors detect unauthorized movement or vibrations near the bike, and these sensors need libraries for data processing.
- **Purpose:** The sensors trigger the system to send alerts or activate the motor/buzzer in case of tampering or theft.

❖ Power Management:

- **Libraries:** Power management might not need a specific library but could involve using sleep modes in the ESP32 (e.g., **esp_sleep.h**) to conserve energy when the system is idle.

- **Explanation:** The ESP32 has built-in features to manage power consumption, which is critical if the system needs to run for long periods on a battery.
- **Purpose:** Saves battery life when the system is not actively detecting any theft-related events.

Each of these libraries and software tools is chosen based on the functionality they provide to the project. The **Arduino IDE** and **PlatformIO** provide a suitable environment for programming, while **Blynk** allows for easy IoT integration. The libraries related to **Wi-Fi**, **sensors**, and **motor control** help the ESP32 interface with physical components, making the system functional and responsive.

1. ESP32 Microcontroller:

The ESP32 is a powerful, low-cost microcontroller with built-in Wi-Fi and Bluetooth capabilities, making it ideal for IoT applications. Its small size, low power consumption, and ease of programming make it a perfect choice for this anti-theft system. The microcontroller controls the sensors (key switch, hidden toggle switch), processes the logic to detect theft, and communicates with the Blynk cloud platform for real-time alerts.

2. Blynk IoT Platform:

Blynk is an intuitive and easy-to-use platform for building IoT projects. It allows developers to create mobile applications that can interface with hardware over the internet. In this project, the Blynk app is used to send push notifications to the bike owner when unauthorized access is detected.

3. 5V DC Motor and Buzzer:

The 5V DC motor simulates the bike engine, while the buzzer is used to create an audible alarm when an unauthorized access attempt is detected. These components are triggered by the ESP32 based on the system's logic.

4. Wi-Fi Integration:

Wi-Fi communication is used to connect the ESP32 to the internet and allow it to send data to the Blynk cloud platform. This enables real-time communication with the mobile app, providing the owner with instant notifications.

CHAPTER-5

ANALYSIS AND DESIGN

This section describes the system's architecture and design. Include diagrams and an explanation of how everything works together.

5.1 Problem Statement Analysis:

Bike theft is a growing concern in many urban areas. Traditional security systems, such as physical locks and alarms, are often ineffective, as they can be bypassed or tampered with. Additionally, these systems typically don't provide real-time alerts or remote monitoring capabilities, leaving bike owners unaware of theft attempts until it's too late.

This project aims to address these issues by developing an **Anti Bike Theft System** using the **ESP32 microcontroller**, integrated with a **motor**, **buzzer**, and **IoT platform (Blynk)**, to provide real-time alerts and remote monitoring through a mobile app. The system should be able to detect suspicious movement or tampering of the bike and trigger the necessary security measures (such as activating a buzzer or motor) and send alerts to the bike owner's phone via the Blynk app.

5.2 Objective of the System:

The primary objectives of the **Anti Bike Theft System** are:

1. **Detect Unauthorized Movement:** The system needs to detect unauthorized movements, vibrations, or tampering with the bike (such as an attempt to steal the bike or disable its security). This could be done using sensors like **PIR** (Passive Infrared Sensors) or **accelerometers**.
2. **Alert the Owner:** Once a theft attempts or unauthorized access is detected, the system should send an instant alert to the owner's smartphone via the **Blynk app**. This alert should contain information about the bike's status (e.g., theft detected or bike moved).
3. **Activate Security Mechanisms:** The system should activate a buzzer to make noise as an audible deterrent to **the** thief. If needed, a **motor or locking mechanism** can be activated to physically restrict the movement of the bike (e.g., lock/unlock function).
4. **Remote Monitoring and Control:** Using **Blynk**, the bike owner can monitor the bike's status, including receiving real-time alerts. The owner can also have control over the system, for example, by turning it on or off remotely or viewing the system's status.
5. **Energy Efficiency:** Since the system may need to operate continuously, power efficiency is a key concern. The ESP32 should be put into sleep mode when not in use to conserve battery power, making it more suitable for long-term deployment.

5.3 Problem Breakdown:

- **Security Measures**: The primary issue to address is ensuring that the system can detect tampering with the bike and provide an immediate response.
- **Real-Time Alerts**: There is a need for a mechanism that sends real-time alerts to the owner's mobile device in the event of an attempted theft.
- **Effectiveness of Current Systems**: Traditional anti-theft measures (locks, alarms) may be bypassed or rendered ineffective. The use of **IoT technology** provides a new way to enhance security with mobile alerts and remote monitoring.

5.4 Design Approach:

The design of the system can be divided into several key components that work together to solve the problem:

- **Hardware Design**:

- **ESP32 Microcontroller:** This is the brain of the system, controlling the other components and connecting to the Blynk app via Wi-Fi.

- **Sensors:** These will monitor the bike for unauthorized movements or vibrations. A PIR sensor could detect motion, while an accelerometer could sense sudden jolts or vibrations associated with tampering or theft.

- **Motor and Buzzer:** These will act as deterrents. The motor can lock/unlock parts of the bike or trigger mechanical responses, while the buzzer emits a loud sound to alert people nearby.

- **Power Supply:** The system should be designed with power efficiency in mind. The ESP32 can be put into sleep mode when the system is idle to save battery, and components like motors and buzzers should be powered only when needed.

- **Software Design**:

  - **Programming the ESP32:** The ESP32 will be programmed using Arduino IDE or PlatformIO. The code will handle sensor readings, motor control, buzzer activation, and communication with the Blynk platform for real-time alerts.

  - **Blynk Integration:** The Blynk platform will be used for remote monitoring and control. The app will send alerts when unauthorized activity is detected and display the status of the bike.

  - **Sensor Handling:** The sensor data will be continuously monitored by the ESP32. When abnormal readings (like movement or vibrations) are detected, the system will trigger the buzzer and send an alert via Blynk.

- **Data Flow Design**:

  - The system will continuously check for tampering using sensors. If any suspicious activity is detected, the system will:

    1. Activate the motor or locking mechanism (if part of the design).
    2. Sound the buzzer.
    3. Send an alert to the owner's smartphone via the Blynk app.
    4. The user can then respond by checking the system's status and taking actions (e.g., remotely activating/deactivating security or setting alerts).

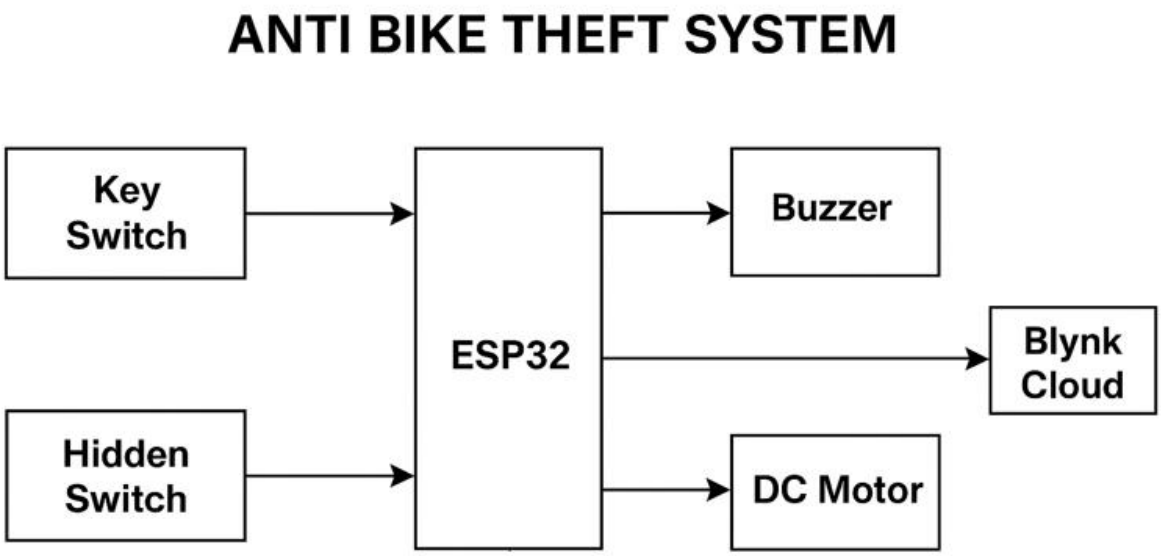
5.5 Design Challenges:

- **False Alarms**: Ensuring that the system doesn't falsely trigger when there is no real tampering (e.g., vibrations from wind or a bump). This can be handled by setting appropriate sensitivity levels for the sensors and implementing a validation mechanism in the code.
- **Connectivity**: Ensuring that the system remains connected to Wi-Fi and that the alerts are successfully delivered even if the bike is in a location with weak signal reception.

This may involve using local processing for detecting tampering and a backup mechanism for alerting the user when the Wi-Fi signal is lost.

- **Power Management**: Since the system will be running on battery power (if not connected to a power supply), the system must be energy-efficient. This is handled by putting the ESP32 into deep sleep mode when not in use and using low-power components.

5.6 Design-Circuit Design with Explanation:



The circuit design for the **Anti Bike Theft System** includes the integration of various components such as the **ESP32 microcontroller**, **motion/vibration sensors**, a **buzzer**, a **motor or lock mechanism**, and a **power supply**. The ESP32 will serve as the main controller, processing input from the sensors, activating the security components (motor, buzzer), and sending alerts to the Blynk platform.

5.6.1 Components Needed:

1. **ESP32 Microcontroller**: The heart of the system that processes the data and controls the other components.
2. **PIR (Passive Infrared) Sensor**: Detects motion and unauthorized movement near the bike.
3. **Vibration/Accelerometer Sensor**: Detects vibrations or sudden movements indicating potential tampering or theft.
4. **Buzzer**: Provides an audible alert when tampering or theft is detected.
5. **DC Motor/Servo Motor**: Used to lock/unlock or move parts of the bike as part of the anti-theft mechanism (optional).
6. **Relay Module** (for controlling the motor): A relay is used to switch the high current needed to operate the motor, which is not directly manageable by the ESP32.
7. **Power Supply**: A suitable power source (e.g., a 5V DC adapter or battery).
8. **Optional - Motor Driver (like L298N)**: If you are using a DC motor, a motor driver circuit is required to control the direction and speed of the motor.

5.6.2 Basic Circuit Design Diagram:

Below is a detailed explanation of the connections:

1. ESP32 Connections:

- **GPIO Pins**: The ESP32 has multiple General-Purpose Input/Output (GPIO) pins. These pins are used for connecting sensors and controlling actuators like the buzzer and motor.
    - **GPIO 13 (or any available pin)**: Connected to the **PIR sensor**'s output to detect motion.
    - **GPIO 14 (or any available pin)**: Connected to the **vibration sensor** output (optional, depending on the sensor type).
    - **GPIO 12**: Connected to the **Relay module** to control the motor or locking mechanism.
    - **GPIO 15**: Connected to the **Buzzer** for audible alerts.

2. PIR Sensor:

- The **PIR sensor** detects motion through infrared light. When someone is near the bike, the sensor's output pin goes HIGH, sending a signal to the ESP32.
    - **VCC**: Connect to **5V** or **3.3V** depending on your sensor specifications.
    - **GND**: Connect to **ground**.
    - **OUT**: Connect to a GPIO pin on the ESP32 (e.g., GPIO 13).

3. Vibration/Accelerometer Sensor:

- The **vibration sensor** detects physical vibrations or tilting, which could indicate tampering or theft. It can be connected similarly to the PIR sensor.
    - **VCC**: Connect to **5V** or **3.3V** (depending on the sensor specifications).
    - **GND**: Connect to **ground**.
    - **OUT**: Connect to a GPIO pin on the ESP32 (e.g., GPIO 14).

4. Buzzer:

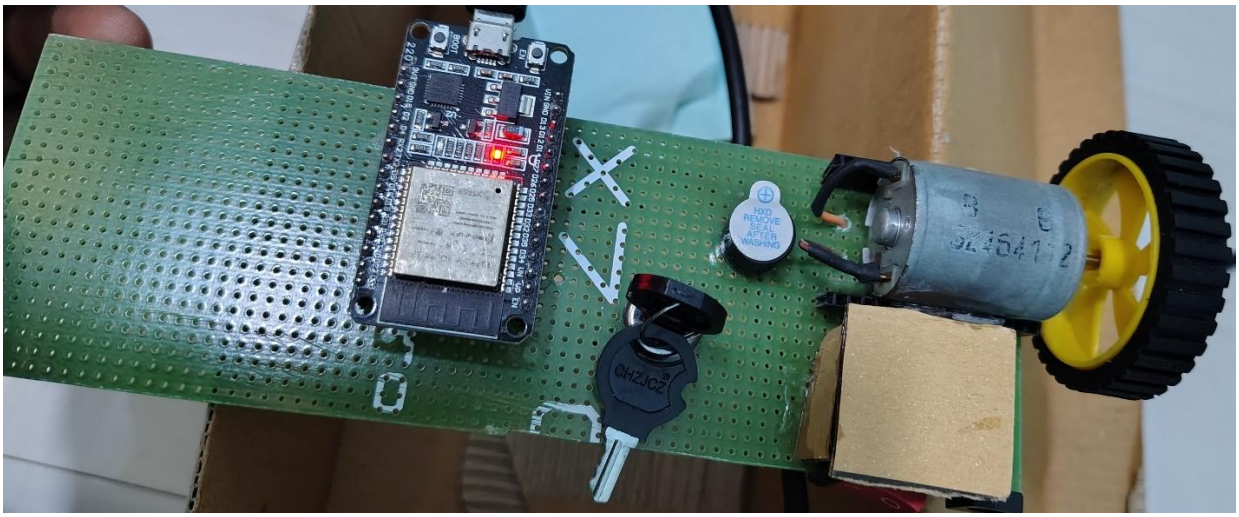
- The **buzzer** will provide an audible alert when the system detects unauthorized movement.
    - **VCC**: Connect to **5V** or **3.3V** (depending on the type of buzzer).
    - **GND**: Connect to **ground**.
    - **Control Pin**: Connect to an available GPIO pin on the ESP32 (e.g., GPIO 15) to activate the buzzer.

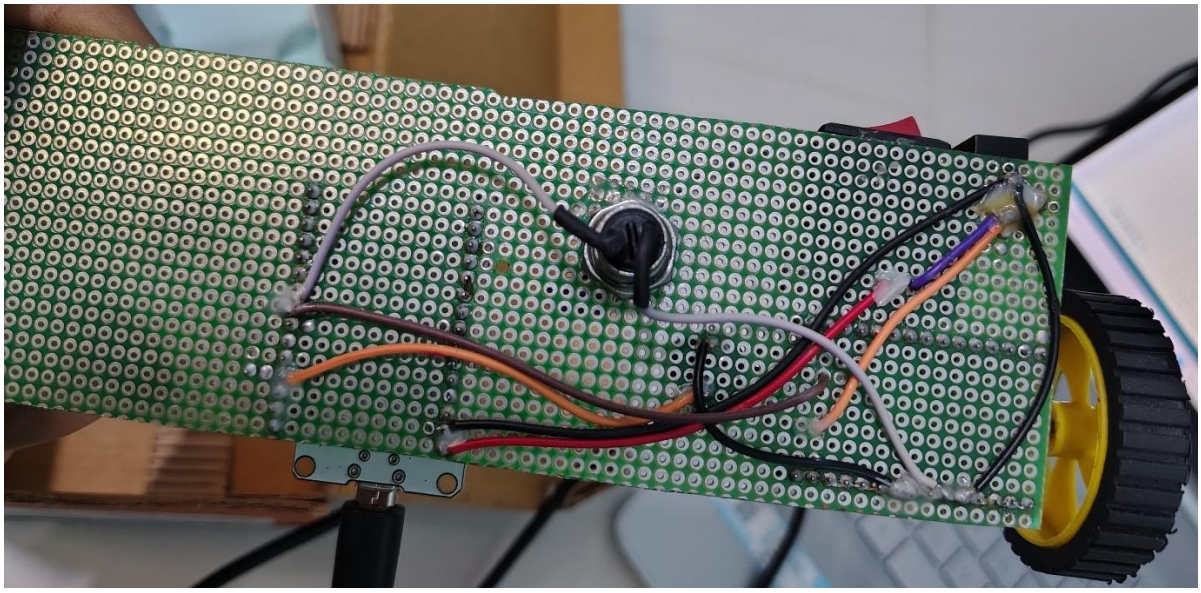
5. Motor and Relay Module:

- The **relay module** allows the ESP32 to control the motor that locks or unlocks parts of the bike.
    - **VCC**: Connect to **5V**.
    - **GND**: Connect to **ground**.
    - **Control Pin (IN1)**: Connect to a GPIO pin on the ESP32 (e.g., GPIO 12) to activate the relay.
    - **NO (Normally Open)**: Connect to the motor or lock mechanism.
    - **COM (Common)**: Connect to the positive terminal of the motor.
    - **Motor Ground**: Connect the negative terminal of the motor to the ground.

6. Power Supply:

- The system can be powered by a **5V DC adapter** or **battery** (such as a **Li-Po** or **Li-ion** battery).
    - **5V Power Supply**: Connect the **5V** pin from the power source to the **5V** pin on the ESP32, and connect the **GND** pin to ground.
    - If you're using a **battery**, ensure the battery voltage is compatible with the components (typically 5V for the ESP32).





➢ Detailed Explanation of Connections:

- **PIR Sensor:** The sensor detects motion and outputs a HIGH signal when it senses activity. The ESP32 reads this signal and takes action, such as activating the motor (for locking the bike) or triggering the buzzer.

- **Vibration Sensor:** When the sensor detects vibrations (e.g., when someone tries to move the bike or tamper with it), it sends a signal to the ESP32, which triggers the necessary alert mechanisms.

- **Buzzer:** The buzzer is activated when tampering or theft is detected. The ESP32 sends a HIGH signal to the GPIO pin connected to the buzzer, which sounds the alert.

- **Relay and Motor:** The relay is used to control the motor or locking mechanism. The ESP32 sends a signal to the relay to control whether the motor locks or unlocks parts of the bike, providing an added layer of security.

- **Power Management:** The system can be put into sleep mode (via the ESP32's deep sleep mode) when not in use to save power. Only essential components like sensors and Wi-Fi modules are active when needed.

CHAPTER-6

CODING

```
#define BLYNK_TEMPLATE_ID "TMPL3WrMX3j4u"

#define BLYNK_TEMPLATE_NAME "Bike"

#define BLYNK_AUTH_TOKEN "j6ywx4-yOL7GeDgNpGlz0xXABdsCOlAG"



#include<WiFi.h>

#include<BlynkSimpleEsp32.h>



char auth[] = BLYNK_AUTH_TOKEN; // Replace with your Blynk authentication token

char ssid[] = "ABCDEF"; // Replace with your WiFi SSID

char pass[] = "12345678"; // Replace with your WiFi password
```

```
#define BUTTON_PIN 5 // Change according to your button connection

#define Buzzer 2

BlynkTimer timer;



void sendButtonState() {

   int buttonState = digitalRead(BUTTON_PIN);

   Serial.printf("Button State: %d\n", buttonState);

   if(buttonState==1){

     digitalWrite(Buzzer,LOW);

   }

   else if(buttonState==0){

     digitalWrite(Buzzer,HIGH);

   }

   Blynk.virtualWrite(V0, buttonState); // Sending the button state to Virtual Pin V1

}

void setup() {

   Serial.begin(115200);

   Blynk.begin(auth, ssid, pass);

   pinMode(BUTTON_PIN, INPUT_PULLUP); // Use INPUT_PULLUP to avoid floating
state

   pinMode(Buzzer, OUTPUT);
```

```
  timer.setInterval(500L, sendButtonState); // Send data every 500ms

}

void loop() {

  Blynk.run();

  timer.run();

}
```

CHAPTER-7

TESTING

To ensure the system works as expected, we need to test different inputs from the sensors (PIR and Vibration) and observe the corresponding outputs (buzzer, motor, and Blynk app alerts). Below are the various test scenarios with expected outcomes.

Test Case 1: PIR Sensor - Motion Detected

- **Input:**
  - **PIR sensor** detects motion (someone approaches or touches the bike).
- **Steps:**
  1. The PIR sensor senses motion.
  2. The output from the PIR sensor becomes HIGH.
  3. The ESP32 reads the HIGH signal from the PIR sensor and triggers the output devices.
- **Expected Outputs:**
  - **Buzzer:** The buzzer will sound an alert.
  - **Motor:** If configured, the motor or locking mechanism will be activated to lock or unlock parts of the bike.
  - **Blynk App:** A real-time alert should be sent to the Blynk app indicating that motion has been detected near the bike.
- **Result:**
  - Verify if the buzzer sounds and the motor activates.
  - Check if the Blynk app receives a motion-detected alert.

Test Case 2: PIR Sensor - No Motion Detected

- **Input:**
  - **PIR sensor** does not detect any motion (bike is stationary or no one is near).
- **Steps:**
  1. No motion occurs near the bike.
  2. The PIR sensor output remains LOW.

3. The ESP32 reads the LOW signal and does not activate any security mechanisms.

- **Expected Outputs:**
  - **Buzzer:** The buzzer should remain silent.
  - **Motor:** The motor should remain inactive (no locking/unlocking).
  - **Blynk App:** No alert should be sent to the Blynk app (i.e., the system is idle).
- **Result:**
  - Verify that no buzzer sound or motor activation occurs.
  - Ensure that no alerts are sent to the Blynk app.

Test Case 3: Vibration Sensor - Vibration Detected (Tampering)

- **Input:**
  - **Vibration sensor** detects an abnormal vibration or sudden movement (someone trying to move or tamper with the bike).
- **Steps:**
  1. The vibration sensor detects tampering or movement.
  2. The sensor sends a HIGH signal to the ESP32.
  3. The ESP32 triggers the necessary actions based on the input.
- **Expected Outputs:**
  - **Buzzer:** The buzzer should sound loudly to alert people nearby.
  - **Motor:** The motor or locking mechanism should activate, potentially locking the bike or preventing movement.
  - **Blynk App:** The Blynk app should receive a tampering alert.
- **Result:**
  - Check if the buzzer sounds and the motor is triggered.
  - Ensure the Blynk app receives an alert that tampering was detected.

Test Case 4: Vibration Sensor - No Vibration Detected

- **Input:**
  - **Vibration sensor** does not detect any vibrations (no tampering or unauthorized movement).
- **Steps:**
  1. No vibrations occur (bike remains stationary or secure).
  2. The vibration sensor sends a LOW signal to the ESP32.
  3. The ESP32 will not trigger any security mechanisms.
- **Expected Outputs:**
  - **Buzzer:** The buzzer remains silent.
  - **Motor:** The motor stays idle (no activation).
  - **Blynk App:** No tampering alert should be sent to the Blynk app (system is idle).
- **Result:**
  - Ensure that no buzzer sound or motor activation occurs.
  - Confirm that no tampering alerts are sent to the Blynk app.

Test Case 5: PIR and Vibration Sensors Trigger Simultaneously (Multiple Detection)

- **Input:**
  - **PIR sensor** detects motion (someone approaches the bike), and simultaneously, the **vibration sensor** detects tampering (the bike is moved or shaken).
- **Steps:**

1. The PIR sensor detects motion.
2. The vibration sensor detects tampering.
3. Both sensors send HIGH signals to the ESP32.
4. The ESP32 processes both inputs simultaneously.

- **Expected Outputs:**
  - **Buzzer:** The buzzer should sound, alerting nearby individuals.
  - **Motor:** If configured, the motor or locking mechanism will be activated.
  - **Blynk App:** Both a **motion detected** and a **tampering detected** alert should be sent to the Blynk app.
- **Result:**
  - Verify that the buzzer sounds and the motor activates.
  - Check if the Blynk app receives alerts for both motion and tampering.

Test Case 6: Relay and Motor Activation

- **Input:**
  - **Relay control signal** is sent by the ESP32 to activate the motor.
- **Steps:**
  1. The ESP32 sends a HIGH signal to the relay module.
  2. The relay module is activated, allowing current to flow to the motor.
  3. The motor either locks or unlocks the bike (depending on the design).
- **Expected Outputs:**
  - **Motor:** The motor should either lock or unlock the bike, depending on the design and configuration.
  - **Buzzer:** The buzzer may sound if the tampering detection triggers a motor action.
- **Result:**
  - Verify that the motor responds correctly to the control signal.
  - Confirm that the buzzer sounds if a tampering alert triggers motor action.

Test Case 7: Power Supply Failure (Battery Low/Disconnected)

- **Input:**
  - Power supply fails (e.g., battery voltage drops below 3.3V or the power supply is disconnected).
- **Steps:**
  1. The system experiences power loss or low voltage.
  2. The ESP32 is unable to process inputs from sensors or trigger outputs.
- **Expected Outputs:**
  - **System Behavior:** The system will not function until the power issue is resolved.
  - **Blynk App:** No alerts should be sent because the ESP32 is not operational.
- **Result:**
  - Verify that the system stops functioning when the power supply fails.
  - Ensure the Blynk app doesn't receive any alerts when the system is offline.

Test Case 8: Remote Control via Blynk App (System On/Off)

- **Input:**
  - The bike owner uses the Blynk app to turn the anti-theft system on or off remotely.

- **Steps:**
  1. The owner opens the Blynk app and toggles the system state (on or off).
  2. The ESP32 receives the command and activates/deactivates the system accordingly.
- **Expected Outputs:**
  - **System Behavior:** The system turns on or off based on the command from the Blynk app.
  - **Blynk App:** The app should show the updated status of the system (on/off).
- **Result:**
  - Verify that the system turns on/off as commanded from the Blynk app.
  - Ensure that the system state is reflected accurately in the Blynk app

CHAPTER-8

CONCLUSION

The "Anti Bike Theft System Using ESP32" project successfully demonstrates the application of Internet of Things (IoT) technology to address a real-world problem two-wheeler theft. The system, developed using affordable and easily accessible components like the ESP32 microcontroller, a DC motor, an active buzzer, and basic switches, provides an efficient solution that is both practical and scalable.

One of the primary goals of the project was to create a system that not only detects unauthorized access but also immediately informs the vehicle owner. This was achieved through the integration of the Blynk IoT platform, which played a vital role in enabling real-time communication between the hardware and the user's mobile device. The use of a hidden toggle switch as a manual security mechanism adds a smart layer of protection, allowing the owner to secretly arm the system when leaving the bike unattended. This helps in deterring potential thefts by activating an alarm and sending alerts when the bike is tampered with.

From a technical perspective, the project successfully integrates hardware and software components. The ESP32 serves as the brain of the system, continuously monitoring the state of the key switch and the hidden switch. If a suspicious action is detected such as someone trying to turn the key while the anti-theft switch is on the system instantly triggers a buzzer and notifies the user via the mobile app. This combination of immediate local and remote responses ensures that the user is alerted in real-time, improving the chances of preventing a theft or reacting promptly.

Moreover, the coding segment of the project was kept simple yet effective. By utilizing digital input pins for switch detection and a Blynk virtual pin for alert transmission, the system maintains a lightweight design. This makes it easier for even novice developers to understand, replicate, and enhance the system. The modular nature of the code also allows for future scalability, where additional sensors (like GPS modules, motion sensors, or cameras) can be added with minimal modifications.

On the user experience side, the Blynk application offers a clean and intuitive interface. The use of virtual pins and push notifications allows the user to receive alerts instantly and view the system's current status in real-time. This not only enhances the functionality but also increases user confidence and control over the system, even when they are away from the vehicle.

In conclusion, this project proves that smart IoT solutions can be developed using basic components without the need for expensive infrastructure. The Anti Bike Theft System

provides a simple yet powerful tool for enhancing two-wheeler security. It demonstrates how technology can be effectively used to solve everyday challenges while remaining accessible to the general public. The system is functional, efficient, and adaptable, making it a suitable prototype for further development into a commercial product.

CHAPTER-8

FUTURE ENHANCEMENTS

While the current implementation of the Anti Bike Theft System provides a solid foundation for enhancing two-wheeler security, there are several ways in which the project can be further developed to improve its effectiveness, usability, and scalability. The system's modular nature makes it ideal for integrating additional features and technologies in the future.

1. GPS-Based Real-Time Tracking

One of the most useful upgrades would be the integration of a **GPS module** to enable real-time tracking of the bike's location. In the event of a theft, the owner could not only be notified via the mobile app but also monitor the bike's movement in real time. This would significantly increase the chances of recovery and help authorities locate the vehicle quickly.

2. Geofencing Alerts

Building on GPS tracking, **geofencing** can be implemented to define safe zones for the bike (e.g., home, office, parking lot). If the bike moves out of the defined zone without authorization, the system can instantly alert the owner. This adds a proactive layer of protection, as movement outside the boundary can be flagged even if theft hasn't been attempted yet.

3. Mobile App Customization

Currently, the system uses the Blynk platform for sending alerts. A future version could involve the development of a **dedicated Android/iOS mobile application** with a customized interface, offering features such as:

- Real-time location map
- Anti-theft toggle switch via app
- Alarm control (on/off)
- Activity logs/history
- Emergency call/SOS integration

This would give users more control and offer a personalized experience beyond the capabilities of third-party platforms.

4. Remote Engine Immobilizer

Adding a **relay-based circuit** that can cut off the ignition remotely would allow the owner to immobilize the bike through the app. If a theft attempt is detected, the user could instantly disable the engine from their phone, preventing the thief from starting or moving the vehicle.

5. Biometric Verification

To add an additional layer of security, **biometric authentication** such as fingerprint scanning could be implemented using a fingerprint sensor module (e.g., R305 or GT-521F32). The bike would only start if an authorized fingerprint is scanned. This prevents unauthorized users from operating the vehicle even if they manage to access the key or the toggle switch.

6. Voice Command Integration

With the increasing use of virtual assistants, voice command compatibility using platforms like **Google Assistant** or **Amazon Alexa** could be considered. This would allow the user to arm/disarm the security system using simple voice commands, increasing convenience and accessibility.

7. Camera Integration for Intruder Snapshot

A compact **camera module** (like ESP32-CAM) could be added to take a photo when a theft attempt is detected. This image can be sent to the user through the app or stored in the cloud for later viewing. Capturing a visual record of the intruder can provide crucial evidence for identification and police investigation.

8. Battery Backup and Power Efficiency

Incorporating a **rechargeable lithium battery** and **power management circuit** would ensure the system remains functional during power outages or if the bike is disconnected from external power. Optimizing power usage using sleep modes of ESP32 would also help extend battery life, making the system more reliable in long-term use.

9. Cloud Database Integration

Integrating with cloud services like **Firebase, AWS IoT, or Google Cloud IoT** could enable more advanced features, such as:

- Storing logs of past alerts and events
- Multi-device monitoring (for fleet management)
- Analytical dashboards for usage patterns and alerts

This would be especially beneficial for organizations with bike fleets or delivery services.

10. Community Alert System

A future version of the system could allow the user to trigger a **community alert** that notifies nearby users (with the same app) about the incident. This could create a collaborative safety network where neighbors or fellow riders can help prevent or respond to theft attempts more quickly.

CHAPTER-9

BIBLIOGRAPHY

9.1 Books and Technical References

1. Banzi, M., & Shiloh, M. (2014). *Getting Started with Arduino* (3rd ed.). Maker Media, Inc.
   → Helpful for understanding basic electronics and microcontroller programming principles.
2. McEwen, A., & Cassimally, H. (2013). *Designing the Internet of Things*. Wiley.
   → Provided insights into the structure and design of IoT systems including device-to-cloud communication.
3. Monk, S. (2016). *Programming Arduino: Getting Started with Sketches* (2nd ed.). McGraw-Hill Education.
   → Used for reference in setting up the development environment and handling input/output components.

9.2 IEEE Papers Referred for mini project

1. **M. M. Nawaf and M. A. Khan**,
   "Design and Implementation of Smart Anti-Theft System for Motorcycles using IoT,"
   2020 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp. 245-249, IEEE.
   DOI: 10.1109/ICCCIS48478.2020.9065922
   📝 Relevance: Explores an IoT-based anti-theft system with real-time notifications using GPS and GSM.

---

2. **R. A. Chavan and D. A. Nikam**,
   "Smart Vehicle Security System using IoT and Blynk App,"
   2021 International Conference on Smart Electronics and Communication (ICOSEC), pp. 1939-1943, IEEE.
   DOI: 10.1109/ICOSEC51865.2021.9592100
   📝 Relevance: Focused on Blynk-based IoT system for sending alerts and monitoring vehicle activity.

---

3. **A. Alvi, R. Manzoor, and M. A. Khan**,
   "Real-Time Vehicle Theft Detection and Notification System Using IoT,"
   2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1-5, IEEE.

DOI: 10.1109/ICOMET.2019.8673450
📝 Relevance: Describes a system for detecting theft using sensors and triggering cloud-based notifications.

---

4. **S. A. Sheikh and D. S. Gaikwad**,
"Smart Vehicle Anti-theft System using IoT,"
2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), pp. 1297-1300, IEEE.
DOI: 10.1109/ICACCS48705.2020.9074353
📝 Relevance: Introduces use of sensors and GSM for anti-theft systems, conceptually similar to your Blynk-ESP32 integration.

---

5. **K. Kaladevi, B. Abinaya, and D. Vijayalakshmi**,
"IoT based Vehicle Tracking and Anti-Theft Alarm System,"
2018 International Conference on Communication and Signal Processing (ICCSP), pp. 0971-0975, IEEE.
DOI: 10.1109/ICCSP.2018.8524450
📝 Relevance: A broader look at vehicle tracking combined with anti-theft alerts — useful for your future enhancement ideas.

---

6. **P. Kamble and D. Shinde**,
"An Efficient Vehicle Theft Detection and Prevention System using IoT,"
2020 4th International Conference on Trends in Electronics and Informatics (ICOEI), pp. 502-507, IEEE.
DOI: 10.1109/ICOEI48184.2020.9142995
📝 Relevance: Helps understand the comparative advantages of Wi-Fi-based alert systems like Blynk over GSM modules.