**NATIONAL INSTITUTE OF TECHNOLOGY,**

**ROURKELA**

****

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**Product Development Laboratory**

**(EC 3712)**

**Digital Platforms for IoT:**

**Sensors, Actuators, Network Interfaces with ESP32, Using Arduino IDE, ReactJS, Flutter, NodeJS, PostgreSQL**

**Presented By:**

**Satwik Sanket Ratha (122EC0668)**

**Sanseet Suna (122EC0585)**

**Lavanya Tamgade (122EC0354)**

**Shyamsunder Tard (122EC0140)**

**Pratyush Senapati (122EC0138)**

**Digital Platforms for IoT:**

**Sensors, Actuators, Network Interfaces with ESP32, Using Arduino IDE, ReactJS, Flutter, NodeJS, PostgreSQL**

**Abstract:**

****The increasing demand for automation in residential and industrial environments necessitates the development of an efficient and scalable IoT-based control system for electrical appliances. In this project, an integrated digital platform utilizing ESP32, sensors, actuators, and network interfaces is proposed to enable real-time remote monitoring and control of appliances such as fans, bulbs, motors, and air conditioners. The system is implemented using the Arduino IDE for firmware development, while ReactJS and Flutter provide intuitive web and mobile interfaces, respectively. A NodeJS-based backend with PostgreSQL ensures seamless data management and device coordination. The proposed solution is initially designed for smart home automation, with future scalability towards industrial applications, enabling centralized control of large-scale electrical systems. By leveraging IoT and cloud-based technologies, the platform enhances energy efficiency, reduces manual intervention, and improves operational reliability. The developed system offers a cost-effective and user-friendly approach to automation, catering to homeowners, industries, and businesses seeking intelligent control solutions.

******Introduction:**

**Internet of Things (IoT):**

The Internet of Things (IoT) refers to a network of interconnected devices embedded with sensors, software, and communication technologies, enabling them to collect, share, and act upon data autonomously. IoT bridges the gap between the physical and digital worlds, finding applications in industries like healthcare, agriculture, manufacturing, smart cities, and

homeautomation. Its scope includes improving efficiency, enabling real-time monitoring, and providing actionable insights across diverse domains.

**Digital Platforms for IoT:**

Digital platforms for IoT provide the infrastructure necessary to connect devices, process data, and manage interactions between hardware and software. These platforms can be implemented using various approaches, such as:

Cloud-Based Platforms: Services like AWS IoT, Google Cloud IoT, and Azure IoT Hub, which offer scalable solutions for managing IoT devices and data.

Edge Computing Platforms: Systems like NVIDIA Jetson and Raspberry Pi that process data locally, reducing latency and dependency on cloud connectivity.

Hybrid Platforms: Combining cloud and edge computing to balance real-time processing and long-term data storage.

In this project, the digital platform for IoT will be implemented using **ESP32**, **Arduino IDE**, **ReactJS**, **Flutter**, **NodeJS**, and **PostgreSQL**. This method provides a balance between cost-effectiveness, ease of development, and flexibility.

**ESP32:**

The **ESP32** is a 32-bit system-on-chip (SoC) microcontroller by **Espressif Systems**, known for its high performance, low cost, and IoT versatility. It features a dual-core or single-core Tensilica Xtensa LX6 processor running at up to 240 MHz, with 520 KB of SRAM and typically 4 MB of flash memory. The ESP32 supports Wi-Fi (802.11 b/g/n) and Bluetooth 4.2/BLE for wireless communication and includes a range of peripherals like GPIO, SPI, I2C, UART, PWM, ADC, and DAC for sensor and actuator integration. Its low-power design and security features, such as secure boot and flash encryption, make it ideal for battery-powered and secure IoT applications. Widely used in smart homes, wearables, and industrial automation, the ESP32 offers a powerful and cost-effective solution for IoT Development.

**Components and Their Roles in This Project:**

* **Sensors**: Sensors are critical for collecting environmental and operational data such as temperature, humidity, motion, or light. In this project, they will interface with the ESP32 to capture and relay data to the system.
* **Actuators**: Actuators convert digital commands into physical actions, like turning on/off appliances or moving mechanical components. They will be controlled by the ESP32 based on sensor inputs and user commands.
* **Network Interfaces with ESP32**: The ESP32, equipped with Wi-Fi and Bluetooth capabilities, acts as the central hub, enabling communication between sensors, actuators, and the digital platform. It ensures real-time data transfer and device control.
* **Arduino IDE**: The Arduino IDE is used for programming the ESP32, offering a user-friendly environment and extensive libraries for managing hardware interactions.
* **ReactJS**: ReactJS is utilized to create a responsive and user-friendly web interface for monitoring and controlling IoT devices in real-time.
* **Flutter**: Flutter is employed to develop a mobile application, enabling cross-platform access and remote control of IoT devices.
* **NodeJS**: NodeJS serves as the backend server, handling communication between the interfaces and the ESP32 while processing data efficiently.
* **PostgreSQL**: PostgreSQL is chosen for storing and managing sensor data, user preferences, and logs, offering scalability and flexibility for handling dynamic IoT data.

**Objectives:**

* **Dual Mode Control:** Develop a platform that enables both automated and manual control of electrical appliances via relay switching.
* **Sensor Integration:** Utilize temperature, humidity, and distance sensors to monitor environmental conditions and trigger automated actions.
* **Real-Time Monitoring:** Establish wireless connectivity (via WiFi) for real-time data transmission to a cloud server, enabling remote monitoring.
* **User Interface:** Provide a user-friendly front end (web/mobile) for manual override and system status visualization.
* **Cost-Effectiveness and Scalability:** Implement the system with affordable components (ESP32, common sensors, and relays) while ensuring room for future expansion.

**Key Features of the Product:**

**Automation and Remote Control**:  
The platform allows users to manage appliances like fans, lights, motors, pumps, and air conditioning systems via web or mobile interfaces, reducing human effort and ensuring precise control.

**Energy Efficiency**:  
The system optimizes energy consumption by enabling smart scheduling and usage tracking for appliances, contributing to cost savings and sustainability.

**Scalability**:  
The platform is designed to accommodate future expansions, allowing users to integrate additional devices and features as needed.

**Ease of Use**:  
Intuitive user interfaces ensure a seamless experience, making it accessible for users of all technical levels.

**Security Features**:  
Includes secure data transmission and access control, ensuring user privacy and system safety.

**Possible Applications:**

**Residential Use**: Automate household appliances like lighting, HVAC systems, and water pumps.

**Industrial Use**: Monitor and control machinery, reduce downtime, and optimize resource usage.

**Smart Energy Management**: Real-time energy monitoring and optimization to reduce waste.

**Literature Survey:**

The development of IoT-based automation systems has been a topic of extensive research. Several studies have explored various architectures and frameworks for smart home and industrial automation:

* **IoT Frameworks and Protocols:**  
  Al-Fuqaha et al. (2015) provide an extensive survey on enabling technologies, protocols, and applications in IoT. Their work highlights the importance of scalable, secure, and efficient communication in IoT systems.
* **Smart Home Automation:**  
  Atzori, Iera, and Morabito (2010) discussed the potential of IoT in home automation, emphasizing how sensor networks and remote-control systems can improve household efficiency and energy usage.
* **Cost-Effective IoT Platforms:**  
  Zanella et al. (2014) analysed the applications of IoT in smart cities and homes, advocating for the use of affordable microcontrollers like the ESP32 to enable scalable and energy-efficient solutions.

**Comparison with Existing Idea and Proposed Idea:**

**Existing IoT Approaches:**

* **Single Mode Operation:** Many existing IoT platforms focus solely on automated control based on sensor thresholds or entirely on manual control via user interfaces.
* **Limited Integration:** Conventional systems often lack the flexibility to switch seamlessly between manual and automated modes, thereby limiting user control and responsiveness in dynamic environments.
* **Higher Cost and Complexity:** Some platforms use complex hardware and expensive sensors, making them less accessible for cost-sensitive applications.

**Proposed Idea:**

* **Dual-Mode Operation:** Our platform uniquely combines both automated and manual control. In automated mode, sensor readings (temperature, humidity, distance) trigger relay actions; in manual mode, users can override these actions via a dedicated front end.
* **Simplified Hardware Architecture:** By using the ESP32 and readily available sensors and relays, the system remains cost-effective while maintaining high functionality.
* **Enhanced Flexibility and User Experience:** The integration of real-time cloud connectivity and a responsive user interface (developed using frameworks like ReactJS and Flutter) provides an improved user experience over traditional, rigid systems.

**Proposed System Model:**

The proposed system is structured into the following key modules:

1. **Sensor Module:**
   * **Sensors:** Temperature (DHT11), Humidity (DHT11), and Distance (HC-SR04).
   * **Data Acquisition:** Continuous reading and validation of sensor data.
2. **Control Module (ESP32):**
   * **Processing Unit:** ESP32 microcontroller that processes sensor data.
   * **Relay Control:** Activates relays based on sensor thresholds (e.g., turning on a relay if temperature exceeds 20°C, if humidity falls below 40%, or if an object is within 15 cm).
3. **Connectivity Module:**
   * **WiFi Interface:** Enables connection to the internet and a cloud server.
   * **Communication Protocol:** Uses HTTP requests to send sensor data and receive manual control commands.
4. **User Interface:**
   * **Manual Control:** Provides a web/mobile interface for users to override automated actions.
   * **Real-Time Monitoring:** Displays sensor readings and system status.
5. **Cloud Backend:**
   * **Data Storage & Processing:** Receives sensor data and commands from the ESP32, facilitating centralized monitoring and control.

**Design Diagram: Hardware & Software:**

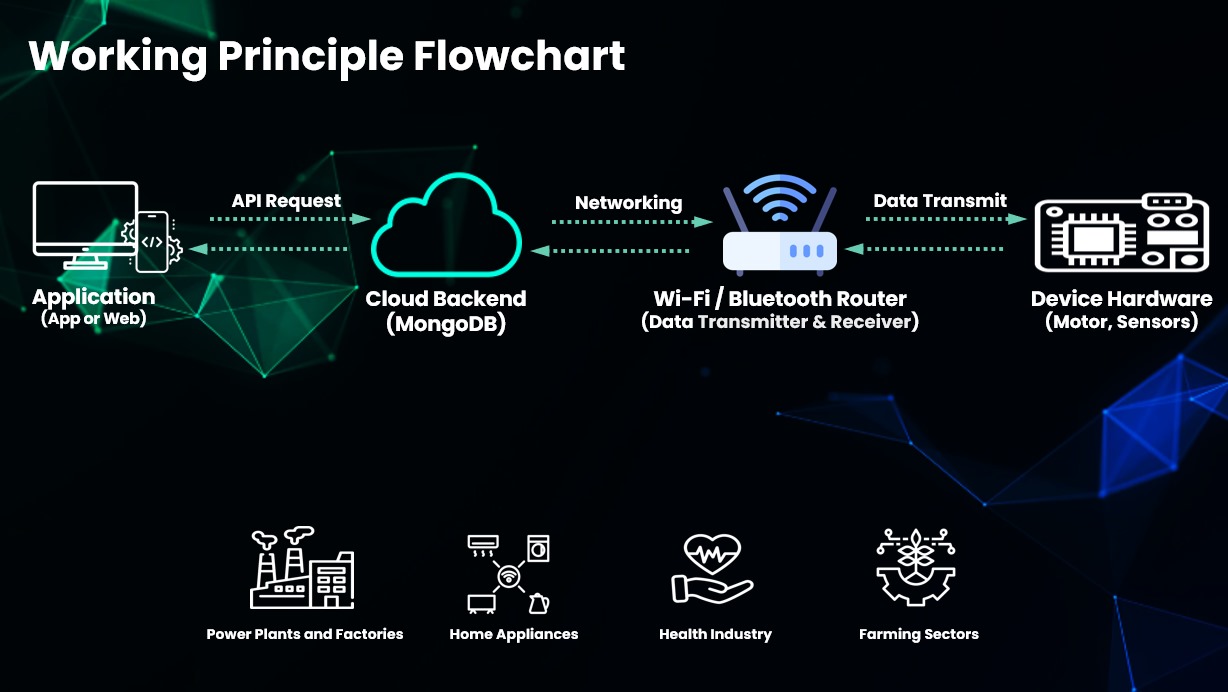
**Hardware Diagram:**

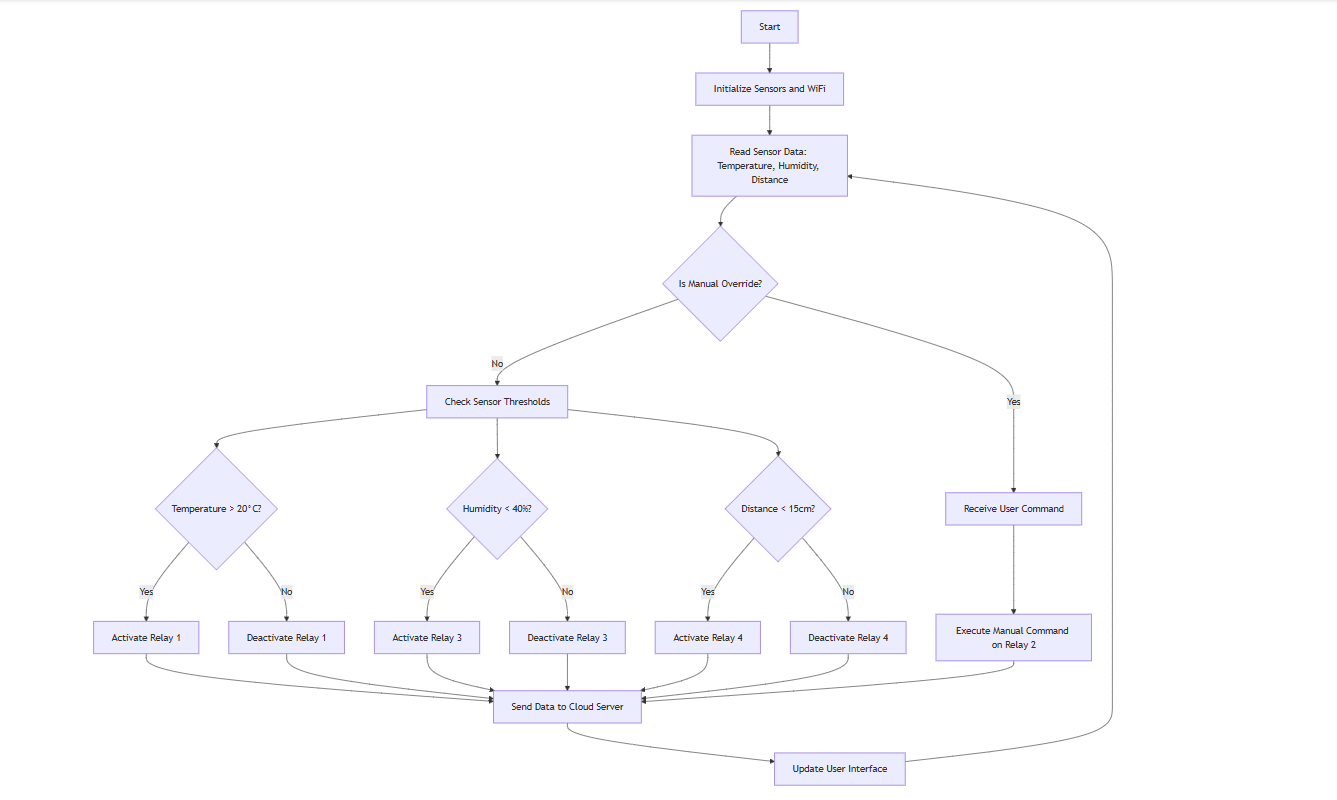
* **ESP32 Microcontroller:** Central hub for processing and communication.
* **Sensors:**
  + DHT11 for temperature and humidity.
  + HC-SR04 for distance measurement.
  + (Optional) PIR sensor for motion detection.
* **Actuators:**
  + Relays (RELAY\_1 to RELAY\_4) for controlling connected appliances.
* **Connectivity:**
  + WiFi module integrated within the ESP32 for cloud communication.

**Software Diagram:**

* **Firmware (Arduino IDE):**
  + Sensor data acquisition and relay control logic.
  + Implementation of dual-mode (automated and manual) control.
* **Cloud Backend:**
  + RESTful API endpoints (e.g., /dht, /pir, /ultrasonic) for receiving data.
* **User Interface:**
  + Web application (ReactJS) and mobile application (Flutter) for real-time control and monitoring.

**Flow Chart:**

****

****

**System Workflow: From User to IoT Device and Back:**

The Smart IoT Platform enables seamless interaction between users, applications, and IoT devices, creating an efficient and responsive ecosystem. The entire process involves user input, backend processing, hardware interaction, and real-time feedback to ensure smooth operation.

**User Interaction with Application:**

* Users interact with the system through a mobile or web application, designed to be user-friendly and intuitive.
* The application enables users to send commands (e.g., turning a light on or off) or retrieve real-time data (e.g., temperature readings).

**API Request to Cloud Backend:**

* The application communicates with a cloud backend (PostgreSQL) via secure APIs.
* These APIs handle requests to store, process, or fetch data as required by the user commands or system needs.
* Example: If the user requests the temperature of a room, the API fetches the data from the cloud database.

**Networking via Wi-Fi/Bluetooth:**

* The cloud backend transfers data to the IoT hardware using communication protocols like Wi-Fi or Bluetooth.
* This step bridges the gap between the digital application and the physical devices, ensuring reliable data transmission.

**Hardware Device Processing:**

* IoT devices like the ESP32 are equipped with sensors and actuators to execute commands or collect data.
* Upon receiving a command (e.g., "turn on the motor"), the hardware processes it and performs the required action.
* Sensors continuously collect data, which is transmitted back to the backend for further use.

**Feedback Loop to Application:**

* The IoT hardware sends feedback to the cloud, such as the status of an appliance (e.g., "Motor is ON") or sensor readings (e.g., "Temperature: 25°C").
* This feedback is displayed in the application in real-time, enabling users to monitor and verify actions or conditions.

**Algorithm:**

**Pseudocode:**

BEGIN

Initialize sensors (DHT11, HC-SR04, PIR)

Initialize ESP32 WiFi and connect to network

Set relay control pins as OUTPUT

LOOP:

Read temperature from DHT sensor

Read humidity from DHT sensor

Read distance using HC-SR04

IF manual override command received THEN

Process manual command for Relay 2 (and others if applicable)

ELSE

IF temperature > 20°C THEN

Set Relay 1 = ON

ELSE

Set Relay 1 = OFF

ENDIF

IF humidity < 40% THEN

Set Relay 3 = ON

ELSE

Set Relay 3 = OFF

ENDIF

IF distance < 15 cm AND distance valid THEN

Set Relay 4 = ON

ELSE

Set Relay 4 = OFF

ENDIF

ENDIF

Display sensor data on Serial Monitor

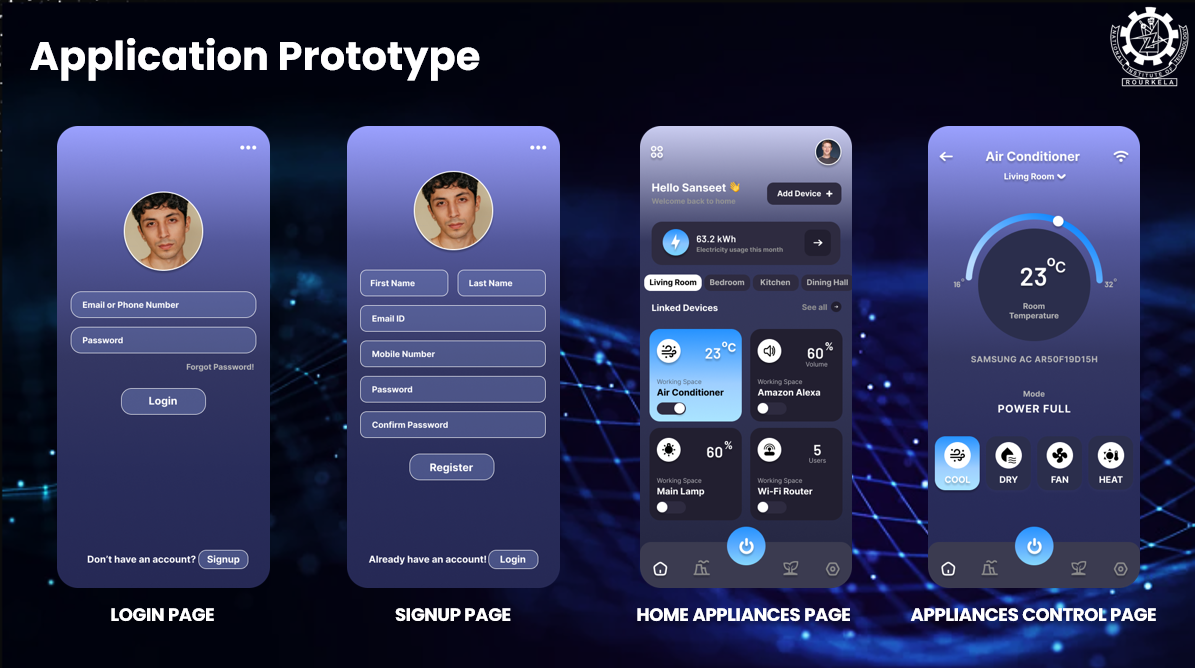
Transmit sensor data to cloud server via HTTP POST

Wait for defined sensor update interval

END LOOP

END

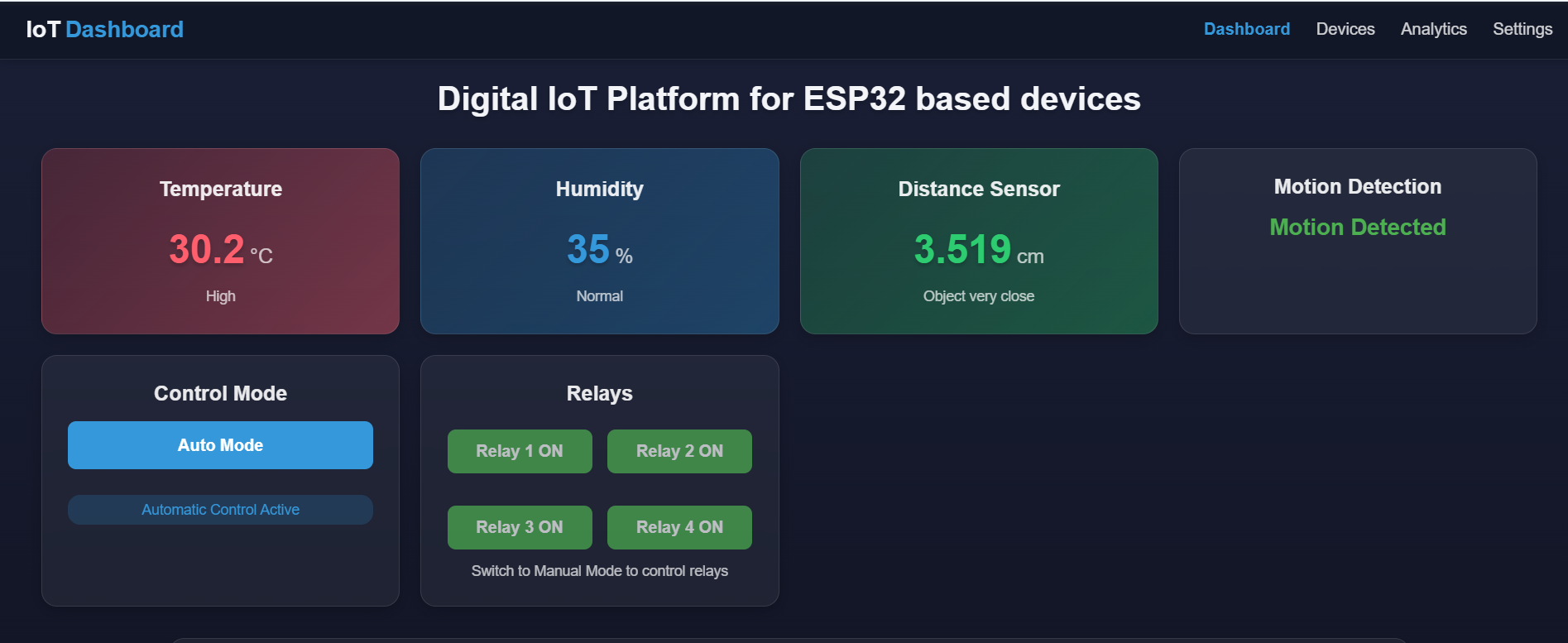
**App Prototype:**

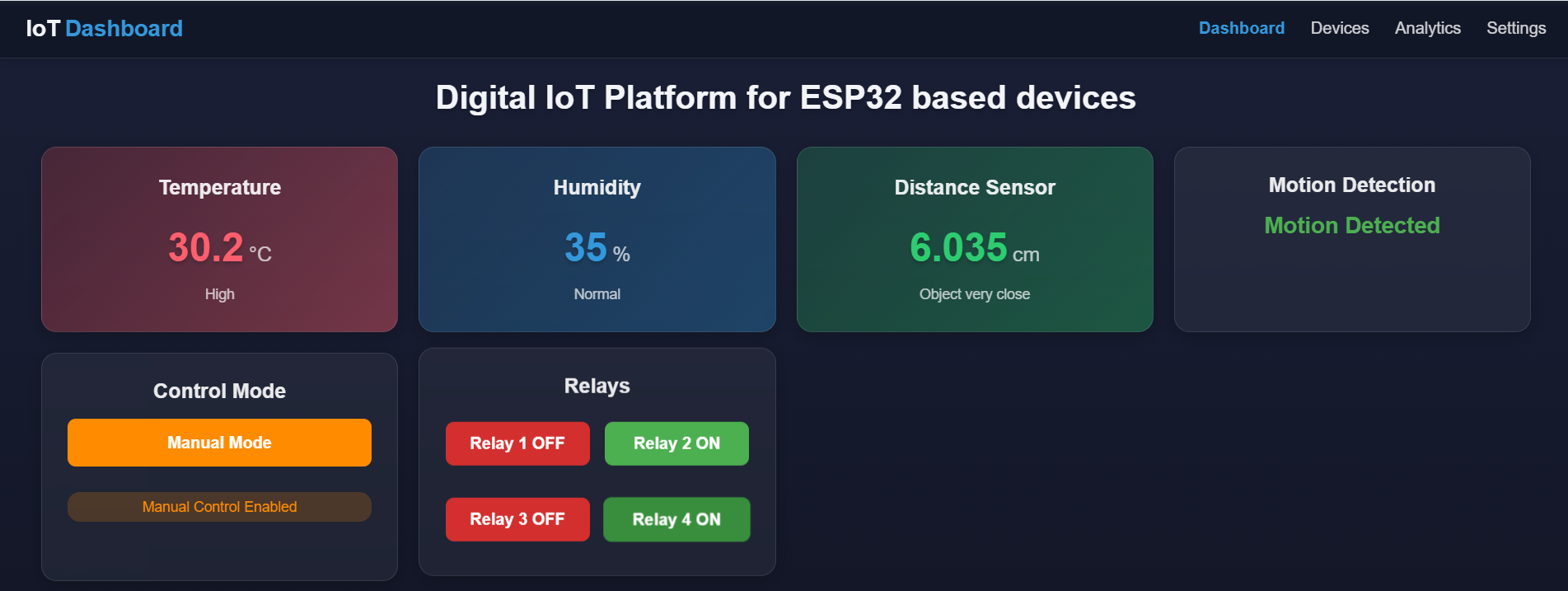


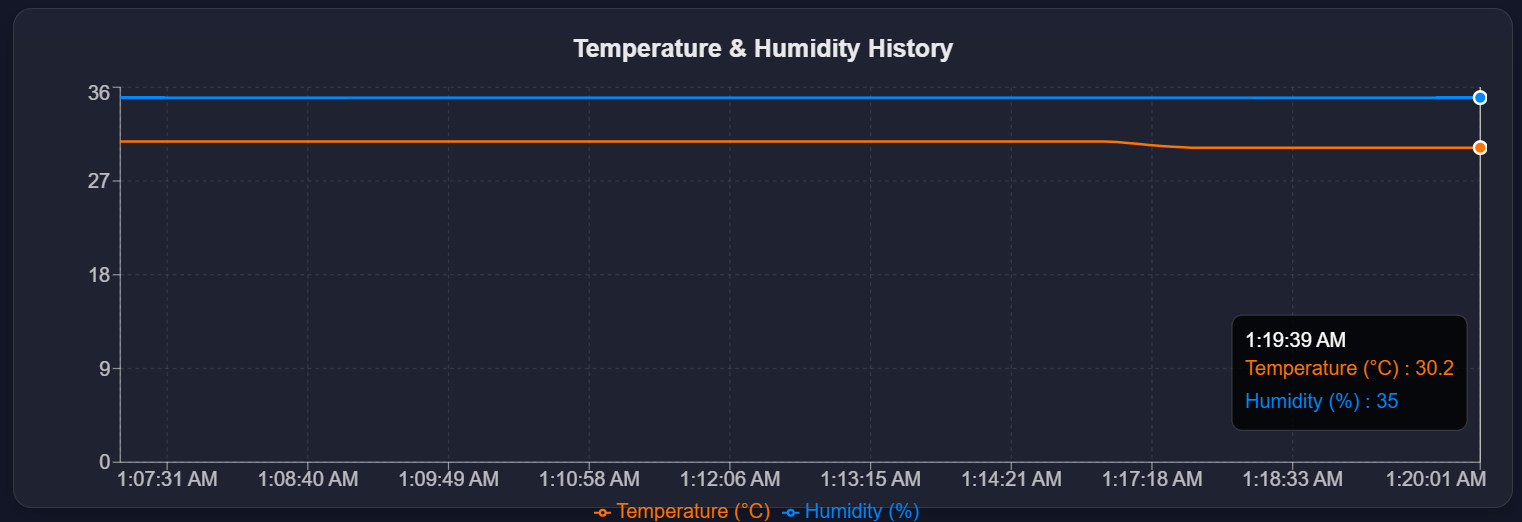


**Results:**

* **Sensor Readings:**
  + Temperature consistently recorded around 24°C.
  + Humidity levels were observed near 76%.
  + Distance measurements correctly flagged objects closer than 15 cm.
* **Relay Activation:**
  + Relay 1 activates when the temperature exceeds 20°C.
  + Relay 3 is triggered when the humidity falls below 40%.
  + Relay 4 activates if an object is detected within 15 cm.
  + Relay 2 remains under manual control via the front end.
* **Communication:**
  + The ESP32 successfully establishes a WiFi connection and sends periodic sensor updates to the cloud server.
  + Data received on the server is available in real time for the user interface, allowing for effective monitoring and control.
* **User Interface:**
  + The front end enables manual override while also displaying live sensor data and relay statuses.





****

**Project Repositories and Application Links:**

* **Web Page:** <https://iot-frontend-tau.vercel.app/dashboard>
* **Web and ESP32 Repository:** <https://github.com/shyamsundertard/iot_platform>
* **App Repository:** <https://github.com/Sanseet/digitalot>
* **App Download Link:** [**https://drive.google.com/file/d/1xPt376kzJAZbqnQ8byHs1rzEPWlfzYwl/view?usp=sharing**](https://drive.google.com/file/d/1xPt376kzJAZbqnQ8byHs1rzEPWlfzYwl/view?usp=sharing)

**Discussion:**

The implemented IoT platform demonstrates a practical approach to integrating automated sensor-driven control with manual intervention capabilities. The dual-mode functionality ensures that users have the flexibility to override automated settings when necessary. Key discussion points include:

* **Reliability:**  
  The ESP32 consistently handles sensor data and relay operations with minimal latency. However, occasional sensor reading inaccuracies (e.g., due to environmental interference) could be improved with calibration and error-checking enhancements.
* **Scalability:**  
  The system’s modular design allows for the integration of additional sensors or actuators, making it suitable for both residential and small industrial applications.
* **User Experience:**  
  With real-time data streaming and an intuitive control interface, the platform delivers a responsive and user-friendly experience. Future improvements could include enhanced security protocols and more sophisticated data analytics for predictive maintenance.
* **Limitations and Future Work:**  
  While the current implementation provides a robust proof of concept, future iterations could explore:
  + Integration with more advanced sensor types for broader environmental monitoring.
  + Implementation of machine learning algorithms for predictive control.
  + Expansion of the user interface to include historical data analysis and remote diagnostics.

**References:**

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications.* IEEE Communications Surveys & Tutorials.
2. Atzori, L., Iera, A., & Morabito, G. (2010). *The Internet of Things: A survey.* Computer Networks.
3. Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). *Internet of Things for Smart Cities.* IEEE Internet of Things Journal.
4. Espressif Systems. (2020). *ESP32 Technical Reference Manual.*