VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

MULTIDISCIPLINARY PROJECT (CO3107)

Assignment

Yolo:Bit-Powered Smart Home Security and Automation System

Advisor: Mai Xuan Toan
Student: Pham Le Huu Hiep - 2252223
Nguyen Ba Vuong - 2252922
Nguyen Dang Duy - 2252116
Ha Kien Hoa - 2252225
Nguyen Tien Hung - 2252280

HO CHI MINH CITY, MAY 2025

Contents

1 Member list & Workload **2**

2 Introduction **3**
2.1 Motivation . . . 3
2.2 Domain Context . . . 3

3 Requirements **4**
3.1 Functional Requirements . . . 4
3.2 Non-functional Requirements . . . 4

4 System Design **5**
4.1 Yolo:bit . . . 5
4.1.1 Module 1 - Sensor Readings . . . 7
4.1.2 Module 2 - Device Controls . . . 7
4.1.3 Module 3 - IoT Smart Home . . . 9
4.1.4 Module 4 - Voice Control . . . 11
4.1.5 Module 5 - FaceAI Recognition . . . 13
4.2 Web application module . . . 14
4.2.1 View sensors . . . 14
4.2.2 Turn on/off the light . . . 15
4.2.3 Change fan speed . . . 15
4.2.4 Turn on/off the pump . . . 16
4.2.5 Open/close the door . . . 16

5 System Development and Implementation **17**
5.1 Yolo:Bit module . . . 17
5.1.1 Submodule 1: Sensors reading . . . 17
5.1.2 Submodule 2: Devices Control . . . 17
5.1.3 Submodule 3: IoT Dashboard . . . 19
5.1.4 Submodule 4: Voice control . . . 20
5.1.5 Submodule 5: Face recognition . . . 21
5.2 Web application module . . . 21
5.2.1 Frontend . . . 21
5.2.2 Backend . . . 22
5.2.3 AI model for fire detection . . . 25
5.2.4 AI model for face recognition . . . 28

6 Source Code and Deployment **32**
6.1 Yolo:Bit module . . . 32
6.2 Web application module . . . 32
6.3 AI model for fire detection . . . 32
6.4 AI model for face recognition . . . 32
6.5 Demonstration . . . 32

7 Conclusion **33**

1 Member list & Workload

No.	Fullname	Student ID	Problems	Percentage of work
1	Pham Le Huu Hiep	2252223	- Learn and implement AI model. - Developing web application. - Write the report.	20%
2	Nguyen Ba Vuong	2252922	- Learn and implement AI model. - Developing web. - Write the report.	20%
3	Nguyen Dang Duy	2252116	- Learn and implement AI model. - Developing web. - Write the report.	20%
4	Ha Kien Hoa	2252225	- Learn and implement AI model. - Developing web. - Write the report.	20%
5	Nguyen Tien Hung	2252280	- Learn and implement AI model. - Developing web. - Write the report.	20%

2 Introduction

2.1 Motivation

In recent years, the demand for smart home solutions has surged, driven by growing needs for enhanced security, convenience, and energy efficiency. Traditional home security systems, while functional, often lack flexibility, real-time responsiveness, and cost-effectiveness. To address these limitations, we introduce the **Yolo:Bit-Powered Smart Home Security and Automation System**—a comprehensive AIoT platform designed for modern smart homes. This system integrates key functionalities such as environmental monitoring, cipher-locked doors, voice control, and facial recognition. Additionally, we aim to enhance these features by enabling automated door unlocking through facial recognition and developing a web application for centralized control and management of the entire smart home system.

2.2 Domain Context

This project belongs to the field of AIoT (Artificial Intelligence of Things), which combines smart devices with intelligent features to make everyday life more convenient and secure. In smart homes, AIoT allows different devices—like sensors, lights, cameras, and door locks—to work together and respond automatically based on the situation. With the help of technologies like voice control and facial recognition, the system can understand commands and make decisions without needing constant human control. By using Yolo:Bit, a user-friendly microcontroller designed for learning and creativity, this project shows how we can build a smart home system that is both useful and easy to develop. It helps demonstrate how technology can make homes safer, more efficient, and more interactive.

3 Requirements

3.1 Functional Requirements

- **Environmental Monitoring:**
  - The system shall displays real-time data on temperature, humidity, and light intensity.
  - The system shall provides a line graph showing the temperature history over time.
- **Fire Detection:**
  - The system shall detect the presence of fire or abnormal heat levels.
  - The system shall trigger an alert (e.g., sound an alarm or notify the user) when a fire is detected.
- **Door Access Control:** The user shall able to turn the door on and off through a dashboard interface.
- **Lightning Control:** The user shall be able to turn the lights on and off through a dashboard interface.
- **Fan Control:** The user shall be able to adjust the fan speed or turn off the fan through the dashboard.
- **Voice Control**: The system shall accept voice commands to control home devices (e.g., lights, fan, door).
- **Pump Control:** The user shall able to turn the pump on and off through a dashboard interface.
- **Face Recognition Access:**
  - The system shall authenticate users through facial recognition.
  - Upon successful recognition, the system shall automatically unlock the door.

3.2 Non-functional Requirements

- **Usability:** The user interface (UI) should be intuitive, user-friendly, and easy to navigate for all users.
- **Performance:** The system should respond to user actions—such as turning lights on/off, changing light color, or adjusting fan speed—within 1 second under normal network conditions.
- **Availability:** The system should be accessible and operational 24/7, ensuring continuous service.
- **Reliability:** The system should maintain a failure rate of less than 5% per month to ensure stable and dependable operation.
- **Scalability:** The system should be scalable to support an increasing number of connected household devices without affecting performance.
- **Compatibility:** The web application should be compatible with major web browsers, including Google Chrome, Safari, Microsoft Edge, and others.

4 System Design

Our project YoloHome will be divided into two parts. The first part is an AIoT system in Yolo:Bit, the second part is a web application.

4.1 Yolo:bit

For an AIoT system with Yolo:Bit, we need several devices as below:

1. **Expansion board for Yolo:Bit**: An extension board for Yolo:Bit that enables connectivity with multiple devices.

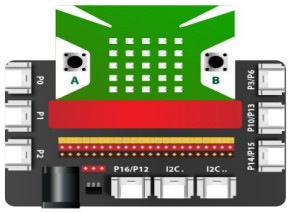


Figure 1: Expansion board for Yolo:Bit.

2. **4-LED RGB**: A module consisting of four LEDs capable of displaying eight different colors: red, orange, yellow, green, blue, indigo, violet, and white.

Figure 2: 4-LED RGB.

3. **DHT20**: A sensor designed to measures temperature and humidity in the surrounding air.

Figure 3: DHT20.

4. **LCD**: A display screen used to show various data, including date, time, temperature, humidity, and light levels.

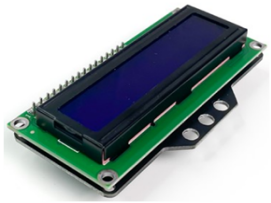


Figure 4: LCD.

5. **Mini fan**: A small fan used to demonstrate the operation of a fan motor.

Figure 5: Mini fan.

6. **Light sensor**: A component that detects light intensity and converts it into a digital value ranging from 0 to 4095 using an ADC (Analog to Digital Converter).

Figure 7: Light sensor.

7. **Mini pump**: A small pump used to illustrate the activity of pump is working or not.

Figure 8: Mini pump.

8. **RC Servo motor**: A motor that operates based on angular position control, used in this project to simulate door opening and closing.

Figure 10: RC Servo motor.

4.1.1 Module 1 - Sensor Readings

This submodule is a core feature of the YOLO project, enabling real-time monitoring of indoor conditions. It uses the DHT20 sensor to measure temperature and humidity, and a light sensor to detect ambient light. The data, along with the current date and time, is shown on the LCD screen.

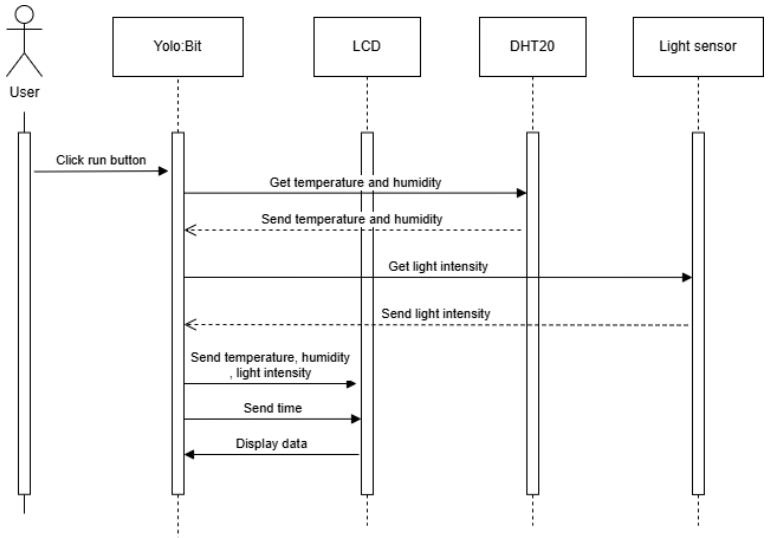


Figure 11: Sequence diagram of sensors.

When a user initiates the program via the Ohstem application, the Yolo sends a request to the DHT20 sensor to obtain temperature and humidity readings. After the DHT20 measures and returns these values, the Yolo requests light intensity data from the Light sensor. Upon receiving all three environmental parameters, the Yolo transmits this information along with the current time to the LCD display, which then presents the complete dataset to the user.

4.1.2 Module 2 - Device Controls

There are four basic device we need to clarify before application. In this path, the order of the whole system is depend on the user, no logically sequence.

- **Door:** When the user click "on" button, the yolobit automatically send open signal to RC servo motor and the RC servo motor open the door. The activity is simiar when the user click "off" button.

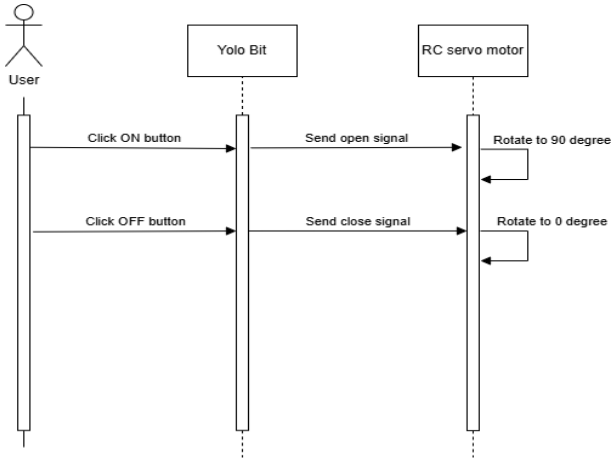


Figure 12: Sequence diagram of door.

- **Fan:** There are 4 buttons that the user can choose. The "1", "2", "3" is the level of the fan. Whenever user click on that three buttons, the yolobit automatically send turn on signal to RC servo motor and turn the fan at "1", "2", "3" level depend on user. Otherwise, the RC servo motor turn off the fan.

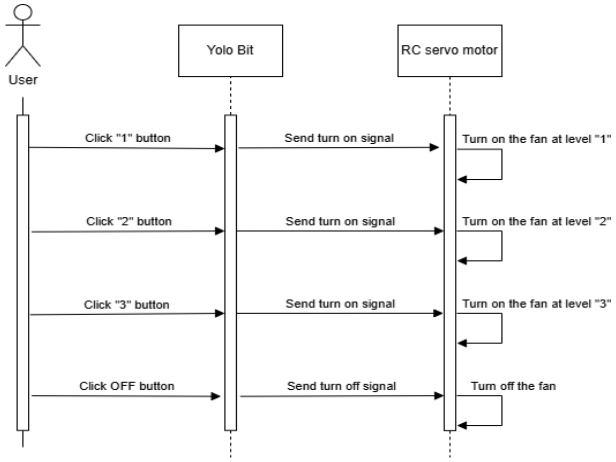


Figure 13: Sequence diagram of fan.

- **LED:** When the user click "on" button, the yolobit automatically send turn on signal to RC servo motor and the RC servo motor turn on the LED. The activity is similar when the user click "off" button.

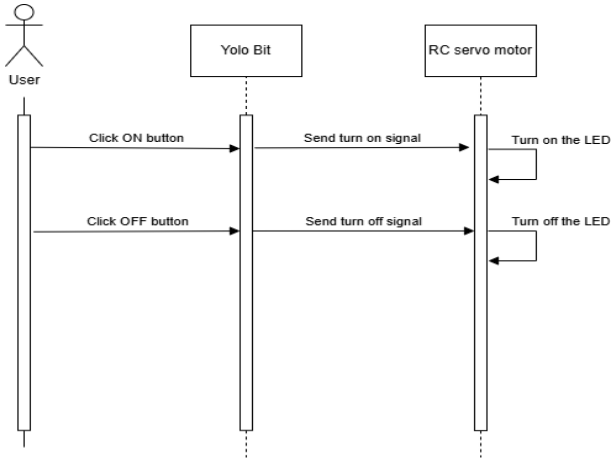


Figure 14: Sequence diagram of LED.

- **Pump:** When the user click "on" button, the yolobit automatically send turn on signal to RC servo motor and the RC servo motor turn on the pump. The activity is similar when the user click "off" button.

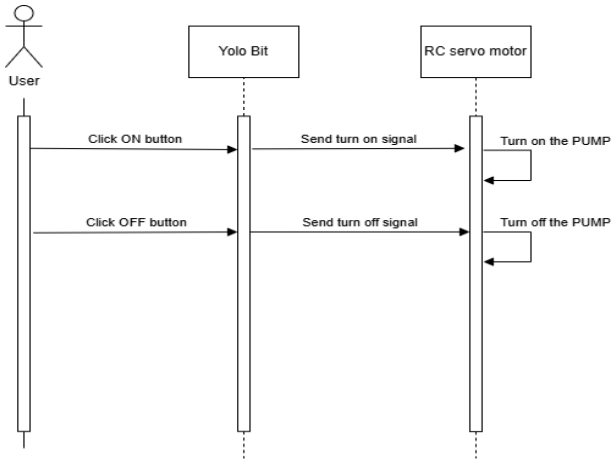


Figure 15: Sequence diagram of Pump.

4.1.3 Module 3 - IoT Smart Home

With the application of IoT, the data from sensors of the smart system can be sent to the Internet to facilitate the management from a far distance. This submodule is an IoT dashboard to monitor and control the devices in our smart home.
The devices can send and receive data from an IoT server based on a protocol called Message Queuing Telemetry Transport (MQTT). In this project, we will use Ohstem server to connect the devices together. In order to send data to Ohstem server, we will use publish mechanism of MQTT; and for receiving data, we will use subscribe mechanism.
Our IoT dashboard features eight functional components:

1. Three boxes to display temperature, humidity and light sensor.
2. Three line graphs to illustrate the history of temperature, humidity and light sensor.

3. A button to turn on/off the light.
4. A button to change the fan speed.

Each dashboard component interacts with the Ohstem server through designated data channels called topics. The diagram below illustrates the communication pathways between devices, the Ohstem server, and the IoT dashboard:

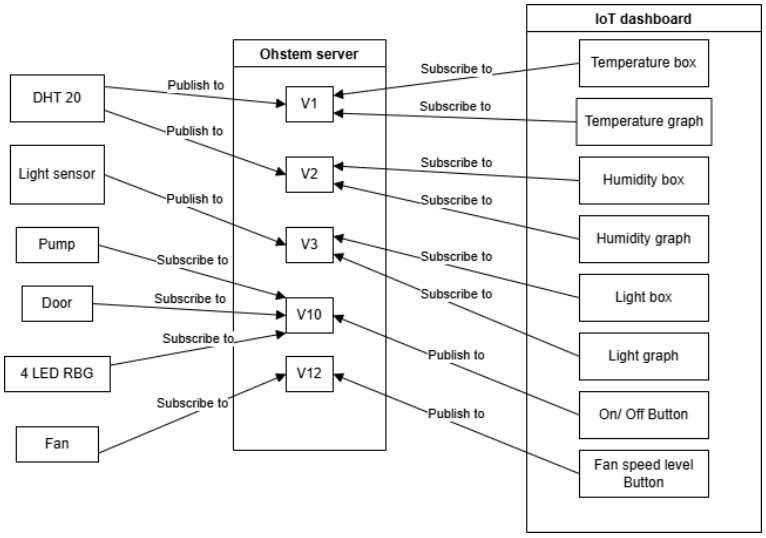


Figure 16: Connection through MQTT

Seven topics in Ohstem server are used in this submodule:

- **V1**: After measuring temperature, DHT20 will send data to V1 by publish mechanism. Temperature box in IoT dashboard will get the temperature by subscribe to V1 and display it on the screen. Temperature graph in IoT dashboard also subscribe to V1 to get the data so that it can draw the graph to illustrate temperature history.
- **V2**: After measuring humidity, DHT20 will send data to V1 by publish mechanism. Humidity box in IoT dashboard will get the humidity by subscribe to V1 and display it on the screen. Humidity graph in IoT dashboard also subscribe to V1 to get the data so that it can draw the graph to illustrate humidity history.
- **V3**: After measuring light intensity, light sensor will send data to V3 by publish mechanism. Light intensity box in IoT dashboard will get the light intensity by subscribe to V3 and display it on the screen. Light intensity graph in IoT dashboard also subscribe to V1 to get the data so that it can draw the graph to illustrate light intensity history.
- **V4**: 4 LED RGB will subscribe to V10 to get the signal whether to turn on or turn off the light. If the signal is 0, the light will be turn off. Otherwise, the light will be turn on. On/Off button in IoT dashboard sends signal to V10 by publish mechanism. If the button is in the state ”On”, signal 1 will be send to V10. Otherwise, signal 0 will be send to V10. Pump and Door are similar to 4 LED RBG.
- **V12**: Fan will subscribe to V12 to get the fan speed. Speed level in IoT dashboard will publish the fan speed to V12.

4.1.4 Module 4 - Voice Control

Instead of typing text commands or clicking buttons, user can control devices much more easily by voice command. We will use an AI model converting speech to text and handle the command from user by connecting to Ohstem server with MQTT.

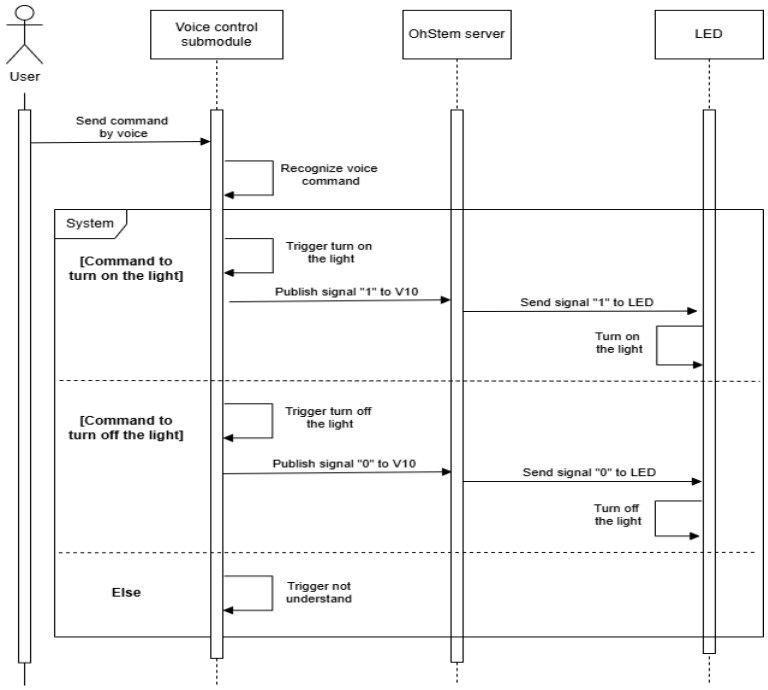


Figure 17: Sequence diagram of voice control LED.

In this submodule, the user will send their command by voice. Then, the AI model in voice control submodule will recognize the voice command and convert speech to text. There are 3 cases in this stage:

- If the command contains ”turn on the light”, the voice control submodule will trigger turn on the light function. The voice control submodule will publish signal ”1” to topic V10 on Ohstem server. Then, Ohstem server will send signal ”1” to 4 LED RGB. The 4 LED RGB will turn on the light.
- If the command contains ”turn off the light”, the voice control submodule will trigger turn off the light function. The voice control submodule will publish signal ”0” to topic V10 on Ohstem server. Then, Ohstem server will send signal ”0” to 4 LED RGB. The 4 LED RGB will turn off the light.

Similar, we have sequence diagram voice control for door, fan and pump.

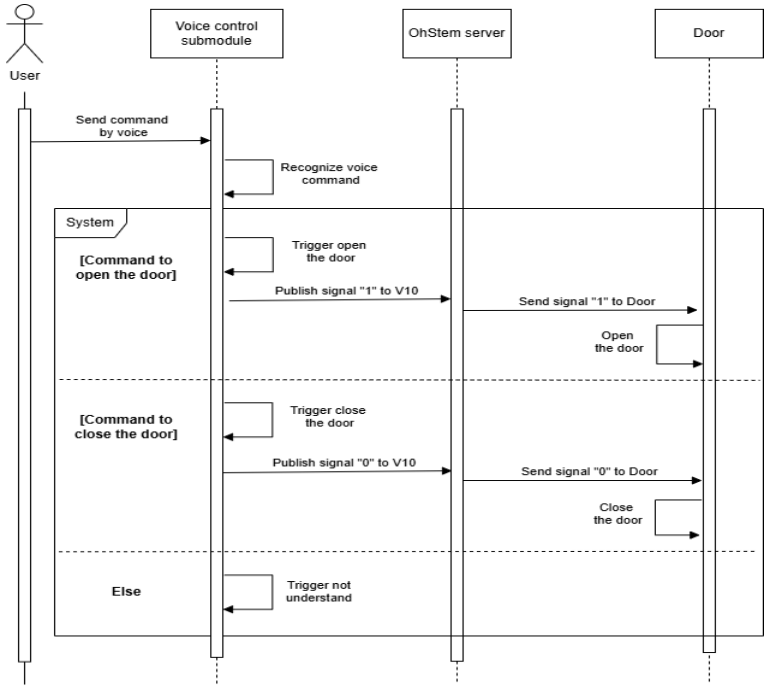


Figure 17.1: Sequence diagram of voice control Door.

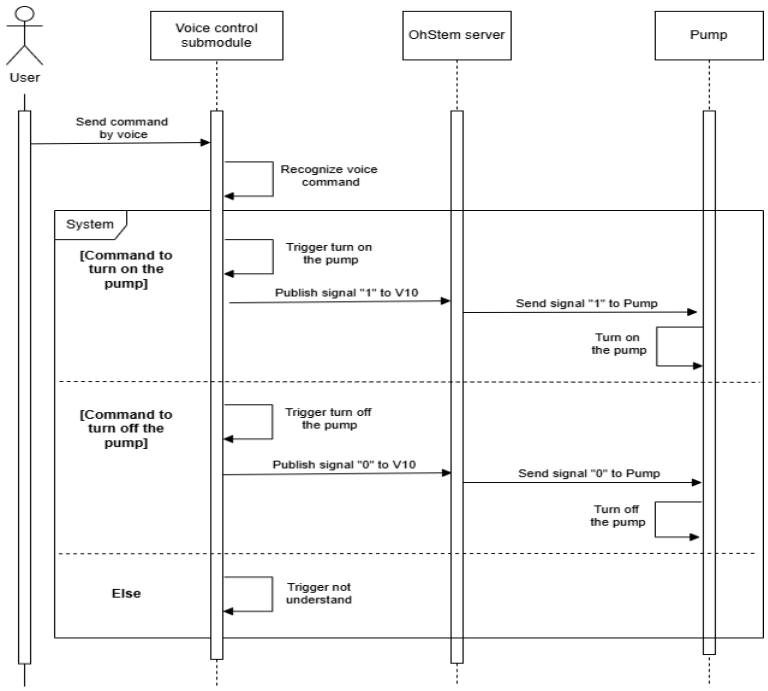


Figure 17.2: Sequence diagram of voice control pump.

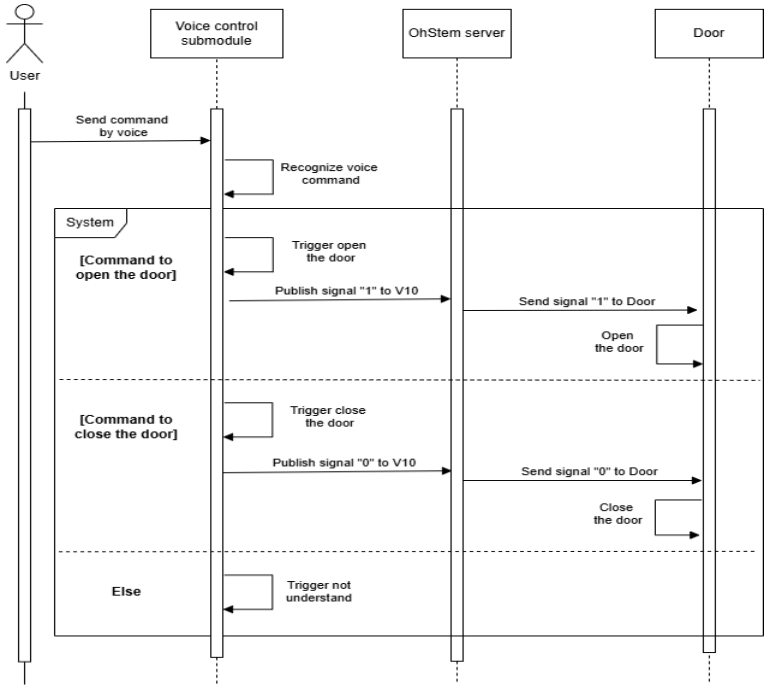


Figure 17.3: Sequence diagram of voice control fan.

4.1.5 Module 5 - FaceAI Recognition

Face recognition is a feature to authenticate users. We will use an AI model to regconize the faces of users who have access to the house, labelling the names of the users and display a greeting message in the Information box of IoT dashboard. If the user is authenticated sucessfully, the door will open.

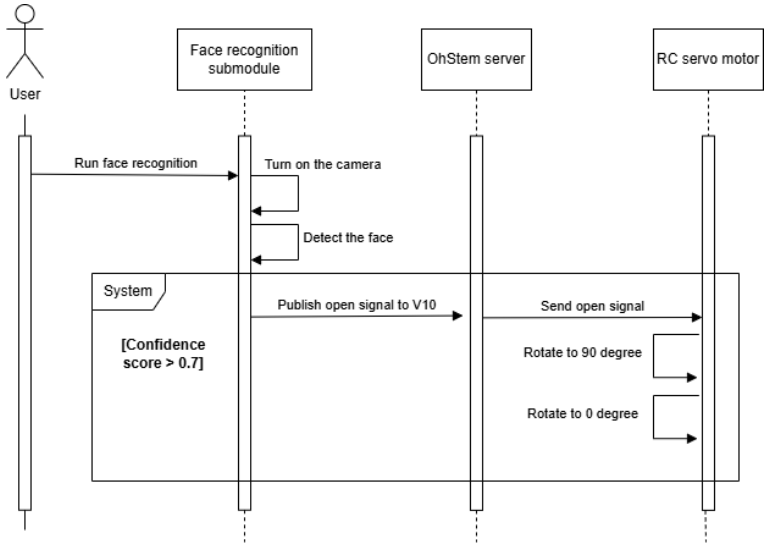


Figure 18: Sequence diagram of face recognition.

Firstly, the user will click the run button to trigger the face recognition submodule. Then, the

face recognition submodule will turn on the camera and begin to perform face detection. If the confidence score of the detection is larger than 0.7, the face recognition submodule will publish an open signal to V10 to make the door open. In this submodule, RC servo motor will subscribe to topic V10 to receive open or close signal from Ohstem server. Ohstem server will send that open signal to RC servo motor. The RC servo motor will rotate to 90 degree, indicating that the door is open. After the user has came in the house, the RC servo motor will rotate back to 0 degree, indicating that the door is closed.

4.2 Web application module

In this part, we decide to design a web application so that users can observe changes in their house and control the devices at home. The use case of the web application is shown as below:

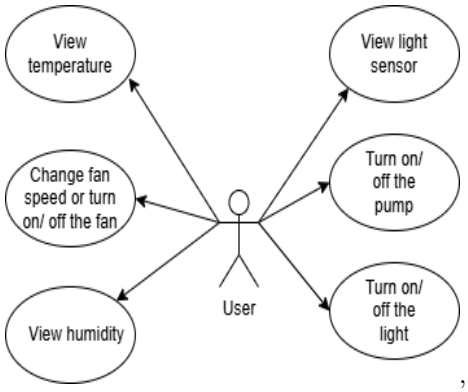


Figure 19: Use case diagram of the web application.

4.2.1 View sensors

	View sensors
Actor	User
Description	User can view environmental data including temperature, humidity, and light intensity.
Trigger	User has initiated the YOLO system.
Precondition	User has successfully authenticated to the web application and all sensors are operational.
Postcondition	User can visualize environmental data on the interface.
Normal flow	1. The web application requests environmental data from the Ohstem server. 2. DHT20 sensor measures temperature and humidity values and transmits them to the Ohstem server. 3. Light sensor measures ambient light intensity and transmits it to the Ohstem server. 4. Ohstem server forwards all environmental parameters to the web application backend. 5. The data is rendered on the user interface through the frontend components.
Alternative flow	None
Exception flow	None

4.2.2 Turn on/off the light

Use case name	Turn on/off the light
Actor	User
Description	User can activate or deactivate lighting in the smart home.
Trigger	User has initiated the YOLO system.
Precondition	User has successfully authenticated to the web application and all control systems are operational.
Postcondition	User can successfully control lighting status.
Normal flow	1. User activates the lighting by clicking the designated button. 2. The web application backend receives signal " 1 " from the frontend and forwards it to the Ohstem server. 3. Ohstem server receives signal " 1 " and transmits it to the 4-LED RGB module. 4. The 4-LED RGB module activates, illuminating the environment.
Alternative flow	1. User deactivates the lighting by clicking the designated button. 2. The web application backend receives signal " 0 " from the frontend and forwards it to the Ohstem server. 3. Ohstem server receives signal " 0 " and transmits it to the 4-LED RGB module. 4. The 4-LED RGB module deactivates, turning off the light.
Exception flow	None

4.2.3 Change fan speed

Use case name	Change fan speed
Actor	User
Description	User can adjust the operational speed of the fan.
Trigger	User has initiated the YOLO system.
Precondition	User has successfully authenticated to the web application and all control systems are operational.
Postcondition	User can successfully modify fan speed settings.
Normal flow	1. User manipulates the slider to select the desired fan speed. 2. The web application backend receives the speed value from the frontend and forwards it to the Ohstem server. 3. Ohstem server receives the speed value and transmits it to the fan device. 4. The fan adjusts its operational speed according to the user's selection.
Alternative flow	None
Exception flow	None

4.2.4 Turn on/off the pump

Use case name	Turn on/off the pump
Actor	User
Description	User can activate or deactivate the pump in the smart home.
Trigger	User has initiated the YOLO system.
Precondition	User has successfully authenticated to the web application and all control systems are operational.
Postcondition	User can successfully control pump.
Normal flow	1. User activates the lighting by clicking the designated button. 2. The web application backend receives signal " 1 " from the frontend and forwards it to the Ohstem server. 3. Ohstem server receives signal " 1 " and transmits it to the pump. 4. The pump activates, illuminating the environment.
Alternative flow	1. User deactivates the lighting by clicking the designated button. 2. The web application backend receives signal " 0 " from the frontend and forwards it to the Ohstem server. 3. Ohstem server receives signal " 0 " and transmits it to the pump. 4. The pump module deactivates, turning off the pump.
Exception flow	None

4.2.5 Open/close the door

Use case name	Open/Close the door
Actor	User
Description	User can open or close door in the smart home.
Trigger	User has initiated the YOLO system.
Precondition	User has successfully authenticated to the web application and all control systems are operational.
Postcondition	User can successfully control the door.
Normal flow	1. User activates the door by clicking the designated button. 2. The web application backend receives signal " 1 " from the frontend and forwards it to the Ohstem server. 3. Ohstem server receives signal " 1 " and transmits it to the door. 4. The door module activates, open the door.
Alternative flow	1. User deactivates the door clicking the designated button. 2. The web application backend receives signal " 0 " from the frontend and forwards it to the Ohstem server. 3. Ohstem server receives signal " 0 " and transmits it to the door. 4. The door module deactivates, close the door.
Exception flow	None

5 System Development and Implementation

5.1 Yolo:Bit module

5.1.1 Submodule 1: Sensors reading

We created three modules named Temperature, Humidity and Luminosity. DHT 20 will read data from three modules every 60 minutes and stored the data into Adafruit. After that, it displays the data to the LCD with the time (hour, minutes, second)

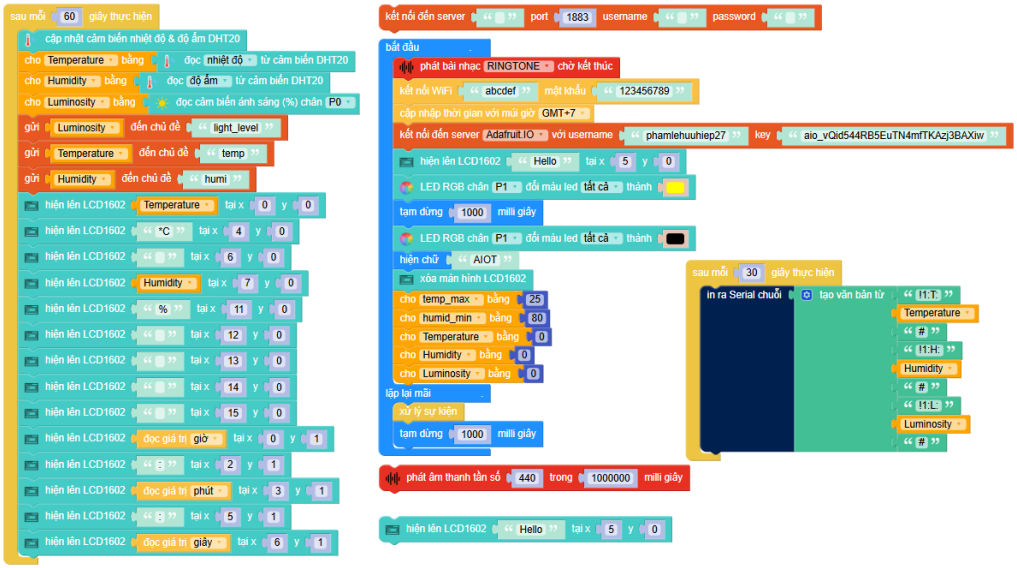


Figure 20: Yolo:Bit block code for Viewing sensors.

Notice that we set the maximum temperature is 25 and minimum humidity is 80.

5.1.2 Submodule 2: Devices Control

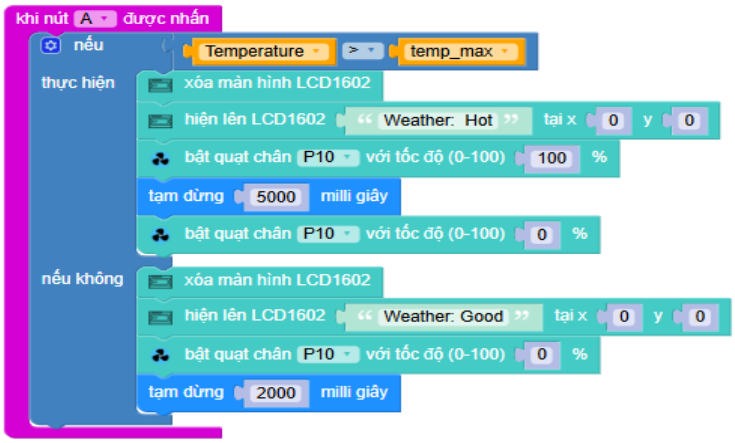
Door: The door received signal to open or close from the user through the web application.

Figure 21: Yolo:Bit block code for door.

Fan: The fan received signal to turn on/ off or change the speed from the user through the web application.

Figure 22: Yolo:Bit block code of fan.

When we press "A" button in the expansion board of Yolo:Bit



- If the current temperature is higher than the temperature max (temp $>$ 25). The fan is automically turn on and the LCD will display "Weather Hot".
- Otherwise, there nothing happen with the fan and the LCD will display "Weather Good".

LED: The LED received signal to turn on/off from the user through the web application.

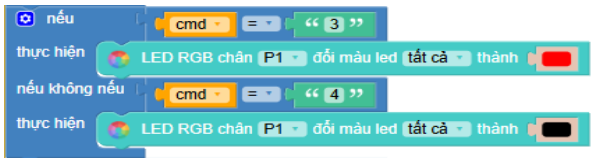
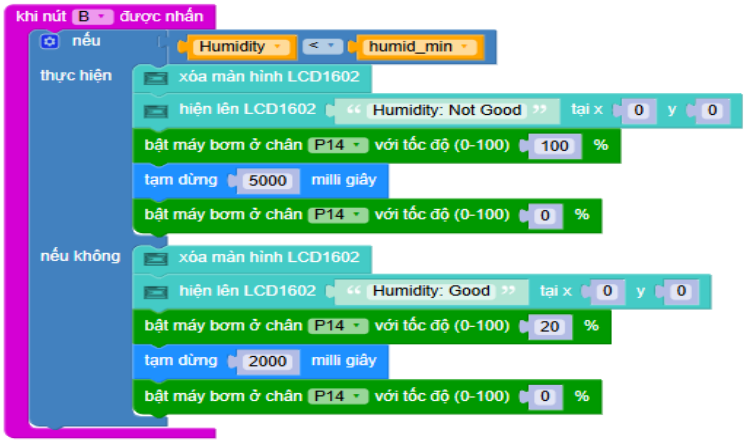


Figure 23: Yolo:Bit block code of LED.

Pump: The pump received signal to turn on/off from the user through the web application.

Figure 24: Yolo:Bit block code of pump

When we press "B" button in the expansion board of Yolo:Bit



- If the current humidity is lower than the humidity min (hud $<$ 80). The pump is automically turn on and the LCD will display "Humidity Not Good".
- Otherwise, there nothing happen with the pump and the LCD will display "Humidity Good".

5.1.3 Submodule 3: IoT Dashboard

The IoT Dashboard based on MQTT protocol will be implemented by the dashboard on Ohstem app. The dashboard after being implemented will look like this:
We will name this dashboard ”Yolo Home”, which is the name of this project. The username for this dashboard will be set as the same as when we subscribe or publish to Ohstem server, so that we can connect to Ohstem server to get and send data.

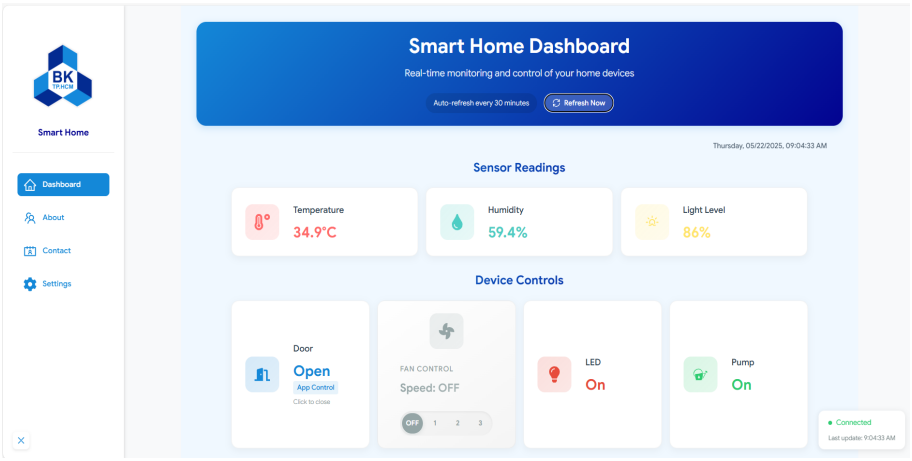
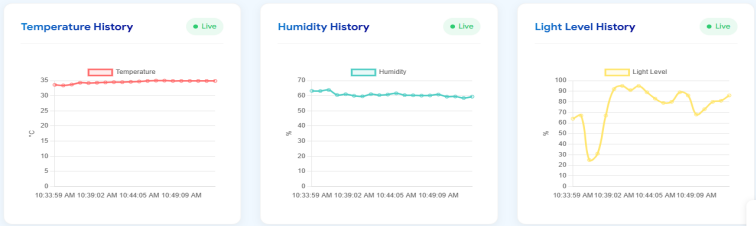


Figure 25: Web dashboard

For the features of the web:

- View the current temperature, humidity and light sensor through the sensors reading.

- Control door, fan, LED and pump through device controls.
- View the history of the temperature, humidity and light intensity.



5.1.4 Submodule 4: Voice control

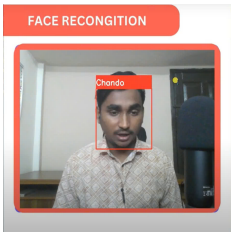
In this section, we will use AI programming on Ohstem app to implement a module to convert speed to text and execute the command.
Firstly, we will connect with the IoT dashboard through our username on Ohstem server. Then, we will begin the task of detecting voice commands. In this project, we will detect voice command in Vietnamese.

Figure 26: Yolo:Bit block code for Voice control

We can use the voice to control door, fan and LED.

5.1.5 Submodule 5: Face recognition

The Face recognition model provided by YOLOHOME software is basically an image classifier. The model is fully supervised, which means data from specific classes are provided to train the model. For example, when model needs to recognize five different people, face images of five people are provided to train the model and the model can only recognize a person who belongs to this group of five. In inference mode, the model outputs the confidence score of the recognized class.



If that confidence score is above a threshold, 0.7 in our case, the corresponding class name is given. Since this model is real time, it is necessary to ensure that model continuously recognize the right class before registering the result. We do this by using a counter to keep track of the number our model recognized correctly in a continuous manner. When a counter for a class is greater than 3, the result is reliable and it will be used to control the system. Apart from the the names of people needed for recognition, there is an empty class with named "Background" added to the model, this is to represent the case where there is no person in the image

5.2 Web application module

5.2.1 Frontend

For the Technology Stack, we design:

- Reactjs as the main frontend framework.
- MQTT for real-time communication.
- Chartjs with react-chartjs-2 for data visualization.
- CSS for styling.

For Main Application Structure, we decide to use Adafruit IO as the MQTT broker for IoT device communication.
The dashboard presents live sensor readings in color-coded cards—temperature in red with a thermometer icon (°), humidity in blue with a droplet icon (%), and light level in orange with a sun icon (%)—for instant at-a-glance monitoring. Intuitive controls include a four-speed fan slider with a spinning icon and a light toggle button that changes bulb color to reflect its On/Off state. Powered by Adafruit IO over WebSockets, all data and historical charts (last 20 points via Chart.js) update in real time, with configurable auto-refresh and clear connection status indicators. A collapsible sidebar provides quick navigation (Dashboard, About, Contact, Settings), where users can adjust refresh intervals and preferences, ensuring a responsive, modern UI across devices.

5.2.2 Backend

The backend of the smart home system is built around the Adafruit IO platform, serving as the core hub for IoT data communication and management. Configuration is handled via environment variables (VITE_ADAFRUIT_USERNAME, VITE_ADAFRUIT_KEY), with data exchanged through Adafruit IO's REST API and WebSocket interfaces. The system manages multiple feeds for temperature, humidity, light level, and device controls (door, fan, LED, pump), enabling real-time data updates and historical data retrieval.

```
.env
VITE_ADAFRUIT_USERNAME=phamlehuuhiep27
VITE_ADAFRUIT_KEY=aio_vQid544RB5EuTN4mfTKAzj3BAXiw
```

Real-time communication is achieved using secure WebSocket connections with automatic reconnection, feed subscriptions, and ping/pong keep-alive mechanisms. MQTT is also supported, using QoS 1 for reliable message delivery and robust connection handling. The backend handles real-time sensor data processing, timestamp management, secure command transmission, and state verification after device control actions.

```
// Adafruit IO Configuration
const ADAFRUIT_CONFIG = {
  username: import.meta.env.VITE_ADAFRUIT_USERNAME,
  key: import.meta.env.VITE_ADAFRUIT_KEY,
  aioUrl: "https://io.adafruit.com/api/v2"
};

// Main API fetch function
const fetchData = useCallback(async () => {
    try {
      const feedsResponse = await fetch
      (${ADAFRUIT_CONFIG.aioUrl}/${ADAFRUIT_CONFIG.u
      sername}/feeds‘,
        {
          headers: {
            'X-AIO-Key': ADAFRUIT_CONFIG.key,
          },
        }
      );
      if (!feedsResponse.ok) {
        throw new Error('Failed to fetch feeds');
      }
      const feeds = await feedsResponse.json();
      const newStates = { ...deviceStates };
      await Promise.all(
        feeds.map(async (feed) => {
          try {
            const dataResponse = await fetch
            (‘${ADAFRUIT_CONFIG.aioUrl}
            /${ADAFRUIT_CONFIG.username}/feeds/${fee
            d.key}/data/last‘,
              {
```

```
              headers: {
                'X-AIO-Key': ADAFRUIT_CONFIG.key,
              },
            }
          );
          if (dataResponse.ok) {
            const data = await dataResponse.json();
            newStates[feed.key] = parseFloat(data.value) || 0;
          }
        } catch (error) {
          console.error(`Error fetching data for ${feed.key}:`, error);
        }
      })
    );
    setDeviceStates(newStates);
    setLastUpdate(new Date());
    setIsConnected(true);
    await fetchHistoricalData();
  } catch (error) {
    console.error('Error fetching data:', error);
    setIsConnected(false);
  }
}, [deviceStates]);

//Device control API endpoints
const handleFanSpeed = async (speed) => {
  try {
    const response = await fetch(
    `${ADAFRUIT_CONFIG.aioUrl}/
    ${ADAFRUIT_CONFIG.username}/feeds/fan/data`,
      {
        method: 'POST',
        headers: {
          'X-AIO-Key': ADAFRUIT_CONFIG.key,
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ value: speed }),
      }
    );
```

Security is enforced through environment-based API key management, WebSocket and MQTT authentication, and input validation. Error handling includes auto-reconnects, logging, fallback mechanisms, and state checks. Performance is optimized through rate-limited data fetching, efficient message handling, and connection pooling.

```
// MQTT connection and message handling
const connectMQTT = useCallback(() => {
  if (clientRef.current) {
    clientRef.current.end();
  }
```

```
const mqttClient =
mqtt.connect('wss://io.adafruit.com', {
  username: 'phamlehuuhiep27',
  password: 'YOUR_AIO_KEY', // Replace with
  your Adafruit IO key
  keepalive: 30, // Reduced keepalive for
  faster response
  reconnectPeriod: 500, // Faster reconnection
  clean: true,
  clientId: `web_${Math.random().to
  String(16).substr(2, 8)}`, // Unique client ID
  protocolVersion: 4
});
mqttClient.on('connect', () => {
  console.log('Connected to Adafruit IO');
mqttClient.subscribe('phamlehuuhiep27/feeds/#',
{ qos: 1 });
});
mqttClient.on('message', (topic, message) => {
  const feedName = topic.split('/').pop();
  const value = message.toString();
  const timestamp = new Date().toISOString();
  // Batch updates for better performance
  if (updateTimeoutRef.current[feedName]) {
  clearTimeout(updateTimeoutRef.
  current[feedName]);
  }
  updateTimeoutRef.current[feedName] =
  setTimeout(() => {
    setFeedData(prevData => ({
      ...prevData,
      [feedName]: value
    }));
    setHistoricalData(prevData => ({
      ...prevData,
      [feedName]: [...(prevData[feedName] || []).slice(-30), { value, timestamp }]
    }));
    setLastUpdate(prev => ({
      ...prev,
      [feedName]: timestamp
    }));
  }, 50); // Small delay to batch updates
});
mqttClient.on('error', (error) => {
  console.error('MQTT Error:', error);
  // Attempt to reconnect immediately on error
  setTimeout(() => connectMQTT(), 500);
});
clientRef.current = mqttClient;
```

```
    return mqttClient;
  }, []);
```

5.2.3 AI model for fire detection

This project uses a custom-trained YOLO model to detect fire in real-time from webcam footage and integrates with Adafruit IO via MQTT for smart device control. It processes video at 640px resolution with 78% confidence, identifying multiple objects including fire. The system communicates with Adafruit IO to manage feeds for sensors (temperature, humidity, fire) and device controls (LED, fan, door, pump). UART communication is handled via a separate Python module (uart.py), sending commands based on MQTT messages. Although fire alert actions are currently commented out, the system is designed to trigger alerts and publish to the fire_detect feed when fire is detected.

Figure 27: Yolo:Bit code block for Fire Detect

In uart.py, it connects to a serial device (like an Arduino or ESP32), reads sensor data, and publishes it to MQTT topics. It automatically detects the correct serial port, opens a connection, and continuously reads incoming data in the format !ID:TYPE:VALUE. It processes this data to extract values for temperature (T), humidity (H), and light level (L), then publishes them to respective MQTT feeds. It also includes a function to send commands back to the device via serial. This setup is commonly used in IoT applications for real-time sensor monitoring and control.

```
import serial.tools.list_ports
def getPort():
    ports = serial.tools.list_ports.comports()
    N = len(ports)
    commPort = "None"
    for i in range(0, N):
        port = ports[i]
        strPort = str(port)
        if "SERIAL" in strPort:
            splitPort = strPort.split(" ")
            commPort = (splitPort[0])
    # return "COM1"
    return commPort
if getPort()!= "None":
    ser = serial.Serial( port=getPort(), baudrate=115200)
    print(ser)
mess = ""
```

```
def processData(client, data):
    data = data.replace("!", "")
    data = data.replace("#", "")
    splitData = data.split(":")
    print(splitData)
    if splitData[1] == "T":
        client.publish("temp", splitData[2])
    elif splitData[1] == "H":
        client.publish("humi", splitData[2])
    elif splitData[1] == "L":
        client.publish("light_level", splitData[2])
mess = ""
def readSerial(client):
    bytesToRead = ser.inWaiting()
    if (bytesToRead > 0):
        global mess
        mess = mess +
        ser.read(bytesToRead).decode("UTF-8")
        while ("#" in mess) and ("!" in mess):
            start = mess.find("!")
            end = mess.find("#")
            processData(client, mess[start:end + 1])
            if (end == len(mess)):
                mess = ""
            else:
                mess = mess[end+1:]
def writeData(data):
    ser.write((str(data)).encode())
```

In fire.py, it integrates a custom-trained YOLO object detection model with Adafruit IO for real-time fire detection and IoT control. It uses a webcam to continuously capture video frames and runs YOLO inference to detect objects. If fire is detected (currently commented out), it is set up to send a serial command ("11") and publish an alert to the fire_detect feed. The script connects to Adafruit IO via MQTT using the Adafruit_IO library, subscribes to multiple feeds (like temp, humi, door_button, fan, etc.), and sends commands to connected devices via UART based on the received messages. This enables both monitoring and controlling smart home components like LEDs, doors, fans, and pumps.

```
from ultralytics import YOLO
import cv2
model = YOLO('best.pt')
import sys
from Adafruit_IO import MQTTClient
import time
import random
from uart import *
#from simple_ai import *
timer = time.time()
AIO_FEED_ID = ["temp", "humi", "light_level",
"led_button", "door_button", "fan",
```

```
"fire_detect","pump","set-h", "set-t"]
AIO_USERNAME =  "phamlehuuhiep27"
AIO_KEY = "aio_vQid544RB5EuTN4mfTKAzj3BAXiw"
AIO_KEY = AIO_KEY.replace(AIO_KEY[:3], "aio")
def connected(client):
    print("Ket noi thanh cong ...")
    for topic in AIO_FEED_ID:
        client.subscribe(topic)
def subscribe(client , userdata , mid ,granted_qos):
    print("Subscribe thanh cong ...")
def disconnected(client):
    print("Ngat ket noi ...")
    sys.exit (1)
def message(client , feed_id , payload):
    print("Nhan du lieu: " + payload + " , feed
    id:", feed_id)
    global timer
    if time.time() - timer < 0.2:
        time.sleep(0.3 - (time.time() - timer))
    if feed_id == "door_button":
        if payload == "1":
            writeData("1")
        else:
            writeData("2")
    if feed_id == "led_button":
        if payload == "1":
            writeData("3")
        else:
            writeData("4")
    if feed_id == "fan":
        if payload == "1":
            writeData("5")
        elif payload == "2":
            writeData("6")
        elif payload == "3":
            writeData("7")
        else:
            writeData("8")
    if feed_id == "pump":
        if payload == "1":
            writeData("9")
        else:
            writeData("10")
    if feed_id == "set-t":
            writeData("00000"+payload)
    if feed_id == "set-h":
            writeData("0000000"+payload)
    timer = time.time()
client = MQTTClient(AIO_USERNAME , AIO_KEY)
```

```
client.on_connect = connected
client.on_disconnect = disconnected
client.on_message = message
client.on_subscribe = subscribe
client.connect()
client.loop_background()
cap = cv2.VideoCapture(0)
class_names = model.names
while True:
    print("waiting")
    time.sleep(1)
    ret, frame = cap.read()
    if not ret:
        break  # Nếu không lấy được frame thì dừng
    results = model.predict(source=frame,
    imgsz=640, conf=0.78,verbose = False)
    r = results[0]
    boxes = r.boxes
    if boxes is not None and boxes.cls.numel() > 0:
        for cls_id in boxes.cls:
            class_id = int(cls_id.item())
            class_name = class_names[class_id]
            # if class_name == "fire":
            #     print("Detecting the FIRE------
            -!!!!!!")
            #     writeData("11")
            #     client.publish("fire_detect",1)
    # Thoát nếu người dùng nhấn 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

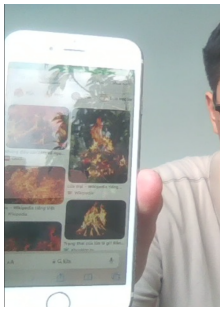


Figure 29: Test the AI model for fire detection.

5.2.4 AI model for face recognition

In Controllerpy, it controls a servo motor connected to an Arduino using the PyFirmata library. It sets up communication through port "COM7" and configures pin 10 for servo control. The rotateServo() function rotates the servo to a specified angle, while the doorAutomate() function

opens or closes a door based on the input value—rotating to 220 degrees to close and 40 degrees to open.

```
from pyfirmata import Arduino, SERVO
PORT="COM7"
pin=10
board=Arduino(PORT)
board.digital[pin].mode=SERVO
def rotateServo(pin, angle):
    board.digital[pin].write(angle)
def doorAutomate(val):
    if val==0:
        rotateServo(pin, 220)
    elif val==1:
        rotateServo(pin, 40)
```

In Datacollect.py, captures face images from a webcam using OpenCV and saves them to build a facial recognition dataset. It starts by loading the Haar Cascade classifier for face detection and prompting the user to enter their ID. The script then opens the webcam feed and continuously detects faces in grayscale frames. Each detected face is saved as a cropped image in the datasets folder with a filename format User.[ID].[count].jpg. A rectangle is also drawn around detected faces for visual feedback. The process continues until 500 face images are collected, after which the video stream is released and all OpenCV windows are closed. This dataset can later be used to train a face recognition model.

```
# pip install opencv-python==4.5.2
import cv2
video=cv2.VideoCapture(0)
facedetect = cv2.CascadeClassifier
("haarcascade_frontalface_default.xml")
id = input("Enter Your ID: ")
# id = int(id)
count=0
while True:
    ret,frame=video.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = facedetect.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        count=count+1        cv2.imwrite
        ('datasets/User.'+str(id)+"."+str(count)+".jpg",
        gray[y:y+h, x:x+w])
        cv2.rectangle(frame, (x,y), (x+w, y+h),
        (50,50,255), 1)
    cv2.imshow("Frame",frame)
    k=cv2.waitKey(1)
    if count>500:
        break
video.release()
cv2.destroyAllWindows()
print("Dataset Collection Done.................")
```

In Testmodel.py, it captures video from the webcam, detects faces using a Haar Cascade classifier, and performs real-time face recognition using the LBPH (Local Binary Patterns Histograms) model previously trained and saved in a file named Trainer.yml. It matches detected faces against known IDs, such as "Chando", from the name_list. Recognized faces are highlighted with rectangles and labeled with names if the confidence score exceeds a threshold (in this case, 50). The webcam feed is embedded into a custom background image (background.png) and displayed in a window. Although the code includes commented lines for triggering a door automation function upon successful recognition (e.g., opening the door when the person is identified), these parts are currently inactive. The program continues running until manually stopped.

```
# pip install opencv-python==4.5.2
import cv2
#from controller import doorAutomate
import time
video=cv2.VideoCapture(0)
facedetect =
cv2.CascadeClassifier("haarcascade_frontalface_defau
lt.xml")
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("Trainer.yml")
name_list = ["", "Chando"]
imgBackground = cv2.imread("background.png")
while True:
    ret,frame=video.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = facedetect.detectMultiScale(gray, 1.3,5)
    for (x,y,w,h) in faces:
        serial, conf = recognizer.predict
        (gray[y:y+h, x:x+w])
        if conf>50:
            cv2.rectangle(frame, (x,y), (x+w, y+h),
            (0,0,255), 1)
            cv2.rectangle(frame,(x,y),(x+w,y+h),
            (50,50,255),2)
            cv2.rectangle(frame,(x,y-40),(x+w,y),
            (50,50,255),-1)
            cv2.putText(frame, name_list[serial],
            (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,
            (255,255,255),2)
        else:
            cv2.rectangle(frame, (x,y), (x+w, y+h),
            (0,0,255), 1)
            cv2.rectangle(frame,(x,y),(x+w,y+h),
            (50,50,255),2)
            cv2.rectangle(frame,(x,y-40),(x+w,y),
            (50,50,255),-1)
            cv2.putText(frame, name_list[serial],
            (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,
            (255,255,255),2)
    frame=cv2.resize(frame, (640, 480))
```

```
    imgBackground[162:162 + 480, 55:55 + 640] =frame
    cv2.imshow("Frame",imgBackground)
    k=cv2.waitKey(1)
    # if k==ord('o') and conf>50:
    #     doorAutomate(0)
    #     time.sleep(10)
    #     doorAutomate(1)
    # if k==ord("q"):
    #     break
video.release()
cv2.destroyAllWindows()
```

In trainingdemo.py, it reads grayscale face images from a dataset folder (datasets), extracts the user ID from each image's filename, and uses these to train a face recognition model. It utilizes the LBPHFaceRecognizer from OpenCV for training. The function getImageID() loads all images, converts them to grayscale using the PIL library, extracts the ID from the filename (e.g., User.1.5.jpg $\rightarrow$ ID = 1), and collects both image data and IDs. These are then used to train the recognizer, which is saved to a file named Trainer.yml for later use in real-time face recognition. During training, each image is briefly displayed for visual feedback, and all OpenCV windows are closed afterward. The script concludes by confirming that the training is complete.

```
import cv2
import numpy as np
from PIL import Image
import os
recognizer = cv2.face.LBPHFaceRecognizer_create()
path="datasets"
def getImageID(path):
    imagePath = [os.path.join(path, f) for f in
    os.listdir(path)]
    faces=[]
    ids=[]
    for imagePaths in imagePath:
        faceImage =
        Image.open(imagePaths).convert('L')
        faceNP = np.array(faceImage)
        Id= (os.path.split(imagePaths)
        [-1].split(".")[1])
        Id=int(Id)
        faces.append(faceNP)
        ids.append(Id)
        cv2.imshow("Training",faceNP)
        cv2.waitKey(1)
    return ids, faces
IDs, facedata = getImageID(path)
recognizer.train(facedata, np.array(IDs))
recognizer.write("Trainer.yml")
cv2.destroyAllWindows()
print("Training Completed.....")
```

6 Source Code and Deployment

6.1 Yolo:Bit module

Link to the Yolo:Bit program: https://app.ohstem.vn/
Please import this file Yolobit_code.json.json to get the Yolo:bit program
https://drive.google.com/drive/folders/1nsaWrA1eJkS3AnMkW-X6GoeSqr0YwBig

6.2 Web application module

Link to the web application repository:
https://github.com/l1ttled1no/demo-dadn

6.3 AI model for fire detection

Link to the model:
https://drive.google.com/drive/u/0/folders/1vm4QF$_w$uyBdmcJ$_t$42d8MSL4xkA6sp3M
https://colab.research.google.com/drive/1t4FS$_P$TZh$_l$CwLo59nNTfqQOWVNGpKL3

6.4 AI model for face recognition

Link to the model:
https://drive.google.com/drive/folders/1Fvba3RsgtkuxaOX6UzhfinX8Cuj7-9jN

6.5 Demonstration

Link to the demonstration of our project:
https://drive.google.com/drive/folders/1nsaWrA1eJkS3AnMkW-X6GoeSqr0YwBig

7 Conclusion

In this project, we designed and implemented a smart home system that enhances convenience, security, and environmental awareness. Key features include real-time monitoring of temperature, humidity, and air quality; a secure door access system using user authentication; and a centralized IoT dashboard for remote control and monitoring.
We also integrated voice commands and face recognition for easier, more secure access. To boost safety, we trained an AI model for real-time fire detection via live video, instantly alerting users to potential threats.
A web application was developed for easy system management, allowing users to access sensor data, configure devices, and receive alerts remotely.
However, some limitations remain: the fire detection model needs more training for accuracy, and the face recognition system struggles under poor lighting. Future improvements include refining these AI models and launching a mobile app for greater accessibility and control.
Looking ahead, our roadmap includes continued training and fine-tuning of all AI models to enhance their precision and robustness. We also plan to extend the system's accessibility by developing a mobile application, allowing users to manage and control their smart home conveniently from smartphones or tablets. By doing so, we aim to provide a more comprehensive, user-friendly, and secure smart home ecosystem that adapts to evolving user needs and technological advancements.

References

[1] Ohstem, Yolo: Home- AI va IoT cho nha thong minh, access from https://drive.google.com/file/d/1$_i$5X7eNRt0A48ETs669PU$_{vfq0MPb8/view}$

[2] chipfc (2024), [AIOT Project] Cau truc server Ohstem, access from https://www.youtube.com/watch?v=t3kNm9uFSxA

[3] Ultralytics (2023), Object Detection, access from https://docs.ultralytics.com/tasks/detect/

[4] Ming (2022), How to use MQTT in Flask, access from https://www.emqx.com/en/blog/how-to-use-mqtt-in-flask