DEVELOPING A REAL-TIME OBJECT DETECTION SYSTEM ON FPGA

Master thesis of Paris-Saclay University

Specialization: M2 Communication and Data Engineering

Master student: NGUYEN Trung Kien

Committee

Arnaud BOURNEL	Chairman
Université Paris-Saclay	
Pierre DUHAMEL	Reporter
CNRS, Laboratory of Signals and Systems	
NGUYEN Linh Trung	Examiner
VNU University of Engineering and Technology	
BUI Duy Hieu	Examiner
VNU Information Technology Institute	
Erwan LIBESSART	Examiner
CentraleSupélec	

Thesis supervision

BUI Duy Hieu	Thesis supervisor
VNU Information Technology Institute	
TRAN Thi Thuy Quynh	Co-supervisor of the thesis
VNU University of Engineering and Technology	
Erwan LIBESSART	Co-supervisor of the thesis
CentraleSupélec	

ABSTRACT

AUTHORSHIP

Hanoi, January 22nd, 2024

Author

ACKNOWLEDGEMENT

Contents

List of Abbreviations

Abbreviation	Definition
IoT	Internet of Things
ADAS	Advanced Driver Assistance Systems
AI	Artificial Intelligence
AMD	Advanced Micro Devices
mAP	Mean Average Precision
ASIC	Application Specific Integrated Circuit
AVC	Advanced Video Coding
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
DMA	Direct Memory Access
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GB	Gigabyte
GE	Gate equivalents
GPU	Graphics Processing Unit
HD	High Definition
HOG	Histogram of Oriented Gradients
LTS	Long Term Support
MJPEG	Motion JPEG
MOT15	Multiple Object Tracking Benchmark 2015
MOTC	Multiple Object Tracking Challenge
MPSoC	MultiProcessor System On Chip
NMS	Non-Maximum Suppression
OS	Operating System
PASCAL	Pattern Analysis Statistical Modelling and Computational Learning
PETS09	Performance Evaluation of Tracking and Surveillance 2009
PTZ	Pan-Tilt-Zoom
RAM	Random Access Memory
RCNN	Regional Convolutional Neural Network
RGB	Red Green Blue
SIFT	Scale-Invariant Feature Transform
SPP	Spatial Pyramid Pooling
SRAM	Static Random Access Memory
SSD	Single Shot Detector
SVM	Support Vector Machine

TSMC	Taiwan Semiconductor Manufacturing Company
UHD	Ultra High Definition
VNU	Vietnam National University
VOC	Visual Object Classes
YOLO	You Only Look Once
ZCU106	Zynq UltraScale+ ZCU106 Evaluation Kit

List of Figures

List of Tables

Introduction

Chapter 1. Real-time Object Detection System

This chapter provides an overview of the application of Internet of Things (IoT) technology in monitoring and alerting fire detection systems, primarily in the context of increasingly serious fire problems that require critical attention. It begins with an introduction to IoT and the importance of its role in improving monitoring skills and its value in fire situations. In addition, this chapter also focuses on the main objective of the thesis: to explore and employ the full potential of IoT-connected hardware devices in both IoT applications and real-world scenarios. This chapter is organized into three sections. Section 1.1 provides an overview of the IoT and its applications in fire monitoring systems. *Section 1.2* discusses the current situation and limitations of existing fire monitoring systems. *Section 1.3* presents modern fire detection methods using current technologies, along with their limitations.

1.1. Introduction to the Internet of Things (IoT) and Its Applications for Fire Monitoring Alert Systems

In recent years, fire accidents have become increasingly dangerous, causing critical damage to both human life and property. Traditional fire monitoring systems, which often rely on manual checks or local alerts, are limited in their ability to respond quickly and effectively, especially in complex or remote environments. To handle this issue, we need smart and effective solutions to improve it in time.

The IoT is one exciting technical development that resolves these issues. As its name suggests, IoT refers to a network of smart devices that can communicate and connect with each other, exchanging data. These devices can collect, transmit, and process data in real-time, thereby enabling automation and making personal decisions.

Unlike standalone devices that work independently, a true IoT system (Figure 1) is a scalable and integrated system where multiple devices work together for a common goal. In the context of fire detection, this goal is always monitoring environmental conditions, identifying early signs of fire, and sending real-time alerts, even when no one is present at that location.

By providing features such as remote monitoring, automated notifications, and log data, IoT-based systems significantly enhance responsiveness and overall safety. Therefore, using IoT for fire monitoring systems is both useful and essential for enhancing early detection and minimizing damage.

Figure 1. An IoT system with interconnected smart devices.

1.2. Current situation and Limitations of the monitoring fire alert system

Fire accidents are now common and can occur in various types of environments, ranging from crowded residential areas to industrial zones, forests, etc.... In an industrial environment, the use of flammable materials, high electricity consumption, and poor ventilation are common reasons that increase the risk of fire. In a residential area, the complex and overloaded electrical network increases the chance of short circuits and electrical fires. In a forest environment, rising global temperatures make it more flammable, and also human activities, such as camping or purposeful burning of the forest, increase the risk of fire. As mentioned, the fire can occur anywhere, any time, whether in crowded urban areas or isolated natural areas.

Therefore, deploying fire monitoring systems with sensors and data-collecting modules in large areas is expensive and challenging. Traditional systems, which rely on wired connections or require continuous human monitoring, are not

realistic, especially in large areas or remote and harsh environments. These systems will face problems such as processing complex data and false fire alerts. This is really a big challenge for the fire warning system. The solution using IoT with sensor data can help provide more accurate and useful results.

Currently, traditional fire monitoring and alert systems are still widely used and have not been fully replaced by IoT-based systems. Although some improvements have been made, many issues and limitations still exist, such as detection delays, limited coverage range, and high maintenance requirements. As a result, developing a fire detection and monitoring system that uses fire, temperature, and humidity sensors combined with wireless communication technologies is a creative, possible, worthy solution to consider.

1.3. Related Works

The development of fire monitoring and alert systems based on IoT has gained increasing attention in recent years. This section presents several existing works and discusses their limitations, highlighting the gap that this project aims to address.

Nowadays, IoT systems typically use various sensors, ranging from tiny microcontrollers to large ones, and are on the way to merging machine learning in the new era of AI, which is now very popular. I plan to use AI in my future work. Back to this section – “Related Works”, I believe that anyone who reads this thesis someday will find it easy to refer to the paper I will share below:

In the first paper [1] (2023), the authors proposed an IoT-based fire detection and monitoring system using temperature, flame, and smoke sensors with an ESP32 microcontroller, and implemented a LoRa Network to detect and alert on forest and farm. In paper [2] (2013), the authors have used an Arduino-based home fire alarm system with a GSM module and temperature sensor to detect fire and send SMS alerts, enhancing user safety and property protection. In the next paper [3] (2025), the authors developed a smart IoT-based system using the same list of sensors as paper 1 to detect fire, but in addition, using Arduino UNO and a gas sensor. The system was implemented on the Blynk platform and uses GSM to send alerts. Paper

[3] (2017), the authors designed an STM32-based wireless fire detection system using a block of sensors and embedded wireless communication technologies like Wi-Fi to detect fire. Next paper [4] (2019) shows a big improvement in the application of robots. Pages 267–278 of the paper focus on the use of robots in fire monitoring and alert systems. The project utilizes STM32 and STC89C51 microcontrollers, drones, to connect air and ground monitoring systems and facilitate communication through ZigBee. This paper [5] (2021) focuses more on the alerting part. The authors use ESP32 and PIR sensors to detect fire and movement, sending alerts through a Telegram bot to a smartphone, with feature alarms by image and temperature parameters. When AI becomes popular, some papers nowadays are applying it in IoT systems to detect and alert fire. For example, paper [6] (2023) presents a system that combines IoT devices with YOLOv5 to enable early forest fire and real-time detection, aiming to reduce false alarms and enhance safety during dry seasons. While the approach in [6] is already technically advanced, [7](2025) adds a suggestive improvement to fire monitoring and alert systems by using IoT and machine learning, and also combining various sensor types, to improve detection accuracy and support remote monitoring. Although this recent work represents the development, the core concept of fire detection systems has already been explored in earlier studies.. Indeed, a paper from 2013 [9] discusses a real-time fire alarm system developed using Raspberry Pi and Arduino Uno. This system features include smoke detection, room image capture, and alert through SMS and a web interface. It only sends alerts to firefighters after user confirmation to reduce false alarms. Although this paper was from 2013, its implementation was quite complete – features like login were already added. However, the web design is still basic and a little bit outdated. However, about implementation, it was a meaningful success, and the authors' idea was forward–thinking.

As mentioned at the beginning of Section 1.3, my thesis does not apply AI in the IoT system. However, this will be explored in future work. To recognize and expand on the ideas presented from the science papers, many of which use wireless

technologies such as Wi-Fi or GSM. This thesis proposes enhancing current approaches by applying radio communication to an IoT-based fire detection and alert system. Once again, I am grateful to these “giants” whose pioneering work has allowed this small idea of a thesis to be developed.

1.4. Conclusions

An IoT system is an essential technology with applications in monitoring and fire detection. Achieving real-time monitoring, high accuracy, processing all scenarios, and overcoming hardware limitations creates serious difficulties. The use of IoT brings a promising solution, enabling real-time monitoring, remote access, and automated alert notifications. This project aims to inherit and integrate the ideas from those previous works, while also contributing further improvements to develop a more responsive and reliable fire alert system. It also introduces new innovations to improve the system’s flexibility and performance in real-life situations. In Chapter 2*In System Architecture Design*, a proposal will be presented, focusing on the integration of hardware and software components, including sensor modules, communication systems, and control logic. All of which will be implemented through software to build the proposed IoT-based fire monitoring solution.

Chapter 2. System Architecture Design

The first chapter of this thesis provided an overview of IoT technology and its applications in fire detection and monitoring. It also discussed the current state of fire accidents occurring in many places today, highlighting the need for more efficient solutions. Furthermore, with the current state of fire, the thesis proposed a new direction by enhancing existing ideas with the use of radio communication. While many previous systems deploy on Wi-Fi or GSM, this thesis focuses on using HC-12 wireless modules for *long-range, low-cost, low-power* communication. The main goal is to develop an IoT-based system that monitors, detects fires, and alerts in real-time.

Building on this ideal, *Chapter 2* presents the proposed system architecture, which consists of both hardware and software components designed to work together for efficient fire monitoring. The system includes: Raspberry Pi and STM32 microcontrollers, HC-12 modules for communication, and a variety of sensors to collect fire-related data. This chapter has three main sections: *Section 2.1 Hardware System Architecture*: in this section, an overview of the hardware components used in this thesis will be presented, including sensors, microcontrollers, and the central processing unit. Each component will be explained with the reason why it is used and a summary of its advantages and disadvantages. *Section 2.2: Software & Firmware System Architecture*: This part details how the software & firmware work across the system. It begins with sensor-side firmware, which includes collecting data, encoding, and transmission. On the server side, it explains how data is received, decoded, stored, and visualized through a Flask-based web application. This section also includes user authentication, account management, and how software sends alerts (using Fuzzy Logic and Threshold Evaluation). *Section 2.3 Conclusions:* This final section summarizes all components introduced in *Chapter 2* and prepares for the system implementation and evaluation that will follow in later chapters, *Chapter 3: Implementation Results and Evaluation*.

2.1. Hardware System Architecture

This section introduces the hardware components used at both the *sensor node* and *the central processing unit*. It explains what each component is, its role in the system, and the reasons behind choosing it for the demo. Subsections cover specific components, including fire sensors, temperature and humidity modules, STM32 for edge processing, and the Raspberry Pi for central control and web server hosting. The following *section 2.2* will discuss the data flow in depth, and how each part works together.

2.1.1. Sensor Node Architecture

(1) Flame Sensor

Flame sensors are a type of detector used to identify the presence of fire or flames, which is possible because fires typically occurs various types of radiation – including *visible light* (such as red, orange, and blue), *infrared radiation* (heat), and maybe *ultraviolet radiation* – and flame sensors are specifically designed to detect the infrared spectrum in the 760 nm to 1100 nm range.

In Figure 2, the PISAFE (Figure 2) flame sensor is a solution for detecting fire using Wi-Fi connectivity and wide-area fire detection (up to 100 meters), while also meeting fire safety certification standards. However, the cost of it is really high, so this sensor will not be used in the context of this thesis.

Instead, to align with the project's budget and for demo purposes, the **KY-026** flame sensor (Figure 3) is selected. Despite its small size and lower cost, the KY-026 is a good option for demo and educational applications for flame detection in this thesis.

Figure 2. Flame sensor PISAFE.

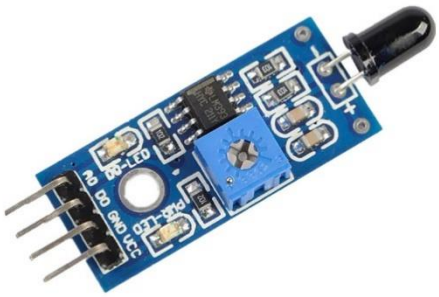

Figure 3. Flame sensor KY-026.

(2) Temperature & Humidity Sensor

A temperature and humidity sensor is a type of sensor used to collect environmental data. When a fire occurs, such as when flammable materials or electrical wires are burning, the temperature of the fire will increase continuously, and the humidity drops quickly. In the process of fire detection and monitoring, just using the flame sensor's evaluation is not enough. Therefore, combining data from both the Temperature & Humidity Sensor and the Flame Sensor can provide a more accurate

result and enhance the fire detection system's reliability, helping to reduce false and prevent incorrect fire alarms. Nowadays, there are many types of sensors collecting environmental data, like Honeywell 5809SS [9] (Figure 4). Similar to the flame sensor, this system is designed to be low-power and low-cost, and as the main objective of the thesis is for demo purposes. Therefore, it uses the DHT11 (Figure 5) sensor to collect temperature and humidity data. The DHT11 is a widely used and accessible sensor in educational environments, especially in university-level projects.

Figure 4: Honeywell 5809SS heat detector.

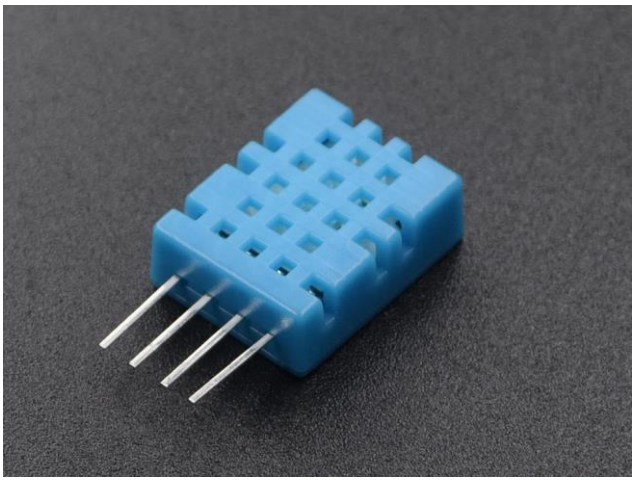

Figure 5: DHT11 - Temperatue & Humidity Sensor.

(3) Edge processor

In an IoT system, the edge processor is a lightweight microcontroller responsible for collecting data from a block of sensors, such as flame sensors,

temperature and humidity sensors. Each microcontroller that handles the data from one or more sensors can call a sensor node within the system.

Usually, edge processors use microcontrollers like the ESP32 (Figure 6), ESP8266, or other microcontrollers, which are favored because of its programming and integration. However, in this thesis, the selected edge processor is the STM32, specifically the STM32F103C8T6 (Figure 7). Although programming the STM32 may be more complex compared to ESP-based microcontrollers, but it is chosen to support the research and application of this work. Additionally, STM32's UART communication mode is effective for serial data transmission, making it an right choice for handling sensor data and transmitt data in this system.

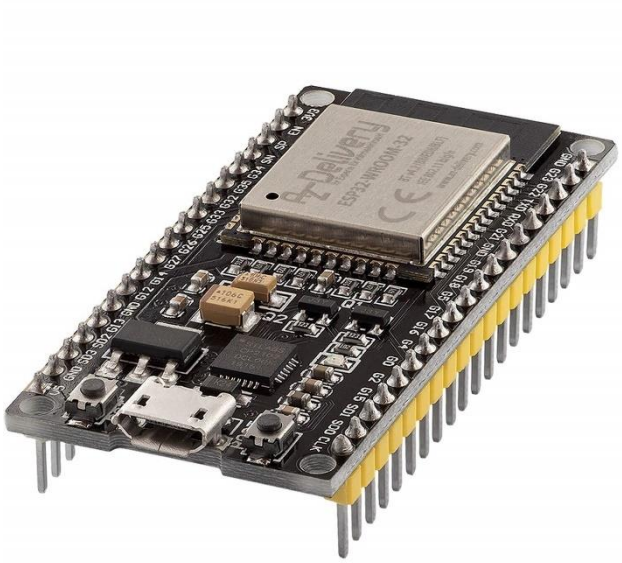

Figure 6: ESP32 Microcontroller.

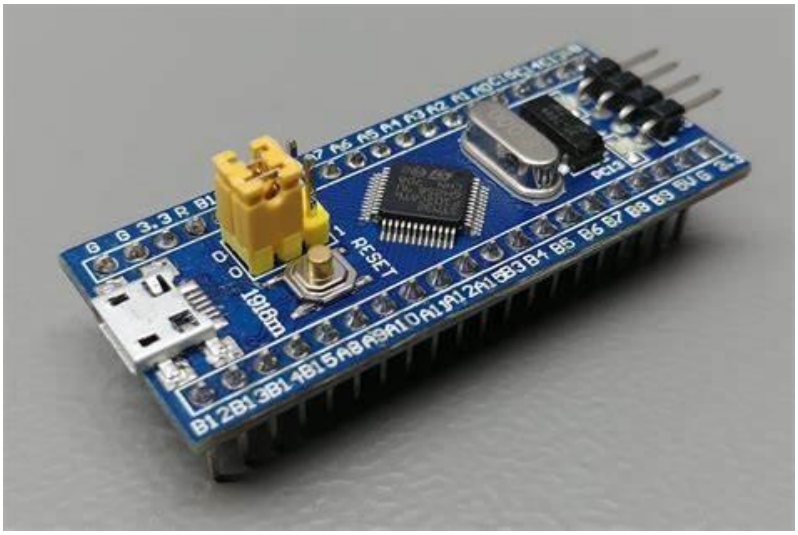

Figure 7: STM32F103C8T6 Microcontroller.

(4) Wireless Communication Module

This module plays an important role in the main idea of this thesis. In the proposed IoT system, the wireless communication module is the part that transmits data between sensor nodes and the central processing unit. Several types of wireless communication technologies were reviewed in *Section 1.3 Related Work*, including Wi-Fi, Zigbee, Cellular, and LoRa.

Among them, Wi-Fi, Zigbee, and Cellular require an existing communication infrastructure such as routers, gateways, or mobile network towers. On the other hand, LoRa can operate in two modes: LoRaWAN, which is based on centralized infrastructure (gateways and network servers), and LoRa (peer-to-peer), which can communicate directly between nodes without any infrastructure.

Despite the advantages of LoRa in terms of long-range communication and energy efficiency, it often comes at a higher cost and requires more complex configuration. In opposition, the HC-12 module offers a low price and is easy to configure, making it suitable for the goals and scope of this thesis project. In the *2.1.1 Node System Architecture*, the HC-12 module is placed in the transmission (Tx) block. After the sensor data is collected and processed by the STM32F103C8T6 microcontroller, the resulting data packet is transmitted to the HC-12 module (Figure 8), which then sends the data wirelessly via radio frequency (RF) communication.

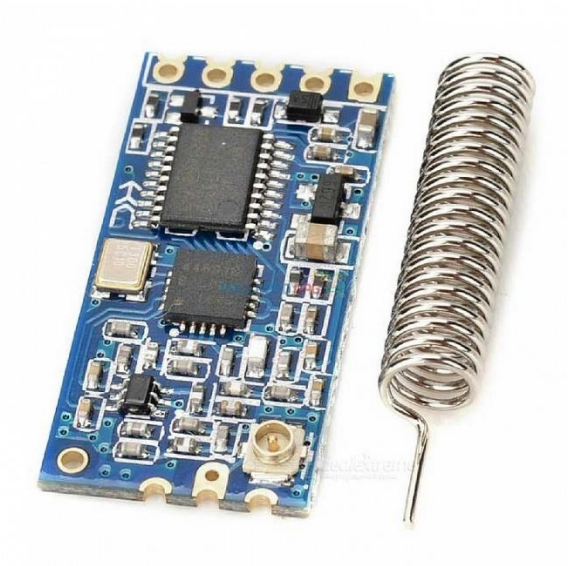

Figure 8: HC-12 Wireless Communication Module.

2.1.2. Central Processing Unit System Architecture

(1) Wireless Communication Module

In the *2.1.2 Central Processing Unit System Architecture*, the HC-12 module (Figure 8) is placed in the receiver (Rx) block. After the data is transmitted from the node to the central processing unit through radio communication, the HC-12 module at the receiver receives incoming signals and transmits them to the central processor for handling.

(2) Central processing unit

In this section, the central processing unit is using for handling incoming data after it has been wirelessly transmitted to the receiver block (in this case, the HC-12 module). There are many types of CPUs suited of performing logical data processing and running a web server. However, with the objective of this thesis, the system using the Raspberry Pi 5 (Figure 9) to explore its practical application in real-world IoT scenarios.

The Raspberry Pi 5 is a single-board computer (SBC) — as the name implies, it is a complete computer built on a single circuit board. With only a power supply and an external monitor, it can working as a fully like a computer. It supports many type of operating systems like : Raspberry Pi OS, Raspberry Pi Lite, Ubuntu,…etc…, among which Pi OS being the most commonly used.

For storage, it uses a microSD card, supporting capacities up to 1 TB, however a normal and reliable setup runs from 32 to 256 GB. The Pi 5 is powered by a quad-core ARM Cortex-A76 processor, provides high performance for embedded applications. In this thesis, the Raspberry Pi 5 equipped 8 GB of RAM, making it the same to many standard desktop PCs.

About connectivity, it provides USB ports, Ethernet, Wi-Fi, Bluetooth, a 3.5 mm audio jack, camera ports and also 2 micro HDMI for connecting screen and support to 4k HD. Importantly, it also features a 40-pin GPIO header, including 26 programmable GPIO pins, with the rest used for power (3.3V/5V - VCC) and ground connections (GND). These GPIO pins allow the Pi to connect with sensors, LEDs, buzzers, and communication modules like the HC-12.

With its wide features, fast performance, and flexibility, the Raspberry Pi 5 is an excellent choice for developing modern IoT systems.

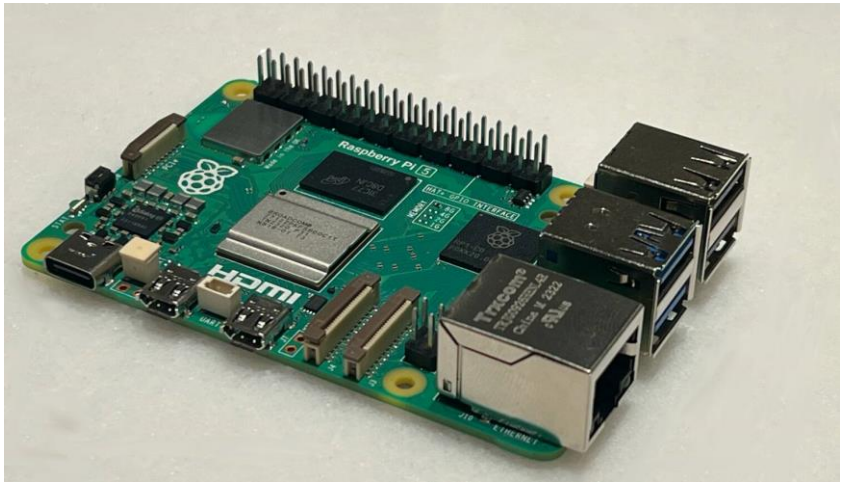

Figure 9: Single board Computer - Raspberry Pi 5.

(3) Touchscreen Monitoring Interface

In a typical IoT-based monitoring system, external touchscreen displays are often connected to the Central Controller Unit (Raspberry Pi 5), especially when building a web-based application. These interfaces enable users to easily interact with the system and monitor its status in real time (Figure 10).

Although this project does not focus on physical user interfaces, adding a touchscreen in the demo setup improves system visualization and user interaction.

Therefore, a touchscreen NoName A (Figure 11) is used. It works well with the custom monitoring software in this project. This addition helps users understand the system more clearly, test it more easily, and consider improvements for future development.

Figure 10: Touchscreen Monitoring.

Figure 11: NoName A Touchscreen.

2.2. Software & Firmware System Architecture

Section 2.1 focuses on introducing the hardware components used in this thesis, along with the reasons for their selection and their unique benefits. *Section 2.2. Software & Firmware System Architecture* discusses the working flow of the full system implementation. *Section 2.2* is divided into two subsections: *Section 2.2.1.*

Firmware at the Node explains how data is handled at the sensor node after collecting data from flame, temperature, and humidity sensors. It also provides the standard requirements for protecting the privacy and security of data during wireless transmission. *Software at the Central Processing Unit* uses software to receive, decode, and process incoming data packets. The decrypted data will be uploaded in real-time onto a web-based platform. This subsection also covers user administration, data logging, notification handling, and other system operations, which will be described in further depth.

2.2.1. Firmware at the Node

(1) Sensor Data Collection

This section will discuss data collection and the configuration of the pins for each type of sensor. In the *Sensor Data Collection* part, there will be two subsections: a) DHT11 and b) KY-026. The application used for configuration in this section is STM32CubeMX, while the firmware for the STM32F103C8T6 will be developed using STM32CubeIDE. Additionally, other tools such as STM32CubeProgrammer (STM32 Utility) will be used for debugging and verifying the specific type STM32 in use.

a) DHT11

DHT11 employs a 1-Wire communication protocol, meaning a single data line is used for both control signaling and data exchange. Initially, the STM32 microcontroller pulls the data line low for 18 ms, then pulls it high for a 20µs pulse to await the sensor's response. To make DHT11 work correctly, the configuration is as follows: *Data pin*: GPIOA Pin 6. *GPIO Configuration*: GPIOA Pin 6 set to mode Output, no pull-up, and no pull-down. *Timing requirements*: Hardware timers are employed to provide accurate microsecond-level delays.

After setting and configuring the DHT11 sensor, the method DHT11_Check_Response() is used to check if it responded. If the return value is one, this means the DHT11 is working successfully and is ready to transmit data. The sensor transmits 5 bytes of data in the following sequence: *Rh_byte1*: Humidity –

integer part, *Rh_byte2*: Humidity – decimal part, *Temp_byte1*: Temperature – integer part, *Temp_byte2:* Temperature – decimal part, and the last is *SUM* to calculate the Checksum to identify sensor data. The checksum is used to verify data. If the total of the first four bytes matches the fifth byte (SUM), the data is accepted and the sensor responds correctly.

b) KY-026

The KY-026 flame sensor has a simple configuration compared to the DHT11. It has a digital output pin (D0) that returns logic values: HIGH (1) for no flame and LOW (0) for flame detection. To work with the KY-026 using the A0 pin (analog output), it can still be connected through PA1, as in the configuration. Data pins: GPIOA Pin 1, GPIOA Pin 0. GPIO configuration: GPIOA Pin 1 is set to input mode, while GPIOA Pin 0 has been enabled in the hardware configuration for the ADC.

```
loop forever
   start DHT11, confirm presence
   Rh_byte1, Rh_byte2, Temp_byte1, Temp_byte2, crc ←
read five bytes

   valid ←  ((Rh_ byte1+Rh_ byte2+ Temp
_byte1+Temp_byte2) & 0xFF) = crc

   Temperature ←  float(Temp  byte1)
   Humidity    ← float(Rh _byte1)
   Fire ← readGPIO(GPIOA, PIN1)
end loop
```

Listing 1: Read DHT11 Data and Fire Sensor State.

(2) Encoding Message

This is a highly important and interesting part when transmitting messages over wireless communication. It ensures compliance with standard requirements for protecting privacy and securing data in wireless transmission.In wireless data security, the main components typically include (

Figure 12): Frame Packing, Security Coding, Channel Coding, Spreading, Frame Synchronization, and Modulation - This part will be in *Wireless Transmission*. In the Encoding Message section, each of these components will be described in detail

through corresponding steps, to clearly understand how the message is processed and prepared before transmission.

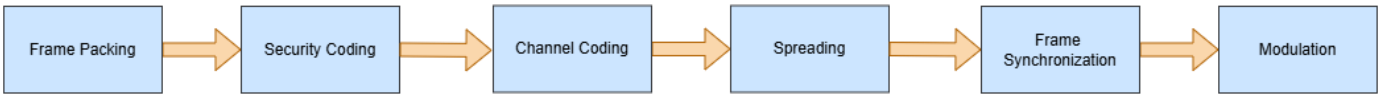

Figure 12: Data protection sequence in wireless transmission.

a) Frame Packing

Input: Temperature, Humidity, Fire state**.**

Output: MavLink frame (with header, payload, CRC).

In the Frame Packing step, the message is converted into a common communication format, similar to how two people communicate using the same language. In this thesis, the MAVLink protocol is used to package the message into a packet structure, making it easier to manage communication between the sender and receiver. MAVLink is a lightweight and efficient communication protocol; it applies in embedded systems and wireless transmission. It helps standardize data exchange and reduces transmission errors. The MAVLink frame consists of three main parts: Header, Payload, and Checksum (CRC). In this thesis, MAVLink version 1 is used. Each frame can be up to 263 bytes, with the Header including 6 bytes, the Payload ranging from 0 to 255 bytes, and the CRC covering 2 bytes.

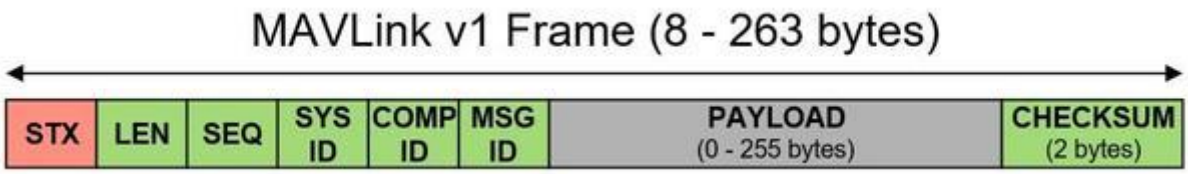

Figure 13: Mavlink v1 Frame.

Diving into the structure of the MAVLink frame, the Header is a crucial component consisting of 6 fields: *Start of Frame (STX)* – 1 byte: This is a fixed start byte with the value 0xFE, which helps the receiver detect the beginning of a MAVLink frame. *Length (LEN)* – 1 byte: This indicates the size of the payload in bytes. Since it is limited to one byte (maximum 0xFF), the maximum payload size is 255 bytes. *Sequence (SEQ)* – 1 byte: A sequence number used to track the order of transmitted messages. *System ID (SYS ID)* – 1 byte: Identifies the system or device that is sending the message. *Component ID (COMP ID)* – 1 byte: Specifies the component in the system that originates the message (e.g., GPS module, Gimbal, etc). *Message ID (MSG ID)* – 1 byte: Identifies the type of message sent (e.g., heartbeat, sensor data, command, log, etc).

The Payload after the header can range from 0 to 255 bytes, depending on the message type. It contains the actual message content, usually in hexadecimal format. After data is collected in the Collecting Data stage, the values for temperature, humidity, and fire status (Intput) are copied into the payload section of the MAVLink message.

Finally, the CRC consists of 2 bytes. It is an important part of the MAVLink frame, as it ensures data integrity. The CRC is computed over the entire frame, not including the STX byte, and any frame with an invalid CRC is considered damaged and will be discarded by the receiver.

b) Security Coding

Input: MavLink frame**.**

Output: PKCS7 padding and bit masking.

This stage improves the message with extra transformations after MAVLink framing in order to reduce the risk of attacks during wireless transmission. Security coding employs a range of block encryption algorithms, including AES-128, AES-256, and others. However, in this thesis, block encryption techniques have not yet been deployed. Instead, this stage is the beginning of future enhancements if block

encryption is to be used later. It also serves as the base for secure message delivery through wireless communication networks.

In this thesis, PKCS7 is employed for padding encoding. After MAVLink framing, the message may not be matched to fixed-size blocks (for example, 16 bytes). As a result, padding is required as *the very first step* to ensure flexibility with future encryption techniques.

PKCS7 calculates the remaining data when the message length is split by the block size (for example, 16 bytes). If the message is not a multiple of the block size, padding bytes are used. For example, if the message is three bytes short, three bytes of 0x03 are added. If the message length is already a multiple of the block size, a complete block of padding with the value 0x10 (standing for 16 in decimal) is appended. After padding, the message is converted into a series of fixed-length blocks that are suitable for encryption.

Bit masking is used to mask and randomize the message even more. In this thesis, a simple type of bit manipulation is used: bit reversal, which reverses each byte in a block at the bit level. This transformation is dependent on the use case or goal of the user; however, it is done here as a lightweight technique for improving bit-level randomness before future encryption incorporation. Bit masking may also be applied after block encoding using algorithms such as AES-128, AES-256, and others.

c) Channel Coding

Input: PKCS7 padding and bit masking.

Output: LDPC encoding.

Because the encrypted communication is transferred wirelessly, mistakes are unavoidable as a result of external interference. To solve this issue, this thesis applies channel coding, especially the LDPC (Low-Density Parity-Check) method, as referenced from Wikipedia [9].

LDPC is a class of error-correcting codes that uses a sparse parity-check matrix, commonly called H. This matrix will be used in both the transmitter and receiver. The

structure and properties of matrix H depend on the type of LDPC chosen for the application.

In the implementation of channel coding, LDPC 1/2 adds parity bits to each encoded byte; using this encoding will double the length of each block after padding and bit masking. This enables the calculation of the syndrome during decoding. A correctly received message will produce a zero syndrome vector, indicating that no errors have been detected. This attribute is critical to LDPC's ability to detect and repair transmission faults quickly.

d) Spreading

Input: LDPC encoding.

Output: LFSR encoding.

This process confuses the signal and prevents listening during wireless transmission by producing pseudo-random sequences, which improves communication dependability across wireless channels. As a result, the message becomes more difficult to identify, more difficult to detect or attack, and stronger when delivered through a harmful environment.

In the Spreading phase, this thesis employs an LFSR (Linear Feedback Shift Register). There are multiple types of LFSR implementations, and more information can be found on Wikipedia. Specifically, the thesis uses a Fibonacci-type LFSR [10].

It is important to note that the input message to the LFSR is the LDPC-encoded message, which has already been doubled in length due to channel coding. Therefore, when applying the LFSR spreading, special attention must be paid to ensure that the same register is used at both the transmitter and the receiver. At the LFSR encoding steps, each block of the message at this stage will be XORed with the output of the Fibonacci LFSR.

Input: LFSR encoding.

Output: Frame Sync packing.

After going through the processes of framing, encryption, and spreading, the message becomes significantly confused, making it difficult for the receiver to identify the beginning or structure of the transmission. Therefore, as the final step, two header bytes (0xEB and 0x90) are added after LFSR encoding.

These specific bytes were chosen because 0xEB and 0x90 are commonly detectable in wireless environments. They frequently appear in the radio spectrum, making them reliable markers. By adding these two bytes at the beginning of the final message, the receiver can more easily synchronize and detect the start of a valid transmission.

(3) Wireless Transmission

Input: Frame Sync packing.

Output: Message transmitted.

Before implementing this stage, it is necessary to introduce the HC-12 wireless module. The module must be properly configured before use, which can be done using Hercules, a serial communication tool that allows access to COM ports.

To interface with the HC-12, a USB to UART converter (RS232 to TTL mode) is used. This mode allows access to both the TX (transmit) and RX (receive) pins, enabling full-duplex communication. After the STM32 encrypts the message, it sends the message through USB UART to the Hercules terminal. If the message is received correctly, the first step in the transmission process is complete.

The next step is to configure the HC-12 module. This module features a SET pin, which enables configuration mode when pulled LOW. According to the documentation [11], enabling the SET pin places the module in default mode with the following parameters:

- Baud rate: 9600.
- Stop bits: 1.
- Parity: None.

To reconfigure the module, the AT command is used to verify the connection (a response of “OK” confirms). From there, configuration can be performed using the following AT commands:

- AT+Bxxx: Set baud rate.
- AT+Cxxx: Set channel.
- AT+Axxx: Set address.

An important requirement is that the baud rate of both the transmitter and receiver HC-12 modules must match. In this thesis, the chosen baud rate is 115200 for high-speed transmission. Therefore, the STM32 UART interface must also be configured to 115200 through STM32CubeMX.

In addition to the baud rate, the wireless channel (AT+Cxxx) must be the same on both sides to ensure communication.

As mentioned in the Encoding Message section, modulation lies within the wireless transmission layer. In this case, the HC-12 module already handles modulation through its integrated RF circuitry. Therefore, after the message (in hexadecimal format) is transmitted to HC-12 through the TX pin (GPIOA Pin A9) - as configured in STM32CubeMX - the wireless transmission process is considered complete.

2.2.2. Software at the Central Processing Unit

(1) Wireless Receiver

Input: Message transmitted.

Output: Message received.

First, regarding the setup, the AT configuration parameters for the receiving side have already been described in the Wireless Transmission section. After the message is transmitted wirelessly, the receiving unit of the Central Processing Unit will collect the incoming data. The data becomes valid if the receiving side detects the first two bytes as 0xEB and 0x90, which are used as synchronization headers.

(2) Decoding Message

a) Frame Synchronization Decode

Input: Message received.

Output: Message unpack Frame Sync.

Once the received message is detected as valid, the Frame Synchronization Decode step removes the first two bytes (0xEB and 0x90), which serve as the synchronization header

b) Spreading Decode

Input: Message unpack Frame Sync.

Output: Message decode Spreading.

Because both the transmitter and the receiver use the same register after applying the LFSR Fibonacci, to decode the spreading, XOR each block again with the LFSR Fibonacci output, just as in the LFSR encoding step.

c) Channel Coding Decode

Input: Message decode Spreading.

Output: Message decode LDPC.

As previously said, Channel Coding is important to the message encoding process. When a message is transmitted wirelessly, external interferences can affect the signal, leading to damaged data. To fix this, a shared parity-check matrix H (described in the *Channel Coding - Encoding Message* section) is used during the LDPC decoding step.

In LDPC decoding, matrix H is applied to compute the syndrome. If the syndrome vector contains any non-zero elements, it show the presence of errors in the received message. The decoder then tries to correct the errors by flipping bits.

In this LDPC version, each data byte pairs up with a parity byte rather than creating one large parity block for a longer segment (e.g., 16 bytes of data and 16 bytes of parity). This design makes error localization and correction easier, since each 2-byte segment can be independently checked and corrected.

For example, suppose the first row of matrix H contains 1s at places 0, 3, and 7. If the 16-bit vector (2 bytes) under evaluation has a mistake in any of these points, XOR-ing the bits at positions 0, 3, and 7 will result in a non-zero syndrome, showing a parity error.

When an error is detected, the algorithm identifies the bit that contributes to the highest number of violated parity equations, and flips that bit. The syndrome is then recalculated. This process repeats until: The syndrome becomes all zero (i.e., no more errors), or a maximum of 10 rounds is reached (as specified in this thesis), which can be changed depending on application requirements. If, after 10 rounds, the syndrome still indicates errors, the message is considered uncorrectable and the message with errors will be discarded.

d) Security Coding Decode

Input: Message decode LDPC.

Output: Bit Masking Decode, and unpadding PKCS7.

After LDPC decoding, if the message contains no errors, the syndrome vector is all zeros. It proceeds to the Bit Masking Decode stage. As mentioned earlier, bit masking in this system only applies a simple bit-reversal operation during encoding, so decoding reverses the bits again to restore the message, prepared for PKCS7 unpadding.

Next, in the PKCS7 unpadding stage, the decoder inspects the last byte of the message, which includes how many padding bytes were added. For example, if the last byte is 0x03, this indicates that three padding bytes were appended during encoding, each with the value 0x03. During unpadding, the decoder checks whether the final three bytes all equal 0x03. If any of the padding bytes are incorrect, the message is considered invalid. If all padding bytes are correct, they are removed, and the original unpadded message is returned.

e) Frame Packing Decode

Input: Bit Masking Decode, and unpadding PKCS7.

Output: Temperature, Humidity, Fire state.

Ở bước này sau khi đã unpadding PKCS7. Nếu byte đàu tiên là 0xFE thì có nghĩa đây là bản tin Mavlink packing. Sau đó ta sẽ kiểm tra các dữ liệu như là byte LEN để kiểm tra xem độ dài của bản tin với Payload có trùng nhau không. Tiếp tới là kiểm tra SEQ để xem thứ tự bản tin gửi tới. Đối với các bản tin quan trọng cần log kĩ lượng thì khi gửi đi thì SEQ là phần rất quan trọng khi có thể log ra được các bản tin bị gửi thiếu số lượng, thứ tự bản tin sai. Cũng như với SYS ID và COMP ID sẽ giúp xác nhận hệ thống, bộ phận nào gửi bản tin đến. Và quan trọng nhất là CRC. CRC ở bên Receiver sẽ được so sánh với CRC tính mới từ byte thứ 2 trở đi (bỏ qua STX) nếu 2 cái CRC này trùng nhau thì là bản tin hợp lệ, còn khác nhau thì bản tin lỗi, bỏ qua bản tin. Từ đó nếu bản tin hợp lệ ta sẽ có Output là 3 giá trị nhiệt độ, độ ẩm, Fire state.

a method where the primary gradient vector was split into two vectors, as shown in **Error! Reference source not found.**. The values of these two vectors for pixel I (x,y) were calculated using, which takes into account the values of the four surrounding pixels I(x+1,y), I(x-1,y), I(x,y+1), and I(x,y-1).

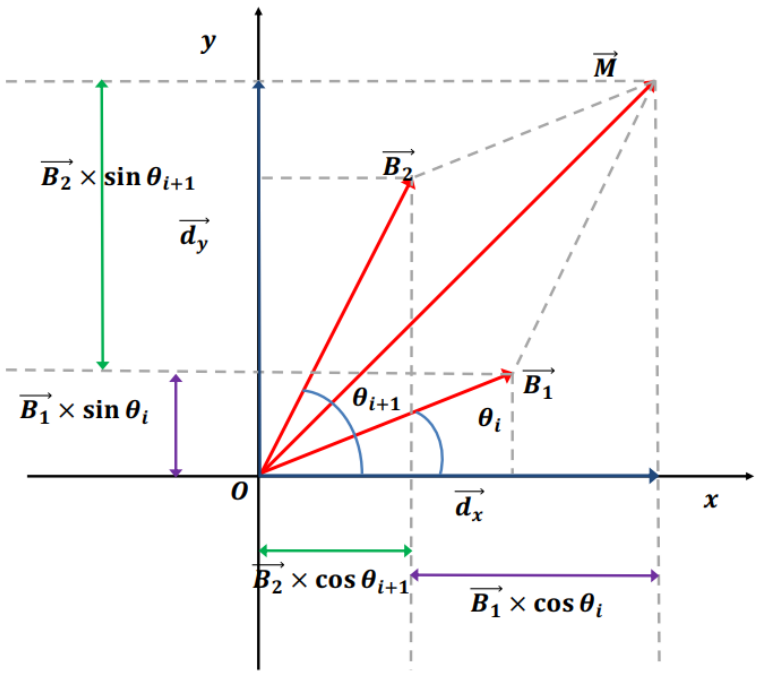

Figure 14 - Decomposition of a vector into the form of two vectors [18].

Two magnitudes are the solutions of the two equations:

$$\left|\left|\overrightarrow{B_{\theta_i}}\right|\right| = \frac{\sin(\theta_{i+1})\, d_x - \cos(\theta_{i+1})\, d_y}{\sin(20)};\ ||\overrightarrow{B_{\theta_{i+1}}}|| = \frac{\cos(\theta_i)\, d_y - \sin(\theta_i)\, d_x}{\sin(20)}$$

This thesis uses this technique to improve the HOG-SVM algorithm and decrease the number of calculations needed to process each sliding window.

real-time applications with power constraints.

As mentioned above, the Object Detection Block will include Bilinear Interpolation Scale Generator Module, HOG-SVM module, combine module, and NMS module.

to form a block, and the cell histograms are normalized using block data.

(a) <u>Gradient calculation and Histogram generation</u>

This

).

Algorithm 1 – Gradient calculation.

will generate all the histograms for the input image used for the next step.

Algorithm 2 – Histogram generation.

(b) <u>Descriptor generation and Result</u>

After

generation.

The combining

remove redundancies. This block is crucial to this thesis's proposed object detection system.

2.3.

2.4. Conclusions

In this

architecture offers a promising pipeline for efficient and accurate human detection for real-world applications.

Chapter 3. Implementation and Evaluations

In the previous

3.1. Experiment setup environment

3.1.1. Mean Average Precision

(1) Mean Average Precision (mAP)

an object

(2) Mean Average Precision (mAP)

Average

Here is a summary of the steps to calculate the AP:

1. *Generate the prediction scores using the model.*
2. *Convert the prediction scores to class labels.*
3. *Calculate the confusion matrix—TP, FP, TN, FN.*
4. *Calculate the precision and recall metrics.*
5. *Calculate the area under the precision-recall curve.*
6. *Measure the average precision.*

The mAP is calculated by finding Average Precision(AP) for each class and then averaging over several classes.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi$$

3.1.2. Frame rate

3.2. Experimental results

3.2.1. Input: PETS09-S2L1 [19]

(1) Input description

-S2L.

(2) Result

and mAP graphs per frame of input PETS209-S2L.

(3) Analysis

low mAP score.

3.2.2. Input: TUD-Stadtmitte [20]

(1) Input description

Input racteristic in the photo is that the moving object is only human moves in groups.

(2) Result

In of 51

.

(3) Analysis

The system can process images in near-real-time but faces significant challenges in accurately identifying and tracking human groups in urban pedestrian settings. The moderate precision combined with low recall and mAP scores suggests that the system might misidentify groups or miss them altogether.

3.2.3. Input: TUD-Campus [21]

(1) Input description

TUD-Campus.

(2) Results

In the

D-Campus.

(3) Analysis

The system

fast.

3.3. Conclusions

Conclusions and Perspective

References

[1] Pradeep, Aneesh; Latifov, Akmaljon; Yodgorov, Ahborkhuja; Mahkamjonkhojizoda, Nurislombek, "Hazard Detection using custom ESP32 Microcontroller and LoRa," *IEEE,* no. 26 May 2023, 2023.

[2] N. I. M. E. N. M. Z. a. K. S. S. K. M. N. N N Mahzan, "Design of an Arduino-based home fire alarm system with GSM module," *Journal of Physics: Conference Series,* vol. 1019, 2917.

[3] N. I. M. E. N. M. Z. a. K. S. S. K. M. N. N N Mahzan, "Design of an Arduino-based home fire alarm system with GSM module," *Journal of Physics: Conference Series,* vol. 1019, 2017.

[4] H. Y. ·. J. L. ·. L. L. ·. Z. J. ·. Y. L. ·. D. Zhou, "Development of Four Rotor Fire Extinguishing System for Synchronized Monitoring of Air and Ground for Fire Fighting," *ICIRA,* p. 267–278, 2019.

[5] N. Komalapati, V. C. Yarra, L. A. Vyas, Kancharla and T. N. Shankar, "Smart Fire Detection and Surveillance System Using IOT," *IEEE,* 2021.

[6] A. E. H. 1. A. S. S. 1. K. 2. B. A. 1. a. I. C. y Kuldoshbay Avazov 1, "Forest Fire Detection and Notification Method Based on AI and IoT Approaches," *Future Internet (ISSN: 1999 - 5903),* vol. 15, no. 2, 2023.

[7] L. WHITE and R. AJAX, "Improved Fire Detection and Alarm Systems.," 2025.

[8] "spntelecom," [Online]. Available: https://spntelecom.vn/cam-bien-nhiet-pisafe-dau-bao-nhiet-khong-day-phong-chay-canh-bao-chay-thong-minh-wifi-dat-tieu-chuan-pccc-phat-hien-chay-no-tu-xa-qua-dien-thoai.

[9] "ManualMachine," Honeywell Home, [Online]. Available: https://digitalassets.resideo.com/damroot/Original/10002/L_5809SSD_D.pdf.

[10] Nguyen, N. S., Bui, D. H., & Tran, X. T., "Reducing temporal redundancy in MJPEG using Zipfian estimation techniques," *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS). IEEE,* pp. 65-68, 2014.

[11] A. Manzanera, "σ-δ background subtraction and the Zipf law," *Progress in Pattern Recognition, Image Analysis and Applications: 12th Iberoamericann Congress on Pattern Recognition, CIARP 2007,* pp. 42-51, 2007.

[12] Lacassagne, L., Manzanera, A., & Dupret, A, "Motion detection: Fast and robust algorithms for embedded systems," *IEEE international conference on image processing (ICIP),* pp. 3265-3268, 2009.

[13] Richefeu, A. M. J., "A robust and computationally efficient motion detection algorithm based on σ-δ background estimation," *Indian Conference on Computer Vision, Graphics & amp; Image Processing,* vol. 9, pp. 16-18, 2004.

[14] Dalal, N., & Triggs, B., "Histograms of oriented gradients for human detection," *IEEE computer society conference on computer vision and pattern recognition,* 2005.

[15] Dollár, P., Belongie, S., & Perona, P., "The fastest pedestrian detector in the west," 2010.

[16] Dollar, P., Wojek, C., Schiele, B., & Perona, P., "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence, 34(4),* pp. 743-761, 2011.

[17] Cristianini, N., & Shawe-Taylor, J., An introduction to support vector machines and other kernel-based learning methods, Cambridge university press, 2000.

[18] Ho, H. H., Nguyen, N. S., Bui, D. H., & Tran, X. T, "Accurate and low complex cell histogram generation by bypass the gradient of pixel computation," *4th NAFOSTED Conference on Information and Computer Science,* pp. 201-206, 2017.

[19] Ferryman, J. & Shahrokni, A., "PETS2009: Dataset and challenge.," *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance,* 2009.

[20] Andriluka, M., Roth, S. & Schiele, B. , "Monocular 3D Pose Estimation and Tracking by Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2010.

[21] Andriluka, M., Roth, S. & Schiele, B., "People-Tracking-by-Detection and People-Detection-by-Tracking," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2008.

[22] A. Suleiman, "An energy-efficient hardware implementation of HOG-based object detection at 1080HD 60 fps with multi-scale support.," 2016.

[23] Nguyen, N. D., Bui, D. H., & Tran, X. T., "A novel hardware architecture for human detection using HOG-SVM co-optimization," *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05).,* vol. 1, pp. 886-893, 2019.

[24] Nguyen, T. A., Tran-Thi, T. Q., Bui, D. H., & Tran, X. T, "FPGA-Based Human Detection System using HOG-SVM Algorithm," *International Conference on Advanced Technologies for Communications (ATC),* pp. 72-77, 2023.

[25] [Online]. Available: https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/.

[26] [Online]. Available: https://www.v7labs.com/blog/mean-average-precision.

[27] T. Panda, R. Banerjee, A. Pal, S. K. Bishnu and A. Chakraborty, "IOT-Based Home Automation for LPG Gas and Fire Detection System With Automated Safety Measures," *IEEE,* no. 16 April 2025, 2025.

[28] Shanghai Xian Dai Architecture,Engineering&Consulting Co. ,China , "Information fusion technology based on wireless fire detection and alarm

system," *Journal of Physics: Conference Series,* pp. 883-887, 21-11-2013.

[29] M. S. B. Bahrudin, "Development of Fire Alarm System using Raspberry Pi and Arduino Uno," *2013 International Conference on Electrical, Electronics and System Engineering (ICEESE),* 2013.