SUP'COM
Higher School of Communication of Tunis

Scope Statement:

SMART FIRE DETECTION SYSTEM

Authors:
Nour Mabrouk
Bacem Ahmed

Faculty Adviser:
MOHAMED BECHA
Kaaniche

Contents

1 Concept

1.1 Project context

With the rise of urbanization and industrial activities, the risk of fire incidents has increased, necessitating improved fire detection and response systems. Early detection of smoke or gas emissions can help prevent potential fires and mitigate their impact on public safety and property. Traditional fire alarms often rely on manual checks and on-site maintenance, which can be time-consuming and inefficient, especially in large or remote locations. To address these challenges, a smart smoke detection system integrated with Cloud of Things (CoT) and Location-Based Services (LBS) technologies is essential. This system enables real-time monitoring of air quality and smoke levels, providing timely alerts and the precise location of detected smoke to help users respond quickly to potential fire hazards. Additionally, if users fail to respond to a notification within 1 minute, the system will automatically contact emergency services, ensuring swift intervention. The integration of an iOS application enhances accessibility, allowing users to monitor and respond to alerts on any device.

1.2 Problem statement

Current smoke detection systems lack real-time connectivity, remote access, and location tracking, which are critical for prompt responses to fire risks. Traditional systems are often limited to specific locations and do not offer users the ability to monitor conditions or receive location-based alerts remotely. In cases where these systems fail to detect smoke in time or do not report the precise location, the consequences can be devastating, leading to property damage and potential harm to human lives. Furthermore, traditional systems lack an automated emergency call feature if users fail to respond, which can delay assistance in critical moments. This limitation creates an urgent need for a modernized solution that can detect smoke, immediately provide the location of potential fire hazards, and automatically contact emergency services if users do not respond within 1 minute, enhancing safety and responsiveness across various environments.

1.3 Objectives

- **Real-time smoke and gas detection**: Continuously monitor smoke and gas levels in the environment and detect anomalies immediately.
- **Threshold-based alerts**: Send instant notifications to users when smoke levels exceed a predefined threshold, enabling rapid response to potential fire hazards.
- **Location tracking via LBS**: Utilize LBS to pinpoint the exact location of detected smoke, allowing emergency responders or users to quickly locate and address the potential fire source.

- **Data logging and historical analysis**: Store detected events, environmental conditions, and location data for later analysis, which can help improve safety protocols and system effectiveness.
- **Automatic emergency call**: If users do not respond to a notification within 1 minute, the system automatically contacts emergency services, ensuring swift assistance in case of fire or gas-related incidents.

2 Client

The clients who will benefit from this solution include **homeowners, business owners, and facility managers** seeking an efficient, real-time smoke detection system to enhance safety and prevent fire-related incidents. This project provides these clients with a smart smoke monitoring system capable of detecting smoke and hazardous gases, instantly alerting users through an iOS application accessible on any device. If users do not respond to a notification within 1 minute, the system automatically calls emergency services to ensure prompt assistance.

With threshold-based alerts and precise location tracking, the system not only warns users when smoke or gas levels surpass safe limits but also pinpoints the exact location of potential hazards. This enables a rapid response to mitigate risks, protect lives, and reduce property damage. Additionally, data logging and historical analysis allow users to review past events and refine safety protocols, making this solution ideal for both residential and commercial environments.

3 Functional need

3.1 Sensors and IoT network

To fulfill the functional needs of our smoke detection system, the following components will be integrated to address the project context, problem statement, and objectives outlined.

- **Smoke and Gas Detection Sensors:** will be utilized to detect the presence of smoke, carbon monoxide, and other hazardous gases in real time, enabling rapid detection of fire-related risks.
- **Environmental Sensors:** Additional sensors such as temperature and humidity sensors can be included to provide context, helping to differentiate smoke from other particles and enhancing the reliability of detections.
- **Microcontroller:** An ESP32 or Raspberry Pi will be employed to process sensor data and manage communication between the sensors and the cloud for real-time alerts and remote monitoring.

3.2 iOS application

The iOS Application will enable users to:

- View real-time sensor data (smoke levels, gas concentration, temperature, etc.) with the option to visualize historical data over customizable time periods.
- Receive alerts based on threshold values for smoke and gas levels, helping users respond promptly to potential fire risks.
- Utilize Location-Based Services (LBS) to send the precise location of detected smoke or gas anomalies.
- Activate an emergency alert system in the event of detected smoke levels exceeding critical thresholds. If users do not respond to a notification within 1 minute, the system will automatically contact emergency services to ensure immediate assistance.

4 Machine learning integration

4.1 Model development

The objective of model development is to construct a machine learning model capable of accurately predicting potential fire risks based on sensor data. The model will utilize specific indicators such as smoke levels, gas concentrations, temperature, and humidity, as well as the location of the detected smoke to provide an information if there is a fire or not .

4.2 MLOps implementation

The MLOps framework will support continuous improvement and operational management of the machine learning model, covering key areas:

1. **Automated model training and deployment:** MLOps tools will automate the training and deployment of the model using real-time environmental and sensor data to provide ongoing risk predictions and updates.
2. **Continuous monitoring:** The framework will continuously monitor the model's performance, track its accuracy, and identify potential drift to maintain high-quality predictions over time.
3. **Alerts and notifications:** When the model detects potential fire hazards, it will trigger automated alerts to users, encouraging prompt action to mitigate risks.
4. **Data versioning and tracking:** MLOps practices will ensure data versioning and tracking of historical environmental and sensor data for future analysis and reference.
5. **Security and compliance:** Robust security measures will protect user data, including encryption, access control, and compliance with privacy regulations.

6. **Model performance optimization:** MLOps tools will aid in optimizing the model, including hyperparameter tuning and resource management, to ensure precise and efficient risk predictions.

7. **Feedback loop:** User feedback will be incorporated to refine model accuracy and enhance its ability to deliver relevant and actionable fire risk insights.

The MLOps implementation is essential to maintaining the accuracy, security, and reliability of the fire risk prediction model, ensuring it adapts to evolving user needs and environmental changes.

5 Equipment

After conducting research, we have identified the following components as necessary for the smoke detection system:

- **Raspberry Pi 4 Model B:** This versatile single-board computer is equipped with multiple general-purpose input/output (GPIO) pins, and it supports various programming languages, such as Python, Java, C, and C++. With built-in dual-band Wi-Fi and Bluetooth, it requires no additional modules, making it ideal for IoT projects. Its enhanced processing power and memory options make it an excellent choice for real-time data processing and sensor integration in applications like smart smoke detection systems.

- **BME680 Environmental Sensor:** The BME680 is an environmental sensor that measures temperature, humidity, pressure, and indoor air quality (IAQ) by detecting volatile organic compounds (VOCs) in the environment. It communicates via I2C or SPI, making it easy to interface with the Raspberry Pi. This sensor is particularly suited for air quality monitoring, helping to detect changes in environmental conditions that may indicate smoke or hazardous gas presence.

- **SGP30 Gas Sensor:** The SGP30 is a digital multi-pixel gas sensor that detects indoor air quality and measures CO_2 and total volatile organic compounds (TVOCs). It also communicates via I2C, enabling straightforward integration with the Raspberry Pi. The SGP30 is ideal for smoke detection and monitoring air pollutants, as it can provide additional data on harmful gases that may signal potential fire hazards.

- **Cables:** To connect the various components of the circuit, multiple wires will be used to ensure stable and reliable data transmission between the sensors and the Raspberry Pi.

6 Technologies choice

6.1 Back-end

- MongoDB: A NoSQL document-oriented database used for storing user data. It is practical and easy to use in conjunction with Jakarta EE.

- MQTT: A lightweight publish-subscribe network protocol used to communicate sensor data to an MQTT broker in the cloud.

6.2 Middleware

- JAX-RS: Java API for RESTful Web Services is a Java programming interface for creating web services with a REST architecture.
- WildFly: WildFly, formerly known as JBoss Application Server or JBoss, is a free and open-source Java EE application server written in Java and released under the GNU LGPL license. It can be used on any operating system that provides a Java virtual machine.
- Mosquitto: Mosquitto is a widely used MQTT broker, serving as an intermediary for efficient and reliable messaging between IoT devices and the back-end system.

6.3 Front-End

- **iOS App with Swift:** A native iOS app built with Swift offers high performance, security, and seamless integration with Apple features. Developed in Xcode, Swift apps are optimized for iPhone/iPad, delivering a smooth, reliable experience tailored to the Apple ecosystem.

6.4 Iot integration

- Flogo: Flogo is an open-source project designed for creating lightweight, event-driven microservices and workflows, making it suitable for orchestrating data flows, data pre-processing, and automation in IoT scenarios.

6.5 MLOPS integration

In our MLOps integration, we harness a suite of technologies to streamline the operational management of machine learning models, ensuring their continuous accuracy and reliability. Key technology components include:

1. CI/CD Pipeline: Our continuous integration and continuous deployment (CI/CD) pipeline, powered by Jenkins, is at the core of our MLOps strategy.
2. Testing technologies: For model validation and data quality assurance, we employ PyTest and Selenium. These testing frameworks aid in verifying the functionality and reliability of our health monitoring system, ensuring that user data is processed accurately.
3. Development technologies: We leverage Python and Jupyter Notebook for model development and experimentation. These technologies provide a versatile and data-centric environment for creating and fine-tuning our machine learning models.

4. Monitoring and logging: Our system relies on Prometheus and Grafana for monitoring and logging. These tools allow us to track the performance of our machine learning models and detect any anomalies, ensuring their continued effectiveness.

5. Data management: MongoDB serves as our database technology, providing a robust and flexible solution for storing and managing user data, health records, and environmental information.

6. Security measures: To safeguard sensitive user data, we implement encryption and access control using HashiCorp Vault.

7. Containerization: Docker containers are used for efficient deployment and scaling, ensuring consistency and ease of management in our system.

8. Stream processing: Apache Kafka is employed to handle real-time data streams from sensors, enabling timely updates and notifications in our health monitoring system.

7 Architecture

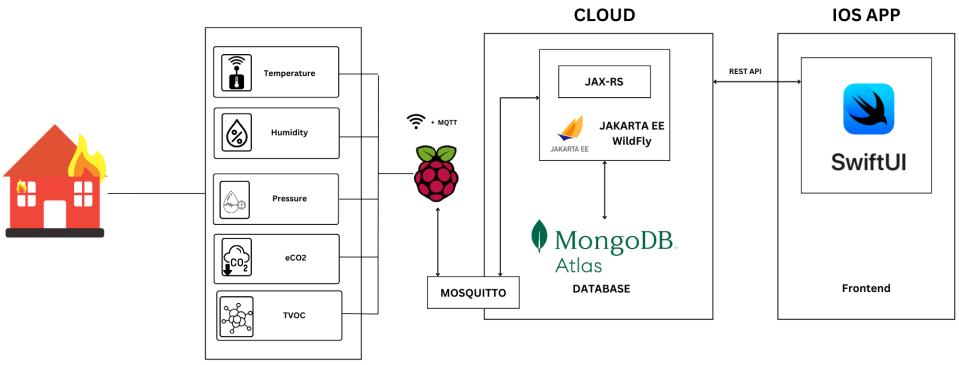

Figure 1: Architecture

The diagram above describes the main architecture of the Health monitoring system which is mainly composed by Backend, Middleware and Frontend.

8 Timeline and tasks

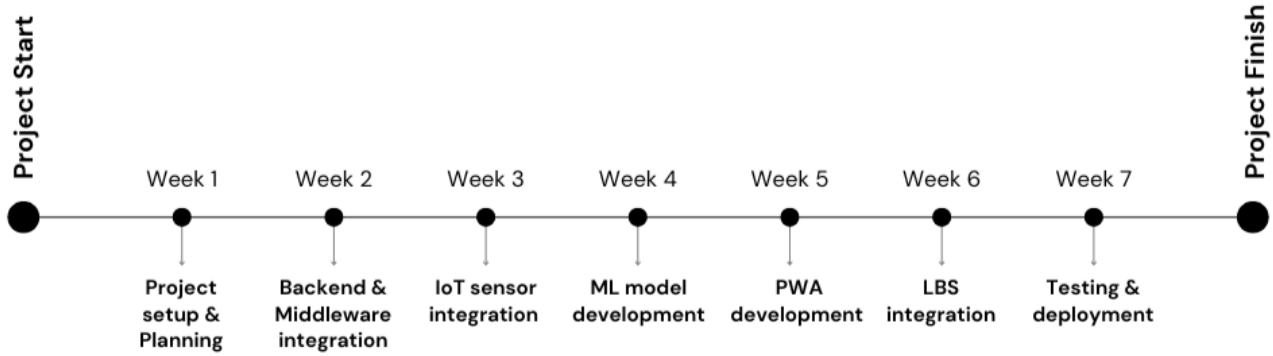

Figure 2: Timeline and Tasks

9 Methodology

Throughout the project, we will be utilizing Extreme Programming (XP), an Agile software development framework renowned for its commitment to delivering superior software quality and enhancing the personal satisfaction of the development team. This methodology is exceptionally well-suited to our project due to its adaptability, specifically tailored for short-duration projects where last-minute requirement changes are common.

9.1 Principles

XP's principles align with those of agile methodologies but are distinguished by their extreme emphasis. XP is founded on:

- High responsiveness to changing customer needs.
- Teamwork, in our context represented by Pair Programming.
- Delivering high-quality work.
- Early, high-quality testing.

9.2 Process

The XP framework normally involves 5 phases or stages of the development process that iterate continuously:

1. Planning: In the initial stage, we create our own user stories and define the desired outcomes. Requirements are provided, and we estimate the stories, creating a release plan that breaks the project into iterations to cover the required functionality incrementally. If certain stories cannot be accurately estimated, we introduce "spikes," indicating that further research is necessary.

2. Design: While designing is an integral part of the planning process, it is essential enough to be highlighted separately. It is closely tied to one of the core XP values we will discuss later – simplicity. A well-thought-out design brings logical structure to the system, helping us avoid unnecessary complexities and redundancies.

3. Coding: This phase involves the actual implementation of code, and it follows specific XP practices, including adherence to coding standards, pair programming, continuous integration, and collective code ownership.

4. Testing: Testing is at the heart of Extreme Programming. It is an ongoing activity encompassing both unit tests, which are automated tests that verify the proper functioning of individual features, and acceptance tests, where customers assess whether the system aligns with the initial requirements.

5. Listening: Listening emphasizes continuous communication and feedback. Coaches and project managers play a pivotal role in conveying the business logic and expected value to the team, ensuring a shared understanding of the project's goals and requirements.

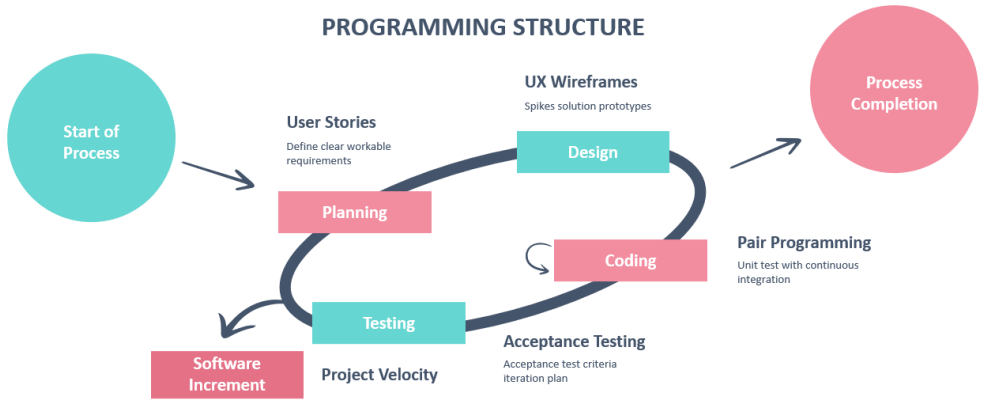

Figure 3: Extreme programming life-cycle

10 Limitations

- Internet Connectivity Interruptions: One limitation lies in the reliance on internet connectivity. In cases of internet outages, real-time monitoring and user alerts for critical health changes may not be achievable.
- Sensor Data Accuracy: There is a risk of false alarms due to inaccuracies in sensor data or anomalies detected by the smoke detection algorithms. This could lead to unnecessary concerns and alerts for users, potentially diminishing trust in the system's reliability.
- Proximity of IoT Devices: The accuracy of smoke detection may be affected by the proximity of IoT devices to each other. If devices are placed too close together, they might register interference from neighboring sensors, impacting the precision of smoke detection.
- Sensor Placement and Thresholds: Proper placement of smoke sensors is critical for effective operation. Inaccurate sensor placement or poorly defined thresholds for smoke detection can result in false readings and unnecessary alerts, complicating the monitoring process.
- Data Volume and Processing: Managing the volume of data generated by multiple smoke detectors can be challenging. Efficient processing and analysis of this data are essential for providing timely alerts and maintaining system performance.

11 Business study

The Business Study section provides an overall view of our project's business model and marketing policy.

11.1 Business Model Canvas (BMC)

The Business Model Canvas serves as a visual representation of our project's main aspects, such as value proposition, customer segmentation, channels, cost structure, revenue stream, and more. The BMC is depicted in the figure below:

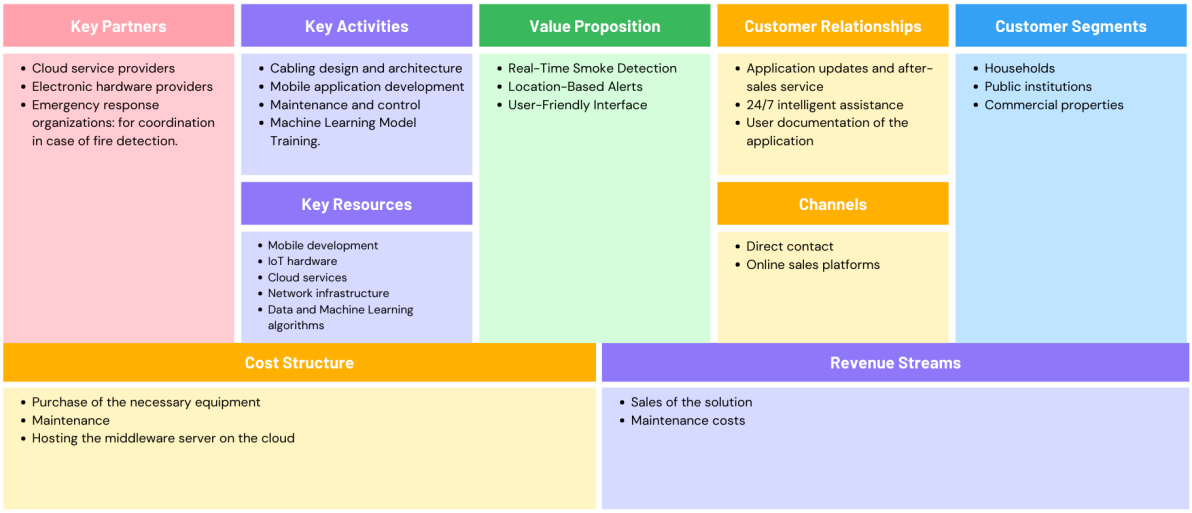

Figure 4: Business Model Canvas

11.2 Marketing policy

Our marketing policy is focused on providing a reliable, high-value smoke detection solution to our customers while establishing a strong presence in the market. It encompasses the following aspects:

11.2.1 Product

- A smart smoke detection system equipped with BME680 and SGP30 sensors for accurate real-time air quality and smoke detection.
- A user-friendly Progressive Web Application (PWA) that provides immediate alerts, tracks smoke levels, and pinpoints the location of potential fire hazards.
- Continuous algorithm enhancement using MLOps for high-accuracy predictions and low false alarms.

11.2.2 Price

- Competitive pricing with flexible packages for homeowners, businesses, and facility managers.
- Discounts for bulk installations and long-term service agreements, along with warranties and comprehensive after-sales support.

11.2.3 Promotion

- A multi-channel promotion strategy, including digital marketing, social media outreach, and participation in safety and smart home technology trade shows.
- Free trials and informational webinars to educate customers on the benefits and applications of the smart smoke detection system.

11.2.4 Place

- Accessible via a web and mobile Progressive Web Application (PWA), making it convenient for users to monitor air quality and receive real-time alerts on any device.

12 Deliverables

- **Conceptual Document:** A comprehensive document detailing the system's specifications, features, and services to be provided.
- **Source Code:** The source code of the various project components will be available on GitHub for transparency and collaboration.
- **Prototype/Simulation:** A functional prototype or simulation of the smart smoke detection system.
- **Demonstration Video:** A video (in mp4 format) demonstrating the proposed solution and showcasing key features and functionalities.