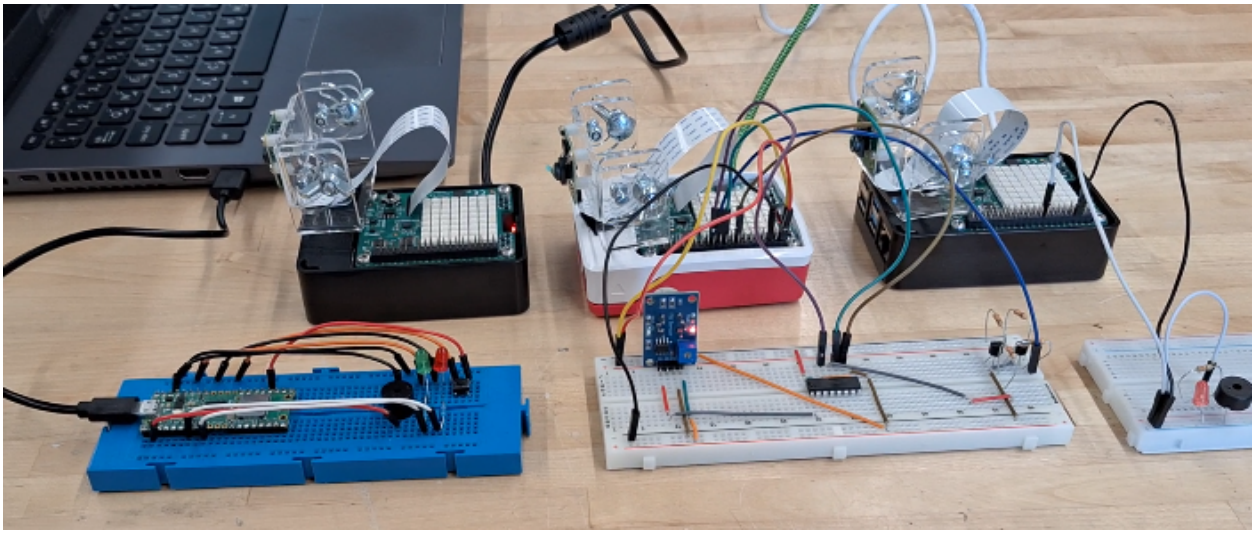
SYSC3010 Computer Systems Development Project
FANS Final Report

Group L1-G8



[1]

Grant Achuzia, 101222695
Javeria Sohail, 101197163
Matteo Golin, 101220709
Saja Fawagreh, 101217326
TA: Sean Kirkby

Created: April 3rd, 2024
Modified: April 7, 2024

Contents

1 Project Description

1.1 Motivation

The motivation for the FANS project was to address the critical problem of preventable fire-related deaths in Canada. Fire safety was a fundamental concern for individuals, families, and communities nationwide. By leveraging modern technological advancements, the FANS system aimed to significantly enhance the effectiveness of fire alarm systems, reducing response times, and ultimately saving lives. The importance of this project could not be overstated, as it directly impacted the safety and well-being of Canadians.

1.2 Problem Statement

The need for the Fire Alarm Notification System (FANS) stemmed from the alarming number of preventable fire deaths in Canada, where 220 people died in fires each year, with at least one in seven of these deaths occurring in homes without working smoke alarms [2]. This critical problem underscored the shortcomings of traditional fire alarm systems, which often lacked advanced communication capabilities and real-time monitoring capabilities [3]. These limitations not only contributed to delayed response times but also to preventable deaths, underscoring the urgent need for an advanced fire alarm solution.

Traditional systems' shortcomings, coupled with the fact that smoke detection systems were sometimes unsafely disarmed by users to avoid false alarms—especially those installed close to kitchen spaces—further exacerbated the problem. The FANS project sought to address these issues by integrating smoke and temperature sensors with Internet of Things (IoT) technology. This approach not only aimed to cover scenarios where smoke may not reach the detecting device but also offered real-time notifications via SMS and email, thus notifying homeowners immediately in the event of an emergency.

By providing configurable thresholds for smoke and temperature alarms and adjustable timeouts that prevented the system from being deactivated in an unsafe manner, FANS aimed to use technological advances to significantly improve the effectiveness of fire detection systems. The ultimate goal of the project was to reduce response times, improve overall fire safety, and thereby mitigate fire disasters and protect the well-being of individuals, families, and communities across Canada [4].

1.3 Overview of Solution

In our project, we implemented the Fire Alarm Notification System (FANS), a real-time system designed for efficient fire detection and alerting. The system comprised several key components, seamlessly integrated to ensure a comprehensive approach to fire safety. We outlined the system's architecture in a detailed deployment diagram, illustrating the interconnections between the various components.

The Alarm System, based on a Raspberry Pi 4, triggered audible and visual alerts promptly in the event of a fire, ensuring occupants were promptly warned. Our Smoke Detection System, also utilizing a Raspberry Pi 4, continuously monitored the environment for smoke using advanced sensors. It served as the primary detection mechanism, activating the alarm and notification systems upon detecting smoke.

Our Notification System, operating on a Raspberry Pi 4, promptly sent out emergency alerts to

predefined contacts, including local authorities and individuals, ensuring rapid response to the detected fire. We employed a real-time cloud database to store critical data, facilitating remote monitoring and configuration, thus ensuring optimal system functionality.

Additionally, we utilized a Pi Pico Device to provide haptic feedback, offering an additional layer of alert through physical sensation, ensuring even those outside the hearing range of the alarm or with hearing impairments were alerted.

Our design emphasized scalability, enabling easy expansion with additional sensors or alarm units as needed. Each component functioned independently, ensuring reliability and ease of maintenance. Communication between components was facilitated through a local network, with the cloud database supporting remote access and control. We incorporated advanced communication protocols for efficient data exchange and alerting. The smoke detection system employed the I2C protocol for sensor communication, while UDP packets facilitated local network communication between the system's nodes. HTTP requests were utilized for interactions with the cloud-hosted real-time Firebase database, ensuring timely updates and access to system data.

2 Final Solution

The final solution for FANS is displayed in Figure 1. This solution uses a distributed systems approach to create a multi-node fire response system controlled via a web interface.

2.1 Deployment Diagram

The primary node in the FANS system is the sensor data collection node, which monitors temperature and smoke concentration levels. All sensor measurements are written to the Firebase cloud database for display by the web UI.

When a measurement is detected above the user-defined threshold, the sensor data collection node sends a UDP message signifying an emergency over the local network to the notification system and alarm system nodes. Upon receipt of this message, the notification system will send emails and SMS notifications to all users in its local database. The alarm system node will begin blinking an LED and sounding an alarm.

In addition to the UDP message, the sensor data collection node will raise an emergency flag in the Firebase database which signifies to the UI and the wearable haptic alarm that an emergency is active. The haptic alarm wearable device will then begin blinking an LED and buzzing until the user disables it or the emergency ends.

The UI provides an interface for checking on the status of an emergency and reviewing sensor data in real-time. Users can disable the emergency response from the UI when the emergency has been resolved, and they can also time out the system's emergency response in scenarios where the system may be falsely triggered (smokey cooking). Finally, the UI also provides an interface for user contact information to be added to the subscriber list for emergency notifications.

2.2 Message Protocol Table

In the Fire Alarm Notification System (FANS), a variety of communication protocols are meticulously integrated to ensure seamless interaction among the system components and with the external cloud database, facilitating a robust and responsive fire alarm solution.

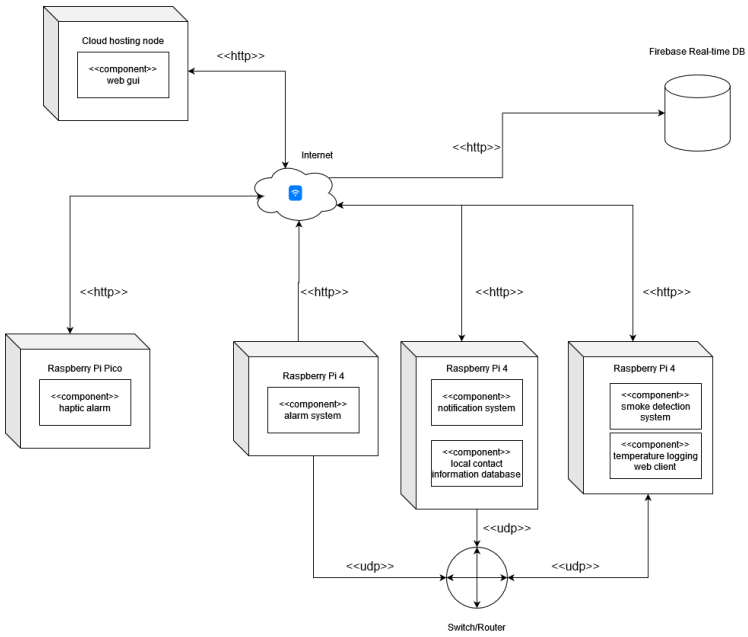


Figure 1: The FANS deployment diagram.

Sender	Receiver	Message	Data Format	Protocol
Raspberry Pi 4	Temperature Sensor	`read_temp`	See section 6.2.1 of datasheet [5]	I2C
Raspberry Pi 4	Smoke Sensor (via ADC)	`read_smoke`	See figure 1.1 of datasheet [6]	SPI [6]

Table 1: Messages for I2C communication in FANS.

The system's core, the Smoke Detection System, communicates with its temperature and smoke sensors using the I2C and SPI protocols over the GPIO pins of a Raspberry Pi 4.

2.2.1 I2C & SPI Communication

The Raspberry Pi 4 utilizes the I2C and SPI protocol to communicate with temperature and smoke sensors, monitoring environmental conditions to detect potential fire hazards. These messages are outlined in 1.

2.2.2 Local Area Network Communication

Nodes within the FANS (smoke detection, alarm, and notification systems) communicate over a local network using UDP packets, facilitating real-time alerts and system coordination. The messages sent over UDP use numerical value to encode messages. The representation agreed upon is shown in Table 2. The messages sent are in Table 3

Message	Value
Emergency	0
No Emergency	1

Table 2: Numerical representation of messages over UDP in FANS.

Sender	Receiver	Message	Data Format	Protocol
Smoke detection system	Notification system	Emergency	0	UDP
Smoke detection system	Alarm system	Emergency	0	UDP
Smoke detection system	Notification system	No emergency	1	UDP
Smoke detection system	Alarm system	No Emergency	1	UDP

Table 3: Messages for local area network communication in FANS.

Sender	Receiver	Message	Data Format	Protocol
Smoke Detection	Cloud DB	`put_sensor_data()`	See listing 1	HTTP (JSON)
GUI	Cloud DB	`update_threshold(new_threshold)`	See listing 2	HTTP (JSON)
Notification	Cloud DB	`query_contact_information()`	See listing 3	HTTP (JSON)
Haptic Alarm	Cloud DB	`emergency()`	See listing 4	HTTP (JSON)
GUI	Cloud DB	`get_sensor_data()`	See listing 5	HTTP (JSON)

Table 4: Messages for cloud database communication in FANS.

2.2.3 Cloud Database Communication

Messages between each node and the Firebase database are shown in Table 4.

Listing 1: Update sensor data message.

```
{
  "method": "PUT",
  "path": "/sensor-data/temperature",
  "headers": {
    "Authorization": "Bearer YOUR_ACCESS_TOKEN",
    "Content-Type": "application/json"
  },
  "body": {
      "data": {
          "temperature": 21.2,
          "timestamp": "2024-03-13T08:37:22"
      }
  }
}
```

Listing 2: Threshold update message.

```
{
  "method": "PUT",
  "path": "/system/threshold",
  "headers": {
    "Authorization": "Bearer YOUR_ACCESS_TOKEN",
    "Content-Type": "application/json"
  },
  "body": {
    "newThreshold": 50
  }
}
```

Listing 3: Request for user contact information.

```
{
  "method": "GET",
  "path": "/user/contact",
  "headers": {
    "Authorization": "Bearer YOUR_ACCESS_TOKEN"
  }
}
```

Listing 4: Request for emergency flag.

```
{
  "method": "GET",
```

```
  "path": "/system/emergency",
  "headers": {
    "Authorization": "Bearer YOUR_ACCESS_TOKEN"
  }
}
```

Listing 5: Request for latest sensor data.

```
{
  "method": "GET",
  "path": "/sensor-data",
  "headers": {
    "Authorization": "Bearer YOUR_ACCESS_TOKEN"
  }
}
```

2.2.4 User Notification Communication

The notification system communicates with users through email, employing standard internet protocols to ensure timely and effective alerts.

Sender	Receiver	Message	Data Format	Protocol
Notification System	User Inbox	Emergency notification	See listing 6	SMTP (Email)

Table 5: Messages for user notification communication in FANS.

Listing 6: Email notification for detected emergency in FANS.

```
FROM: notification@example.com
TO: user@example.com
SUBJECT: Fire Alarm Notification

Dear [User's Name],

This is an emergency notification. Please exit the building.

Emergency detected: [Date, Time]

Stay safe!
```

2.3 Sequence Diagrams

The FANS system will satisfy six core use cases highlighted and illustrated in the message sequence diagrams below.

2.3.1 Trigger Emergency Use Case

The trigger emergency use case shown in Figure 2 represents the critical functionality of the FANS, where the system detects a potential fire through its smoke detection system and initiates a series of automated responses to mitigate the situation. As depicted in the message sequence diagram, the process begins when the smoke detection system identifies smoke and potentially high temperatures indicative of a fire. This detection triggers the system to send a notification to the alarm system and the notification system. The alarm system responds by sounding an audible and visible alert to notify occupants of the building immediately. Concurrently, the notification system sends out emergency alerts via SMS and email to all registered users, ensuring they are informed of the danger regardless of their current location. This response ensures that all the users are promptly alerted to the emergency, resulting in a quick evacuation and response to the detected threat.

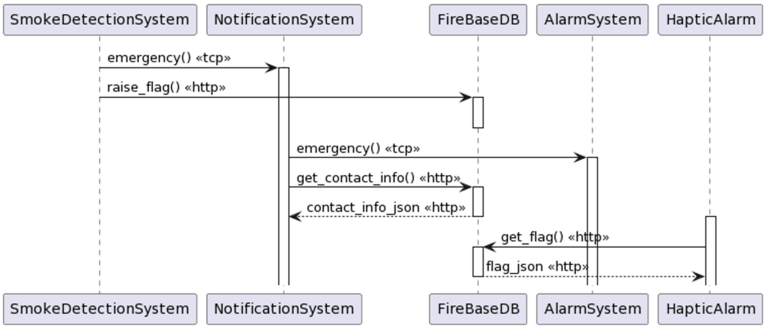


Figure 2: The sequence diagram for the trigger emergency use case.

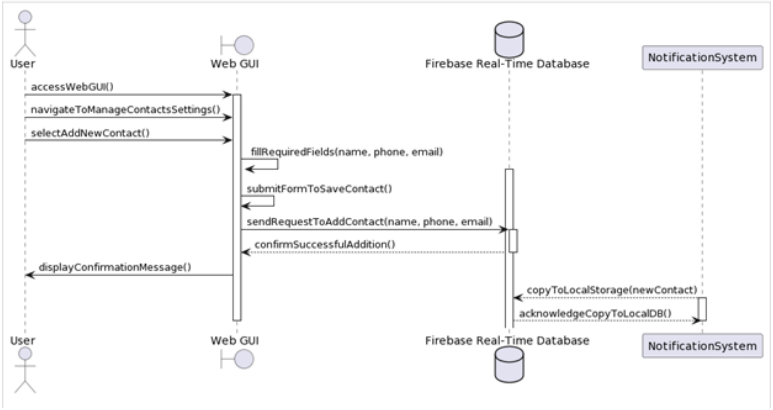


Figure 3: The sequence diagram for the adding new contact information use case.

2.3.2 Add New Contact Information Use Case

This use case shown in Figure 3 outlined the process by which users added new contact information to the FANS database. Through the web GUI, users entered new contact details such as phone numbers and email addresses, enabling the system to send emergency notifications. Upon submission, these details were successfully updated in the real-time database. The message sequence diagram for this use case (not shown) illustrated the interactions between the user, the web GUI, and the database, culminating in the notification system being updated with the new contact information. This ensured that the FANS could reach the user through various channels in the event of an emergency, thereby enhancing the system's effectiveness in communicating critical alerts.

2.3.3 Change Emergency Threshold Use Case

Adjusting the smoke detection threshold was an essential feature that allowed users to customize the sensitivity of the FANS based on environmental conditions and personal preferences. This use case shown in 4 involved a user accessing the web GUI to modify the threshold settings determining when the smoke detection system should trigger an alert. The message sequence diagram (not shown) visualized the steps involved, from the user's interaction with the GUI to update the threshold to the real-time database's role in storing this new setting. The smoke detection system then retrieved and applied the updated threshold, ensuring that the FANS operated according to the user's specifications, balancing sensitivity and minimizing false alarms.

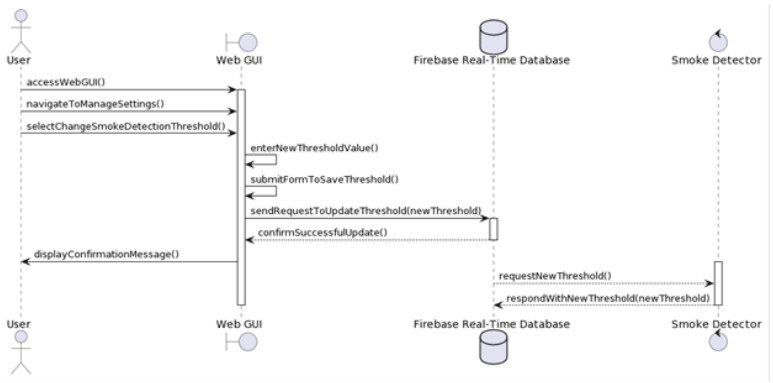


Figure 4: The sequence diagram for the changing emergency threshold use case.

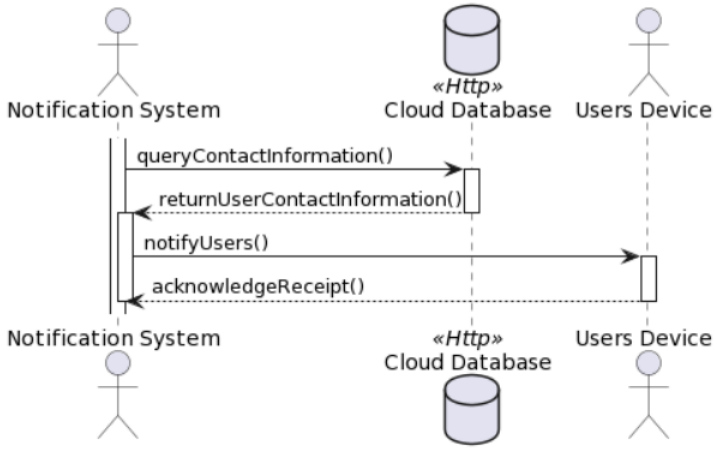


Figure 5: The sequence diagram for the emergency response use case.

2.3.4 Emergency Response Use Case

The scenario in Figure 5 highlights the FANS's capability to notify users in the event of a detected fire. Upon detecting a fire, the notification system successfully queried the cloud database for contact information and then sent out alerts via SMS and email to all users. The message sequence diagram for this use case (provided above) captured the sequence of these interactions, showcasing how the system ensured that occupants were informed and ready to address the emergency promptly.

3 Discussion of Final Design

Project FANS (Fire Alarm Notification System) was created to address the urgent issue of preventable fire deaths in Canada by utilizing modern technologies to improve the effectiveness of fire alarm systems. Our final design successfully integrates smoke and temperature sensors with Internet of Things (IoT) technology, providing real-time notifications and improving fire safety measures for individuals and communities across Canada.

Throughout the development of FANS, our team adhered to the specifications and objectives outlined in our initial proposal and detailed design document. Our goal was to develop a system that

would not only detect fire based on smoke and temperature changes but also immediately notify people of emergencies through a comprehensive and responsive communication system.

1. Complete implementation: We are proud that our project was realized completely and without omissions. Every component, from the smoke detection system to the emergency call system, was implemented as planned. Thank you to our commitment to rigorous planning and effective teamwork, we were able to overcome challenges and ensure that all project objectives were met.

2. Consistency of protocols: There was no deviation from the originally proposed communication protocols. The use of the I2C protocol for sensor communication, UDP for local network communication and HTTP requests for interaction with the Firebase real-time database were maintained throughout the project. This consistency in the choice of protocols ensured seamless integration of the system components and reliable performance of the FANS.

Our project not only met the expected design specifications, but also performed excellently in the test phases. Here are some important results and data from our tests:

- Sensitivity of the sensors: the smoke and temperature sensors used in the FANS showed high sensitivity and accuracy during testing. They were able to detect slight changes in environmental conditions, ensuring early detection of potential fire hazards. The responsiveness of the sensors is crucial to the effectiveness of the system, as they issue warnings in good time and enable rapid action to be taken.

- Efficiency of the notification system: The notification system was tested to ensure that it could immediately trigger an alarm when a fire was detected. The test results showed that the SMS and email notifications were sent without significant delay, usually within seconds of the emergency being detected. This rapid response is crucial for the safety of residents and minimizing potential damage.

- Scalability of the system: A key aspect of our design was its scalability, allowing easy integration of additional sensors or alarm units. Tests confirmed that adding new components did not affect the performance of the system, proving the flexibility and adaptability of the design to different environments.

- GUI functionality: The Fire Alarm Notification System (FANS) graphical user interface provides an interactive platform for real-time monitoring and control, enabling direct alarm management and seamless integration with the system components and real-time database, significantly improving responsiveness and ease of use.

The fire detection system represents a significant advance in fire protection, effectively meeting the need for an integrated, responsive and reliable fire detection and notification system. Through careful planning, unwavering commitment to our project goals and meticulous adherence to our design, we have developed a system that will significantly impact the well-being of Canadians and provide a new level of protection from fire hazards. The successful completion of this project demonstrates not only our team's technical capabilities but also our commitment to making a meaningful contribution to public safety.

4 Contributions

This section lists all of the contributions of each member on the FANS development team.

4.1 Code Contributions

Code file contributions listed in this table will use Unix file paths to describe which files were worked on by each contributor. The asterisk signifies the wildcard operator, so `alarm-system/*` captures all files within the alarm-system/ directory. All paths correspond to paths within the project repository on GitHub.

For a more granular view of contributions, where multiple authors have worked on the same file, please review the Git commit history.

Author	Code files
Grant Achuzia	• `docs/*` • `web-app/*` • `pico_alarm/*` • `README.md`
Saja Fawagreh	• `docs/*` • `notifier/*` • `notifier/tests/test_email_notification_system.py` • `README.md`
Javeria Sohail	• `docs/*` • `pico_alarm/*` • `haptic_alarm_test/*` • `README.md`
Matteo Golin	• `alarm-system/*` • `sensor-pi/*` • `web-app/main.py` • `web-app/templates/index.html` • `web-app/templates/settings.html` • `docs/*` • `notifier/templates/*` • `notifier/tests/test_email.py` • `notifier/messages.py` • `README.md`

Table 6: The individual code contributions of each FANS team member.

4.2 Report Contributions

Author	Report sections
Grant Achuzia	Project Success, Learning Experiences, Appendix A
Saja Fawagreh	Message Protocols, Sequence Diagrams, Discussion of Final Design
Javeria Sohail	Motivation, Problem Statement, Final Design Solution
Matteo Golin	Contributions, Deployment Diagram, Project Extensions

Table 7: The individual final report contributions of each FANS team member.

References

[1] P. Matoušek, *Thick smoke on black background*. [Online]. Available: https://www.freeimages.com/photo/thick-smoke-on-black-background-1633270 (visited on 02/11/2024).

[2] (Oct. 7, 2022), [Online]. Available: https://www150.statcan.gc.ca/n1/pub/11-627-m/11-627-m2022035-eng.htm (visited on 02/12/2024).

[3] A. Erickson. (Sep. 15, 2023), [Online]. Available: https://www.digitize-inc.com/blog/fire-alarm-whats-new-2023.php (visited on 02/12/2024).

[4] "Smoke alarms: A sound you can live with." (Sep. 29, 2020), [Online]. Available: https://www.getprepared.gc.ca/cnt/rsrcs/sfttps/tp201011-en.aspx (visited on 02/11/2024).

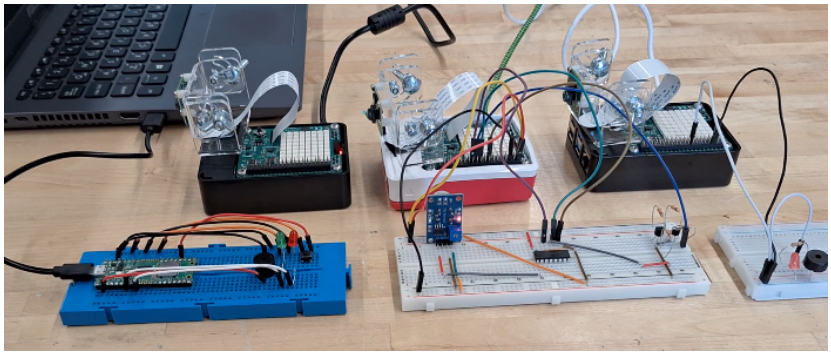
[5] "Mems pressure sensor: 260-1260 hpa absolute digital output barometer." (), [Online]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiGq4Dy4_GEAxX1HNAFHSf-BRwQFnoECBcQAQ&url=https%3A%2F%2Fwww.st.com%2Fresource%2Fen%2Fdatasheet%2Flps25hb.pdf&usg=AOvVaw2FTksZHmMhimTAl6Qq3eFF&opi=89978449 (visited on 03/13/2024).

[6] "Mcp3004/3008." (), [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP3004-MCP3008-Data-Sheet-DS20001295.pdf (visited on 03/13/2024).

A GitHub Repository README

The following content is the README file taken from the FANS GitHub repository.

FANS (Fire Alarm Notification System)



FANS is a comprehensive multi-system solution for fire-related emergencies that require prompt notification of persons in the affected area.

FANS utilizes smoke and temperature sensors to accurately detect fires in its surrounding environment. Upon detection, FANS issues SMS and email notifications to individuals in the surrounding area.

FANS provides a dashboard for live monitoring of its environment, as well as an interface for customizing things like its sensitivity to pressure changes.

Table Of Contents

Contributors

SYSC3010 group L1-G8, under guidance of TA Sean Kirkby.

- Grant Achuzia
- Javeria Sohail
- Matteo Golin
- Saja Fawagreh

Repository Organization

FANS is composed of several different nodes. Within the project repository, each node is given its own directory containing its required files.

- The data collection system is stored in `sensor-pi/`
- The notification system is stored in `notifier/`
- The web GUI is stored in `web-app/`
- The alarm system is stored in `alarm-system/`
- The haptic alarm system is stored in `pico_alarm/`

In addition, the `docs/` directory contains documentation on our system design. Within `docs/`, the `assets/` folder contains all of the UML class, sequence, state machine and deployment diagrams which describe the FANS system and which are used throughout our reports. The `design-report/` folder contains all of the LaTeX source files that compose the FANS design report, and the `proposal/` folder contains the LaTeX source files for building our initial project proposal.

Installation and Setup Guide

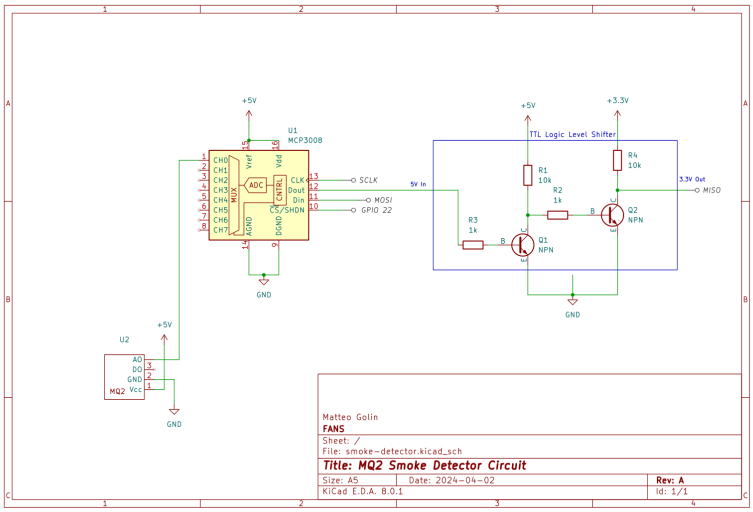
Guides to install and run each system node are provided below. Once all nodes have been set up, the system will be functional.

Sensor-Pi

In order to set up the `sensor-pi` sensor data collection system, you will need:

- Raspberry Pi 4
- Sense hat
- Breadboard
- Wires
- 2x 10k resistors
- 2x 1k resistors
- 2x NPN bipolar-junction transistors
- Flying fish MQ2 smoke detector module

In order to assemble the circuit, place the components on your breadboard according to the following schematic. Power sources can come from the Raspberry Pi or an external source. The labelled pin outputs (GPIO 22, SCLK, MOSI, MISO) are all meant to be connected to the corresponding pin on the Raspberry Pi 4. Please see its pinout sheet to locate this pins.



Ensure that the following files are present in the same directory for it to function properly:

- `firebase_config.json`: This file should include the configuration details required to connect to the Firebase database, such as the API key, authentication domain, database URL, and storage bucket.

Example `firebase_config.json` file:

```
{
  "apiKey": "yourKey",
  "authDomain": "yourDomain",
  "databaseURL": "https://your-database-url",
  "storageBucket": "someStorageBucket"
}
```

Once the circuit is built and connected, the node can be started by running the software located in the `sensor-pi` directory from the Raspberry Pi. Python 3.11 must be previously installed. The following commands will download the repository and start the software when run in the terminal:

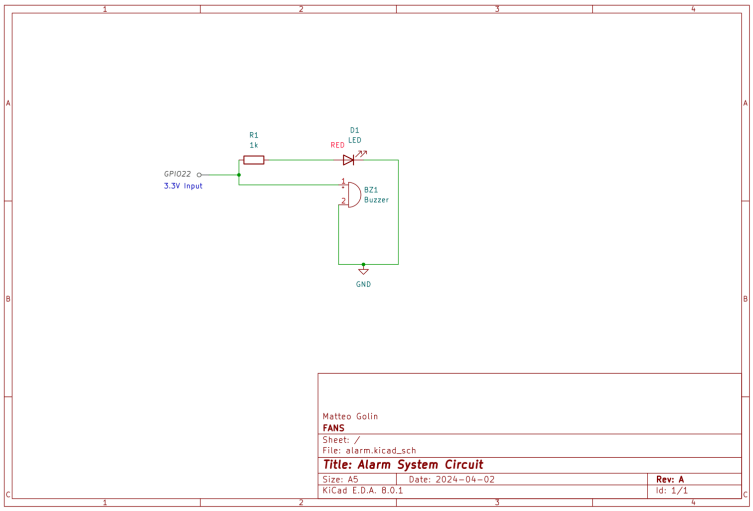
```
git clone https://github.com/SYSC3010-W24/sysc3010-project-l1-g8.git
cd sysc-project-l1-g8/sensor-pi
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
python3 main.py
```

Alarm System

In order to set up the `alarm-system` node, you will need:

- Raspberry Pi 4
- Breadboard
- Wires
- Passive piezoelectric buzzer
- Red LED
- 1k resistor

In order to assemble the circuit, place the components on your breadboard according to the following schematic. The 3.3V input signal should be connected to the Raspberry Pi's GPIO 22 pin. Please see the Pi's pinout sheet to connect the pins properly.



Once the circuit is assembled and connected, the node can be started by running the software located in the `alarm-system` directory from the Raspberry Pi. Python 3.11 must be previously installed. The following commands will download the repository and start the software when run in the terminal:

```
git clone https://github.com/SYSC3010-W24/sysc3010-project-l1-g8.git
cd sysc-project-l1-g8/alarm-system
python3 -m venv venv
source venv/bin/activate
```

```
pip install -r requirements.txt
python3 main.py
```

Notifier

In order to set up the `notifier` node, you will need:

- Raspberry Pi 4

The node can be started by running the software located in the notifier directory on the Raspberry Pi. Prior installation of Python 3.11 is required. Execute the following commands in the terminal to download the repository and start the software:

```
git clone https://github.com/SYSC3010-W24/sysc3010-project-l1-g8.git
cd sysc-project-l1-g8/notifier
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
python3 email_notification_system.py
```

Additionally, ensure that the following files are present in the same directory for it to function properly:

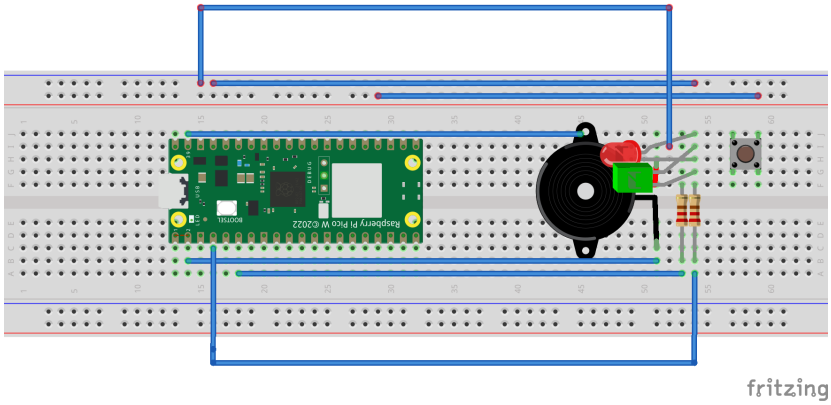
- `fans_credentials.json`: This file should contain the username and password required for authentication to the email server, enabling the system to send notifications via email.
- `firebase_config.json`: This file should include the configuration details required to connect to the Firebase database, such as the API key, authentication domain, database URL, and storage bucket.
- `twilio_credentials.json`: This file should contain the authentication credentials for accessing the Twilio API, including the account SID and authentication token, enabling the system to send notifications via SMS.

Pico-Alarm

In order to set up the `pico_alarm` buzzer alarm and led notifier system, you will need:

- Raspberry Pi Pico W
- Piezoelectric Buzzer
- 1 red and 1 green LED
- Breadboard
- Wires
- 2x 10k resistors

Hardware Assembly



1. **Connect the Buzzer** to GPIO 1 on the Pico W.
2. **Attach the Red LED** to GPIO 6 with a 10k resistor in series.
3. **Attach the Green LED** to GPIO 5 with a 10k resistor in series.
4. **Set up the Button** on GPIO 18, ensuring it's properly debounced with a 10k resistor.
5. **Wire everything** according to the schematic on a breadboard, ensuring secure connections.

Software Installation and Deployment for Pico_Alarm System

1. **Connect the Raspberry Pi Pico W** to your computer. Use a USB cable to establish the connection. Ensure the Pico is in MicroPython mode.
2. **Open Thonny IDE** on your computer. It's recommended to use Thonny for working with MicroPython on Raspberry Pi Pico due to its built-in support.
3. **Prepare the Python Script**: Have the `pico_alarm.py` script ready. This script includes the logic for monitoring the Firebase database and activating the buzzer and LEDs.
4. **Configure the Script**:
   - **WiFi Credentials**: Within the script, locate the `do_connect()` function. Replace `"your_wifi_ssid"` and `"your_wifi_password"` with your actual WiFi network SSID and password.
   - **Firebase URL**: Find the `firebase_url` variable in the script. Change its value to your Firebase database URL where the emergency flag will be checked.
5. **Deploy the Script to Pico W**:
   - In Thonny, select MicroPython (Raspberry Pi Pico) as the interpreter from the bottom right corner.
   - Open the `pico_alarm.py` script in Thonny.
   - Click on 'File' > 'Save As...' and choose to save the script on your Raspberry Pi Pico.
6. **Run the Script**:
   - With the `pico_alarm.py` script open in Thonny, press the green 'Run' button to execute the script on the Pico W.
   - The script will automatically start monitoring the Firebase database for any emergency flags and activate the buzzer and LEDs accordingly.

Note: The Raspberry Pi Pico W must be powered continuously for the alarm system to function. It can remain powered via the USB connection to your computer or through an external 5V power source.

Operational Notes

- The system checks the Firebase database at regular intervals for any emergency flags.
- The buzzer and red LED activate to indicate an emergency, with the green LED indicating normal operations.
- The button serves as an acknowledgment mechanism to stop the alarm and reset the system to its normal state.

Web App

In order to set up the `web-app` node, you will need:

- Access to the internet

FANS web application is made using Flask, an easy to use Python micro framework. The `static/` folder contains css files for styling the app as well as a JS script for dynamically changing between themes. The `templates/` folder contains html files, and the `tests/` folder contains tests for making sure the application gets data from firebase.

Ensure that the following files are present in the same directory for it to function properly:

- firebase_config.json: This file should include the configuration details required to connect to the Firebase database, such as the API key, authentication domain, database URL, and storage bucket. Example firebase_config.json file:

```
{
  "apiKey": "yourKey",
  "authDomain": "yourDomain",
  "databaseURL": "https://your-database-url",
  "storageBucket": "someStorageBucket"
}
```

To gain access to the website, run the following commands in your terminal:

```
git clone https://github.com/SYSC3010-W24/sysc3010-project-l1-g8.git
cd sysc-project-l1-g8/web-app
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
python3 main.py
```

In the terminal, click on the second http link to access the web app on a network server. Here's an example of what the second link may look like:
`Running on http://192.168.X.X:5000/`


Home Settings Logout
Welcome Grant Achuzia
Grant's Fire Alarm
Clear
Grant's Pager
Connected
Timeout System
3600
System timeout in seconds (maximum 4 hours/7200 seconds).
SUBMIT TIMEOUT
Temperature
Current Temperature: 38.13 °C
Temperature of system environment
Temperature
Temperature (C)
Time
Smoke
Current Smoke Level: 173.86 PPM
Smoke concentration of system environment
Smoke Concentration
Smoke (PPM)
Time