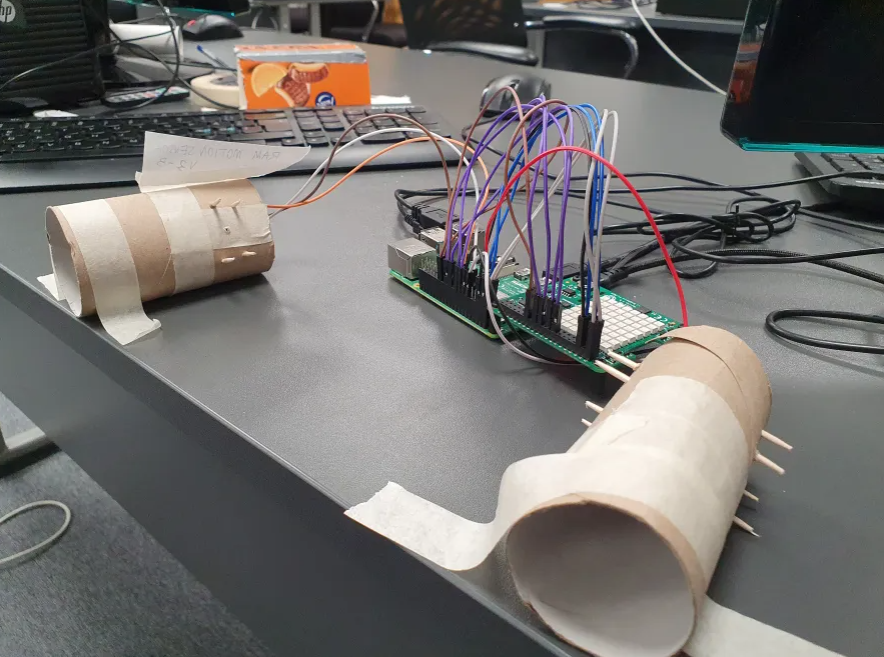
RAM Motion Sensor



Riku Sänkiaho, Anthony Haapaniemi ja Mika Rasinkangas

Table of Contents

[1. General overview 3](#_Toc154057083)

[2. Functionality 4](#_Toc154057084)

[2.0 Mode selection 4](#_Toc154057085)

[2.1 Mode 1 – Room booking 4](#_Toc154057086)

[2.2 Mode 2 – Motion alarm 4](#_Toc154057087)

[2.3 Mode 3 – Monitoring system 5](#_Toc154057088)

[3. Diagrams 6](#_Toc154057089)

[4.Implementation 8](#_Toc154057090)

[4.1 Materials 8](#_Toc154057091)

[4.2 Hardware Setup 8](#_Toc154057092)

[4.3 Software and development 9](#_Toc154057093)

[4.4 Database 9](#_Toc154057094)

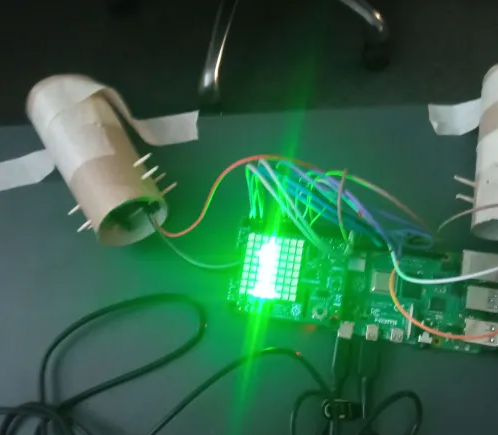
[5. Further development ideas 9](#_Toc154057095)

[6. Credits / Sources 9](#_Toc154057096)

# 1. General overview

The RAM Motion Sensor is a setup consisting of two PIR motion sensors and a Raspberry Pi with the Sensehat extension, with three modes for different applications of the sensors. The sensors are meant to be attached to both side of a doorway, one outside and one inside a room.

The first (and most extensive) mode allows users to reserve a room for a specific amount of time, access it with a username/password combination and then monitors the amount of people within. Reservation is lost at the end of the time limit or if the room is empty for a specified amount of time.



The second mode is a simple motion sensor alarm that can be set to be active for a set time and requires a password to be entered if the alarm is triggered.

*RAM motion sensor in mode 1. The screen shows there’s one person inside the room.*

The third mode is an inverse of the previous one that sets off an alarm if no movement is detected for a long time.

All modes are capable of logging the date and time of all motion activity into an SQL database, with basic user administration for safety.

The purpose behind this project was for our group to learn the basics of programming with embedded systems, gain further coding experience and add another project to our resume. This project was part of Opiframe’s Embedded systems class in the fall of 2023.

Made in Python on a Raspberry Pi 4.

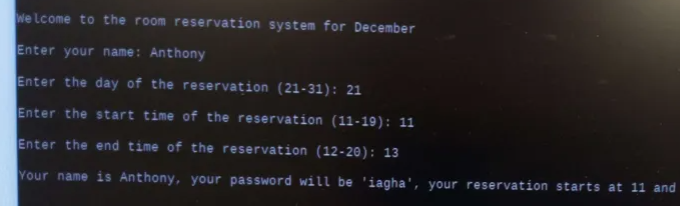
# 2. Functionality

## 2.0 Mode selection

The system is controlled via the terminal on the Raspberry Pi. On startup the user is presented with four options on the terminal (the three modes and exit). To choose one the corresponding number must be typed in.

## 2.1 Mode 1 – Room booking

Initially the user is prompted to enter their name, day and start/end hours for their reservation. A random five-letter password is automatically generated and displayed to the user for later identification.



*The terminal in mode 1. The randomly generated password consists of five random letters.*

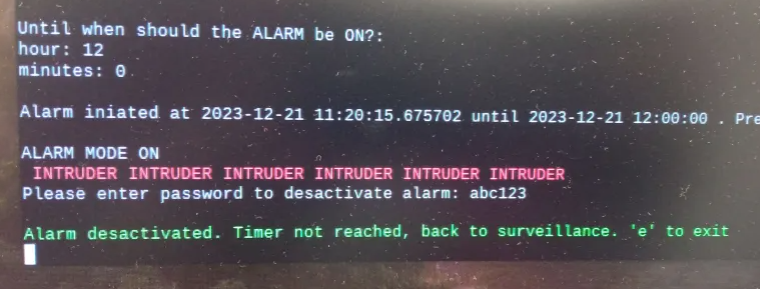
When the reserved time slot activates the outer sensor will wait for activity. Upon detecting movement, the user will be prompted to type in their username and password, and if typed correctly, the reservation will become active and room usage will be monitored from this point onwards.

The motion sensors keep track of movement in and out of the room and display the current amount of room occupants on the Sensehat display. Should the room become empty the reservation will only stay active if someone re-enters the room within a set amount of time.

At the end of the reservation time or if the room is idle for too long the program automatically shuts down the current reservation and allows a new reservation to be made.

## 2.2 Mode 2 – Motion alarm

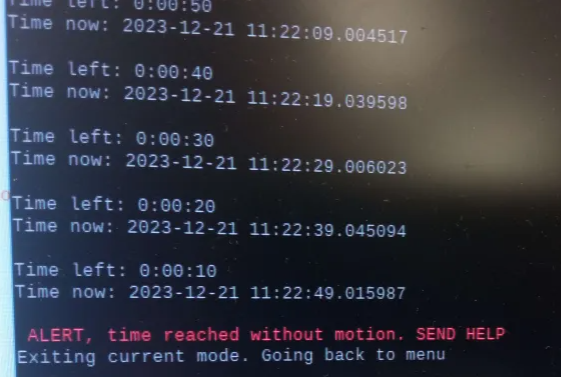
The user will be asked to type in the hour and hour the alarm will deactivate, after which it automatically becomes active for the duration. Any motion detected by the sensors will set off the alarm, which can only be turned off by entering the password. The alert mode itself can also be turned off by inputting the password at any point.



*Terminal in mode 2. The alarm has been triggered and deactivated.*

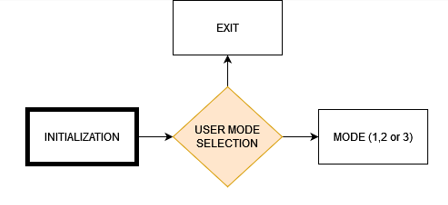
## 2.3 Mode 3 – Monitoring system

This mode is an inverted version of the motion alarm that activates immediately as soon as the mode is selected, and a timer will start counting down. The sensors will constantly scan for activity and upon detection will reset the timer to 24 hours (by default), effectively monitoring that there’s activity at least once a day in its radius. If the timer reached zero an alarm is triggered.

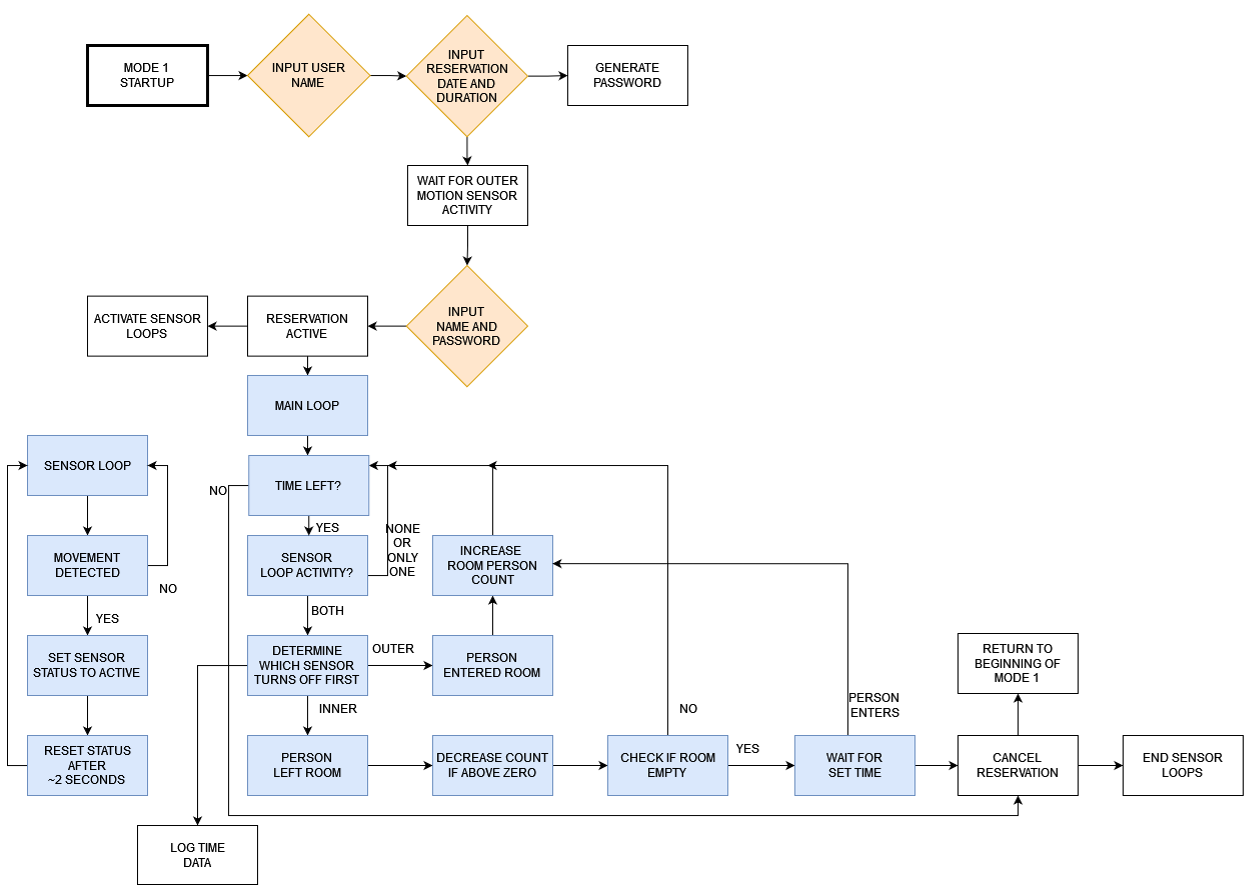


*Terminal in mode 3 after time has run out.*

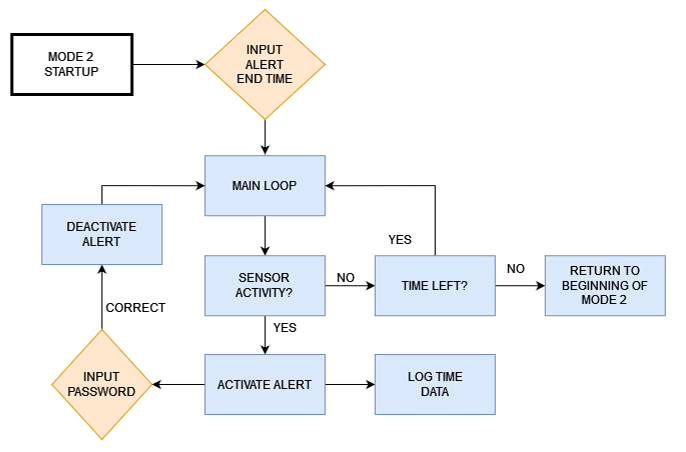
# 3. Diagrams



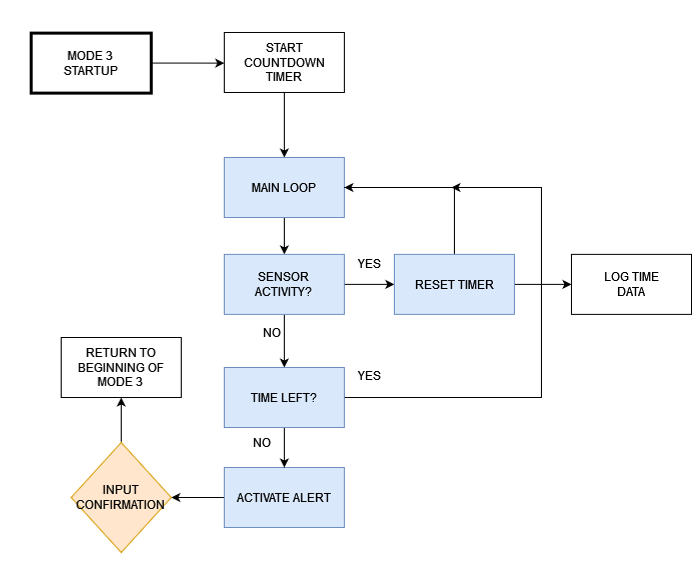
*Mode selection diagram.*



*Mode 1 diagram.*



*Mode 2 diagram.*



*Mode 3 diagram.*

# 4.Implementation

## 4.1 Materials

* Raspberry 4 model B
* Sensehat
* 2 x PIR motion sensor
* 11 x female to male jumper cables (for the Sensehat connection)
* 6 x female to female jumper cables (three cables per motion sensor)
* A display monitor is required to use the terminal.

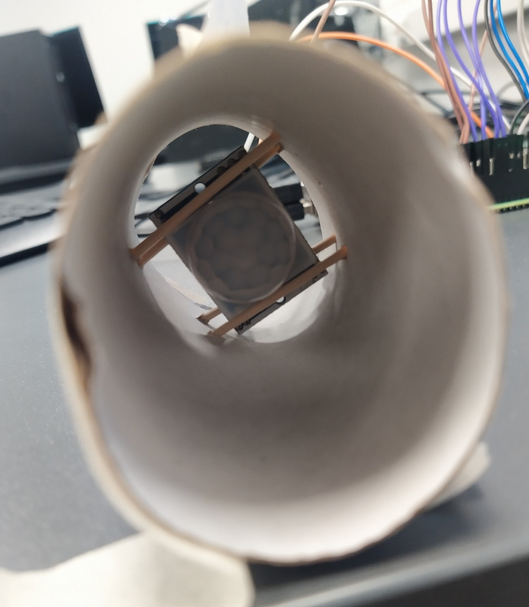
## 4.2 Hardware Setup

The base of the project is the Raspberry Pi. A Sensehat attachment is connected with pin cables instead of simply putting it on top of the Raspberry PI in order leave room for other attachments, which includes two PIR motion sensors.

The Sensehat requires a total of eleven pins to connect to the Raspberry Pi properly.

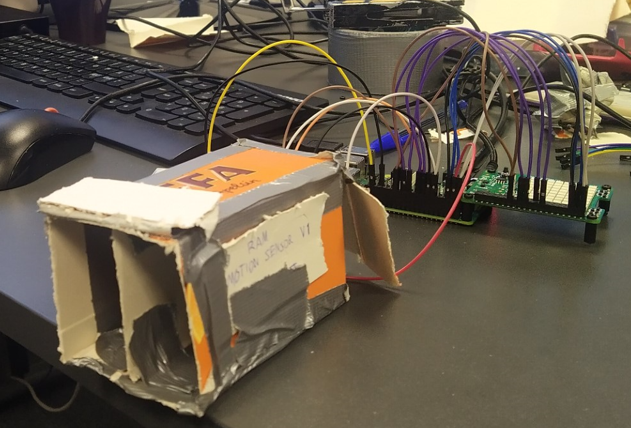
The motions sensors require three pins each. Due to lack of available power outputs, one sensor is attached to 5V while the other uses 3.3V power.

The motion sensors available scanning area must be limited into small cones, and they must be stuck in place next to each other, about 25 cm / 10 inches apart. For this configuration two empty toilet rolls were taped onto a table and the motion sensors were fastened in place inside the using four toothpicks as pictured below.



*The PIR motion sensor kept in place by toothpicks.*

## 



*The early design used cardboard “lanes” to focus the motion sensor area.*

## 4.3 Software and development

Python was chosen as our primary programming language and the bulk of the coding was done on the Raspberry Pi itself using Geany. Git was set up along with SSH access, but remote work was unfeasible for the most part due to the Raspberry being confined to the classroom and constant testing being required when adjusting motion sensor functionality.

## 4.4 Database

Sqlite was chosen as the database for this project along with a graphic UI made with Python and the Tkinter-library. Reservation information is stored in the table along with timestamps. The UI includes a basic login system with a username and password for security.

# 5. Further development ideas

- Make the device and database available remotely via the internet, via a web app or similar.

- Improve the visual design of the device.

- Extend the reach of the sensors from the Raspberry Pi or find wireless alternatives.

- Add new modes.

- Optimize and clean up code.

# 6. Credits

Riku Sänkiaho — python programming (mode selection, mode 1, motion sensor functionality, debugging), hardware setup, documentation.

Anthony Haapaniemi — python programming (mode selection, all modes, debugging), Github management, hardware setup.

Mika Rasinkangas — python programming (mode 1, debugging), sqlite database, UI development.

Github: https://github.com/AntHaa78/Projekti