Emergency Info Hub

User Usability/Videos

Osama Abou Hajar - C00220135.
4th Year Software Eng.

Supervisor Paul Barry

INSTITUTE of TECHNOLOGY CARLOW
Institiúid Teicneolaíochta Cheatharlach

Abstract

Emergency Info Hub (EIH), Is a central website that helps the emergency services to prepare for, respond to & recover from disaster, by providing all needed data for the targeted building (E.g. Number of people, area size and emergency exits).

The main objective of this project is giving the number of trapped people under rubbles or inside a building, by tracking their number using a simple movement sensor fitted on the main gate and face detection technology and save this number to the cloud to be used when a disaster happens

This document is the user manual provides system requirements and installation instructions. It also contains the code listings.

Table of Contents

User Guidance Videos:

TO easy use and setup instructions, YouTube videos playlist has been created to cover all the project functions, code, and setup instructions.

The list of modules covered in the playlist:

- 01 - INTRO TO EIH - EVERY LIFE MATTERS (video)
- 02 - How to calibrate EIH Hardware (video)
- 03 How TO Setup the hardware environment, EIH (video)
- 04 EIH Front-end Functionality Overview (video)
- 05 Front-end Framework Structure and Python AnyWhere (video)
- 06 - How to change Address AutoComplete country restrictions (video)
- 07 - how to add new Admin and how the login function works (video)
- 08 How to add new Building and how to establish the connection with the hardware (video)

The link to the full Playlist including all videos:

https://www.youtube.com/playlist?list=PL4amH-KHpsvIujZYx7ztRXOgMtDIQeFQZ

Also, you can always refer to the user manual on GitHub markdown and the technical manual PDF within the project documentation.

- **Project main Readme file**
  - https://github.com/OAbouHajar/projectEIH/blob/master/README.md
- **Project front-end instructions Readme file**
  - https://github.com/OAbouHajar/projectEIH/blob/master/eih-front-end-webapp/README.md
- **Project back-end instructions Readme file**
  - https://github.com/OAbouHajar/projectEIH/blob/master/eih-raspberrypi-body-detect/README.md

The user manuals mentioned below are the same one on GitHub.

EIH: Emergency Info Hub

The project link: https://www.e-hub.ie/

What is EIH?

Emergency Info Hub (EIH), is a fourth-year student project, has been designed to be a central website helps the emergency services to prepare for, respond to & recover from disaster, by providing all needed data for the targeted building (E.g. Number of people, area size and emergency exits). The main objective of this project is giving the number of trapped people under rubbles or inside a building, by tracking their number using a simple movement sensor fitted on the main gate and face detection technology and save this number to the cloud to be used when a disaster happens.

See the documentation for more details.

EIH - Structure:

The structure of the EIH GitHub repository is:

```
.
├── eih-front-end-webapp/     # The Fron-end folder and files which create the website to output the data.
├── eih-raspberrypi-body-detect/    # The Back-end folder and files where all the hardware work takeing place.
├── LICENSE                         # EIH LICENSE agreement for any use of the project.
└── README.md
```

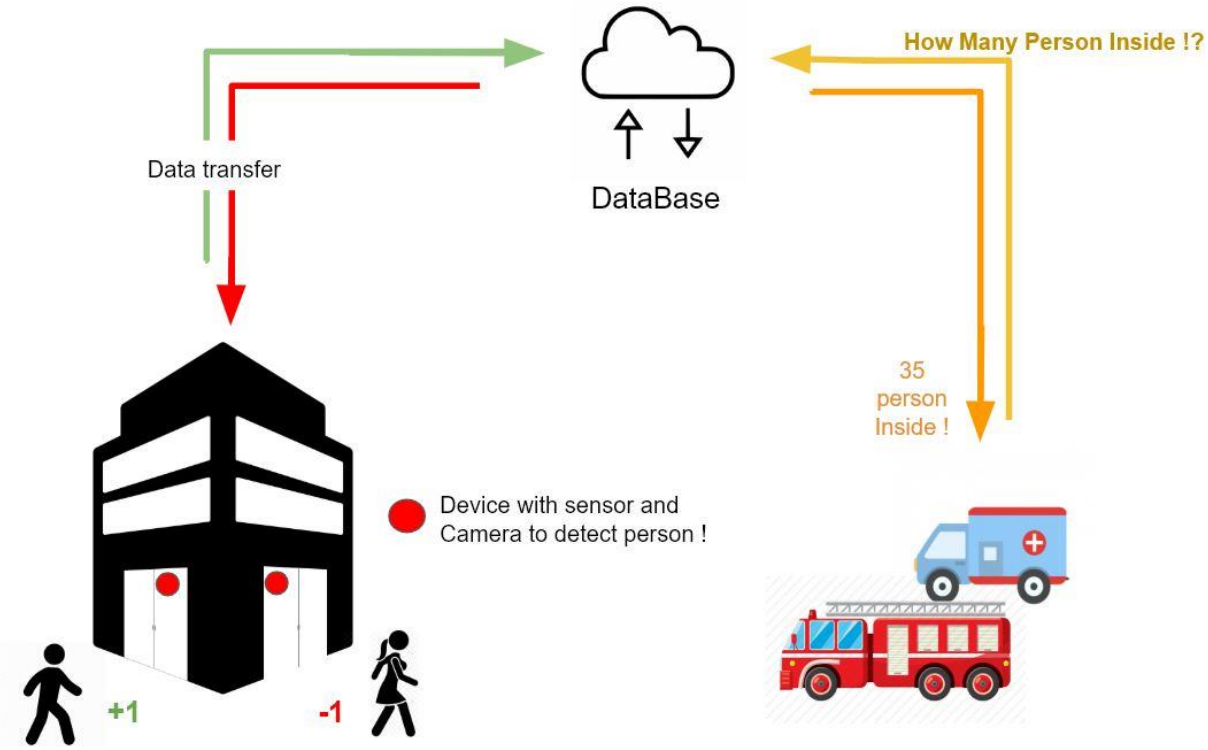
EIH Requirements

This project consists of two parts:

- Hardware: on this link (Hardware)
- Website: on this link (Website)

to run EIH you have to follow the inaction of the REAMDME.MD files for the Front-End and the Backend.

How EIH works?



It is shown in the diagram above:

- EIH briefly works as the following steps:
  - When Someone approaches the door, the sensors work, and the camera takes a picture of the people waking IN Or OUT
  - The Picture will be processed by a body-detect algorithm, to detect how many people in the picture.
  - The total number of people inside the building will be saved on the could (GOOGLE FIREBASE).
  - The Emergency Services will be able to reach this detail by the address search at any time.

EIH Story

The story of EIH started THE day when this photo was taken, back in 2012 during the Syrian war.

The story of EIH started at that day when this photo was taken (the website background), back to 2012 during the Syrian war,
when I was among those guys running to help the people stuck under rubbles. during this moments the one and the only question you will hear is: **ANYBODY HERE?**

The hours, the minutes and even the Seconds might be a reason to give someone a new life. calling again and again... **ANYBODY HERE?**, feeling bad if any one left behind with no help.

ANYBODY HERE? it's the reason to start, and here where we are: EIH - "EVERY LIFE MATTERS".

EIH - Front-End:

EIH front-end build using (FLASK) python framework, Jinja2 HTML, CSS, JavaScript web design languages.

The structure of the front-end is:

```
.
├── configuration/       # Configure the parameters and initial settings for some computer programs
├── static/              # Static web pages are HTML documents stored as files in the file system
├── templates/           # All the HTML files are stored inheriting the base, html page
├── app.py               # The main webApp python file where all python code and functions
├── .env                 # All the environment variables
├── requirements.txt     # EIH environment requirements.
├── test_flask.py        # Unit test, to make sure the DB connection is established and the webApp load successfully.
└── README.md            #
```

app.py

-- This is the main file for the webApp, all the function and the variables are getting processed, to run the file you need to have the following dependencies installed on your machine or the server-side for online deployment.

To run the app you need to have Python3 installed and flask, in addition to the (request), (firebase), (python-dotenv), (pyrebase) dependencies as the following:

```
curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
python get-pip.py
python3 get-pip.py
pip3 install flask
pip install requests
pip install python-firebase
pip3 install firebase
pip install python-dotenv
pip3 install pyrebase
```

This dependency will be responsible for all the processes and the connections between the Database and the WebApp.

.env File

-- The .env file contains all the environment variables, where all the Tokens, API_Keys, and other types of credentials can be stored as follows:

```
    $ export GOOGLE_API_KEY='GOOGLE-API-KEY-FOR-THE-CREATED-PROJECT-ON-THE-FIREBASE'
    $ export GOOGLE_MAP_API='GOOGLE-MAPS-API-KEY-TO-SHOW-THE-MAP-ON-RESULTS'
    $ export AUTH_DOMAIN='<YOU-PROJECT-NAME-ON-FIREBASE>.firebaseio.com'
    $ export DB_URL='https://<YOU-PROJECT-NAME-ON-FIREBASE>.firebaseio.com'
    $ export STOREG_BUKET='<YOU-PROJECT-NAME-ON-FIREBASE>.appspot.com'
    $ export SERVICE_ACCOUNT='<THE-PATH-TO-THE-DB-CONFIG-JSON-FILE>/projecteih-firebase-adminsdk-dmd9b-dfbc30ba25.json'
    $ export DB_FILE_URL='https://<YOU-PROJECT-NAME-ON-FIREBASE>.firebaseio.com/<THE-JSON-FILE-NAME-ON-FIREBASE>.json'
```

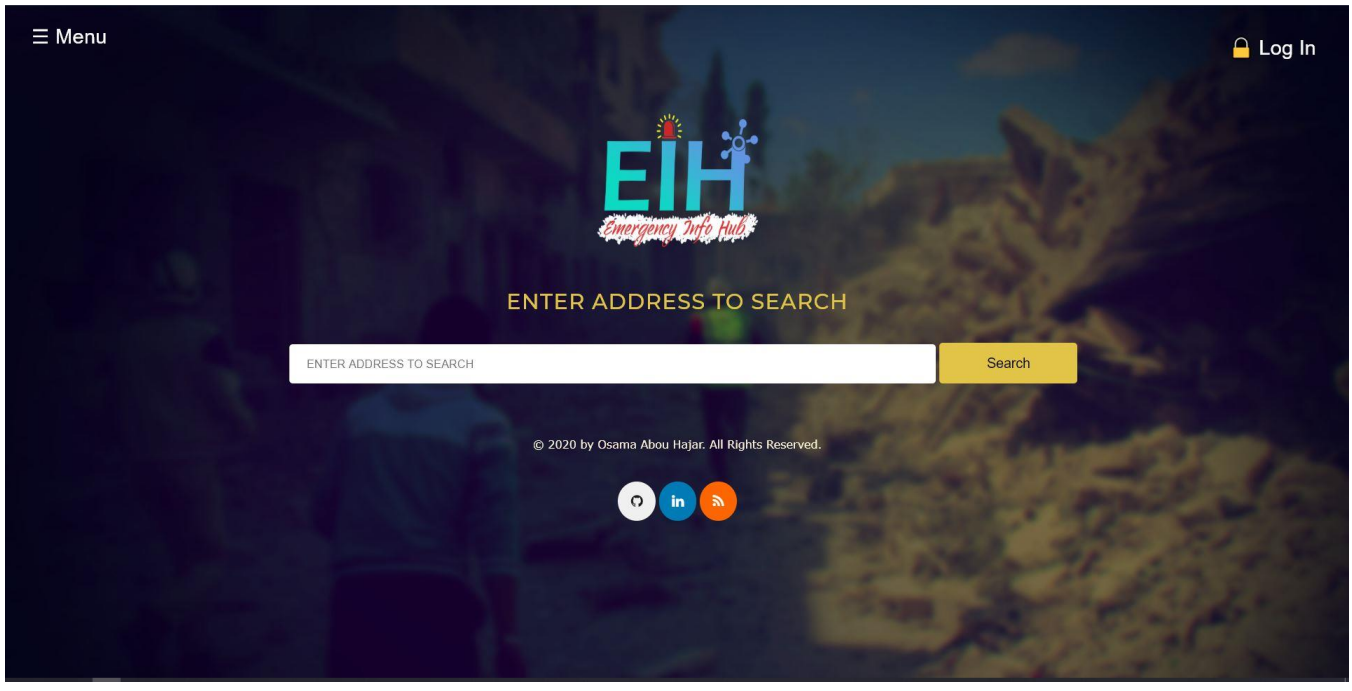
For more details about Google firebase and Google Address autocomplete and how to get this API_KEYS and Tokens, You have to register and read Google documentations

- (Goolge Firebase)
- (Goolge Maps API)

EIH - Screens & Code Functionality:

Search Screen:

It is the main screen of the project, it contains several components and functions to run.



1. **The Search input:** The search box has an integrated Address autocomplete, which creates compatibility to have one search mechanism. In this screen, the user can search for any Irish addresses The search box uses the Google Autocomplete API to give the user the ability to get the correct address and save their time during emergency moments. To be able to list the address from other countries than Ireland, the country code within the restrictions attribute in the files below should

- `templates/googlAutoCompleteGeneral.html`
- `templates/googlAutoComplete.html`

```
// Change the country code from 'IE' to the country where is EIH used
// to be able to list al locations with in this country
autocomplete.setComponentRestrictions({
    'country': 'IE'
});
```

For more information about the (Google Addresses Autocomplete) and the (countries code).

2. **The Search Button:** The search button will send a `POST` request to the `app.py` file to run the `display_form():` function. This function will be calling other functions to get the data and check from the database and check if the address is already registered and active or not.

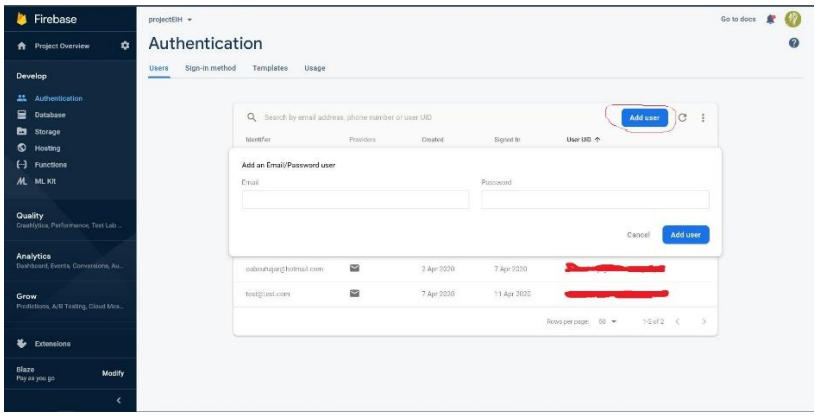
- if the functions return `True` the `retult.html` page will be displayed with all the data coming from the Database, in addition to Google Map box using the `maps2.html` file.

- if the functions return `False` the app will be redirected to the main page with alert no data for the searched address founded displayed.

Add Admin Screen:

The registration from the front-end is not an option, to gain access to the website the administration office, should grant you with the login credentials, that by registering your email and password to the user's authentications table in the firebase API side

as the following picture:



Locations Screen:

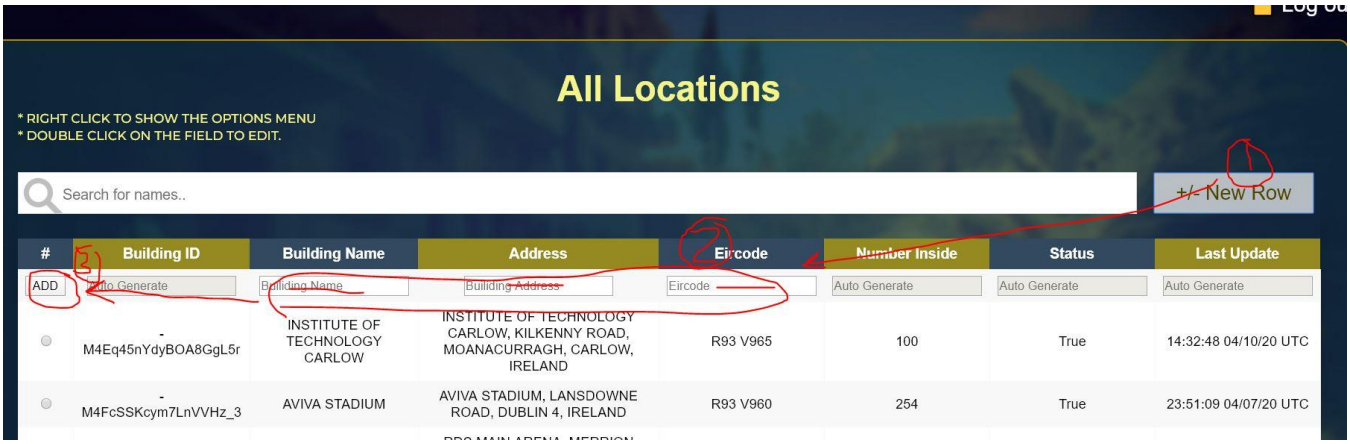
This screen will display all the locations within the databases and give a fully dynamic table to the admins to add, delete, update, and reset the number of people for each row.

- Public access all locations page:

- Admits access all locations page with the menu options on the right mouse click:

Add Building Feature

This is one of the most important parts to link the hardware to the database:
To add building, you have to login as Admin (If you have no Access you should contact the admin body create a credential to gain access). Then you follow the steps in the picture:



- Once you have registration completed the System will show you the Building registration ID, This ID should be used on the hardware side to link the backend to this actual building, as it shows in the picture.

if the Alert! The massage was telling the building is already in the system, that means the building should be activated again and the building ID is already assigned.

Update Building Feature

On the All locations page, the table supports the edit mode, double click on the row you want to edit and you can edit one row at the time, then you sleeted the radio bottom for the edited row and select the update option from the menu.

Reset Building Feature

To Reset a building, select the building row, and select the reset option from the table menu. The number of people inside the selected building will be reset to zero.

Delete Building Feature

The Delete option from the table menu will update the `active` value in the database to `False`.

```
{"active": False}
```

Activate Building Feature

The Activate option from the table menu will update the `active` value in the database to `True`.

```
{"active": True}
```

About Us Screen

The project description and the story of the project.

Case Study Screen

Redirect to an external link to show the case study for EIH as a final year project.

Contact Screen

A contact form to allow the users to contact the admins to give feedback or to report a problem.

EIH - API and Database Structure:

EIH data stored as an API on Google firebase real time database. The data are formatted as JSON with dictionary key and value.

```
data =
{
   '<BUILDING-ID>':
    {
        'active': '<BOOLEAN-VALUE>'
        "deviceId": '<STRING-SET-BY-THE-USERS-FROM-THE-BACK-END>'
        "known_name": '<THE-BUILDING-KNOWN-NAME>',
        "address": '<THE-ADDRESS-GENERATED-BY-GOOGLE-ADDRESS-AUTOCOMPLETE>',
        "eircode": '<THE-BUILDING-EIRCODE>',
        "numberOfPeopleINDetect": '<INTEGER-INITIALIZED-TO-ZERO>',
        "timeUpdated": '<AUTO-GENERATED-BY-THE-SERVER-TIME>'
    }
}
```

- **BUILDING-ID**: is auto-generated by the firebase with the push request, so when the user adds new building data the key will be generated and associated with this data.
- **ACTIVE**: Boolean data type, True to active and in work building, False for non-active or deleted address.
- DEVICE-ID: String data type, created by the user to differentiate between the hardware.
- **KNOWN NAME** is the known and public name for the address, which makes a better understanding by the reader than the proper address.
- **ADDRESS**: String datatype, the address will be autocompleted by the google autocomplete function, the address is considered the unique value to each address where the search will be depending on.
- **EIRCODE**: for the time of this project was done, the Eircode was not used by all the address in Ireland, which is hard to be used for the search and identity.
- **NUMBER-OF-PEOPLE-DETECTED**: this is the value where the number of people gets stored and updated, the initial value is zero when the building gets added.
- **TIME-UPDATED**: is the time when any changes happening to the data, this value will be assigned by the server time.

EIH - Flask Unit Test

- The file `test_flask.py` has been designed to ensure the connections to the database (firebase) has been established, and the webapp pages load successfully.

Run The Test file:

To run the test file you need to run the following command.

```
$ python3 test_flask.py -v
```

The test result will be all the tests passes.

In the case of implementing a new function to the App, you must write a new test match the structure of the tests.
For more details, you can referee to the comment within each file, or contact the developer by email.

EIH - Back-End

The structure of the back-end is:

```
.
├── cred/                   # Configure the parameters and initial settings for some computer programs
├── draft/                  # some files
├── images/                 # where the image got save to be processes by the body detect algorithm
├── detect.py               # the body detect algorithm
├── eihIR.py                # the main file for EIH back-end project
├── get-pip.py              # to install get
├── post-data-to-api.py     # The file post the data to the clould after the number get detected
├── requirments.txt         # all the for this project
├── setup.py                # to setup the project enviroument
└── README.md               #
```

Software Requirements - Installation:

To run the hardware part of EIH: 1- You need Python 3.5 or later to run mypy. You can have multiple Python versions (2.x and 3.x) installed on the same system without problems.

In Ubuntu, Mint, and Debian you can install Python 3 like this:

```
$ sudo apt-get install python3 python3-pip
```

For other Linux flavors, macOS and Windows, packages are available at

http://www.python.org/getit/

-- The file setup.py is made too easy to install all the requirements if you have Python 3 and pip3 installed on your device already, you can use the following command to set up the EIH project in full.

```
$ python setup.py
```

file setup.py

Once your setup is completed you will have the following dependencies installed on your system.

- **Dependencies**:
  - Requests
  - python-firebase
  - python-opencv
  - python-scipy
  - ipython
  - firebase
  - imutils

- python_jwt
- gcloud
- sseclient
- parse
- requests_toolbelt
- pyrebase

Export the Environment Variables

- once your setup.py done you have to set your environment variables to gain the access to your database credentials:

```
 $ export FIREBASE_DB_URL='https:<PROJECT-NAME>.firebaseio.com/<JSON-FILE-NAME>.json'
$ export REG_BUILIDING_ID='<THE-KEY-YOU-GET-AFTER-ADDING-THE-BUILDING-TO-THE-SYSTEM-FROM-THE-FRONT-END>'
$ export DEVICE_ID='<THE-DEVICE-ID-YOU-CHOSE>'
$ export PROJECT_API_KEY='<GOOGLE-FIREBASE-PRJECT-API-KEY>'
```

Hardware Requirements

You need the list of hardware parts to be connected:

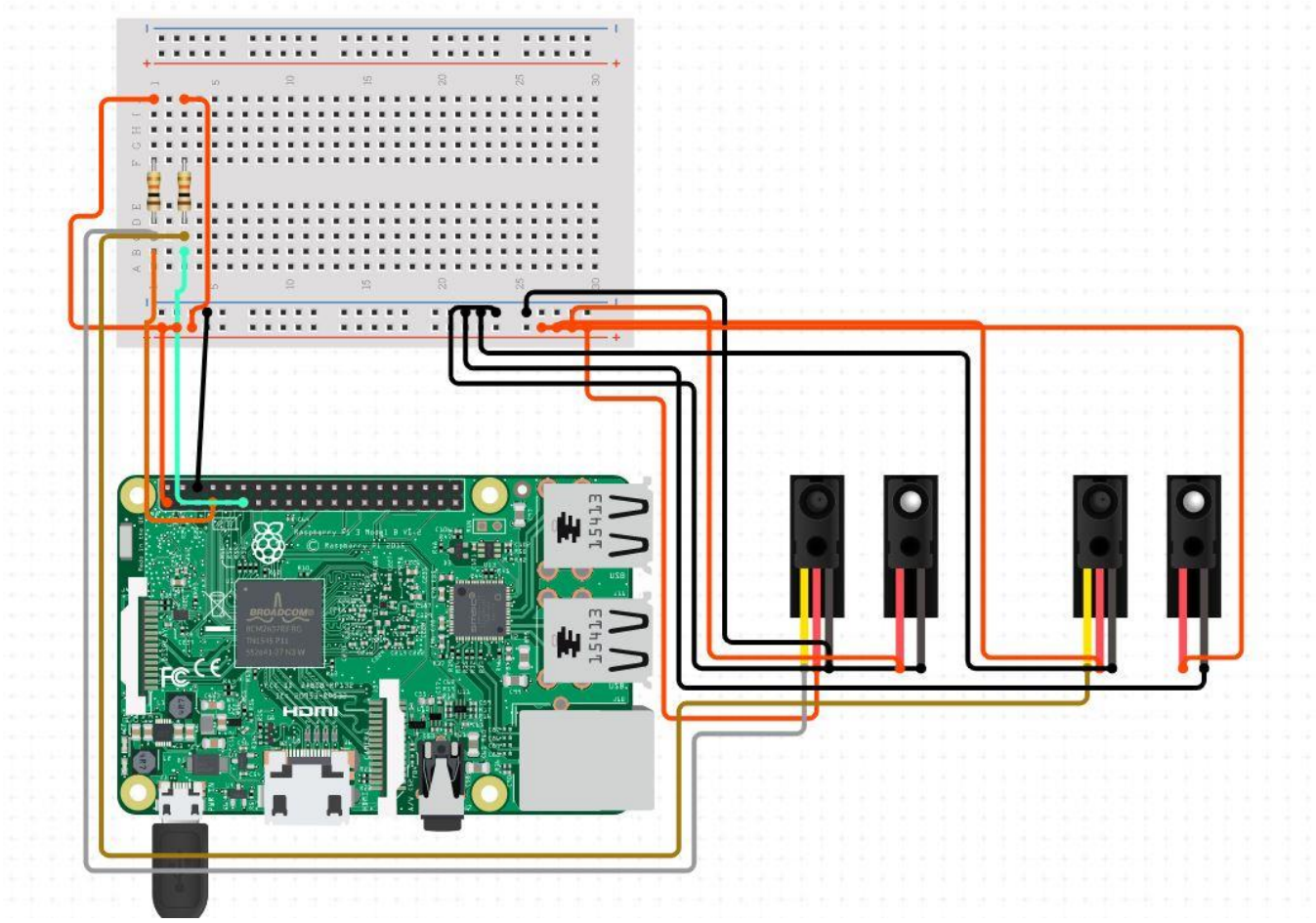
- Hardware List:
  - Raspberry Pi 4 with 8GB RAM && 8GB Memory Card
  - Raspberry Pi Camera
  - IR Break Beam Sensor LEDs

Hardware Collaboration

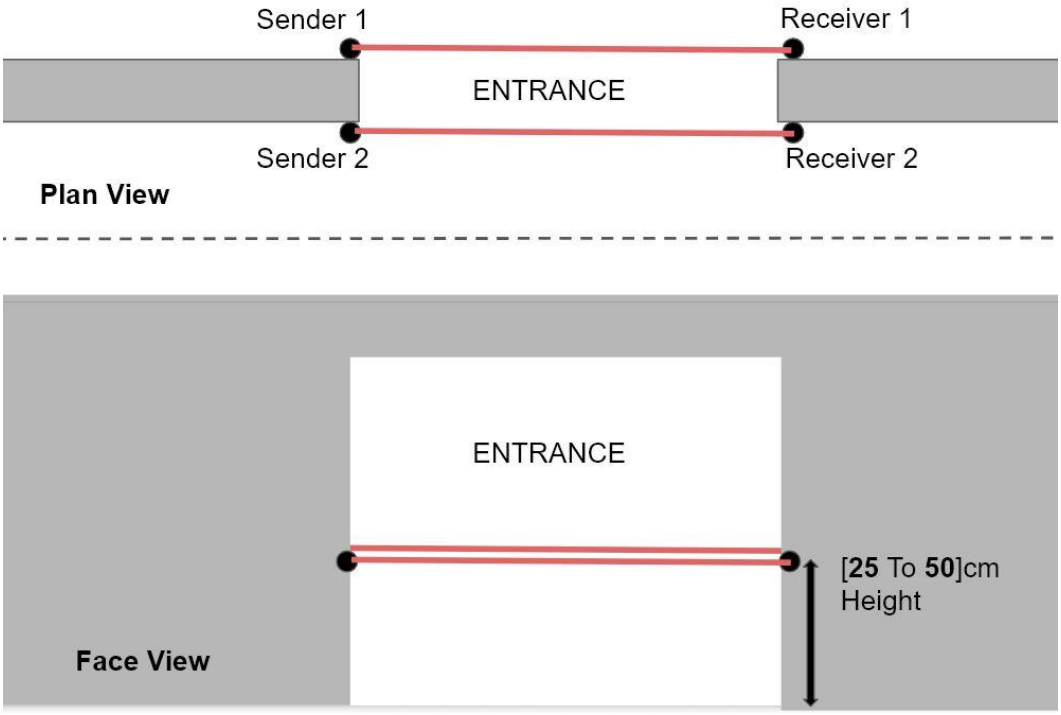
To have the best performance of your hardware the following instructions should be followed:

IR Break Beam Sensor LEDs

- Sensor one should be connected to GIPO 4 PIN(7)
- Sensor two should be connected to GIPO 17 PIN(11) As is shown in the picture below:

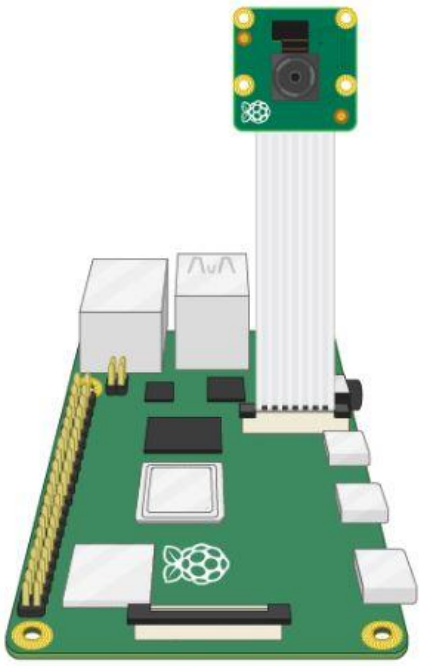


- Both sensors should be installed on both side of the door, with range [25cm] to [50cm] height from the ground taken the kids and adults height under consideration as shown in the picture below:



Raspberry Pi Camera

- the camera connects to the camera port on the Raspberry Pi as it is shown in the picture below.#

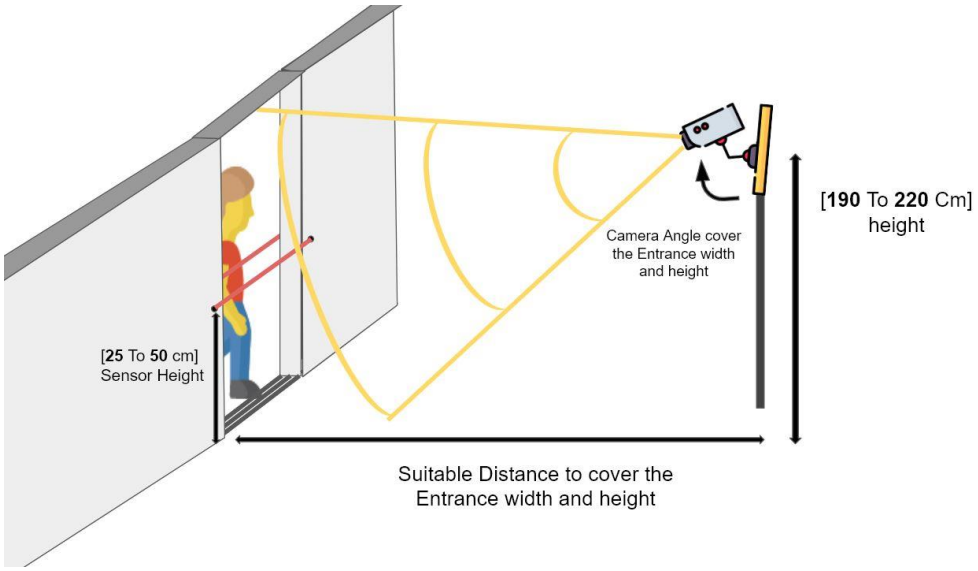


- Now you need to enable camera support using the **raspi-config** program you will have used when you first set up your Raspberry Pi:
    - Use the cursor keys to select and open Interfacing Options, and then select Camera and follow the prompt to enable the camera.
    - To test that the system is installed and working.

Try the following command:

```
$ raspistill -v -o test.jpg
```

- The camera should be installed closer to the midpoint of the entrance, with a height range [190cm] to [220cm], with distance and an angle cover the full Entrance width and height.



How To Run EIH Backend?

To Run EIH Project you do need the same software package to both directions **IN** or **OUT** and that by telling the system what direction to run with the argument passed as the following commands:

Before you run the project, This environment variables should be export and save to run the project.

```
    $ export FIREBASE_DB_URL='https://<YOUR-PROJECT-NAME>.firebaseio.com/<THE-JSON-FILE-NAME>.json'
    $ export REG_BUILIDING_ID='<THE-BUILDING-ID-FROM-FRONT-END-REGISTARTION>'
    $ export DEVICE_ID='<THE-DDEVICE-ID-YOU-CHOSE>'
    $ export PROJECT_API_KEY='<THE-FIREBASE-API-KEY>'
```

To get `REG_BUILIDING_ID` you have to follow the instructions on (the fron-end add new building)

To Run The Project for [OUT] Direction

```
$ python eihIR.py out
```

To Run The Project for [IN] Direction

```
$ python eihIR.py in
```

And this will change the calculations in the file post-data-to-api.py in the following code.

```
if args["numberOUT"] is not None:
    # substact the number coming from the API to the number coming from the RassPi
    newNumber = preNumber- currentNumberOUT
elif args["numberIN"] is not None:
    # add the number coming from the API to the number coming from the RassPi
    newNumber = preNumber + currentNumberIN
```

To Run the Project In The Both Direction, You need Two Hardware Set As It's Shown In The Picture