Visitor-Counter: Development of a device and software to capture and count visitors

Stefan Düx

*Dept. of Computer Science*

*of UAS Technikum Wien*

Vienna, Austria

if20b245@technikum-wien.at

Rebekka Tscheppen

*Dept. of Computer Science*

*of UAS Technikum Wien*

Vienna, Austria

if20b164@technikum-wien.at

Jessica Isabella Görög

*Dept. of Computer Science*

*of UAS Technikum Wien*

Vienna, Austria

if20b094@technikum-wien.at

Lukas Varga

*Dept. of Computer Science*

*of UAS Technikum Wien*

Vienna, Austria

if20b167@technikum-wien.at

Dominic Grabner

*Dept. of Computer Science*

*of UAS Technikum Wien*

Vienna, Austria

if20b219@technikum-wien.at

***Abstract - Nowadays digital surveillance is used in public spaces with hightech equipment, to secure high risk locations or workspaces and many more use-cases. Alongside all these conditions is it interesting to see, if a simple people detection can be realized within a low budget solution. Lite resources for such a project include cheap hardware, free software libraries and components to handle data. Opening a new room for technical implementations of the UAS Technikum Wien, a system to detect people and count visitors is needed. Therefore, the result of this paper will be a simple solution to realize the visitor counter and what technical implementations are needed to do so. Problems and optional results are also provided. The most critical part is the access to high or low quality resources - regarding hardware and software. Testing results show that free to use libraries can not reach their potential in combination with weak hardware such as the Raspberry Pi 4B.***

***Keywords: digital surveillance, visitor counter, people detection, simple solution, free software libraries***

HARDWARE

For the hardware a Raspberry Pi 4B and the referring raspberry pi camera v2 module were used [1]. At the beginning it was planned to power the mini-computer via PoE (Power over Ethernet), but the power source had malfunctioned and was set to fire with the second try. Because of this, the original USB-C power supply was used. In cooperation with the Project Kitchen (technical workshop owned by university), a customized case was printed by 3D technology [2].

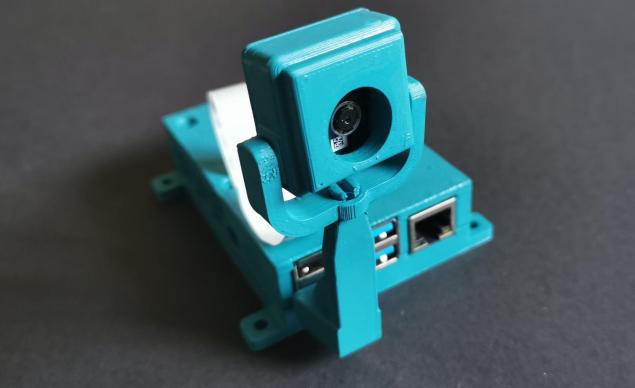


Figure 1. Hardware, assembled 3D case.

For the configuration of the Raspberry Pi 4B, a light operational system needed to be installed, because of the reduced capacity. Therefore, Raspberry Pi OS lite was installed, which comes without a “GUI” (graphical user-interface) which means in our case an operational system with only a terminal to use. Furthermore, the packages, dependencies and libraries were installed, the repository loaded and the autostart prepared. This means that the people detection in the form of the live image processing starts right after the device is started.

PYTHON

The code is programmed in Python (vers. 3.8) which was adapted with “pdm” (Python package and dependency manager) to manage dependencies and standards. Using class based coding we created the file pyproject.toml to manage all dependencies on a central location which includes the following dependencies:

"paho-mqtt",

"opencv-python",

“imutils",

"pyyaml",

"requests",

"flask>=2.2.2",

"matplotlib>=3.6.1"

The code was subclassified to different modules, containing “HttpStreamer.py”, “Logger.py”, “main.py” and “MqttSender.py”. Additionally, the config.yaml file includes the actual mqtt broker which gets the live data sent via the paho-mqtt library.

Using the openCV library for the image processing, we made use of the HOGDescriptor\_getDefaultPeopleDetector method to detect people in a given stream or frame [3]. This method can also be used to detect other objects like cars, which adds to the open options of advancing the project in the future. This part can also easily be adapted or exchanged if more valuable resources are obtained (paid libraries with higher efficiency).   
The flask library is used to send the live-stream of the camera module with the adapted picture (including the frames marking the detected people) to the frontend. This streams a picture via mimetype (multipart/x-mixed-replace) via HTTP as a frame.

NODE-RED

Node-RED is used to transfer events from the mqtt-broker to our database (InfluxDB).

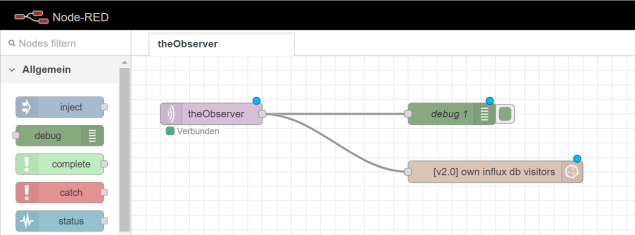


Figure 2. Node-RED, nodes.

INFLUXDB

InfluxDB is used as our database which saves data, transferred by Node-RED from mqtt-broker into a timeline. This saved data can be queried and used to visualize in Grafana (see subchapter).

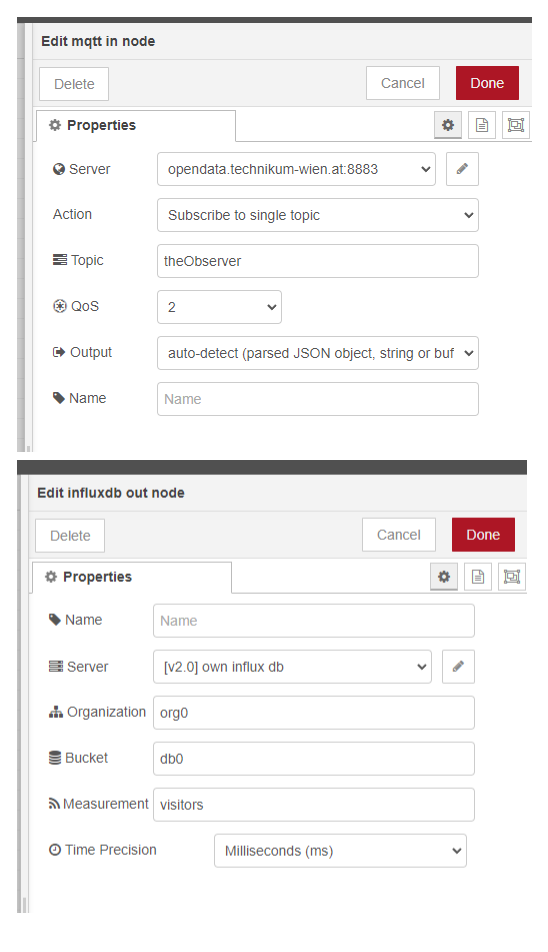


Figure 3. InfluxDB, settings.

WEBPAGE

For the Framework Angular was chosen, because we were already familiar with the basics of angular due to working on projects using it in previous semesters. To further facilitate the setup the basic setup was done using the Ionic framework [4]. Ionic is a tool used for cross platform development which helps facilitate certain aspects of web-development, in our case it took care of routing. Overall the angular frontend allows for an easy way to present the data gathered by the Raspberry Pi.

To provide the possibility to visualize the data which gets processed by the PI-Module and software, a simple Webpage was set up. The webpage takes the output from the VisitorCounter-Software and presents it in different tabs. The “home”-page provides a life number of visitors.

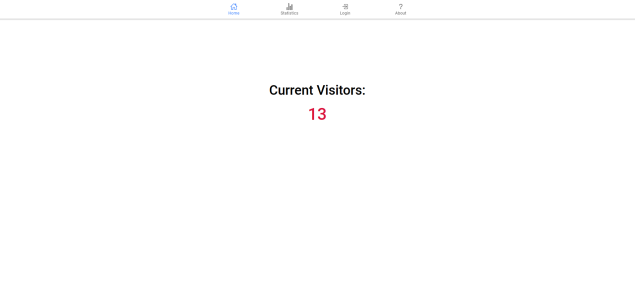


Figure 4. GUI, Visitors.

The “statistics“ page shows a frame with a timeline and its related visitor counts. In the about tab, general Information about the project can be presented. On opening the webpage the home-tab is displayed, which shows the live visitor counts to realize the live update the webpage is subscribed to a mqtt-broker that publishes the current number on every update it receives from the Observer-software. The statistics page contains an iframe which is provided by Grafana.

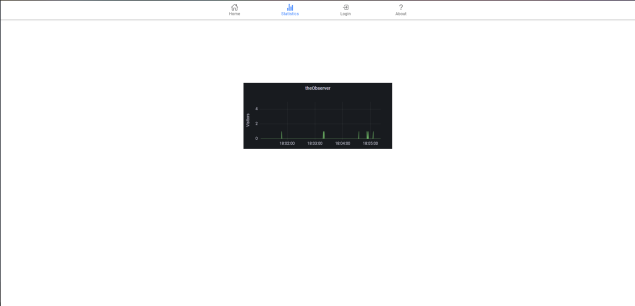


Figure 5. GUI, database entries.

This iframe contains the visualized visitor counts for each timestamp in the specified timeframe, in this case the overall time since the Pi has started recording. Last but not least the about page is used to provide general information about the project.

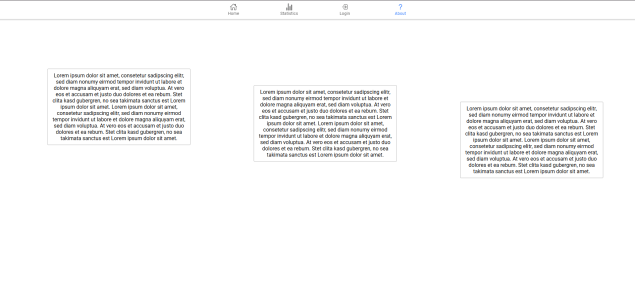


Figure 6. GUI, about information.

Possible outlook for project: Add the login page, so you can display the live-recording from the Pi with drawn rectangles on recognized visitors. The possibility to do this has already been implemented and can be added via a simple img tag to the endpoint: <http://localhost:5000/video_feed>.

MQTT-BROKER

The Pi sends a constant stream of detected visitors with every processed frame. To handle the huge amount of requests a protocol with high performance and speed was needed, to handle the sheer amount of data. MQTT is the protocol that many IoT-devices use to communicate their data and fits perfect for this use-case. The raspberry pi publishes its data under the topic theObserver. The data can now be read by everyone by simply subscribing to this topic. For MQTT-Support in Angular we used the PAHO-library, which makes it possible to subscribe to MQTT-Brokers in Angular via websockets. Every subscriber will get a notification every time a new update is done under the topic by the raspberry. There is a MQTT-Broker set up by the FH under the address opendata.technikum-wien.at at port 8883 and topic “theObserver”, however there is a limitation when subscribing to a topic using angular and PAHO, a module for mqtt-support. When trying to subscribe to a topic using angular you have to use websockets to communicate since it does not support the mqtt-protocol. However the FH-Broker does not have websockets configured and we can therefore not use it yet. As an alternative it is possible to use an open source public broker like mosquitto or hivemq to test the concept.

Possible outlook for project: Once websockets are configured on FH-Server you can change the code to implement it, since it is not recommended to communicate visitor data via public services.

GRAFANA

Grafana is an open source analytic web-application which has ways to analyze and visualize data on multiple platforms and databases. Queries get copied to Grafana and thereupon visualized in a graph [5].

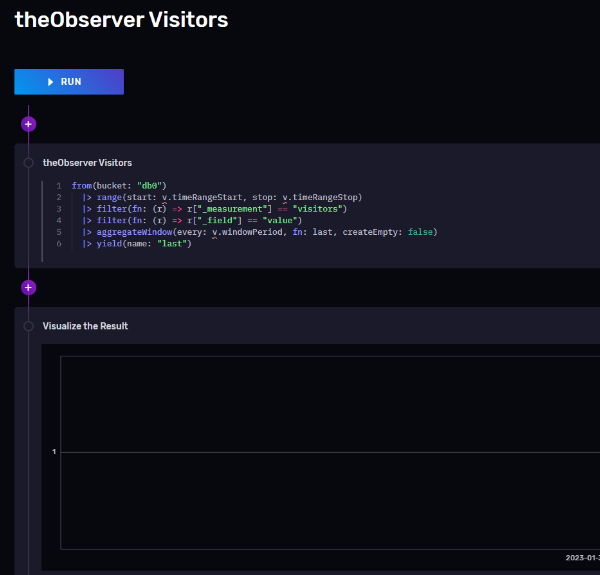


Figure 7. Grafana, queries.

The visitor counts from the Observer-software are persisted in a Influx-Database with a timestamp. Grafana provides a convenient way to take time related data and create graphs which can then be exported and used in the frontend via a simple iframe. In our case the visitors over a certain time period are displayed under the statistics tab. For now the whole timeline with its visitors is displayed.

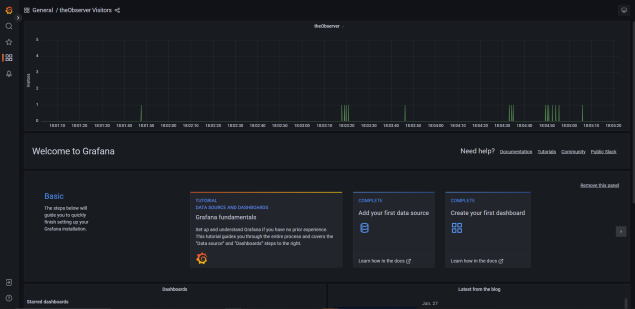


Figure 8. Grafana, GUI queries.

Possible outlook for project: It is possible to extend the statistics that are displayed, with for example average visitors per day, a display for the highest or lowest number of visitors in a month/day, to give more insight into visitor numbers of the showroom.

REFERENCES

1. J. Wang, L. Long, *Raspberry Pi 4B-based Design of Home Intelligent Access Control System.* China, Dalian: IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), May 2022.
2. C. Kim, D. Espalin, M. Liang, H. Xin, et al., *3D Printed Electronics with high performance, multi-layered electrical interconnect.* USA, Texas: Keck Center for 3D Innovation, October 2017.
3. S. Parveen, J. Shah, *A motion detection system in Python and openCV.* India, Tirunelveli: 3rd International conference on intelligent communication technologies and virtual mobile networks, March 2021.
4. O.C. Novac, D.E. Madar, C.M. Novac, G. Bujdoso, et al., *Comparative study of some applications made in the Angular and Vue.js frameworks.* Romania, Oradea: 16th International conference on engineering of modern electric systems (EMES), July 2021.
5. L. Manases, D. Zinca, et al, *Automation of Network Traffic Monitoring using Docker images of Snort3, Grafana and a custom API.* Romania, Sovata: 21st RoEduNet Conference: Networking in Education and Research, October 2021.