Smart Welcome Gadget

Eira, Leon, Teemu

19.12.2025

1 Introduction

Have you ever wondered how many people have visited the same museum on the same day as you? Or how many people in total have visited the local shop now that it is December and everyone is buying Christmas presents? Or what about all the passers-by who went towards their unknown destinations in silence, without being noticed at all?

For this course, we decided to implement a smart welcoming gadget. The goal is to create a device that would count the visitors of a shop or passers-by on a hiking trail, and greet them happily. One of the most influential inspirations for this project came from Oulu city's counters located next to the bike lane. Nevertheless, those counters only count the number of bicyclers and walkers, but they lack interaction. Therefore, in our implementation, the greeting will be done with a light show. Due to humans being social creatures, by interacting, some people may even get a sense of community and communication.

In addition to the interaction of greeting people, the data that is being collected via the device holds a monetary value. Since the number of visitors or passers-by can be tracked, a trend can be compiled to show the number of people in relation to a specific time frame. This allows one, for example, to analyze people's behavior and possibly even estimate the most popular visiting times for a museum. In addition, improving and developing the light show functionality further could open new business opportunities in the field of events and art. For example, the Lumo Light Festival is held in Oulu yearly, and Oulu is also the European Capital of Culture 2026. In other words, in the best possible example scenario, the creative potential of the light show would be utilized in order to be a part of such events.

With these thoughts in mind, our team naturally has to think about the implementation of the device. For this project, a microcontroller (Raspberry Pi Pico W2), wires, and a breadboard were provided to us on behalf of the IoT course staff. The motion sensor was bought by the team, and the LED lights and the resistor were kindly provided to us by FabLab. It was also mandatory to implement sensor data collection, data processing and output control, and the descriptions of the functionalities are the following:

1. **Sensor data collection:**

   In our implementation of the project, an infrared motion sensor is used to monitor movement. This is done by utilizing the HC-SR501 PIR module.

2. **Data processing:**

   Data from the movement sensor is saved, opening up different possibilities for its use. Within our current scope, the goal is to compile the data into a simple graph or a trend that shows the number of people in relation to a specified time frame.

3. **Output control:**

   The output in our case is two LED lights, which are wired into the breadboard together with a resistor. Whenever movement is detected, the LED light turns on, and data is saved.

In addition to the mandatory requirements – sensor data collection, data processing and output control, it was instructed to implement at least two more functionalities. When deciding which functionalities to implement, the focus was on usability and how to add value to the data collected. Therefore, our team decided to implement the following functionalities:

1. **Wireless connectivity:**

   Internet connectivity is implemented in our case. Originally, the device worked on a local WiFi network as it started an HTTP server and listened for a connection whenever plugged into a power source. After testing our device and receiving feedback from the demo, we opted to use MQTT as our communication protocol. Using MQTT has way less overhead compared to HTTP and seems to be better suited for IoT devices in general.

2. **Mobile App:**

   A mobile app is a great way to improve the user experience and target usability. The app in question is a simple "hub" for the device, where different values can be modified and interactions can be set, such as at which point the LED turns on, showing graphs etc.

3. **Database:**

   A database is implemented to store data. The implementation utilizes Google Sheets and Apps Script. A timestamp is stored in the database whenever a motion is detected. The goal is to compile a trend or a graph by utilizing the data collected in the database.

2 Architecture

Our project's sensor layer consists of a HC-SR501 PIR module, which is a motion sensor that detects movement. It is connected to a Raspberry Pi Pico

W2 microcontroller. When the sensor detects motion LEDs connected to the microcontroller are blinked.

The networking layer enables communication between the microcontroller, cloud services and application. When the sensor detects motion a counter is incremented and this value is sent forward to a cloud database and MQTT broker for future use. The microcontroller sends the data via a wifi network. HTTP is used to communicate with the Google Sheet database and MQTT with the HiveMQ cloud broker.

HiveMQ and the Google Sheet database make up the data management layer. HiveMQ is a MQTT broker that routes the messages between the microcontroller and the application. Google Sheets are used as a cloud database for storing values and timestamps related to them. They can later be utilized in graphs and analysis and other visualization.

Application layer is comprised of an application apk made with MIT App Inventor. This is where the user interacts with the system. The movement count is displayed to the user and they can modify the interval at which the LEDs are blinked from the app. The project Flowchart can be seen below in Figure 1.

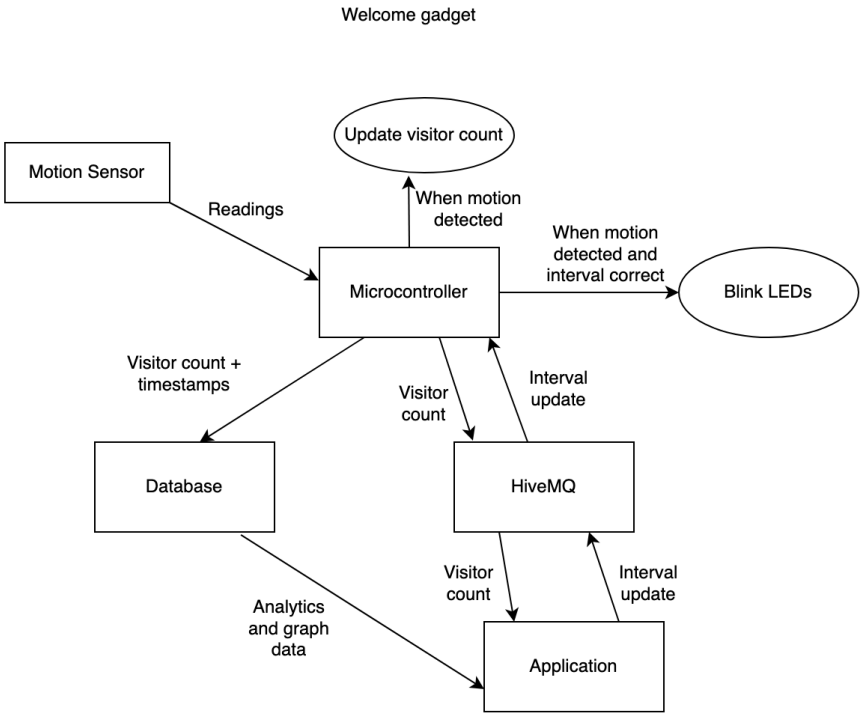

Figure 1: Project Flowchart

3 Methods & Tools

During this project, multiple tools were used in hopes of achieving a functional, fun and coherent device. To implement the hardware side of the project, MicroPython was used together with the Thonny Python IDE. MIT App Inventor was used to create the mobile application, HiveMQ was used to transmit data via MQTT, and Google Sheets was used for the database. Also, in addition to the course material and exercise documentation, literary resources from the internet, such as Raspberry Pi forums, were also used to support our work.

When it comes to project management, our team used an iterative prototyping approach. Due to noticing that testing would be a crucial part of implementing this device early on, our implementation strategy quite naturally fell to follow the incremental implementation method. In this method, new features are implemented into the project in smaller, more manageable steps rather than all at once. In our case, as defined in the implementation plan, this means we started with the motion detection and counting. When those were functioning correctly, the output control (LED) was integrated into the system. After that, the focus shifted to the application and wireless communication. In conclusion, this approach enabled us to do unit testing first, followed by integration testing later.

Our team's practices and working styles were flexible throughout the whole project. Each member completed some work individually, but most of the work was done together. We held team meetings both online and in-person, and a Telegram group was created for communication at the beginning of this project.

3.1 Source Code

The project code can be found on GitHub.

3.2 Microcontroller

We use a Raspberry Pi Pico W2 microcontroller in our project. The role of the microcontroller is to read motion sensor data from a PIR sensor. These readings are relayed forward to a cloud database and an MQTT server. The microcontroller also controls LEDs connected to it based on instructions from the user. To make the operation more robust, the code prevents very quick repeated counts.

All this is implemented using a MicroPython script developed in Thonny. The microcontroller communicates via a Wi-Fi network, sending motion sensor reading updates to the cloud and timestamped data to a database. It receives requests from the MQTT via messages that are used to update the LED blinking interval.

3.3 Phone application

The phone app for the project was implemented with MIT App Inventor. It's a powerful, yet relatively simple tool to develop working prototypes of a phone app. The phone application connects and communicates with the device through HiveMQ. The app receives updates from the device and can send light show interval changes to the device. The application can also communicate securely with our database, using an authenticated Google service account.

The user interface of the mobile app is shown in Figure 2. The live counter is located on the upper part of the screen, under the title "Number of People Detected Today:". Under the live counter, there is a "Subscribe" button. When the device and the mobile app are connected, a text cue is given to the user by showing the text "Connected" and "SUBSCRIBED". There is also a section for the lights and the switch for turning on or off the light show. The user is also able to input the interval for the light show.

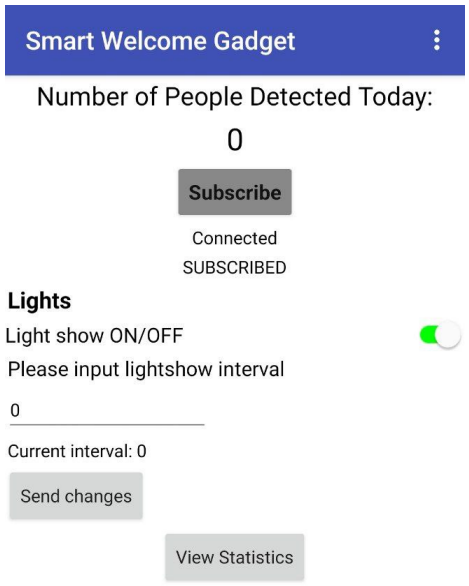

Figure 2: Mobile Application user interface

To control the application, visual coding was done in the MIT App Inventor. There are seven code blocks in total: one for the screen, one for each button, one for the switch, one for the MQTT connection, and one for communication. In addition, a global variable *currentcount* is defined. The visual code of this mobile app is shown in the Figure 3.

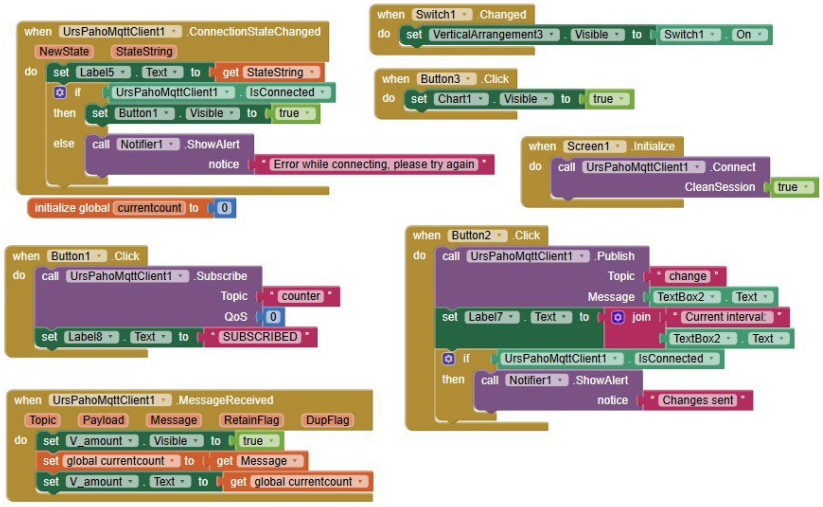

Figure 3: Photo from the MIT App Inventor: visual code for controlling the mobile app

3.4 Cloud

For implementing the cloud, the team opted for HiveMQ paired with an extension. HiveMQ has its own cluster for cloud-related operations, and the implementation was mainly done with the help of instructions from the course's Exercise 3. In addition, for the sake of MIT App Inventor, an extension was utilized to ensure smooth and clear communication between HiveMQ and the mobile app.

In practice, the mobile app first connects to the HiveMQ broker and subscribes to the topic "counter". This means that all the messages of the topic "counter" from the microcontroller would end up in the mobile app and show the live count to the user. One can also send a message to the topic "change" from the app if they want to change the interval of the light show. As the microcontroller is also subscribed to the "change" topic, the user input can be saved and updated.

3.5 Database

In this project, Google Sheets was utilized to implement the database. Inspiration for the database implementation was taken from various sources, but this was the most influential source. The team decided to proceed with Google Sheets since it is lightweight and has good functionality for data analysis. As mentioned earlier, data collected by the device needs to be stored for further processing. In practice, the microcontroller sends timestamps to the database whenever movement is detected. This operation is being done using a short Apps Script shown in Figure 4. The mobile app retrieves data from the database using an authenticated service account.

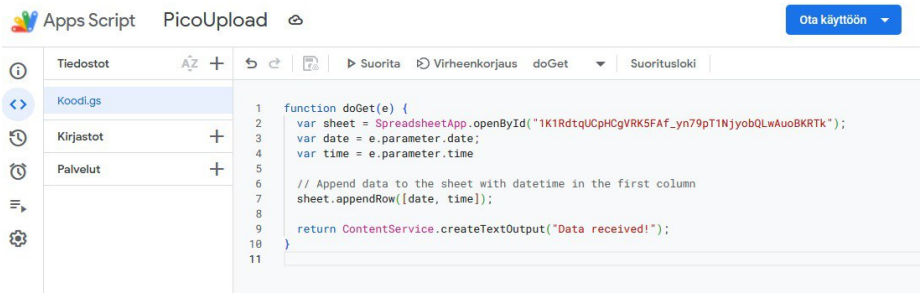

Figure 4: Screenshot of the Apps Scripts for the database implementation.

4 Evaluation

The evaluation criteria were defined early on, due to testing being an important part of any project. The selected criteria for evaluation were kept in mind constantly, and they are the following:

- Correctness of counts
- Latency
- How the device handles itself in busy scenarios
- Hard limitations/non-plausible use cases

Two evaluation scenarios were also created in order to compare the application's performance in different settings.

- **1.st scenario:** A real-life non-busy time period of 2 minutes. Limited number of people. Only one person is coming at a time.
- **2.nd scenario:** A real-life busy scenario of 2 minutes. Multiple people are coming and going at the same time.

5 Results

During the project, functionalities were tested separately and gradually at first. After the components worked on their own, we could implement the parts together and evaluate according to plan. The two evaluation scenarios were carried out and analyzed based on the evaluation criteria defined earlier.

5.1 The first evaluation scenario

The first evaluation scenario was created to depict a real-life non-busy time period of two minutes. Only a limited number of people were allowed, and only one person was coming at a time. The results of the first evaluation scenario is shown in Figure 5, where the number of motions is shown in the y-axis, and the actual motion detected is shown in the x-axis.

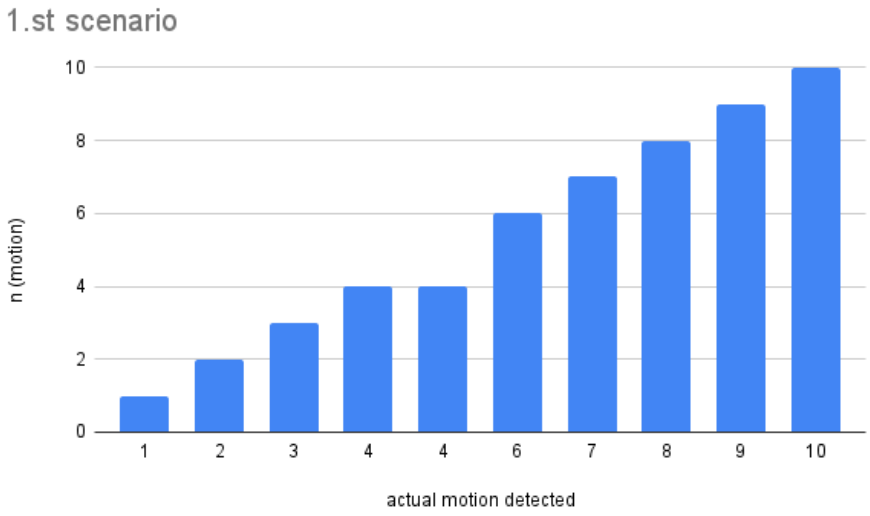

Figure 5: Visualization of the results of the first evaluation scenario

When it comes to comparing the results from the first evaluation scenario to the evaluation criteria, the correctness of counts was mainly correct during the test time period. When the fifth movement should have been detected, the counter still showed four. But when the sixth motion was detected, the counter was updated correctly. Therefore, our team looks at this case as more related to latency rather than the correctness of counts. For this scenario, the evaluation criteria regarding the device's performance in busy scenarios can not be fully evaluated, since the first evaluation scenario focuses on depicting a non-busy time period. Similar to the device's performance in busy scenarios, the hard limitations/non-plausible use cases can not be fully evaluated due to the

limitation of only one person coming at a time.

5.2 The second evaluation scenario

The second evaluation scenario was created to depict a real-life busy scenario of two minutes, where multiple people are coming and going at the same time. The results of the first evaluation scenario is shown in Figure 6, where the number of motions is shown in the y-axis, and the actual motion detected is shown in the x-axis.

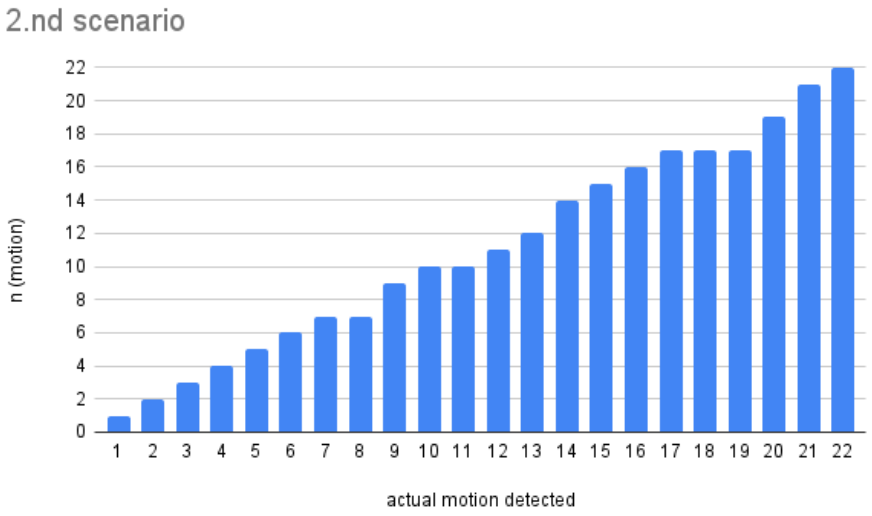

Figure 6: Visualization of the results of the second evaluation scenario

When looking at the results from the second evaluation scenario and comparing them to the evaluation criteria, similar results to the first evaluation scenario can be detected when it comes to the correctness of counts and latency. Nevertheless, due to the missing restriction that only one person could enter at a time, the latency problem appeared more often, and the time period to update the count correctly took more time. Since there were only three people in our team, we were not able to create an extremely busy scenario to test the device's maximum capacity. Therefore, even in the second evaluation scenario, the criteria of hard limitations/non-plausible user cases cannot be fully evaluated. Nevertheless, despite running back and forth, we were not able to cause a non-plausible user case, and the device handled itself okay.

6 Discussion

Discuss the results of the evaluation. What weaknesses does your application have? In what scenarios does it work well? Return to the application area you described in Introduction – put the results in context.

Sensor data collection in our implementation was a success. The sensor did as well as it could in detecting movement. The most severe issues of the sensor is its limited configuration and its need for a long calibration. The minimum sensitivity of the sensor was 3 meters and the minimum "blocking" period was 3 seconds. This leads to many false positives and the sensor can't "keep up" with the Pico W very well. Summarized, the sensor works as intended, but there most likely are better ones for our specific use case.

Data processing was a mixed success. We have data from our testing and we can use it to make graphs and to inspect trends. The bottleneck in our case was that it is an encumbering task to deploy a graph using MIT App inventor coupled with Google Sheets and App inventors built-in graphing solution. We simply do not have enough time to read through all the documentation and to test the implementation thoroughly.

Output control was the first thing that we managed to implement in this project. The logic is pretty simple and it works well. The only issue in the implementation is a small, random bug that causes the led to stay on a while longer than intended. This happens rarely and lasts for less than five seconds.

Wireless connectivity in our project is solid and there are no glaring issues. Both the app and the device communicate using MQTT and are subscribed to each other. The weak link in our connectivity is Pico W to database connectivity. This is due to having to communicate using HTTP, but this issue is mitigated by using a queue for the database updating.

The mobile application we made for this project is simple, but functional. The app functions as intended and is easy to use and understand.

The database in our implementation is quite unique. While using Google Sheets is a relatively simple and effective way to store data, its lacking aspect is communication using MQTT. It is very time consuming and overall unfeasible to have it communicate directly with HiveMQ.

Overall, the project was a success. We reached almost all the goals in our scope and made a functional, user-friendly IoT device. We started early and made good progress throughout the course, except for the migration from HTTP to MQTT, where developing the changes got a bit rushed.