

UNIVERSITY OF WATERLOO



SE 101 Introduction to Methods of Software Engineering

Fall 2022

December 2nd, 2022

Course Project Final Report

Team Hot Wheels

Full Name	Student ID	User Name
Abhinav Balasubramanian	21004672	a3balasu
Daniel Larkin	21004120	ddlarkin
Dhyan Patel	20999506	djpatel
Kieran Hulsman	21011088	kghulsma
Nihal Menon	21040511	nmenon

Introduction

Our project was inspired by our personal experiences studying at UWaterloo. We often found that we would enter study spaces and find them heavily occupied and unable to study in, leading to unnecessary time spent on relocating. This felt like an inconvenience that we were capable of finding the solution to. For our project, we created a system where we can keep count of the traffic for a given space. Our approach consisted of two main sections, hardware and software, and their integration with each other. Put simply, to keep track of the number of people in a given space, we set up a network of sensors that triggers and updates a counter. This information is then sent and processed behind the scenes, eventually making its way onto a website where students can access live traffic data.

Background Research

The official name of a device that tracks the traffic in a room is “people counter”. When brainstorming different algorithms, our group studied various implementations of people counters that exist in the real world. Companies like [SenSourceInc](#) use IR sensors paired with Machine Learning, while others use cameras with computer vision. One trait in common with all the products we found on the market was the use of a detection algorithm that determines the direction and number of objects. With those key principles in mind, our group decided to construct a product that utilises a pair of ultrasonic sensors and a thermal camera. We decided to use a thermal camera as opposed to a regular camera to respect student’s privacy and avoid any issues.

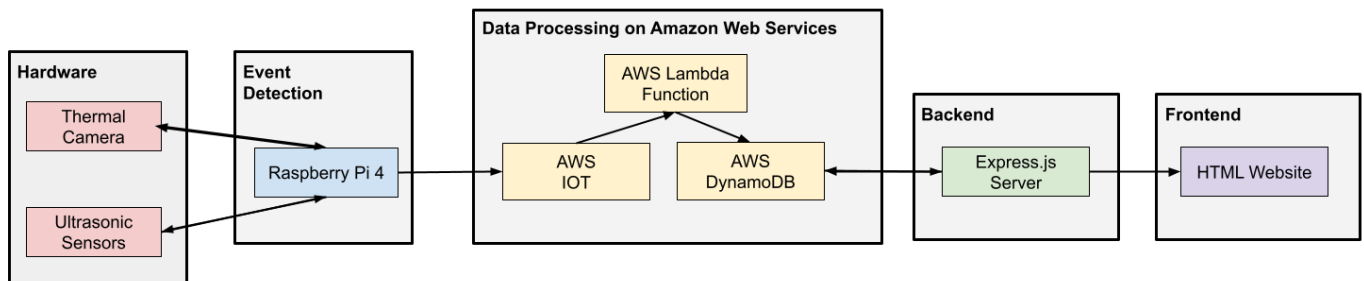
In order to wire the ultrasonic sensors as well as the thermal camera to the Raspberry Pi, we needed to use tutorials that demonstrated the process of calculating the necessary voltage output and input in order to set up an input to a Raspberry Pi. We also needed to use the official documentation and [diagrams](#) of the Raspberry Pi that displayed the GPIO pin layout for the Raspberry Pi 4. This was crucial as we had limited pins and had to connect three different inputs to the same Raspberry Pi.

Object-detection algorithms were a foreign concept for our group. Before we decided on an implementation, we needed to understand *how* the algorithm works. By watching videos by 3Blue1Brown about [convolutions](#), and reading various [Medium articles](#) on what is actually being done behind-the-scenes, our group had a starting point on how to develop our own. From our findings, it revolves around the concepts we learned in MATH 115 about matrix multiplication and some new concepts on how images are pre-processed in machine learning models.

The group had no experience with AWS before starting the project. The main tutorial we followed was a [YouTube video](#), which explained how to send data from the Raspberry Pi to an AWS IoT topic. From there (Lambda, DynamoDB, etc...), research mostly consisted of looking up how to do small, specific parts of the project.

Implementation

ForAllTimes Block Diagram



Ultrasonic Sensors

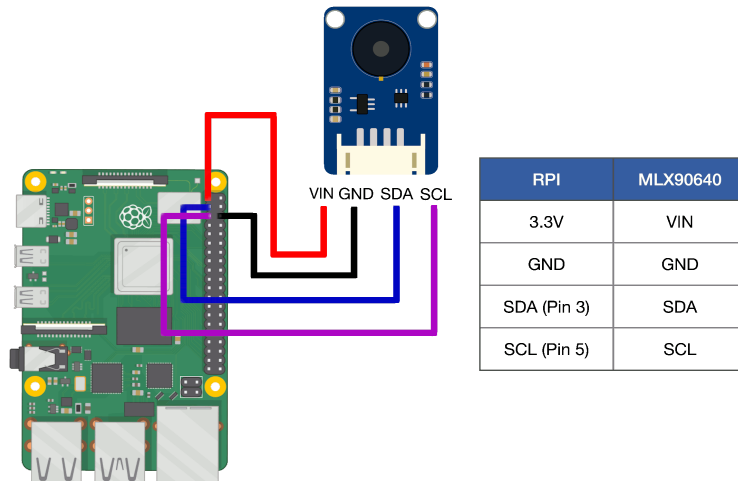
The two ultrasonic sensors are connected to the Raspberry Pi's GPIO pins through a breadboard, jumper wires and resistors (to account for the difference in the output voltage of the sensors and the input voltage of the Raspberry Pi). The Raspberry Pi runs a Python script in order to determine whether a person is walking through. The Pi starts off with a base measurement of the distance to the wall. It then repeatedly sends pulses and measures distances to an object. When the current measurement drops under a certain tolerance from the wall measurement, the respective sensor is "triggered" and the program listens to the second sensor. Only when the second sensor is triggered and returns to the original wall measurement does the Raspberry Pi send a JSON object to AWS containing the direction of movement (exit or entry), time and temperature array.

Thermal Camera

The purpose of the thermal camera is to send a thermal image to the machine learning model in the form of a multidimensional array. This is then used to calculate the number of people entering or exiting the room at a given time. The process of setting up and integrating the thermal camera has 3 steps:

1. Setting up the hardware:

Before any temperature can be read, the thermal camera needs to be connected to the Raspberry Pi. The thermal camera allows roughly 3 frames-per-second visualisation of 768 pixels and takes 3.3V. Because of this, it can be directly connected to the Raspberry Pi's GPIO pins like such:



2. Downloading necessary libraries and software:

After connecting the camera with the Raspberry Pi certain libraries need to be downloaded and settings need to be enabled in order for the Raspberry Pi to detect and work with the camera. In terminal settings I2C needs to be enabled and the command “`sudo i2cdetect -y 1`” needs to be run to ensure the Raspberry Pi registers the camera module and the camera is wired correctly.

3. Receiving temperature and resizing the data:

Using “`adafruit_mlx90640`”, the library that corresponds to the thermal camera, we built a function that when called, activates the camera module and captures an image. Which then is converted to a multidimensional array which can be passed into the machine learning model.

AWS

Once the ultrasonic sensor detects an object, it triggers the thermal camera to take a photo. That data is formatted into a json object, and sent from the Raspberry Pi to AWS via the MQTT protocol. The data is then forwarded to a Lambda function for processing. The function runs our object detection algorithm, and updates the live count on our DynamoDB database to be read by Node.js.

Object-Detection

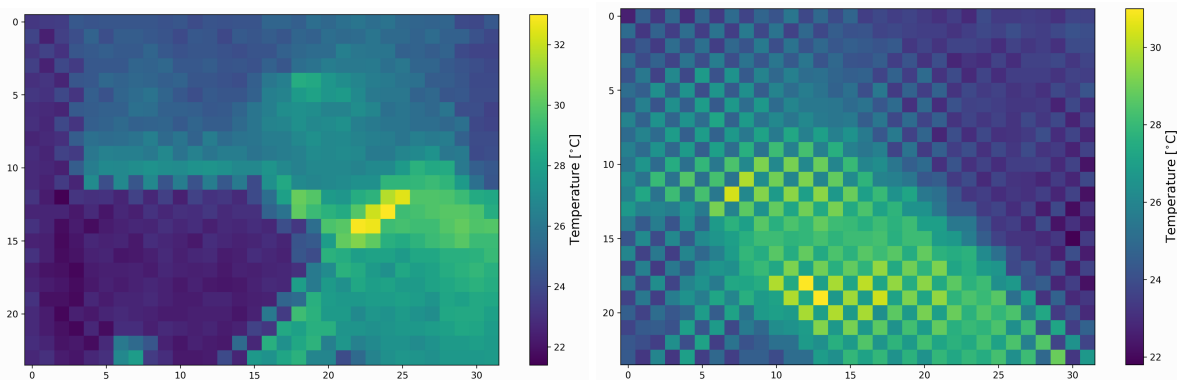
The detection algorithm is used to determine the number of people passing through at a certain time. The object-detection algorithm is split into 3 main steps.

1. Preprocessing

Before any manipulations can be done, the temperature has to first be converted into the correct data structure. As opposed to a built-in array, we chose to use Numpy for its vast features and added functionality for data manipulation. Using Numpy, the raw temperature array was transformed into a 24x32 matrix. In order to check if the data was converted

correctly, we plotted the matrix on a climate graph using the popular data visualisation library Matplotlib.

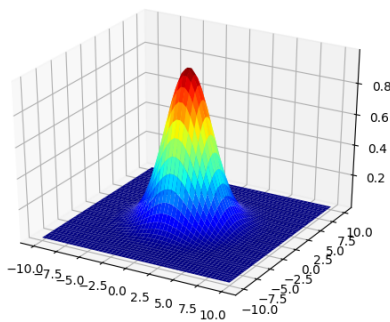
Figure 1. Examples of the raw data plotted on a climate graph.



2. Gaussian Smoothing

The raw data contained a lot of noise from the temperature. Trying to run a detection algorithm on it would yield poor results and a much higher chance for mislabeling data.

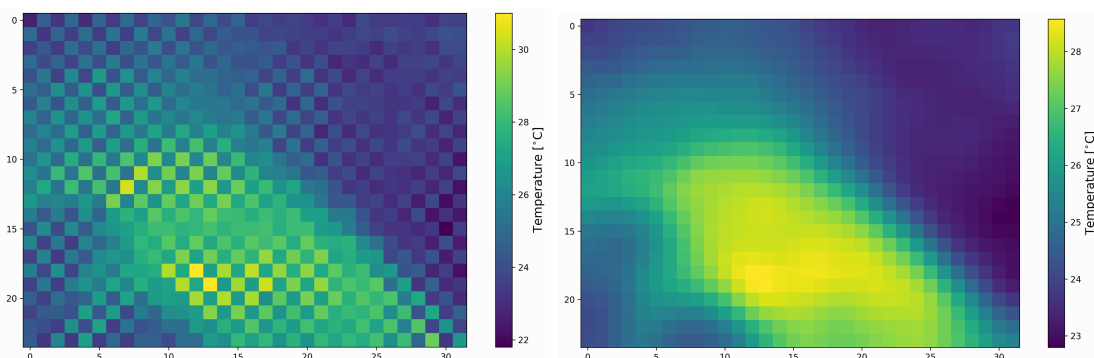
Figure 2. 2D Gaussian Distribution.



A trick that many data scientists use to clean up the noise is running the matrix through a Gaussian Smoothing algorithm. Many of us are familiar with a Gaussian distribution. Without getting into too much of the fine details, at a high-level, the filter iterates through every cell in the matrix and sets it equal to the weighted average of all its neighbouring pixels. The unique aspect of the Gaussian filter is that the weights are based on a normal distribution, with the centre cell having the most influence on the final value. It is clear to see that one

pass through the Gaussian filter yields significantly better results with the edges of the object having a much greater contrast to the background. We also noticed that most of the “checkerboard” noise smooths out.

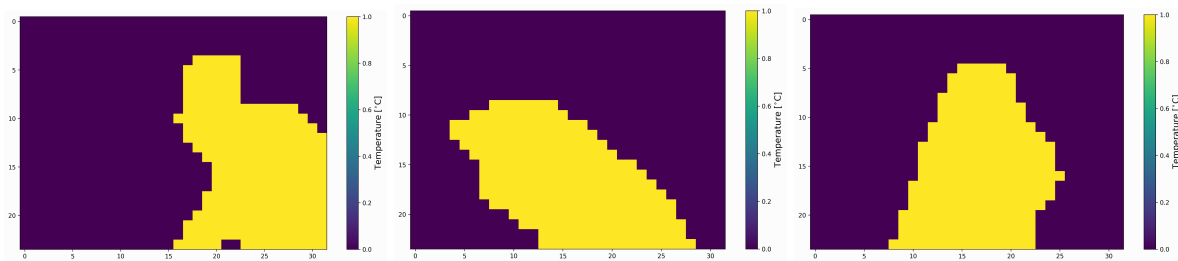
Figure 3. Comparison of raw data and data that has been passed through a Gaussian filter.



3. Connected Components Labelling

The last step of our algorithm involved binarizing the array and then determining all the connected components. Based on our sample data and testing, we calculated the average human's body temperature to show up as a range between ~ 27.5 - 33.0 degrees celsius in the readings. We used that fact to set our tolerance to 27.5 degrees celsius. Then, we iterated through the matrix to set all values lower than our tolerance to 0 and the rest to 1, thereby binarizing the matrix. Finally, all that was left to do was determine the number of groups of 1s, or more academically, the number of connected components. The algorithm to do so is very similar to a popular coding interview question [“How Many Islands”](#). We used an implementation provided by the Scipy.ndimage library called “label”. The final result is surprisingly accurate.

Figure 4. Examples of the final binarized matrix.



Backend

In order to display the current count of people in the lounge on the website, the backend needed to get access to the data in our DynamoDB database. To achieve this, we used Node.js as well as an AWS module in order to make a request to the database. Once the data is received from the database, we use Express.js to display this data on the website using the module handlebars.

Frontend

When creating the website, we wanted to keep the design simple and opted for a one-page design. We prioritised keeping the counter as the centre of attention but added extra features to enhance user experience. For example, we embedded a google maps link of where the SE lounge was located and added a comment section to display user feedback.

Figure 5: Front page displaying traffic counter

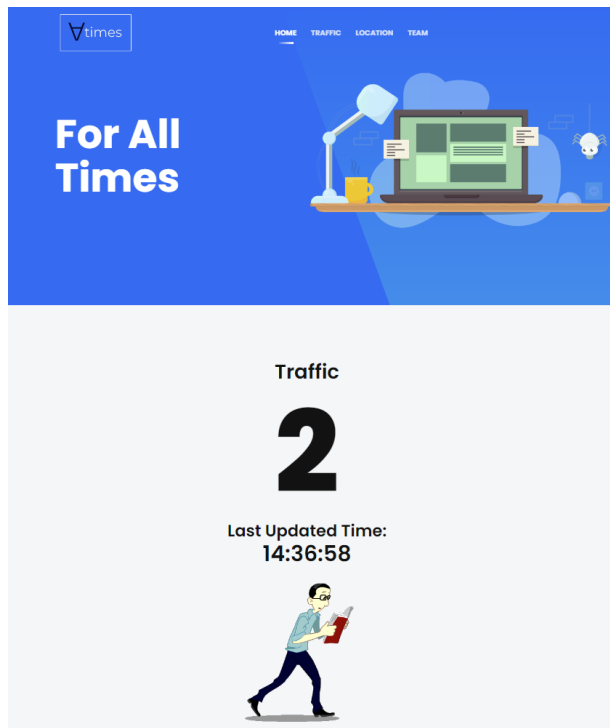
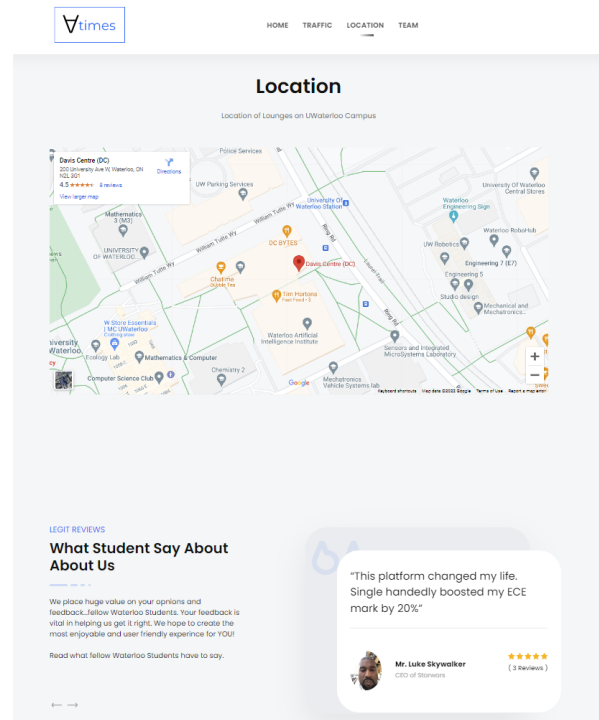


Figure 6: Map and comment section on website



Group Member's Contribution

Abhinav worked on the circuit that connected the ultrasonic sensors to the Raspberry Pi. He also developed the object-detection algorithm based on background research conducted on image processing and connect component labelling. Since there was no pre-existing model for thermal images, he had to develop the algorithm from scratch by applying similar concepts to regular computer vision. Finally, he worked on creating the animation for the implementation that was displayed in the video. The animation was created using the Python library Manim.

Daniel worked on wiring the ultrasonic sensors and thermal camera to the Raspberry Pi. He ensured that the ultrasonic sensors were connected to the Raspberry Pi correctly by using appropriate resistors to match the voltage of the sensors. He also worked on the front end of the project. He designed our logo from scratch and created the website, "For All Times" with Dhyan. Lastly, Daniel helped create the video.






Dhyan worked on wiring and connecting the thermal camera to the Raspberry Pi. He configured the Raspberry Pi settings to be compatible with the sensors and camera modules, while also conducting background research on alternate compatible libraries since most

libraries were out of date. Downloading the necessary libraries and creating the thermal image processing script which collected thermal image data. Also developed a script that converted all collected data into an organised multidimensional array. Lastly, he worked on building the website with Daniel and creating the video.

Kieran mainly focused on AWS. Using the MQTT protocol, he got the Raspberry Pi to publish data to an AWS IoT topic. Upon receiving the data, AWS triggered a Lambda function to process the data. Once the Lambda function was triggered, it determined the number of people entering/exiting, and updated the count accordingly. He used a DynamoDB table to keep track of the count, which Nihal pulled from in the backend. He and Abhinav also edited the video together.

Nihal was involved in connecting the Raspberry Pi to the ultrasonic sensors. He also worked on the ultrasonic sensor logic and code which involved continually sending pulses from the ultrasonic sensors and recognizing a change in measurement of distance to the opposing wall. He also worked on the backend of the website which first involved using asynchronous Node.js functions in order to get the data from the DynamoDB database. Next, he created an express application and used modules including HandleBars in order to display the counter data on the website.

Signatures:

				
Abhinav Balasubramanian	Daniel Larkin	Dhyani Patel	Kieran Hulsman	Nihal Menon

Final Product Evaluation

The final product was a reasonably accurate representation of the project proposal and initial idea. The project appropriately addressed the goal of keeping track of the amount of people in the SE lounge at a time. Every aspect of the project proposal: data collection, data transfer, and displaying the data on a website worked out as intended although there were changes in the approach to achieve them. For instance, we initially proposed using React in order to structure the website although this didn't end up being necessary in order to display the data on the website. Another area of slight difference between the final product and the project proposal was the initial lack of mention of the involvement of AWS. Yet, the role of AWS was quite crucial in transferring the data and invoking actions once the ultrasonic sensors had been triggered and the Raspberry Pi sent the data. The other aspects of the project including the ultrasonic sensor and thermal camera data collection worked out as initially proposed.

Design Trade Offs

Due to limitations with time and cost, we were unable to 3D-print an aesthetic protective case for the setup. As such, the layout of the Raspberry Pi with the ultrasonic sensors remains uncovered in its current state. One early decision we made was to do the processing of data on AWS as opposed to on the Raspberry Pi due to limitations of memory on the actual Raspberry Pi. In doing so, we were able to avoid potential issues with our Raspberry Pi being unable to handle the processing of data. Due to limitations with the number of Raspberry Pi's we were able to get access to as well as the number of ultrasonic sensors, we decided to limit the scope of the project to only include one door in the Software Engineering Lounge as it wouldn't have been feasible to place such as setup on both doors. In doing so, there are possible inaccuracies in the count but this is an issue that could be addressed in a straightforward manner with more resources.

Future Work

ForAllTimes has the potential to be a university wide resource for students in any programs looking to study in any lounge at any time. Moving forward we plan on expanding the project's range to all lounges on campus. To implement this we will install Raspberry Pi's into each lounge individually and expand our backend to be able to handle the new influx in data. To provide users with the best experience possible we aim to upgrade our website into a multi-page site with different pages for individual lounges, which contain the description of the lounge, its location, a live counter and a graph representing the traffic within the lounge over the course of the day. Additionally, we wish to integrate a deep learning model within the website that uses historical traffic data to predict when a certain lounge will be busy or empty. For the convenience of students and all users the website will have a similar app implementation that will allow users to check the traffic of lounges on the fly from their phones. We understand that there may be discrepancies between the counter and the amount of individuals actually in the room and therefore we are creating a report bug section to allow users to inform us on any bugs, issues or discrepancies they might run into while using ForAllTimes. Technically we hope to upgrade the thermal cameras and ultrasonic sensors. Upgrading the thermal camera will allow us to increase the detection range and accuracy. While upgrading the ultrasonic sensors will increase accuracy but also prevent them from burning out after constant use over a long period of time.

References

Bi-Directional Visitor Counter Based IOT and Raspberry Pi with Voice ...

<http://www.ijaema.com/gallery/35-july-2019.pdf>

“Embedding Google Maps.” *Google Maps Help*, Google,

<https://support.google.com/maps/answer/144361?hl=en&co=GENIE.Platform%3DDesktop#:~:text=Click%20Share%20or%20embed%20map,Click%20Embed%20map.&text=Copy%20the%20text%20in%20the,of%20your%20website%20or%20blog>

“Highlighting a Region within Google Maps.” *Google*, Google,

https://developers.google.com/maps/documentation/javascript/adding-a-google-map#maps_add_map-html.

Hrisko, Joshua. “High Resolution Thermal Camera with Raspberry Pi and

MLX90640.” *Maker Portal*, Maker Portal, 25 Aug. 2021,

<https://makersportal.com/blog/2020/6/8/high-resolution-thermal-camera-with-raspberry-pi-and-mlx90640>

“SMBUS2.” *PyPI*, <https://pypi.org/project/smbus2/>

Veeru. “How to Get Raspberry Pi to Interact with Amazon Web Services & Push Data into the Dynamodb.” *Electronics Innovation*, 27 July 2020,

<https://electronicsinnovation.com/how-to-get-raspberry-pi-to-interact-with-amazon-web-services-push-data-into-the-dynamodb/>