`

Computer Vision Truck Tracking Project

Cornell University

Dr. Zhang

MAE 4220 Internet of Things

Final Project Report

Katarina Duric (kd374)

Brooke Hudson (bah253)

Sabian Grier (srg294)

Tyler Wisniewski (ttw24)

Table of Contents

Executive Summary

The vision of our project was to create a truck tracking system which encompasses electronics that can successfully support the YOLO version 8 algorithm designed for tracking specifically trucks among other vehicles in traffic. Furthermore, our goal was to integrate the hardware into a waterproof and durable enclosure resistant to any weather conditions. What is more, the system ought to be able to support LoRaWAN capabilities, and transmit data wirelessly via LoRaWAN to The Things Network (TTN). The retrieved data from TTN will contain packets of vehicle identifiers, particularly the vehicle type identifier, which will differentiate between different truck types (for the purposes of our community partner's data interpretation): an assigned vehicle ID which will enable maintaining a count of vehicles and particularly trucks in real time, as well as a timestamp at which a truck was detected. The successfully transmitted data will only be provided to our community partner Blueprint Geneva, and they will be responsible for interpreting the truck traffic data.

The primary motivation behind this project was to help eradicate the issue of pollution in Geneva, which is impacted by emission of bad smells and dangerous chemicals, ultimately affecting the health of Geneva's residents. The environmental group has a dedicated website It Stinks, logging resident concerns in hopes to draw attention to the severity of the issue. Therefore, the primary challenge we have completed throughout this semester is demonstrating a functional system which can track truck driving by running the computer vision (CV) algorithm on the integrated electronics in real time, so that BluePrint Geneva has clear and digestible data on the fundamental pollutants. According to our last communication with Blueprint Geneva, the truck trucking system will be integrated with the rest of their air quality equipment, which will help determine the correlation between air quality monitor readings and our obtained truck traffic readings. Hence, our community partner will be able to gauge the air pollution within the City of Geneva with a more informed perspective.

The contributions we have made to the first iteration of this project are being able to transmit all the detected vehicles in traffic in real-time, by performing testing on the Cornell University campus. The camera focus is adjustable for a variety of distances, and the electronics are integrated in a robust and compact enclosure which ought to be able to resist any weather conditions. The system's design is adaptable to the ranges of local powerlines, and converts the input voltage to the desired 5 V for our system's operation. The current limitations of the project are the unknown powerline source, which is dependent on Blueprint Geneva's decision with regards to City of Geneva regulations. The next steps of the project ought to be performing fieldwork in Geneva, alongside our community partners, once the location is confirmed in order to ensure operation with the provided input voltage as well as the performance of the integrated enclosure and its electronics.

Social Context:

Blueprint Geneva is an environmental group raising concerns for the Geneva community, monitoring pollution around the City of Geneva. Landfills and the traffic necessary to transport trash can often have negative impacts on the community. The landfill itself decreases quality of living, such as emitting bad smells and emitting dangerous chemicals for people's health. The smell issues from these chemicals are also severe enough that the Blueprint Geneva website It Stinks collects complaints about bad smells around the Ontario County of Geneva. The emissions from the landfill and trucks that transport trash to them include CO_2 and nitrogen compounds (Kryder-Reid). These truck chemicals as well as those directly from the landfill and trash disposal are known to cause health issues and decrease air quality (Kryder-Reid). While these issues first impact people with underlying health conditions (asthma, etc.), over time the severity of the issue can grow to impact the greater community.

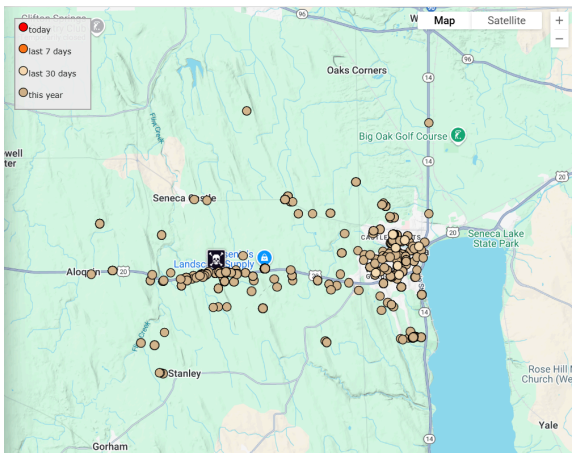

Figure 1: Logged complaints in greater Geneva area on the Blueprint Geneva's It Stinks website

These issues apply as well on a much larger scale. Landfills and such toxic gas emissions exist all across the country and the world, and therefore community monitoring practices are important for many communities. The BBC reported on health and smell impacts for a British community near a landfill within two weeks of this report (Burnell). In this real world example of the reported impacts of landfills on communities, many of the theoretical aspects can be seen in practice. The elderly and those with health conditions in this news coverage suffer first, while health issues ultimately impact the entire community. Devices used to monitor these damages and impacts can be used for benefits on a larger scale as well.

In order to prevent problems that decrease the quality of life for everyone, the community members must come together to monitor these impacts. There is a current solution with many shortcomings - high schoolers volunteer for Blueprint Geneva (Blueprint Geneva). They count the cars and trucks manually to estimate the emissions, potentially leading to estimations, as their daily availabilities vary. High schoolers are not available at all hours of the day, and so estimations must be made. Apart from the potential human error of measurements in a high traffic environment, different types of trucks can emit different gas emissions, and detecting

them without a robust system differentiating the trucks to recognize the severity of emissions is not a possibility with the current circumstances.

Hence, our team has devoted ourselves to work with Blueprint Geneva on designing a robust system, incorporating hardware that supports a CV algorithm for tracking trucks in traffic. By deploying our system in a high traffic Genevan neighbourhood, our community partners can use our data, along with their obtained air quality data, to gain a deeper understanding on the correlation between truck emissions in traffic with their air quality results. What is more, such a system would represent a crucial step towards eradication of harmful and damaging emissions and resolvement of many residential air quality complaints.

Overall, smells from the landfills, and emissions from the traffic are contributing to a rise in community issues for quality of life and health concerns. The current monitoring solutions are inadequate, inaccurate, and unable to measure the traffic at many important times when they can even be implemented. The creation of a technological solution, measuring the incoming traffic more accurately, efficiently, and most importantly, autonomously, would mean a great help to the community of Geneva. Hence, with the clear motivation towards helping the community, throughout this report, we will be demonstrating the successfully integrated and tested system for tracking trucks in traffic.

Refined project scope

IoT System - Vision, Design and Objective:

In order to reiterate the established project scope established from April Design Review, the goal of our project is to create a truck tracking system which encompasses electronics that can successfully support YOLO version 8 algorithm designed for tracking specifically trucks among other vehicles in traffic, as well as integrate the hardware into a waterproof and endurant enclosure resistant to any weather conditions. What is more, the system ought to be able to support LoRaWAN capabilities, and transmit data wirelessly via LoRaWAN to The Things Network (TTN). Based on the conversation with our community partners, we have established a system where the retrieved data from TTN will contain packets of vehicle identifiers, particularly the type of the truck which is assigned a specifier number from 0 to 13 (with 13 being the maximum numerical id of identifiable truck types), the confidence that the detected vehicle is a truck, and a timestamp of when the vehicle was recorded on the TTN Live screen.

Requirements:

The project scope expanded while working on the project since some software which we were given was ultimately not fully compatible with the hardware (although the hardware was chosen to be *mostly* compatible with some changes). Furthermore, some of the aspects of the project remained uncertain, such as power supply specifications, and precise location of where the completed project unit will be placed, due to the fact that there is a lack of communication about the specifics from our community partners at Blueprint Geneva. Through conversation with professors and teaching assistants, we have received approval to proceed with the project assuming that there will be power supply from the city powerline, as well as that there will be a gateway nearby to which we can connect to for the purposes of LoRaWAN data transmission.

Our project scope includes running the trained CV algorithm on a camera that can be used to track traffic, sending the data back to Blueprint Geneva for analysis using LoRaWAN, as well as weather-proofing the device using an enclosure. Our project scope was further modified to make adjustments to the CV algorithm for truck tracking provided by the professor's PhD students Jintao Gu and Mike Liao. The scope further included debugging of software in order for it to be compatible with our hardware and get it working. The ways in which we have debugged our software and made adjustments are further discussed in the *Technological Development* section.

Our project scope does not include training the actual YOLO version 8 model. The algorithm behind tracking the trucks and detecting vehicle identifiers was performed by the Professor's PhD students and provided to us. The modifications made to the algorithm were specifically for the case of making the software compatible with our project's hardware. Furthermore, interpreting truck traffic data is outside of the project scope, the data will be transmitted via LoRaWAN and provided to our community partner Blueprint Geneva for analysis. Lastly, the original design requirements and the original response from our community partner stated that a power outlet would be available, so power supply was not considered. Based on the statement from the community partners that a powerline will be provided, our design counted towards down converting the powerline voltage, which is further described in the *Technological Development* section.

Community Partners Feedback

Lastly, apart from the previously described community partner engagement, our community partner – Blueprint Geneva – has shaped our project in a sense of providing us with the vehicle identification information they would like us to transmit over LoRaWAN, as well as the fact that the system ought to be functional without any WiFi connection, making it adaptable for integration anywhere in Geneva, New York, which is why our team decided to pursue LoRaWAN functionalities of the Feather Board, as has been taught in lecture. Based on our most recent communication with the community partners, the data collected from this project will serve as additional information for their air quality monitoring program, hence our project will be correlated with their own air quality equipment for better analysis of the truck traffic impact on the pollution.

Technological development

Overall Hardware Diagram and Enclosure Diagram

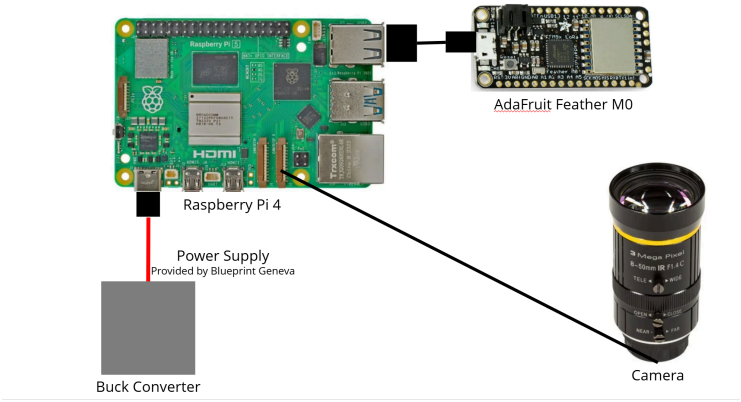

Figure 1: Updated Diagram of Electronics required for running the CV algorithm and wireless LoRaWAN Transmission

Justification for Sensor Selection and Components Selection:

Camera and Lens Selection : *Arducam for Raspberry Pi HQ Camera Module, 12.3MP IMX477 Raspberry Pi Camera with Arducam 8-50mm C-Mount Zoom Lens for IMX477 Raspberry Pi*

Our chosen camera has adjustable focus and zoom. Our community partner has indicated that they intend for our unit to be used in multiple locations. The adjustability of this camera makes it versatile enough to be placed around the city of Geneva and collect valuable traffic data. The camera allows for manual adjustment and aperture of F1.4, which makes it ideal for tracking even at very high distances. Throughout testing the camera focus, we determined that it can capture distances as small as 0.2 m, and as large as 20-27 meters with the adjusted focus. Furthermore, when it comes to the field of view, the camera has 45 degrees at wide 8 mm focal length, and telephoto 5.35 degrees at 50 mm focal length.

Raspberry Pi : *Raspberry Pi 4B 8GB RAM 4564*

Thus far, a significant element of our design effort was attempting to debug the YOLO CV algorithm on our Raspberry Pi Zero 2W. Ostensibly, the algorithm was not supposed to be very compute intensive but the onboard 512 MB is not a lot. To help identify the source of our YOLO issue, we tested the algorithm on a Raspberry Pi 4B which helped us identify the challenges we were experiencing where a function of the Pi Zero's onboard ram, YOLO CV algorithm, needs ~450 MB of available ram in order to run. The 4B currently runs the CV algorithm, processing frames from the video feed with YOLO v8. With our pivot to the new board, we are able to process images in seemingly real time.

AC-DC Buck Converter: *Output 5V, 85~264 V VAC, 120 ~ 370 VDC input 1866-4133-ND*

This AC-DC converter converts alternating current we will be obtaining from the local powerline, which ought to be 120 V according to the North America standards (and fits in the proposed input range of 85~265 V VAC) and converts it into a stepped down DC voltage of 5 V, which ought to be the input voltage for the Raspberry Pi 4. The buck converter has a live (L), neutral (N), and ground (GND) ports we can connect to the powerline through an AC power cord, where each of the three wires will be connected to the city of Geneva's powerline once the project is handed out to the community partners. Since we use the power supply from the powerline, we do not need to worry about power management and consumption.

Adafruit Feather Mo with RFM95 Lora Radio *3178*

The Feather Mo RFM95 Lora Radio microcontroller is the same board used for the training labs at the beginning of this course. Based on our initial design review feedback, we decided to pivot away from the Adafruit LoRa radio module due to potential incompatibility issues with the Raspberry Pi. The design choice to use the microcontroller simplifies The LoRaWAN and TTN communication by allowing the use of the code infrastructure built for the training labs to handle the IoT component. By plugging the Feather board into the Raspberry Pi via USB, we can both power the board and achieve simple and seamless data transfer using serial communication. After receiving data via serial communication, we will then send our data to the nearest gateway and upload it to The Things Network using the procedure we used during training labs.

Enclosure Box: Amazon Electrical Junction Box *B0B87V7QTH*

Given that enclosures already exist for the environments relevant to our use case, it was far more pragmatic to select one that meets our requirements and specifications rather than design a new one that might prove unsuitable. We selected an Electrical junction box with a clear front enabling our camera to collect video footage and a backplate with mounting holes allowing us to securely attach our electrical hardware.

Manual External Switch Button: *Amazon Pushbutton Switch B0DSFMWQL5*

We are using an external switch button for our power connection, which will switch power to our electrical components inside the enclosure. It will be connected in series with the line or neutral inputs to the buck converter. We drilled a small mourning hole that enables the button to sit flush with the outside of the box so power can be switched easily without opening the enclosure.

Wireless Communication and System Software Integration:

CV Software

The software changes were based on the script that runs the trained YOLO model. Additionally, the YOLO model itself was chosen now that there was hardware to test. First the YOLO model itself was changed. The Raspberry Pi Zero 2 W was unable to handle version 11, and there were too many outdated dependencies in version 5. Therefore, version 8 was used, which was not outdated like version 5 and was less computationally expensive than version 11.

In addition to choosing a reasonable YOLO version for the model based on hardware constraints, the script had to be made to process images using the trained CV model (made with YOLO v8). This Raspberry Pi uses libcamera (this is because it uses the bookworm Raspberry Pi OS instead of the older bullseye Raspberry Pi OS). OpenCV is used to analyse the images, but OpenCV is not compatible with libcamera (ie, the cv2.VideoCapture(<camera number>) command will not work, even though it is compatible with older Raspberry Pi models that use Legacy Camera Stack, which is compatible with OpenCV instead of libcamera. Since only code that was compatible with Legacy Camera Stack was provided, changes were made to allow communication between the camera and OpenCV (all OpenCV commands are referred to as cv2.<command>). The following function was defined to read the camera frame *using libcamera*:

```
Import subprocess
Import numpy as np

def get_frame():
        cmd = [“libcamera-jpeg”, “-n”, “--output”, “-”]
        proc = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.DEVNULL)
        jpeg_bytes = b’ ’
        while True:
                s = proc.stdout.read(1024)
                if not s:
                        break
                jpeg_bytes += s
        frame = cv2.imdecode(np.frombuffer(jpeg_bytes, dtype=np.uint8), cv2.IMREAD_COLOR)
        return frame
```

To use this function, instead of ret, frame = cap.read() from the old script, frame = get_frame() is used instead. The frame is then passed into the OpenCV functions as before. In case changes were made by future student groups, this adaptation was made very flexible. There is a flag new_rasos = 1 at the beginning of the file. If a student group in the future uses a different Raspberry Pi that uses the Legacy Camera Stack instead of libcamera, this flag can be changed to 0 and will use the old OpenCV functions that are compatible with that system.

When it comes to testing of the CV algorithm in real time with the integrated hardware setup, we opted for performing preliminary testing through focusing the camera lens on different truck images, and detecting if it is able to recognize the type of the truck, the probability of the truck being recognized. For instance, the image used in this case was an image of a blue-white semi-truck that was shown to the camera lens from a phone screen. As can be seen from the picture below (Figure 2), the probability of the detected truck was only 0.47, due to the image being blurry.

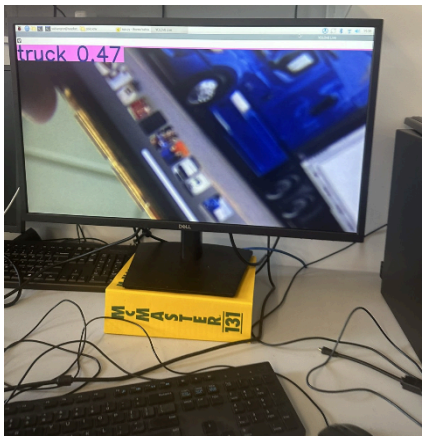

Figure 2: Initial testing of the CV algorithm running in real time with the integrated system. The lens were too focused, hence the probability of a truck detection was only 0.47.

What we concluded throughout the experiment was that the CV algorithm had the highest probability of detecting the truck once the lens was oriented in the correct direction, and laid horizontally, as shown in Figure 3. The CV algorithm was able to achieve a high probability of truck detection of 0.92, which is much larger than the probability detected in the initial testing, and the type of the truck was correctly identified and color-coded to be pink.

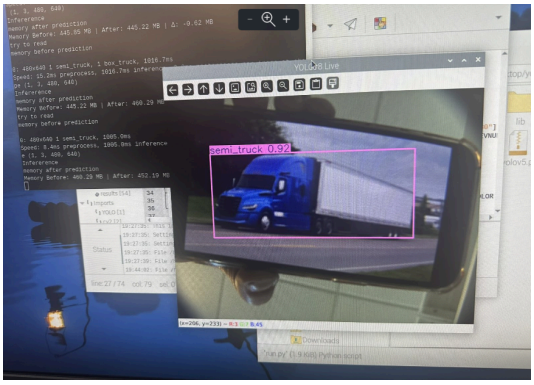

Figure 3: Further testing of the CV algorithm running in real time with the integrated system. The lens were correctly adjusted and oriented, hence the high probability of 0.92 was detected by the algorithm.

As can be seen from Figure 4a and 4b below, the CV algorithm's accuracy has a high dependence on the orientation of the camera lens, where it can inaccurately categorize the truck type have a much lower confidence in the probability of that type being detected.

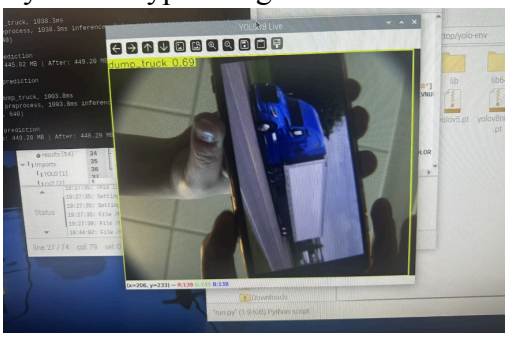

Figure 4a (on the left): The CV algorithm detecting a car due to the incorrect focus and orientation of the lens. Figure 4b (on the right): The CV algorithm detecting a dump truck with a low probability due to the orientation being inaccurate by a 90 degree shift.

Another significant aspect to note, is how much memory the algorithm exhausts when running it on Raspberry Pi 4 in real time. As can be inferred from the experimental images above, the CV algorithm was run on Raspberry Pi 4, and Figure 3 denotes how much memory each of the frames takes up out of the total RAM memory in real time. Hence, it was determined that each calculation in the frame takes up about 10-15 MB of memory, and that the overall program takes about 400-450 MB in total to run in real time. Therefore, when we tried to run and optimize the same algorithm on Raspberry Pi Zero 2 W, we were unsuccessful in doing so considering that it exhausts the maximum RAM the Zero 2 W possesses (with its total RAM being 512 MB as stated beforehand). The successful CV algorithm integration was only reflected in the Raspberry Pi 4, which is why we will be proceeding with it for the final integration.

On-Site Testing of the CV Algorithm

After completing the full electronics integration into the enclosure, we tested the CV algorithm's functionality in real time on campus location. In our initial test, we tested with stationary cars in the Upson Parking Lot. As can be detected from the images below,

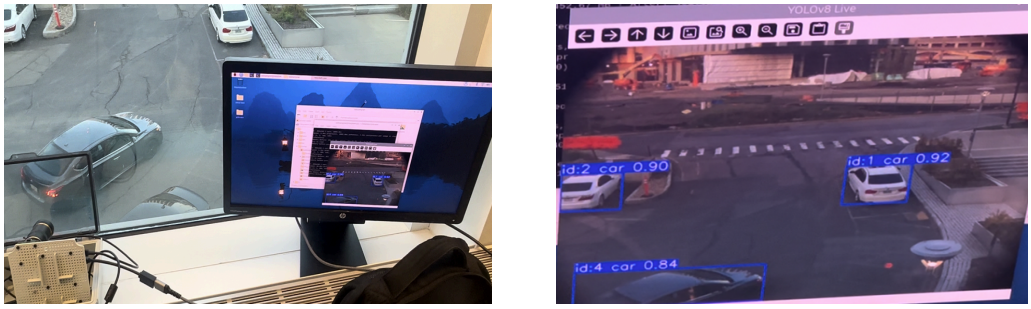

Figure 5a and 5b: Testing at the Upson Parking Lot location. Figure 5a on the left shows testing setup at the parking lot, while Figure 5b shows the monitor output of the CV algorithm running in real time

Our second testing location was at Duffield Hall. Even with testing for a long period of time, we were unfortunately not able to detect any trucks in real time, as none were passing by the campus. However, this test was very crucial, as it enabled us to detect all buses in real time.

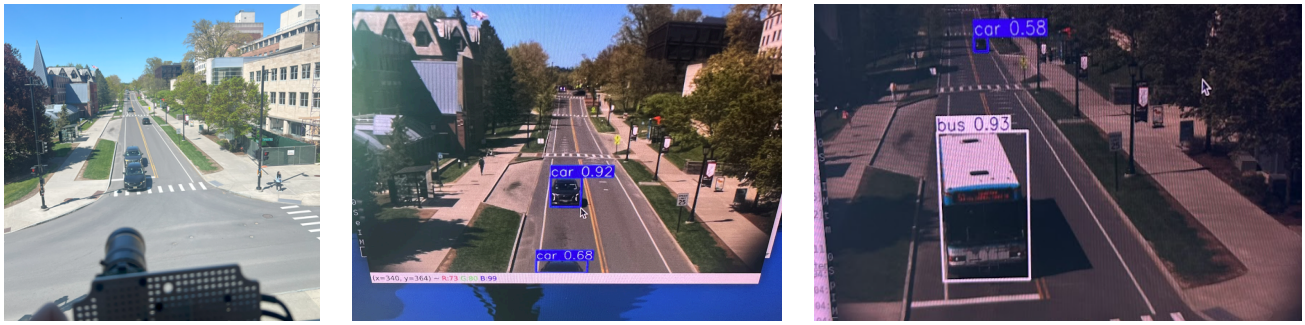

Figure 6a, 6b, 6c: Testing at Duffield Hall. Figure 6a on the left shows the testing setup, Figure 6b in the middle shows the algorithm running at moving cars in real time, and Figure 6c on the right shows the bus being detected moving in real time. All the vehicles detected are with high confidence displayed due to a well tuned and correctly oriented camera.

As can be inferred from the images above, with the right orientation of the camera and adjustment of the lens, the camera detects the vehicles currently in movement with a high confidence level.

LoRaWAN Software (and Raspberry Pi - Feather Board Communication)

The Feather Board handles all LoRaWAN communication. We achieved this by modifying and recycling the code we wrote for the training labs. As previously mentioned, using a wired serial connection, we can power and communicate with the Feather Board from our Raspberry Pi. By writing a simple function, we can search through the devices connected to the USB ports of our Raspberry Pi and identify our microcontroller. Once we are able to identify the Feather Boards's port, we open a serial connection on that port. From the Pi's standpoint, sending data is as simple as writing it to serial. On the feather side, we use Serial.begin() in setup with the same baud rate as specified on the Pi. We then perform a polling loop on serial, and when data is available, it will be transmitted to TTN with LoRaWAN using the same method we performed in training lab 5.

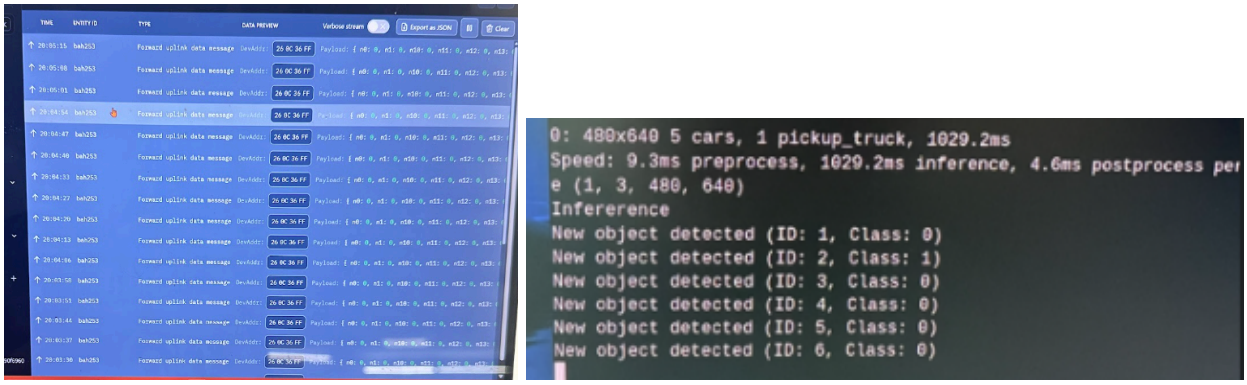

Figure 7a: Results from the TTN Live Screen of the data being sent through LoRaWAN
Figure 7b: Monitor Output of active vehicle detection

From the YOLO model our pi is able to collect vehicle ID denoting the vehicle type and a confidence which quantifies how confident the model is in its prediction. We needed to modify the model such that it is able to track vehicles through multiple frames. In the previous implementation, it would uniquely identify every vehicle in each new frame independent of whether that vehicle was in the previous frame. In this approach, there would be multiple LoRaWAN transmissions identifying the same vehicle resulting in inaccurate traffic counting. In our discussions with our community partner, they have indicated that they intend to use our unit in multiple locations and cannot ensure there will always be a stable wifi connection which motivated us to use LoRaWAN as our communication protocol. Considering our relative difficulty gaining meaningful feedback from our community partner, we were unable to finalize a means of efficiently providing the data they are seeking. We made the decision to save the data to CSV format using the TTN MQTT method in lab 6 .

Enclosure and Sensor/Component Integration into the Enclosure

Figure 8: The enclosure (prototype sensing package) including the mounted AC-DC buck converter and the pushbutton switch. The picture on the right shows the blue pushbutton switch integrated onto the enclosure, as well as the black wire input enclosure for the incoming power line wires into the AC-DC buck converter.

The camera and lens are placed in the top right corner of the enclosure. This position avoids inconsistencies in the clear outer face of the junction box. The TICONN logo obscures the top left, while much of the bottom causes the camera to appear blurry due to residue from stickers used to cover this portion during packaging and shipping. Additionally, by placing the

camera higher within the enclosure, we achieve a much greater field of view due to its position relative to the ground. This is an important factor to consider since our final placement location is still undecided based on the correspondence with our partners at Blueprint Geneva. The Raspberry Pi is positioned in the top left corner of the enclosure since it must be near the camera.

Our Power Grommet and button are placed on the bottom of the enclosure. This ensures they are easily accessible, regardless of mounting position and volunteer height. The placement of these components on the bottom enables the enclosure to shelter them from rain and the elements. It's important to note that both these interfaces are waterproof, utilizing threaded nut interfaces and rubber gasket rings. Bottom placement is not a critical design choice but instead a redundant safeguard, which is a good contingency since heavy water damage would result in catastrophic failure of the system. The Buck Converter is placed in the lower right, close to the powerline and switch. The Featherboard M0 is on the bottom left because the space is available, and the LoRaWAN signal does not care about the clouded enclosure face.

Legal Requirements

It was important to make sure that the video capturing complied with NY state law. Video capturing is legal when there is a clearly visible sign that there is video recording in progress and/or the video surveillance system can be clearly seen [1]. The video cannot also take place where there is a "reasonable expectation to privacy (such as a bathroom or changing room) [1]. This means that the surveillance is legal in a public place, such as a public roadway, and that there may need to be a sign about the recording if the surveillance camera does not appear to be obvious to the average viewer.

Community Partners involvement in the Technological Development

While our community partners have clarified their needs and how their project will integrate with the rest of their air quality equipment, they did specify any requirements for our system. Hence, we built our design solely based on our team's decisions.

Timeline and organization

Task	*Due Date*	*Completed*
Ordering all required parts and completing the Preliminary Design Report	4/7	✅
Receiving materials	4/14	✅
Milestone 1: Assembling the Hardware and Testing Serial Communication		
Ensuring validity of individual components	4/20	✅
Assembling the components		
Testing serial communication between the devices		
Milestone 2: Getting YOLO to work on Pi Zero 2 W using the provided video		
First attempt of integrating YOLO on Pi Zero 2 W	4/20	✅
Second Design Review	4/25	✅
Integration of YOLO on Pi Zero 2 W Finalized	4/30	✅
Milestone 3: Raspberry Pi Communication with Feather board and LoRaWan		
LoRaWAN Integration	4/30	✅
Milestone 4: Preliminary integration with enclosure	5/2	✅
Milestone 5: Complete prototype		
First On-Site Test	5/8	✅
Create User Instruction Manual	5/11	✅
Final Design Review	5/11	✅

Table 1: Updated Project Timeline and Milestones Table, with dates and completion indicated

Summary of Updates/Changes:
Following the April Design Report and the Second Design review, our group has been able to successfully integrate all the electronics into our waterproof housing, as well as perform thorough tests of vehicle detection in traffic on campus. As demonstrated in the *Technological Development* section, we have been able to detect all moving vehicles in traffic in real time, with a very high confidence of the correct vehicle being detected. The system was tested in two locations on campus, and ought to be tested with the community partners once they decide on the location with regards to the City of Geneva policies.

Work distribution:
When it comes to work distribution for the project from last design report to this design report, our group has been meeting during the lecture, and outside of class, and predominantly collaborated together on the majority of aspects of the project. Regardless, the initially proposed member responsibility breakdown has remained as the primary responsibility for each of the members. Brooke has predominantly been in charge of running and integrating the Computer Vision algorithm on Raspberry Pi 4 and Raspberry Pi Zero 2 W, with help and collaboration from other members. Katarina and Sabian have helped with software debugging, tested the system as well as LoRaWAN transmission, testing the serial communication among the hardware. Tyler has been in charge of layout of the components within the enclosure.

Plans for next steps

An important focus for future groups who may take on this project is more rigorous testing. As a new project, we don't have any past experience, data, or community partner feedback to incorporate into our initial prototype. Over this semester, we have extensively tested each feature, independently and as an integrated system to validate overall functionality. With the lack of a test site from our community partener we have no knowledge of its long term performance in its intended environment and under harsh weather conditions. This semester we were able to deliver a final project that met all the requirements of the project scope, but rigorous and extended testing will help identify areas in which our first prototype can be improved.

Throughout the semester, we had difficulty communicating with our Community Partner at Blueprint Geneva and received no insight as to where our device will be located. With this limitation, we chose a camera with a variable focal length lens to make it suitable for a range of locations. If future groups have more precise information about their test location, they can design a device specific for that location because the adjustable camera introduces challenges. Because of the adjustable zoom and focal length, you are responsible for manually adjusting focus. However, to focus the camera a display must be used to observe the video feed to ensure that it is focused at the correct range. However for practical reasons, there is no display integrated in our enclosure making focus calibration a challenging process once deployed.

In our design we encountered a number of barriers. First was our limited communication with our community partner. Their limited correspondence hindered our ability to gain a strong understanding of their needs, limiting our ability to design a system specific for their use case. In our system we are not allowed to store or save video footage due to legal regulations. With this being the case, without someone physically monitoring the video feed, there is no way to validate YOLO model performance. This semester our group worked under the assumption that we will have access to an outlet for power and a LoRaWAN gateway based on what we were told by Blueprint Geneva. However, if this is not the case our device is unusable in certain test locations.

For future groups, we recommend performing testing in the deployment location if possible. Furthermore, if the unit is located in only one location, they should select a camera specific for that application to mitigate camera focus challanges. Lastly, they should confer with Blueprint Geneva to understand how researchers and volunteers want to access the traffic data and what elements of the data are important for the analysis. At the end of the semester we had a conversation with Mike Liao where he mentioned he had found a workaround that made YOLO functional on the Pi Zero 2W. If it's possible to pivot back to that platform, that would result in a meaningful decrease in the bill of materials. Additionally the choice of a cheaper camera would mean significant cost savings. Moving in this direction would result in a smaller and cheaper device making it more financially sustainable and more conducive for mass deployment at scale.

Works Cited

[1]https://recordinglaw.com/united-states-recording-laws/one-party-consent-states/new-york-recording-laws/