Bus Passenger Counter with Analytics & Dashboard Visualization

ECE4872 Senior Design Project

IntelliBus
Project Faculty Advisor, Dr. Vijay Madisetti

Shadman Ahmed, CMPE, sahmed85@gatech.edu
Noah Chong, CMPE, nchong8@gatech.edu
Yue Pan, CMPE, ypan331@gatech.edu
Thomas Talbot, CMPE, ttalbot7@gatech.edu

Submitted

December 13, 2021

Table of Contents

Executive Summary

The IntelliBus system is an IoT device and software suite that tracks bus location and the aggregate number of riders on buses. The ongoing pandemic has shown that human congestion and high traffic on public areas and transportation can lead to higher virus transmission and exposure. Our team's solution brings real-time passenger data into the hands of public transit authorities that helps reduce costs and increase public transit efficiency.

The IntelliBus system solves the human congestion and traffic problem through an IoT ecosystem comprised of a microcontroller with sensors connected through cellular LTE-M networks and a AWS cloud-based web application that analyzes and visualizes incoming sensor data. Each IoT module costs $161.00 per unit with an expected $30,915.00 cost of labor to install, design and develop the IoT ecosystem. Performance is based on the accuracy and latency of both the IoT devices and web application. Acceptable tolerances are ±5% and 5 minutes, respectively. The IntelliBus system has shown to be well within those tolerances in both small-scale testing and real-world demonstrations on Georgia Tech buses.

Solutions similar to the proposed project exist outside of the United States, including China-based "Beijing Bus", that displays public transportation capacity via a mobile app. Real-time bus data has measurable benefits outside of the pandemic as well, including optimization of public transportation fleet schedules and valuable meta-data that could be applied towards better city planning, infrastructure, and economics. Integration with a larger traffic authority, such as Georgia Tech PTS or MARTA, would be the logical next step for the IntelliBus project.

Nomenclature

APC (Advanced Passenger Counting) - usually refers to the sensors, embedded computers, and network connections that make real-time people tracking possible in a public setting

AWS (Amazon Web Services) - a subsidiary of Amazon that offers cloud computing platforms and application programming interfaces to individuals, governments, and businesses

CapEx (Capital Expenditure)- spends to buy, maintain, or improve fixed assets, such as servers or equipment.

CPU (Central Processing Unit) - the portion of a computer that fetches and executes instructions

GPIO (General-Purpose Input/Output) - uncommitted digital signal pin on an integrated circuit that has no predefined purpose and is unused by default

GPS (Global Positioning System) - satellite-based radionavigation system owned by the United States government

GUI (Graphical User Interface) - a user interface that includes graphical elements, such as windows, icons and buttons

HTTP (Hypertext Transfer Protocol) - application layer protocol designed for communication between web browsers and web servers

IoT (Internet of Things) - a network of interconnected devices – from simple sensors to smartphones and wearables

IC (Integrated Circuit) - a set of electronic devices integrated onto a small piece of material (usually silicon) to achieve a certain function

LiDAR (Light Detection And Ranging) - a remote sensing method that uses light in the form of a pulsed laser to measure ranges and distances

LTE-M (Long Term Evolution for Machines) - low power wide area technology that enables a wide range of cellular devices and services

MARTA (Metropolitan Atlanta Rapid Transit Authority) - the principal public transport operator in the Atlanta metropolitan area

ML (Machine Learning) - is a method of data analysis that automates analytical model building

noSQL (Non Structured Query Language) - provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

OpEx (Operational Expenditure)- is an ongoing cost for running a product, business, or system.

PERT (Program Evaluation and Review Technique) - variation of the critical path method that examines tasks and dependencies to calculate the minimum time to finish a project

PIR (Passive Infrared) - a type of motion sensor that detects motion by sensing its higher infrared emission than the surrounding environment

PTS (Parking and Transportation Services) - the primary traffic authority at Georgia Tech

QFD (Quality Function Deployment) - method used to identify specific links between customer attributes and engineering requirements

RAM (Random Access Memory) - computer memory that can be read in any order - often used to store dynamic data and machine code

RTOS- A real-time operating system is an operating system intended to serve real-time applications that process data as it comes in, typically without buffer delays.

SDK (Software Development Kit) - a collection of software development tools in one installable package

SIM (Subscriber Identification Module) - a smart card inside of a mobile device that carriers an identification number unique to the user

TCP/IP (Transmission Control Protocol and Internet Protocol) - the two primary transport and network layer protocols that govern computer connections to the internet

USB (Universal Serial Bus) - industry standard that establishes protocols for data transfer and power supply between devices

Wi-Fi - a group of wireless network protocols based on Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards

Bus Passenger Counter with Analytics & Dashboard Visualization

1. Introduction

The team will design IntelliBus, an affordable solution for real-time people counting and data analysis system for buses. It is designed to aid transportation authorities achieve higher utilization of fleet resources and provide better service to their customers. The team requests $700 to prototype the IntelliBus.

1.1 Objective

The objective of IntelliBus is to create an affordable solution for bus authorities to track the number of passengers on board a certain bus route. Two main components are needed to implement this design.

First, the hardware end consisting of a sensor array, a microcontroller with built-in LTE-M and GPS support will be installed on the buses. The hardware end is responsible for using the microcontroller to process the sensor readings and calculate the number of passengers on board the bus. A software routine will periodically transmit the people count and current GPS location to the cloud server. To keep power and Internet data usage within a reasonable threshold, the system will transmit live people count data every ten seconds.

Second, the software end deployed on the cloud will be responsible for processing and visualizing the data transmitted from buses. Using cloud technologies, the team attempts to create a web application backed by a cloud-based database. Upon receiving updates from the hardware installed on buses, the cloud software will produce a real-time visualization of the changes throughout a bus's route.

1.2 Motivation

The team's motivation for IntelliBus comes from the current state of public transportation, which the COVID-19 crisis has tremendously altered. Regulations regarding the population density within confined spaces such as buses are in place globally. With decreased vehicle capacity and fluctuating demand, it is crucial for transportation authorities to revise their route schedules for better profitability, efficiency, and quality of service. Accurate data support is the key. A tool for analyzing and visualizing the number of passengers on board a specific bus will help decision makers understand passenger traffic along the routes and identify hotspots or underutilization. In non-pandemic times, IntelliBus can also provide general assistance to transportation authorities in optimizing fleet resource deployment, for example, evaluating the effectiveness of a new route.

1.3 Background

The need for people counting systems is prevalent. During COVID-19, stores can use it to limit the number of customers inside, and bus companies can use it to notify people how crowded an oncoming bus is. Traditionally, there are mainly two types of designs for a people counting system. The first type utilizes embedded sensors, such as PIR and LiDAR sensors, and is adopted in convenient stores and theft-prevention products. The PIR technology relies on interpreting a person's movement into an infrared source different from the surrounding environment. The LiDAR sensor reads distance measurements

between itself and the object in front. Given a certain threshold, an array of LiDAR sensors translate the distance measurements into movements before them. The second type uses video-based recognition technologies. This type can be seen in bus companies; for example, public transportation authorities in Beijing and Singapore display the crowdedness of buses on their mobile app. Adding an edge computing device to the already installed security camera on buses, companies like Beijing Transport can achieve real-time decoding of the video feed, thus producing a live count of the number of passengers on board. However, implementing the same technology may face problems in the US due to concerns regarding personal privacy, and potential costs of revising existing structures may jeopardize the low-cost intention of IntelliBus.

People counting systems are limited in value if the result is only shown locally to the driver or store owner. The connection between the hardware end consisting of microcontroller and sensors and the cloud is the core of IoT systems. In the recent decade, with the lowering cost of digital ICs and their growing versatility, IoT devices have been increasingly integrated into society as a stepping stone toward the future interconnected world. The emergence of IoT devices enables and inspires the design of IntelliBus. Alongside the hardware components, the cost of cloud service has also been dropping, with many providers such as Google Cloud and AWS offering free credits for users. The organization and properties of IoT systems make designs like IntelliBus low-cost and mass-deployable.

2. Project Description, Customer Requirements, and Goals

The goal of this project was to design and prototype an IoT device that will track bus location and the aggregate number of riders. The design includes two main components: an embedded passenger counting system and a cloud-based web application. The counting system contains a microcontroller, LiDAR sensors, and an LTE-M and GPS module to send data to the cloud service. A

pair of LiDAR sensors can be installed by the bus doors to capture passengers entering and exiting. The microcontroller processes the sensor inputs and calculates the aggregate passenger count. The LTE-M module transmits the current number of passengers and GPS coordinates of the bus to an AWS gateway as quickly as the network will allow.

The cloud-based web application consists of a noSQL database to store time-stamped passenger counts and a web application that provides dashboards to end-users. AWS IoT connects the cellular IoT device and the back-end noSQL database. A web dashboard actively refreshes with the latest passenger counts and bus locations from the database. Transportation departments and bus riders can analyze the passenger traffic statistics and bus maps on a web application that displays the dashboards.

In general, stakeholders are the people who will be affected by the project. They include people who have considerable influence over the project or those who are interested in its outcome. High power, highly interested stakeholders will engage the most with the project. The majority of the project effort will be to meet the goals of the project advisor and team members. High power, less interested stakeholders are heavily invested in the project's outcome but do not need to be notified of all the details. Capstone Design Expo judges and state and local governments will influence the project's success, and it is essential to keep them satisfied. Low power, highly interested stakeholders need to be informed about the project's progression and help with technical and marketing details. Regular communication with transportation departments, bus passengers, and competitors can help the project avoid significant problems. Low power, less interested stakeholders should be monitored occasionally, but regular communication is not necessary. It will be moderately important to consider the project's impact on the Georgia Tech Community, prospective customers, the media, and future recruits.

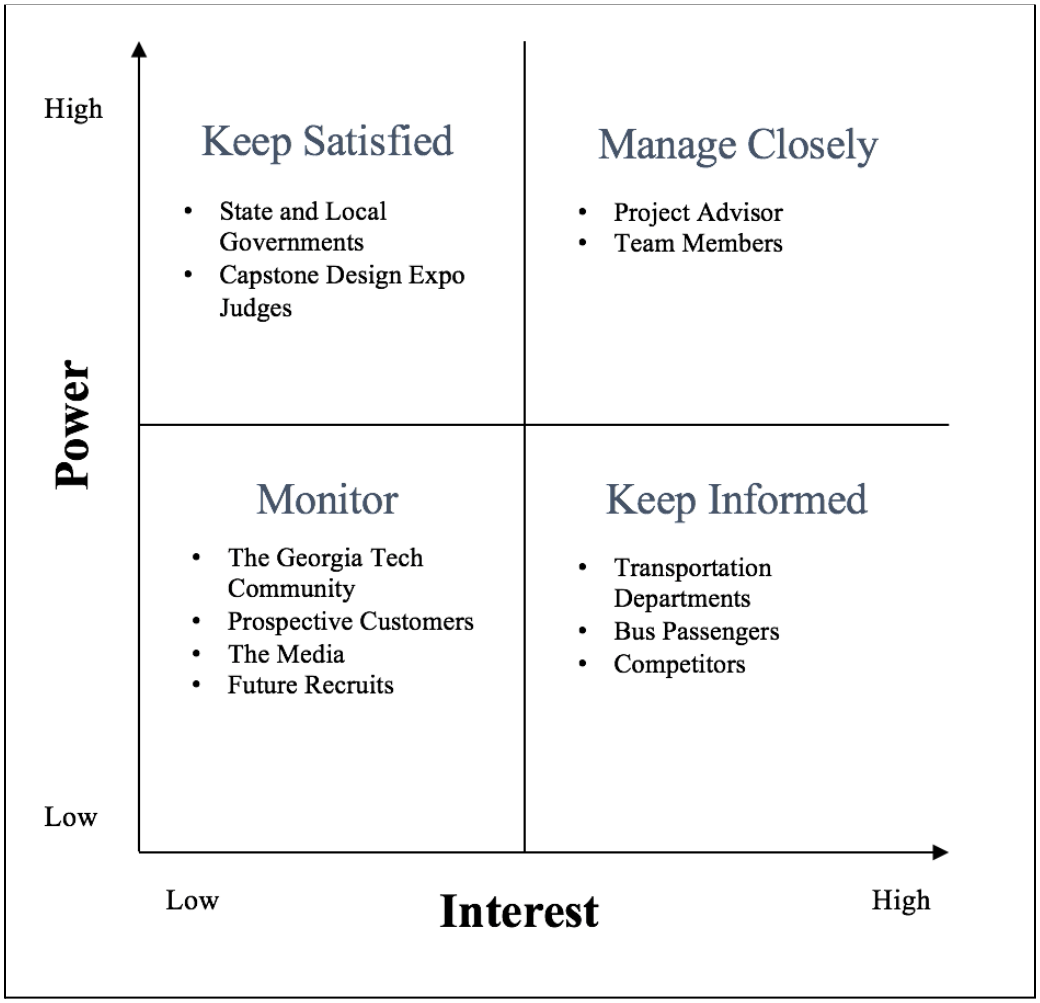


Figure 1. Stakeholder "2x2" Chart.

The target user of the product is transportation departments that provide bus services. The IoT device should use an existing bus power supply and be installed in under four hours. To be implemented across an entire bus fleet, the device should cost no more than $800. Passengers accessing the web application for aggregate rider counts and location services should receive updates within 5 s. The customer needs for the project include the following:

Customer Needs

- Accurate display of passenger counts and bus location
- Total cost less than $800
- Charging capability

- Setup IoT device and connect to cloud service in under 4 hours
- Fit inside the doorway of a bus
- Timely web application response

The product will fail if it cannot respond to user requests in a punctual fashion. There are two potential network bottlenecks within the proposed system. Between the IoT device and the cloud database, there should be a minimum end to end throughput of 5.6 kbps to allow enough time for data processing. Transportation departments and passengers should receive HTTP responses from the AWS virtual machine within 5 s. Beyond timing performance, the device should easily integrate into a modern bus infrastructure. The embedded system should charge using a 5 V USB cable and should not hinder passenger movement. To be scalable, the total cost of the cloud microservices and hardware should be less than $250, and the back-end should support up to 15 IoT device connections. The engineering requirements for the project include the following:

Engineering Requirements

- Transmit from the IoT device to the cloud with a minimum throughput of 5.6 kbps
- Embedded system cost under $200
- Respond to thousands of HTTP requests in less than 5 s
- Total weight should be less than 256 g
- Total area of the embedded system should be less than 680 cm^2
- Support up to 15 device connections per node
- Charge using a 5V USB power supply
- Analyze passenger count data on a cloud service for less than $50

The relationship between the customer needs and engineering requirements were mapped into the Quality Function Deployment (QFD) Chart in Appendix A.

The target customer for this project is transportation companies with hundreds of buses. The per-bus cost of the product must be kept within an acceptable threshold to make it economically viable. Due to the project's limited timeframe, the team has decided to use infrared sensors rather than image tracking software to monitor passenger movement. The use of cameras to identify passengers and their traveling habits leads to privacy concerns. Such data collection cannot be justified for this project; thus, pivoting towards infrared, non-identifying sensors helps achieve the project's goals without fear of litigation. The embedded processor in the cellular IoT development kit contains 256 KB of RAM [1]. The limited processor memory will relegate data analysis and visualization to the cloud service. The embedded processor will do little more than receive sensor inputs, package the data, and send it to the web app over the LTE-M network.

The project's computer networking standards impose additional timing constraints. The data transfer between the IoT device and the cloud system will be limited by the 200 kbps maximum uplink throughput of the LTE-M network [2]. User HTTP requests will be restricted by the response time and end to end throughput of the underlying TCP/IP infrastructure. The constraints for the project include the following:

Constraints

- Per-Bus cost
- Accuracy of sensors
- Privacy concerns
- Throughput between IoT device and cloud system
- HTTP response time

- Embedded processor RAM

3. Technical Specifications & Verification

The two major components of the project include the embedded passenger counter and the cloud-based web application. The calculation of the aggregate passenger occurs in under 2 s to allow enough time for transmission to AWS. The relationship between the response time of the device and the customer requirements can be seen in the Quality Function Deployment (QFD) chart in Appendix A.

Table 1. Embedded Passenger Counter Specifications

Feature	Updated Value	Measured Value
Calculation of aggregate passenger count	< 2 s	20 ms
Sensor Housing Dimensions	< 680 cm^2	246.125 cm^2
Sensor Range	> 762 mm	800 mm
LTE-M Connectivity	600 MHz -2.5 GHz LTE Carrier frequency	2.4 GHz LTE-M
Arduino Nano Code Size	< 32 KB	3.72 KB
nRF9160-DK Code Size	< 1 MB	31.3 KB

Table 2. Cloud-based Web Application Specifications

Feature	Updated Value	Measured Value

Model Size	10 GB	10 GB
Live Map Update Frequency	< 5 seconds	3 seconds
Num. Supported Devices	> 15 connections/shard	1000 connections/shard
Analytics Dashboard Update Frequency	< 5 minutes	5 seconds

4. Design Approach and Details

4.1 Design Concept Ideation, Constraints, Alternatives, and Tradeoffs

System Overview

There are two main components to IntelliBus: the IoT embedded device and the web applications. The IoT device consists of a microcontroller, tracking sensors, and LTE-M module. The web application is responsible for providing a real-time dashboard of bus locations, passenger counts, and viewing pertinent graphical statistics and analytics. Figure 2 shows the overall system architecture of IntelliBus:

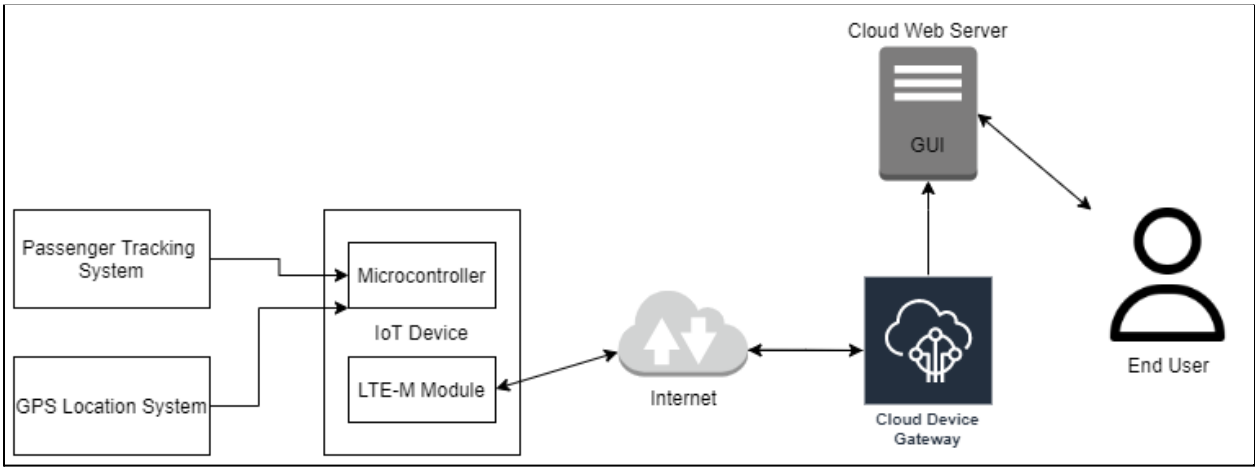


Figure 2. The overall system architecture of IntelliBus.

Passenger Tracking

Passenger tracking provides the count of the number of passengers on a bus. Passenger counts are acquired by tracking when passengers exit and enter the vehicle. The embedded software program calculates the aggregate number of passengers based on sensor inputs.

A pair of LiDAR sensors will be used to track passengers as they enter and exit the bus. The two LiDAR sensors are first connected to an Arduino Nano, which handles the pre-processing for the sensor readouts. The Arduino Nano reads LiDAR measurements and determines if the object is within the set threshold. Then, it forwards two digital logic signals to the microcontroller, indicating the triggered state for each LiDAR sensor. The order of when the sensors are triggered will indicate if a person is entering or exiting the bus, as shown in Figure 3.

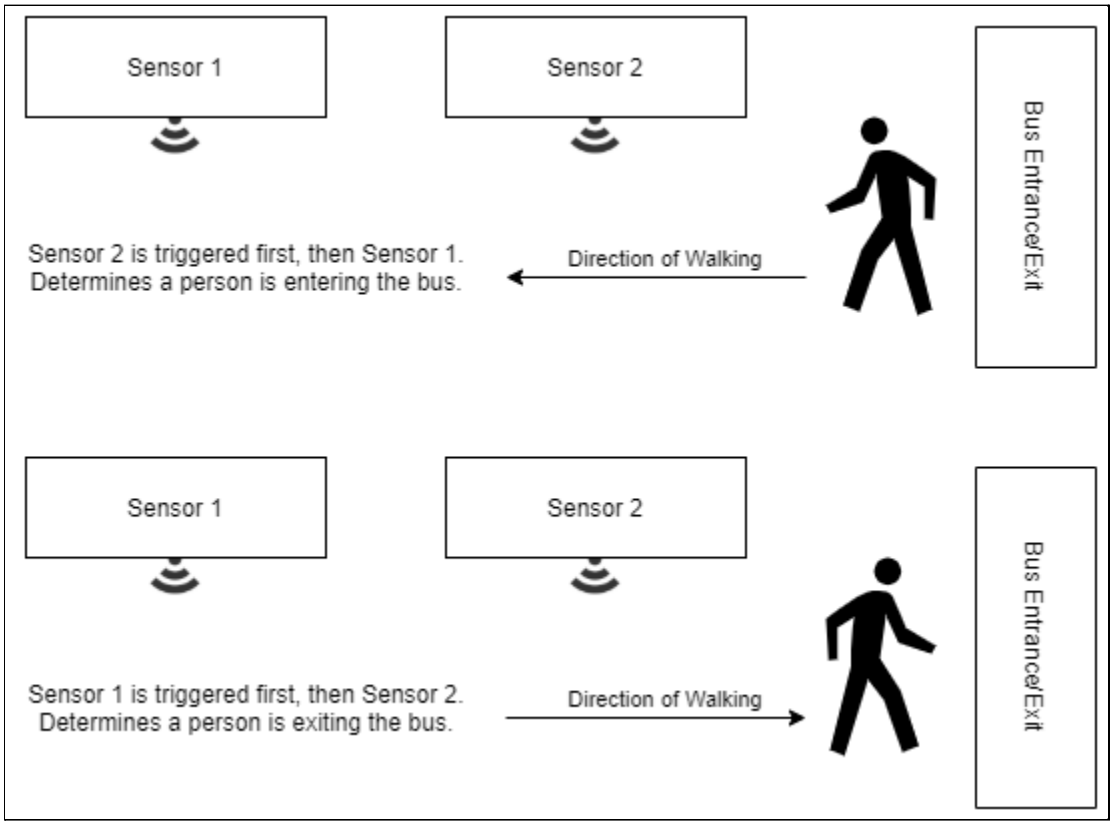


Figure 3. A pair of LiDAR sensors can determine if a passenger is entering or exiting the bus.

LiDAR sensors were finally chosen for their low cost, low power, and ease of setup at the expense of high accuracy. Most importantly, LiDAR offers the fastest respond time of all, able to accurately record a densely moving queue of people. Other sensor devices that were considered:

- Pressure Sensors were considered for their low cost and high accuracy. Pressure sensors work similarly to infrared sensors to figure out if a person entered or exited a bus based on which pressure sensor was triggered first by footsteps. However, pressure sensors did not meet the constraints for ease of setup and compactness. Pressure sensors require more wiring overhead and higher maintenance cost due to wear and tear from passenger footsteps.
- Camera Sensors: A video-based human recognition model was considered for its high accuracy and ease of setup since existing camera hardware exists in most buses for security purposes. However, ethical implications plague this solution for privacy concerns of running video feeds through Artificial Intelligence recognition models.
- PIR Sensors were first considered for their low-cost and ease of setup. However, as the team installed them on the first prototype, the team finds out that most PIR sensors will need a recovery time between each detection. In other words, we measured that after one trigger, the PIR sensors need around two seconds of time to be able to trigger for the next time. Considering a queue is highly likely to move with less than a two-second interval between each person, we abandoned the PIR approach and moved onto LiDAR sensors.

Location Service

Location services provide the real-time position of the bus through latitude and longitude coordinates. The bus location is sent over the internet to the cloud services that give the end-users real-time maps of the bus location and provide location-based data points for analysis.

A GPS module is used to gather the real-time location of the bus. There is a GPS module embedded in the Nordic nRF9160. GPS modules are highly accurate, low cost, low power, and easy to set up modules. No alternatives were considered as GPS is the de facto method for gathering location positions through satellite.

Internet Connectivity and Communication

Internet connectivity allows pertinent data gathered from the passenger tracking system and GPS system to be sent to the cloud services to process, view, store, and analyze real-time passenger counts and bus locations. Utilizing standard MQTT or HTTP protocols [3,4], data is sent over the LTE medium of communication.

An LTE-M module is utilized for bi-directional communication to the cloud device gateway with passenger count data and GPS location. The LTE-M module connects to the IoT device microcontroller using serial communication that passes data it receives and transmits. The module has its own microcontroller and firmware to control radio communications to the internet and provides a level of abstraction by communicating with the IoT device microcontroller through serial commands.

LTE-M communication is utilized for its high accuracy and high reliability but at a higher cost. The LTE-M module provides consistent connection and range to the internet due to 99.9% coverage of at least a 3G network in the United States [3]. Consistent internet connections are critical for an IoT device that produces real-time sensor data that needs to be processed. WiFi was considered as an alternative wireless network medium because of its low cost. However, WiFi restricts the IoT device's mobility. A WiFi-enabled device must stay within a fixed range of a base station to maintain a consistent internet connection.

IoT Device Microcontroller

The IoT Device requires a RTOS microcontroller board to facilitate central processing of all data to and from GPIO peripherals. The microcontroller contains software programs to receive data from the passenger tracking system and GPS module, which it uses to send data (and receive data) from the cloud device gateway through the LTE-M modem. The microcontroller utilizes standard GPIO signals and serial communications to gather input and output from I/O devices.

Device Gateway

Device gateways provide the way to consume IoT device data to the cloud for collection, analysis, storage, and near real-time updates to end-users. The Device Gateways maintain long-lived, bidirectional connections, which enables IoT devices to send and receive messages at any time with low latency. Device gateways are the integrator for communication between end-users and IoT devices. The Gateways use MQTT to receive and send data from IoT devices and uses a standard to process payloads sent through the internet.

AWS IoT services are used as the device gateway. All cloud providers have IoT services to handle tracking, communication, monitoring, and management of IoT devices in the field. The AWS IoT service provides an easy way to connect to other cloud services for web interfaces, database storage and data analysis. The AWS IoT services also has an extensive Software Development Kits (SDK), management and networking dashboards that allow for rapid deployment and management of IoT devices and their data.

AWS IoT services are chosen as the device gateway for their low cost, ease of setup, and ease of use. The cloud provides extensive savings by not requiring large initial CapEx investments and allowing OpEx spending by paying only how much compute resources are used. Cloud services are easy to set up from their web console and SDKs. They have extensive documentation and support for industry standards with no significant trade-offs for this project's implementation. All cloud services

are easily integrated and scaled to meet the demand for resources. With 99% uptime [4], redundancy, and scalability, cloud services are preferred over dedicated servers. Servers require maintenance and reduce agility in the development lifecycle and have a significant upfront investment.

Graphical User Interface (GUI)

The GUI allows intuitive interactions for users to view and perceive data requested. The GUI serves as the interface for passenger end-users to view live maps of the bus location and its passenger count. Transportation department end-users view statistical and graphical dashboards to analyze useful data from bus routes and passengers. Analytics GUI includes bar graphs for daily overall passenger counts, line graphs of passenger counts by time intervals.

A web interface is utilized to present users with the GUI. AWS is used to host the web application and easily integrates with the IoT service on the same cloud provider. The web interface utilizes APIs to communicate with IoT services and provides an accurate and safe method to present users with the IoT device data.

Hosting the web application on the cloud provides the same benefits as mentioned for cloud IoT services. Web applications also provide ease of use from any web-enabled device. Cloud services scale as the demand from users increases. Scalability is achieved by ramping up web and IoT services during peak times and ramping down at lower usage times to save money.

4.2 Preliminary Concept Selection and Justification

Passenger Detection System

An array of LiDAR sensors is the optimal choice for implementing the Passenger Detection System on IntelliBus due to its low cost, ease of setup, fast response, and ability to identify the direction of human traffic flow. The placement of LiDAR sensors is shown in the following figure.

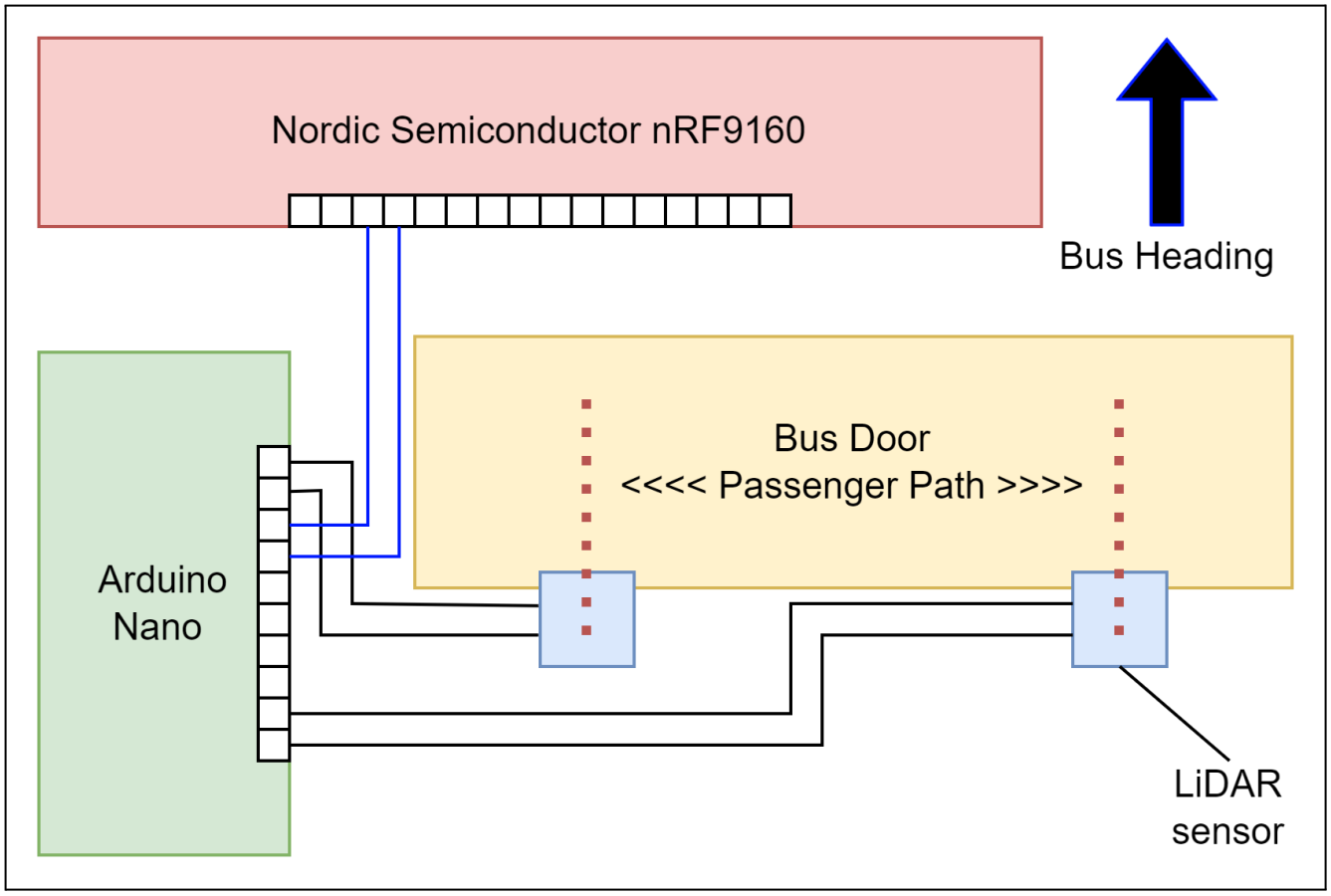


Figure 4. Layout of LiDAR sensors and microcontrollers

One sensor array consisting of two LiDAR sensors will be installed on the bus door. The LiDAR sensors will face perpendicular to the motion of passengers. When the Arduino Nano detects a LiDAR sensor's measured distance being less than the set threshold, it will send a logically high signal to the GPIO pin on the Nordic microcontroller. Using the timing difference on the two LiDAR sensors' triggering in a sensor array, the program will know which sensor is triggered first and distinguish between a person getting on and getting off the bus. With this setup, the team can develop a software program for keeping a count of passengers on board, and IntelliBus will work for buses whose doors are used for both entering and exiting. For buses with two doors, this solution is easily scalable by installing another sensor array on the second door, and only trivial changes to the counting logic will suffice the passenger counting goals.

The Nordic Semiconductor nRF9160DK

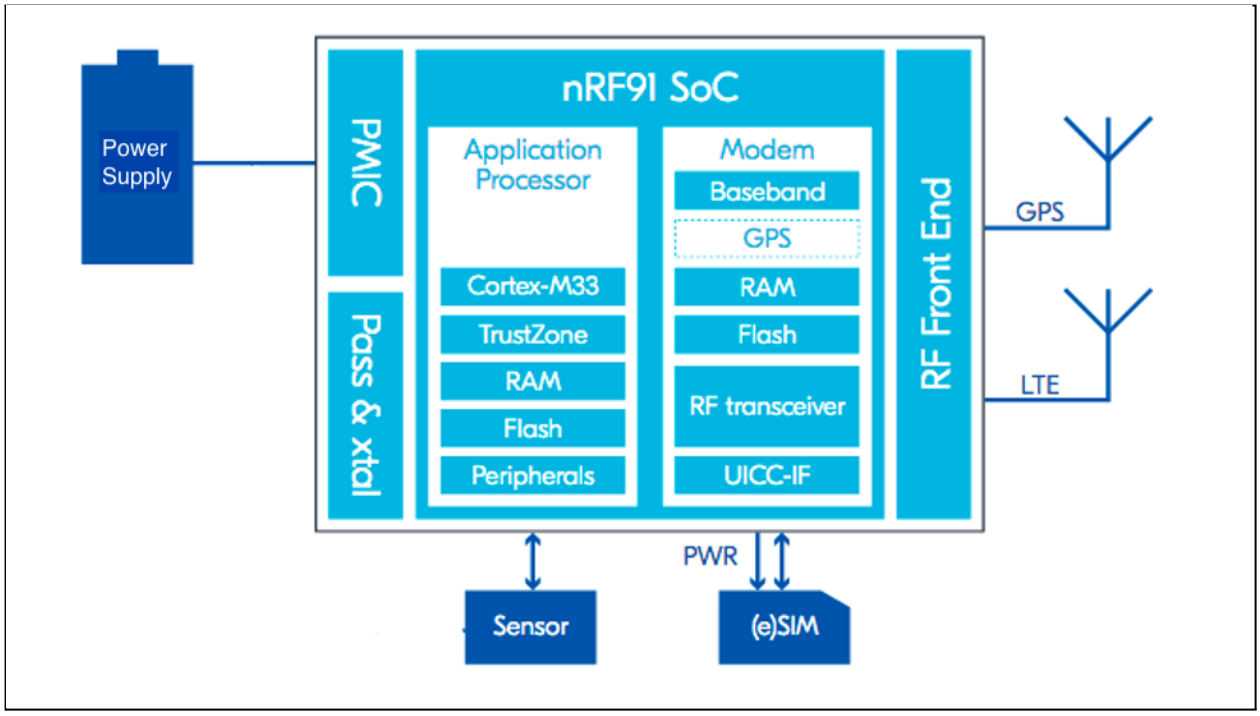


Figure 5. nRF9160 DK cellular IoT development kit with LTE and GPS modem [5].

The nRF9160 microcontroller is the optimal choice for the project because of its built-in LTE and GPS capabilities, application processor, and 24 GPIOs. The modem is optimized for global bus operations supporting all LTE frequency bands in the 698-960 MHz and 1710-2200 MHz ranges. For ease of development, the nRF9160 comes bundled with an iBasis SIM card preloaded with 10 MB of data [5]. Other commercial cellular IoT kits do not come with GPS support or data SIM cards and require additional purchases. The device requires a 3.0-5.5 V power supply from an external source but can also be powered using 5V USB. The application processor includes a 64 MHz Arm Cortex-M33 CPU with 10 MB of flash and 256 KB of RAM dedicated for the application, which are adequate hardware resources for the passenger counting program. The 24 GPIOs are configured to interface with the Arduino, or even directly with LiDAR sensors [1].

AWS Microservices

Amazon's cloud services, AWS, is the ideal choice for the project because it provides and manages an elaborate ecosystem to host all of our needed computational resources. The AWS web dashboard and SDK provides an easy way to develop and manage all applications. AWS takes advantage of a microservice architecture and promotes development in modular parts. These microservices are interconnected through APIs to make the IntelliBus's GUI. The overall AWS service architecture for this project can be seen in Figure 6.

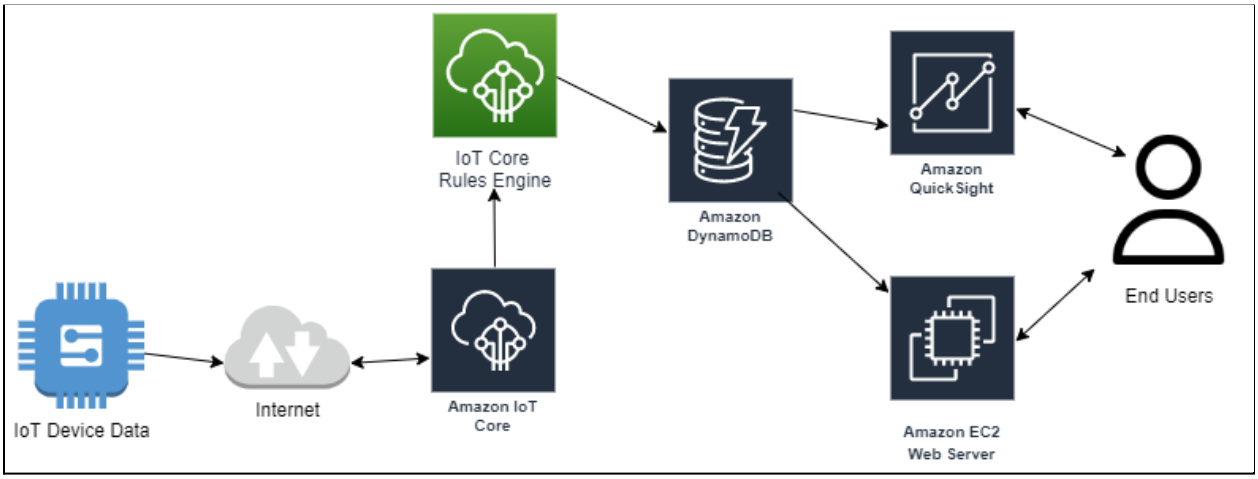


Figure 6. The AWS Architecture of IntelliBus.

AWS IoT Core service is the chosen device gateway for the IoT device connection. The AWS IoT Core provides a Rules Engine [6], software to process incoming IoT device data and forward it to Amazon DynamoDB. DynamoDB is a noSQL database that stores incoming device data in tables [7]. The DynamoDB table is then accessible to the AWS Elastic Compute Cloud (EC2) virtual machine. The EC2 instance is dedicated hardware to host the web application for the GUI [8]. The web application pulls from the database to present the end-users with live maps and passenger counts.

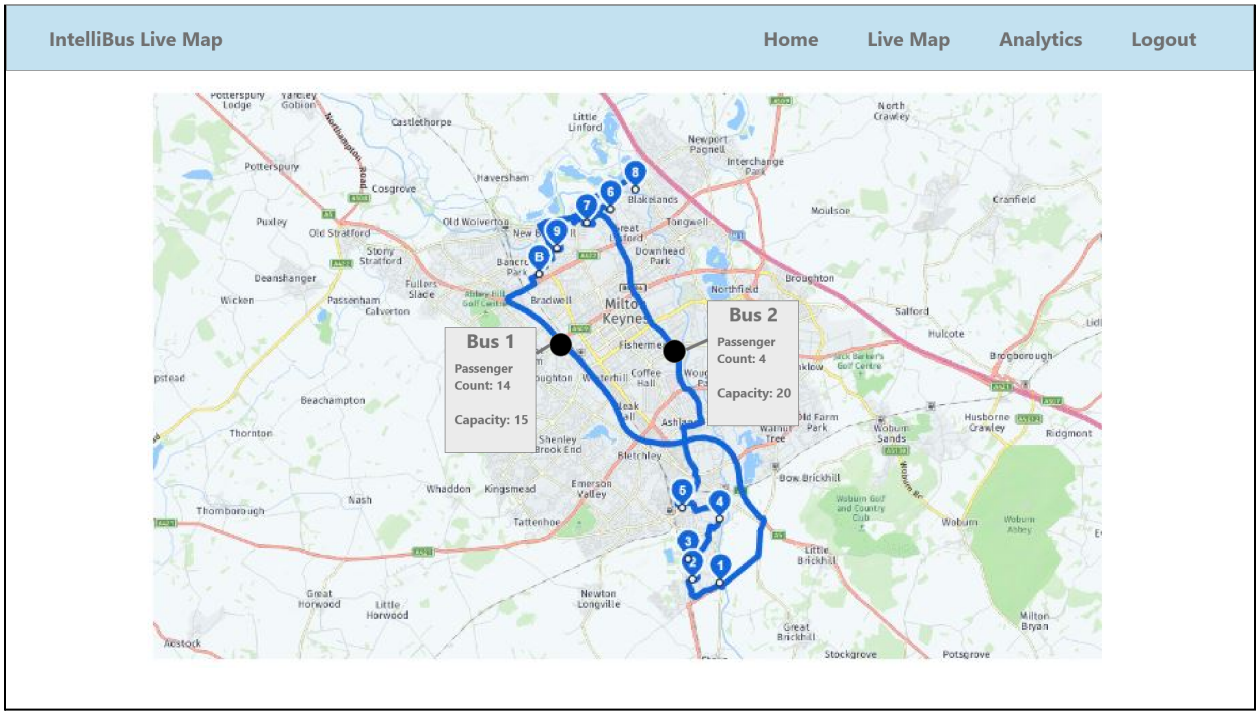


Figure 7. The proposed GUI for the live map and passenger count.

The DynamoDB table also connects to AWS API Gateway that provides web-embeddable GUI dashboards.

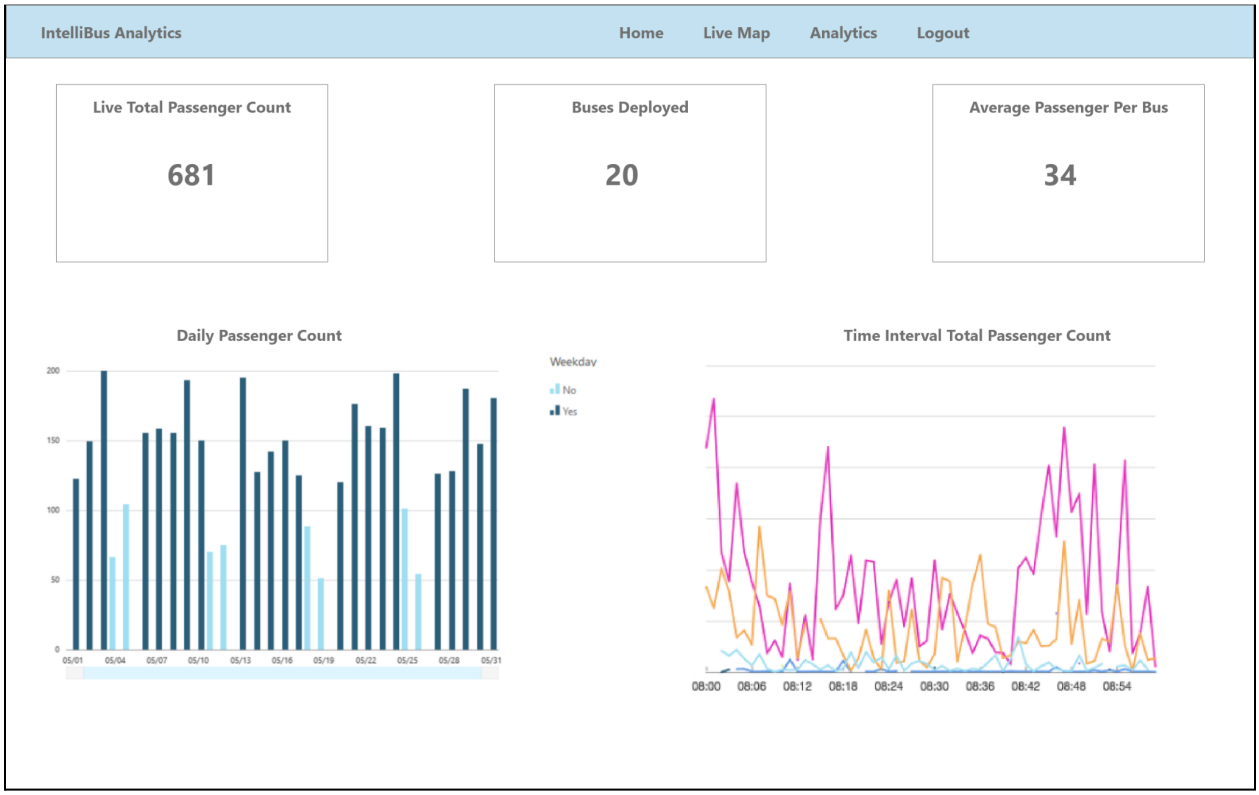


Figure 8. The proposed GUI dashboard.

Critical Path

The technical critical path for this project involved setting up cloud services, including the web app, servers and database, setting up the IoT hub and management software with the project's various devices, and developing and testing integrations between the web app and analytical software.

These key technical issues will be addressed early on in the design process. Integration between the various sensors and DK cellular devices with the IoT hub and cloud services will not happen in isolation. Instead, shortcomings in technical design were discovered as soon as both hardware and backend services were configured. Device connectivity and IoT communication issues, for example, were addressed after only a few weeks into the project to provide ample time to strategize and pivot towards a solution.

Contingency Plan

The team anticipated a low probability of failure in the project's various subsystems, as seen in Table 3.

Table 3. Contingency Plan

Risk	Probability	Response
PIR sensors comprise	Low	Resort to other options for motion sensing that may come at a higher cost, for example an infrared source and receiver solution.
LTE-M configuration failure	Low	Utilize WiFi modules to connect to AWS IoT core and send/receive data over the internet.
AWS Downtime	Very Low	Utilize Azure IoT and web services as the next best alternative.
IoT Device failure	Low	Contact Nordic Semiconductor with technical questions and inquire about receiving a refund or new device.

4.3 Engineering Analyses and Experiment

4.3.1 Prototype Testing and Analysis

Prototype testing is accomplished using unit testing, system integration testing, and user acceptance testing. These testing methods are standard practices for product and software testing and will provide a framework to validate this design.

4.3.1.1 Unit Testing

Unit testing is used to verify all software aspects of IntelliBus. This method ensures that software development is broken into modules and that each module is verified and tested for project functionality and constraints. IoT device unit testing will be used to verify individual I/O devices are sending and receiving data to the microcontroller as expected.

Web application unit testing involves dependency injections to validate individual software components independent of their external dependencies to other web applications parts or APIs. Dependency injections allowed for isolation of code and tests if its data control and manipulation are as expected.

4.3.1.2 System Integration Testing

System integration is essential in testing that all subsystems communicate with each other as expected. System integration was broken into two aspects: hardware integration testing and software integration testing.

Hardware integration testing ensured that all I/O devices have the physical capabilities to communicate with the microcontroller. Passenger tracking integration was tested and verified by ensuring GPIO signals are triggered when a person walks past it and is sent to the microcontroller with logic analyzers. GPS and LTE-M module integration can be tested with serial communication debuggers to ensure data is transferred to the microcontroller.

Software integration testing ensures IoT devices, cloud gateway, and web applications are able to function together to receive and share data. Software integration testing verified data is being sent from the IoT device through MQTT and received by the cloud gateway. LTE-M module's SDK provides console debug options to verify connections using network tests which provide network download, upload, and latency (ping). LTE-M configuration was tested to ensure it has upload speeds greater than 5.6 kbps (Appendix A).

AWS CloudWatch service provided debugging and event listening to ensure network packets are being delivered to the IoT Core's gateway. Event listening verified that the IoT device data is transmitted and received over the internet to and from AWS. AWS web application integration testing

was done by ensuring API responses from DynamoDB and GUI using AWS CloudWatch for network analysis.

4.3.1.3 User Acceptance Testing (UAT)

UAT is when actual users test if the software application can carry out its required tasks as it was designed for end-users. The UAT test adheres to the attributes of the design requirements and interviews end-users to gather feedback. This feedback improved and validated the product until all customer requirements were satisfied (Appendix A).

UAT tests all web application aspects from two groups of end-user types: average public transportation passengers and transportation department management. Regular public transportation passengers test the live map and passenger counts to ensure that the web GUI was intuitive and the map interface was accessible, convenient and error-free. The analytics dashboard GUI was tested by transportation department employees to validate that the data presented was customizable, practical, and contextual.

4.4 Codes and Standards

The most significant codes and standards that apply to the IntelliBus project include the following:

1. Universal Serial Bus (USB) is an omnipresent power socket for cell phones, MP3 players, and microcontroller boards. Features of USB power delivery include maximum power of 100W, bidirectional charging capability, power management across multiple peripherals, and low-power device optimizations [9]. During prototyping, the IoT development board will connect to a PC for power and data delivery over USB.
2. Message Queuing Telemetry Transport (MQTT) and Hypertext Transfer Protocol (HTTP). MQTT is an ISO/IEC 20922:2016 standard that leverages TCP/IP (or other network transport

protocols) to provide lightweight, open and simple publish/subscribe messaging transport [10]. HTTP is an RFC 2616 consensus standard issued by the Internet Engineering Task Force (IETF) in their published memorandum called Request for Comments (RFC). HTTP provides generic and stateless hypermedia information for virtually any digital data [11]. MQTT and HTTP are widely adopted and allow for rapid development and deployment to send and receive data from IoT devices. MQTT and HTTP allow for bidirectional data transfer between IoT devices, Cloud Gateways, Cloud Services, and end-users.

3. I2C is a prevalent serial communication protocol for embedded devices. It is highly scalable and low-power. The two ports, SDA and SCL, are for clock and data. First, the address of the slave is sent on the bus. Then, data on the I2C bus is transferred in 8-bit packets (bytes), and each byte is followed by an acknowledge bit. This bit signals whether the device is ready to proceed with the next byte. With an address-based data transfer scheme, one I2C master on the microcontroller can connect to more than 100 I2C slaves. For the project, the GPS module can be an I2C slave to the main microcontroller, and communication between them must follow the I2C protocol [12].
4. LTE-MTC low power wide area (LPWA), or otherwise known as LTE-M, is a technology that leverages existing mobile networks to provide long range, low power IoT connectivity [13]. Current LTE-M technology is developed privately by businesses attempting to capitalize on the growing IoT market. Standard organization 3GPP specifies a 1.08 MHz bandwidth and 1Mbps peak download/upload rate for LTE-M [14]. Mainstream cellular providers such as AT&T and Verizon offer competitive rates for LTE-M connectivity. During prototyping, we will connect the infrared sensors to the LTE-M network and send data wherever the device is located.

5. Project Demonstration

The team contacted the PTS department, Georgia Tech's bus service provider, for demonstration assistance and acquiring a single-entrance bus. With approval from PTS, the team installed the embedded system on the bus and configured the LTE module to connect to the backend services. A live demonstration of the passenger counting system tracking Georgia Tech students is shown in the last 90 seconds of the Final Video link below. During the final project demonstration at the Georgia Tech Capstone Design Expo, the following items were demonstrated:

- Passenger counting mechanism with LiDAR, Arduino, and nRF9160
- Accurate and reliable transmission via the LTE-M network to the backend
- Successfully receiving and visualizing passenger count in the map display on front-end GUI

Our updated embedded system technical specifications are shown in *Table 1*. Starting from the top of *Table 1*, the delay for the calculation of aggregate passenger count was measured to be 20 ms. In the passenger counting logic program, the calculation of the current passenger count happens every 20 ms in an individual software thread (see link #1 below). The 3D printed LiDAR sensor housing has a 13.75 cm width and a 17.9 cm length (see page 39 of Thomas Talbot's Design Notebook). The rectangular area of the sensor housing dimension was measured to be 246.125 cm^2. The advertised range of the VL53L0X LiDAR sensors is 30 mm - 1000 mm [15]. Thomas and David performed an experimental test of this specification and verified that the VL53l0X sensors accurately detect objects less than 800 mm away (see Thomas Talbot's Design Notebook pages 44-49). At the start of the project, we determined that we would need a cellular microcontroller with an LTE carrier frequency of 600 MHz - 2.5 GHz. The nRF9160 uses a 2.4 GHz LTE-M antenna [5]. This modem worked correctly in the Atlanta, GA area around the Georgia Tech campus. Future testing is needed to determine if the nRF9160 is accurate in other coverage areas. The

static code size of the Arduino Nano program (link #2) is 3.72 KB, which is less than the 32 KB of flash memory on the Arduino Nano [16]. The static code size of the nRF9160 program (link #1) is 31.3 KB, which is less than the 1 MB of flash on the nRF9160 [5]. Shadman and David measured a 2.5-second average delay from the nRF9160 to the API endpoint via AWS Cloud Watch. The live map update rate and analytics dashboard update rates in *Table 2* will vary based on the client's network speed. However, it is safe to estimate that the map update rate and the analytics dashboard update rate will be greater than 2.5 seconds.

The IntelliBus system architecture can be broken up into three major functional modules: the embedded passenger counting system (LiDAR, Arduino Nano, nRF9160), the data pipeline (MQTT Broker, Kinesis Stream, EC2 Web Server), and the front-end webpage (map display and analytics dashboard). The embedded passenger counting system was tested independently before connecting to the back-end data pipeline by running a simpler version of the program in (link #2) that ignores the AWS IoT functions. Thomas and David simulated a bus environment by pointing the LiDAR sensors at a plank of wood a set distance away (see 3:26 in the Final Video). The data pipeline was tested by implementing a mock APC that published random passenger counts and GPS coordinates to the AWS backend. The front-end webpage was tested by creating the map display and the analytics dashboard with hard-coded static passenger counts and coordinates stored in JSON objects. Once the data pipeline and the front-end displays were tested independently, it was straightforward for the web page to pull live data from the pipeline and update graphics accordingly.

List of links to external documentation:

1. Passenger counting program running on the nRF9160 during the project demonstration ~ https://github.com/IntelliBus-SeniorDesign/intellibus-embeddedapp/blob/nRF9160/intellibus_awsiot_counting_reset.c.

2. Program that reads LiDAR sensor distances on the Arduino Nano ~ https://github.com/IntelliBus-SeniorDesign/intellibus-embeddedapp/blob/nano/LiDARWithArduino.cpp.
3. Thomas Talbot's Design Notebook ~ https://github.com/IntelliBus-SeniorDesign/intellibus-embeddedapp/blob/nano/sd21f47_Talbot_Thomas%20(1).pdf.
4. Final Video ~ https://www.youtube.com/watch?v=lCoO3CcgHN8.
5. Photograph 1 of the prototype embedded passenger counting system ~ https://github.com/IntelliBus-SeniorDesign/intellibus-embeddedapp/blob/nano/IntelliBusEmbeddedSystemPic1.jpg
6. Photograph 2 of the prototype embedded passenger counting system ~ https://github.com/IntelliBus-SeniorDesign/intellibus-embeddedapp/blob/nano/IntelliBusEmbeddedSystemPic2.jpg.

6. Schedule, Tasks, and Milestones

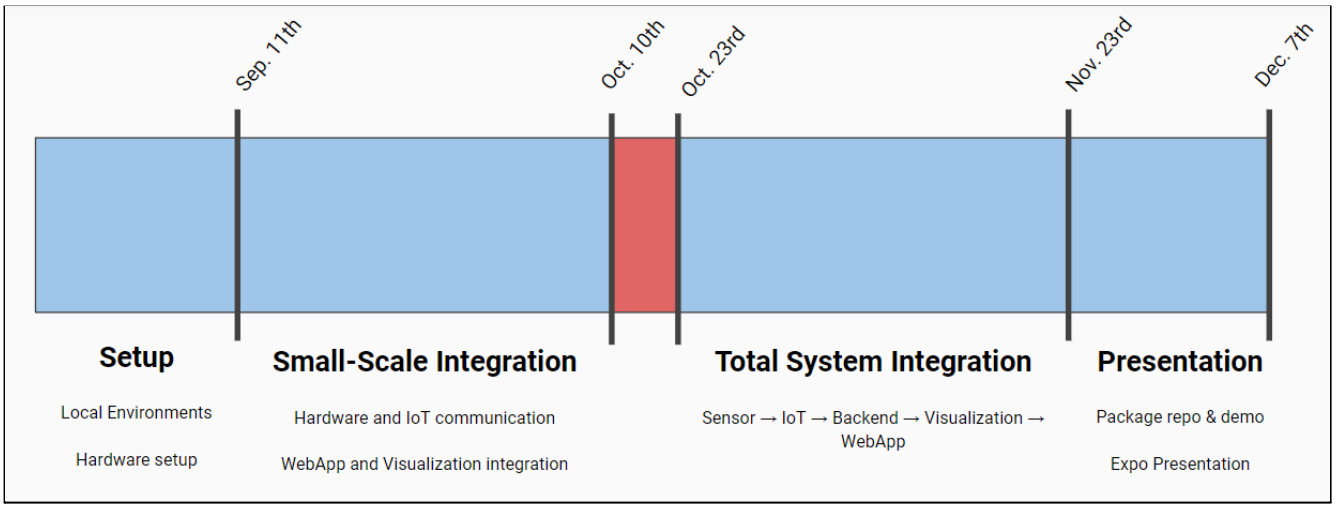


Figure 9. The simplified Team Timeline.

Reflecting on the project, our group did an exceptional job at staying on schedule and reaching planned milestones. Figure 9 shows our team's simplified schedule, based on our earlier comprehensive GANTT/PERT chart described in detail in the paragraphs below. Our team broke our project up into 4 distinct phases: Setup, Small-Scale Integration, Total System Integration, and Presentation. Doing this allowed our team to work in parallel for the first half of the project. With the AWS cloud being the backbone of our device and front-end communications, our hardware and software teams had full freedom to develop independently, without worrying about development blockers. During our small-scale integration, the hardware team worked closely with the IoT team, developing MQTT communication pipelines so that data from the sensors could be sent up to the cloud for processing. While that was going on, the front-end team developed the API fetch routines and coordinated with the IoT team as to the data necessary to visualize web-components.

For what actually happened, our team closely followed the schedule aside from minor hiccups. The time denoted in red from October 10th to October 23rd was set aside for purchasing new hardware and testing with LiDAR instead of PIR sensors. In addition, our total system integration ran past November 23rd to about December 4th due to interruptions caused by the Thanksgiving break.

The IntelliBus team designed and implemented this prototype during the fall 2021 semester. Appendix B contains the list of project tasks, the person assigned, and the risk level associated with each task. Tasks in which there is a large amount of experience among team members are deemed Low Risk. Team members have limited background knowledge of High Risk tasks, and these components may take up most of the labor budget. Appendix C contains a GANTT chart for the project showing the start date, finish date, and predecessors for completion. Appendix D shows the PERT chart for the project. Each edge of the PERT chart is labeled with an optimistic time, most likely time, and pessimistic time for success.

The critical path for the project was determined using the PERT chart in Appendix D. The critical path is the route from the start node to the finish node with the longest expected duration. Using the expected time calculations in Appendix B, the length of the critical path ABHTUWXZ is 20.6267 weeks. Based on the PERT chart analysis, the probability that the team will complete the project one week before the GT Capstone Expo can be found using the following Z statistic:

$$Z_s = (T_s - T_e)/\sigma_T$$

T_s is the time associated with finishing the project one week before the GT Capstone Expo. It includes the time spent working on documentation and planning in ECE 4871 (1/14/21 – 04/16/21) plus the time during the fall semester one week before the Design Expo (08/23/21 – 11/30/21). Therefore, $T_s = 13.14 + 14.14 = 27.28$weeks. T_e is the duration of the critical path found using the expected time calculations in Appendix B. σ_T is the standard deviation of the critical path found from Appendix B.

$$Z_s = (T_s - T_e)/\sigma_T = (27.28 - 20.6267)/1.8322177 = 3.613130208$$

Using a Z table, $P(Z < 3.613130208) = 0.99985$. There is a 99.985% probability that the team will complete the project one week before the GT Capstone Expo.

7. Marketing and Cost Analysis

7.1 Marketing Analysis

There are virtually no existing commercial products that are specifically designed to count the number of people on a moving bus. However, the idea has been proposed in a recent research paper where the number of passengers on a public bus was tracked using infrared sensors, and the location of the bus could be viewed on an Android application [17]. For comparison to the larger marketplace, the proposed product can be categorized as a people counting system. People counting systems include any device that tracks the number of individuals that move through a particular passage or entrance over a fixed time. The global market size of people counting systems was $730.1 million in 2018 and is projected to reach $1,491.3 million by 2026 [18]. The majority of counters are targeted at the retail industry. The Bi-Directional People Counter by Immotion gathers data from more than 1000 infrared sensors to track the preferences and behaviors of customers on the floor of a department store. The proposed product provides a web application where users can access counts and maps of bus routes. In contrast, the Immotion counter leaves the software analysis up to the user [19].

Similar products also exist in corporate settings. The Monnit Wi-Fi Infrared Motion Sensors use passive infrared technology to track the occupancy of desks, conference rooms, and hallways [20]. The device connects to a Wi-Fi network and communicates to the iMonnit Online Sensor Monitoring and Notification System, where sensor data can be reviewed and exported. Although the product in [20] provides customers with a secure online platform to analyze motion sensor readings, it would not

be helpful inside of a bus because it must stay within range of a Wi-Fi access point. Thus, the proposed new product is differentiated by its mobility and the application software included in the purchase.

7.2 Cost Analysis

The equipment for a prototype of the IntelliBus system includes the IoT development kit, LiDAR sensors to track passenger motion, and additional wires and connectors. The Nordic Semiconductor cellular IoT development kit is the most expensive item in the project budget at $139.00 [5]. The VL53L0X ToF distance sensor costs $14.95 each [15]. The Arduino Nano costs $5.80 [16]. The costs of wires and connectors are estimated to be around $1. The embedded software development environment nRF Connect SDK is included with the IoT development kit purchase [5]. The prototype will use approximately $25.00 if Amazon Web Services Cloudwatch and Kinesis Stream [21]. The total equipment cost for the prototype is approximately $200.70.

Table 4. Equipment Costs

Product Description	**Quantity**	**Unit Price ($)**	**Total Price ($)**
nRF9160 DK Cellular IoT Development Kit	1	$139.00	$139.00
VL53L0X ToF Distance Sensor	2	$14.95	$29.90
Arduino Nano	1	$5.80	$5.80
Wires and connectors	1	$1.00	$1.00
Amazon Web Services (Cloudwatch and Kinesis Stream)	1	$25.00	$25.00
Total Cost			**$200.70**

The Development costs in Table 5. were calculated assuming a typical engineer's salary of $45.00 per hour [22]. Back-end code development is a particularly high-risk task and will take the most significant amount of labor hours.

Table 5. Development Costs

Project Component	**Labor Hour**	**Labor Cost**	**Part Cost ($)**	**Total Component Cost ($)**
Embedded System				
Assembly	8	$360.00	$200.70	$560.70
Code Development	50	$2,250.00		$2,250.00
Quality Assurance	50	$2,250.00		$2,250.00
Front-End Development				
Code Development	172	$7,740.00		$7,740.00
Quality Assurance	55	$2,475.00		$2,475.00
Back-End Development				
Code Development	212	$9,540.00		$9,540.00
Quality Assurance	60	$2,700.00		$2,700.00
Demo Preparation	50	$2,250.00		$2,250.00
Group Meetings	30	$1,350.00		$1,350.00
Total Labor Cost	687	$30,915.00		$30,915.00

Total Part Cost			$200.70	
Total Cost (Labor + Part)				**$31,115.70**

Using a fringe benefit of 30% of total labor cost and an overhead of 65% of total materials, labor cost, and fringe, the total development cost of the prototype is $66,643.83 as seen in Table 6. The overhead cost includes factors indirectly tied to the product's production, such as rent and utilities.

Table 6. Total Development Cost

Parts	$200.70
Labor	$30,915.00
Fringe Benefits % of Labor	$9,274.50
Subtotal	$40,390.20
Overhead, % of Material, Labor & Fringe	$26,253.63
Total	$66,643.83

The production will consist of 100 units sold over five years at $778.27 per unit. The cellular IoT kit cannot be purchased at a bulk discount, and the cost for each unit is $139.00 [5]. The wires and sensors can be purchased at a discounted price of $8 and $10, respectively. The cloud services will cost $20 per unit. Technicians employed at $20 per hour will assemble the embedded system on the bus infrastructure in two hours. A technician will spend one hour testing the device, and a software engineer will need an additional hour to assess and verify the product's software. Advertising to local transportation departments will make up 6% of the total sales. Assuming the product has a residual value of zero dollars and a total lifespan of twenty years, the annual fixed amortization over all input costs is $25.86. For 100 units sold at $778.27, the total revenue and total profit of the production are $77,827 and $8,500. The expected profit per unit is $85 yielding a 10.9% payback.

Table 7. Selling Price and Profit Per Unit (Based on 100 unit production)

Parts Cost	$139.00 (cellular IoT kit) + $10 (sensors) + $8 (wires) + $20 (cloud services) = $177
Assembly Labor	$40.00
Testing Labor	$65.00
Total Labor	$105.00
Fringe Benefits, % of Labor	$31.50
Subtotal	$313.50
Overhead, % of Material, Labor & Fringe	$203.78
Subtotal, Input Costs	$517.28
Sales Expense	$46.69
Amortized Development Costs	$129.30
Subtotal, All Costs	$693.27
Profit (Per Unit)	$85
Percent Profit (Per Unit)	10.9%
Selling Price	$778.27

8. Conclusion & Current Status

Ultimately the team was able to complete all the major tasks of the project shown in Appendix B. We created an IoT device that uses LiDAR sensors to capture passengers entering the bus, transmit the rider count over a cellular network, and then create charts and a map display showing the bus capacity in real-time. The team was able to successfully test out the passenger counting portion of the project in a real-world environment in a Georgia Tech bus. The team also demonstrated a full system integration of the project at the Senior Design Expo. Looking back on the Project Proposal from

ECE4871, the team estimated that the project would be complete by November 23rd. There were several tasks in the fall semester that took longer than originally expected. In the embedded system, setting up the full hardware environment took nearly the entire 15 weeks.

To start, the first nRF9160-DK that we ordered broke during the second week of school. It then took us two weeks to get a proper replacement. During September and October, the team tested three different versions of PIR sensors. A major shift in the design occurred around week nine of the project when the hardware team decided to abandon PIR sensors and use LiDAR sensors to track passenger motion. The LiDAR sensors were chosen for their superior performance and reliability. In the remaining four weeks of the project, the hardware team was able to recreate the passenger tracking system using the VL53L0X LiDAR sensors, a dedicated Arduino Nano for reading the LiDAR sensors, and the nRF9160 for LTE-M communication. 3D printing a usable sensor-housing for the project also added significant delay. Thomas printed out four different designs before reaching the final iteration. Combined with longer than expected wait times at the on-campus maker space, it took the team eight weeks to get a usable 3D print.

The data pipeline and website were developed exactly as we planned except with a change in using AWS Quicksight. AWS Quicksight was considered for its BI functionality but did not fit the constraints of how many API calls we could make to get data and the price. Instead, we focused solely on creating an Analysis Web page with various graphs on data that we collected through testing. We pivoted our project to be able to integrate to users existing BI solutions if a customer desired such a solution. The data pipeline was developed exactly as envisioned by using the IoT Core rules engine, data streaming with Kinesis, and storing our data in a warehouse (using DynamoDB). Our solution was fully modular as expected with an API that made all our data accessible.

A future team may be interested in adding GPS capability to the project. For buses that don't already have position tracking, being able to show the current location of the bus is valuable information for riders. Further testing could also be done to determine the optimal sensor height on the bus. At lower heights, passenger movement is more variable (shuffling of legs, luggage, etc.). However, if the LiDAR sensors are placed too high, the passenger counting system may skip shorter riders. Future teams should also note that our prototype was designed for a bus with a single entrance. Passenger counting systems designed for city and county transit authorities may not function the same for a university system. A university system bus typically has two bus entrance doorways, while a local bus usually charges a fare and has a single front entrance.

The applications of the people counting technology in our project are not limited to bus transportation. Busses, airplanes, trains, and subway systems all have a fresh demand for controlling crowds in the wake of the COVID-19 pandemic. There are a handful of contemporary products that provide passenger counting capabilities similar to IntelliBus. The Transit Bus App is a commercial advanced passenger counting system that has recently broken ground on the MARTA and CobbLinc in Atlanta, GA [23]. The Transit Bus app provides real-time bus tracking and trip planning with categorical crowding information by describing buses on a mobile app as not crowded, some crowding, or crowded. Similar to IntelliBus, Transit uses side-by-side infrared beams to detect human movement [24]. In the United States, the Automated Passenger Counting market is expected to increase from $155 in 2019 to $245 million by 2025. COVID-19 sparked the increased demand for real-time transit information from passengers and proper fleet management from transit authorities. The majority of this new growth is in APC systems for buses that offer a mobile application service [25]. An increase in real-time public transit information could increase ridership and cause second-hand benefits to the environment. It is evident that public transportation is reducing greenhouse gasses and conserving

energy. When a solo commuter switches his or her commute from a private vehicle to public transit it can reduce CO2 emissions by 20 pounds per day—more than 4,800 pounds in a year [26]. Providing riders with a convenient way to view live bus capacities and routes could attract more people to use public transportation and help slow the effects of global climate change.

9. Leadership Roles

Shadman Ahmed will serve as Overall Team Coordinator, Backend Software Lead, and Webmaster. Noah Chong will serve as Frontend Software Lead, Expo Coordinator, and Webmaster. Thomas Talbot will serve as Embedded Systems Co-Lead and Documentation Coordinator. Yue Pan will serve as Embedded Systems Co-Lead and Testing Lead. Leadership roles are not permanently assigned, and team members may be called upon to assist in other areas of the project depending on the workload.

10. References

[1] Nordic Semiconductor, “Development kit for LTE-M/NB-IoT/GPS/Bluetooth Low Energy”, nRF9160 DK datasheet, Oct. 2018, [Revised Nov. 2020].

[2] B. E. Benhiba, A. A. Madi, and A. Addaim, "Comparative Study of The Various new Cellular IoT Technologies," in Proc. 2018 International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), Kenitra, 2018, pp. 1-4, doi: 10.1109/ICECOCS.2018.8610508.

[3] “Goal 9: Build RESILIENT infrastructure, promote inclusive and sustainable industrialization and foster innovation,” Sep-2017. [Online]. Available: https://sdg.data.gov/9-c-1/. [Accessed: 12-Apr-2021].

[4] “Amazon Compute Service Level Agreement,” *Amazon Web Services*. [Online]. Available: https://aws.amazon.com/compute/sla/. [Accessed: 12-Apr-2021].

[5] “nRF9160 DK,” *Nordic Semiconductor*, 2020. [Online]. Available: https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF9160-DK. [Accessed: 14-Feb-2021].

[6] “AWS IoT Core,” *AWS IoT Core Overview*. [Online]. Available: https://aws.amazon.com/iot-core/. [Accessed: 12-Apr-2021].

[7] “Amazon DynamoDB,” *Amazon Dynamo DB | NoSQL Key Value Database*. [Online]. Available: https://aws.amazon.com/dynamodb/. [Accessed: 12-Apr-2021].

[8] *Create an EC2 instance and install a web server*, 2021. [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateWebServer.html. [Accessed: 12-Apr-2021].

[9] "USB Charger (USB Power Delivery)," *USB*. [Online]. Available: https://usb.org/usb-charger-pd. [Accessed: 16-Mar-2021].

[10] *Information technology – Message Queuing Telemetry Transport (MQTT)*, ISO Standard ISO/IEC 20922:2016, June 2016. [Online]. Available: https://iso.org/standard/69466.html

[11] *Hyper Text Transfer Protocol – HTTP/1.1*, RFC 2616, June 1999. [Online]. Available: https://ietf.org/rfc/rfc2616.txt

[12] "I2C Bus Specification," *I2C Info – I2C Bus, Interface and Protocol*. [Online]. Available: https://i2c.info/i2c-bus-specification. [Accessed: 26-Mar-2021].

[13] "Long Term Evolution for Machines: LTE-M," *Internet of Things*, 19-May-2020. [Online]. Available: https://www.gsma.com/iot/long-term-evolution-machine-type-communication-lte-mtc-cat-m1/#:~:text=LTE%2DM%20is%20the%20simplified,CatM1%2C%20suitable%20for%20the%20IoT

[14] K. Flynn, "A Global Partnership," *Standards for the IoT*. [Online]. Available: https://www.3gpp.org/news-events/1805-iot_r14. [Accessed: 25-Mar-2021].

[15] "Adafruit VL53L0X time of Flight Micro-LIDAR Distance Sensor Breakout," *Adafruit Learning System*, 2021. [Online]. Available: https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout. [Accessed: 06-Dec-2021]

[16] "Arduino Nano," *Arduino Official Store*. [Online]. Available: https://store.arduino.cc/products/arduino-nano. [Accessed: 09-Dec-2021].

[17] B. K. Harini, A. Parkavi, M. Supriya, B. C. Kruthika, and K. M. Navya, "Increasing Efficient Usage of Real-Time Public Transportation Using IOT, Cloud and Customized Mobile App," *SN*

Computer Science, 09-May-2020. [Online]. Available: https://link.springer.com/article/10.1007/s42979-020-00161-8. [Accessed: 01-Apr-2021].

[18] "People Counting System Market Size, Share, Trends And Forecast," *Verified Market Research*, 03-Dec-2020. [Online]. Available: https://www.verifiedmarketresearch.com/product/people-counting-system-market/. [Accessed: 31-Mar-2021].

[19] *Counting Solutions*, 01-Jul-2017. [Online]. Available: https://www.im-motion.com/counting-solutions/. [Accessed: 15-Mar-2021].

[20] Monnit, "Monnit Wi-Fi Infrared Motion Sensor", MS-IR-WDS-01 datasheet, 2016

[21] "Amazon QuickSight Pricing" *Amazon*, 2021. [Online]. Available: Amazon QuickSight Pricing - Business Intelligence Service - Amazon Web Services. [Accessed: 2-Apr-2021].

[22] "Engineers: Employment, pay, and outlook : Career Outlook," *U.S. Bureau of Labor Statistics*, 2016. [Online]. Available: https://www.bls.gov/careeroutlook/2018/article/engineers.htm#:~:text=The%20U.S.%20Bureau%20of%20Labor,median%20wage%20for%20all%20workers. [Accessed: 20-Mar-2021].

[23] "Hello, Atlanta," *Transit*. [Online]. Available: https://transitapp.com/region/atlanta. [Accessed: 06-Dec-2021].

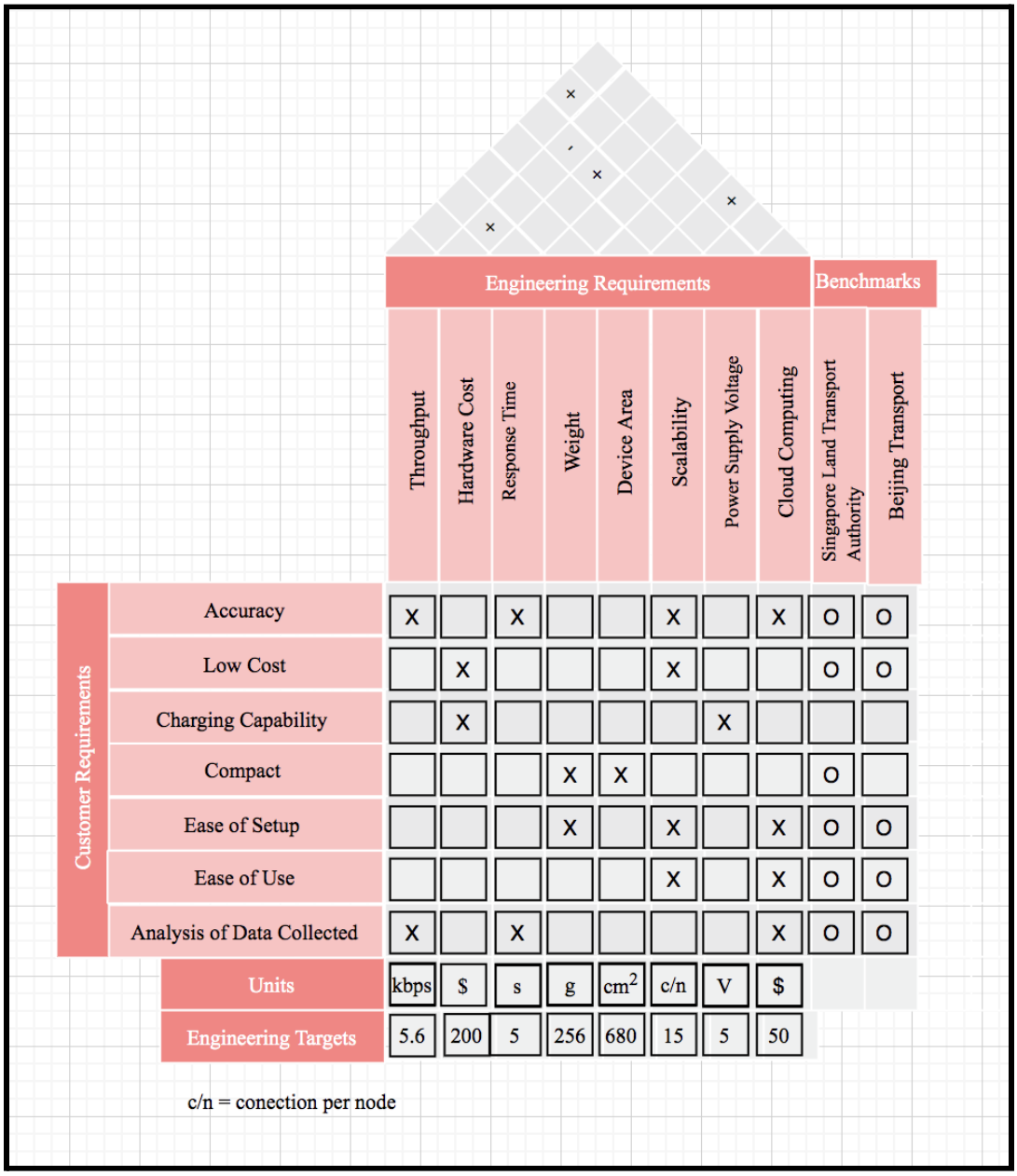
[24] Transit, "What do bus riders want? Crowding info.," 31-Aug-2020. [Online]. Available: https://archive.transitapp.com/what-do-bus-riders-want-crowding-info-db89b6d0b5ec. [Accessed: 18-Oct-2021].

[25] MarketsandMarkets, "Automated passenger counting system market worth $245 million by 2025 - exclusive report by MarketsandMarkets™," *Automated Passenger Counting System Market worth $245 million by 2025 - Exclusive Report by MarketsandMarkets™*, 29-Sep-2020.

[Online]. Available: https://www.prnewswire.com/news-releases/automated-passenger-counting-system-market-worth-245-million-by-2025---exclusive-report-by-marketsandmarkets-301139567.html. [Accessed: 06-Dec-2021].

[26] "Public Transportation Reduces Greenhouse Gases and Conserves Energy," *apta*, 2007. [Online]. Available: https://www.apta.com/wp-content/uploads/Resources/resources/reportsandpublications/Documents/greenhouse_brochure.pdf. [Accessed: 06-Dec-2021].

Appendix A - Project QFD Chart



Customer Requirements	Throughput	Hardware Cost	Response Time	Weight	Device Area	Scalability	Power Supply Voltage	Cloud Computing	Singapore Land Transport Authority	Beijing Transport
	Engineering Requirements								Benchmarks	
Accuracy	X		X			X		X	O	O
Low Cost		X				X			O	O
Charging Capability		X					X			
Compact				X	X				O	
Ease of Setup				X		X		X	O	O
Ease of Use						X		X	O	O
Analysis of Data Collected	X		X					X	O	O
Units	kbps	$	s	g	cm^2	c/n	V	$		
Engineering Targets	5.6	200	5	256	680	15	5	50		

c/n = conection per node

Appendix B - Tasks, Persons Assigned, and Risk Level

The expected time of a task was found using the following formula:

$t_e = (t_o + 4t_m + t_p)/6$

The standard deviation of a task was found using the following formula:

$\sigma = (t_p - t_o)/6$

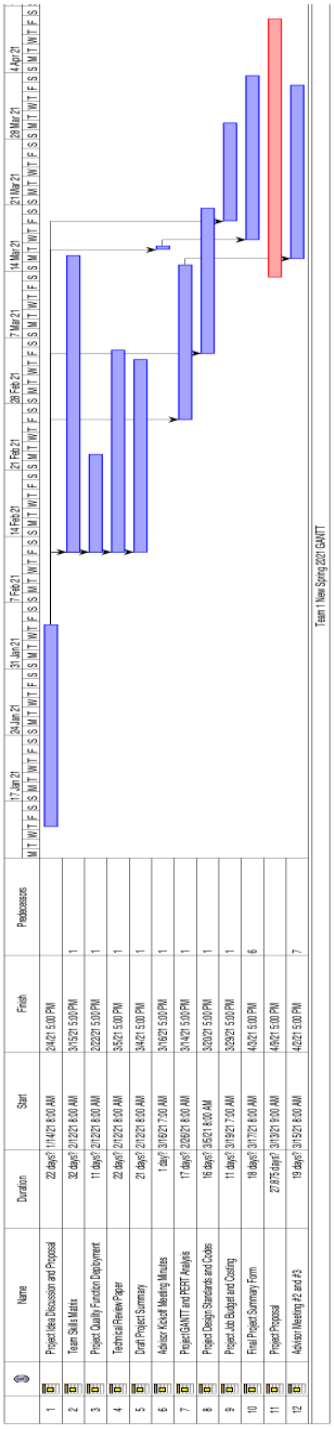
Letter	Task	Predecessor	Task Lead	Risk Level	t_o	t_m	t_p	Expected Time	Standard Deviation
A-K	**Planning and Documentation (ECE 4871)**	Start	All	Low					
A	Project Idea Discussion and Proposal	Start	All	Low	2	3	4	3	0.3333
B	Technical Review Paper	A	All	Low	1	2	3	2	0.3333
C	Project QFD	A	All	Low	1	1.5	2	1.5	0.16667
D	Draft Project Summary	A	All	Low	2	3	4	3	0.3333
E	Kick Off Meeting Minutes	A	Thomas	Low	0.14	0.28	1	0.37666	0.14333
F	Project GANTT and PERT Analysis	A	All	Medium	1	2	3	1.83333	0.3333

G	Project Design Standards and Codes	A	All	Medium	1	2	3	2	0.3333
H	Project Job Budgeting and Costing	B	All	Medium	0.5	1	2	1.083333	0.25
I	Final Project Summary Form	D	All	Low	2	3	4	3	0.3333
J	Project Proposal	C,F,G,H	All	High	2	3	4	3	0.3333
K	Remaining Two Meeting Minutes	E	Shadman, David	Low	0.14	0.28	1	0.37666	0.1433
L-P	**Embedded System Setup and Integration**	H	Thomas, David	Medium					
L	Order IoT development kit and infrared sensors	H	Thomas, David	Low	0.28	1	2	1.046667	0.28667
M	Set up hardware environment	L	Thomas, David	Medium	0.14	0.28	1	0.37666	0.1433
N	Program microcontroller to read sensor inputs	M	Thomas, David	Medium	0.42	1	2	1.07	0.2633
O	Test and debug the sensor processing code	N	Thomas, David	Medium	0.42	2	3	1.9033333	0.43
P	Simulate the bus environment and verify accuracy of data acquisition	O	Thomas, David	High	1	2	3	2	0.3333

Q	Enable communication between the embedded system and Backend Software	M, U	Noah, Shadman	Medium	1	2	4	1.5	0.5
R	Integration of sensor data processing and communication to the backend software including testing and debugging	Q	Noah, Shadman	Medium	1	3	4	2.8333	0.5
S	Put the embedded device code into production	R,P	Thomas, David	Low	0.42	1	2	1.07	0.2633
T-V	**Back-End Software Development**	H	Shadman	High					
T	Setup web app and servers and database (Cloud Serices)	H	Shadman	Low					
U	Setup and connect IoT Hub and IoT Management with devices	T	Shadman	Medium	1	2	4	2.166667	0.5
V	Connect Power BI with our IoT Hub for data analytics	U	Noah, Thomas	High	1	2	4	2.1666667	0.5

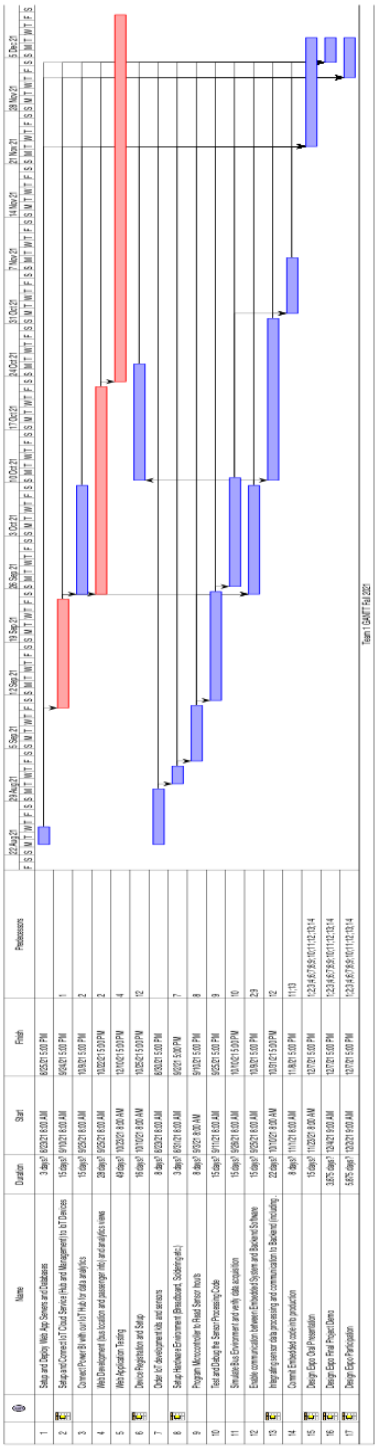
W-Y	**Front-End Software Development**	H	Noah	High					
W	Web Development (for bus location and passenger info) and analytics views	H, U	Noah, Shadman	Low	3	4	6	4.166667	0.5
X	Web App Testing	W	Noah, Shadman	Low	4	6	7	5.83333	0.5
Y	Device Registration and Setup	Q	Noah, Shadman	Low	1	2	4	2.166667	0.5
Z-β	**Presentations and Design Expo**	Y,S,V,X	All	Medium					
Z	4872 Oral Presentation	Y, S, V, X	All	Low	1	2	3	2	0.3333
α	4872 Final Project Demonstration	Y, S, V, X	All	Medium	0.28	0.42	1	0.493333	0.12
β	4872 Participation in Design Expo	Y, S,V, X	All	Medium	0.28	0.70	1	0.68	0.12

Appendix C - Spring and Fall 2021 Project GANTT Chart



	Name	Duration	Start	Finish	Predecessors
1	Project Idea Discussion and Proposal	22 days	1/14/21 8:00 AM	2/4/21 5:00 PM	
2	Team Skills Matrix	32 days	2/12/21 8:00 AM	3/15/21 5:00 PM	1
3	Project Quality Function Deployment	11 days	2/12/21 8:00 AM	2/22/21 5:00 PM	1
4	Technical Review Paper	22 days	2/12/21 8:00 AM	3/5/21 5:00 PM	1
5	Draft Project Summary	21 days	2/12/21 8:00 AM	3/4/21 5:00 PM	1
6	Advisor Kickoff Meeting Minutes	1 day	3/16/21 7:00 AM	3/16/21 5:00 PM	1
7	Project GANTT and PERT Analysis	17 days	2/26/21 8:00 AM	3/14/21 5:00 PM	1
8	Project Design Standards and Codes	16 days	3/5/21 8:00 AM	3/20/21 5:00 PM	1
9	Project Job Budget and Costing	11 days	3/19/21 7:00 AM	3/29/21 5:00 PM	1
10	Final Project Summary Form	18 days	3/17/21 8:00 AM	4/5/21 5:00 PM	6
11	Project Proposal	27.875 days	3/31/21 5:00 AM	4/9/21 5:00 PM	
12	Advisor Meeting #2 and #3	19 days	3/15/21 8:00 AM	4/2/21 5:00 PM	7

Team 1 New Spring 2021 GANTT



	Name	Duration	Start	Finish	Predecessors
1	Setup and Develop Web App Services and Databases	3 days	9/25/21 8:00 AM	9/29/21 5:00 PM	
2	Setup and Connect to IoT Cloud Service Hubs and Management for IoT Devices	15 days	9/30/21 8:00 AM	10/20/21 5:00 PM	1
3	Connect Power Bi with our IoT Hub for data analytics	15 days	9/29/21 8:00 AM	10/19/21 5:00 PM	2
4	Web Development (data isolation and passenger info) and analytics review	39 days	9/29/21 8:00 AM	10/22/21 5:00 PM	2
5	Web Application Testing	49 days	10/3/21 8:00 AM	12/10/21 5:00 PM	4
6	Device Registration and Setup	18 days	10/13/21 8:00 AM	11/25/21 5:00 PM	12
7	Order IoT development kits and sensors	9 days	9/29/21 8:00 AM	10/8/21 5:00 PM	
8	Setup Hardware Environment (Breadboard, Soldering etc.)	3 days	9/30/21 8:00 AM	10/2/21 5:00 PM	7
9	Program Microcontroller for Road Sensor Inputs	8 days	9/30/21 8:00 AM	10/11/21 5:00 PM	8
10	Test and Debug the Sensor Processing Code	15 days	10/11/21 8:00 AM	10/29/21 5:00 PM	9
11	Simulate Bus Environment and verify data acquisition	15 days	10/29/21 8:00 AM	11/16/21 5:00 PM	10
12	Enable communication between Embedded System and Backend Software	15 days	10/25/21 8:00 AM	11/12/21 5:00 PM	2;3
13	Integrating scenario data processing and communication to Backend (including	22 days	10/18/21 8:00 AM	11/16/21 5:00 PM	12
14	Commit Embedded code into production	9 days	11/17/21 8:00 AM	11/29/21 5:00 PM	11;13
15	Design Expo Draft Presentation	15 days	11/29/21 8:00 AM	12/17/21 5:00 PM	1;2;3;4;5;7;8;9;10;11;12;13;14
16	Design Expo Final Project Demo	3.875 days	12/14/21 8:00 AM	12/17/21 5:00 PM	1;2;3;4;5;7;8;9;10;11;12;13;14
17	Design Expo Participation	5.875 days	12/2/21 8:00 AM	12/17/21 5:00 PM	1;2;3;4;5;7;8;9;10;11;12;13;14

Team 1 GANTT Fall 2021

Appendix D - Project PERT Chart