Public Transportation Efficiency with Smart Stop Density Monitoring System

Institution: Sixfab Company

Internship Period: 08.07.2025-19.10.2025

Submission Date:

Student Full Name	Student Surname	Student ID	Student University	Student Department	Student Unit
Ömer Ercan	DAYAN	150122050	Marmara University	Engineering Faculty	Computer Engineering

Content

Abstract 3

Problem Description 4

Solution Approaches 5

1. Description of General Solution Idea 5
2. Improved Solutions 6
   - 2.1. Basic Change Detection System 6
   - 2.2. Cloud-Integrated Monitoring System 9
   - 2.3. Optimized IoT Observation Framework 9

General Hardware and Cots 10

Future Work 11

Terminology and References 11

Abstract

This report presents the design and implementation of the PTE_SSDMS (Public Transportation Efficiency with Smart Stop Density Monitoring System), an intelligent bus stop density monitoring system. The system, developed using Raspberry Pi 4 and the Sixfab modem kit, provides basic image processing capabilities and real-time data transmission. This enables real-time analysis of passenger density at bus stops, aiming to inform both passengers and management units.

The application was developed using Python-based software and integrated with ThingsBoard, allowing data to be processed and visualized on a cloud-based platform. Multiple versions of the project were tested, and system performance was continuously improved through version control managed on GitHub. The results demonstrate that the system functions successfully and indicate that future versions can be enhanced by integrating more advanced image processing techniques and AI-based analytics, thus expanding the overall scope of the project.

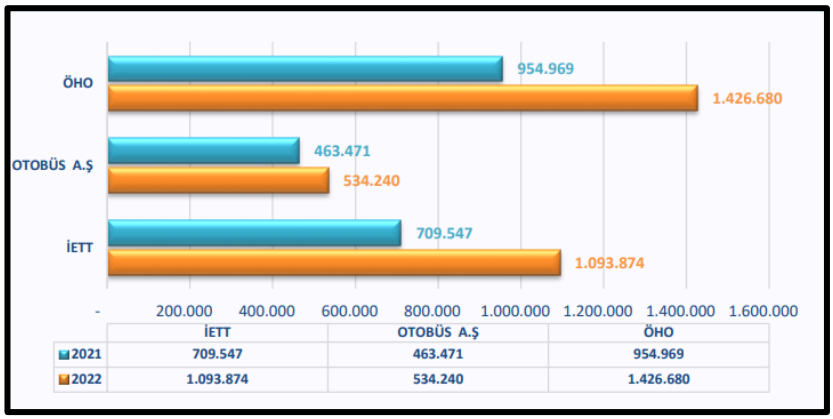
Problem Description

Imagine working at a company where you're expected to be in the office every weekday at 9:00 AM. A citizen wakes up early one morning but doesn't even have time to make a cup of coffee—let alone enjoy a proper breakfast. "I'll grab a coffee on the way," the person thinks, rushing out the door. Still half-asleep from a poor night's rest, the person aims to catch the 08:00 bus, leaving home at 07:45.

But at the stop, the person finds at least 25 other people waiting to board the same bus.

The person thinks, "Maybe I should've taken the other line from the next stop—would I have made it in time?" As the person hesitates, the bus that arrives is already nearly full. Squeezing in is the only option. And worse still, this is just the beginning: after the bus, the person needs to transfer to the metro—which will be even more crowded.

This isn't just one person's bad morning; it's a daily struggle shared by thousands of city residents. And this struggle is reflected in actual data: in Istanbul alone, between January and June 2022, the number of bus passengers reached millions each month—ranging from 2.5 million in January to over 3.1 million in May. These figures clearly show that in a megacity like Istanbul, managing such a massive volume of passengers is not optional; it requires serious planning and constant adjustments to prevent inefficiencies and frustration.



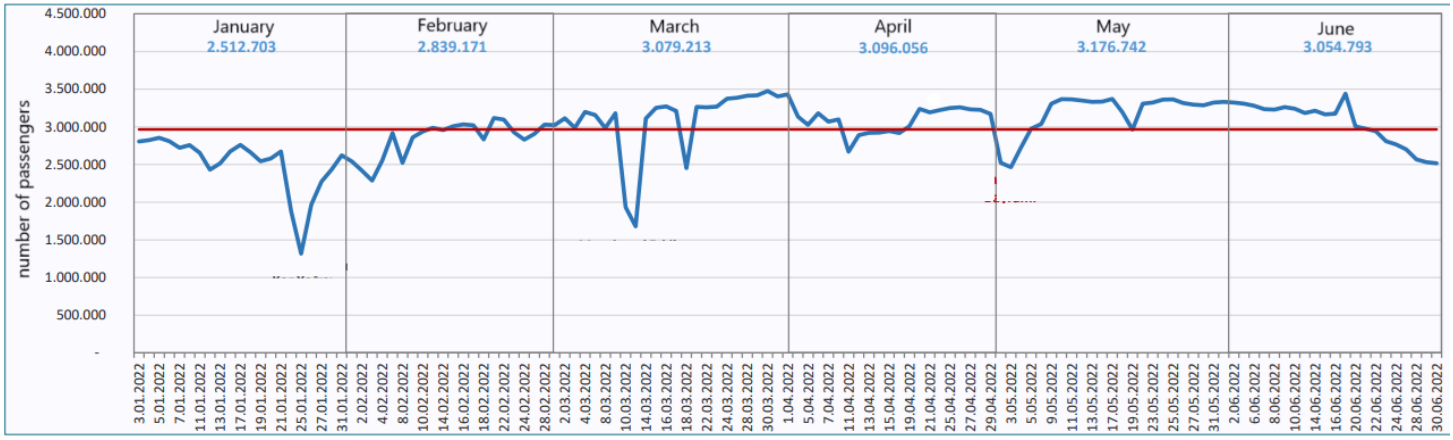
Grahp 1: Number of bus trips by month in 2022

To make things worse, the same overcrowded bus stop becomes almost completely deserted later in the day. The buses pass by empty. While the line is overloaded in the morning, it runs nearly idle at other times, leading to fuel waste and inefficiency.

But what if we could see the crowd levels at bus stops in advance?

What if just one person could avoid being late by choosing a different route?

This system is designed to minimize exactly that problem.



Grahp 2: Number of bus passengers by month in 2022

Solution Approaches

1. Description of General Solution Idea

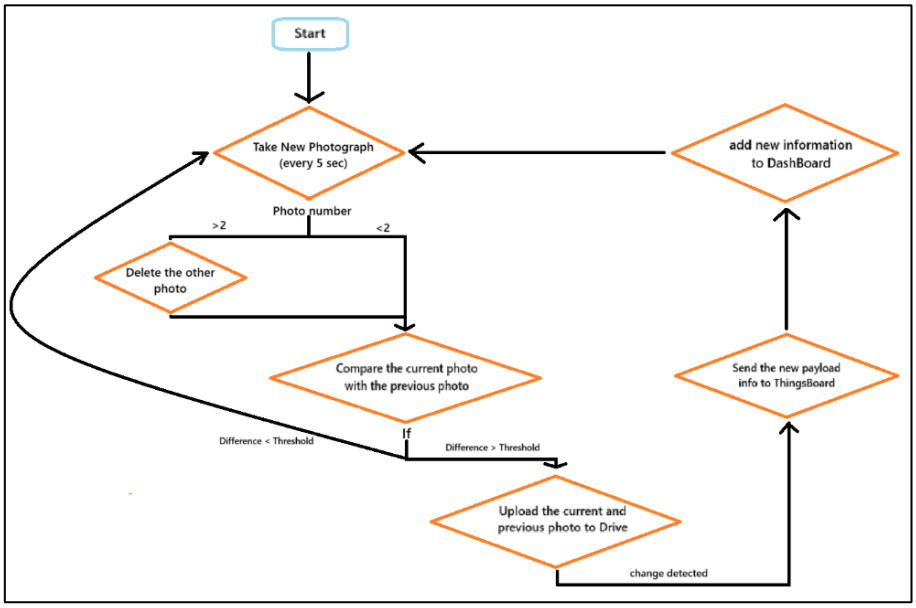
To explore potential solutions for the main problem, I reviewed various ideas aimed at addressing the issue. Each approach proposed a different method and perspective, so I analyzed their common points to determine whether a more efficient and sustainable solution could be developed. As a result, I designed and implemented the **Public Transportation Efficiency with Smart Stop Density Monitoring System** project.

As the name suggests, the main goal of this project is to enable more effective and smarter management of public transportation. Its primary application area is bus-based public transport systems, which are widely used around the world. The key aim is to reduce congestion and crowding, particularly during peak hours such as the morning and evening rush periods.

Our system is built on a Raspberry Pi with a camera module that captures real-time images of bus stops and applies simple image processing techniques to detect changes. The Python-based software connects to an IoT platform, such as ThingsBoard, to send change notifications and upload captured data to the cloud. In its current form, the system can detect changes in bus stop activity and lays the groundwork for future enhancements like density analysis, prediction, and advanced optimization studies.

2. Improved Solutions

2.1. Basic Change Detection System



- **The Idea Description**

The first solution, Basic Change Detection System, was designed as a fundamental prototype to validate the overall workflow of an automated monitoring system using a Raspberry Pi and camera module.

The primary objective was to determine whether a visible change occurred in the environment by comparing consecutive images, without focusing on identifying the exact nature or source of the change. This approach aimed to create a quick, low-cost starting point for more complex solutions in the future.

It integrated three core steps: capturing images at fixed intervals, performing a pixel-level comparison between current and previous images, and sending change notifications to ThingsBoard along with uploading photos to Google Drive for record-keeping.

- **Solution Steps**

1. **System Initialization:** The Raspberry Pi powers up, initializes the camera module, and checks for internet connectivity to ensure ThingsBoard and Google Drive integration.
2. **Image Capture Loop:** Photos are taken every 5 seconds and stored temporarily in local memory for quick access.
3. **Image Comparison:** The most recent photo is compared with the previously saved one using a pixel-based difference algorithm.
4. **Threshold Evaluation:** If the computed difference exceeds a defined threshold (default 30%), the system concludes that a significant environmental change has occurred.
5. **Cloud Actions:** The triggered images (current and previous) are uploaded to Google Drive, then a change event notification is sent to ThingsBoard, updating its dashboard.
6. **Cleanup and Repeat:** The system keeps only the last two images for memory efficiency, deletes older files, and restarts the loop from step 2.

- ## Detailed Explanation, CodeSnapshot & GitHub Link

This version operated on a simple yet functional pipeline:

1. **Image Capture:** Photos were taken every 5 seconds using the Raspberry Pi camera module.
2. **Comparison Algorithm:** The algorithm compared the most recent image with the previously saved one, using pixel-level difference to detect significant environmental changes. A configurable threshold value determined whether a change was relevant or a false trigger caused by lighting variations or sensor noise.
3. **Cloud Integration:** Upon detecting a change, both images were uploaded to Google Drive for archival purposes, while ThingsBoard received a notification, updating its dashboard to reflect the detected event.,

The approach deliberately avoided heavy processing techniques such as object recognition or video analysis to keep system resource usage minimal. By focusing on building a working baseline, it allowed testing of hardware integration, workflow timing, and cloud connectivity before moving on to more complex solutions.

Code Snapshot & GitHub Link:

The relevant code for image capture, pixel comparison, and Google Drive upload routines is available here:

https://github.com/ErcanPasha/inter_project

Key scripts used:

- *camera_loop.py* – Handles continuous photo capture.
- *compare_image.py* – Implements pixel-difference comparison.
- *upload_drive.py* – Uploads flagged photos to Google Drive.

- ## Pros/Cros and Issues Encountered

Pros

- **Simple and Fast Deployment:** Quick to implement and validate the core monitoring workflow without heavy computational requirements.
- **Low Hardware Requirements:** Uses only a Raspberry Pi 4, a camera module, and internet connectivity.
- **Cloud Connectivity Demonstration:** Successfully integrated with ThingsBoard for dashboard visualization and Google Drive for image archiving.
- **Configurable Threshold:** Allowed fine-tuning of change sensitivity without modifying the algorithm logic.

Cons

- **False Positives:** Pixel-level comparison frequently flagged lighting changes or minor sensor noise as real events.
- **No Change Localization:** The algorithm detected "a change" but not *where* or *what* changed.
- **Cloud Dependency:** Upload performance heavily depended on internet stability, sometimes causing delays.

- **Timing Issues:** Occasional mismatches between camera capture timing and comparison logic caused workflow delays.

Issues Encountered

During implementation, several challenges emerged, primarily involving threshold adjustment and cloud integration. Fine-tuning the sensitivity required multiple test iterations to reduce false positives, while network instability occasionally disrupted ThingsBoard and Google Drive uploads. Additionally, some timing mismatches between image capture and comparison caused minor workflow delays, highlighting the limitations of this basic prototype for larger-scale real-time applications.

- ## Hardware Compenent

2.2. Cloud-Integrated Monitoring System

- **The Idea Description**
- **Solution Steps**
- **Detailed Explanation, CodeSnapshot & GitHub Link**
- **Pros/Cros and Issues Encountered**
- **Hardware Compenent**

2.3. Optimized IoT Observation Framework

- **The Idea Description**
- **Solution Steps**
- **Detailed Explanation, CodeSnapshot & GitHub Link**
- **Pros/Cros and Issues Encountered**
- **Hardware Compenent**

General Hardware and Cots

a- Raspberry Pi 4
b- Sixfab Cellular Modem Kit
c- Raspberry Pi Camera Module v2

Future Work

Terminology and References