Progress in Engineering Application and Technology Vol. 3 No. 2 (2022) 938–944
© Universiti Tun Hussein Onn Malaysia Publisher's Office

PEAT

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/peat
e-ISSN : 2773-5303

Simulation of GPS-Based Train Positioning Information for LRT Kelana Jaya Line

Raja Iman Muhaimin Raja Ismail[1], Karthigesu Nagarajoo[1]*, Beant Singh Devinder Singh[2]

[1]Department of Transportation Engineering Technology, Faculty of Engineering Technology,
University Tun Hussein Onn Malaysia, 84600 Pagoh, Johor MALAYSIA

[2]r2pMY, D-08-07, Menara Suezcap 2, Jalan Kerinchi, 59200, Kuala Lumpur MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/peat.2022.03.02.093
Received 07 July 2022; Accepted 07 November 2022; Available online 10 December 2022

Abstract: Public Address and Passenger Information System (PA/PIS) is significant in railway industry as it provides multiple information to the passengers including the train's current and next station location. The information displayed is reliant on the Train's Positioning Information (TPI) which was gained from multiple sources such as Radio Frequency Identification (RFID) and Global Positioning System (GPS). For Light Railway Transit (LRT) Kelana Jaya Line, signalling and RFID had been used in the past as a source of TPI. However, issues arise from the use of both methods, where signalling is not future-friendly as it needs to update its configuration every time new fleets arrived and RFID is declared obsolete from the manufacturer, hence the use of GPS as a source for TPI was proposed. The objective of this project is to understand the PA/PIS and GPS concept for TPI, and to analyse the feasibility of the system. The concept of TPI, a brief detail of LRT Kelana Jaya Line, and the working principle of GPS is researched to help with the process of creating the simulator. This information would provide an idea of how to develop the GPS-based TPI simulator by creating a flowchart and block diagram of the system. As a result, A GPS simulator is developed to simulate a system where GPS works as a source for TPI. To conclude, a complete simulation of how the GPS broadcasts its signal which includes coordinates of the train was done.

Keywords: LRT Kelana Jaya Line, GPS, Python Simulation, Train Positioning Information

1. Introduction

Public Address/ Passenger Information System (PA/PIS) is an integral part of a railway system to ensure passengers are provided with reliable information regarding train's arrival and departure time

[1], general information such as train delays, the location of the next station, and also emergency alerts. PA/PIS also provides information such as showing the sides of the train that is opening the doors. For this research, LRT Kelana Jaya Line (KLJ) will be the subject of analysis as there has been changes in train's source of Train Positioning Information (TPI) for their PIS system.

The initial method or source of TPI is signalling. However, this method is not future proof. This is because when LRT KLJ adds a new fleet or set of trains in a project named KLAV27, compatibility issues arise. As signalling requires software and configuration update each time a new set of train arrives, it would be a hassle and costly if there are more fleets to be added in the future. Therefore, signalling was deemed unsuitable.

After signalling, LRT KJL used Radio Frequency Identification (RFID) as their source for TPI [2]. RFID is much better than signalling because it does not need constant update if a new fleet of train is added. This is as all the new fleets just need to install RFID readers. For RFID to work, the train and the stations need to have RFID tag and reader. When the train arrives at the station, the reader will read the tag and pass the TPI to the PA/PIS system. However, RFID also have their own problems. First and foremost, the supplier of the RFID equipment for LRT KLJ has declared their product obsolete, hence if the product needs repair or replacement, the service of existing fleets and track will be disrupted. This is because when the supplier is changed, all of the existing devices need to be changed to the new supplier's products. Next, maintenance issue arises as whenever the trackside needed repair and maintenance, any disturbed RFID tags would disrupt the service of the LRT.

As mentioned in the introduction, signalling as an initiating method is not future proof because of its expandability issues. Any new fleets of trains that are added needs to have software and configuration update to be able to function. For RFID, the product was declared obsolete hence making the method out of the question. This is because it makes replacing damaged components impossible. All the existing RFID tag and readers needs to be replaced with different suppliers thus making the method a hassle.

Therefore, GPS as a source for Train Positioning Information for the PA/PIS of LRT Kelana Jaya Line is proposed. As GPS is an open-source software that is easily accessible and modifiable, it is easier for systems to incorporate GPS into new fleets of trains [3].

The aim of this project is firstly, to determine the GPS coordinates of each station in LRT Kelana Jaya Line. Next, to review the system of GPS-based simulator for Train Positioning Information. Finally, from information stated above, to develop a GPS simulation model using modified parameter in Python language for Train Positioning Information

For this project, the main scope is focused on LRT Kelana Jaya Line (KJL) as the use of GPS-based Train Positioning Information for triggering PA/PIS was proposed to the line. Therefore, the background of LRT KLJ and the addition of new train fleet under the KLAV27 project is explored [4]. Next, the secondary scope of the project is the GPS system for TPI in railway industries. As the TPI is used to trigger the PA/PIS, the system will be explored and elaborated in this project. Finally, this project is limited towards the creation of GPS simulator system for GPS-based TPI using Python language.

2. Methodology

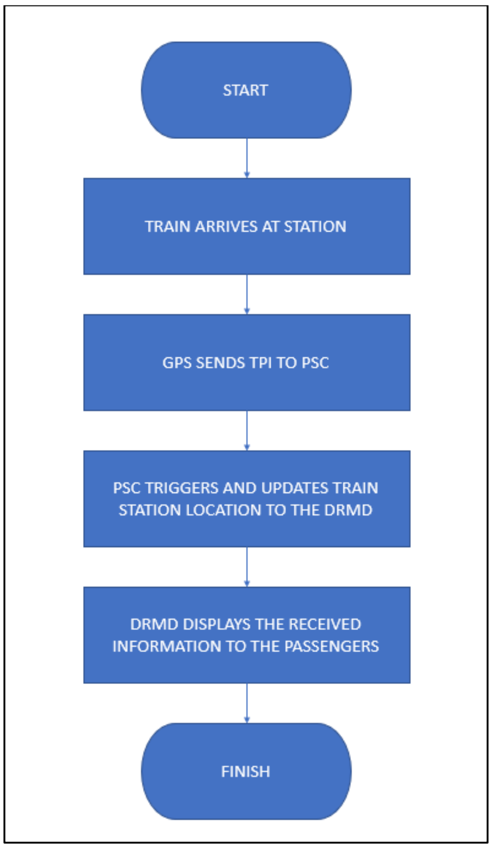

START
TRAIN ARRIVES AT STATION
GPS SENDS TPI TO PSC
PSC TRIGGERS AND UPDATES TRAIN STATION LOCATION TO THE DRMD
DRMD DISPLAYS THE RECEIVED INFORMATION TO THE PASSENGERS
FINISH

Figure 1: Flowchart of the GPS System

The system starts when the train is arriving at the station. The GPS antenna will receive signals from the satellite and the passes the information to a PA/PIS System Controller (this device is generalized as the official device name was trademarked by the company). The information was passed in form of message to the controller. This message informs the controller regarding the position of the train. Other than that, the PA/PIS System Controller also receives message from the Vehicle Communication Controller (VCC), informing whether the station is either terminus station or not, the status of the PIS devices, and the direction of the train route. The PA/PIS System Controller (PSC) will then update the information to the DRMD devices. Next, the DRMD devices will display Doors Opening Warning video and Door Closing Warning video on the side of the open door, Isolated Door.

Warning video on the faulty/isolated door, and Opposite Side Door Warning video on the side opposite of the opening door. Included in the DRMD display videos are the name of the station it has arrived at, and the name of the next station. For the next part of the system, it begins when the train is leaving the station. As the train leaves, the PSC will trigger the PIS and the DRMD to display the videos mentioned above. In addition to that, the DRMD will display a full line view video, showing all the stations on the LRT Kelana Jaya Line. This video will alternate with a zoomed in video of the next station until the train arrives at the next station. Finally, as it arrives, the first part of the system is repeated again.

To summarize the system, the GPS Router will pass the Train Positioning Information to the PA/PIS System Controller which will then update the information to the DRMD displays. The displays will then show the passengers the updated information corresponding to stations the train has arrived at. The PSC will also update the DRMD once the train leaves the station.

2.1 GPS Simulator

The testing of the system uses the Python language for the simulation of GPS, as mentioned before. The objective of the simulator is to broadcast the GPS signals or coordinate to the PSC. The simulated coordinates specify the location of the train as it travels along the service route.

To make this simulator from scratch is quite hard, hence an example from our company's simulator is used as a guideline for this project's simulator. The company's simulator functions similarly to what my simulator is supposed to do so all it takes is to understand how the simulator works and modify it a little bit as not to just copy and paste the simulator.

As shown in the block diagram below, this is how the GPS simulator will be implemented in the system.

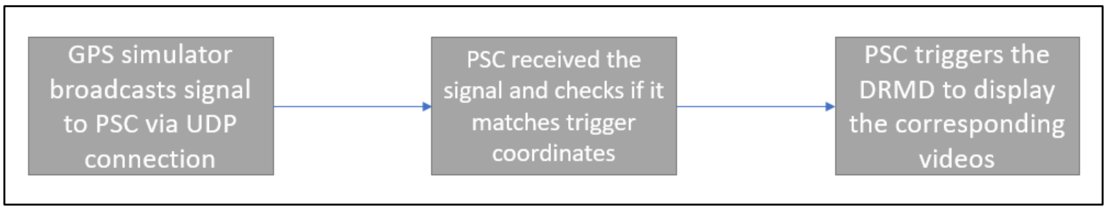

GPS simulator broadcasts signal to PSC via UDP connection
PSC received the signal and checks if it matches trigger coordinates
PSC triggers the DRMD to display the corresponding videos

Figure 2: Block Diagram of GPS system

3. GPS Simulator for TPI

3.1 LRT Kelana Jaya Line Coordinates.

For the GPS simulator, there are multiple pre-requisites step to take before starting with the simulator itself. First of all, for this simulator, the GPS signals are taken from GPS coordinates along the Kelana Jaya Line, which is, from Gombak to Putra Heights. To do this, Google Earth Pro is used to obtain all the latitudes and longitudes of the stations. The coordinates are in the form of degrees and minutes. In the Python, the codes I used can broadcast GPS signals in the form of decimal degrees. Therefore, by using a converter available online, all the coordinates are converted. Figure below shows the coordinates in Google Earth Pro.

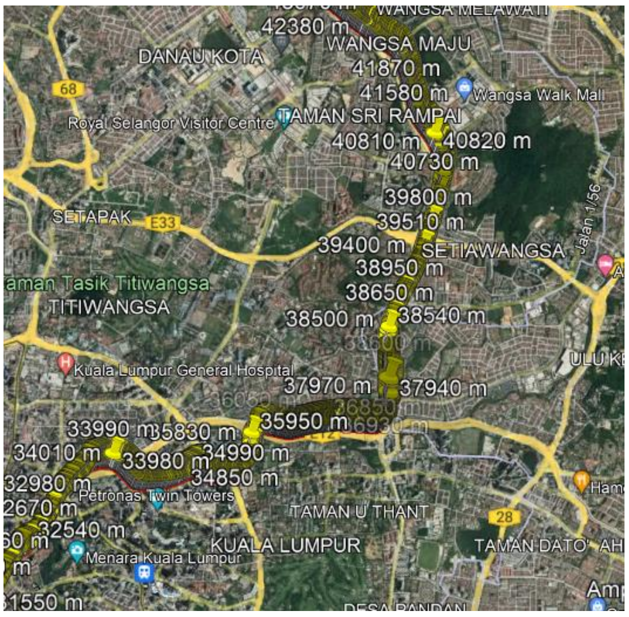
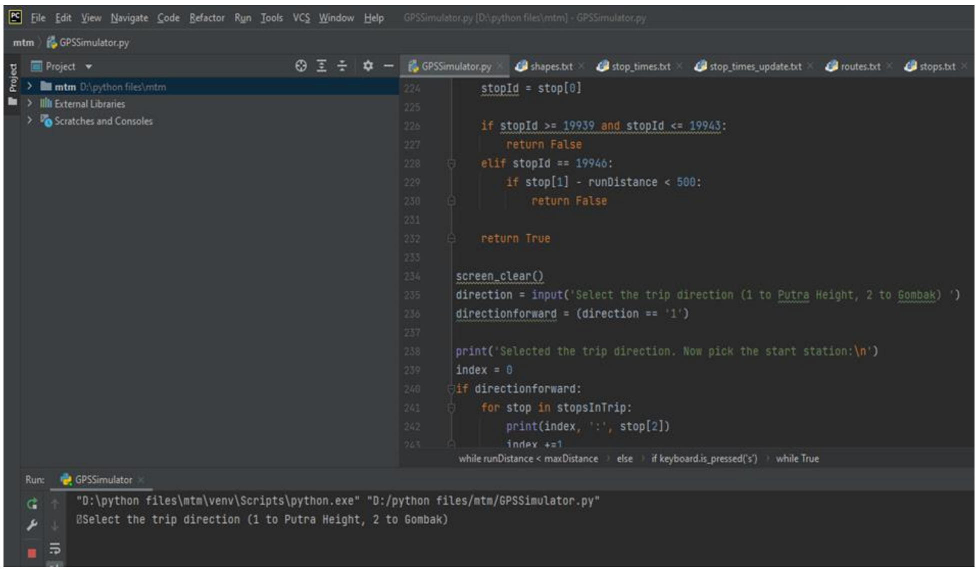


Figure 3: Google Earth Pro Coordinates

3.2 GPS Simulator Functions

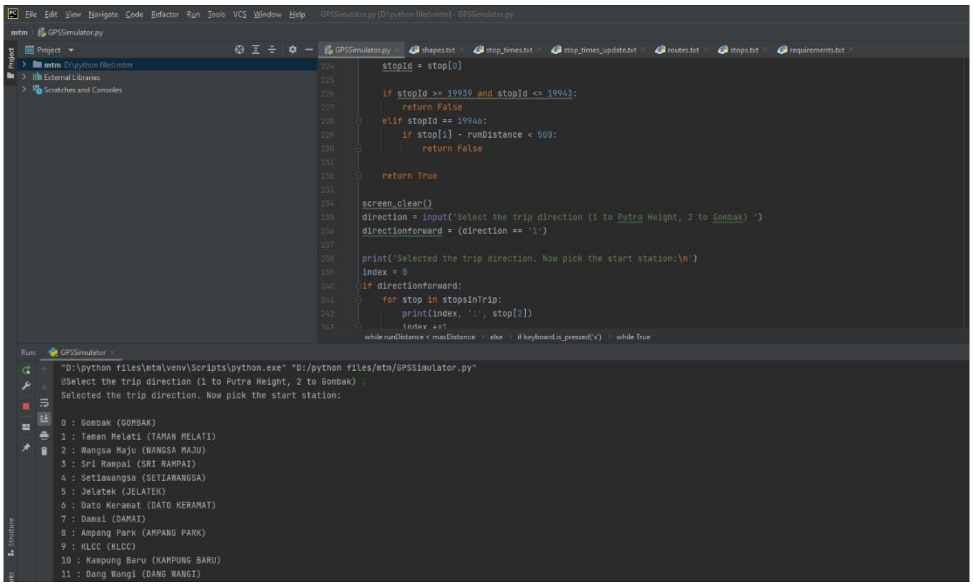
For the GPS simulator output, the code itself contains 450 lines of codes which can be referred in Appendix B. At first, from Figure 4, it will require users to choose the direction of the train, whether towards Gombak which is northbound or Putra Height which is southbound. After that, users will be prompted to input the name of the starting station as if they are really at that station as shown in Figure 10. At this point, the simulator will start broadcasting the GPS coordinates. Other than that, this simulator also includes the speed of the train, the distance to the station, the name of the next station, and the acceleration of the train. Figure 5 shows section of train station selection prompt.



```python
    stopId = stop[0]

    if stopId >= 19939 and stopId <= 19943:
        return False
    elif stopId == 19946:
        if stop[1] - runDistance < 500:
            return False

    return True

screen_clear()
direction = input('Select the trip direction (1 to Putra Height, 2 to Gombak) ')
directionforward = (direction == '1')

print('Selected the trip direction. Now pick the start station:\n')
index = 0
if directionforward:
    for stop in stopsInTrip:
        print(index, ':', stop[2])
        index +=1
```

```
"D:\python files\mtm\venv\Scripts\python.exe" "D:/python files/mtm/GPSSimulator.py"
Select the trip direction (1 to Putra Height, 2 to Gombak)
```

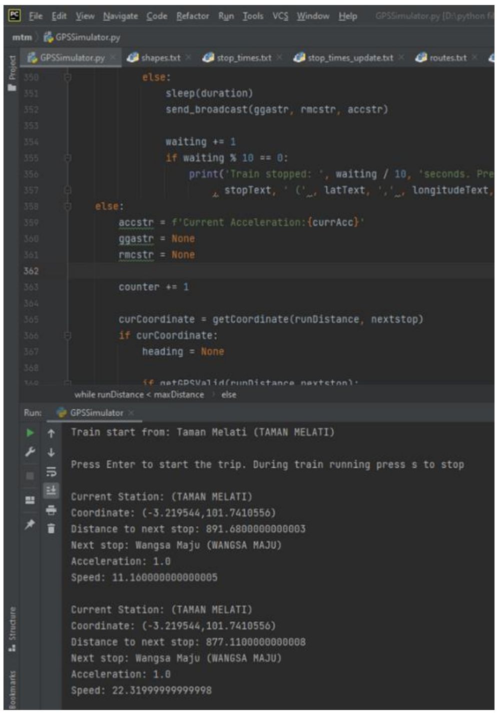
Figure 4: Trip direction prompt



```
"D:\python files\mtm\venv\Scripts\python.exe" "D:/python files/mtm/GPSSimulator.py"
Select the trip direction (1 to Putra Height, 2 to Gombak) 2
Selected the trip direction. Now pick the start station:

0 : Gombak (GOMBAK)
1 : Taman Melati (TAMAN MELATI)
2 : Wangsa Maju (WANGSA MAJU)
3 : Sri Rampai (SRI RAMPAI)
4 : Setiawangsa (SETIAWANGSA)
5 : Jelatek (JELATEK)
6 : Dato Keramat (DATO KERAMAT)
7 : Damai (DAMAI)
8 : Ampang Park (AMPANG PARK)
9 : KLCC (KLCC)
10 : Kampung Baru (KAMPUNG BARU)
11 : Dang Wangi (DANG WANGI)
```

Figure 5: Train Selection Prompt

The distance is in metres, the speed is in metres per second, and the acceleration is in metres per second squared. When nearing the next station, the train will decelerate and reduce its speed. Other than that, this simulator will also simulate the scenario when the train starts its departure, it will increase its acceleration for each second. As seen from Figure 6 below, the speed increases for each output after the train covers a certain distance.

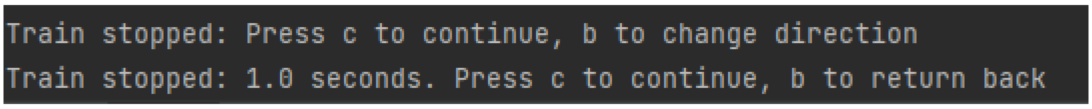


```
                else:
                    sleep(duration)
                    send_broadcast(ggastr, rmcstr, accstr)

                    waiting += 1
                    if waiting % 10 == 0:
                        print('Train stopped: ', waiting / 10, 'seconds. Pre
                              , stopText, ' (', latText, ',', longitudeText,
        else:
            accstr = f'Current Acceleration:{currAcc}'
            ggastr = None
            rmcstr = None

            counter += 1

            curCoordinate = getCoordinate(runDistance, nextstop)
            if curCoordinate:
                heading = None

                if getGPSValid(runDistance, nextstop):
```

```
Train start from: Taman Melati (TAMAN MELATI)

Press Enter to start the trip. During train running press s to stop

Current Station: (TAMAN MELATI)
Coordinate: (-3.219544,101.7410556)
Distance to next stop: 891.6800000000003
Next stop: Wangsa Maju (WANGSA MAJU)
Acceleration: 1.0
Speed: 11.160000000000005

Current Station: (TAMAN MELATI)
Coordinate: (-3.219544,101.7410556)
Distance to next stop: 877.1100000000008
Next stop: Wangsa Maju (WANGSA MAJU)
Acceleration: 1.0
Speed: 22.31999999999998
```

Figure 5: The acceleration of the train

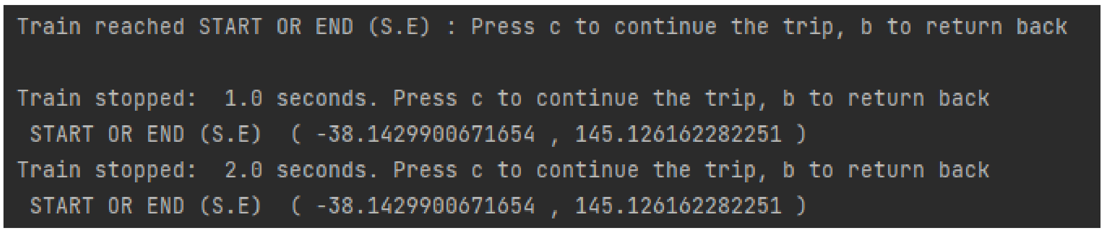
When the train is stopped, the user will be prompted to either continue onto the next station or change the direction as if the user is changing train which heads the other direction.



```
Train stopped: Press c to continue, b to change direction
Train stopped: 1.0 seconds. Press c to continue, b to return back
```

Figure 6: Screenshot of train stopped in Python

The interval of the broadcast is 1 second. This means that every 1 second, the simulator will broadcast the GPS signal like in Figure 6 to the PSC. After the train has reached the final station. An output stating that the train has stopped will appear as seen from Figure 8.



```
Train reached START OR END (S.E) : Press c to continue the trip, b to return back

Train stopped:  1.0 seconds. Press c to continue the trip, b to return back
 START OR END (S.E)  ( -38.1429900671654 , 145.126162282251 )
Train stopped:  2.0 seconds. Press c to continue the trip, b to return back
 START OR END (S.E)  ( -38.1429900671654 , 145.126162282251 )
```

Figure 7: Train reached final station

4. Conclusion

To conclude this project, a complete understanding of how GPS works the coordinates of train stations in LRT KJL is achieved. After that, a simulator sent by the company is successfully reviewed. Finally, a simulation model of the GPS-based TPI is successfully developed by using Python as its programming language and modified some parameters for self-use.

Acknowledgement

The authors would like to thank r2pMY and the Faculty of Engineering Technology, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] Crato, N. (2010). How GPS Works. Figuring It Out, 49-52.

[2] MRT Integration. (n.d.). Retrieved December 25, 2021, from http://mrt.com.my/lrt_kelana/index.htm/

[3] myrapid. (n.d.). Retrieved December 2021, 24, from https://myrapid.com.my/bus-train/rapid-kl/lrt/

[4] Thomas Albrecht, K. L. (2013). A Precise and Reliable Train Positioning System. International Conference on Intelligent Rail Transportation (ICIRT), 134-139.