UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

IOT system for Bluetooth based Origin-Destination studies

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

In partial fulfillment of the requirement for the

Degree of

MASTER SCIENCE

By

MHD Munzer Alsallakh

Norman, Oklahoma

2019

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/321133

IOT system for Bluetooth based Origin-Destination studies

A THESIS APPROVED FOR THE

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

_____

Dr. Hazem Refai, Chair

_____

Dr. Thordur Runolfsson

_____

Dr. Samuel Chang

©Copyright by MHD Munzer Alsallakh 2019

All Rights Reserved.

To my Soulmate, my better half and my ever-supportive wife, Bushra Aljbawi.

To the woman I owe everything, the woman that all languages don't have enough words to thank her, the woman that means all in life, my Mom.

To the man who supported, advised and was always there for me, my Father

To my father and mother in law who were a significant source of motivation and support.

To my brothers, extended family, friends, colleagues and those who supported me.

I dedicate this work to you all.

MHD MUNZER ALSALLAKH

Acknowledgments

Foremost, I would like to express my genuine gratitude and sincere thanks to my advisor *Dr.Hazem Refai*, for the continuous support of my M.Sc. study and research, for his enthusiasm, guidance and immense knowledge.

Besides my advisor, I would like to deliver my appreciation to my committee members, *Dr. Thordur Runolfsson* and *Dr. Samuel Chang.* I would also like to thank Michelle Farabough and Krista Pettersen for helping me editing this thesis.

Many thanks to my colleagues who participated in this research, specially: Jiamiao Zhao, Siraj Muhammad, Mohamat Irban bin Ali Kaja Najumudeen, Nabil Asfari, Mohammed Afifi and Samuel Chan.

A big thanks to all of my friends and OU family at OU-Tulsa who were a huge support for me during my study.

Table of Contents

Acknowledgments iv

Table of Contents v

List of Tables x

List of Figures xiii

Chapter 1 - Introduction 1

Chapter 2 - Background and Related Work 5

2.1 Smart cities and IoT 5

2.2 Traffic measurements and O/D collecting method 6

2.2.1 O/D Surveys 8

2.2.2 Anonymous cellular data tracking 8

2.2.3 Automatic License Place Recognition 9

2.2.4 Bluetooth MAC addresses matching 9

2.2.5 Inductive Loops for vehicle classification 11

Chapter 3 - System setup and configuration 13

3.1 System Overview 13

3.1.1 iVCCs IoT node 14

3.1.2 Access Point 16

3.2 Bluetooth Classic and Bluetooth Low Energy 20

3.2.1 BT Classic 20

3.2.2 BLE 21

3.2.3 Comparison between classic BT and BLE 21

3.2.4 BLE architecture 22

3.3 Antenna selection: 27

3.3.1 Selected Antennas' specification 29

3.3.2 Radio Frequency Antenna Gain Patterns 30

- 12-degree antenna 30

- 21-degree antenna 30
- 30-degree antenna 31
3.4 Commercial BTA Sniffer Integration 31
Chapter 4 – Testing and data collection 33
4.1 On-campus Roadside Test 33
4.1.1 Initial Setup 34
- Detected BTAs 34
- Matched BTAs 34
- Unmatched BTAs 35
- Repeated BTAs 35
4.1.2 Second Setup 37
- Detected BTAs 37
- Matched BTA 38
- Unmatched BTAs 39
- Repeated BTAs 39
- Matching algorithm 41
- BTA Randomization 42
4.2 On-campus Roadside Test using RF Multiplexer 43
4.2.1 Third Setup 45
- Detected BTAs 45
- Matched BTAs 45
- Unmatched BTA 45
4.2.2 Fourth Setup 46
- Detected BTA 46
- Matched BTA 46
- Unmatched BTA 46
4.3 On-campus Sideroad Test (Two Simultaneous Setups) 47

4.3.1 Fifth setup 48
4.3.2 Sixth setup 49
- Matched BTA 49
- Unmatched BTA 49
- Unique BTA 50
4.3.3 Data division 50
4.3.4 Number of Records 50
4.3.5 Number of matches 53
4.4 South Yale Setup 56
4.4.1 Seventh Setup 56
- Detected BTA 57
- Unique BTA 57
- Matched BTA 57
4.4.2 Eighth setup 58
- Detected BTA 58
- Unique BTA 58
- Matched BTA 58
4.4.3 Ninth setup 58
- Detected BTA 59
- Unique BTA 59
- Matched BTA 59
4.5 Integration of commercial system 59
4.5.1 Tenth Setup 59
- Detected BTA 60
- Unique BTA 60
- Matched BTA 61
4.5.2 Eleventh Setup 61

- Detected BTA 62
- Unique BTA 63
- Matched BTA 63
4.5.3 Twelfth Setup 63
4.5.4 Data processing 64
4.6 Highway Deployment 65
4.6.1 First field-test: 65
- Detected BTA 66
- Matched BTA 66
- Distribution 67
4.6.2 Second field-test: 68
- Data collection: 68
4.6.3 Third field-test 79
4.7 iVCC Sensors Highway Layout 84
Chapter 5 – Vehicle BTAs and Class Association 90
5.1 Classification algorithm 91
5.1.1 Data parsing 91
5.1.2 Classification 93
5.2 MAC to Vehicle assignment algorithm 95
5.2.1 Filtering BTAs 96
5.2.2 BTA/Vehicle assignment algorithm 98
5.3 Privacy 99
Chapter 6 – Conclusion and Future Work 100
Conclusion: 100
Future Work: 101
Appendix A: Algorithms and codes 106
1. BLE Dongle Software: 106

2. Finding matched BTAs 120
3. Parsing Iteris Data 122
4. Parsing iVCC sensor's data 123
5. Filtering BTAs Algorithm: 125

List of Tables

TABLE 1. CHARACTERISTICS OF DIRECTIONAL ANTENNAS 27
TABLE 2. SPECIFICATIONS OF SELECTED ANTENNAS 29
TABLE 3. NUMBER OF CAPTURED BTAS PER ANTENNA DURING MORNING TEST 34
TABLE 4. NUMBER OF BTAS CAPTURED BY ONLY ONE ANTENNA DURING THE MORNING TEST 35
TABLE 5. NUMBER OF CAPTURED BTAS DETECTED BY SYSTEM ANTENNA 37
TABLE 6. NUMBER OF UNMATCHED BTAS CAPTURED BY ONLY ONE ANTENNA 39
TABLE 7. EXAMPLE OF COLLECTED DATA 42
TABLE 8. NUMBER OF CAPTURED BTAS PER ANTENNA – MUX USED- (THIRD SETUP) 45
TABLE 9. NUMBER OF UNMATCHED BTAS PER ANTENNA – MUX USED- (THIRD SETUP) 45
TABLE 10. NUMBER OF CAPTURED BTAS PER ANTENNA – MUX USED- (FOURTH SETUP) 46
TABLE 11. NUMBER OF UNMATCHED BTAS PER ANTENNA – MUX USED- (FOURTH SETUP) 46
TABLE 12. NUMBER OF CAPTURED BTAS PER ANTENNA FOR THE FIFTH SETUP 48
TABLE 13. NUMBER OF CAPTURED BTAS PER ANTENNA FOR THE SIXTH SETUP 49
TABLE 14. NUMBER OF UNMATCHED BTAS BETWEEN THE FIFTH AND SIXTH SETUPS 49
TABLE 15. NUMBER OF UNIQUE BTA COLLECTED BY BOTH SETUPS 50
TABLE 16. COLLECTED BTA BY EACH PART OF THE DAY (SIDE12) 51
TABLE 17. COLLECTED BTA BY EACH PART OF THE DAY (SIDE21) 51
TABLE 18. COLLECTED BTA BY EACH PART OF THE DAY (LEARNING1) 51
TABLE 19. COLLECTED BTA BY EACH PART OF THE DAY (LEARNING2) 52
TABLE 20. COLLECTED BTA BY EACH PART OF THE DAY (MAIN12) 52
TABLE 21. COLLECTED BTAS BY EACH PART OF THE DAY (MAIN21) 52
TABLE 22. COMPARISON BETWEEN SIDE ANTENNAS 53
TABLE 23. COMPARISON BETWEEN LEARNING ANTENNAS 54
TABLE 24. COMPARISON BETWEEN MAIN ANTENNA 54
TABLE 25. UNIQUE BTAS COLLECTED BY SIDE12 PER PERIOD 55
TABLE 26. UNIQUE BTAS COLLECTED BY SIDE21 PER PERIOD 55
TABLE 27. UNIQUE BTAS COLLECTED BY LEARNING1 PER PERIOD 55
TABLE 28. UNIQUE BTAS COLLECTED BY LEARNING2 PER PERIOD 55
TABLE 29. UNIQUE BTAS COLLECTED BY MAIN21 PER PERIOD 55
TABLE 30. UNIQUE BTAS COLLECTED BY MAIN12 PER PERIOD 55
TABLE 31. NUMBER OF BTA COLLECTED BY 57
TABLE 32. NUMBER OF UNIQUE BTA COLLECTED BY 57

TABLE 33. NUMBER OF BTA COLLECTED BY THE 58
TABLE 34. NUMBER OF UNIQUE BTA COLLECTED BY THE 10DB REECE AND THE STANDARD REECE 58
TABLE 35. NUMBER OF BTA COLLECTED BY THE 10DB REECE AND 15DB REECE 59
TABLE 36. NUMBER OF UNIQUE BTA COLLECTED BY THE 10DB REECE AND THE 15DB REECE 59
TABLE 37. COMPARISON BETWEEN ITERIS- AND REECE-COLLECTED 60
TABLE 38. NUMBER OF BTA COLLECTED BY BOTH SYSTEMS USING OMNIDIRECTIONAL ANTENNA 60
TABLE 39. COMPARISON BETWEEN ITERIS- AND REECE-COLLECTED DATA UTILIZING 12-DEGREE ANTENNA 62
TABLE 40. NUMBER OF UNIQUE BTA COLLECTED BY BOTH SYSTEMS UTILIZING 12-DEGREE ANTENNA 63
TABLE 41. COMPARISON BETWEEN DETECTION TIMES OF PRE-KNOWN BTA 63
TABLE 42. NUMBER OF DETECTED BTAS BY BOTH SYSTEMS 66
TABLE 43. COLLECTED MACS STATISTICS (FIRST DAY) 70
TABLE 44. COLLECTED MACS STATISTICS (SECOND DAY) 70
TABLE 45. COLLECTED MACS STATISTICS (THIRD DAY) 71
TABLE 46. NUMBER OF MATCHED MACS BETWEEN THE TWO ANTENNAS (FIRST DAY) 71
TABLE 47. NUMBER OF MATCHED MACS BETWEEN THE TWO ANTENNAS (SECOND DAY) 71
TABLE 48. NUMBER OF MATCHED MACS BETWEEN THE TWO ANTENNAS (THIRD DAY) 71
TABLE 49. OUTPUT OF THE MACS FILTERING ALGORITHM (LEFT DZ TO RIGHT DZ) 72
TABLE 50. OUTPUT OF THE VIDEO-SENSOR TIMESTAMPS MATCHING ALGORITHM 73
TABLE 51. RESULTS OF VIDEO-SENSOR TIMESTAMPS MATCHING ALGORITHM 73
TABLE 52. OUTPUT OF THE BT-VEHICLE ASSIGNMENT ALGORITHM 74
TABLE 53. NUMBER OF DETECTED VEHICLES AND THEIR CLASSES (THIRD DAY - CAMERA) 75
TABLE 54. NUMBER OF DETECTED VEHICLES AND THEIR CLASSES (PART OF THE THIRD DAY - CAMERA) 76
TABLE 55. NUMBER OF DETECTED VEHICLES AND THEIR CLASSES (PART OF THE THIRD DAY - SENSORS) 77
TABLE 56. NUMBER OF DETECTED VEHICLES AND THEIR CLASSES (CAMERA) 81
TABLE 57. NUMBER OF DETECTED VEHICLES AND THEIR CLASSES (SENSORS) 82
TABLE 58. RESULTS OF THE REAL TIME MACS DIRECTIONALITY AND BTA-VEHICLE ASSIGNMENT ALGORITHMS 83

TABLE 59. PERCENTAGES OF ASSIGNED VEHICLES 83
TABLE 60. RECEIVED POWER -SENSOR FACES RECEIVER ANTENNA- ANECHOIC CHAMBER 85
TABLE 61. RECEIVED POWER -BATTERY FACES RECEIVER ANTENNA- ANECHOIC CHAMBER..... 85
TABLE 62. RECEIVED POWER -BLE MODULE FACES RECEIVER ANTENNA- ANECHOIC CHAMBER 86
TABLE 63. RECEIVED POWER -SENSOR FACES RECEIVER ANTENNA- OUTSIDE 87
TABLE 64. RECEIVED POWER -BATTERY FACES RECEIVER ANTENNA- OUTSIDE 87
TABLE 65. RECEIVED POWER -BLE MODULE FACES RECEIVER ANTENNA- OUTSIDE..... 88

List of Figures

FIGURE 1. SYSTEM OVERVIEW. 13
FIGURE 2. IVCCS SYSTEM OVERVIEW. 15
FIGURE 3. ILLUSTRATION OF DETECTION ALGORITHM[]. 15
FIGURE 4. BEAGLEBONE BLACK COMPONENTS. 17
FIGURE 5. REECE'S EXTENSION CAPE. 17
FIGURE 6. SIERRA'S RV50 INDUSTRIAL GATEWAY. 18
FIGURE 7. UBERTOOTH ONE BLUETOOTH SNIFFER 19
FIGURE 8. NORDIC NRF52840 BLE DONGLE. 20
FIGURE 9. BLUETOOTH LOW ENERGY STACK ARCHITECTURE. 22
FIGURE 10. GATT SERVER PROFILE ARCHITECTURE. 26
FIGURE 11. 12-DEGREE ANTENNA AGP. 30
FIGURE 12. 21-DEGREE ANTENNA AGP. 30
FIGURE 13. 30-DEGREE ANTENNA AGP. 31
FIGURE 14.VANTAGE VELOCITY TRAFFIC MONITORING CONCEPT. 32
FIGURE 15. VELOCITY FIELD UNIT OVERVIEW. 32
FIGURE 16. INITIAL SETUP WITH ANTENNA POV. 33
FIGURE 17. REPEATED BTAS FREQUENCY DURING THE MORNING TEST FOR THE MAIN ANTENNA. 36
FIGURE 18. REPEATED BTAS FREQUENCY DURING THE MORNING TEST FOR THE LEARNING ANTENNA. 36
FIGURE 19. REPEATED BTAS FREQUENCY DURING THE MORNING TEST FOR THE SIDE ANTENNA. 37
FIGURE 20. REPEATED BTAS FREQUENCY FOR THE MAIN ANTENNA DURING THE PEAK EVENING HOUR TEST. 39
FIGURE 21. REPEATED BTAS FREQUENCY FOR THE SIDE ANTENNA DURING THE PEAK EVENING HOUR TEST. 40
FIGURE 22. REPEATED BTAS FOR LEARNING ANTENNA FREQUENCY DURING THE PEAK EVENING HOUR TEST. 40
FIGURE 23. TWO-WAY, 2.4 GHZ RADIO FREQUENCY MUX. 43
FIGURE 24. SECOND TEST LOCATION WITH ANTENNAS DZS AND MUX. 44
FIGURE 25. TEST SETUP THREE FOR SIMULTANEOUS TESTING OF TWO SYSTEMS WITH DZS ANTENNA. 47

FIGURE 26. SETUP FOR TEST THREE. 48
FIGURE 27. FOURTH TEST SETUP LOCATIONS WITH ANTENNAS DZ 56
FIGURE 28. A DEPICTION OF THE FIFTH TEST SETUPS. 60
FIGURE 29. A DEPICTION OF THE TWO SETUPS WITH DIRECTIONAL ANTENNAS OF THE SIXTH TEST 61
FIGURE 30. DEPICTION OF THE TWO SYSTEMS OF SIXTH TEST. 62
FIGURE 31. LOCATION OF TWO SETUPS OF THE FIRST SETUP. 65
FIGURE 32. DISTRIBUTION OF BTAS DURING A 24-HOUR PERIOD AT AVC10. 67
FIGURE 33. DISTRIBUTION OF BTAS DURING A 24-HOUR PERIOD AT AVC68. 67
FIGURE 34. A DEPICTION OF THE LAYOUT OF THE SECOND DEPLOYMENT. 68
FIGURE 35. LOCATION OF THE SECOND DEPLOYMENT. 69
FIGURE 36. FHWA VEHICLE CLASSIFICATION. 70
FIGURE 37. DISTRIBUTION OF THE CLASSES PASSING THROUGH THE DZS (THIRD DAY -CAMERA). 76
FIGURE 38. DISTRIBUTION OF THE CLASSES PASSING THROUGH THE DZS (PART OF THE THIRD DAY -CAMERA). 77
FIGURE 39. DISTRIBUTION OF THE CLASSES PASSING THROUGH THE DZS (THIRD DAY - SENSORS). 78
FIGURE 40. LOCATION OF THE THIRD DEPLOYMENT. 79
FIGURE 41. LAYOUT OF THE THIRD DEPLOYMENT. 80
FIGURE 42. DISTRIBUTION OF THE CLASSES PASSING THROUGH THE DZS (CAMERA). 81
FIGURE 43. DISTRIBUTION OF THE CLASSES PASSING THROUGH THE DZS (SENSORS) 82
FIGURE 44. DIFFERENT PLACEMENT SCENARIOS FOR THE SENSOR INSIDE THE ENCLOSURE. 84
FIGURE 45. RECEIVED RSSI FOR DIFFERENT SENSOR POSITIONS (ANECHOIC CHAMBER). 86
FIGURE 46. RECEIVED RSSI FOR DIFFERENT SENSOR POSITIONS (OUTSIDE). 88
FIGURE 47. OVERVIEW OF DATA FLOW INSIDE THE SYSTEM. 91
FIGURE 48. FLOW DIAGRAM OF PARSING ALGORITHM. 92
FIGURE 49. EXAMPLES OF TIMESTAMPS SENT BY THE SENSOR. 92
FIGURE 50. DETAILED VEHICLES' CLASSES DESCRIPTION 95
FIGURE 51. FLOW DIAGRAM OF BTAS FILTERING ALGORITHM. 97
FIGURE 52. FLOW DIAGRAM OF THE BTA/VEHICLE ASSIGNMENT ALGORITHM. 98
FIGURE 53. DATA FLOW BETWEEN THE REECE AND THE IVCC SENSOR 107
FIGURE 54. FLOW DIAGRAM OF THE MATCHED BTAS ALGORITHM 121

Abstract

Designing and modelling a transportation system is a complicated, yet crucial task that demands comprehensive study of public needs. An important aspect of the process is specifying the characteristics of traffic schemes, which include vehicle classification, origin/destination (O/D), travel time (TT), and vehicle occupancy, in addition to other factors. A more thorough understanding of these factors will lead to improved transportation planning.

This thesis proposes the development of an Internet of Things (IoT) system that integrates two systems, namely Bluetooth (BT) identification and vehicle classification, for monitoring route choices per vehicle class. The extant system consists of one BT identification/vehicle classification unit deployed at an Oklahoma port of entry, along with a number of BT identification stations deployed at various locations across Oklahoma's roadways. As vehicles travel over magnetometer nodes, sensors measure changes in magnetic field (i.e., vehicle magnetic signature) for defining each vehicle's time of arrival and time of departure. Stated times will be used to estimate magnetic length of a passing vehicle for the purpose of classifying the vehicle. During this process, the BT ID of detected BT mobile devices in the vehicle is captured using BT identification stations.

Algorithms were developed to associate detected BT addresses to the corresponding vehicles with exceptional accuracy. BT addresses are planned to be sent alongside the vehicle group to a server where they will be matched by multiple stations as the vehicle travels on observed roadways. Hence, active monitoring of route choice and TT per vehicle class is achieved using only inexpensive BT stations.

Chapter 1 - Introduction

The accelerating expansion of modern cities is increasing pressure on transportation system infrastructure. Analyzing infrastructure performance is a pivotal task for both transportation and urban planners. To address mounting challenges, investigative methods—like TT and Origin Destination (O/D) studies—were developed to deliver enhanced knowledge to roadway planners. These studies play a crucial role in optimizing the performance and efficiency of transportation infrastructures. They also assist in facilitating the procedure of transporting people and goods from one place to another. Traffic studies focus on vehicle movement on roads and highways, pedestrian movement, and environmental effects on infrastructure conditions. O/D studies provide vital input for analyzing transportation initiatives, making it increasingly feasible to understand traffic patterns in an area of interest during a particular period of time.

Departments of transportation (DOTs) have commonly been focused on planning, managing, and maintaining the services they provide, relying on costly and unreliable manual processes for collecting traffic data. For example, DOTs have traditionally relied heavily on manual surveys for collecting O/D pairs. As such, the data might be subject to biased opinions and to errors. Also, this method is considered expensive in both time and cost. However, recent developments in technology have urged the discovery of less costly methods for collecting traffic data.

The use of IoT systems, for example, has helped resolve a number of issues challenging transportation systems. IoT is believed to have the ability to reshape the world's transportation systems. Most aspects of our transportation systems (e.g. traffic lights, vehicles, roads) are now equipped with one or multiple IoT nodes for sensing the

environment, analyzing the collected data and, sometimes, taking actions. IoT nodes transmit important information to traffic control sites and aid in decision-making based on the received data. Advancements in IoT technology have played a major factor in changing the volume, quality, and cost basis of collected data.

The work detailed in this research presents an overview of an IoT system architecture that delivers a cutting-edge solution for monitoring route choices per vehicle class by utilizing Bluetooth (BT) technology. The system collects BT addresses (BTAs) in various locations throughout the state, and then associates each with reported vehicle class via the intelligent vehicle counting and classification sensor (iVCCs). This sensor was developed at the WECAD lab at the University of Oklahoma as a state-of-the-art solution for detecting and classifying vehicles based on changes in the earth's magnetic field in the vicinity of the sensor relative to a passing vehicle.

The project uses two types of stations:

1. **Detection and classification stations**. These stations are located on a state's port of entry and collect BTAs using a BT sniffer through directional antennas. Alongside BTA detection, iVCCs nodes will be installed accordingly to detect and report vehicles' arrival and departure time stamps. This information will be sent via Bluetooth Low Energy (BLE) to an Access Point (AP) that resides on the roadside. Time stamps will then be used to classify vehicles into one of four groups to be then associated with the corresponding BTA.

2. **Detection stations:** These stations will be installed at various locations throughout the state and will be responsible for collecting BTAs that travel across their detection zones (DZs).

All stations will send their collected data to the cloud, where the BTAs are analyzed and matched across all the stations to extract the O/D data per vehicle.

An algorithm was proposed to collect BTAs traveling across DZs and to apply data analysis methods for predicting vehicles' lanes using the Received Signal Strength Indicator (RSSI) reported by the BT sniffer. Also, a communication scheme using BLE was developed to transmit data between the APs and the sensor nodes, and then use received data in the vehicle classification model.

This thesis is organized as follows. Chapter 2 provides the necessary background for describing the motivation behind the presented research, including the definition of smart cities and the features that characterize a smart city. An introduction to IoT and its ever-growing impact on different aspects of our lives is also featured. Origin destination and other traffic metrics, along with methods for collecting O/D data and related research work, are part of this chapter, as well. Chapter 3 provides an overview of various embedded systems used in the final IoT system. These include

- iVCCS nodes and the algorithm used to detect vehicles based on changes in the earth's magnetic field;
- Roadside Embedded Extensible Computing Equipment (REECE) system architecture, an overview of the system, and its components;
- Ubertooth One, along with the description of BT sniffer functionalities, as well as the directional antennas selection schema and the integration of the commercial BT sniffer;

- Nordic nRF52840 BLE dongle for facilitating communication between AP and iVCCs; and
- an introduction to BT and BLE.

Chapter 4 describes various setups used in BTAs data collection and the environments in which they took place (i.e., on-campus and highways). Also, this chapter provides a comprehensive comparison between tested setups and procedures for analyzing collected data. Chapter 5 provides an in-depth description of the algorithms used in this thesis project (i.e. vehicle classification algorithm and vehicle/BTA association algorithm).

Chapter 2 - Background and Related Work

2.1 Smart cities and IoT

Urban populations are expected to exceed 60% of the total world population. Due to rapid advancements in innovative technologies—especially in the field of IoT, smart cities have emerged as an important consideration for modern societies. The International Data Corporation (IDC) projects that in 2022, $158 billion will be spent on smart cities development [1]. Governments, companies, and academic research centers around the globe are racing to make discoveries in this field.

A smart city is defined as a city that uses smart computing technologies to connect all physical, computational, and social infrastructures (e.g., education, healthcare, real estate, and transportation systems), making them more intelligent [2][3]. Smart cities follow a forward-looking approach that focuses on increasing flexibility, transformability, and self-awareness of many aspects of the city [4].

The main tenant of developing a city smart is merging smart computing technologies to the city's infrastructure, providing the infrastructure with real-time awareness, enabling independent decision making, and integrating analytical and communication abilities [5]. Advancements in smart computing technologies—especially IoT—has made the adoption of smart cities more realistic and helped facilitate the process of making cities smarter.

IoT is a network of smart devices that can process data, sense, and sometimes interact with the surrounding environment. Most importantly, IoT devices are capable of interacting with other devices. Studies [6] [7] predict that the number of IoT devices will reach tens of billions by 2020, with a dramatic increase from 0.9 billion in 2009[8][48]

Having billions of devices connected to each other and to the internet will pose a significant challenge in several areas (e.g., Machine to Machine [M2M] communication), where traffic is likely to comprise 45% of the total internet traffic [9]. Of interest is that such challenges are urging researchers in both academic institutions and industries to thoroughly examine the scope of IoT.

Advances in IoT have created promising opportunities for solving a variety of everyday problems; they have also opened new horizons in various domains, including healthcare, agriculture, industry, and transportation. Although an impressive number of applications have been proposed in these domains, only a small portion have been developed and are available for use in everyday society. Of those in operation, the applications are believed to enhance the quality of every aspect of our lives [10].

Notably, the Internet has become an everyday fixture in modern society. An increasing number of devices are manufactured with communication abilities and embedded sensors. At the same time, the cost of technology fabrication continues to decrease. This synergy continues to boost the momentum of IoT adoption.

2.2 Traffic measurements and O/D collecting method

O/D data has always been one of several vital factors leveraged for improving the transportation infrastructure. O/D is utilized to characterize traffic patterns in a particular area of interest during a particular period of time. Knowledge from this data is considered decisive for transportation departments that are planning broad transportation infrastructure systems; building travel demand forecasting models; managing arterial conditions; analyzing traffic performance [11], [12]; and planning for freight transport [13]. Real-time O/D information aids in scaling down congestion, as it offers insights about

important factors like TT and traffic load. This information offers travelers alternative routes in an effort to avoid congested traffic conditions.

The ever-growing size of freight flowing across the US highways has led to a notable increase in truck traffic on highways. The dramatic increase is affecting both traffic safety and the condition of the transportation infrastructure. As a result, proper and special planning is necessary to mitigate these issues. Analyzing freight travel patterns using O/D data offers better insights about truck traffic flows and cargo types. Stakeholders can make the most of this data by utilizing it for managing goods transportation.

DOTs across the nation have conducted several studies (e.g., O/D, traffic count, TT, and others) to gain a better understanding of traffic patterns. Compared with other methods, O/D data provides insights not only about the number of vehicles on a road, but also about intended travel destinations and anticipated travel routes. Analyzing combined O/D pairs improves decision makers' choice of where to invest in transportation infrastructure [14].

Conducting O/D studies requires the ability to identify/re-identify vehicles. Currently, several technologies are employed by DOTs for collecting O/D data, including Automatic License Plate Recognition (ALPR), manual surveys, anonymous cellular data tracking, and BT MAC addresses matching. A study [15] by the Washington DOT investigated the effectiveness of multiple O/D data collection methods by comparing the accuracy and reliability of such technologies.

Additional information about methods used by research institutes and DOTs for collecting O/D data is provided in the sections below.

2.2.1 O/D Surveys

An O/D survey serves a way of collecting travel information for determining traffic patterns in a project study area. Surveyors utilize a number of methods for delivering questionnaires (e.g., home or roadside interviews, postcards). Survey responses provide crucial information about the journey that other methods cannot, like start and end point; anticipated routes; frequency; purpose; driver age/gender; and vehicle ownership. However, O/D survey data collection also has several drawbacks:

- Costly in both time and money,
- Cannot provide real-time data,
- Depends heavily on the cooperation of respondents, and
- Imperfect in both spatial and temporal scales of data [16]

2.2.2 Anonymous cellular data tracking

Cellular network data allows researchers to analyze mobility patterns via smart phones that are turned on during a traveler's journey. Cellular data is used to track travelers by examining the communication towers travelers are connected to. This method for studying O/D pairs has the advantage of protecting driver privacy, as no private information is exposed during testing. Another benefit is that data is consistently generated, ensuring data retrieval at any time during the study period. However, cellular data tracking lacks the ability of identifying vehicle groups or capturing travel purpose, as there is no way of extracting this information [17].

2.2.3 Automatic License Place Recognition

ALPR technology utilizes high-speed, automated camera systems installed on streetlights, traffic lights, and toll gates. Cameras capture and then process license plate characters using optical character recognition (OCR), along with location, date, and time stamp [18]. Data is sent to a remote server for matching plates as they pass different locations, and then deriving O/D pairs.

In addition to O/D data, ALPR provides critical information, like TT, number of cars passing a specific route, and vehicle class [19]. In addition to privacy issues, this method is also considered costly and complex.

2.2.4 Bluetooth MAC addresses matching

BTAs collection has emerged as a leading technology in the field of passive data collection techniques for traffic studies, primarily due to its inexpensive cost, ease of implementation, and the effectiveness of collected data [20]. Also, the increased use of BT technology in devices—such as smart phones, wireless headsets, laptops, and integrated BT vehicle systems—has favored the use of device IDs for traffic studies [21].

Bluetooth Traffic Monitoring Systems (BTMSs) can identify/reidentify vehicles across a wide range of interests, providing the ability to extract important information, like TT and O/D pairs [22]. Murphy et al. (2002) verified and confirmed that BT devices are discoverable in moving vehicles. Notably, however, BT sensors must be distributed in a thoughtful way to ensure study area coverage while maintaining the low-cost benefits of BTMS. Notably, BT sensors obtain a high level of accuracy detection. Khliefat et al. have proposed a method for optimizing the localization of BT sensors for traffic studies in urban networks.

BTM stations are spread throughout areas of interest and equipped with BT sniffers for capturing unique Media Access Control (MAC) addresses. This 48-bit identifier floats around connected devices in different ranges (i.e., coverage areas) [25]. The first three octets of the MAC address are referred to as an Organizationally Unique Identifier (OUI), which are assigned by the Institute of Electrical and Electronics Engineers (IEEE) and provide information about the BT chip manufacturer. The last three octets are assigned by the manufacturer and record the unique addresses of the devices [26].

BT sniffers scan the spectrum to detect and temporarily store all MAC addresses in a given detection zone. Capturing a MAC address at different locations provides the opportunity to calculate a vehicle's TT, speed, and O/D data as the vehicle travels from one zone to the other [27].

In 2007, a group of researchers designed a portable BT monitoring unit for measuring highly accurate TT and O/D data. The BT unit collects MAC addresses from vehicles passing through a detection zone, and then matches the MAC address with information gathered at different locations around the studied area [28]. Tsubota et al. studied the detection probability of MAC addresses from moving Bluetooth devices and the factors that affect this probability. The researchers also designed a model for estimating the detection probability based on the position of the installed sniffers and the speed of the experimental cars. Stevanovic et al. studied the accuracy and reliability of Bluetooth sniffers when measuring Arterial TTs, and authors examined the change of antennas attached to the sniffer by positioning the Bluetooth device inside the vehicle. This permitted them to discover the effect of vehicle speed on the detection procedure.

Several studies have compared Bluetooth traffic data gathering performance with that of traditional methods. Combined, these studies suggest that TT reported by both BTMSs and loop detectors are nearly identical [31], and that average TT estimated by BTMSs were 4-7 % higher than that of ALPR [32].

2.2.5 Inductive Loops for vehicle classification

Inductive loops are considered one of the most reliable vehicle classification technologies. These systems typically consist of a wire and an electronic detection unit, wherein the wire is coiled to form a loop that can be shaped into a square, circle, or rectangle and then installed under the roadway surface [33]. The electronic unit transmits an AC current through the wire loops at frequencies between 10 and 200 kHz, causing a magnetic field around the wire. Inductive loops act as a metal detector, so that when a vehicle passes above the loop, the system acts as the core of the coil and causes an increase in inductance. However, this effect (i.e., ferromagnetic sensing) is not considered a direct method for counting the vehicle. Instead, this distinction is given to eddy currents induced in the loops, because the vehicle itself has an additional effect on the inductance of the loop. In this way, levels decrease to the point at which the detector can sense a change in inductance [34].

Vehicle detection loops are used to detect a vehicle passing a certain point on a roadway, which could be utilized in any number of applications, like measuring the occupancy of a parking garage, leveraging a control trigger for access gates, or triggering a ticketing system [35]. However, the generated magnetic signatures, if saved and analyzed, may offer important data like the speed of the vehicle, number of axels, and the

length of the vehicle. This leads to a highly accurate classification of the passing vehicles. It is important to note that inductive loops are not considered helpful for O/D studies due to the fact that they don't uniquely identify each vehicle, which is the key factor in studying vehicles routes for passengers.

Chapter 3 - System setup and configuration

3.1 System Overview

Assimilating a BT identification system with a vehicle detection and classification system for collecting O/D measurements per vehicle class requires the integration of several hardware and software systems. Figure (1) illustrates a conceptual diagram of the various embedded systems used in the design of the developed system presented herein.

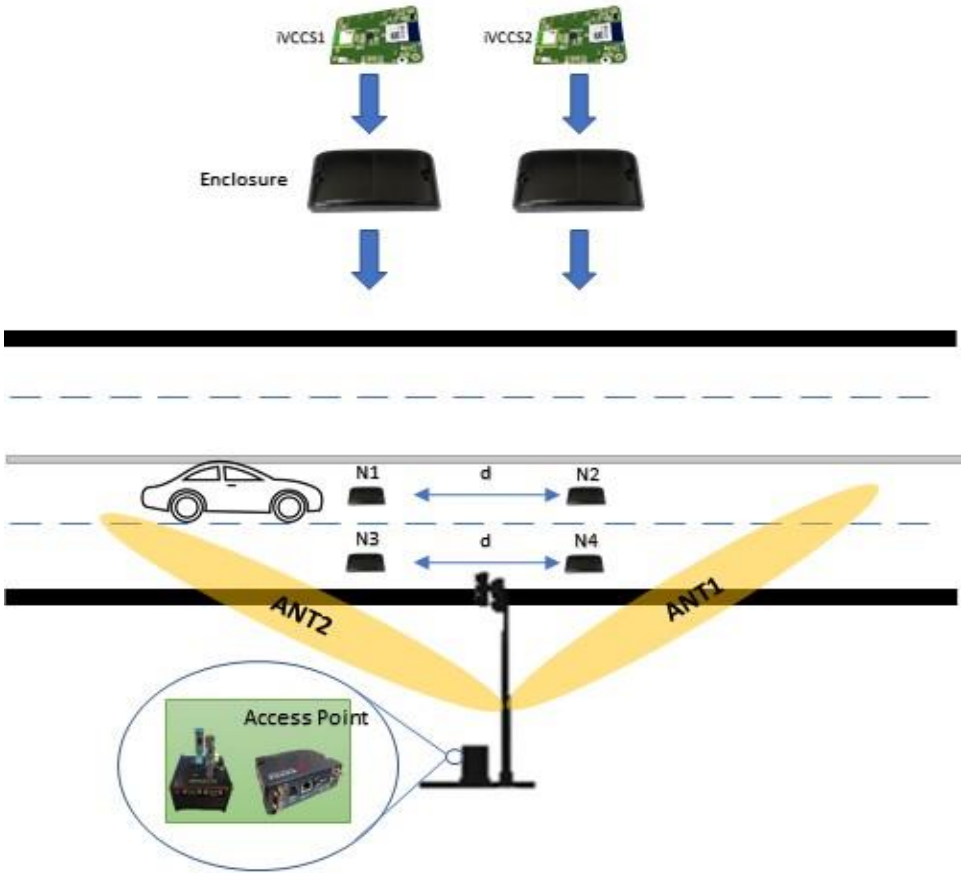


Figure 1. System overview.

Two iVCCs are deployed on each lane. Sensor nodes report time of arrival and departure to the roadside AP. Subsequently, the AP will classify the passing vehicle based on reported times, and then try to attach each vehicle group to BTAs that were collected

using the Ubertooth. Following is a description of the different hardware systems composing the system.

3.1.1 iVCCs IoT node

The iVCCs is a low-cost, low-power, flexible IoT system designed to accommodate smart cities applications leveraging a compact 46 x 30 x 6 mm design and state-of-the-art components. The sensor is built around the nRF52840 SoC, which has the following features [36]:

1. 32-bit ARM® Cortex™-M4 CPU with floating point unit
2. Fully multiprotocol SoC that supports BT 5, BT mesh, Thread, Zigbee, 802.15.4, ANT, and 2.4 GHz proprietary stacks
3. Support for numerous digital peripherals and interface (e.g., High-speed SPI, Quad SPI, I2S, USB, and others)
4. Exceptionally low energy consumption—as low as 1.5 µA at 3 V

The system is equipped with sophisticated algorithms that utilize the six-freedom degree inertial KMX62 sensor for detecting and counting vehicles passing its detection zone. The detection algorithm commences by monitoring and analyzing changes in the magnetic flux, based on the metal structure of a passing vehicle. This information is reported by KMX62. The controller then calculates the magnitude of the three reported magnetic axes (x, y, z) and advances them to the algorithm that uses predefined thresholds to detect the arrival and departure of a vehicle [37]. After that, the sensor reports times to the REECE using BLE. Figure (2) shows the top and bottom sides of the system. Figure (3) shows an overview of the system detection algorithm.

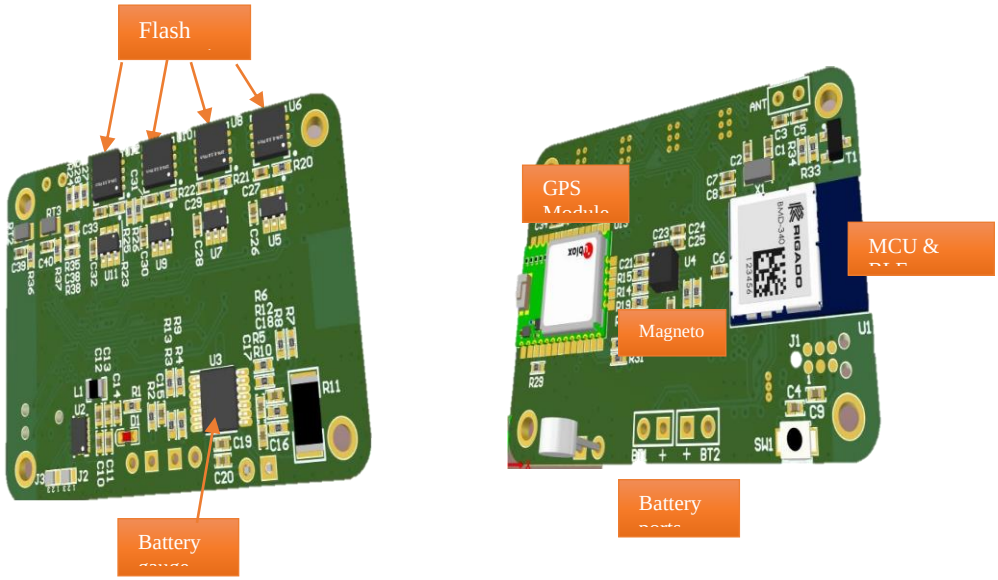


Figure 2. iVCCs system overview.

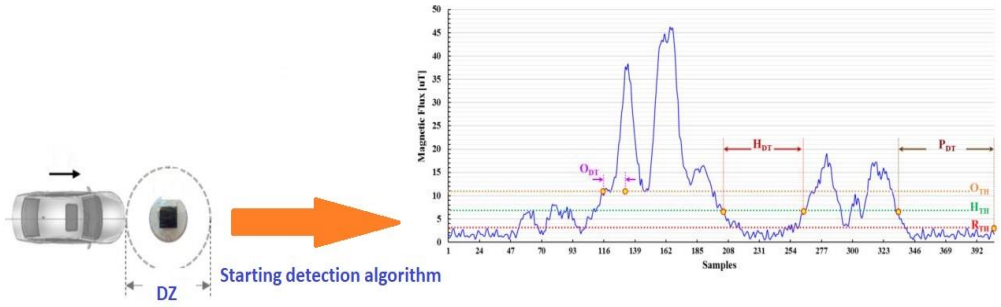


Figure 3. Illustration of detection algorithm[].

An iVCCS can synchronize its real-time clock (RTC) to a specific time zone using the Quectel's L96 GPS module, which is provided with an embedded chip antenna and supports the reception of up to three GNSS systems (GPS + GLONASS + Galileo). These systems are equipped with 33 tracking channels and 99 acquisition channels [38]. The sensor stores its data in the four on-board, 64 Mb flash memory unit to protect against loss

of connection with AP. The sensor is also equipped with a TI bq35100 battery gauge for measuring connected battery capacity.

3.1.2 Access Point

The access point is composed of several different systems, as detailed in the sections below.

3.1.2.1 REECE Unit

The REECE is a low-power, Linux-based, embedded system designed to collect, and process data captured by connected sensors through several available interfaces (e.g., USB, RS232) [39]. The system also communicates with the cloud using an internet gateway (e.g., Sierra's RV50) connected through USB. REECE system components include:

- **Beaglebone Black**: an open-source, low-power development board based on a TI AM335x Arm Cortex A8 CPU. The board also has two PRU 32-bit microcontroller units, 512MB DDR3 RAM, 4GB 8-bit eMMC on-board flash storage, and a 3D graphics accelerator. On the connectivity side, Beaglebone Black is equipped with an Ethernet, HDMI, and client/host USB ports [40]. Following is an overview of the Beaglebone black board.

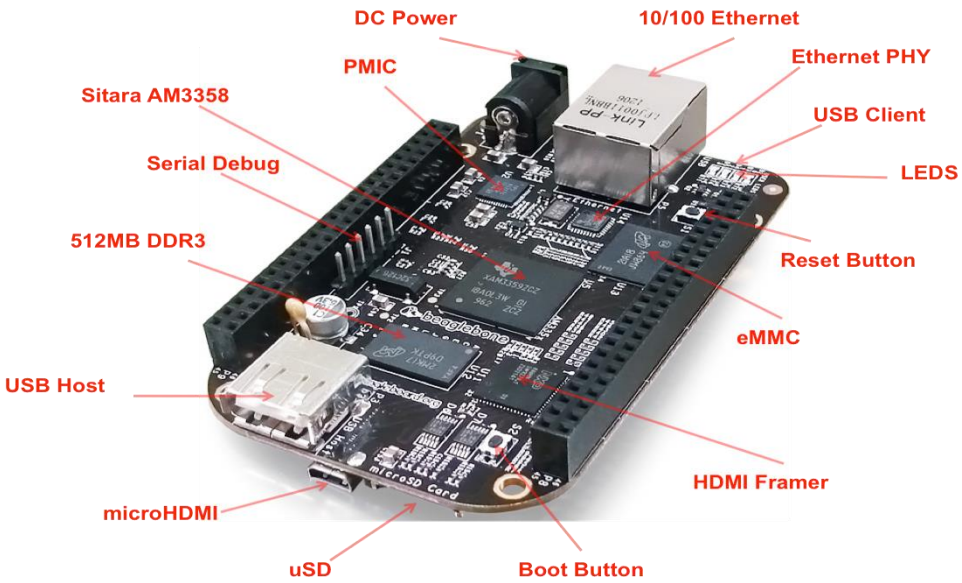


Figure 4. Beaglebone Black components.

- **Extension cape:** designed by Innovative Traffic Systems & Solutions (ITSS), LLC to further extend the features of the Beaglebone, including two RS232 connectors and four USB ports, as well as enable easy access to needed functionalities.

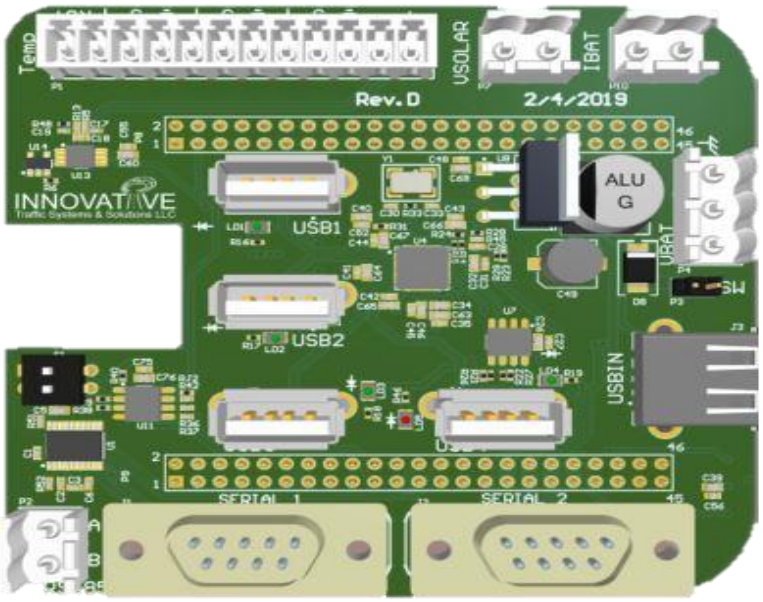


Figure 5. REECE's extension cape.

3.1.2.2 Sierra's RV50

AirLink® RV50 is the industry's lowest-power and most-rugged LTE gateway, designed to connect critical assets and infrastructure. The RV50 provides real-time remote connectivity for SCADA, distribution management systems, and metering. With a very low power consumption feature, the RV50 dramatically reduces infrastructure costs when running on battery or solar power. It also provides programmability for edge computing applications and uses the ALEOS Application Framework (AAF). Features in this gateway include [41]:

- LTE-advanced performance at 2G power consumption, using less than 1 watt and making it ideal for solar-powered applications
- Ruggedized, industrial grade form factor
- Edge computing application enabled with ALEOS Application Framework (AAF)
- Remote, secure network management in either the cloud or the enterprise



Figure 6. Sierra's RV50 industrial gateway.

RV50 will provide internet connectivity to the REECE using Ethernet, thus enabling it to send stored data to the cloud.

3.1.2.3 Ubertooth One

Ubertooth is a low-cost, open source 2.4 GHz wireless platform designed by Mike Ossmann at Great Scott Gadgets in 2011. The technology offers an affordable, off-the-shelf platform for monitoring and analyzing data packets. Ubertooth is built around the LPC175x ARM Cortex-M3 microcontroller and uses TI CC2400 as a wireless transceiver. Ubertooth operates within the 2.4 GHz Industrial, Science, and Medicine (ISM) band with a narrow bandwidth of 1MHz. This feature allows it to capture both BT classic and BLE packets—decode captured packets; find the master BTA; measure the RSSI, noise and signal-to-noise ratio (SNR); and report number of errors in captured packets [42].



Figure 7. Ubertooth One Bluetooth sniffer.

3.1.2.4 Nordic Dongle

Nordic nRF52840 Dongle (PCA10059) is a low-cost, versatile USB development board for BLE, ANT, 802.15.4, and user-proprietary 2.4 GHz applications that uses the nRF52840 SoC. The dongle has the following key features [43]:

- RF52840 flash-based ANT/ANT+™, BLE SoC solution
- Button and LEDs for user interaction
- 15 GPIO available on a castellated edge
- Onboard USB bootloader with buttonless support

- USB support

  The dongle was used to enable BLE on the AP. Notably, BLE is the communication protocol both iVCCs and AP will use to wirelessly exchange data. Nordic nRF52840 Dongle was chosen for several reasons, including:

- Identical microprocessor and Bluetooth stack, as used in the sensor
- Extremely low cost
- Supports BT 5
- Two operating modes—long range and short range
- Compact size

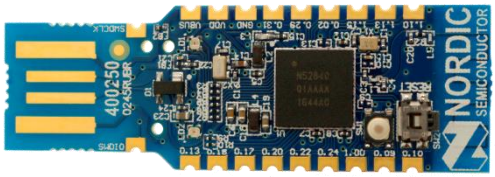


Figure 8. Nordic nRF52840 BLE dongle.

3.2 Bluetooth Classic and Bluetooth Low Energy

3.2.1 BT Classic

BT is a universal wireless communication technology that enables both portable and fixed devices to transmit and receive data over short ranges. BT operates in the 2.4 GHz ISM band and works within a wireless personal area network (WPAN). BT defines two types of connectivity topologies, namely piconet and scatternet. A Piconet WPAN is established by a **Master** device that initiates the connection and one or more peripherals (i.e., **Slaves**). Each unit can simultaneously communicate with up to seven units for each piconet. Moreover, each unit can simultaneously belong to several piconets. A scatternet

group is composed of operational BT piconets that are overlapped in time and space. BT devices can simultaneously be members of several piconets, thus increasing the possibility that data can be exchanged beyond the coverage area of each piconet [44].

3.2.2 BLE

BLE (or BT smart) is an intelligent, battery-friendly version of classic BT technology (i.e., BR/EDR). This WPAN technology operates in the 2.4 GHz ISM and was first introduced in 2011 by the BT Special Interest Group (SIG) to provide low power solutions for IoT applications for devices that are powered by a small battery for long periods. BLE is used when small amounts of data are sent at lower rates up to 1Mbps (e.g., healthcare, fitness, and home automation applications).

The following sections provide a comparison between the BT classic technology and BLE.

3.2.3 Comparison between classic BT and BLE

Compatibility: BLE is not backward compatible with the BT Classic, although both technologies can be implemented in the same chipset. This allows the device to operate as a dual-mode device when needed.

Power consumption: Ultra-low power consumption is the primary motivation behind designing BLE, as it permits device longevity from several months to several years while operating on a coin-cell battery. BT Classic is used for streaming and data transfer applications, which require the device to operate on larger batteries. As such, classic devices are considered power hungry applications.

Range: BT Classic has a longer communication range (up to 800 feet in BT 5.0) than BLE, which is affected by its low power consumption.

Throughput: Signaling rate of BLE is 1Mbps. However, practical transfer rate is between 100 and 250 Kbps. BT Classic v 5.0, on the other hand, promises a transfer rate of 2 Mbps.

Number of slaves: BLE Master can hold communication links for a greater number of slaves than BT Classic, which is limited to seven active slaves. Maximum number of slaves in BLE depends on the stack used and the application preserved memory of the central device.

Radio interface: BLE operates in the same spectrum range as BT Classic (i.e., 2.4 ~ 2.4835 GHz). However, whilst BT Classic operates within 79 1-MHz radio frequency (RF) channels, BLE operates within 40 2-MHz RF channels.

Connection time: BLE utilizes three of its 40 channels for discovery purposes, allowing it to initialize connections quicker than BT Classic, which utilizes 32 channels.

3.2.4 BLE architecture

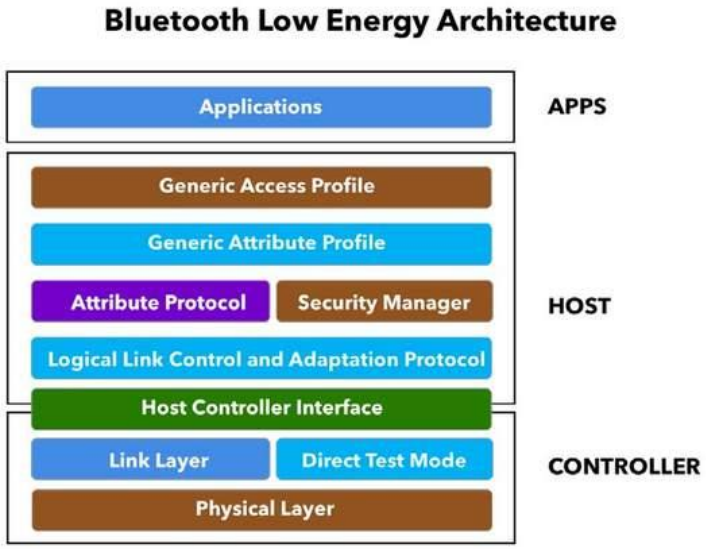


Figure 9. Bluetooth Low Energy stack architecture[54].

Physical layer: The physical layer is responsible for modulating/demodulating the RF radio holding the data. BLE uses adaptive frequency hopping as a modulation technique, as well as lazy acknowledgment, 24-bit CRC, and 32-bit integrity check.

Link layer: The link layer plays an intermediate role between the physical and higher layers, providing the latter a way to interact with the radio [45]. This layer acts as a five-state machine (e.g., standby, advertise, scan, initiate, scan), which will be discussed below in the Generic Access Profile (GAP) section.

Direct test mode: The direct test mode validates radio operations at the physical level (e.g., frequency offset and drift, modulation characteristics, and receiver sensitivity, among others [46]).

Host Controller Interface (HCI): HCI is a standard protocol defined by BT specifications that facilitates communication between the host and controller stacks.

Logical Link Control and Adaptation Protocol (L2CAP) layer**:** The L2CAP layer provides connectionless data services to higher layers, translating protocols from higher levels and sending them through BLE packets to lower levels [45].

Generic Access Profile (GAP)**:** The GAP layer is the cornerstone for interoperation between BLE devices, as it provides a framework for all BLE devices to follow for facilitated communication. The framework includes [45]:

Device roles. Various roles in which a BLE device can play during the communication procedure, including:

a- Broadcaster: device that only sends advertisement data to nearby devices without receiving from/connecting with any other device

b- **Observer:** device that only listens to devices sending advertising packets, without connecting to the devices

c- **Central:** observers that initiate and maintain connections with other devices

d- **Peripheral:** device that sends advertising packets to other devices and accepts connection requests from central devices, subsequently establishing a link with these devices

BLE devices can fulfill multiple roles at the same time (e.g., device can be central and send connection requests to peripherals in one link, as well as peripheral and send advertising packets on another link).

Advertising and scanning procedures

a- **Advertising**: BLE peripherals send connection parameter "advertisement packets" on one of three primary advertising channels (i.e., 37, 38, and 39) and are sent on prespecified intervals.

b- **Scanning**: Centrals willing to initiate connection will search three primary advertising channels for advertising packets. Once connectable devices are discovered, connection requests are sent to the peripherals. Hence, if a peripheral is available and ready to connect, a connection link will be created automatically after receiving the connection request, and the peripheral will immediately stop advertising. After both devices start sending data packets, the central device will then be referred to as a *Master* and the peripheral as a *Slave.*

The Generic Attribute profile (GATT)

The GATT layer describes the hierarchical data structure exposed to connected BLE devices as it's defined in the BT specifications. The GATT role commences once two devices are connected. GATT layer has two roles:

1. **Server**: device that exposes its data and accepts requests, commands, and confirmations from the GATT client, and
2. **Client:** devices connected with servers that read exposed data.

Note that BLE devices can simultaneously function as both server and client. GATT manages important BLE features that are pivotal for establishing connections.

Characteristics: lowest GATT data structure representing a data point desired to be exposed.

Services: logic entities containing a group of data points (i.e., characteristics). Note that a *service* can contain one or more characteristics, each distinguished by a unique ID (**UUID)**. UUIDs are characterized as either 128-bit for custom built services or 16-bit for SIG predefined services (e.g., Generic Access 0x1800, Blood Pressure 0x1810 and Indoor positioning 0x1821).

For example, indoor positioning has multiple characteristics including, latitude, longitude, local east coordinate, and local north coordinate.

- **Profiles:** conceptual entity not existing on the BLE device; each is considered a group of services that were compiled by either Bluetooth SIG or by developers.

More specifically, following are features of the BLE dongle after programming.

GAP role: central

GATT role: client

Number of allowed peripherals: 8

Transmit and receive buffer size: 256 bytes

Services: Nordic UART service with a UUID of 0x0001

Characteristics: received data, and transmitted data

Used microprocessor peripheral: application designed for available USB port and to program the device to act as virtual COM. This feature facilitates the operation for sending received data to REECE.

The complete code will be found in Appendix A.

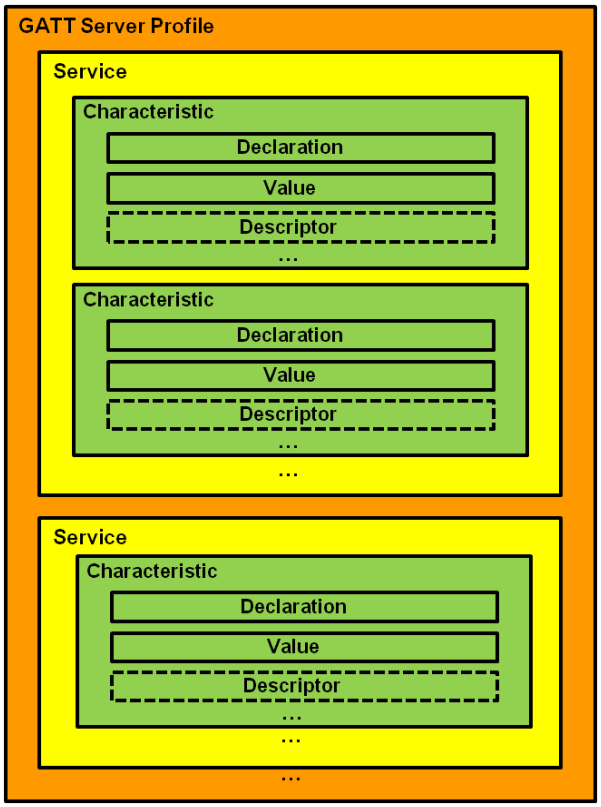


Figure 10. GATT server profile architecture [55].

3.3 Antenna selection:

For the purpose of this thesis, directional antennas will be used to enable the Ubertooth of capturing BTAs of vehicles passing through the detection zone in each site, and then reporting this information to the REECE. Because of the pivotal role antennas play in this project, careful attention was devoted to selecting an appropriate antenna for mounting it to the system to collect data. Antenna selection for capturing BTA is crucial for successful detection, as the antenna is considered a primary factor affecting detection probability in a detection zone. Antenna selection impacts collected data effectiveness, as well as the way in which iVCC sensors are assigned to road segments. Because the proposed system will utilize directional antennas, as opposed to omnidirectional antennas, a number of characteristics must be carefully studied. Potential use of the following antennas was investigated, as shown in Table 1.

Table 1. Characteristics of Directional Antennas

Company	**L-com**	**L-com**	**L-com**	**Afar**
Type	Grid	Grid	Grid	Parabolic grid reflector
Gain	30 dBi	26.5 dBi	24 dBi	24 dBi
3 dB angle (horizontal beam-width)	5.3 deg	6.5 deg	8 deg	8 deg
Dimensions	59" Dia.	47.2 x 35.43 inches	39.5 x 23.5 inches	42 X 24 inches

Company	**L-com**	**Afar**	**L-com**	**L-com**
Type	Grid	Flat Panel	Enclosed Yagi	Enclosed Yagi
Gain	20 dBi	18 dBi	14.5 dBi	12 dBi
3 dB angle (horizontal beam-width)	12 deg	21 deg	30 deg	45 deg
Dimensions	15.7" x 23.6 inches	12.4 x 12.4 x 1 inches	18.2 x 3.0 Ø inches	11.2 x 3.0 Ø inches

In addition to Afar's 21-degree antenna, L-com's 12-degree and 30-degree antennas, were selected as a result of their compact size, their rather small beam-width, and their high gain.

3.3.1 Selected Antennas' specification

Table 2. Specifications of Selected Antennas

Company	L-com	Afar	L-com
Type	Grid	Flat Panel	Enclosed Yagi
Frequency range	2.4 – 2.5 GHz	2.4 - 2.5 GHz	2.4 - 2.5 GHz
Gain	20 dBi	18 dBi	14.5 dBi
horizontal beam-width	12 deg	21 deg	30 deg
Vertical Beam-width	17 deg	21 deg	30 deg
VSWR	< 1.5:1	1.5: 1	< 1.5:1
Impedance	50 ohms	50 ohms	50 ohms
Dimensions	15.7" x 23.6 in	12.4 x 12.4 x 1 in	18.2 x 3.0 Ø in
Weight	3.3 lbs.	3.3 lbs.	1.8 lbs.
Wind loading	100 mph->20 lbs. 120mph->31 lbs.	100 mph->34.7lbs. 125 mph ->54.2 lbs.	100 mph->11.4 lbs. 125mph->17.8 lbs.
Max input power	100 watts	100 watts	50 watts

3.3.2 Radio Frequency Antenna Gain Patterns

Antenna Gain Pattern (AGP) is an important antenna feature, as it characterizes antenna gain with respect to directionality and provides information regarding effects of directional antenna sending and receiving angles. Following are AGPs for selected antennas.

- 12-degree antenna

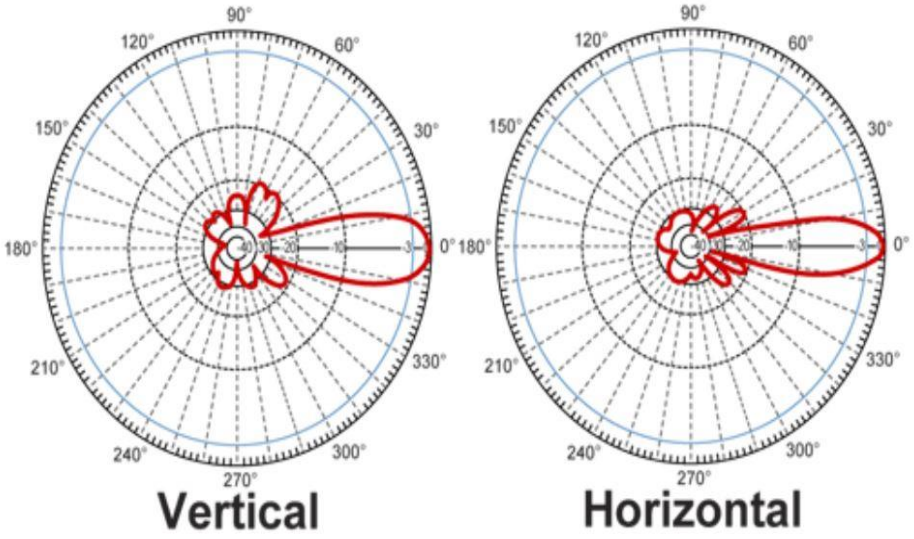


Figure 11. 12-degree antenna AGP.

- 21-degree antenna

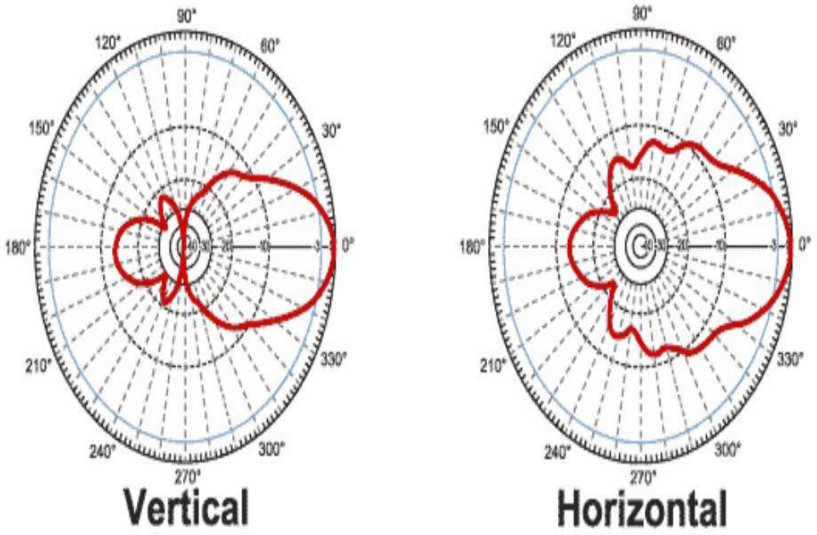


Figure 12. 21-degree antenna AGP.

- 30-degree antenna

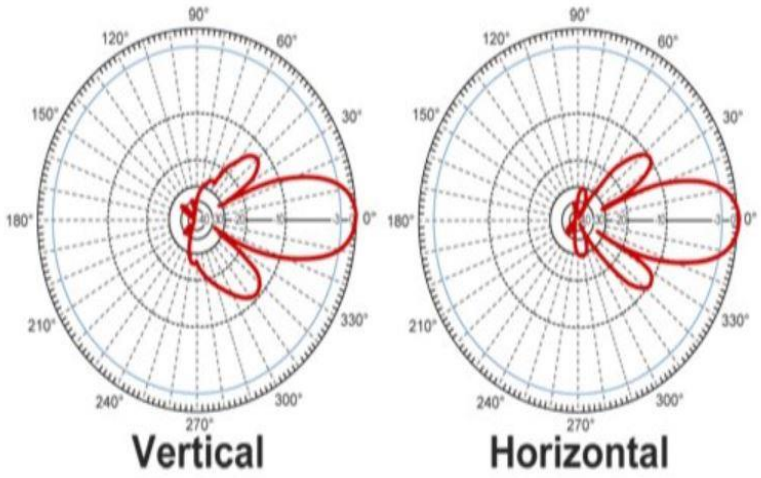


Figure 13. 30-degree antenna AGP.

3.4 Commercial BTA Sniffer Integration

Validating the proposed system's data collection is vital for the success of both the O/D study and the correct assignment of vehicle BTA pairs. Hence, a second method for sniffing BTA must be implemented. The majority of commercial systems applying BTA detection in the transportation field are used for calculating TT, and they commonly utilize Bluetooth sniffing. Iteris Vantage Velocity was chosen for comparison due to the manufacturer's long history in the industry for Bluetooth TT estimations and smart cities solutions. The Iteris Vantage Velocity field unit is placed inside roadside traffic cabinets for sniffing both Wi-Fi and Bluetooth addresses within the antenna's field of view. Each field unit is a Linux box equipped with several peripherals, including USB, Ethernet, and a serial port. Received BTAs are transmitted to host software residing inside the Linux box. Notably, they can also be sent to a server by way of UDP (User Datagram Protocol) messages in prespecified intervals.

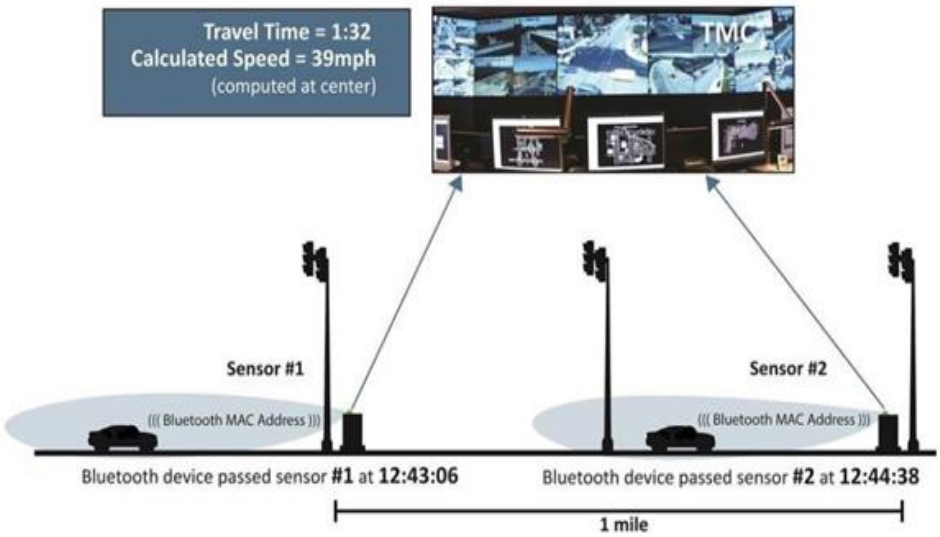


Figure 14.Vantage velocity traffic monitoring concept.

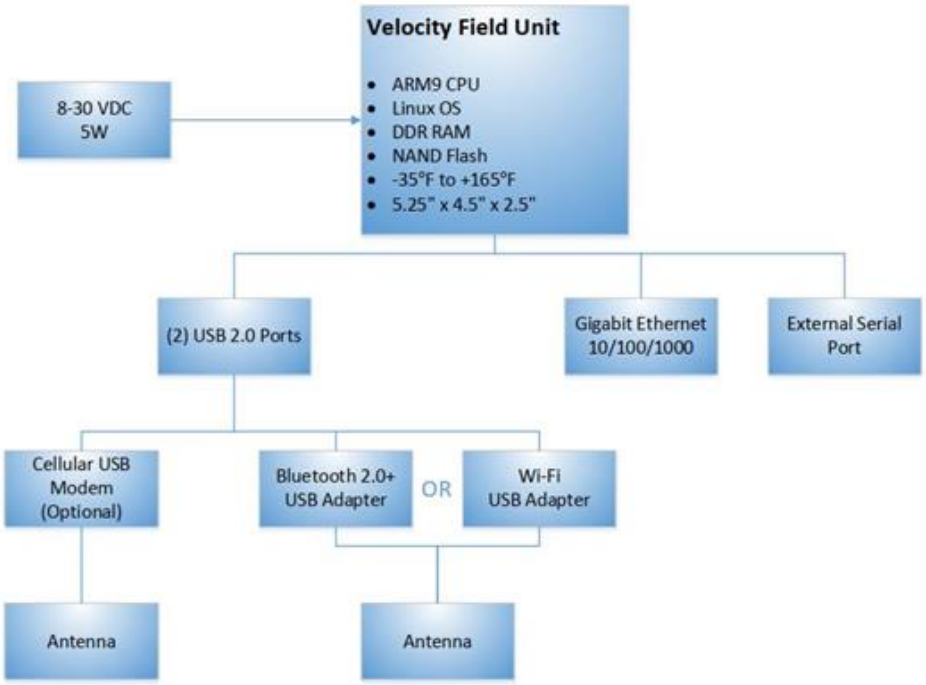


Figure 15. Velocity field unit overview.

Chapter 4 – Testing and data collection

4.1 On-campus Roadside Test

Several on-campus tests were conducted to determine data collection accuracy and to address issues prior to highway deployment. Conducted tests occurred on several days, each focusing on a specific issue.

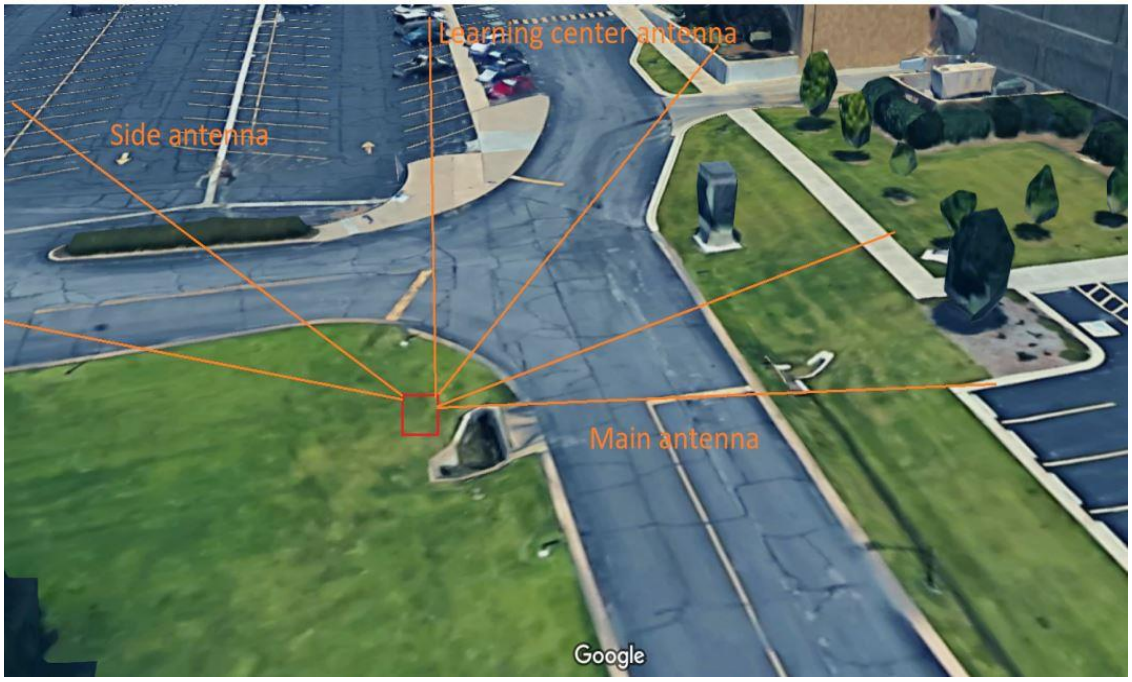


Figure 16. Initial setup with antenna POV.

Two setups were designed for comparing antenna data. The first featured a combination of two 21-degree antennas pointing to the side and to the main road and a third 30-degree antenna pointing to the learning center. The second setup featured two 12-degree antennas pointed in similar fashion and a third 30-degree antenna, namely the one aim at the learning center antenna on campus. The test commenced at 4 pm and ended at 5 pm. Figure 16 shows the on-campus site location.

4.1.1 Initial Setup

The initial test commenced during morning peak hours, starting at 7:45 am. Data was collected for one hour and used three Ubertooth devices, each attached to a REECE and an antenna.

- Detected BTAs

Table 3. Number of Captured BTAs Per Antenna During Morning Test

Antenna	Collected BTAs
Main	723 (52.7%)
Side	249 (18.1%)
Learning	402 (29.2%)
All	1374

Table 3 suggests that most vehicles passing through the side and learning antenna DZs also passed through the main antenna DZ. In fact, the majority of the 723 vehicles traveled through the learning antenna DZ. On campus, Learning/Main road is typically busier than Side/Main road, which is reflected in Table 3 that shows that the Learning/Main O/D pair had more detected BTAs during morning than the Main/Side pair.

- Matched BTAs

The number of matched BTAs detected by antenna pairs is described below.

- Main antenna detected 79.1% (197) and 70.6% (284) of BTA captured by Side and Learning antennas, respectively.
- Side antenna detected 23.3% (169) and 45.3% (182) of BTA captured by Main and Learning antennas, respectively.

- Learning antenna detected 37.3% (270) and 78.3% (195) of BTAs captured by Main and Side antennas, respectively.

Given the numbers characterizing traffic flow throughout the testing period, the following statements can be made.

1. Most vehicles entered the campus from the Main road, and then a larger portion of vehicles

   traveled on the Learning center road rather than the Side road.

2. Traffic flow between the Side/Learning pair was larger than the Side/Main pair.

- Unmatched BTAs

Table 4. Number of BTAs captured by only one antenna during the Morning Test

Antenna	Unmatched BTAs
Main	430 (59.47%)
Side	58 (23.30%)
Learning	110 7.36%)

- Repeated BTAs

Given that the test occurred on a campus sideroad, it is likely that several vehicles passed the same DZ several times during the test. Figures 17 through 19 show the frequency of repeated BTA per antenna.

a- Antenna aimed at the main street or main antenna

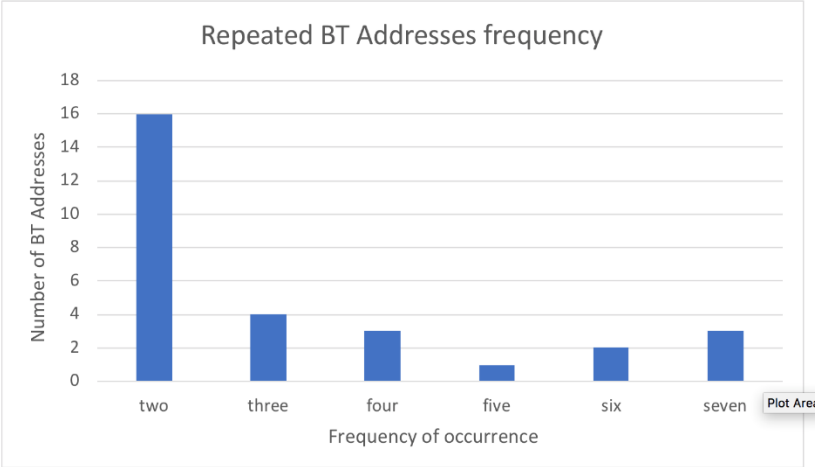


Figure 17. Repeated BTAs frequency during the morning test for the Main antenna.

b- Antenna aimed at the learning center or learning antenna

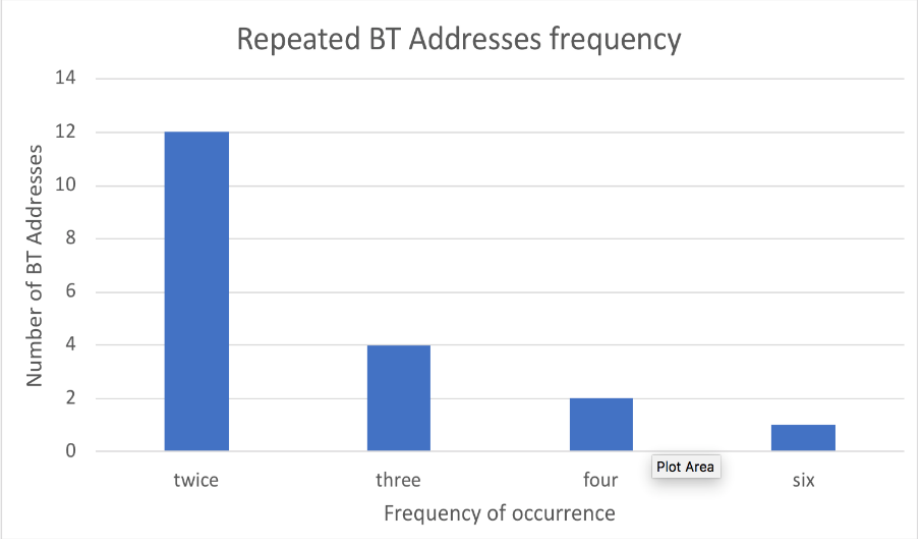


Figure 18. Repeated BTAs frequency during the morning test for the Learning antenna.

c- Antenna aim at the side walk or side antenna

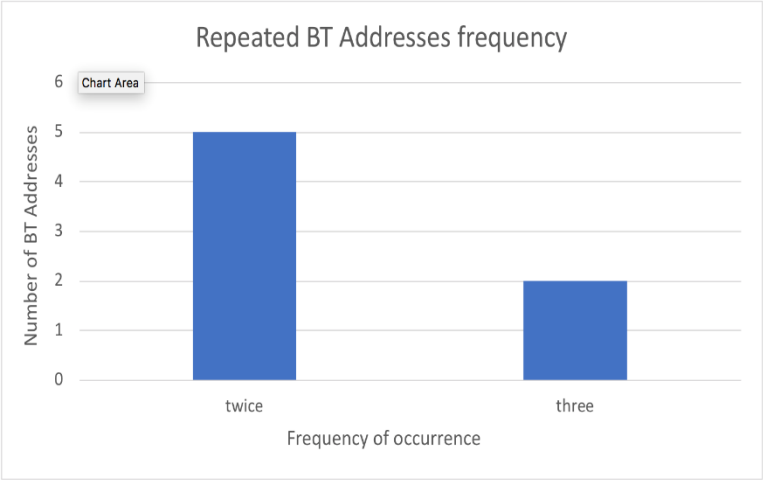


Figure 19. Repeated BTAs frequency during the morning test for the Side antenna.

4.1.2 Second Setup

The second test was conducted for one hour during an evening peak period, beginning at 4 pm. Data was collected using three Ubertooth devices—each attached to a REECE and an antenna.

- Detected BTAs

Table 5. Number of Captured BTAs Detected by System Antenna

Antenna	Collected BTAs
Main	443 (37%)
Side	303 (25.3%)
Learning	449 (37.7%)
All	1195

Table 5 describes various aspects of peak evening hour traffic flow. The majority of vehicles passed through the Main and Learning DZs. More specific insights about traffic directionality can be extracted from the percentage of matched BTA.

- Matched BTA

Number of matched BTAs detected by antenna pair is shown below.

- Main antenna detected 74.2% (225) and 65% (292) BTAs captured by Side and Learning antennas, respectively.
- Side antenna detected 45.8% (203) and 43.2% (194) BTAs captured by Main and Learning antennas, respectively.
- Learning antenna detected 67.1% (297) and 72.6% (220) BTAs captured by Main and Side antennas, respectively.

Given that the numbers characterize traffic flow during the test period, the following statements can be extracted.

1- Percent of vehicles taking Main/Learning and Main/Side pairs is relatively close, meaning that during the evening period, drivers had no dominant tendency to select a specific route during the testing day. However, the Main/ Learning antenna pair was dominant over Main/Side antenna pair.

Unlike the morning hour test, traffic flow during peak evening hours was the same for Side/Learning and Side/Main antenna pairs.

- Unmatched BTAs

Table 6. Number of unmatched BTAs captured by only one antenna

Antenna	Unmatched BTAs
Main	121 (27.31%)
Side	79 (26%)
Learning	131 (29.17%)

- Repeated BTAs

a- Main antenna

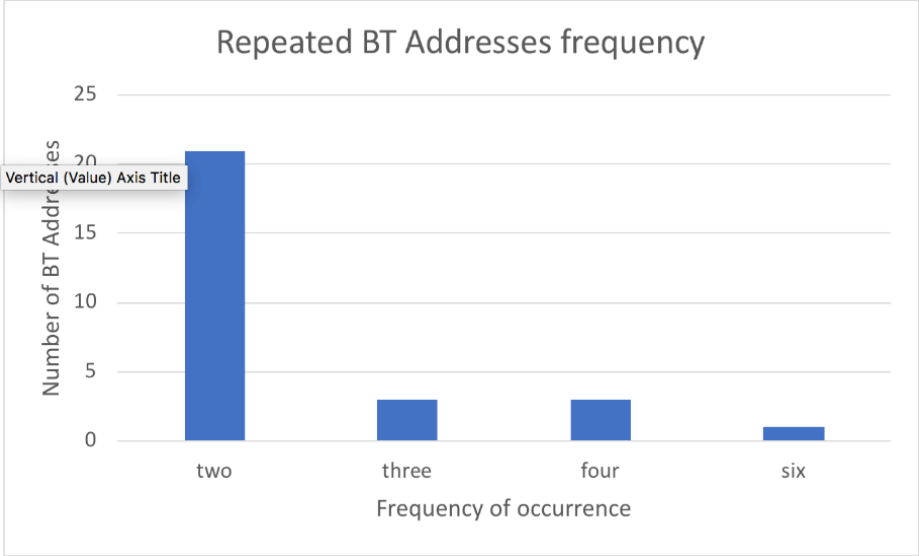


Figure 20. Repeated BTAs frequency for the main antenna during the peak evening hour test.

b- Side antenna

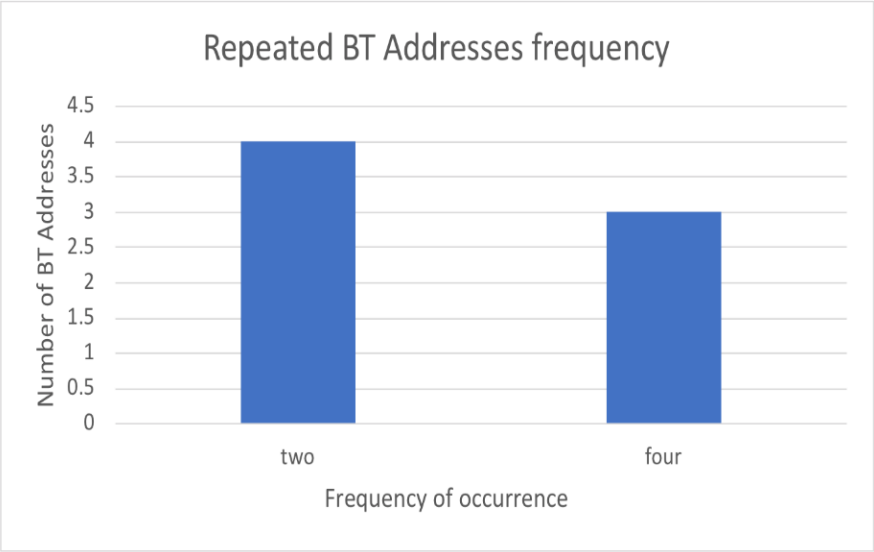


Figure 21. Repeated BTAs frequency for the Side antenna during the peak evening hour test.

c- Learning antenna

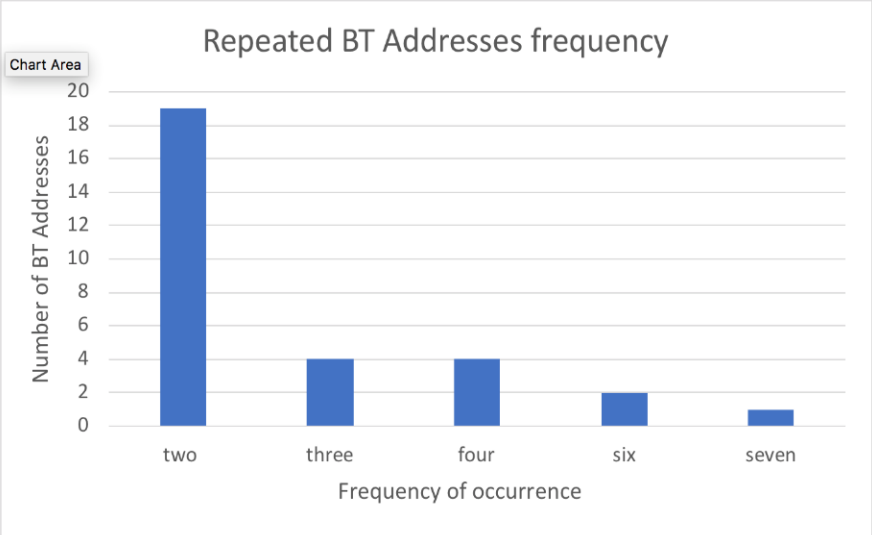


Figure 22. Repeated BTAs for Learning antenna frequency during the peak evening hour test.

- Matching algorithm

a- Data overview

Before processing data, a closer examination of data provided by Ubertooth is required.

Regarding starting code, Ubertooth provides the following data.

1- Lower address part of the BTA – LAP

2- Detection time represented as UNIX time stamp.

3- BTA channel

4- Signal level

5- Noise level

6- SNR

In this test, only the following is necessary.

1- Lower Address Part (LAP)

2- Detection time

Detection code in Ubertooth firmware was modified to detect only LAP and detection time.

Also, Ubertooth was prevented from writing its own BTA to the database by adding the following code:

```
if (btbb_packet_get_lap(pkt) != 0x9e8b33)   // 9e8b33 is the Ubertooth BTA
{
printf("systime=%u LAP=%06x \n", (int)systime, btbb_packet_get_lap(pkt))
fflush(stdout); // we need to flush the standard output after every write procedure in order
for the JavaScript code to capture the data.
}
```

An example of collected data is shown below.

Table 7. Example of collected data

Time	MAC
946685217	1b058b
946685217	1b058b
946685219	e65b1e

b- Data processing

After collecting data from Ubertooth, some processing was required. Python was used for data processing. Each data set was represented as a Pandas Data frame, which is a 2-dimensional, labeled data structure with columns for potentially different types. Pandas Data Frames are the most popular structures for processing data sets, as many related functions facilitate data processing. Data processing codes are presented in Appendix A.

- BTA Randomization

The release of BT 4.0 core specification brings with it BT Smart privacy (i.e., LE privacy). This feature enables the Master device in a communication link to advertise a random BTA to change at timing intervals determined by the manufacturer; hence, the O/D study will be affected due to the inability of capturing the same BTA. BT address randomization was studied for both iOS and Android systems. Several tests were conducted to determine properties of randomization. Results indicated that BT address is randomized only in BLE connections. Thus, given that a mobile phone using iOS is connected to a watch or any other BLE device, captured addresses will not be real addresses. Instead, they will be randomized and are not useful. However, when using BT Classic (i.e., devices that transfer media via Bluetooth, like headphones, car monitors, and BT adapters inside

vehicles), the real address will be detected and written to the database. Notably, Android devices can be discovered without a connection.

4.2 On-campus Roadside Test using RF Multiplexer

A proposed research task suggests investigating an added Radio Frequency Multiplexer (RF MUX) into the design, thus enabling the system to utilize one REECE with one Ubertooth for multiple antennas and reducing system cost. A two-way RF signal combiner operating on a dual 2.4 GHz and 5 GHz band is shown in Figure 23.



Figure 23. Two-way, 2.4 GHz radio frequency MUX.

A test was conducted using the same specifications as previously described, only with the addition of the MUX. Figure 24 provides a depiction of the site location.

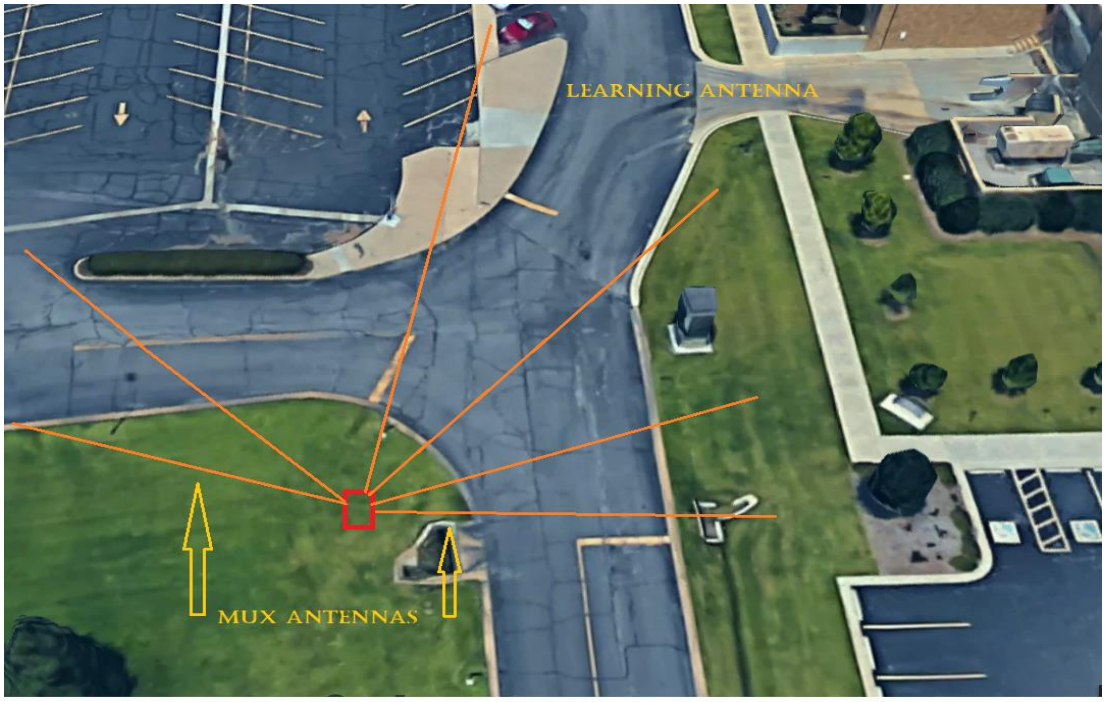


Figure 24. Second Test location with antennas DZs and MUX.

Procedures are listed below.

1- The Third setup utilized a 30-degree beam-width angle antenna (i.e., Learning antenna) pointed toward the Learning center. Two 12-degree beam-width angle antennas were pointed toward the Side road and toward the Main road (i.e., MUX antenna). Both were connected to an RF MUX. Test duration was 90 minutes during the evening peak hours.

2- The Fourth setup utilized a 30-degree beam-width angle antenna (i.e., Learning antenna) pointed toward the Learning center. Two 21-degree beam-width angle antennas were pointed toward the Side road and toward the Main road (i.e., MUX antenna). Both were connected to an RF MUX. Test duration was 90 minutes during the evening peak hours.

4.2.1 Third Setup

- Detected BTAs

Table 8. Number of Captured BTAs per Antenna – MUX used- (Third setup)

Antenna	Matched BTAs
MUX antenna	2650 (55%)
Learning	2169 (45%)
All	4819

- Matched BTAs

MUX antenna detected 80.26% (1741 BTA) of BTA captured by Learning antenna.

- Unmatched BTA

Table 9. Number of Unmatched BTAs per Antenna – MUX used- (Third setup)

Antenna	Matched BTAs
MUX antenna	3855 (44.4%)
Learning	4819 (55.6%)
All	8674

4.2.2 Fourth Setup

- Detected BTA

Table 10. Number of Captured BTAs per Antenna – MUX used- (Fourth setup)

Antenna	
MUX	598 (15.5%)
Learning	289 (6%)

- Matched BTA

MUX antenna detected 55.18% (2650) of BTAs captured by Learning antenna.

- Unmatched BTA

The table below shows the number of BTAs captured by one antenna, but not the other.

Table 11. Number of Unmatched BTAs per Antenna – MUX used- (Fourth setup)

Antenna	Unmatched BTAs
MUX	866(32.7%)
Learning	427(19.7%)

Since RF MUX was utilized, there was no evidence of directionality, primarily because when BTA is captured by the MUX antenna, it will be immediately written to the database without any indication of source. Hence, the issue cannot be resolved and will undermine the entire purpose of the O/D study. In response, RF MUX use was discontinued.

4.2.3 Data processing

The code used in this test was similar to that used in a previous test, with the addition of a function to remove false positive BTAs. As stated in the Ubertooth manual, the device

might detect some LAPs that are not originally BTA. These should be considered noisy data. The code is discussed in Appendix A.

4.3 On-campus Sideroad Test (Two Simultaneous Setups)

The previously described tests were conducted to provide information about data collected from various antenna types and to compare captured data quality. However, to achieve optimal results, both setups should be operating in the same environment under the same conditions. Figure 25 shows a depiction of the simultaneous test setup.

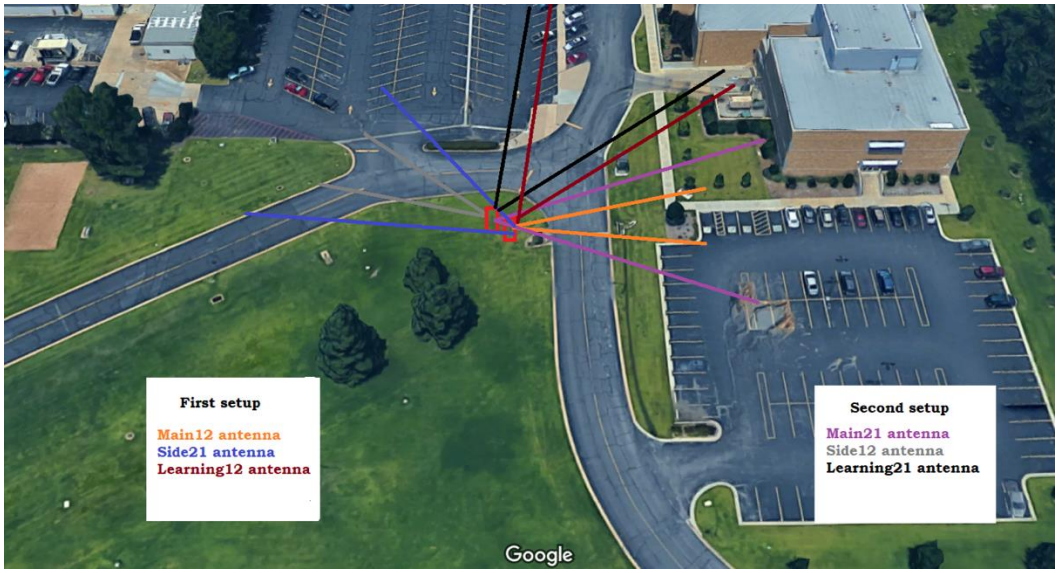


Figure 25. Test setup three for simultaneous testing of two systems with DZs antenna.

Configuration for test setups.

1- The Fifth setup utilized a 30-degree antenna (Learning1) pointed toward the Learning center; a 21-degree antenna pointed toward the Side road (Side21); and a 12-degree beam-width angle pointed toward the Main road (Main12).

2- The sixth setup utilized a 30-degree antenna (Learning2) pointed toward the Learning center; a 12-degree beam-width angle pointed toward the Side road (Side12); and a 21-degree antenna pointed toward the Main road (Main21).

The nine-hour test began at 8:42 am and ended at 5:45 pm. The system recorded all BTAs, without exception.



Figure 26. Setup for test three.

4.3.1 Fifth setup

Table 12. Number of Captured BTAs per Antenna for the Fifth Setup

Antenna	Detected BTAs
Side21	79,280 (35%)
Learning1	84,477 (38%)
Main12	60,169(27%)
All	223,926

4.3.2 Sixth setup

Table 13. Number of Captured BTAs per Antenna for the Sixth Setup

Antenna	Detected BTAs
Side12	73,157 (40%)
Learning2	72,298 (39.6%)
Main21	37,054 (20.4%) *
All	182,509

*Main21: Started at noon due to a technical problem.

- Matched BTA
  - Side21 detected 95% (70,079) of BTA captured by Side12
  - Learning2 detected 98% (70,963) of BTA captured by Learning1
  - Main21 detected 93% (34,579) of BTA captured by Main12
- Unmatched BTA

Table 14. Number of Unmatched BTAs Between the Fifth and Sixth Setups

Antenna	**Unmatched BTAs***
Side12/Side21	3,079/ 5,147
Learning1/Learning2	3,424/ 1,336
Main12/Main21	5,699/ 2,476**

* Detected by one antenna only

** Starting afternoon

- Unique BTA

Table 15. Number of Unique BTA Collected by Both Setups

Antenna	Unique BTAs
Side12	1,285
Side21	1,729
Learning1	1,741
Learning2	1,393
Main12	2,276
Main21	920

4.3.3 **Data division**

Further research provided additional information about the data, according to five groupings listed below.

1. Morning: 8:42 to 9:47 am
2. Idle1: 9:48 am to 12:03 pm
3. Noon: 12:04 to 1:04 pm
4. Idle2: 1:05 to 4:29 pm
5. Evening: 4:30 to 5:45 pm

4.3.4 **Number of Records**

Number of BTA collected by each antenna for specified times of day are shown below.

Table 16. Collected BTA by each part of the day (Side12)

Antenna	side 12_MO	side 12_I1	side 12_NO	side 12_I2	side 12_EV	Total
Detected BTAs	3,456 (4.72%)	15,007 (20.51%)	9,466 (12.94%)	23,722 (32.42%)	21,507 (29.41%)	73,158

Table 17. Collected BTA by each part of the day (Side21)

Antenna	side 21_MO	side 21_I1	side 21_NO	side 21_I2	side 21_EV	Total
Detected BTAs	4,815 (6.07%)	17,070 (21.53%)	7,569 (9.54%)	27,590 (34.8%)	22,237 (28.06%)	79,281

Table 18. Collected BTA by each part of the day (Learning1)

Antenna	Learning 1_MO	Learning 1_I1	Learning 1_NO	Learning 1_I2	Learning 1_EV	Total
Detected BTAs	5,923 (5.74%)	18,882 (20.51%)	10,852 (12.94%)	27,585 (32.42%)	21,235 (29.41%)	84,477

Table 19. Collected BTA by each part of the day (Learning2)

Antenna	Learning 2_MO	Learning 2_I1	Learning 2_NO	Learning 2_I2	Learning 2_EV	Total
Detected BTAs	4,815 (6.07%)	13,597 (21.53%)	7,569 (9.54%)	27,590 (34.8%)	22,237 (28.06%)	72,298

Table 20. Collected BTA by each part of the day (Main12)

Antenna	Main 12_MO	Main 12_I1	Main 12_NO	Main 12_I2	Main 12_EV	Total
Detected BTAs	3,456 (4.72%)	15,007 (24.94%)	9,466 (15.73%)	23,722 (39.42%)	21,507 (35.74%)	72,298

Table 21. Collected BTAs by each part of the day (Main21)

Antenna	Main 21_NO	Main 21_I2	Main 21_EV	Total
Detected BTAs	7,569 (9.54%)	27,590 (34.8%)	22,237 (28.06%)	37,054

4.3.5 Number of matches

a- Side antennas

Table 22. Comparison Between Side Antennas

Antenna	Matched BTAs	Not matched*
Side12_MO/ Side21_MO	3,280/ 4,297	176/ 518 (5.09% - 10.75%)
Side12_I1/ Side21_I1	14,729/ 16,565	278/ 505 (1.85% - 2.95%)
Side12_NO/ Side21_NO	8,706/ 7,068	760/ 501 (8.02% - 6.61%)
Side12_I2/ Side21_I2	22,641/ 26,431	1081/ 1159 (4.55% - 4.2%)
Side12_EV/ Side21_EV	19,595/ 18,599	1912/ 3638 (8.89% - 16.36%)

* Only detected by one antenna

Legends: MO: morning period; I1: first idle period; NO: noon period; I2: second Idle period; EV: evening period

b- Learning antenna

Table 23. Comparison Between Learning Antennas

Antenna	Matched BTAs	Not matched*
Learning_1_MO/ Learning_2_MO	5,188/ 4613	735/ 113 (12.4% - 2.59%)
Learning_1_I1/ Learning_2_I1	18,138/ 13162	744/ 435 (3.94% - 3.19%)
Learning_1_NO/ Learning_2_NO	10,472/ 7448	380 / 121 (3.5%- 1.39%)
Learning_1_I2/ Learning_2_I2	21,200/ 26668	658 / 922 (2.38% - 3.58%)
Learning_1_EV/ Learning_2_EV	19,444/ 21974	1791 / 263 (8.43% - 1.31%)

* Only detected by one antenna

c- Main antennas

Table 24. Comparison Between Main Antenna

Antenna	Matched BTAs	Not matched*
Main_12_NO / Main _21_NO	8,397/ 7156	1069 / 413 (21.25% - 11.66%)
Main _12_I2/ Main _21_I2	20,531/ 26676	3191 / 914 (13.48% - 4.68%)
Main _12_EV / Main _21_EV	19,138/ 20453	2369 / 1784 (14.41% - 12.73%)

* Only detected by one antenna

- Unique BTA

Table 25. Unique BTAs collected by Side12 per period

Antenna	Unique BTAs
Side12_MO	130
Side12_I1	364
Side12_NO	191
Side12_I2	547
Side12_EV	348

Table 26. Unique BTAs collected by Side21 per period

Antenna	Unique BTAs
Side21_MO	172
Side21_I1	426
Side21_NO	239
Side21_I2	757
Side21_EV	445

Table 27. Unique BTAs collected by Learning1 per period

Antenna	Unique BTAs
Learning1_MO	192
Learning 1_I1	463
Learning 1_NO	256
Learning 1_I2	715
Learning 1_EV	457

Table 28. Unique BTAs collected by Learning2 per period

Antenna	Unique BTAs
Learning 2_MO	158
Learning 2_I1	386
Learning 2_NO	217
Learning 2_I2	574
Learning 2_EV	360

Table 30. Unique BTAs collected by Main12 per period

Antenna	Unique BTAs
Main12_MO	277
Main 12_I1	753
Main 12_NO	362
Main 12_I2	1310
Main 12_EV	713

Table 29. Unique BTAs collected by Main21 per period

Antenna	Unique BTAs
Main 21_NO	158
Main21_I2	527
Main 21_EV	304

4.4 South Yale Setup

The use of power attenuators was investigated for the purpose of reducing employed antenna sensitivity. Results showed that noise was reduced by receiving antenna and reducing the effect of side lobes of antenna radiation pattern, in addition to a decrease in detected BTA number. Used attenuators were fixed at 15dB and 10dB. Attenuator was attached to a 12-degree antenna with 20 dB power gain.

4.4.1 Seventh Setup

A 15dB attenuator was attached to only one 12-degree antenna, while the other was operating normally. Testing occurred between 7:15 and 8:15 p.m. on David Boren Blvd. at Yale Ave. in Tulsa, OK. Figure 34 shows both setups.

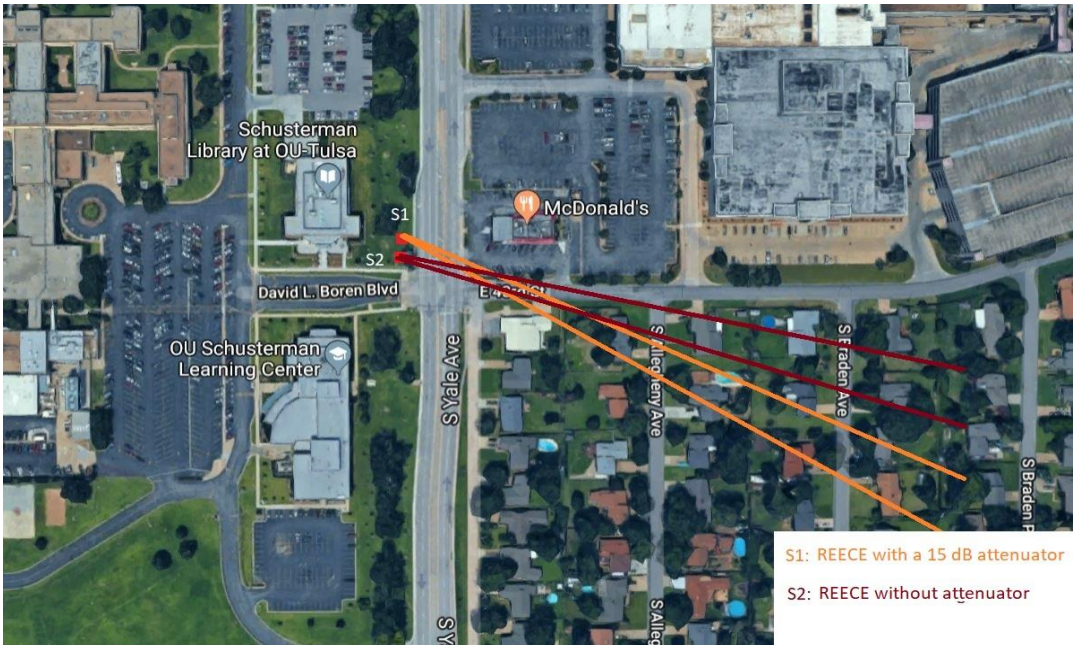


Figure 27. Fourth Test setup locations with antennas DZ

- Detected BTA

Table 31. Number of BTA Collected by 15db REECE and Standard REECE

REECE with 15 dB attenuator	REECE without attenuator
1986	29992

- Unique BTA

Table 32. Number of Unique BTA Collected by 15db REECE and Standard REECE

REECE with 15 dB attenuator	REECE without attenuator
204	619

- Matched BTA

The REECE with power attenuator detected 32% of unique codes detected by the REECE without power attenuator. Data collected by REECE equipped with a 15dB attenuator was expected to decrease, as received power was reduced.

4.4.2 Eighth setup

In the second test setup, a 10dB attenuator was attached to one of the 12-degree antennas. Data was collected between 6:53 and 7:53 am at the same location using a setup identical to the first test setup.

- Detected BTA

Table 33. Number of BTA Collected by the 10dB REECE and Standard REECE

REECE with 10 dB attenuator	REECE without attenuator
8401	18171

- Unique BTA

Table 34. Number of Unique BTA Collected by the 10dB REECE and the Standard REECE

REECE with 10 dB attenuator	REECE without attenuator
617	753

- Matched BTA

The REECE with power attenuator detected 66.4% of the unique codes detected by the REECE without a power attenuator. Unlike results from the first setup, the percentage of matched data between REECE without power attenuator and REECE with a 10dB attenuator was larger than REECE with a 15dB attenuator.

4.4.3 Ninth setup

In the third test setup, a 10dB attenuator was attached to one of the 12-degree antennas and a 15dB attenuator to the second 12-degree antenna. Testing occurred between 10:15 and 10:45 am at the same location using the same orientation as previous tests.

- Detected BTA

Table 35. Number of BTA Collected by the 10dB REECE and 15dB REECE

REECE with 10 dB attenuator	REECE with 15 dB attenuator
5047	2170

- Unique BTA

Table 36. Number of Unique BTA Collected by the 10dB REECE and the 15dB REECE

REECE with 10 dB attenuator	REECE with 15 dB attenuator
296	225

- Matched BTA

Matched BTA between 10 dB and 15dB REECE units was 76.5%.

4.5 Integration of commercial system

4.5.1 Tenth Setup

The first task of the initial test was comparing BTAs detected by the system under review to those detected by Iteris Velocity. Two Iteris devices with omnidirectional antennas were utilized with two REECE equipped with 12-degree antennas. Testing was conducted between 7 and 8 pm at the same location during which time 700 vehicles traveled through the DZs.



Figure 28. A depiction of the Fifth test setups.

- Detected BTA

Table 37. Comparison between Iteris- and REECE-collected Data with Omnidirectional Antenna.

	Iteris	REECE
Right Side	60	2413*
Left Side	52	12307*

- Unique BTA

Table 38. Number of BTA Collected by Both Systems using Omnidirectional Antenna

	Iteris	REECE
Right Side	52	285
Left Side	50	395

- Matched BTA

Right side Iteris unit detected 6% (17) BTA detected by the corresponding REECE. Left side Iteris unit detected 10% (39) BTA detected by the corresponding REECE.

4.5.2 Eleventh Setup

This test utilized a 12-degree antenna leveraging both systems for comparing BTAs detected by both systems. The test was conducted between 7:40 and 8:40 pm at the same location with 600 vehicles traveling through DZs. Figures 36 and 37 show both setups.

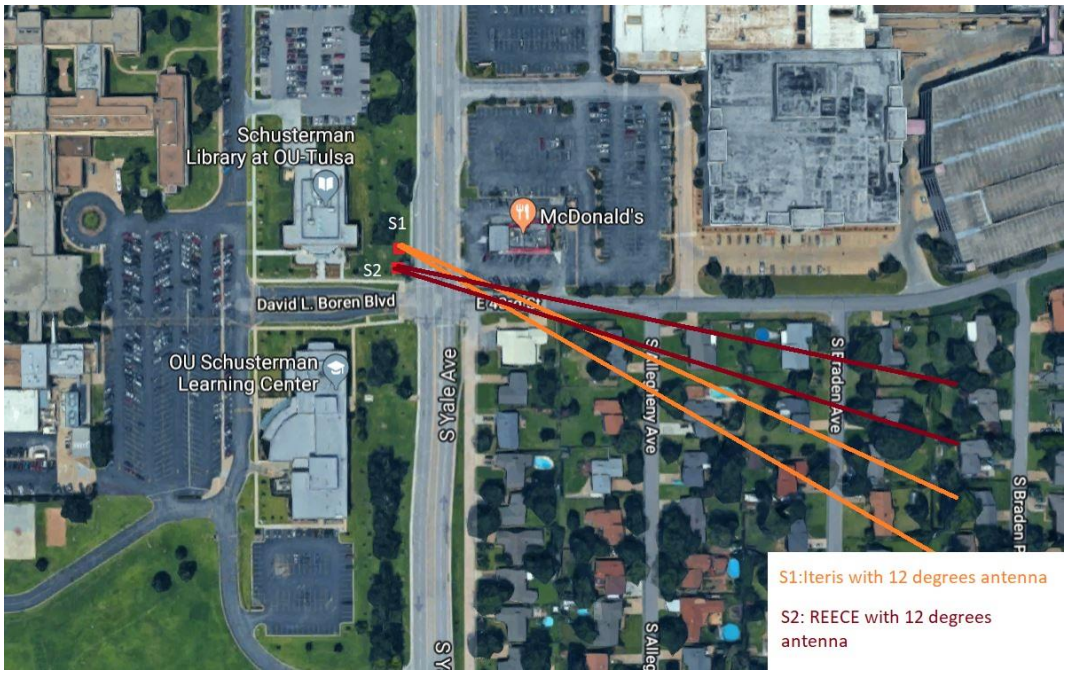


Figure 29. A depiction of the two setups with Directional antennas of the Sixth test



Figure 30. Depiction of the two systems of Sixth test.

- Detected BTA

Table 39. Comparison Between Iteris- and REECE-collected Data utilizing 12-degree Antenna

	Iteris	REECE
Number of BTAs	218	14257*

*The current system archived all detected BTA without removing duplicates or utilizing queues.

- Unique BTA

Table 40. Number of Unique BTA Collected by Both Systems utilizing 12-degree Antenna

	Iteris	REECE
Number of unique BTAs	101	529

- Matched BTA

Iteris unit detected 15% (79) of BTAs detected by the proposed system. The percentage of detected BTAs increased, primarily because utilized antenna had greater power gain than the Iteris omnidirectional antenna.

4.5.3 Twelfth Setup

In addition to data collection for previous tests, a car equipped with a BT device connected to a cell phone was driven through the DZs to determine the number of times each system detected pre-known BTA. This test was completed twice for each of the following setups.

1- Iteris Vantage and REECE with omnidirectional antenna

2- Iteris Vantage and REECE with 12-degree antenna

Table 41. Comparison Between Detection Times of Pre-known BTA for the Proposed System and Iteris Velocity

	Iteris	REECE
Omnidirectional antenna	0/2	1/2
12-degree directional antenna	1/2	2/2

Results reported above matched researcher expectations. Iteris Vantage Velocity detects full BTAs, which requires more time than the developed system's ability to detect only the

lower half of the BTA-LAP. Responding to a need for additional system inspection, both systems were deployed in a highway environment for longer periods of time.

4.5.4 Data processing

Due to various Iteris vantage velocity data structures, data processing procedures should be applied to data captured before comparison with data of the developed system. Following is an example of the data captured by Iteris.

07/08/2018 05:50:23 PM, Test_Unit, 00:1E:B2:08:E1:83

The data reporting commences with date and time of detection, followed by unit and full address. Notably, data and time formula vary from the developed timestamp formula. Hence, the first step is changing the data and time to a UNIX timestamp. After that, each data point should be parsed for extracting LAP of the full BTA. The parsing code is discussed in Appendix A.

4.6 Highway Deployment

4.6.1 First field-test:

After many tests were conducted on the OU-Tulsa campus and a number of software and hardware issues were resolved, the system was deployed on Highway 169 in Tulsa, OK. Two sites measuring 5.5 miles in length were deployed at AVC10 and AVC68, as shown in Figure (31).

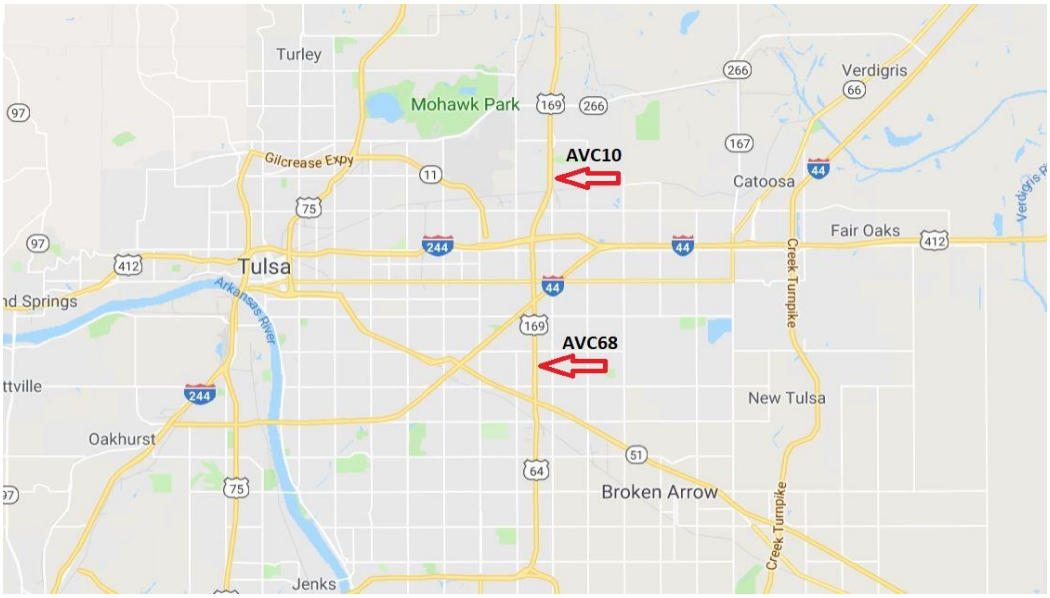


Figure 31. Location of two setups of the first setup.

Both the proposed system and an Iteris unit were deployed at each site.

1- AVC10 was equipped with a 12-degree antenna and an Iteris unit with an omnidirectional antenna.

2- AVC68 was equipped with a 21-degree antenna and an Iteris unit with an omnidirectional antenna.

The following results were reported after capturing BTAs for 24 hours, beginning at 3 pm.

- Detected BTA

Table 42. Number of Detected BTAs by Both Systems

Unit	Detected MACs
AVC10_REECE	27,892
AVC10_Iteris	11,392
AVC68_REECE	35,941
AVC68_Iteris	8,946

- Matched BTA

Number of matched BTA detected by a pair of antennas is shown below.

- AVC10_REECE antenna detected 24.1% (8,664) of BTA captured by AVC68_REECE.
- AVC10_Iteris antenna detected 1.5% (416) of BTA captured by AVC10_REECE.
- AVC68_REECE antenna detected 33.5 % (9,350) of BTA captured by AVC10_ REECE.
- AVC68_Iteris antenna detected 1.3 % (473) of BTA captured by AVC68_ REECE.

- Distribution

The following figures show the distribution of collected BTAs during the 24-hours test.

- AVC10:

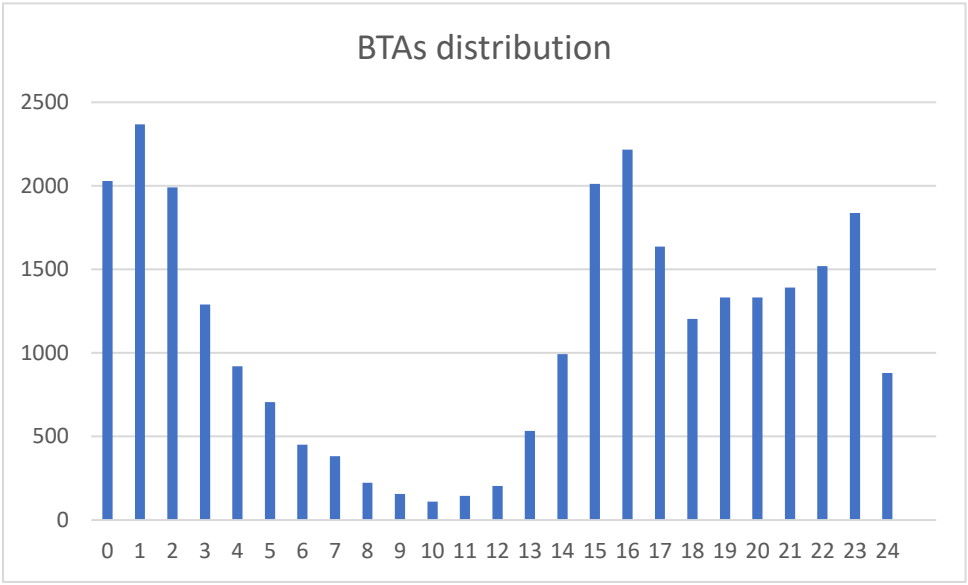


Figure 32. Distribution of BTAs during a 24-hour period at AVC10.

- AVC68:

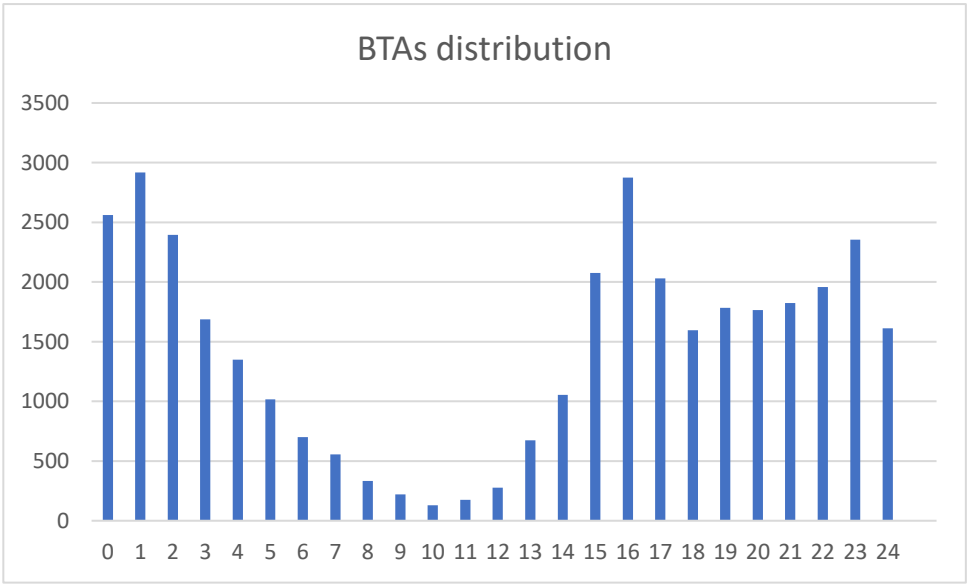


Figure 33. Distribution of BTAs during a 24-hour period at AVC68.

4.6.2 Second field-test:

- Data collection:

After testing the MACs collection system in the previous setup, further deployments were needed to validate the integrity of the two systems (i.e., MAC collection and Vehicle Classification) and to determine the best positioning scheme of the sensors within the DZ. Accordingly, the full system was deployed on I-44 at one of ODOT's data collection stations (AVC19). The test lasted three days, starting April 24 for 3.5 hours each day. Following are deployment layout and location.

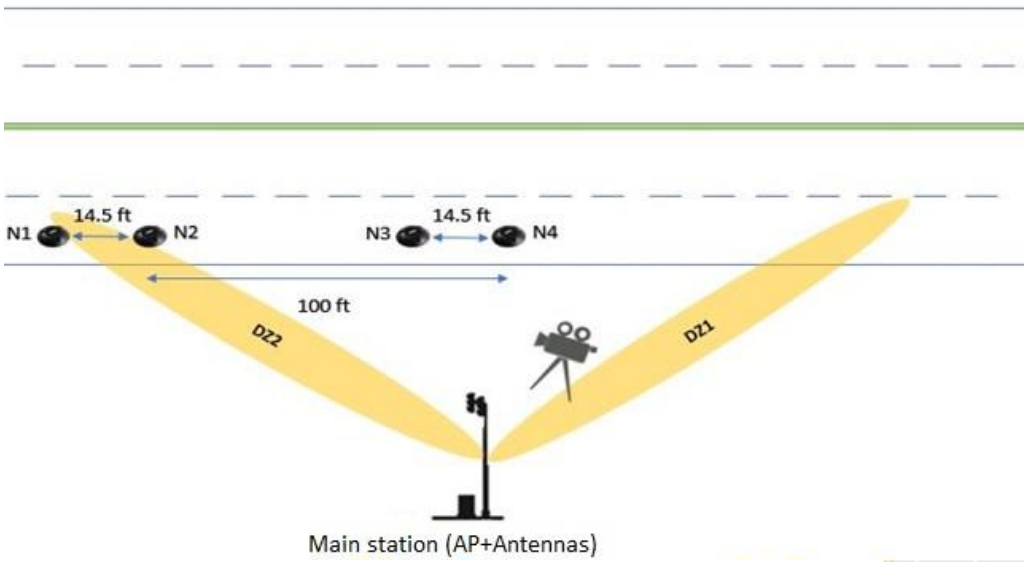


Figure 34. A depiction of the layout of the second deployment.



Figure 35. Location of the Second deployment.

A digital camera was used to record vehicles passing through the DZ during the deployment. The recording was used as a validation measure for classification system accuracy, as well as to understand conditions underlying the system's decision. Vehicle classes driving on the first lane were extracted from the recording and archived in a database. The following chart was used to manually classify vehicles according to class.

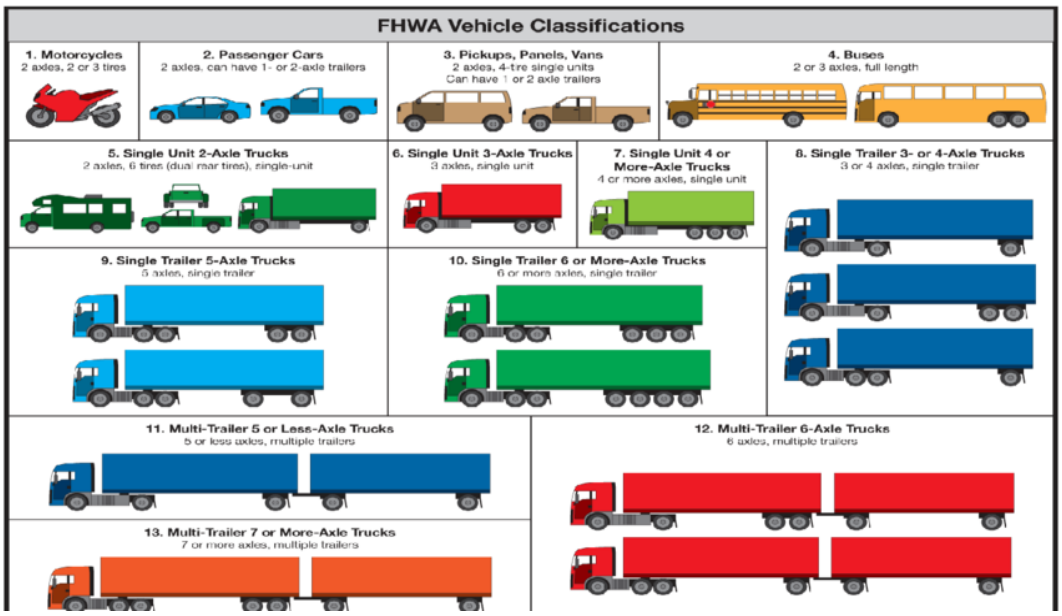


Figure 36. FHWA vehicle classification.

Collected data statistics are illustrated in the following tables.

Table 43. Collected MACs statistics (First day)

04/24/2019		
	Left antenna	Right antenna
Start time	12:29:48 PM	12:31:30 PM
End time	3:28:14 PM	3:29:12 PM
Number of records	8389	46534
Number of unique records	1428	3725

Table 44. Collected MACs statistics (Second day)

04/25/2019		
	Left antenna	Right antenna
Start time	2:17:45 PM	2:14:13 PM
End time	6:33:29 PM	6:33:27 PM
Number of records	130700	131233
Number of unique records	7246	6910

Table 45. Collected MACs statistics (Third day)

04/26/2019		
	Left antenna	Right antenna
Start time	11:26:18 AM	11:30:04 AM
End time	2:37:37 PM	2:37:35 PM
Number of records	65683	48589
Number of unique records	5133	4352

4.6.2.1 Data processing:

In order to analyze the collected data, several data processing algorithm were developed.

1- MACs filtering. Spatial analysis of the collected MAC addresses requires prober detection of the traveling MAC address directionality. The algorithm is designed to facilitate filtering MAC addresses based on their directionality. The algorithm considers data sets of MACs collected by both antennas and filters them, retaining only MACs traveling from one DZ to another.

After applying the algorithm on the datasets, the following results were obtained.

Table 46. Number of matched MACs between the two antennas (First day)

Traffic Direction	Number of records
West bound traffic	5
East bound traffic	6

Table 47. Number of matched MACs between the two antennas (Second day)

Traffic Direction	Number of records
West bound traffic	3422
East bound traffic	3625

Table 48. Number of matched MACs between the two antennas (Third day)

Traffic Direction	Number of records
West bound traffic	2126
East bound traffic	2061

Table 49. Output of the MACs filtering algorithm (Left DZ to Right DZ)

codes	TA	First RSSI	TD	Second RSSI
158857	1556296195	-18.4643	1556296204	-17
570451	1556296200	-16.7778	1556296210	-29
102154	1556296202	-16	1556296207	-6
771171	1556296203	-20	1556296208	-23
107192	1556296208	-16	1556296221	-22
459316	1556296212	-5	1556296217	-25
318711	1556296213	-17	1556296219	-13.6
144962	1556296214	-22	1556296215	-27
124999	1556296217	-18	1556296219	-27.5
489138	1556296219	-24	1556296224	-23.5

Legends:

Codes: Detected MAC address

TA: First occurrence of MAC address in left antenna's database

First RSSI: Average value of RSSI calculated from left antenna database

TD: Last occurrence of MAC address in right antenna's database

Second RSSI: Average value of RSSI calculated from right antenna database

2- Video-sensor matching algorithm. This algorithm is responsible for matching timestamps of the vehicles passing the DZ using the monitored lane with those reported by the sensors.

After checking all timestamps, the algorithm ceases operation by archiving matched timestamps into a database. Following is an example of the resultant dataset.

Table 50. Output of the Video-Sensor timestamps matching algorithm

ID	Vehicle Class	Sensor	TA	TD
2	2	364099	1556296584.92	1556296585.14
3	2	364099	1556296591.67	1556296591.86
4	5	364099	1556296597.27	1556296597.62
5	3	364099	1556296602.67	1556296602.91
6	3	364099	1556296609.38	1556296609.63
7	3	364099	1556296610.48	1556296610.74
8	2	364099	1556296620.67	1556296620.88
10	3	364099	1556296638.66	1556296638.73
11	9	364099	1556296649.69	1556296650.51
12	6	364099	1556296658.11	1556296658.72
13	6	364099	1556296661.22	1556296661.83

Legends:

ID: ID of vehicle within the video database

Vehicle Class: Vehicle class with provided ID

Sensor: ID of the sensor matched with vehicle timestamp

TA: Vehicle time of arrival at sensor

TD: Vehicle time of departure at sensor

Table 51. Results of Video-Sensor timestamps matching algorithm

Number of vehicles(video)	Number of matched Vehicles (Sensor1)	Number of matched Vehicles (Sensor2)
793	287	692

The number of detected vehicles on the sensor side was low due to several issues during deployment.

a- Sensors were not able to establish a stable connection link with the AP due to high attenuation in the deployment environment.

b- Sensor2 and Sensor3 were not able to save timestamps to flash memories installed on the board due to sensor hardware problems.

c- Sensor2 and Sensor4 lost sync with the GPS satellites for a period of time, causing incorrect timestamps. This data was not considered in the algorithm.

After using the algorithm to process each sensor's dataset, the classification algorithm was applied. The video recording was used to validate classification algorithm accuracy—85% for 276 vehicles. Classification accuracy is expected to improve after solving sensor issues.

3- MAC to vehicle assignment: After using timestamps reported by the sensors to classify vehicles passing through DZ, timestamps were also used to match each detected MAC address for vehicles traveling from DZ's left side to right side.

The algorithm was designed to find all MAC addresses captured within a threshold prior to vehicle detection by the first sensor (sensor1).

Following is an example of the output of the matching algorithm.

Table 52. Output of the BT-Vehicle assignment algorithm

MAC address	Group	BT DetectionTime	STA	STD	BT DepartureTime
941544	3	1556296657	1556296658.11	1556296659.99	1556296671
899086	2	1556296662	1556296663.94	1556296665.31	1556296669
96284	2	1556296738	1556296739.37	1556296740.69	1556296745
721944	2	1556296742	1556296744.73	1556296745.96	1556296747
127167	2	1556296764	1556296766.60	1556296767.87	1556296765
744044	2	1556296769	1556296769.58	1556296771.10	1556296774
778122	2	1556296787	1556296787.45	1556296788.85	1556296794
136411	2	1556296807	1556296809.25	1556296810.48	1556296812
869021	2	1556296824	1556296826.50	1556296827.87	1556296825

Legends:

MAC address: Address assigned to the vehicle

Group: Vehicle group assigned to the MAC address

BT Detection Time: First occurrence of MAC address within DZ

STA: Time of arrival to first sensor (Sensor 2)

STD: Time of departure from second sensor (Sensor 4)

BT Departure Time: Last occurrence of MAC address within DZ

The algorithm matched 125/276 vehicles with their corresponding MAC addresses. (276 is the number of vehicles classified by the classification algorithm, as previously discussed).

4.6.2.2 Data analysis:

Collected data was analyzed after applying the processing algorithms. Following is a list of information extracted from the data.

1- Vehicle classes

a. Distribution of classes for vehicles traveling on the first lane passing through the DZs during the full day of deployment on the third day and extracted from the video recording.

Table 53. Number of detected Vehicles and their classes (Third day - Camera)

Number of vehicles	G1	G2	G3	G4
1333	10 (0.7%)	1101 (82.59%)	66(4.9%)	156(11.7%)

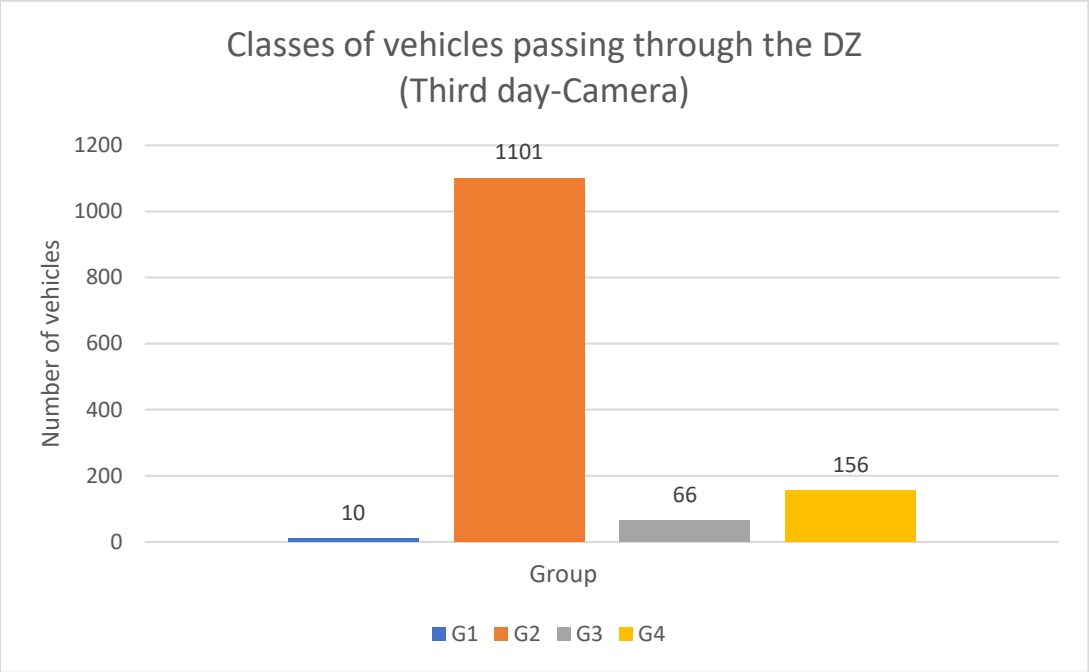


Figure 37. Distribution of the classes passing through the DZs (Third day -Camera).

b. Distribution of classes for vehicles traveling on the first lane passing through the DZs during the time sensors were working, and then extracted from the video recording.

Table 54. Number of detected Vehicles and their classes (Part of the third day - Camera)

Number of vehicles	G1	G2	G3	G4
275	0 (0%)	225 (81.8%)	18(6.5%)	32(11.6%)

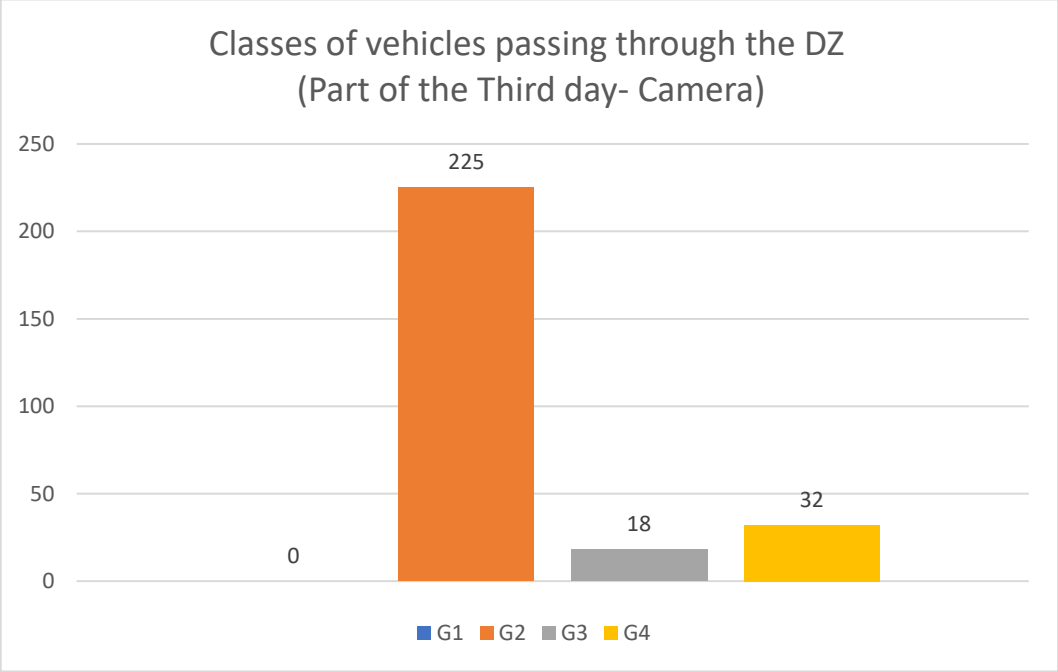


Figure 38. Distribution of the classes passing through the DZs (Part of the Third day -Camera).

c. Distribution of classes for vehicles traveling on the first lane, passing through the DZs during the time sensors were working, and then extracted from the following classification algorithm.

Table 55. Number of detected Vehicles and their classes (Part of the third day - Sensors)

Number of vehicles	G1	G2	G3	G4
275	3 (1.08%)	223 (81.15%)	13(4.7%)	36(13.04%)

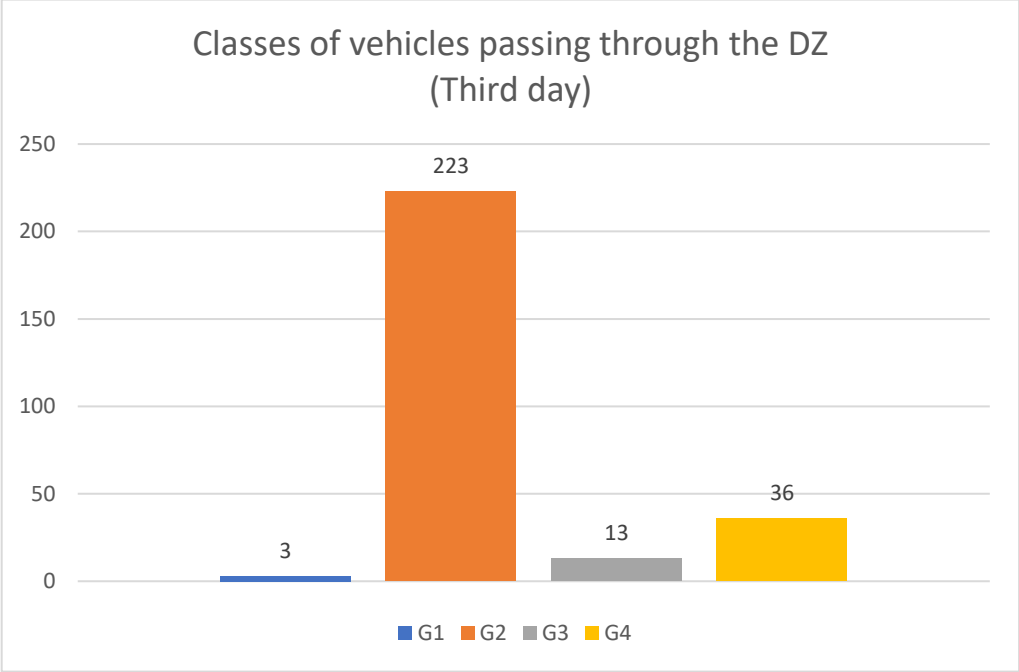


Figure 39. Distribution of the classes passing through the DZs (Third day - Sensors).

Total number of unique MACs was 4353.

d. TT from left DZ to right DZ was 5 seconds, with an SD of 3.63.

e. TT from right DZ to the left DZ was 6.6 seconds, with an SD of 5.1.

f. Average vehicle speed was 58.16 mph with an SD of 18.27. Results were extracted from vehicles that were matched using video recording.

4.6.3 Third field-test

4.6.3.1 Data collection.

In this test, the system was deployed at one of ODOT's AVC sites (AVC516) for 4.5 hours on June 10. The test aimed to examine the newly developed algorithms and the solutions that were suggested to maintain a better communication link between AP and sensors. Following are deployment location and layout.

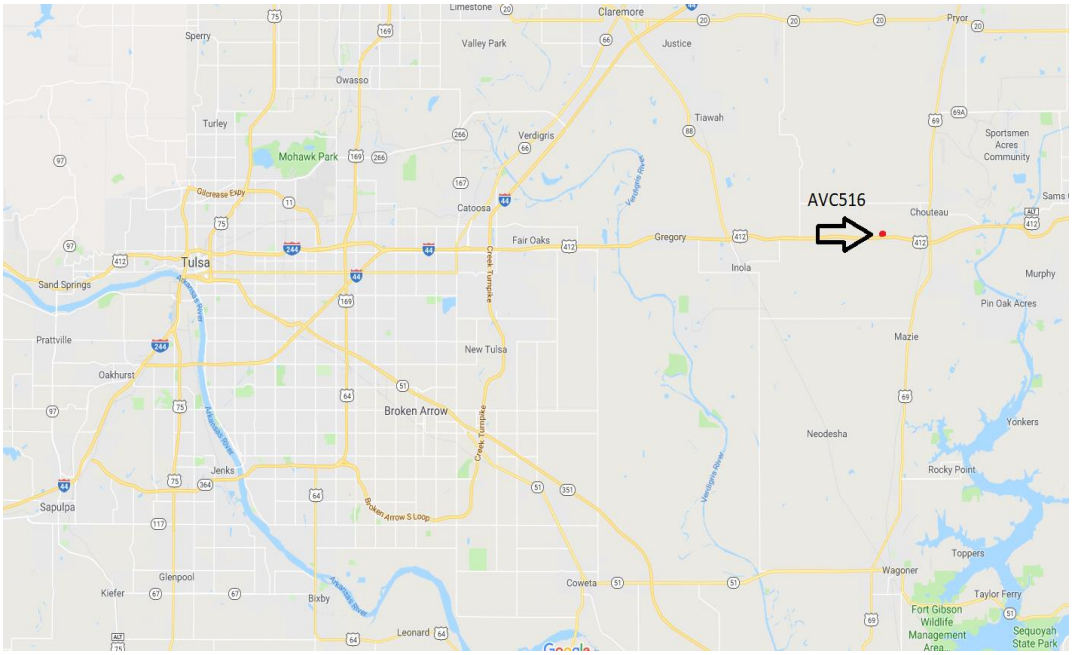


Figure 40. Location of the third deployment.

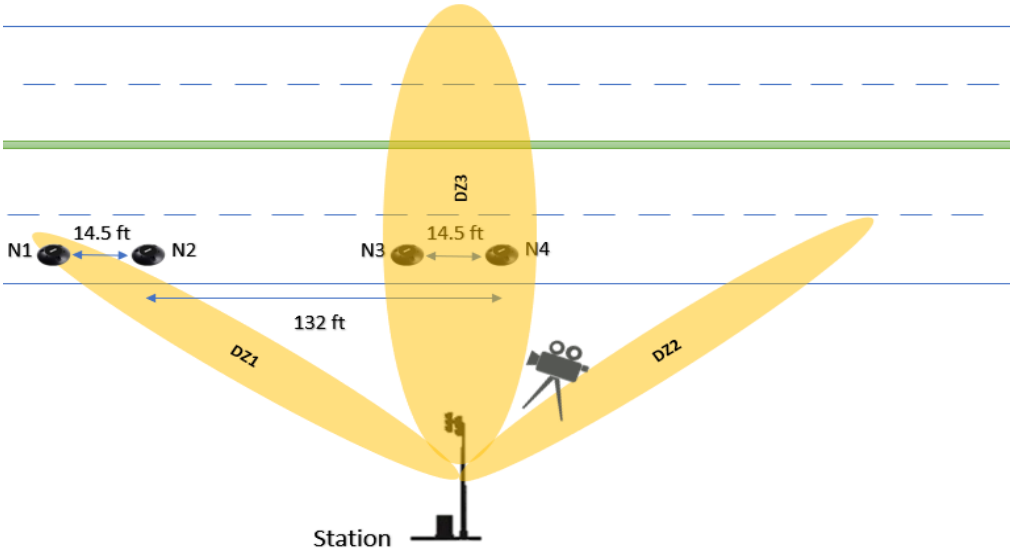


Figure 41. Layout of the third deployment.

Figure 41 shows an illustration of the deployment layout including the 30-degree unidirectional antenna that was used to establish an improved communication link. During the deployment, the antenna was instrumental in achieving very good results, as the AP was able to receive data from sensors during the full length of the deployment.

All algorithms used in the previous test were modified to enable the system to assign BTA to its associated vehicle in real time. To accomplish this, the following changes were made.

1- The REECE that collected BTAs from the right side of the road sent data to the other REECE via a serial cable.

2- The REECE responsible for collecting BTAs on the left side were watching the serial port and reading all data sent by the other REECE, and then archived the information into MACs2_DB database.

3- A python script utilized threads monitored in classified vehicle databases. The script verified number of unassigned vehicles and commenced the assignment script for each.

All scripts are discussed in Appendix A.

4.6.3.2 Data processing:

Collected data was analyzed after applying processing algorithms. Following is information extracted from the data.

1- Vehicle classes:

a- Camera:

Data represents vehicle groups that were recorded during the full length of the deployment. The total numbers of vehicles was 1076.

Table 56. Number of detected Vehicles and their classes (Camera)

Group	G1	G2	G3	G4
	(3) 0.27 %	(883)82.06%	(59)5.4%	(131)12.7%

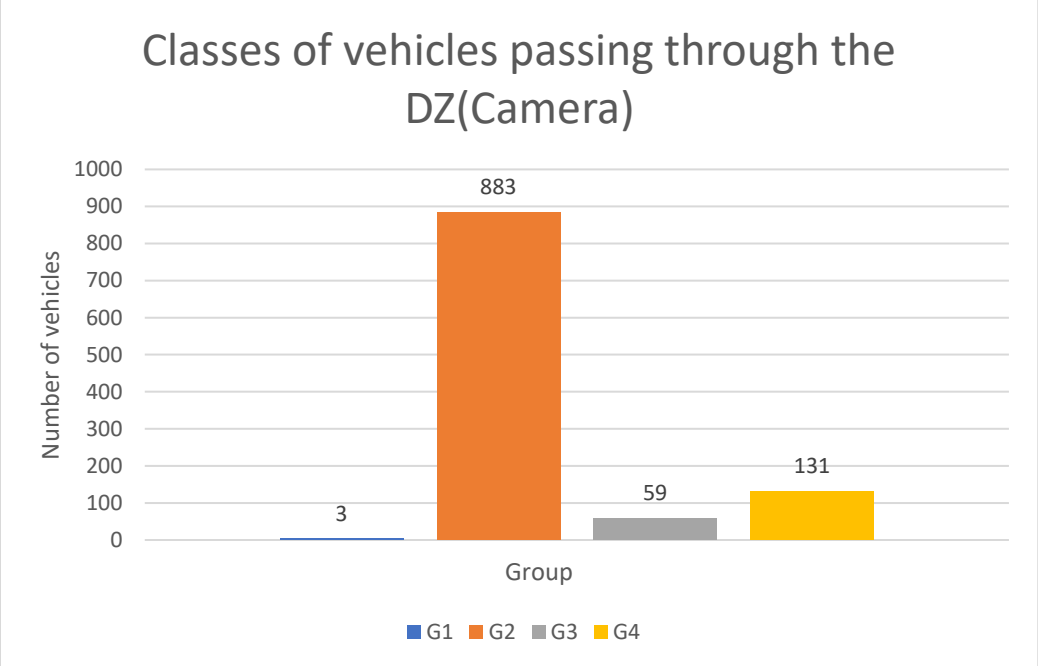


Figure 42. Distribution of the classes passing through the DZs (Camera).

b- Sensors

This data represents vehicle group information extracted from the developed classification algorithm based on reported sensors timestamps.

Total number of vehicles was 777.

Table 57. Number of detected Vehicles and their classes (Sensors)

Group	G1	G2	G3	G4
	23 (2.9%)	(626) 80.56%	(29) 3.7%	(99)12.74%

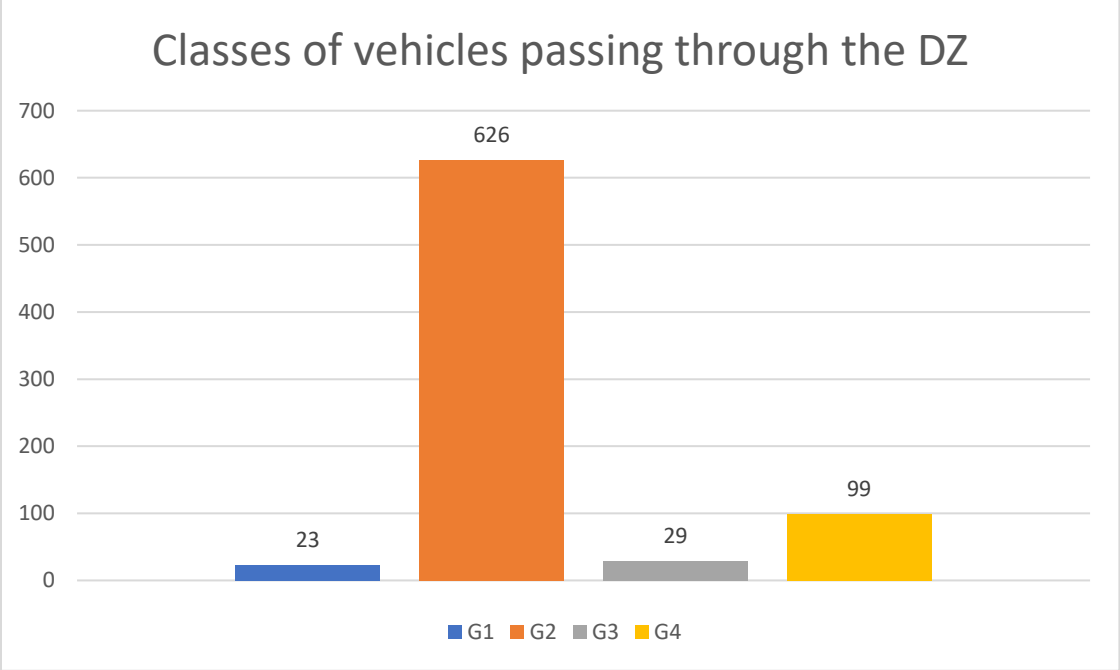


Figure 43. Distribution of the classes passing through the DZs (Sensors)

Following are the results of the real time, MACs directionality, and BTA-vehicle assignment algorithms.

Table 58. Results of the Real time MACs directionality and BTA-vehicle assignment algorithms

Traffic Direction	Number of Unique MACs	Assigned MACs
East bound traffic	337	96

The following table shows the percentages of each group of the assigned vehicles:

Table 59. Percentages of assigned vehicles

Group	G1	G2	G3	G4
	8.54%	81.69%	5.63%	4.22%

The results of the deployment suggest the following:

1- Only a third of the MACs that were traveling from the left DZ to the right DZ were assigned to their possible corresponding vehicles.

2- The algorithms were not able to assign all of the MACs to their corresponding vehicles due to these reasons:

   a- We only deployed the sensors on one of the two lanes, which lead to the inability to assign the MACs that were travelling on the second lane

   b- In cases were vehicles have multiple MACs, only one of the MACs would be assigned

   c- The sensors were reporting erroneous timestamps in some periods during the deployment, which lead to the inability of accurately detecting and classifying the vehicles during these periods.

4.7 iVCC Sensors Highway Layout

For reliable, accurate vehicle speed and length estimation, proper iVCCs installment is required inside the DZ. Also, since all sensors deployed on the ground inside specially designed enclosures, several sensor positioning scenarios inside the enclosure were investigated. First, the sensor is positioned to face the AP, and then, the battery faced the AP. Finally, the antenna of the BLE module was placed onto the sensor facing AP. The following figure shows the aforementioned placement scenarios.



Figure 44. Different placement scenarios for the sensor inside the enclosure.

The test was conducted in two environments at various distances: first, inside an anechoic chamber with the sensor fixed inside the enclosure and positioned on a table, and second, outside, with the sensor fixed inside the enclosure and positioned on the ground. Measurements were taken with a Vector Signal Analyzer (VSA)-a tool offering quick, high-resolution spectrum measurements, demodulation, and advanced time-domain analysis. This tool is also useful for characterizing burst, transient, or modulated signals used in communication, video, broadcast, radar, and ultrasound imaging applications [47].

- Anechoic chamber:
- Sensor faces VSA antenna

Table 60. Received power -Sensor faces receiver antenna- Anechoic chamber

Distance(ft)	Power(dBm)					Average power
2	-53.93	-57.34	-54.58	-53.53	-54.74	-54.824
4	-59.14	-60.52	-59.37	-58.77	-59.5	-59.46
6	-59.01	-63.9	-60.73	-62.13	-65.05	-62.164
8	-63.41	-60.67	-57.39	-60.09	-59.95	-60.302

- Battery faces VSA antenna

Table 61. Received power -Battery faces receiver antenna- Anechoic chamber

Distance(ft)	Power(dBm)					Average power
2	-54.82	-51.7	-52.06	-51.88	-53.04	-52.7
4	-56.99	-61.1	-59.88	-59.03	-59.63	-59.326
6	-56.47	-59.47	-59.24	-54.25	-56.47	-57.18
8	-59.39	-61.56	-60.94	-63.24	-62.37	-61.5

- BLE module faces VSA antenna

Table 62. Received power -BLE module faces receiver antenna- Anechoic chamber

Distance(ft)	Power(dBm)					Average power
2	-52.9	-52.12	-51.72	-52.67	-55.57	-52.996
4	-49.25	-52.68	-51.44	-51.17	-50.17	-50.942
6	-53.88	-57.24	-60.27	-59.48	-56.87	-57.548
8	-50.73	-53.31	-51.16	-51.06	-50.14	-51.28

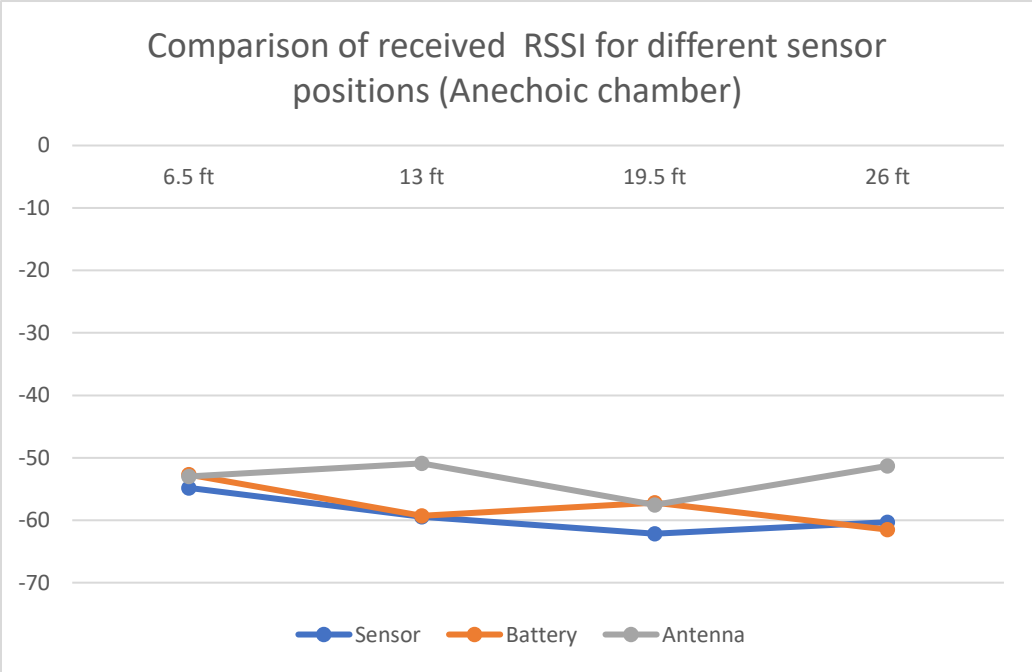


Figure 45. Received RSSI for different sensor positions (Anechoic chamber).

- Outside, with sensor on ground.
- Sensor faces the VSA antenna

Table 63. Received power -Sensor faces receiver antenna- Outside

Distance(ft)	Power(dBm)					Average power
2	-56.16	-56.79	-55.25	-57.24	-58.18	-56.724
4	-67.5	-63.87	-63.11	-63.19	-61.94	-63.922
6	-68.74	-70.07	-70.15	-71.14	-70.71	-70.162
8	-	-	-	-	-	-

- Battery faces VSA antenna

Table 64. Received power -Battery faces receiver antenna- Outside

Distance(ft)	Power(dBm)					Average power
2	-63.39	-63.21	-65.23	-60.71	-60.1	-62.528
4	-65.9	-64.93	-65.64	-65.15	-64.54	-65.232
6	-68.96	-67.72	-67.35	-68.28	-68.14	-68.09
8	-69.36	-69.14	-71.01	-63.55	-65.64	-67.74
10	-69.68	-70.67	-70.63	-70.6	-70.15	-70.346

- BLE module facing VSA antenna

Table 65. Received power -BLE module faces receiver antenna- Outside

Distance(ft)	Power(dBm)					Average power
2	-46.1	-45.34	-43.65	-46.14	-45.66	-45.378
4	-49.49	-52.09	-49.84	-49.94	-50.32	-50.336
6	-57.73	-54.54	-56.02	-55.58	-57.16	-56.206
8	-60.31	-58.4	-59.55	-60.1	-61.07	-59.886
10	-65.02	-62.81	-63.05	-64.32	-62.54	-63.548
15	-68.71	-68.06	-67.85	-66.71	-68.89	-68.044
20	-	-	-	-	-	-

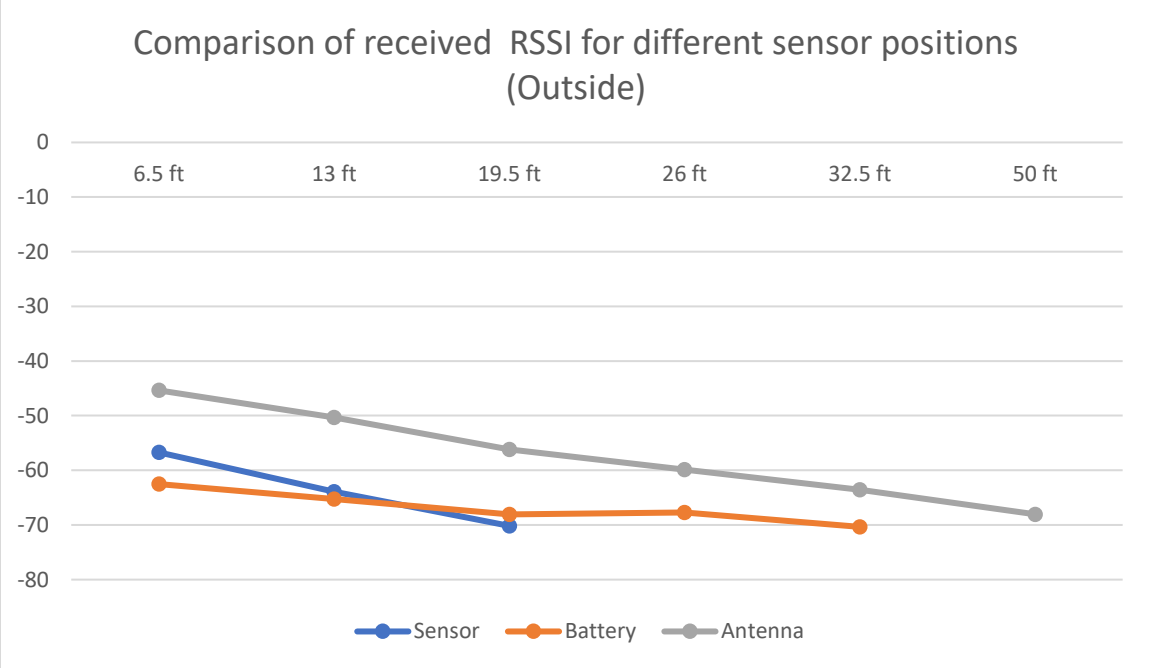


Figure 46. Received RSSI for different sensor positions (Outside).

Collected power measurements suggest the following conclusions.

1- Placing the sensor on the ground results in a decrease in sensitivity and, thus, reduced connection distance.

2- Placing the sensor so that its BLE module antenna faces the AP results in superior sensitivity.

Chapter 5 – Vehicle BTAs and Class Association

Providing reliable route choices per vehicle class data requires precise vehicle classification and highly accurate BTA-to-vehicle class attachment. Assigning a MAC address to the corresponding vehicle class can be accomplished by matching reported sensor timestamps to MAC detection timestamps. However, received MAC timestamps provide no indication of vehicle travel lane or direction. Thus, using only temporal analysis won't be enough to achieve high accuracy MAC/vehicle class assignment. To overcome this issue and to provide a more reliable solution, unidirectional antennas were used instead of omni directional antennas.

Adding multiple unidirectional antennas would limit DZ of each antenna to specific highway segments; hence, adding a spatial dimension to temporal data. Using two antennas establishes a precedent for using two DZs. In the meantime, two iVCCs were installed between the DZs. Given that a vehicle enters the first DZ (DZ1), MACs will be captured by the first antenna. After that, the two sensors will detect the vehicle and report timestamps to the AP for classifying the vehicle. Next, the vehicle will enter DZ2, and its MAC will be captured and saved in the MAC database. This will, in turn trigger, the assignment algorithm.

5.1 Classification algorithm

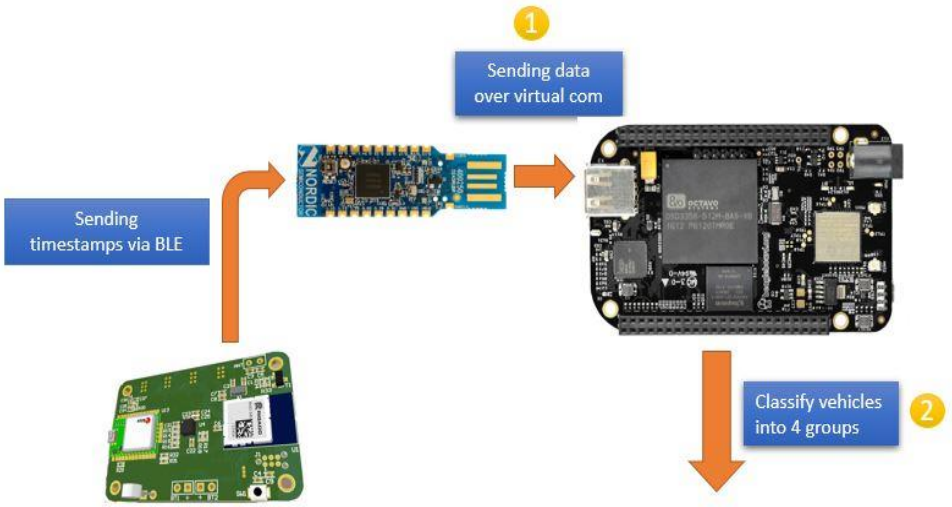
Software implementation for parsing data received by the BLE dongle, and then classifying the detected vehicle, is based on received timestamps.

Software development is divided into two parts:

1- Read data received by the dongle through a virtual COM, and then parse it.

2- Extract information from part one and execute the classification algorithm.

Following is an overview of the system data flow.



Group	FHWA-S_F
G1	F_1
G2	F_2—F_3
G3	F_4—F_7
G4	F_8—F_{13}

Figure 47. Overview of data flow inside the system.

5.1.1 Data parsing

The BLE dongle enables REECE to communicate with the sensors. Accordingly, it will receive timestamps from sensors before forwarding them to REECE. The algorithm flow diagram is shown below, followed by examples of data sent from iVCCs.

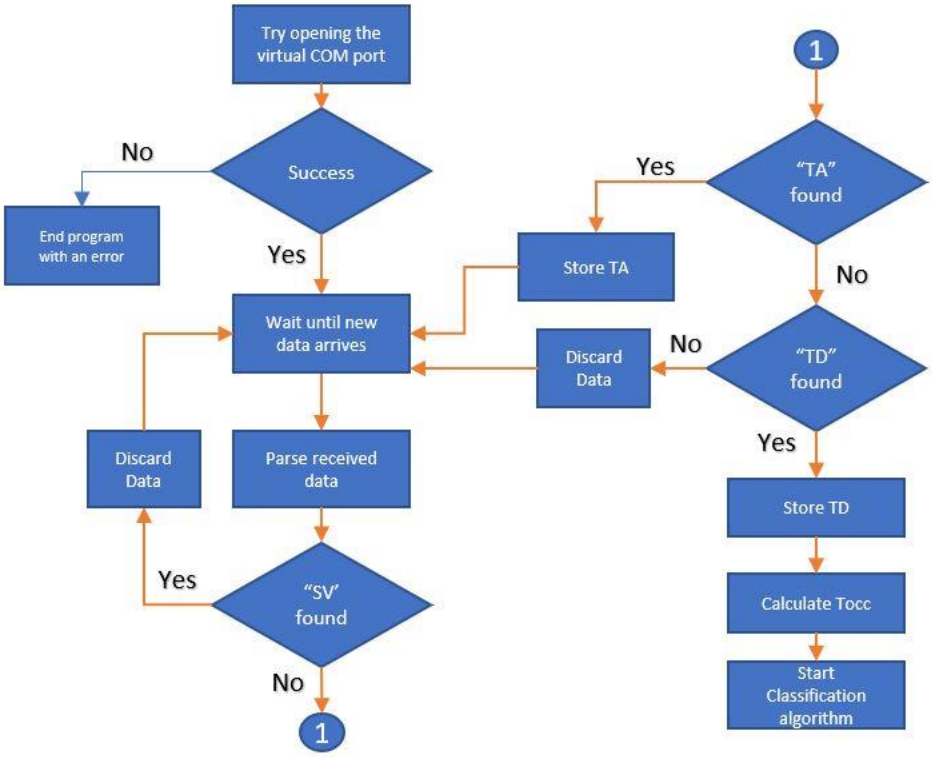


Figure 48. Flow diagram of parsing algorithm.

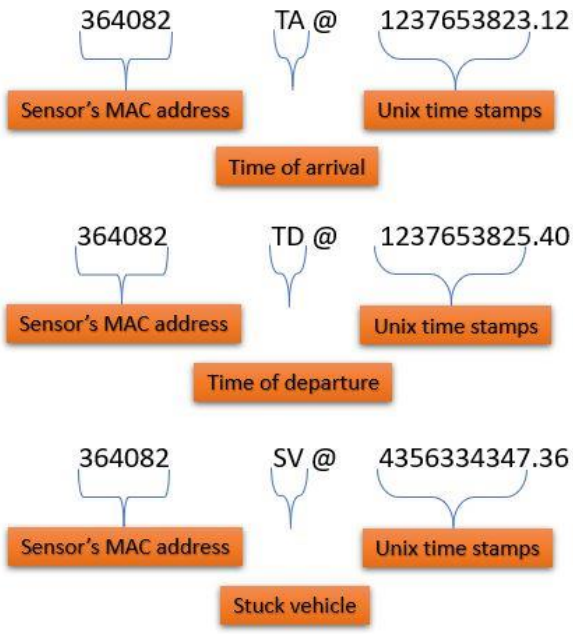


Figure 49. Examples of timestamps sent by the sensor.

Unix timestamps are a method for measuring time in seconds. The first timestamp counter was initiated at the Unix epoch on January 1, 1970, at UTC. Each timestamp represents the difference between a date and the Unix epoch. The full Parsing algorithm is described in Appendix A.

5.1.2 Classification

Length-based classification is used for classifying detected vehicles into one of four groups, as described in [47]. The method relies on two sensors in each lane at a predefined distance. Each sensor reports both arrival and departure time of the vehicle. After parsing the collected data, vehicle velocity between sensor nodes N_A and N_B can be calculated using the following formula:

$$\bar{v}_i \approx \frac{2 \times d^{(N_A \to N_B)}}{T_A^{(N_B)} - T_A^{(N_A)} + T_D^{(N_B)} - T_D^{(N_A)}}$$

where d is the distance between the two nodes.

Calculated velocity will be used in accordance with the difference between T_A and T_D (Tocc) for calculating vehicle magnetic length (**VML**):

$$\overline{VML} = \bar{v} \times T_{Occ}^{(N_i)} = \bar{v} \times \frac{T_D^{(N_A)} - T_A^{(N_A)} + T_D^{(N_B)} - T_A^{(N_B)}}{2}$$

VML is the disturbance in the Earth's magnetic field caused by the vehicle's metal mass [47]. The classification algorithm uses calculated VML to classify the vehicle to one of the

predefined groups, as described in Figure 50. Following is the code used to implement the classification algorithm.

```
def ClssifyVehicle(SensorIndex):
    if SensorIndex == 1:
       lane = "L1"
    elif SensorIndex == 3:
        lane = "L2"

    Vi = ((2.0 * distance) ) / ((TA[SensorIndex] - TA[SensorIndex - 1]) + (TD[SensorIndex]
          - TD[SensorIndex - 1]))

    Lm = (Vi / 2 ) * (Tocc[SensorIndex] + Tocc[SensorIndex - 1])

    if (Lm < 2.984):
        print("G1" + " " + str(TA[SensorIndex - 1]) + " " + str(TD[SensorIndex]) + " "
              + str(Vi) + " " + str(Lm) + " " + lane)
        sys.stdout.flush()

    elif (Lm > 2.984 and Lm < 10.971):
        print("G2" + " " + str(TA[SensorIndex - 1]) + " " + str(TD[SensorIndex]) + " "
              + str(Vi) + " " + str(Lm) + " " + lane)
        sys.stdout.flush()

    elif (Lm > 10.971 and Lm < 14.727):
        print("G3" + " " + str(TA[SensorIndex - 1]) + " " + str(TD[SensorIndex]) + " "
              + str(Vi) + " " + str(Lm) + " " + lane)
        sys.stdout.flush()


    elif (Lm > 14.727):
        print("G4" + " " + str(TA[SensorIndex - 1]) + " " + str(TD[SensorIndex]) + " "
             + str(Vi) + " " + str(Lm) + " " + lane)
        sys.stdout.flush()
```

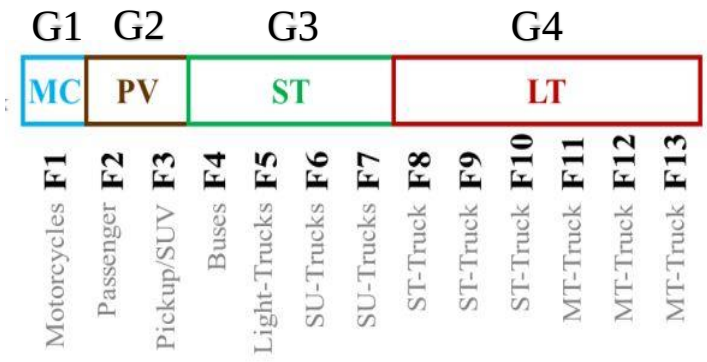


figure 50. Detailed vehicles' classes description

5.2 MAC to Vehicle assignment algorithm

The proper assignment of BTAs/vehicle class pairs will depend, as discussed earlier, on both spatial and temporal analysis. BT RSSI reported form the Ubertooth will play the most important role in the spatial dimension of the analysis. A study [49] was conducted to use BT RSSI to estimate the distance between connected devices. The relation between distance and RSSI was studied in different environments (e.g. indoor hall, Electro Magnetic Compatibility (EMC) chamber). RSSI is an indication of the power level the antenna receives—the higher the RSSI, the stronger the signal. Yang et al. [50] introduced several models (i.e., Leas Square Estimation, Three-border and Centroid method) to predict distance using reported RSSI. In this work, the researchers stated that RSSI-based triangulation positioning methods yield very good results. Another research group used a back propagation neural network model optimized by particle swarm optimization to predict distance and reduce positioning error [52].

An algorithm was developed to use the RSSI of each detected BTA to predict the lane of the corresponding vehicle. The algorithm makes decisions by examining different received RSSIs for each BTA, and then finding the Mode of the values. That way the data

will be as close as possible to the real RSSI and will reduce the effect of the environment, thus reducing the error of prediction.

The algorithm is divided into two parts, as described below.

5.2.1 Filtering BTAs

The first part of the algorithm will work exclusively on selecting BTAs that satisfy two conditions:

1- BTAs should be detected by both antennas

2- BTAs should be detected by the second antenna *after* being detected by the first one; difference in detection times should fall within a prespecified threshold.

After storing BTAs that satisfy the previous conditions, the algorithm will attempt to find the Mode of RSSI values for each BTA in both DZs. Given there was no Mode, the mean of the values will be calculated, along with the BTA, initial detection in the DZ1, and Mode of RSSIs in DZ1 (or the Mean). First time of detection in DZ2 and Mode of RSSIs in DZ2 (or the Mean) will be stored in database MACS_DF. Following is the flow diagram of the algorithm.

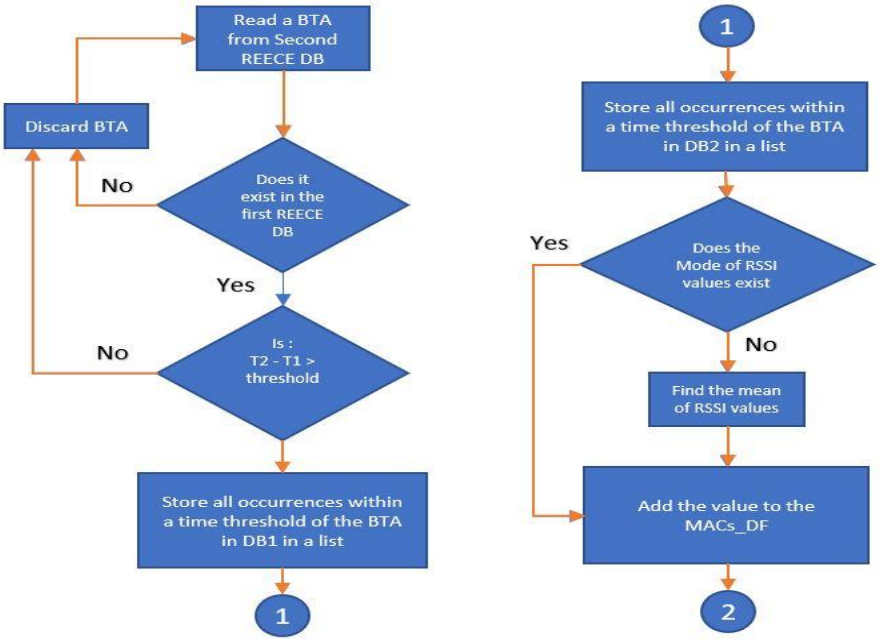


Figure 51. Flow diagram of BTAs filtering algorithm.

The full code is discussed in Appendix A.

5.2.2 BTA/Vehicle assignment algorithm

The second part of the algorithm focuses on finding the lane in which the BTA was detected. Each sensor is placed on a lane, having two sensors on each lane. All sensors report timestamps to the AP, which, in turn, classifies the vehicle. Next, database Sensors DB will store the following: TA1 (i.e., time of arrival for the first sensor on each lane); TD2 (i.e., time of Departure for the second sensor on each lane); and vehicle class. The algorithm starts by filtering the resultant database from the previous step by comparing each record of the sensor database to records in MACs DB, selecting only BTAs that fall within a specific time threshold before TA1 and after TD2. The two highest RSSI values will be assigned to vehicles on lane1 and lane 2, respectively. Figure (52) shows the flow diagram of the algorithm.

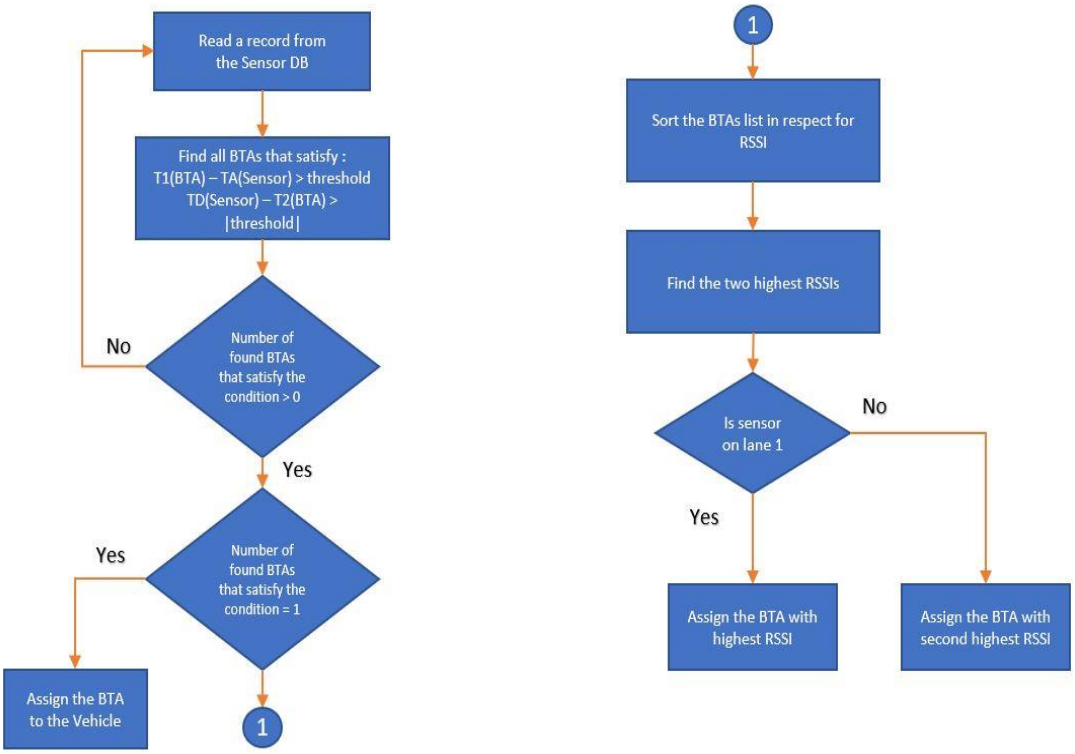


Figure 52. Flow diagram of the BTA/Vehicle assignment algorithm.

A full description of the algorithm is provided in Appendix A.

5.3 Privacy

Because of its uniqueness, collecting and analyzing BTAs always introduces privacy concerns. These legitimate fears should be addressed by researchers, offering BT users assurance that their data will not be shared or used in any way to threaten their privacy. Protocols were followed for the research conducted in this thesis to adhere to the conduct of protecting driver privacy, as follows:

1- BTA is in no way connected to the person using the BT device.

2- Only the lower half—LAP—of the BTA was collected.

3- BTAs databases are cleared each time 20,000 records have been recorded.

Chapter 6 – Conclusion and Future Work

Conclusion:

This thesis reported research about the design of an IoT system that combines two systems:

1- BT identification system that uses BT sniffers to detect vehicles' MAC addresses and match them as vehicles travel across the state of Oklahoma
2- Vehicle classification system that make use of the iVCCSs to classify vehicles at POIs

The system will be used to enable traffic agencies of studying route selection per vehicle class in real time using cost-effective equipment. Information extracted from the system will aid in decision-making for transportation system infrastructure improvements.

Several algorithms were designed to enable the system in achieving stated goal, including:

1- MACs collection: using Ubertooth to collect MACs of vehicles passing through the studied area
2- Vehicles classification: receiving timestamps of passing vehicles using BLE, and then utilizing timestamps to classify vehicles
3- MAC-Vehicle assignment: utilizing data provided by previous algorithms to match each vehicle with its possible MAC address

The system achieved an accuracy of 85% for classifying vehicles using timestamps reported by iVCCs and was able to attach 30% of vehicles with their MAC address

Future Work:

Several modifications can be added to the system to increase algorithms' accuracy, including:

1- Build a model to estimate RSSI of detected MACs based on distance
2- Modify classification algorithm thresholds to reduce classification algorithm errors
3- Send collected data to a server, which in turn will assign vehicles to their prospective MACs, and then validate assignment accuracy by monitoring the vehicle route selection

References

[1] https://www.idc.com/getdoc.jsp?containerId=prUS44159418

[2] Harrison, C., Eckman, B., Hamilton, R., Hartswick, P.,Kalagnanam, J., Paraszczak, J., &Williams, P. (2010). Foundations for Smarter Cities. IBM Journal of Research and Development, 54(4).

[3] Washburn, D., Sindhu, U., Balaouras, S., Dines, R. A., Hayes, N. M., & Nelson, L. E. (2010). Helping CIOs Understand "Smart City" Initiatives: Defining the Smart City, Its Drivers, and the Role of the CIO. Cambridge, MA: Forrester Research, Inc.

[4] Giffinger, R., Fertner, C., Kramar, H., Kalasek, R., Pichler-Milanović, N., & Meijers, E. (2007). Smart Cities: Ranking of European Medium-Sized Cities. Vienna, Austria: Centre of Regional Science (SRF), Vienna University of Technology. Available from http://www.smartcities.eu/download/smart_cities_final_report.pdf

[5] Chourabi, H., Nam, N., Walker, S., Gil-Garcia, J.Ramon., Mellouli, S., Nahon, K., Pardo,T., & Jochen Scholl, H. 2012 45th Hawaii International Conference on System Sciences, Understanding Smart Cities: An Integrative Framework

[6] Gartner. (2014, March 19). Gartner says the Internet of Things will transform the data center. Retrieved from http://www.gartner.com/newsroom/id/2684616

[7] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," IDC iView: IDC Anal. Future, vol. 2007, pp. 1–16, Dec. 2012

[8] https://www.statista.com/statistics/764026/number-of-iot-devices-in-use-worldwide/

[9] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M., IEEE COMMUNICATIONS SURVEYS &AMP TUTORIALS · JANUARY 2015 Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications

[10] Atzori, L., Iera, A., Morabito, G., The International Journal of Computer and Telecommunications Networking (October 2010) The Internet of Things: A Survey

[11] D. S. Kim, J. D. Porter, S. Park, A. Saeedi, A. Mohseni, N. Bathaee, and M. Nelson, Bluetooth Data Collection System for Planning and Arterial Management.

[12] W. Wang, J. Attanucci, and N. Wilson, "Bus Passenger Origin-Destination Estimation and Related Analyses Using Automated Data Collection Systems," J. Public Transp., vol. 14, no. 4, pp. 131–150, Dec. 2011

[13] Large Truck Origin Destination Monitoring | Freight Planning, New Jersey, 2008

[14] Blogg, M., Semler, C., Hingorani, M., Troutbeck, R. Australasian Transport Research Forum 2010 Proceedings 29 September – 1 October 2010, Travel Time and Origin-Destination Data Collection using Bluetooth MAC Address Readers.

[15] Y. Wang, B. N. Araghi, Y. Malinovskiy, J. Corey, and T. Cheng, "Error Assessment for Emerging Traffic Data Collection Devices: WSDOT Research Report," Seattle, Washington, 2014

[16] Calabrese, Francesco, Giusy Di Lorenzo, Liang Liu, and Carlo Ratti. "Estimating Origin-Destination Flows Using Mobile Phone Location Data." IEEE Pervasive Computing 10, no. 4 (April 2011): 36–44.

[17] G. Schiffer, R., Tools of the Trade Conference on Transportation Planning for Small and Medium-sized Communities (September 2016), Alternate Methodologies for Origin-Destination Data Collection, retrieved from http://www.trbtoolsofthetrade.org/files/theme/C2-4-presentation.pdf

[18] Electronic Frontier Foundation, https://www.eff.org/pages/automated-license-plate-readers-alpr

[19] License Plate Matching Techniques, https://www.fhwa.dot.gov/ohim/handbook/chap4.pdf

[20] Carpenter, C., Fowler, M., J. Adler, T. Journal of the Transportation Research Board (January 2012) Generating Route-Specific Origin–Destination Tables Using Bluetooth Technology

[21] Stephanie Box (December 2011) Arterial Roadway traffic Data Collection Using Bluetooth Technology [22] Vinagre Díaz, J., González, A., Wilby, M., IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 17, NO. 1, JANUARY 2016, Bluetooth Traffic Monitoring Systems for Travel Time Estimation on Freeways

[23] Murphy, P., Welsh, E., Frantz, J. P. 2002. Using Bluetooth for short-term ad hoc connections between moving vehicles: a feasibility study. Vehicular Technology Conference. IEEE 55th Vehicular Technology Conference. VTC Spring 2002 (Cat. No.02CH37367), 1, 414-418. doi:10.1109/VTC.2002.1002746

[24] Khliefat, I. and Shatnawi, I. (2017) An Optimization of Bluetooth Sensor Locations for Origin Destination in an Urban Network. Journal of Transportation Technologies, 7, 367- 375.

[25] J.C.Haartsen,"The Bluetooth radio system,"IEEEPers. Commun.Mag., vol. 7, no. 1, pp. 28–36, Feb. 2000.

[26] https://macaddresschanger.com/what-is-bluetooth-address-BD_ADDR

[27] Juan José Vinagre Díaz, Ana Belén Rodríguez González, and Mark Richard Wilby, Bluetooth Traffic Monitoring Systems for Travel Time Estimation on Freeways, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 17, NO. 1, JANUARY 2016

[28] Philip John , T., Darcy M, B., Staneley E, Y., James, W., Nicholas, G., James R, S., , Transportation Research Board 88th Annual Meeting, Washington DC, 2009, Continuing Evolution of Travel Time Data Information Collection and Processing

[29] Tsubota, T., Yoshii, T., International Symposium of Transport Simulation (ISTS'16 Conference), June 23~25, 2016, An Analysis of the Detection Probability of MAC Address from a Moving Bluetooth Device

[30] Stevanovic, A., Olarte, C. L., Galletebeitia, A., Galletebeitia , B., and Kaisar, E, 18th World Congress on Intelligent Transport Systems, October 2011, Testing Accuracy and Reliability of MAC Readers to Measure Arterial Travel Times

[31] Wang, Y., Vrancken, J. L. M., Seidel, P. 2011. Measure travel time by using Bluetooth detectors on freeway. In Proceedings of the ITS World Congress (pp. 1–5).

[32] Malinovskiy, Y., Wu, Y.-J., Wang, Y., Lee, U. K. 2010. Field Experiments on Bluetooth-Based Travel Time Data Collection. In Transportation Research Board 89th Annual Meeting. Retrieved from http://trid.trb.org/view.aspx?id=910907

[33] http://diamondtraffic.com/technicaldescription/124

[34] Traffic Detection by Inductive Loops, Joel Johnson, Yr 12, Physics ERT 3.1 Final, 11/4/16

[35] https://en.wikipedia.org/wiki/Induction_loop

[36] https://www.nordicsemi.com/Products/Low-power-short-range wireless/nRF52840

[37] Mohammed, S. (July 2018), Intelligent Power Aware Algorithms for Traffic Sensors.

[38] https://www.quectel.com/product/l96.htm

[39] Alamiri, M, (Dec 2017), IoT Systems for Travel Time Estimation

[40] https://beagleboard.org/black

[41] https://www.sierrawireless.com/products-and-solutions/routers-gateways/rv50/

[42] https://hackerwarehouse.com/product/ubertooth-one/

[43] http://infocenter.nordicsemi.com/pdf/nRF52840_Dongle_User_Guide_v1.0.pdf

[44] Lee, J., Su, Y., Shen, C., The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON) Nov. 5-8, 2007, Taipei, Taiwan, A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi

[45] Novelbits site, https://www.novelbits.io/basics-bluetooth-low-energy/

[46] Nordic semi info center page, https://infocenter.nordicsemi.com.

[47] https://literature.cdn.keysight.com/litweb/pdf/5990-6405EN.pdf

[48] D. Evans, "The Internet of things: How the next evolution of the Internet is changing everything," CISCO, San Jose, CA, USA, White Paper, 2011.

[49] Jung, J., Kang, D., Bae, C., CSNC 2013: The Eighth International Conference on Systems and Networks Communications, Distance Estimation of Smart Device using Bluetooth.

[50] Wang, Y., Zhao, Y., Cuthbert, L., 2013 IEEE 10th Consumer Communications and Networking Conference (CCNC), Bluetooth Positioning using RSSI and Triangulation Methods

[51] Li, G., Geng, E., Ye, Z., Xu, Y., Lin, J., Pang, Y. sensors open access journal (August 2018) Indoor Positioning Algorithm Based on the Improved RSSI Distance Model

[52] iAP: http://clipart-library.com/clipart/82967.htm

[53] BLE: http://www.clker.com/clipart-ble.html

[54] https://www.novelbits.io/basics-bluetooth-low-energy/

[55] http://microchipdeveloper.com/wireless:ble-gatt-data-organization

Appendix A: Algorithms and codes

1. BLE Dongle Software:

The described application is written in Embedded C using SEGGER Embedded studio for ARM. It uses Nordic UART service (NUS) which is a simple GATT-based service with two properties: TX (characteristic with "notify" properties) and RX (characteristic with "write" properties). Data received from the peer is passed to the application, and the data received from the application of this service is sent to the peer as Handle Value Notifications. This module demonstrates how to implement a custom GATT-based service and characteristics using the SoftDevice (Bluetooth Stack) . The service is used by the application to send and receive ASCII text strings to and from the peer. The software also uses USB Communications Device Class (CDC) Abstract Control Model (ACM). The ACM subclass describes the bidirectional communication flow popularly known as Virtual Serial Port. The class has two interfaces:

- COMM interface that contains one IN endpoint. The COMM interface is used to notify the host about serial state.
- DATA interface that contains one IN and one OUT bulk endpoint. The DATA interface allows to send or receive bulk data.

Thus, data will be sent from the REECE to the application (resides on the dongle) using the DATA interface of the virtual serial port. After that, the USB CDC ACM module will use the NUS service to send the data to the SoftDevice stack and then to the peer. The same procedure applies when the peer send data to REECE but in a reverse order. Following is

an illustration of the data flow between the REECE and the iVCC sensor from the BLE dongle point of view:

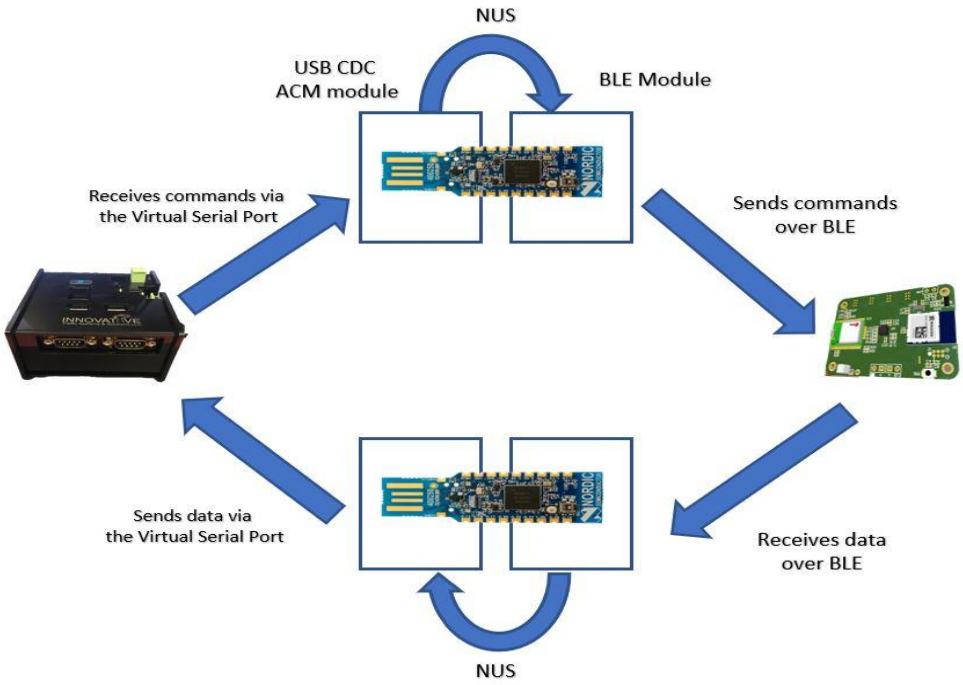


Figure 53. Data flow between the REECE and the iVCC sensor.

The code starts by calling important libraries needed by the software to function probably, then defining some variables:

```
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include "nrf.h"
#include "ble_advdata.h"
#include "ble_advertising.h"
#include "ble_conn_params.h"
#include "ble_nus.h"
#include "ble_nus_c.h"
#include "app_util_platform.h"
#include "bsp_btn_ble.h"
#include "nrf_drv_usbd.h"
#include "nrf_drv_clock.h"
#include "nrf_gpio.h"
#include "nrf_delay.h"
#include "nrf_drv_power.h"
#include "app_usbd_core.h"
#include "app_usbd.h"
#include "app_usbd_string_desc.h"
#include "app_usbd_cdc_acm.h"
#include "app_usbd_serial_num.h"
#include "nordic_common.h"
#include "app_error.h"
#include "app_uart.h"
#include "ble_db_discovery.h"
#include "app_timer.h"
#include "app_util.h"
#include "bsp_btn_ble.h"
#include "ble.h"
#include "ble_gap.h"
#include "ble_hci.h"
#include "ble_conn_state.h"
#include "nrf_sdh.h"
#include "nrf_sdh_ble.h"
#include "nrf_sdh_soc.h"
```

```
#include "ble_nus_c.h"
#include "nrf_ble_gatt.h"
#include "nrf_pwr_mgmt.h"
#include "nrf_ble_scan.h"
#include "nrf_log.h"
#include "nrf_log_ctrl.h"
#include "nrf_log_default_backends.h"
#include "string.h"

#define APP_BLE_OBSERVER_PRIO   3                       /**< BLE observer priority of
the application

#define APP_DEFAULT_TX_POWER    8                       // Transmit power
#define UART_TX_BUF_SIZE        256                     /**< UART transmission buffer
size. */
#define UART_RX_BUF_SIZE        256                     /**< UART receiver buffer size.
*/
#define NUS_SERVICE_UUID_TYPE   BLE_UUID_TYPE_VENDOR_BEGIN  /**<
UUID type for the Nordic UART Service

#define ECHOBACK_BLE_UART_DATA  1       /**< Echo the UART data that is
received over the Nordic UART Service (NUS) back to the sender. */



BLE_NUS_C_ARRAY_DEF(m_ble_nus_c,
NRF_SDH_BLE_CENTRAL_LINK_COUNT);       /**< creating a NUS module
instances array, with the size of NRF_SDH_BLE_CENTRAL_LINK_COUNT . */
BLE_DB_DISCOVERY_ARRAY_DEF(m_db_disc,
NRF_SDH_BLE_CENTRAL_LINK_COUNT);  /**< creating a database discovery
instances array, with the size of NRF_SDH_BLE_CENTRAL_LINK_COUNT . */
NRF_BLE_GATT_DEF(m_gatt);                                   /**< creating a GATT
module instance. */
NRF_BLE_SCAN_DEF(m_scan);                                    /**< creating a Scanning
Module instance. */

static uint16_t m_ble_nus_max_data_len = BLE_GATT_ATT_MTU_DEFAULT -
OPCODE_LENGTH - HANDLE_LENGTH; /**< Maximum length of data (in bytes)
that can be transmitted to the peer by the Nordic UART service module. */
static char m_nus_data_array[BLE_NUS_MAX_DATA_LEN];                  /**< create
a data array
```

After that, important USB CDC ACM library initializations are called. Starting by defining which USB port will act as a DATA interface, then creating a data array to hold the data received by the module. All the instances will be passed to the module through the APP_USBD_CDC_ACM_GLOBAL_DEF.

```
#define CDC_ACM_DATA_INTERFACE        1
#define CDC_ACM_DATA_EPIN      NRF_DRV_USBD_EPIN1
#define CDC_ACM_DATA_EPOUT     NRF_DRV_USBD_EPOUT1


static char m_cdc_data_array[BLE_NUS_MAX_DATA_LEN];

/** @brief CDC_ACM class instance */
APP_USBD_CDC_ACM_GLOBAL_DEF(m_app_cdc_acm,
                            cdc_acm_user_ev_handler,
                            CDC_ACM_COMM_INTERFACE,
                            CDC_ACM_DATA_INTERFACE,
                            CDC_ACM_COMM_EPIN,
                            CDC_ACM_DATA_EPIN,
                            CDC_ACM_DATA_EPOUT,
                            APP_USBD_CDC_COMM_PROTOCOL_AT_V250);
```

The main function starts by initializing multiple hardware and software modules like logger unit, Timers, CPU's clock, SoftDevice stack, Power management, NUS service etc.

```
int main(void)
{
    ret_code_t ret;
    static const app_usbd_config_t usbd_config = {
        .ev_state_proc = usbd_user_ev_handler
    };
    // Initialize.
```

```
    log_init();
    timers_init();
    app_usbd_serial_num_generate();
    ret = nrf_drv_clock_init();
   APP_ERROR_CHECK(ret);
    NRF_LOG_INFO("USBD BLE UART example started.");

    ret = app_usbd_init(&usbd_config);
    APP_ERROR_CHECK(ret);

    app_usbd_class_inst_t const * class_cdc_acm =
app_usbd_cdc_acm_class_inst_get(&m_app_cdc_acm);
    ret = app_usbd_class_append(class_cdc_acm);
    APP_ERROR_CHECK(ret);
    buttons_leds_init();
    db_discovery_init();
    power_management_init();
    ble_stack_init();
    gatt_init();
    nus_c_init();
    scan_init();
    ret = app_usbd_power_events_enable();
    APP_ERROR_CHECK(ret);
    // Start execution.
    printf("BLE UART central example started.\r\n");
    NRF_LOG_INFO("BLE UART central example started.");
    scan_start();


    // Enter main loop.
    for (;;)
    {
      while (app_usbd_event_queue_process())
          {
            /* Nothing to do */
          }
      idle_state_handle();
    }
}
```

scan_init () is the function responsible for initializing the scanning procedure of the central device, it is responsible of setting all the scanning parameters, e.g. the physical layer, scan timeout, filter policy, scan duration, etc.

```
static void scan_init(void)
{
    ret_code_t          err_code;
    nrf_ble_scan_init_t init_scan;
    ble_gap_scan_params_t scan_params;
    memset(&scan_params, 0, sizeof(ble_gap_scan_params_t));
    scan_params.extended      = true,
    scan_params.active        = 1,
    scan_params.interval      = NRF_BLE_SCAN_SCAN_INTERVAL,
    scan_params.window        = NRF_BLE_SCAN_SCAN_WINDOW,
    scan_params.timeout       = NRF_BLE_SCAN_SCAN_DURATION,
    scan_params.filter_policy = BLE_GAP_SCAN_FP_ACCEPT_ALL,
    scan_params.scan_phys     = BLE_GAP_PHY_CODED | BLE_GAP_PHY_1MBPS,
    memset(&init_scan, 0, sizeof(init_scan));

    init_scan.connect_if_match = true;
    init_scan.conn_cfg_tag     = APP_BLE_CONN_CFG_TAG;
    err_code = nrf_ble_scan_init(&m_scan, &init_scan, scan_evt_handler);
    APP_ERROR_CHECK(err_code);
    err_code = nrf_ble_scan_filter_set(&m_scan, SCAN_UUID_FILTER, &m_nus_uuid);
    APP_ERROR_CHECK(err_code);
    err_code = nrf_ble_scan_filters_enable(&m_scan, NRF_BLE_SCAN_UUID_FILTER,
false);
    APP_ERROR_CHECK(err_code);
    err_code = sd_ble_gap_tx_power_set(BLE_GAP_TX_POWER_ROLE_SCAN_INIT,
BLE_CONN_HANDLE_INVALID,
                                       APP_DEFAULT_TX_POWER);
    APP_ERROR_CHECK(err_code);
}
```

The central device keeps scanning all the time until it finds a connectable peripheral, then the flow control will be passed to ble_nus_c_evt_handler which is responsible of processing all the events on the BLE side that are related to the NUS service. When a NUS BLE event happens, the handler is triggered and starts checking the type of the event in order to take the proper action, three events are noticed in the code:

1- Discovery complete: if the central finds a peripheral that applies to the filters assigned in scan_init () It starts the connection procedure, enables the notification from the NUS service and logs the statues of the connection
2- Data received: if the central device receives any data through the BLE link from a connected device, it transfers the data immediately to the USB CDC ACM module using app_usbd_cdc_acm_write () which will write the received data to the virtual serial port.
3- Device Disconnected: if a connected peripheral is disconnected the central device starts scanning for other connectable devices.

```
static void ble_nus_c_evt_handler(ble_nus_c_t * p_ble_nus_c, ble_nus_c_evt_t const *
p_ble_nus_evt)
{
    ret_code_t err_code;
    ret_code_t ret ;

    switch (p_ble_nus_evt->evt_type)
    {
        case BLE_NUS_C_EVT_DISCOVERY_COMPLETE:
            p_ble_nus_c->handles.nus_tx_cccd_handle = p_ble_nus_evt-
>handles.nus_tx_cccd_handle;
            p_ble_nus_c->handles.nus_rx_handle = p_ble_nus_evt->handles.nus_rx_handle;
            p_ble_nus_c->handles.nus_tx_handle = p_ble_nus_evt->handles.nus_tx_handle;
```

```
            err_code = ble_nus_c_tx_notif_enable(p_ble_nus_c);
            APP_ERROR_CHECK(err_code);
            NRF_LOG_INFO("Connected to device with Nordic UART Service.");
            break;

        case BLE_NUS_C_EVT_NUS_TX_EVT:
        ret = app_usbd_cdc_acm_write(&m_app_cdc_acm,
                                     p_ble_nus_evt->p_data,
                                     p_ble_nus_evt->data_len);

            break;

        case BLE_NUS_C_EVT_DISCONNECTED:
            NRF_LOG_INFO("Disconnected.");
            scan_start();
            break;
    }
}
```

Another important handler is the ble_evt_handler, this handler takes control whenever a BLE events occurs. After being triggered by an event it starts by checking the type of the event. Some of the events are the following:

1- Device connected: once a peripheral is connected, the application starts by logging the connection handler, checking if the number of connected devices is still less than the prespecified constant NRF_SDH_BLE_CENTRAL_LINK_COUNT and make sure the device's connection handle is passed to the NUS event handler and it is added to the discovery database.

2- Device disconnected: when a peripheral disconnects, the program logs the connection handler and the reason of disconnection and reinitiate the NUS service.

3- Physical layer update request: in case the connected peripheral requested an update, this event will trigger the handler to log the request then updates the physical layer modulation.

```
Static void ble_evt_handler(ble_evt_t const * p_ble_evt, void * p_context)
{
ret_code_t err_code;
ble_gap_evt_t const * p_gap_evt = & p_ble_evt->evt.gap_evt;

switch(p_ble_evt->header.evt_id)
{
   case BLE_GAP_EVT_CONNECTED:

NRF_LOG_INFO("Connection 0x%x established, starting DB discovery.",
           p_gap_evt->conn_handle);

APP_ERROR_CHECK_BOOL(p_gap_evt->conn_handle <
NRF_SDH_BLE_CENTRAL_LINK_COUNT);

err_code = ble_nus_c_handles_assign( & m_ble_nus_c[p_gap_evt->conn_handle],
p_ble_evt->evt.gap_evt.conn_handle, NULL);
APP_ERROR_CHECK(err_code);

err_code = bsp_indication_set(BSP_INDICATE_CONNECTED);
APP_ERROR_CHECK(err_code);

err_code = ble_db_discovery_start( & m_db_disc[p_gap_evt->conn_handle],
p_ble_evt->evt.gap_evt.conn_handle);
if (err_code != NRF_ERROR_BUSY)
APP_ERROR_CHECK(err_code);

err_code = sd_ble_gap_tx_power_set(BLE_GAP_TX_POWER_ROLE_CONN,
p_gap_evt->conn_handle, APP_DEFAULT_TX_POWER);
APP_ERROR_CHECK(err_code);

if (ble_conn_state_central_conn_count() !=
NRF_SDH_BLE_CENTRAL_LINK_COUNT)
scan_start();
```

```
break;

err_code = bsp_indication_set(BSP_INDICATE_CONNECTED);
APP_ERROR_CHECK(err_code);


case BLE_GAP_EVT_DISCONNECTED:

NRF_LOG_INFO("Disconnected. conn_handle: 0x%x, reason: 0x%x",
            p_gap_evt->conn_handle,
                    p_gap_evt->params.disconnected.reason);

ble_nus_c_init_t init;
init.evt_handler = ble_nus_c_evt_handler;
err_code = ble_nus_c_init( & m_ble_nus_c[p_gap_evt->conn_handle], & init);
APP_ERROR_CHECK(err_code);

break;


case BLE_GAP_EVT_TIMEOUT:
if (p_gap_evt->params.timeout.src == BLE_GAP_TIMEOUT_SRC_CONN)
   {
      NRF_LOG_INFO("Connection Request timed out.");
   }
   break;



   case BLE_GAP_EVT_SEC_PARAMS_REQUEST:
   // Pairing not supported.
   err_code = sd_ble_gap_sec_params_reply(
   p_ble_evt->evt.gap_evt.conn_handle,
BLE_GAP_SEC_STATUS_PAIRING_NOT_SUPP, NULL, NULL);
   APP_ERROR_CHECK(err_code);
   break;

   case BLE_GAP_EVT_CONN_PARAM_UPDATE_REQUEST:
   // Accepting parameters requested by peer.
   err_code = sd_ble_gap_conn_param_update(p_gap_evt->conn_handle,
   & p_gap_evt->params.conn_param_update_request.conn_params);
```

```
    APP_ERROR_CHECK(err_code);
    break;

    case BLE_GAP_EVT_PHY_UPDATE_REQUEST:
    {
        NRF_LOG_DEBUG("PHY update request.");
    ble_gap_phys_t const
    phys =
    {
    .rx_phys = BLE_GAP_PHY_AUTO,
    .tx_phys = BLE_GAP_PHY_AUTO,
    };
    err_code = sd_ble_gap_phy_update(p_ble_evt->evt.gap_evt.conn_handle, & phys);
    APP_ERROR_CHECK(err_code);
    } break;

    case BLE_GATTC_EVT_TIMEOUT:
    // Disconnect on GATT Client timeout event
    NRF_LOG_DEBUG("GATT Client Timeout.");
    err_code = sd_ble_gap_disconnect(p_ble_evt->evt.gattc_evt.conn_handle,

BLE_HCI_REMOTE_USER_TERMINATED_CONNECTION);
    APP_ERROR_CHECK(err_code);
    break;

    case BLE_GAP_EVT_PHY_UPDATE:
    {
        ble_gap_evt_phy_update_t
    const * p_phy_evt = & p_ble_evt->evt.gap_evt.params.phy_update;

    if (p_phy_evt->status ==
BLE_HCI_STATUS_CODE_LMP_ERROR_TRANSACTION_COLLISION)
    {
    // Ignore LL collisions. \
            NRF_LOG_DEBUG("LL transaction collision during PHY update.");
    break;
    }


    ble_gap_phys_t
    phys = {0};
```

```
    phys.rx_phys = p_phy_evt->tx_phy;
    phys.rx_phys = p_phy_evt->tx_phy;
    NRF_LOG_INFO("PHY update %s. PHY set to %s.",
    (p_phy_evt->status == BLE_HCI_STATUS_CODE_SUCCESS) ?
    "accepted": "rejected",
    phy_str(phys));
    } break;


    case BLE_GATTS_EVT_TIMEOUT:
    // Disconnect on GATT Server timeout event.
    NRF_LOG_DEBUG("GATT Server Timeout.");
    err_code = sd_ble_gap_disconnect(p_ble_evt->evt.gatts_evt.conn_handle,

BLE_HCI_REMOTE_USER_TERMINATED_CONNECTION);
    APP_ERROR_CHECK(err_code);
    break;

    default:
    break;
    }
    }
```

nus_data_handler is another handler that processes the data received from the Nordic UART BLE Service and sends it to the USBD CDC ACM module, when a BLE_NUS_C_EVT_NUS_TX_EVT (Data received over BLE) event occurs it triggers this handler to take control. This handler starts by logging the status of the link, then copies the received data to the m_nus_data_array using the following:

memcpy(m_nus_data_array, p_evt->p_data, p_evt->data_len);

After that, it adds the end line string “\r\n” to the end of the received data as long as its size is less than BLE_NUS_MAX_DATA_LEN, then sends the data to the USB module by executing the following code:

ret_code_t ret = app_usbd_cdc_acm_write(&m_app_cdc_acm, m_nus_data_array, length);

If the sending fails, the handler prints an error message stating that the module is not available.

```
static void nus_data_handler(ble_nus_c_evt_t * p_evt)
{

   if (p_evt->evt_type == BLE_NUS_C_EVT_NUS_TX_EVT)
   {
      //bsp_board_led_invert(LED_BLE_NUS_RX);
      NRF_LOG_DEBUG("Received data from BLE NUS. Writing data on CDC ACM.");
      NRF_LOG_HEXDUMP_DEBUG(p_evt->p_data , p_evt->data_len);
      memcpy(m_nus_data_array, p_evt->p_data, p_evt->data_len);

      // Add endline characters
      uint16_t length = p_evt->data_len;
      if (length + sizeof(ENDLINE_STRING) < BLE_NUS_MAX_DATA_LEN)
      {
         memcpy(m_nus_data_array + length, ENDLINE_STRING, sizeof(ENDLINE_STRING));
         length += sizeof(ENDLINE_STRING);
      }

      // Send data through CDC ACM
      ret_code_t ret = app_usbd_cdc_acm_write(&m_app_cdc_acm,
                                              m_nus_data_array,
                                              length);
      if(ret != NRF_SUCCESS)
      {
         NRF_LOG_INFO("CDC ACM unavailable, data received: %s", m_nus_data_array);}}
```

2. Finding matched BTAs

The function **Calc_MatchedPrct(data_frame1, data_frame2, data_frame3)** will apply the algorithm and calculate the number and percentage of matched BTAs between **data_frame1** and the other two data frames combined. For example, to find the matched data percentage between the main antenna and the other two antennas, the following should be used:

Calc_MatchedPrct (Main, Side, Learning)

The function commences with initializing an empty list—**MatchedBTAs_indices**—that holds the indices of matched BTA and a counter -**dropped**- that holds the number of unmatched BTAs. Data frames 2 and 3 are then merged into one to hasten searching algorithm processing. A loop will be going through all the BTAs in **data_frame1** to match with the same BTA in the merged list. If the BTA exists in the merged list, its index will be added to the **MatchedBTAs_indices**; if not, the counter will be increased to find an unmatched BTA. Eventually, the function prints the number and percentage of matched BTA.

```
def Calc_MatchedPrct (data_frame1, data_frame2, data_frame3):
""":param  data_frame1: Main list
                data_frame2: the first list to be compared
                data_frame3: the second list to be compared

  :return: indices_data_frame1: A list containing the indices of the matched data
points

   MatchedBTAs_indices = [] //An empty list to save matched data indices
   dropped = 0 // A counter for unmatched data
```

```
    Merged_Frames = [data_frame2, data_frame3]
    Merged_data = pd.concat(Merged_Frames, ignore_index=True)
    Merged_data.set_index("codes", inplace=True)
    data_frame1.set_index("codes", inplace=True)
    for i in data_frame1.index:
        M = Merged_data[Merged_data.index == i]
        if len(M) == 0:
            dropped += 1
        else:
            MatchedBTAs_indices.append(i)
    print('Number of matched BTAs is: ' + str(len(data_frame1) - dropped) + ", Percentage
of matched BTAs is :" + str(100 - ((dropped / len(data_frame1)) * 100)) + '%')
    return MatchedBTAs_indices
```

Following is the flow diagram of the algorithm.

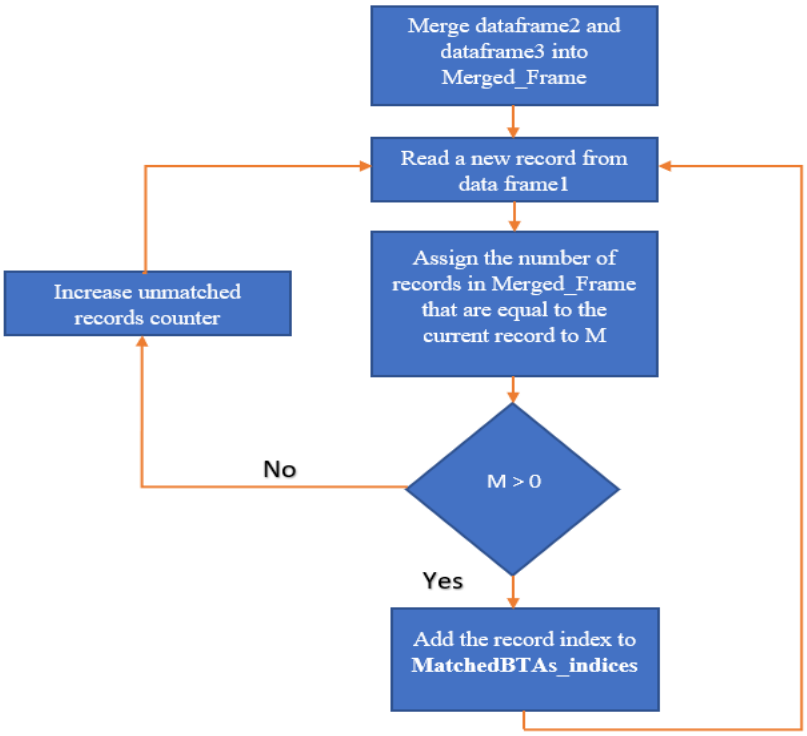


Figure 54. Flow diagram of the Matched BTAs algorithm

3. Parsing Iteris Data

The parsing code initiates by opening and reading the data set. Two lists—Macs and Times—are declared to save LAP and timestamps, respectively. Next, each line of data is searched for the string “2018” and then divided accordingly. To change the data and time formula, this information should be divided into year, month, day, hour, minutes, and seconds; each should be assigned to a variable.

The function **datetime.datetime(int(year), int(Month), int(Day), int(Hour), int(Min), int(Sec))** will change them into a UNIX time stamp.

```
import pandas as pd
import datetime
file = open(r"Data\Iteris_unit.txt", 'r')
lines = file.readlines()
file.close()
Macs = []
Times = []
for line in lines:
    start = line.find("2018")
    if start > 2:
        Mac = line[start - 7:start - 1]
        Time = line[start:start + 19]
        year = Time[0:4]
        Month = Time[5:7]
        Day = Time[8:10]
        Hour = Time[11:13]
        Min = Time[14:16]
```

```
        Sec = Time[17:19]

        dt = datetime.datetime(int(year), int(Month), int(Day), int(Hour), int(Min), int(Sec))
        Macs.append(Mac)
        Times.append(dt.timestamp())

Data_frame = pd.DataFrame(
    {
        'codes': Macs,
        'Time': Times
    }
)
writer = pd.ExcelWriter(r'Data\AVC68_now.xlsx')
Data_frame.to_excel(writer, 'Sheet1', index=False, index_label=False )
writer.save()
```

4. Parsing iVCC sensor's data

Initially, REECE must open a virtual COM "**ttyACM0,**" with the following features:

a. **Baud rate**: 115200

b. **Data bits**: 8

c. **Parity**: None

d. **Stop bits**: 2 bits

e. No flow controls

Once the data is ready for review, function **ser.readline()** will read the virtual COM buffer, and then data will be stored in the variable **read_buffer**. Notably, the virtual COM buffer should be cleared after each reading procedure, using the function **ser.flushInput().**

Software is designed to simultaneously communicate with eight sensors, thus an array **SensorMACs** will store sensors' MAC addresses. Given that a sensor reports a vehicle as stuck, received data will contain the string "**SV**." The program will print "**Stuck Vehicle**" and ignore the vehicle. Given that the sensor reports vehicle arrival and departure, sent data will have the string "**TA**" and "**TD**," respectively. The software will parse received data to find arrival time or departure time by examining all numbers after an "**@**" character, and then storing the characters in the corresponding index of the **TA** or **TD** array. Next, the difference in time **Tocc** will be calculated, where **Tocc = TD – TA**. If both **TA** and **TD** are detected for the same sensor, the classification algorithm will commence. The code for this procedure is provided below.

```
import serial
import sys

distance = 3.5
NumberOfNodes = 4


SensorMACs = ["36408d", "364093","3642fc", "36h4093", "364096", "364099"]
BoolSensor = [False, False, False, False, False, False]
pairs = [False, False, False, False, False, False]
TA = [0, 0, 0, 0, 0, 0]
TD = [0, 0, 0, 0, 0, 0]
Tocc = [0, 0, 0, 0, 0, 0]
Vi = 0.0
Lm = 0.0
loc = 0;


ser = serial.Serial(
    port='/dev/ttyACM0',
    baudrate=115200,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS
)
while 1:
```

```
    ser.flushInput()
    read_buffer = ser.readline()
    sys.stdout.write(read_buffer)
    sys.stdout.flush()
    temp = read_buffer.find("SV")
    if  temp != -1:
        sensor[loc] = False
    else:
        for j in range (NumberOfNodes):
            temp1 = read_buffer.find(SensorMACs[j])
            if  temp1 != -1:
                loc = j
        TA_Loc = read_buffer.find("TA")
        TD_Loc = read_buffer.find("TD")
        if  TA_Loc != -1 :
            BoolSensor[loc] = True
            TA[loc] = float(read_buffer[TA_Loc + 3 : len(read_buffer)])
        elif  TD_Loc != -1 :
            if BoolSensor[loc] == True:
                TD[loc] = float(read_buffer[TD_Loc + 3 : len(read_buffer)])
                Tocc[loc] = TD[loc] - TA[loc]
                sys.stdout.flush()
                BoolSensor[loc] = False
                pairs[loc] = True
                if ((loc + 1) % 2 == 0):
                    if (pairs[loc - 1] == True):
                        ClassifyVehicle(loc)
                        pairs[loc - 1] = False

ser.close()
```

5. Filtering BTAs Algorithm:

```
import pandas as pd
import Process_functions as pf
import statistics

FirstDBLocation = "Deployement\\REECE1.xlsx"
SecondDBLocation = "Deployement\\REECE2.xlsx"
SensorDBLocation = "Deployement\\Sensor.xlsx"

FirstDB = pd.read_excel(FirstDBLocation)
SecondDB = pd.read_excel(SecondDBLocation)
SensorDB = pd.read_excel(SensorDBLocation)
MACsIndices = []
```

```
MACsList = []
FirstDBUnique = FirstDB.codes.unique()
SecondDBUnique = SecondDB.codes.unique()


for MACCounter in range(len(SecondDBUnique)):
    # Find all the instances of SecondDBUnique in FirstDB to check if each code is
existed in both DBs
    TempHolder = pf.look_in_list(FirstDB, SecondDBUnique[MACCounter])
    # check to see if the number of occurrences is only one and TempHolder is 0, so it
doesn't exist
    if isinstance(TempHolder, int) and TempHolder == 0:
            continue
    # Assigning the first occurrence of the element in FirstDB to temp
    Temp = TempHolder[0]
    # Perform a search to find all he occurrences of each element of SecondDB inside
itself
    TempLocation = pf.look_in_list(SecondDB, SecondDBUnique[MACCounter])

    if isinstance(TempLocation, int):
        TempIndex = TempLocation
    else:
        TempIndex = TempLocation[0]

    if SecondDB['Time'][TempIndex] < FirstDB['Time'][Temp]:
        continue
    elif SecondDB['Time'][TempIndex] - FirstDB['Time'][Temp] < 20:
   # MACSIndices is a list of all MACs the were captured by both antennas and in the
wanted order.
        MACs = {}
        MACsIndices.append(TempIndex)
        FirstRx = FirstDB.loc[FirstDB['codes'] == SecondDB['codes'][TempIndex]]
        try:
            ModeFirst = statistics.mode(FirstRx.RSSI)
        except statistics.StatisticsError:
            ModeFirst = statistics.mean(FirstRx.RSSI)
        SecondRx = SecondDB.loc[SecondDB['codes'] == SecondDB['codes'][TempIndex]]
        try:
            ModeSecond = statistics.mode(SecondRx.RSSI)
        except statistics.StatisticsError:
            ModeSecond = statistics.mean(SecondRx.RSSI)
        MACsCodes = {'codes': SecondDB['codes'][TempIndex]}
        MACs.update(MACsCodes)
        MACsTA = {'TA': FirstDB['Time'][Temp]}
        MACs.update(MACsTA)
```

```
        MACsFRx = {'FirstRx': ModeFirst}
        MACs.update(MACsFRx)
        MACsTD = {'TD': SecondRx.Time.values[len(SecondRx.Time.values) - 1]}
        MACs.update(MACsTD)
        MACsSRx = {'SecondRx': ModeSecond}
        MACs.update(MACsSRx)
        MACsList.append(MACs)

MACs_DF = pd.DataFrame(MACsList, columns=['codes', 'TA', 'FirstRx', 'TD',
'SecondRx'])
```

6. Real time assignment algorithm:

```
import time
from MySQLdb import _mysql
from MySQLdb.constants import FIELD_TYPE
import threading
import pandas as pd
import numpy as np
import traceback
import sys


def removeoutliers(dataframe):
    if len(dataframe) >= 3:
        temp = dataframe.copy()
        temp['SNR'] = pd.to_numeric(dataframe['SNR'])
        temp['TIME'] = pd.to_numeric(dataframe['TIME'])
        temp.sort_values(['SNR', 'TIME'], axis=0, ascending=True, inplace=True)
        q1, q3 = np.percentile(temp["SNR"], [25, 75])
        tempDF = dataframe.loc[(temp['SNR'] >= q1) & (temp['SNR'] <= q3)]
        return tempDF
    else:
        return dataframe

def returnRSSI(dataframe):
    try:
        RSSI = dataframe.mode(dropna=True).iloc[0]['SNR']
    except statistics.StatisticsError:
        RSSI = dataframe.mean().iloc[0]['SNR']
    return RSSI


def table_size():
    global DBsize
```

```
    db = _mysql.connect(host="localhost", user="BTS", passwd="temppwd",
db="MACs")



    db.query("select count(*) from Sensors_DB")
    result = db.store_result()
    DBsize = result.fetch_row()[0][0]
    print(DBsize)


def assign(SensorID):
    SNRs = []
    s1 = time.time()
    indices = []
    AssignStart = False
    db = _mysql.connect(host="localhost", user="BTS", passwd="temppwd",
db="MACs")
    query = "select TA from Sensors_DB where Seq =" + str(SensorID)
    try:
        db.query(query)
        result = db.store_result()
        TA = result.fetch_row(maxrows=0)[0][0]
        try:
            query = "select * from MACs_DB where TIME between " + str(int(TA) - 5) + "
and " + str(int(TA))
            db.query(query)
            result = db.store_result()
            LeftMACs = result.fetch_row(maxrows=0, how=1)
        except:
            exc_info = sys.exc_info()
            print(exc_info)
        tempDF = pd.DataFrame.from_records(LeftMACs)
        print("Left MACs are", tempDF)
        query = "select * from MACs_UnDB where TIME between " + str(int(TA) - 5) + "
and " + str(int(TA))
        db.query(query)
        result = db.store_result()
        LeftMACsUn = result.fetch_row(maxrows=0, how=1)
        tempDFUn = pd.DataFrame.from_records(LeftMACsUn)
        if (len(tempDFUn) == 0):
            return
        print("UnLeft MACs are", tempDFUn)
        try:
            query = "select * from MACs2_DB where TIME between " + str(int(TA) + 1) + "
and " + str(int(TA) + 20)
```

```
            db.query(query)
            result = db.store_result()
            RightMACs = result.fetch_row(maxrows=0, how=1)

            tempDF2 = pd.DataFrame.from_records(RightMACs)
            print("Right MAcs are", tempDF2)
        except:
            exc_info = sys.exc_info()
            print(exc_info)

        s3 = time.time()

        for i in range(len(tempDFUn)):
            if (len(tempDF2) > 0):
                try:
                    Unique_leftDF = tempDFUn.copy()
                    Right_MACsDF = tempDF2.copy()
                    Unique_leftDF['TIME'] = pd.to_numeric(tempDFUn['TIME'])
                    Right_MACsDF['TIME'] = pd.to_numeric(tempDF2['TIME'])
                    MutuableRecords = tempDF2.loc[(tempDF2['MAC'] ==
tempDFUn['MAC'][i]) & (
                                (Right_MACsDF['TIME'] - Unique_leftDF['TIME'][i]) > 0)]
                except:
                    exc_info = sys.exc_info()
                    print(exc_info)
                if len(MutuableRecords) != 0:
                    print("MU is", MutuableRecords)
                    FirstSRecords = tempDF.loc[tempDF['MAC'] ==
tempDFUn['MAC'][i].strip()]

                    try:
                        FirstSRecords1 = removeoutliers(FirstSRecords)
                        SNR = returnRSSI(FirstSRecords1)
                        indices.append(i)
                        SNRs.append(int(SNR))
                        AssignStart = True
                    except:
                        exc_info = sys.exc_info()
                        printf(exc_info)
                else:
                    SNRs.append(-100)
                    continue
        if AssignStart:
            query = "select vehicle_Group from Sensors_DB where seq=" + str(SensorID)
            db.query(query)
            result = db.store_result()
```

```
            VehicleGroup = result.fetch_row(maxrows=0, how=1)
            try:
                max_index = SNRs.index(max(SNRs))
            except:
                exc.info = sys.exc_info()
                print(exc_info)

            MAC = tempDFUn.iloc[max_index]['MAC'].decode('utf-8')
            VehicleG = VehicleGroup[0]['vehicle_Group'].decode('utf-8')

            print("MAC is " + MAC + ", " + VehicleG + ", " + TA)

            try:
                query = "insert into MACVEH (MAC,CLASS,TIME) VALUES (\"" + MAC +
"\",\"" + VehicleG + "\",\"" + TA + "\")"
                db.query(query)
            except:
                exc_info = sys.exc_info()
                print(exc_info)

        s4 = time.time()
    except:
        print("Oops")
        return


def main():

    db = _mysql.connect(host="localhost", user="BTS", passwd="temppwd",
db="MACs")
    rows_index = 1
    while (1):

        t2 = threading.Timer(6, table_size)
        t2.start()
        t2.join()
        print("DBsize", DBsize)

        while (rows_index < int(DBsize)):

            query = "select TD from Sensors_DB where Seq =" + str(rows_index)
            db.query(query)
            result = db.store_result()
            num_rows = result.num_rows()
            if num_rows <= 0:
                continue
```

```
            else:
                TD = result.fetch_row(maxrows=0)


                if len(TD) != 0:
                    if ((time.time() - (float(TD[0][0]) + 18000)) > 21.0):
                        t3 = threading.Thread(target=assign, args=[rows_index])
                        t3.start()
                        t3.join()
                        rows_index += 1
                    else:
                        rows_index += 1
                        continue
                print(rows_index)


if __name__ == "__main__":
    main()
```