International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-12, October 2019

An affordable solution to V2V Communication using XBee

Ravishankar Dudhe, Nathan Matias, Ali Mohammad Jowkar, Aaron Anthony Dsouza

Abstract—This paper proposes the creation of a platform-agnostic Vehicle to Vehicle (V2V) communication system that combines a GPS module, XBee-PRO 900HP ® and Raspberry Pi 3 Model B+ Single Board Computer (SBC) to create a low-cost V2V solution. The choice of XBee® enables the use of Digi Internationals' proprietary DigiMesh® protocol, which creates an ad-hoc mesh network. Using the XBee® devices data (longitude, latitude, speed, heading) obtained from the GPS modules can be shared between the vehicles (nodes). For testing purposes, the system was programmed in python with a local SQLite database used to store the obtained data. This approach allows basic V2V communication between devices with different operating systems if a compatible XBee® device is attached to the SBC.

Keywords—Vehicle to Vehicle Communication; XBee®; DigiMesh®; Python.

I. INTRODUCTION

By the year 2050, there are projected to be nearly 3 billion light-duty vehicles on the road up by almost a billion from the year 2020[1]. As the number of vehicles on roads continues to rise, road safety has become a major concern. Globally nearly 20-50 million people are injured per year due to traffic accidents [2]. Statistics reveal that nearly 94% of accidents are caused by human error and are preventable [3]. Vehicle-to-vehicle communication is a technology that aims to improve drivers' awareness and assist them in making the right decisions. Vehicular Ad-Hoc Networks (VANET) are derived from the concept of Mobile Ad-hoc networks (MANET). VANETs build an ad-hoc network between vehicles and road infrastructure. IEEE 802.11p [4], long-term evolution-vehicle (LTE-V) [5], Bluetooth [6] and IEEE 802.15.4/ZigBee [7] are all proposed transmission standards that can facilitate the creation of a VANET. Current V2V and V2X (vehicle to everything) chips use a combination of DSRC/802.11p that operates in the 5.9 GHz band and Wi-Fi (802.11,).

Revised Manuscript Received on October 30, 2019.
***** Correspondence Author
Dr. Ravishankar Dudhe*, Associate Professor Senior Scale, MAHE Dubai
Nathan Matias, undergraduate ECE student at Manipal Academy of Higher Education Dubai.
Ali Mohammad Jowkar, undergraduate mechatronics student at Manipal Academy of Higher Education Dubai.
Aaron Anthony Dsouza, undergraduate ECE student at Manipal Academy of Higher Education Dubai.

Modern V2X chips are designed to take advantage of upcoming 5G technology.

The goal of our research is to develop a V2V solution that is low cost, platform-independent and easily integrated with other Original Equipment Manufacturer (OEM) security systems to enhance vehicle safety. To achieve this, we have used a combination of XBee®, GPS and Raspberry Pi 3B+ SBC.

II. RELATED WORK

Although more advanced Vehicular ad-hoc networks (VANET) are currently available, we tested the viability of using a commercially available RF modules (XBee) to achieve basic V2V communication; sharing location information and vehicle classification (heavy, light. etc.).

[8] proposed a system that used a combination of GPS, Wi-Fi, ZigBee® and 3G broadband units, with an SBC used as the processing unit. Using a combination of parallel CPU and GPU processing they attained real-time V2V data streaming in less than a second and a 4 to 10-time improvement in processing speed compared to conventional methods.

The ZigBee® system used above required two devices a ZigBee® Coordinator and the ZigBee® router. XBee® systems on the other need require only a single XBee® device that acts as a router and coordinator [9]. Our work attempts to test the viability of a system based on only a single XBee® running the DigiMesh® protocol [10].

III. PROPOSED WORK

Fig. 1. Illustrates the embedded systems architecture.

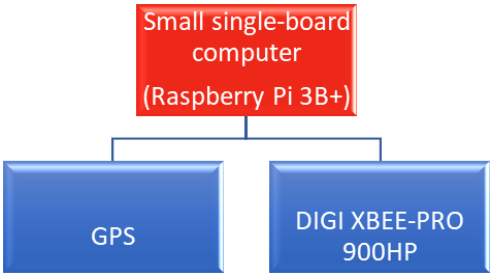

Fig. 1. System Architecture.

The Linux based Raspberry Pi 3B+ utilizes the BCM2837 system-on-chip (SoC) which is a 1.4GHz 64-bit quad-core processor. This processor is capable enough for basic V2V communication.

The 40 GPIO pins and 4 USB 2.0 ports allow multiple peripheral sensors or devices to be connected to the board [11].

The RF module (XBee- PRO 900HP ®) works in the 900MHz band and this offers a range advantage over the 2.4 GHz band of the ZigBee®. XBee® supports data rates of up to 200 Kbps at shorter distances. The choice of the XBee-PRO 900HP® is due to it supporting Digi Internationals proprietary networking topology called DigiMesh® for use in wireless end-point connectivity solutions. It supports three networking topologies such as point-to-point, point-to-multipoint, and mesh networks [12].

The choice of DigiMesh® was due to the following features [13]:

- *Self-healing*: This is the ability of a node to enter and leave the network without causing the entire network to fail.
- *Peer to peer architecture:* Unlike ZigBee® there is no hierarchy All nodes can route data and are interchangeable.
- *Route discovery:* Routes are discovered and created only when needed. DigiMesh® is a Reactive routing protocol like Ad-hoc On-Demand Distance Vector routing (AODV).

The GPS/GNSS module is used for acquiring location information, speed, and heading. The module used in our research was the u-blox NEO-7N® and u-blox NEO-M8N®. These modules are low-cost GPS/GNSS modules, that can give meter level accuracy [14] in an outdoor environment but proved to be inaccurate in a partially enclosed space showing up to a 100% divergence from the actual mean distance. The largest variation was approximately 15 meters which is in line with the ± 15-meter tolerance [14] of most commercial GPS systems.

Table I highlights the issues with relying solely on a GPS based system to acquire distance information. Table II shows that heading can be trusted even with poor reception.

Table- I: Calculated Distance Accuracy

Distance (meters)	Optimum Reception Mean μ (meters)	Poor Reception Mean μ (meters)	Optimum Reception Percent Error (PER)	Poor Reception Percent Error (PER)
5	5.9664	20.8119	19.3291%	316.2396%
10	9.9637	20.7771	0.3623%	107.771%
20	22.3252	32.8963	11.625%	64.481%
30	30.6096	37.2690	2.032%	24.23%
40	40.1891	22.7162	0.472%	43.209%

Table- II: Calculated Angle Accuracy

Angle (◦)	Optimum Reception Mean μ (◦)	Poor Reception Mean μ (◦)	Optimum Reception Percent Error (PER)	Poor Reception Percent Error (PER)
90	79.5424	100.4943	11.6195%	11.6604%
100	103.5728	106.0359	3.57%	6.0359%
360/0	0.5144	16.4945	0.5144%	16.4945%

IV. METHODOLOGY AND PROGRAM FLOW

For our research, only the most basic V2V system was tested that relied on the GPS for location information and the XBee® for transmission and reception. [8] used their ZigBee® modules received signal strength for trilateration to find the distance between vehicles in an indoor environment and combined this system with a GPS for outdoor location tracking. Our focus was to find the reliability of using an XBee ® for transmission and reception of GPS/GNSS data.

The V2V program has been constructed to be modular and scalable. There are few dependencies between the various classes so debugging can be achieved easily. New modules such as voice recognition or GPS navigation can be added to the code base relatively easily. Fig. 2. illustrates the program structure.

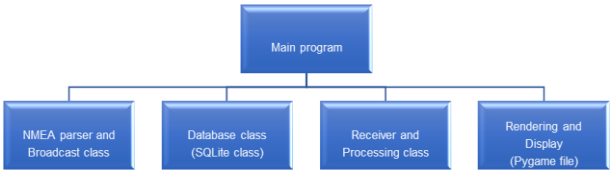

Fig. 2.Program Structure

A. NMEA parser and broadcast class

This class is used for parsing and storing the NMEA sentences in a data container for further processing by the other classes. This class also contains functions for broadcasting the parsed NMEA sentences that can be called by the main program. GPRMC sentences were chosen for our V2V communication since they contain the most relevant data for terrestrial vehicles. The local nodes MAC address, name, timestamp of GPS reception, flag, longitude, latitude, speed and vehicle category are all broadcast to the surrounding nodes (vehicles). Conversion of latitude and longitude from DDDMM.mmmm format to Decimal Degrees format also takes place within this class.

The XBee® library and XCTU software provided by Digi International simplifies the coding processing, enabling speed and range tests to be carried out within a few lines of code. Broadcasting is achieved by using the XBee libraries inbuilt broadcast functions.

B. Database class (SQLite class)

This class handles all the database related functions. The initialization of the database and the connection object are created here. Sorting, deletion, creation, and updating of entries are done using functions from within this class.

The deletion function removes vehicles that are out of the range of the local node and vehicles that have stayed in the database for too long without being updated.

Various test functions also exist in this class to assist in troubleshooting. The database class contains no dependencies from other classes, meaning extra features can be added to the class without interfering with the program flow.

There are ten columns within the V2V database MAC address, name, flag, timestamp, longitude, latitude, speed, distance, angle and vehicle type (light, heavy, etc.).

C. Receiver and Processing class

All reception and processing of broadcast data from remote nodes is achieved using this class. All remote nodes send their data as python strings. The strings must be split into individual data fields and converted into more appropriate data types. The longitude and latitude extracted from the received strings are used to calculate the relative distance and angle between the local node and surrounding nodes. All this information is then stored in a tuple consisting of ten data entries, which is then returned by the main function within this class. The returned value is then stored in the V2V SQLite database.

The distance is calculated using the geopy distance function which returns the geodesic distance.

D. Rendering and Display (Pygame file)

The pygame file contains the functions needed to display the position of the remote nodes relative to the local node as well as to display other GUI information. This file creates a car object each time a new node is added to the database and draws that object onto the screen. The local node will act as the origin or (0,0) in Cartesian coordinates and will be displayed at the center of the window. Heavy and light vehicles are differentiated based on color (red or blue respectively).

E. Main Program

This program checks if a V2V database exists and creates a new one if it has not been built yet. It calls the various function from within the other classes and handles the program flow. The main program oversees exception handling and prevents the classes from returning None type values. Within the main program, function calls to the pygame file, parser class, receiver class, and database class occur.

The script will automatically restart to prevent the software from terminating in case it encounters an exception.

The main program contains code to flush the input buffer of the XBee to ensure that only the latest data is used in calculations.

Table- III: Pseudo Code

1. Create an SQLite database with ten columns and create a connection to the database.
2. Open the XBee serial port and flush the device.
3. Enter the rendering loop (pygame) and start reading the NMEA sentences received from the GPS/GNSS module, as well as read the received data within the XBee buffer.
4. Broadcast the GPS data from the local node to all the remote nodes.
5. Add the GPS data for the local XBee to the SQLite database.
6. Process the remote nodes transmitted data. First split the received string and convert the data back into the appropriate primitives. Calculate the distance between the local and remote nodes using the geodesic distance formula. Calculate the angle as well. Return a tuple with all the information contained within it.
7. Insert the returned tuple into the database with the MAC address of the XBee devices acting as the KEY.
8. Delete data within the database that has existed for over 1 minute or if the distance between the local and remote nodes is above 400 meters.
9. Begin drawing the vehicles to the screen.

V. RESULTS AND DISCUSSION

A. Hardware Setup

We tested this system using two devices. One system was the raspberry pi 3b+ with a 1.4GHz Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit processor and a 1GB LPDDR2 SDRAM. The second system we used for testing was a windows machine with a 2.20 GHz Intel® Core™ i7-8750H Processor and 16GB.

Table IV below lists the hardware devices used in our system.

Table- IV: Hardware

Device	Function	Implementation
XBee-PRO 900HP ®	The XBee embedded modules provide wireless connectivity and facilitate V2V communication	Uses the proprietary Digi mesh algorithm to establish a mesh network between the vehicles.
GPS	Used to obtain the vehicle location information, time and speed.	NMEA sentences based on the NMEA 0183 protocol which contain the vehicles location and speed are received by the GPS module
Raspberry Pi 3B+	Single-board computers used to process incoming and outgoing data.	Utilizes a Linux OS and runs the python interpreter
Raspberry Pi touchscreen	Display the vehicle location	Interaction with the V2V system through a GUI.

B. Software Setup

Our choice of using python was due to the massive number of libraries available and the ease of rapidly prototyping and testing different program structures in this language. A shift to JAVA would significantly improve execution time. The table below list the software that were used during the development stages of this system

Table V list the software that are used to implement our V2V system:

Table- V: Software

Software	Methodology
IDLE IDE (Python with SQLite) [15]	Python is used to parse the GPS data, transmit data, receive data, and store data in a SQLite database. The GUI is also built using python
XCTU [16]	Software developed by Digi International. Used to configure the XBee® devices and set the network protocol.

The system was tested in an outdoor environment with a total of three XBee® modules used simultaneously.

Fig. 3. shows the results of our V2V testing for two vehicles. There are 10 columns in total. The first column holds the MAC addresses of the remote and the local nodes which acts as key or unique identification. The “A” under the flag column indicated that the GPS is successfully receiving location data. The lack of values for distance and angle in the case of the first entry indicates that it is the local node. The final column identifies if a vehicle is a light or heavy vehicle. The SQLite database created by our V2V program updates as soon as an external message is received by the XBee unit. The database has also been successfully tested with a third node.

device_id	flag	lat	lng	speed	distance	angle
0013A200418...	A	25.133422	55.425479833...	0.702	0	0.0
0013A200418...	A	25.133622833...	55.425135166...	1.7945	41.267	302.766

Fig. 3. Sample data from important fields in our SQLite Table

C. GUI

Fig. 4. is an image taken from our GUI . The blue icon with an angle 0.0 and centered in the middle of the screen is the local node. The blue icon with an angle of 1.426 is another vehicle or node. The local node will always remain stationary while the other vehicles will rotate around it, with their distance changing based on their nearness to the local node.

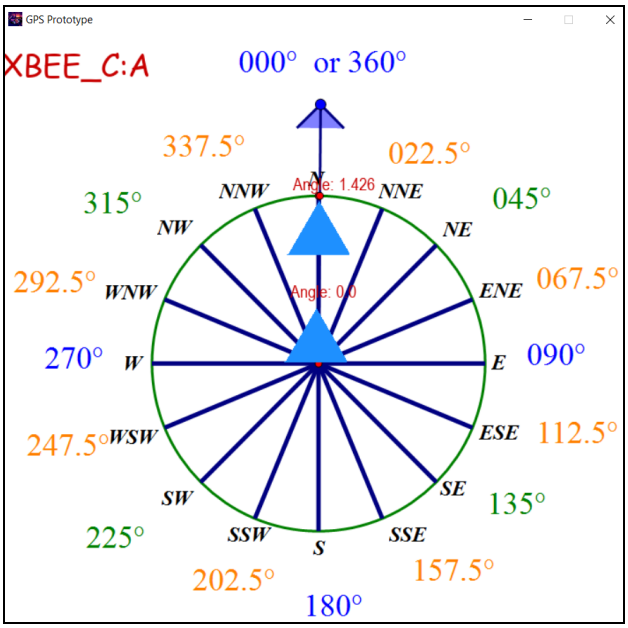

Fig. 4. GUI Output

D. Execution Time

Table VI lists the execution time of our V2V program. The results below were obtained by taking random execution time samples from the program loop.

Table-VI : Execution Time

Windows System (2 nodes) (seconds)	Windows System (3 nodes) (seconds)
0.9433	0.9606
0.9871	0.949
1.0664	1.0199
0.9563	0.9603
0.896	0.8668
0.9454	0.9889

The XBee® establishes connection within a second and as shown in Table VI the difference in execution time between a two-node system and three-node system is insignificant. With proper optimization, we believe the program has the capabilities of running consistently within a second. One improvement to the program would be to accelerate the time taken to parse an NMEA sentence received from the GPS module. A third party NMEA parsing library should help reduce the execution time by a few milliseconds.

A major improvement to the execution time of the code would be migrating the codebase to a bytecode language such as Java or a compiled language such as C++. These languages are significantly faster and can take advantage of the multiple cores present in most modern CPUs. The disadvantage of the C++ approach is ensuring that the code remains platform independent (though compiler dependencies will still exist).

VI. CONCLUSION

As discussed in this paper basic V2V communication can be achieved at a reasonable cost. The XBee® unit is commercially available and easy to setup. Although this system doesn't provide the large data transfer rates offered by more expensive alternatives, it can be installed in any vehicle and supplies the driver with important emergency information such as the distance of vehicles outside the drivers' line of sight. Other uses would be communication with road infrastructure; enabling road speed limits and surface conditions to be broadcast to all vehicles in the vicinity.

ACKNOWLEDGMENT

We would like to thank Manipal Academy of Higher Education Dubai for allowing us to use the campus facilities and premises for the duration of our research.

REFERENCES

1. "What cars will we be driving in 2050? - Fuel Freedom Foundation", Fuel Freedom Foundation, 2019. [Online]. Available: https://www.fuelfreedom.org/cars-in-2050/. [Accessed: 13- Jul- 2019].
2. "Road Safety Facts — Association for Safe International Road Travel", Association for Safe International Road Travel, 2019. [Online]. Available: https://www.asirt.org/safe-travel/road-safety-facts/. [Accessed: 13- Jul- 2019].
3. "Automated Vehicles for Safety", NHTSA, 2019. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety. [Accessed: 13- Jul- 2019].
4. Han, C.; Dianati, M.; Tafazolli, R.; Kernchen, R.; Shen, X. Analytical Study of the IEEE 802.11p MAC Sublayer in Vehicular Networks. IEEE Trans. Intell. Transp. Syst. 2012, 13, 873–886.
5. Molina-Masegosa, R.; Gozalvez, J. LTE-V for Sidelink 5G V2X Vehicular Communications: A New 5G Technology for Short-Range Vehicle-to-Everything Communications.
6. Bluetooth Special Interest Group. Bluetooth Core Specification Versions: 4.0; 4.1; 4.2. June 2010; December 2013; December 2014.
7. IEEE. IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs); Technical Report; IEEE: Piscataway, NJ, USA, 2006.
8. Chia, Goh & Isa, Dino. (2012). Low cost approach to real-time vehicle to vehicle communication using parallel CPU and GPU processing. International Journal of Advanced Computer Science and Applications. 3. 10.14569/IJACSA.2012.031205.
9. Zigbee.org, 2019. [Online]. Available: http://www.zigbee.org/wp-content/uploads/2014/11/docs-05-3474-20-0csg-zigbee-specification.pdf. [Accessed: 13- Jul- 2019].
10. Digi.com, 2019. [Online]. Available: https://www.digi.com/resources/documentation/digidocs/pdfs/90000991.pdf. [Accessed: 13- Jul- 2019].
11. Static.raspberrypi.org, 2019. [Online]. Available: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf. [Accessed: 13- Jul- 2019].
12. Digi.com, 2019. [Online]. Available: https://www.digi.com/resources/documentation/digidocs/pdfs/90002173.pdf. [Accessed: 13- Jul- 2019].
13. [10]Digi.com, 2019. [Online]. Available: https://www.digi.com/resources/documentation/digidocs/pdfs/90001496.pdf. [Accessed: 13- Jul- 2019].
14. U-blox.com, 2019. [Online]. Available: https://www.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_%28UBX-15031086%29.pdf. [Accessed: 13- Jul- 2019].

15. A. Functions et al., "SQLite Python", *SQLite Tutorial*, 2019. [Online]. Available: http://www.sqlitetutorial.net/sqlite-python/. [Accessed: 13- Jul- 2019].
16. "XCTU - Next Gen Configuration Platform for XBee/RF Solutions | Digi International", Digi.com, 2019. [Online]. Available: https://www.digi.com/products/iot-platform/xctu. [Accessed: 13- Jul- 2019].

AUTHORS PROFILE

Dr. Ravishankar S. Dudhe is presently working as an Associate Professor Senior Scale, Program coordinator for the ECE/EEE department and research Co-coordinator in the School of Engineering and Information Technology, Manipal Academy of Higher Education, Dubai. He received his M. Tech as well as Ph.D. in Microelectronics from IIT Bombay. Dr. Ravishankar S. Dudhe has 26 years of teaching and research experience and is the author of multiple textbooks.

Nathan Matias is an undergraduate ECE student at Manipal Academy of Higher Education Dubai. His interests lie in electronics, creative programming and mathematics.

Ali Mohammad Jowkar is an undergraduate mechatronics student at Manipal Academy of Higher Education Dubai. He enjoys working on projects and research that have an impact on society.

Aaron Anthony Dsouza is an undergraduate ECE student at Manipal Academy of Higher Education Dubai. He takes great joy in working with embedded systems.