A WIRELESS SENSOR NETWORK BREADCRUMB TRAIL FOR AN AUTONOMOUS VEHICLE

by

Manu Chaudhary

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2019

ABSTRACT

MANU CHAUDHARY. A Wireless Sensor Network Breadcrumb Trail for an Autonomous Vehicle. (Under the direction of DR. JAMES M. CONRAD)

Research in the area of wireless sensor networks has seen unprecedented growth in the last decade. Human deployable ad-hoc networks are being used to communicate essential information in places where there is no network or the network is destroyed due to a disaster like a flood, hurricane, earthquake and other geologic processes. The research in the field of autonomous vehicle to reach a GPS location along a trail in an unstructured outdoor environment inspired us to make a breadcrumb network. The term breadcrumb is inspired from the well-known fairy tale Hansel and Gretel, in which Hansel uses breadcrumbs dropped on a trail to trace the way back home. The prototype of this work has been made using RaspberryPis, GPS modules, and XBees. The breadcrumb network will help in localizing its nodes and send the GPS coordinates to the autonomous vehicle. This work has analyzed all the complications which might occur in case a breadcrumb GPS does not work.

ACKNOWLEDGEMENTS

Thanks goes first and foremost to my advisor Dr. James M. Conrad for his guidance through my graduate career. Dr. Conrad is tireless and consistent in managing his students, without his help none of the work in thesis would have been possible. I sincerely thank Dr. Conrad for believing in me and guiding me through the work performed in this document. Thanks also goes to Dr. Arun Ravindran and Dr. Aidan Browne for providing support and guidance, and accepting their positions as my committee members. I would like to thank Dr Sam Shue, Dr. Benjamin Rhoades and David Grabowsky for their constant support and guidance. Thanks also goes to all of my colleagues and classmates in the masters program who, through their presence, have helped lighten the load of many long hours. I would like to thank Shantanu Mhapankar, Sunny Arokia Swamy, Raj Gupta, Karim Erian, and Lauren Johnson for being there for help, advice, and moral support throughout my graduate career. I would like to thank god, and to my family for all of their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FFD	Full Function Device
FIRE	Fire Information and Rescue Equipment
HMD	Head Mounted Display
IEEE	Institute of Electrical and Electronics Engineers
IPv4	Internet Protocol Version 4
MAC	Medium Access Control
MUBCS	multiple User breadcrumb System
PDR	Pedestrian Dead Reckoning
RFD	Reduced Function Device
RMG	Relative Measurement Graph
RSSI	Received Signal Strength Indicator
SNR	Signal-to-noise ratio
UART	Universal Asynchronous Receiver/Transmitter
USGS	United States Geological Survey
WSN	Wireless Sensor Network

CHAPTER 1: INTRODUCTION

1.1 Motivation

Research in the area of Wireless Sensor Network(WSN) has shown unprecedented growth in the last decade. These networks are presently being used for industrial applications, environmental monitoring, precision agriculture, security and surveillance. The research in the field of autonomous vehicles in unstructured human uninhabitable environment presents a big challenge. Today, the world is facing an increasing frequency of natural disasters, large scale accidents, and terrorism. In these situations, people may need a network which can be established by the first responder quickly and can be used by an autonomous vehicle to follow the path and reach the destination.

The main motivation for this work comes from the situation where security personnel are doing surveillance in outdoor human uninhabitable place. In order to maintain a communication with the base station they keep on establishing a network. This network can be used to convey their physiological information. The same network can be used by an autonomous vehicle to traverse a path by following the GPS coordinates of the temporary network. In critical situations, an autonomous vehicle can provide even a life saving assistance to the first responder. The main inspiration for using a human deployable network to assist the autonomous vehicle comes from the well-known fairy tale Hansel and Gretel, in which Hansel uses breadcrumbs dropped on a trail to trace the way back home. While critically analyzing the topic, an exhaustive study was carried to find an easy alternative solution for the problem, then establishing a breadcrumb network by the first responder.

In the age of internet, one can trace the path followed by first responder all the way to its destination by sending its GPS coordinates continuously to a online server, and an autonomous vehicle can travel to that point with the help of its telemetry sensors through the unstructured environment. In our study, we found that the solution of using the internet not optimal because, in most of the uninhabitable places, internet facilities are still not established. Even if internet is available, it would not be continuous. With the limited information of the path followed by first responder, it is very difficult for any vehicle to trace along that path with a very high probability that the vehicle will land up in no return situation. Thus, after carrying out an extensive study, one can conclude that there is a need of a Breadcrumb network which can assist the first responder to send GPS coordinates and vital information to the autonomous vehicle. The GPS coordinates communicated by the network to the autonomous vehicle is used by the vehicle to reach the first responder and assist him in his desired mission. For completing up this research, we are assuming that the breadcrumbs placed by the first responder are kept in a trail or vehicle movable locations so that vehicle can trace the breadcrumbs. The whole idea of establishing a breadcrumb was to make a fail-safe network. During our study and implementation, we found various problems associated with the breadcrumb network.

1.2 Problem Statement

The breadcrumb network required to be made should be highly accurate in its working. The first problem was to decide a suitable hardware and make a breadcrumb network. The most basic and biggest problem was that a breadcrumb placed in a highly dense vegetative environment might not give us the GPS reading. In that situation, there will a portion of trail path which will be completely unknown to the autonomous vehicle. The task is to localize the missing breadcrumb. The GPS reading also have slight variations in the values of longitude, latitude and altitude at different times. The another major issue was to approximate the most accurate reading given by the GPS at different times. The breadcrumbs might get dysfunctional due to hardware issue or human intervention. We have to devise a method that each breadcrumb stores the GPS coordinates of a node placed before it and after it, so that autonomous vehicle knows the GPS Coordinates of the next breadcrumb node. In the external environment, it is very tough to know the limit of the range while placing the breadcrumb nodes. If a node goes beyond the communication range of the previously installed node, the communication between the node stops. When a new breadcrumb node is installed, it should be able to identify its neighbouring nodes by discovering the nearby nodes and identify the previously installed node.

1.3 Contribution

The breadcrumb networks is an active area of research with a large number of applications like environmental monitoring, precision agriculture, security and surveillance. The application of a breadcrumb network to provide a traversing path for autonomous vehicle has a numbers problems whose solution still required to found. This research contribution are as follows:

1. To design a prototype of breadcrumb network using RaspberryPis.
2. To implement a Node Discovery, Power-On and Deployed algorithm in breadcrumb nodes.
3. Solving the problem of localizing the breadcrumb not giving the latitude, longitude and altitude and implemented it on RaspberryPis.
4. Installing some breadcrumb nodes in between without the GPS modules and approximating its location with the help of previous and next node installed.

1.4 Organization of Thesis

This thesis is organized into six chapters. Chapter 2 provides a background information about different types of breadcrumb networks and existing research in the usage of breadcrumb networks. Chapter 3 describes the initial hardware and software design. Chapter 4 describes the System Overview, which contains the algorithm designed to make a prototype of a breadcrumb network and steps for configuring the prototype. Chapter 5 summarizes the tests and observations. Chapter 6 presents the conclusions of this research and future work.

CHAPTER 2: BACKGROUND STUDY

To make a breadcrumb network for an autonomous vehicle, an extensive background study was carried out to know all the types of breadcrumb networks used for wireless communication. In this study, a large number of breadcrumb networks were studied to know their merits and demerits. An extensive study was also done on the ZigBee protocol.

2.1 Introduction

A wireless sensor breadcrumb network is made up of a large number of sensor nodes used for establishing a communication path for transmitting essential information. This information is transmitted through a self-configuring network of sensor motes. The other very important problem solved by these motes is the task of localization. The self-localization capability is a highly required characteristic for a wireless sensor networks [2]. In environmental monitoring applications like water quality monitoring or precision agriculture, forest fire observation, the data measured are meaningless without knowing the location. The location estimation enables innumerable applications, like intrusion detection, road traffic observation, surveillance, and exploration using autonomous vehicles. Wireless sensor breadcrumb network's research is a rapidly developing area for a myriad of applications, like collecting physiological data of firefighters working in a building, monitoring pipelines, or monitoring wearable location awareness systems.

The term breadcrumbs is a reference to the well-known fairy tale Hansel and Gretel, in which Hansel uses breadcrumbs to trace the path back home [3]. There are many ways by which the breadcrumb network can be configured. The two major types of breadcrumb networks are human deployable breadcrumb networks and self deployable breadcrumb networks. The majority of breadcrumb systems we came across were using hardware based on 900 MHz and 2.4 GHz frequencies.

2.2 Breadcrumb Networks

Breadcrumb nodes are small and low-priced devices that work as relays; that is, their only aim is to forward packets between nodes. As an example of their use, in an emergency situation, the first responders are provided with a large number of breadcrumb devices with a mobile radio. They will drop these breadcrumb nodes, for establishing a communication channel. Figure 2.1 shows a breadcrumb-based

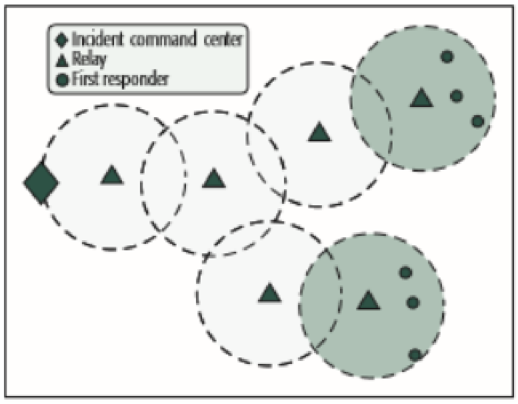


Figure 2.1: A breadcrumb-based network conceptual schema [3]

network. The base station establishes communication with the mobile node with the assistance of relays placed by the first responders, thus expanding the coverage area.The advantages of the breadcrumb networks are as follows:

1. Helps in the creation of multihop network.
2. Establishes a communication channel between the first responders after the relays are placed.
3. Guarantees an efficient and reliable communication
4. An increased coverage area is obtained.

5. The probability that network will get partitioned is reduced.

Extensive research has been conducted to address the deployment decision problem. All the deployment algorithms share several important characteristics. These algorithms monitor the link quality by measuring received signal strength indicator (RSSI), bandwidth or signal-to-noise ratio (SNR) [4, 5]. These measurements can be collected with the help of probe packets artificially infused in the network or by means of control messages acquired directly using the routing protocol. A threshold for deploying a breadcrumb is set. When the link quality goes below a particular threshold, the user should drop a new breadcrumb. For example, the algorithms used in [4] and [5] use a pre-defined triggering threshold for all the applications. An efficient deploying algorithm should be capable of observing changes in the network to fulfill the relays needs.

2.3 Human Deployable Breadcrumb Networks

An extensive survey to look for human deployed breadcrumb networks was carried to know the deployment criteria used by them and the success achieved. The first human deployed breadcrumb network we came across was LifeNet [6]. LifeNet is an ad-hoc network which can be used in emergency situations like fire in the building. It is a static network which is required to be deployed in the building to establish communication between the firefighters and the base station. This network infrastructure can be used for positioning, sensing, and communication. The firefighters use a wearable computing equipment called as head mounted display for receiving navigational information. The sensor system is established on the sensor nodes referred as RELATE Bricks (Figure 2.2), a variant of a RELATE node as introduced by Hazas et al. [7]. The data processing and the communication for Bricks is handled with the help of Particle Computer sensor node using a 8-bit PIC18 microcontroller and 868 MHz TR1001 radio front-end [8]. The Communication and the time synchronization primitives are completed using the AwareCon network stack [9]. The sensor board is

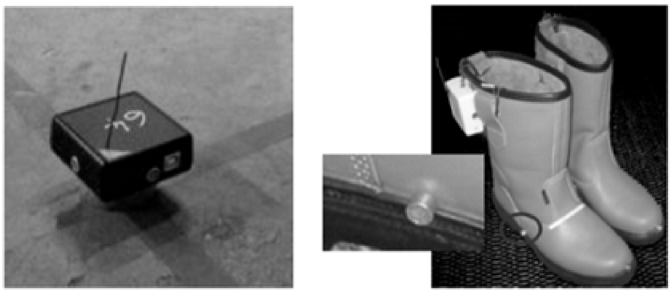


Figure 2.2: RELATE Brick sensor node (left). Boot is attached with RELATE sensor nodes (right) [7]

composed of four 40 kHz ultrasound transducers, power supply, and a temperature sensor. They are using the transducers, which acts as receivers and transmitters for ultrasonic bi-directional sensing. With the help of bidirectional ultrasonic sensing, these devices can calculate their positions with respect to each other. Additionally, they placed two of the ultrasonic sensing nodes on the pair of heavy duty boots. It is used to perform spatial discovery of the nodes and to measure the distance and the angle-of-arrival relative to discovered nodes. The Fire Information and Rescue Equipment (FIRE) project at UC Berkeley's Mechanical Engineering Department aimed to design and implement a decision support tools for the assistance of firefighters in order to raise their safety, effectiveness, and efficiency in emergency situations, mainly in massive complex buildings [10]. To solve the problem of safety and localization of firefighters, they created hardware and software tools. The strategy used by them is to use a wireless sensor network (WSN) called SmokeNet to trace the firefighters working in a large building and supply essential information to all the persons concerned. The key information includes the location of firefighter, fire and physiological data. The interface for these information is the FireEye head-mounted display(HMD) and the electronic Incident Command System (eICS) software [11]. For solving the problem of localization, the building should be pre-installed with beacon motes that continuously broadcast a static information at a 40 Hz frequency. SmokeNet consists of a large number of pre-installed nodes located at each smoke detector and doorway in a building. Firefighters are equipped with MICA2 mote, a head-mounted display (HMD) and a small wearable computer. [12]. In HMD, the locations of the firefighters and messages from the incident command are provided to help them in their task. The research by Souryal et al. is based on the requirement to extend the range of single hop communication [5]. A single hop range is restricted by harsh radio propagation conditions or distance. In this paper, a deployment procedure for the sensor nodes is suggested, taking the consideration of real-time measurements and Physical layer characteristics. They have implemented a prototype based on 900 MHz TinyOS motes [13, 14]. The deployed relays used for range extension have been referred to as breadcrumbs. To access the exact location for the mote placement, a mobile mote continuously performs the link assessment and deployment algorithm, giving a visual indication regarding the new relay deployment position. The proposed relay algorithm adapts to the environment automatically and helps in determining the location to place the new relay. The new relay is placed when the user reaches to the edge of the existing network.

2.4 Self Deployable Breadcrumb Networks

Most of the breadcrumb networks which are used or required are in the case of an emergency situation like fire fighting, natural disasters like earthquake. The need for these networks arises when there is no pre-existing network or the pre-existing network is destroyed. In some cases, like a fire in a building, it is tough for the firefighters to deploy the network by installing the sensor motes at appropriate locations so that their connectivity with the base station outside the building is maintained. To solve this problem of a human requirement in establishing the network, we found the research completed by Liu et al. which suggested a concrete method of establishing a breadcrumb network without any human involvement [15]. The breadcrumbs used here are very small size motes, contained in a dispenser. These breadcrumbs move out of the dispenser automatically as per the need. The dispenser is fitted at the back of firefighters as shown in Figure 2.3. The base station used here is a USB-ported

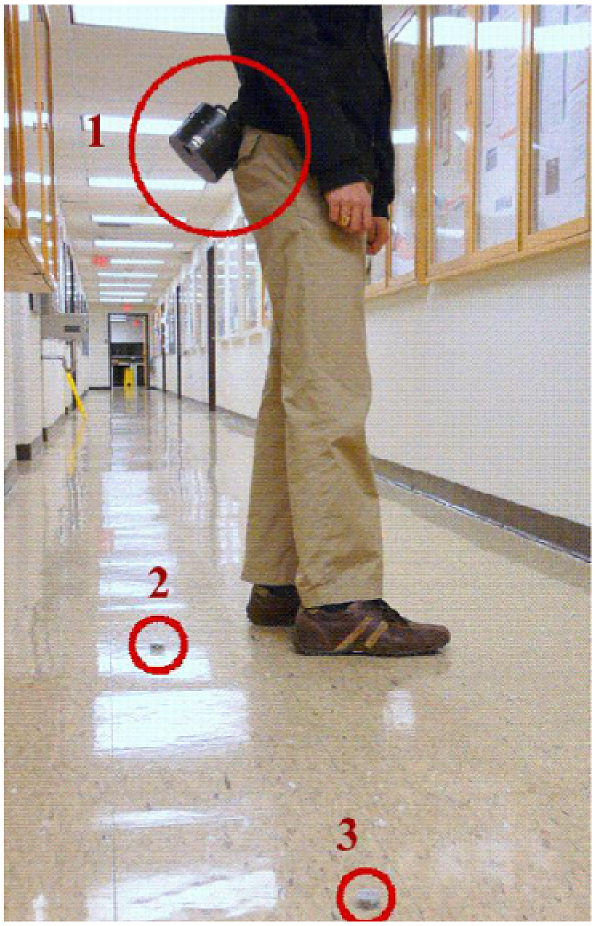


Figure 2.3: Automatic breadcrumb system working, using a dispenser (1) and breadcrumbs (2 and 3) [15]

device attached to a laptop computer. The Texas Instruments Inc. evaluation board SmartRF04EB has been used to program the breadcrumb, base station, dispenser using IAR programming environment [16]. ZigBee networks have been used for implementation because of the advantage of ZigBee over Bluetooth and WiFi in terms of power, bandwidth, and cost [17]. A very important concept used in this network is adaptive power control. It means dynamically increasing the power levels of remote breadcrumbs to increase their chances to reconnect to the network. The paper also addresses the issue of reduction in signal strength due to the difference in height of dispenser and breadcrumb. To resolve the height effect problem, a unique technique called as adaptive threshold adjustment is proposed. All the hardware used by Liu et al. work on 2.4 GHz frequency range [15]. The benefit of using higher frequency is that the antenna size gets reduced. Since antenna size gets reduced, a large number of breadcrumbs can fit inside a dispenser. Their research also addresses the issue of reduction in signal strength due to the difference in the height of dispenser and breadcrumb. It focuses on the requirement that in case of multi-user application, the user with the most number of breadcrumbs in dispenser should disperse its breadcrumbs. Liu et al. work proposes the utility function based algorithm UF, which is as follows: When the request message is broadcasted by the requester, the algorithm gets initiated. After the algorithm gets initiated, neighbors send the information regarding the number of breadcrumbs to the requester. After a predefined time, the value of utility functions is calculated by the requester for each of its neighboring nodes and sends a message to the user with the highest number of breadcrumbs to deploy a new breadcrumb. The UF coordination algorithm maintains connectivity up to 87% greater distances than the baseline greedy coordination algorithm and also maintaining a high packet delivery ratio. By using this network shown in Figure 2.4,

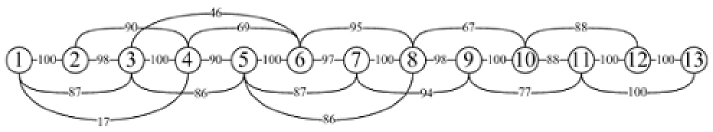


Figure 2.4: Network topology used by Liu et al. [15]

topology, they achieved 200 percent link redundancy at the price of 123 percent node redundancy. Liu et al. further extended their research on breadcrumb networks by using their breadcrumbs for various emergency response applications [15]. A similar method of automatic deployment of the breadcrumb is used by Lai et. al for pipeline monitoring [18]. These pipelines are used for various purposes like water pipelines for carrying water, oil pipelines for carrying petroleum products. The prototype made for monitoring the pipeline from inside is called as TriopusNet. TriopusNet automatically deploys these sensors with the help of a deployment algorithm. The flowing node inside the pipeline latches itself to the inside wall of the pipe by extending its mechanical arms. The TriopusNet (Figure 2.5) system is made up of a breadcrumb network of wireless sensor nodes. This system releases a new node from the repository and executes a node replacement algorithm, whenever a node has a low battery or experiences some fault. The TriopusNet node consists of:

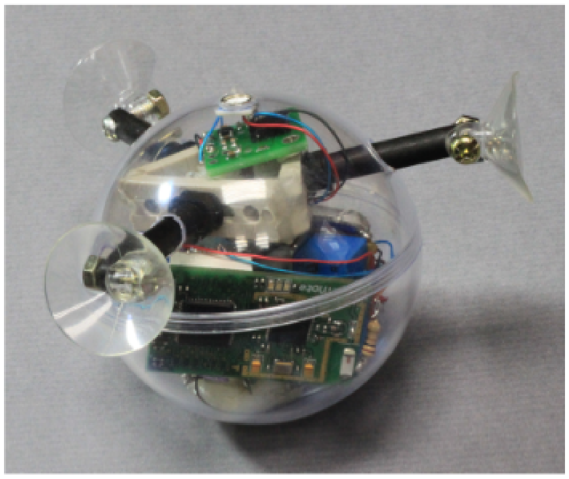


Figure 2.5: The final TriopusNet node prototype [18]

1. A wireless sensor mote termed as Kmote.
2. A motor, which is driving three mechanical arms for latching detaching to the pipes inner surface.
3. A spherical case for waterproofing the nodes
4. Gyroscope sensors and pressure sensors for node localization.

The Kmote circuit used in TriopusNet is a standard TelosB mote, which contains a MSP430 microcontroller and a CC2420 radio stack [19, 20]. It is compatible with Tiny-OS. Because pipes are of variable diameters, the arms stroke length should be large enough to touch the pipe inner surface on all the sides. The Figure 2.5 shows the final prototype of TriopusNet. The main benefit of TriopusNet is that it reduces human effort in maintaining and deploying WSN infrastructure inside pipes. The TriopusNet provides a promising strategy to automate the sensor deployment and replacement inside pipelines. A very unique work which we came across is completed

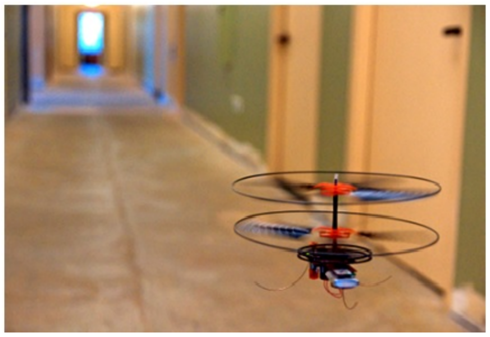


Figure 2.6: SensorFly node flying in a hallway [21]

by Purohit et al. [21]. They proposed and implemented an aerial sensor network called as SensorFly for indoor emergency application. The SensorFly node used in the prototype implementation is shown in Figure 2.6. It is a miniature, low cost sensor network, which has the capability to achieve three-dimensional sensing, self deploy and adapt to network disruptions and node destruction in harsh environments like fire in a building. The processor used in the first version of the sensorFly node was made using ARM7 based LPC2148 [22]. The second and third version of sensorFly nodes incorporates an additional external processor and an AVR AtMega644 for radio function [23]. The navigation sensors used are Accelerometer, Gyroscope, compass ultrasonic ranger, Nanotron nanoLoc RToF ranging, laser ranger and vision. They assessed the platform in a real fire monitoring scenario with the help of CFAST indoor fire simulation models [24]. SensorFly can completely eliminate the cost of building large sensing infrastructure. The Figure 2.7 shows the final prototype of SensorFly implemented and tested successfully.

2.5 Localization using Breadcrumb Networks without Preinstalled Network

Breadcrumb systems (BCS) are very useful in emergency situations to establish an ad-hoc network. Breadcrumb networks can solve the problem of localization, which

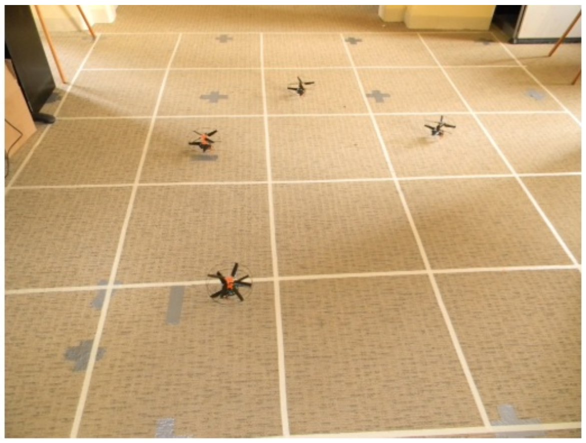


Figure 2.7: Experimental setup established with 4 SensorFly nodes [21]

is highly essential in emergency situations like localizing a firefighter working dousing fire in a building. In the research of Liu et al. a breadcrumb network is implemented for multiple users, which utilizes efficient and automatic coordination among multiple users to accomplish better utilization of limited number of breadcrumbs [25]. This system is termed as MUBCS (multiple user breadcrumb system). This paper proposes a utility function (UF) based algorithm. UF helps in maintaining longer breadcrumb chain lengths with high system reliability and fairness by using suitable benefit and cost functions. UF does not require any prior user mobility models, thus making the design suitable for practical real life applications. For localizing a node without the help of any pre-installed network we came across a study by Wang et al. [26]. In this study, they have suggested an indoor localization technique by using identifiable signatures in the indoor environment. These identifiable signatures may be due to distinct pattern on smartphone accelerometer caused by an elevator, a corridor corner in a building may hear a unique WiFi access points, a particular location may experience uncommon magnetic fluctuations. It is hypothesized that these kinds of signatures can be used for internal landmarks of the building. They named their technique as UnLoc and believe in its real world deployment. The Figure 2.8 provides us the architecture of unLock. The research of Chandra et al. describes a wearable

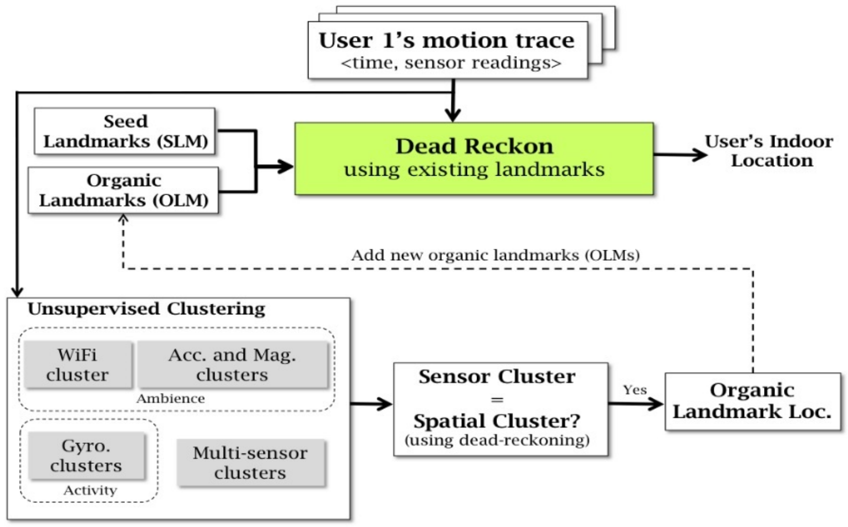


Figure 2.8: UnLoc architecture [26]

location awareness system, which can determine the location of its user within a given building if map of the building is available [27]. The system uses a modest number of ultrasonic range transceivers as sensing elements. A simulation model of the propagation of ultrasonic signals inside the building is used for calculating the expected readings. There have been a lot of research done on combining PDR with wireless sensor networks (WSN) [28–30]. The research of Li et al. [31] pointed out a very important way of exactly locating the location of a breadcrumb. In their research, each firefighter is provided with foot-mounted PDR units and also carries small deployable motes called as breadcrumbs. The breadcrumbs are placed in a dispenser and it is attached to the back of firefighter. The self deployed breadcrumb system is used to transfer data and relative distances among system nodes i.e, breadcrumbs and firefighters. By collecting these location measurements, a localization algorithm is made, which provides the position estimates for both firefighter and breadcrumb. A PDR (Pedestrian Dead Reckoning), which is a mounted on the foot of firefighter is made up of a gyroscope and accelerometer as shown in Figure 2.9.

The accelerometer and gyroscope keep the record of acceleration and angular ve-

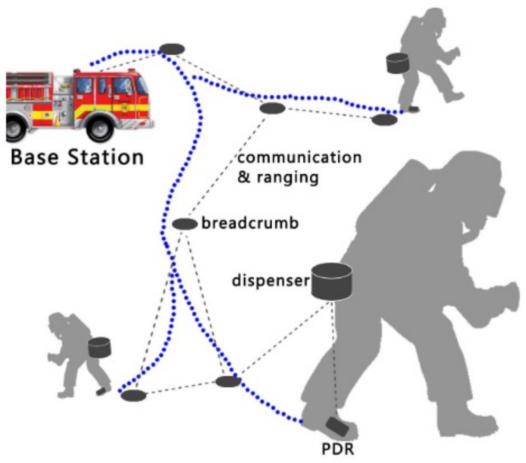


Figure 2.9: Entire System Overview of Wang et al. [26]

locity all time. With the help of inertial dynamic equations and combining it with the zero velocity update error correction, the PDR provides firefighter displacement estimate between two time points. Since the firefighters are moving, the breadcrumbs get dropped from the dispenser, creating a chain to transmit data from firefighter to the base station located outside the building. A breadcrumb estimates the distance between itself and another breadcrumb with the help of receive signal strength indicator (RSSI). By accumulating all these position related measurements, a relative measurement graph (RMG) is constructed, where breadcrumbs and firefighters are nodes of the graph and the measurements made are the edges of the graph. A collaborative localization algorithm is applied on the RMG to estimate the position of breadcrumbs and firefighters. The localization algorithm developed by them is highly scalable. This allows the nodes to join and leave the graph dynamically. The solution provided is unique for localization using breadcrumb networks.

2.6 ZigBee / XBee Personal Area Network PAN

ZigBee is a global standard for making low-cost, low-power and low-data-rate wireless mesh networking built on the IEEE 802.15.4 standard [1]. The below list describes the three types of nodes that make a ZigBee based network. They include a coordinator, end devices, and routers.

1. Coordinator: A coordinator is a full function device and is responsible for overall network management. Each network has only one coordinator. The coordinator performs the following functions:
   - Selects the channel to be used by the network.
   - Starts the network.
   - Assigns how addresses are allocated to nodes or routers.
   - Permits other devices to join or leave the network.
   - Holds a list of neighbors and routers.
   - Transfers application packets.
2. Router: A router is used in tree and mesh topologies to expand network coverage. The function of a router is to find the best route to the destination to transfer message.
3. End Device: The end device can be connected to a router or coordinator. It operates at low duty and power. The end device performs the following functions:

   (a) Joins or leaves a network

   (b) Transfers application packets

Figure 2.10 shows an example of a complete PAN with one central coordinator.

2.6.1 ZigBee Protocol

All the protocols of the ZigBee are shown breiefly in Figure 2.11.

The study related to all the breadcrumb networks and ZigBee protocol used for wireless communication gave a great insight in designing a breadcrumb network for an autonomous vehicle.

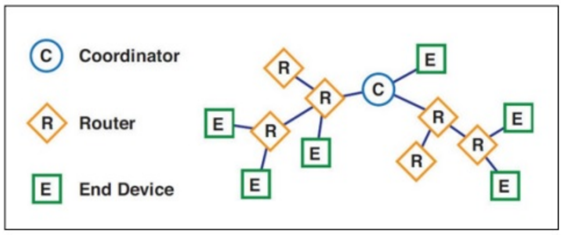
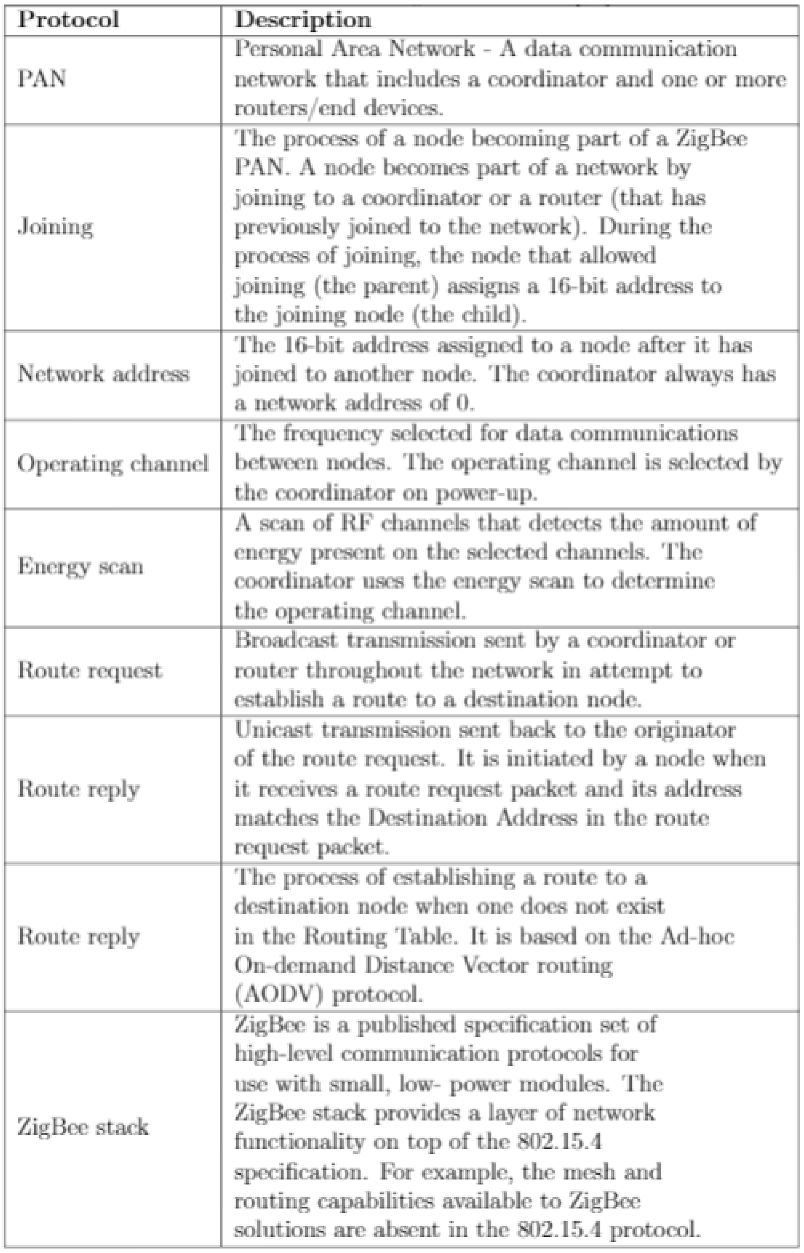


Figure 2.10: ZigBee Personal Area Network [1]



Protocol	Description
PAN	Personal Area Network - A data communication network that includes a coordinator and one or more routers/end devices.
Joining	The process of a node becoming part of a ZigBee PAN. A node becomes part of a network by joining to a coordinator or a router (that has previously joined to the network). During the process of joining, the node that allowed joining (the parent) assigns a 16-bit address to the joining node (the child).
Network address	The 16-bit address assigned to a node after it has joined to another node. The coordinator always has a network address of 0.
Operating channel	The frequency selected for data communications between nodes. The operating channel is selected by the coordinator on power-up.
Energy scan	A scan of RF channels that detects the amount of energy present on the selected channels. The coordinator uses the energy scan to determine the operating channel.
Route request	Broadcast transmission sent by a coordinator or router throughout the network in attempt to establish a route to a destination node.
Route reply	Unicast transmission sent back to the originator of the route request. It is initiated by a node when it receives a route request packet and its address matches the Destination Address in the route request packet.
Route reply	The process of establishing a route to a destination node when one does not exist in the Routing Table. It is based on the Ad-hoc On-demand Distance Vector routing (AODV) protocol.
ZigBee stack	ZigBee is a published specification set of high-level communication protocols for use with small, low- power modules. The ZigBee stack provides a layer of network functionality on top of the 802.15.4 specification. For example, the mesh and routing capabilities available to ZigBee solutions are absent in the 802.15.4 protocol.

Figure 2.11: ZigBee Protocols [1]

CHAPTER 3: HARDWARE AND SOFTWARE SETUP

This chapter covers the hardware setup and the software used to testify the research performed for the thesis. It begins with an overview of the hardware components. It also stepwise explains the connections of the components to make the entire system. It also explains the software components and libraries used to implement the complete breadcrumb network described in Chapter 4.

3.1 Hardware Setup

The main aim of the thesis is to build a prototype of the breadcrumb network, which can collect its GPS data along with the GPS data of previous and next node and transmit it towards Moving Node. The Moving node stores all the data in a MySQL database. The GPS data collected in the coordinator node is used by the autonomous vehicle to traverse the path along the trail. For this thesis, RaspberryPi boards have been used for implementing the breadcrumb network. For making the breadcrumb network, three types of breadcrumb nodes have been used. These nodes are Coordinator Node, Static Breadcrumb Nodes and Moving Node. A Tool is also made using RaspberryPi and Xbee to help in the installation the new Node. The Figure 3.2 shows the prototype of a static breadcrumb node with GPS. The main aim of our breadcrumb network is to provide latitude, longitude, altitude and time information of the previous and next breadcrumb node to the Moving Node node as fast as possible. The nodes should be self configuring,i.e, they should be able to identify their previous and next nodes and also they should be able to tell the user the location of their placement. The placement indication is given by glowing an LED for 15 seconds, based on the signal strength relative to the previous node. The Moving

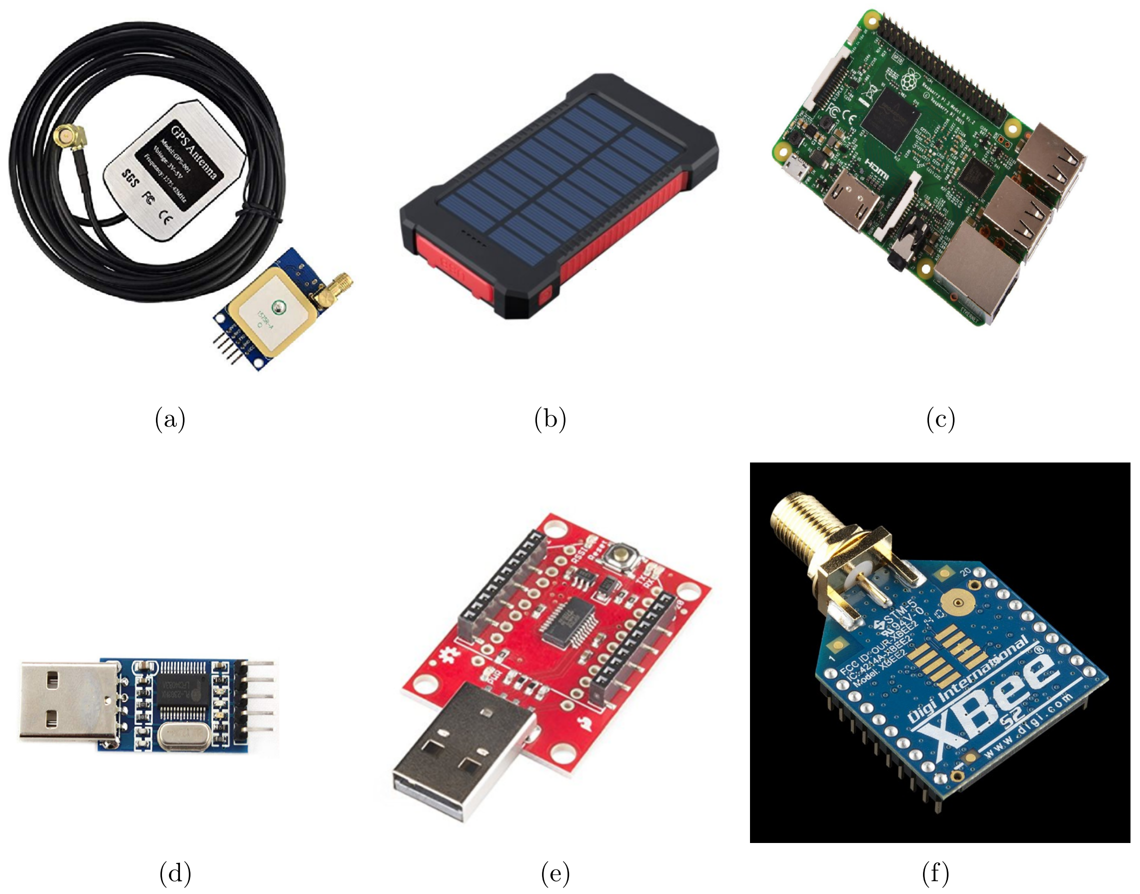


Figure 3.1: Individual Components used to make a breadcrumb node: (a) Gowoops GPS Module with 3m Active Antenna; (b) Dual USB Portable Solar Battery Charger Solar Power Bank; (c) RaspberryPi; (d) Solu USB To RS232 TTL PL2303HX Auto Converter Module; (e) Sparkfun XBee Explorer Dongle; and, (f)XBee S2 Module

node uses these GPS coordinates to reach the destination. The GPS modules used in the breadcrumb node is Gowoops GPS Module with 3m Active Antenna. The GPS modules are connected to the RaspberryPi using Solu USB To RS232 TTL PL2303HX Auto Converter Module Converter Adapter. To transfer the GPS coordinate information of the previous and next node to the Moving Node, S2 Series XBee modules have been used. A SparkFun XBee Explorer Dongle is used to connect XBee S2 to USB port of RaspberryPi. Figure 3.1 shows all the individual components used for making a breadcrumb node. The RaspberryPi board is a development board built on the Broadcom BCM2837B0 processor. Some specifications and features of the board

Table 3.1: Characteristics of RaspberryPi [32]

Processor	BCM 2835 ARM11 700 MHz
Board Power Draw	600 mA
Video Ouput	HDMI
GPIO Pins	40
SD Card	MicroSD
Ethernet Port for Internet Access	10/100 MB
Operating System	Rapbian

are listed in Table 3.1. All the radio communication between the breadcrumb nodes

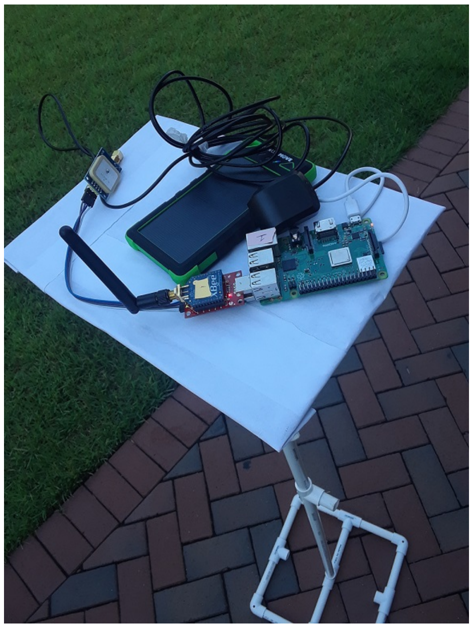


Figure 3.2: A Breadcrumb Node

is handled by XBee S2 radio modules of Digi International. The radios provides the 802.15.4 protocol on a simple to implement and low-powered device. Table 3.2 shows some of the important data-sheet specifications of the XBee S2 modules.

3.2 Interfacing with XBee 802.15.4 Radios

All data to be sent using the XBee is sent through UART interface connection. Devices that have a UART interface can connect directly to the pins of the RF module. The Figure 3.2 shows the Interfacing of the RaspberryPi to the XBee module using XBee Explorer Dongle. To send data with the XBee module, it first needs to be

Table 3.2: Characteristics of XBee S2 [1]

Indoor/Urban Range	up to 133 ft. (40 m)
Outdoor RF line-of-sight Range	up to 400 ft. (120 m)
Transmit Power Output (software selectable)	2mW (+3dBm)
RF Data Rate	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 230400 bps (non-standard baud rates also supported)
Receiver Sensitivity	-95 dBm
Supply Voltage	2.8 to 3.4 V
Operating Current (Transmit)	40mA (@ 3.3 V)
Power-down Current	<1 uA @ 25 C
Operating Frequency Band	ISM 2.4 GHz
Dimensions	0.960 inch x 1.087 inch(2.438cm x 2.761cm)
Operating Temperature	40 to 85 Celsius(industrial)
Antenna Options	Integrated Whip, Chip, RPSMA, or U.FL Connector
Supported Network Topologies	Point-to-point, Point-to-multipoint, Peer-to-peer & Mesh
Number of Channels (software selectable)	16 Direct Sequence Channels
Addressing Options	PAN ID and Addresses, Cluster IDs and Endpoints (optional)

transferred to the XBee module over serial via UART. Once received by the XBee, the data is queued in the module Data-In (DI) buffer. The buffer acts as a first in, first out (FIFO) queue, delivering data as the CSMA/CA algorithm. The device will also keep data queued as it is receiving transmissions. If the DI buffer is full, all new data is discarded. Similarly, all radio frequency (RF) data received is placed into the modules Data-Out buffer until the device receives it over UART. Any new transmissions received when the DO buffer is full are discarded. Both the DI and DO buffers are 100 bytes in size. The baud rate chosen for UART communications in this thesis is 57600 baud [33]. Figure 3.3 shows the data flow between two XBees connected to micro-controllers.

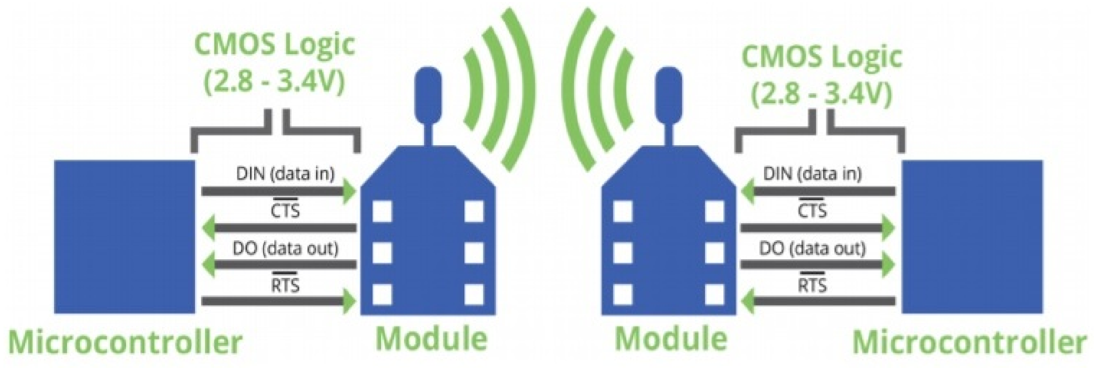


Figure 3.3: Uart Data Flow [1]

3.2.1 Serial Data

A device sends data to the XBee S2 RF Module's UART through pin 4 as an asynchronous serial signal. The signal should be high, when the device is not transmitting. For successful serial communication, the UART of both devices (the microcontroller and the XBee S2) should have compatible settings for the baud rate, parity, start bits, stop bits, and data bits. Each data byte consists of a 8 data bits, start bit (low) and a stop bit (high). Figure 3.4 shows the serial bit pattern of data passing through the device. Figure 3.4 shows UART data packet 0x1F (decimal number 31) as transmitted through the device [1].

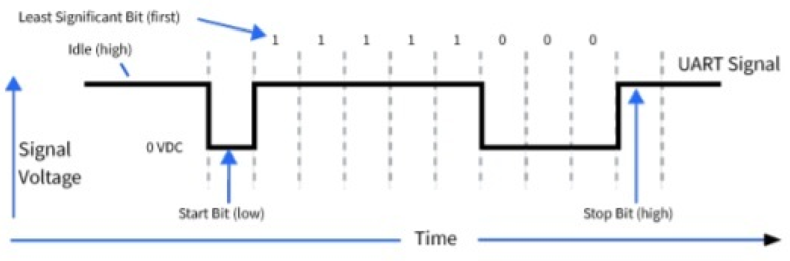


Figure 3.4: Data Byte [1]

3.2.2 Serial Buffers

The XBee S2 modules maintains internal buffers to collect serial and RF data that it receives. The serial receiver buffer collects incoming serial characters and holds them until the device can process them. The serial transmit transmits the data out the serial port until it receives via the RF link. Figure 3.5 shows the process of device buffers collecting received serial data. The operating mode of an XBee radio module

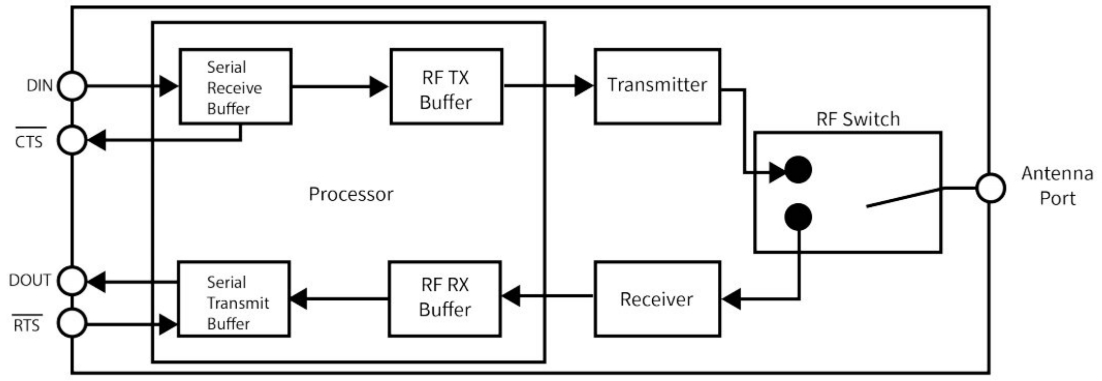


Figure 3.5: XBee Internal block diagram [1]

establishes the way a user or any micro-controller attached to the XBee communicates with the module through the Universal Asynchronous Receiver/Transmitter (UART) or serial interface.

3.2.3 Modes of Operation in XBee

Radio modules can work in three different operating modes, depending on its configuration and firmware:

- Application Transparent (AT) operating mode
- API operating mode
- API escaped operating mode

3.2.3.1 AT Mode

AT Mode is called as the Application Transparent Mode. The protocol link established between the two XBees is transparent and it appears to be a direct serial link between the communicating nodes. Even though the data transmission and reception between the nodes is unprocessed, still the message passed between nodes is encapsulated with needed information such as address and error checking bytes [34].

3.2.3.2 API Mode

API mode is an alternative to transparent AT mode. API operating mode requires that the communication with the module is done through API frames. In API Mode, the data is packaged with other information like checksum, destination address and the type of packet. The receiving node also accepts the data with information such as signal strength, type of packet, checksum and source address. The advantage is that the user can build a packet that includes important data, such as destination address, and the receiving node can pull the packet information such as source address of the data. Although, the API Mode is more programming intensive, but it gives greater flexibility and increased reliability [34]. With API operating mode, one can:

- Configure the XBee module itself
- Configure remote modules in the network.
- Manage data transmission to multiple destinations.
- Receive success/failure status of each transmitted RF packet.
- Identify the source address of each received packet.

Depending upon the value of AP parameter, the radio module operates in one of two modes: API (AP=1) or API escaped (AP=2) operating mode.

Table 3.3: Characteristics of a Frame [34]

Field	Description
Start delimeter	The first byte of a frame consisting of a special sequence of bits which indicate the beginning of a data frame. Its value is always 0x7E. This allows for easy detection of a new incoming frame.
Length	Specifies the total number of bytes included in the frame data field. Its two-byte value excludes the start delimiter, the length, and the checksum.
Frame data	Composed by the API identifier and the API identifier-specific data. The content of the specific data depends on the API identifier (also called API frame type).
Checksum	The last byte of the frame. It helps test data integrity and is calculated by taking the hash sum of all the API frame bytes that came before it, excluding the first three bytes (start delimiter and length).

3.2.3.3 API escaped operating mode

This mode increases the reliability of transmission by preventing the conflicts using the special characters like the start-of-frame byte (0x7E). Since 0x7E appears at the start of each API packet, a module knows that a new packet has begin if 0x7E is received at any time [34].

3.2.4 API Frames

The structure of API frame in API non-escaped (API=1) and API escaped (API=2) is shown in Figure 3.6 and Figure 3.7 respectively.

If the module is operating in API escaped operating mode, some bytes in the Length, Frame data, and Checksum frame fields may need to be escaped. In ap-

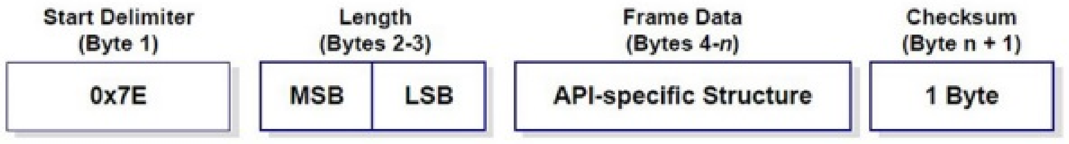


Figure 3.6: API 1 Frame Structure [34]

plication programming interface (API) mode for communicating data, all messages

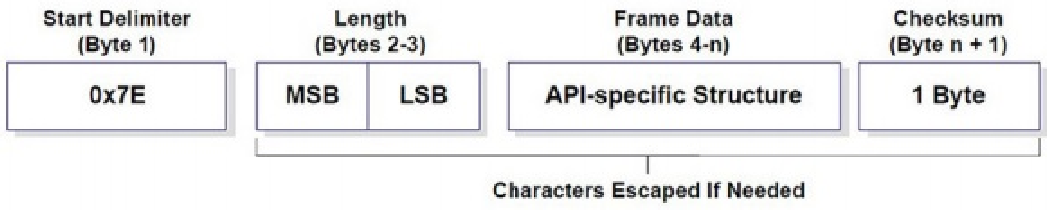


Figure 3.7: API 2 Frame Structure [34]

intended to be sent to other nodes must be packaged into a frame format before being sent to the XBee module. All messages received from the module are also packaged in this format.

3.3 Software Used

3.3.1 XCTU Software

XCTU is a free software used for configuring Digi RF modules. It includes new tools that make it easy to set-up, configure and test XBee RF modules [34]. XCTU software has been used to configure the XBee S2 in API escaped (API=2) operating mode. XCTU software is installed in PC. XCTU cannot be installed in RaspberryPi. After configuring the XBee S2 in API=2 mode, the XBee can be connected to RaspberryPi.

3.3.2 Python XBee Library and Python GPS library

The software programming language used in the making of the breadcrumb network is done using Python. GPS and XBee library of Python helped in making the breadcrumb network.

3.3.3 MySQL Server and PhpMyAdmin

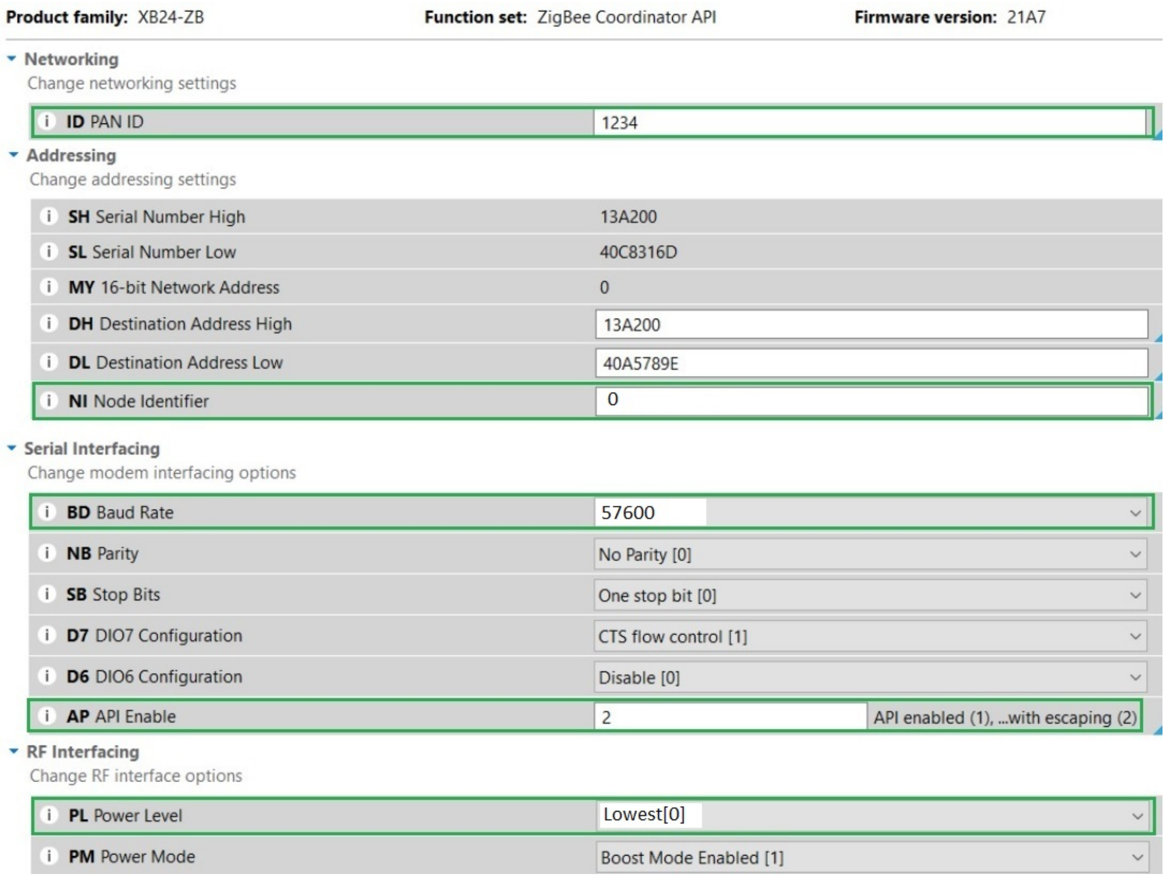
To store the GPS data arriving from all the router nodes at the Moving node, the MySQL database is installed in coordinator RaspberryPi. The data reaching at the coordinator node is stored in tables inside the MySQL database.

CHAPTER 4: SYSTEM OVERVIEW

This chapter provides a detailed overview of the full software architecture implementing the hardware and software of chapter 3. The breadcrumb network consists of RaspberryPis, GPS modules and XBees. The main purpose of the breadcrumb network is that each node collects the GPS Coordinates of the previous and next node and send the GPS data periodically to the Moving Node. The collected GPS coordinates are used by the autonomous vehicle to traverse its path through a trail. The purpose of the breadcrumb network is to provide the GPS coordinates to the autonomous vehicle so that the autonomous vehicle can follow that path and reach its destination.

4.1 XCTU Software

For proving the concept of a breadcrumb network, a breadcrumb network prototype is made using eight breadcrumbs. The first step for making a breadcrumb network is to configure the XBees in API 2 mode. The API mode is essential for making a breadcrumb network. In API mode, the data is first placed into a frame and then transmitted. For making the network, an XBee configured in Coordinator API 2 mode is attached to a USB port of a RaspberryPi and other five have to be configured into Router API 2 mode. For initial configuration of the XBee, XCTU software is required. XCTU software is the graphical software package, which is used for configuring the different parameters in an XBee. Most of the parameters are kept as default. The parameters which are essentially required to be configured are ID PAN ID, NI Node Identifier, BD Baud Rate, PL Power, BH Broadcast Radius, NH Maximum Hops and AP API Enable of the XBee. Some of these parameters are shown in Figure 4.1 inside the green squares. The PAN ID should be the same for both the Coordinator and the Routers. With the help of PAN ID, the system identifies all XBees belonging to the same network. Since the XBee are operated in API 2 mode, the destination address of


Product family: XB24-ZB **Function set:** ZigBee Coordinator API **Firmware version:** 21A7

Networking
Change networking settings

ID PAN ID	1234

Addressing
Change addressing settings

SH Serial Number High	13A200
SL Serial Number Low	40C8316D
MY 16-bit Network Address	0
DH Destination Address High	13A200
DL Destination Address Low	40A5789E
NI Node Identifier	0

Serial Interfacing
Change modem interfacing options

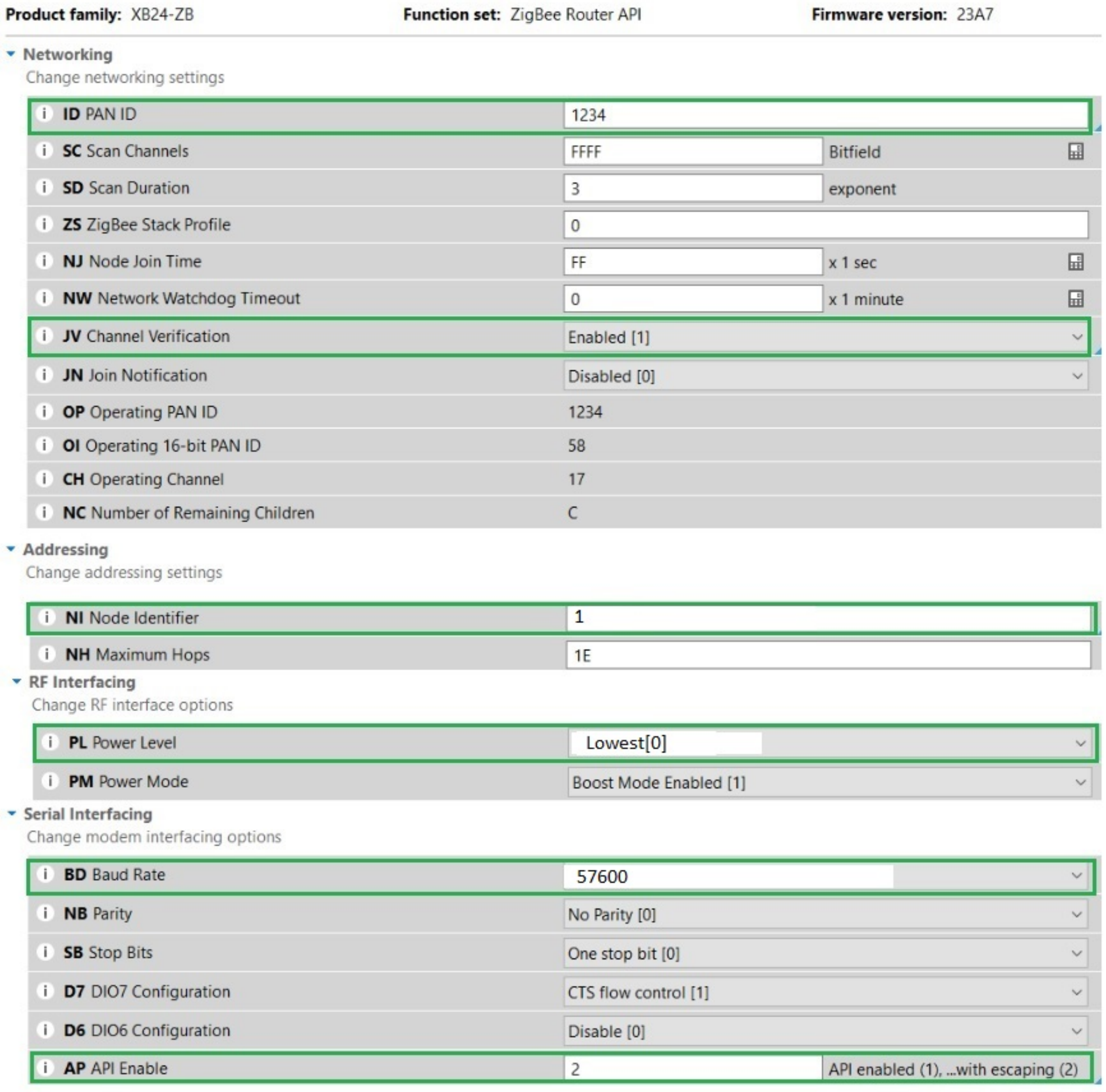
BD Baud Rate	57600
NB Parity	No Parity [0]
SB Stop Bits	One stop bit [0]
D7 DIO7 Configuration	CTS flow control [1]
D6 DIO6 Configuration	Disable [0]
AP API Enable	2 API enabled (1), ...with escaping (2)

RF Interfacing
Change RF interface options

PL Power Level	Lowest[0]
PM Power Mode	Boost Mode Enabled [1]

Figure 4.1: Configuration settings of Coordinator

the Router does not possess any importance. The baud rate used in XBees is 57600. The reason for using API 2 mode is that the API escaped (API 2) operation involves escaping character sequences in an API frame to improve reliability, especially in noisy RF environments. The Routers are also required to be configured using XCTU software. All the Routers firmware is first updated with API 2 Mode. When a Router is operated in API 2 mode, it uses a destination address of 0, irrespective of the value of destination address placed. After installing the firmware, the Router is put to the default settings. Most of the settings will be kept as default except a few as shown in Figure 4.2 inside the green color squares. After configuring the XBees, they are



Product family: XB24-ZB **Function set:** ZigBee Router API **Firmware version:** 23A7

Networking
Change networking settings

ID PAN ID	1234	
SC Scan Channels	FFFF	Bitfield
SD Scan Duration	3	exponent
ZS ZigBee Stack Profile	0	
NJ Node Join Time	FF	x 1 sec
NW Network Watchdog Timeout	0	x 1 minute
JV Channel Verification	Enabled [1]	
JN Join Notification	Disabled [0]	
OP Operating PAN ID	1234	
OI Operating 16-bit PAN ID	58	
CH Operating Channel	17	
NC Number of Remaining Children	C	

Addressing
Change addressing settings

NI Node Identifier	1
NH Maximum Hops	1E

RF Interfacing
Change RF interface options

PL Power Level	Lowest[0]
PM Power Mode	Boost Mode Enabled [1]

Serial Interfacing
Change modem interfacing options

BD Baud Rate	57600	
NB Parity	No Parity [0]	
SB Stop Bits	One stop bit [0]	
D7 DIO7 Configuration	CTS flow control [1]	
D6 DIO6 Configuration	Disable [0]	
AP API Enable	2	API enabled (1), ...with escaping (2)

Figure 4.2: Configuration settings of a Router

attached to the USB port of RaspberryPis. The XBee pin length is 0.1 inch. It cannot be connected to the breadboard, to further connect it to the GPIO pins of RaspberryPi.

4.2 Use of GPS Module

The GPS connection to a RaspberryPi is made using USB to RS232 TTL PL2303HX Auto Converter Module. This module helps in connecting the GPS to the USB port of the RaspberryPi. The connection diagram is shown in Figure 4.3.

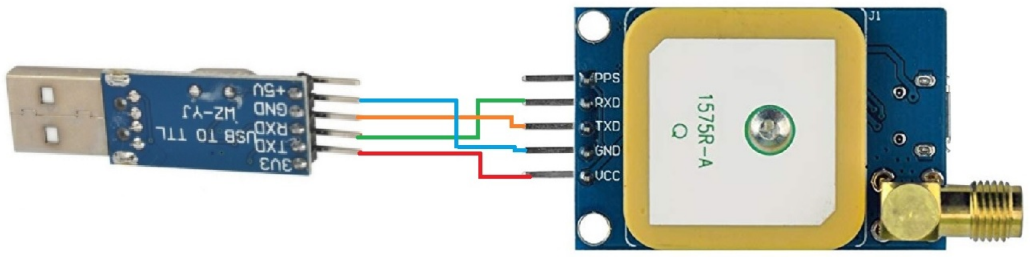
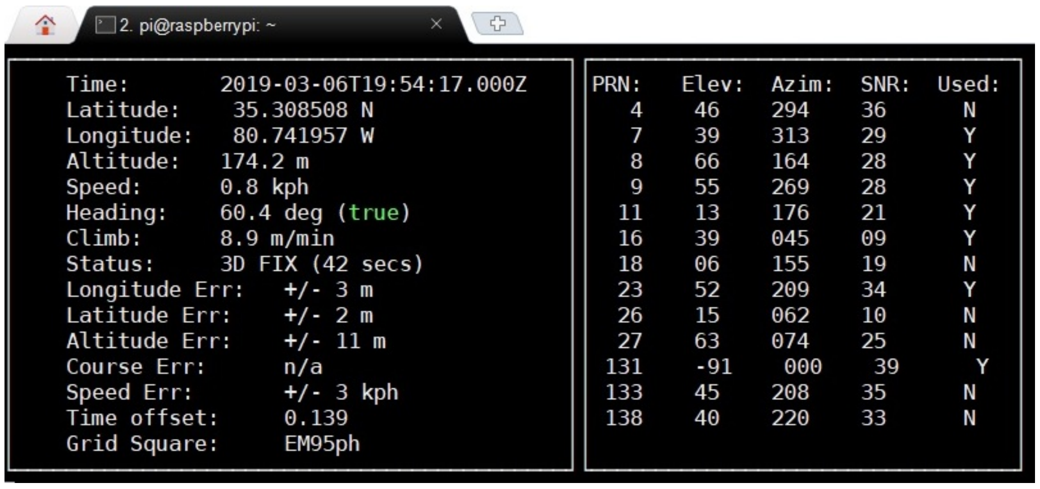


Figure 4.3: GPS connection with USB To RS232 TTL PL2303HX Auto Converter Module

The GPS device data is received in the RaspberryPi with the help of GPSD. GPSD is a daemon that receives data from a GPS receiver and provides the data back to multiple applications. When GPSD starts, it is able to open up the proper socket and when GPS is locked, some data is received from the GPS module. The data obtained by running the GPSD on the terminal of RaspberryPi is shown in Figure 4.4.



```
Time:         2019-03-06T19:54:17.000Z      PRN:   Elev:  Azim:  SNR:  Used:
Latitude:      35.308508 N                    4     46     294    36      N
Longitude:     80.741957 W                    7     39     313    29      Y
Altitude:     174.2 m                         8     66     164    28      Y
Speed:        0.8 kph                         9     55     269    28      Y
Heading:      60.4 deg (true)                11     13     176    21      Y
Climb:        8.9 m/min                      16     39     045    09      Y
Status:       3D FIX (42 secs)               18     06     155    19      N
Longitude Err:     +/- 3 m                   23     52     209    34      Y
Latitude Err:      +/- 2 m                   26     15     062    10      N
Altitude Err:      +/- 11 m                  27     63     074    25      N
Course Err:        n/a                      131    -91      000    39      Y
Speed Err:         +/- 3 kph                133     45     208    35      N
Time offset:       0.139                    138     40     220    33      N
Grid Square:       EM95ph
```

Figure 4.4: Result obtained by Runnning GPSD on the terminal

For acquiring the appropriate data from the raw GPS data, Python3 GPS library is used.

4.3 Block Diagram of the Prototype of a Breadcrumb Network

In this work, eight breadcrumb nodes are used. Six breadcrumb nodes are static Nodes, including the Coordinator Node. Each breadcrumb node is composed of RaspberryPi, GPS and an XBee. There is one Moving Node, which acts like a autonomous vehicle. A Moving Vehicle is composed of a RaspberryPi and a XBee module. One Node Deployment tool is also made using RaspberryPi and XBee to assist in the deployment of the new node. The basic prototype of the breadcrumb network is shown in Figure 4.5 and explained in upcoming sections.

4.4 Different Types of Nodes Working in Breadcrumb Network

In order to make a breadcrumb network, four types of breadcrumb nodes are used. The first node is Coordinator. The coordinator node is the main controller node in XBee network. This node is build using a RaspberryPi, GPS module, and an XBee S2 module. This node functionality is to collect the GPS data and forward it to its next node. The second type of breadcrumb nodes used are Static Router Nodes. The Static Router Nodes are also build using RaspberryPi, GPS and XBee. All the Static Router Nodes have exactly the same program working inside them. The third type of node do not have a GPS connected to it. A single node in our breadcrumb network prototype doesnot have GPS connected. This node will try to approximate its latitude and longitude using the GPS data collected from its previous and next node. The fourth type of node is a Moving Node, which acts like a future autonomous vehicle. A Node Deployment tool is made using a RaspberryPI and XBee, which assist in the deployment of the new node at the farthest possible distance. The Algorithms working in all the types of nodes is explained below:

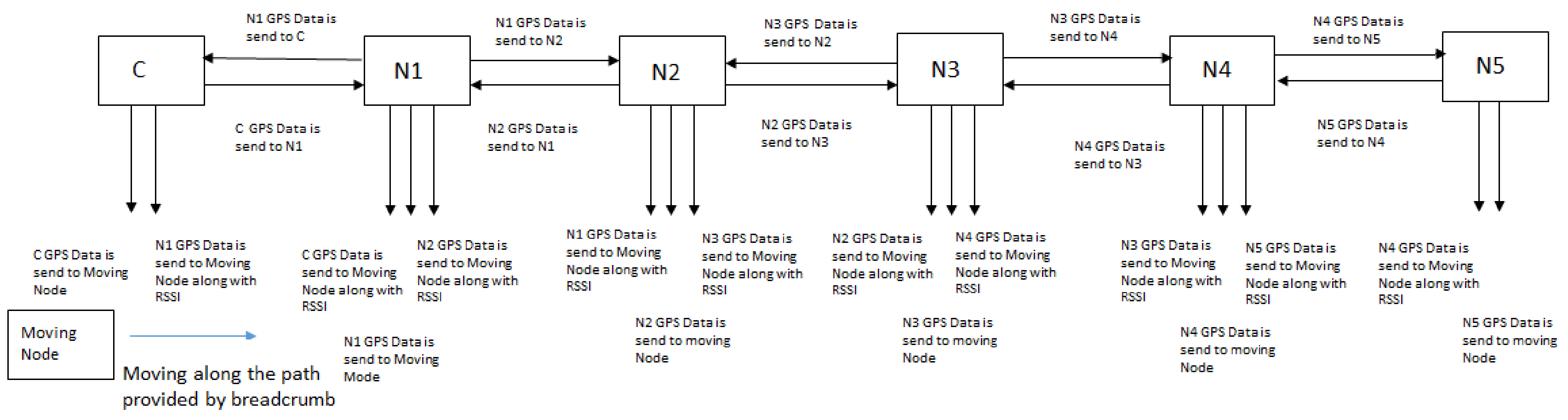

N1 GPS Data is send to C
N1 GPS Data is send to N2
N3 GPS Data is send to N2
N3 GPS Data is send to N4
N4 GPS Data is send to N5
C
N1
N2
N3
N4
N5
C GPS Data is send to N1
N2 GPS Data is send to N1
N2 GPS Data is send to N3
N4 GPS Data is send to N3
N5 GPS Data is send to N4
C GPS Data is send to Moving Node
N1 GPS Data is send to Moving Node along with RSSI
C GPS Data is send to Moving Node along with RSSI
N2 GPS Data is send to Moving Node along with RSSI
N1 GPS Data is send to Moving Node along with RSSI
N3 GPS Data is send to Moving Node along with RSSI
N2 GPS Data is send to Moving Node along with RSSI
N4 GPS Data is send to Moving Node along with RSSI
N3 GPS Data is send to Moving Node along with RSSI
N5 GPS Data is send to Moving Node along with RSSI
N4 GPS Data is send to Moving Node along with RSSI
Moving Node
Moving along the path provided by breadcrumb
N1 GPS Data is send to Moving Mode
N2 GPS Data is send to moving Node
N3 GPS Data is send to moving Node
N4 GPS Data is send to moving Node
N5 GPS Data is send to moving Node

Figure 4.5: Breadcrumb Network Prototype

4.4.1 Algorithm Implemented in the Static Router Nodes

4.4.1.1 Node Discovery

The static node is a self configuring node. When the Node is Power On, a Node Discovery Algorithm begins. During the Node Discovery, the node tries to search for the Node IDs and MAC Addresses of all the XBees in its vicinity. The Node discovery algorithm works for 15 seconds. During the node discovery, all the MAC Addresses and Node IDs are stored in a textfile, which acts like a Node table. The Node IDs are put in XBees while configuring them. The Node IDs of all the Router XBees are put in increasing order starting from one. The NodeID of the moving node is negative one. After scanning the nearby nodes Node IDs, the node knows the previously installed node from the highest Node IDs received during scanning. The algorithm increases the highest Node ID received by one to get the Node ID of the current node, and further increases it by one to get node of the next node. Algorithm 1 describes the Node Discovery Algorithm working in a Static Breadcrumb Node. The Coordinator Node coordinates the entire system. Each network of XBees contains only one coordinator node.

Algorithm 1 Node Discovery in Static Nodes

```
1: The node is Powered ON.
2: Open XBee Port.
3: The Node sends a broadcast signal to nearby nodes for their MAC Addresses and
   Node IDs.
4: The nearby Nodes send their Node IDs and MAC Address and these are stored
   in a textfile.
5: Node Discovery is done for only 15 seconds. In 15 seconds, all the nearby nodes
   send their Node IDs and MAC Address.
6: Close XBee Port.
```

4.4.1.2 Installation Algorithm

Installation Algorithm helps in installing the breadcrumb nodes to maximum separable distances to increase the coverage area. The good range for communication to work between the nodes is determined by the RSSI values. If the area is a plane without much trees, vegetation and concrete structues, the signal strength decreases slowly and the distance covered is large. A Node Deployment tool is used to identify the newly installed node when powered on. It will send "Hello Messages" to the newly powered on node, which gets the RSSI value by reading the "DB" parameter from the API2 frame. As the distance increases, the signal strength decreases. When the RSSI value reaches above 70, the node gives an indication to install it by glowing an externally attached LED for 15 seconds. Algorithm 2 describes the Installation Algorithm working in Static Breadcrumb Node.

Algorithm 2 Installation Algorithm in Static Nodes(Do Forever Till Node is not Placed)

```
1: Open XBee Port.
2: Node waits to read the data coming from previous node.
3: After data is read, it will get the RSSI value by reading the “DB” parameter.
4: Close XBee Port.
5: If the RSSI value $>$ 70, the Node is installed. LED glows for 15 seconds. Program
   returns to main program.
6: Else, the program keeps on repeating until the RSSI $>$ 70.
```

4.4.1.3 GPS Data Reading

Once the node is installed, it begins the GPS data Reading Algorithm. The GPS device is connected to a USB port of RaspberryPi. As the USB port is read by RaspberryPi, the algorithm takes out the latitude, longitude, altitude and time information from the raw GPS data received and store the data in the three queues(named as q, qNext and qPrevious). The data from these queues will be fetched to send data to Moving Node, previous node and the next node. All the three queues are of size 5. After the queues are full, the program returns to the main program. Algorithm 3 describes the GPS Data Reading in Static Breadcrumb Node.

Algorithm 3 GPS Data Reading in Static Nodes(Repeat till the Queues get fill)

```
1: Open the textfile to get the discovered Node MAC address and Node IDs.
2: Read the textfile to get the highest Node ID, which is the Node ID of the previous
   node installed.
3: The Node ID of the current node is PreviousNodeID +1.
4: Read the Raw GPS data coming to the node.
5: Extract the latitude, longitude, time and altitude data from Raw GPS Data.
6: Attach the Node ID of the current node to the Data.
7: Store the Data in Queue q, qPrevious and qNext, all are kept at size 5.
8: When the Queues get full with GPS data, return to the main program.
9: Else, if data is not available, fill the Queue with Data not available and return to
   the main program.
```

4.4.1.4 Unicasting Data to Previous, Next and Moving Node

After receiving data from the previous node and next node, the unicasting algorithm begins. This algorithm will fetch the GPS data stored in three separate queues(queue q, qPrevious and qNext) and unicast the data to moving node, previous node and next node. If the GPS data is not available, then data not available is stored in the queue and the node will try to approximate its latitude and longitude using the haversine formula (Equation 4.1), RSSI distance approximation and interpolation. After approximating the GPS location, the approximated data is forwarded to the moving vehicle, previous node and next node. The breadcrumb node will also forward the data received from previous and next node towards the Moving Node. Algorithm 4 describes the unicasting in Static Breadcrumb Node.

4.4.1.5 Data Reading Algorithm

Data Reading Algorithm starts after the GPS data reading algorithm. This algorithm will be waiting for the data to come to the XBee port. It will wait for 30 seconds for the arrival of new data. The algorithm will try to receive four data by running the receiver code four times. When a data is received, it will determine whether the data is coming from previous node or next node and accordingly place the data in the separate queues. Algorithm 5 describes the Data Reading in Static Breadcrumb

Algorithm 4 Unicasting Data to Previous, Next and Moving Node in Static Nodes

```
1: Open the text file where discovered Nodes MAC address and Node IDs are stored.
2: Read the text file and get the highest Node ID, which is the Node ID of the
   previous node installed.
3: The Node ID of the current node is PreviousNodeID +1.
4: The Node ID of the next Node is PreviousNodeID+2.
5: Check the Data of GPS stored in Queue q.
6: If GPS data is available, Discover the MAC address of Moving Node and unicast
   the Data Received from own GPS to the Moving Node.
7: Discover the previous MAC address of previous node and unicast GPS data from
   queue qPrevious.
8: Discover the next Node MAC address and unicast GPS data from queue qNext.
9: If GPS data is not available, call the Function CalculateApproximateGPSLocation
   to calculate the appoximate GPS location.
10: Discover the MAC address of moving node and unicast the Approximate GPS
    data calculated.
11: Discover the MAC address of previous node and unicast the Approximate GPS
    data by fetching the data from Queue qPrevious of size 5.
12: Discover the MAC address of next node and unicast own GPS data to next node
    by fetching the data from Queue qNext of size 5.
13: Discover the MAC address of Moving Node and unicast the Data Received from
    Previous node, which is stored in Queue qPreviousNodeData.
14: Discover the MAC address of Moving Node and unicast the Data Received from
    Next node, which is stored in Queue qNextNodeData.
```

Node.

4.4.1.6 Approximate GPS Location

For calculating the approximate GPS location, the breadcrumb node will fetch data of the previous node and next node and store them in separate queues. The distance between the two nodes can be calculated using haversine formula [35] as described below.

$$\begin{aligned} x &= long\beta - long\alpha \\ y &= lat\beta - lat\alpha \\ a &= sin(\frac{y}{2})^2 - cos(lat\alpha) \times cos(lat\beta) \times sin(\frac{x}{2})^2 c \quad = 2 \times tan^{-1}(\sqrt{a}, \sqrt{1-a})) \end{aligned} \tag{4.1}$$

$$D = c \times R$$

Where:

Algorithm 5 Data Reading Algorithm(Execute four times) in Static Nodes

```
1: Open the textfile to get the discovered Node MAC address and Node IDs.
2: Read the text file and get the highest Node ID, which is the Node ID of the previous node installed.
3: The Node ID of the current node is Previous NodeID +1.
4: The Node ID of the next Node is Previous NodeID+2.
5: Open XBee Port.
6: Receive the Data coming towards the Node. The receiver will get timeout after 30 seconds
7: Get the data and RSSI value from frame and concatenate them.
8: Check and identify, whether the concatenated data(data+RSSI) is coming from previous node or Next Node.
9: If data is coming from previous node, store it in queue named as qPreviousNodeData of maximum size 30.
10: If data is coming from next node, store it in queue named as qNextNodeData of maximum size 30.
```

$lat\alpha$: Latitude of a node

$long\alpha$: Longitude of a node

$lat\beta$: Latitude of another node

$long\beta$: Longitude of another node

a : Square of Half the chord length between the two points

c : Angular distance (radians)

R : Radius of the Earth $\cong$ 6370.0 kilometers(km)

D : Final, calculated distance between two Nodes

The node can also calculate its distance from the previous and next with the help of RSSI value. RSSI can be used to measure the distance between two points. The unique work of Shue et al. [36] has mentioned that RSSI is a good means to measure distances in an outdoor environment. The RSSI (in dBm) using the log-distance path loss model can be expressed as:

$$RSSI = 10nLog_{10}d + A \tag{4.2}$$

where n is the path loss exponent, d is the transmission distance in meters, and A is the reference value, which is the RSSI at 1 meter away from the transmitter. Most

Table 4.1: Path Loss Exponent values for different environments

Environment	**Path-Loss Exponent(n)**
Free Space	2.0
Urban Area Cellular Radio	2.7 to 3.5
Indoor Residential	1.4 to 1.8

wireless transceivers represent RSSI in -dBm (decibel milliwatts) which represents the amount of attenuation or lost power during transmission.

$$d = 10^{\frac{RSSI-A}{10n}} \tag{4.3}$$

The path loss exponent, n can be calculated for each environment by recording RSSI value at known distances and reverse solving for n. Typical values of n can be observed in Table 4.1.

The value of RSSI at a distance of 1 meter calculated in an open environment is = 47 dBm. The value of path loss exponent is 2. The distance of previous node to unlocalized current node is d= (d1/(d1+d2))*D where d1 is transmission distance in meters between previous node and current node calculated using Equation 4.4,

$$d1 = 10^{\frac{RSSI_1-47}{10n}} \tag{4.4}$$

and d2 is the transmission distance between current node and next node calculated using Equation 4.5.

$$d2 = 10^{\frac{RSSI_1-47}{10n}} \tag{4.5}$$

The ratio of the distance of previous node to unlocalized current node and distance between previous and next node is r= d/D. Using the interpolation, the latitude and longitude code can be approximated. This concept of approximating the latitude and longitude of the node without GPS can be understood from Figure 4.6.

4.4.1.7 Main Function working in Static Node

The main function will call all the algorithms described from Algorithm 1 to 6 in a sequential manner. First, the node discovery algorithm is executed. After successful node discovery and identification of the previous and next node, the node installation

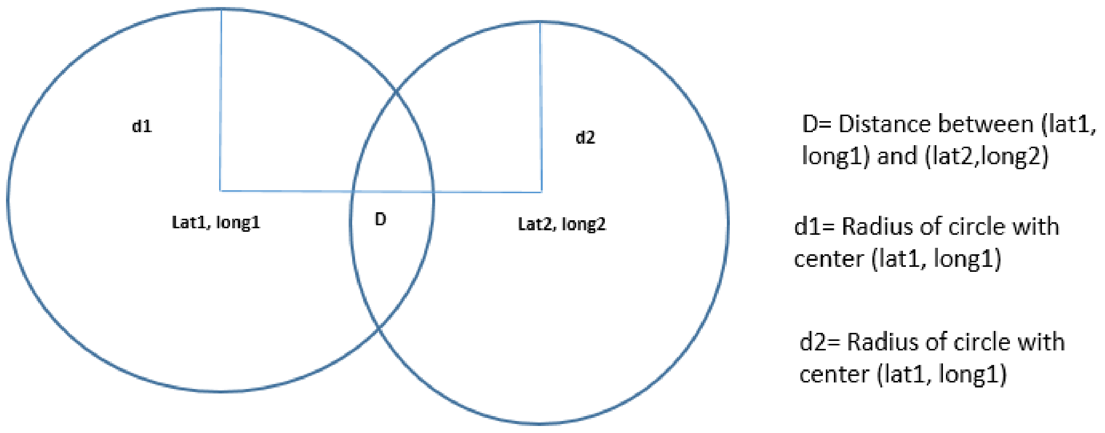

d1
d2
Lat1, long1
D
Lat2, long2
D= Distance between (lat1, long1) and (lat2,long2)
d1= Radius of circle with center (lat1, long1)
d2= Radius of circle with center (lat1, long1)

Figure 4.6: Intersection area of the circles formed using radius calculated by RSSI is the location of unknown node

Algorithm 6 Calculate Approximate GPS Location in Static Nodes

```
1: Fetch the GPS data stored in queue qPreviousNodeData.
2: Distance of PreviousNode to CurrentNode can be calculated by using Equation (4.4).
3: Fetch the GPS data stored in queue qNextNodeData.
4: Distance of NextNode to CurrentNode is calculated using Equation (4.5).
5: Use the Haversine Formula as show in Equation (4.1), to calculate the distance between Previous and Next node, which is returned as D.
6: Distance of Unlocalised current node to Previous Node is d = (d1/(d1+d2))*D
7: Ratio r = d/D
8: Using interpolate, find the Latitude and Longitude of unknown Node using the ratio r and GPS coordinates of previousNode and nextNode.
9: Return the Approximated Latitude and Longitude to the function, which is unicasting data to Previous Next and Current Node.
```

algorithm is executed. After the node is installed, the GPS data reading algorithm is initiated. The main function initiate the Data Reading algorithm, then unicasting the data to previous, next and moving node and then GPS Data reading algorithms after flushing the previously stored data in the queues(q, qNext and qPevious).

Algorithm 7 Main Function in Static Node

```
1: Node Discovery Algorithm
2: Installation Algorithm
3: GPS Data Reading Algorithm
4: Start the Infinite While Loop.
5: Data Reading Algorithm.
6: Unicasting Data to Previous, Next and Moving Node
7: Flush the Queue q, qPrevious and qNext.
8: GPS Data Reading.
```

4.4.1.8 Inter-Node Communication

The internode communication diagram between the static nodes is described in Figure 4.7, which explains the communication of Node N1 with its nearby nodes, i.e, Node C and Node N2. It explains all the communication carried between a static node with its nearby nodes, moving vehicle and the tool for installation.

4.4.2 Algorithm Implemented in the Coordinator

The Coordinator node is the first node of the XBee Network, which controls the entire network. The coordinator is the first node to be powered on after the Moving Node. It scans the network and complete the node discovery. Coordinator node do not have any previously installed static node. Because of this, there is no communication between the previously installed node with the current node in the Coordinator. Also, since the Coordinator do not have any previous node, it cannot calculate its approximated GPS location. The Coordinator should be placed at a place where it can receive GPS signals continuously. The internode Communication between Coordinator Node and its nearby Nodes is shown in Figure 4.8. Algorithms 8, 9, 10, 11 and 12 describes the complete algorithm working in the Coordinator Node.

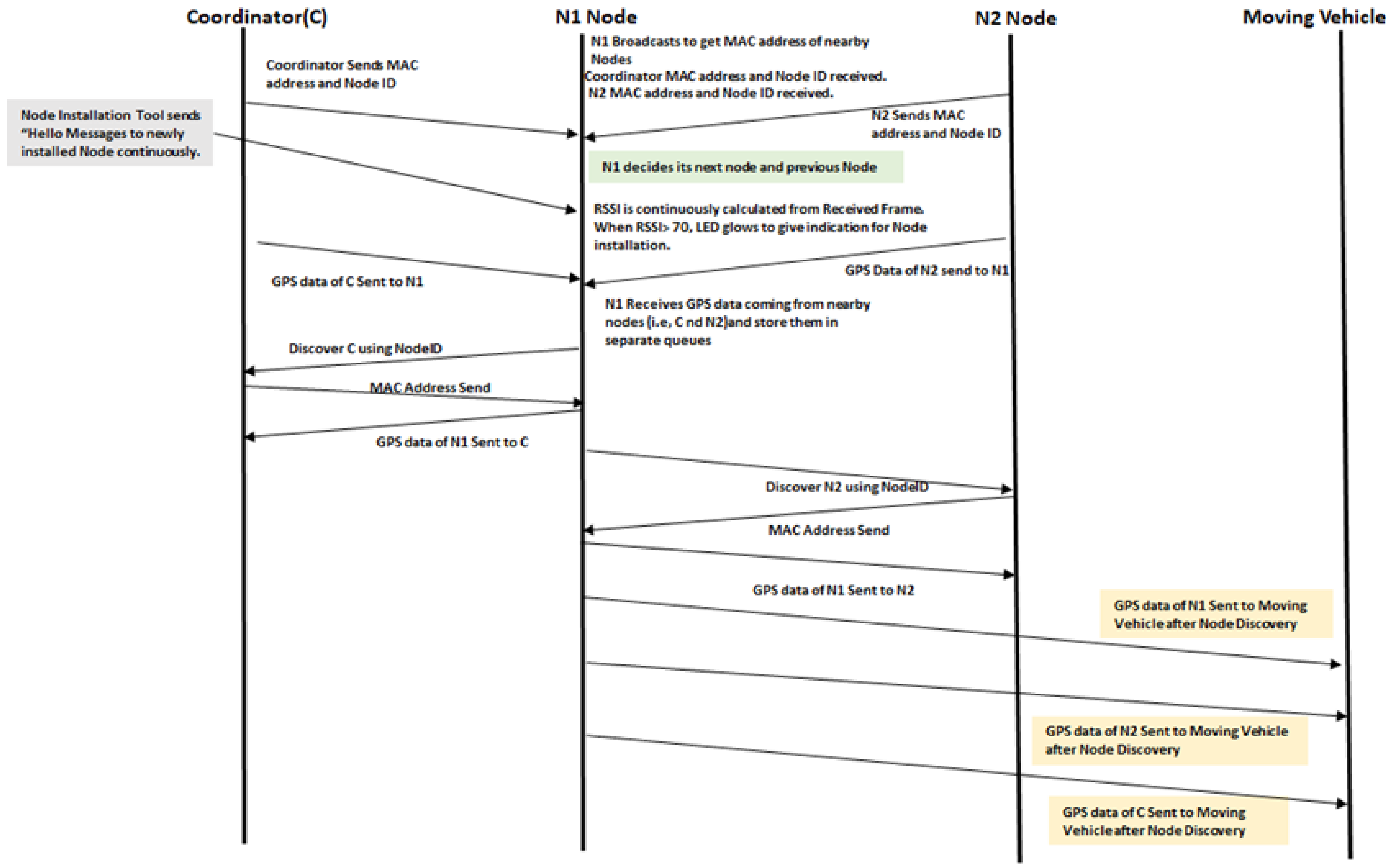

Coordinator(C)
N1 Node
N2 Node
Moving Vehicle
Coordinator Sends MAC address and Node ID
N1 Broadcasts to get MAC address of nearby Nodes
Coordinator MAC address and Node ID received.
N2 MAC address and Node ID received.
Node Installation Tool sends "Hello Messages to newly installed Node continuously.
N2 Sends MAC address and Node ID
N1 decides its next node and previous Node
RSSI is continuously calculated from Received Frame. When RSSI> 70, LED glows to give indication for Node installation.
GPS data of C Sent to N1
GPS Data of N2 send to N1
N1 Receives GPS data coming from nearby nodes (i.e, C nd N2)and store them in separate queues
Discover C using NodeID
MAC Address Send
GPS data of N1 Sent to C
Discover N2 using NodeID
MAC Address Send
GPS data of N1 Sent to N2
GPS data of N1 Sent to Moving Vehicle after Node Discovery
GPS data of N2 Sent to Moving Vehicle after Node Discovery
GPS data of C Sent to Moving Vehicle after Node Discovery

Figure 4.7: Communication between a Static Node with its nearby Nodes

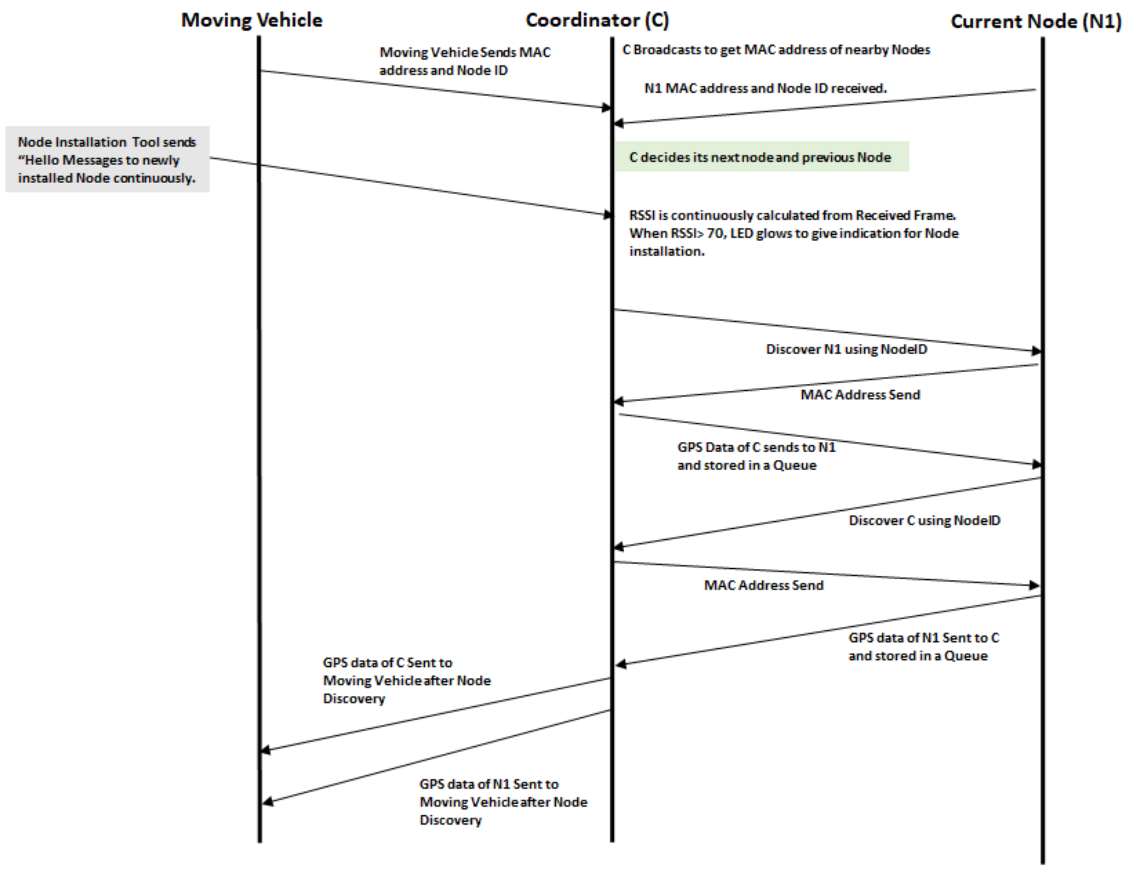

Moving Vehicle
Coordinator (C)
Current Node (N1)
Moving Vehicle Sends MAC address and Node ID
C Broadcasts to get MAC address of nearby Nodes
N1 MAC address and Node ID received.
Node Installation Tool sends "Hello Messages to newly installed Node continuously.
C decides its next node and previous Node
RSSI is continuously calculated from Received Frame. When RSSI> 70, LED glows to give indication for Node installation.
Discover N1 using NodeID
MAC Address Send
GPS Data of C sends to N1 and stored in a Queue
Discover C using NodeID
MAC Address Send
GPS data of N1 Sent to C and stored in a Queue
GPS data of C Sent to Moving Vehicle after Node Discovery
GPS data of N1 Sent to Moving Vehicle after Node Discovery

Figure 4.8: Communication between a Coordinator Node with its nearby Nodes

Algorithm 8 Node Discovery in Coordinator Node

```
1: The node is Powered ON.
2: Open XBee Port.
3: The Node sends a broadcast signal to nearby nodes for their MAC Address and
   Node ID.
4: The nearby Nodes send their Node IDs and MAC Address and these are stored
   in a textfile.
5: Node Discovery is done for only 15 seconds. In 15 seconds, all the nearby nodes
   send their Node IDs and MAC Address.
6: Close XBee Port.
```

Algorithm 9 GPS Data Reading in Coordinator Node(Repeat till the Queues get fill)

```
1: Open the textfile to get the discovered Node MAC address and Node IDs.
2: Read the textfile to get the highest Node ID, which is the Node ID of the previous node installed.
3: The Node ID of the current node is PreviousNodeID +1.
4: Read the Raw GPS data coming to the node.
5: Extract the latitude, longitude, time and altitude data from Raw GPS Data.
6: Attach the NodeID of the current node to the Data.
7: Store the Data in Queue q and qNext, all are kept at size 5.
8: When the Queues get full with GPS data, return to the main program.
9: Else, if data is not available till more than 20 iterations, fill the Queue with "Data not available" and return to the main program.
```

Algorithm 10 Unicasting Data to Next and Moving Node in Coordinator Node

```
1: Open the textfile where discovered Nodes MAC address and Node IDs are stored.
2: Read the textfile and get the highest Node ID, which is the Node ID of the previous node installed.
3: The Node ID of the next Node is PreviousNodeID+2.
4: Discover the MAC address of Moving Node and unicast the Data Received from own GPS to the Moving Node.
5: Discover the next Node MAC address and unicast own GPS data to next node by fetching the data from Queue qNext of size 5.
6: Discover the MAC address of Moving Node and unicast the Data Received from Next node, which is stored in Queue qNextNodeData.
```

Algorithm 11 Data Reading Algorithm in Coordinator Node (Execute four times)

```
1: Open the textfile to get the discovered Nodes MAC address and Node IDs.
2: Read the text file and get the highest Node ID, which is the Node ID of the previous node installed.
3: The Node ID of the next Node is PreviousNodeID+2.
4: Open XBee Port.
5: Receive the Data coming towards the Node. The receiver will get timeout after 30 seconds
6: Get the data and RSSI value from frame and concatenate them.
7: Check and identify, whether the concatenated data(data+RSSI) is coming from previous node or Next Node.
8: If data is coming from next node, store it in queue named as qNextNodeData of maximum size 30.
```

Algorithm 12 Main Function in Coordinator Node

```
1: Node Discovery Algorithm
2: Installation Algorithm
3: GPS Data Reading Algorithm
4: The nearby Nodes send their Node IDs and MAC Address and these are stored
   in a text file.
5: Start the Infinite While Loop.
6: Unicasting Data to Next and Moving Node
7: Data Reading Algorithm.
8: Flush the Queue q and qNext.
9: GPS Data Reading.
```

4.4.3 Algorithm Implemented in Breadcrumb Nodes without GPS

In order to make the system cost effective in its future implementation at a large scale, some nodes in the system are without GPS and their GPS location is calculated with the help of the GPS location of their previous and next nodes. After the GPS location is calculated inside the node, it is forwarded to the moving vehicle, previous node and the next node. Algorithms 13 to 18 describes the complete algorithm working in the Node without GPS.

Algorithm 13 Node Discovery in Node Without GPS

```
1: The node is Powered ON
2: Open XBee Port.
3: The Node sends a broadcast signal to nearby nodes for their MAC Address and
   Node ID.
4: The nearby Nodes send their Node IDs and MAC Address and these are stored
   in a textfile.
5: Node Discovery is done for only 15 seconds. In 15 seconds, all the nearby nodes
   send their Node IDs and MAC Address.
6: Close XBee Port.
```

4.4.4 Algorithm Implemented in the Node Deployment Tool

In Wireless sensor networks, the nodes must be installed at the farthest possible distances, so that the least number of nodes are required to make a network. But the main problem, which is encountered while working in the outside field is that it is very difficult to measure the exact location for node placement and if a node

Algorithm 14 Installation Algorithm in Node Without GPS (Do Forever Till Node is not Placed)

```
1: Open XBee Port.
2: Node waits to read the data coming from previous node.
3: After data is read, it will get the RSSI value by reading the “DB” parameter.
4: Close XBee Port.
5: If the RSSI value $> 70$, the Node is installed. LED glows for 15 seconds. Program
   returns to main program.
6: Else, the program keeps on repeating until the RSSI $> 70$.
```

Algorithm 15 Unicasting Data to Previous, Next and Moving Node in Node Without GPS

```
1: Open the text file where discovered Nodes MAC address and Node IDs are stored.
2: Read the text file and get the highest Node ID, which is the Node ID of the
   previous node installed.
3: The Node ID of the current node is Previous Node ID $+1$.
4: The Node ID of the next Node is PreviousNodeID$+2$.
5: since no GPS is connected, call the Function **CalculateApproximateGPSLo-
   cation** to calculate the appoximate GPS location.
6: Discover the MAC address of moving node and unicast the Approximate GPS
   data calculated.
7: Unicast the Approximated GPS data calculated in step 5.
8: Unicast the Approximated GPS data calculated in step 5.
9: Discover the MAC address of Moving Node and unicast the Data Received from
   Previous node, which is stored in Queue qPreviousNodeData.
10: Discover the MAC address of Moving Node and unicast the Data Received from
    Next node, which is stored in Queue qNextNodeData.
```

Algorithm 16 Data Reading Algorithm in Node Without GPS (Execute four times)

```
1: Open the textfile to get the discovered Nodes MAC address and Node IDs.
2: Read the textfile and get the highest Node ID, which is the Node ID of the
   previous node installed.
3: The Node ID of the current node is PreviousNodeID $+1$.
4: The Node ID of the next Node is PreviousNodeID$+2$.
5: Open XBee Port.
6: Receive the Data coming towards the Node. The receiver will get timeout after
   30 seconds
7: Get the data and RSSI value from frame and concatenate them.
8: Check and identify, whether the concatenated data(data+RSSI) is coming from
   previous node or Next Node.
9: If data is coming from previous node, store it in queue names as qPreviousNode-
   Data of maximum size 30.
10: If data is coming from next node, store it in a queue named as qNextNodeData
    of maximum size 30.
```

Algorithm 17 Calculate Approximate GPS Location

```
1: Fetch the GPS data stored in queue qPreviousNodeData.
2: Distance of PreviousNode to CurrentNode can be caculated by using Equation (4.4).
3: Fetch the GPS data stored in queue qNextNodeData.
4: Distance of NextNode to CurrentNode is Equation (4.5).
5: Use the Haversine Formula as show in Equation (4.1), to calculate the distance between Previous and Next node, which is returned as D.
6: Distance of Unlocalised current node to Previous Node is d = (d1/(d1+d2))*D
7: Ratio r = d/D
8: Using interpolate, find the Latitude and Longitude of unknown Node using the ratio r and GPS coordinates of previousNode and nextNode.
9: Return the Approximated Latitude and Longitude to the function, which is unicasting data to Previous Next and Current Node
```

Algorithm 18 Main Function in Node Without GPS

```
1: Node Discovery Algorithm.
2: Installation Algorithm.
3: Start the Infinite While Loop.
4: Data Reading Algorithm.
5: Unicasting Data to Previous, Next and Moving Node
```

is placed outside the Zigbee Netowork range, their is a network breakdown and all other nodes installed after that donot work. In order to prevent this mistake, while a node is being installed, there should be tool a which is sending data "Hello" message towards the newly installed node. The newly installed node should measure the RSSI from the frame received continously. When the RSSI value increases above 70, the LED attached to the node should glow to give indication to the user that there is a sufficient separation and new node should be installed now.

4.4.5 Algorithm Implemented in the Moving Node

The moving node acts like an autonomous vehicle. It receives the data continously from the nodes which are sending data towards it. From this data, the moving node knows the latitude and longitude information of the next node even before reaching near it. In this way, the autonomous vehicle gets to know about the location of the next node similar to the fairy tail of Hansel and Gratel.

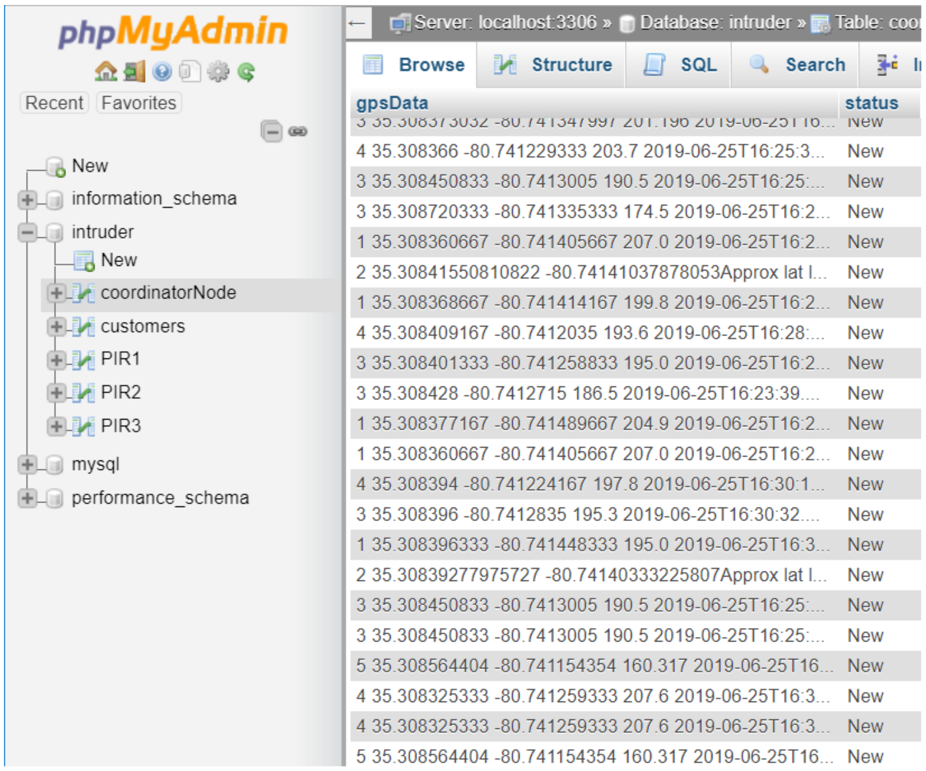
Algorithm 19 Main Function in the Node used as Deployment Tool

```
1: The node is powered ON.
2: Open XBee Port.
3: The Node sends a broadcast signal to nearby nodes for their MAC Address and
   Node ID.
4: The nearby Nodes send their Node IDs and MAC Address and these are stored
   in a textfile.
5: Node Discovery is done for only 15 seconds. In 15 seconds, all the nearby nodes
   send their Node IDs and MAC Address.
6: Close XBee Port.
7: Identify the newest power on node.
8: Send Hello Messages to newly power on Node to help in its installation.
9: Power OFF the Node after the node has been installed.
```

Algorithm 20 Main Function of Moving Node

```
1: Open the XBee Port.
2: Receive the Data coming towards the Node. The receiver will get timeout after
   1000 seconds.
3: Get the data and RSSI value from frame and concatenate them.
4: Store the data MySQL.
5: Close the XBee port.
6: Repeat the above steps continuously.
```

The Moving Node node stores all the data in the MySQL database table installed inside the RaspberryPi. The phpMyAdmin server is also installed inside the RaspberryPi. All the data stored in the MySQL database can be accessed by using a laptop connected to the same network or by using the IP address of the Moving Vehicle RaspberryPi. The data stored in the PHPMyAdmin server is as shown in Figure 4.9.



gpsData	status
3 35.308373032 -80.741347997 201.196 2019-06-25T16...	New
4 35.308366 -80.741229333 203.7 2019-06-25T16:25:3...	New
3 35.308450833 -80.7413005 190.5 2019-06-25T16:25:...	New
3 35.308720333 -80.741335333 174.5 2019-06-25T16:2...	New
1 35.308360667 -80.741405667 207.0 2019-06-25T16:2...	New
2 35.30841550810822 -80.74141037878053Approx lat l...	New
1 35.308368667 -80.741414167 199.8 2019-06-25T16:2...	New
4 35.308409167 -80.7412035 193.6 2019-06-25T16:28:...	New
3 35.308401333 -80.741258833 195.0 2019-06-25T16:2...	New
3 35.308428 -80.7412715 186.5 2019-06-25T16:23:39....	New
1 35.308377167 -80.741489667 204.9 2019-06-25T16:2...	New
1 35.308360667 -80.741405667 207.0 2019-06-25T16:2...	New
4 35.308394 -80.741224167 197.8 2019-06-25T16:30:1...	New
3 35.308396 -80.7412835 195.3 2019-06-25T16:30:32....	New
1 35.308396333 -80.741448333 195.0 2019-06-25T16:3...	New
2 35.30839277975727 -80.74140333225807Approx lat l...	New
3 35.308450833 -80.7413005 190.5 2019-06-25T16:25:...	New
3 35.308450833 -80.7413005 190.5 2019-06-25T16:25:...	New
5 35.308564404 -80.741154354 160.317 2019-06-25T16...	New
4 35.308325333 -80.741259333 207.6 2019-06-25T16:3...	New
4 35.308325333 -80.741259333 207.6 2019-06-25T16:3...	New
5 35.308564404 -80.741154354 160.317 2019-06-25T16...	New

Figure 4.9: GPS data of all the Router Nodes stored at PhpMyAdmin Server

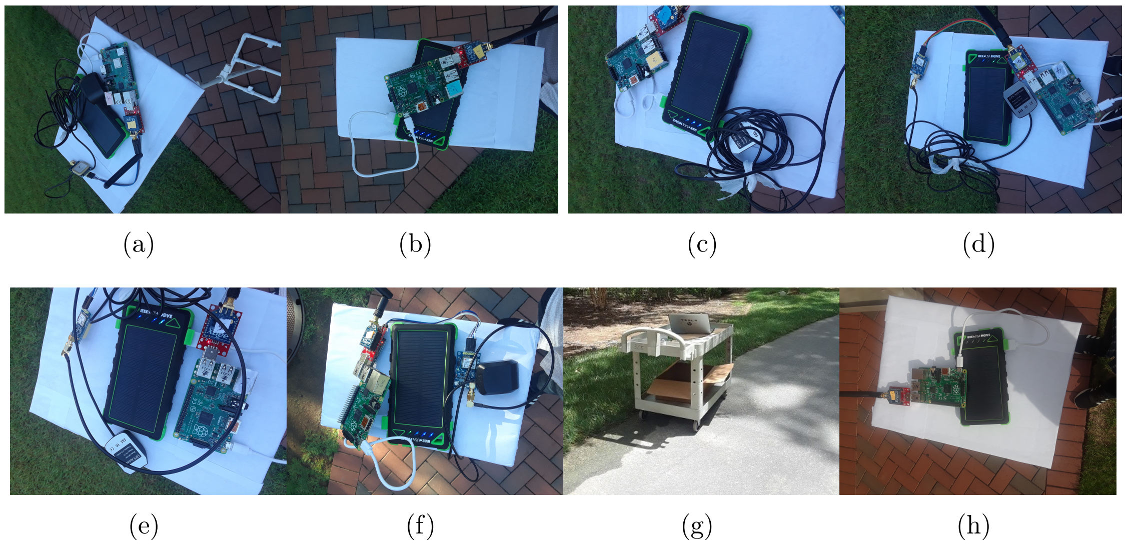


Figure 4.10: Location of the Static breadcrumb nodes: (a) Node1; (b) Node2; (c) Node3; (d) Node4; (e) Node5; (f) Node6; (g) Moving Node; and, (h) Tool for Installation

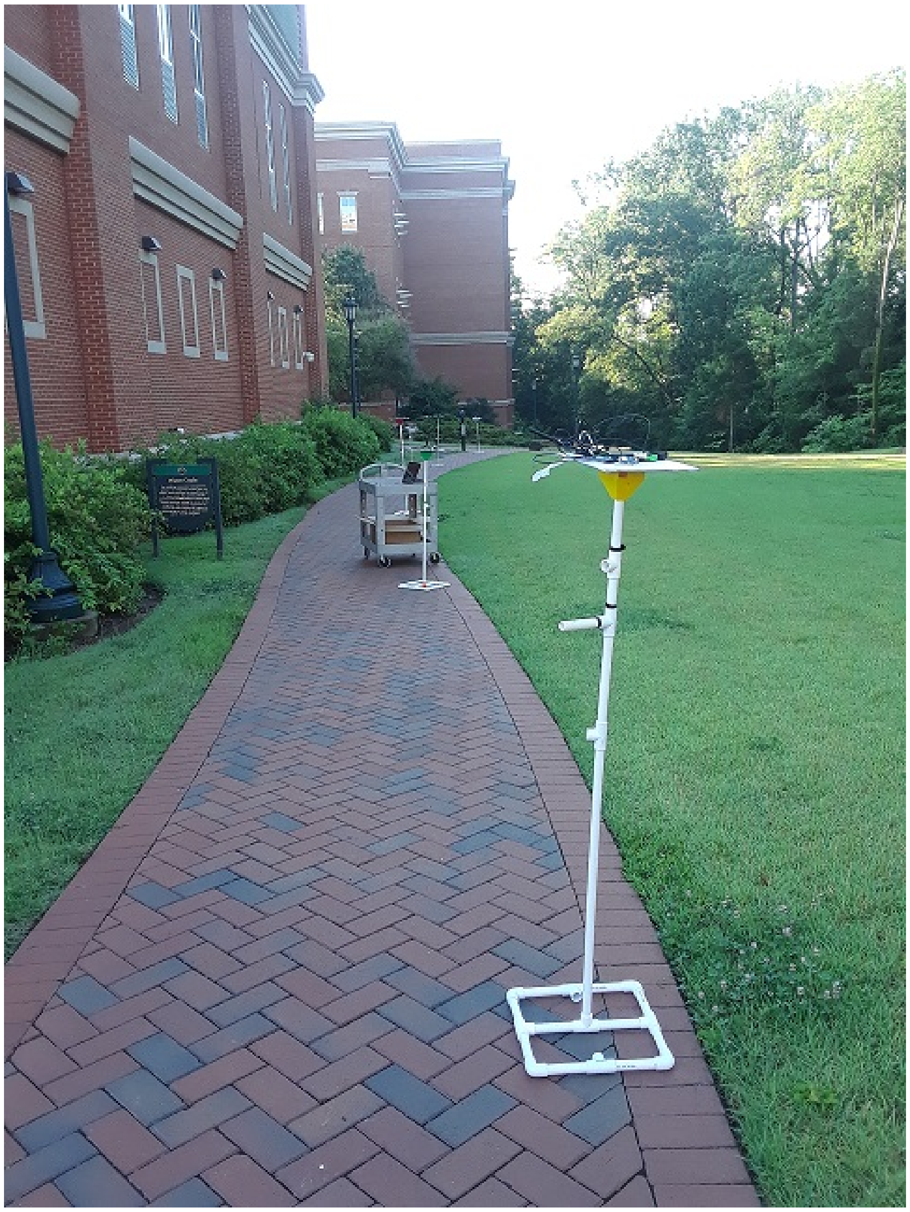


Figure 4.11: Entire System Setup

The entire system setup of eight Nodes, of which five are Router nodes, one Coordinator, one Moving Node and one Node Deployment Tool is shown in Figure 4.10. Figure 4.11 shows the location of each node.

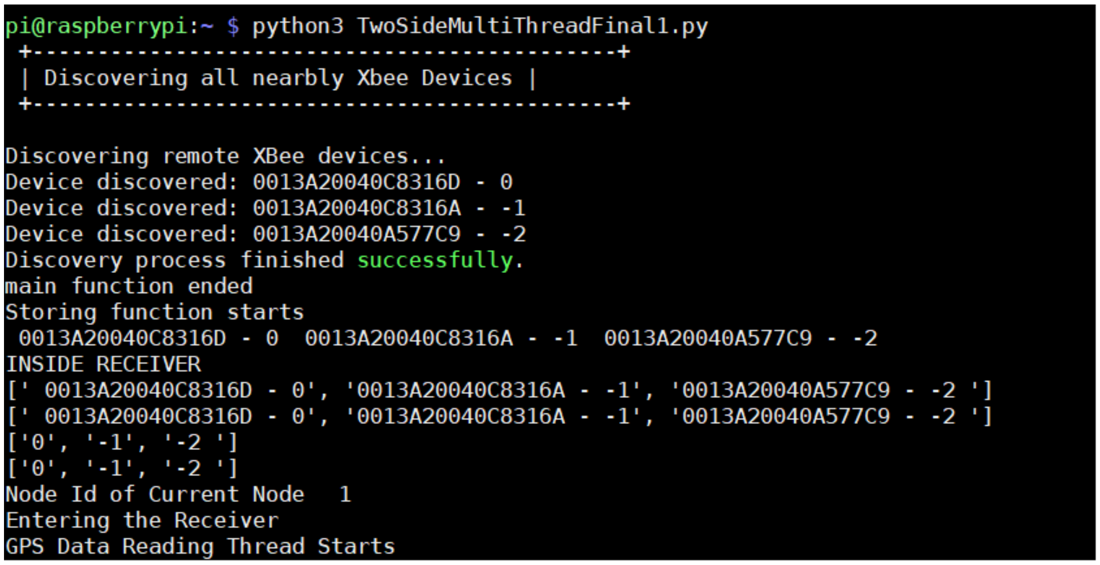
CHAPTER 5: TESTS AND OBSERVATIONS

5.1 Breadcrumb Network Prototype

For the purpose of testing the breadcrumb network prototype, all the XBees are configured at the lowest power level. All the breadcrumbs are kept at a height of 1 meter and they are at line of sight to each other. The tests were performed in a trail near the EPIC building of UNC Charlotte.

5.1.1 Node Discovery and Determination of Previous Node, Current Node and the Next Node

The breadcrumb network designed contains the same program in all the router nodes with GPS and a slightly different program in the nodes without GPS. When the Node is powered on, it will start doing the Node Discovery by sending broadcast message to all the nodes in the system to get their MAC Addresses. In the initial fifteen seconds, all the nearby nodes will send their MAC addresses along with their Node Identifier. These MAC Addresses and Node Identifiers are stored in the textfile. After the Node identification, the program will identity the maximum value Node ID present in the textfile. This Node ID is the Node ID of the previously installed node. The program increases the maximum received Node ID by one to get the Node ID of the currently powered on node and get the Node ID of the next node by increasing the Node ID of currently powered on node by one. The Node IDs are put in increasing order while configuring the XBees using XCTU software. The Coordinator is given the Node ID of zero. The other Router nodes are given Node IDs from 1 to 5. The process of Node Identification can be seen in Figure 5.1.



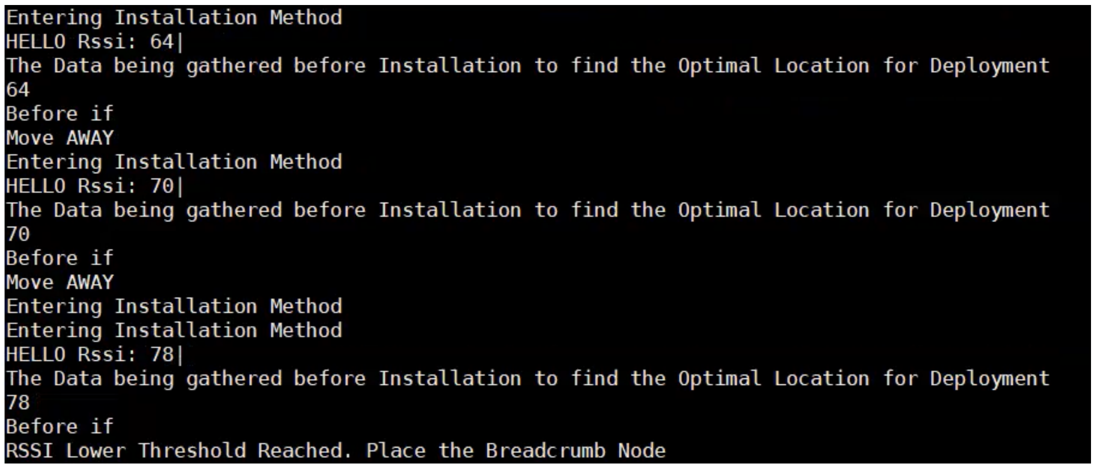
```
pi@raspberrypi:~ $ python3 TwoSideMultiThreadFinal1.py
 +-----------------------------------------+
 | Discovering all nearbly Xbee Devices |
 +-----------------------------------------+

Discovering remote XBee devices...
Device discovered: 0013A20040C8316D - 0
Device discovered: 0013A20040C8316A - -1
Device discovered: 0013A20040A577C9 - -2
Discovery process finished successfully.
main function ended
Storing function starts
 0013A20040C8316D - 0  0013A20040C8316A - -1  0013A20040A577C9 - -2
INSIDE RECEIVER
[' 0013A20040C8316D - 0', '0013A20040C8316A - -1', '0013A20040A577C9 - -2 ']
[' 0013A20040C8316D - 0', '0013A20040C8316A - -1', '0013A20040A577C9 - -2 ']
['0', '-1', '-2 ']
['0', '-1', '-2 ']
Node Id of Current Node   1
Entering the Receiver
GPS Data Reading Thread Starts
```

Figure 5.1: Device Discovery by Breadcrumb Node

5.1.2 Installation Algorithm

After the execution of the Node Discovery to find the previous and the next node, the Node Installation Algorithm begins. Once the Node is powered on, a Node installation Tool is also powered on. The Node Installation tool identifies the newly powered on node and send "Hello Messages" to it. The RSSI value is extracted from the frame received. The RSSI value in -dBm keeps on increasing when the distance increases. When the RSSI value reaches to 70, an external LED glows, signalling the installation of the Node. Figure 5.2 gives relation between different power levels, RSSI and distances.



```
Entering Installation Method
HELLO Rssi: 64|
The Data being gathered before Installation to find the Optimal Location for Deployment
64
Before if
Move AWAY
Entering Installation Method
HELLO Rssi: 70|
The Data being gathered before Installation to find the Optimal Location for Deployment
70
Before if
Move AWAY
Entering Installation Method
Entering Installation Method
HELLO Rssi: 78|
The Data being gathered before Installation to find the Optimal Location for Deployment
78
Before if
RSSI Lower Threshold Reached. Place the Breadcrumb Node
```

Figure 5.2: New Node Installation

In order to calculate the maximum range reliable communication between two XBees devices, the RSSI values at different distances for 0 Power Level(+10 dBm) is shown in Figure 5.3. As the distance increases the RSSI value decreases.

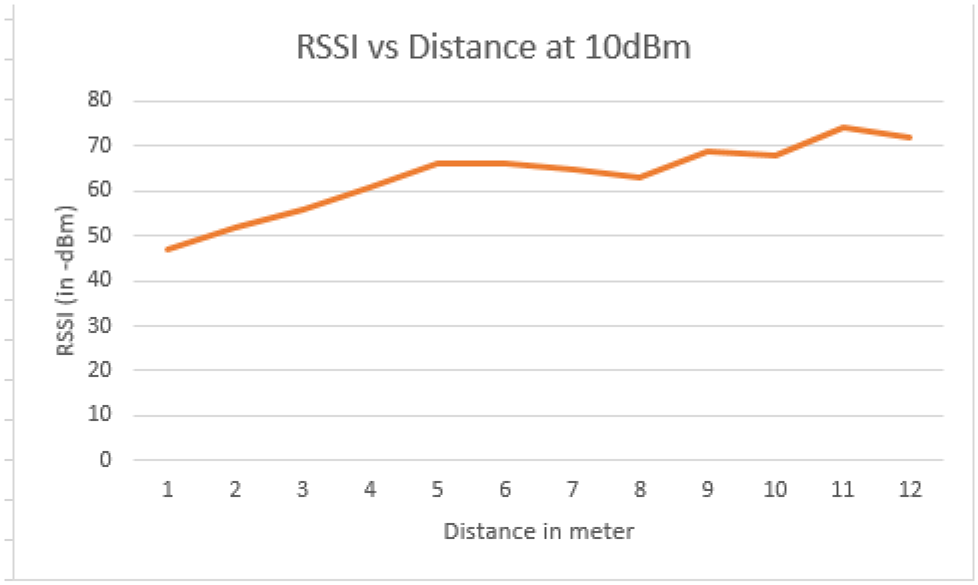


Figure 5.3: RSSI vs Distance at 0 Power Level

5.1.3 GPS Data of all the Breadcrumb Nodes Collected in MqSQL Database of Coordinator

The breadcrumb network prototype consists of five Routers, one Coordinator Node and one Moving node. The results collected in the MySQL database is as shown in Figure 5.5. The breadcrumb network will give the GPS location of the Node to the Moving node well ahead, so that the moving node can navigate smoothly through unknown terrain. The actual location for the test case results in shown in Figure 5.4. The location of these GPS coordinates on the Google Map is shown in Figure 5.6

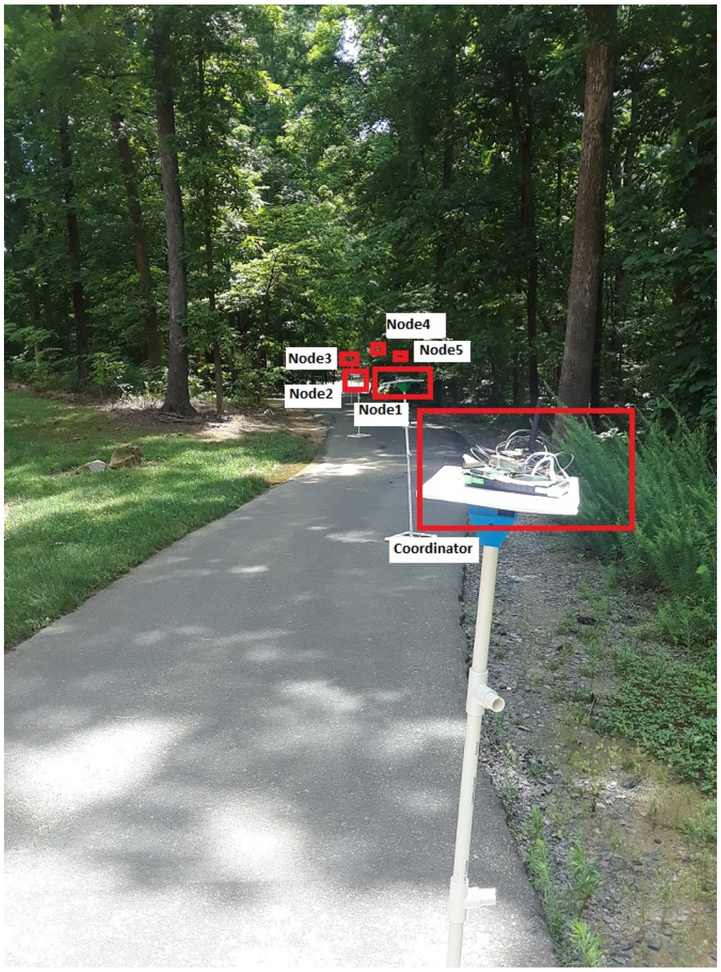
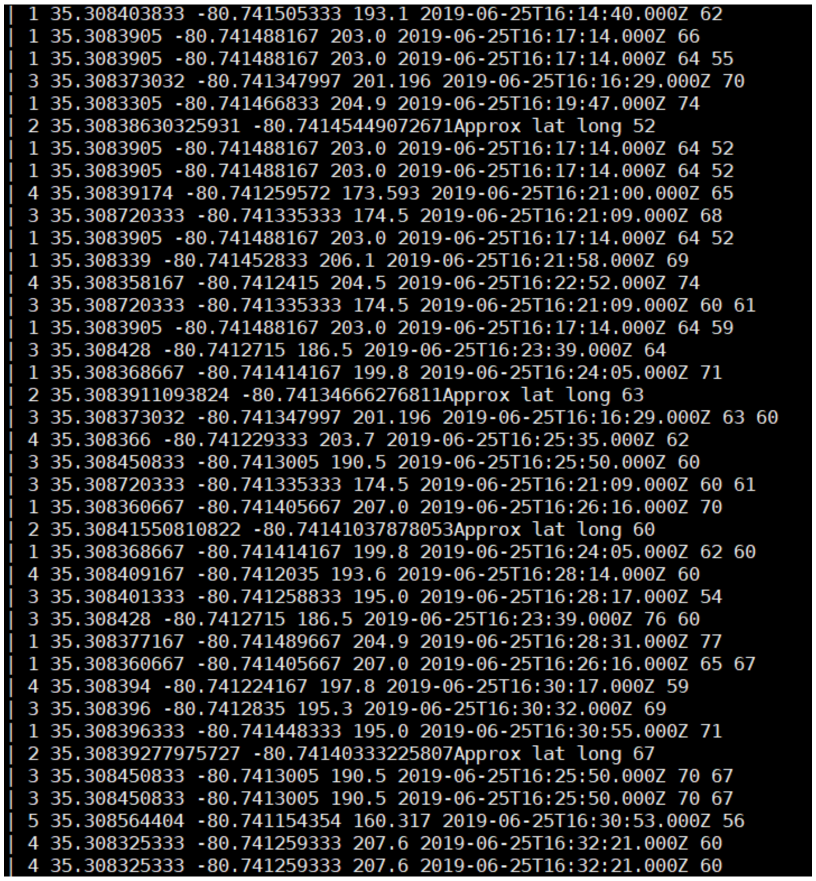


Figure 5.4: Location of the Testing



```
| 1 35.308403833 -80.741505333 193.1 2019-06-25T16:14:40.000Z 62
| 1 35.3083905 -80.741488167 203.0 2019-06-25T16:17:14.000Z 66
| 1 35.3083905 -80.741488167 203.0 2019-06-25T16:17:14.000Z 64 55
| 3 35.308373032 -80.741347997 201.196 2019-06-25T16:16:29.000Z 70
| 1 35.3083305 -80.741466833 204.9 2019-06-25T16:19:47.000Z 74
| 2 35.30838630325931 -80.74145449072671Approx lat long 52
| 1 35.3083905 -80.741488167 203.0 2019-06-25T16:17:14.000Z 64 52
| 1 35.3083905 -80.741488167 203.0 2019-06-25T16:17:14.000Z 64 52
| 4 35.30839174 -80.741259572 173.593 2019-06-25T16:21:00.000Z 65
| 3 35.308720333 -80.741335333 174.5 2019-06-25T16:21:09.000Z 68
| 1 35.3083905 -80.741488167 203.0 2019-06-25T16:17:14.000Z 64 52
| 1 35.308339 -80.741452833 206.1 2019-06-25T16:21:58.000Z 69
| 4 35.308358167 -80.7412415 204.5 2019-06-25T16:22:52.000Z 74
| 3 35.308720333 -80.741335333 174.5 2019-06-25T16:21:09.000Z 60 61
| 1 35.3083905 -80.741488167 203.0 2019-06-25T16:17:14.000Z 64 59
| 3 35.308428 -80.7412715 186.5 2019-06-25T16:23:39.000Z 64
| 1 35.308368667 -80.741414167 199.8 2019-06-25T16:24:05.000Z 71
| 2 35.3083911093824 -80.74134666276811Approx lat long 63
| 3 35.308373032 -80.741347997 201.196 2019-06-25T16:16:29.000Z 63 60
| 4 35.308366 -80.741229333 203.7 2019-06-25T16:25:35.000Z 62
| 3 35.308450833 -80.7413005 190.5 2019-06-25T16:25:50.000Z 60
| 3 35.308720333 -80.741335333 174.5 2019-06-25T16:21:09.000Z 60 61
| 1 35.308360667 -80.741405667 207.0 2019-06-25T16:26:16.000Z 70
| 2 35.30841550810822 -80.74141037878053Approx lat long 60
| 1 35.308368667 -80.741414167 199.8 2019-06-25T16:24:05.000Z 62 60
| 4 35.308409167 -80.7412035 193.6 2019-06-25T16:28:14.000Z 60
| 3 35.308401333 -80.741258833 195.0 2019-06-25T16:28:17.000Z 54
| 3 35.308428 -80.7412715 186.5 2019-06-25T16:23:39.000Z 76 60
| 1 35.308377167 -80.741489667 204.9 2019-06-25T16:28:31.000Z 77
| 1 35.308360667 -80.741405667 207.0 2019-06-25T16:26:16.000Z 65 67
| 4 35.308394 -80.741224167 197.8 2019-06-25T16:30:17.000Z 59
| 3 35.308396 -80.7412835 195.3 2019-06-25T16:30:32.000Z 69
| 1 35.308396333 -80.741448333 195.0 2019-06-25T16:30:55.000Z 71
| 2 35.30839277975727 -80.74140333225807Approx lat long 67
| 3 35.308450833 -80.7413005 190.5 2019-06-25T16:25:50.000Z 70 67
| 3 35.308450833 -80.7413005 190.5 2019-06-25T16:25:50.000Z 70 67
| 5 35.308564404 -80.741154354 160.317 2019-06-25T16:30:53.000Z 56
| 4 35.308325333 -80.741259333 207.6 2019-06-25T16:32:21.000Z 60
| 4 35.308325333 -80.741259333 207.6 2019-06-25T16:32:21.000Z 60
```

Figure 5.5: GPS Location of all Nodes Collected in MySQL database of Moving Node

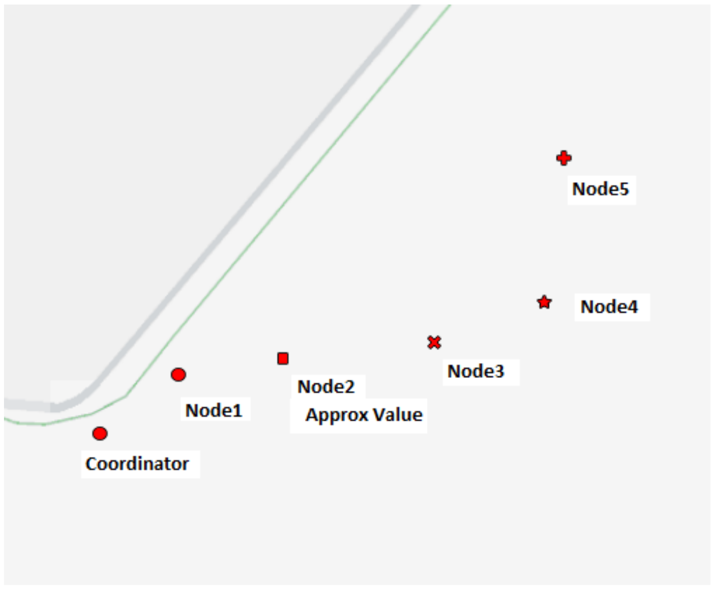


Figure 5.6: Location of all the Nodes seen on Google Map

5.1.4 Approximating the Location of the Node not Having GPS

All the Nodes in the breadcrumb network doesnot have a GPS. The Nodes without GPS try to approximate their GPS coordinates i.e, latitude and longitude with the help of latitude and longitude of the previous node and the next node. These results are fairly accurate. The distance between the actual location and approxinate location is 1.5 meters. GPS is a very costly device. By avoiding putting GPS at some of the breadcrumb nodes decreases the cost of the prototype by substantial amount.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

This research work sought to solve the problem of providing GPS coordinates to an autonomous vehicle operating in an unstructured human uninhabitable environment. The breadcrumb network designed is tested on a prototype of five routers, one coordinator node and one moving node. The breadcrumb nodes automatically do the node discovery to determine the previous and next node. A tool is also made, which will help the user in establishing the nodes at maximum distances. This network can be extended to any number of nodes. We have incorporated all the problems which can cause the system to fail while designing the network. This network can be extremely helpful in an event of natural calamities like hurricane, floods, earthquake and other geologic processes. This system finds application in an event of security personnel doing some combing operation in an area where there is no network. The breadcrumb network can provide vital GPS coordinates to an autonomous vehicle to reach the last deployed node by traversing the nodes and provide help to the first responder.

In future, a lot of work can be done on increasing the efficiency of the system further. This system can be in future connected to autonomous vehicle and can become very useful to work in an unstructured human uninhabitable environment. This system will be a big step towards making a autonomous vehicle travel smoothly in unstructured environment to long distances without human intervention.

REFERENCES

[1] "XBee/XBee-PRO S2C Zigbee RF Module." `https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf`, 2018.

[2] G. Mao, B. Fidan, and B. D. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[3] K. Miranda, A. Molinaro, and T. Razafindralambo, "A survey on rapidly deployable solutions for post-disaster networks," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 117–123, 2016.

[4] J. Q. Bao and W. C. Lee, "Rapid deployment of wireless ad hoc backbone networks for public safety incident management," *GLOBECOM - IEEE Global Telecommunications Conference*, pp. 1217–1221, 2007.

[5] M. R. Souryal, J. Geissbuehler, L. E. Miller, and N. Moayeri, "Real-time deployment of multihop relays for range extension," *Proceedings of the 5th international conference on Mobile systems, applications and services - MobiSys '07*, p. 85, 2007.

[6] M. Klann, T. Riedel, H. Gellersen, C. Fischer, M. Oppenheim, P. Lukowicz, G. Pirkl, K. Kunze, M. Beuster, M. Beigl, O. Visser, and M. Gerling, "LifeNet: an Ad-hoc Sensor Network and Wearable System to Provide Firefighters with Navigation Support," *Adjunct Proc Ubicomp 2007*, vol. M, no. 1, pp. 124–127, 2007.

[7] M. Hazas, C. Kray, H. Gellersen, H. Agbota, G. Kortuem, and A. Krohn, "A relative positioning system for co-located mobile devices," *Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys '05*, p. 177, 2005.

[8] C. Decker, A. Krohn, M. Beigl, and T. Zimmer, "The particle computer system," *2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005*, vol. 2005, pp. 444–448, 2005.

[9] A. Krohn, M. Beigl, C. Decker, and T. Riedel, "Syncob: Collaborative time synchronization in wireless sensor networks," *4th International Conference on Networked Sensing Systems, INSS*, no. July, pp. 283–290, 2007.

[10] J. Wilson, V. Bhargava, A. Redfern, and P. Wright, "A wireless sensor network and incident command interface for urban firefighting," *Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous 2007*, 2007.

[11] J. Snydal, A. Van Pelt, and J. Wilson, "FireEye: Needs and Usability Assessment of a Head-Mounted Display for Firefighters," 2005.

[12] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," *2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005*, vol. 2005, pp. 364–369, 2005.

[13] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler., "TinyOS: An Operating System for Sensor Networks, Technicle report, University of Berkeley," 2007.

[14] A. Juels, "RFID security and privacy: A research survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, 2006.

[15] H. Liu, J. Li, Z. Xie, S. Lin, K. Whitehouse, J. A. Stankovic, and D. Siu, "Automatic and robust breadcrumb system deployment for indoor firefighter applications," *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10*, p. 21, 2010.

[16] S. E. B. Troubleshooting, "Design Note DN300 Design Note DN300," pp. 1–13.

[17] K. Pothuganti and A. Chitneni, "A Comparative Study of Wireless Protocols," *IECON Proceedings (Industrial Electronics Conference)*, no. January 2014, pp. 46–51, 2014.

[18] T. Lai, W.-J. Chen, K.-H. Li, P. Huang, and H.-H. Chu, "TriopusNet: automating wireless sensor network deployment and replacement in pipeline monitoring," *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pp. 61–72, 2012.

[19] CrossBow, "TelosB," *North*, 2008.

[20] S. Cc, "Chipcon Chipcon," *ReVision*, pp. 2004–2006, 2004.

[21] A. Purohit, Z. Sun, F. Mokaya, and P. Zhang, "SensorFly: Controlled-mobile Sensing Platform for Indoor Emergency Response Applications," *Processing in Sensor*, vol. 2009, no. April, pp. 223–234, 2011.

[22] NXP, "LPC2146 ARM7 Microcontroller Datasheet," no. August, 2011.

[23] K. I. Sram and W. E. Cycles, "8-bit Atmel Microcontroller with 64K Bytes Programmable ATmega644," 2013.

[24] NIST, "NIST Special Publication 1026 CFAST – Consolidated Model of Fire Growth and Smoke Transport (Version 6) Technical Reference Guide," *Nist Special Publication*, vol. 1026, no. April, 2009.

[25] H. Liu, Z. Xie, J. Li, K. Whitehouse, J. Stankovic, S. Lin, and D. Siu, "Efficient and reliable breadcrumb systems via coordination among multiple first responders," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, pp. 1005–1009, 2011.

[26] H. Wang, S. Sen, and A. Elgohary, "No need to war-drive: Unsupervised indoor localization," *Proceedings of the 10th international conference on Mobile systems, applications, and services Pages 197-210*, pp. 197–210, 2012.

[27] M. Chandra, M. T. Jones, and T. L. Martin, "E-textiles for autonomous location awareness," *IEEE Transactions on Mobile Computing*, vol. 6, no. 4, pp. 367–379, 2007.

[28] A. Correa, M. Barcelo, A. Morell, and J. L. Vicario, "Enhanced inertial-aided indoor tracking system for wireless sensor networks: A review," *IEEE Sensors Journal*, vol. 14, no. 9, pp. 2921–2929, 2014.

[29] B. E. van Issum and N. H. Chamberlain, "37—The free diameter and specific volume of textile yarns," *Journal of the Textile Institute Transactions*, vol. 50, no. 11, pp. T599–T623, 1959.

[30] J. Schmid, T. Gadeke, W. Stork, and K. D. Muller-Glaser, "On the fusion of inertial data for signal strength localization," *Proceedings of the 8th Workshop on Positioning Navigation and Communication 2011, WPNC 2011*, pp. 7–12, 2011.

[31] J. Li, Z. Xie, X. Sun, J. Tang, H. Liu, and J. A. Stankovic, "An automatic and accurate localization system for firefighters," *Proceedings - ACM/IEEE International Conference on Internet of Things Design and Implementation, IoTDI 2018*, pp. 13–24, 2018.

[32] M. Richardson and S. Wallace, *Getting started with raspberry PI*. O'Reilly Media, Inc., 2012.

[33] X. X.-p. R. F. Modules, X. X.-p. R. F. Modules, and B. R. East, "Digi International Inc .," 2013.

[34] "XCTU User Guide," 2018.

[35] B. B. Rhoades, *A Novel Framework for Integrating Legacy Vehicles into an Intelligent Transportation System*. PhD thesis, University of North Carolina at Charlotte, 2018.

[36] S. Shue and J. M. Conrad, "Procedurally generated environments for simulating RSSI- localization applications," in *Proceedings of the 20th Communications & Networking Symposium*, CNS '17, (San Diego, CA, USA), pp. 7:1–7:11, Society for Computer Simulation International, 2017.