# Design and development of Intelligent Mobile Robots (IMRs) for disaster mitigation and firefighting

## Sixth Deliverable

## Report submitted on 15th October 2017


## PI: Dr. Muhammad Bilal Kadri

## Co-PI: Dr. Tariq Mairaj Rasool Khan

# Table of Contents

# Chapter 1. Introduction

This report is the sixth deliverable for the National ICTR&D funded research project titled *"Design and development of Intelligent Mobile Robots (IMRs) for disaster mitigation and firefighting"*.

Sixth deliverable includes:

a) A working module of the IMR with a built in capability to traverse and unknown space and generate a map.·
b) A prototype IMR to be tested for path following, fire detection.
c) The IMR will be tested for collision avoidance.
d) All the bottlenecks faced in the development will be reported.

# Chapter 2. Working model of the Intelligent Mobile Robot (IMR)

The Intelligent Mobile Robotics Lab has so far designed, developed and constructed several different robots in order to achieve the tasks and the goals for this project. Much of the work has been done to create academic test platforms, dummy mobile platforms and sensor platforms which have proven to be instrumental in determining effectiveness of sensors in several different environments, indoor, outdoor, fire test site and also have played an instrumental role in determining the kinds and types of actuators needed and how powerful they need to be.

For the purpose of academic research, IMR Lab designed different robot models and procured some catering to different objectives of the IMR Lab project, which include testing of algorithms in lab environment, testing of robot actuators and sensors in different conditions and also for collaborative control of robots. We call our robots MUSAFIR bots and IMR bots, the MUSAFIR bots are the ones which have primary focus on the research aspect of this project and the IMR bots which are geared towards the final implementation and testing for the work done in this project.

## 2.1 MUSAFIR Robots

MUSAFIR Robots as mentioned in this chapter and in previous reports are the bots whose primary focus is for research purposes and are not targeted for final fire related tests. However, these have been very much useful in working on algorithms and testing different techniques for wheeled mobile robots. Figures 2.1-2.5 show the different MUSFIR Robots and how they evolved to include different sensors and be used in different environments other than lab environment.

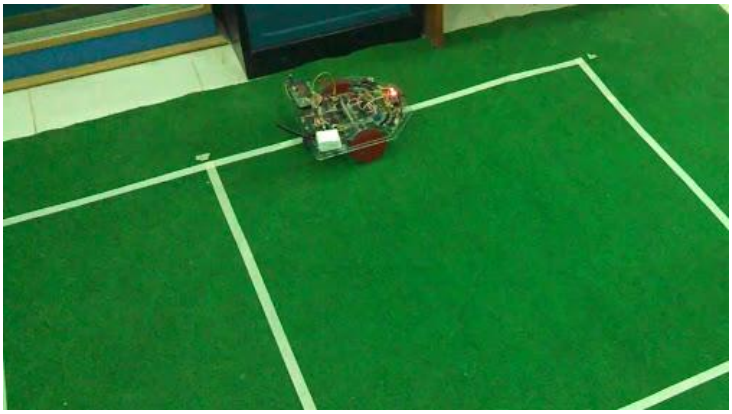


Figure 2.1: MUSAFIR v1 – while being tested in the IMR LAB

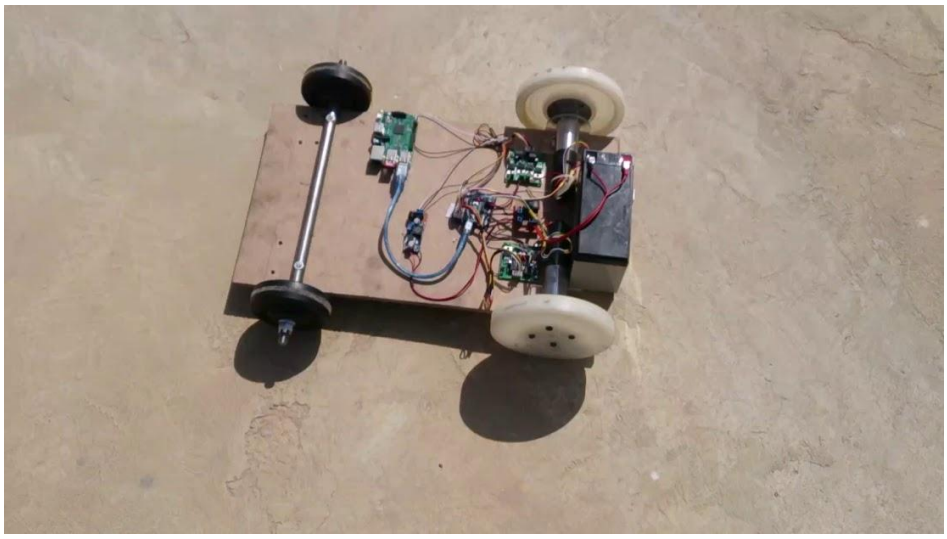


Figure 2.2: MUSAFIR v2 – with basic circuitry – OUTDOOR Test

Figure 2.3: MUSAFIR v2: with improvements to Wheels and Sensors – IMU/GPS/Odometry

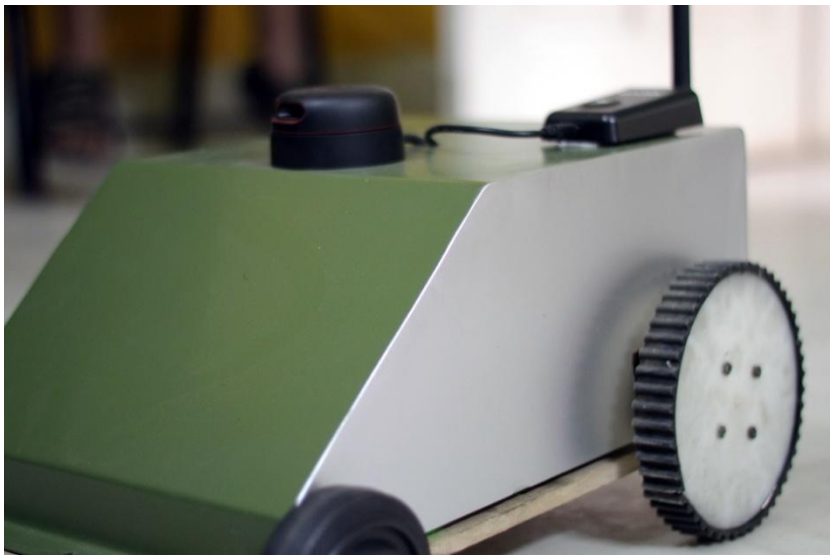


Figure 2.4: MUSAFIR v2 with cover – LIDAR and WiFi shown outside

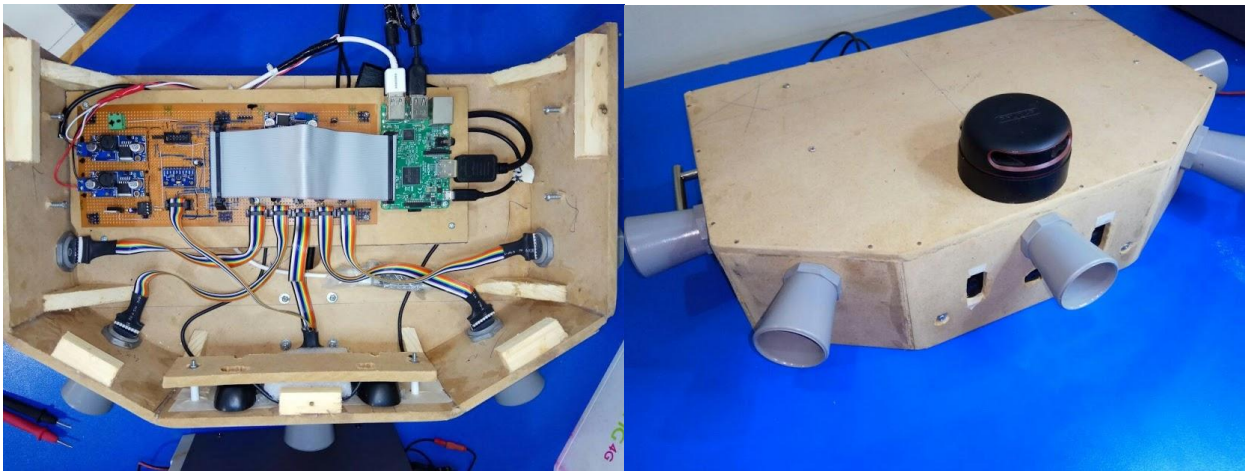


Figure 2.5: MUSAFIR Robot Sensor Head – LIDAR / UTs / IMU / Thermal Cam / Stereo Vision / GPS and interface to motor control and drivers.

As seen from the above figures, much work was done on the MUSAFIR robots and the platforms to achieve better understanding of the final work.

## 2.2 IMR Robots (Wheeled and non-Wheeled)

The IMR Lab final project goal is to design and develop firefighting robots which are capable enough to work in unknown areas and autonomously just with on board systems and sensors. The robots have to be robust enough to traverse on uneven and rough terrain and also be able to avoid obstacles and hurdles. We developed several test robotic platforms, wheeled and belt driven mechanisms and tried different iterations of differential drive robots.

Figures 2.6 and 2.7 show some of the final designs that we used for further tests and improvements.

Figure 2.6: Wheeled IMR Robot – Bare Chassis

Figure 2.7: Belt Driven IMR Robot – Bare Chassis

## 2.3 Experimentation in the test environment

A test area was also setup with walls and multiple gateways and paths to different areas which can be closed and opened as will to test the robustness of the algorithms and the maneuverability of the robots. The IMR Robots were tested on the open area and grounds of PAF KIET which are un-paved and rough terrain for a mobile robot. We were able to move the robots to and for from the workshop to the Test are and do several sensor tests and other tests in the test area.

Figure 2.8: IMR Wheeled Robot – on its way to the Test Site

The test environment and methodology also requires for a fire to be created which is to be detected by the robot and also the robot be able to douse the fire by means of water or using a fire extinguisher. We tested the capability of the robot to drag a filled and un-filled water hose across uneven and rough terrain and also its ability to aim water towards a stationary target.

Figure 2.9: Wheeled IMR Robot with Fire hose mechanism – driving on uneven terrain

To test the capability of the robot to extinguish fires, we tested the water hose dragging of the robot and the capability of the robot to aim the water hose at a target, this proved to be quite difficult as this involved very high loads which the robot needed to drag along and the filled water hose and the unfilled hose both have extremely different load profiles and effects resulting in the robot behaving differently in both conditions even when the motor code was same.

We also had to waterproof the electronics and the motors of the robot such that any leakage or spilling of the water would not destroy the work.

Figure 2.10: Water proofed IMR Wheeled Robot – also Water Reseviour

Figure 2.11: IMR Wheeled Robot with Water Hose – max distance – 35 to 45 feet forward

## 2.4 Bottlenecks faced with W-IMR

The real world tests of the real IMR Robots have provided us with data and observations which are simply impossible to be found otherwise, we now have a much better understanding of how the robots have to work and behave and most of our observations have led us to do further changes and improvements to the IMR Robots design.

### 2.4.1 Fire Hose

The fire hose which the robot has to drag has to be of a specific type which does not get flattened when empty and it has to be unwound from a roller with external help so that the robot does not

have to drag the entire load when unwinding the hose. Our initial tests were with a different hose which did not have either of the above mentioned properties.

### 2.4.2 Water Pump

The maximum distance the robot can aim the water towards is about 35 to 45 Feet, this depends entirely on the water pump that is used at the back to pump out water. Further improvements in the range can be obtained by using a different type of head for the hose.

### 2.4.3 Water Reservoir

The robots ability to douse fires also depends on the water reservoir and its capacity to hold water.

## 2.5 Modifications proposed

The robot tests provided us with several insights which have resulted in modifications at different levels of the robot design and are being incorporated for future work in the IMR Lab.

### 2.5.1 Better Water Proofing

The robot needs a better solution for water proofing of the electronics, for this, IP67 rated enclosures have been proposed.

### 2.5.2 Plumbing

The robot plumbing currently has the water pipes and everything going from the top portion of the robot and the electronics is right below it, we have proposed to change this such that the plumbing is setup at the bottom of the robot. This provides dual benefit of securing the electronics and also bringing the center of gravity of the robot down, which improves overall stability and grip of the robot.

### 2.5.3 Plumbing

The robot plumbing currently has the water pipes and everything going from the top portion of the robot and the electronics is right below it, we have proposed to change this such that the plumbing is setup at the bottom of the robot. This provides dual benefit of securing the electronics and also bringing the center of gravity of the robot down.

# Chapter 3. Map generation from IMR and other navigation aids

## 3.1 Introduction

Map generation is the task of using the data obtained from distance, range and proximity sensors and converting that into a usable map where the robot is able to locate itself in it and also be able to navigate in a given map.

Given a map, an autonomous robot can be expected to drive in the mapped area without fear of colliding with walls and other obstacles, considering that the environment is static and there are no dynamic obstacles involved. However, generating such a map is itself a tedious and non-trivial task. Maps can be of different types and sizes and may span from map of a single room to a whole floor plan layout to a building to the maps which use satellite imagery and other methods to map entire areas and cities.

## 3.2 Types of Maps

The Maps may be of different types, topological or metric. Metric maps show the world in a way that humans can more easily relate to it as they show the world with objects and distances between them in proper scaled manner, see figure 3.1. Topological maps on the other hand are more suited for computational work and are only concerned with the points of interests and their relation with other points, resulting in a graph with nodes and the distances among them as cost parameters of a graph, figure 3.2

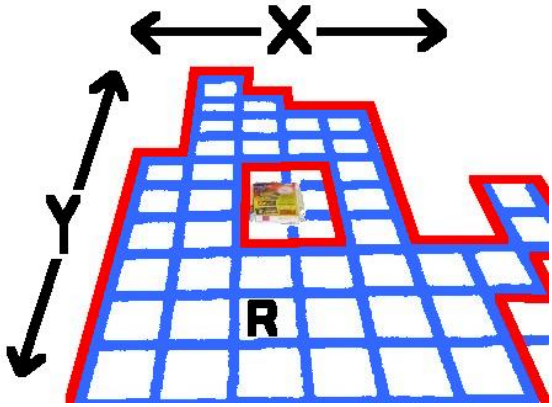



Figure 3.1: Metric Map showing a grid, robot and 2 dimensional data

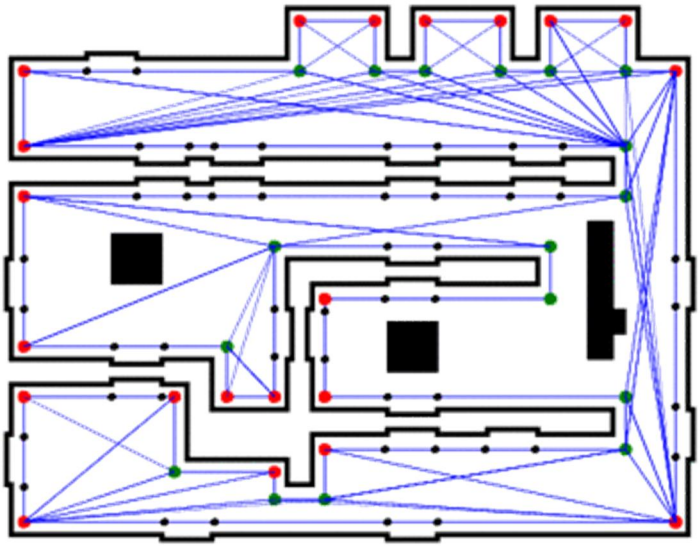


Figure 3.2: Topological Map (inside graph view) the map is shown inside a metric representation

### 3.3 Map generation in IMR Lab

Our work in IMR Lab focuses on robots which can move autonomously in areas which they have not mapped or chartered before and hence our task is further complicated by the need to generate a map at runtime while the robot is in operation in a dynamic environment where the obstacles can move around due to different circumstances and actions. This type of real time or run time mapping is coupled with the ability for the robot to be able to locate itself inside such a map, this complete process is known as SLAM or Simultaneous Localization and Mapping.

### 3.4 Use of SLAM for map generation

SLAM has been one of the most challenging problems of Robotics and work has been on going to produce SLAM algorithms that can cope with the uncertainties in the sensor data, the environment, large areas and such since the past 10 years and more. We surveyed several different SLAM approaches, including but not limited to GMapping, OrthoSLAM, Visual SLAM, OrbiSLAM, OrbSLAM, Hector SLAM, RGBDSLAM to name a few, some of these methods require specific sensors and availability of accurate odometry, some require a start point to be manually given and start location be manually provided in order for the algorithm to work. Most of the SLAM algorithms require the use of a LIDAR Laser scanner to provide the basic information which is then coupled with odometry information, information from inertial measuring sensors and or with visual sensors, cameras.

We started our work testing different algorithms and approaches to SLAM and finally have narrowed down to a just a few methods which have shown us promising results, the best results so far have been from HECTR SLAM, which uses only the LIDAR information and can self-generate the map on the fly and also generate relative odometry on its own.

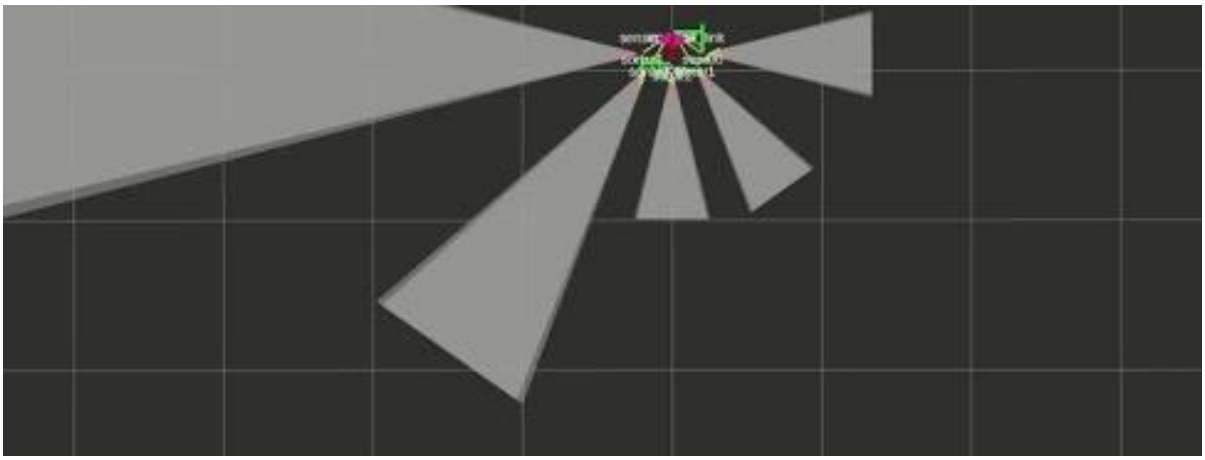


Figure 3.3: UltraSonic rays visualized as cones coming out of sensors form Robot

Map generation requires that the robot is aware of its surroundings to a great degree of detail and for that sensors like simple proximity sensors or range sensors like ultra sonic sensors cant be used as the primary source for reason that the data points are very sparse and there is not enough density in the points, see figure 3.3 and distances provided by these sensors, on the other hand, LASER Range finders, LIDARs (figure 3.4) or RGBD cameras (figure 3.5) provide point clouds which give very detailed information.

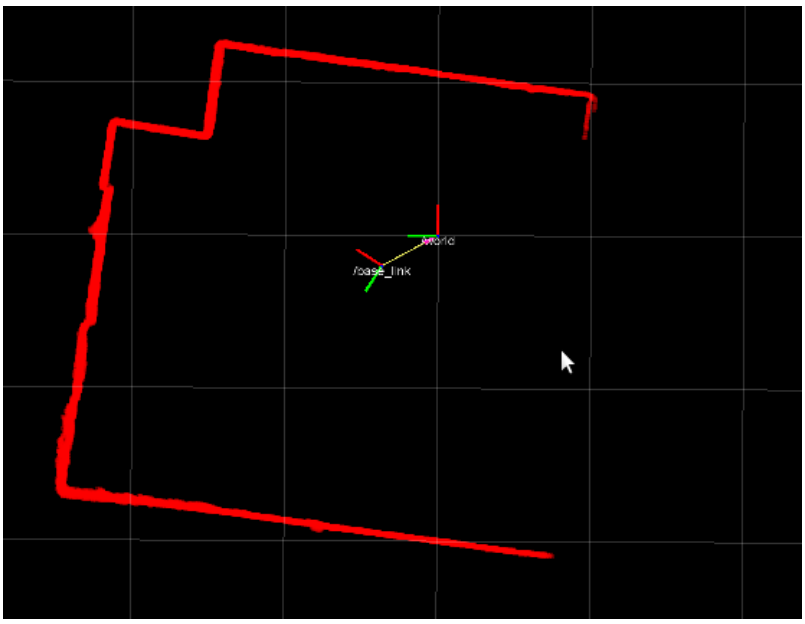



Figure 3.4: LASER Scan data being visualized in RVIZ

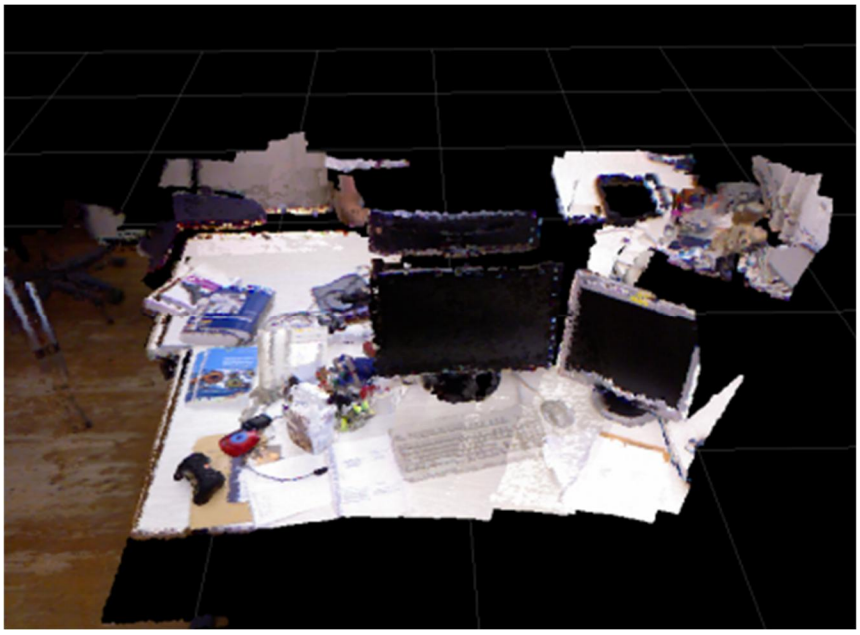


Figure 3.5: an RGBD Point Cloud visualized in RVIZ

## 3.5 Hector SLAM

From the official ROS Hector SLAM Wiki

> *"hector_mapping is a SLAM approach that can be used without odometry as well as on platforms that exhibit roll/pitch motion (of the sensor, the platform or both). It leverages the high update rate of modern LIDAR systems like the Hokuyo UTM-30LX and provides 2D pose estimates at scan rate of the sensors (40Hz for the UTM-30LX). While the system does not provide explicit loop closing ability, it is sufficiently accurate for many real world scenarios. The system has successfully been used on Unmanned Ground Robots, Unmanned Surface Vehicles, Handheld Mapping Devices and logged data from quadrotor UAV"*

at IMR Lab, we don't have access to the Hokuyo LIDAR sensors, but we were able to use the Chinese cheaper variant, RP LIDAR A2 to work with the Hector Slam ROS Node and were able to generate map of the Lab and the floor while tele-operating the MUSAFIR robot. Figures 3.6 to 3.8 show how the Hector SLAM Method builds up a map form the incoming LIDAR Data.

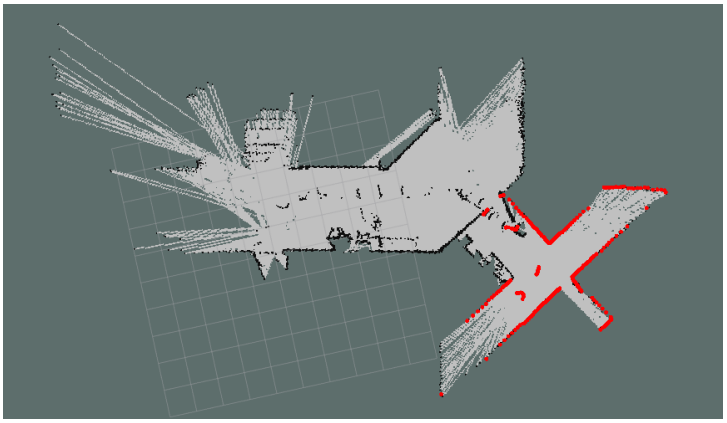


Figure 3.6: New LIDAR Data shown in red while previously generated map is behind the robot

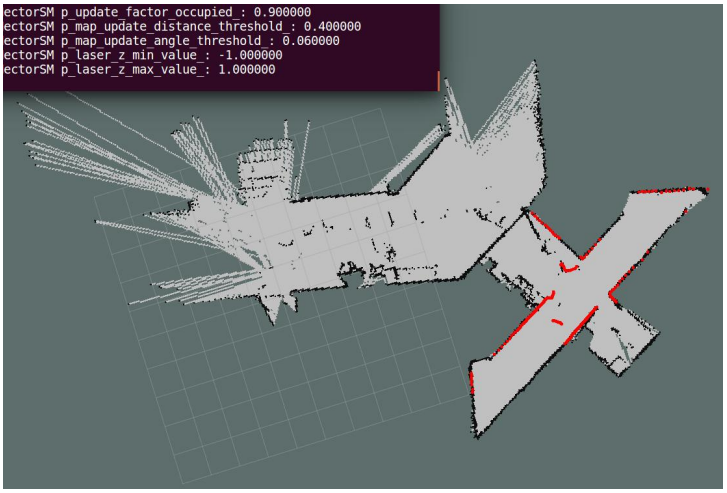



Figure 3.7: The Map is further expanded as the robot moves

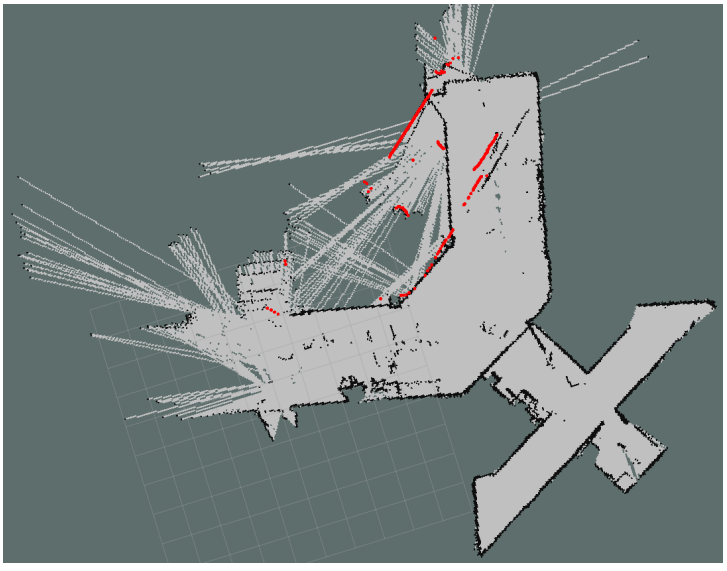


Figure 3.8: Due to high speed turns, the SLAM algorithm failed to keep up with the data

# Chapter 4. Bottlenecks faced in the development of lab based test environment

## 4.1 Overview of the lab based environment for testing formation control of mobile robots

We used Zumo 32U4 robots for testing purposes. We used these because the size of the actual robot is very large for testing in the lab. Zumo robot is equipped with built-in Arduino-compatible ATmega32U4 microcontroller, LCD, encoders for closed-loop motor control, and proximity sensors for obstacle detection. The robot is compact enough to qualify as a mini sumo robot, but it's high-performance motors and integrated sensors make it versatile enough to serve as a general-purpose small robot.

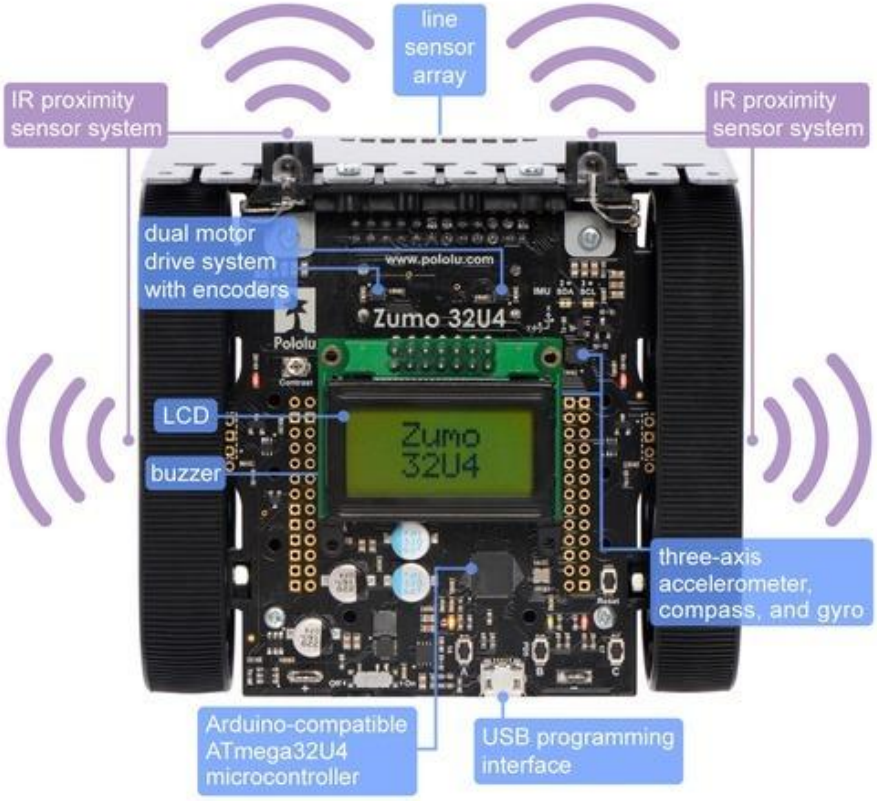
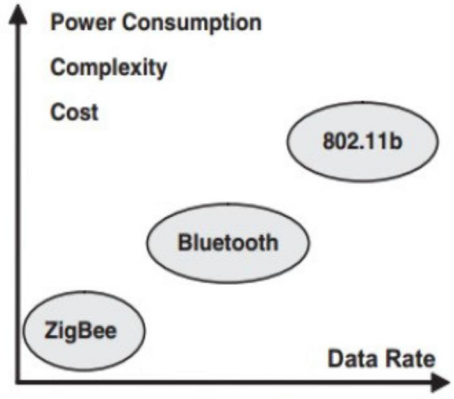



Figure 5.1 Zumo32U4

**Wi-Fi Technology (IEEE 802.11.a/b/h/g):** The Wi-Fi technology allows different devices like laptops, personal computers (PCs), cell phones, and personal digital assistants (PDAs) to communicate between one another or to connect to the Internet without needing a cable connection. The Institute of Electrical and Electronics Engineers (IEEE) defined the Wi-Fi network protocols IEEE 802.11a, 802.11b, and 802.11 g operating in the unlicensed radio bands of 2.4 GHz and 5 GHz. Therefore, any kind of standard Wi-Fi–certified device is able to operate

all over the world with data rates of 11Mbps for IEEE 802.11b or 54 Mbps for IEEE 802.11a. Of course, the greater the distance to the access point, the lower the performance. Wi-Fi technology lacks a low-power mode and is also not very highly integrated. Thus, a low-powered and highly integrated WSN cannot use this technology, which is why we give just an overview of this existing technology.

**Bluetooth Technology (IEEE 802.15.1):** The Bluetooth wireless communications technology provides a personal area network (PAN) for exchanging data between Bluetooth-capable devices within a certain proximity. Bluetooth technology has a low-power mode and high integrated devices and operates in the unlicensed 2.4-GHz band, but it is limited to short-distance communications. Therefore, this technology is not the most appropriate for developing a WSN. For this reason, Bluetooth is just mentioned and described as an existing technology.




| | Data Rate | Typical Range | Application Examples |
|---|---|---|---|
| ZigBee | 20 to 250 Kbps | 10–100 m | Wireless Sensor Networks |
| Bluetooth | 1 to 3 Mbps | 2–10 m | Wireless Headset Wireless Mouse |
| IEEE 802.11b | 1 to 11 Mbps | 30–100 m | Wireless Internet Connection |

Figure 5.2: Comparison between ZigBee, Bluetooth and IEEE 802.11b

For our project we chose Zigbee as it is evident from the comparison graph that the Zigbee consumes the least power, is less complex and costs the least.

## 4.2 Communication between the robots and the base station

### XBee Series 2

We used XBee Series 2 for our project. It uses a microchip from Ember Networks that enables several different flavors of standards-based ZigBee mesh networking. Mesh networking is the heart of creating robust sensor networks, the systems that can generate immensely rich data sets or support intricate human-scale interactions.

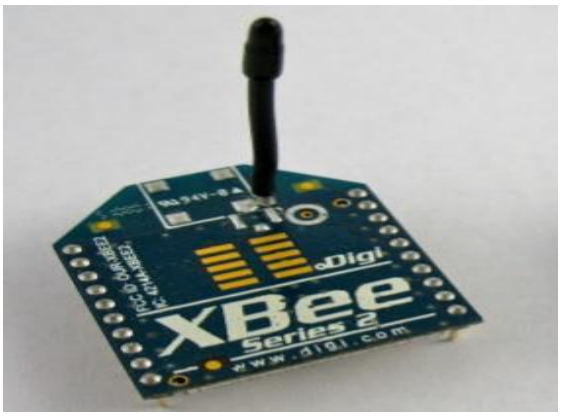


Figure 5.3 : Xbee Series 2

## Block Diagram

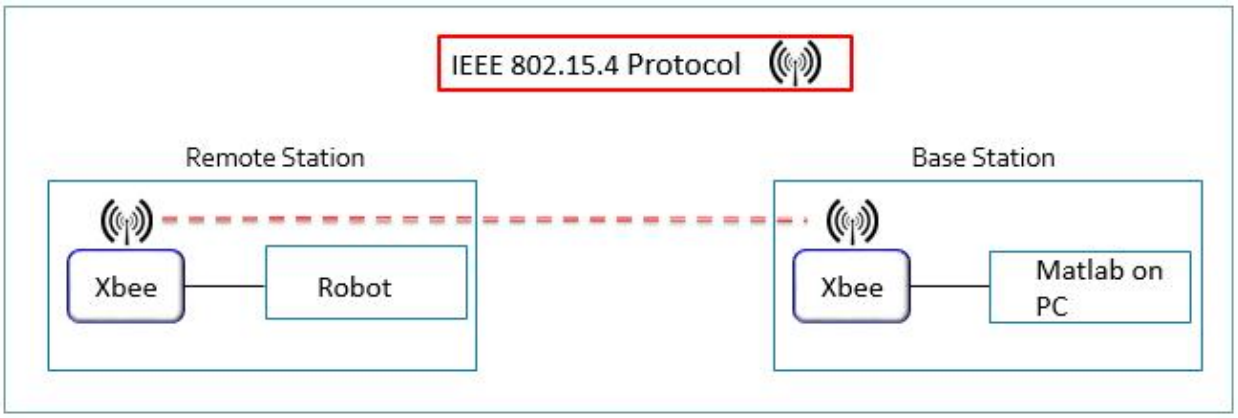



Figure 5.4 : Xbee Series 2

XBee can be used for serial connection in two different ways. They are as follows

1. AT Command Mode
2. API Frame Mode

### 4.2.1 AT Command Mode.

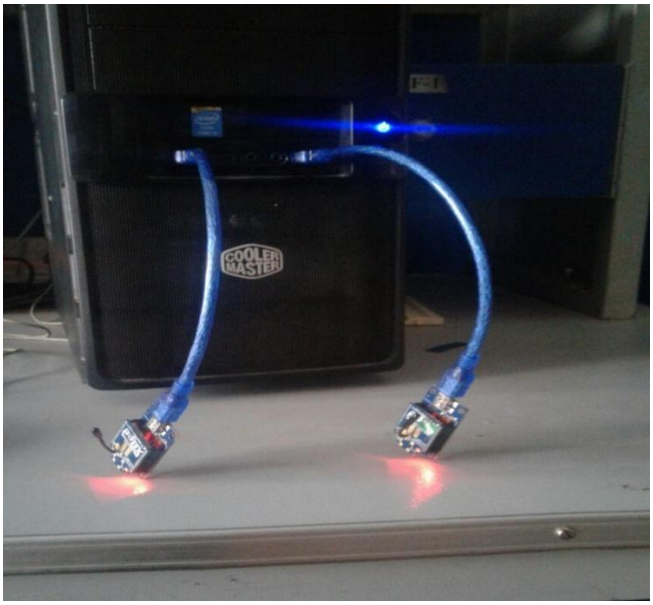


Figure 5.5: XBee setup at NDT LAB NUST-PNEC

To establish communication between XBee first we need to install firmware on Xbee modules. For this purpose "XCTU" is a used which is a free multi-platform application designed to enable developers to interact with Digi RF modules through a simple-to-use graphical interface.

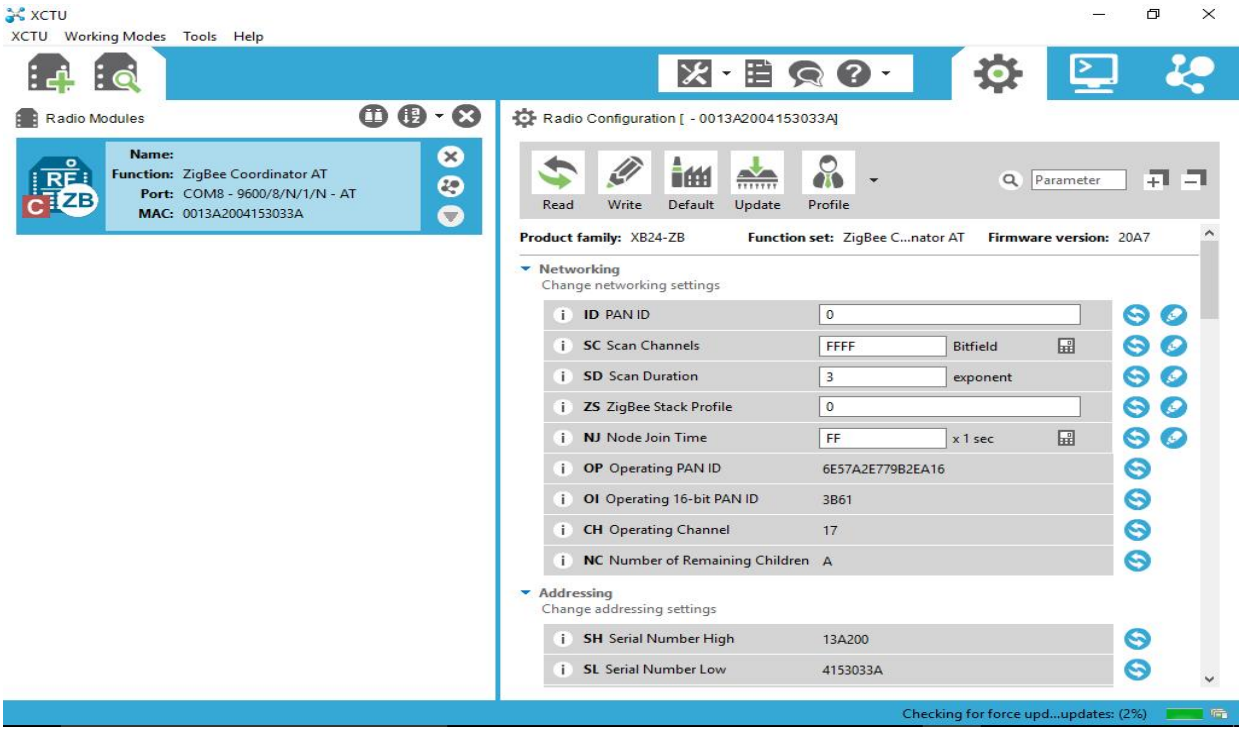



Figure 5.6: XCTU GUI

Remote node is programed with “Router AT” firmware and base station node is programed with “Coordinator AT” firmware. A Matlab script is developed to send data from Coordinator to Router. For our project we need to send the speed in our data. The data can be viewed from the serial port of the Router. We used “TeraTerm” a serial port terminal software to view the packets.

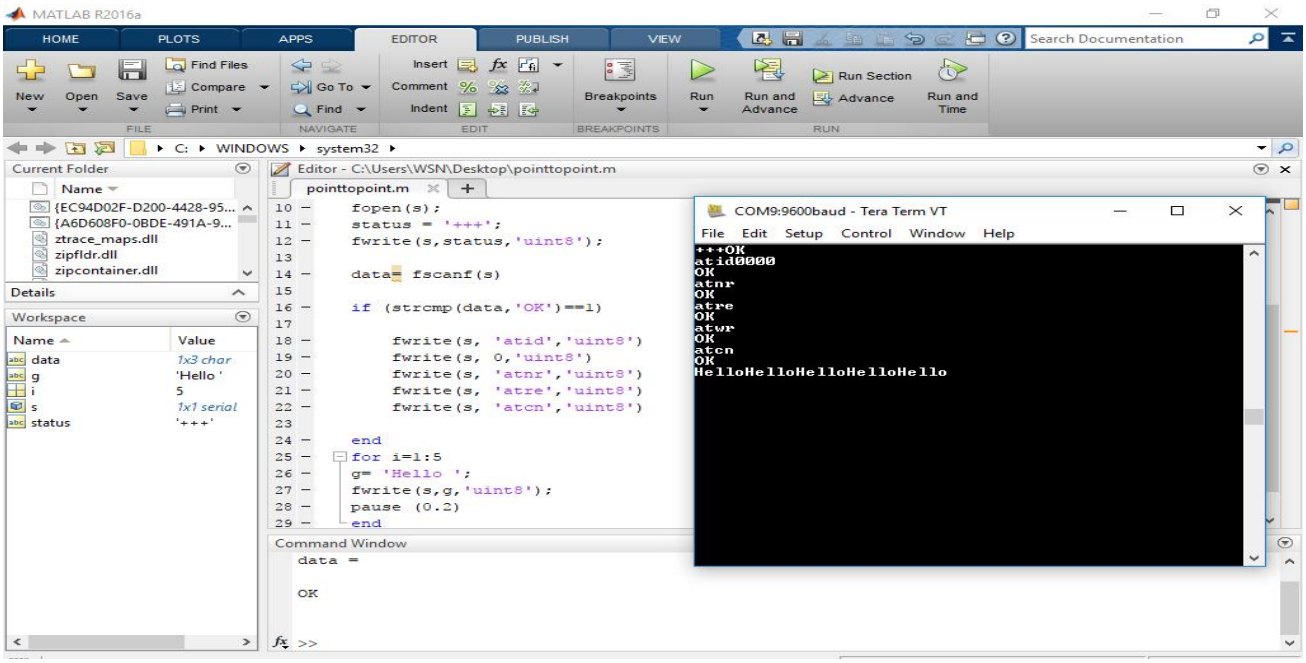



Figure 5.7: Matlab script is run at coordinators and TeraTerm is used to show that router has received packets

The Coordinator sends “Hello” packet byte by byte to the Router attached at other USB port. Initially we need to make sure that both the XBees does have the save PAN ID. We set PAN ID to “0000” because if an XBee is reset at any stage it uses 0000 as its default PANID. “ATID” command is used to set the PANID.

A router will normally find the coordinator on the same PAN and join itself to the network. Occasionally this won’t happen properly. So we forced each radio to rebuild its networking setup from scratch by issuing a network reset with “ATNR”. We had done this on both radios.

Sometimes setting both radios back to factory defaults and reconfiguring them will flush out a bad setting that was left over from a previous setup. The “ATRE” command will wipe out radio’s custom configuration and leave the firmware set cleanly to factory defaults. Follow it with the “ATWR” command to write those defaults to the firmware, then go back to the configuration steps and try putting in your settings again. ATCN is used to make the radio into transmit mode

### 4.2.2 API-frames Mode

A more robust solution than AT Command as it follows a protocol. The frame consists of “Start byte”, “Sender and receiver Address”, “Data” and “checksum”. When a frame is received by any remote station it returns an acknowledgement packet. This acknowledgment confirms the packet delivery.

For this remote node is programed with “Router AT” firmware and base station node is programed with “Coordinator API” firmware. We tested API frame mode in two different setting. The first setting was API-frame with broadcast packet and API-frame with node specific packet. The broadcast can work up to maximum of four Xbee module. It works by sending a packet at least four times. API frame with node specific packet sends packet to the designated node in the network. This is done by selecting the address of the robot in the packet. In the figure below a Matlab script is built to send API frames.

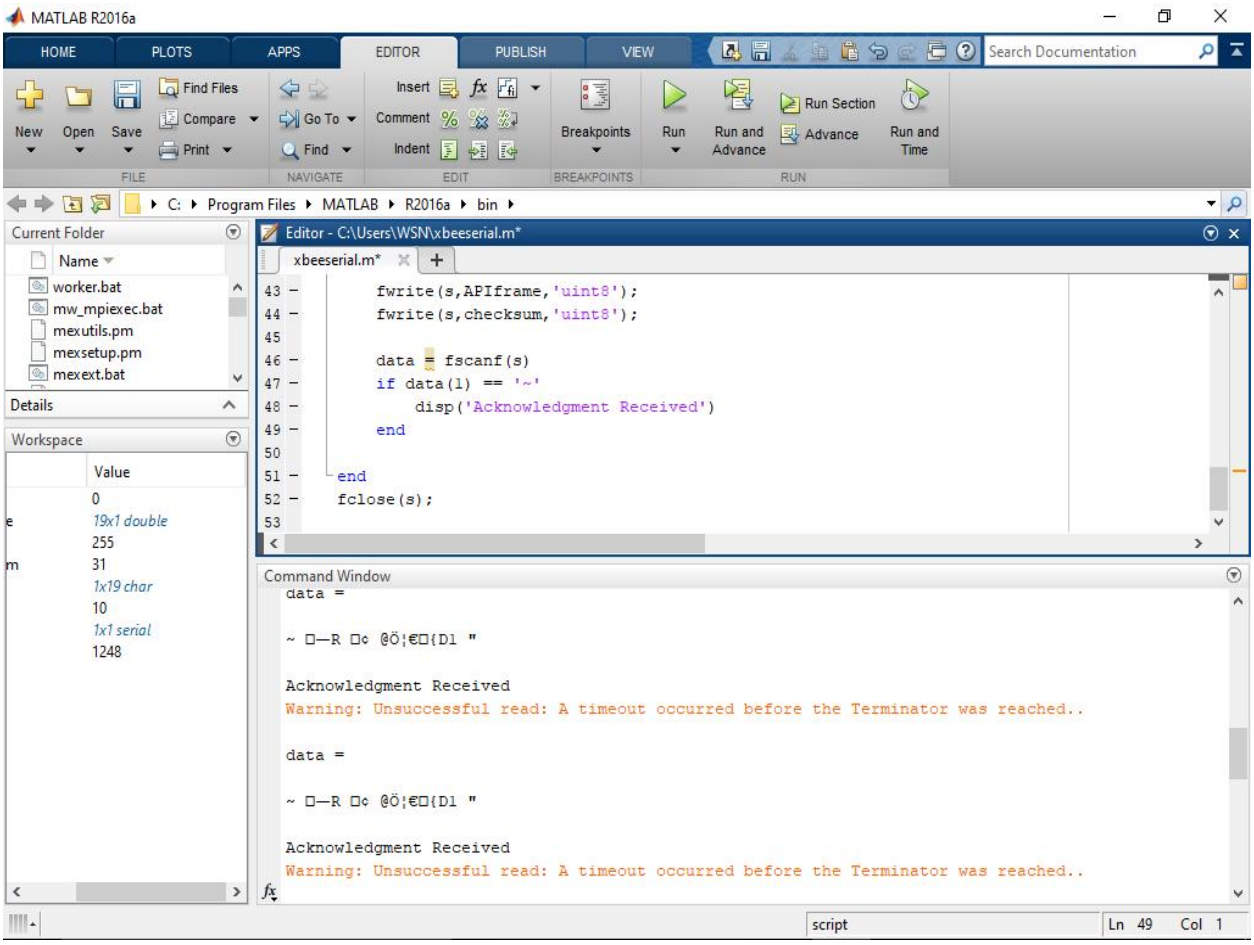



Figure 5.8: Matlab Command Windows show the acknowledgement packet

Viewing a stream of API frames displayed as ASCII characters in a terminal program will look something that is shown in the command window of the Matlab

Note that repeating tilde (~) character. It is the ASCII equivalent of 0x7E, the start byte, and is an indication that rather than seeing garbage, you are seeing good data acknowledgement.

After that we used XCTU software is used to generate frames and to check the robustness and latency

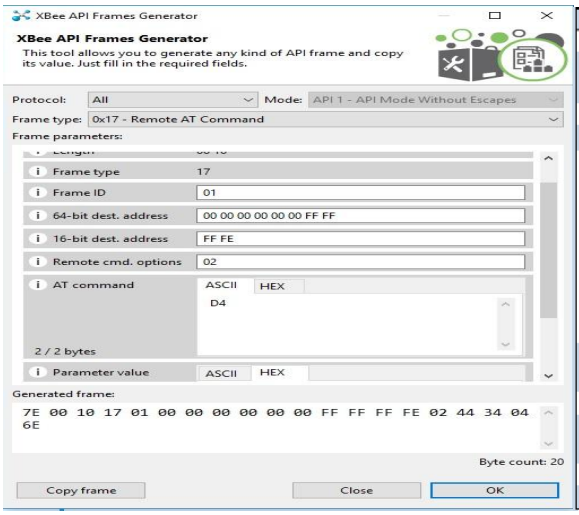



Figure 5.9: Frame generator in XTCU

| Byte | Example | Description |
|---|---|---|
| 0 | 0x7e | Start byte – Indicates beginning of data frame |
| 1 | 0x00 | Length – Number of bytes (ChecksumByte# – 1 – 2) |
| 2 | 0x10 | |
| 3 | 0x17 | Frame type - 0x17 means this is a AT command Request |
| 4 | 0x52 | Frame ID – Command sequence number |
| 5 | 0x00 | 64-bit Destination Address (Serial Number) |
| 6 | 0x13 | MSB is byte 5, LSB is byte 12 |
| 7 | 0xA2 | |
| 8 | 0x00 | 0x0000000000000000 = Coordinator |
| 9 | 0x40 | 0x000000000000FFFF = Broadcast |
| 10 | 0x77 | |
| 11 | 0x9C | |
| 12 | 0x49 | |
| 13 | 0xFF | Destination Network Address |
| 14 | 0xFE | (Set to 0xFFFE to send a broadcast) |
| 15 | 0x02 | Remote command options (set to 0x02 to apply changes) |
| 16 | 0x44 (D) | AT Command Name (Two ASCII characters) |
| 17 | 0x02 (2) | |
| 18 | 0x04 | Command Parameter (queries if not present) |
| 19 | 0XF5 | Checksum |

Figure 5.10: Example of frame

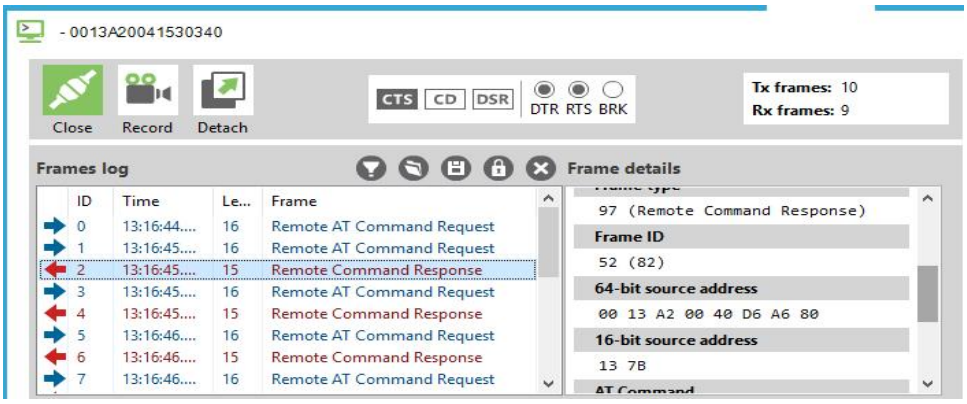



Figure 5.11: Console window showing sent packets and received acknowledgements

## 4.3 Bottlenecks faced in communication using XBee Modules

### 4.3.1 AT Command Mode.

In AT Command Mode neither there is acknowledgment nor any packet damage detection system. So therefore we cannot verify that the data reached to its destination correctly. Therefore it is not a robust solution.

### 4.3.2 API-frame Mode

In API-frame with broadcast packet occasionally a huge delay in acknowledgement was observed and therefore it is also not suitable to use this method specially in extreme conditions where a single second would be of extreme value.

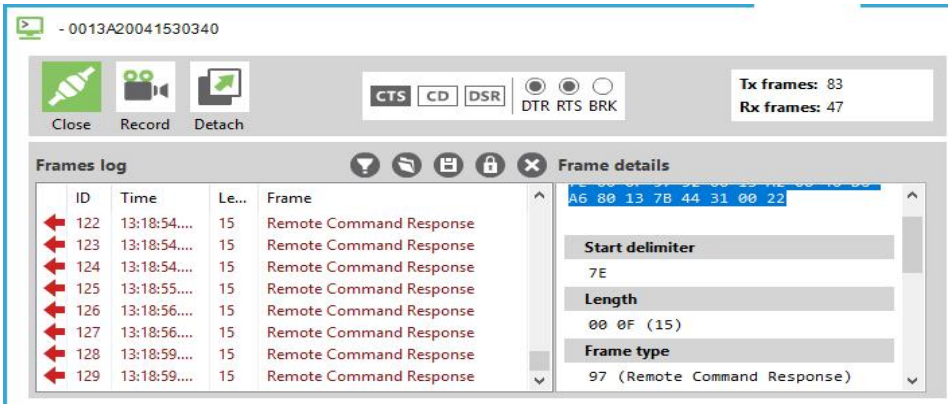



Figure 5.12: Console window showing sending of packet and receiving of acknowledgement

On the top-right side of the Console Window note the TX & RX frames. TX frames are greater than RX frames. This is because the sometimes when the Coordinator sends the frame to the Router and the Buffer is already full of frames and therefore it needs to discard that frame.

## 4.4 Suggested configuration

We tested different configurations and we can conclude that for our project the most suitable configuration would be API-frame Mode with node-specific packet i.e. unicast. The reason for choosing this method is because the API-frame with broadcast setting takes greater time as it first sends a packet and waits to avoid any collision with the previous packet.

## 4.5 Receiving speed commands on Zumo 32U4

The Zumo robot allows PWM control of each of the left and right motors by setting an integer speed value between -400 and +400. For this purpose, the on-board Arduino is programmed in C to wait for data on its serial port and issue speed commands to the motors accordingly. The serial lines of robot are usually connected to the LCD header for displaying debugging data. By removing the LCD, the header can be used to supply power to and communicate with an Xbee module mounted on each robot. A direct connection between the Rx pin of the robot board and Xbee breakout board is sufficient for reading data received wirelessly by the Xbee module. The pin connections are shown in the following figure.

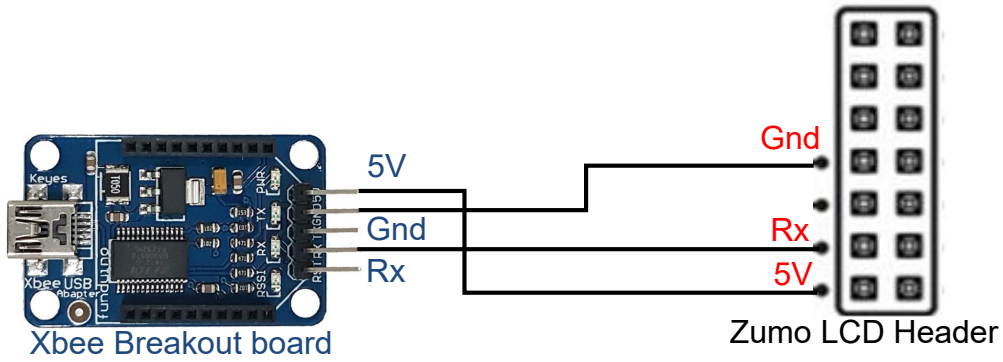



Figure 5.13: Connection of Zigbee module to

With connections hardwired as shown, an API frame sent via the coordinator at the PC base station generated by MATLAB is picked up by the desired Xbee module. The module extracts the payload which are 2 bytes consisting of left and motor speeds and transmits it to the Arduino serial port. The Arduino on the Zumo robot thus infinitely waits for 2 bytes in its serial buffer followed by a termination character (NULL) and sets speeds of the motors by scaling up the numbers from -128 to 127 to the -400 to +400 range. The expected update rate of speed is 5 Hz, therefore, the robot comes to a halt if no command is received from the Xbee module for more than 1 second.

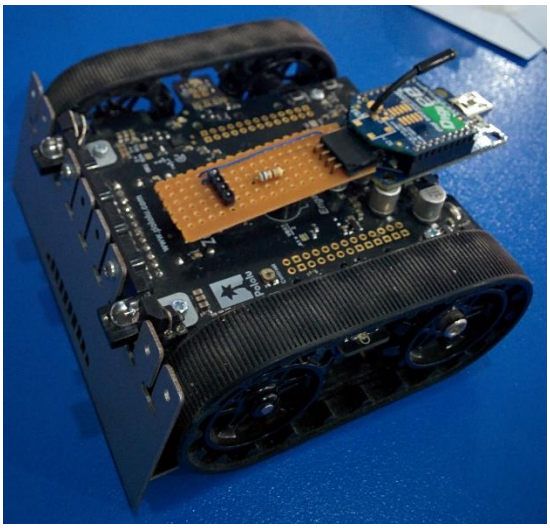


Figure 5.14: Zumo mounted with Xbee module for remote control

### 4.6 Further steps

The robot has been shown to be controlled remotely over the wireless network from the base station. Using feedback from the camera over the test area, a “go-to-goal” test will first be carried out in which the robot will navigate to given target coordinates in the test area. Once multiple robots are remotely running, formation algorithms can be tested in a feedback loop.

# Chapter 5. Improvements in the design of the mechanical structure of IMR

## 5.1 Initial design of the IMR

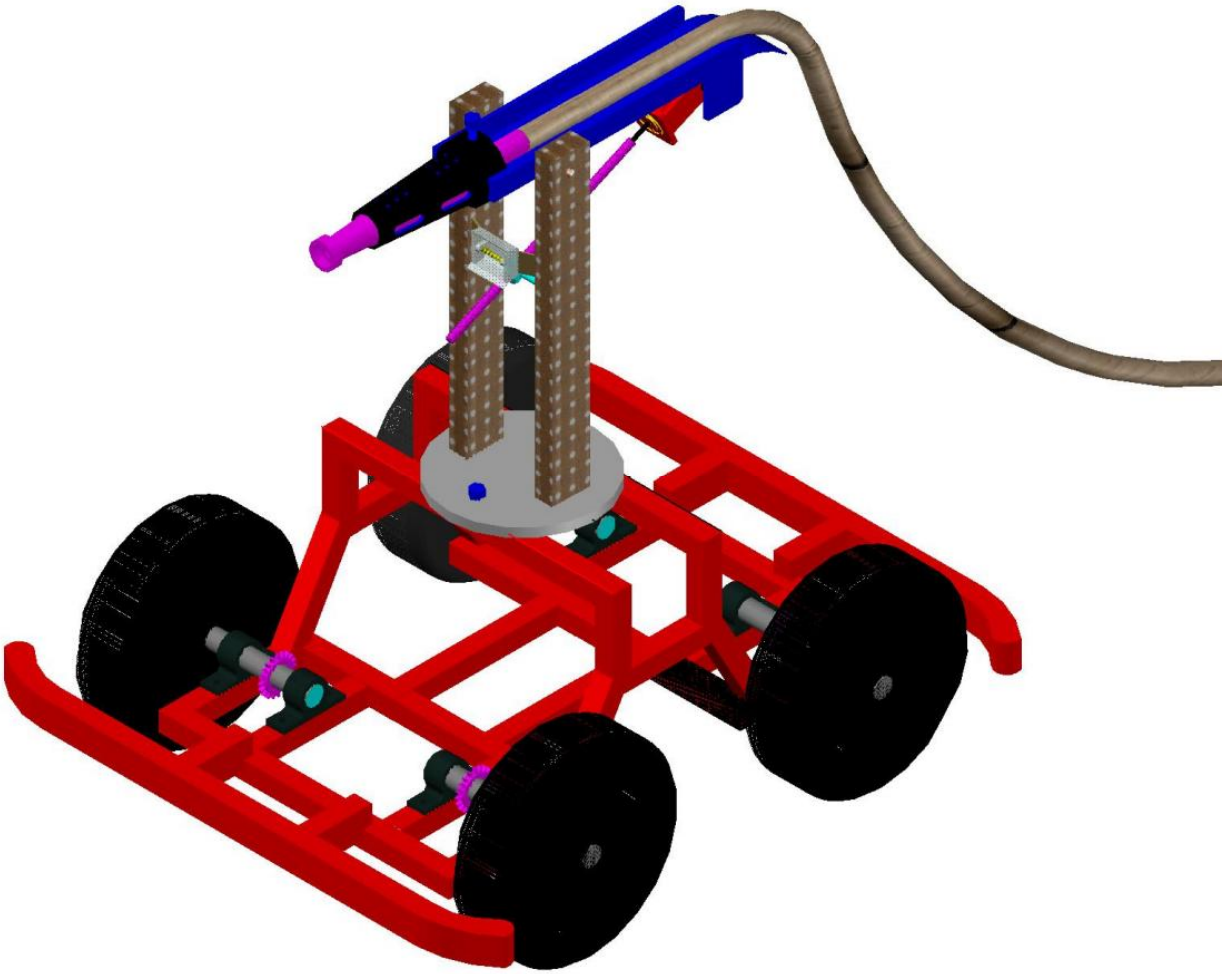
We have designed two different types of mobile robots.

a) Wheel based robot
b) Tank type robots

After many improvement cycles the design for the wheel based robot has been finalized. The robot has been fabricated and tested with fire hose. The results are quite satisfactory.

## 5.2 Wheel based robots with fire hose

### 5.2.1 Design of wheel based robot with fire hose in AutoCad



### 5.2.2 Wheel based robot with fire hose

## 5.3 Problems and challenges faced in the initial porotype

The tank type robot was initially designed with a rubber belt. The robot was tested in the fire test area. The robot has some maneuvering issues. The belt is quite weak to cater for the load it has to bear. The belt is not a good option when the robot has to be tested in high temperature environments and also it does not have a long life.

## 5.4 Proposed modifications in the design of the tank based IMR

In the  modified tank based design we have used link chain in place of rubber belt. The design of the robot has also been modified. The DC motors along with the placement of these motors in the chassis has been updated. The placement and design of the fire hose has been improved. In the initial design of fire hose, that was tested with the wheel based robot, due to the high torque prodeuced at the fire hose there was some slippage problem. The height of the fire hose i.e. the koment arm has been koved near to the centre of gravity of the robot. Consequently the magnitude of the torque has been reduced.