CS-684 Final Project Report

Hexapod Navigation using WiFi RSSI

Meet Udeshi 14D070007
Prathu Baronia 14D070046
Yogesh Mahajan 14D070022

April 2018

Contents

1 Introduction **1**

2 Problem Statement **3**

3 Requirements **5**

3.1 Functional Requirements 5

3.2 Non-Functional Requirements 5

3.3 Hardware Requirements 5

3.4 Software Requirements 6

4 System Design and Working **7**

4.1 Hardware Design 7

4.1.1 Hexapod 7

4.1.2 WiFi Localization 8

4.2 Software Design 10

4.2.1 Hexapod 10

4.2.2 Control Format 10

4.3 Localization 11

4.4 Localization 12

5 Test Results **13**

5.1 Characterizing and Testing Localization Setup 14

6 Discussion of the System **17**

6.1 The Controller 17

6.2 The Power Supply 17

6.3 Localization 18

7 Future Work **19**

8 Conclusion **21**

9 References **23**

Chapter 1

Introduction

Location estimation is a common application of wireless sensor networks (WSN). Data collected within a WSN is often meaningless without information about the location in which it was collected. Therefore, the need for node position information is important. While nodes can have positions manually assigned to each device, it is more efficient if the nodes can determine their own locations since networks tend to consist of hundreds of devices, and it would be incredibly tedious to prepare each one with an individual location. In order to determine the location, first nodes must have some method to estimate distance relative to each other. Also an ever increasing number of indoor robotic usage demands cheap ans scalable localization solution.

It would be preferable to use radio propagation to estimate distance as the network is already using wireless transmissions to communicate. There are two methodologies primarily used to interpret distance from wireless transmissions: Received Signal Timing Interval (RSTI) ans Received Signal Strength Indication.

The most important factors that complicates the distance calculation from received signal is multi-path reflections and side-band interface. RSTI based systems are more immune to these effects but are more costly, difficult to implement and not feasible in indoor application where distances are relatively smaller, which leads to another method of using RSSI in indoor application in spite of the fact that RSSI based localization is very sensitive to many environmental conditions and configurations.

Chapter 2

Problem Statement

The most important objective of this project is to implement an efficient and reliable WiFi RSSI based localization system and use it to navigate the hexapod in designated area. Previous implementations have shown that RSSI could give accuracy up to 10 cm in $2m \times 2m$ area. This decides the size of test area. The body of hexapod must not be larger than the maximum accuracy error, hence limiting the size of hexapod.
Mechanics indicates that the hexapod should have minimum 18 degrees of freedom to be able to maneuver over common terrain found in indoor environment such as slope and stairs.

Chapter 3

Requirements

3.1 Functional Requirements

- Hexapod should have at least 18 degrees of freedom
- Localization system should be able to detect the location with maximum error of 10cm

3.2 Non-Functional Requirements

- Size of hexapod should be smaller than $25cm \times 25cm$
- Maximum current consumption of motors must be less than 7A
- Localization system should give accurate position within 500 ns

3.3 Hardware Requirements

Component	**Description**	**Count**
Raspberry Pi 3	Main controller for controlling servos and determining location for trilateration	1
Arduino	Controlling beacons	4
Xbee	4 beacons and 1 receiver	5
Servos	Locomotion of the hexapod	18
5V LiPo Battery	Power up the hexapod	1
SPI Servo Controller	Require total 18 channels	2

3.4 Software Requirements

- **Adafruit Servo Controller Library**: Enables easy control of servo controller using I2C interface on RPi3. The library supports 16 channel per driver
- **XBEE Python Library**: RPi3 uses serial communication library to get data from the receiver Xbee.
- **Arduino XBee Library**: Beacons use Arduino compatible version of XBee library to broadcast messages.

Chapter 4

System Design and Working

4.1 Hardware Design

Hardware design has two prime sections, the Hexapod and localization hardware. Since RPi3 is quite sufficient to handle both work loads, we have common controller for both of them. Now we will discuss both aspects in details in following subsection.

4.1.1 Hexapod

Designing complete Hexapod is a challenging task and requires mechanical design expertise. Due to lack of time and knowledge of mechanical design we have decided to use an open source hexapod structural design. We have chosen Archrobotic's Hexy project due to their good support and assistance. The Hexy uses 18 9MG servo motors and is laser cut from an 6mm acrylic sheet. servo motors are controlled by RPi using two servo controllers. Each servo controller handles 9 servos. Servo controller is interfaced with RPi using I2C bus with fixed addressing mode. The whole structure is powered via 5V LiPo battery mounted on the bot itself.

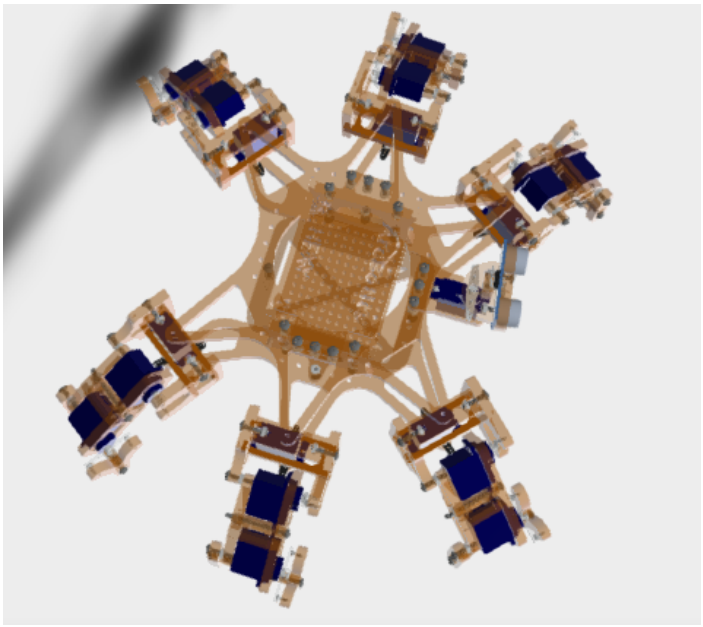

Figure 4.1: 3D Model of Hexapod

4.1.2 WiFi Localization

Why XBee?

The radio module selected was the XBee series of wireless transceivers. As XBee has several modules, such as WiFi, ZigBee, and 800 MHz ZigBee implementations, along with the XBee footprint becoming a standard with manufacturers, the XBee was inexpensive and versatile enough to meet the specifications desired. The ZigBee series 2 modules were chosen for their low-cost and minimal power requirements. They also operate in the standard 2.4 GHz frequency range, which means that the hardware could eventually build off of existing infrastructures as the XBee ZigBee uses the same protocol, IEEE 802.15.4, as many wireless routers. This particular module also uses directional antennas instead on chip or unidirectional antennas. This feature is important as unidirectional antennas are extremely expensive, and chip antennas have poor propagation qualities.
The XBee also keep the same packet structure between the different modules, so a library could be created to allow the user to easily change wireless modules without heavily modifying the application code. The XBee also provides access to RSSI values, which can be used to localize each sensor node through triangulation techniques. This can be especially useful within a swarm of robots providing a relative distance between each other and sharing mapping information.

XBee Configuration

XBee modules are available in several different models, but the most common are the series 1 and series 2. The series 1 implements the IEEE 802.15.4 standard, which covers the MAC and Physical Layers of the OSI networking model. As a result, the series 1 only perform peer to peer communications and cannot support multihop alone because of the lack of a networking protocol. The series 2 implement the complete Zigbee protocol, which supports multihop communication. To cover a large area, multihop communication is preferred, so XBee series 2 is used.
All XBee modules are serially connected to the corresponding controller via UART interface as we don't need high data rate for localization application. One can shift to other interfaces according to the application requirement.
The XBee has two modes of operation: Transparency mode and API mode. Transparency mode allows the XBee to serve as a "cable replacement", where all packets are broadcasts to all modules on the network. API mode requires the user to create specific packets for interfacing with module and sending messages throughout the network. The API mode provides many features such as RSSI value as a part of received packet, hence eliminating the need of custom hardware to calculate RSSI. So we have configured XBees in API mode.

Beacon Placement

For this project we have decided to use $1.5m \times 1.5m$ arena with one beacon at each corner. Beacon XBees are configured in boost mode to increase the signal strength. The Hexapod is equipped with one receiver XBee which sends received packets to the RPi.

4.2 Software Design

Overall software flow of the project can be summarized using following flowchart

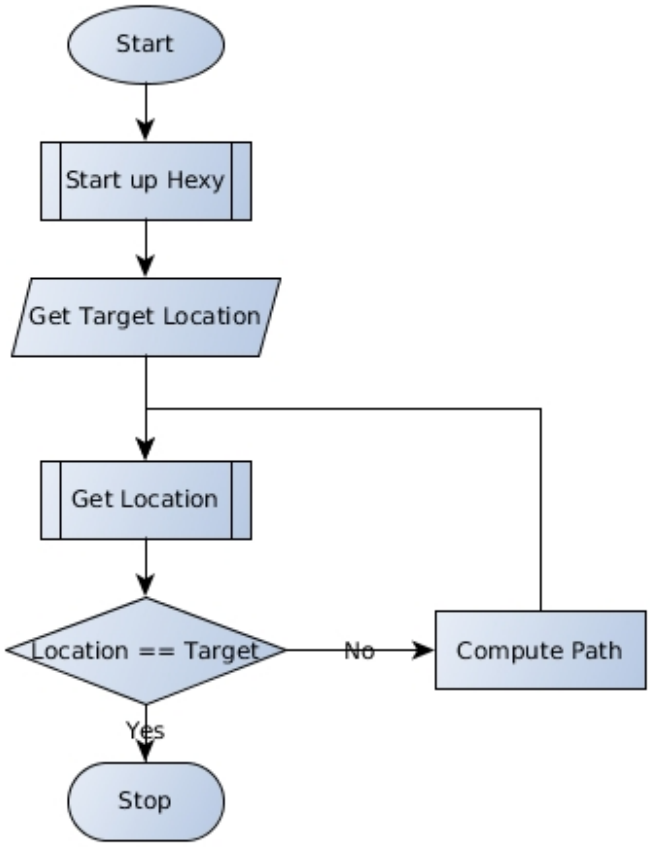

Figure 4.2: Syatem Control Flow

4.2.1 Hexapod

We have implemented a python library based on Archrobotic's source code to control the hexapod using RPi. The library provides high level interface functions to control the hexapod.

4.2.2 Control Format

Each move of the hexapod can be visualized as series of servo movements in sync. Most of the moves cam be broken down into series of basic movements of each leg. For each leg we have provided a function to reposition it. Angular offsets of servo motors can be configured using code snippet in *core.py*.

```
    """ joint_key convention:
    R - right, L - left
    F - front, M - middle, B - back
    H - hip, K - knee, A - Ankle
    key : (channel, minimum_pulse_length, maximum_pulse_length) """

joint_properties = {

    'LFH': (0, 305, 485), 'LFK': (1, 200, 480), 'LFA': (2, 200, 640),
    'LMH': (3, 245, 398), 'LMK': (4, 290, 610), 'LMA': (5, 200, 500),
    'LBH': (6, 265, 455), 'LBK': (7, 230, 520), 'LBA': (8, 155, 615),

    'RFH': (9, 180, 450), 'RFK':(10, 320, 530), 'RFA':(11, 200, 700),
    'RMH':(12, 470, 600), 'RMK':(13, 320, 550), 'RMA':(14, 200, 640),
    'RBH':(15, 400, 660), 'RBK':(16, 250, 460), 'RBA':(17, 200, 630),

    'N': (18, 150, 650)
}
```

4.3 Localization

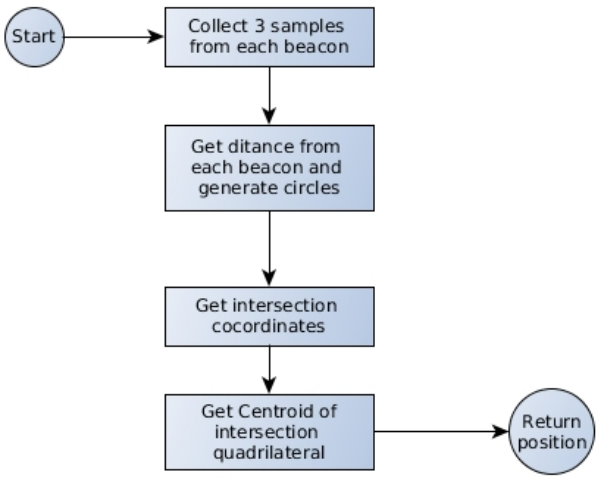

Figure 4.3: Localization Control Flow

Each XBee beacon has is own first order RSSI to distance mapping function. These functions explicitly have to found using profiling for each XBee. One profiled this map remains more or less constant for the XBee transmitter receiver pair irrespective of other pairs in vicinity. So

we can increase the number of number of nodes to cover more area. Following figures shows the distance vs RSSI plots for one of the 4 used XBee beacons. Each beacon broadcasts a packet containing its ID after **every 200 ms**. The receiver collects at least 3 packets from each beacon and then decodes the location using average RSSI of received packets.

4.4 Localization

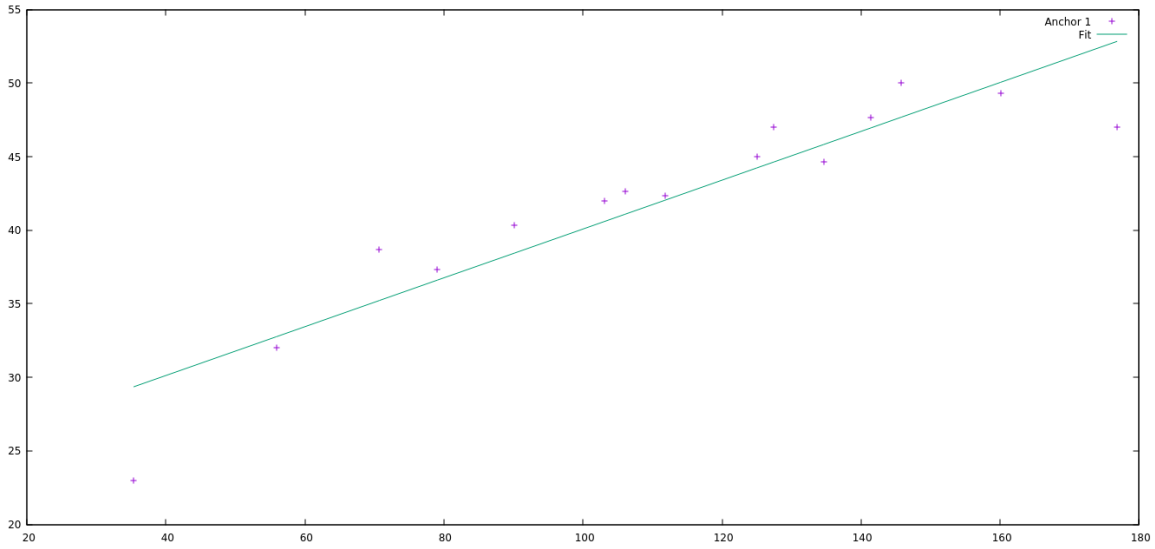

Figure 4.4: Distance Vs RSSI for Anchor 1

Chapter 5

Test Results

Following picture shows the test setup used for testing the system. The square arena has dimensions of $1.5m \times 1.5m$ and was divided in small square regions of $25cm \times 25cm$. The four beacons are placed at each of the corners.

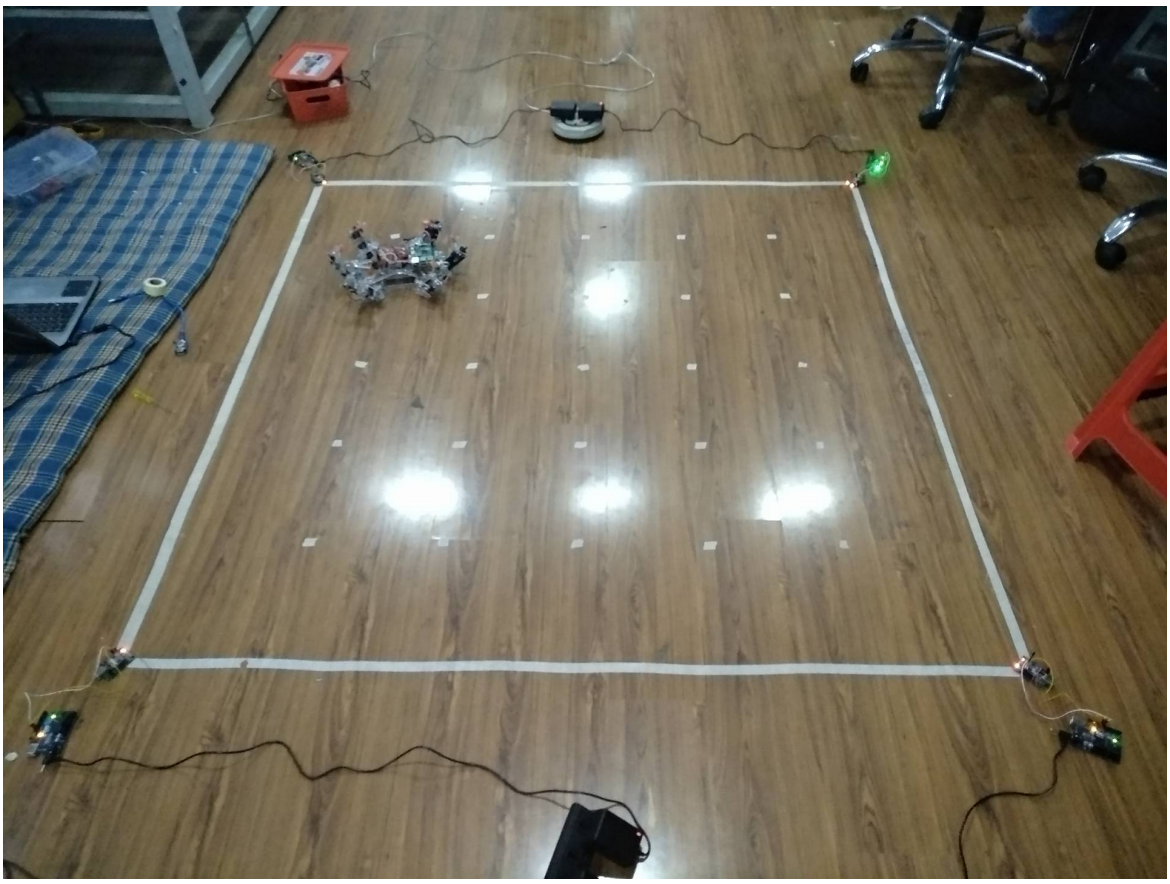

Figure 5.1: Test Setup

The test video of hexapod can be found here:

- https://youtu.be/1PUA8-5pAxc
- https://youtu.be/FjIFfGolqqk

5.1 Characterizing and Testing Localization Setup

We have characterized every beacon using RSSI and distance from the beacon in the given arena to get the mapping function corresponding to each node. Following figures shows the profiles we got. These transfer functions are follows linear function for distances in given arena. For larger areas we may have to use exponential model of RRSI.

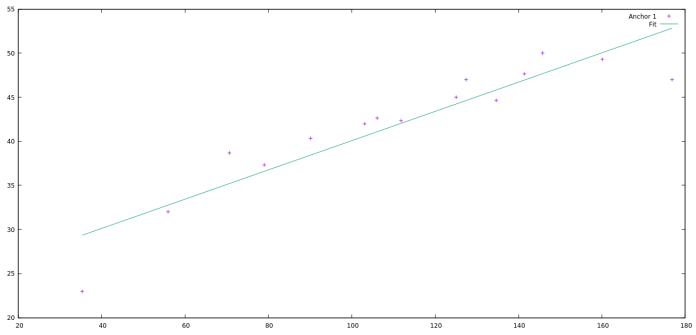

Figure 5.2: Beacon 1

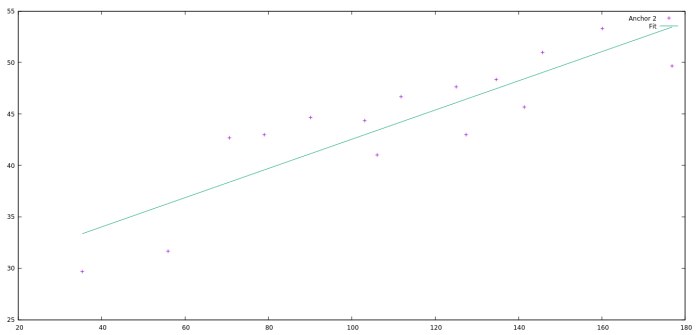

Figure 5.3: Beacon 2

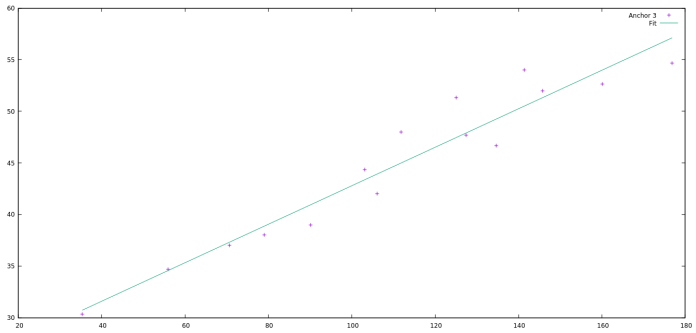

Figure 5.4: Beacon 3

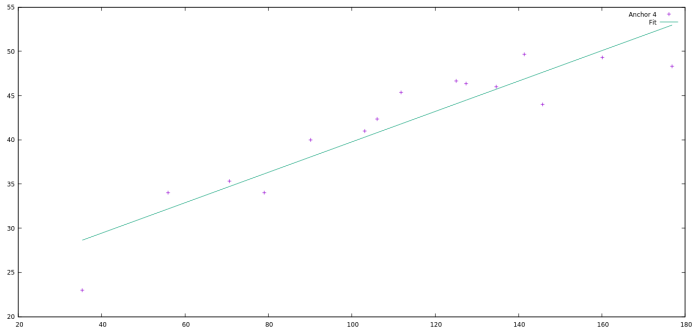

Figure 5.5: Beacon 4

Following screenshot shows the received location for receiver kept at coordinate $(150, 50)$.

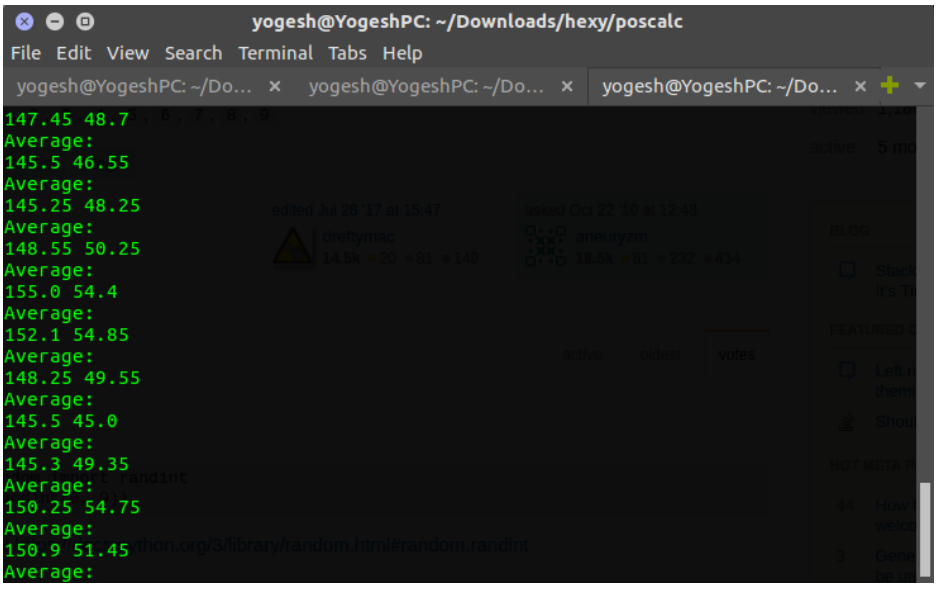

Figure 5.6: Received location averaged over 3 samples. error is within **5cm** limit

Chapter 6

Discussion of the System

We were able to make the system work as planned but some last minute problems with the battery disappointed us. The hexapod is able to maneuver all the programmed moves as planned and localization setup is able to locate with the maximum error of 10 cm.

6.1 The Controller

Initially we have planned to use Tiva-Cas the main controller for our project. Because of following limitations/shortcomings of the Tiva-C board, we switched to more flexible and versatile controller RPi

- Tiva C has limited floating point computation power. Running trilateration and hexapod control loop simultaneously is difficult on Tiva-C. RPi helped us by eliminating the need of two separate controllers.
- Implementing complete hexapod controller was time consuming task. Off the self python library for servo controller helped us a lot by saving valuable time.
- RPi's wireless connectivity helped us by avoiding to have long cables between hexapod and computer used for debugging and testing.

6.2 The Power Supply

We had some hard time finding the correct battery for hexapod because of smaller available volume and larger current requirements. Also we needed output voltage of around 5V. Initially we planned to have a voltage converter and use 11.1V LiPo battery. Because of lack of time and resources we couldn't do it in time and ended up using 6V battery which went

bad just before the demonstration and we couldn't complete the demonstration as planned. We have tried off the shelf buck converter too. but it was limited by the current requirements of hexapod which exceeds little more than 4A depending on the load and surface.
We have added schematic and layout of the high current output 9-12V to 5V voltage converter in our submission.

6.3 Localization

Initially we were going to use exponential model for profiling the RSSI curve, but we experimentally found out that linear model was more accurate and we switched to it. We have also tried Linear Regression based model to determine the location. It seems to be promising but needs some more effort to choose correct model.

Chapter 7

Future Work

- Hexapods control structure is divided into sequence of movements of individual legs one by one, which produces jittered transition. Smoother movements can be achieved by simultaneously controlling all the servos. This can be done using the method discussed in Archrobotic's Hexy controller.
- Power supply is one of the major concern for the hexapod. Using the voltage converter with LiPo battery would help to reduce weight too.
- Currently all the parts of the hexapod are in different STL file and hence takes large time to laser-cut. Total time can be reduced by consolidating all the parts in one file. This would reduce the cutting cost too.
- RPi is sort of an overkill for the task. An efficient and more simpler controller can be used but it needs lot of time to optimize.
- More number of nodes can be used to increase the size of arena and to cover up the shortcomings of obstacles.
- RSSI Vs distance follows exponential decay for smaller distances and then follows linear model. More accurate model should include both the cases. Linear regression based model may be more effective.

Chapter 8

Conclusion

- We have successfully realized and controlled a relatively cheap hexapod which can be useful in academia.
- An energy efficient and simple RSSI based system implemented in this project is very effective for indoor application where location needs to pre accurate within 10cm or more.
- Implemented localization system is able to detect location with within accuracy of 6cm.

Chapter 9

References

- Reference Paper: Development of a portable XBee C library and RSSI triangulation localization framework
- Reference Paper: Utilization of XBee ZigBee modules and MATLAB for RSSI localization applications
- Hexapod Design: Archrobotic's Git Repository
- Hexapod Assembly: Getting started with Hexy
- Controlling Hexapod: Archrobotic's controller