Table of Contents

[Abstract 3](#_Toc292617365)

[1 Introduction 3](#_Toc292617366)

[2 Specification 4](#_Toc292617367)

[2.1 Tracking system’s operation 4](#_Toc292617368)

[2.1.1 Front end network 4](#_Toc292617369)

[2.1.2 Back end network 4](#_Toc292617370)

[2.2 Hardware 4](#_Toc292617371)

[2.2.1 Detector 4](#_Toc292617372)

[2.2.2 Tag 4](#_Toc292617373)

[2.2.3 Proxy 4](#_Toc292617374)

[2.3 Controller 4](#_Toc292617375)

[2.4 Locating Algorithm 4](#_Toc292617376)

[2.5 Web App 4](#_Toc292617377)

[3 Test Plan 4](#_Toc292617378)

[3.1 Unit Tests 4](#_Toc292617379)

[3.2 Hardware 4](#_Toc292617380)

[3.2.1 Power supply unit 4](#_Toc292617381)

[3.2.2 Tag 4](#_Toc292617382)

[3.2.3 Detector 4](#_Toc292617383)

[3.2.4 Proxy 4](#_Toc292617384)

[3.3 Software 4](#_Toc292617385)

[3.3.1 Controller 4](#_Toc292617386)

[3.3.2 Web App 4](#_Toc292617387)

[3.3.3 Database 4](#_Toc292617388)

[3.4 System Tests 5](#_Toc292617389)

[3.4.1 Test case 1: 5](#_Toc292617390)

[3.4.2 Test Case 2: 5](#_Toc292617391)

[3.4.3 Test Case 3: 5](#_Toc292617392)

[3.4.4 Test Case 4: 5](#_Toc292617393)

[3.5 Acceptance Tests 5](#_Toc292617394)

[3.5.1 Test case 1: 5](#_Toc292617395)

[3.6 Final Results 5](#_Toc292617396)

[3.6.1 Device dimensions 5](#_Toc292617397)

[3.6.2 Power consumption 5](#_Toc292617398)

[3.6.3 Accuracy 5](#_Toc292617399)

[4 Conclusions 6](#_Toc292617400)

[4.1 What we learned 6](#_Toc292617401)

[4.2 Good Decisions 6](#_Toc292617402)

[4.3 Bad Decisions 6](#_Toc292617403)

[4.4 Meeting our Expectations 6](#_Toc292617404)

[4.5 Meeting our Sponsors Expectations 6](#_Toc292617405)

[4.6 Group Dynamics 6](#_Toc292617406)

[5 Appendices 6](#_Toc292617407)

[5.1 Installation 6](#_Toc292617408)

[5.1.1 Detector Deployment 6](#_Toc292617409)

[5.1.2 Tag Deployment 6](#_Toc292617410)

[5.1.3 Controller Setup 6](#_Toc292617411)

[5.1.4 Proxy Setup 6](#_Toc292617412)

[5.2 Maintenance 6](#_Toc292617413)

[5.2.1 Replacing Batteries 6](#_Toc292617414)

[5.3 User Guide 6](#_Toc292617415)

[5.3.1 Web-based application 6](#_Toc292617416)

[5.3.2 Controller application 6](#_Toc292617417)

[5.4 Useful Links 6](#_Toc292617418)

[6 References 6](#_Toc292617419)

# Abstract

TIU tracking system is a capstone project conducted by Portland State University students and sponsored by Intel Corp. The system will be used to track locations of Test Interface Units (TIUs) in the Intel Validation Lab.

In this paper, we propose a general architecture for the tracking system. The main goal of the architecture aims to achieve a low-cost, yet effective tracking system. The infrastructure of the tracking system is built upon RF transceivers which communicate wirelessly in a mesh network. The asset tags will be attached to tracked devices (TIUs). These asset tags will broadcast messages to detectors. Messages that contain received signal strength indication (RSSI) values will be routed though the mesh network to a server. The server will analyze the received data, calculate the locations, and store them in the database. A web application will then be able to access the database and display location information in an interactive 2D map.

The primary figures of merit that were used in the decision making process include accuracy, power consumption, size, and cost. The last administrative idiom this paper conveys is a road map which will guide our team to the final objective. The means by which this is accomplished is through a tentative schedule, timeline, a list of milestones and expected deliverables.

# Introduction

Location aware computing has a bright future in the fields of personal navigation, personal security, prompt healthcare and entertainment. Furthermore, information on the physical location of mobile nodes can greatly help in urban search and rescue missions, as well as enable geographical routing in ad hoc multi-hop networks. The determination of physical location is sometimes referred to as location estimation, location identification, localization, and positioning or geolocation identification.

Chip manufacturers today face dramatic upheaval across the industry. Companies (such as Intel) need to find ways to remain profitable at a time when economic turbulence is curtailing consumer buying power. In the midst of these changes, plant managers must find a way to address the problem of improving processes while becoming more cost-effective.

As part of this effort, Intel wants to automate tracking of the Test Interface Units (TIU) at different locations of the Intel lab, with the intention of wirelessly locating a TIU’s physical position at all times. Given the size and complexity of Intel validation lab, locating Test Interface Units (TIU) can be a tedious task since the test equipments have to be broken down to identify the TIU it is using. Therefore, an effective TIU tracking system rises as an urgent need.

By tracking location of the TIU’s physical position in the lab, Intel can minimize downtime and decrease labor costs caused by the size and complexity of Intel validation lab. This solution enables:

* Increased equipment utilization
* Decreased loss and theft
* Reduced costs
* Faster production
* Avoid time wasted on manual searches and inventory checks.

This capstone project will be a prototype system that can wirelessly track the location of at least 10 TIUs in an indoor environment. The duration of the project will last 6 months.

# Specification

## System Overview

Figure 3-1 shows a network of detectors that are geometrically located in a grid array. The density of the grid array will primarily be a ratio of accuracy and cost. For the scope of this project, the reasonable number of detectors would be from five to seven. The location of the proxy must be within range of at least one detector.

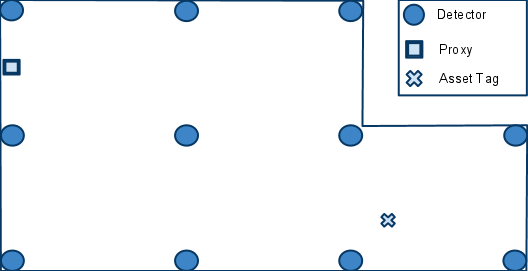


Figure : A hypothetical room with detectors positioned in a grid array. Also shown is an arbitrarily placed proxy.

There are two distinct networks which compose this system: the front-end network, which includes all the physical nodes, and the back-end network, which includes all the logical nodes (Figure 3-2). The proxy is the gateway between the two networks.

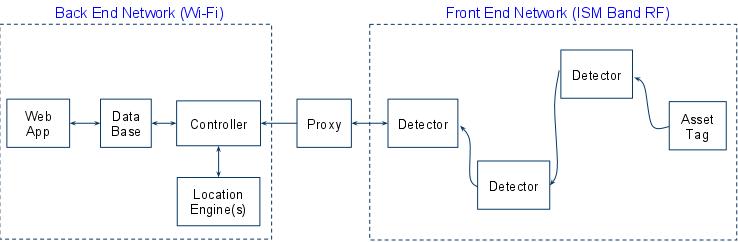


Figure : Front-end and back-end network topologies.

There are three important aspects to the communications that take place in the front-end network.

* Each asset tag will broadcast a message periodically. Neighboring detectors will detect the broadcast, and determine the RSSI of the broadcasting asset tag.
* When a detector determines the RSSI of a broadcasting asset tag, it must then transmit this data towards the proxy using a routing table. The routing table will be configured in such a way as to ensure the data moves in a direction that is physically towards the proxy.
* The locating application will periodically receive bursts of RSSI information from various detectors. Since it has no knowledge of how many detectors actually detected a broadcasting asset tag, it will wait a certain amount of time that assumes all the RSSI information from the various detectors has finished coming in.

Now we will further discuss some finer points of the second aspect. Figure 3-3 shows the paths taken as a result of an asset tag’s broadcasting. When an asset tag broadcasts, the nearest detectors are able to determine the corresponding RSSI values. The detectors that detected the asset tag broadcast are denoted by a connection with a green arrow. Each detector will have a routing table programmed into them. The routing table will contain information that steers the messages physically toward the proxy. In the example shown in Figure 3-3, the routes taken are in red. The blue dash paths are routes that exist but are not taken.

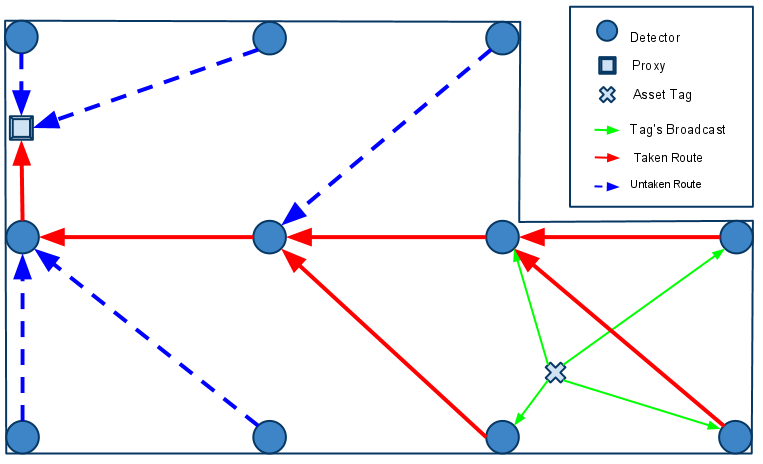


Figure : An example of the routing paths as a result of an asset tag’s broadcasting.

## Software

### Front-end Software

The front-end software consists of the firmware for the detectors, and the asset tags.

For a detector to determine the RSSI of an asset tag, the asset tag must broadcast a message. The message has six fields, as shown in Figure 3-4. The way the fields are used depends on the origin of the message. There are two possible origins of a message. The first type of origin is an asset tag. A message can be identified as coming from an asset tag by examining the DID, SID, and RSSI value fields. If all three fields contain the value zero, then the message originates from an asset tag. Otherwise, the message originates from a detector.



Figure : Message format.

When receiving a message whose DID is not zero, a detector will do four things:

1. Replace the DID of the message with a DID contained in its routing table
2. Replace the SID of the message with its own ID
3. Replace the RSSI value with that inferred from the reception of the asset tag broadcast
4. Rebroadcast the message

An example of this message passing sequence is show in Figure 5. **(Need modified)**

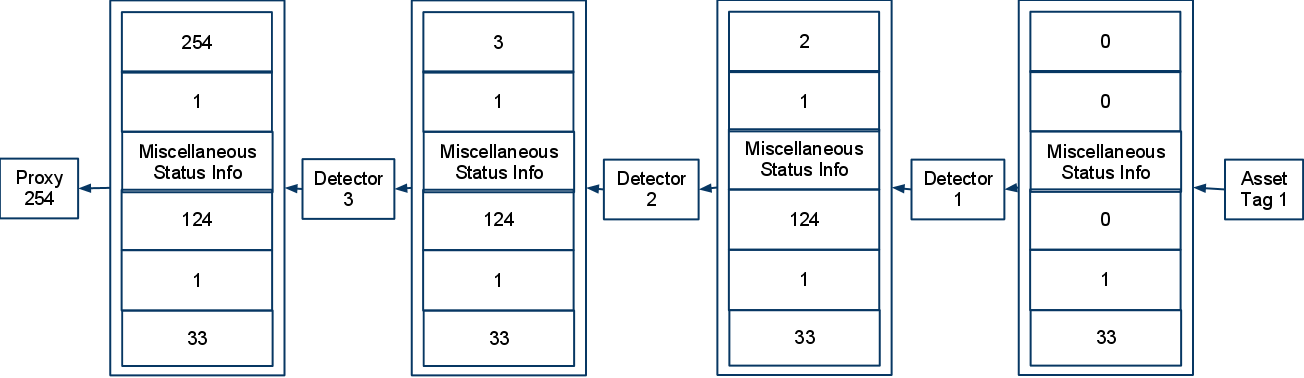


Figure : A example of a message originating from a asset tag, and being relayed to a proxy.

### Back-end Software

The back-end network, composed exclusively of logical nodes, will rely on normal internet communication. This allows applications to leverage TCP/IP and many APIs that have been built upon it. Also, we gain the advantage of being able to have standard inter-nodal communication regardless of if they exist on the same computer or not. This increases the scalability of the system.

The back-end software system consists of four components:

* SQL Database
* Web Application
* Controller
* Location Engine

#### SQL Database

The SQL database is a central component that provides a link among the web application, the controller, and the location engine. Specifically, the database stores the following data:

* Location update interval
* Geometry of the tracking area
* RSSI-distance model or RF fingerprinting
* Identifications and locations of the detectors
* Identifications, locations, and battery levels of the TIUs
* Mapping table between asset tags’ ID and TIUs’ ID

We agree on using SQL since it is a standard that is easy to use, secure and scalable.

#### Web application

The web application provides an interface for users to access the location information in the database. Its functionalities are:

* Display the locations and battery levels of all tracked TIUs on an interactive 2D map
* Allow users to search for a TIU via its ID. In case the ID is not complete or unique, the application will highlight the locations of all matched TIUs
* Allow users to view location history of the TIUs
* Alert users when a TIU is brought out of bounds or when it runs out of battery

The web application consists of two components: the client side and the server side. The client side will be implemented using modern web technologies such as HTML5, JavaScript, CSS3, and AJAX. The server side could be built upon one of the two most popular web server applications, i.e. Microsoft IIS and Apache.

#### Controller

The controller application is a GUI application that allows an administrator to manage the system. It allows the administrator to

* Modify the geometry of the tracking area
* Remap asset tags’ ID to TIUs’ ID
* Add or remove devices (detectors and asset tags)

The controller also receives RSSI data from the proxy, preprocesses them before feeding them into the location engine.

#### Location Engine

The location engine receives RSSI values from the controller and uses those values to calculate the locations of the asset tags. The derived locations will then be stored in the database.  
In case of using battery to power the asset tags, this application also receives battery level reports of the asset tags from the detectors and update the data in the database. This application should be written in a programming language that supports SQL database access and network communication.

The controller and the locating applications are discussed herein as two separate applications. However, they can definitely be combined into one single application.

## Hardware

### Tag

An asset tag consists of two major parts:

* Microcontroller: ATMega328P – 32 TQFP
* RF transceiver: RFM12B-S2 Wireless Transceiver Module

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Unit | Device Name | Description |
| 1 | Microcontroller | ATMega328P – 28 DIP |  |
| 2 | RF Transceiver | RFM12B-S2 Wireless Transceiver |  |
| 3 | Voltage regulator | XC6221 |  |
| 4 | Power supply | 6V Battery (Dual 3V Coin cells) | Two 3V coin cells connected in series |
| 5 |  |  |  |

The microcontroller controls the RF transceiver. The RF transceiver is responsible for communicating information to the system.

### Detector

A detector consists of two major parts:

* Microcontroller: ATMega328P – 28 DIP
* RF transceiver

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Unit | Device Name | Description |
| 1 | Microcontroller | ATMega328P – 28 DIP |  |
| 2 | RF Transceiver | RFM12B-S2 Wireless Transceiver |  |
| 3 | Voltage regulator | XC6221 |  |
| 4 | Power supply | 9V Battery | Required 3.3V input voltage |
| 5 |  |  |  |

The microcontroller controls the RF transceiver. The RF transceiver is responsible for communicating information to the system.

### Proxy

The proxy consists of three major parts:

* Microcontroller unit
* RF transceiver unit: RFM12B-S2 Wireless Transceiver Module
* Wi-Fi transceiver: WiFly GSX Module

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Unit | Device Name | Description |
| 1 | Microcontroller | ATMega328P – 28 DIP |  |
| 2 | RF Transceiver | RFM12B-S2 Wireless Transceiver |  |
| 3 | Wi-Fi transceiver | WiFly GSX Breakout |  |
| 4 | Power Supply | 9V |  |
| 5 |  |  |  |

The microcontroller is responsible for relay data between the two transceivers.

### Parts Selection

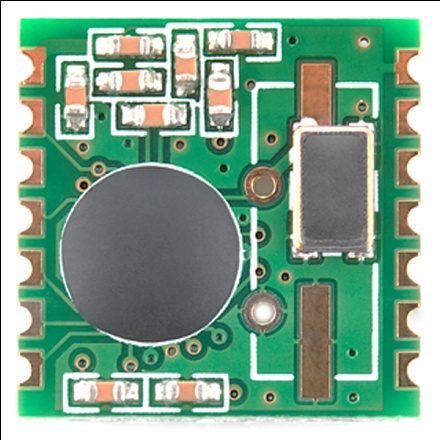


Figure : RFM12B-S2 Wireless Transceiver Module

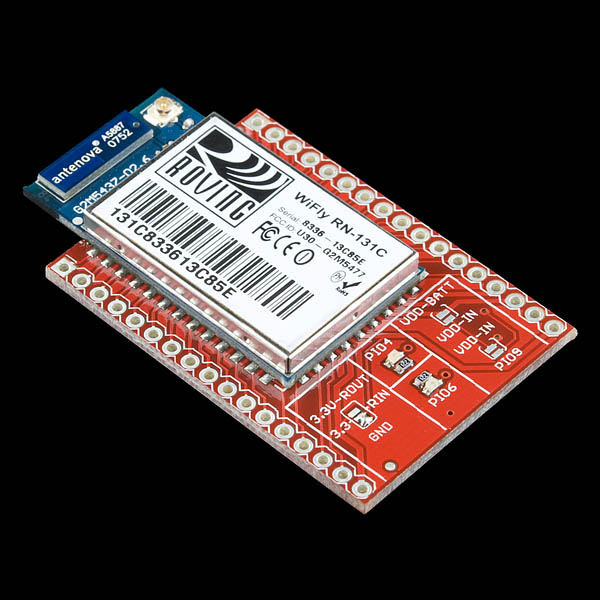


Figure : WiFly GSX Breakout

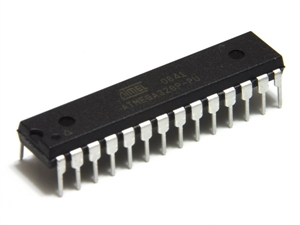


Figure : ATMega328P – 28 DIP Pinout

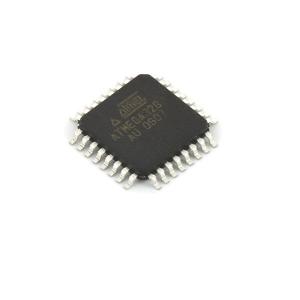


Figure : ATMega32P – 32 TQFP Pinout

Refer to datasheets provided by manufacturers for more information (See **Useful Links** section in **Appendices**).

# Test Plan

## Unit Tests

## Hardware

### Power supply unit

### Tag

### Detector

### Proxy

## Software

### Controller

### Web App

### Database

## System Tests

### Test case 1:

1 Tag, 1 Detector, Serial

### Test Case 2:

1 Tag, 1 Detector, Proxy

### Test Case 3:

1 Tag, Multiple Detectors

### Test Case 4:

Multiple Tags, Multiple Detectors, Proxy, Database

## Acceptance Tests

### Test case 1:

10 Tags, 8 Detectors, Proxy, Controller, Web App

Accuracy Requirement Met?

Power Requirement Met?

Update Interval Requirement Met?

## Final Results

### Device dimensions

### Power consumption

### Accuracy

# Conclusions

## What we learned

## Good Decisions

## Bad Decisions

## Meeting our Expectations

## Meeting our Sponsors Expectations

## Group Dynamics

# Appendices

## Installation

### Detector Deployment

### Tag Deployment

### Controller Setup

### Proxy Setup

## Maintenance

### Replacing Batteries

## User Guide

### Web-based application

### Controller application

## Useful Links

# References