**Portland State University**

**Department of Electrical and Computer Engineering**

11

**Capstone 2011**

**TIU Tracking Final Report**

**Sponsor: Intel**

**Advisor: Prof. Robert Daasch**

**Members: Dung Le**

**Tri Truong**

**Lynh Pham**

**Man Hoang**

**Daniel Ferguson**

**Version History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| 1.0 | Daniel Ferguson | 05/02/2011 |  |  | Report outline |
| 1.2 | Tri Truong  Lynh Pham | 05/27/2011 |  |  | Draft |
| 1.3 | Dung Le  Tri Truong  Lynh Pham  Daniel Ferguson | 06/05/2011 |  |  | Modify report |
| 1.4 | Man Hoang | 06/07/2011 |  |  | Added Web App  Improve formatting |
| 1.5 | Dung Le  Tri Truong  Lynh Pham  Man Hoang  Daniel Ferguson | 06/08/2011 |  |  | Final report |

**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 app 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.

Table of Contents

[1 Introduction 1](#_Toc295308346)

[2 Specification 2](#_Toc295308347)

[2.1 System Overview 2](#_Toc295308348)

[2.2 Software 4](#_Toc295308349)

[2.2.1 Front-end Software 4](#_Toc295308350)

[2.2.2 Back-end Software 11](#_Toc295308351)

[2.3 Hardware 20](#_Toc295308352)

[2.3.1 Tag 20](#_Toc295308353)

[2.3.2 Detector 21](#_Toc295308354)

[2.3.3 Proxy 23](#_Toc295308355)

[2.3.4 Parts Selection 23](#_Toc295308356)

[3 Test Plan 24](#_Toc295308357)

[3.1 Unit Tests 24](#_Toc295308358)

[3.1.1 Power Supply 24](#_Toc295308359)

[3.1.2 Tag 25](#_Toc295308360)

[3.1.3 Detector 26](#_Toc295308361)

[3.2 Integration Tests 26](#_Toc295308362)

[3.2.1 RF Network and Proxy Communication 26](#_Toc295308363)

[3.2.2 Fully Integrated System 27](#_Toc295308364)

[3.3 Acceptance Tests 28](#_Toc295308365)

[3.3.1 Tag’s Size 28](#_Toc295308366)

[3.3.2 Power 28](#_Toc295308367)

[3.3.3 Accuracy 29](#_Toc295308368)

[3.4 Final Results 31](#_Toc295308369)

[3.4.1 Device Dimensions 31](#_Toc295308370)

[3.4.2 Power Consumption 32](#_Toc295308371)

[3.4.3 Accuracy 32](#_Toc295308372)

[4 Discussions 33](#_Toc295308373)

[5 Conclusions 33](#_Toc295308374)

[5.1 What We Learned 33](#_Toc295308375)

[5.2 Good Decisions 33](#_Toc295308376)

[5.3 Bad Decisions 34](#_Toc295308377)

[5.4 Group Dynamics 34](#_Toc295308378)

[6 Appendices 35](#_Toc295308379)

[6.1 Installation 35](#_Toc295308380)

[6.1.1 Detector Deployment 35](#_Toc295308381)

[6.1.2 Programming Firmware 35](#_Toc295308382)

[6.1.3 Software Maintenance 38](#_Toc295308383)

[6.1.4 Source codes 38](#_Toc295308384)

[6.2 User Guide 39](#_Toc295308385)

[6.2.1 Web App 39](#_Toc295308386)

[6.2.2 Controller Application 45](#_Toc295308387)

[6.3 Hardware Design Specification 50](#_Toc295308388)

[6.3.1 Detectors 50](#_Toc295308389)

[6.3.2 Asset Tags 55](#_Toc295308390)

[6.3.3 RFM12B Breakout Board 59](#_Toc295308391)

[6.4 Useful Links and Reference 60](#_Toc295308392)

# 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.

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:

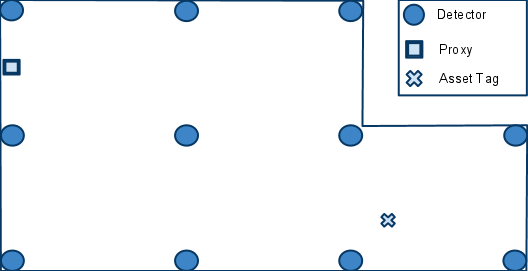
* Increased equipment utilization
* Decreased loss and theft
* Reduced costs
* Faster production
* Avoid time wasted on manual searches and inventory checks.

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.

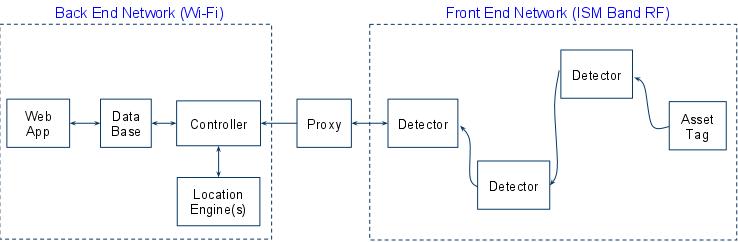
# Specification

## System Overview

Figure below 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.



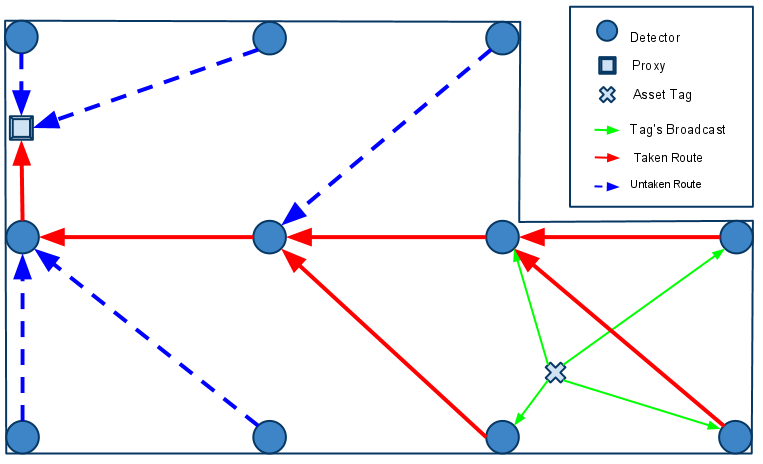
There are two distinct networks which compose this system: the front-end network, which includes all the physical nodes, and the back-end network, which includes all the logical nodes. The proxy is the gateway between the two networks.



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 below 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 below, the routes taken are in red. The blue dash paths are routes that exist but are not taken.



## Software

### Front-end Software

This section describes the firmware that is programmed into the tags, detectors and proxy. The codes are adapted to Arduino platform, using Arduino and RF12 library (provided by Jeelabs)

#### Basic Communication

The below functions are provided by jeelabs rf12 library for basic communicating purposes:

|  |  |
| --- | --- |
| Function | Description |
| rf12\_initialize() | Init radio module |
| rf12\_recvDone() | Keep the reception and transmission going |
| rf12\_sendStart() | Switch to transmission mode and send a packet. |
| rf12\_lowbat() | This checks the status of the RF12 low-battery detector |
| rf12\_canSend() | Check if radio is ready to send data |

##### Tag

Tag is typically in a low power sleep mode. It periodically wakes up, broadcasts its ID and battery level, then goes back to sleep.

Initialize MCU & Radio

Broadcast

Sleep

Battery check

Note:

* Tag performs battery check at beginning of every cycle, battery threshold: 3.2V
* Radio is initialized to broadcast at 433MHZ, -15dB, 49.2kbps, and 1 second sleep time.
* Each tag needs a unique ID

##### Detector

Detectors are responsible for:

* Picking up data from tags, determine tag’s signal strength, and transmitting them to nearby detectors or the proxy.
* Picking up data from other detectors and transmitting them to nearby detectors or the proxy.

Initialize MCU & Radio

Listening for packet

Battery check

Read RSSI

Construct new packet

From tag?

Yes

Modify packet route

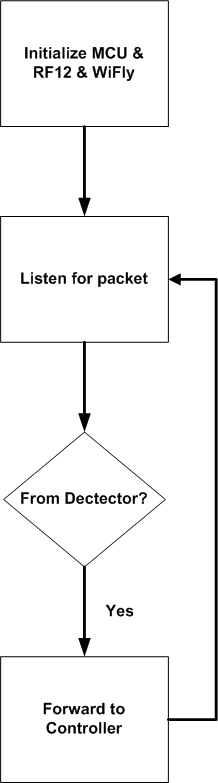
No

Transmit packet

Note:

* Detectors are always awake listening to tags and other detectors
* RSSI values are read from analog input (analogRead() - arduino library function). MCU has 10-bit analog to digital converters.
* Data packets are defined in the Mesh protocol below
* Radio is initialized to broadcast at 433MHZ, 0dB (max), 49.2kbps

##### Proxy



Initially, the proxy configures the MCU, RF12 and Wifly modules. Configuring the MCU and RF12 is a standard process. The process used to configure the Wifly is unique. Configuring the Wifly involves setting it to connect to an Ad-Hoc network that has already been created on the controller computer. In order to do so, the wifly must be told what the WPA password is.

There are other methods to configure the Wifly, but in general, the main goal is to get the Controller computer and proxy connected to each other via a TCP/IP connection. The wifly module has very good documentation located at

<http://www.sparkfun.com/datasheets/Wireless/WiFi/WiFlyGSX-um2.pdf>

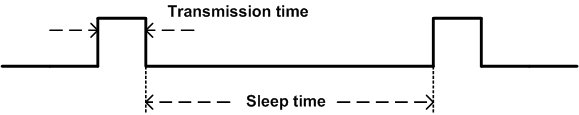
<http://forum.sparkfun.com/viewtopic.php?f=32&t=25129> <http://www.sparkfun.com/datasheets/Wireless/WiFi/rn-131-ds.pdf>

#### Power Management

##### Battery Life

* C is the capacity of battery (mAh)
* I is the active current (mA)
* D is duty cycle, which is transmission time / transmission cycle

We have this on the assumption that we *ignore the sleep current*.



For example, assuming that:

* Transmission time: 100ms every 10 seconds
* Active current 100mA (RF transceiver + MCU)
* Negligible sleep current

Approximate battery life is

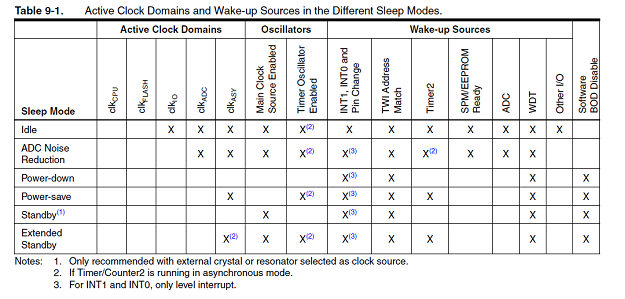
##### Transmission Time

It is good to know the transmission time of the module so that we can calculate the transmission ratio. Also, transmission time place an important role in collision avoidance (the shorter transmission time, the smaller chance of collision).

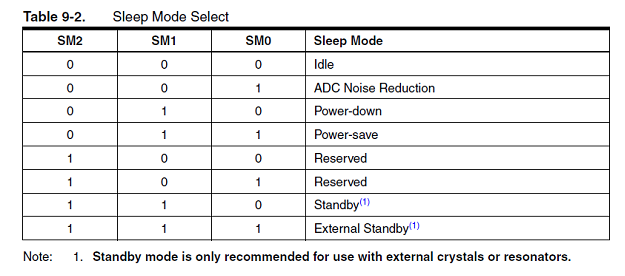
##### Sleep Mode

The sleep mode (implemented on tags) exploits power-down mode in ATMega328P and uses watchdog timer to periodically wake up the chip. The ATMega328P is Pico-power version. Below is the DC characteristic from datasheet:

Below is the reference for sleep mode and wake up source



Below are the register bits for setting up power-down mode:



Initialize MCU/Radio/WDT

Broadcast

Turn radio off

Battery check

Sleep

Turn radio on

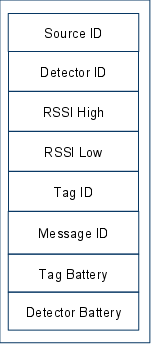
WDT interrupt/Wakeup

Note:

* MCU sleep mode implementations provided AVR library (avr/sleep.h)
* Radio sleep mode implementations provided by rf12 library (rf12\_sleep())

#### Mesh Protocol

For a detector to determine the RSSI of an asset tag, the asset tag must broadcast a message. The message has eight fields. 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 value field. If the field contains the value zero, then the message originates from an asset tag. Otherwise, the message originates from a detector.



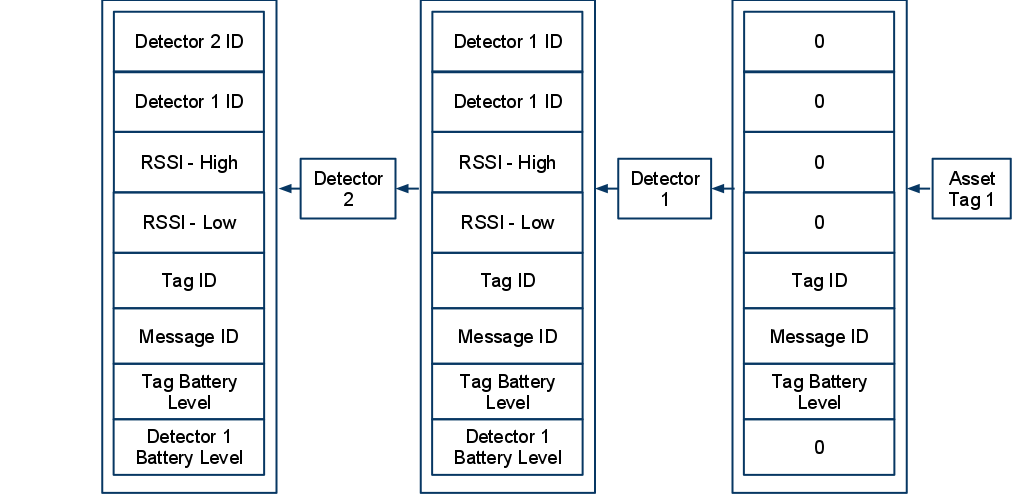
When receiving a message whose DID is zero, it assumes the origin is a tag, and it will act accordingly:

1. Replace the DID of the message with its own ID
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. Replace the Detector Battery level with that of its own.
5. Rebroadcast the message

When receiving a message whose DID is not zero, it assumes the origin is another detector, and it will act accordingly:

1. Replace the SID of the message with its own ID
2. Rebroadcast the message

An example of this message passing sequence is show in below figure.



### 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 App
* Controller
* Location Engine

#### SQL Database

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

* Location update interval
* Identifications, locations, and battery levels of the detectors
* Identifications, locations, and battery levels of the TIUs
* Mapping table between asset tags’ ID and TIUs’ ID

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

#### Web App

The web app 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
* Allow an administrator to add, modify, and remove tags and detectors
* Allow an administrator to configure the tracking area

Technologies used:

* PHP
* HTML5, CSS3, JavaScript, AJAX

##### Structure

The figure below shows the overall structure of the Web App.

MySQL Database

Common.php

TIUTracking.php

Common.js

FlexTable.js

InfoBox.js

Main.js

Map.js

TabControl.js

Timer.js

Vector2D.js

fx.js

Common.css

TIUTracking.css

MapUpload.php

Images

Request.php

* Common.php
  + Contains database information (host, name, username, password)
  + Declares global constants
  + Implement helper functions to be used in **Request.php** and **MapUpload.php**
* Request.php
  + The main client-server interface that handles all requests from client except configuring the tracking area (see Client-Server Communications section)
* TIUTracking.php
  + The main client interface that interacts directly with users
* MapUpload.php
  + Handles the configuration of the tracking area
  + Allows an administrator to change the map resolution and upload a new map image
* Common.css
  + Contains common CSS styles used in **TIUTracking.php** and **MapUpload.php**
* TIUTracking.css
  + Contains CSS styles dedicated to **TIUTracking.php**
* Images folder
  + Contains images used in **TIUTracking.php** and **MapUpload.php**
  + Contains the map image
* Common.js
  + Declares global constants and utility functions used in other JavaScript files
* FlexTable.js
  + Implements **TFlexTable** class, a flexible table that creates, modifies, and deletes cells on demand
* InfoBox.js
  + Implements **TInfoBox** class, which is used to display tags’ and detectors’ info in a tooltip-like manner
* Main.js
  + The main script handles the dynamic contents of **TIUTracking.php** as well as communicating with the server
* Map.js
  + Implements **TMap** class, an interactive 2D map with smooth zooming and panning
* TabControl.js
  + Implements **TTabControl** class, which adds logics to a tab panel
* Timer.js
  + Implements **TTimer** class, which is used to create the animation of the map as well as to periodically send request to the server for new tags’ info
* Vector2D.js
  + Implements **TVector2D** class, which is used in **Map.js**
* fx.js
  + An open source animation library from <http://fx.inetcat.com/>
  + It has been modified to improve the efficiency (speed and memory)

##### Client – Server Communications

This section discusses the format of the requests and their corresponding responses that are handled by **Request.php**.

The response is in JSON format and consists of three fields:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| request | String | Request identifier. This is the value of the **request** parameter. |
| status | Integer | 0 = No Error  1 = Session End  2 = Invalid Argument |
| data | Object | If status is 0, this field contains the returned data for the corresponding request and its type is request specific (see the following table). Otherwise, it is an error message. |

The following table describes all the requests handled by **Request.php** and the value of **data** field of the corresponding responses on success.

|  |  |  |
| --- | --- | --- |
| Request | Params | Value of ‘data’ field on success |
| Log in | request = log-in  username = ...  password = ... | Zero (SessionID regenerated, username cookie set) |
| Log out | request = log-out | Zero (All cookies cleared) |
| Get tags’ info | request = get-tags | [  {  s: String, // Timestamp  i: Integer, // Tag ID  a: String, // Asset ID  x: Double, // X coordinate  y: Double, // Y coordinate  b: Integer // Battery info  },  ... // For other tags  {} // End of array  ] |
| Add tag | request = add-tag  tag-id = ...  asset-id = ... | {  s: String, // Timestamp  i: Integer, // Tag ID  a: String, // Asset ID  x: Double, // X coordinate, default -1  y: Double, // Y coordinate, default -1  b: Integer // Battery info  } |
| Delete tag | request = del-tag  tag-id = ... | ID of the deleted tag |
| Get detectors | request = get-detectors | [  {  i: Integer, // Detector ID  x: Double, // X coordinate  y: Double, // Y coordinate  b: Integer // Battery info  },  ... // For other detectors  {} // End of array  ] |
| Add detector | request = add-detector  detector-id = ...  x = ...  y = ... | {  i: Integer, // Detector ID  x: Double, // X coordinate  y: Double, // Y coordinate  b: Integer // Battery info  } |
| Delete detector | request = del-detector  detector-id = ... | ID of the deleted detector |

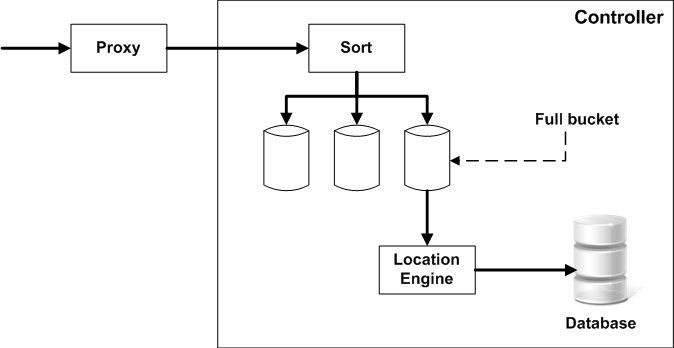
#### Controller

The controller has two major modes of operation

1. Calibrating
2. Locating

In addition to the two main operating modes, it also has logging capabilities to allow for simple diagnostics.

In calibrating mode, a Tag is placed in a known location, and data, representing an RF fingerprint, is collected and stored in an offline database. An x and y coordinate, tag Id, and calibration block id must be specified for each calibration block. If there are multiple tags currently in operation within the tracking region, the controller will ignore the tags that are not specified for calibration.

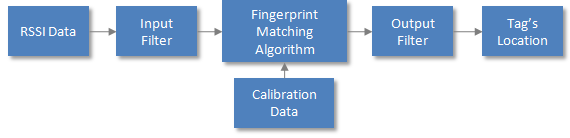


In locating mode, the controller collects and sorts incoming data, based on Tag Id, into “buckets”. Once a bucket is filled, it is handed to the location engine for location calculation. The result from the location engine is then stored into the database for future retrieval by the web application.

#### Location Engine

The location engine receives RSSI values from the controller and uses those values to calculate the locations of the asset tags. The location engine employs fingerprint algorithm which compares tag RSSI to sets of calibrated RSSI to find a best matching set. This is based on the fact that each location has a unique and consistent RSSI pattern. Since the calibrated RSSI set is associated with a calibrated location, the engine can use this result to provide location of the tag.

Data recorded from calibrating are RSSIs from a tag placed at the calibrating location (these signals measured by the surrounding detectors).

****

##### Euclidean Distance Method

Fingerprint matching uses Euclidean distance calculated as follow:

Where:

* *d* is Euclidean distance
* *N* is number of detectors
* is the RSSI value from detector i in locating phase
* is the RSSI value from detector i in calibrating phase

The minimum Euclidean distance from the set of calculated distances indicated a best matching result.

##### Aliasing

Aliasing is an issue of the fingerprint approach since two arbitrary locations can accidentally have a similar RSSI pattern, resulting in similar Euclidian distances. This causes incorrect or unstable output results.

To address aliasing, the algorithm needs to determine whether there are possible aliasing results in the calculated Euclidean distance set. If there is such case:

* If the locations are close to each others, then the algorithm chooses to interpolate their locations to provide location of tag.
* If the two locations are not close to each others, the algorithm will decide to referencing the nearest detector using the strongest RSSI in the calibration set before determining the location of tag

##### (n -1) Supporting Model

Locating algorithm uses a so-called (n-1) model to support its decisions. The (n-1) model performs n Euclidean distance calculations on n-1 RSSI values of the incoming RSSI set and at each time, an RSSI value from a detector is left out. This will give n results. Since there are less RSSI in used, prediction from this model will likely to:

* Support the main algorithm when it is unsure about a decision
* Eliminate effect of bad data from single detector

The core algorithm is described as in the below flow chart.

Calculate Euclidean distances

Sort distance list

Aliasing?

Yes

Referencing (n-1) model

No

Adjacent locations?

Interpolating

Referencing nearest detector

No

Yes

Give Tag location

Note:

* Calculations are performed using an input packet associated with a tag
* Calibrated data needs to be provided before calling locating method

##### Input & Output Filters

Input filter use previous input to predict noise in the current input packet. If the filter finds that input is unstable, it will ignore the packet and return the previous location of tag.

The output filter’s purpose is to provide stabilized output by having a moving average of the past N location calculations.

Data flow diagram:

Extract input packet from controller

Compare to previous packet

Is stable?

Yes

Retrieve previous tag location

No

Send to fingerprint engine

Send Tag location to output filter

Store tag location (engine output)

## Hardware

The system consists of three main devices: tags, detectors and proxy. Tag is attached to the tracking asset and broadcast its data to detectors. Detectors are at fixed locations, they receive data from tag or from another detector. All data will be relayed toward proxy by detectors. For system mobility, proxy is a standalone module that connects to local wireless LAN and relays data to back-end system. Since the system requires large range of tracking, tags, detectors and proxy are active broadcasting devices and need to be powered from power source.

Front-end system devices use RF technologies for communicating and locating. Data that travel through the front-end system contain device info which is used for device identification and location calculation. The primary datum used for location calculation is the Received Signal Strength Indication (RSSI) (measured by detector as receiving a broadcast from a tag. The hardware is designed to give reliable signal strength reading as well as data transferring.

### Tag

An asset tag consists of two major parts:

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

Table : Tag Specifications

|  |  |
| --- | --- |
| Specification | Description |
| Microcontroller | ATMega328P – 32 TQFP |
| RF Transceiver | RFM12B-S2 Wireless Transceiver |
| Voltage regulator | XC6221 |
| Power supply | 6V Battery (two 3-V coin cells in holder) |
| Program Interface | ISP 6-pin header |
| Operating Frequency Band | ISM 433 MHz |
| Power Indicator | No |
| RF Indicator | No |
| PCB size | 0.9930 X 1.0030 inches |
| Microcontroller | 25.4mm x 25.4mm |
| RF Transceiver | 20.3mm x 25.4mm |
| Outline Dimension | 1” x 1” x 1” |

Table : Electrical Characteristics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Specification | Min | Typ | Max | Unit |
| Power Supply | 3.3 |  | 6.0 | V |
| Transmit Current |  | 30 |  | mA |
| Idle Current |  | 40 |  | µA |
| Power Dissipation | 99 | 99 | 99 | mW |
| Operating Temperature | -25 |  | +85 | 0C |

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: RFM12B-S2 Wireless Transceiver

Table : Detector Specifications

|  |  |
| --- | --- |
| Specification | Description |
| Microcontroller | ATMega328P – 28 DIP |
| RF Transceiver | RFM12B-S2 Wireless Transceiver |
| Voltage regulator | MIC5205 150mA Low-Noise LDO Regulator |
| Power supply | 9V Battery |
| Program Interface | ISP 6-pin header |
| Operating Frequency Band | ISM 433 MHz |
| Power Indicator | Yes (regular red LED) |
| RF Indicator | Yes (green LED) |
| PCB size | 25.4mm x 88.9mm |
| Outline Dimension | 3.5” x 1” |

Table : Electrical Characteristics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Specification | Min | Typ | Max | Unit |
| Power Supply | 3.3 | 9.0 | 16.0 | V |
| Transmit Current |  | 30 |  | mA |
| Receive Current |  | 30 |  | mA |
| Idle Current |  | 30 |  | mA |
| Power Dissipation | 99 | 99 | 99 | mW |
| Operating Temperature | -25 | - | +85 | 0C |



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: ATMega328P – 28 DIP
* RF transceiver: RFM12B-S2 Wireless Transceiver Module
* Wi-Fi transceiver: WiFly GSX Module

Table : Proxy Specifications

|  |  |
| --- | --- |
| Specification | Description |
| Microcontroller | ATMega328P – 28 DIP |
| RF Transceiver | RFM12B-S2 Wireless Transceiver |
| Voltage regulator | LM317 |
| Power supply | 6V Battery |
| Program Interface | USB |
| Operating Frequency Band |  |
| RF Transceiver | ISM 433 MHz |
| Wi-Fi Transceiver | 2.4 GHz Standard |
| Power Indicator | Yes |
| Transmission Indicator | Yes |
| Outline Dimension | Prototype on bread board |
|  |  |

Table : Electrical Characteristics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Specification | Min | Typ | Max | Unit |
| Input Voltage |  |  | 3.3V | V |
| Current Draw |  | 200 |  | mA |
| Operating Temperature | -25 |  | +85 | 0C |

The microcontroller is responsible for receiving data from the RF12, and transmitting data through the Wifly.

### Parts Selection

These are selected carefully to make sure they meet these main requirements: small size, low power consumption and low price. Also, it is important that they are available and well-supported.

|  |  |  |  |
| --- | --- | --- | --- |
| D:\Study\Capstone Proj\Docs\Photos\09582-08.jpg | D:\Study\Capstone Proj\Docs\Photos\download.jpg | D:\Study\Capstone Proj\Docs\Photos\ATMegaDIP.jpg | D:\Study\Capstone Proj\Docs\Photos\328PQFP.jpg |
| **RFM12B-S2 Wireless Transceiver Module** | **WiFly GSX Breakout** | **ATMega328P – 28 DIP** | **ATMega32P – 32 TQFP** |

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

# Test Plan

## Unit Tests

### Power Supply

|  |  |  |  |
| --- | --- | --- | --- |
| Test Writer: Lynh Pham | | | |
| Test Case Name | Power Supply Unit | Test ID #: | Pwr\_01 |
| Description: | Measure output voltage of voltage regulator. Input voltage is supplied by power supply in capstone lab. Record output voltage versus input voltage values. | Type: | white box  black box |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Lynh Pham | | Date: | | | 5/6/2001 |
| Hardware Version: | | **Voltage regulator Mic5205** | | Time: | | | 1:00pm |
| Setup: | | Connect voltage meter to jumper VCC output (JP\_PWR) to measure output voltage | | | | | |
| Test | Input Voltage | Expected Output | Pass | | Fail | N/A | Comments |
| 1 | Check the red LED | Should be on with the connected jumper | x | |  |  |  |
| 2 | Connect 6V DC (4xAA batteries) to power jack | Output should be 3.3V+3% | x | |  |  |  |
| 3 | Connect 9V DC battery to power jack | Output should be 3.3V+3% | x | |  |  |  |
| 4 | Connect 9V adapter to power jack | Output should be 3.3V+3% | x | |  |  |  |
| Overall test result: | | | x | |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Lynh Pham | | Date: | | | 5/6/2011 |
| Hardware Version: | | **Voltage regulator XC6221** | | Time: | | | 1:10pm |
| Setup: | | Connect voltage meter to jumper VCC output (JP\_PWR) to measure output voltage | | | | | |
| Test | Input Voltage | Expected Output | Pass | | Fail | N/A | Comments |
| 1 | Check the red LED | Should be on with the connected jumper | x | |  |  |  |
| 2 | Connect 6V DC (4xAA batteries) to power jack | Output should be 3.3V+3% | x | |  |  |  |
| 3 | Connect 6V DC battery to power jack | Output should be 3.3V+3% | x | |  |  |  |
| Overall test result: | | | x | |  |  |  |

### Tag

|  |  |  |  |
| --- | --- | --- | --- |
| Test Writer: Lynh Pham | | | |
| Test Case Name | Asset Tag Electrical Testing | Test ID #: | Tag\_01 |
| Description: | Checks the asset tag responding RSSI outputs. | Type: | white box  black box |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Lynh Pham | | Date: | | | 5/6/2011 |
| Hardware Version: | | Asset tag | | Time: | | | 1:20pm |
| Setup: | | Remove the breakout board then connect the programmer cable to the SPI on the board. Power the testing board. Connect the base station to a PC by USB cable. | | | | | |
| Step | Action | Expected result | Pass | | Fail | N/A | Comments |
| 1 | Compile the final firmware version. | Arduino software should generate no warning/error | x | |  |  |  |
| 2 | Download firmware to MCU | AVR Studio should generate no error | x | |  |  |  |
| 3 | Turn off power |  | x | |  |  |  |
| 4 | Remove the cable from SPI |  | x | |  |  |  |
| 5 | Insert the breakout board into the tag |  | x | |  |  |  |
| 6 | Turn on power |  | x | |  |  |  |
| 7 | Open software controller | A new window should open | x | |  |  |  |
| 8 | Click Start | RSSI signal should be showed up on the window with Tag ID | x | |  |  |  |
| 9 | Repeat step 1-8 for other tags |  | x | |  |  |  |
| Overall test result: | | | x | |  |  |  |

### Detector

|  |  |  |  |
| --- | --- | --- | --- |
| Test Writer: Lynh Pham | | | |
| Test Case Name | Detector Electrical Testing | Test ID #: | Det\_01 |
| Description: | Checks the detector receiving RSSI from the asset tag. | Type: | white box  black box |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Lynh Pham | | Date: | | | 5/6/2011 |
| Hardware Version: | | Detector | | Time: | | | 1:40pm |
| Setup: | | Connect the programmer cable to the SPI on the board. Power the testing board and an asset tag. Connect the base station to a PC by USB cable. | | | | | |
| Step | Action | Expected result | Pass | | Fail | N/A | Comments |
| 1 | Check the red LED | Should be on when turn the power switch on | x | |  |  |  |
| 2 | Compile the final firmware version. | Arduino software should generate no warning/error | x | |  |  |  |
| 3 | Download firmware to MCU | AVR Studio should generate no error | x | |  |  |  |
| 4 | Turn off power | The red LED should be off | x | |  |  |  |
| 5 | Remove the cable from SPI |  | x | |  |  |  |
| 6 | Turn on power | The red LED should be on | x | |  |  |  |
| 7 | Open software controller | A new window should open | x | |  |  |  |
| 8 | Check the green LED | Should be blinking with the connected jumper | x | |  |  |  |
| 9 | Click Start | RSSI signal should be showed up on the window with detector ID and Tag ID | x | |  |  |  |
| 10 | Repeat step 1-9 for other detectors |  | x | |  |  |  |
| Overall test result: | | | x | |  |  |  |

## Integration Tests

### RF Network and Proxy Communication

|  |  |  |  |
| --- | --- | --- | --- |
| Test Writer: Dung Le | | | |
| Test Case Name | RF network and Proxy communication testing | Test ID #: |  |
| Description: | Test functionality of RF network and Proxy. Use Wi-Fi proxy to relay all data from detectors to the controller. | Type: | white box  black box |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Dung Le | | Date: | | | 5/22/2011 |
| Hardware Version: | | Detector | | Time: | | | 11:00am |
| Setup: | |  | | | | | |
| Step | Action | Expected result | Pass | | Fail | N/A | Comments |
| 1 | Program one tag and one detector with test IDs and transmit interval of 1 second |  | x | |  |  |  |
| 2 | Turn on tag, detector and proxy | Detector blinking every 1 second | x | |  |  |  |
| 3 | Open controller |  | x | |  |  |  |
| 4 | Turn on proxy to local network | The output console window says “HELLO” | x | |  |  |  |
| 5 | Use controller to collect data | data on output console window | x | |  |  |  |
| Overall test result: | | | x | |  |  |  |

### Fully Integrated System

Multiple Tags, Multiple Detectors, Proxy, Database

|  |  |  |  |
| --- | --- | --- | --- |
| Test Writer: Daniel | | | |
| Test Case Name | Locating with multiple detectors testing | Test ID #: |  |
| Description: | Test functionality of the tracking system with one tag, multiple detectors and proxy. Calibrating and Locating procedures are conducted. Locations of tags are display on 2D map on Web app which exploited data from database on CAT server.  Location: capstone lab | Type: | white box  black box |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Daniel | | Date: | | | 5/22/2011 |
| Hardware Version: | | Detector, Tag, Proxy (PCB) | | Time: | | |  |
| Setup: | |  | | | | | |
| Step | Action | Expected result | Pass | | Fail | N/A | Comments |
| 1 | Program tag and detectors with IDs and transmit interval of 1 seconds |  |  | |  |  |  |
| 2 | Power up tag and detectors, then distribute detectors at four corners of the lab | Detector’s LEDs blink at approximately 1 second | x | |  |  |  |
| 3 | Open the controller |  |  | |  |  |  |
| 4 | Place tag at 1st location |  |  | |  |  |  |
| 5 | Calibrate | Calibrated blocks shown on controller | x | |  |  |  |
| 6 | Repeat 3-4 (total 4 locations) |  |  | |  |  |  |
| 7 | Place tag at 1st location (exact) |  |  | |  |  |  |
| 8 | Locate | Tag’s locations shown on controller | x | |  |  |  |
| 9 | Repeat 7-8 (total 4 locations) |  |  | |  |  |  |
| Overall test result: | | | x | |  |  | 3/4 locations are located correctly |

## Acceptance Tests

### Tag’s Size

|  |  |  |  |
| --- | --- | --- | --- |
| Test Writer: Lynh Pham | | | |
| Test Case Name | Tag’s Size testing | Test ID #: |  |
| Description: | Using ruler to measure the actual size of asset tag (PCB). The required size of a tag must be 1” x 1” x 1”. | Type: | white box  black box |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Lynh Pham | | Date: | | | 5/6/2011 |
| Hardware Version: | | Tag (PCB) | | Time: | | | 1:45pm |
| Setup: | | Prepare three tags and ruler for measurement. | | | | | |
| Step | Action | Expected result | Pass | | Fail | N/A | Comments |
| 1 | Measure one tag’s size | 1” x 1” x 1” | x | |  |  | 0.9930” x 1.0030” |
| 2 | Repeat measurement with two more tags |  |  | |  |  |  |
| 3 | Average size |  |  | |  |  | 1” x 1” x 1” |
| 4 |  |  |  | |  |  |  |
| Overall test result: | | | x | |  |  |  |

### Power

|  |  |  |  |
| --- | --- | --- | --- |
| Test Writer: Lynh Pham | | | |
| Test Case Name | Consumed power of asset tag | Test ID #: |  |
| Description: | Measure power consumed by a tag during stressed testing condition in order to predict battery lifetime of the asset tag. Expected battery lifetime is more than one month with normal operating conditions. | Type: | white box  black box |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Lynh | | Date: | | | 5/6/2011 |
| Hardware Version: | | Tag (PCB) | | Time: | | | 1:50pm |
| Setup: | | A tag with 1 sec broadcast interval, 3ms transmit window, multi-meter | | | | | |
| Step | Action | Expected result | Pass | | Fail | N/A | Comments |
| 1 | Insert 2 coin cell batteries |  |  | |  |  |  |
| 2 | Power a detector |  |  | |  |  |  |
| 3 | Check the green LED | Should be blinking | x | |  |  |  |
| 4 | Use a multi-meter measure voltage of coin cell batteries | Should be 6.4V at beginning | x | |  |  |  |
| 5 | Repeat step 4 every 12 hours until the voltage drop below 3.3V | The voltage should decrease a little bit. The batteries should be last long 76 days | x | |  |  |  |
| Overall test result: | | | x | |  |  |  |

### Accuracy

|  |  |  |  |
| --- | --- | --- | --- |
| Test Writer: Tri Truong | | | |
| Test Case Name | Accuracy of the tracking system testing | Test ID #: |  |
| Description: | Test the accuracy of the tracking system by evaluating the error of the calculated results returned by the controller during locating process. Test with four detectors and one tag at Smith Food Court, PSU campus. Expected accuracy is 1.5m. | Type: | white box  black box |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tester Information | | | | | | | |
| Name of Tester: | | Daniel – Dung Le | | Date: | | | 5/22/2011 |
| Hardware Version: | | Detector, tag (PCB), proxy (prototype) | | Time: | | |  |
| Setup: | | Set up four detectors in the 12m x 6m tracking area as shown in Figure xx. | | | | | |
| Step | Action | Expected result | Pass | | Fail | N/A | Comments |
| 1 | Program tag and detectors with IDs and transmit interval of 1 seconds |  |  | |  |  |  |
| 2 | Power up tag and detectors, then distribute detectors at four corners of the lab | Detector’s LEDs blink at approximately 1 second | x | |  |  |  |
| 3 | Open the controller |  |  | |  |  |  |
| 4 | Open web app and configure detectors location |  | x | |  |  |  |
| 5 | Place tag at 1st location |  |  | |  |  |  |
| 6 | Calibrate | Calibrated blocks shown on controller | x | |  |  |  |
| 7 | Repeat 3-4 (total 18 locations) |  |  | |  |  |  |
| 8 | Place tag at 1st location (exact) |  |  | |  |  |  |
| 9 | Locate | Tag’s locations shown on web app | x | |  |  |  |
| 10 | Repeat 8-9 (total 18 locations) | Detector’s LEDs blink at approximately 1 second | x | |  |  |  |
| Overall test result: average accuracy ~2m | | | x | |  |  | See result below |

**Setup:**

Detector are set up as shown in the below figure. Calibration points are marked with corresponding block ID numbers. In this tracking area, we use four detectors and calibrate on 15m x 6m area with 3m x 3m grid. Hence, the total number of calibration points is 18 points.

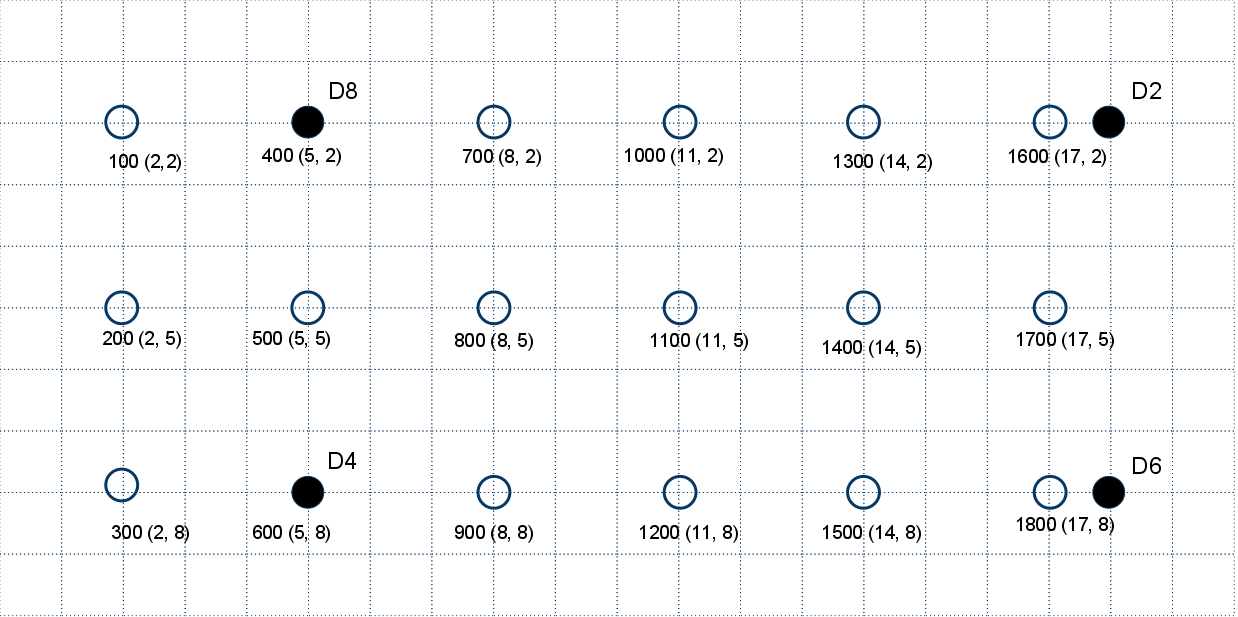


Figure ‑: Layout of the tracking area – Smith Food Court, SMSU

The error of measurement at each locating point was recorded. The error is defined as the Euclidean distance between the actual location and calculated location. The measurements were conducted at exact calibration points, 0.5m, and 1m far away from the calibration points.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Location** | | **Error as distance (m)** | | |
| Block ID | X | Y | Exact | 1m | 0.5m |
| 100 | 2 | 2 | 0 | 0.5 | 2.5 |
| 200 | 2 | 5 | 0 | **confused** | 2.5 |
| 300 | 2 | 8 | **confused** | 3 | 4 |
| 400 | 5 | 2 | 0 | 2 | 0.5 |
| 500 | 5 | 5 | 0 | 2 | 2.5 |
| 600 | 5 | 8 | 0 | 2.5 | 0.5 |
| 700 | 8 | 2 | 0 | 2.2 | 2.5 |
| 800 | 8 | 5 | 0 | 2.5 | 1 |
| 900 | 8 | 8 | 0 | 4 | 3 |
| 1000 | 11 | 2 | 0 | 2.2 | 2.5 |
| 1100 | 11 | 5 | 3 | 3 | 3 |
| 1200 | 11 | 8 | 0 | 2 | **confused** |
| 1300 | 14 | 2 | 0 | 1 | 4 |
| 1400 | 14 | 5 | 2 | 5 | 3 |
| 1500 | 14 | 8 | 0 | 4 | 1 |
| 1600 | 17 | 2 | 0 | 2.2 | 3 |
| 1700 | 17 | 5 | 0 | 1.8 | 1.5 |
| 1800 | 17 | 8 | 0 | 1 | 0.5 |

Some measurements were noted as “confused” when the calculated location of the tag oscillated between two or more different locations on the map. These oscillations were caused by aliasing of the signal signature in the tracking area, where some locations have nearly the same signal signature.

## Final Results

Feasible experiments and acceptance tests were conducted to ensure our design meet the requirements. Technically, the designed TIU tracking system has satisfied basic requirements about size, power, and accuracy. We also conducted experiments with multi-tags and multi-detectors. In such testing, small adjustments of broadcast time interval and transmit window in firmware need to be done to reduce signal collisions. Otherwise, the system will corrupt since message packets will be dropped frequently and the controller will have lack of data to calculate out tags’ locations.

A fully testing with 10 detectors and 8 tags could not be done because of time and space constraint. It is hard to find a large area on campus where we can conduct such experiment. The largest dimension of the area we has had done experiments and testing is about 18m x 8m in Smith Food Court, SMSU, on PSU campus.

### Device Dimensions

As shown the acceptance test for tags’ size, the physical dimensions of the final products are:

* Tag’s size: 1”x1”x1”
* Detector’s size: 3.5”x1”

The tag’s size satisfies the requirements given by the sponsors. There is no constraint about the size of detectors. With such small size, detectors are quite portable and can be mounted on ceiling of the tracking area.

### Power Consumption

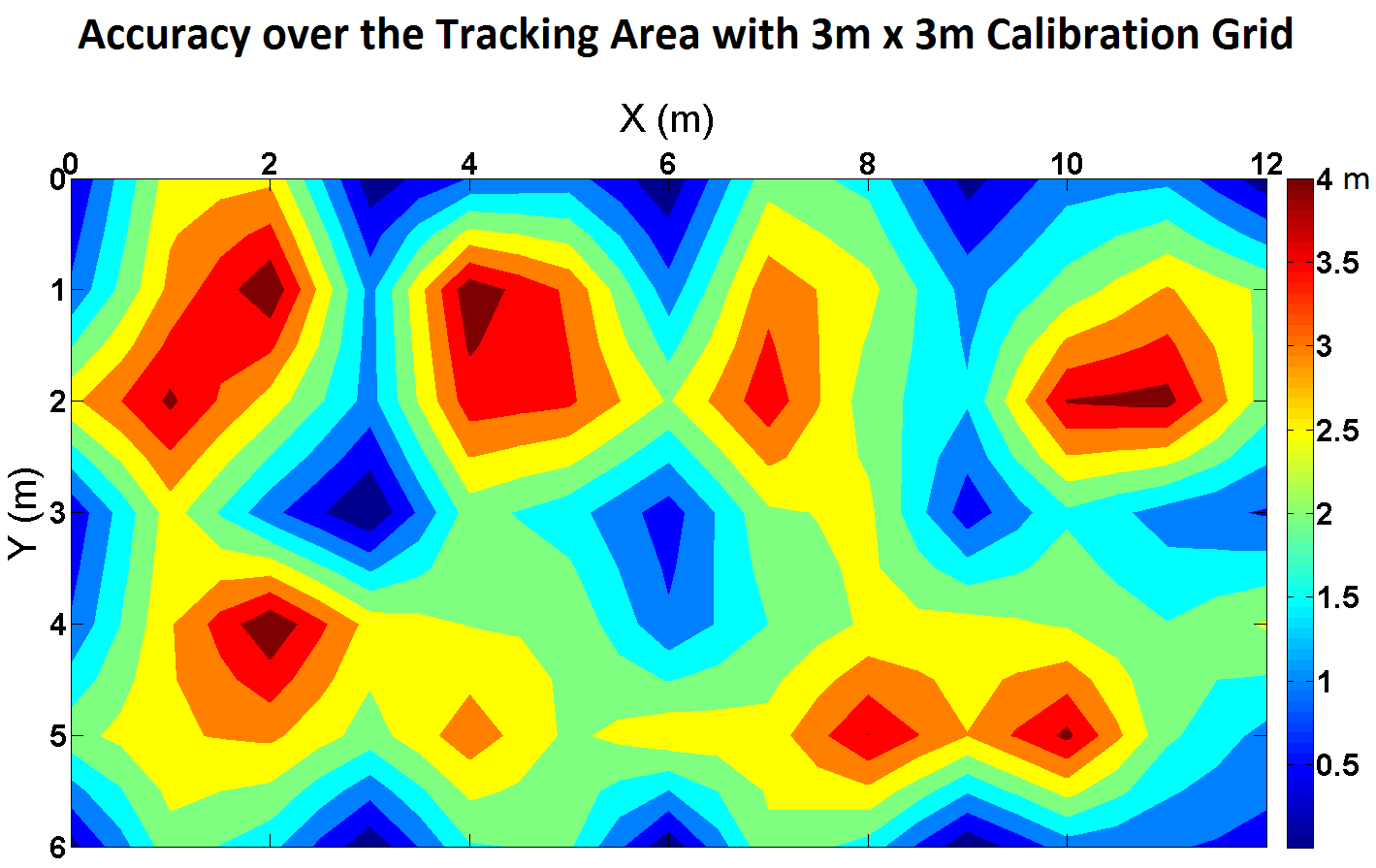
Because of time constraint, we cannot test the power consumption of a tag in one month. Instead, we applied stressed conditions on tag’s operation which is specified as follow:

* 240mAh coin cell battery
* 30mA transmit current
* 40µA sleep current
* 1sec broadcast interval
* 3ms transmit window
* 0.3% duty cycle

The estimated batter life would be

In reality, a tag will broadcast at lower frequency than stressed frequency. Hence, the requirement of power consumption of tags can be easily achieved.

### Accuracy



An accuracy map was plotted to show the error of locating results over the tracking area. The color indicates the error between the actual location and calculated location for each measurement. The number of calibrated points is 18. The number of located points is 54. We did not locate the tag at every location in the tracking area. The colored map is interpolated to show the contour field.

# Discussions

Antenna design

More testing

* Calibration density
* Detector placement

Improve testability

Different algorithms

Environment & signal strength

System configuration and detector installation

# Conclusions

We have a complete system that achieves small size, low cost, and battery life that exceeds 1 month. As for accuracy, when a tag is placed on exactly a location that was calibrated, success rates are at least 90%. The average accuracy, or error, over the tested tracking area is about two meters. Basically, we have achieved all requirements given by the sponsor: size, power, and accuracy. In fact, we specified a challenging accuracy which is 1.5m (5 ft.). However, we claim that such accuracy level is difficult to be achieved with limitation of radio technology as well as with prototypes in this capstone project.

There is also the trade-off between the system’s cost and performance. Obviously, tracking TIU in Intel’s validation lab requires high reliability rather than high accuracy since the system must tolerate to noise in lab environment while TIUs are pretty large PCBs that can be easily found within 5m.

Further work to be done includes refining the testability of the system, analyzing antenna radiation pattern, and improving the robustness of the location algorithms. The hardware design could be optimized further to improve the system’s performance and reliability.

## What We Learned

This project presented many opportunities to expand our horizon. On a high level, we were responsible for bringing to fruition an entire system from a mere idea. This required requirement gathering, cost analysis, case studies, creativity, cooperation, experimentation, trials, and solid group communication. Every one of these brought with it a myriad of educational experiences and opportunities that we will be able to take with us as we go into the future. On a technical level, this project allowed us to bring together many different technologies under one roof. From microcontrollers, circuit layout, wireless communications, math, databases, and user experience. The variety of technologies gave us an opportunity to bring one component up at a time, and slowly integrate them into a single, seamless system. This level of engineering is a powerful capability to add to our resumes. Finally, our team had to overcome and compensate for a dynamic group environment. It required us to make sure our thoughts were crystal clear, and our words were concise. We learned how to civilly express ourselves when we felt constructive, and when we felt critical.

In this project, we also have experience in how to solder surface mount devices. It is actually easy than we thought at first. We were trained soldering techniques in pre-capstone course (ECE 411) and it is amazing that these practices are quite useful in capstone project.

## Good Decisions

Deciding to use Received Signal Strength Indication (RSSI) seems to be the right choice for an indoor locating system. We have had a lot of discussions about medium and methods using for the tracking system. We focused on radio frequency wave as the medium for detection and tracking and RSSI was selected as basis for calculating method. Within indoor environments, propagation loss model with triangulation methods is difficult to be implemented since reflection, attenuation, and multipath effect will strongly affect radio signal. A unique radiation pattern, or fingerprint, was assumed to be representative for a location, even though aliasing could not be eliminated out of noisy environments.

## Bad Decisions

We decided to use through-hole components on detector. This was based on the situation that we were inexperienced to surface mount soldering, detector hardware does not have size constraint, and we also had few through-hole MCUs laying around. After finished the soldering, we agreed that surface mount design would be a better choice not only because the soldering were not as difficult as we thought but also it would have saved us a lot of PCB making cost.

Also, because of the heavy load of works needed to be done, we did not spend enough time to provide full features on hardware such as routing all the pins to external ports for later development and advance debugging.

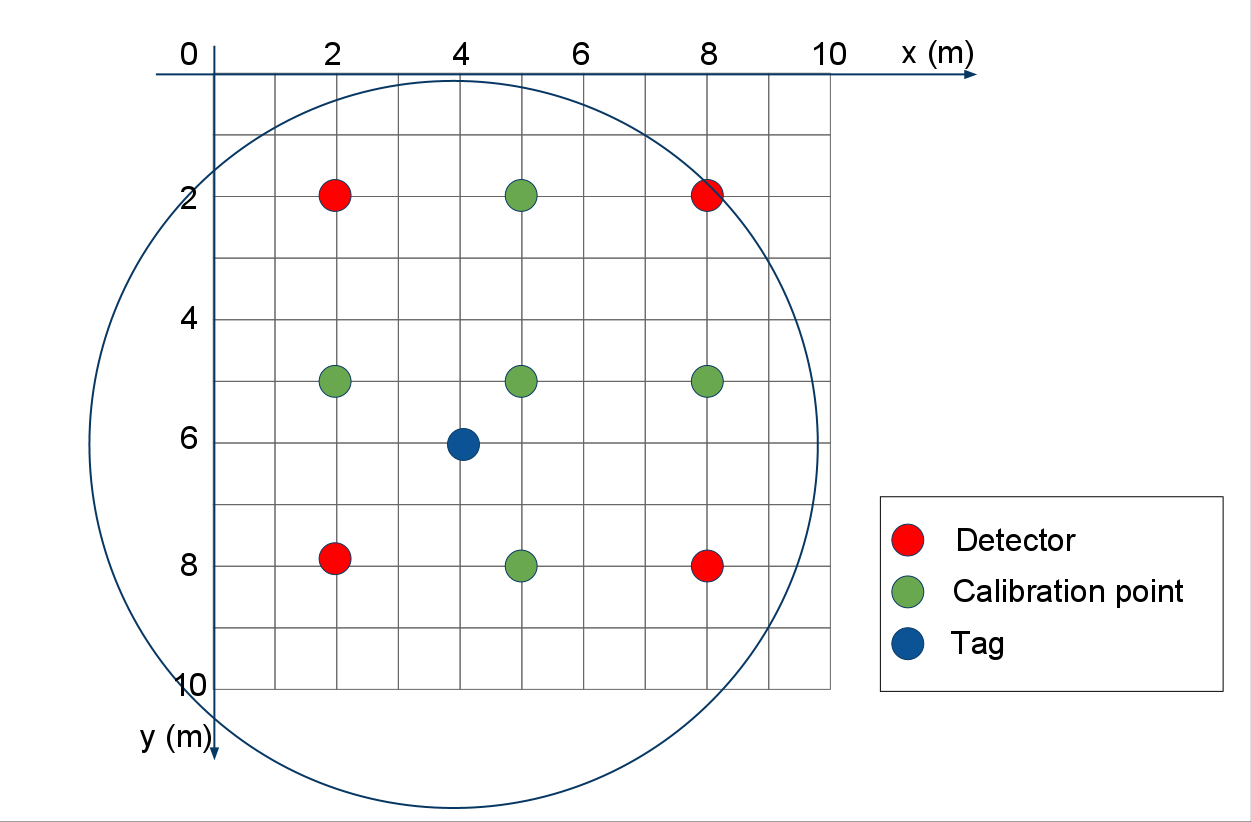
## Group Dynamics

Conflicts in group during the project were unavoidable. However, we always tried to resolve them in a constructive way to help the team move forward. Actually, conflicts could be seen as positive factors that contribute to the success of the project since we had more discussions, arguments for right choices and selecting best ideas. That is also a valuable lesson for us since we might deal with similar situations in future when we work in companies.

# Appendices

## Installation

### Detector Deployment



### Programming Firmware

Both tags and detectors are designed to support programming. The devices’ firmware can be updated using 6-pin ISP header on printed circuit board. An easy way to reprogram the chip is using AVR Studio with ISP mkII Programmer. Refer to online tutorials from manufacturers for more information (also see **Useful Links** section).



ISP mkII programmer (Atmel Corp.)

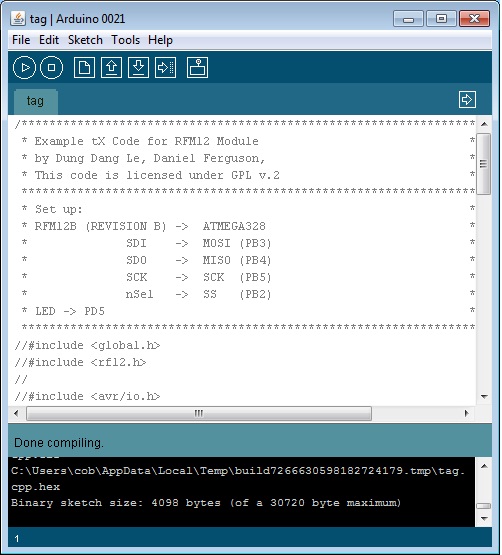
**What you need**

* AVR Studio
* Arduino Environment
* rf12 library (modified version, support RSSI analog reading), can be found in Firmware/arduinolibs
* Copy RF12 folder into ...arduino-0021\libraries\ (Windows machine). In Linux, you will find it somewhere else...)
* Device firmware can be downloaded from Main Site of TIU Tracking project.

**Compile Source**

* Open firmware source in Arduino IDE
* Hold Shift key and hit compile button
* Go to the end of printout window, copy the “C:\Users\...\Temp\build3902875270031311054.tmp\....cpp.hex” to clipboard

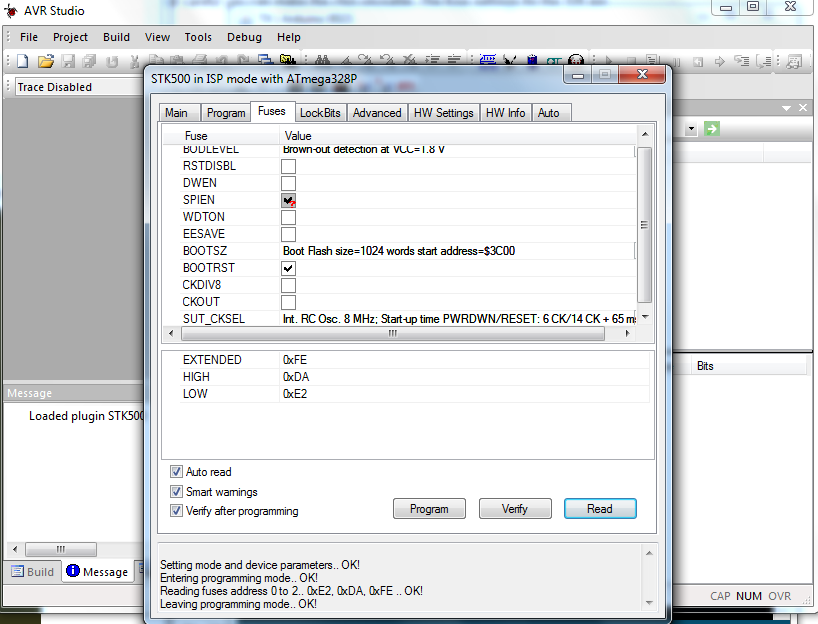
This is the compiled hex file that can be used for burning device flash.



Arduino Compiler

**Programming**

* Open AVR Studio
* Make sure mkII programmer is connected to PC and the target device.
* Make sure the device is powered.
* Burn fuses:
  + Fuse settings for chips that are off-arduino and 3.3V power supply is shown in below figure.
  + Program fuse bits
* Programming Flash on chip
  + Select **Program** tab
  + Paste the path copied above
  + Click **Program** to flash the chip

****

Fuse settings

### Software Maintenance

### Source codes

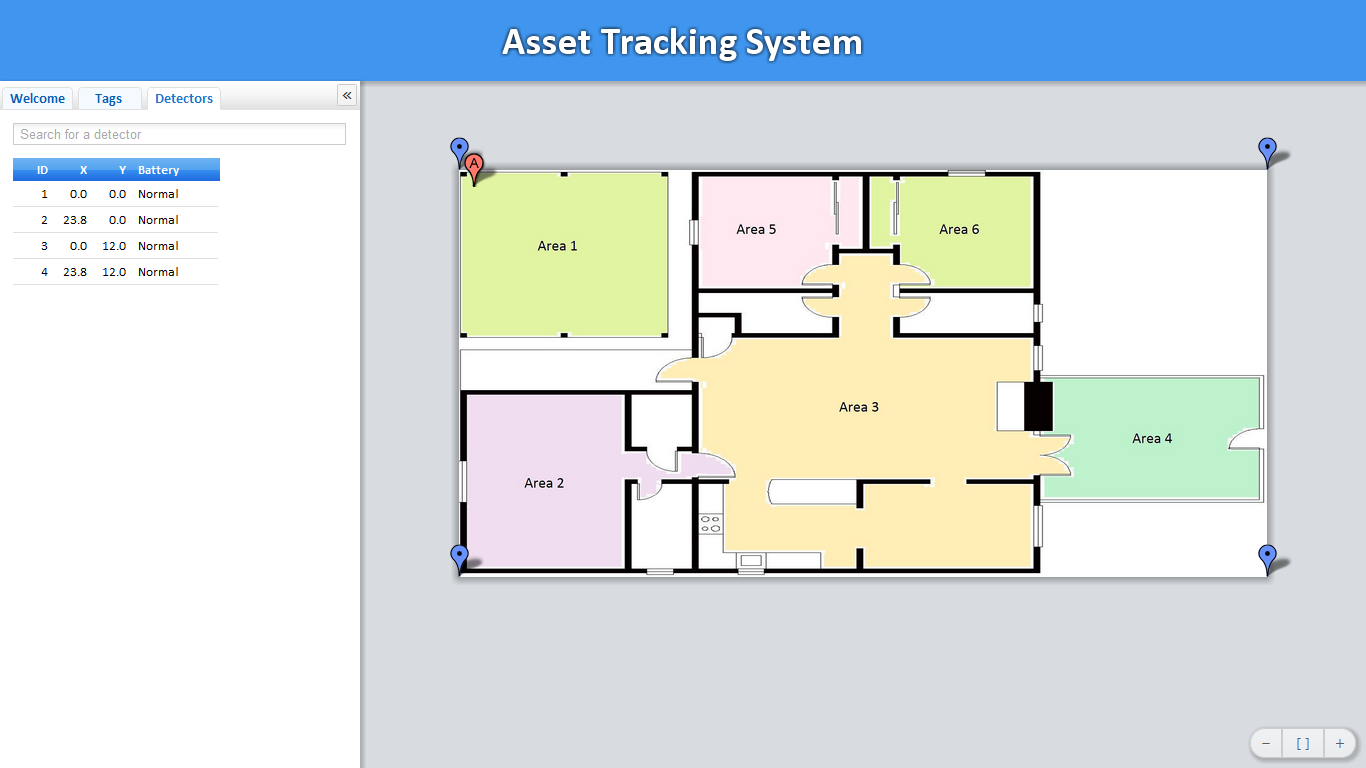
All source codes are open and available on ***Project Hosting on Google Code*.**

URL: <http://code.google.com/p/tiu-tracking/>

## User Guide

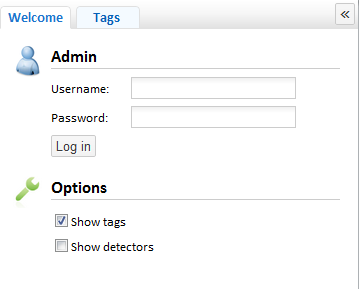
### Web App

#### Users



Web-based application user interface

Users can select/deselect on Options to display/hide tags and detectors on 2D map.

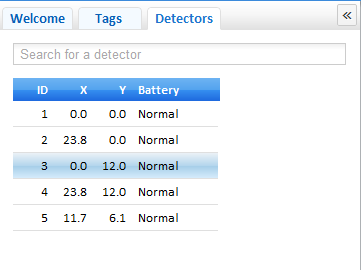


Options Table

**Display detectors’ info**

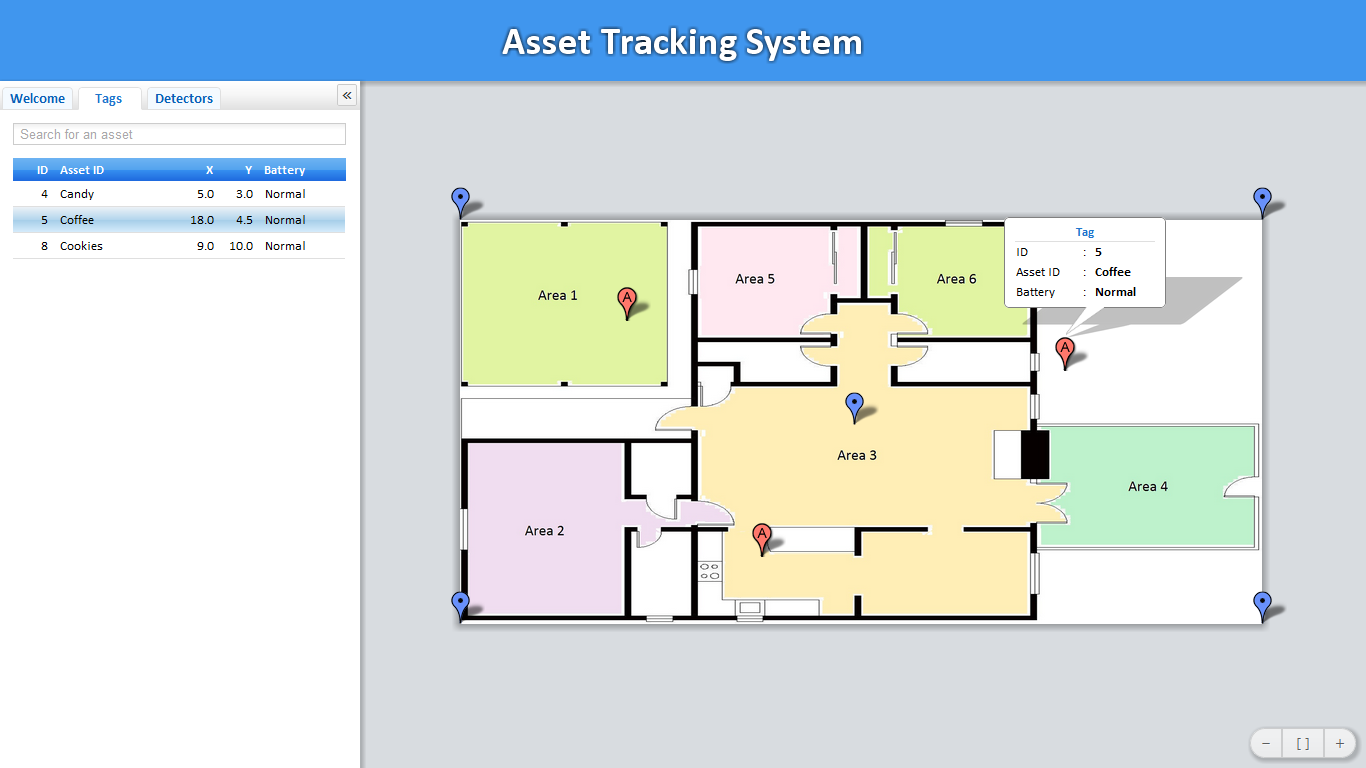
****

Display Detectors’ info on 2D map



Detector’s info table

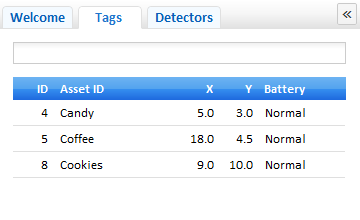
**Find an asset on the map**

****

Show tag’s info on 2D map

Each asset (TIU) has a unique ID which is mapped to asset tag ID shown on the map of the Web app. To display a tag’s info, including ID, Asset ID, and battery status, either of the following ways are applicable:

* Click the read bubble on 2D map. Note that corresponding line on the table is highlighted.
* Move the mouse over the red bubble.
* Select an asset tag displayed on the table (left side).



Tag’s info table

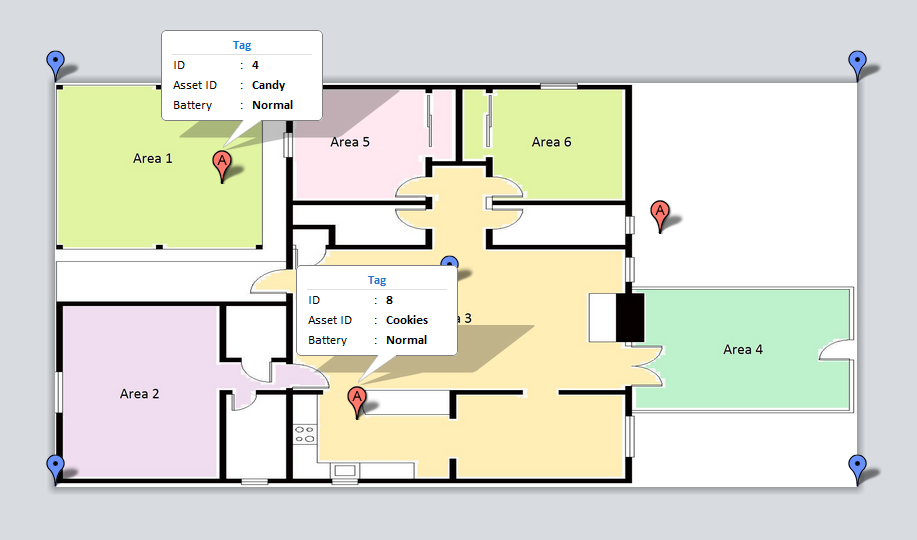
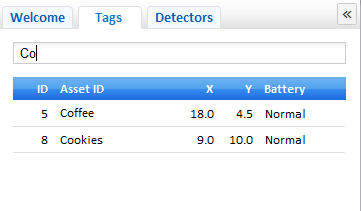
****

Figure ‑: Multi-tags’ info are displayed on 2D map

To find an asset tag quickly, users can enter Asset ID into Search textbox. **Instant Filter** function returns matched results.



Search a tag with search textbox

**Zooming and Panning**

The web app highly supports zooming and panning functions. Users can experience by dragging mouse, rolling mouse wheel to zoom in and zoom out. Basic zooming functions are provided at the right bottom corner of the web app.

**Recommended Web browsers**

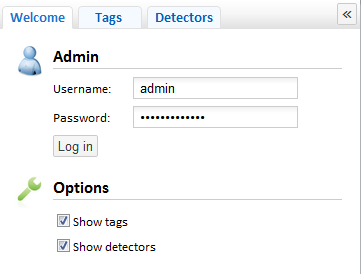
The Web-based application of the TIU Tracking system has deployed latest web technologies which are supported so well on current Web browsers such as Chrome, Mozilla Firefox, and Safari. We strongly recommend using Chrome for the highest performance of the application.

#### Administrator

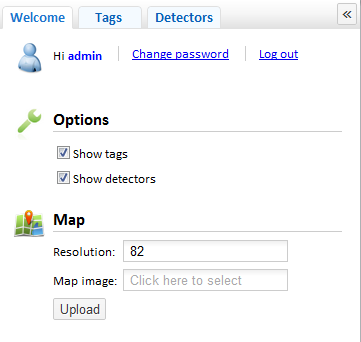
**Log in session**

In order to have privilege to configure the system, users need to log in as administrator. In Admin mode, the administrator can:

* Add/Remove tags
* Add/Remove detectors
* Modify detectors’ locations
* Upload a new map
* Change password, add/modify authorized users who can log in

****

Log in session

****

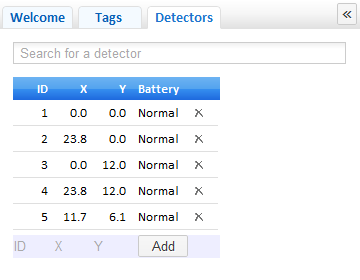
Admin control table

**Upload a new map**

In Admin mode, users can upload a new map for the tracking area.

* Click on **Map Image** box
* In **Open File** Dialog Box, select a new map (\*.jpg, \*.png, …)
* Specify an appropriate resolution in **Resolution** textbox
* Click **Upload**

**Add detectors in tracking area**

****

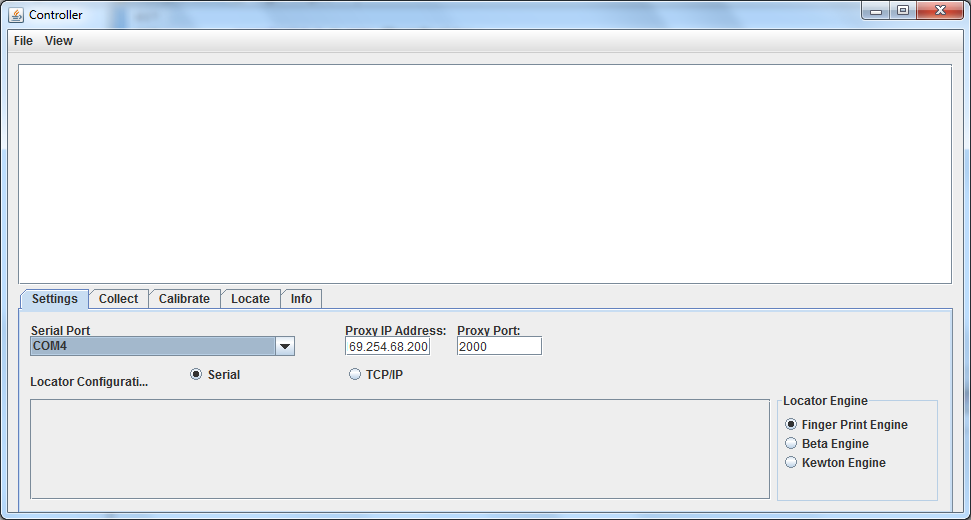
Add detectors to the system

To add a detector to the system,

* Enter ID and (X, Y) location of the detector. Detector’s location can be specified by clicking on 2D map.
* Click **Add** to finish the task
* Detectors’ info can be modified by add a new detector with the same ID.

### Controller Application

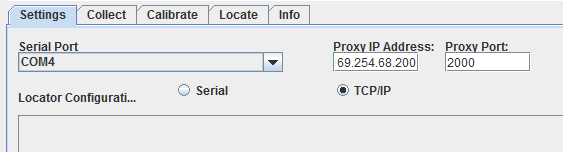
The controller software application provides utilities to calibrate, locate tags, and diagnostics tools. The GUI application has five different tabs for corresponding tasks, including Settings, Collects, Calibrate, Locate, and Info.



The controller application user interface

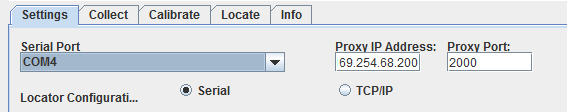
#### Set up Connection Between Proxy and Controller

**Wi-Fi Proxy connection**

****

Wi-Fi Proxy Connection Settings

**Serial connection**

****

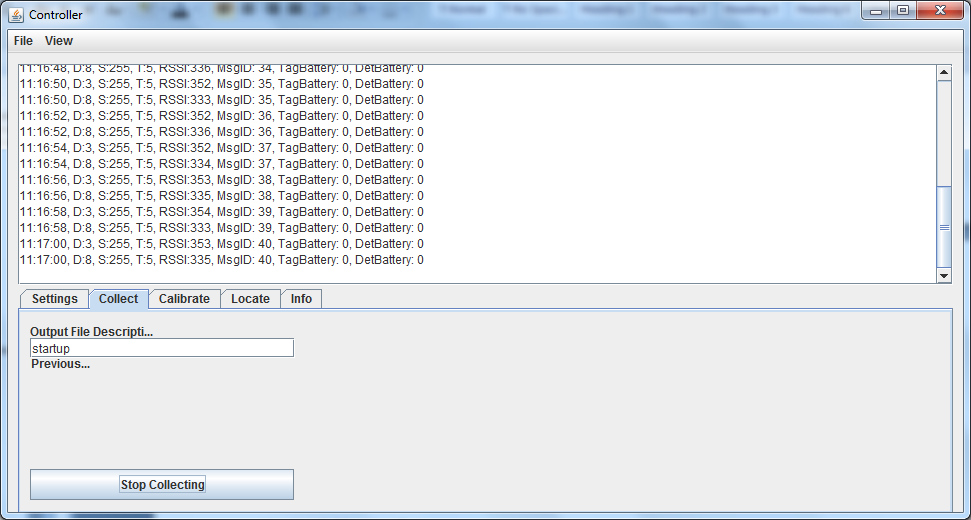
Serial Connection Settings

To run the controller using serial-connected Arduino board as a base,

* Plug the Arduino board to USB port on computer
* Run Controller Java application
* In Setting tab, select **Serial** radio button
* Select Serial Port (e.g. COM4)

#### Run Controller application

**Data Collecting Mode**

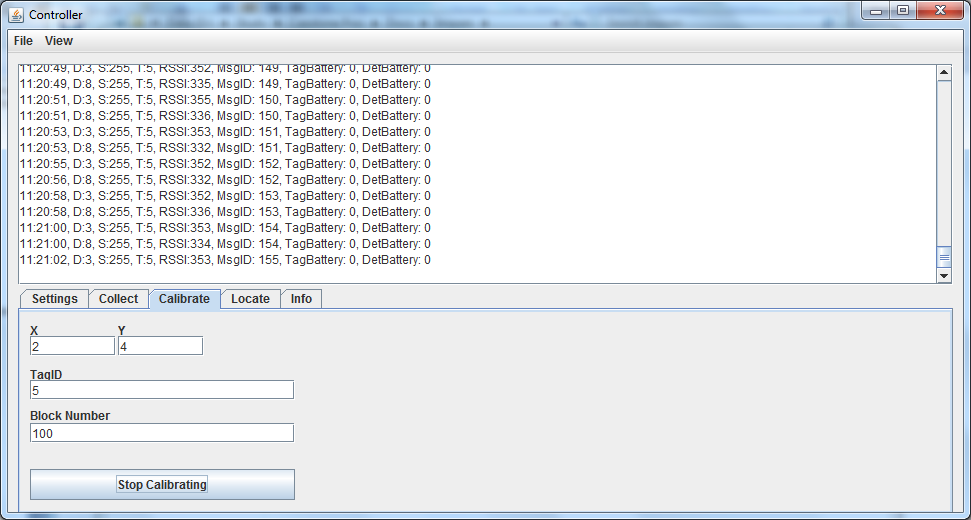
****

Controller runs in data collecting mode

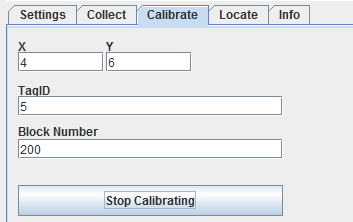
Data Collecting Mode provides useful diagnostic tools to collect RSSI from detectors, test tags and detectors. Data is saved in \*.csv file which is convenient for filtering on Excel.

* Select **Collect** tab
* Enter data file name
* Click **Start Collecting**. Then, click **Stop Collecting** to finish.

**Calibrating Mode**

****

Controller runs in calibrating mode



Parameters and Location Settings in one calibration

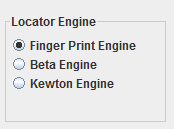
To calibrate the tracking area:

* Select **Calibrate** tab
* Enter (X, Y) value of the current calibration point
* Enter the tag ID used in calibration process
* Enter block ID
* Click **Start Calibrating**. Then, click **Stop Calibrating** to finish

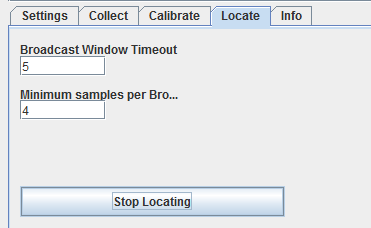
**Locating Mode**

To run locating mode:

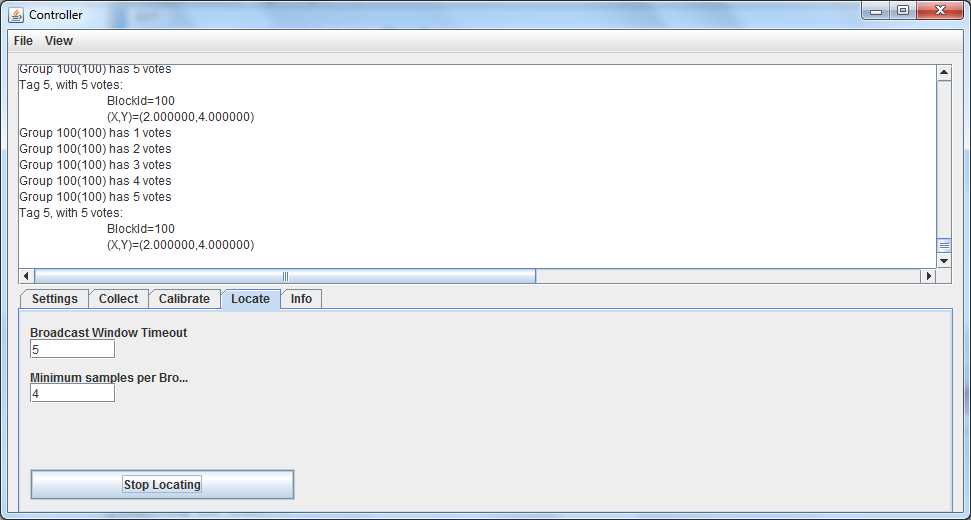
* Select Location Engine in **Settings** tab (current implementation in Finger Print Engine)
* Select **Locate** tab, specify parameters if necessary
* Click **Start Locating**. Then, click **Stop Locating** to finish



Choose Location Engine (Locating algorithm)



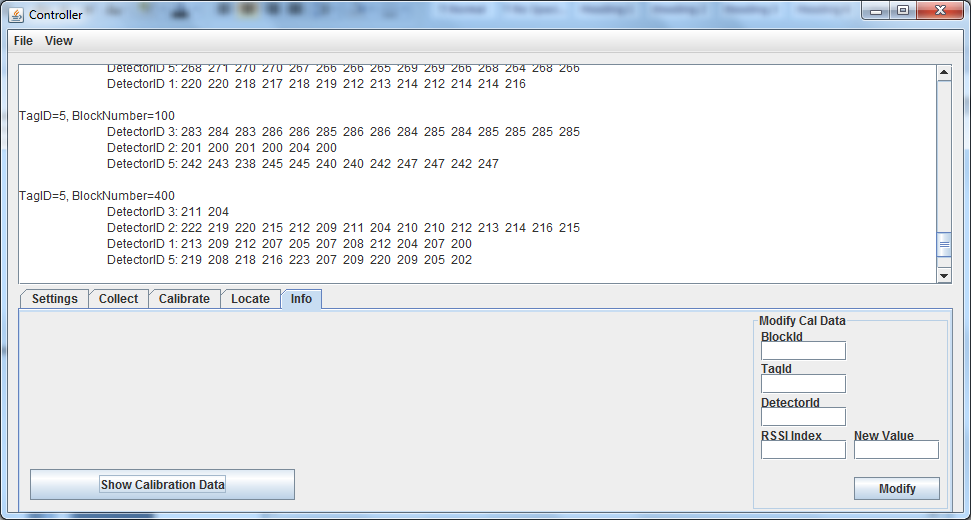
Locate tab settings

****

Controller run in Locating Mode

**Viewing Calibration Data**

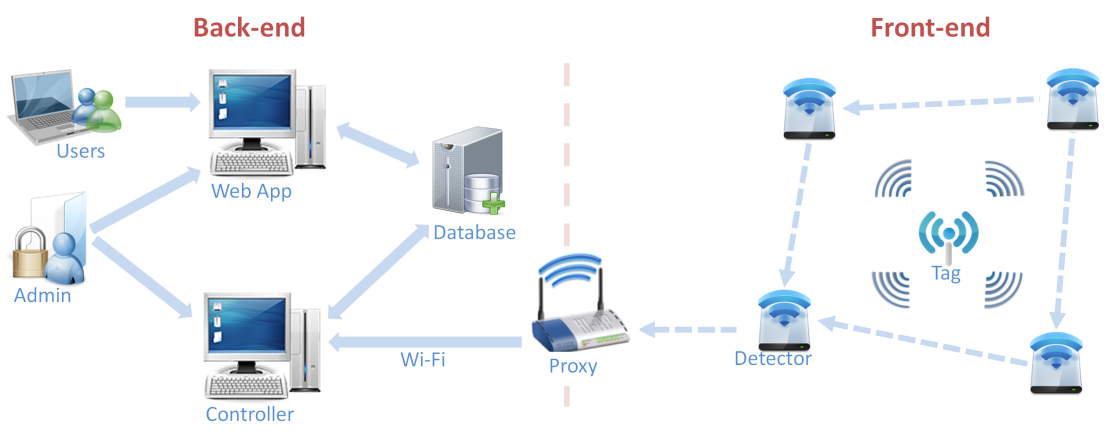
Select **Info** tab, and click Show Calibration Data to view data of all blocks.

****

View calibration data file

## Hardware Design Specification

The infrastructure of the tracking system consists of detectors, asset tags, proxies and a server, as shown in below figure.



TIU Tracking System topology

Our system is made up of two networks. The front end network, composed of tags and detectors, and a back end network composed of the controller, location engine, database, and web app.  
The proxy is a device that links the two networks together by relaying messages from the front end to the controller in the backend. The Tags are affixed to the TIU's, and periodically broadcast. The broadcast gives the detectors an opportunity to determine the signal strength of the Tags. This knowledge is then relayed through nearby detectors until it reaches the proxy. The proxy then forwards the messages to the controller which feeds it to the locating engine. Then, the locating engine calculates a location, returns the result to the controller, where it is inserted into the Database. The web app then pulls the results from the database and displays them on an interactive 2D map of the tracking area.

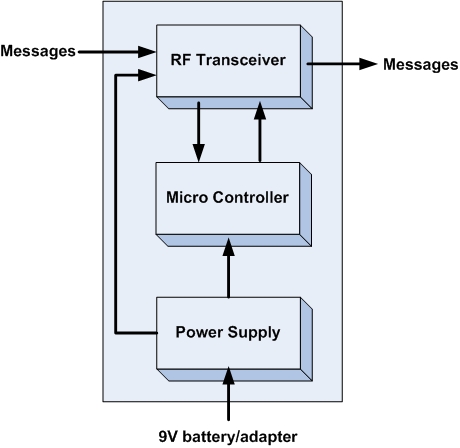
The following sections provide more details about hardware design in the front-end network. These sections will help readers reproducing the PCB products effectively. All design specifications and schematics are delivered.

### Detectors

A detector consists of two major parts: a microcontroller, and a RF transceiver. The microcontroller controls the RF transceiver. The RF transceiver is responsible for communicating information to the system.

Detectors are responsible for gathering RSSI data for the locating algorithm. They must always be listening for a tag to broadcast. Upon receiving a broadcast from a tag, or another detector, it rebroadcasts the message, which results in messages always propagating toward the proxy

#### Functionality



|  |  |
| --- | --- |
| Module | Detectors |
| Inputs | * Power supply: 3.3V DC * Messages from asset tags and other detectors |
| Outputs | * Messages from asset tags and other detectors |
| Functionality | A detector consists of two major parts: a microcontroller, and a RF transceiver. The microcontroller controls the RF transceiver. The RF transceiver is responsible for communicating information to the system. |

#### What is on the Board?

* A power switch on/off
* A 3.3V power regulator which can sustain maximum 16V input voltage.
* An ATMega328p MCU by Atmel with 8MHz
* A 6pins SPI connector for programming.
* A RFM12B wireless RF module for 434MHz ISM band by Hope RF.
* A LED with jumper (on/off) for checking power.
* A LED with jumper (on/off) for checking signal transmission.

#### Advantages

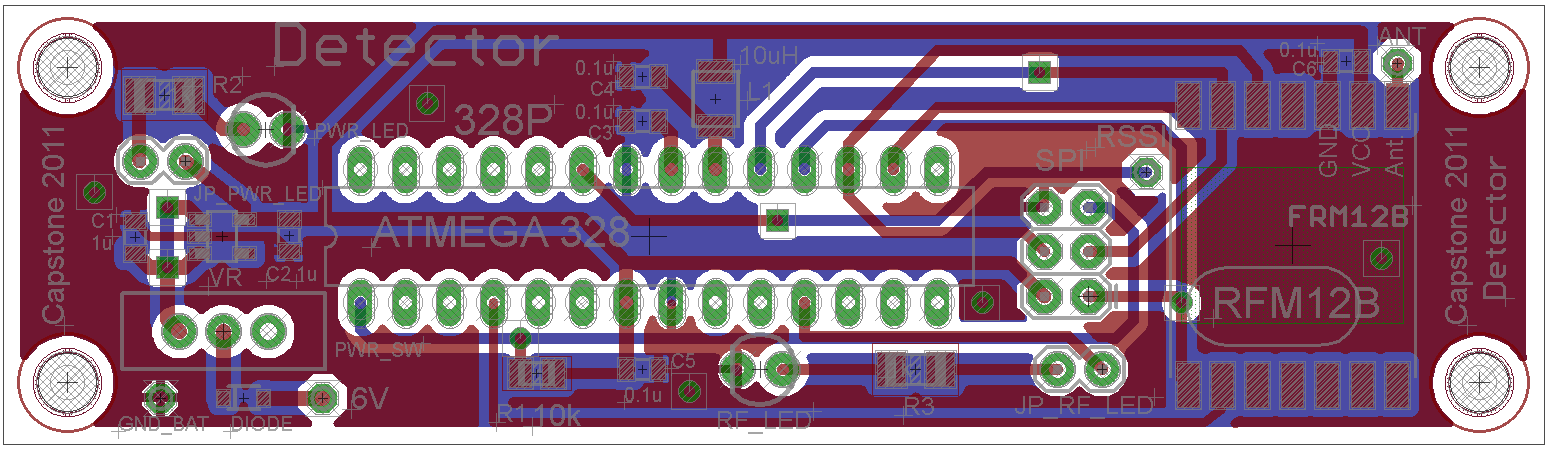
* By including a 3.3V regulator, the detector can be powered a DC power adapter, or various types of 3.6 - 16V batteries.
* The RFM12B module is a low cost option with sufficient power and range to provide reliable communication around the lab - a basic packet protocol can be implemented in just 2..3 Kb of C code.
* Had 2 LEDs for debugging power and signal transmission.
* Small size 1” x 3.5”.
* There are many possible microcontrollers we could use to implement a solution. The ATMega328p is a good choice because they are supported in Arduino platform.
* RF12B transceiver is a good choice because of low price, good support and low power consumption.

#### Schematic



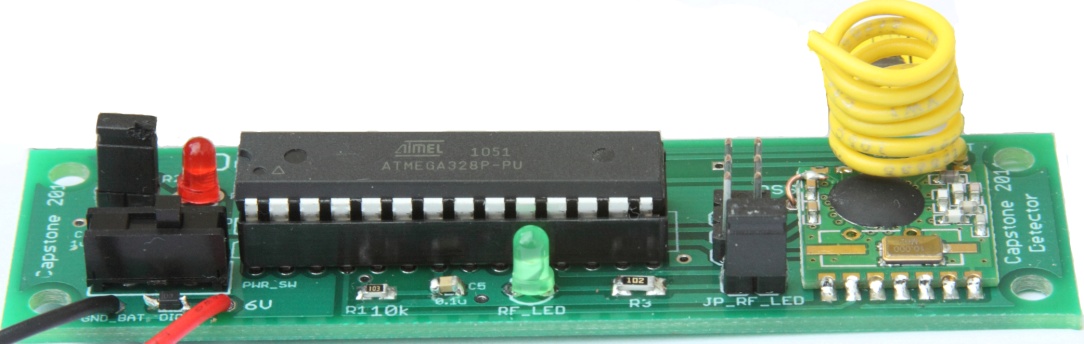
Schematics of a detector

#### Board Layout



Two-layer PCB layout of a detector with shown ground plane

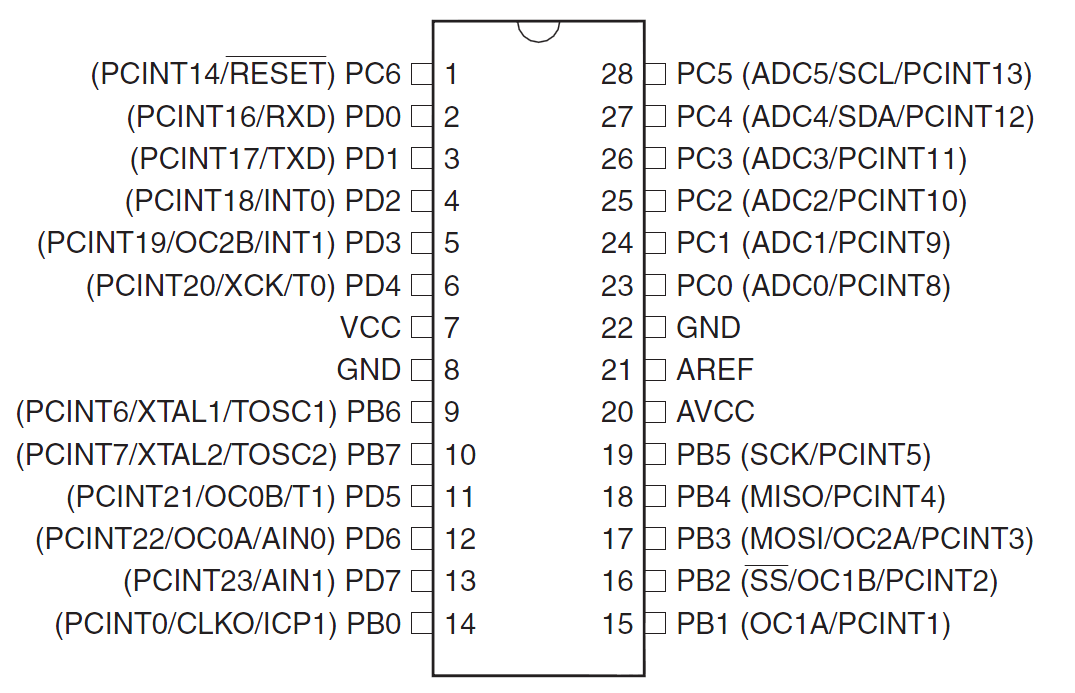
#### Real Board



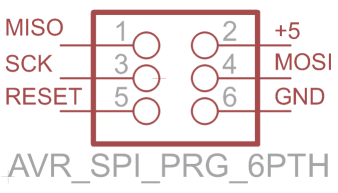
PCB prototype of a detector

#### Port/Pin Mapping

**ATMega328P – 28 pins**

****

**SPI connector**

****

|  |  |  |
| --- | --- | --- |
| Pin | Name | Description |
| 1 | MISO | master in / slave out |
| 2 | +5V | Regulated +5V |
| 3 | SCK | SPI clock |
| 4 | MOSI | master out / slave in |
| 5 | RESET | Reset |
| 6 | GND | ground |

**RFM12B**

****

|  |  |  |
| --- | --- | --- |
| Pin | Name | Description |
| 1 | SDO | Serial data output |
| 2 | nIRQ | Interrupts request output |
| 3 | FSK/DATA/nFFS | Transmit FSK data input/Received data output/FIFO select |
| 4 | DCLK/CFIL/FFIT | Clock output/Extrenal filter capacitor/FIFO interrupts |
| 5 | CLK | Clock output for external microcontroller |
| 6 | nRES | Reset output |
| 7 | GND | Ground |
| 8 | ANT | Antenna |
| 9 | VDD | VCC |
| 10 | GND | Ground |
| 11 | nINT/VDI | Interrupt input |
| 12 | SDI | SPI data input |
| 13 | SCK | SPI clock input |
| 14 | nSEL | Chip select |

#### Part List

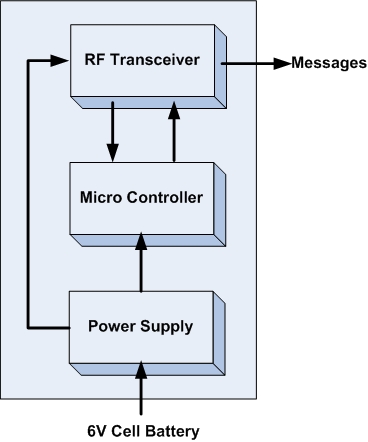
|  |  |  |
| --- | --- | --- |
| Part | Value | Details |
| ATMega328 | Atmel's ATMega328 | Atmel's ATMega328 8-Bit Processor in 28 pin DIP |
| RFN12B | RFM12B-S2 | RFM12B-S2 Wireless Transceiver |
| Socket IC | Socket for ATMega328p | Socket IC open frame 28Pos 0.3” |
| SPI | SPI 6 pins connector | 6 POS connector header |
| PWR\_SW | Power switch | SPDT Mini Power Switch |
| VR | Voltage regulator 3.3V | IC VREG LDO 3.3V |
| DIODE | Diode | DIODE SBR 1A 30V SOD-323 |
| C1, C1 | 1uF | CAP CERM 1UF 25V Y5V 0805 |
| C3, C4, C5, C6 | 0.4uF | CAP CERM .10UF 50V 5% 0805 SMD |
| L1 | 10uH | INDUCTOR 10UH 10% SA TYPE SMD |
| R1 | 10K | RES 10K OHM 1/4W 5% 0805 SMD |
| PWR\_LED | Power LED 3mm | Power LED 3mm |
| RF\_LED | LED 3mm | LED for signal transmission |
| JP\_RF\_LED | 2 pins male header | 2 pins male header |
| JP\_PWR\_LED | 2 pins male header | 2 pins male header |

### Asset Tags

An asset tag consists of two major parts: a microcontroller, and a RF transceiver. The microcontroller controls the RF transceiver. The RF transceiver is responsible for communicating information to the system.

Tags are attached to TIUs. They are the beacon with which locations can be calculated. It is critically important that tags preserve power. Therefore they are typically in a low power state, but periodically wake up to broadcast, and then immediately go back to their low power state

#### Functionality



|  |  |
| --- | --- |
| Module | Asset Tags |
| Inputs | Power supply: 3.3V DC |
| Outputs | Messages containing tag’s ID |
| Functionality | An asset tag consists of two major parts: a microcontroller, and a RF transceiver. The microcontroller controls the RF transceiver. The RF transceiver is responsible for communicating information to the system. |

#### What is on the Board?

* A power switch on/off
* A 3.3V power regulator which accepts 2.2V to 6V as external power source.
* An ATMega328p MCU by Atmel with 8MHz
* A 6pins SPI connector for programming.
* A RFM12B wireless RF module for 434MHz ISM band by Hope RF.
* A jumper which connect to DP3 pin of ATMega328p for debug/mode.
* 2x7pins headers female for connecting with RFM12B breakout board.

#### Advantages

* Designed from the ground up to support very low-power use with batteries.
* The RFM12B module is a low cost option with sufficient power and range to provide reliable communication around the lab - a basic packet protocol can be implemented in just 2..3 Kb of C code.
* Very small size 1” x 1”
* Very low power consumption
* There are many possible microcontrollers we could use to implement a solution. The ATmega 328p surface mount packet is a good choice because they are the Arduino technology platform and very small size.
* FR12B transceiver is a good choice because of low price, good support and low power consumption

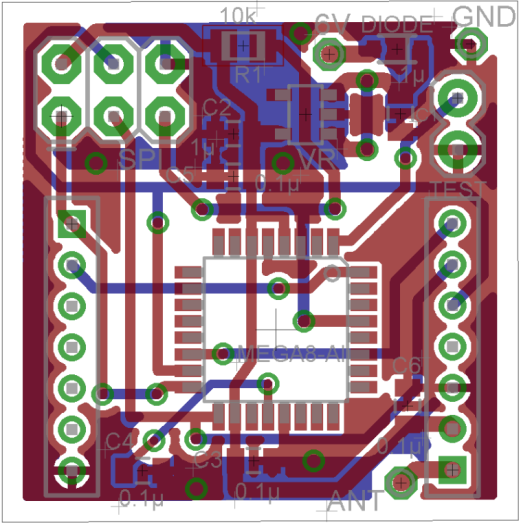
#### Battery Life

#### Schematic



Schematics of a tag

#### Board Layout



Two-layer PCB layout of a tag with shown ground plane

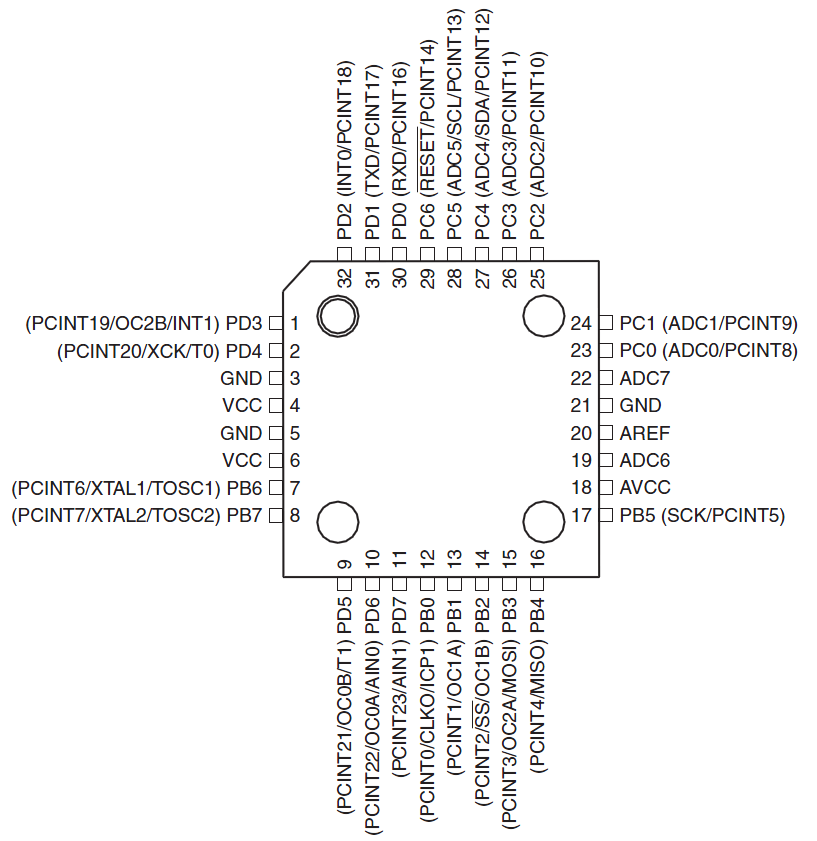
#### Real board



Asset Tag: main board with MCU and breakout board with RF chip

#### Port/Pin Mapping

**ATMega328P – 32 pins**

****

ATMega328P 32TQFP pin-out (top view)

#### Part List

|  |  |  |
| --- | --- | --- |
| Part | Value | Details |
| ATMega328 | IC MCU AVR 32K FLASH 32TQFP | IC MCU AVR 32K FLASH 32TQFP |
| SPI | SPI 6 pins connector | 6 POS connector header |
| VR | Voltage regulator 3.3V | IC VREG LDO 3.3V |
| DIODE | Diode | DIODE SBR 1A 30V SOD-323 |
| C1, C1 | 1uF | CAP CERM 1UF 25V Y5V 0805 |
| C3, C4, C5, C6 | 0.4uF | CAP CERM .10UF 50V 5% 0805 SMD |
| R1 | 10K | RES 10K OHM 1/4W 5% 0805 SMD |
| LEFT | 2mm 7-pin Socket | 2mm 7-pin Socket |
| RIGHT | 2mm 7-pin Socket | 2mm 7-pin Socket |
| TEST | 2-pins header | 2-pins header |
| Battery | Coin Cell battery 20mm 240mAh | Coin Cell battery 20mm 240mAh |

### RFM12B Breakout Board

#### What is on the Board?

* A RFM12B wireless RF module for 434MHz ISM band by Hope RF.
* An copper straight-wire antenna.

#### Advantages

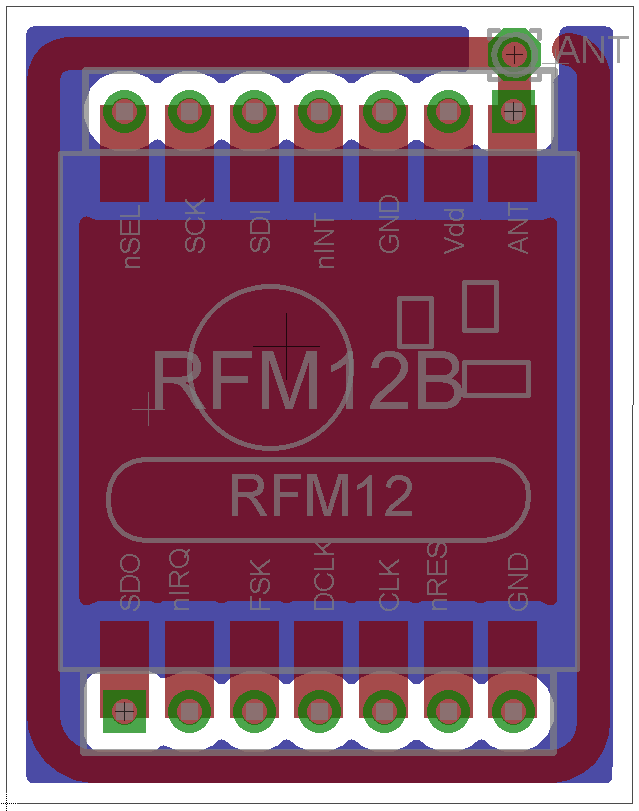
* The breakout board for the RFM12B which gives you access to the pins.
* Easy to solder. And implement.
* Had an antenna on it.
* Very small size 0.8” x 0.9”

#### Schematic



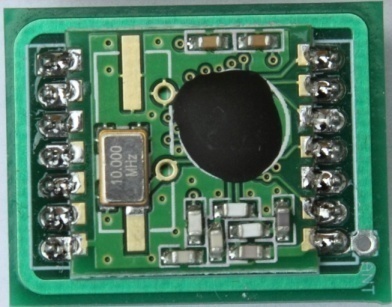
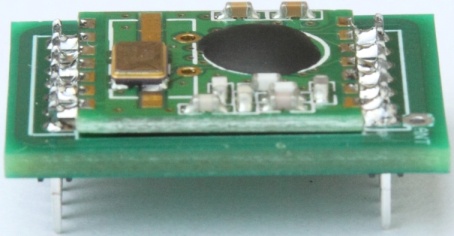
Schematics of a breakout board

#### Board Layout



PCB layout of a breakout board with shown ground plane

#### Real Board



#### Part List

|  |  |  |
| --- | --- | --- |
| Part | Value | Description |
| RFN12B | RFM12B-S2 | RFM12B-S2 Wireless Transceiver |
| LEFT | 2mm 7-pin male header | 2mm 7-pin male header |
| RIGHT | 2mm 7-pin male header | 2mm 7-pin male header |

## Useful Links and Reference

1. “TIU Tracking System Demo”, http://www.youtube.com/watch?v=sZbXoZNrWNc
2. “Web app Demo”, http://www.youtube.com/watch?v=XX9sif2QWpw&feature=related
3. “ISP mkII Programmer User Guide”, http://www.atmel.com/dyn/resources/prod\_documents/AVRISPmkII\_UG.pdf
4. “AVR studio”, http://www.atmel.com/dyn/products/tools\_card.asp?tool\_id=2725&category\_id=163&family\_id=607&subfamily\_id=760
5. “Arduino environment”, <http://www.arduino.cc/en/Main/Software>
6. “Source code and documentation”, <http://tiu-tracking.googlecode.com/svn/trunk>