Automated Wireless Asset Tracking for Underground Mines

Final Report

For Tracking & Monitoring Software

Version 1.0

By: Philip Kurowski

Supervisor: Dr. Abdallah Shami

Table of Contents

[1. Abstract 4](#_Toc416027094)

[2. Literature Review 5](#_Toc416027095)

[3. Modern Software Requirements Specification 6](#_Toc416027096)

[3.1 Introduction 6](#_Toc416027097)

[3.2 Overall Description 8](#_Toc416027098)

[3.3 Requirements 10](#_Toc416027099)

[4. Software Design Specification 14](#_Toc416027100)

[4.1 Introduction 14](#_Toc416027101)

[4.2 Logical Architecture 15](#_Toc416027102)

[4.3 Detailed Description of Components 16](#_Toc416027103)

[4.4 Design Rationale 43](#_Toc416027104)

[5. Walkthrough Report 44](#_Toc416027105)

[6. Software Testing Plan 45](#_Toc416027106)

[6.1 Introduction 45](#_Toc416027107)

[6.2 Relationship to other documents 45](#_Toc416027108)

[6.3 System overview 45](#_Toc416027109)

[6.4 Features to be tested/not to be tested 45](#_Toc416027110)

[6.5 Pass/Fail criteria 46](#_Toc416027111)

[6.6 Approach 46](#_Toc416027112)

[6.7 Suspension and resumption 46](#_Toc416027113)

[6.8 Testing materials (hardware/software requirements) 46](#_Toc416027114)

[6.9 Test cases 46](#_Toc416027115)

[6.10 Testing schedule 46](#_Toc416027116)

[7. Test Case Specifications 48](#_Toc416027117)

[7.1 Master Module 48](#_Toc416027118)

[7.2 Tracking Module 50](#_Toc416027119)

[7.3 Messaging Module 51](#_Toc416027120)

[7.4 Reports Module 52](#_Toc416027121)

[7.5 Attendance Module 52](#_Toc416027122)

[8. Conclusion / Recommendations 53](#_Toc416027123)

[9. Gantt chart 54](#_Toc416027124)

[10. References 55](#_Toc416027125)

[11. Appendix A: Use Case Specification Documents 56](#_Toc416027126)

[11.0 Use Case Specification: Assign Shift 56](#_Toc416027127)

[11.1 Use Case Specification: Assign Tag 57](#_Toc416027128)

[11.2 Use Case Specification: Broadcast Text 58](#_Toc416027129)

[11.3 Use Case Specification: Create Alert 59](#_Toc416027130)

[11.4 Use Case Specification: Create Attendance Report 60](#_Toc416027131)

[11.5 Use Case Specification: Create Member 61](#_Toc416027132)

[11.6 Use Case Specification: Create Message Details Report 62](#_Toc416027133)

[11.7 Use Case Specification: Create Miner Report 63](#_Toc416027134)

[11.8 Use Case Specification: Create Miner Block Report 64](#_Toc416027135)

[11.9 Use Case Specification: Create Miner Position Report 65](#_Toc416027136)

[11.10 Use Case Specification: Create Router Report 66](#_Toc416027137)

[11.11 Use Case Specification: CreateSensor 67](#_Toc416027138)

[11.12 Use Case Specification: Create Tag 68](#_Toc416027139)

[11.13 Use Case Specification: Create Tag Report 69](#_Toc416027140)

[11.14 Use Case Specification: Create User 70](#_Toc416027141)

[11.15 Use Case Specification: Create Vehicle Operation Report 71](#_Toc416027142)

[11.16 Use Case Specification: Daily Report 72](#_Toc416027143)

[11.17 Use Case Specification: Load Map 73](#_Toc416027144)

[11.18 Use Case Specification: Log In 74](#_Toc416027145)

[11.19 Use Case Specification: Monthly Report 75](#_Toc416027146)

[11.20 Use Case Specification: Receive Message 76](#_Toc416027147)

[11.21 Use Case Specification: View Member Position 77](#_Toc416027148)

[11.22 Use Case Specification: View Path Chart 78](#_Toc416027149)

[11.23 Use Case Specification: Yearly Report 79](#_Toc416027150)

[12. Appendix B: Test Case Results 80](#_Toc416027151)

[12.1 Master Module 80](#_Toc416027152)

[12.2 Tracking Module 80](#_Toc416027153)

[12.3 Reports Module 80](#_Toc416027154)

[12.4 Attendance Module 80](#_Toc416027155)

[13. Appendix C: Source Code 81](#_Toc416027156)

[14. User Manual 82](#_Toc416027157)

[15. Vitae 83](#_Toc416027158)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 05/04/15 | 1.0 | Finished test case specifications for reports, attendance modules | Philip Kurowski |
| 04/04/15 | 1.0 | Finished test case specifications for master, tracking, and messaging modules | Philip Kurowski |
| 03/04/15 | 1.0 | Finished test plan | Philip Kurowski |
| 10/03/15 | 1.0 | Added abstract | Philip Kurowski |
| 15/02/15 | 1.0 | Updated design documents | Philip Kurowski |
| 13/02/15 | 1.0 | Updated requirements: software interfaces | Philip Kurowski |
| 23/11/14 | 1.0 | Partially filled out test plan  Added literature review  Added design rationale  Added walkthrough report | Philip Kurowski |
| 20/11/14 | 1.0 | Added sequence diagrams to section 4.3.6  Completed SDS | Philip Kurowski |
| 14/11/14 | 1.0 | Completed SDS Introduction  Added class diagrams to section 4.3.1 – 4.3.5 | Philip Kurowski |
| 12/11/14 | 1.0 | Completed applicable standards.  Completed SRS. | Philip Kurowski |
| 11/11/14 | 1.0 | Completed section 1  Added UCD to section 2  Completed section 3 (minus interfaces, applicable standards, and functionality) | Philip Kurowski |
| 10/10/14 | 1.0 | Filled out SRS introduction. | Philip Kurowski |
| 29/09/14 | 1.0 | Filled out fields. | Philip Kurowski |
|  |  |  |  |
|  |  |  |  |

# Abstract

Underground mining sites are hazardous environments that require reliable communication system throughout sites to ensure consistent functioning of mining equipment and miners’ safety. While mine-wide communications networks are typically wired, this communication principle is both difficult to deploy and unable to withstand disaster situations that may occur underground. This project is application layer software that provides full visualization and interfacing with an underground mine that has established ZigBee communications with routers and end-devices. A user is able to receive network information through the Tracking and Monitoring Software and easily find miners, track miners’ paths, send alert messages, and send broadcast messages to communicate with miners. The software also contains organizational features such as the ability to produce attendance and equipment-use reports meant to achieve smooth business processes. The Tracking and Monitoring Software serves as an all-purpose safety and business interface tool between aboveground users and underground miners that will allow immediate response to hazardous mining events.

Keywords: ZigBee, mining, tracking, monitoring

# Literature Review

Underground mines are hazardous environments with tough working conditions. There is a strong need for reliable mine-wide communications systems in order to achieve smooth functioning of mining equipment and ensure miners’ safety. Most existing systems for underground mines are based on wired communication principle. Wired communications are both difficult to deploy in inaccessible locations and are unable to withstand the potential disaster conditions that arise working underground.

There exist wireless-based technology for tracking controls and sensors in networks; one of these includes OpenGN. OpenGN is a software system made for controlling and monitoring building systems like lighting, fire alarms, or security systems. While being able to monitor controls in an area, OpenGN does not provide the functionality of installing movable sensors that is necessary in tracking the many moving miners and mining vehicles found in an underground mine.

In terms of network technologies, there exist Bluetooth and WiFi, that could be used to connect underground network devices but these are specialized for mid to high rates and include voice, video, or other capabilities. While the features are many, most are unnecessary in tracking controls and sensors. The ZigBee standard is specifically aimed at at control-sensor networks as a cost effective low rate solution.

This project will interface with a ZigBee network and provide full tracking capabilities not found in other alternatives. This will address the issues found in other alternatives by specifically targeting a mining scenario instead of a general building / wireless automation system.

# Modern Software Requirements Specification

## Introduction

### Purpose

This software requirements specification describes the external behavior of the tracking and monitoring software system. It also describes nonfunctional requirements, design constraints, and the other factors integral in the description of the tracking and monitoring software.

### Scope

This SRS applies to the Tracking and Monitoring Software used in a ZigBee-based automated wireless asset tracking system for underground mines. The software is associated with the use case diagram found in [**section 2.0.1.4**](#_Diagrams_of_the).

The completed application will let users:

* Track positions of miners and mining vehicles.
* Project miner and mining vehicle paths on a map.
* Specify ZigBee networks nodes including routers, end devices, and sensors.
* Generate attendance reports.
* Generate reports based on network node information, miner information, and vehicle information.
* Communicate via text with miners.

The completed application will not:

* Allow communicate via speech
* Automate communications to end devices (done in coordinator firmware)

### Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Coordinator** | A device that maintains all network knowledge. There is only one in the ZigBee network. |
| **End Device** | A device in end device mode. It can only communicate with coordinators. |
| **FFD** | Full-function device, can be implemented as a coordinator or an end device. |
| **Git / GitHub** | Source control and online hub that will host versions of the described software codebase. |
| **RFD** | Reduced-function device, can only be implemented as an end device. |
| **Router** | In this scope, a router is any FFD that connects to the coordinator. Routers are stationary and provides connectivity for the coordinator and end devices. |
| **TMS** | Acronym for Tracking & Monitoring Software, the software described in this SRS. |
| **Visual Studio** | Integrated development environment for Windows applications. Includes C# integration. |
| **ZigBee** | A wireless packet transfer specification built on IEEE 802.15.4 that uses a single coordinator, multiple routers and multiple end devices to build a low rate control / sensor network. |

### References

The referenced documents are use case specification documents and can be found in the [Use Case Documents](#_Use_Case_Description) section of the report.

|  |  |  |
| --- | --- | --- |
| **Document No.** | **Document Title** | **Date** |
| 1 | AssignShift.docx | 10/05/14 |
| 2 | AssignTag.docx | 10/05/14 |
| 3 | BroadcastText.docx | 10/10/14 |
| 4 | CreateAlert.docx | 10/05/14 |
| 5 | CreateAttendanceReport.docx | 10/17/14 |
| 6 | CreateMember.docx | 10/14/14 |
| 7 | CreateMessageDetailsReport.docx | 10/20/14 |
| 8 | CreateMinerReport.docx | 10/20/14 |
| 9 | CreateMinerBlockReport.docx | 10/20/14 |
| 10 | CreateMinerPositionReport .docx | 10/20/14 |
| 11 | CreateRouterReport .docx | 10/20/14 |
| 12 | CreateSensor.docx | 10/11/14 |
| 13 | CreateTag.docx | 10/05/14 |
| 14 | CreateTagReport.docx | 10/20/14 |
| 15 | CreateUser.docx | 10/20/14 |
| 16 | CreateVehicleOperationReport.docx | 10/15/14 |
| 17 | DailyReport.docx | 10/18/14 |
| 18 | LoadMap.docx | 10/09/14 |
| 19 | LogIn.docx | 10/07/14 |
| 20 | MonthlyReport.docx | 10/18/14 |
| 21 | ReceiveMessage.docx | 10/16/14 |
| 22 | ViewMemberPosition.docx | 10/07/14 |
| 23 | ViewPathChart.docx | 10/05/14 |
| 24 | YearlyReport.docx | 10/18/14 |

### Overview

The rest of the SRS examines the specifications of the Tracking and Monitoring Software. The second section of the SRS describes the use case model and core functionality from a high level point of view. The third section of the SRS will go over the specific program requirements of the Tracking and Monitoring Software, including detailed descriptions of functional, and non-functional requirements along with constraints, licensing, and legal notices.

## Overall Description

### Use-Case Model Survey

#### Introduction

The use-case model provides an overview of the tracking and monitoring software’s core functionalities from a system perspective. The separation of user access levels is shown as are all features of the software that are both initiated by network events and initiated by the user. Use-cases are divided into 5 modules: Master, Tracking, Messaging, Attendance, and Reports.

The use-case model contains the four system actors including a user, an administrator, the network coordinator, and the TMS server itself. Exceptional cases are shown by extends stereotypes and include cases of different attendance reports.

#### Use-Case Model Hierarchy

##### Tracking & Monitoring Software

The package contains all system modules including those used for administrative tasks, receiving & displaying tracking data, and generating reports. The software functions as a Windows application that automatically tracks ZigBee network devices and lets a user perform all-purpose monitoring of miners and vehicles in an underground mine.

The system actors are:

* **User**: A person that is using the TMS to retrieve data about the mining site and monitor operations underground.
* **Admin**: A user with administrative authority. This type of user can create other user account and is preinstalled in the system.
* **Coordinator**: The ZigBee network coordinator interacts with the TMS to send and receive messages to and from the network.

#### Diagram of the Use-Case Model



Figure 1: The TMS use case model

### Assumptions and Dependencies

#### Coordinator is up-to-date

The FFD acting as the coordinator is up to date with the latest firmware that can communicate with the TMS.

#### Server can connect

The server running the TMS has the appropriate ports to communicate with the network.

## Requirements

### Use-Case Specifications

All use cases and their corresponding description documents can be found in the [Use Case Specification Documents](#_Use_Case_Specification) section of the report.

### Functionality

#### Display mine plan and structure

The image containing the mining site plan will be displayed along with all routers set in the site.

#### Display Path Chart for Miners

A miner’s path can be drawn on the mining site plan. The path is the visited route of routers that the miner has visited.

#### Display Miner Position

A miner’s position can be highlighted on the map

#### Generate Attendance Reports

Available attendance reports include daily, monthly, and yearly reports for miner attendance and general operation time reports for vehicles.

#### Generate Network Node Reports

Available reports include miners, miner’s position, miners blocked, router, tag, and sent / received message reports.

#### Add / Edit Network Nodes

Routers, sensors, and end devices can be added and edited from the application.

### Usability

#### Training Time

Under an hour to be able to add and edit components, and monitor data.

#### User Interface & Usability Design Standards

Spacing & positioning, size, grouping, and intuitiveness will all be applied in forms according to this standard: <http://msdn.microsoft.com/en-us/library/aa468595.aspx#humanux_topic2>

### Reliability

##### *Availability*

TMS server should be up 99.99% of the time.

### Performance

#### Throughput

At least 10 updates / second should be received from the coordinator to allow real time tracking.

### Supportability

#### MSDN C# Coding Conventions

Microsoft’s C# programming guide as shown as outlined:

<http://msdn.microsoft.com/en-us/library/ff926074.aspx>

### Design Constraints

#### ZigBee Network

The physical system will be a network of a coordinator, several routers and many end devices as described in ZigBee specifications.

#### C# Language

All application layer code will be written in C# as a Windows Forms application.

#### Microsoft Visual Studio 2013

The application source code will be a visual studio solution.

### Online User Documentation and Help System Requirements

N/A

### Purchased Components

#### Full Function Devices (FFDs)

Multiple FFDs will be programmed and used for testing as a coordinator and routers to simulate an underground mine.

#### Reduced Function Devices (RFDs)

Multiple RFDs will be used to test end device functionality.

### Interfaces

#### User Interfaces

##### Master / Tracking Interface

* This will be the main view that is open at all times of operation. From this view, we can access all other views via a menu bar on top of the screen.
* The tracking module will be contained on the right side of the screen as a map image above a legend. The legend shows the images that represent routers, blocked routers, and active miners. Clicking on routers or miners in the map will display a small box that shows information about said router or miner.

##### Messaging Interface

* This interface will appear on the master interface at the top right.
* It will contain a button to send a text and a button to open the list of received texts.
* The sending text window will contain a field for a simple text message and a list of miners to send to.
* The received texts window will contain a list ordered by date received that will show the message text when clicked.

##### Reports Interface

* A simple window containing a drop down of the report types on the left. Selecting a report type will reveal the input method to input the valid parameters for the report.
* When all fields are filled out, a generate report button will be enabled.

##### Attendance Interface

* A simple window containing a list of miner names and a checkbox group of the different attendance report types (Daily, Monthly, Yearly).
* Depending on the report type, the user will be presented with a date time picker, a month picker, or a year picker for daily, monthly, and yearly reports respectively.
* There will be two tabs: Miner, as described above, and Vehicle, which will allow selecting a vehicle and generating an operation time report.

#### Hardware Interfaces

N/A

#### Software Interfaces

##### Coordinator firmware

The TMS will communicate directly with the network coordinator using a custom protocol to exchange network topology information.

###### Path Messages

Messages received from the coordinator will include an end device tag ID and a router ID.

|  |  |
| --- | --- |
| **Router ID** | **End Device ID** |
| 0x0000 | 0x0000 |

###### Alert Messages

Messages sent to the coordinator with a value of 0 as the initial byte will represent a single alert to the specified end device.

|  |  |
| --- | --- |
| **Message type** | **Tag ID** |
| 0x1 | 0x0000 |

###### Broadcast Messages

Messages sent to the coordinator with a value of 1 as the initial byte will represent a mine-wide alert to all end devices.

|  |  |
| --- | --- |
| **Message type** | **Other content** |
| 0x1 | … |

#### Communications Interfaces

##### USB

The port at which the network coordinator will connect to the server hosting the tracking and monitoring software.

### Licensing Requirements

* Philip Kurowski hereby grants Mircom Group a worldwide, perpetual, non-exclusive, non-transferable license to all software for Mircom Group’s use in connection with the establishment, use, maintenance and modification of the system implemented by Philip Kurowski. Software shall mean executable object code of software programs and the patches, scripts, modifications, enhancements, designs, concepts or other materials that constitute the software programs necessary for the proper function and operation of the system as delivered by Philip Kurowski and accepted by the Customer.
* Any and all licenses, product warranties or service contracts provided by third parties in connection with any software, hardware or other software or services provided in the system shall be delivered to Mircom Group for the sole benefit of Mircom Group.
* Mircom Group may supply to Philip Kurowski or allow Philip Kurowski to use certain proprietary information, including service marks, logos, graphics, software, documents and business information and plans that have been authored or pre-owned by Mircom Group. All such intellectual property shall remain the exclusive property of Mircom Group and shall not be used by Philip Kurowski for any purposes other than those associated with delivery of the system.

### Legal, Copyright and Other Notices

* Customer may make copies of the software for archival purposes and as required for modifications to the system. All copies and distribution of the software shall remain within the direct control of Customer and its representatives.
* Customer may make modifications to the source code version of the software, if and only if the results of all such modifications are applied solely to the system. In no way does this Software License confer any right in Customer to license, sublicense, sell, or otherwise authorize the use of the software, whether in executable form, source code or otherwise, by any third parties, except in connection with the use of the system as part of Customer’s business.
* All express or implied warranties relating to the software shall be deemed null and void in case of any modification to the software made by any party other than Philip Kurowski.
* Philip Kurowski represents and warrants to Mircom Group that:
* it has all necessary rights and authority to execute and deliver this Software License and perform its obligations hereunder and to grant the rights granted under this Software License to Customer;
* the goods and services provided by contractor under this Software License, including the software and all intellectual property provided hereunder, are original to Philip Kurowski or its subcontractors or partners; and
* The software, as delivered as part of the system, will not infringe or otherwise violate the rights of any third party, or violate any applicable law, rule or regulation.
* Philip Kurowski further represents and warrants that, throughout the System Warranty Period, the executable object code of software and the system will perform substantially in accordance with the System Specifications and Agreement. If the software fails to perform as specified and accepted all remedies are pursuant to the policies set forth in the Specification and in the Agreement. **No warranty of any type or nature is provided for the source code version of the software which is delivered as is.**
* Except as expressly stated in this Agreement, there are no warranties, express or implied, including, but not limited to, the implied warranties of fitness for a particular purpose, of merchantability, or warranty of no infringement of third party intellectual property rights.

### Applicable Standards

#### C# Coding Standards

All code will adhere to Microsoft’s outlines C# coding conventions:

<http://msdn.microsoft.com/en-us/library/ff926074.aspx>

#### Windows Forms Usability Standards

All UI and windows forms will take into account Microsoft’s UI and Usability standards:

<http://msdn.microsoft.com/en-us/library/aa468595.aspx#humanux_topic2>

# Software Design Specification

## Introduction

### Purpose of this document

The main objectives of the SDS are to provide both a high level description of the system architecture and enough low level details to allow a developer to fully implement the system. The role of each module will be described to provide a high level description of the system and the connectivity between the modules. Each module will then be described by a class diagram and a detailed overview will be presented for each module’s class’ attributes and operations to describe implementation.

### Scope of the development project

The scope defined in [section 1.0.2](#_Scope) applies to the SDS as well.

### Definitions, acronyms, and abbreviations

All definitions found in [section 1.0.3](#_Definitions,_Acronyms_and) apply to the SDS as well.

### References

The SRS associated with this design can be found [prepended to this document](#_Modern_Software_Requirements). Use case specifications are located at the [end of the report](#_Use_Case_Description).

### Overview of document

The rest of the SDS examines the design of the Tracking and Monitoring Software. The second section of the SDS describes the high level design of the TMS’ modules and is followed by a detailed look into the classes associated with each module. The final sections of the design document contains the reasoning for various choices related to architecture and class structure that were made throughout the system design phase.

## Logical Architecture

### Overview

The Tracking & Monitoring Software’s logical architecture is divided into five modules: a Master module for general purpose management tasks, a Tracking module for tasks dealing with observing the mining site network data, a Messaging module for text communication with end devices in the network, and Report and Attendance modules for generating reports based on network data and member attendance. See [section 2.0.1.](#_Diagram_of_the)4 for overall use case diagram.

### Master Module

General management tasks, this module accesses the system database to insert, update, and delete data related to members, network nodes, and users.

### Tracking Module

Primary data visualization, provides an interface for viewing and receiving network data received from the coordinator.

### Messaging Module

Contains data and functions that allow the user to communicate with miners via text messaging to end devices.

### Attendance Module

Provides an interface for the user to generate reports on individual miner attendance.

### Reports Module

Provides an interface for the user to generate reports on all miners, routers, and tags. Manipulates and organized data needed for the reports into an easily readable format.

## Detailed Description of Components



Figure 2: The full system class diagram.

### Master Module Class Diagram



Figure 3: Master module class diagram

#### MasterController

The controller for the master module. This performs all the tasks in the master module of the use case diagram and serves as a façade to access all other modules.

##### MasterController Attributes

N/A

##### MasterController Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| assignShift(member : Member, shift : Shift) | Adds the shift to the specified member’s list of shifts and saves the shift to the database. |
| createMember() | Adds a new member entry to the system database. |
| createSensor() | Adds a new sensor entry to the system database. |
| createUser() | Adds a new User entry to the system database. |
| createTag(tagId : String) | Adds a new Tag entry to the system database. |
| loadMap() : void | Displays a map image on the tracking panel. |
| logIn(username : string, pass : string) | Authenticates the user, closes the login form and opens the mainform if authentication works. |
| updateMember() | Edits a member’s data in the database. |
| updateSensor() | Edits a sensor’s data in the database. |
| updateUser() | Edits a member’s data in the database. |

##### MasterController Design Specification/Constraints

N/A

##### MasterController States and Transitions

N/A

#### LoginForm

The view class providing an interface for the user to log in.

#### LoginForm Attributes

N/A

##### LoginForm Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| OnLoginPress() | Delegate for when the user presses a login button in the view. |

##### LoginForm Design Specification/Constraints

N/A

##### LoginForm States and Transitions

N/A

#### MainForm

The controller for the master module. This performs all the tasks in the master module of the use case diagram and serves as a façade to access all other modules.

##### MainForm Attributes

N/A

##### MainForm Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| OnAttendancePress() | Delegate for when the user presses a button to enter the attendance module. |
| OnLoadMapPressed() | Delegate for when the user presses a button to open a map loading interface. |
| OnMessagingAlertPress() | Delegate for when the user presses a button to open an alert creation interface. |
| OnMessagingBroadcastPress() | Delegate for when the user presses a button to open a message broadcasting interface. |
| OnReportsPress() | Delegate for when the user presses a button to enter the reports module. |
| OnTrackingMapClick() | Delegate for when the user presses a button to enter the tracking module. |

##### MainForm Design Specification/Constraints

N/A

##### MainForm States and Transitions

N/A

#### MemberForm

The main interface for creating and editing members. This form allows users to get to forms for assigning shifts and tags to members.

##### MemberForm Attributes

N/A

##### MemberForm Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| OnAssignShiftPress() | Delegate for when the user presses a button to open a shift assignment form. |
| OnAssignTagPress() | Delegate for when the user presses a button to open a tag assignment form. |

##### MemberForm Design Specification/Constraints

N/A

##### MemberForm States and Transitions

N/A

#### ShiftForm

The interface that lets a user specify a range for a member’s work shift and assign it to the member.

##### ShiftForm Attributes

N/A

##### ShiftForm Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| OnAssignPress() | Delegate for the form’s main confirm button. Pressing this calls the controller to assign a shift to the member that this form’s MemberForm describes. |

##### ShiftForm Design Specification/Constraints

N/A

##### ShiftForm States and Transitions

N/A

#### TagForm

The interface that lets a user enter data to create a new tag.

##### TagForm Attributes

N/A

##### TagForm Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| OnUpdateButtonPress() | Delegate for the form’s main confirm button. Pressing this calls the controller to verify that the entered tag is valid and generate the tag. |

##### TagForm Design Specification/Constraints

N/A

##### TagForm States and Transitions

N/A

#### UserForm

The interface that lets an admin user enter data to create other user accounts. It contains textfields for necessary user data.

##### UserForm Attributes

N/A

##### UserForm Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| OnUpdateButtonPress() | Delegate for the form’s main confirm button. Pressing this calls the controller to create a database row for a new user with the data entered on this form. |

##### UserForm Design Specification/Constraints

N/A

##### UserForm States and Transitions

N/A

### Tracking Module Class Diagram



Figure 4: Tracking module class diagram

#### TrackingController

The controller for the tracking module. This performs all the tasks in the tracking module of the use case diagram and serves as the entry point of incoming communication from the coordinator.

##### TrackingController Attributes

N/A

##### TrackingController Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| listenForCoordinator(portNo : int) | Opens a port to continuously listen to for messages from the coordinator. |
| parseMessage(message : Byte[]) | Given a message from the coordinator, this function updates members’ router path to give a live position tracking. |
| showMemberPosition() | This instantiates a RouterInfoForm appropriately on the map that displays the position of the router that the specified member is at. |
| showPath(member : Member) | Determines the member’s router history and draws a path on the map. |

##### TrackingController Design Specification/Constraints

N/A

##### TrackingController States and Transitions

N/A

#### Map

Model representing the mine plan map that will show the mining facility’s tunnels. This will be drawn in the tracking panel and all routers and members will be drawn to scale on top of it.

##### Map Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| mapPlan | Image | The raw image data to be displayed on the tracking panel. This is an 800 x 700 pixel image of the mine site plan. |
| Scale | float | The pixel : meter scale of the map plan. |

##### Map Operations

N/A

##### Map Design Specification/Constraints

* Image constrained to 800 x 700 pixels

##### Map States and Transitions

N/A

#### Member

Model representing a miner’s or a mining vehicle’s employment and personal data.

##### Member Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| memberNo | String |  |
| fName | String |  |
| mName | String |  |
| lName | String |  |
| Address | String |  |
| Province | String |  |
| City | String |  |
| pinNo | Int |  |
| phoneNo | String |  |
| mobileNo | String |  |
| returnDate | Date |  |
| isVehicle | Boolean | Method of differentiating between miners or vehicles. This is true if the member is a miner, false if a vehicle. |

##### Member Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| appendRouter(router : Router) | Appends the inputted router to the member’s router path list. |
| getRouterPath() | Returns the member’s list of routers that they have connected to. The last router in the list is the member’s current position. |

##### Member Design Specification/Constraints

N/A

##### Member States and Transitions

N/A

#### Router

Model representing the physical location and properties of a ZigBee router that has been placed in the underground mine. This is updated periodically using data from incoming messages sent by the coordinator to the TMS server.

##### Router Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| id | String | The unique ID associated with the specific router. |
| Address | String | The network address of the router represented as a hexadecimal address. |
| Location | String | The name of the location that the router is placed at. Ie: Left tunnel, committee room, etc… |
| x | int | The x position of the router on the map. |
| Y | int | The y position of the router on the map. |
| isBlocked | Boolean | Whether or not the router is in a blocked zone of the site. |

##### Router Operations

N/A

##### Router Design Specification/Constraints

N/A

##### Router States and Transitions

N/A

#### RouterInfoForm

This form appears over top of a router icon on the map. Its purpose is to display all router information including a list of all connected members when a user clicks it on the map or selects a member connected to it.

##### RouterInfoForm Attributes

N/A

##### RouterInfoForm Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| setPosition(x : int, y : int) | Sets the position on the screen where the router should appear. This should be over top of a router icon. |
| setRouter(router : Router) | Sets the router whose information will be displayed on the form. |

##### RouterInfoForm Design Specification/Constraints

N/A

##### RouterInfoForm States and Transitions

N/A

#### Sensor

<EachClass.Documentation>

##### Sensor Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Id | String |  |

##### Sensor Operations

N/A

##### Sensor Design Specification/Constraints

N/A

##### Sensor States and Transitions

N/A

#### Shift

Model representing a miner’s work shift. A miner can have many work shift that are described by a start and end time while a work shift belongs to one particular miner.

##### Shift Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| start | DateTime | The start time of the shift. |
| End | DateTime | The end time of the shift |

##### Shift Operations

N/A

##### Shift Design Specification/Constraints

N/A

##### Shift States and Transitions

N/A

#### Site

Simple model used to represent the mining site. This acts as a parent object of all routers in the system and is described by the name of the mining site.

##### Site Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| name | String | The name of the mining site. |

##### Site Operations

N/A

##### Site Design Specification/Constraints

N/A

##### Site States and Transitions

N/A

#### Tag

The ZigBee tag associated with an end device. This is used in coordinator messaging to allow the system to pair members with end devices and is the main identification of miners in the ZigBee network.

##### Tag Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| id | String | The ZigBee network id of the end device. |

##### Tag Operations

N/A

##### Tag Design Specification/Constraints

N/A

##### Tag States and Transitions

N/A

#### TrackingPanel

The main view associated with the tracking module. Contains the UI for selecting miners, viewing the map, and viewing miner positions and paths.

##### TrackingPanel Attributes

N/A

##### TrackingPanel Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| drawPath(points : Point[]) | Draws a polygon line with vertices at each point representing routers on the map. |
| selectMember() | Delegate for selecting a miner from the list of miners. This invokes other controller actions including viewing the miner position on the map. |

##### TrackingPanel Design Specification/Constraints

N/A

##### TrackingPanel States and Transitions

N/A

### Messaging Module Class Diagram



Figure 5: Messaging module class diagram

#### MessagingController

The main controller responsible for executing messaging module tasks.

##### MessagingController Attributes

N/A

##### MessagingController Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| getMessage() | Provides a simple input box for the user to enter message contents and the type of message. |
| broadcastMessage(content : String) | Creates a broadcast message packet to be forwarded to the coordinator. |
| createAlert(content : String) | Creates an alert message to be forwarded to the coordinator. |

##### MessagingController Design Specification/Constraints

N/A

##### MessagingController States and Transitions

N/A

#### Message

Model representing messages that are sent over the network through the coordinator. The messages are received at end devices and displayed to the users of those devices.

##### Message Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| content | String | The content of the message that end device users will see. |
| Time | DateTime | The time of creating the message. |
| messageType | Enumerator | Specifies the type of message: a broadcast message to the entire mining site, or an alert message. |

##### Message Operations

N/A

##### Message Design Specification/Constraints

N/A

##### Message States and Transitions

N/A

#### User

Model representing the user accounts that can log into the system.

#### User Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| username | String | Unique username for logging in. |
| password | String | Secret password for logging in. |
| fName | String | User’s first name. |
| lName | String | User’s last name. |

##### User Operations

N/A

##### User Design Specification/Constraints

N/A

##### User States and Transitions

N/A

### Attendance Module Class Diagram



Figure 6: Attendance module class diagram

#### AttendanceController

The main controller for the attendance module. This generates the various attendance range reports for selected members.

##### AttendanceController Attributes

N/A

##### AttendanceController Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| generateDaily(start : Date, end : Date) | Generates a report of all work shift by daily intervals in the provided day range. Creates an ExportReportView displaying a PDF of the report. |
| generateMonthly(start : Date, end : Date) | Generates a report of monthly intervals in the provided month range. Creates an ExportReportView displaying a PDF of the report. |
| generateYearly(start : Date, end : Date) | Generates a report of yearly intervals in the provided year range. Creates an ExportReportView displaying a PDF of the report. |

##### AttendanceController Design Specification/Constraints

N/A

##### AttendanceController States and Transitions

N/A

#### AttendanceView

The main interface used by the user to select one of the three attendance report ranges for a member in the mine.

##### AttendanceView Attributes

N/A

##### AttendanceView Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| OnRadioButton() | Delegate for selecting a report type. Depending on the report range selected, the form will prepare the appropriate inputs. |
| setDailyInput() | Creates appropriate input for a daily report. This includes two days being the start and end days of the report for a user to specify. |
| setMonthlyInput() | Creates appropriate input for a monthly report. This includes two month being the start and end month of the report for a user to specify. |
| setYearlyInput() | Creates appropriate input for a yearly report. This includes two years being the start and end years of the report for a user to specify. |

##### AttendanceView Design Specification/Constraints

N/A

##### AttendanceView States and Transitions

N/A

#### ExportReportView

A simple window used to display generated PDF reports.

##### ExportReportView Attributes

N/A

##### ExportReportView Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| showReport(reportPath : String) | Opens the PDF report located at the path. |

##### ExportReportView Design Specification/Constraints

N/A

##### ExportReportView States and Transitions

N/A

### Reports Module Class Diagram



Figure 7: Reports module class diagram

#### ReportsController

The main controller for the reports module. This performs the functions necessary to build the various available reports.

##### ReportsController Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| reportType | Enumerator | Specifies the type of report that will be generated when the user initiates to generate a report. |

##### ReportsController Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| generateMessageDetailsReport() | Generates a report containing all messages that have been created by the user using the messaging module organized by time sent. |
| generateMinerReport() | Generates a report containing all information of all miners organized by memberNo. |
| GenerateMinerBlockReport() | Generate a report of miners that have connected to blocked routers. |
| generateMinerPositionReport() | Generates a report of all of a particular miners’ positions travelled to in the mine. |
| generateRouterReport() | Generates a report containing all router information on site in a table. |
| generateTagReport() | Generates a report |
| generateReport() | Uses the reportType to pick which operation to execute to generate the correct report type. |
| setReportType(reportType : Enumerator) | Called externally to specify the type of report that will be generated when generateReport() is called. |

##### ReportsController Design Specification/Constraints

N/A

##### ReportsController States and Transitions

N/A

#### ReportsView

<EachClass.Documentation>

##### ReportsView Attributes

N/A

##### ReportsView Operations

|  |  |
| --- | --- |
| **Name** | **Description** |
| OnListItemSelected() | Delegate for selecting a report type from a drop down list. |
| setMessageDetailsInput() | Creates the required input for a message details report. This clears the previous input layout as no input is required. |
| setMinerInput() | Creates the required input for a miner details report. This clears the previous input layout as no input is required. |
| setMinerBlockInput() | Creates the required input for a miner block report. This includes a list of all miners in the system to select from. |
| setMinerPositionInput() | Creates the required input for a miner position report. This includes a list of all miners in the system to select from. |
| setRouterInput() | Creates the required input for a router details report. This clears the previous input layout as no input is required. |
| setTagInput() | Creates the required input for a tag details report. This clears the previous input layout as no input is required. |

##### ReportsView Design Specification/Constraints

N/A

##### ReportsView States and Transitions

N/A

### Sequence Diagrams

#### AssignShift Diagram:



Figure 8: AssignShift sequence diagram

#### AssignTag Diagram:



Figure 9: AssignTag sequence diagram

#### CreateMember Diagram:



Figure 10: CreateMember sequence diagram

#### CreateSensor Diagram:



Figure 11: CreateSensor sequence diagram

#### CreateTag Diagram:



Figure 12: CreateTag sequence diagram

#### CreateUser Diagram:



Figure 13: CreateUser sequence diagram

#### LoadMap Diagram:



Figure 14: LoadMap sequence diagram

#### Login Diagram:



Figure 15: Login sequence diagram

#### ReceiveMessage Diagram:



Figure 16: ReceiveMessage sequence diagram

#### ViewMemberPosition Diagram:



Figure 17: ViewMemberPosition sequence diagram

#### ViewPathChart Diagram:



Figure 18: ViewPathChart sequence diagram

#### BroadcastText Diagram:



Figure 19: BroadcastText sequence diagram

#### CreateAlert Diagram:



Figure 20: CreateAlert sequence diagram

#### CreateAttendanceReport Diagram:



Figure 21: CreateAttendanceReport sequence diagram

#### CreateReport Diagram:



Figure 22: CreateReport sequence diagram

### Relational Database Schema

#### System Relational Diagram:



Figure 23: Full system relational schema showing how data is saved to the database

## Design Rationale

### Controllers & Architecture

The choice of creating a controller for each module is meant to separate the modules to the user’s point of view. The TMS will not require that all modules have to work together always, only that they store or retrieve data from the same database. In this sense the Tracking, Messaging, and Master modules store data; and the Reports and Attendance modules retrieve data. It is designed like this to be able to easily update each module knowing that retrieving modules will have no impact on other modules.

### Data Storage

The TMS will run on a dedicated machine so all data will be stored. This is also necessary for the creation of reports so that the user can see a long history of underground activity.

# Walkthrough Report

No feedback was given during the walkthrough. However, since then it has been decided to replace the functionality of specifying the conditions for an alert message with simply sending a message labelled as an alert. The broadcast text use case was also changed from a send text and now allows the TMS user to send a message that will be received by all mining personnel.

# Software Testing Plan

## Introduction

The objective of the tests outlined in this plan is to provide development and QA team members with a framework that can be used to verify all functional and non-functional requirements specified in the SRS document.

## Relationship to other documents

The tests are designed to verify the correct functionality of all class operations described in the SDS and to validate that all functional and non-functional requirements specified in the SRS are present and verified.

## System overview

All modules will depend on the Master module to be tested because it provides the necessary database data required for testing.

Besides entering data using the master module function, all modules: tracking, reports, attendance, and messaging will be tested independently.

## Features to be tested/not to be tested

### To be tested

#### Add / Edit Network Nodes

Routers, sensors, end devices tags should be able to be added to the system

#### Display mine plan and structure

The image containing the mining site plan should be displayed along with all routers set in the site upon loading the map.

#### Display Path Chart for Miners

A miner’s path should be drawn on the mining site plan after selecting them.

#### Display Miner Position

A miner’s position should be highlighted on the map after selecting them.

#### Generate Attendance Reports

Daily, monthly, and yearly reports for miner attendance and general operation time reports for vehicles reports should be able to be generated.

#### Generate Network Node Reports

Miners, miner’s position, miners blocked, router, tag, and sent / received message reports should be able to be generated..

### Not to be tested

#### Firmware communication functions among nodes

Proper communication functions between routers and coordinator will not be tested as part of this project.

#### Messaging module USB communication

Due to unforeseen circumstances related to receiving necessary hardware for the project, this test plan will not include results for the messaging module. This includes sending broadcast and alert messages across a USB connection.

## Pass/Fail criteria

### Display mine plan and structure

Pass if the image containing the mining site plan is displayed and all routers in the system are visible and in the correct position.

### Display Path Chart for Miners

Pass if a path is drawn upon selecting a miner and the path contains all previously connected to routers for that miner.

### Display Miner Position

Pass if the miner’s position is highlighted on the map by a router info window.

### Generate Attendance Reports

Pass if:

* Daily report shows all work shift per day for the day range
* Monthly report shows all day hours per day of the month
* Yearly reports show all month hours for a year
* General operation time reports for vehicles reports show all day hours for the day range.

### Generate Network Node Reports

Pass if miners, miner’s position, miners blocked, router, tag, and sent / received message reports are generated and contain all pertinent information.

### Add / Edit Network Nodes

Pass if routers, sensors, end devices tags are added to the system and saved to the database.

## Approach

The TMS consists of multiple modules and so the test plan is designed to allow most modules to be tested concurrently and to later be integrated only with the master module.

In the cases of inputting data in specific formats in the master module, the formats will not be tested because the TMS uses Masked Text Fields that do not allow input that deviates from the expected format.

## Suspension and resumption

All test items must be repeated if the Master module is updated because all modules depend on successful functionality of the Master module.

The tracking, messaging, reports, and attendance module test cases can all be run independently as long as the master module is verified.

## Testing materials (hardware/software requirements)

* Large area to set routers: underground tunnels or hallways
* Full Function Devices (FFDs)
* Reduced Function Devices (RFDs)
* Windows Computer

## Test cases

## Testing schedule

Testing will take place across 16 work days and is planned to start February 6, 2015 as shown in the [Gantt Chart](#_Gantt_Chart).

# Test Case Specifications

## Master Module

### Test items

#### Adding Members

A new member should be added to the database if all input is correct.

#### Adding Routers

A new router should be added to the database if all input is correct.

#### Adding Sensors

A new sensor ID should be added to the database if all input is correct.

#### Adding Tags

A new tag ID should be added to the database if all input is correct.

### Input specifications

#### Adding Members

##### Valid member data

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MemberID** | **Is Vehicle** | **Pin No** | **Tag** | **First** | **Middle** | **Last** | **Address** | **Province** | **City** | **Phone** | **Mobile** |
| M0001 | False | 111111 | E0001 | Mark | “” | Miner | 12 Addr Ave | ON | London | (555)555-555 | “” |

##### Invalid member data

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MemberID** | **Is Vehicle** | **Pin No** | **Tag** | **First** | **Middle** | **Last** | **Address** | **Province** | **City** | **Phone** | **Mobile** |
| M0001 | False | “” | E0001 | Mark | “” | Miner | 12 Addr Ave | ON | London | (555)555-555 | “” |

Invalid because the Pin No is mandatory but null.

#### Adding Routers

##### Valid router data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Router ID** | **Address** | **Location** | **Position X-Y** | **Is Blocked?** |
| R0001 | Address 1 | 1 | 20-100 | false |

##### Invalid router data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Router ID** | **Address** | **Location** | **Position X-Y** | **Is Blocked?** |
| R0001 |  | 1 | 20-100 | false |

Assuming that a router with ID “R0001” already exists.

#### Adding Sensors

##### Valid sensor ID

|  |
| --- |
| **Sensor ID** |
| S0001 |

##### Duplicate sensor ID

|  |
| --- |
| **Sensor ID** |
| S0001 |

Assuming that a sensor with ID “S0001” already exists.

#### Adding Tags

##### Valid tag ID

|  |
| --- |
| **Sensor ID** |
| E0001 |

##### Duplicate tag ID

|  |
| --- |
| **Sensor ID** |
| E0001 |

Assuming that a tag with ID “E0001” already exists.

### Output specifications

#### Adding Members

##### Valid member data

The input fields are cleared and the new member is added to the list and database.

##### Invalid member data

Message box appears notifying user of invalid input.

#### Adding Routers

##### Valid router data

The input fields are cleared and the new router is added to the list and database.

##### Invalid router data

Message box appears notifying user of invalid input.

#### Adding Sensors

##### Valid sensor ID

The input fields are cleared and the new sensor is added to the list and database.

##### Duplicate sensor ID

The submit button is disabled, disallowing users from submitting an invalid ID.

#### Adding Tags

##### Valid tag ID

The input fields are cleared and the new tag is added to the list and database.

##### Duplicate tag ID

The submit button is disabled, disallowing users from submitting an invalid ID.

### Environmental needs

#### Computer running Windows with the Tracking and Monitoring Software (TMS) installed

A fresh install of the TMS is required with an empty database.

### Special procedural requirements

### Intercase dependencies

#### Adding tag must succeed before adding member can succeed

There must be a valid tag in the system for a member to be added because a member requires an assigned tag.

## Tracking Module

### Test items

#### Adding Members

A new member should be added to the database if all input is correct.

#### Adding Routers

A new router should be added to the database if all input is correct.

#### Adding Sensors

A new sensor ID should be added to the database if all input is correct.

#### Adding Tags

A new tag ID should be added to the database if all input is correct.

### Input specifications

#### Display Path Chart for Miners

##### Miner with path

Select a miner from the tracking display that has connected to at least two different routers.

##### Miner without path

Select a miner from the tracking display that has not connected to any routers.

#### Display Miner Position

##### Select active miner from tracking display

Select an active miner from the list of active miners on the tracking display.

### Output specifications

#### Display Path Chart for Miners

##### Miner with path

A heat trail will appear on the map leading to the current position of the miner.

##### Miner without path

Nothing will appear on the map.

#### Display Miner Position

##### Select active miner from tracking display

The router info form will appear with its corner pointing at the router that the selected miner is connected to.

### Environmental needs

#### Computer running Windows with the Tracking and Monitoring Software (TMS) installed

An install of the TMS is required with some miners and routers loaded into the system.

### Special procedural requirements

#### Test path message receiving

There must be some way to receive messages from some source that will allow the system to emulate receiving a valid path message specified in the software interfaces section.

### Intercase dependencies

#### Master module test cases must succeed before data can be tracked

There must be valid members, tags, and routers in the system for the components to be tracked.

## Messaging Module

### Test items

#### Sending an alert

A specific active miner on site should be notified via their end device.

#### Sending a broadcast

Every active miner on site should be notified via their end device.

### Input specifications

#### Sending an alert

The user chooses specific active from the messaging form.

#### Sending a broadcast

The user selects to broadcast all miners from the messaging form.

### Output specifications

#### Sending an alert

A specific active miner on site should be notified via their end device.

#### Sending a broadcast

Every active miner on site should be notified via their end device.

### Environmental needs

#### Computer running Windows with the Tracking and Monitoring Software (TMS) installed

An install of the TMS is required with some miners and routers loaded into the system.

#### Active programmed coordinator connected to the TMS

There should be a programmed FFD connected directly to the TMS via USB that is capable of receiving a byte stream consisting of a byte and an end tag ID.

### Special procedural requirements

### Intercase dependencies

#### Master module test cases must succeed before data can be tracked

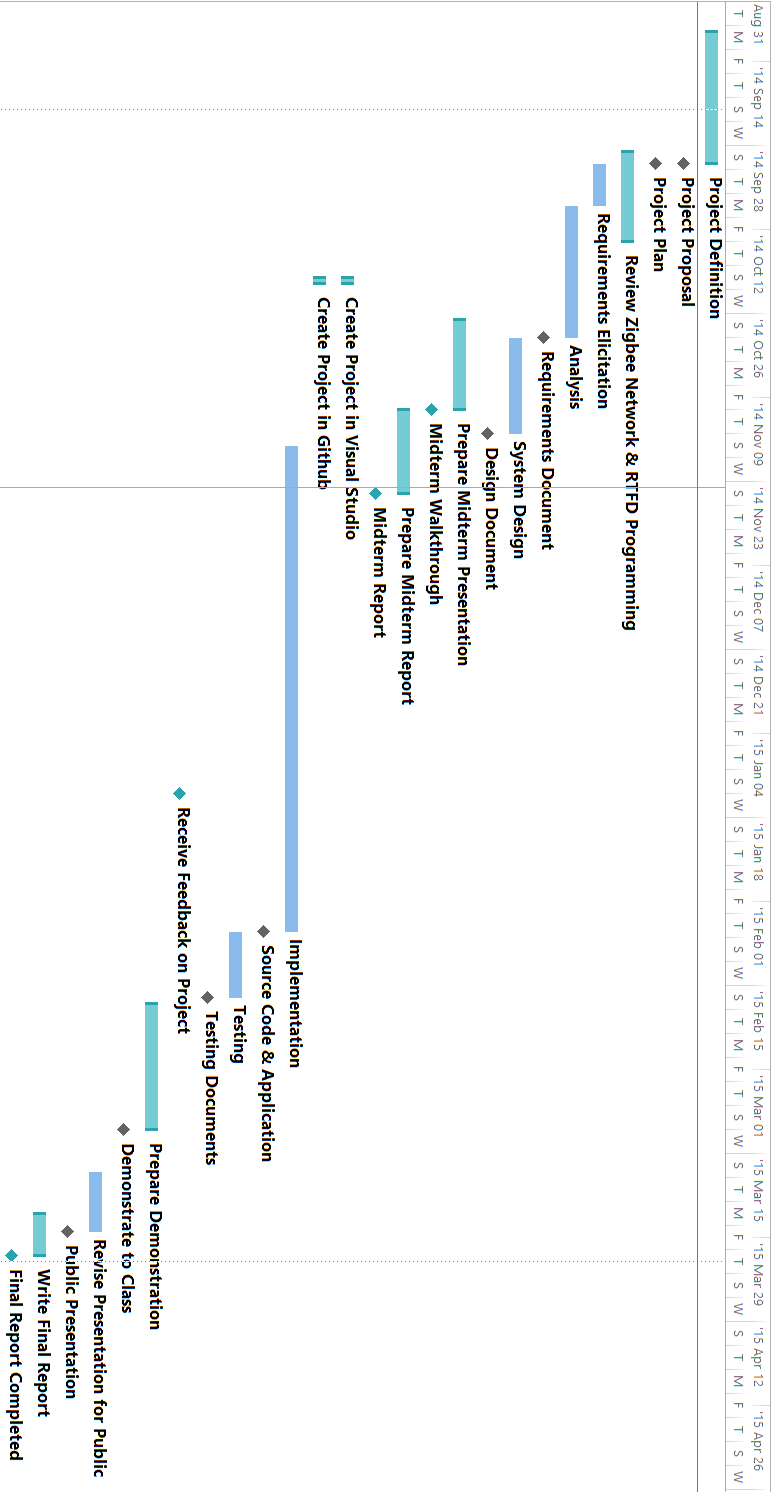
There must be valid members, tags, and routers in the system for the messages to be constructed.

## Reports Module

## Attendance Module

# Conclusion / Recommendations

# Gantt chart



# References

Bandyopadhyay, L. K., S. K. Chaulya, and P. K. Mishra. "Chapters 8 & 11." *Wireless Communication in Underground Mines: RFID-based Sensor Networking*. New York: Springer, 2010. N. pag. Print.

Microsoft. "C# Coding Conventions (C# Programming Guide)." *MSDN*. Microsoft, 2014. Web. 20 Nov. 2014.

Microsoft. "How to Create the Best User Experience for Your Application." *MSDN*. Microsoft, 2014. Web. 20 Nov. 2014.

Mircom Group. “OpenGN System Overview.” Mircom. Mircom Group, 2014. Web. 20 Nov. 2014

<http://www.mircom.com/opengn>

# Appendix A: Use Case Specification Documents

## 

### Brief Description

This use case allows the user to specify a work shift for an employee. The work shift is a day and a start / end time that is assigned to a miner.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has successfully completed the LogIn use case.

#### Selected miner

The user has selected a miner to assign a shift to.

### Flow of Events

#### User selects to assign shift

* In the miner window, the user initiates the use case by clicking to assign a shift to the miner.
* The shift data window is opened.

#### User enters shift data

* The user enters a date
* The user enters a start time and an end time

#### Confirm shift

* The user confirms the shift specifications and assigns it to the miner.

### Exit Conditions

#### Shift is created and assigned

A work shift is added to the database and assign the selected miner.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

A tag refers to the ZigBee tag given to end devices in a ZigBee network. The purpose of assigning a tag is to pair an end device with a miner or a vehicle that is working in the mine.

### Participating actor

#### User

The TMS user.

### Entry conditions

#### User logged in

The user has completed the LogIn use case and thus supplied the TMS with valid login credentials.

#### Tag created

The user has completed the CreateTag use case at least once so that there is at least one tag to assign to a member.

### Flow of Events

#### User selects assign tag

* The user has initiated the use case by selecting to assign a tag from the menu.

#### User selects tag

* The user selects the tag that they would like to assign.

#### User selects member

* The user selects the member that they would like to assign the tag to.

#### Confirm selection

* The user confirms the tag and user pair.
* The system updates the database to update the selected member’s tag id.

### Exit Conditions

#### Member tag is updated

The selected member is assigned a tag and data is updated appropriately in the system database.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

A person using the TMS can broadcast a text message to members with an end device. The message is encapsulated and sent through the ZigBee network to the appropriate member.

### Participating actors

#### User

The TMS system user.

#### Coordinator

The central node of the ZigBee network. The coordinator contains all information of routers in the mine and is able to return their location and end device connections.

### Entry conditions

#### User logged in

The user has completed the LogIn use case.

#### The TMS is online

The network is active and the TMS server is connected to the coordinator.

### Flow of Events

#### Start up

* The user initiates the use case by selecting the Send Text option from the messaging module.

#### Enter message content

* The user is presented with an input dialog for a text message and whether or not the message is a broadcast or an alert

#### Confirm & Send

* Message content is confirmed
* The message is encapsulated and sent to the coordinator

### Exit Conditions

#### Coordinator receives message

The coordinator has received the message packet and proceeds to send it to the appropriate end devices.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

Alarms are received by miners on their end devices. This use case allows the user to define alarm contents and conditions for which the coordinator will send an alarm to miners.

### Participating actor

#### User

The TMS system user.

#### Coordinator

The central node of the ZigBee network. The coordinator contains all information of routers in the mine and is able to return their location and end device connections.

### Entry conditions

#### User authenticated

The user has logged in to the system.

#### Network implemented

The coordinator and routers are placed throughout the network.

### Flow of Events

#### Start up

* The user initiates the use case by pressing a Create Alert option.

#### Input Dialog Window

* An input dialog window appears
* The user enters the contents of the alarm message
* The alarm contents are saved to the TMS database for use in message details reports.

#### Coordinator receives alarm

* The coordinator receives the packet containing the alarm data and generates the appropriate functions to set the alarm.

### Exit Conditions

#### The Coordinator received alarm

The coordinator is now prepared to send alarm messages to active end devices.

#### The Alarm is saved

The alarm is saved to the database.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

This use case’s purpose is to generate an attendance report for a miner. The report contains miner information and worked shift times and can be either daily, monthly, or yearly.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User is logged in

The user has completed the LogIn use case.

#### Members have been created

At least miner has been added to the system to be tracked.

### Flow of Events

#### Initiate report window

* The user selects the type of report from a Reports menu drop down.

#### Select miner

* The user selects a miner to be reported from a drop down list

#### User selects date interval

* If the user selects daily
  + See DailyReport use case.
* If the user selects monthly
  + See MonthlyReport use case.
* If the user selects yearly
  + See YearlyReport use case.

### Exit Conditions

#### PDF Report generated

A yearly, monthly, or daily report is generated and presented to the user in PDF form.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

The purpose of this use case is to create a profile of a member (a miner or a mining vehicle) that represents a member assigned to an end device in the mine.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User is logged in

The user has completed the LogIn use case.

### Flow of Events

#### User selects to create member

* In the main menu window, the user initiates the use case by clicking to create a member.
* The member creation window is opened

#### User enters sensor data

* The user enters the member’s name, address, phone number, etc… in text fields and presses the confirm button

### Exit Conditions

#### Member is created

Member data is created and saved to the database.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

This use case allow a user to create a report containing the content and date of every message that has been created by the user.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has successfully completed the LogIn use case.

### Flow of Events

#### User selects to create message details report

* The user selects message details from the list of report in the main window’s menu

#### Report is shown to user

* A report of all broadcast messages created by the user is shown in another window in PDF

### Exit Conditions

#### PDF Report generated

A PDF report is generated and can be saved to the computer.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

This use case allow a user to create a report containing all personal information of miners in the system organized by member number.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has successfully completed the LogIn use case.

### Flow of Events

#### User selects to create miner report

* The user selects miner from the list of report in the main window’s menu

#### Report is shown to user

* A report of all miners in the system is shown in another window in PDF

### Exit Conditions

#### PDF Report generated

A PDF report is generated and can be saved to the computer.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

This use case allow a user to create a report containing the router information and date of every instance that a particular miner has entered a blocked zone.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has successfully completed the LogIn use case.

### Flow of Events

#### User selects to create miner block report

* The user selects miner block from the list of report in the main window’s menu
* An input window is shown that lets the user specify the miner on which to report

#### User selects miner

* The user selects the miner’s member number from a list

#### Report is shown to user

* A report of all instances of the selected miner entering a blocked zone is shown in another window in PDF

### Exit Conditions

#### PDF Report generated

A PDF report is generated and can be saved to the computer.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

This use case allow a user to create a report containing the router information and date of every position that a particular miner has entered.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has successfully completed the LogIn use case.

### Flow of Events

#### User selects to create miner position report

* The user selects miner position from the list of report in the main window’s menu
* An input window is shown that lets the user specify the miner on which to report

#### User selects miner

* The user selects the miner’s member number from a list

#### Report is shown to user

* A report of all routers that the selected miner has connected to is shown in another window in PDF

### Exit Conditions

#### PDF Report generated

A PDF report is generated and can be saved to the computer.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

This use case allow a user to create a report containing all network of routers in the system organized by router ID.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has successfully completed the LogIn use case.

### Flow of Events

#### User selects to create router report

* The user selects router from the list of report in the main window’s menu

#### Report is shown to user

* A report of all routers on site is shown in another window in PDF

### Exit Conditions

#### PDF Report generated

A PDF report is generated and can be saved to the computer.

### Quality requirements

#### <Quality requirement one>

## Use Case Specification: CreateSensor

### Brief Description

This use case allows a user to create a sensor in the TMS system that corresponds to a sensor in the ZigBee underground mine site.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has successfully completed the LogIn use case.

### Flow of Events

#### User selects to create sensor

* In the main menu window, the user initiates the use case by clicking to create a sensor.
* The sensor creation dialog is opened

#### User enters sensor data

* The user enters the sensor name in a textfield and presses the confirm button

### Exit Conditions

#### Sensor added to the database

* The sensor is added to the TMS database

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

The user enters the ZigBee tag of a new end device. The new tag can be assigned to a miner and represents an end device in the ZigBee network.

### Participating actor

#### User

An authenticated user interacting with the monitoring software.

### Entry conditions

#### User is logged in

The user has supplied valid login credentials in the LogIn use-case.

### Flow of Events

#### Tag list window opens

* The user opens a window that shows all the tags in the database.

#### Fills in tag information

* The tag name is entered and the user submits the tag to be created.
* If the tag name already exists, notify the user of this and ask for a different name.
* If the tag name does not exist, insert the tag to the database.

### Exit Conditions

#### Tag added to database

A new entry for the created tag is added to the system database.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

This use case allow a user to create a report containing all tags of end devices in the system organized by tag ID.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has successfully completed the LogIn use case.

### Flow of Events

#### User selects to create router report

* The user selects tag from the list of reports in the main window’s menu

#### Report is shown to user

* A report of all tags and their assigned miner on site is shown in another window in PDF

### Exit Conditions

#### PDF Report generated

A PDF report is generated and can be saved to the computer.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

The purpose of this use case is to create a user account for an employee that will be using the TMS but without administrative privileges.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User is logged in

The user has completed the LogIn use case.

#### User is an admin

The user has administrative privileges.

### Flow of Events

#### User selects to create user

* In the main menu window, the user initiates the use case by clicking to create a user.
* The user creation window is opened

#### User enters sensor data

* The user enters the user’s name, username, and a password in text fields and presses the confirm button

### Exit Conditions

#### User is created

User data is created and saved to the database.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

The purpose of this use case is to generate a portable report containing information about a vehicle’s operation. The information on the vehicle’s operation is gathered throughout tracking

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user has completed the LogIn use case.

#### Vehicle has been tracked

At least one vehicle has been tracked

### Flow of Events

#### User clicks report

* The user clicks to generate a vehicle operation report.
* A list of all vehicles is presented.

#### User selects vehicle

* The user picks the vehicle from a drop down list.
* A report on the vehicle’s operation data is presented to the user.

### Exit Conditions

#### PDF Report Generated

A PDF report is generated that displays the specified vehicle’s information.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

A special case of CreateAttendanceReport where the user selects to create a daily report. A daily report contains miner shift information for each day in some specified day range.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### Create Attendance Report use case initiated

The user has begun the Create Attendance Report use case and selected a miner to report.

### Flow of Events

#### Enter date range

* The user enters a start date and an end date.

#### Create report

* A report is constructed using the selected miners’ work shifts.

### Exit Conditions

#### PDF report generated

A PDF report is generated containing daily work shift information for the selected miner.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

The purpose of this use case is to load an image file to the application and provide the user with a mine plan of the mining site.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### User logged in

The user completed the LogIn use case.

### Flow of Events

#### User initiates use case

* The user enters the master module drop down and selects to load a map.

#### User selects file

* The user selects an 800x700 pixel jpg, jpeg, or bmp image file from their files using a file explorer dialog
* The image is loaded and displayed onto the screen

### Exit Conditions

#### Map is displayed on screen

The user is given a visual of the mine plan on the screen alongside the main controls.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

The user enters his / her username and password to enter the system. No other use cases can proceed without logging in.

### Participating actor

#### User

A TMS user with valid login credentials.

### Entry conditions

#### Account created

There is at least one account on the system to log in with.

### Flow of Events

#### Application Start Up

* The user starts up the application.
* Login screen in presented to the user.

#### User enters credentials

* User enters username
* User enters password

#### Authentication

* Username and password are verified
  + If they match open the main screen.
  + If not, notify user that credentials are invalid.

### Exit Conditions

#### User logged in

The user profile is loaded and the main screen is presented to the user.

#### Invalid login

The user is notified that their username and password does not match any combinations in the system.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

A special case of CreateAttendanceReport where the user selects to create a monthly report. A monthly report contains miner shift information for each day in some specified month.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### Create Attendance Report use case initiated

The user has begun the Create Attendance Report use case and selected a miner to report.

### Flow of Events

#### Enter month

* The user enters a month to generate a report through.

#### Create report

* A report is constructed using the selected miners’ work shifts.

### Exit Conditions

#### PDF report generated

A PDF report is generated containing monthly work shift information for the selected miner.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

The TMS server is constantly awaiting communication from the network coordinator.

### Participating actor

#### Coordinator

The central node of the ZigBee network. The coordinator contains all information of routers in the mine and is able to return their location and end device connections.

### Entry conditions

#### Coordinator is connected

There is an active connection between the TMS server and the coordinator.

### Flow of Events

#### Coordinator sends message

* The coordinator sends a formatted message to the TMS including the end device and the router IDs.

#### TMS server receives message

* The message is received at the server’s socket.

#### Parse message

* The message type is parsed and its content is stored appropriately in the TMS database.

### Exit Conditions

#### Database updated

The message protocol is analyzed and its appropriate information is saved to the TMS database.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

Miners traverse the underground mine holding end devices that connect to strategically placed routers throughout the mining tunnels. A router has a location property which is also the member’s location when the member passes by and connects to that router.

### Participating actor

#### User

The TMS user that is tracking miners or mining equipment.

### Entry conditions

#### Physical network set up

The coordinator, routers, and end devices must be properly placed throughout the tracking area.

### Flow of Events

#### User selects a member

* The user selects a member from the list of active members in the mine or my clicking the map.
* The member is only selectable if they are associated with a tag.

#### Display member position

* The member is cross referenced with the router it is at and the router’s position in the mine is displayed alongside the member information.
* The member’s information is displayed on the map.

### Exit Conditions

#### Member position is displayed

* A tabular view displays the member’s location.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

The use case involves the user viewing a graphical representation of a miner’s current path in the map. As a miner has travelled through the mine’s tunnels, the server has received messages from the network coordinator that specified the end device tag connected to each router. The history of visited routers is drawn as a path on the screen.

### Participating actor

#### User

The TMS system user.

### Entry conditions

#### The user has loaded a map

The user has completed the LoadMap use case.

#### The user has created the tag

There is at least one end device tag created that can be monitored.

#### The miner / vehicle has connected to a router

The miner or vehicle to be tracked has connect to at least one router or else there is no path to display.

### Flow of Events

#### Select a tag

* The user selects a tag from the tag list. This is the tag associated with the miner or vehicle that is to be displayed.

#### Draw path

* Any paths on the map are cleared.
* Path data is loaded from the received messages and drawn on the screen.

### Exit Conditions

#### Path shown on screen

The map area of the screen is populated with a set of lines connecting each router that represents the path that the miner is travelling.

### Quality requirements

#### <Quality requirement one>

## 

### Brief Description

A special case of CreateAttendanceReport where the user selects to create a yearly report. A yearly report contains miner shift information for each day in some year.

### Participating actor

#### User

A person using the TMS.

### Entry conditions

#### Create Attendance Report use case initiated

The user has begun the Create Attendance Report use case and selected a miner to report.

### Flow of Events

#### Enter year

* The user enters a year to generate a report for.

#### Create report

* A report is constructed using the selected miners’ work shifts.

### Exit Conditions

#### PDF report generated

A PDF report is generated containing yearly work shift information for the selected miner.

### Quality requirements

#### <Quality requirement one>

# Appendix B: Test Case Results

## Master Module

## Tracking Module

## Reports Module

## Attendance Module

# Appendix C: Source Code

The source code of the TMS system is hosted on GitHub at <https://github.com/pkurowsk/TMS>.

# User Manual

# Vitae

**Education**

Candidate for B.E.Sc.-Software, 2015 University of Western Ontario

Skills

* Programming experience with C++, Java, C#, Python, JavaScript, and bash
* Intermediate experience using software design patterns and writing maintainable and testable code
* Server development experience using node.js to create RESTful web services, database applications, and web server games
* Game development experience with Unity, libgdx, XNA Game Studio, and HTML5 / JavaScript
* Knowledge of Google Play, App Store and Windows Marketplace TCR
* Experience developing test cases and assuring code coverage using JUnit & Sahi
* Able to define specifications, use case diagrams, design diagrams, and relational schemas

**Work Experience**

### May – August 2014 Cloud Computing Research Position at the University of Western Ontario

* Implemented a novel high availability-aware scheduling algorithm as a Java GUI that works with Linux-based OpenStack installations to optimize the placement of application components
* Developed two OpenStack filters to schedule virtual machines based on the availability & delay of servers in a cloud computing system using data from OpenStack’s MySQL database
* Developed data visualization UI components for network resources and application availability
* Wrote bash scripts to create an interface with OpenStack and Java, to automate database entry for testing large data sets, and to automate installation of database modifications
* Modified OpenStack source code to create a testing environment for virtual machine scheduling

### May – August 2013 Game Developer (Co-op) at Slightly Social

* Worked in an agile programming environment to create and ship Road Trip Friends, an online iOS multiplayer battle racing game, using Unity and the Nextpeer multiplayer SDK
* Developed networked multiplayer protocols and software components to allow players to race against and attack each other in five person online matches
* Created UIs for online scoreboard data, and in-game HUD for live rankings and a player feed
* Managed all Windows Phone and BlackBerry production, post-production, and distribution
* Used C++ and C# to develop Windows Phone 8 and BlackBerry plugins to perform native functionality from the Unity engine such as in-app purchasing and advertisements

**Engineering Projects**

### September 2014 – Present Video Game Projects

* Developed a 3D top-down shooter where players fly a space ship around a planet’s surface to defend the planet from enemies, includes local high scores, multiple stages, and animations
* The project code can be found here: <https://github.com/pkurowsk/surface-rupture>
* Used the Unity engine’s Editor API to develop a project-wide localization tool and a dialog tree editor tool that saves dialog data to resources and allows in-game language switching
* The project code can be found here: <https://github.com/pkurowsk/localization-dialog-editor-unity>
* Used node.js and HTML5 to develop an online versus game of Slapsies where players are matched up and play as either attacker or defender (<https://github.com/pkurowsk/slapsies-online>)