4-2013

Cyber-Adaptive Physical Systems for Automated Construction Progress Monitoring and Asset Tracking

Syed Hammad Rasheed

CYBER-ADAPTIVE PHYSICAL SYSTEMS FOR AUTOMATED CONSTRUCTION PROGRESS MONITORING AND ASSET TRACKING

by

Syed Hammad Rasheed

A thesis submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Master of Science in Engineering (Civil)
Department of Civil and Construction Engineering
Western Michigan University
April 2013

Thesis Committee:

Abiola Akanmu, Ph.D., Chair
Osama Abudayyeh, Ph.D.
Upul Attanayake, Ph.D.

CYBER-ADAPTIVE PHYSICAL SYSTEMS FOR AUTOMATED CONSTRUCTION PROGRESS MONITORING AND ASSET TRACKING

Syed Hammad Rasheed, M.S.E.

Western Michigan University, 2013

Despite the benefits of building information models as a means of communication, collaboration and integration in construction projects, their use is often limited to the design and tendering/bidding stage. Potential of building information models lies in its implementation during construction, operations and maintenance phases of a facility's lifecycle. Tight Integration of building information model and physical facility by means of computational resources is required to achieve this potential such that changes in one environment are automatically reflected in the other. This is termed a cyber-adaptive physical systems approach. This research investigates the applicability of a cyber-adaptive physical systems approach in the construction industry. Suitable technologies and limitation of existing approaches was determined by brief literature review and three prototypes were developed for automated component status tracking, asset tracking in constructed facilities and automated progress tracking using earned value management. The benefits of these systems include access to real-time progress information for quick decision making and the potential for enhancing real-time collaboration between the job site and main office. These prototypes were also tested for reliability in different case-studies.

ACKNOWLEDGMENTS

First, I would like to thank God for giving me the strength and power to complete this research. I would also like to thank the Department of Civil and Construction Engineering for their continued support and help throughout my academic tenure here. Specially, I would like to thank my advisor, Dr. Abiola Akanmu, who went far and wide to help me guide through the challenges I faced and groomed me to use my full potentials.

I would also like to thank my committee members, Dr. Osama Abudayyeh and Dr. Upul Attanayake for taking time to review my work. I would also like to acknowledge my colleagues for their help and support during this research.

Lastly, I would like to thank my parents and sibling for their continued motivation and prayers.

Syed Hammad Rasheed

TABLE OF CONTENTS

Table of Contents - Continued

Chapter

Table of Contents - Continued

Chapter

Table of Contents - Continued

Table of Contents - Continued

LIST OF TABLES

LIST OF FIGURES

List of Figures - Continued

CHAPTER 1

INTRODUCTION

1.1 Background

Over the years, there have been significant attempts to improve the construction project delivery process through the integration of innovative information and communication technologies (such as virtual models, wireless sensing and other data acquisition technologies) with the construction process. These have resulted in incremental changes with marginal level of improvement; thus there are still opportunities for improvement. It is increasingly being recognized that a more effective approach will involve an adaptive synergic integration of information and communication technologies into the project delivery process. This chapter introduces the research project by describing the issues with existing integration approaches. This chapter presents the aim and objectives of the research, the research process which was followed, and an overview of the structure of the thesis.

1.2 Problem Statement

A number of researchers have identified that construction site personnel spend significant amount of time manually tracking and recording progress data (Cheok et al. 2001; Navon and Sacks 2007). Data collected using manual methods are usually incomplete or sometimes inaccurate due to unwillingness of site workers to record these data. Studies have shown that efficient and effective construction progress data tracking

is crucial for management of construction operations (Akanmu et al., 2012; Song et al. 2012). In recent years, there has been significant improvements to existing 'component level' progress tracking approaches through the isolated use of radio frequency identification (RFID) tags (Goodrum et al. 2006; Razavi and Haas 2011 and Song et al. 2006) to the use of integrated approaches involving the integration of RFID tags and virtual models (Chin et al. 2008; Motamedi and Hammad 2009). These approaches still involve manually embedding status information into the tags. Using this approach, site personnel will need to spend significant amount of time gathering the required data in addition to accomplishing the assigned task. An example is the use of RFID tags for progress tracking by Chin et al. (2008); construction crew will need to manually scan tagged steel components to record component status information. In-order for the industry to fully adopt this integrated approaches, construction workers will need to be motivated to collect and record required progress data, thus reducing the reliability and access to real-time information. Grau et al. (2009) identified that the use of information technologies in construction is to greatly simplify the complexity and burden associated with existing construction practices and not to contribute to it.

Effective project management requires the development of a comprehensive integrated approach to report progress rapidly (Lee and Peña-Mora, 2006). This will help project managers make corrective decisions in a timely fashion. It is increasing being recognized that a more effective integrated approach will involve a tight integration and coordination between virtual models and the physical components in an adaptive way; this is termed a cyber-adaptive physical systems (CaPS) approach. This will enable

automated tracking of component status and update of the virtual model. CaPS approach has implications for real-time construction progress tracking using variety of project management tools such as earned value analysis. Hence, there is need to explore the application of CaPS for construction progress tracking. Opportunities exist for exploring other applications of this approach e.g. for facility management.

1.3 Research Objectives

The aim of the research was to investigate the applicability of the cyber-adaptive physical systems approach in enhancing integration between virtual models and the physical construction/constructed facility. The specific objectives are:

- To identify the requirements for a cyber-adaptive physical systems integration between virtual models and the physical construction. This will be carried out through detailed analysis of literature;
- To investigate the most appropriate mechanisms that can be used to achieve tight integration between virtual models and the physical construction/constructed facility, such that real-time data collection and aggregation can be undertaken for tracking applications;
- To develop software and systems architectures which will highlight mechanisms and integration approaches required for developing sample prototypes for CaPS applications;

- To develop and experiment with CaPS on laboratory-scale prototypes. This experiment will enable tracking the status of key components in the laboratory-scale physical prototype;
- Test the performance of the developed prototypes by implementing on actual construction sites/constructed facilities.

1.4 Research Scope

This research focuses on investigating CaPS approach to construction component tracking. This is explored through the development and implementation of three prototype systems for adaptive progress tracking and facility management. The prototypes are integrated using the RFID-RTLS system. The study also explored approaches to improving the accuracy of the RFID-RTLS system for the desired applications.

1.5 Research Contribution

The outcome of this research provides opportunities for enhanced access to real-time information by the project team during construction and improved lifecycle performance monitoring and control during the post construction phase of buildings. The detailed contributions of the research are contained in Chapter 5– a brief summary is presented here:

- Technologies and techniques for enhancing adaptive integration between virtual models and the physical components/constructed facility. They provide a basis for

future research into how these technologies and techniques can be collectively integrated to develop a more comprehensive cyber-physical system approach for progress monitoring and control;

- Three automated real-time monitoring and control systems, which integrate virtual models and the physical construction, have been developed. These systems have the following capabilities: real-time progress monitoring and control and facility management. The adoption of this system has the following implications for the construction industry:
    - Reduction of construction project delivery times. An example of this will be a reduction in time wastage due to improved tracking and visualization of status of tracked materials in real-time during construction and post-construction phases;
    - Reduction of uncertainty and construction risks because the activities and processes can be more closely monitored and controlled.

1.6 Thesis Structure

This thesis is organized as follows (Figure 1.1):

Chapter One: Introduction

This chapter presented the general introduction to the research and discusses the background to the research. It discusses the need for the research and outlines the aim and objectives. It also provides a structure of the thesis.

Chapter Two: Virtual-Physical Integration Efforts in Construction

This chapter reviews existing technologies and integration efforts in the architecture, engineering and construction industry. It also identifies the research gaps and opportunities for improvement. The gaps highlighted in this chapter are used to introduce the need for the use of cyber-adaptive physical systems in the next chapter.

Chapter Three: Towards Cyber-Adaptive Physical Systems Approach to Construction

Chapter 3 presents an introduction to the application of cyber-adaptive physical systems approach in construction. This chapter also highlights the requirements and enabling technologies for cyber-adaptive physical systems integration in construction. The requirements and enabling technologies are used to develop software and system architectures. The software and system architectures provide basis for development of three laboratory scale prototypes which are described in the chapter.

Chapter Four: Case Studies and Results

Chapter 4 presents case studies of the application of the prototype systems developed in chapter 3. The chapter also presents results obtained from the implementation of two of the developed prototypes.

Chapter Five: Conclusions and Recommendations

This chapter summarizes the findings of this research, presents the limitations and provides recommendations for future research.

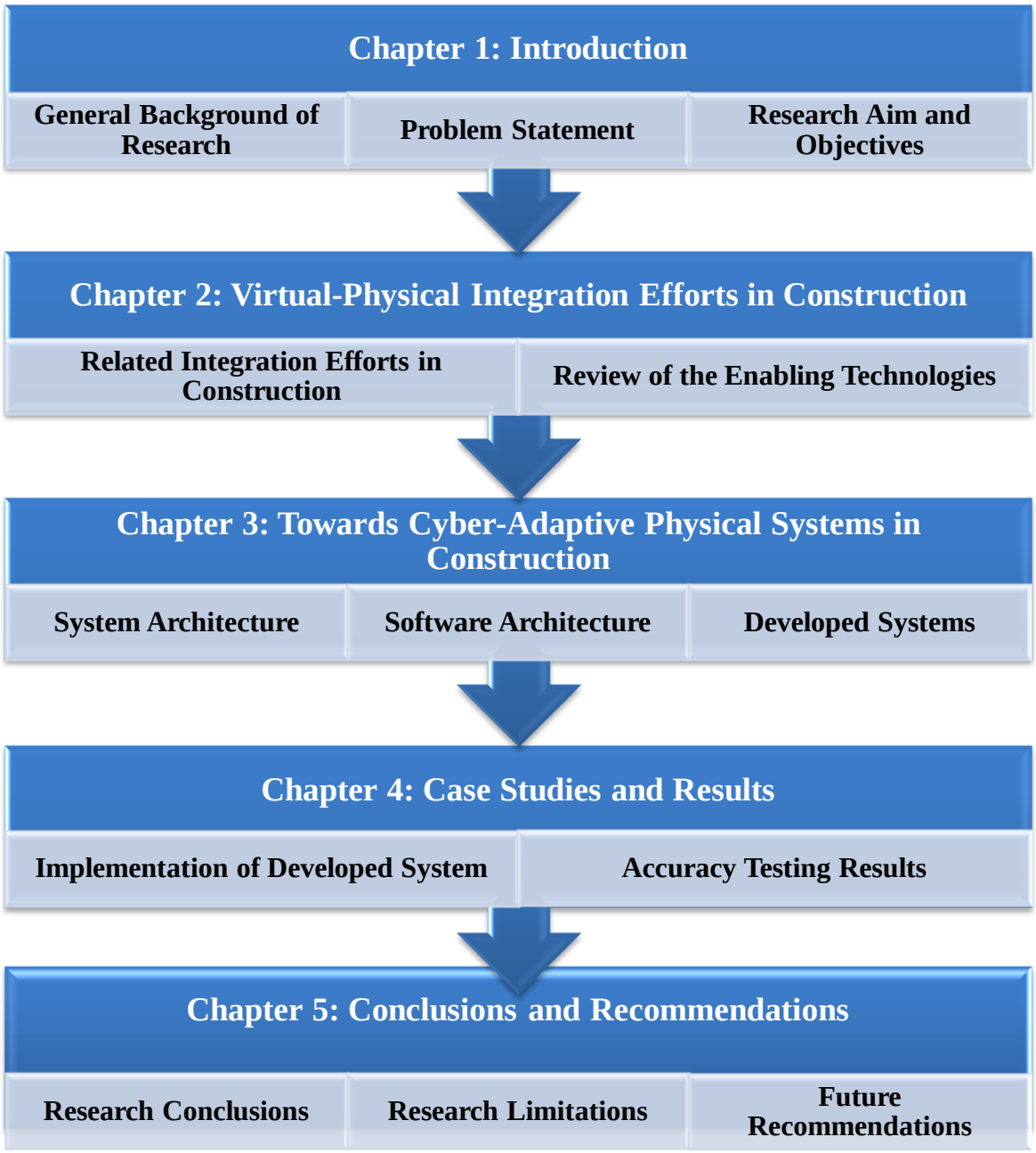

Figure 1.1: Graphical representation of thesis structure

CHAPTER 2

RESEARCH BACKGROUND

2.1 Introduction

Over the years, there have been a number of efforts in the construction industry towards integrating virtual models and the physical construction/constructed facility for automated progress monitoring and facility management operations. These efforts emerged as a result of increasing development and innovation in data acquisition technologies. This chapter discusses the different enabling technologies utilized in the construction industry such as wireless sensors and other data acquisition technologies such as photographs, laser scanners, photogrammetry, communication networks and mobile devices. This chapter also discusses related efforts made by different researchers to integrating these technologies for various tracking and monitoring applications. The limitations and opportunities for future research are described in the concluding section of the chapter.

2.2 Enabling Technologies in Integration Approaches

This section describes the key enabling technologies used in automated progress monitoring and facility management in the construction industry. The technologies

include Barcodes, Wireless sensors and RFID, Cameras, Digital Scanners, Photogrammetry, GPS, Mobile Devices and Communication network:

2.2.1 Barcode

Barcodes have long been used in a number of industry sectors for variety of tracking applications. In-spite of the low cost attributed to barcodes; barcodes have a relatively short reading range and requires a line of sight to the reader. Barcodes become unreadable if the surface becomes dirty or scratched. (Goodrum et. al, 2006).

2.2.2 Photography

For many years, photography has proved to be a very useful means of capturing site progress and activities (Brilakis et al., 2005). With recent advances in digital photography, this method of data capturing is very cost-effective. Using digital cameras, photographs can be taken at different times concurrently on an ongoing construction project and compared for identifying discrepancies between asbuilt and asplanned design. Previous research efforts can be traced back to Oglesby et al. (1989) who reported that site photographs allow analysts to focus on the details of the work while being away from site tensions and confusions, and to perform productivity analysis based on time-lapsed photographs. However, the lack of advanced technologies for automation had made the process time consuming and unattractive. Abeid et al. (2003) presented a Photo-Net II wherein time-lapse digital movies of construction activities were linked with critical path activities. In Photo-Net II, time-lapse photography was used as a source of spatial as-built

information. In a more recent study, Golparvar-Fard et al. (2007) presented an Augmented Reality (AR) system wherein 3D models are superimposed over time-lapsed photographs. Despite the advantages of photographs, severe weather, illumination and shadow inhibit the utilization of photographs for construction progress monitoring.

2.2.3 3D Laser Scanning

A laser scanner is a device that analyzes a real-world object or environment to collect data on its shape and appearance. The collected data can then be used to construct digital, 3D models useful for a wide variety of applications. 3D laser scanners enable the acquisition of accurate three dimensional as-built information in the form of dense point clouds or laser scans. The laser scanner can be used to scan the construction site at different times to generate data. This data can then be used to estimate the quantities of work performed within the time interval considered between two successive scans (El-Omari and Moselhi, 2009). Laser scanners have been used for construction quality control (Akinci et al., 2006; Jaselkis et al., 2006), condition assessment (Gordon et al. 2004), component tracking (Teizer et al., 2005) and progress monitoring (El-Omari and Moselhi, 2008; Bosche and Haas, 2008; Su et al., 2006). Inspite of the benefits, there are still a number of challenges in implementing such technology on construction sites. These challenges include mixed pixel phenomenon and discontinuity of the spatial information as well as scanning range and sensor calibration (Kiziltas et al., 2008). For example, the laser scanner cannot capture the point cloud of a moving object in its line of sight. In addition, the farther the laser scanner is from the objects, the less the level of detail within the captured components. Another potential disadvantage of laser scanning

technology is that the operating field conditions of many construction sites, including dirt, extremes of temperature and humidity must be considered because of the sensitivity of laser scanning equipment (Dickinson et al., 2009). In addition to this laser scanned data requires extensive manual post processing to remove the obstructions and clutters (undesired points) from the point cloud making the process very time-consuming.

2.2.4 Photogrammetry

Photogrammetry techniques have long assisted in realizing computer aided design (CAD) and virtual models of existing structures for architectural purposes (Dickinson et al., 2009). Photogrammetry is a three dimensional coordinate measuring technology that uses photographs as the medium for measurement. Photogrammetry uses two dimension images and basic triangulation principle to compute the locations of points in three dimensions (generally using computer software). However, it is still very difficult to get computers to recognize general objects in images except for structured scenes with severe limits on objects so humans remain an integral part photogrammetric approach to as-built models (Dickinson et al., 2009). Also, unlike the laser scanning approach, photogrammetry approaches rely on good images as such, lighting will be an issue.

2.2.5 Global Positioning Systems (GPS)

GPS is an outdoor satellite-based worldwide radio-navigation system formed by a constellation of 24 satellites, ground control stations, and end users. GPS uses triangulation from these satellites in order to determine a three-dimensional position. To

triangulate, a GPS receiver computes the distances to at least four different satellites at any given time. Distances to the receiver are computed measuring the travel time of radio signals from each one of these four satellites. Atmospheric conditions and satellites' location above the receiver influence the resulting position and its accuracy but in general the location accuracy for GPS is fairly accurate outdoors.

GPS is an established location technology offering a wide range of off-the-shelf positioning solutions for the construction industry. Guidance from one location to another (navigation), monitoring the movement of people or assets tracking, creating maps (mapping), and precise timing (timing) are the other GPS basic functions. Although GPS is limited to outdoor environments, some common construction applications such as automation of processes, guidance of equipment and materials management remains basically unexplored.

2.2.6 Wireless Sensing Technologies

Wireless sensors are generally responsible for information exchange (i.e., data transfer), serving as a bridge between the cyber (virtual) and the physical (actual) worlds (Xia et al., 2011). In the construction domain, sensors are deployed to obtain information about facilities that are being constructed, processes and equipment that are being utilized in constructing these facilities, and monitoring the as-is condition of an infrastructure throughout its service life (Akinci and Anumba, 2008). Wireless sensor technology offers increased flexibility in terms of placement, reduced cost of installation and maintenance compared with wired sensors; this makes it suitable for adoption in the construction

industry. Wireless sensors can be used to capture data from the physical construction process/activities and to update information in the virtual model. Examples of such wireless sensors include RFID tags, Placement sensors, ZigBee etc.

2.2.6.1 Radio Frequency Identification (RFID) systems

Radio frequency identification (RFID) is an automatic identification solution that streamlines identification and data acquisition. Table 2.1 summarizes the main differences between barcodes and RFID tags. Information can be written to tags attached to building components to be accessed by personnel on the construction site using RFID readers. RFID tags are classified as passive and active tags (Era-build, 2006). Passive tags have no power, RFID reader signal acts as a power signal for such tags making them very low cost. The active tags have an on-tag power supply like a battery, which emits a constant signal containing identification information. RFID-based systems have been used in different applications in construction and maintenance, such as component tracking and locating, inventory management, equipment monitoring, progress management, facilities and maintenance management, tool tracking and quality control (Kasim, 2008; Motamedi and Hammad, 2009). While RFID tags have shown great potential for identifying and tracking construction components (Hammad and Motamedi, 2009; Chin et al., 2008), the use of this tags for construction progress monitoring still involves manually updating the tags of component status.

2.2.6.2 RFID-RTLS (Real-time Location Sensing System):

Owing to the short coming of RFID tags, such as requirement of manual reading of tag using hand held readers and limited accuracy of the sensed tag location, researchers have started to explore the potential of RFID – RTLS (Real time locating systems) systems RFID-RTLS system in addition to sensing the specialized RFID tag can also locate it. Generally, the location accuracy of the system varies based on the environment (±1m outdoors; ±3m indoors) and also on the number of readers with a direct line-of-sight connection to the tag.

Table 2.1: Differences between Barcodes and RFID technology (Domdouzis, 2007)

Barcodes	**RFID**
Human intervention is required	RFID tags can be detected without human intervention
Scanning of one item per time is required	Simultaneous scanning of many items is achieved.
The barcode reader and the barcode must be in straight line in order for the barcode to be read by the reader.	No line of sight is required
Barcodes do not provide unique codes for each product but only for a group of products, thus making it impossible to have detailed information on each product separately	RFID technology allows the use of electronic product codes which are unique for each product and allow the retrieval of detailed information about a specific product.

Barcode communication is affected by the existence of objects.	RFID systems are not affected by dirt or dust.
Minimum storage capacity	Storage capacity is higher than barcode's capacity.

These inaccuracies occur because of a phenomenon known as Multipath Effect (jumping of radio signals from solid objects) apart from the inaccuracies caused by the interference of the radio signals with other wireless systems. The applications of RFID-RTLS system explored were in component tracking (Akanmu et. al, 2012) and in freight container management (Park et. al, 2006) but opportunities exist for utilizing this technology for progress monitoring, inventory management, etc.

- Location Accuracy for RFID systems

Recent studies have shown that different methods can be implemented to improve the location accuracy of RFID technologies (Ni et. al, 2004; Zhao et. al, 2007; Li et. al, 2012). Ni et. al (2004) developed LANDMARC approach. This approach was developed based on the principle that environmental factors which lead to the variation in the detection range will affect the reference tags (tags with known location) as it will affect the tracking tags. After spreading the reference tags into a grid in area of interest, the tracking tag is introduced. With the RF (radio frequency) readers the received signal strength for both tracking and reference tag is captured. Based on the minimum Euclidean distance k-nearest neighbor tags were selected. The location of the tracking tag is determined by the sum of weighted nearest neighbor coordinate. The accuracy of the

determined location in this approach is based on the density of reference tags but higher density of tags results in interference of signals. Implementation of this method also results in long latency in reporting of location. Zhao et. al (2007) improved on accuracy of LANDMARC approach by introducing the concept virtual tags, also known as VIRE approach. In this method instead of increasing the density of reference tags, authors suggest placing virtual tags and determining their signal strength based on linear interpolation. However, the signal strength calculated .for virtual tags using linear interpolation since they are related by polynomial relationship which is computationally expensive. Li et. al (2012) improved the VIRE approach by introducing the logarithmic model which incorporates the negative impact of obstructions (such as walls and floors) when calculating signal strengths for virtual tags.

2.2.6.3 Zigbee

The strength of sensors is that they can form networks and cooperate according to various models and architectures, capable of processing, sensing, computing and communication throughout the distributed mesh networks (such as the ZigBee networks). Zigbee wireless technology is a low-cost, low-power, short range communication system for data transfer applications. The low cost property allows this technology to be widely utilized in wireless control and monitoring applications. ZigBee devices can quickly attach, exchange information, detach, and then go to deep sleep to achieve a very long battery life. The low power-usage enables longer life with smaller batteries, and the mesh networking provides high reliability and larger range. Xuesong et al. (2008) identified the ZigBee wireless sensor network as having great potential for resource tracking. The key

features of ZigBee wireless technology are low complexity, low cost, low power consumption, low data rate transmissions, and they are supported by cheap fixed or moving devices (Verdone et al., 2008).

Wireless sensors have proven to be effective in tracking individual physical components and their information on site, which can be linked to the virtual model for progress monitoring and documenting as-built information. Also, tagging components on site with RFID tags will enable capturing of design changes from the model and transmitting these to the physical components on site in real-time. Placement sensors can also track when building components have been erected or installed on site, thus eliminating manual input of information. Thus, wireless sensors offer opportunities to reduce rework, thereby enhancing adherence to the project schedule and budget.

2.2.7 Mobile Devices

Mobile devices are portable/small sized computing devices having a display screen for reading and writing information. Mobile devices have long been found useful in the construction industry for progress management (Leung et al., 2008; El-Omari and Moselhi, 2008), maintenance (Kim et al., 2008) and safety applications (Yabuki et al., 2002; Wang, 2008). Examples of such mobile devices include personal data assistant (PDA), smart-phones and tablet PCs. The tablet PCs offers several advantages compared with the PDAs and smart phones: Tablet PCs are capable of accommodating several kinds of information such as project models and specifications. The major drawback of PDAs is the small screen size and limited ability to easily and quickly enter data. Most

mobile devices have external features for data capture such as barcode scanners and RFID readers. Construction personnel can easily embed information through the screen to be written to RFID tags before installing/erecting building components. They can also scan the tags and barcodes using the embedded RFID and barcodes readers respectively, to read the tag information. Another important feature of mobile devices is the wireless connectivity. Data captured using mobile devices can be transferred wirelessly to a local or remote server.

2.2.8 Communication Networks

The communication network is one of the most important technology for enhancing bi-directional coordination between virtual model and the physical construction, as it enables the transfer and exchange of information between mobile and fixed devices such as PDAs, tablet PCs and desktop computers (Aziz et al., 2006; Chin et al., 2005; Sorensen et al., 2009). Examples of some communication networks being used in the construction industry include the internet, wireless local area network (WLAN) (Wi-Fi) and the wireless personal area network (WPAN) (comprising of ultra-wide band (UWB) , Zigbee and Bluetooth). With these communication networks, data can be transferred or exchanged wirelessly between devices on the construction site, and between the construction site and the remote office, thus enhancing collaboration amongst the project team. The choice of communication network depends on a number of factors such as range, cost, data transfer rate, network topology and battery life (Xuesong et al., 2008).

2.3 Trends in Construction Progress Monitoring

Numerous researchers have investigated the use of automated systems such as wireless sensors to capture as-built information about building facilities and components. This has involved material and tool tracking, equipment monitoring, progress monitoring, facilities and maintenance management, etc. Jaselskis and El-Misalami (2003) conducted several pilot tests to explore the use of passive RFID technology and handheld readers, in the receiving process of palletized pipe hangers and pipe supports at job site laydown yards. The pilot tests indicated the usefulness of RFID in receiving the unique engineered materials, but technical difficulties were encountered because the RFID handheld readers had to be within a few inches of a tag for proper reading. In order to determine the feasibility of RFID technology for long ranges, Song et al. (2006) proposed a system where the RFID system is installed on a portal frame using 4 antennas through which a loaded flatbed trailer could be driven, thereby simulating the transport of the pipe spools. In the field test with the fixed RFID system, 20 RFID active tags were placed on the pipe spools which were put on a trailer to be driven through the portal structure. The field test indicated that RFID technology can automatically identify pipe spools with high accuracy and precision on construction sites where large pipe spools are present and long ranges are involved. Active RFID tags can be placed within tool casings to be read by either mobile or stationary RFID readers. Goodrum et al. (2006) explored the use of active RFID tags for tool tracking and inventory. The RFID tags with 32 Kb memory, 3.6 V battery, and antennae operating at 915 MHz were read by a PDA and used to track and inventory tools located in mobile truck or mobile gang boxes. The results showed that

RFID tags have the potential to improve tool inventory and allocation on a construction jobsite but the tools will have to be manually scanned with the PDA at close range.

In order to enhance the capabilities of RFID tags, Song et al. (2007) developed a 'Material Tracker' composed of RFID and ZigBee technology which can identify the data of bulk materials which is loading through RFID reader. 'Material Tracker' can also check the flow and location of materials in real time using ZigBee technology. The material ID data is scanned by the RFID reader installed on the material tracker. The data location of the material is received by ZigBee and the collected data is monitored. From the field test carried out, it is possible to track material location within a planned range through the movement of material tracker due to the short coverage range of RFID. On a large construction site, it will be difficult to manually move the material tracker. It was also not possible to get location data from the ZigBee sink nodes due to communication interference between the sensor devices; it is necessary to determine the communication range of the material tracker for locating materials.

Precast/prefabricated components can be tagged for the purpose of monitoring their location and as built condition on the construction site. Gajamani and Varghese (2007) conducted an experimental investigation on the use of RFID for schedule and inventory monitoring in real time, using a scaled building model. The RFID technology used is the 125 KHz passive RFID system which uses hand held readers, since the read range required is short. Tagged precast components are scanned and identified in a developed 3D model of the component. This information is used to validate and thus, update a Microsoft Project schedule and inventory. This approach can be used for

progress monitoring of components but has set-backs because tasks other than the ones that involve building components were not considered and the time of actual finish of the task depends on the time of scanning the tag on the component. Identification of material moving logistics or tracking location using RFID requires human effort, time and errors are prone to occur. Unlike the RFID system, the RFID-RTLS system has shown great potential for both outdoor and indoor component tracking applications by providing location information of tagged components. An important distinguishing feature of this technology is its ability to zone or map defined areas of interest. This feature makes the RFID-RTLS system suitable for tracking the status of tagged components within mapped zones such as staging and storage areas, and 'as-built' installation points in buildings. Being able to capture this status information provides opportunity for automatically updating the virtual model for real-time progress monitoring. A recent study by Akanmu et al. (2012) involved the use of RFID-RTLS system for bi-directional coordination between virtual models and the physical components by tracking the component on-site to staging area to 'installation' location; this process was automated and didn't involve human interaction except when the tags were assigned to components.

2.4 Integration Support for Project Lifecycle Processes

Systems integration has been identified as one of key approaches to help the construction industry to improve productivity and efficiency (Shen et al., 2008). Hence, a number of researchers have explored the application of data acquisition technologies and virtual technologies (databases, building information models, etc) for different aspects of the construction project lifecycle process, namely: supply chain management, progress

monitoring, facility management and deconstruction. Some of the most relevant are discussed in the following sections.

2.4.1 Supply Chain Management

Chin et al. (2005) examined the utilization of 4D CAD and RFID tags for supply chain management. RFID tags were placed on structural elements such as structural steel and curtain walls, to sense their status from the ordering stage through the delivery, receipt, and finally to the erection stage. 125 kHz RFID tags were adopted and the status of the components at each location was identified using a RFID reader to scan the tags for their ID number. Status information about components was shared and communicated through a Web-based collaboration system between the different stakeholders. Once the component was installed and the status sensed, the tag is returned to the manufacturing industry for reuse on other components. The progress information sensed by the RFID readers is reflected on a 3D model of the structure in ArchiCAD 9.0TM such that the system searches for the component with the scanned tag ID and changes its color in the 3D model to indicate its progress status. The authors limited the research to location tracking but installation instructions and design changes (which often occur during construction) could be written to the 3D model and captured in the tags. Easy access to this information could greatly enhance productivity on the project site.

2.4.2 Construction Progress Monitoring

Memon et al. (2005) developed a Digitalizing Construction Monitoring (DCM) Model which is a system that integrates 3D CAD drawings and digital images for project progress monitoring. The model estimates the percentage of progress by integrating the digital images and AutoCAD drawings to develop the actual progress bar chart. The digital images are captured from the construction site, the 3D model is developed by using Photomodeler software and AutoCAD is used to display the 3D CAD information of the intended design. Using the DCM interface, the construction personnel can input the 3D CAD drawing, digital image and as-planned schedule into the database. Information from the 3D CAD drawings and digital images are integrated using event oriented programming (Visual Basic 6.0) and compared with the as-planned schedule of work to estimate the percentage of progress. The percentage of progress is used to produce the as-built schedule bar chart in Microsoft project. Digital cameras can only capture images within their view. Thus, information about the entire project will require a number of cameras and manual input is needed to direct the cameras to capture desired information. Also, this is not suitable for capturing individual components but processes. Thus, this makes it difficult to determine the source of a discrepancy because more information is needed.

Golparvar-Fard et al. (2009) also utilized digital cameras and 4D CAD for the visualization of progress deviations. This approach involves the superimposition of 4D CAD of the as-planned model into time-lapse photography and videotaping of the as-built progress data collection. The progress information can also be visualized through

augmenting the as-built photograph with the as-planned data. Data was collected from as-built and as-planned environments and merged to form a 4D as-planned simulation and time-lapsed photographs. The as-planned model was superimposed on the site photographs. This superimposed image enables discrepancies to be detected, quantified and cost values extracted. This process could also be time consuming as a large number of photographs will need to be taken to adequately capture the as-built information. In the case of bad weather such as snow, the use of photographs will be inadequate. In a similar study, Kim et al. (2009) employed image processing techniques along with a closed-circuit television (CCTV) camera and a wireless local area network to monitor the construction operations of a cable stayed bridge. The image data of the construction process was captured using the CCTV camera and automatically transferred from the construction site to the head office storage device via a WLAN. This enabled construction managers at the head office to remotely monitor the construction progress of the cable stayed bridge via an Internet connection. The construction progress was analyzed using image processing techniques along with the 3D CAD model of the bridge for comparisons between the acquired data and as-planned data. As a camera cannot adequately capture the whole construction process, a number of cameras will be needed and a lot of time is required to process the images.

Laser scanned data has also been integrated with 3D model for progress tracking. Boche et al. (2008) presented an approach which involves integrating 3D CAD modeling and time-stamped 3D laser scanned data for automated project progress tracking. The approach enables an automated recognition of 3D CAD model objects from site laser

scans. Scans obtained from different days or from different site locations are compared by recognizing the 3D objects which differ in each scan to determine the construction progress. This object recognition approach can recognize the 3D model objects being built. It also enables tracking of the progress of the construction of pre-fabricated and installed elements. The authors carried out an experiment using a 3D model and 5 laser scans of a building. First, the scans were compared with the 3D model for object recognition. Subsequently, the scans were analyzed based on the proposed approach to identify project progress in 3D. This approach will require taking a number of scans from different sides, thus making this approach time consuming.

Contrary to the above efforts which deal with tracking progress of activities, Wenfa (2008) developed an integrated model of radio-frequency identification (RFID) and four-dimensional computer-aided design (4D CAD) for tracking the status of construction components. Construction components (such as pipes, equipment, steel columns and beams) are tagged with RFID passive tags and a RFID reader is used to track their status from the manufacturing or fabrication plant to the construction site where they are installed. Once the components are installed, construction personnel use a Personal Digital Assistant (PDA) with an embedded RFID reader to manually track the status of the components. This status information is captured in a 4D CAD model of the initial project schedule. This enables comparisons between actual and initial work schedules. Since passive tags are utilized, tag information has to be read manually at close range. This makes this approach unsuitable for project sites with a large number of components.

2.4.3 Facility Management

Menzel et al. (2008) demonstrated the capability of RFID and a Web-based virtual prototype in enhancing access to maintenance information via PDAs and laptops. Integrated RFID-based information was designed and installed into a Web based prototype supporting facilities management in order to monitor the performance of buildings. HVAC components were tagged with active RFID tags integrated with temperature sensors. These tagged components were linked with their virtual representations and context information was captured. Real time data stored on active RFID-tags were monitored by the tradesmen to test working conditions and comfort aspects. The tag triggers an alert if it detects an extremely high temperature on a particular component and the component gets checked. Trades-men and engineering experts can use the Web-based virtual prototype to identify the tagged components. It is difficult to retrieve historical performance data about the tagged component except via chatting with the engineering experts through the web interface.

Motamedi and Hammad (2009) proposed the use of passive RFID tags and BIM for lifecycle management of facility components. The authors proposed permanently attaching RFID tags to facility components where the memory of the tags is populated with BIM information taken from a BIM database. The BIM data stored on the tags provides a distributed database that allows access to different players who do not have real-time access to a central database. The authors demonstrated the feasibility of this approach by carrying out two case studies: The first case study demonstrated how the proposed approach will facilitate progress monitoring of construction projects. The status

and timing information of a building component obtained from the BIM database can be merged with geometrical information from a 3D model to produce a 4D model which will help project managers and the facility management team to better visualize the status of the facility. In the second case study, RFID tags were used for storing information about fire safety equipment. Crucial information is converted to binary codes and stored on tags attached to extinguishers and valves. This information includes history and condition of the extinguishers and valves for inspectors and maintenance personnel without access to the central database. With the floor plans, the inspectors and maintenance personnel can visually locate the tags and access information from the tag using RFID readers. This approach is not suitable where a large number of components are to be tagged, as passive tags have limited memory and will need to be read at close range.

Meadati et al. (2010) investigated the integration of RFID and BIM for enhancing access to facility management information during the operation and maintenance phase of buildings. The authors propose permanently attaching RFID tags to building components and manually uploading building information (such as operation and installation manuals, specifications, warranty) into the corresponding virtual components in an Autodesk Revit Architecture model. During the operations and maintenance phase, the facility manager can easily scan a tagged component with an RFID reader and the component is highlighted in the BIM model. The highlighted component enables the facility manager to identify the component and access the embedded information in the Revit model.

2.4.4 Asset Management

Goodrum et al. (2008) demonstrated the capability of active RFID tags and mobile readers to improve the efficiency of tracking tools and their availability. The authors proposed attached 32KB memory size active RFID tags were installed inside the most commonly used hand tools such as corded hammer drill, a portable band saw, and a reciprocating saw. Operation and maintenance information data was stored in the memory of the tags. The system was implemented in different scenarios such as Interior renovation project, plant maintenance project, plant expansion project and construction of generator and chiller station for endurance and exposure testing of the tags in realistic use and varying temperature cases. Authors concluded that although the active RFID technology had significant potential to improve tool inventory and allocation, the potential limitation that needed to be addressed was the determination of the location of the tracked tagged since the system was not able to pin point the location or distance of the tracked tag.

2.4.5 Earned Value Management

Recent studies suggest that researchers have used earned value management as a metric for monitoring progress of construction projects. Earned value is a management technique which links the schedules and technical performance metrics resource planning and usage. Researchers have integrated different data acquisition technologies and virtual models; such as laser scanners and 4D CAD (Turkan et. al, 2012), RFID and 3D building information model (Azimi et.al, 2012), RFID, 3D photographs and schedule (El-

Omari and Moselhi, 2011), time lapsed photography with 4D model CAD (Golparvar-Fard et. al, 2009) with earned value to determine the progress of the construction project. The challenges associated with these studies are either some approaches requires extensive manual processing of information (scanning bar codes or registering point clouds from laser scanners), post processing of information and /or not fully automated approaches.

2.5 Opportunities for Improving on Previous Integration Approaches

While previous studies have demonstrated the potential for significant improvements to the project delivery process based on better modeling of a proposed facility and tracking of construction components and resources, the following are the limitations of the existing work:

- There are limited mechanisms for integrating virtual models and the physical components in an adaptive way such as to enable automated real-time status tracking. For example, when a component arrives on the job site, the staging area or when the component is installed, there is no means of automatically updating the virtual model to reflect this. The existing approaches involve manually embedding status information in the tags which is unreliable and time consuming;
- There are limited opportunities to automatically capture project status using earned value management. Existing approaches still require the project team to manually embed progress information the tags. Access to this real-time

information will depend on workers motivation, thus reducing the reliability of the captured data;

- Existing approaches to indoor asset tracking using the RFID-RTLS system does not allow visually tracking the location of tagged component based on the context of the user. Opportunities exist for exploring the use of RFID-RTLS system for navigating and visually tracking the location of tagged components within constructed facilities.

2.6 Summary

In spite of several efforts to integrate virtual models and the physical construction, the benefits of virtual models have not been fully utilized. Virtual models contain semantic representations of construction components, and as-planned data which is constantly updated and required on the construction site. Virtual models enable visual identification of construction components which is required for identifying and distinguishing each component. On the other hand, the physical construction and constructed facility contains physical representation of the virtual components. Thus, this chapter presented related integration efforts and gaps in the construction industry.

A number of enabling technologies were discussed. However, of all the technologies described, RFID-RTLS tags have the potential for identifying and distinguishing components which can be linked to their virtual representation in the model. This link will enable monitoring or status tracking from the model thus providing opportunity for control. The next chapter introduces the CaPS in the construction

industry. It also highlights the role of the data acquisition technologies in enhancing the CaPS integration.

CHAPTER 3

TOWARDS A CYBER-ADAPTIVE PHYSICAL SYSTEM IN CONSTRUCTION

3.1 Introduction

Automated construction progress monitoring can be achieved by integrating physical systems with their corresponding virtual design using sensing technologies. The level of automation can be enhanced a by adapting the virtual design to the physical system using computing systems. This chapter describes the CaPS approach. Based on the review from the preceding chapter, this chapter presents the requirements for CaPS in construction. The enabling technologies for achieving CaPS are presented. This chapter also describes the development of three prototype systems to illustrate the CaPS approach. The software, system architecture and use case analysis for each prototype is presented.

3.2 Cyber-Adaptive Physical Systems

The need for control of physical operations emerged as a result of growing developments and innovation in computing systems design. Increasing investigations into the development of new sensors for improvement of physical and engineered operations offers opportunities for enhancing the level of automation and adaptivity between the virtual design and the physical systems. For example, the development of the radio frequency identification real-time location sensing system (RFID-RTLS) (which provides real-time location tracking of tagged components) is an improvement over the existing

RFID system. The RFID-RTLS system greatly reduces the computational effort required to track the previous RFID tags. Furthermore, RFID-RTLS also offers opportunity for tightly integrating (in an adaptive way) physical components with their corresponding virtual representation in the model using applications; the resulting system can be termed a cyber-adaptive physical system (CaPS).

3.3 Key features of a Cyber-adaptive physical systems approach to construction

For a system to be considered a CaPS, it must be able to bridge the virtual design to the physical world through sensors, storage and applications, intelligent and decision systems (Wu et al. 2011). Thus, the key requirements of CaPSs are illustrated in Figure 3.1 (Bhave et al. 2010; Xia et al. 2011):

- **The Physical to Cyber Bridge:** This is the sensing process, which involves using sensors and data acquisition technologies to acquire information about components or phenomenon.
- **The Cyber to Physical Bridge:** This represents the actuation which shows how the sensed information affects the system. In the context of construction, the actuation is taken to mean making control decisions from the sensed information and/or using the sensed information to physically control building components.

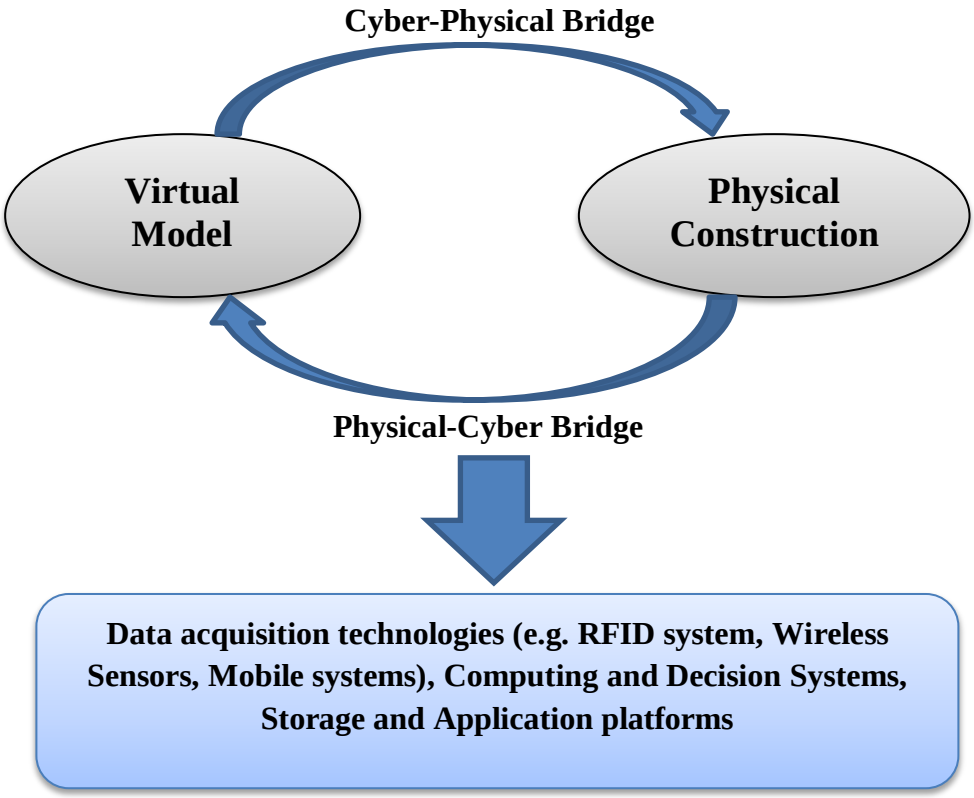

Figure 3.1: Key features of Cyber-adaptive Physical Systems

3.4 Requirements for a Cyber-adaptive physical systems approach to construction

CaPS aims to deeply integrate virtual design and the physical system. CaPSs are different from wireless sensor networks, embedded/real-time systems, traditional desktop computing etc. The following are some of the defining characteristics of CaPS:

- **Context-adaptive:** The context of the physical environments needs to be understood by the virtual design. This is defined by the level of interaction between the virtual design and the physical system using the RFID-RTLS system. The RFID-RTLS system has a context adaptive feature which can be integrated

with the physical components for tracking and monitoring purposes. This is enhanced by the integrating software.

- **Real-time capability:** The system needs to be closely monitored to capture time critical and relevant information. Information retrieval, aggregation, processing and communication must occur before critical information is obsolete or replaced by new or contradicting information. This is the demonstrated by the capability of the developed CaPS system to provide real-time contextual information at the point of need e.g. through a visualization tool (in the case of BIM).
- **High degree of automation:** CaPS can be partially or completely autonomous. The level of automation defines the efficiency of the system. This is particularly important in tracking, monitoring of resources and safety applications.

3.5 Enabling Technologies

Based on the literature review in Chapter 2, the following technologies have been identified as been suitable for CaPS to construction:

3.5.1 RFID-RTLS system

The RFID-RTLS system (shown in Figure 3.2) consists of a master reader, slave readers, RTLS tags and software. The RTLS tags are battery powered active tags usually attached to assets/components and personnel being tracked. The tags relay location information every predetermined time interval (also known as cycle time) to the master and slave reader. The slave and master readers are positioned at the boundaries of areas

of interest. The slave readers determine and relay the range of the tags to the master reader. In addition to determining the range of the tags, the master reader captures the location data from each slave reader and relays it to the location engine software (installed in the associated computer system). The location engine software determines the location of RTLS tags based on time-of-arrival (TOA) of radio frequency signal from the tags to the reader; the distance between the tags and reader (with the master and slave readers positioned at known reference locations) are determined based on the round-trip time of the signal travelling between the two entities.

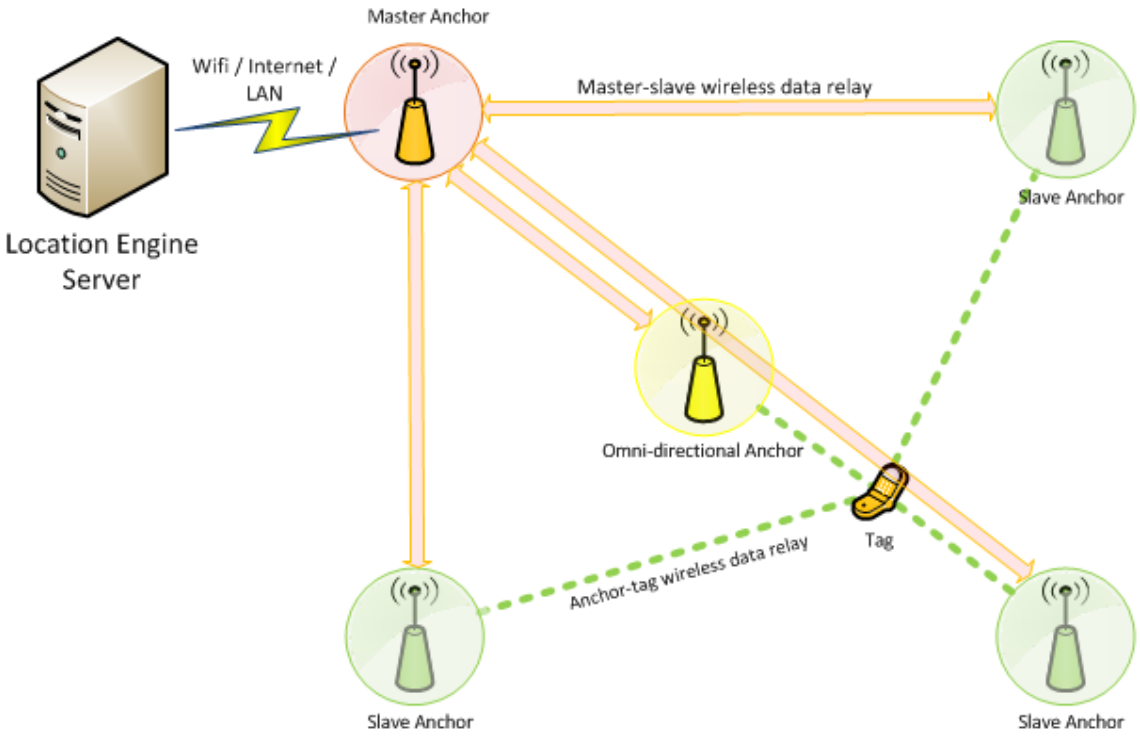

Figure 3.2: Component overview of the RFID-RTLS system

3.5.2 Building Information Model (BIM)

Building Information Model (BIM) represents all the information related to a building's physical and functional characteristic along with project life cycle information (Salman et. al, 2008). BIM is usually used as a central database of information by stake holders to help make informed decision on a project. Potential applications for BIM are in facility management, cost estimation, construction sequencing (schedules for material orders, delivery, etc), and clash detection between building components. BIM enables visualization of building systems and subsystems in 3 and 4 dimensional.

3.5.3 Mobile Tablet

Mobile tablets such as PDA (Personal Digital Assistant), Tablet PCs have long been found to be useful in the construction industry (details of industry applications are described in the Chapter 2). Some specific applications of mobile devices in the architecture, engineering and construction (AEC) industry includes on-site blue print access, data capturing and visualization. These devices can also be used for project management applications such as activity review, monitoring, status updates (Chen and Kamara, 2008). iPad is typical example of mobile devices. The iPad has an inbuilt gyroscope for measuring the position and orientation of users. It also has an inbuilt camera which can be used for capturing the real world view. This real world view can be overlaid on building plans or models (Augmented Reality) such that as users move within a space, the locations of the users are captured using the gyroscope, thus providing opportunities for the delivery of relevant context related information.

3.6 Prototype Development

Based on the enabling technologies described above, three prototype systems were developed. These prototypes enable automated component tracking on the job site, asset tracking within indoor environments and earned value analysis. Each of the developed prototypes is described above:

3.6.1 Prototype #1: Automated Component Status Tracking based on Spatial Mapping

3.6.1.1 Requirements

In light of current practices in the construction industry (as discussed in Chapter 2) opportunities exist for improving on-site component tracking. One of such opportunities lies in automated component status tracking. This is an improvement over the existing or current practice of manually embedding status information into RFID tags or moving tagged components through fixed readers to capture component status. This process is error prone as the data collection process is dependent on workers motivation, thus reducing the reliability of the collected data. Another opportunity lies in the automated tracking of component installation status. Although there have been efforts involving the use of RFID and GPS for installation status tracking, the use of GPS is limited to outdoor environments. This becomes particularly important because as the building moves from partially to fully completed; access to component location data becomes unrealistic. To overcome the highlighted challenges, a CaPS to automated component status tracking must satisfy the following requirements:

- Be able to track accurate location of tagged components;
- Be able to understand on-site spatially mapped locations;
- Be able to update with the virtual models with the tag data.

3.6.1.2 Prototype System Design

The prototype system was developed by integrating a physical laboratory scale prototype (Figure 3.3) and its corresponding virtual model using the RFID-RTLS system. The overall setup of the prototype system is shown in Figure 3.4. The key steps involved in the prototype system development include the following:

- Development of the Virtual model

  A virtual model of a small scale building was developed using Autodesk Revit which was exported into Autodesk Navisworks. The model serves the purpose of enabling visualization of status of tracked building components and information captured from the project site. The model also enables embedding of model updates or critical information that needs to be communicated to the construction site in real-time. Autodesk Navisworks was utilized because it offers a .NET application programming interface (API), which enables users to write custom plug-ins to drive Autodesk Navisworks from outside the graphical user interface (GUI) and automate tasks like changing material properties.

- Development of the Physical Laboratory Scale Model

A laboratory scale physical prototype of the Navisworks model was constructed. The physical prototype is 1m x 0.75m x 1m in dimensions and consists of six detachable components (Figure 3.3). The physical prototype was designed as detachable components to enable easy removal when tracking placement during the experiment. The components of the laboratory scale physical building prototype were tagged with RTLS tags as shown in Figure 3.3.

Figure 3.3: Laboratory scale physical prototype

- Development of Prototype Application

A prototype application was developed to integrate the virtual model and the physical laboratory scale prototype using Visual Studio .NET 2010 (C#). A brief graphical representation of the system architecture for the prototype is shown in Figure 3.4. The prototype application demonstrates the potential of the developed system for:

- Communicating with the RFID-RTLS system reader to acquire location coordinates and tag ID data;
- Collecting the location data of each tagged component;
- Filtering out the inaccurate location data of each tagged component;
- Assigning 'On-site', 'Stored' or 'Installed' status to each virtual component based on the comparison of the current tag location with the planned location;
- Storing the captured location data and status for each tagged component in a MySQL database and;
- Monitoring changes in the database and reporting the updates in the virtual model based on the status of tags.

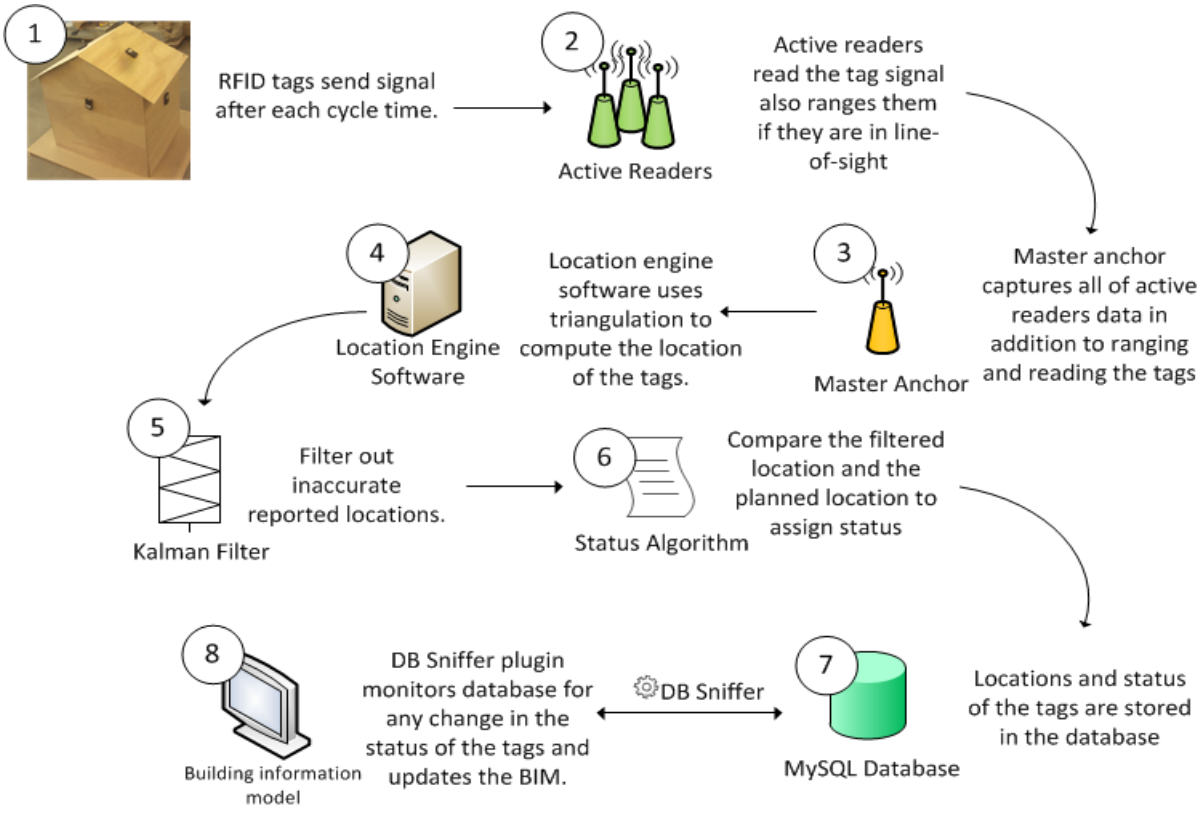

Figure 3.4: Comprehensive system overview for component tracking prototype

- Kalman Filter Algorithm:

Previous studies suggest that Kalman filter can be used to estimate the location of the moving tags with fairly good accuracy (Rohrig and Muller, 2009), it can be applied to stationary locations assuming the noise is a function of time (Bekkali et. el, 2007).

Kalman filter addresses the general problem of trying to estimate the state $x \in R^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_{k+1} = A_k x_k + B u_k + w_k \tag{3.1}$$

with a measurement $y \in R^m$ that is

$$y_k = H_k x_k + v_k \tag{3.2}$$

Equation 1 implies that the next predicted state (x_{k+1}) is a linear combination of the previous state (x_k), control input (u_k), process noise ($w_k \sim N(0,Q_k)$) and measurement noise ($v_k \sim N(0,R_k)$). A and B represent the matrices which relate the state at time k and k+1 and control input to state x respectively. Equation 2 implies that the measured (observed) value (y_k) is a linear combination of the state value (x_k) and the measurement noise (v_k). H represents a matrix which relates the state to the measurement z_k. Kalman Filtering is an iterative process which start by an initial value of state and its error covariance, which results in "prediction" (known as *priori* state) of the state before the "correction" (known as *posteriori* state) is applied for better estimation of process states (Figure 3.5) for more details or refer to Welch & Bishop (1995).

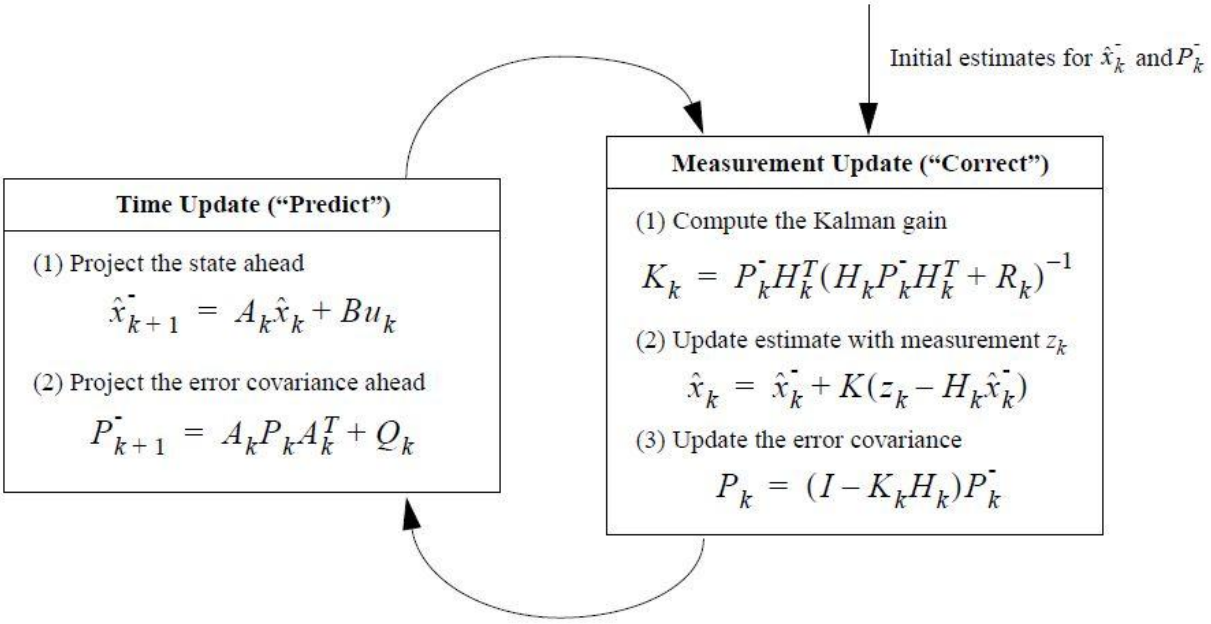

Figure 3.5: Brief overview of Kalman Filter (Welch and Bishop, 1995)

For this case it is assumed the accuracy of system will be sufficient when the component is tracked when it is 'on-site' and in 'storage'. For tracking the tag to the 'installation' location higher level of accuracy will be required hence the filter process was only applied to the tracking of tag when they were being 'installed'. Since the installation location of the tag is known, it can be assumed that the tag is fixed. Assume that the process noise Q is negligible ($w_k = 0$), and state of this system is not modified by any control input (B = 0). Also, since there will no transition among predicted and current state, A is assumed to be zero. With these assumptions the process model (Equation 3.1) becomes,

$$x_{k+1} = x_k \tag{3.4}$$

For the observation model, H matrix is assumed to be one with the measurement noise (R) to be variance of the input data for each dimension (x and y). Based on this assumption equation (Equation 3.2) is reduced to,

$$y_k = x_k + v_k \tag{3.5}$$

To initiate the filter, initial state estimate is set to the average of the input data (x_0) and the initial error covariance (P_0) is determined. Value for P_0 is determined using the population covariance of the two data sets (reported x coordinate and reported y-coordinate). This process is repeated for each tag.

3.6.1.3 Software Architecture

The software architecture (Figure 3.7) for the developed prototypes is distributed over two middleware, namely RTLS-DB and DBSniffer. Both middleware are connected to database which serves as a bridge to information from the RTLS-DB-BIM (Building Information Model). RTLS-DB conveys the information from the RTLS system to the database which is extracted by DBSniffer for use with BIM

- RTLS-DB Middleware:

  This middleware was developed to capture location information from the RFID-RTLS location engine software. The middleware also served as a terminal for the user to input the zoning information (such as the name, size and location of the zone) into the system. A statistical filter known as the Kalman filter was embedded into the middleware which filtered out the inaccurate reported location of the tag (For code refer to APPENDIX A – RFID-DB C). All the information from the middleware

was stored into the database from where the DBSniffer middleware could extract information.

- DBSniffer Middleware:

  This middleware was developed as a plugin for Autodesk Navisworks Manage 2013 using .NET Application Programming Interface (API). This middleware was also connected to the database and using the timer function, it scanned the database table (‘taginfo’) for tag information. As soon as the database is populated with the tag information (e.g. TagID) from the RTLS system, the plugin launches a search query within Navisworks for any 3D component with the corresponding tag. Based on the zone reported in the database, the plugin automatically updates the color of the searched 3D component. If the reported zone is ‘On-site’, the component color changes to red, if reported zone is ‘Storage’, color changes to white and if it is ‘Installed’ the color changes to green (For code, refer to the APPENDIX B – DBSniffer Code).

Both middleware were connected to a centralized database which acts as a bridge to relay information between both RFID-RTLS software and Autodesk Navisworks. The database server (MySQL 5.6.10) was installed on the workstation and a database schema was setup. Two database tables were created to capture the tag information. The first is called ‘accuracy_table’. The ‘accuracy_table’ stores the Tag IDs, RSSIs, reported ranges, and locations of the tags. The second table called ‘taginfo’, stores the last known work zone within which the tag was last found, and the corresponding date and time. (Figure 3.6).

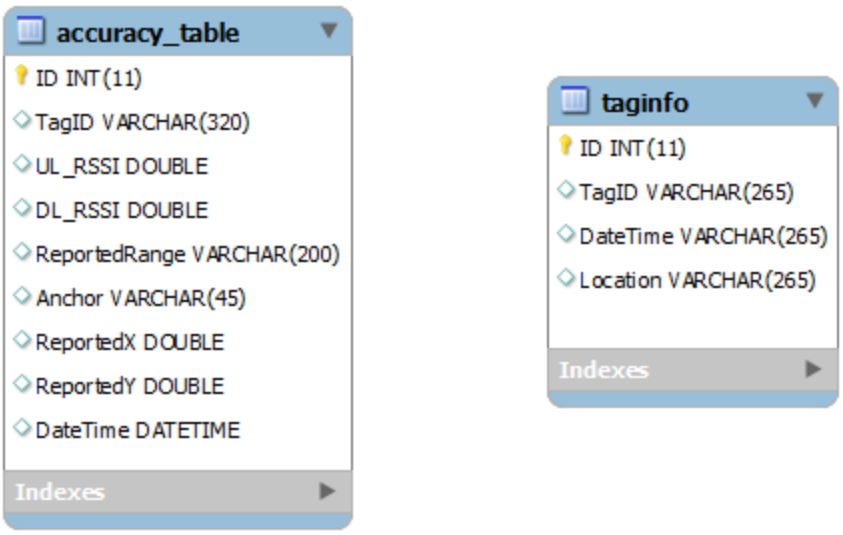

Figure 3.6: Database table structure for component tracking prototype

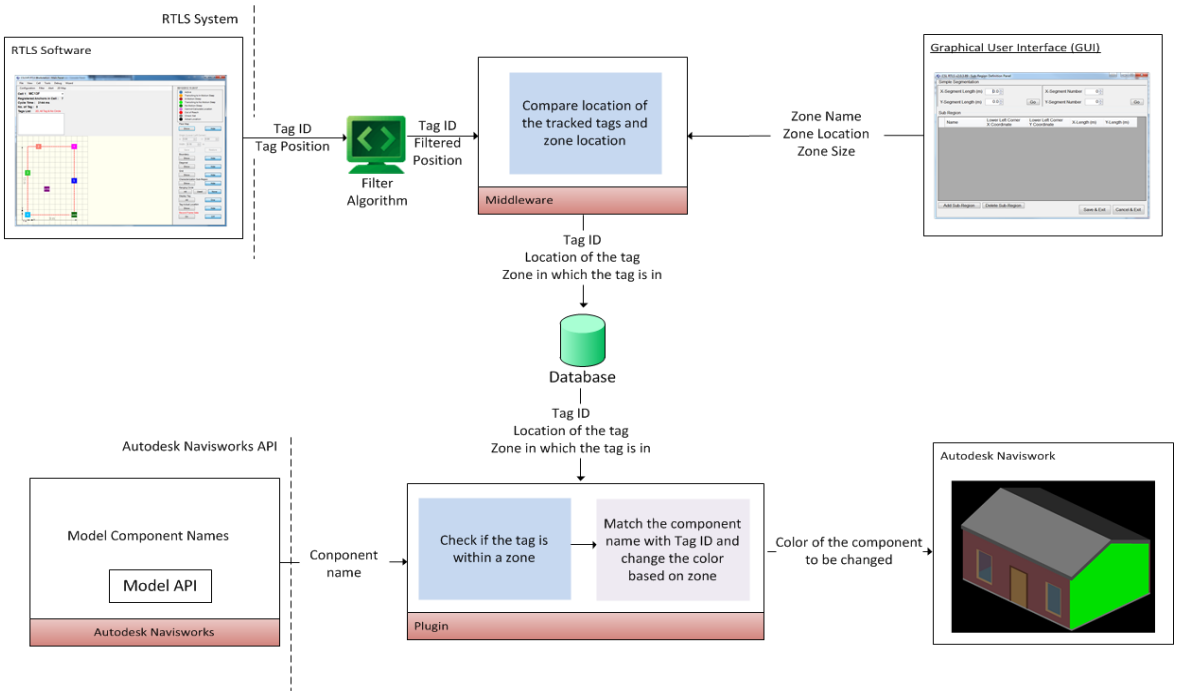

Figure 3.7: Software architecture for Prototype #1.

3.6.1.4 Use-case Analysis

The use-case diagram for the prototype is shown in Figure 3.8. The use-case depicts how the project manager will interact with the developed prototype system.

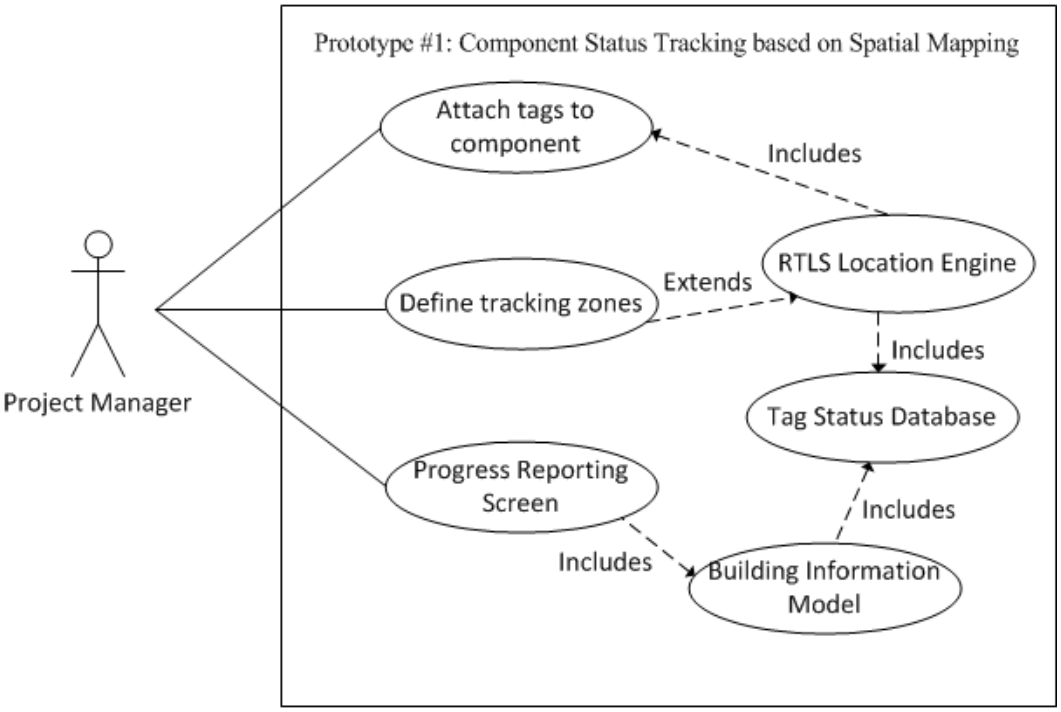

Figure 3.8: Use-case for component tracking prototype.

Primary Actor: Project Manager / Site Supervisor

Short Description: The developed prototype tracks the tagged components and updates the associated virtual model based on location of component. This helps project manager to track the status of components and make decisions.

Preconditions:

- Building components are tagged with RFID-RTLS tags either from the manufacturer's yard or on arrival on site.
- A virtual model of the facility must be developed.
- The virtual components must be linked with the corresponding physical components.
- RFID-RTLS system must be deployed on the site.
- All the tags must be powered on.

Basic Flow:

- 3D model of the project must be created with the component name being the tag IDs.
- The project manager must make sure that the appropriate components are tagged.
- After launching the RFID-RTLS software, the project manager must define zone under the Observation Area in the software window.
- There can be three possible zones, 'On-site', 'Storage' and 'Installed'.
- Once the zones appear on the interface the project manager must click 'Start / Stop' button in the DBSniffer plugin in Autodesk Navisworks.
- System will take over from there and will update the virtual counter part of the physical component based on its location.

Post-conditions:

- The 3D model is updated with different colors based on the status of the component.

3.6.1.5 Case Study for developed prototype:

To determine the accuracy of the RFID-RTLS system for tracking components in mapped locations, two experiments were conducted one indoor (laboratory) and the other outdoor (field). The details of the experiments are described below:

- Indoor Experiment

The indoor experiment was carried out in the Cyber-Physical Construction Laboratory in the Department of Civil and Construction Engineering, College of Engineering and Applied Sciences, Western Michigan University, Kalamazoo. The aim of this indoor experiment was to test the functionality of the developed prototype system, prior to testing in the outdoor environment. The laboratory was mapped out as a "construction site". The master and slave readers were deployed at varying interval from 4m-13m on the perimeter of the laboratory. The Omni-directional reader was placed in the center of the laboratory to ensure 360^0 coverage of the laboratory. An assumed area of 2m x 2m was also mapped out for component storage. The physical laboratory scale prototype was placed at a known location within the laboratory (this indicates the as-built location of the physical prototype). The plans of the laboratory test area indicating the locations

of the readers, staging area and physical laboratory scale prototype are shown in Figure 3.9.

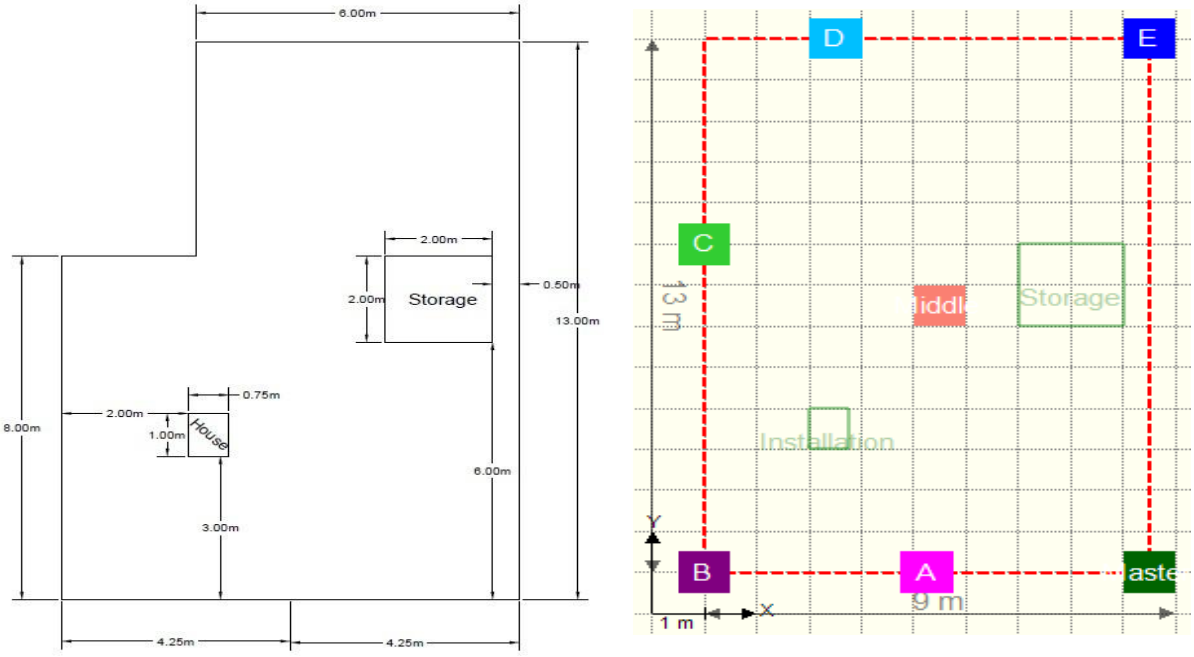

Figure 3.9: Physical Plan of indoor testing area (left), RFID-RTLS map showing mapped locations and locations of the readers (right).

- Outdoor Experiment

The outdoor experiment was carried out in the employee's parking lot next to the College of Engineering and Applied Sciences, Western Michigan University, Kalamazoo. An area of 6m x 12m was mapped out as the "construction site". An assumed entrance was also mapped out. The readers were deployed at an interval of 6m along the perimeter of the mapped area with the exception of the Omni-directional reader which was placed in the middle of the mapped area. A storage area of 2m x 2m was also mapped out within

the experimental ‘construction site’. The plan of the test area showing the location of the readers is shown in Figure 3.10.

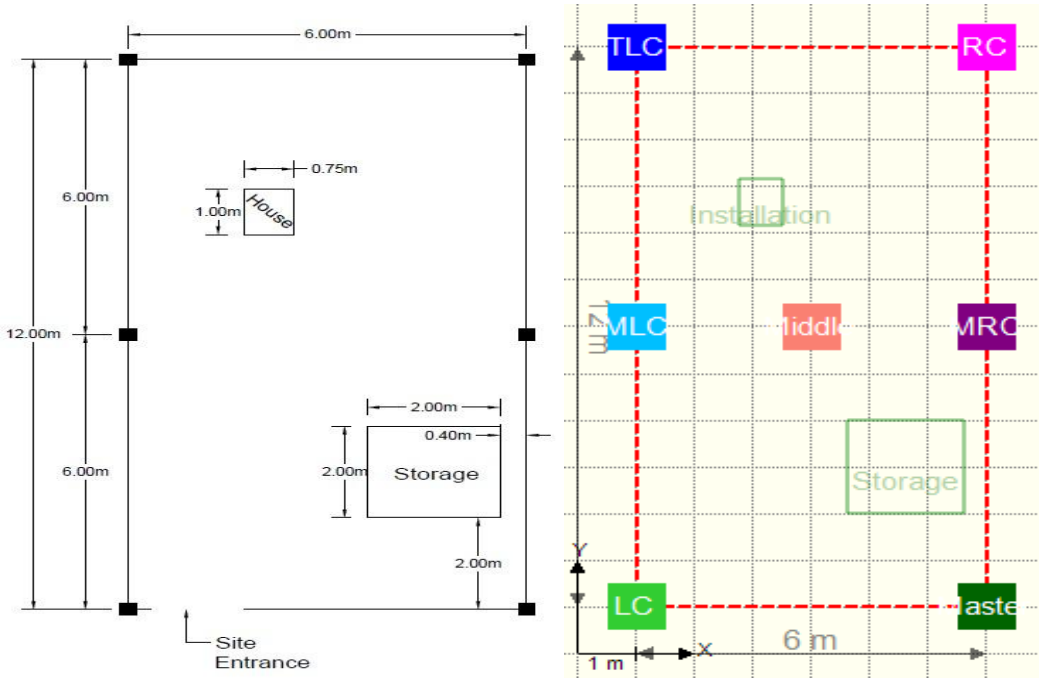

Figure 3.10: Plan of outdoor testing area (left), RFID-RTLS map showing the mapped areas and the location of readers (right).

- Methodology

For both indoor and outdoor experiments, positions of the readers were captured manually and defined in the prototype application along with the storage and the installation locations for the tags. The components of the physical laboratory scale prototype (such as wall and roof) were tagged with the RTLS tags. The logistical sequence of the components was simulated similar to the actual construction site. The sequence commenced by bringing the components on-site, then storing the

components in the storage area and finally installing the components in the as planned installation location. During the experiment, the following parameters were noted:

- The range of the tagged components from each reader as determined by the system. The RFID-RTLS system uses the range to compute the estimated location of the tagged components;
- The actual location of each tagged component. Measuring the actual location of the tagged components is important for determining the accuracy and reliability of the RFID-RTLS system for components status tracking. This can be compared with the estimated location to identify the deviation or localization error;
- The time of recognition of the tags (by the RFID-RTLS system) as the tagged components arrives the observation area. This is important as it determines when the virtual model will be updated of the status of the tagged component and;
- The actual time of introduction of the tags into the observation area.

The above parameters were captured at every stage of the sequence i.e. during component arrival on site, in the storage area and in the 'installed' location or position.

3.6.2 Prototype #2: Asset tracking using Tablet PC integrated with RFID-RTLS system

3.6.2.1 Requirements

Being able to effectively track assets is critical for both supply chain and facility management operations. In current practice, on-site tracking of components or assets is generally done manually. As discussed in Chapter 2, although researchers have tried to automate this process by integrating different technologies (such as GPS and RFID tags) opportunities still exist for improvements.. One of the challenges of asset tracking is the significant effort required in locating tags. Existing approach involves using propriety visualization software embedded in the RFID system to track components. This visualization software are capable of assisting users navigate through spaces to locate components of interest. To improve in existing efforts, a prototype system was developed to enable facility managers navigate and locate tagged components based on their context within buildings.

3.6.2.2 System Architecture

The system architecture is illustrated in Figure 3.11 brings together the RFID-RTLS system and iPad mobile device as a framework for enhancing asset tracking within facilities. The system architecture is described as follows:

Tagged assets are ranged by the slave and master anchors (these are deployed at the perimeter of the area under observations). The observation area is also called a cell). It is imperative that the anchors have a line-of-sight connection to tags for ranging.

Distance or ranging data between each tag and slave anchor is captured by the slave anchors and communicated wirelessly to the master anchor. The master anchor communicates the distance data to the location engine server (Figure 3.12). The server processes the range data by computing the tag locations using the triangulation method. Triangulation method is used to determine the location of the tag based on the range of the tag from each anchor. Within the location engine, the range of the tags are first sorted in ascending order based on the Received Signal Strength Indicator (RSSI) implying that for each tag, the anchor range with highest RSSI is assigned highest priority and then the subsequent ranges are sorted. At most, seven ranges with highest RSSIs are taken into account and based on that, the location of the tag is determined. The location engine server then communicates this location data to the iPad via wireless network connection.

The iPad has a location visualization application (CS5900 mobile software for RTLS systems) installed. This software enables a user to view the real-time positions and easily navigate to a RTLS tagged component. On launching the software from the iPad, the interface in Figure 3.13 is initiated. The uniform resource locator (URL) pointing to the location of the location engine server running on the work is embedded in the interface as shown in Figure 3.13. This enables the tags and location data from the RTLS location

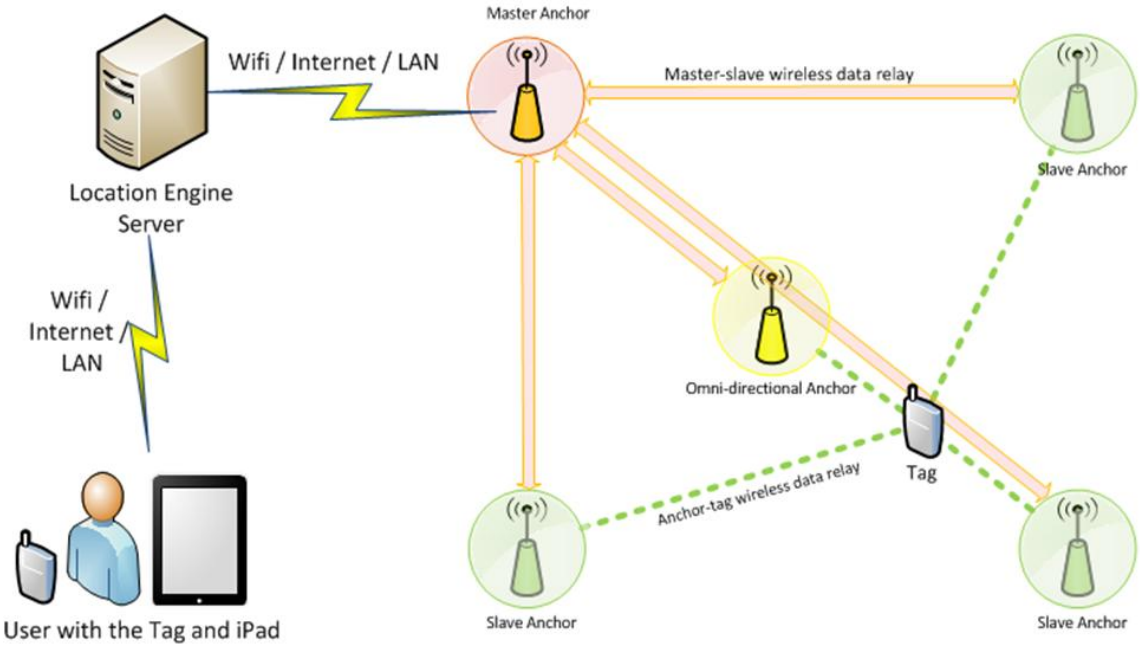

Figure 3.11: System architecture for asset tracking prototype

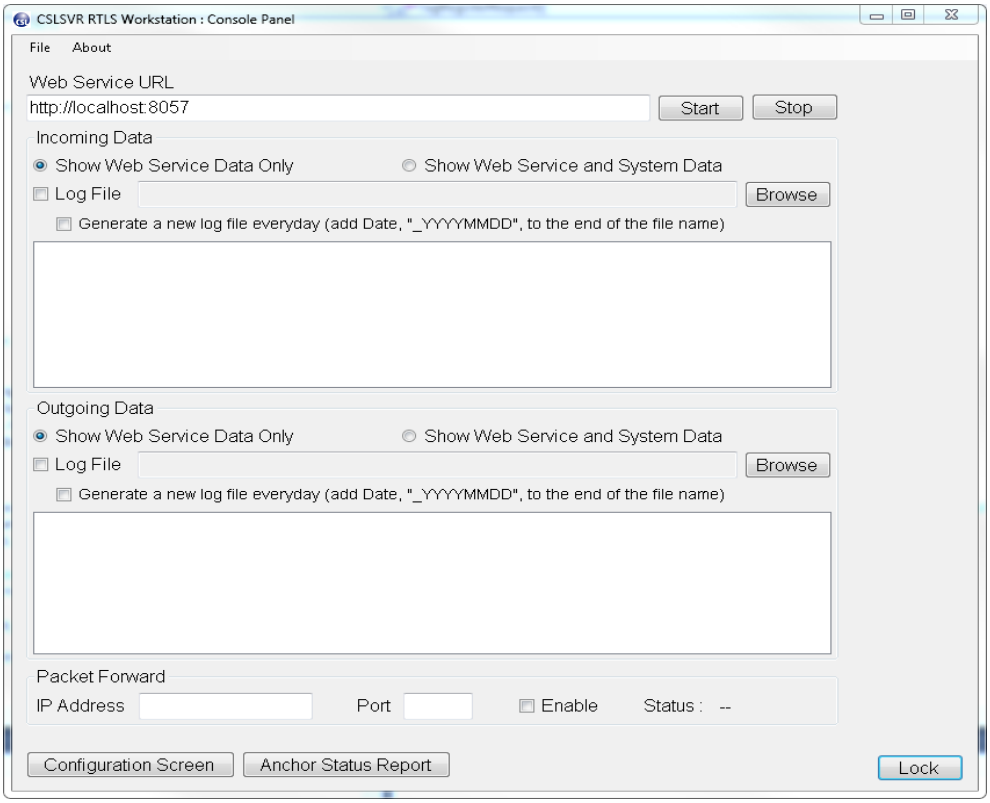

Figure 3.12: Interface for the location engine server (RTLS middleware)

engine server to be observed on the iPad interface (Figure 3.13). The number of tags, their locations and orientation are shown on the iPad screen using the camera of the iPad. The reported location of the tags is based on the orientation and direction of iPad as shown in Figure 3.14. One of the tags must be attached to the user and specified on the interface of the iPad (Figure 3.13); this tag serves as the reference tag aiding the iPad to understand the location of the user. Once the user selects any of the tags shown on the iPad's interface, the distance between the tagged component and the user's current location is shown as in Figure 3.14. This enables the user locate any selected tag within the building or observation area.

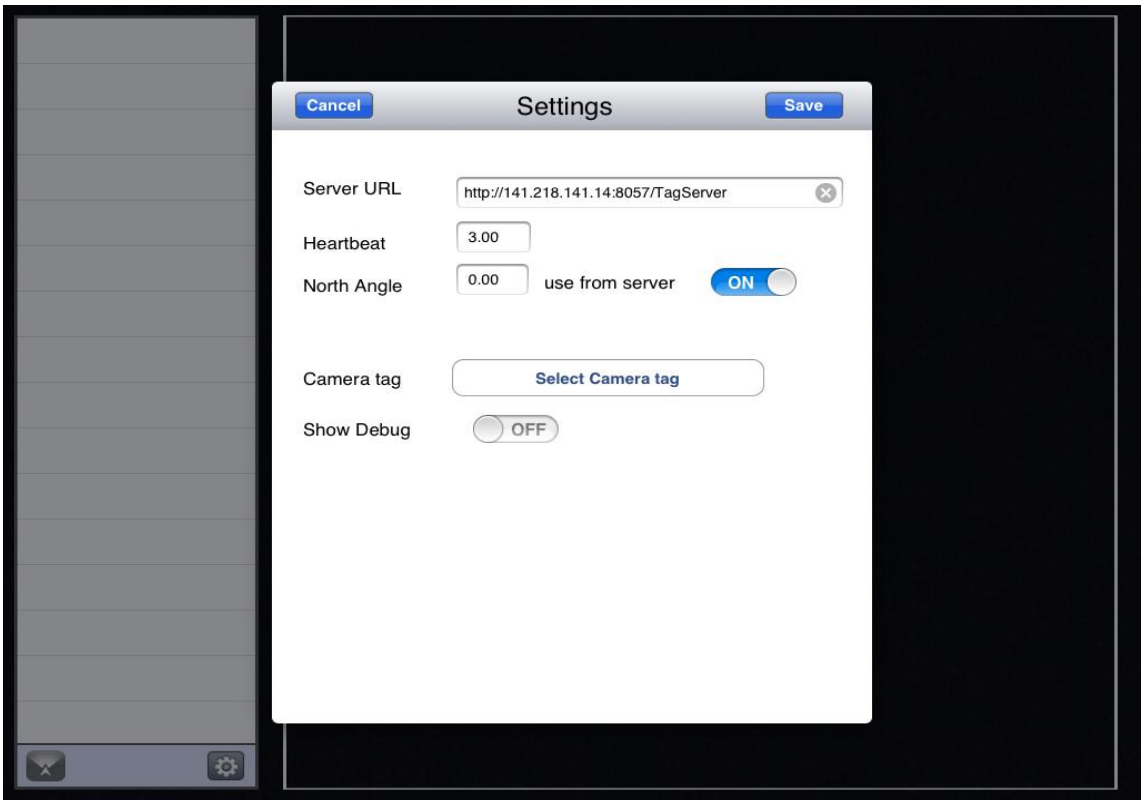

Figure 3.13: iPad configuration screen while associating the iPad to the Location Engine Server.

Figure 3.14: iPad interface showing 2D Map of lab (left) and distance of tracked asset (right)

3.6.2.3 Software Architecture

The software architecture for this prototype is shown in Figure 3.15. The software architecture is integrated via the Location Engine Server (LES) (installed on the workstation). The RFID-RTLS master anchor communicates tag information, such as the Tag ID, ranges from each anchor, Received Signal Strength Indicator (RSSI) and the Date and Time when the tag reported the last position, to the LES. The LES processes the tag information and determines the position of the tags based on the information relayed by the RFID-RTLS system.

Based on the location of the tag associated with the iPad, LES uploads all the Tag IDs, their position relative to the iPad and the distance of the tags from the iPad. This information is displayed on the iPad interface. The location improvement algorithm can also be embedded in LES to help improve the accuracy of tags being tracked.

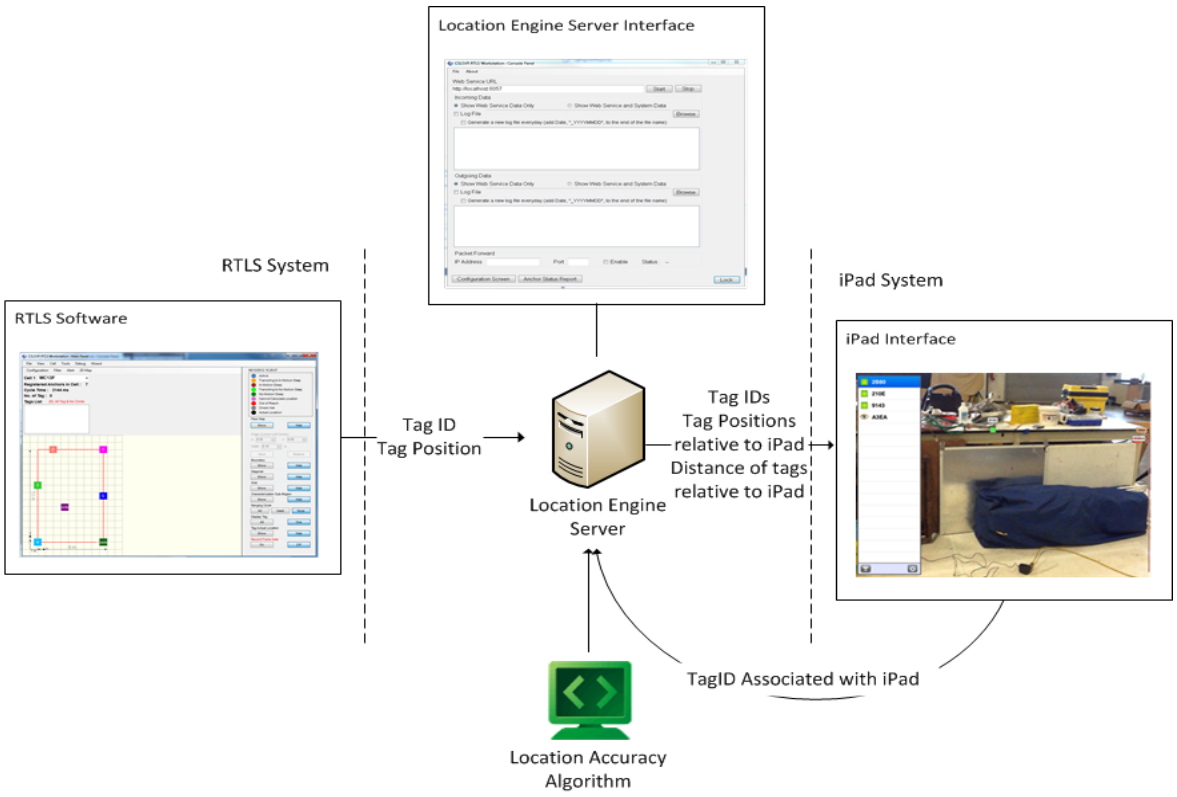

Figure 3.15: Brief software architecture for asset tracking prototype.

• Location Improvement Algorithm

For improving the accuracy of the reported location, a correction factor method is developed in this study. The correction factor method is based on the LANDMARC approach as discussed in Chapter 2. The developed approach is split into two steps. The first step is to calculate the correction factor of the reference tag (tags with known locations) based on the actual known position and the reported position by the system. The second step includes the application of correction factors on the tracking tags (the tags with unknown location) based on the nearest neighbor reference tags.

Similar to the LANDMARC approach, the reference tags are dispersed in the area of interest at fixed intervals. For each reference tag, the reported and actual (physical) locations (X and Y coordinate) were captured. The reported locations were filtered using the Kalman Filter to reduce the noise in the reported location data. The average of both the filtered x-location ($\bar{x}$) and filtered y-location ($\bar{y}$) is computed and the correction factors for each axis is determined using,

$$CF_z = \frac{\bar{z}}{Actual\ z\text{−coordinate}} \tag{3.6}$$

Where, *z* is either x-coordinate or y-coordinate.

This process is repeated on every reference tag and two correction factors (for x and y) for each tag is determined.

For each tracking tag, the Euclidean distance (localization error) of the tracking tag to each reference tag is calculated using,

$$LE = \sqrt{\sum(\text{Reported } z \text{ of Reference Tag} - \text{Filtered } z \text{ of Tracking Tag})^2} \tag{3.7}$$

Where, *z* is the x-coordinate and y-coordinate.

Nearest neighbor reference tags are selected based on the determined LE and a presumed threshold within which the tracking tag is assumed to be near the reference tag. If there are *i*=1,2,…,n nearest neighbor tags close to a tracking tag *k*, weights for the *ith* nearest neighbor tag is computed using,

$$W_i = \frac{1/LE_i^2}{\sum_{i=1}^{n} 1/LE_i^2} \tag{3.8}$$

The corrected location of the tracking tag *k* is then determined using,

$$z - coordinate\ of\ tracking\ tag = \sum_{i=1}^{n} CF_i \times Reported - z\ of\ tracking\ tag \tag{3.9}$$

Where *z* is either x or y-axis.

3.6.2.4 Use-case analysis

Figure 3.16 describes how a project manager or site supervisor will track the tagged asset on a mobile tablet.

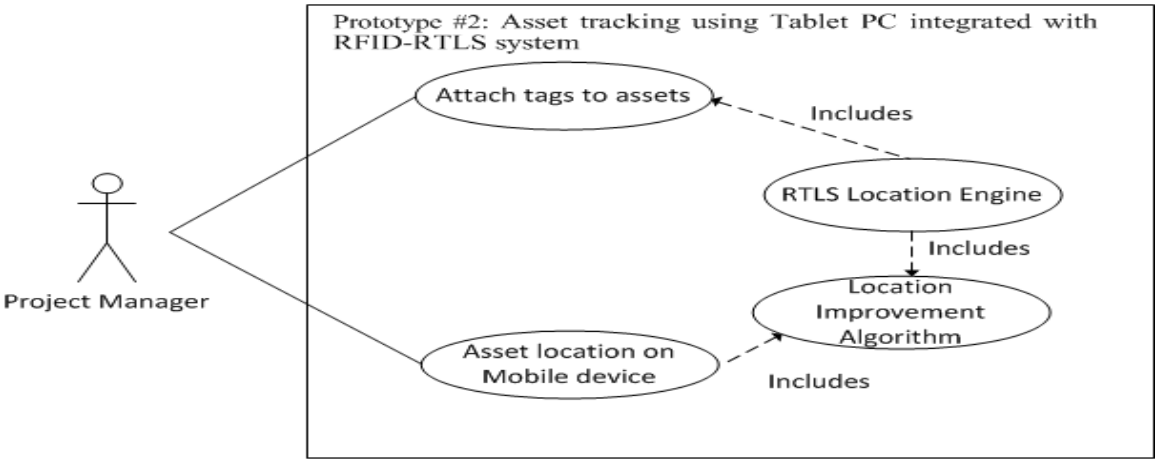

Figure 3.16: Use-case for the asset tracking prototype.

Primary Actor: Project Manager / Site Supervisor

Short Description: This use-case illustrates how the facility manager will track tagged assets using the integrated TRFID-RTLS system.

Preconditions:

- RFID-RTLS system must be deployed on the site and connected to a LES.
- All the tags must be powered on.

Basic Flow:

- Mobile device is connected to the workstation running the location engine server using the Universal Resource Locator (URL) address pointing towards the server.
- Camera tag (tag associated with the mobile device) is selected within the interface of the application installed on the mobile device (Figure 3.13).
- Once the mobile device is connected, the location of the assets associated with tags can be tracked.
- The project manager can select the tag from the interface to see the direction and bearing of that asset from the location of mobile device.

3.6.2.5 Case Study for Asset tracking Prototype

To access the accuracy of the developed system, several tests were carried out the College of Engineering and Applied Sciences, Western Michigan University. Two laboratories (the CPC laboratory and the Plastics lab) were selected to conduct experiments because both of labs had different item density per square foot of the laboratory.

- Cyber Physical Construction (CPC) Laboratory

The RTLS system anchors were deployed at the perimeter of the laboratory. A total of seven anchors were used for the test. A 2D map of the anchor setup is shown in Figure 3.18. Six anchors were setup on the perimeter of the laboratory (1, 2, 3, 4, 5, and Master anchor as shown in Figure 3.17). Anchors setup on the perimeter can only establish the line-of-sight connection with the tags when they are within the area of influence (these are marked by red circles in Figure 3.17). The seventh anchor was setup in the middle (as shown in Figure 3.17 and Figure 3.18) because that anchor was Omni-directional meaning that it can establish the line-of-sight connection with the tags in a circular radius given there is no obstruction between the tag and the anchor. However, the anchors require the line-of-sight connection to the tags for the determining the location of the tags.

Three tools (a hammer, drill machine and a grinder) were tracked as a part of this experiment. These tools were chosen because they are easily misplaced. The tools were placed in random locations before the start of the experiment and were kept at the same location during the experiment to log data. The logged data consisted of the tag IDs, reported location by the system (x and y coordinate) with respect to an origin (in this case Anchor 4 as in Figure 3.17), reported range from each anchor to the asset tag, received signal strength indicator (RSSI) of the tag and date and time of reporting. The actual locations of the tags from the origin (Anchor 4) were also determined during the experiment. Euclidean distance (localization error) was calculated based on the reported and actual values to quantify the accuracy of the system.

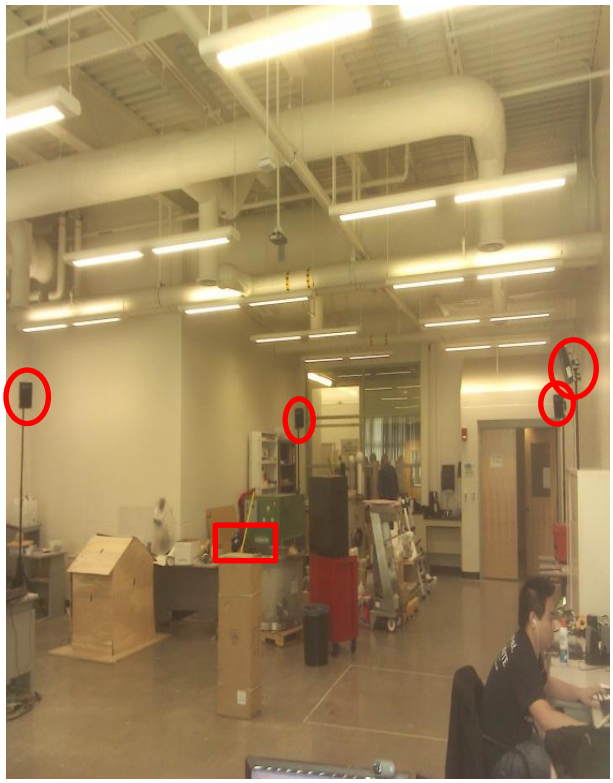

Figure 3.17: RFID-RTLS anchor setup in CPC Laboratory

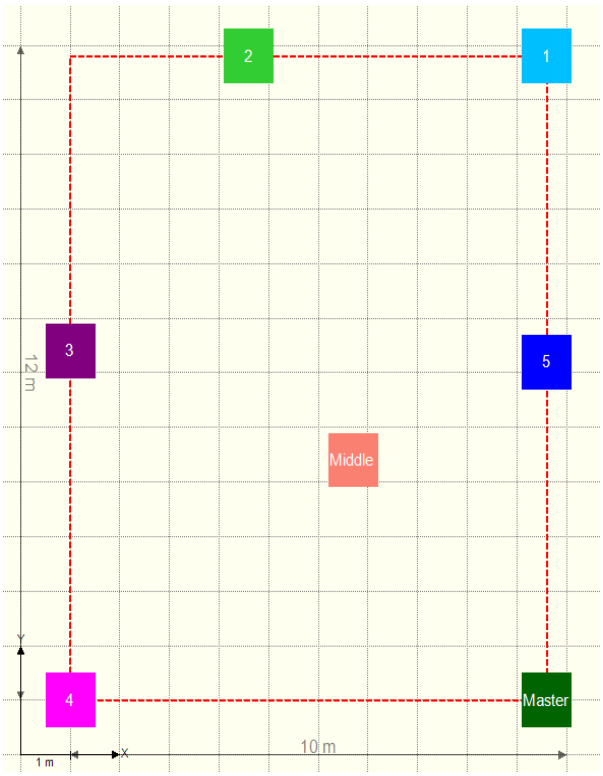

Figure 3.18: Plan of RFID-RTLS anchor setup for CPC Laboratory

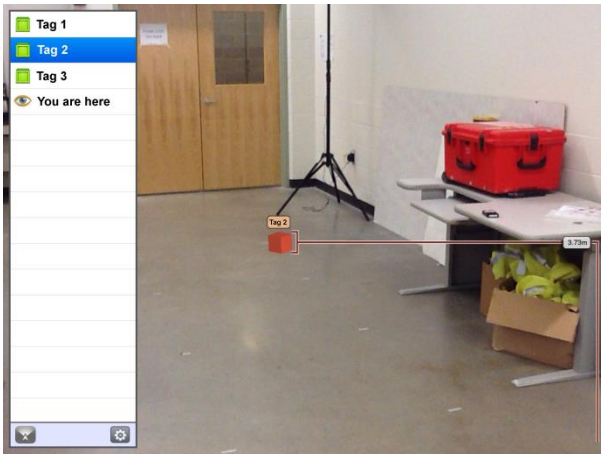

Figure 3.19: Heads-up display view displayed on iPad.

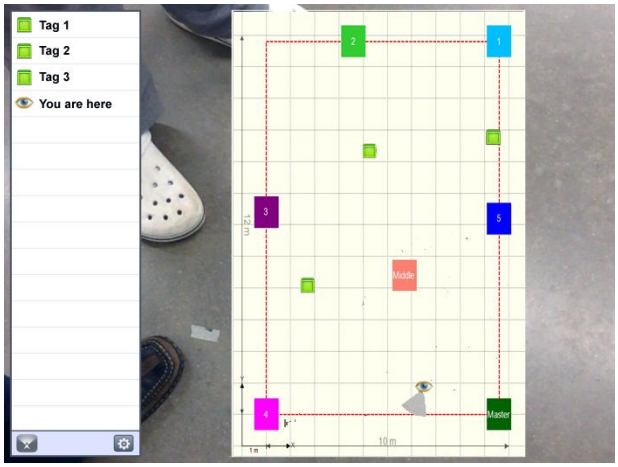

Figure 3.20: 2D Map displayed on the iPad.

- Plastics laboratory

The Plastics laboratory had more bulk items compared with the CPC laboratory (Figure 3.21). Implying that the item density per square foot of this lab is relatively higher than that of compared with the CPC laboratory. A total of seven anchors were also deployed in the laboratory (Figure 3.21). Six anchors were deployed on the perimeter of the area under observation (1, 2, 3, 4, 5, and Master as in Figure 3.21) and the Omni-directional anchor was placed in the center of the observation area (Middle in Figure 3.22). The anchors are marked in Figure 3.21 using circles and the Omni-directional anchor is marked by a square.

Three tools (a wire plier, and two small solar car assemblies) were tracked as part of this experiment. Before the start of the experiment, these tools were randomly placed in different locations and subsequently kept at the same location during the experiment to log data. Similar to the CPC laboratory, the parameter recorded from this experiment were same as the previous experiment and included tag IDs, reported location by the system (x and y coordinate) with respect to an origin (in this case Master as in Figure 3.22), reported range from each anchor to the asset tag, received signal strength indicator (RSSI) and date and time of reporting**.**

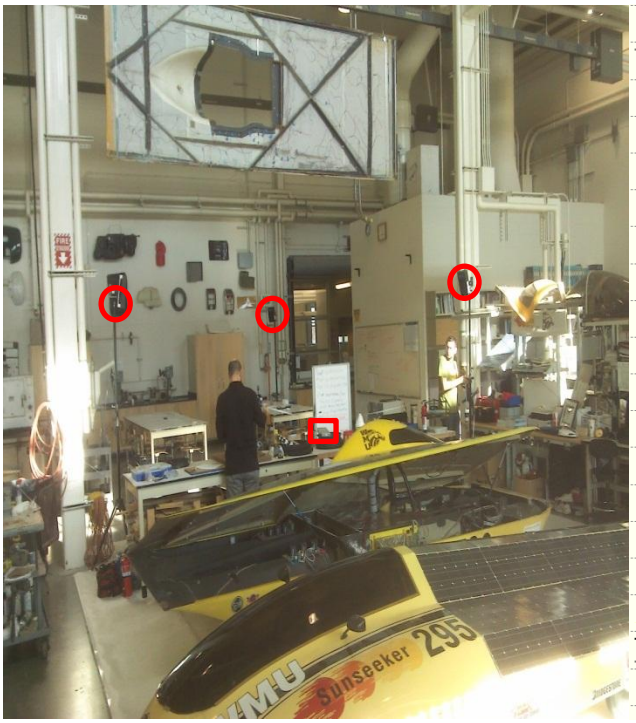

Figure 3.21: RFID-RTLS anchor setup in the Plastics lab.

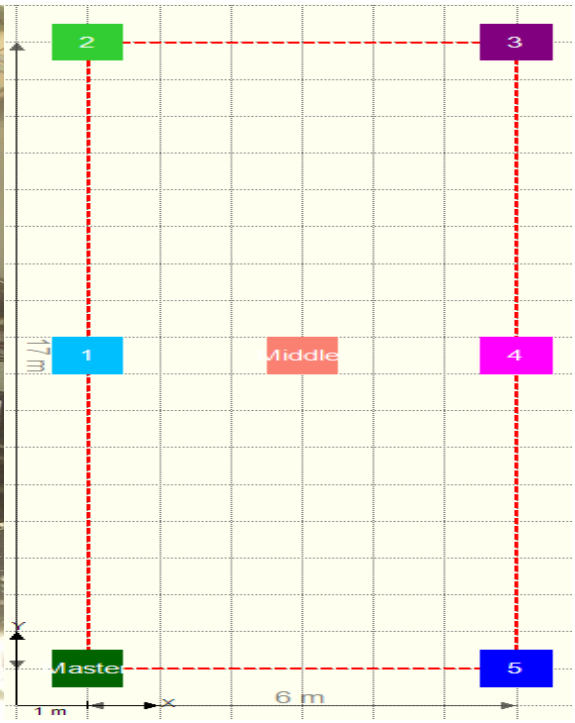

Figure 3.22: Plan of RFID-RTLS anchor setup in the Plastics lab.

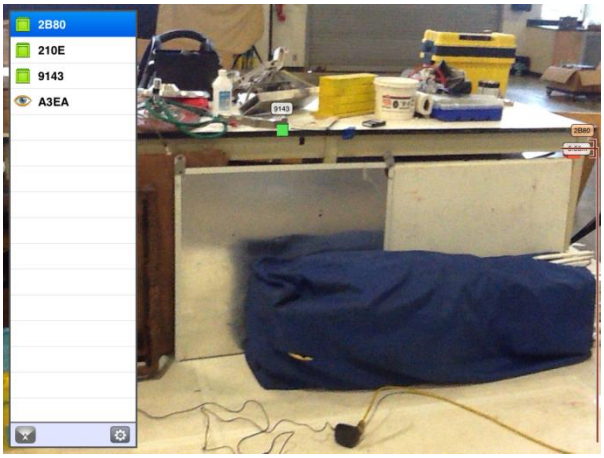

Figure 3.23: Heads-up display view displayed on iPad

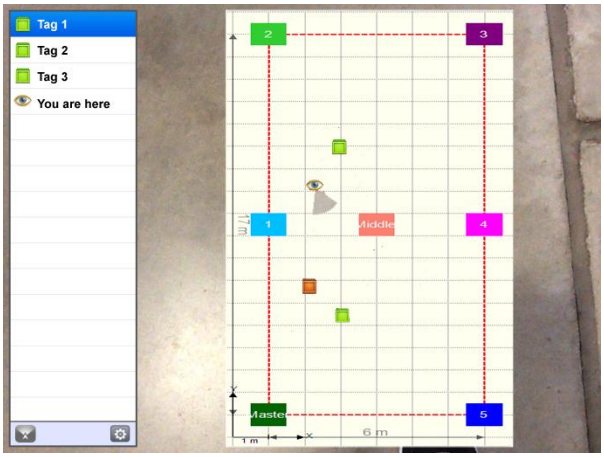

Figure 3.24: 2D Map of the test site displayed on the iPad

3.6.2.6 Tests for correction factors

To improve the accuracy of the location data reported by the system, two separate experiments were performed in the test site. As discussed in the Section 3.6.2.5 the tags were dispersed at a fixed interval of 2m (Figure 3.25 and Figure 3.26) throughout the test sites. The actual and reported locations were recorded. The test results were also compared with other methods such as the LANDMARC approach and Virtual tag approach.

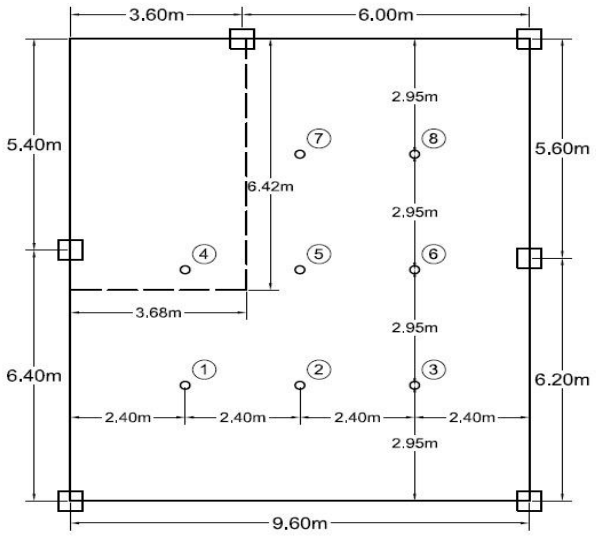

Figure 3.25: Reference tag setup plan for the CPC Lab.

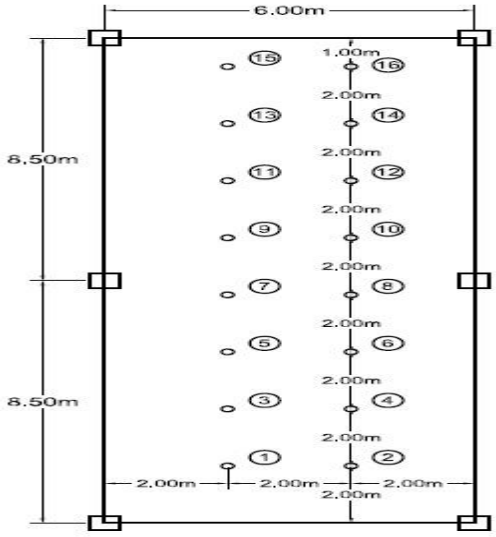

Figure 3.26: Reference tag setup plan for the Plastic lab.

3.6.3 Prototype #3: Automated Construction Progress Control based on real-time Earned Value Analysis

Schedule, cost and quality are three major indicators for the construction project control as discussed in Chapter 2. Recent studies have suggested that the earned value

analysis is a good indicator of construction progress but little progress has been made towards automating progress control using the earned value management. Frameworks developed in recent studies have been limited to manual scanning of tags to update the schedule and the earned value. Such studies are also limited to particular resource type (either material or equipment or both but not labor) to determine the earned value of project, leaving prospects to improve on the current work.

3.6.3.1 System Architecture

To develop an automated system for earned value management, a system architecture has been developed which involves the integration of RFID-RTLS system, Building Information Model (BIM), and Database (DB) using Visual Studio .NET 2012 (C#). The system architecture shown in Figure 3.27 is described as follows: Tagged resources, such as material, labor or equipment are ranged by the slave anchors. The range data along with the received signal strength indicators (RSSI) is sent wirelessly to the master anchor by the slave anchors. In addition to ranging the tags and determining the RSSI, the master anchor relays the information of the slave anchors to the LES. The range data is sorted based on the RSSI of the tag. The range data with the highest RSSI value is given the most priority. Eventually, at most seven sorted range values are chosen for computing the location of the tag. The computed tag locations are forwarded to middleware.

Two middleware were developed within the RFID-RTLS Location Engine Server (LES) and Autodesk Navisworks namely; RTLS-DB-EVM and EVMPanel. Both middleware were programmed using Visual Studio 2010 (C#). EVMPanel was developed

as a plugin for Autodesk Navisworks using the software's .NET API Both middleware were connected to a database which acted as a bridge to exchange information between the RFID-RTLS and BIM.

EVMPanel (For code, refer to APPENDIX D – EVMPanel) captures the user defined resource type, cost, and activity assigned to a particular tag from a custom graphical user interface (GUI). This plugin also extracts the schedule details (such as planned, actual start and end dates for the activities), the list and planned cost of activities to be performed. All of the data is collected in the database; where the RTLS-DB picks up and processes the cost data.

RTLS-DB-EVM (For code, refer to APPENDIX C – RTLS-DB-EVM Code) checks for the current activity (by comparing the actual start and current date of each activity in the database) and lists all the defined resources associated with that activity. EVMPanel computes the actual cost of resources based on the location of the attached tag. The EVMPanel computes the cost data as follows:

- If the resource type is material, the actual cost is computed as the material is tracked to its installed location. The cost per unit quantity / time defined in the DB and will be the actual cost incurred based on the resource type (material, labor/equipment).

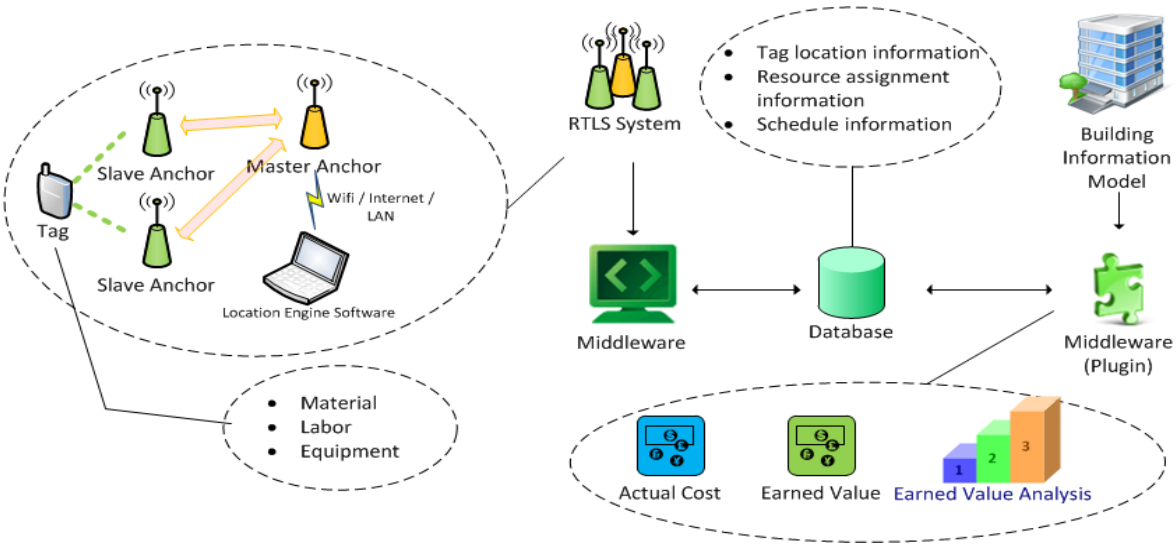

Figure 3.27: System Architecture for the developed EVM prototype.

- If the resource type is labor, it is assumed that the labor is working when they are within the vicinity of the installation zone (known as the 'Labor' zone). RTLS-DB-EVM computes the total time spent by the labor tag, when it is within the 'Labor' zone. The actual cost is calculated when the labor moves out from the work zone by multiplying the time spent by the unit cost defined in the database by Equation 3.10.

$$\text{Actual Cost (labor)} = \text{Time spent} \times \text{Unit Price} \quad (3.10)$$

- For the equipment, it is assumed that the equipment will be working when the equipment moves between the storage and installation location of components. RTLS-DB-EVM computes the total time spent by the equipment as it moves between the 'Storage' zone and the 'Installation' zone. The total time is used to calculate the actual cost of the equipment. The actual cost is calculated in a similar way as the labor by multiplying the time by the unit cost defined in the database by Equation 3.11.

$$\text{Actual Cost (labor)} = \text{Time spent} \times \text{Unit Price} \quad (3.11)$$

The planned value (budgeted cost of work scheduled) is based on the planned activities and their associated costs. EVMPanel determines these planned values from the planned cost associated with the scheduled activities specified in the Timeliner in Autodesk Navisworks. The earned value (budgeted cost of work performed) is based on the amount of work performed at a particular instance. The earned value is also captured in the EVMPanel and is specified by the user using the GUI.

After the computing the actual cost, the RTLS-DB-EVM stores the data and the timing information in the database. EVMPanel extracts the cost information (actual cost) and the values (both planned and earned) from the database and presents it visually in real-time on a graph via a custom GUI developed within the plugin (Figure 3.28). This figure shows the Actual cost in addition to the planned and earned values. It also indicates if the project is on-schedule and on-budget based on the earned value parameters which include cost variance percent and schedule variance percent.

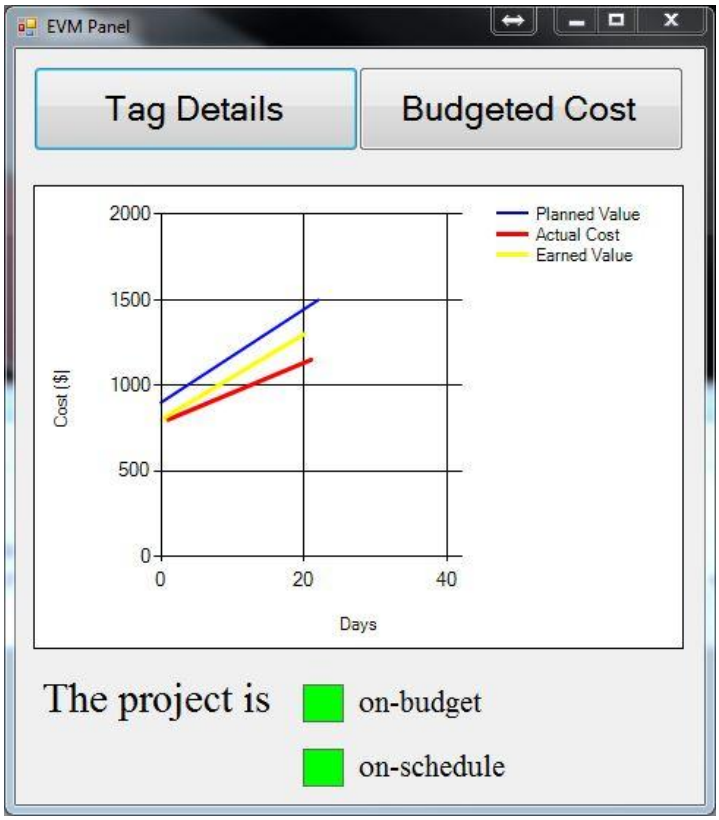

Figure 3.28: Graphical User Interface showing progress using graph in EVMPanel

3.6.3.2 Software Architecture

The software architecture (Figure 3.30) for the developed prototypes is distributed over two middleware. Both middleware were connected to a centralized database which acts as a bridge to relay information between both RFID-RTLS software and Autodesk Navisworks. The database server (MySQL 5.6.10) was installed on the workstation and a database schema was setup. Four database tables were modeled to capture information namely: accuracy_table', 'taginfo', 'activitydetails' and 'tagdetail'. The accuracy_table' stores the Tag IDs, RSSIs, reported ranges from each anchor to the tag and reported location from the system. . The 'taginfo' stores the TagIDs, last known zone within

which the tag was reported , the date and time at which the record was added. The 'activitydetails' table stores the Activity ID and their associated actual start and end dates. The 'tagdetail' stores the resource type, resource cost, activity, and description. This information is assigned to a particular Tag ID which is also defined in this table.

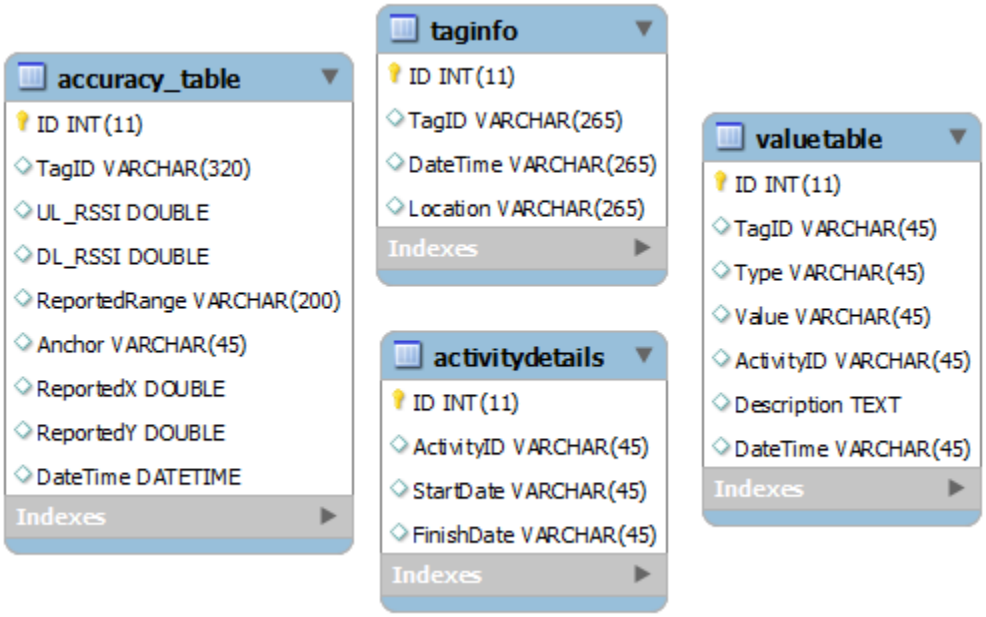

Figure 3.29: Database table structure for earned value prototype

RTLS-DB-EVM was developed to capture location information and all associated data (such as tag IDs, reported locations) from the RFID-RTLS location engine software. This RTLS-DB also served as a terminal for the user to input the zoning information (such as the name, size and location) into the system. RTLS-DB-EVM also checked for the current activity stored in the 'activitydetails' database table by comparing the current dates and the actual dates (start and end) of the activities. Actual cost was determined by the middleware based on the location of resource tags associated with the current activity

as discussed in the System architecture. All the information from the middleware was stored in the database. EVMPanel was developed as a plugin for Autodesk Navisworks Manage 2013 using .NET Application Programmability Interface (API). This middleware was also connected to the database. The middleware uses the timer function to scan the database table 'taginfo' for tags information. As soon as the tag ID is populated in the database, the plugin launches a search query within the Navisworks for a 3D component with matching ID. Based on the zone reported in the database, the plugin automatically updates the color of the searched 3D component. If the reported zone is empty (on-site) the component color is changed to red, if reported zone is 'Storage', color changes to white and if it is 'Installed' the color changes to green. Using the timer function, this plugin scans for changes in the 'valuetable' database table. Based on the associated activity, the plugin captures all the cost data, processes it and presents it in a graphical form. In addition to extracting the information from the database, this plugin also stores information in the database. This information includes schedule details (Activity ID, actual start and end dates) in the 'activitydetail' database table. Through a custom GUI, user defined resource type, resource cost, and activity assigned to a particular tag can be captured and stored in the 'tagdetail' database table. (For code, refer to APPENDIX).

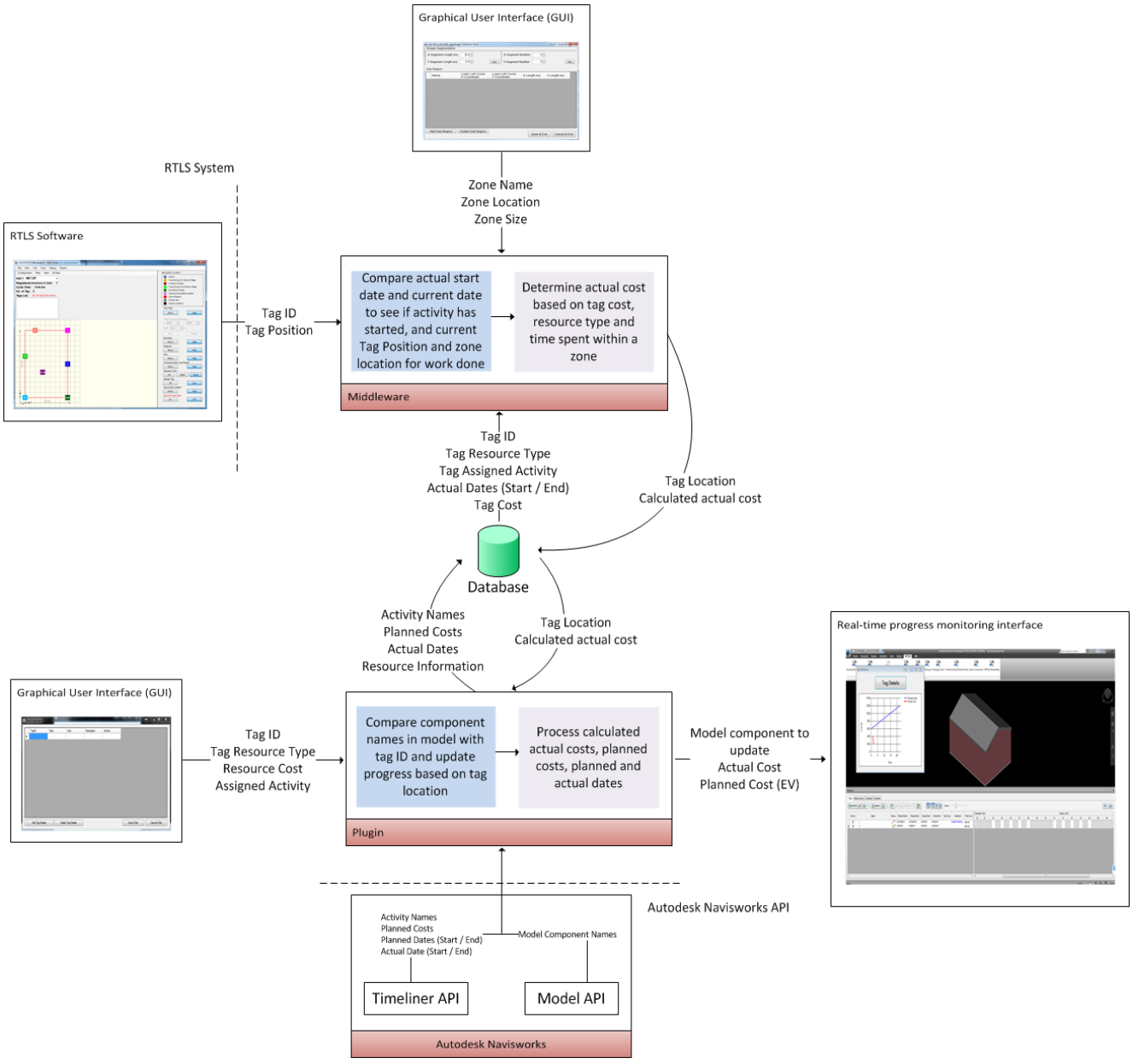

Figure 3.30: Brief software architecture for the project control prototype.

3.6.3.3 Use case analysis

This use-case (Figure 3.31) describes how a project manager can track the progress of a construction project on an actual job-site based on the cost of resources (material, labor and equipment) on the jobsite.

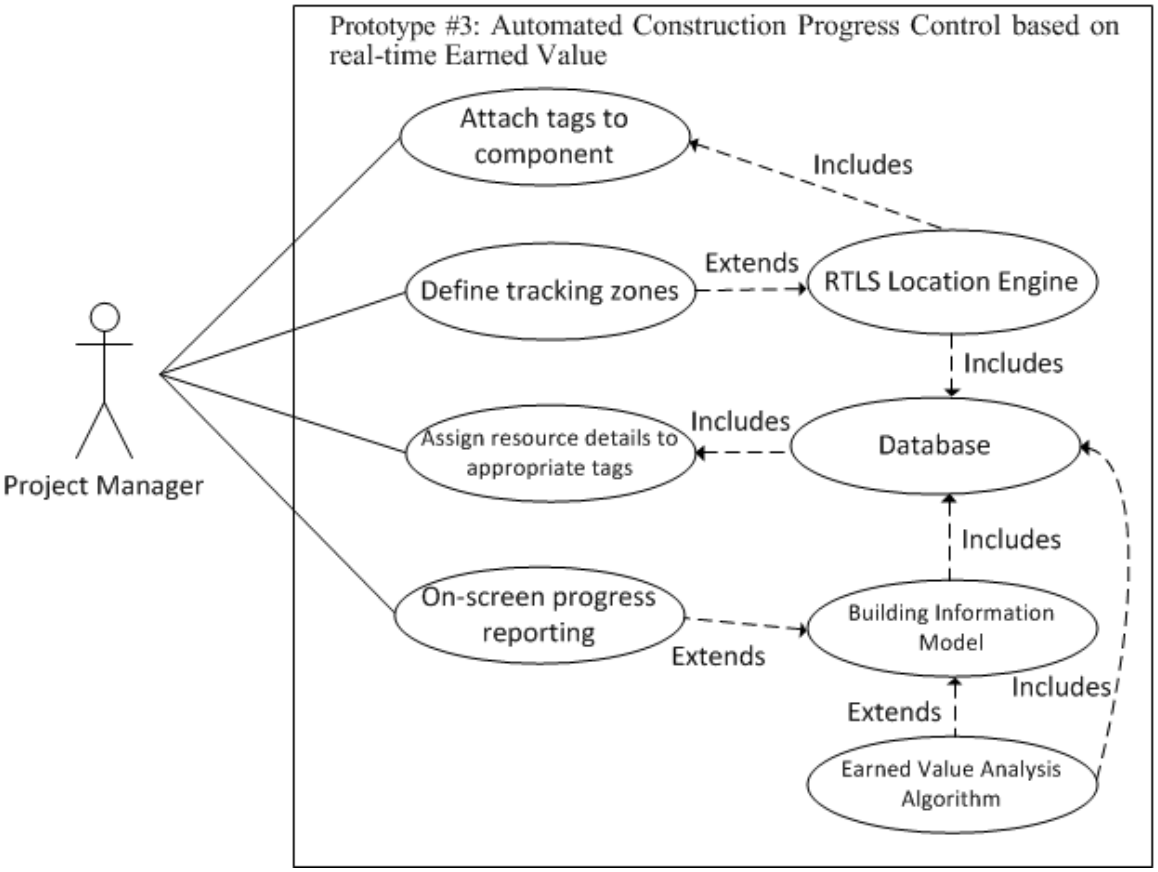

Figure 3.31: Use case analysis diagram for the developed third prototype.

Primary Actor: Project Manager

Short Description: The project manager tracks the construction progress based on the cost of tagged material, labor and equipment.

Preconditions:

- RTLS-RFID anchors must be setup on the perimeter of the area of interest.
- All the tags must be powered on.

Basic Flow:

- The Project Manager must make sure that the appropriate resource (material, labor and/or equipment) are tagged.
- The process starts when the Project Manager accesses the EVMPanel in Autodesk Navisworks and defines the resource type, resource cost, activity assigned to a particular Tag ID under the Tag Assignment interface.
- The Project Manager must also define the actual start and end date for the activities
- The Project Manager then launches the RFID-RTLS software, defines zones under the Observation Area in the software window. There can be three possible zones, 'On-site', 'Storage' and 'Installed'.
- Once the zones appear on the interface the Project Manager must click 'Start / Stop' on the EVMPanel.
- System will return the cost data based on the information provided.

Post-conditions:

- The 3D model is updated with different colors based on the status of the component.
- A graph of the actual and planned cost is displayed.

3.7 Summary

This chapter has described the development of three prototype systems namely; automated component tracking on the job site, asset tracking within indoor environments and earned value analysis. The automated component tracking system demonstrates how the status of tagged components can be automatically captured and visualized in the model based on the context of the tag. This was demonstrated by in the CPC laboratory at Western Michigan University by integrating RFID-RTLS system with BIM. The RFID-RTLS tags can identify and distinguish components. The technology has shown tremendous potential for real-time location tracking of tagged components; this makes the technology suitable for tracking and monitoring the installation of components during construction. This adaptive system has potential opportunities for improving progress monitoring and control of construction activities. The second developed prototype was also used to illustrate the concept of cyber-adaptive systems and its potential for asset tracking. This system enables automated tracking of tagged assets through the integration of RFID-RTLS system with i-Pad. The system provides the location of tagged components through the GUI of the iPad based on the context of the user. With this system, facility managers can easily navigate to tagged components based on their context.

Cyber-adaptive physical systems have also been demonstrated using RFID-RTLS system for integrating virtual models and physical laboratory scale prototype for earned value management. This prototype offers tremendous opportunities for real-time progress

tracking within spatially mapped activity zones or spaces. The next chapter presents the results of the implementation of two of the developed systems.

CHAPTER 4

CASE-STUDY AND RESULTS

4.1 Introduction

Two of the developed prototypes (Prototype #1 and #2) for automated construction progress monitoring and asset tracking were implemented and the results are discussed in the concluding section of this chapter.

4.2 Prototype #1: Automated Component Status Tracking using the RFID-RTLS System

The results from Prototype #1 include the ability of the prototype to automatically capture and update the model with RFID-RTLS tag status information and test results (to determine the accuracy of the developed prototype for automated model update).

4.2.1 Developed Prototype System

The developed prototype system was clearly able to capture the status of the tagged components and update the model. Whenever the location of the tagged components changed, the colors of the corresponding virtual components were automatically updated. For example, the color of the virtual component changed to red when the corresponding physical component was on-site, changed to white when the corresponding physical component was in the storage area, and finally changed to green when the corresponding physical component was installed. Figure 4.1 shows the update in virtual component color as a result of the components' status change.

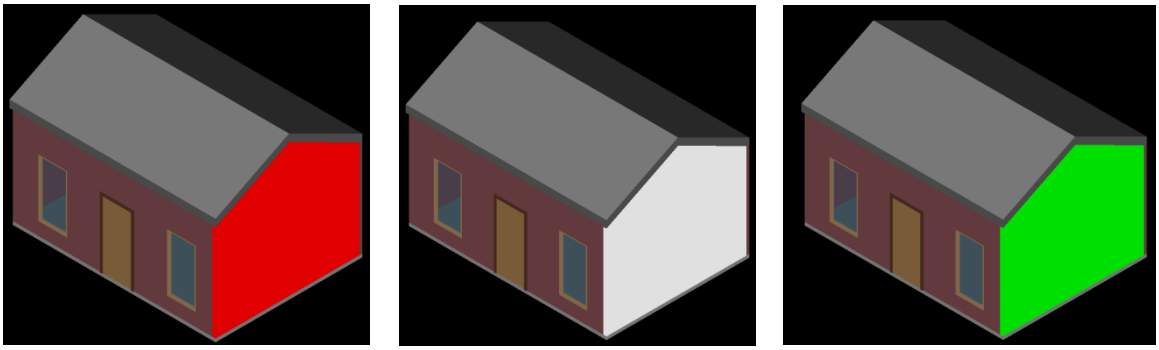

Figure 4.1: BIM alerting that the wall component is on-site (left), stored (center), installed (right)- use the interface

4.2.2 Test Results

It is important to have a stable location data so that the status update to the virtual model is stable. Hence, the aim of the test was to determine the accuracy of the RFID-RTLS system for tracking components within spatially mapped locations and updating the virtual model with the status information. For both the indoor and outdoor tests, the actual and estimated locations of the tagged components were captured. The accuracy of the RFID-RTLS system was determined by computing the localization error for each tag. The localization error is the distance between the estimated location and the actual location. This was calculated in meters using both the estimated x and y coordinates of the tagged component (i.e., the tag) obtained from the RTLS, and the actual location of the tags (i.e., actual x and y coordinates).

$$\text{Localization error} = \sqrt{(\text{Actual x coordinate} - \text{Estimated x coordinate})^2 + (\text{Actual y coordinate} - \text{Estimated y coordinate})^2} \quad (4.1)$$

The estimated location of the tagged components within the mapped out locations was captured from the system for about 45-60 seconds translating to about 30 observations. Six tags were used for the experiment: the tags correspond to the number of components of the physical laboratory scale prototype. This section shows the localization errors for all of the tagged components i.e. when arrived on-site, in a storage area and when they were actually installed respectively.

4.2.2.1 On-site Area

As discussed in Chapter 3, the logical sequence for component tracking was implemented during the experiment. Phase 1 included bringing the material 'on-site'. Figure 4.2 shows the localization error for one of the tagged components (for the indoor and outdoor experiment) computed for 30 observations of reported locations (when the components were within the on-site mapped area). The localization error was higher indoors than outdoors due to multipath effects experienced with the laboratory. The outdoor localization error was less than 0.5m; this indicates that the tagged components are within the 'on-site' mapped area.

Similarly Figure 4.3, Figure 4.4, Figure 4.5, Figure 4.6, and Figure 4.7 also represent the localization errors of five different tags for both the indoor and outdoor experiments. The graphs suggest that localization errors for the outdoor results are less than 1m for Tag 1 (Figure 4.6), Tag 2 (Figure 4.3) and Tag 6 (Figure 4.7) and less than 0.5m for Tag 3, Tag 4 and Tag 5.

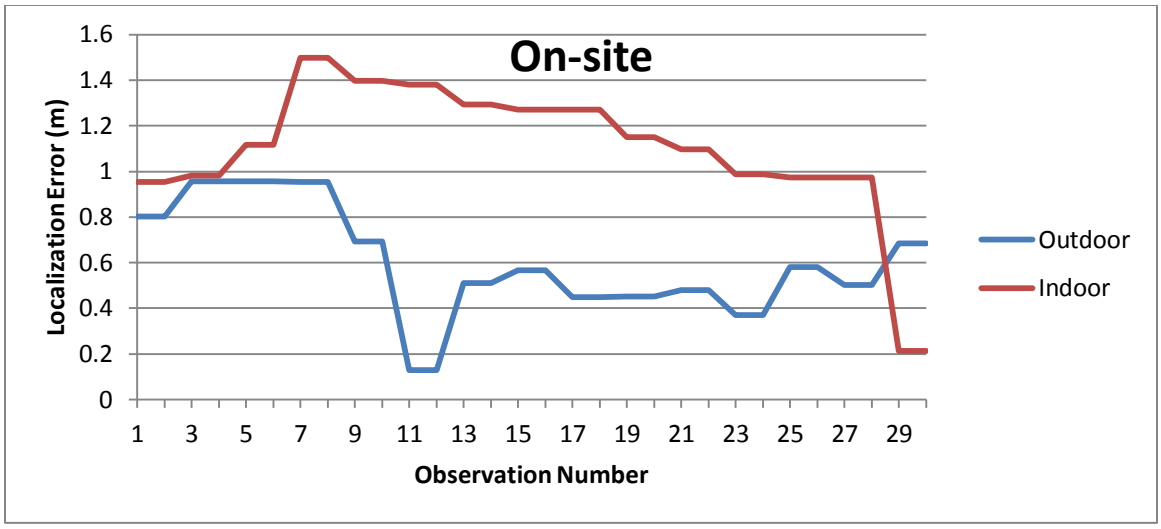

Figure 4.2: Tag 1 'On-site' localization errors for both indoor and outdoor observation.

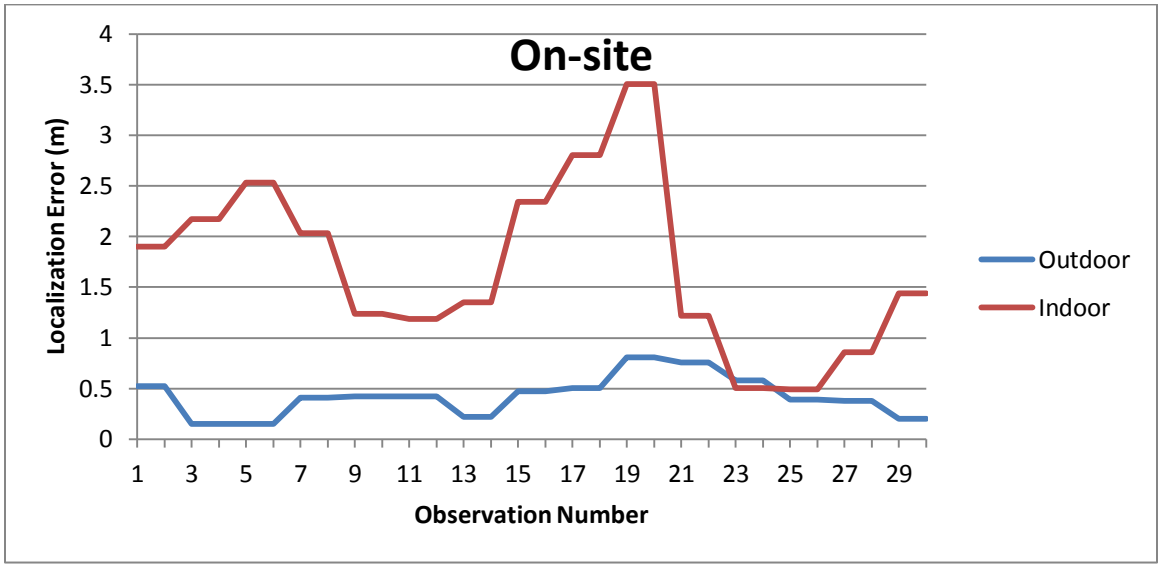

Figure 4.3: Tag 2 'On-site' localization errors for both indoor and outdoor observation.

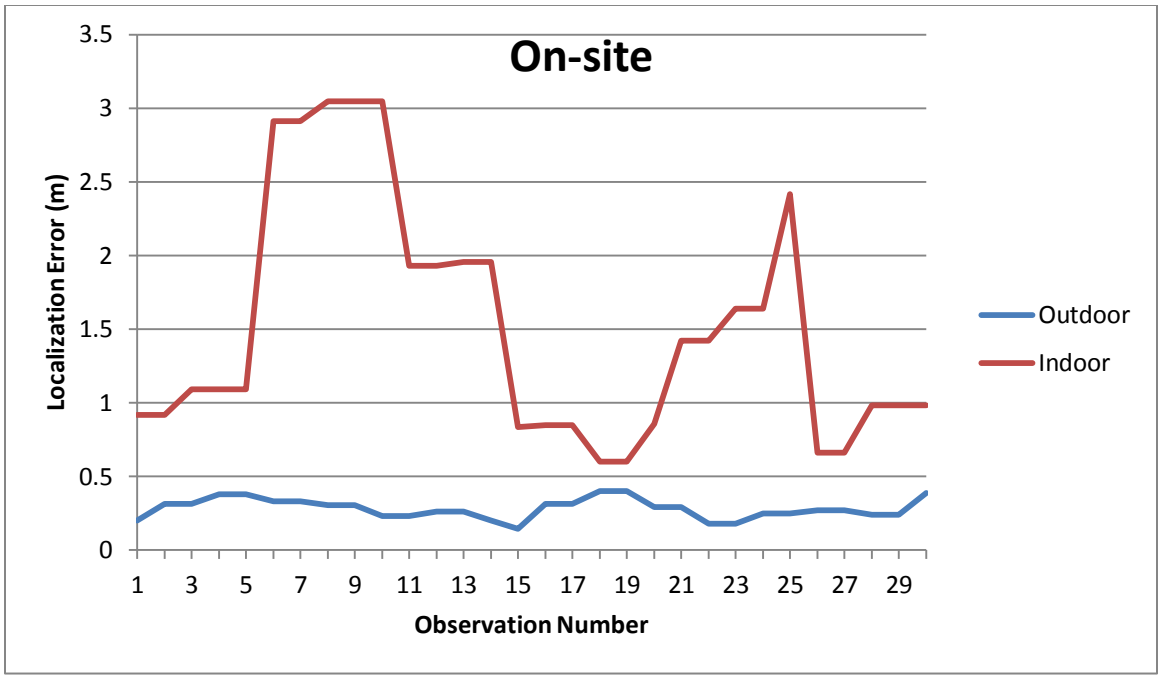

Figure 4.4: Tag 3 'On-site' localization errors for both indoor and outdoor observation.

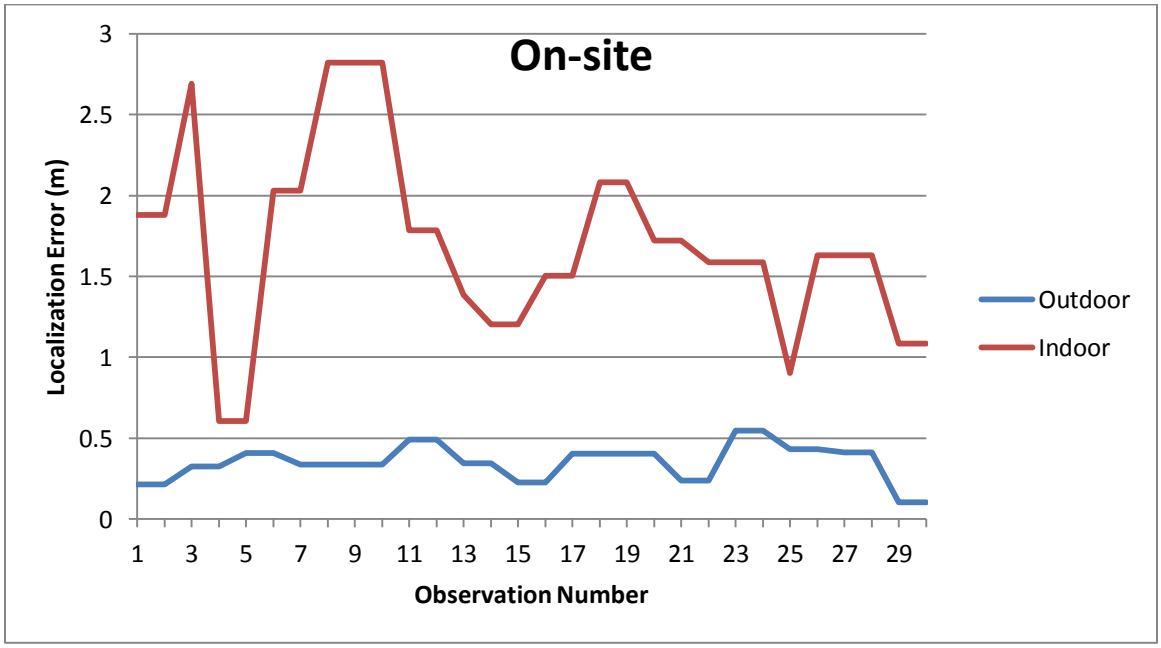

Figure 4.5: Tag 4 'On-site' localization errors for both indoor and outdoor observation.

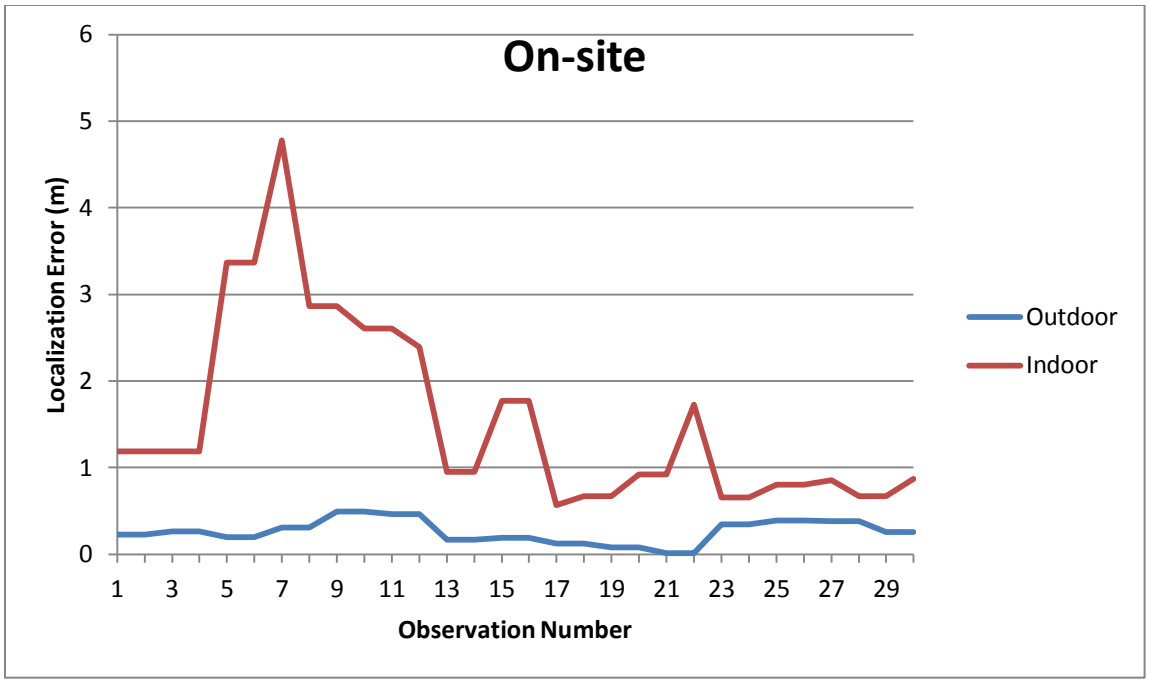

Figure 4.6: Tag 5 'On-site' localization errors for both indoor and outdoor observation.

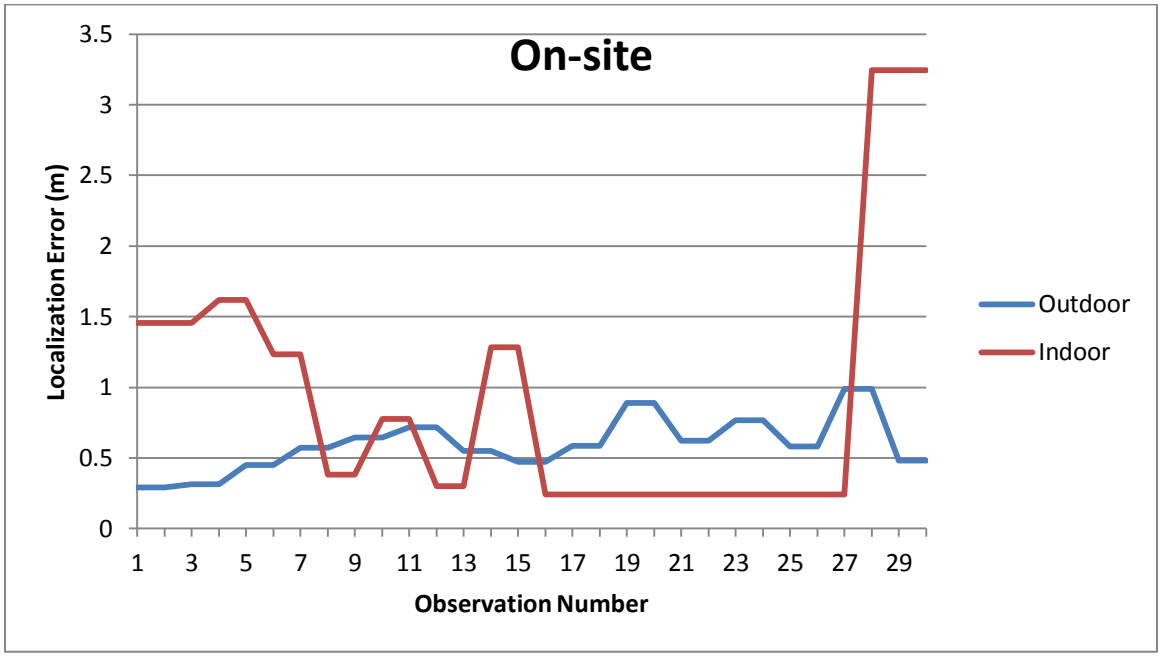

Figure 4.7: Tag 6 'On-site' localization errors for both indoor and outdoor observation.

4.2.2.2 Storage Area

Phase 2 of the logistical sequence implies that the component should be hauled to the storage. Figure 4.8 is the graphical representation of Tag 1 when it was in the storage area for both outdoor and indoor experiment. Localization errors were calculated for 30 observations of locations reported by the system. From the figure, it can be observed that the localization error for indoor is significantly higher (less than 4m) when compared to outdoor (less than 0.5m).

Similarly, Figure 4.9, Figure 4.10, Figure 4.11, Figure 4.12, and Figure 4.13 also represent the localization errors from five different tags when they were in the storage area. From these figures, it can be observed that the outdoor localization error is less than 0.5m for Tag 4 (Figure 4.11), Tag 5 (Figure 4.12) and Tag 6 (Figure 4.13), and less than 1.5m for Tag 1 (Figure 4.8), Tag 2 (Figure 4.9) and Tag 3 (Figure 4.10). Indoor localization errors reported were less than 2m for Tag 3 (Figure 4.10) and Tag 4 (Figure 4.11), less than 2.4m for Tag 1 (Figure 4.8), less than 5m for Tag 5 (Figure 4.12), less than 8m for Tag 2 (Figure 4.9). The goal of tracking the status of the components is to have the model updated when the components are within the mapped location. Thus, localization error observed during both Phase 1 and Phase 2 of the logistical sequence is acceptable. This provided a stable status update to the model.

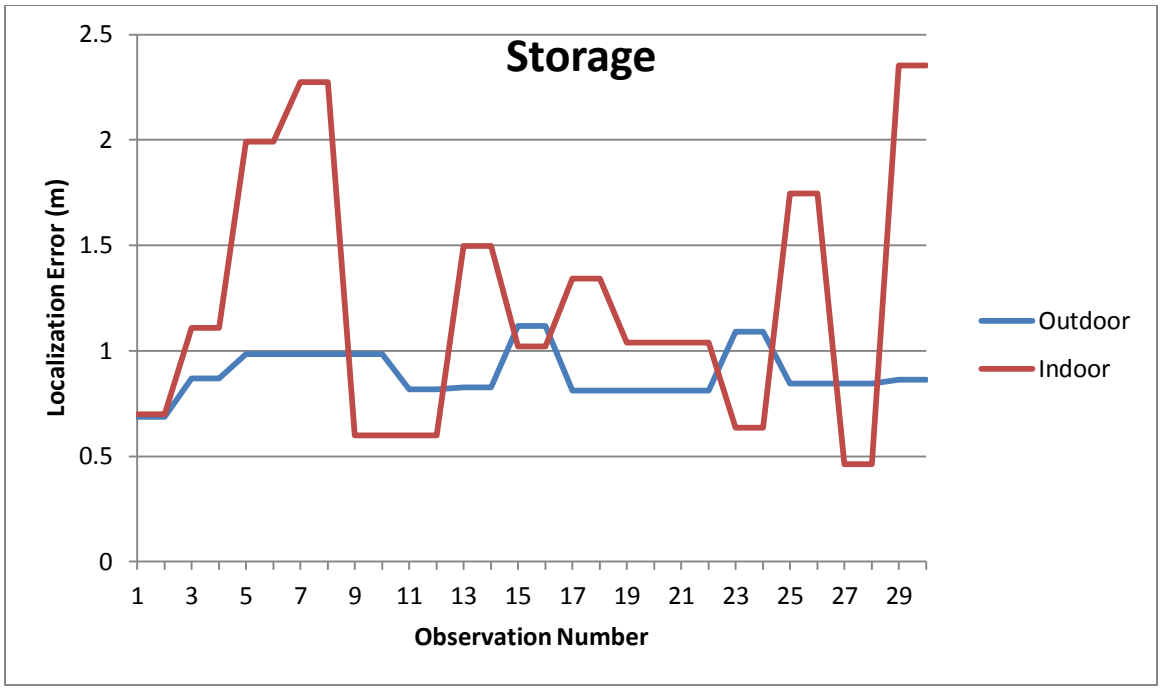

Figure 4.8: Tag 1 'Storage' localization errors for both indoor and outdoor observation.

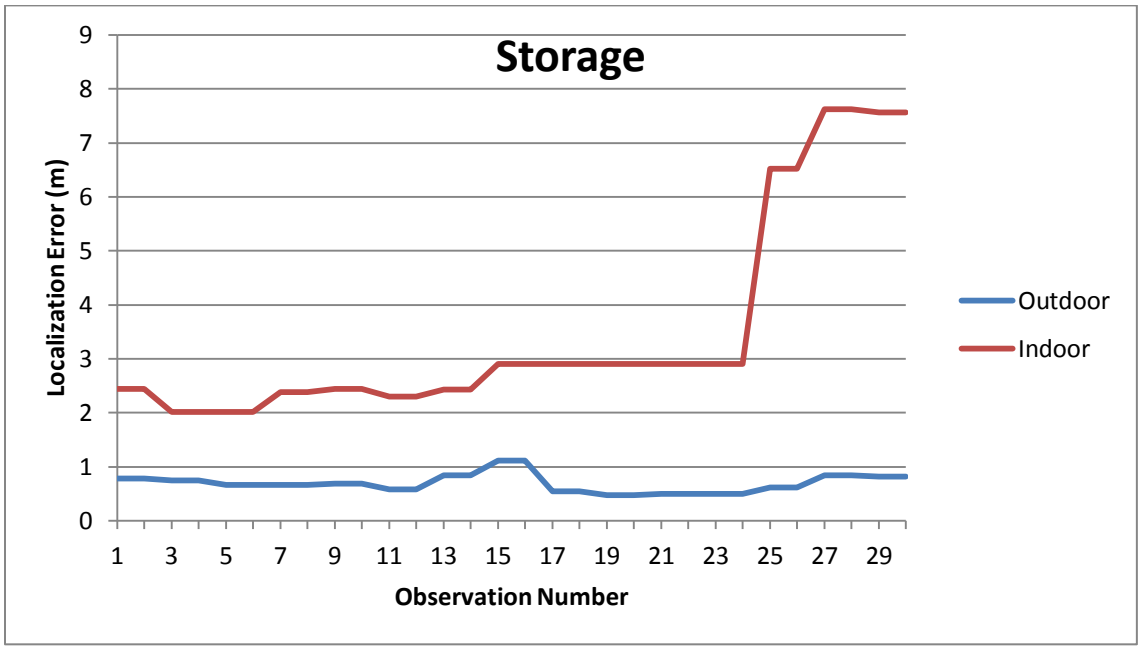

Figure 4.9: Tag 2 'Storage' localization errors for both indoor and outdoor observation.

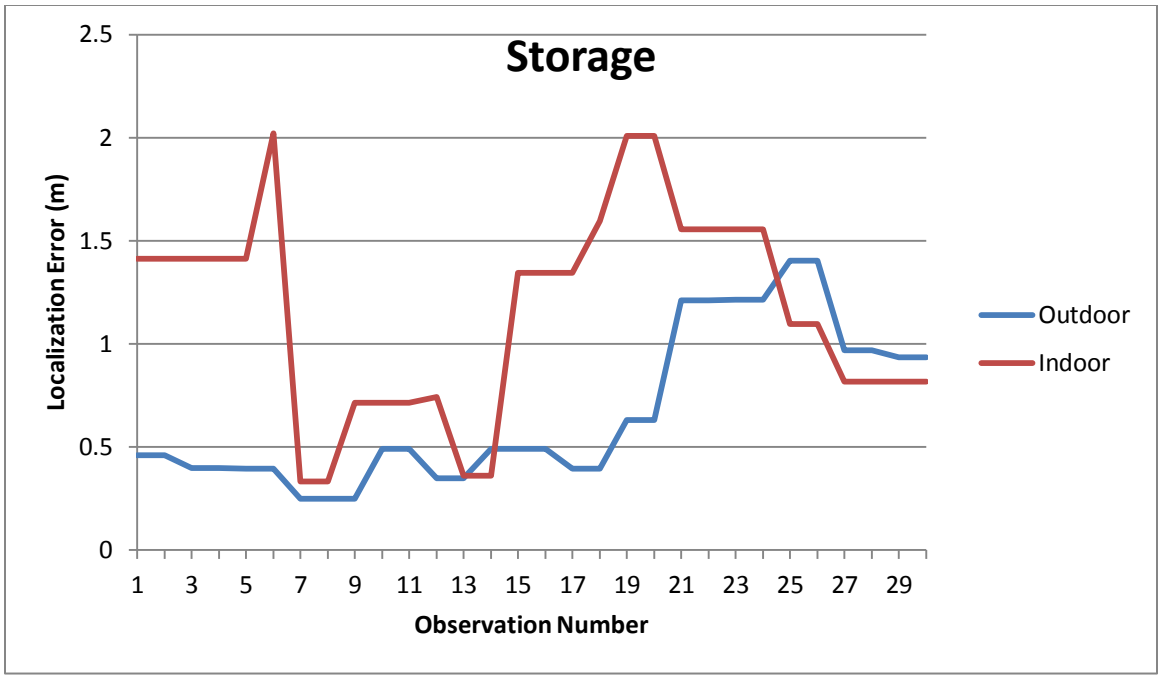

Figure 4.10: Tag 3 'Storage' localization errors for both indoor and outdoor observation.

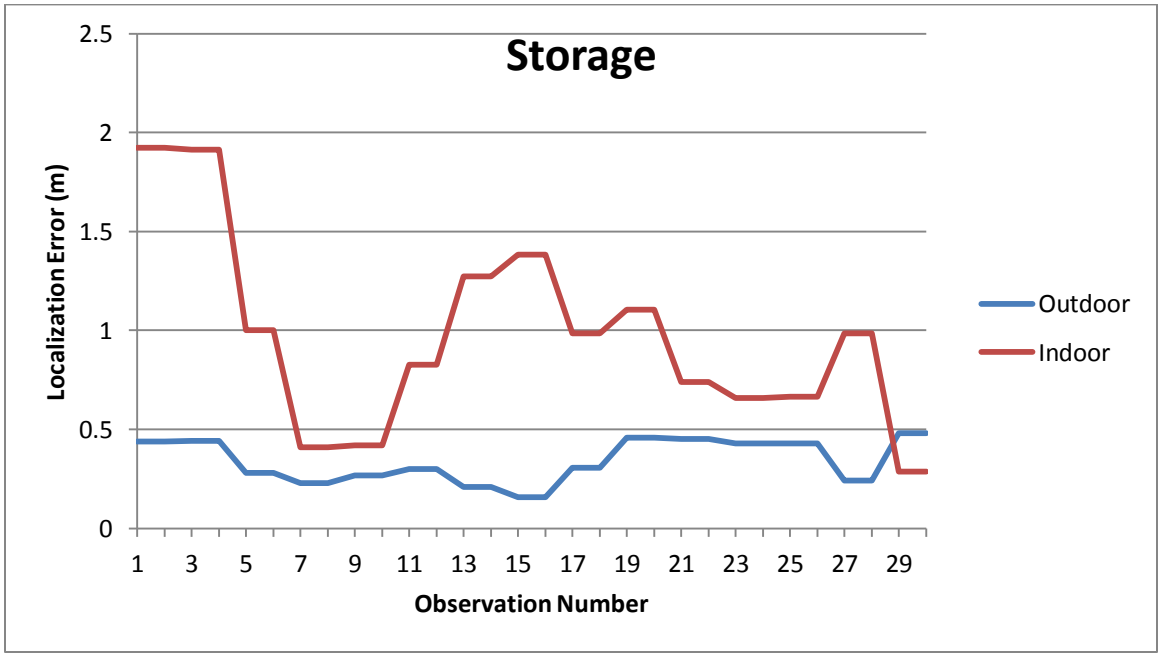

Figure 4.11: Tag 4 'Storage' localization errors for both indoor and outdoor observation.

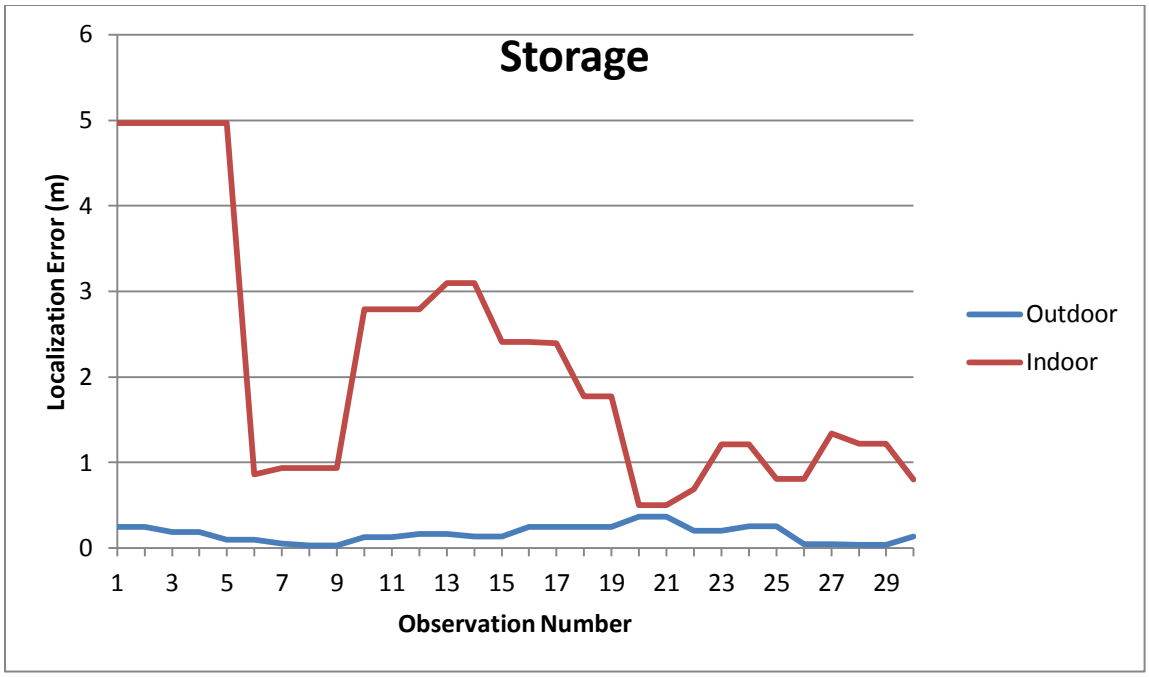

Figure 4.12: Tag 5 'Storage' localization errors for both indoor and outdoor observation.

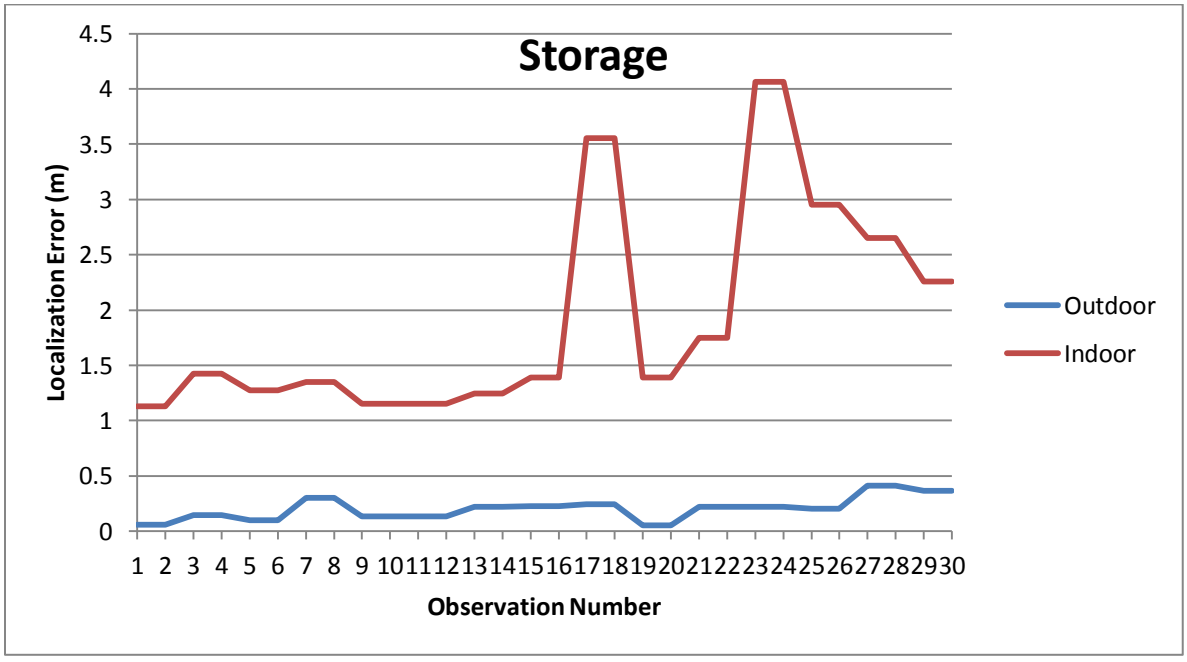

Figure 4.13: Tag 6 'Storage' localization errors for both indoor and outdoor observation

4.2.2.3 Installation Area/Point

The third logistical sequence for component tracking required the materials to be hauled to the installation location. For the 'installed' location tracking (**Error! Reference ource not found.**, Figure 4.15, Figure 4.16, Figure 4.17, Figure 4.18, and Figure 4.19), the localization error was higher indoors than in the outdoor environment. However, in the outdoor environment, the localization error was less than 1m. This resulted in false and delay in virtual model updates. Even when the component was installed, the virtual model sometimes indicated that the component was uninstalled. The level of accuracy needed for tracking 'installed' status of the tagged components is high. Thus, it was necessary to have no localization error for 'installed' location tracking. Accuracy in 'installed' location tracking provides opportunities for real-time construction progress tracking by integrating the developed prototype system with schedule and cost forecasting applications such as earned value analysis. As extreme accuracy of the RFID-RTLS system is required for real-time status update to the model, there is need to investigate how the accuracy of the RTLS system can be enhanced.

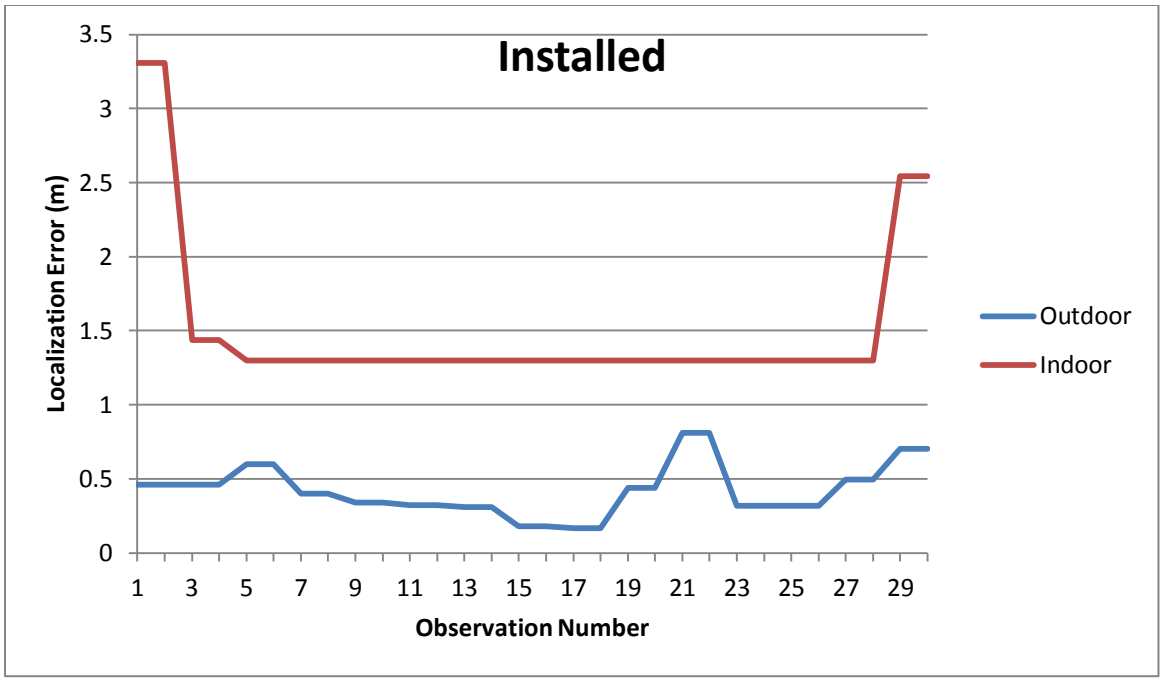

Figure 4.14: Tag 1 'Installed' localization errors for both indoor and outdoor observation.

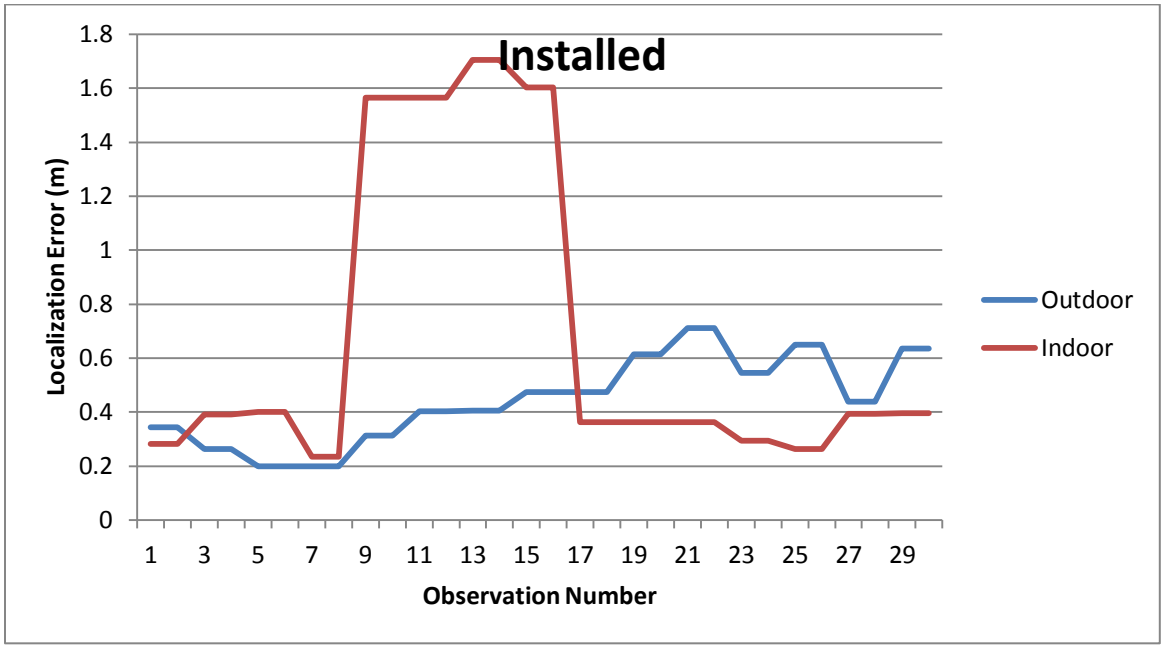

Figure 4.15: Tag 2 'Installed' localization errors for both indoor and outdoor observation.

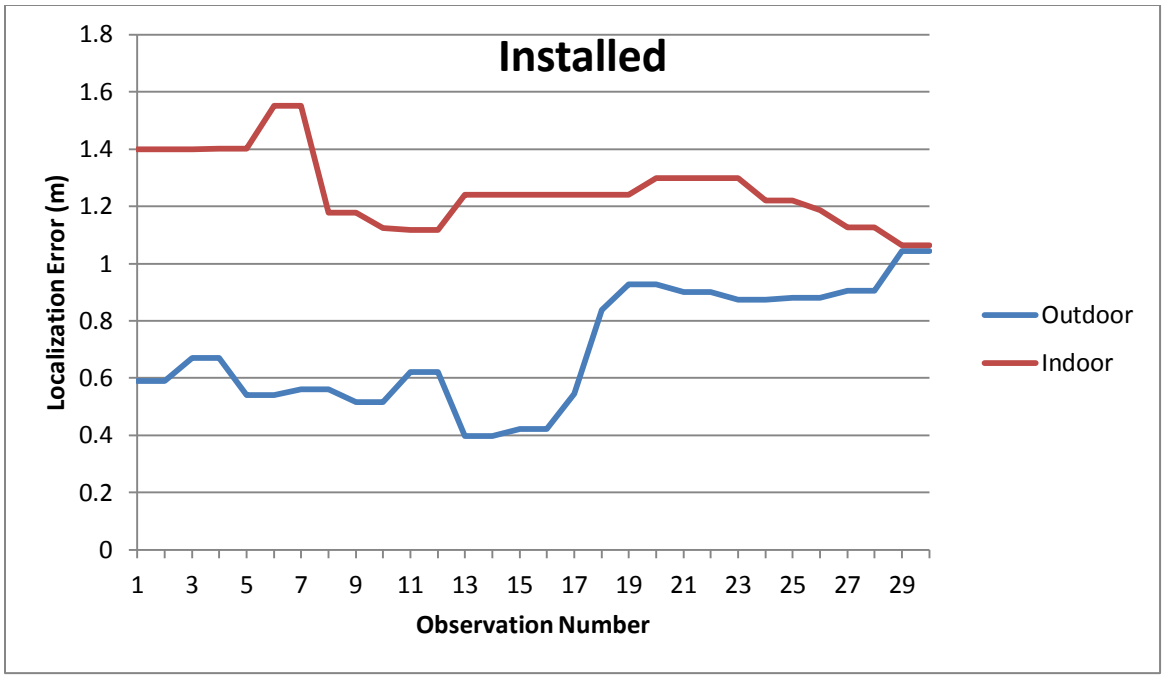

Figure 4.16: Tag 3 'Installed' localization errors for both indoor and outdoor observation.

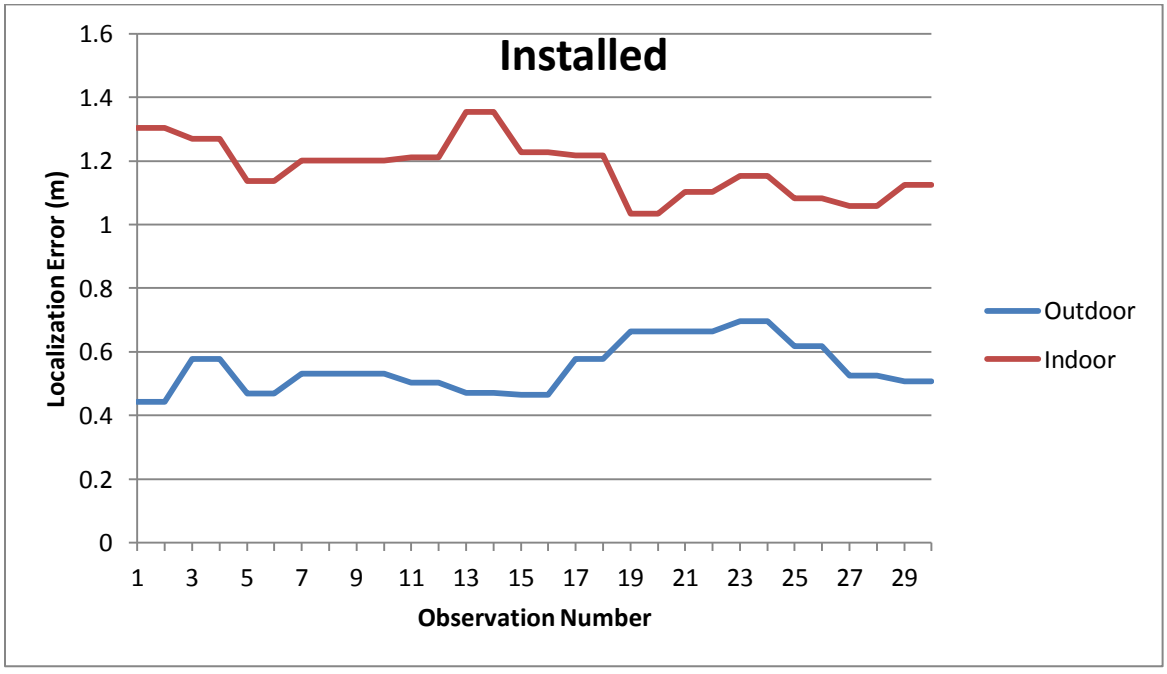

Figure 4.17: Tag 4 'Installed' localization errors for both indoor and outdoor observation.

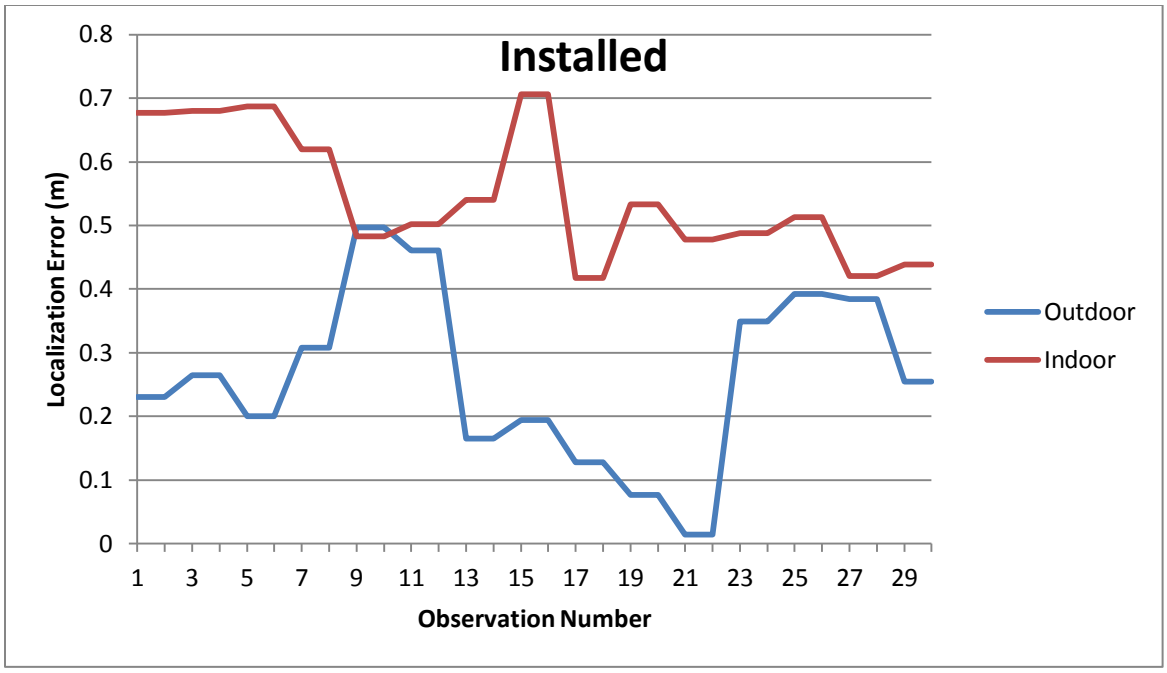

Figure 4.18: Tag 5 'Installed' localization errors for both indoor and outdoor observation.

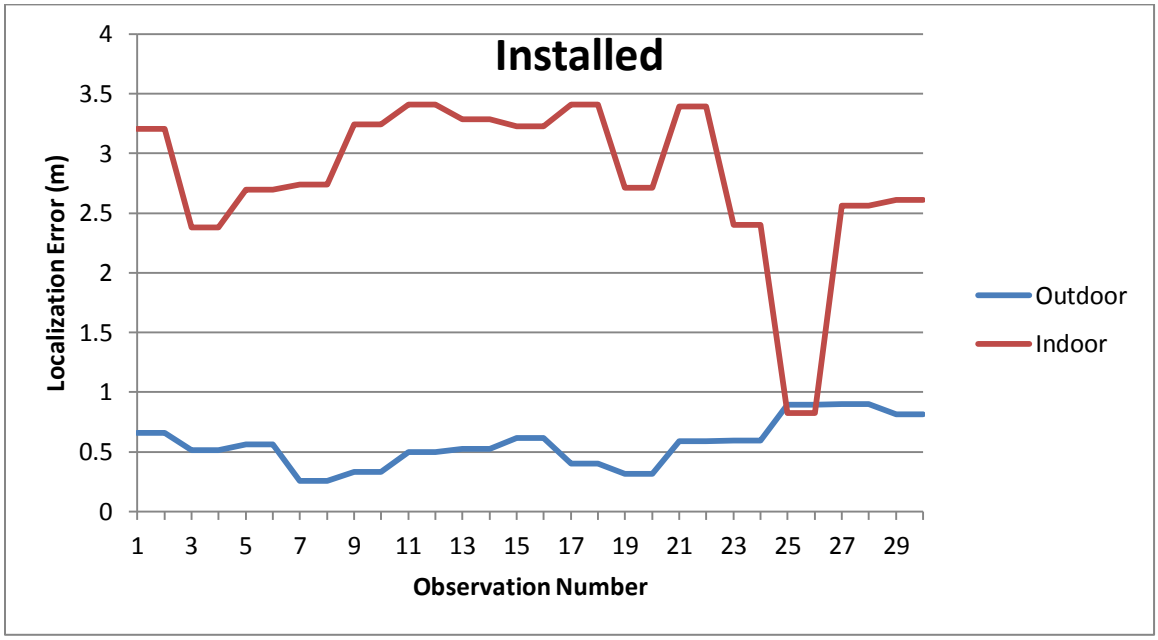

Figure 4.19: Tag 6 'Installed' localization errors for both indoor and outdoor observation.

4.2.3 Filtered Results

Since the installed location required accurate location data, Kalman Filter was used to improve the accuracy of the system by reducing the localization error. Details of the Kalman filter algorithm are presented in the preceding chapter. The results from the Kalman filter are reported for both the indoor and outdoor environment data.

4.2.3.1 Indoor

For the indoor experiment, Figure 4.20 (Tag 1), Figure 4.21 (Tag 2), Figure 4.22 (Tag 3), Figure 4.23 (Tag 4), Figure 4.24 (Tag 5), and Figure 4.25 (Tag 6) represents the localization error when the components were installed. The graphs also present the localization error from Kalman filtered value for each set of data from each tag.

From the figures, it can be observed that the Kalman filtered localization errors were very consistent and the 'noise' from the original reported localization errors was removed. It was observed that the localization error are and less than 2.5m for Tag 1 (Figure 4.20) less than 0.9m for Tag 2 (Figure 4.21), less than 0.7m for Tag 3 (Figure 4.22), less than 1.3m Tag 4 (Figure 4.23), less than 0.6m for Tag 5 (Figure 4.24) and less than 2.2m for Tag 6 (Figure 4.25). These values are more consistent than the original results. This ensures stability of the results required by the system to update the virtual model.

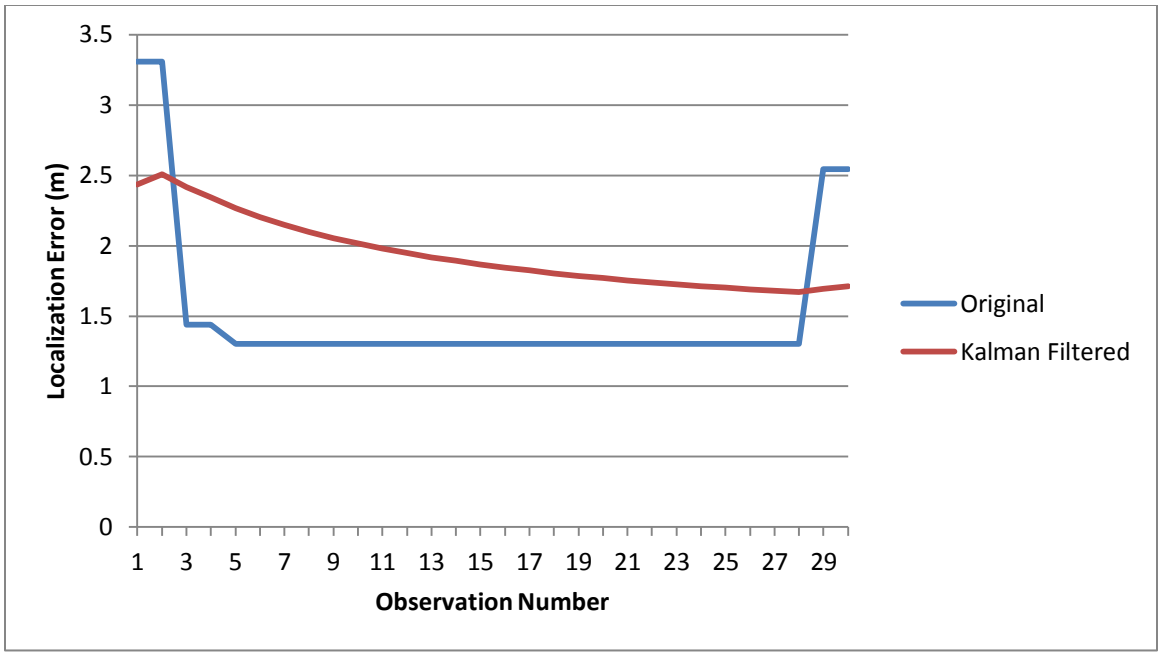

Figure 4.20: Tag 1 'Installed' localization errors for both the original and Kalman filter results.

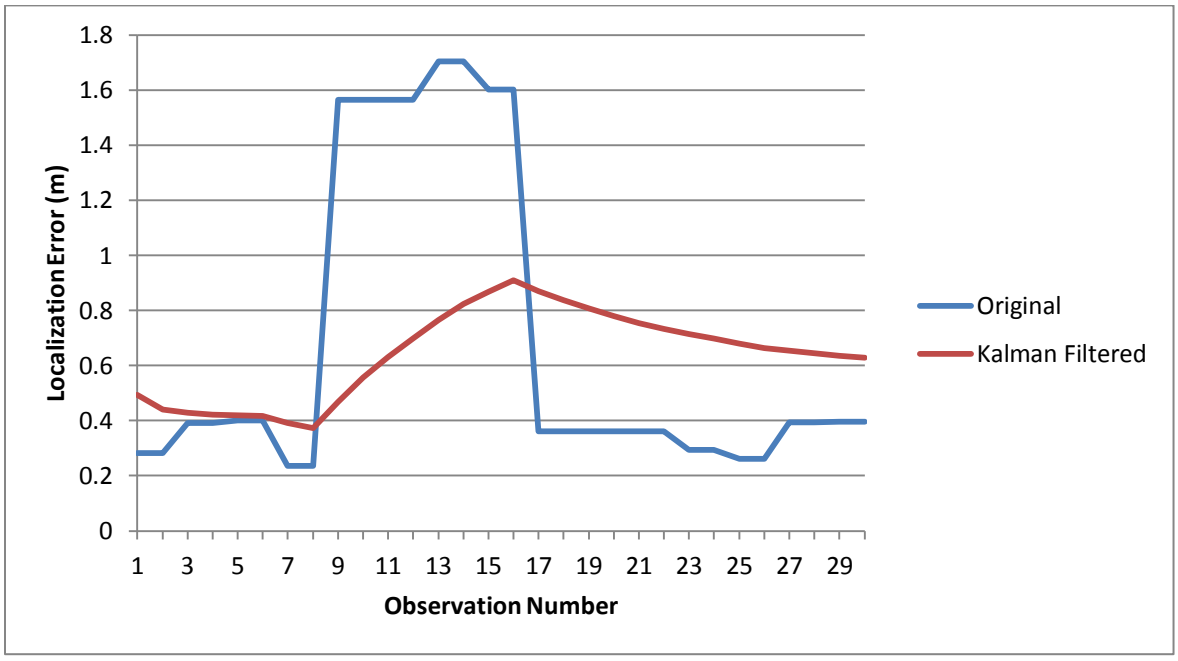

Figure 4.21: Tag 2 'Installed' localization errors for both the original and Kalman filter results.

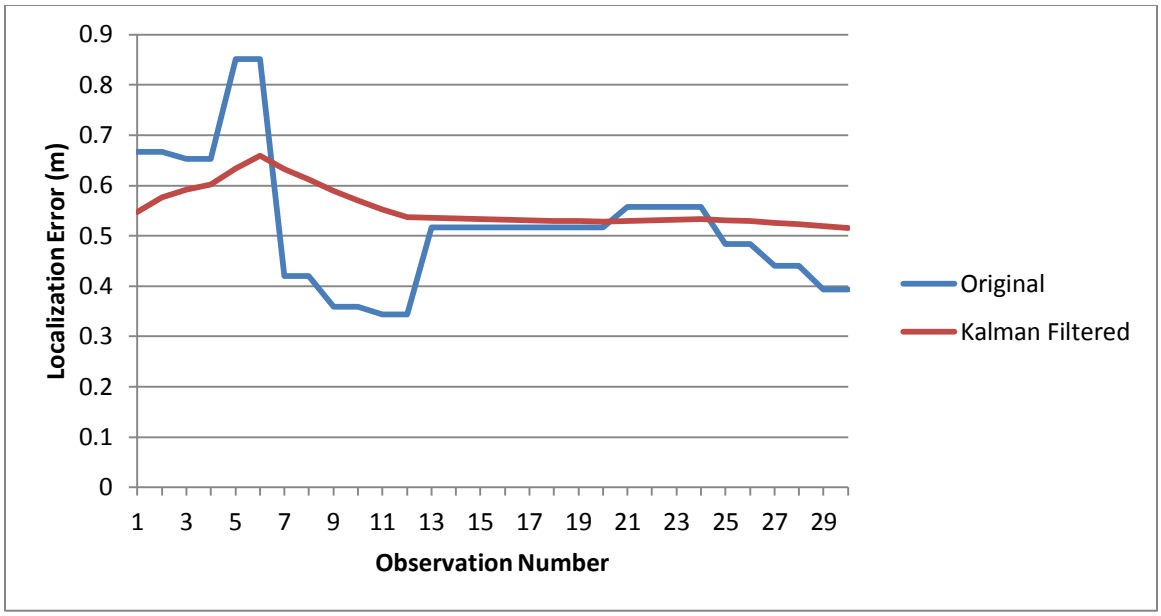

Figure 4.22: Tag 3 'Installed' localization errors for both the original and Kalman filter results.

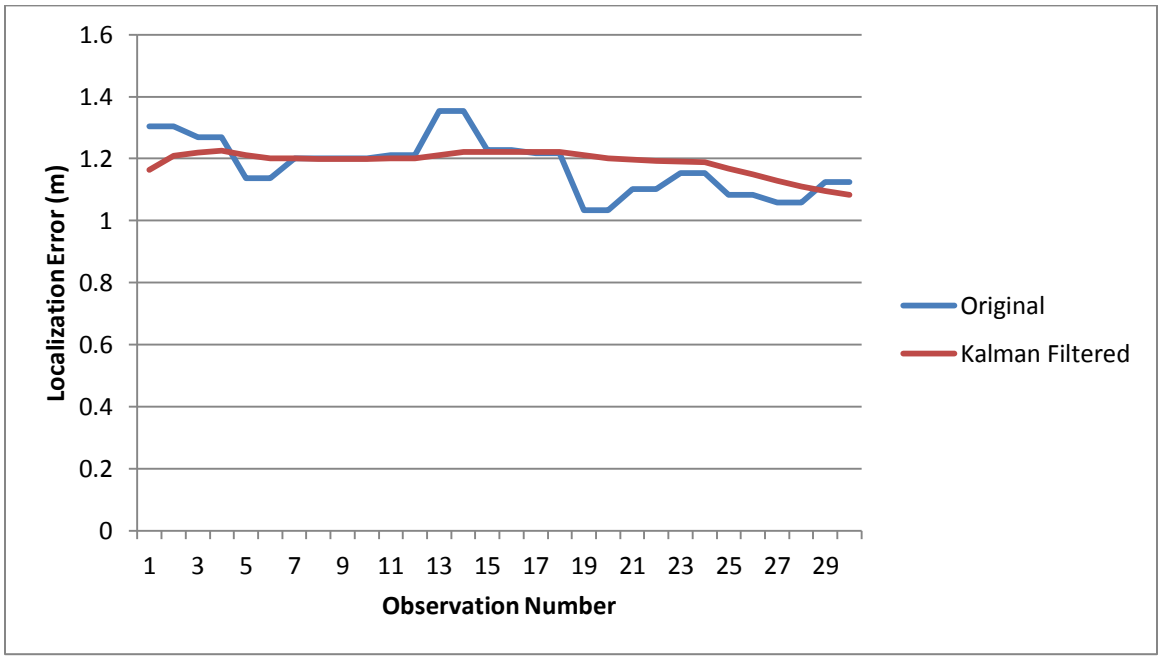

Figure 4.23: Tag 4 'Installed' localization errors for both the original and Kalman filter results.

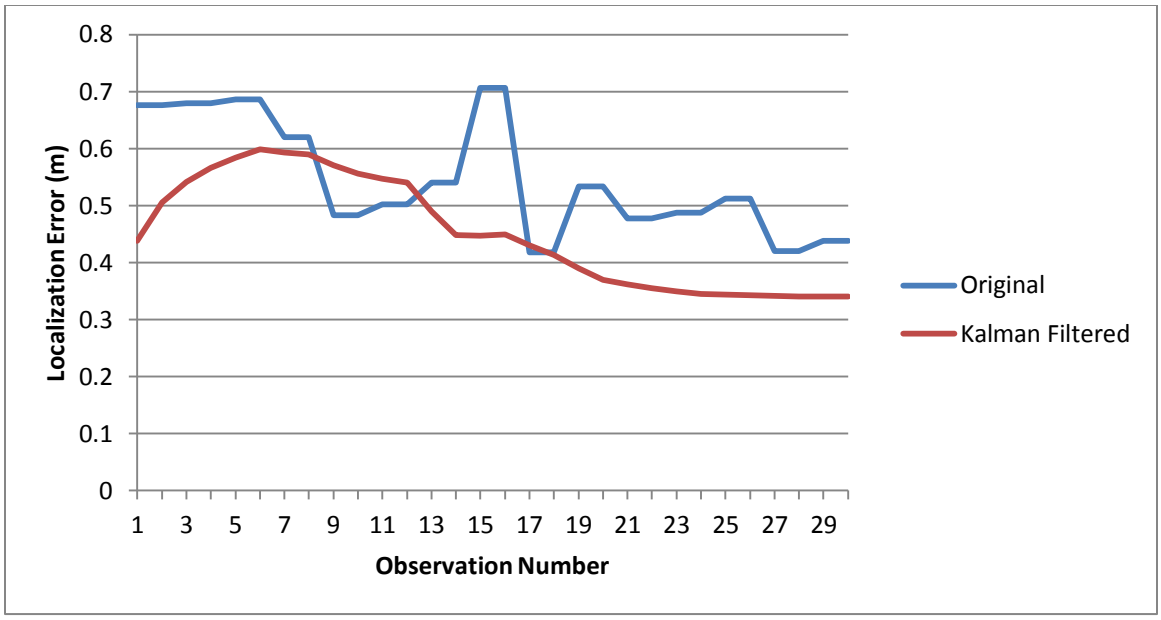

Figure 4.24: Tag 5 'Installed' localization errors for both the original and Kalman filter results.

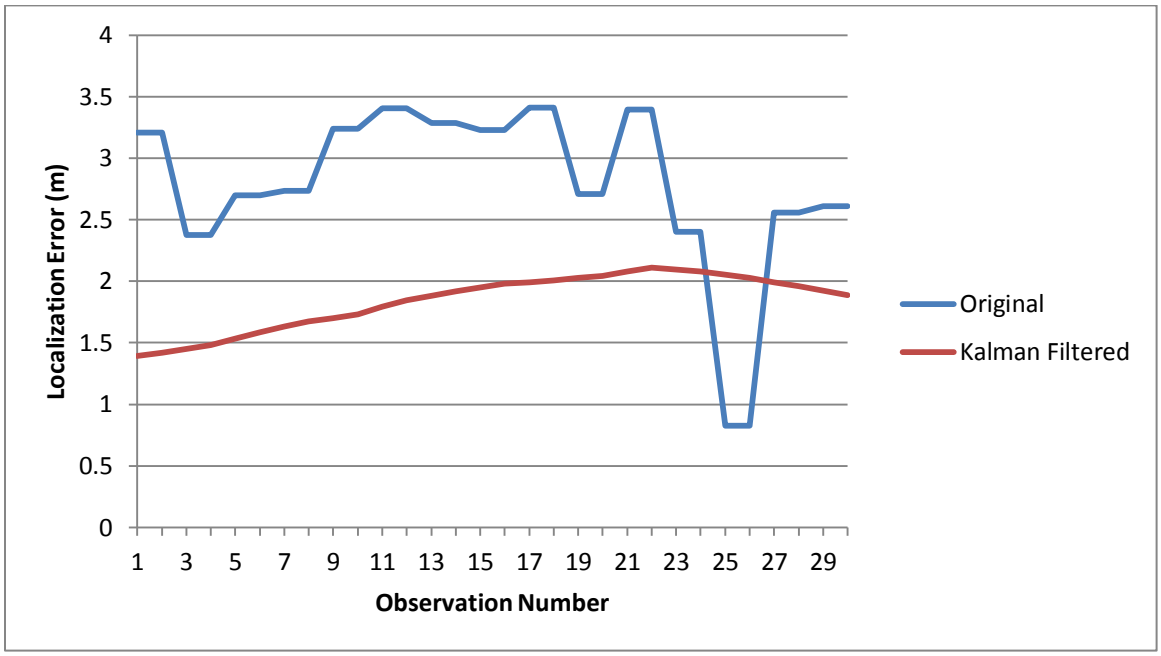

Figure 4.25: Tag 6 'Installed' localization errors for both the original and Kalman filter results.

Statistical significance of localization errors before and after Kalman Filter as applied was compared by plotting boxplots. Boxplots have four major components, lower whisker, box, box divider (red line inside the box) and upper whisker Lower whisker represents 25% (from 0-25%) of the data, box represents 50% (from 25% - 75%) of the data and upper and the upper whisker represents the rest of 25% (75% to 100%) of the population. The box divider represents the median of the data. Based on the size of the box, variation in the data can be graphically visualized: the bigger the size of the box, the bigger the variation in the data and vice versa.

Figure 4.26, Figure 4.27, Figure 4.28, Figure 4.29, Figure 4.30, and Figure 4.31 represents the boxplots for the localization error (LE) for both the reported locations and the filtered results. From each figure, the first boxplot represents the reported localization error (LE - Original) while the second boxplot represents the filtered localization error (LE - Filtered).

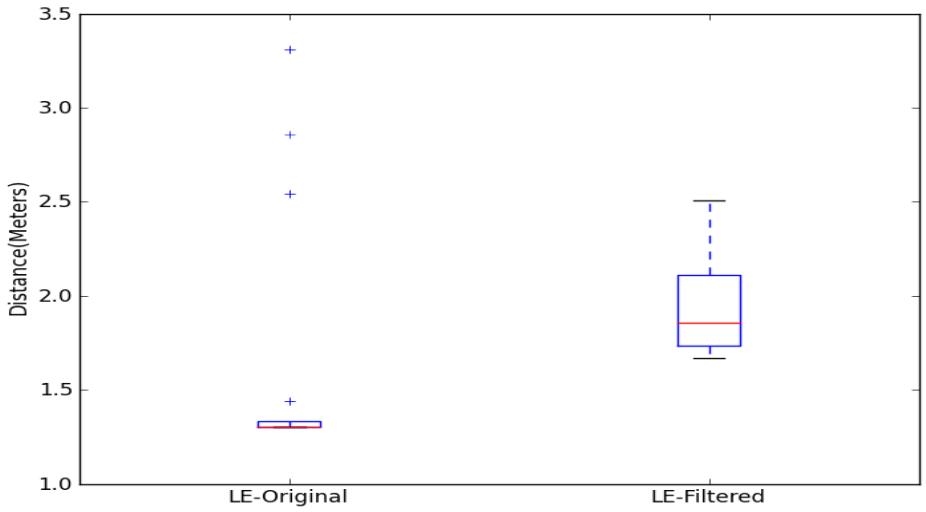

Figure 4.26: Tag 1 'Installed' localization error boxplots for both the original and Kalman filter results.

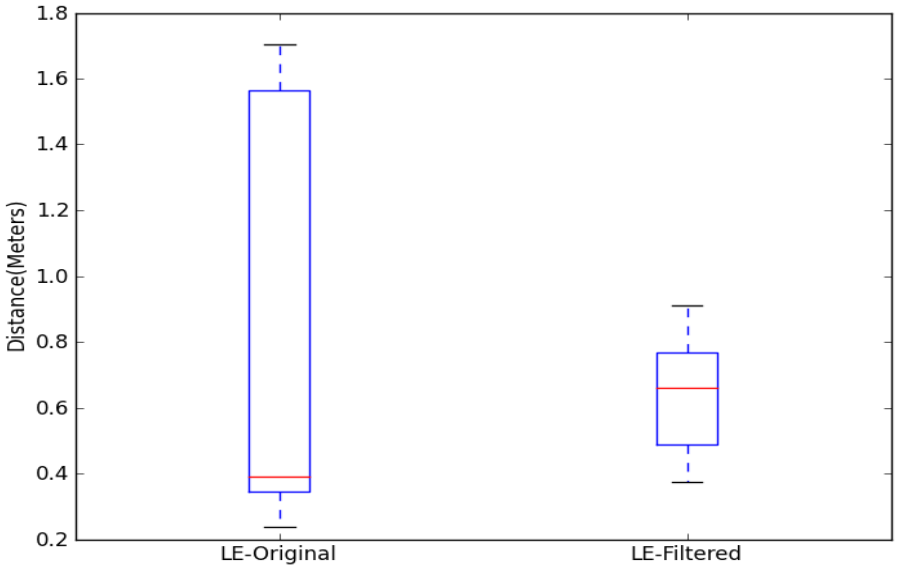

Figure 4.27: Tag 2 'Installed' localization error boxplots for both the original and Kalman filter results.

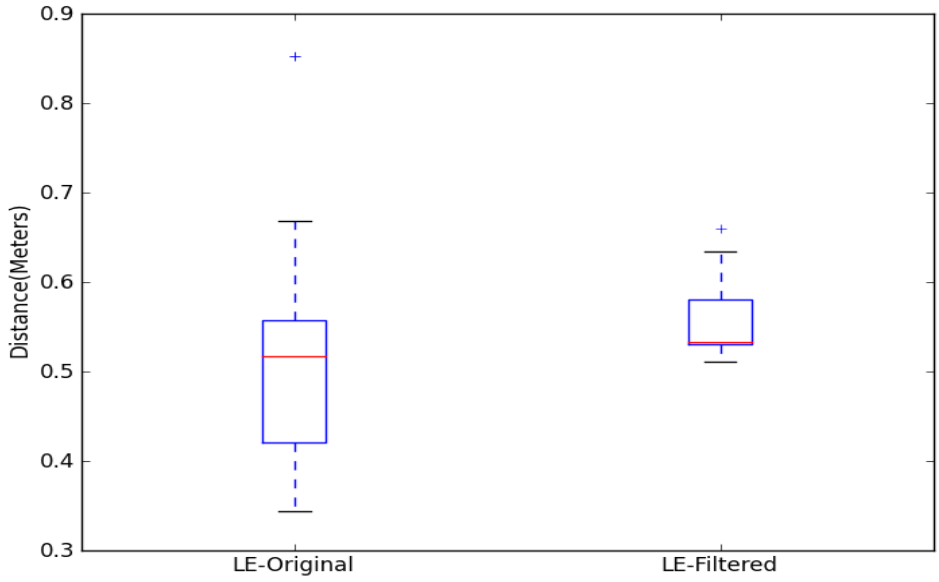

Figure 4.28: Tag 3 'Installed' localization error boxplots for both the original and Kalman filter results.

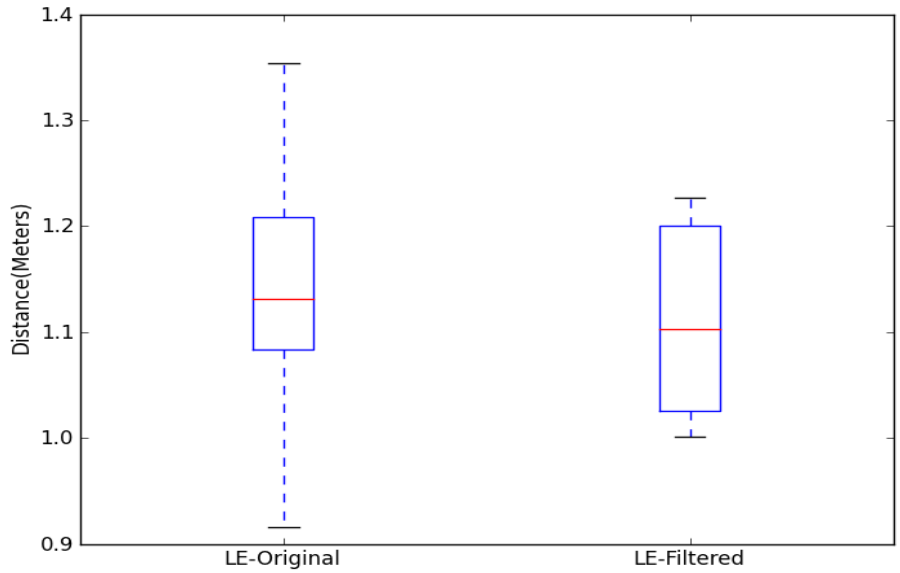

Figure 4.29: Tag 4 'Installed' localization error boxplots for both the original and Kalman filter results.

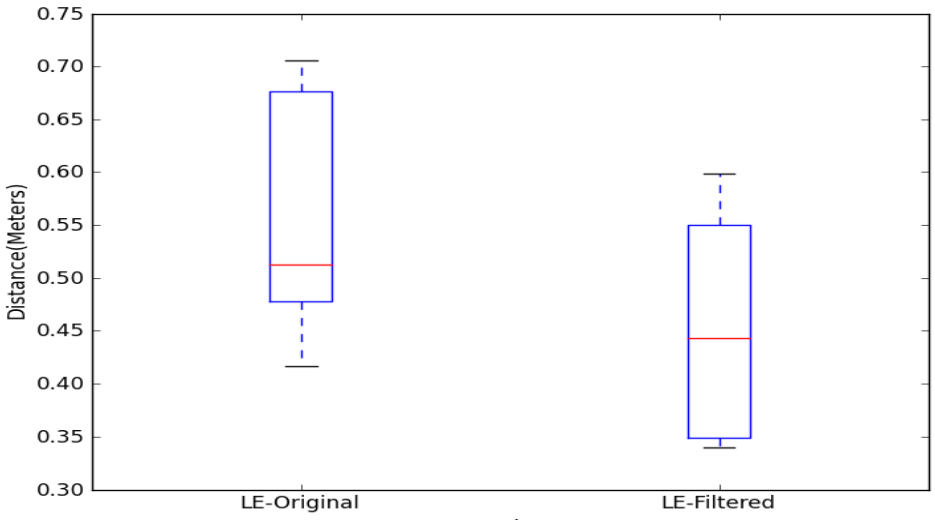

Figure 4.30: Tag 5 'Installed' localization error boxplots for both the original and Kalman filter results.

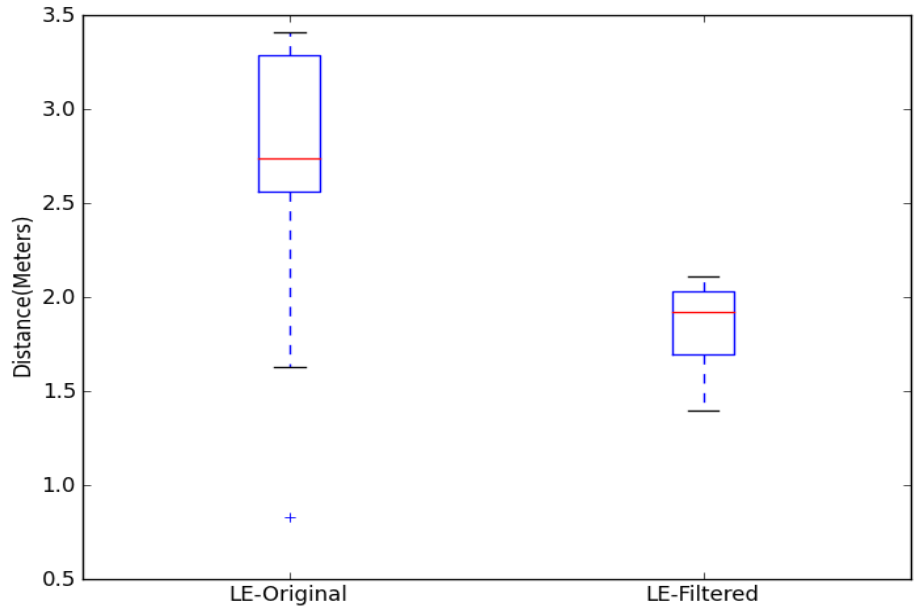

Figure 4.31: Tag 6 'Installed' localization error boxplots for both the original and Kalman filter results.

From the indoor experiment, it was observed that Tag 1 had a tolerance of ±3.4m as the original localization error and ± 2.5 as the filtered localization error. Tag 2 had a tolerance of ±1.7m as the original localization error and ± 0.91m as the filtered localization error. Tag 3 had a tolerance of ±0.85m as the original localization error and ± 0.66m as the filtered localization error. Tag 4 had a tolerance of ±1.35m as the original localization error and ± 1.23m as the filtered localization error. Tag 5 had a tolerance of ±0.71m as the original localization error and ± 0.59m as the filtered localization error. Tag 6 had a tolerance of ±0.41m as the original localization error and ± 2.11m as the filtered localization error.

4.2.3.2 Outdoors

Similar to the indoor results, Figure 4.32 (Tag 1), Figure 4.33 (Tag 2), Figure 4.34 (Tag 3), Figure 4.35 (Tag 4), Figure 4.36 (Tag 5), and Figure 4.37 (Tag 6) represent the localization error when the components were installed in the outdoor environment. The graphs also present the localization error from Kalman filtered value for each set of data from each tag.

From the figures, it can be observed that the Kalman filtered localization errors are very consistent with the 'noise' from the original reported localization errors removed. It can be observed that the localization errors are less than 0.5m for Tag 5 (Figure 4.36), less than 0.7m for both Tag 2 (Figure 4.33) and Tag 4 (Figure 4.41), less than 0.8m for Tag 1 (Figure 4.38), less than 0.95m for both Tag 3 (Figure 4.40) and Tag 6 (Figure 4.37). Similar to the results of the indoor experiment, these values are more consistent than the original results. This ensures the stability required by the system to update the virtual model.

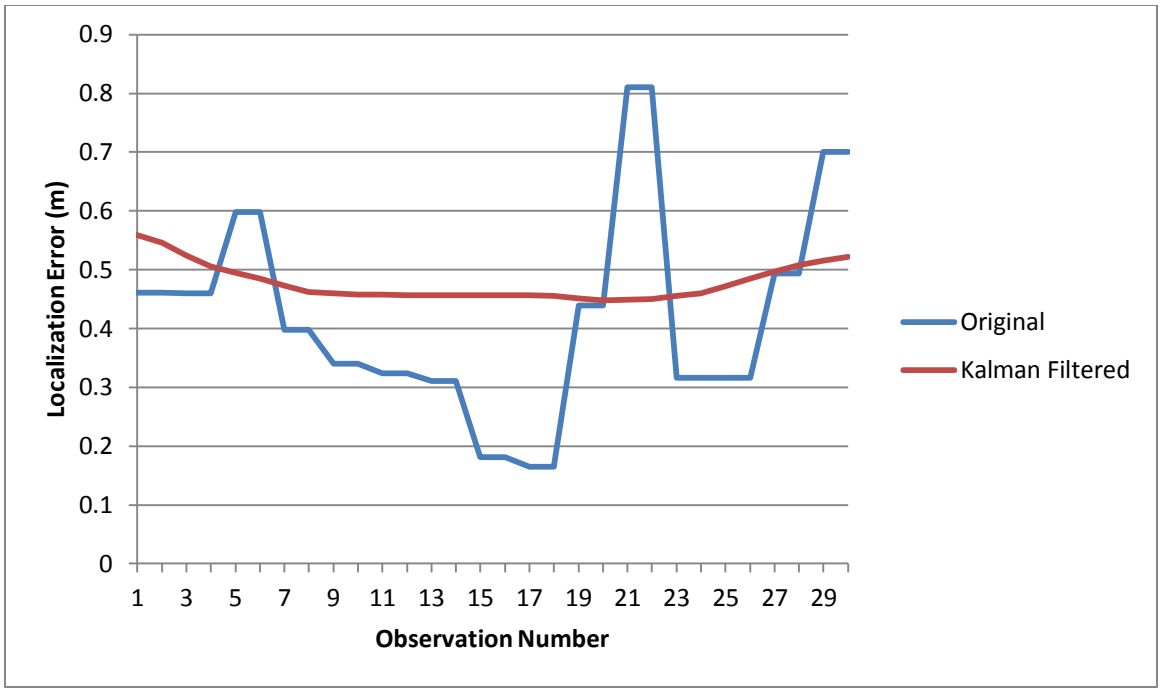

Figure 4.32: Tag 1 'Installed' localization errors for both the original and Kalman filter results.

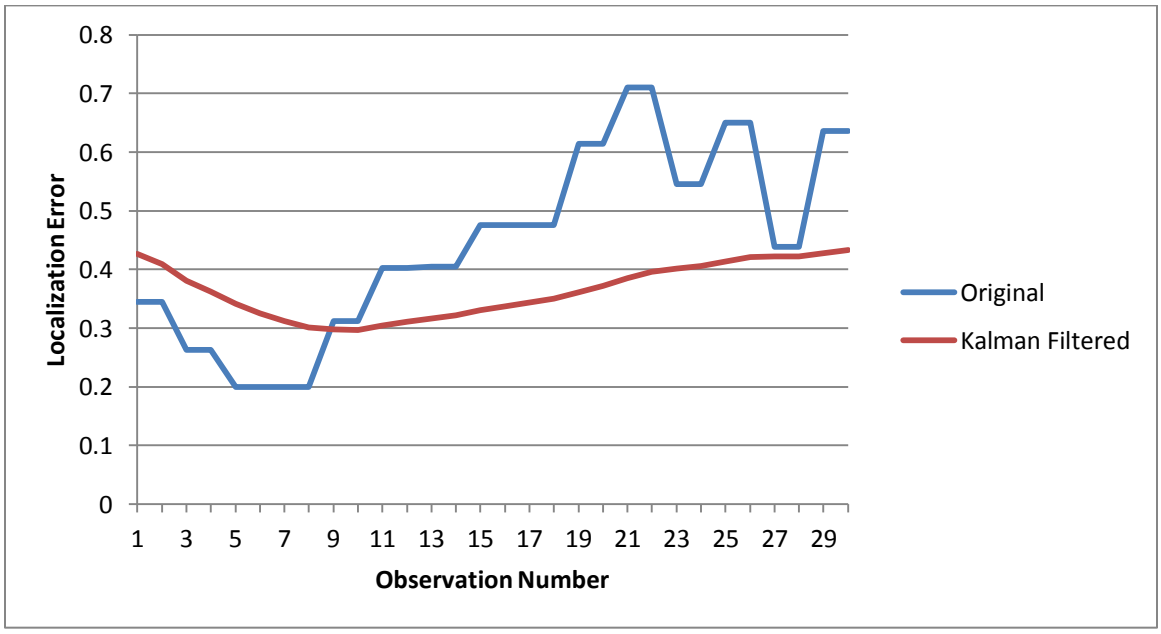

Figure 4.33: Tag 2 'Installed' localization errors for both the original and Kalman filter results.

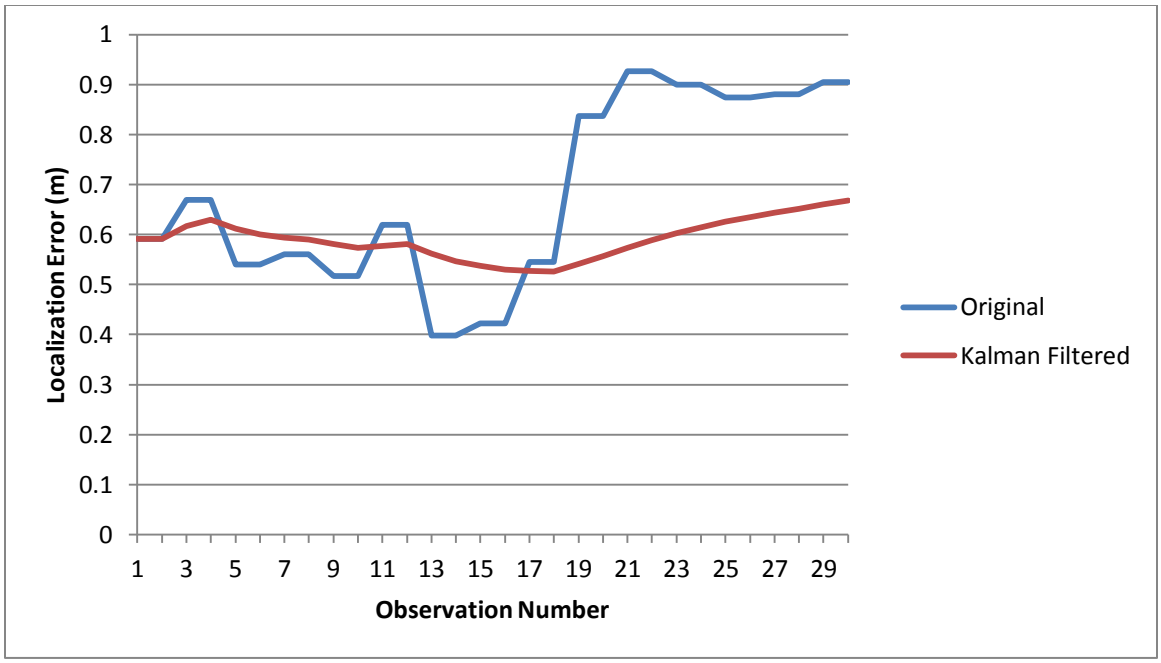

Figure 4.34: Tag 3 'Installed' localization errors for both the original and Kalman filter results.

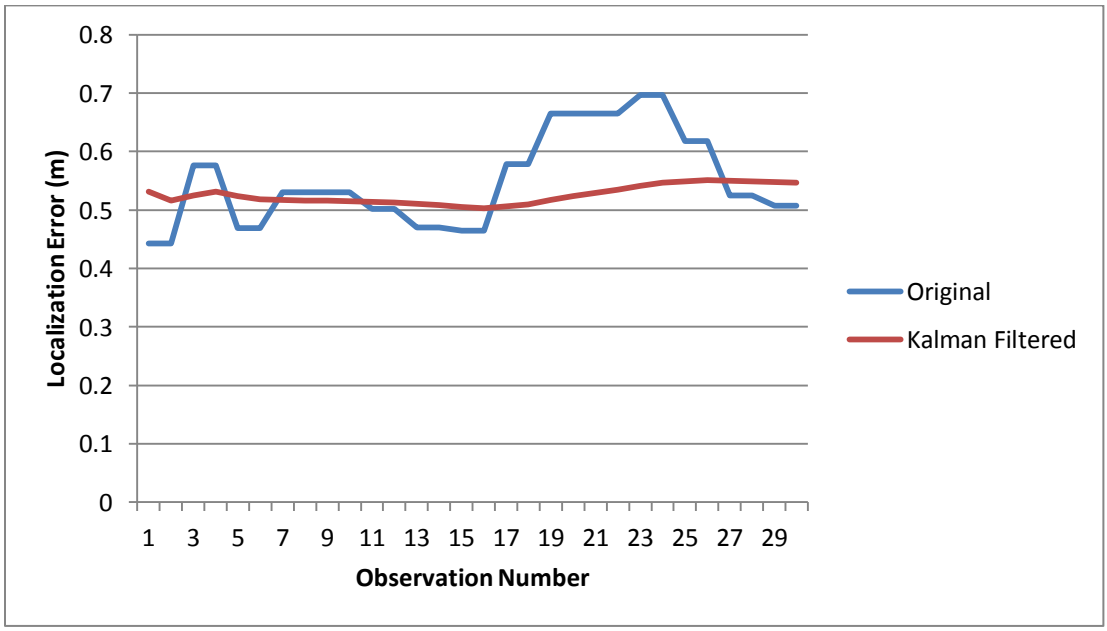

Figure 4.35: Tag 4 'Installed' localization errors for both the original and Kalman filter results.

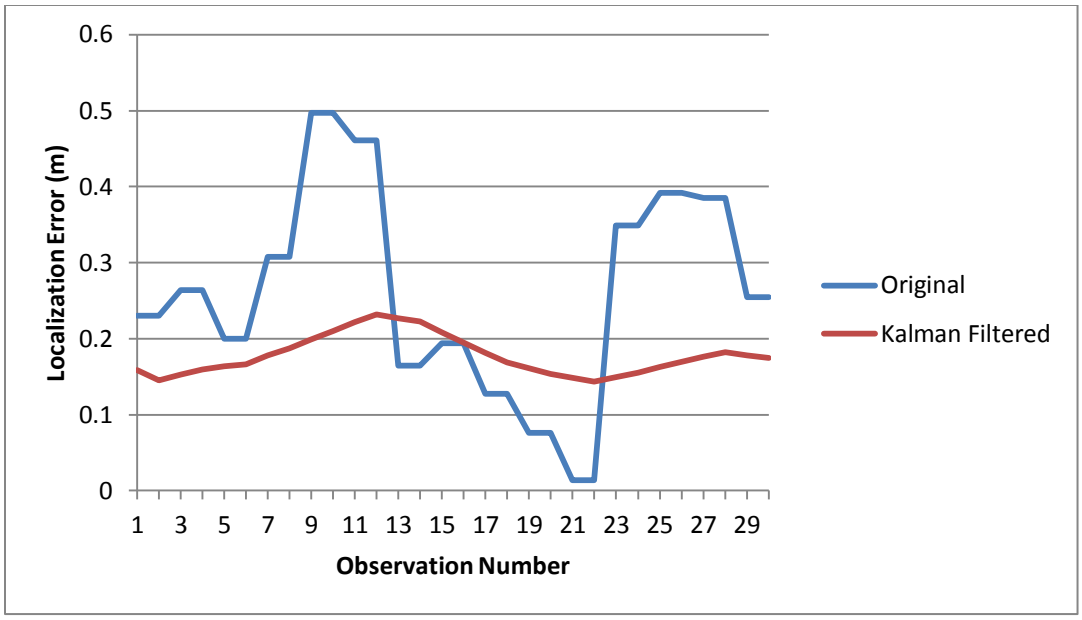

Figure 4.36: Tag 5 'Installed' localization errors for both the original and Kalman filter results.

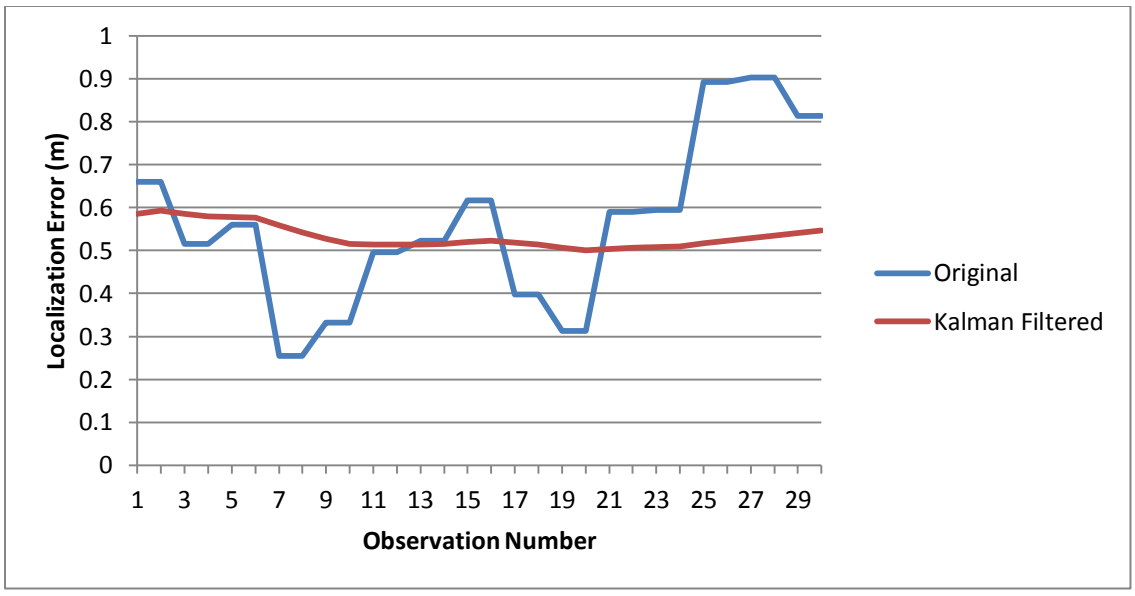

Figure 4.37: Tag 6 'Installed' localization errors for both the original and Kalman filter results.

Similar to the indoor results, the statistical significance of localization errors before and after Kalman filter was applied was compared using boxplots. Figure 4.38, Figure 4.39, Figure 4.40, Figure 4.41, Figure 4.42 and Figure 4.43 represent the boxplots of the localization error (LE) for both the reported locations and the filtered results. From the figures, the first boxplot represents the reported localization error (LE - Original) and the second boxplot represents the filtered localization error (LE - Filtered). It can be observed that the variation in the filtered localization error (LE – Filtered) is less than the reported localization error (LE – Original). The filtered values are more consistent when compared to the reported results; hence a tolerance can be set for spatially mapped 'installation' location for the system to update the virtual model when the tag is within that area.

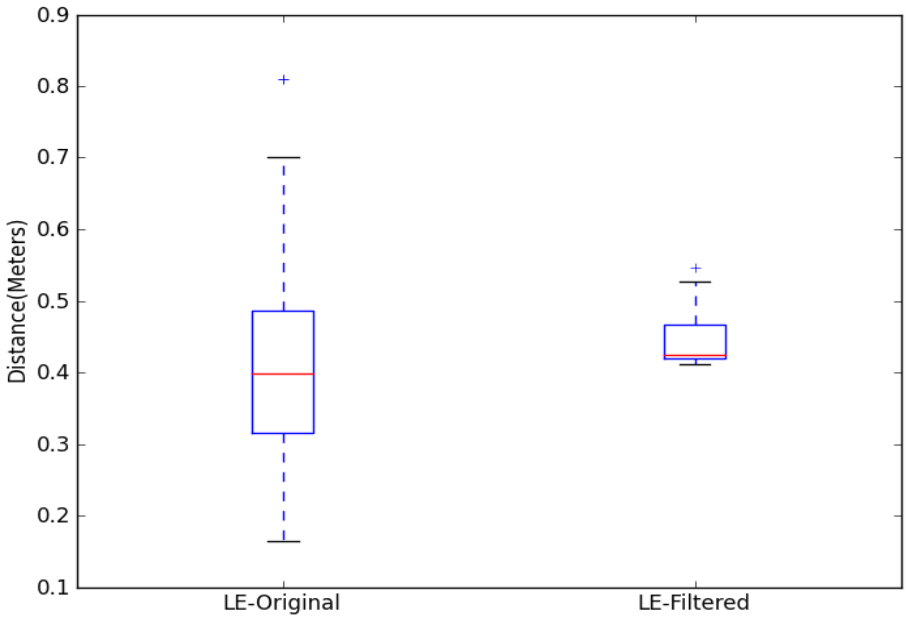

Figure 4.38: Tag 1 'Installed' localization error boxplots for both the original and Kalman filter results.

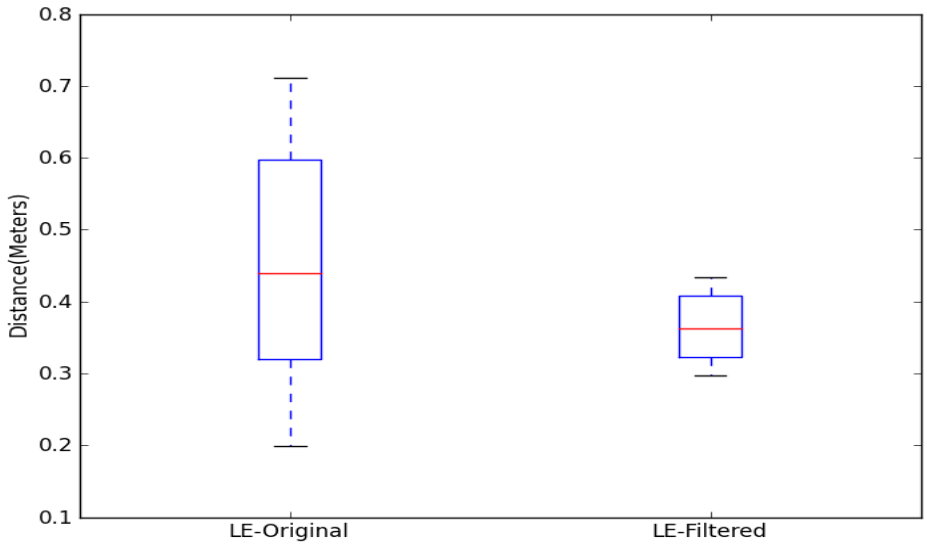

Figure 4.39: Tag 2 'Installed' localization error boxplots for both the original and Kalman filter results.

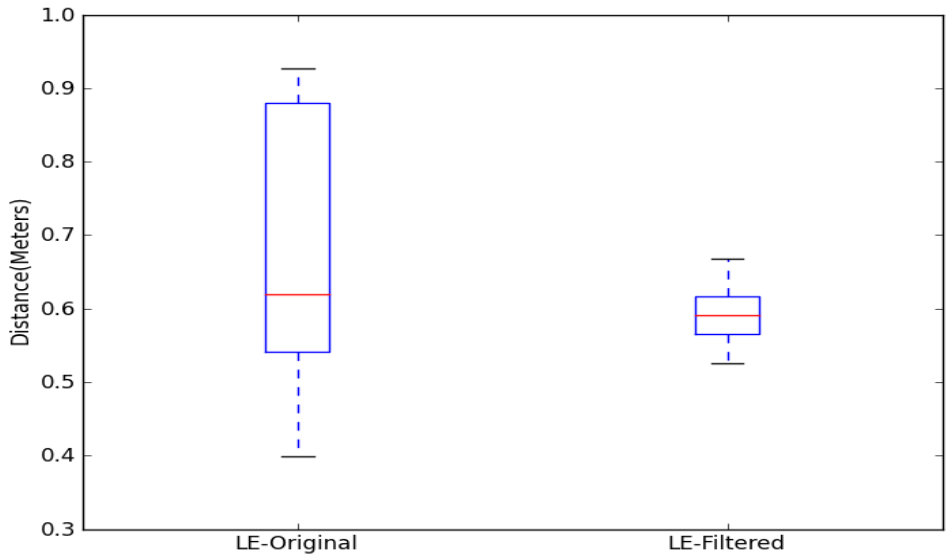

Figure 4.40: Tag 3 'Installed' localization error boxplots for both the original and Kalman filter results.

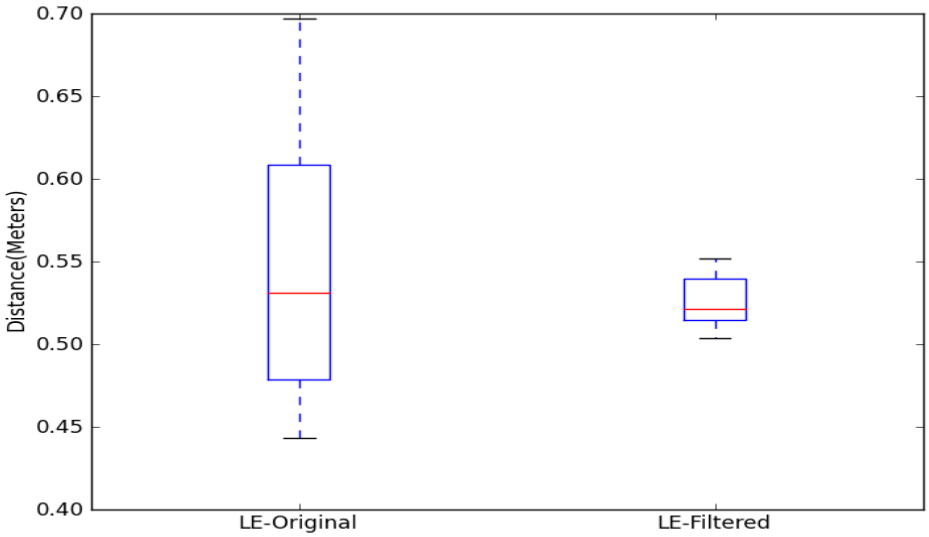

Figure 4.41: Tag 4 'Installed' localization error boxplots for both the original and Kalman filter results.

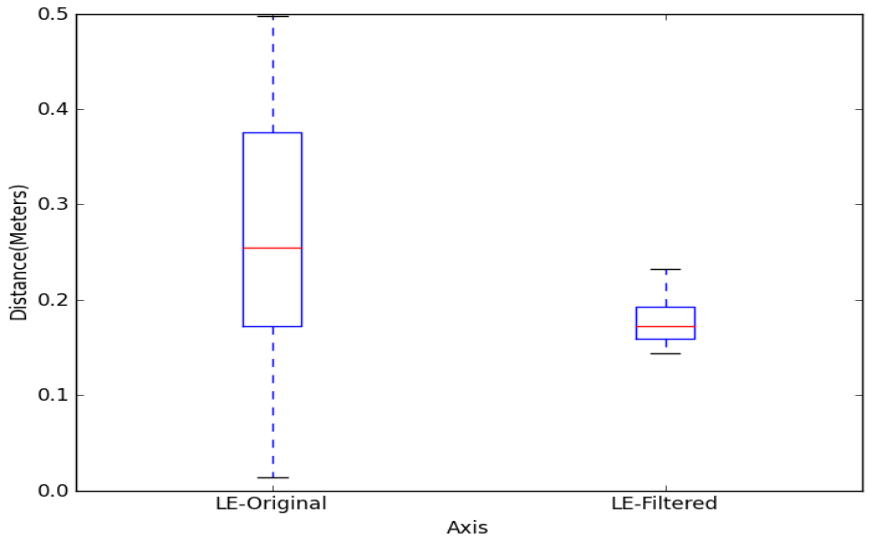

Figure 4.42: Tag 5 'Installed' localization error boxplots for both the original and Kalman filter results.

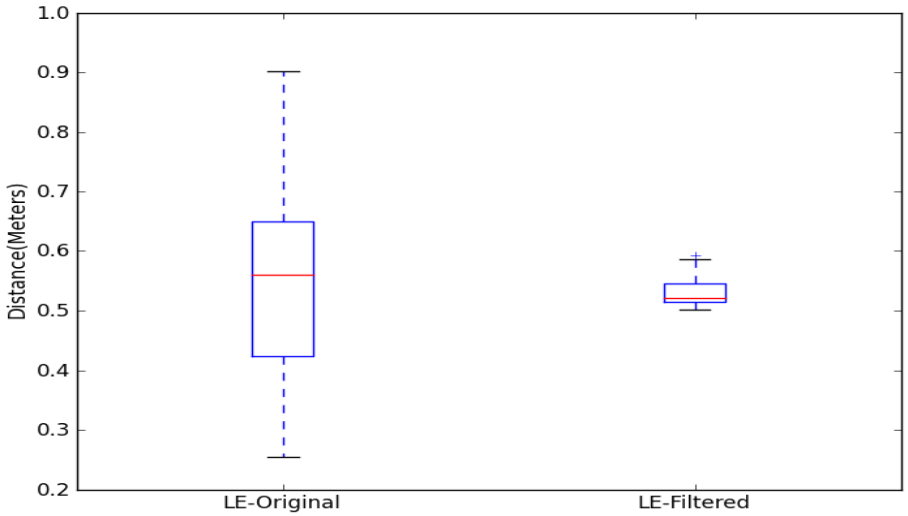

Figure 4.43: Tag 6 'Installed' localization error boxplots for both the original and Kalman filter results.

From the outdoor experiment, it was observed that Tag 1 had a tolerance of ±0.81m as the original localization error and ± 0.55m as the filtered localization error. Tag 2 had a tolerance of ±0.71m as the original localization error and ± 0.43m as the filtered localization error. Tag 3 had a tolerance of ±0.93m as the original localization error and ± 0.67m as the filtered localization error. Tag 4 had a tolerance of ±0.7m as the original localization error and ± 0.55m as the filtered localization error. Tag 5 had a tolerance of ±0.5m as the original localization error and ± 0.24m as the filtered localization error. Tag 6 had a tolerance of ±0.9m as the original localization error and ± 0.59m as the filtered localization error.

4.3 Asset tracking using Tablet PC integrated with RFID-RTLS system

The aim of this experiment was to assess the feasibility and accuracy of the developed system in visually tracking tagged assets using the iPad mobile device. This experiment was performed in two laboratories: CPC and Plastics laboratory. For both laboratory tests, the estimated (reported by the RFID-RTLS system) and actual location of the tagged assets were captured. Using Equation 4.1, the location error was calculated for the tagged assets. This section presents the results for the asset tracking prototype. A typical unfiltered result is presented for each laboratory. Results from the Correction Factors method are also presented. Details of the Correction Factor method are presented in the preceding chapter.

4.3.1 General Results

Localization errors for the experiment conducted in the CPC Laboratory are shown in Figure 4.44. From the figure, it can be observed that the localization error for Tag 1 is high (<3.1m). The localization error of Tag 2 is less than 1.4m compared with Tag 3 which has a localization error less than 2.3m.

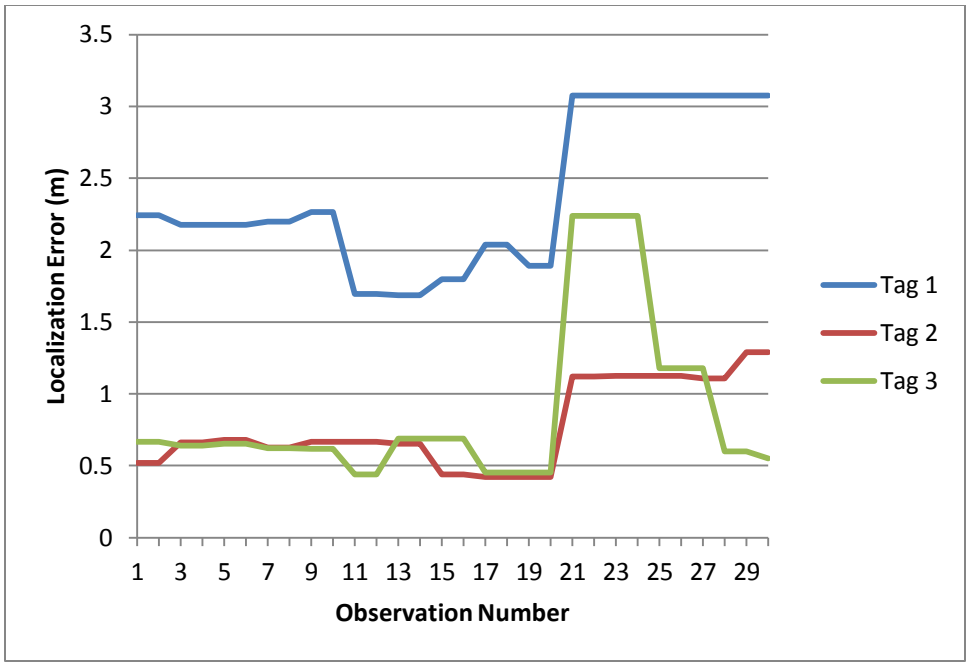

Figure 4.44: Localization Errors for three assets tracked in CPCLab.

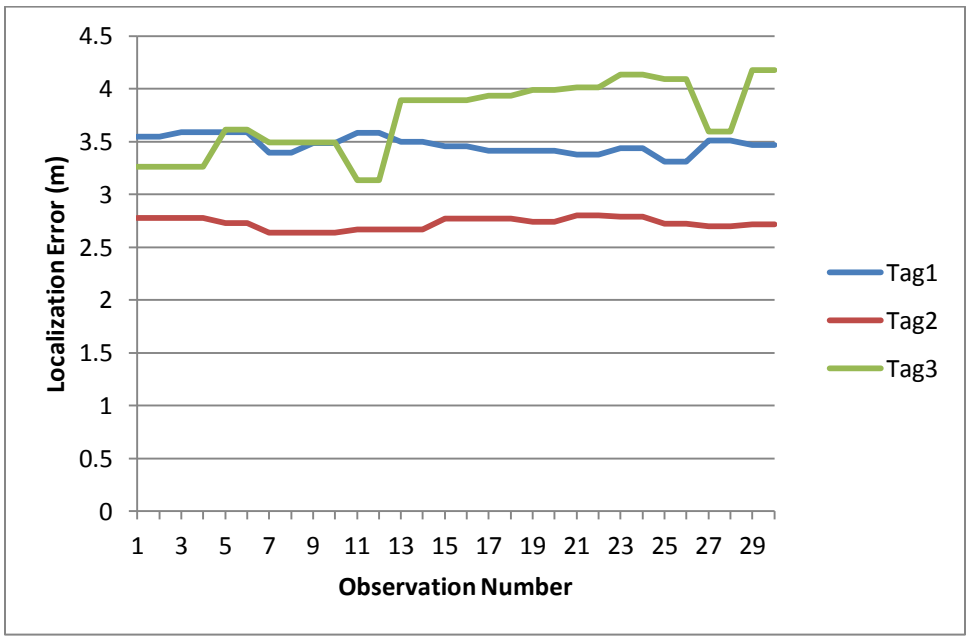

Figure 4.45: Localization Errors for three assets tracked in Plastics Lab.

Localization errors for the experiment conducted in the Plastics Lab are shown in Figure 4.45. From the figure, it can be observed that the localization error for Tag 3 is high (<4.2m). Tag 1's is localization error less than 3.6mcompared with Tag 2 which has a localization error less than 3.6m.

It is important to have a reduced localization error so that the system can relay accurate location of tagged assets in the iPad GUI. In line with this, the developed Correction Factor method was applied to the reported locations from the system. The results are shown in the subsequent section.

4.3.1.1 CPC Lab

The results before and after the application of correction factor method are shown in Asset 1 (Figure 4.46), Asset 2 (Figure 4.47), and Asset 3 (Figure 4.48). From Asset 1 the correction factor reduced the localization error to less than 1.5m compared with the RFID-RTLS system localization, For Asset 2, the localization errors are less than 3m (reported by the RFID-RTLS system).

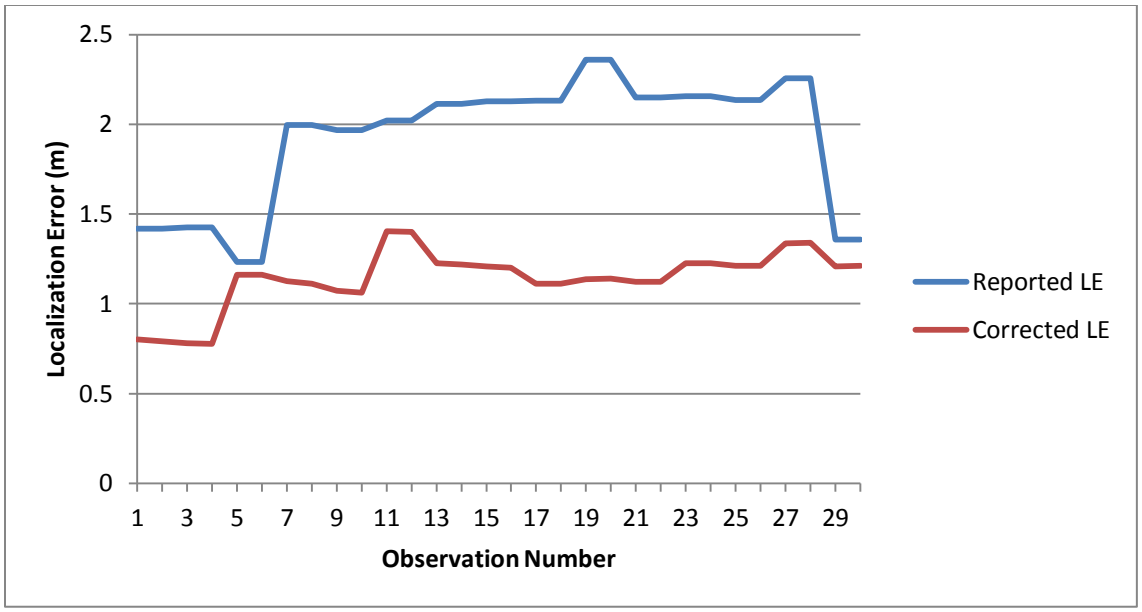

Figure 4.46: Asset 1 localization error for both the Original and Corrected Location for CPC Lab

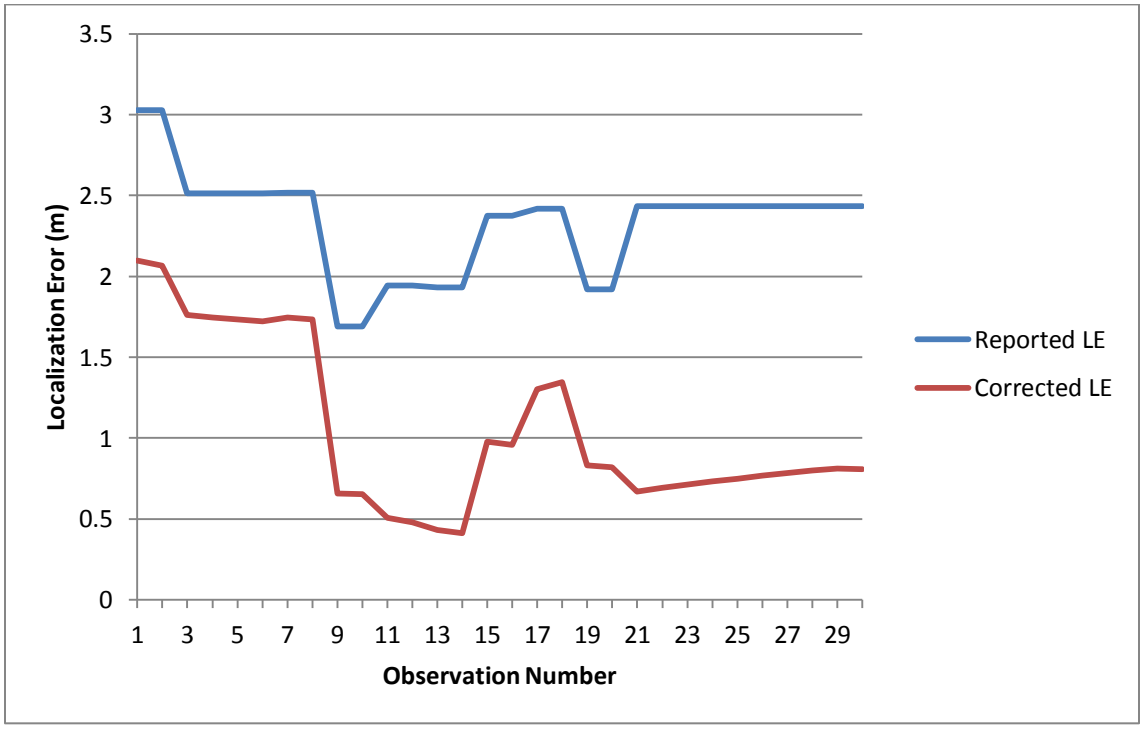

Figure 4.47: Asset 2 localization error for both the Original and Corrected Location CPC Lab

For Asset 3 (Figure 4.48), results from the correction factor method was worse than that achieved from the RFID-RTLS system.

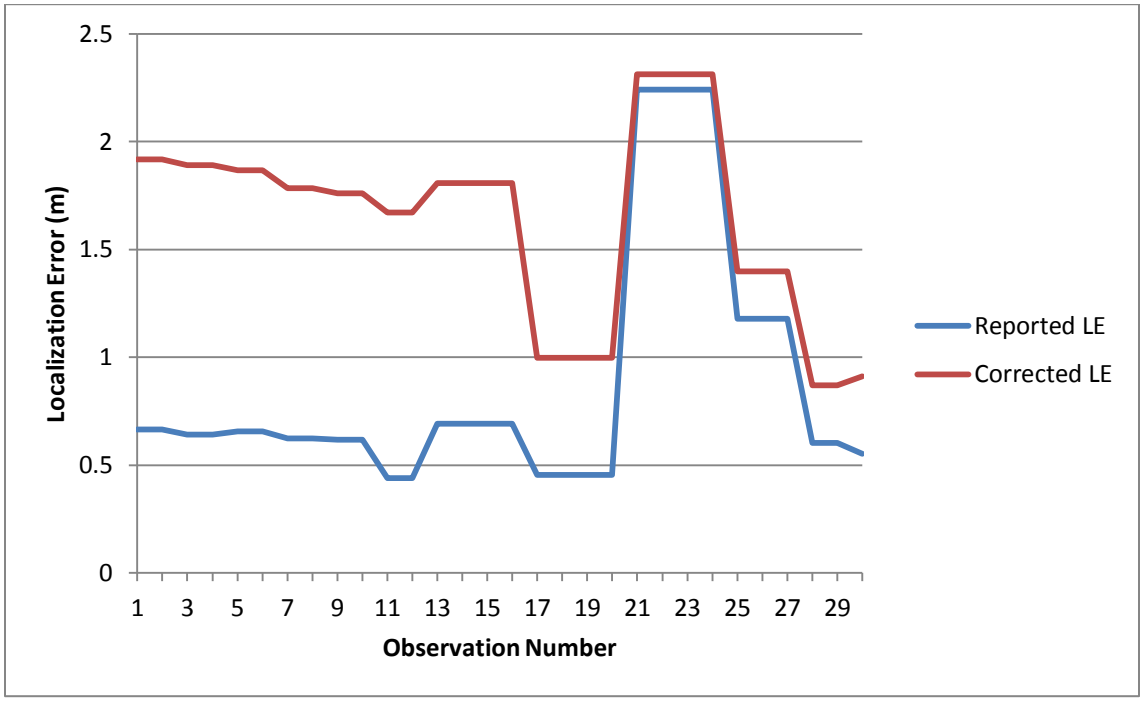

Figure 4.48: Asset 3 localization error for both the Original and Corrected Location CPC Lab

4.3.1.2 Plastics Lab

The results before and after the application of correction factor method are shown in Asset 1 (Figure 4.49), Asset 2 (Figure 4.50), and Asset 3 (Figure 4.51).

For all three tagged assets, the correction factor method produced better localization errors than that from the RFID-RTLS system. For all of the figures, the localization is reduced by 1m and generally follows the trend of the filtered value.

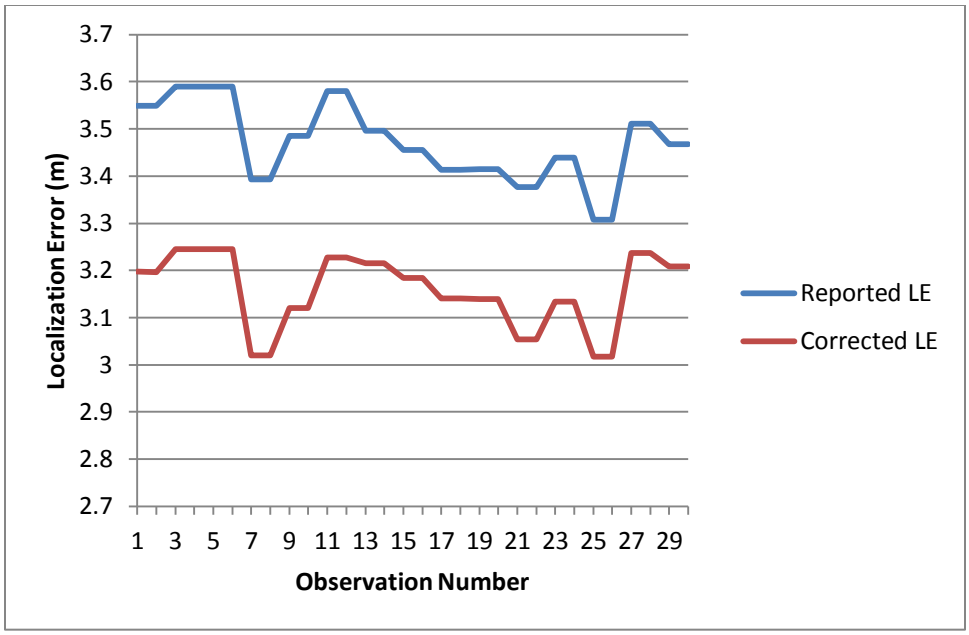

Figure 4.49: Asset 1 localization error for both the Original and Corrected Location for Plastic Lab

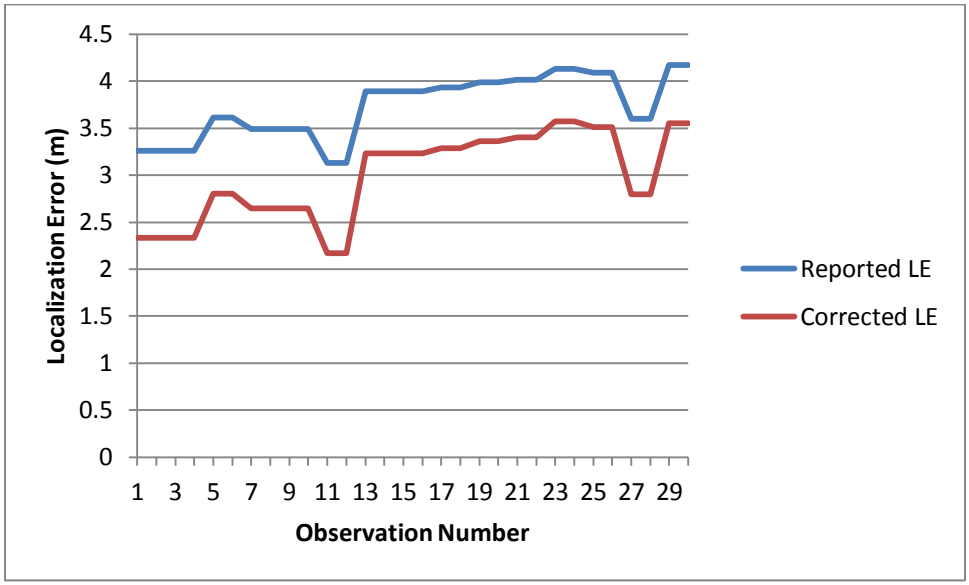

Figure 4.50: Asset 2 localization error for both the Original and Corrected Location for Plastic Lab

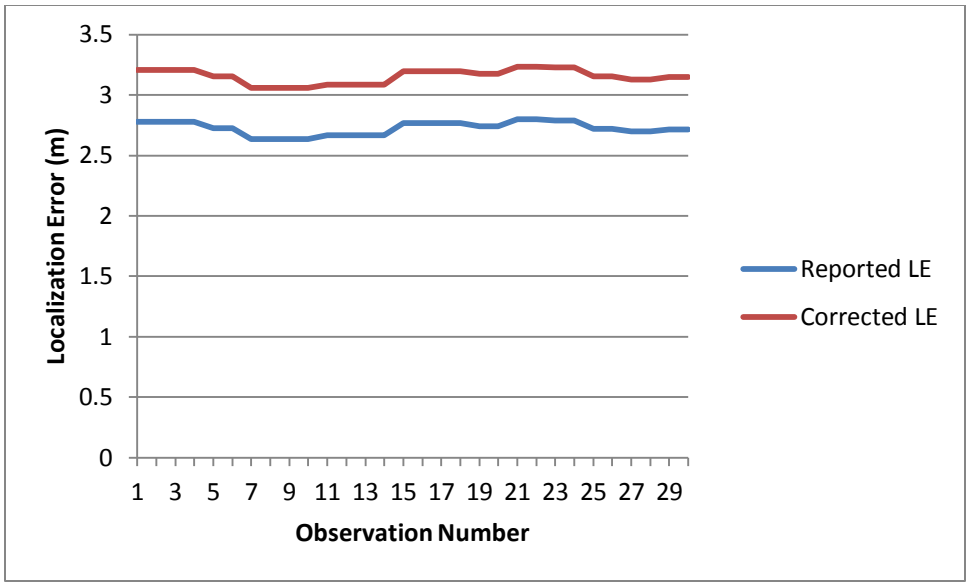

Figure 4.51: Asset 3 localization error for both the Original and Corrected Location for Plastic Lab

To compare the results with the other methods, the following graphs (Figure 4.52 and Figure 4.53) were plotted. The correction factor method was compared with two other methods which included the LANDMARC approach (Ni et. al, 2004) and Virtual tags method (Li et. al, 2012). From the figures it can be observed that the Correction Factor method improved on the accuracy of the reported location in four out of the six situations for Asset 2 in the CPCLab and Asset 3 in the Plastics lab.

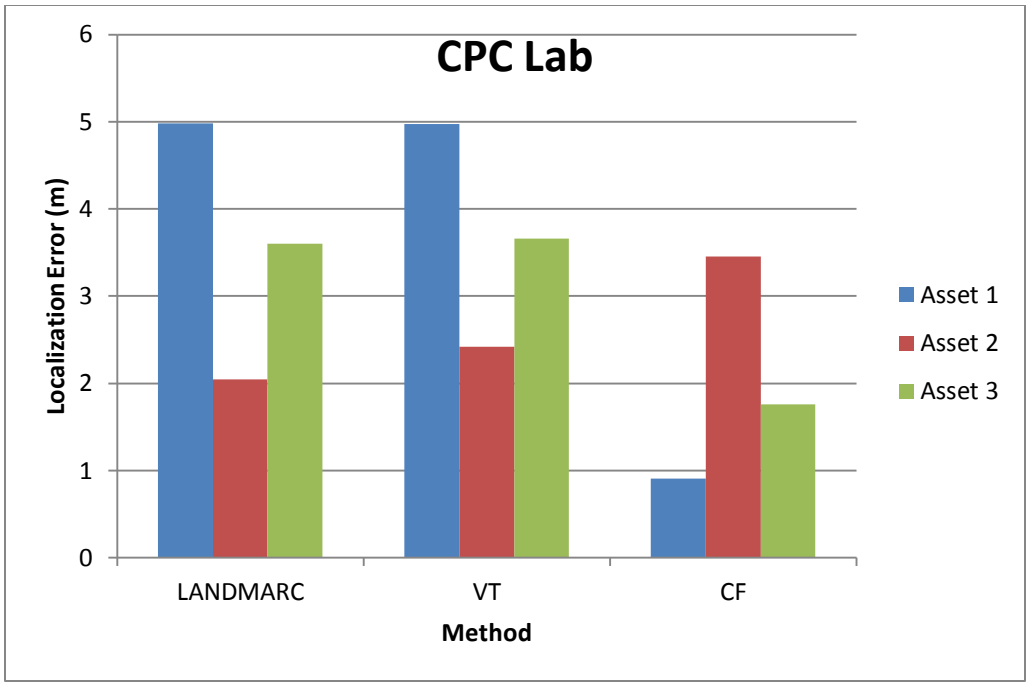

Figure 4.52: Comparative bar chart for different methods for location accuracy in CPC lab

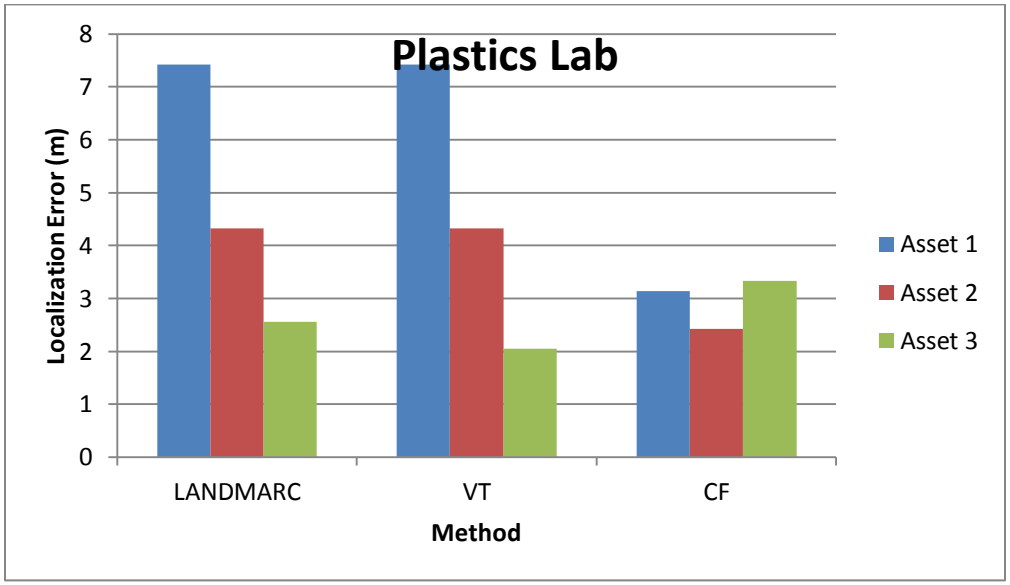

Figure 4.53: Comparative bar chart of different methods for location accuracy

4.4 Discussion and Summary

4.4.1 Prototype #1

RFID-RTLS system has potential for providing locations of tagged components within spatially mapped zones. Linking the tagged components with their virtual representations in the model offers opportunities for tracking component status without manually embedding status information in the tags. This status information is meant to be visualized in the model; however, the RFID-RTLS system suffers from multipath issues and interferences. For virtual model update, the location data from the RFID-RTLS system needs to be stable so that the status of the tagged components can be easily visualized. In-order to reduce the inaccuracies in the radio frequency signal, Kalman filter was used to filter the noise from the location data. The developed system was test indoors and outdoors to observe the performance of the system.

The developed system performed better with less localization error in the outdoor tests than in the indoor tests. For the 'on-site' mapped area, the localization error was less than 1m for the outdoor test and greater than 1m for the indoor test. In an actual construction site, the site will be large enough to accommodate the tagged materials. At a localization error of 1m, the model update will remain stable since the tagged component is still within the site boundary. Furthermore, for the storage area, it can be observed that the localization error for indoor test is also significantly higher (less than 8m) compared to outdoor (less than 1.5m) test. On an actual job site, the storage and staging areas will be of a reduced size than the actual site boundary as much more precision is required for

the storage area compared with the on-site area. In the case of tracking the installed location of tagged components, high accuracy is required. This will enable a steady up-date to the virtual model. As such, Kalman filter was applied to the resulting reported coordinates. This reduced the localization errors significantly. For example, from the indoor tests of Tag 1, the original localization error has a tolerance of ±3.4m while the kalman filter produced a localization error of ± 2.5m. Likewise for Tag 2, the original localization error has a tolerance of ±1.7m while the kalman filter produced a localization error of ± 0.91m. The effect of kalman filter on the original data was consistent with all the tag values. In-spite of the improvement in the indoor test results, the localization error for the outdoor test was much lower. For example, from the outdoor tests of Tag 1, the original localization error has a tolerance of ±0.81m while the kalman filter produced a localization error of ± 0.55m. Likewise for Tag 2, the original localization error has a tolerance of ±0.71m while the kalman filter produced a localization error of ± 0.43m. This shows that significant multipath was experienced indoors than in the outdoor environment.

Localization error in the indoor experiment was more than that achieved in the outdoor experiment. This means that the RFID-RTLS system performs better outdoors than in indoor environments. In the context of a construction site, as the jobsite becomes more crowded and congested; the accuracy of the system will greatly reduce. Testing the performance of the developed system indoors and outdoors is important, especially for partially completed sites or renovation projects. In such cases, as the tag transitions from a partially completed to full completed structure, it is important to have a stable update to

the model. The performance of the tags under these varying enclosed conditions will differ. Tolerances can be applied to such a system to enable stable update in the model.

4.4.2 Prototype #2

Prototype #2 was developed for navigating and tracking assets within the constructed facility. The system was tested in two different laboratories with varying levels of congestion and activity. A high localization error was observed from both laboratories. The CPC laboratory was less congested with less activity level compared with the Plastic laboratory. This is evident in the localization errors attained from the experiments. From Figure 4.44, it was observed that the localization error for Tag 1 is less than 3.1m and the localization error for Tag 2 is less than 1.4m. While, the localization errors for the experiment conducted in the Plastics Lab (as shown in Figure 4.45) for Tag 1 and 2 was less than 3.6m. It is important to have a reduced localization error so that the system can relay accurate location of tagged assets in the iPad GUI. In line with this, the developed Correction Factor method was applied to the reported locations. For the CPC laboratory test, the Correction Factor reduced the localization error to less than 1.5m for Asset 1. For these observations, the localization errors for Asset 2 are less than 2m. For Asset 3, results from the correction factor method was worse than that achieved from the RFID-RTLS system. A reason for this might be that the reference tag used as the nearest tag was actually further away from the Asset tag. On the contrary, applying the Correction Factor method to the Plastics Laboratory test result yielded reduced localization errors. This is illustrated in Figures 4.49, 4.50 and 4.51.

For these tests, few tags were used as reference tags. Since the Correction Factor method was applied based on the nearest tag logic, the closed reference tags to the Asset tags were at a considerable distance. Better results would have been obtained if more reference tags were used. Opportunities exist for future work on exploring the quantity of tags required for such tests. However, the Correction Factor method showed potential for reducing the errors from RFID-RTLS system tags.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Introduction

This chapter presents a general summary of the overall research, draws conclusions and makes recommendations for further research. This research focused on the application of cyber-adaptive physical systems integration between virtual models and the physical construction/constructed facility. This research presented the requirements for this cyber-adaptive physical system integration. The research also described the development of three prototype systems. The research presented the software, system architecture, and implementation of the prototype systems. The research also presents results from the implementation of these prototypes. The research concludes that cyber-adaptive physical systems integration has the potential to significantly improve access to real-time information for the construction process/resulting facility management. This chapter also includes with recommendations for future research.

5.2 General Summary

This section reviews the aim and objectives of this research and compares it with the research undertaken. The aim of the research was to investigate the applicability of the cyber-adaptive physical systems approach in enhancing integration between virtual models and the physical construction/constructed facility. The specific objectives of the research project were:

- To identify the requirements for a cyber-adaptive physical systems integration between virtual models and the physical construction;
- To investigate the most appropriate mechanisms that can be used to achieve the CaPS integration between virtual models and the physical construction/constructed facility, such that real-time data collection and aggregation can be undertaken for tracking applications;
- To develop software and systems architectures which will highlight mechanisms and integration technologies required for developing sample prototypes for cyber-adaptive physical systems applications;
- To develop and experiment with cyber-adaptive physical systems on laboratory-scale prototypes. This experiment will enable tracking the status of key components in the laboratory-scale physical prototype;
- Test the developed prototypes testing on actual construction sites/constructed facilities.

The specific tasks undertaken in this research, with respect to the research objectives, are summarized below:

- ***Objective 1: To identify the requirements for a cyber-adaptive physical systems integration between virtual models and the physical construction:***

The requirements for cyber-adaptive physical systems integration between virtual models and the physical construction was carried out through content analysis of literature. These requirements are presented in Chapter 3 of the thesis.

- ***Objective 2: To investigate the most appropriate mechanisms that can be used to achieve tight integration between virtual models and the physical construction/constructed facility, such that real-time data collection and aggregation can be undertaken for tracking applications:***

A review was undertaken of enabling technologies for various integration approaches in the construction industry. This was used to determine the most suitable technologies for enhancing CaPS integration between virtual models and the physical construction/constructed facility. The results of this review were used to determine the most effective mechanisms to enhance the proposed integration approach.

- ***Objective 3: To develop software and systems architectures which will highlight mechanisms and integration technologies required for developing sample prototypes for cyber-adaptive physical systems applications:***

Based on the review and mechanisms identified, software and system architectures were developed for three prototypes. These architectures served as a basis for developing the prototype systems. The details of these architectures are presented in Chapter 3.

- ***Objective 4: To develop and experiment with cyber-adaptive physical systems on laboratory-scale prototypes. This experiment will enable tracking the status of key components in the laboratory-scale physical prototype:***

Three systems were developed and experimented with in the laboratory using small scale laboratory prototypes. A system for integrating virtual models and the physical construction components using an RFID-RTLS system was also developed. This

system was tried indoors in the CPC Laboratory and implemented outdoors on the Western Michigan University campus. A second prototype was also developed through the integration of RFID-RTLS system and iPad for asset tracking. The third prototype was an extension of the first prototype for automated progress tracking based on earned value management. Although, this was not tested, it has potential opportunities for providing real-time progress information to help project managers with active decision making. The details of the development and implementation of the prototype systems are presented in Chapter 3 and Chapter 4.

- ***Objective 5: Test and validate the developed prototypes testing on actual construction sites***

Two of the developed systems were tested in the laboratory and the field. The laboratory results were affected by multipath effects caused by obstructions and interferences as a result multiple Wi-Fi, Bluetooth and Zigbee signals in the facility. While these tests confirmed the link between the physical laboratory prototype and its associated virtual model, and the effectiveness of the control algorithms, the interference limited the accuracy of the location information for the RTLS tags. This second prototype system was tested in two indoor locations: the CPC and Plastics Laboratory. The system also suffered from inaccuracies of the RFID-RTLS system. However, algorithms were developed to reduce the errors encountered in the system. The results of the test are presented in Chapter 5.

5.3 Conclusions

Based on the findings outlined in the preceding chapters of the thesis, the following conclusions can be drawn:

The requirements for effective cyber-adaptive physical systems integration were identified as context adaptive, having real-time capability and high level of automation. Based on these requirements, further system specific requirements for integrating the virtual models and the physical construction were investigated fortracking accurate location of tagged components, understanding on-site spatially mapped locations and updating the virtual models with the status information;

Adapting virtual models to the physical construction has potential opportunities for improving progress monitoring and control of the construction process (and/or the constructed facility);

Enabling technologies such as 3D laser scanning, photography and photogrammetry are suitable for capturing construction data and activities, and the images generated can be linked to virtual models for progress monitoring and detection of discrepancies between as-built and as-planned models. However, their support for CaPS is limited;

RFID-RTLS tags have significant advantages over standard RFID tags in facilitating the implementation of CaPS in construction. This is because of their capability for both spatial mapping and real-time location tracking;

The developed prototype systems demonstrated the applicability of cyber-adaptive physical systems approach for integrating physical components and the virtual model using the RFID-RTLS system. These systems offer opportunities for enhancing construction progress tracking and improving the lifecycle management and control of the constructed facility (or subsets thereof);

Results from the laboratory and field experiments on the 'integration of virtual models and the physical construction using RTLS system' showed that the RTLS system offers tremendous opportunities for enhancing real-time construction progress monitoring, and process control. However, the use of RTLS system is more suitable for outdoor environments where there is less interference from other Wi-Fi, Bluetooth and Zigbee signals. The presence of these signals caused a multipath effect in the real-time locations generated from the RTLS system. The use of Kalman filter for smoothing the errors or inaccuracies in the system proved effective. However, for installation status tracking, a higher degree of accuracy is required.

5.4 Contribution to Knowledge

This research contributed to knowledge in three key areas and this is presented in more detail below:

- **Definition of Use Cases for Cyber-Adaptive Physical Systems in Construction**

The existing approaches to integrating virtual models and the physical construction have potential for better visualization of the constructed facility and for

progress monitoring. However, this research has demonstrated that these have severe limitations in terms of adapting the virtual models with the physical construction components. It also demonstrated how the cyber-adaptive physical systems approach proposed in this thesis overcomes the limitations of conventional approaches and enhances construction progress monitoring, process control and asset tracking in the constructed facility.

- **Investigation of RFID-RTLS System as a Sensing Technology for Cyber-Adaptive Physical Systems Integration in Construction**

The level of automation required for CaPS integration requires tracking placement of tagged components without manual input into the tags. This level of automation also requires a means of adaptive context tracking. These requirements resulted in investigation into sensing technologies having real-time location sensing potential and spatial capability. The RTLS system was discovered as the only available technology for this CaPS integration. The location and spatial sensing potential makes the RFID-RTLS technology suitable for tracking placement and location of tagged components.

- **Development of Prototype Systems**

Three prototype systems were developed and described in Chapter 3 These prototype systems serve as early CaPS applications in the construction industry and they demonstrate potentials for integrating virtual models and the physical components using the RTLS system. This integration has specific benefits for access to real-time progress information which is essential for active decision making.

5.5 Research Limitations

There are a number of limitations to the research described in this thesis. These are highlighted below:

- Only limited field/outdoor testing of the CaPS system was undertaken. There is a need to demonstrate the practical functionality of the systems on a real construction site before robust conclusions can be drawn on the suitability of the approach to full scale construction projects. This would also help to establish the organizational, contractual and other project-specific constraints that will need to be addressed.
- Kalman filter was used to model and filter the noise from the results. In order to develop more accurate results, there is need for a more detailed study of the reported location data obtained from the RFID-RTLS system. This will help understand and develop more suitable algorithms for reducing the errors or inaccuracies in the RFID-RTLS system results.

5.6 Recommendations for Further Research and Development

The research has identified a number of areas for further research and improvements in construction industry practices. The key recommendations for further research and development are as follows:

- **Comprehensive CaPS for Construction Progress Monitoring**

It has been identified from this research that each data acquisition technology provides different levels of support for the CaPS approach. A number of researchers have

demonstrated the capability of each data acquisition technology. There is a need to investigate how each of these technologies can collectively fulfill the requirements for effective CaPS integration of virtual models and the physical construction for construction progress monitoring. This will involve identifying the type of status information that needs to be captured or sensed, the most suitable data acquisition technologies for these and how to position these devices on a full scale construction site for effectively capturing the as-built information (e.g. identifying the types of camera and laser scanner to use, and how to best position them to capture information). Also, the information from each data acquisition technology will need to be fused together to produce the desired progress information. Since this will involve the use of multiple data acquisition technologies, there is need to also investigate the value added benefit of using these technologies.

- Future Applications of CaPS in Construction

This research has investigated one of the early applications of CaPS to construction. The potentials of the CaPS approach shows that it has the capability of enhancing effective monitoring and control of construction activities. Hence, there is scope for applying this approach to other areas such as site layout management, carbon footprint analysis and workspace management.

- Development of Sensors better suited for CaPS applications in Construction

In order to develop the prototypes, this research was constrained by the available sensors. Most of the available sensors are designed to meet applications in other industry

sectors such as mechanical, automobile and health industry. There are few sensors specifically for construction applications and these sensors do not meet the level of automation required for CaPS application in construction. Having identified the requirements for CaPS for specific applications, there is need to develop sensors and other instrumentation tailored to the specific application.

5.7 Concluding Remarks

This research has shown how CaPS can be applied to construction in enhancing integration between virtual models and the physical construction/constructed facility through the development and implementation of prototype systems. The developed systems illustrate how the current construction practices can be modified and also shows the benefits of the proposed approach for the construction industry. These benefits include access to real-time progress information which will aid the construction project team in quick decision making and potentials for enhancing real-time communication. Another benefit is that the proposed approach can facilitate improvements in facility management practices by enhancing the current process of asset tracking. By demonstrating the potential of the cyber-adaptive physical systems integration approach, this research has opened the door to new CaPS applications in the construction industry.

REFERENCES

Abeid, J., Allouche, E., Arditi, D., & Hayman, M. (2003). PHOTO-NET II: a computer-based monitoring system applied to project management. Automation in construction, 12(5), 603-616.

Ahmadi, H., Abdelzaher, T., Han, J., Pham, N., & Ganti, R. K. (2011, April). The sparse regression cube: A reliable modeling technique for open cyber-physical systems. In Cyber-Physical Systems (ICCPS), 2011 IEEE/ACM International Conference on (pp. 87-96). IEEE.

Akanmu, A. A., Anumba, C. J., & Messner, J. I. (2012). An RTLS-Based Approach to Cyber-Physical Systems Integration in Design and Construction. International Journal of Distributed Sensor Networks, 2012.

Akinci, B., & Anumba, C. (2008). Editorial-Sensors in Construction and Infrastructure Management.

Akinci, B., Boukamp, F., Gordon, C., Huber, D., Lyons, C., & Park, K. (2006). A formalism for utilization of sensor systems and integrated project models for active construction quality control. Automation in Construction, 15(2), 124-138.

Azhar, S., Hein, M., & Sketo, B. (2008, April). Building information modeling (BIM): Benefits, risks and challenges. In *Proceedings of the 44th ASC Annual Conference* (pp. 2-5).

Azimi, R., Lee, S., AbouRizk, S. M., & Alvanchi, A. (2011). A framework for an automated and integrated project monitoring and control system for steel fabrication projects. Automation in Construction, 20(1), 88-97.

Aziz, O., Lo, B., King, R., Darzi, A., & Yang, G. Z. (2006, April). Pervasive body sensor network: an approach to monitoring the post-operative surgical patient. In Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on (pp. 4-pp). IEEE.

Bekkali, A., Sanson, H., & Matsumoto, M. (2007, October). RFID indoor positioning based on probabilistic RFID map and kalman filtering. In Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on (pp. 21-21). IEEE.

Bhave, A., Garlan, D., Krogh, B., Rajhans, A., & Schmerl, B. (2010, May). Augmenting software architectures with physical components. In Proc. of the Embedded Real Time Software and Systems Conf.(ERTS2 2010) (pp. 19-21).

Bosche, F., & Haas, C. T. (2008). Automated retrieval of 3D CAD model objects in construction range images. Automation in Construction, 17(4), 499-512.

Brilakis, I., Soibelman, L., & Shinagawa, Y. (2005). Material-based construction site image retrieval. Journal of computing in civil engineering, 19(4), 341-355.

Build, E. R. A. (2006). Review of current state of radio frequency identification (RFID) technology, its use and potential future use in construction.

Chen, Y., & Kamara, J. M. (2008). Using mobile computing for construction site information management. Engineering, construction and architectural management, 15(1), 7-20.

Cheng, M. Y., & Chen, J. C. (2002). Integrating barcode and GIS for monitoring construction progress. Automation in Construction, 11(1), 23-33.

Chin, S., Yoon, S., Kim, Y., Ryu, J., Choi, C., and Cho, C. (2005). "Real time 4D CAD+RFID for project progress management." Proceedings, Construction Research Congress 2005, ASCE, Reston, VA, 168–172.

Chin, S., Yoon, S., Choi, C., & Cho, C. (2008). RFID+ 4 D CAD for Progress Management of Structural Steel Works in High-Rise Buildings. Journal of Computing in Civil Engineering, 22(2), 74-89.

Dickinson, J. K., Pardasani, A., Ahamed, S. S., & Kruithof, S. (2009, June). A survey of automation technology for realising as-built models of services. In 1st International Conference on Improving Construction and Use Through Integrated Design Solutions, CIB IDS (pp. 365-381).

Domdouzis, K. (2007). Applications of wireless sensor technologies in construction.

El-Omari, S., & Moselhi, O. (2008). Integrating 3D laser scanning and photogrammetry for progress measurement of construction work. Automation in construction, 18(1), 1-9.

El-Omari, S., & Moselhi, O. (2009). Data acquisition from construction sites for tracking purposes. Engineering, Construction and Architectural Management, 16(5), 490-503.

El-Omari, S., & Moselhi, O. (2011). Integrating automated data acquisition technologies for progress reporting of construction projects. Automation in Construction, 20(6), 699-705.

Fard, M. G., Staub-French, S., Po, B., & Tory, M. (2006). Requirements for a mobile interactive workspace to support design development and coordination. In Joint International Conference on Computing and Decision Making in Civil and Building Engineering (pp. 3587-3596).

Jaselskis, E. J., & El-Misalami, T. (2003). Implementing radio frequency identification in the construction process. Journal of Construction Engineering and Management, 129(6), 680-688.

Memon, Z. A., Majid, M. Z. A., & Mustaffar, M. (2005, July). An automatic project progress monitoring model by integrating AutoCAD and digital photos. In ASCE International Conference on Computing in Civil Engineering, Cancun, Mexico.

Gajamani, G. K., & Varghese, K. (2007). Automated project schedule and inventory monitoring using RFID. Proceedings ISARC.

Golparvar-Fard, M., Peña-Mora, F., & Savarese, S. (2009). Application of D4AR–A 4-Dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication.

Goodrum, P. M., McLaren, M. A., & Durfee, A. (2006). The application of active radio frequency identification technology for tool tracking on construction job sites. Automation in Construction, 15(3), 292-302.

Jaselskis, E., Cackler, E., Walters, R., Zhang, J., and Kaewmoracharoen, M. (2006), "Using scanning lasers for real-time pavement thickness measurement", CTRE Project 05-205, National Concrete Pavement Tech Center, Iowa State University.

Kasim, N. (2008). Improving materials management on construction projects (Doctoral dissertation, Loughborough University).

Kim, Y., Shin, H., & Cha, H. (2008, April). Y-mac: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In Proceedings of the 7th international conference on Information processing in sensor networks (pp. 53-63). IEEE Computer Society.

Kim, C., Hyoungkwan K., and Yeonjong, J. (2009), "Bridge Construction Progress Monitoring Using Image Analysis", 26th International Symposium on Automation and Robotics in Construction (ISARC 2009).

Kiziltas, S., & Akinci, B. (2009). Contextual information requirements of cost estimators from past construction projects. Journal of Construction Engineering and Management, 135(9), 841-852.

Lee, S., & Peña-Mora, F. (2006). Visualization of construction progress monitoring. In Proc. of Joint Int. Conf. on Computing and Decision Making in Civil & Building Engrg (pp. 2527-2533).

Leung, S. W., Mak, S., & Lee, B. L. (2008). Using a real-time integrated communication system to monitor the progress and quality of construction works. Automation in construction, 17(6), 749-757.

Li, H., Chen, Z., Yong, L., & Kong, S. C. (2005). Application of integrated GPS and GIS technology for reducing construction waste and improving construction efficiency. Automation in Construction, 14(3), 323-331.

Li, N., Li, S., Becerik-Gerber, B., & Calis, G. (2011). Deployment strategies and performance evaluation of a virtual-tag-enabled indoor location sensing approach. Journal of Computing in Civil Engineering, 26(5), 574-583.

Li, X., Falcon, R., Nayak, A., & Stojmenovic, I. (2012). Servicing wireless sensor networks by mobile robots. Communications Magazine, IEEE, 50(7), 147-154.

Motamedi, A., & Hammad, A. (2009). Lifecycle management of facilities components using radio frequency identification and building information model.

Navon, R., & Sacks, R. (2007). Assessing research issues in automated project performance control (APPC). Automation in Construction, 16(4), 474-484.

Ni, L. M., Liu, Y., Lau, Y. C., & Patil, A. P. (2004). LANDMARC: indoor location sensing using active RFID. Wireless networks, 10(6), 701-710.

Oglesby, C. H., Parker, H. W., & Howell, G. A. (1989). Productivity improvement in construction (pp. 176-180). New York: McGraw-Hill.

Oloufa, A. A., Ikeda, M., & Oda, H. (2003). Situational awareness of construction equipment using GPS, wireless and web technologies. Automation in Construction, 12(6), 737-748.

Rohrig, C., & Muller, M. (2009, October). Indoor location tracking in non-line-of-sight environments using a IEEE 802.15. 4a wireless network. In Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on (pp. 552-557). IEEE.

Sørensen K. B., Christiansson P., Svidt K., Jacobsen K., Simoni T. (2008). "Towards Linking Virtual Models with Physical Objects in Construction using RFID - Review of Ontologies." Proceedings of the CIB-W78 25th International Conference on Information Technology in Construction (Rischmoller L., editor). Santiago de Chile, July 15-17 2008, 418-428.

Shen, X., Chen, W., & Lu, M. (2008). Wireless sensor networks for resources tracking at building construction sites. Tsinghua Science & Technology, 13, 78-83.

Song J., Hass C. T., Caldas C., Ergen E. and Akinci B. (2005). "Automating the task of tracking the delivery and receipt of fabricated pipe spools in industrial projects." Automation in Construction.

Song, J., Lee, N., Yoon, S., and Kwon, S. (2007). "Material Tracker for Construction Logistics." Proceedings of ISARC, 63-67.

Su, Y. Y., Hashash, Y. M. A., & Liu, L. Y. (2006). Integration of construction as-built data via laser scanning with geotechnical monitoring of urban excavation. Journal of construction engineering and management, 132(12), 1234-1241.

Teizer, J., Kim, C., Haas, C. T., Liapi, K. A., & Caldas, C. H. (2005). Framework for real-time three-dimensional modeling of infrastructure. Transportation Research Record: Journal of the Transportation Research Board, 1913(1), 177-186.

Turkan, Y., Bosché, F., Haas, C. T., & Haas, R. (2012). Toward Automated Earned Value Tracking Using 3D Imaging Tools. Journal of Construction Engineering and Management.

Verdone, R., Fabbri, F., & Buratti, C. (2008, September). Area throughput for CSMA based wireless sensor networks. In Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on (pp. 1-6). IEEE.

Wang, L. C. (2008). "Enhancing construction quality inspection and management using RFID technology." Automation in Construction, 17, 467–479.

Welch, G., & Bishop, G. (1995). An introduction to the Kalman filter.

Wu, F. J., Kao, Y. F., & Tseng, Y. C. (2011). From wireless sensor networks towards cyber physical systems. Pervasive and Mobile Computing, 7(4), 397-413.

Xia, F., Vinel, A., Gao, R., Wang, L., & Qiu, T. (2011). Evaluating ieee 802.15. 4 for cyber-physical systems. EURASIP Journal on Wireless Communications and Networking, 2011(1), 596397.

Xuesong, S., Wu, C., Ming, L. (2008). "Wireless Sensor Networks for Resources Tracking at Building Construction Sites." Tsinghua Science and Technology, 13, 78-83.

Yabuki, N., Shimada, Y. and Tomita, K. (2002). "An On-Site Inspection Support System Using Radio Frequency Identification Tags and Personal Digital Assistants." Proceedings of the 2002 CIB w78 Conference, Aarhus School of Architecture, 12–14 June 2002, International Council for Research and Innovation in Building and Construction.

Zhao, Y., Liu, Y., & Ni, L. M. (2007, September). VIRE: Active RFID-based localization using virtual reference elimination. In Parallel Processing, 2007. ICPP 2007. International Conference on (pp. 56-56). IEEE.

Appendix A

RTLS-DB Code

APPENDIX A – RFID-DB CODE

Modified code in CSL RTLS Program v2.0.3.89,

In visual studio, access FrmMain.cs

Add following code before namespace:

```
using System.Linq;
using RTLSProgram.MultipathCharacterization;
using MySql.Data;
using MySql.Data.MySqlClient;
```

Modify tagpositionarrive function:

```
public void TagPositionArrive(PositionNotifyPacket pp)
            {
#if false
            MessageBox.Show("false")
                    if (pp.device is ISlaveDevice)
                    {
                            Device       master       =       (pp.device       as
ISlaveDevice).GetMaster();
                            if (master != null && master is ICellMember)
                                    if      (pp.FromCell      ==      (master      as
ICellMember).GetCell())
                                    {

       ((IRangingDevice)pp.device).SavePosition(pp);
                                            foreach (TagUserData tud in rtls.TagList)
                                            {
                                                    if  (tud  !=  null  &&  tud.tag  ==
pp.device)
                                                    {
                                                            if (pp.position != null)
                                                            {

       tud.timeUpdatePosition = pp.raw.Time;

                                                                    if  (pp.FromCell  ==
tud.tag.GetCell())
                                                                    {

       tud.used_distance.Clear();
                                                                            foreach
(ReferencePosition rp in pp.position.used_ref)
```

```
                                                                {
        foreach (RangingFilter.DeviceDistance dd in rp.distance)
                                                                    {
        if (!tud.used_distance.Contains(dd))
        tud.used_distance.Add(dd);
                                                                    }
                                                                }
                                                            }
                                                            foreach    (CellPanel
panel in cellPanel_List)
                                                            {
                                                                if
(panel.currentCell != null)
                                                                    if
(panel.currentCell.masterAnchor == master)
                                                                    {
        //panel.TagPositionArrive(pp);
        panel.ltTags_invalidate = true;
        panel.pnlMap_invalidate = true;
        panel.CheckInvalidate();
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
 #endif
                                        if (pp.device is Tag)
                                        {
                                            if (pp.device is ICellMember)
                                                if     (pp.FromCell     ==     (pp.device     as
ICellMember).GetCell())
                                                {
                                                    if(pp.source    is    TrilaterationCC    &&
pp.source.id == RTLS.LocationEngineId_TOA)
        ((IPositioningDevice)pp.device).SavePosition(pp);
                                                    TagUserData             tud             =
rtls.GetTagUserData(pp.device.ID);
                                                //Hammad addition
                                                CellUserDataV02 cell = null;
                                                if             (rtls.Config.CellDisplayMode             ==
SystemConfigV02.CellPanelDisplayMode.Global)
                                                    cell = UserCellSelection();
                                                else
                                                {
```

```
                            if (activeCellPanelIndex < cellPanel_List.Count)
                                cell                                                =
cellPanel_List[activeCellPanelIndex].currentCell;
                            }

                            if
(cell.cellcfg.regional_correction_table.region_list.Count == 0)
                            {


                                if (DBExist(pp.device.ID))
                                {
                                    DBUpdate(pp.device.ID, "");
                                    DBAdd("taglogs", pp.device.ID, "");
                                }
                                else
                                {
                                    DBAdd("taginfo", pp.device.ID, "");
                                    DBAdd("taglogs", pp.device.ID, "");
                                }
                            }


                            foreach          (CorrectionRegion          k          in
cell.cellcfg.regional_correction_table.region_list.ToArray())
                            {

                                /*if          ((Convert.ToDouble(k.box.X0)          <=
Convert.ToDouble(tud.tag.GetPosition().X)    &&    Convert.ToDouble(k.box.X1)    >=
Convert.ToDouble(tud.tag.GetPosition().X))    &&    (Convert.ToDouble(k.box.Y0)    <=
Convert.ToDouble(tud.tag.GetPosition().Y)    &&    Convert.ToDouble(k.box.Y1)    >=
Convert.ToDouble(tud.tag.GetPosition().Y)))
                                {
                                    if (DBExist(pp.device.ID))
                                    {
                                        DBUpdate(pp.device.ID, k.Name);
                                        DBAdd("taglogs", pp.device.ID, k.Name);
                                    }
                                    else
                                    {
                                        DBAdd("taginfo", pp.device.ID, k.Name);
                                        DBAdd("taglogs", pp.device.ID, k.Name);
                                    }
                                }*/
                            }

                            //End hammad addition



                                        if(tud != null)
 //                                     foreach (TagUserData tud in rtls.TagList)
                                        {

 //                                           if  (tud  !=  null  &&  tud.tag  ==
pp.device)

                                            {
```

```
#if NanoLoc
                                                        if      (pp.source      is
Nanotron.nanoLOC.AVR_DK.LocationEngine)
                                                        {
      tud.last_nanoloc_position_packet = pp;
                                                              if  (pp.position  !=
null)
      tud.nanoloc_position = pp.position;
                                                        }
#endif
                                                        if (pp.position != null)
                                                        {
                                                              if   (pp.source   is
TrilaterationCC && pp.source.id == RTLS.LocationEngineId_TOA)
                                                              {
      tud.lastPosition = pp.raw.Time;
                                                                    if   (tud.cell
!= null)
      tud.lastPositionCycle = tud.cell.cycleCount;
                                                                    if
(pp.FromCell == tud.tag.GetCell())
                                                                    {
      tud.used_distance.Clear();
                                                                          if
(pp.position.used_ref != null)
      foreach (ReferencePosition_Distance rp in pp.position.used_ref)
      {
      foreach (RangingFilter.DeviceDistance dd in rp.distance)
      {
            if (!tud.used_distance.Contains(dd))
                  tud.used_distance.Add(dd);
      }
      }
                                                                    }
                                                              }
                                                        if
(rtls.Config.CellDisplayMode == SystemConfigV02.CellPanelDisplayMode.Global)
                                                        {
      cellGroupPanel.ltTags_invalidate = true;
      cellGroupPanel.pnlMap_invalidate = true;
```

```
        cellGroupPanel.CheckInvalidate();
                                                                }
                                                                else
                                                                {
                                                                    foreach
(CellPanel panel in cellPanel_List)
                                                                    {
                                                                        if
(panel.currentCell    !=    null    &&    panel.currentCell.cell    !=    null    &&
panel.currentCell.cell == tud.tag.GetCell())
                                                                        {

        //panel.TagPositionArrive(pp);

        panel.ltTags_invalidate = true;

        panel.pnlMap_invalidate = true;

        panel.CheckInvalidate();
                                                                        }
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
```

Add the following functions

```
public bool DBAdd(string table, string TagID, string Location)
        {
            string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";
            MySqlConnection conn = null;

            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                DateTime dt = DateTime.Now;

                //NEW COMMAND
                cmd.CommandText  =  "INSERT  INTO  "+table+"(ID,  TagID,  Location,
DateTime) VALUES(NULL, @TagID, @Location, @DateTime)";
                cmd.Prepare();
                cmd.Parameters.AddWithValue("@TagID", TagID);
```

```
                cmd.Parameters.AddWithValue("@Location", Location);

                //get date and time
                DateTime dt1 = DateTime.Now;
                string dtActual1 = dt1.ToString("yyyy-MM-dd HH:mm:ss");
                cmd.Parameters.AddWithValue("@DateTime", dtActual1);
                cmd.ExecuteNonQuery();
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
                return false;

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }

            }
            return true;

        }

        public bool DBExist(string TagID)
        {
            string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;


                cmd.CommandText = "SELECT * from taginfo WHERE TagID='" + TagID +
"'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                // dr.Read();
                //textbox contain the id or name which you want to check in table
                if (dr.Read())
                {
                    return true;
                }
                else
                {
                    return false;
                }
```

```
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return true;
        }

       public bool DBUpdate(string TagID, string Location)
       {
           string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";

           MySqlConnection conn = null;
           MySqlConnection conn2 = null;

           try
           {
               conn = new MySqlConnection(cs);
               conn.Open();

               MySqlCommand cmd = new MySqlCommand();
               cmd.Connection = conn;

               cmd.CommandText = "SELECT * from taginfo WHERE TagID='" + TagID +
"'";
               cmd.CommandType = CommandType.Text;

               cmd.Connection = conn;

               MySqlDataReader dr;
               dr = cmd.ExecuteReader();

               //textbox contain the id or name which you want to check in table
               if (dr.Read())
               {
                 try
                 {
                     conn2 = new MySqlConnection(cs);
                     conn2.Open();

                     MySqlCommand cmd2 = new MySqlCommand();
                     cmd2.Connection = conn2;
                     DateTime dt = DateTime.Now;
                     cmd2.CommandText = "UPDATE `taginfo` SET `DateTime`='" +
dt.ToString("yyyy-MM-dd HH:mm:ss") + "', `Location` = '"+ Location +"' WHERE
`ID`='" + dr[0] + "'";
                     cmd2.Prepare();
```

```
                cmd2.ExecuteNonQuery();
                return true;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
                return false;

            }
            finally
            {
                if (conn2 != null)
                {
                    conn2.Close();
                }
            }
        }
    }
    catch (MySqlException ex)
      {
          MessageBox.Show("Error:" + ex.ToString());
        return false;

      }
      finally
      {
          if (conn != null)
          {
              conn.Close();
          }
      }
    return true;

}
```

Appendix B

DBSniffer Code

APPENDIX B – DBSNIFFER CODE

Create a new windows form in VisualStudio and code as follows,

Create ‘Examiner.cs’:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows.Forms;

//Add two new namespaces
using Autodesk.Navisworks.Api;
using Autodesk.Navisworks.Api.Plugins;

namespace Examiner
{
   [PluginAttribute("Examiner.Examiner",
                    "SHR",
                    ToolTip = "DB Sniffer for progress monitoring.",
                    DisplayName = "DB Sniffer")]

   public class Examiner : AddInPlugin
   {
      public override int Execute(params string[] parameters)
      {
          SearchForm searchForm = new SearchForm();
          searchForm.Show();
         return 0;
      }
   }
}
```

Create ‘SearchForm.cs’

SearchFrom.cs

```
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Data;
using System.Net;
using System.Linq;
```

```
using System.Text;
using MySql.Data;
using MySql.Data.MySqlClient;
using Autodesk.Navisworks.Api;
using Autodesk.Navisworks.Api.Plugins;
using Autodesk.Navisworks.Api.Controls;
using ComApi = Autodesk.Navisworks.Api.Interop.ComApi;
using ComApiBridge = Autodesk.Navisworks.Api.ComApi;
using Autodesk.Navisworks.Api.Interop.ComApi;


namespace Examiner
{
   public partial class SearchForm : Form
   {
       bool runScript = false;
       string                                cs                                =
"server=localhost;userid=root;password=hammad;database=rtls";
       MySqlConnection conn = null;
       MySqlCommand cmd = new MySqlCommand();
       MySqlDataReader dr;


      public SearchForm()
      {
         InitializeComponent();
      }

      public void startend_Click(object sender, EventArgs e)
      {
          if (runScript)
          {
              runScript = false;

MessageBox.Show(Autodesk.Navisworks.Api.Application.Gui.MainWindow, "Stopped");
              timer.Stop();
          }
          else
          {
              runScript = true;

MessageBox.Show(Autodesk.Navisworks.Api.Application.Gui.MainWindow, "Started");
              timer.Start();

              try
              {
                  conn = new MySqlConnection(cs);
                  conn.Open();
               }
               catch (MySqlException ex)
               {
                   MessageBox.Show("Error:" + ex.ToString());

               }
          }
```

```
            }

            public void timer_Tick(object sender, EventArgs e)
            {
                if (runScript)
                {
                    try
                    {
                        cmd.CommandText = "SELECT * FROM rtls.taginfo ORDER BY
`DateTime` ASC";
                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        dr = cmd.ExecuteReader();
                        //dr.Read();
                        DataTable dt = new DataTable();
                        dt.Load(dr);
                        foreach (DataRow row in dt.Rows)
                        {
                            ChangeColor(row[1].ToString(), row[3].ToString());
                            //MessageBox.Show(row[0].ToString()+",
"+row[1].ToString()+", "+row[2].ToString()+", "+row[3].ToString());
                        }

                    }
                    catch (MySqlException ex)
                    {
                        MessageBox.Show("Error:" + ex.ToString());

                    }


                }
            }

            public void ChangeColor(string ObjectName, string Location)
            {
                try
                {
                    IEnumerable<ModelItem> items =

Autodesk.Navisworks.Api.Application.ActiveDocument.Models.RootItemDescendantsAndSe
lf.
                      Where(x =>
                          //Search for ClassDisplayName matching the text, regardless
of upper/lower case
                           (x.DisplayName.ToLower() == ObjectName.ToLower())

                            );
                    //Select the items in the model that are contained in the
collection

Autodesk.Navisworks.Api.Application.ActiveDocument.CurrentSelection.CopyFrom(items
);

                    //Change the color of the selected item
                    if (ObjectName == Location)
```

```
                    {

Autodesk.Navisworks.Api.Application.ActiveDocument.Models.OverridePermanentColor(i
tems, Autodesk.Navisworks.Api.Color.Green);
                    }
                    else if (Location == "")
                    {

Autodesk.Navisworks.Api.Application.ActiveDocument.Models.OverridePermanentColor(i
tems, Autodesk.Navisworks.Api.Color.Red);
                    }
                    else
                    {

Autodesk.Navisworks.Api.Application.ActiveDocument.Models.OverridePermanentColor(i
tems, Autodesk.Navisworks.Api.Color.White);
                    }


                    //Deselect all

Autodesk.Navisworks.Api.Application.ActiveDocument.CurrentSelection.Clear();

                }

                catch (Exception ex)
                {
                    MessageBox.Show("Error in Changing Object Color");
                }


            }




        }
    }
```

Create ‘SearchFormDesigne.cs’:

```
namespace Examiner
{
   partial class SearchForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
```

```
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.startend = new System.Windows.Forms.Button();
            this.timer = new System.Windows.Forms.Timer(this.components);
            this.SuspendLayout();
            //
            // startend
            //
            this.startend.Font = new System.Drawing.Font("Microsoft Sans Serif",
18F,      System.Drawing.FontStyle.Regular,      System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.startend.Location = new System.Drawing.Point(77, 20);
            this.startend.Name = "startend";
            this.startend.Size = new System.Drawing.Size(222, 58);
            this.startend.TabIndex = 0;
            this.startend.Text = "Start / Stop";
            this.startend.UseVisualStyleBackColor = true;
            this.startend.Click += new System.EventHandler(this.startend_Click);
            //
            // timer
            //
            this.timer.Interval = 1000;
            this.timer.Tick += new System.EventHandler(this.timer_Tick);
            //
            // SearchForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(387, 102);
            this.Controls.Add(this.startend);
            this.Name = "SearchForm";
            this.Text = "DB Sniffer Plugin";
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Button startend;
        private System.Windows.Forms.Timer timer;
```

```
    }
}
```

Appendix C

RTLS-DB-EVM Code

APPENDIX C – RTLS-DB-EVM CODE

Modified code in CSL RTLS Program v2.0.3.89,

In visual studio, accessing FrmMain.cs

Add following code before namespace

```
using System.Linq;
using RTLSProgram.MultipathCharacterization;
using MySql.Data;
using MySql.Data.MySqlClient;
```

Modify tagpositionarrive function

```
public void TagPositionArrive(PositionNotifyPacket pp)
            {
#if false
           MessageBox.Show("false")
                   if (pp.device is ISlaveDevice)
                   {
                          Device       master       =       (pp.device       as
ISlaveDevice).GetMaster();
                          if (master != null && master is ICellMember)
                                 if      (pp.FromCell      ==      (master      as
ICellMember).GetCell())
                                 {

       ((IRangingDevice)pp.device).SavePosition(pp);
                                        foreach (TagUserData tud in rtls.TagList)
                                        {
                                               if  (tud  !=  null  &&  tud.tag  ==
pp.device)
                                               {
                                                      if (pp.position != null)
                                                      {

       tud.timeUpdatePosition = pp.raw.Time;

                                                             if   (pp.FromCell   ==
tud.tag.GetCell())
                                                             {

       tud.used_distance.Clear();
```

```
                                                                    foreach
(ReferencePosition rp in pp.position.used_ref)
                                                                    {
       foreach (RangingFilter.DeviceDistance dd in rp.distance)
                                                                           {
       if (!tud.used_distance.Contains(dd))
       tud.used_distance.Add(dd);
                                                                           }
                                                                    }
                                                             }
                                                             foreach    (CellPanel
panel in cellPanel_List)
                                                             {
                                                                    if
(panel.currentCell != null)
                                                                           if
(panel.currentCell.masterAnchor == master)
                                                                           {
       //panel.TagPositionArrive(pp);
       panel.ltTags_invalidate = true;
       panel.pnlMap_invalidate = true;
       panel.CheckInvalidate();
                                                                           }
                                                             }
                                                       }
                                                 }
                                           }
                                     }
                              }
#endif
                              if (pp.device is Tag)
                              {
                                     if (pp.device is ICellMember)
                                           if     (pp.FromCell     ==     (pp.device     as
ICellMember).GetCell())
                                           {
                                                  if(pp.source    is    TrilaterationCC    &&
pp.source.id == RTLS.LocationEngineId_TOA)
       ((IPositioningDevice)pp.device).SavePosition(pp);
                                                  TagUserData              tud              =
rtls.GetTagUserData(pp.device.ID);
                                     //Hammad addition
                                     // checking cells
                                     CellUserDataV02 cell = null;
```

```
                        if              (rtls.Config.CellDisplayMode              ==
SystemConfigV02.CellPanelDisplayMode.Global)
                            cell = UserCellSelection();
                        else
                        {
                            if (activeCellPanelIndex < cellPanel_List.Count)
                                cell                                                =
cellPanel_List[activeCellPanelIndex].currentCell;
                        }

                        if
(cell.cellcfg.regional_correction_table.region_list.Count == 0)
                        {
                            if (DBExist(pp.device.ID))
                            {
                                DBUpdate(pp.device.ID, "");
                                DBAdd("taglogs",  pp.device.ID,  "",  "",  "",  "",
"");
                            }
                            else
                            {
                                DBAdd("taginfo",  pp.device.ID,  "",  "",  "",  "",
"");
                                DBAdd("taglogs",  pp.device.ID,  "",  "",  "",  "",
"");
                            }
                        }

                        foreach           (CorrectionRegion           k           in
cell.cellcfg.regional_correction_table.region_list.ToArray())
                        {
                            //MessageBox.Show("X0:" + Convert.ToDouble(k.box.X0) +
" ,X1:" + Convert.ToDouble(k.box.X1) + "\nY0:" + Convert.ToDouble(k.box.Y0) + "
,Y1:"        +        Convert.ToDouble(k.box.Y1)        +        "\nX:"        +
Convert.ToDouble(tud.tag.GetPosition().X)        +         "         ,Y:"        +
Convert.ToDouble(tud.tag.GetPosition().Y));
                            if            (Convert.ToDouble(k.box.X0)            <=
Convert.ToDouble(tud.tag.GetPosition().X)    &&    Convert.ToDouble(k.box.X1)    >=
Convert.ToDouble(tud.tag.GetPosition().X)    &&    Convert.ToDouble(k.box.Y0)    <=
Convert.ToDouble(tud.tag.GetPosition().Y)    &&    Convert.ToDouble(k.box.Y1)    >=
Convert.ToDouble(tud.tag.GetPosition().Y))
                            {
                                //MessageBox.Show("IF");

                                if (DBExist(pp.device.ID))
                                {
                                    DBUpdate(pp.device.ID, k.Name);
                                    DBAdd("taglogs", pp.device.ID, k.Name, "", "",
"", "");
                                }
                                else
                                {
                                    DBAdd("taginfo", pp.device.ID, k.Name, "", "",
"", "");
                                    DBAdd("taglogs", pp.device.ID, k.Name, "", "",
"", "");
```

```
                                }

                                //conditions for values
                                //material earned value

                                if      (pp.device.ID      ==      k.Name      &&
DBCheckIDType(pp.device.ID)             ==             "Material"             &&
!CompareDates(DBCheckActivityDate(pp.device.ID, "StartDate")))
                                {
                                    if                (!DBValueExist(pp.device.ID,
DBCheckIDAssignedActivity(pp.device.ID), DBCheckIDType(pp.device.ID)))
                                    {
                                        DBAdd("ValueTable",    pp.device.ID,    "",
DBCheckIDType(pp.device.ID),              DBCheckIDValue(pp.device.ID).ToString(),
DBCheckIDAssignedActivity(pp.device.ID).ToString(),  DBCheckIDDesc(pp.device.ID)  +
" is installed");
                                    }
                                }
                                if        (k.Name        ==        "Labor"        &&
DBCheckIDType(pp.device.ID)             ==               "Labor"               &&
!CompareDates(DBCheckActivityDate(pp.device.ID,          "StartDate"))         &&
CompareDates(DBCheckActivityDate(pp.device.ID, "FinishDate")))
                                {
                                    //MessageBox.Show("Labor started");
                                    if (startLaborTime == DateTime.MinValue)
                                    {
                                        //MessageBox.Show("Labor Time started");
                                        startLaborTime = DateTime.Now;
                                        string format = "HH:mm:ss";
                                        startLaborTime.ToString(format);
                                    }

                                }
                                //MessageBox.Show("ID:" + pp.device.ID + ", Area:"
+ k.Name + ", IDType:" + DBCheckIDType(pp.device.ID));
                                //Equipment earned value
                                if       (k.Name       ==       "Storage"       &&
DBCheckIDType(pp.device.ID)             ==             "Equipment"             &&
!CompareDates(DBCheckActivityDate(pp.device.ID,          "StartDate"))         &&
CompareDates(DBCheckActivityDate(pp.device.ID,          "FinishDate"))         &&
!startEquipmentTimeNote)
                                {
                                    //MessageBox.Show("Equipment started");
                                    startEquipmentTimeNote = true;
                                    //MessageBox.Show("Equipment time start");
                                    startEquipmentTime = DateTime.Now;
                                    string format = "HH:mm:ss";
                                    startEquipmentTime.ToString(format);
                                }

                                //equipment time stop
                                if  (startEquipmentTime  !=  DateTime.MinValue  &&
k.Name    ==    "Labor"    &&    DBCheckIDType(pp.device.ID)    ==    "Equipment"    &&
!CompareDates(DBCheckActivityDate(pp.device.ID,          "StartDate"))         &&
CompareDates(DBCheckActivityDate(pp.device.ID, "FinishDate")))
                                {
```

```
                                    //MessageBox.Show("Equipment Time stopped");
                                    DateTime endEquipmentTime = DateTime.Now;
                                    string format = "HH:mm:ss";
                                    endEquipmentTime.ToString(format);
                                    TimeSpan   totalTime   =   endEquipmentTime   -
startEquipmentTime;
                                    double  EquipmentCost  =  totalTime.TotalHours  *
DBCheckIDValue(pp.device.ID);
                                    DBAdd("ValueTable",      pp.device.ID,      "",
DBCheckIDType(pp.device.ID),                                EquipmentCost.ToString(),
DBCheckIDAssignedActivity(pp.device.ID), DBCheckIDDesc(pp.device.ID) + " equipment
work cost");
                                    startEquipmentTime = DateTime.MinValue;
                                }


                                /*
                                //Equipment time input
                                //equipment time start
                                if  (DBCheckIDType(pp.device.ID)  ==  "Equipment"  &&
startEquipmentTimeNote  ==  true  &&  !CompareDates(DBCheckActivityDate(pp.device.ID,
"StartDate"))  &&  CompareDates(DBCheckActivityDate(pp.device.ID,  "FinishDate"))  &&
startEquipmentTime == DateTime.MinValue && k.Name == "WorkingArea")
                                {
                                    MessageBox.Show("Equipment time start");
                                    startEquipmentTime = DateTime.Now;
                                    string format = "HH:mm:ss";
                                    startEquipmentTime.ToString(format);
                                }
                                 */
                                //labor earned value
                                //MessageBox.Show("ID:" + pp.device.ID + ", Area:"
+ k.Name + ", IDType:" + DBCheckIDType(pp.device.ID));
                                /*
                                if  (CompareDates(DBCheckActivityDate(pp.device.ID,
"StartDate")))
                                {
                                    MessageBox.Show("Start True");
                                }
                                else
                                {
                                    MessageBox.Show("Start False");
                                }

                                if  (CompareDates(DBCheckActivityDate(pp.device.ID,
"FinishDate")))
                                {
                                    MessageBox.Show("End True");
                                }
                                else
                                {
                                    MessageBox.Show("End False");
                                }*/

                            }
                        }
```

```
                        //MessageBox.Show("ELSE");
                        //Labor timing input
                        if     (startLaborTime     !=     DateTime.MinValue     &&
DBCheckIDType(pp.device.ID) == "Labor")
                        {
                            bool LaborWorking = true;
                            foreach         (CorrectionRegion         k         in
cell.cellcfg.regional_correction_table.region_list.ToArray())
                            {
                                //MessageBox.Show("X0:"                            +
Convert.ToDouble(k.box.X0)  +  "  ,X1:"  +  Convert.ToDouble(k.box.X1)  +  "\nY0:"  +
Convert.ToDouble(k.box.Y0)  +  "  ,Y1:"  +  Convert.ToDouble(k.box.Y1)  +  "\nX:"  +
Convert.ToDouble(tud.tag.GetPosition().X)        +         "         ,Y:"         +
Convert.ToDouble(tud.tag.GetPosition().Y));
                                if (k.Name == "Labor")
                                {
                                    if        (Convert.ToDouble(k.box.X0)        <=
Convert.ToDouble(tud.tag.GetPosition().X)    &&    Convert.ToDouble(k.box.X1)    >=
Convert.ToDouble(tud.tag.GetPosition().X)    &&    Convert.ToDouble(k.box.Y0)    <=
Convert.ToDouble(tud.tag.GetPosition().Y)    &&    Convert.ToDouble(k.box.Y1)    >=
Convert.ToDouble(tud.tag.GetPosition().Y))
                                    {
                                        LaborWorking = true;
                                    }
                                    else
                                    {
                                        LaborWorking = false;
                                    }

                                }
                            }
                            if (!LaborWorking)
                            {
                                //MessageBox.Show("Time stopped");
                                DateTime endLaborTime = DateTime.Now;
                                string format = "HH:mm:ss";
                                endLaborTime.ToString(format);
                                TimeSpan     totalTime     =     endLaborTime     -
startLaborTime;
                                double    LaborCost    =    totalTime.TotalHours    *
DBCheckIDValue(pp.device.ID);
                                DBAdd("ValueTable",        pp.device.ID,        "",
DBCheckIDType(pp.device.ID),                                LaborCost.ToString(),
DBCheckIDAssignedActivity(pp.device.ID),  DBCheckIDDesc(pp.device.ID)  +  "  labor
work cost");
                                startLaborTime = DateTime.MinValue;
                            }
                        }


                        //End hammad addition


                                        if(tud != null)
```

```
//                                          foreach (TagUserData tud in rtls.TagList)
                                            {
//                                                  if (tud != null && tud.tag ==
pp.device)
                                                    {
#if NanoLoc
                                                          if      (pp.source      is
Nanotron.nanoLOC.AVR_DK.LocationEngine)
                                                          {
        tud.last_nanoloc_position_packet = pp;
                                                                if (pp.position !=
null)
        tud.nanoloc_position = pp.position;
                                                          }
#endif
                                                          if (pp.position != null)
                                                          {
                                                                if   (pp.source   is
TrilaterationCC && pp.source.id == RTLS.LocationEngineId_TOA)
                                                                {
        tud.lastPosition = pp.raw.Time;
                                                                      if   (tud.cell
!= null)
        tud.lastPositionCycle = tud.cell.cycleCount;
                                                                      if
(pp.FromCell == tud.tag.GetCell())
                                                                      {
        tud.used_distance.Clear();
                                                                            if
(pp.position.used_ref != null)
        foreach (ReferencePosition_Distance rp in pp.position.used_ref)
        {
        foreach (RangingFilter.DeviceDistance dd in rp.distance)
        {
               if (!tud.used_distance.Contains(dd))
                      tud.used_distance.Add(dd);
        }
        }
                                                                      }
                                                                }
                                                                if
(rtls.Config.CellDisplayMode == SystemConfigV02.CellPanelDisplayMode.Global)
```

```
                                                                {
        cellGroupPanel.ltTags_invalidate = true;
        cellGroupPanel.pnlMap_invalidate = true;
        cellGroupPanel.CheckInvalidate();
                                                                }
                                                                else
                                                                {
                                                                      foreach
(CellPanel panel in cellPanel_List)
                                                                      {
                                                                            if
(panel.currentCell    !=    null    &&    panel.currentCell.cell    !=    null    &&
panel.currentCell.cell == tud.tag.GetCell())
                                                                            {
        //panel.TagPositionArrive(pp);
        panel.ltTags_invalidate = true;
        panel.pnlMap_invalidate = true;
        panel.CheckInvalidate();
                                                                            }
                                                                      }
                                                                }
                                                          }
                                                    }
                                              }
                                        }
                                  }
                            }
```

Add the following functions

```
  public bool DBAdd(string table, string TagID, string Location, string Type,
string Value, string ActivityID, string Description)
        {
            string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";
            MySqlConnection conn = null;

            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
```

```
                cmd.Connection = conn;
                DateTime dt = DateTime.Now;

                //NEW COMMAND
                if (table == "ValueTable")
                {
                    cmd.CommandText = "INSERT INTO " + table + "(`TagID`, `Type`,
`Value`, `ActivityID`, `Description`, `DateTime`) VALUES (@TagID, @Type, @Value,
@AID, @Description, @Datetime)";
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@Type", Type);
                    cmd.Parameters.AddWithValue("@Value", Value);
                    cmd.Parameters.AddWithValue("@AID", ActivityID);
                    cmd.Parameters.AddWithValue("@Description", Description);

                }
                else
                {
                    cmd.CommandText = "INSERT INTO " + table + "(ID, TagID,
Location, DateTime) VALUES(NULL, @TagID, @Location, @DateTime)";
                    cmd.Prepare();
                    cmd.Parameters.AddWithValue("@Location", Location);
                }


                cmd.Parameters.AddWithValue("@TagID", TagID);

                //get date and time
                DateTime dt1 = DateTime.Now;
                string dtActual1 = dt1.ToString("yyyy-MM-dd HH:mm:ss");
                cmd.Parameters.AddWithValue("@DateTime", dtActual1);

                //execute query
                cmd.ExecuteNonQuery();
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());
                return false;

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }

            }
            return true;

        }
      public bool DBExist(string TagID)
        {
            string                                cs                                =
"server=localhost;userid=root;password=hammad;database=rtls";
```

```
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;


                cmd.CommandText = "SELECT * from taginfo WHERE TagID='" + TagID +
"'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                // dr.Read();
                //textbox contain the id or name which you want to check in table
                if (dr.Read())
                {
                    return true;
                }
                else
                {
                    return false;
                }
             }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return true;
        }
      public bool DBValueExist(string TagID, string Activity, string Type)
      {
          string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";
          MySqlConnection conn = null;
          try
          {
              conn = new MySqlConnection(cs);
              conn.Open();

              MySqlCommand cmd = new MySqlCommand();
              cmd.Connection = conn;
```

```
                cmd.CommandText = "SELECT * from valuetable WHERE TagID='" + TagID +
"' AND ActivityID ='"+Activity+"' AND Type = '"+Type+"'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                // dr.Read();
                //textbox contain the id or name which you want to check in table
                if (dr.Read())
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return true;
        }
        public string DBCheckIDType(string TagID)
          {
              string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";
              MySqlConnection conn = null;
              try
              {
                  conn = new MySqlConnection(cs);
                  conn.Open();

                  MySqlCommand cmd = new MySqlCommand();
                  cmd.Connection = conn;


                  cmd.CommandText = "SELECT * from TagDetails WHERE TagID='" + TagID
+ "'";
                  cmd.CommandType = CommandType.Text;
                  cmd.Connection = conn;
                  MySqlDataReader dr;
                  dr = cmd.ExecuteReader();
                  string TagIDType = null;
                  while (dr.Read())
                  {
                      TagIDType = (string)dr["Type"];
                  }
```

```
                //MessageBox.Show(TagIDType);
                // dr.Read();
                //textbox contain the id or name which you want to check in table

                return TagIDType;

            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return "";
        }
      public double DBCheckIDValue(string TagID)
        {
            string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;


                cmd.CommandText = "SELECT * from tagdetails WHERE TagID='" + TagID
+ "'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                double TagIDValue = 0;
                while (dr.Read())
                {
                    TagIDValue = Convert.ToDouble((string)dr[3]);
                }

                return TagIDValue;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
```

```
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return 0;
        }
      public string DBCheckIDDesc(string TagID)
    {
        string cs = "server=localhost;userid=root;password=hammad;database=rtls";
        MySqlConnection conn = null;
        try
        {
            conn = new MySqlConnection(cs);
            conn.Open();

            MySqlCommand cmd = new MySqlCommand();
            cmd.Connection = conn;


            cmd.CommandText = "SELECT * from TagDetails WHERE TagID='" + TagID +
"'";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = conn;
            MySqlDataReader dr;
            dr = cmd.ExecuteReader();
            string TagIDDesc = null;
            while (dr.Read())
            {
                TagIDDesc = (string)dr["Description"];
            }

            return TagIDDesc;
        }
        catch (MySqlException ex)
        {
            MessageBox.Show("Error:" + ex.ToString());

        }
        finally
        {
            if (conn != null)
            {
                conn.Close();
            }
        }
        return "";
    }
      public string DBCheckIDAssignedActivity(string TagID)
      {
          string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";
          MySqlConnection conn = null;
          try
          {
```

```
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;


                cmd.CommandText = "SELECT * from TagDetails WHERE TagID='" + TagID +
"'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                string TagIDAA = null;
                while (dr.Read())
                {
                    TagIDAA = (string)dr["ActivityID"];
                }

                return TagIDAA;
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }
            return "";
        }
        public string DBCheckActivityDate(string TagID, string DateType)
    {
            string cs = "server=localhost;userid=root;password=hammad;database=rtls";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();

                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                cmd.CommandText = "SELECT * from TagDetails WHERE TagID='" + TagID +
"'";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                string TagIDActivityID = null;
                while (dr.Read())
                {
                    TagIDActivityID = (string)dr["ActivityID"];
```

```
            }
            dr.Close();
            MySqlCommand cmd2 = new MySqlCommand();
            cmd2.Connection = conn;
            cmd2.CommandText = "SELECT * from activitydetails WHERE ActivityID ='"
+ TagIDActivityID + "'";
            cmd2.CommandType = CommandType.Text;
            cmd2.Connection = conn;
            MySqlDataReader dr2;
            dr2 = cmd2.ExecuteReader();
            string TagDate = null;
            while (dr2.Read())
            {
                if (DateType == "StartDate")
                {
                    TagDate = (string)dr2["StartDate"];
                }
                else if (DateType == "FinishDate")
                {
                    TagDate = (string)dr2["FinishDate"];
                }
            }
            dr2.Close();
            return TagDate;
        }
        catch (MySqlException ex)
        {
            MessageBox.Show("Error:" + ex.ToString());

        }
        finally
        {
            if (conn != null)
            {
                conn.Close();
            }
        }
        return "";
    }
      public bool DBUpdate(string TagID, string Location)
      {
          string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";

          MySqlConnection conn = null;
          MySqlConnection conn2 = null;

          try
          {
              conn = new MySqlConnection(cs);
              conn.Open();

              MySqlCommand cmd = new MySqlCommand();
              cmd.Connection = conn;
```

```
                cmd.CommandText = "SELECT * from taginfo WHERE TagID='" + TagID +
"'";
                cmd.CommandType = CommandType.Text;

                cmd.Connection = conn;

                MySqlDataReader dr;
                dr = cmd.ExecuteReader();

                //textbox contain the id or name which you want to check in table
                if (dr.Read())
                {
                  try
                  {
                      conn2 = new MySqlConnection(cs);
                      conn2.Open();

                      MySqlCommand cmd2 = new MySqlCommand();
                      cmd2.Connection = conn2;
                      DateTime dt = DateTime.Now;
                      cmd2.CommandText  =  "UPDATE  `taginfo`  SET  `DateTime`='"  +
dt.ToString("yyyy-MM-dd  HH:mm:ss")  +  "',  `Location`  =  '"+  Location  +"'  WHERE
`ID`='" + dr[0] + "'";
                      cmd2.Prepare();
                      cmd2.ExecuteNonQuery();
                      return true;
                  }
                  catch (MySqlException ex)
                  {
                      MessageBox.Show("Error:" + ex.ToString());
                      return false;

                  }
                  finally
                  {
                      if (conn2 != null)
                      {
                          conn2.Close();
                      }
                  }
                }
            }
            catch (MySqlException ex)
              {
                  MessageBox.Show("Error:" + ex.ToString());
                return false;

              }
              finally
              {
                  if (conn != null)
                  {
                      conn.Close();
                  }
              }
            return true;
```

```
}
public bool CompareDates(string Date)
{
    DateTime ActivityDate = DateTime.Parse(Date);
    DateTime thisDay = DateTime.Now;
    int result = DateTime.Compare(ActivityDate, thisDay);

    if (result < 0) {
        return false;
    } else {
        return true;

    }
}
```

Appendix D

EVMPanel Code

APPENDIX D – EVMPANEL CODE

Create a new windows form in VisualStudio and code as follows,

Create ‘Examiner.cs’:

Examiner.cs

```
//-----------------------------------------------------------------------
// NavisWorks Sample code
//-----------------------------------------------------------------------

// (C) Copyright 2010 by Autodesk Inc.

// Permission to use, copy, modify, and distribute this software in
// object code form for any purpose and without fee is hereby granted,
// provided that the above copyright notice appears in all copies and
// that both that copyright notice and the limited warranty and
// restricted rights notice below appear in all supporting
// documentation.

// AUTODESK PROVIDES THIS PROGRAM "AS IS" AND WITH ALL FAULTS.
// AUTODESK SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTY OF
// MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE.  AUTODESK
// DOES NOT WARRANT THAT THE OPERATION OF THE PROGRAM WILL BE
// UNINTERRUPTED OR ERROR FREE.
//-----------------------------------------------------------------------
//
// This sample demonstrates basic LINQ style searching capabilities
// in the .NET API
//
//-----------------------------------------------------------------------


using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows.Forms;

//Add two new namespaces
using Autodesk.Navisworks.Api;
using Autodesk.Navisworks.Api.Plugins;

namespace Examiner
{

    [PluginAttribute("Examiner.Examiner",
                     "SHR",
                     ToolTip = "Earned Value Management Controls.",
                     DisplayName = "EVM Panel")]
```

```
    //[Strings("Examiner.SHR.name")] //name of the Localisation file

   public class Examiner : AddInPlugin
   {
      public override int Execute(params string[] parameters)
      {
          SearchForm searchForm = new SearchForm();
          searchForm.Show();
         return 0;
      }
   }
}
```

SearchForm.cs

```
//-----------------------------------------------------------------------
// NavisWorks Sample code
//-----------------------------------------------------------------------

// (C) Copyright 2009 by Autodesk Inc.

// Permission to use, copy, modify, and distribute this software in
// object code form for any purpose and without fee is hereby granted,
// provided that the above copyright notice appears in all copies and
// that both that copyright notice and the limited warranty and
// restricted rights notice below appear in all supporting
// documentation.

// AUTODESK PROVIDES THIS PROGRAM "AS IS" AND WITH ALL FAULTS.
// AUTODESK SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTY OF
// MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE.  AUTODESK
// DOES NOT WARRANT THAT THE OPERATION OF THE PROGRAM WILL BE
// UNINTERRUPTED OR ERROR FREE.
//-----------------------------------------------------------------------
//
// This sample demonstrates basic LINQ style searching capabilities
// in the .NET API
//
//-----------------------------------------------------------------------

using System;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using System.Collections.Generic;
using System.IO;
using System.Data;
using System.Net;
using System.Linq;
using System.Text;

//hammad
using MySql.Data;
using MySql.Data.MySqlClient;
using Autodesk.Navisworks.Api;
using Autodesk.Navisworks.Api.Plugins;
using Autodesk.Navisworks.Api.Timeliner;
using Autodesk.Navisworks.Api.Controls;
```

```
using ComApi = Autodesk.Navisworks.Api.Interop.ComApi;
using ComApiBridge = Autodesk.Navisworks.Api.ComApi;
using Autodesk.Navisworks.Api.Interop.ComApi;


namespace Examiner
{
   public partial class SearchForm : Form
   {
       /*bool runScript = false;
       string cs = "server=localhost;userid=root;password=hammad;database=rtls";
       MySqlConnection conn = null;
       MySqlCommand cmd = new MySqlCommand();
       MySqlDataReader dr;*/


      public SearchForm()
      {
         InitializeComponent();
         //this.cost.ChartAreas[0].AxisY.ScaleView.Zoom(0, 2);
         cost.ChartAreas[0].AxisX.Minimum = 0;

      }


      private string DumpTaskInfo(TimelinerTask task, UInt32 level, string Info)
      {
          for (Int32 i = 0; i < level; i++)
          {
              //Debug.Write("  ");
          }

          /*
          StringBuilder builder = new StringBuilder();
          builder.Append(task.DisplayName);
          if (task.PlannedStartDate != null && task.PlannedEndDate != null)
          {
              builder.Append("      Planned      Start      Date      -      "      +
task.PlannedStartDate.ToString()    +    "    Planned    End    Date    -    "    +
task.PlannedEndDate.ToString()     +     "Actual     Start     Date     -     "     +
task.ActualStartDate.ToString());
          }
           */
          if (Info == "PS")
          {
              return task.PlannedStartDate.ToString();
          }
          if (Info == "PE")
          {
              return task.PlannedEndDate.ToString();
          }
          if (Info == "AS")
          {
              return task.ActualStartDate.ToString();
          }
          if (Info == "AE")
```

```
            {
                string AE = task.ActualEndDate.ToString();
                if (AE != "")
                {
                    return AE;
                }
                else
                {
                    return "";
                }


            }
            if (Info == "Name")
            {
                return task.DisplayName.ToString();
            }

            MessageBox.Show(task.DisplayName.ToString());
            //MessageBox.Show(builder.ToString());

            foreach (TimelinerTask child in task.Children)
            {
                DumpTaskInfo(child, level + 1, "");
            }
            return "";
        }


        private void AssignTag_Click(object sender, EventArgs e)
        {
/*                                                              string      cs      =
"server=localhost;userid=root;password=hammad;database=rtls";
            MySqlConnection conn = new MySqlConnection(cs);
            conn = new MySqlConnection(cs);
            conn.Open();

            MySqlCommand cmd = new MySqlCommand();
            cmd.Connection = conn;

            try
            {
                cmd.CommandText  =  "SELECT  *  FROM  tagdetails  ORDER  BY  ActivityID
ASC";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
                dr = cmd.ExecuteReader();
                // dr.Read();
                //textbox contain the id or name which you want to check in table
                while (dr.Read())
                {
                    DocumentTimeliner                doc_timeliner                =
Autodesk.Navisworks.Api.Application.MainDocument.GetTimeliner();
                    foreach (TimelinerTask task in doc_timeliner.Tasks)
                    {
```

```
                    if (dr[5].ToString() == DumpTaskInfo(task, 0, "Name"))
                    {
                        MessageBox.Show(DumpTaskInfo(task, 0, "AE"));
                    }

                }
            }


        }
        catch (MySqlException ex)
        {
            MessageBox.Show("Error:" + ex.ToString());

        }
        finally
        {
            if (conn != null)
            {
                conn.Close();
            }
        }*/
        TagAssignment AssignTags = new TagAssignment();
        timer.Start();
        AssignTags.Show();

        // TURN THIS BACK ON!!
        //
    }

    private void EVM_Click(object sender, EventArgs e)
    {
        GraphDisplays GraphDisplay = new GraphDisplays();
        GraphDisplay.Show();
    }

    private void timer_Tick(object sender, EventArgs e)
    {
        cost.Series.Clear();
        // actual cost series creation
        Series AC = new Series();
        AC.LegendText = "Actual Cost";
        AC.Color = System.Drawing.Color.Red;
        AC.ChartType = SeriesChartType.Line;
        AC.BorderWidth = 3;
        //database details
        string                                  cs                                      =
"server=localhost;userid=hammad;password=hammad;database=rtls";
        MySqlConnection conn = null;
        //value and housekeeping
        double value = 0;
        DateTime CompleteDateTime = DateTime.MinValue;
        TimeSpan timeDifferenceAC = new TimeSpan(0, 0, 0);
        TimeSpan totalTimeAC = new TimeSpan(0, 0, 0);
```

```
            //earned value series creation
            int i = 1;
            Series EV = new Series();
            EV.LegendText = "Earned Value";
            EV.Color = System.Drawing.Color.Blue;
            EV.ChartType = SeriesChartType.Line;
            EV.BorderWidth = 2;
            DateTime Today = DateTime.Now;

            DocumentTimeliner                    doc_timeliner                    =
Autodesk.Navisworks.Api.Application.MainDocument.GetTimeliner();
            TimeSpan totalTime = new TimeSpan(0, 0, 0);
            double totalCost = 0;

            foreach (TimelinerTask task in doc_timeliner.Tasks)
            {
                //earned value
                DateTime PlannedStart = DateTime.Parse(DumpTaskInfo(task, 0, "PS"));
                //processing

                if (i == 1)
                {
                    totalTime = Today - PlannedStart;

                    //actual cost
                    DateTime  ActualStart  =  DateTime.Parse(DumpTaskInfo(task,  0,
"AS"));
                    totalTimeAC = Today - ActualStart;
                    i = i + 1;
                }


                TimeSpan timeDifference = Today - PlannedStart;
                int day = timeDifference.Days - totalTime.Days;

                //final variables
                day = Math.Abs(day);
                totalCost = (double)task.TotalCost + totalCost;

                //add to series
                EV.Points.Add(new DataPoint(day, totalCost));

            }
            cost.Series.Add(EV);

            //actual cost
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();
                MySqlCommand cmd = new MySqlCommand();
                cmd.Connection = conn;
                cmd.CommandText = "SELECT * FROM valuetable ORDER BY DateTime ASC";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr;
```

```
                dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    value = value + Convert.ToDouble(dr[3].ToString());
                    CompleteDateTime = DateTime.Parse(dr[6].ToString());

                    timeDifferenceAC = Today - CompleteDateTime;
                    int dayAC = timeDifferenceAC.Days - totalTimeAC.Days;

                    dayAC = Math.Abs(dayAC);
                    AC.Points.Add(new DataPoint(dayAC, value));


//MessageBox.Show(Autodesk.Navisworks.Api.Application.Gui.MainWindow, value + ". "
);
                }

//MessageBox.Show(Autodesk.Navisworks.Api.Application.Gui.MainWindow, value + ", "
+ Math.Abs(timeDifferenceAC.Days));

                //AC.Points.Add(new DataPoint(dayAC, value));
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Error:" + ex.ToString());

            }
            finally
            {
                if (conn != null)
                {
                    conn.Close();
                }
            }

            cost.Series.Add(AC);

            //color change
            try
            {
                conn = new MySqlConnection(cs);
                conn.Open();
                MySqlCommand cmd = new MySqlCommand();
                cmd.CommandText = "SELECT * FROM rtls.taginfo ORDER BY `DateTime`
ASC";
                cmd.CommandType = CommandType.Text;
                cmd.Connection = conn;
                MySqlDataReader dr = cmd.ExecuteReader();
                //dr.Read();
                DataTable dt = new DataTable();
                dt.Load(dr);
                foreach (DataRow row in dt.Rows)
                {
                    ChangeColor(row[1].ToString(), row[3].ToString());
                    //MessageBox.Show(row[0].ToString()+",     "+row[1].ToString()+",
"+row[2].ToString()+", "+row[3].ToString());
```

```
            }

        }
        catch (MySqlException ex)
        {
            MessageBox.Show("Error:" + ex.ToString());

        }

    }


    public bool CompareDates(string Date)
    {
        DateTime ActivityDate = DateTime.Parse(Date);
        DateTime thisDay = DateTime.Now;
        int result = DateTime.Compare(ActivityDate, thisDay);

        if (result < 0)
        {
            return false;
        }
        else
        {
            return true;
        }
    }

    private void EVM_Click_1(object sender, EventArgs e)
    {
        GraphDisplays GD = new GraphDisplays();
        GD.Show();
    }

    public void ChangeColor(string ObjectName, string Location)
    {
        try
        {
            IEnumerable<ModelItem> items =

Autodesk.Navisworks.Api.Application.ActiveDocument.Models.RootItemDescendantsAndSe
lf.
           Where(x =>
               //Search for ClassDisplayName matching the text, regardless of
upper/lower case
                (x.DisplayName.ToLower() == ObjectName.ToLower())

                  );
            //Select the items in the model that are contained in the collection

Autodesk.Navisworks.Api.Application.ActiveDocument.CurrentSelection.CopyFrom(items
);

            //Change the color of the selected item
            if (ObjectName == Location)
            {
```

```
Autodesk.Navisworks.Api.Application.ActiveDocument.Models.OverridePermanentColor(i
tems, Autodesk.Navisworks.Api.Color.Green);
                }
                else if (Location == "")
                {

Autodesk.Navisworks.Api.Application.ActiveDocument.Models.OverridePermanentColor(i
tems, Autodesk.Navisworks.Api.Color.Red);
                }
                else
                {

Autodesk.Navisworks.Api.Application.ActiveDocument.Models.OverridePermanentColor(i
tems, Autodesk.Navisworks.Api.Color.White);
                }


                //Deselect all

Autodesk.Navisworks.Api.Application.ActiveDocument.CurrentSelection.Clear();

            }

            catch (Exception ex)
            {
                MessageBox.Show("Error in Changing Object Color");
            }


        }
    }

}
```

SearchFormDesigner.cs

```
namespace Examiner
{
   partial class SearchForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
```

```
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea1 =
new System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend1 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Title title1 = new
System.Windows.Forms.DataVisualization.Charting.Title();
            System.Windows.Forms.DataVisualization.Charting.Title title2 = new
System.Windows.Forms.DataVisualization.Charting.Title();
            this.timer = new System.Windows.Forms.Timer(this.components);
            this.AssignTag = new System.Windows.Forms.Button();
            this.cost = new System.Windows.Forms.DataVisualization.Charting.Chart();
            ((System.ComponentModel.ISupportInitialize)(this.cost)).BeginInit();
            this.SuspendLayout();
            //
            // timer
            //
            this.timer.Interval = 1000;
            this.timer.Tick += new System.EventHandler(this.timer_Tick);
            //
            // AssignTag
            //
            this.AssignTag.Font = new System.Drawing.Font("Microsoft Sans Serif",
18F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.AssignTag.Location = new System.Drawing.Point(127, 17);
            this.AssignTag.Name = "AssignTag";
            this.AssignTag.Size = new System.Drawing.Size(222, 58);
            this.AssignTag.TabIndex = 1;
            this.AssignTag.Text = "Tag Details";
            this.AssignTag.UseVisualStyleBackColor = true;
            this.AssignTag.Click += new System.EventHandler(this.AssignTag_Click);
            //
            // cost
            //
            this.cost.BorderlineColor = System.Drawing.Color.Black;
            this.cost.BorderlineDashStyle =
System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.Solid;
            chartArea1.Name = "ChartArea1";
            this.cost.ChartAreas.Add(chartArea1);
            legend1.Name = "Legend1";
            this.cost.Legends.Add(legend1);
```

```
            this.cost.Location = new System.Drawing.Point(12, 93);
            this.cost.Name = "cost";
            this.cost.Size = new System.Drawing.Size(444, 317);
            this.cost.TabIndex = 8;
            this.cost.Text = "cost";
            title1.Docking                                                  =
System.Windows.Forms.DataVisualization.Charting.Docking.Bottom;
            title1.Name = "Activity Number";
            title1.Text = "Days";
            title2.Docking                                                  =
System.Windows.Forms.DataVisualization.Charting.Docking.Left;
            title2.Name = "Cost ($)";
            title2.Text = "Cost ($)";
            this.cost.Titles.Add(title1);
            this.cost.Titles.Add(title2);
            //
            // SearchForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(470, 431);
            this.Controls.Add(this.cost);
            this.Controls.Add(this.AssignTag);
            this.Name = "SearchForm";
            this.Text = "EVM Panel";
            ((System.ComponentModel.ISupportInitialize)(this.cost)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Timer timer;
        private System.Windows.Forms.Button AssignTag;
        private System.Windows.Forms.DataVisualization.Charting.Chart cost;

    }
}
```

TagAssignment.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Windows.Forms;

using MySql.Data;
using MySql.Data.MySqlClient;
using Autodesk.Navisworks.Api.Timeliner;

namespace Examiner
{
    public partial class TagAssignment : Form
    {
        private bool changesMade = false;
        public TagAssignment()
        {
            InitializeComponent();
            /*
            string                                  cs                                  =
"server=localhost;userid=root;password=hammad;database=rtls";
            MySqlConnection conn = null;
            MySqlCommand cmd = null;
            DataTable dataTable = new DataTable();

            try
            {
                string sql = "SELECT * from rtls.tagdetails";

                conn = new MySqlConnection(cs);

                cmd = new MySqlCommand(sql, conn);

                conn.Open();

                using (MySqlDataAdapter da = new MySqlDataAdapter(cmd))
                {
                    da.Fill(dataTable);
                }

                TagsData.DataSource = dataTable;
                TagsData.DataMember = dataTable.TableName;
            }
            catch (Exception ex)
            {
                MessageBox.Show(string.Format("An     error     occurred     {0}",
ex.Message), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (conn != null) conn.Close();
            }
            */

            string                                  cs                                  =
"server=localhost;userid=hammad;password=hammad;database=rtls";
            MySqlConnection conn = null;
            try
            {
                conn = new MySqlConnection(cs);
```

```
            conn.Open();

            MySqlCommand cmd = new MySqlCommand();
            cmd.Connection = conn;


            cmd.CommandText = "SELECT * from rtls.tagdetails";
            cmd.CommandType = CommandType.Text;
            cmd.Connection = conn;
            MySqlDataReader dr;
            dr = cmd.ExecuteReader();
            // dr.Read();
            int i = 0;
            //textbox contain the id or name which you want to check in table
            while (dr.Read())
            {
                TagsData.Rows.Add();
                //TagsData.ClearSelection();
                //TagsData.Rows[i].Selected = true;
                TagsData.Rows[i].Cells[0].Value = dr[1].ToString();
                TagsData.Rows[i].Cells[1].Value = dr[2].ToString();
                TagsData.Rows[i].Cells[2].Value = dr[3].ToString();
                //TagsData.ClearSelection();
                i = i + 1;
                //MessageBox.Show(i);
            }

        }
        catch (MySqlException ex)
        {
            MessageBox.Show("Error:" + ex.ToString());

        }
        finally
        {
            if (conn != null)
            {
                conn.Close();
            }
        }
    }


    private void btnAdd_Click(object sender, EventArgs e)
    {
        TagsData.Rows.Add();
        TagsData.Focus();
        changesMade = true;
    }

    private void btnDelete_Click(object sender, EventArgs e)
    {
        if (TagsData.CurrentRow != null)
        {
            TagsData.Rows.Remove(TagsData.CurrentRow);
            changesMade = true;
```

```
            }
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            if (changesMade)
            {

                string                              cs                              =
"server=localhost;userid=hammad;password=hammad;database=rtls";
                MySqlConnection conn = new MySqlConnection(cs);
                conn.Open();
                MySqlCommand cmd2 = new MySqlCommand();
                cmd2.Connection = conn;
                cmd2.CommandText = "TRUNCATE tagdetails;TRUNCATE activityDetails";
                cmd2.ExecuteNonQuery();

                try
                {
                    conn = new MySqlConnection(cs);
                    conn.Open();
                    for (int i = 0; i < TagsData.Rows.Count; i++)
                    {
                        MySqlCommand cmd = new MySqlCommand();
                        cmd.Connection = conn;
                        cmd.CommandText = "INSERT INTO rtls.tagdetails (`TagID`,
`Type`, `Cost`, `Description`, `ActivityID`) VALUES (@TagID, @Type, @Cost, @Desc,
@Activity)";
                        cmd.Prepare();
                        cmd.Parameters.AddWithValue("@TagID",
TagsData.Rows[i].Cells[0].Value.ToString());
                        cmd.Parameters.AddWithValue("@Type",
TagsData.Rows[i].Cells[1].Value.ToString());
                        cmd.Parameters.AddWithValue("@Cost",
TagsData.Rows[i].Cells[2].Value.ToString());
                        cmd.Parameters.AddWithValue("@Desc",
TagsData.Rows[i].Cells[3].Value.ToString());
                        cmd.Parameters.AddWithValue("@Activity",
TagsData.Rows[i].Cells[4].Value.ToString());

//MessageBox.Show(TagsData.Rows[i].Cells[0].Value.ToString()    +    ",    "    +
TagsData.Rows[i].Cells[1].Value.ToString()        +        ",        "        +
TagsData.Rows[i].Cells[2].Value.ToString()        +        ",        "        +
TagsData.Rows[i].Cells[3].Value.ToString());
                        //execute query
                        cmd.ExecuteNonQuery();

//MessageBox.Show(TagsData.Rows[i].Cells[0].Value.ToString()    +    ",    "    +
TagsData.Rows[i].Cells[1].Value.ToString()        +        ",        "        +
TagsData.Rows[i].Cells[2].Value.ToString()        +        ",        "        +
TagsData.Rows[i].Cells[3].Value.ToString());
                    }

                    MySqlCommand cmd3 = new MySqlCommand();
                    cmd3.Connection = conn;
```

```
                    DocumentTimeliner               doc_timeliner               =
Autodesk.Navisworks.Api.Application.MainDocument.GetTimeliner();
                    foreach (TimelinerTask task in doc_timeliner.Tasks)
                    {
                        cmd3.CommandText    =    "INSERT    INTO    `activitydetails`
(`ActivityID`,  `StartDate`,  `FinishDate`)  VALUES  ('"  +  DumpTaskInfo(task,  0,
"Name") + "', '" + DumpTaskInfo(task, 0, "AS") + "', '" + DumpTaskInfo(task, 0,
"AE") + "')";
                        cmd3.Prepare();
                        //execute query
                        cmd3.ExecuteNonQuery();
                    }


                }
                catch (MySqlException ex)
                {
                    MessageBox.Show("Error:" + ex.ToString());

                }
                finally
                {
                    if (conn != null)
                    {
                        conn.Close();
                    }
                }
                }
                changesMade = false;
                this.Close();
            }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private string DumpTaskInfo(TimelinerTask task, UInt32 level, string Info)
        {
            for (Int32 i = 0; i < level; i++)
            {
                //Debug.Write("  ");
            }

            /*
            StringBuilder builder = new StringBuilder();
            builder.Append(task.DisplayName);
            if (task.PlannedStartDate != null && task.PlannedEndDate != null)
            {
                builder.Append("     Planned     Start     Date     -     "     +
task.PlannedStartDate.ToString()    +    "    Planned    End    Date    -    "    +
task.PlannedEndDate.ToString()     +     "Actual     Start     Date     -     "     +
task.ActualStartDate.ToString());
            }
             */
            if (Info == "PS")
```

```
            {
                return task.PlannedStartDate.ToString();
            }
            if (Info == "PE")
            {
                return task.PlannedEndDate.ToString();
            }
            if (Info == "AS")
            {
                string AS = task.ActualStartDate.ToString();
                if (AS != "")
                {
                    //return AS.Substring(0, AS.Length - 11);
                    return AS;
                }
                else
                {
                    return "";
                }
            }
            if (Info == "AE")
            {
                string AE = task.ActualEndDate.ToString();
                if (AE != "")
                {
                    //return AE.Substring(0, AE.Length - 11);
                    return AE;
                }
                else
                {
                    return "";
                }

            }
            if (Info == "Name")
            {
                return task.DisplayName.ToString();
            }

            MessageBox.Show(task.DisplayName.ToString());
            //MessageBox.Show(builder.ToString());

            foreach (TimelinerTask child in task.Children)
            {
                DumpTaskInfo(child, level + 1, "");
            }
            return "";
        }

    }
}
```

```
namespace Examiner
{
    partial class TagAssignment
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.TagsData = new System.Windows.Forms.DataGridView();
            this.btnAdd = new System.Windows.Forms.Button();
            this.btnDelete = new System.Windows.Forms.Button();
            this.btnSave = new System.Windows.Forms.Button();
            this.button1 = new System.Windows.Forms.Button();
            this.TagID = new System.Windows.Forms.DataGridViewTextBoxColumn();
            this.Type = new System.Windows.Forms.DataGridViewTextBoxColumn();
            this.Cost = new System.Windows.Forms.DataGridViewTextBoxColumn();
            this.Description                          =                          new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.Activity = new System.Windows.Forms.DataGridViewTextBoxColumn();

((System.ComponentModel.ISupportInitialize)(this.TagsData)).BeginInit();
            this.SuspendLayout();
            //
            // TagsData
            //
            this.TagsData.AllowUserToAddRows = false;
            this.TagsData.AllowUserToDeleteRows = false;
            this.TagsData.Anchor                                                    =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top      |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.TagsData.ColumnHeadersHeightSizeMode                               =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
```

```
            this.TagsData.Columns.AddRange(new
System.Windows.Forms.DataGridViewColumn[] {
            this.TagID,
            this.Type,
            this.Cost,
            this.Description,
            this.Activity});
            this.TagsData.Location = new System.Drawing.Point(12, 12);
            this.TagsData.Name = "TagsData";
            this.TagsData.RowHeadersWidth = 25;
            this.TagsData.RowTemplate.Height = 24;
            this.TagsData.Size = new System.Drawing.Size(780, 346);
            this.TagsData.TabIndex = 1;
            //
            // btnAdd
            //
            this.btnAdd.Location = new System.Drawing.Point(12, 364);
            this.btnAdd.Name = "btnAdd";
            this.btnAdd.Size = new System.Drawing.Size(160, 26);
            this.btnAdd.TabIndex = 7;
            this.btnAdd.Text = "Add Tag Details";
            this.btnAdd.UseVisualStyleBackColor = true;
            this.btnAdd.Click += new System.EventHandler(this.btnAdd_Click);
            //
            // btnDelete
            //
            this.btnDelete.Location = new System.Drawing.Point(173, 364);
            this.btnDelete.Name = "btnDelete";
            this.btnDelete.Size = new System.Drawing.Size(159, 26);
            this.btnDelete.TabIndex = 8;
            this.btnDelete.Text = "Delete Tag Details";
            this.btnDelete.UseVisualStyleBackColor = true;
            this.btnDelete.Click += new System.EventHandler(this.btnDelete_Click);
            //
            // btnSave
            //
            this.btnSave.Location = new System.Drawing.Point(546, 364);
            this.btnSave.Name = "btnSave";
            this.btnSave.Size = new System.Drawing.Size(120, 26);
            this.btnSave.TabIndex = 9;
            this.btnSave.Text = "&Save && Exit";
            this.btnSave.UseVisualStyleBackColor = true;
            this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(672, 364);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(120, 26);
            this.button1.TabIndex = 10;
            this.button1.Text = "&Cancel && Exit";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // TagID
            //
```

```
            this.TagID.HeaderText = "TagID";
            this.TagID.Name = "TagID";
            //
            // Type
            //
            this.Type.HeaderText = "Type";
            this.Type.Name = "Type";
            //
            // Cost
            //
            this.Cost.HeaderText = "Cost";
            this.Cost.Name = "Cost";
            //
            // Description
            //
            this.Description.HeaderText = "Description";
            this.Description.Name = "Description";
            //
            // Activity
            //
            this.Activity.HeaderText = "Activity";
            this.Activity.Name = "Activity";
            //
            // TagAssignment
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(806, 401);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.btnSave);
            this.Controls.Add(this.btnDelete);
            this.Controls.Add(this.btnAdd);
            this.Controls.Add(this.TagsData);
            this.Name = "TagAssignment";
            this.Text = "Tag Assignment";
            ((System.ComponentModel.ISupportInitialize)(this.TagsData)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.DataGridView TagsData;
        private System.Windows.Forms.Button btnAdd;
        private System.Windows.Forms.Button btnDelete;
        private System.Windows.Forms.Button btnSave;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.DataGridViewTextBoxColumn TagID;
        private System.Windows.Forms.DataGridViewTextBoxColumn Type;
        private System.Windows.Forms.DataGridViewTextBoxColumn Cost;
        private System.Windows.Forms.DataGridViewTextBoxColumn Description;
        private System.Windows.Forms.DataGridViewTextBoxColumn Activity;
    }
}
```