Internet of Things-Based Smart Classroom Environment

Amir R. Atabekov
Kennesaw State University

Internet of Things-Based Smart Classroom Environment

Master's Thesis

By

Amir Atabekov
MSCS Student
Kennesaw State University
Department of Computer Science

In fulfillment of the Requirements for the Degree of

Master of Science in Computer Science

May 2016

Internet of Things-Based Smart Classroom Environment

This thesis is approved for recommendation to the Graduate Council.

[illegible]

Jing (Selena) He
Thesis Advisor, Committee Chair

[illegible]

Hisham Haddad
Professor, Committee Member

[illegible]

Ying Xie
Professor, Committee Member

[illegible]

Ken Hoganson
Professor, Committee Member

DEDICATION

To my mother: Roziya Atabekova

To my sister: Zuhra Atabekova

To my two brothers: Farhad and Rustam

The loving memory of my father: Ravshan Atabekov

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Dr. Selena He, for her guidance, motivation and mentorship. This thesis wouldn't exist without her guidance.

I would also like to thank Dr. Hisham Haddad for motivating me to participate in the 2016 ACM SAC Student Research Competition and winning the second place. And also for providing valuable suggestions and comments about my thesis work.

I would like to thank Dr. Ying Xie for his valuable courses in Advanced Algorithms, HPCC and Advanced Database Systems.

I would like to thank Dr. Ken Hoganson and the Department of Computer Science for offering me the position of Graduate Research Assistant.

This thesis is funded in part by the Kennesaw State University's Office of the Vice President of Research (OVPR) Pilot Grant, by the College of Science and Mathematics Interdisciplinary Research Opportunities (IDROP) Program and by the Department of Computer Science Mini-Grants Award.

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Internet of Things (IoT) is a novel paradigm that is gaining ground in the Computer Science field. There's no doubt that IoT will make our lives easier with the advent of smart thermostats, medical wearable devices, connected vending machines and others. One important research direction in IoT is Resource Management Systems (RMS). In the current state of RMS research, very few studies were able to take advantage of indoor localization which can be very valuable, especially in the context of smart classrooms. For example, indoor localization can be used to dynamically generate seat map of students in a classroom. Indoor localization is not the only concept which was not thoroughly researched in RMS. Another valuable proposition is to treat physical chairs as "smart" devices, which can report their occupancy, user information, and duration of presence to a cloud data store. Interconnected smart chairs consisting of pressure sensors, RFID readers, wireless communication capabilities, indoor localization and useful mobile application can serve as a powerful tool for instructors and other stakeholders. In this thesis we propose a complete smart classroom system consisting of smart chairs, anchor nodes, cloud storage and Android application. Implementation of indoor localization is a challenging and intricate task. Furthermore, since GPS chips cannot be used indoors, different and more challenging techniques have to be used. We developed a special protocol to handle communication and data flow of localization between smart chairs and the master node. Finally, the system was evaluated and special algorithm was developed to improve the accuracy of indoor localization in the context of smart classroom.

TABLE OF CONTENTS

CHAPTER 1

Introduction

In this chapter we provide a brief introduction to IoT, followed by our motivation, problem statement, research methodology and the contribution of this thesis.

1.1 Introduction

Internet of Things (IoT) is a novel paradigm that is gaining ground in the Computer Science field. The first Internet of Things appliance was a vending machine in early 1980s, which was developed at Carnegie Melon University [1]. The machine was designed to report over the internet whether or not it had cold drinks available. In a broader sense (IoT) is interconnectedness of multiple devices that can report, monitor, or provide other value or services that are of value to end users. IoT can refer to devices from Smart Thermostats that allow homeowners control temperature over the internet to medical wearable devices that can alert emergency services of any abnormality in vital signs. There's little doubt that IoT has the potential to make people's everyday life easier as interconnected devices become more and more ubiquitous.

1.2 Motivation

The management of classrooms, halls, offices, and public spaces in any organization and the efficient use of these resources when they are needed are challenging problems. With the rise of IoT, the management of these resources can be automated. Methods to automatically record the activities undertaken and monitor resources' usage inside class rooms in real-time are usually intricate. Taking students attendance-monitoring systems as a motivational example, in the past, instructors had to call the names of students or the students had to sign their names on the attendance check sheets. The former is a time consuming process, while the latter is unreliable and the attendance sheets could be lost or damaged. Subsequently, facial and voice recognition methods were proposed to verify the identity of students. Recently, Radio-Frequency (RF) communication based methods [2], [3] and smartphone based methods [4], [5] were extensively investigated. The former has some limitations to cover the whole area of a classroom by applying Near Field Communication (NFC) or Bluetooth technologies, while the latter requires students to use special application on their smartphones. Different from the aforementioned techniques, the proposed smart chair system can automatically check whether the chair is occupied or not by using multiple sensing technologies. Moreover, student identification can be automatically performed using RFID sensors, which can read RFID tags off student identification cards. Additionally, the tardiness or leave-early information can be analyzed by exploiting the integrated timestamps.

In this thesis we are proposing a smart classroom environment, composed of interconnected smart chairs and sensors, by employing IoT paradigm in order to simplify classroom management, attendance tracking and classroom interaction. A complete smart chair system was implemented to demonstrate the viability of the proposed smart classroom environment. The system consists of multiple smart chairs, sink node and anchor nodes. The system will use indoor localization to dynamically generate the seat map of students with their names and positions displayed on the instructor's screen.

1.3 Problem statement

After extensive literature research, several problems were identified with current state of classroom management in IoT.

- The use of indoor localization was poorly explored in the context of smart classroom environment. This is a lost opportunity because indoor localization can be useful to provide a dynamic seat map of students and can greatly improve in class interaction.
- The use of chairs as “smart” devices was not thoroughly considered. Most current research was focused on classroom in general and not on individual chairs which can be equipped with multiple sensors such as pressure sensors and RFID readers.

Implementation of indoor localization for smart classroom is intricate and difficult. The inclusion of expensive GPS chips in IoT devices isn’t practical since most systems require devices to be relatively small. Moreover, GPS is of little use indoors, since the accuracy of GPS suffers greatly when not in direct view of the sky. Currently, techniques used in indoor localization focus on multilateration, exploiting: Time of Arrival (ToA), Time difference of Arrival (TDoA) and Received signal strength indicator (RSSI) [6]. ToA requires time synchronization of very high accuracy; TDoA requires the use of two transmission mediums of different propagation speeds both of which limit its use in IoT.

1.4 Research methodology

The following research methodology was used in this work:

1. Perform extensive literature research on smart classrooms in IoT
2. Perform extensive literature research on current indoor localization techniques
3. Identify weaknesses in current IoT based resource management and find ways to make a better smart classroom

4. Explore different IoT implementation approaches such as Arduino, Raspberry Pi, Ti Launchpad and choose the optimal platform to implement the smart classroom on.
5. Explore and choose different components and sensors needed for this system.
6. Choose the most optimal indoor localization technique and implement it.
7. Assemble the system, solder the components, connect all sensors and program them
8. Develop the client mobile application which will generate the seat map and display chair occupancy information.
9. Evaluate the proposed smart classroom for performance and accuracy

1.5 Contribution

Each of the issues described in the problem statement will be thoroughly addressed. The contribution of this thesis is as follows:

1. Extensively explored and implemented indoor localization in the context of smart classroom environment
2. Developed a complete smart classroom environment with chairs as “smart” devices in mind.

The smart chair system has valuable commercial prospects, which can be helpful to build intelligent resource management systems, intelligent dynamic ticketing system and etc. In the proposed smart classroom concept, all the chairs are interconnected wirelessly and report all the information about its occupants and their identities to the sink node. The proposed system can greatly improve in class interaction, through the use of indoor localization, which will provide graphical seat map of students with their names displayed on instructors' screen. This graphical representation of the seat map will relieve instructors of uncomfortable moments when they don't remember particular student's name.

CHAPTER 2

Literature review

In this chapter, we summarized important concepts related to smart classroom followed by a summary of related work performed in resource management systems.

2.1 Internet of Things

IoT has emerged as a new network paradigm, which allows various physical entities in the world to connect with each other. The observed or generated information of these entities have a great potential to provide useful knowledge across different service domains, such as building management, energy-saving systems, surveillance services, smart homes, smart cities, etc. [8]. IoT was first proposed in 1999 by Kevin Ashton, who is the co-founder of Auto-ID center at the Massachusetts Institute of Technology (MIT) [9]. One foundational technology of IoT is the Radio-Frequency IDentification (RFID) technology, which allows microchips transmit the identification number of the objects to a reader through wireless communication. Through RFID technology, the physical objects can be identified, tracked, and monitored automatically. Nowadays, RFID technology has been widely adopted in logistics, pharmaceutical production, retailing, and supply chain management [10,11]. Another foundational technology of IoT is Wireless Sensor Networks (WSNs), which adopt interconnected intelligent sensors to periodically sense the monitored environment and send the information to the sink (or base station), at which the gathered/collected information can be further processed for end-user queries [12]. The applications include disaster control, environment and habitat monitoring, battlefield surveillance, traffic control, and health care applications [13]. Additionally, many other technologies and devices such as Near Field Communication (NFC) [14], short-range wireless communication (*i.e.*, ZigBee [15] and Bluetooth [16]), universal mobile accessibility (*i.e.*, Wi-Fi hotspots [17], and cellular networks [18]), social networking [19] and cloud computing [20] support internet of things to compose a extensive network infrastructure.

2.2 Indoor localization

Indoor localization has been a trending research area for many years. GPS does not provide reliable positioning indoors. Most current research in indoor localization, focus on determining mobile node's position using existing pervasive radio technologies such as WiFi, Zigbee, Bluetooth and etc. The popularity of using existing wireless infrastructure to determine position comes from its advantage, which doesn't require purchasing extra equipment and existing widespread deployment of these technologies.

Currently, indoor RSSI localization can be categorized into three types: plain multilateration [24], fingerprinting based localization [25] and statistical localization methods [26]. Multilateration methods that use RSSI only, may suffer from performance degradation in complex situations due to multipath fading and temporal dynamics [27]. A more accurate approach for indoor positioning is using fingerprinting localization [28]. Fingerprinting consists of two phases; first phase is a learning phase, which involves forming a database that contains signal fingerprints at each known location. The second phase is online phase where mobile nodes location is found by matching the RSSI fingerprint to the one's already stored in the database. Statistical methods involve refining the position by attempting to reduce location deviation caused by environmental factors [7].

2.3 Related resource management systems.

Several research papers related to classroom management were published. A classroom access control system which tries to solve classroom access with RFID card readers was proposed [38]. In this case the system consists of one Arduino Uno and one master node and communicates wirelessly through ZigBee. Social network integration was also explored. The typical usage scenario of the system is as follows:

1. Teacher enters a classrooms and swipes his smart card against NFC Reader of the classroom node.
2. Classroom node, which consists of Arduino Uno, NFC reader and RF link, transmits the RFID key of the teacher to the master node through ZigBee
3. Master node consists of Arduino Ethernet, which receives the RFID key of the teacher and authenticates it against the teacher database and stores the record on Xively.
4. Client application, developed with Google maps API, shows a map of classrooms and whether or not they are occupied and also displays the name of the professor who occupies the classroom.

In another research [39], the authors proposed a platform for the Internet of Things and performs a case study of a Smart Office. The authors propose to develop a web-based authoring tool that will store and handle the ontologies created by the web-based authoring tool for each service domain. Also to demonstrate the feasibility of the Integrated Semantics Service Platform (ISSP), the authors developed a prototype service for an office domain using ISSP. Ontologies express meanings of data and relationships using knowledge representation. The ISSP consists of two software packages: the web-based authoring tool and the integrated semantic service server. The web-based authoring tool provides four main input fields:

1. Service domain topic. This field is used to allow developers in each service domain to create the class topic which will become a super-class of an ontology created with the values of the second input field to represent particular service domain knowledge in a smart city.
2. Ontology schema and relationship. This field is used to input the name of the class, object properties, data properties, domain, range and restriction in order to create an ontology reflecting the service domain knowledge.
3. Reference resources. This field is used to input information about IoT resources such as URLS for RESTfull APIs.
4. Semantic web rule language. This field is used to input SWRL for reasoning based on added ontologies.

Another paper [40] proposed IoT-Based user-driven service modeling environment that consists of the user, the IoT service market and IoT service platform. The authors focused on creating an easy to use web authoring tool which can be used by non-technical users to create their own IoT services. In other words, the IoT based smart space management service can be defined by one user and deployed to another user to increase its value through the service personalization and customization. Multiple places can contain various objects such as lamps, heaters, roll screens and etc. The data generated by each device is in itself heterogeneous, so the authors proposed RESTful Smart Space Gateway (RSSG) that provides multiple communication protocols and the object data translation to represent data as the web standard. The proposed environment provides predefined context to the user to support a simple and easy to use method of service definition. The authors use a concept of ontology which is defined as an explicit and formal specification of conceptualization. Ontology is needed to define sensor data in order to provide context-aware IoT services and to describe the virtual world that represents the service domain. The environment proposed base ontology such as:

- Place ontology: This ontology describes the service domain characteristics such as home, school, office.
- Object ontology: This ontology describes the sensors and actuators to represent real-world entities such as lamps, computers etc.
- Context ontology: The service situation that describes a certain phenomenon.
- Service ontology: It describes the service condition and service behavior.

The proposed environment also describes User-Driven service modeling process where a user can select Service domain, Actuator function, Service condition which are predefined by the system and can be selected by the user.

Most previous research focuses on bigger picture of IoT such as service platforms and web services. The use of indoor localization was poorly explored in the context of smart classroom environment. This is a lost opportunity because indoor localization can be useful to provide a dynamic seat map of students and can greatly improve in class interaction. The use of chairs as "smart" devices was not thoroughly considered. Current research on smart classrooms was focused on classroom in general and not on individual chairs which can be equipped with multiple sensors such as pressure sensors and RFID readers.

CHAPTER 3

Technologies and preliminary work

This chapter describes different technologies used in this work such as Arduino, XBee, REST and indoor localization. The last sub-section of the chapter describes the preliminary smart chair system which serves as a proof-of-concept for smart classroom.

3.1 Arduino and XBee

Arduino is a flexible and easy to use open micro-controller and development environment. Arduino was first launched in 2005, which is based on a board with a single micro-controller and input/output pins for communications and control of physical objects and the environment. The functionality to connect and control physical objects directly relates to IoT. Hence Arduino gained popularity in short time because of the simplicity to use and cheap price in basic model. Different boards with various capabilities have been developed since 2005, such as UNO, Mega, Leonardo, Minim Due, Yun, etc. [21]. Arduino Yun is the same size as the Arduino Uno, with Uno being the most common Arduino board. However, the difference is that the Yun features a small Linux SoC that runs on a separate processor as well as an onboard Wi-Fi chip so we can connect the board to a local Wi-Fi network. Arduino Yun is perfectly suited to serve as a sink node while Arduino Unos can be used as slave nodes to collect information before sending it to Yun. In addition, Arduino boards offer enough capability and functionality for the smart chair system with its availability of myriad of third party components at low cost.

Arduino Yun contains two microprocessors. One is ATmega32u4 and the other is Atheros AR9331, which governs the Wi-Fi functionality and runs a Linux distribution based on

OpenWrt. The board has built-in Ethernet and Wi-Fi support, a USB-A port, micro-SD card slot, 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a 3 reset buttons [22], as shown in Fig. 3.1.

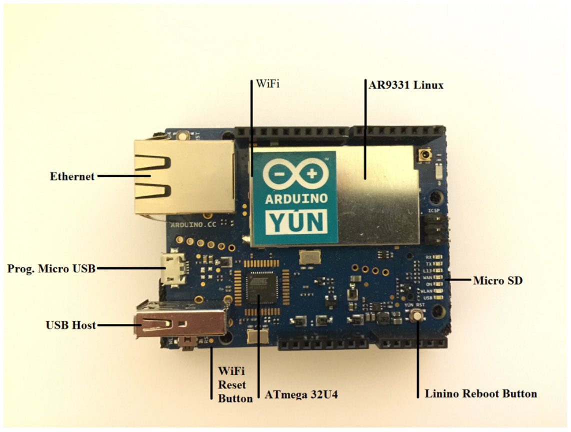


Fig. 3.1. Arduino Yun board

XBee is a low power radio communication device produced by Digi. XBee Series 1 uses IEEE 802.15.4 standard for wireless communication. XBee is inexpensive and can be used to expand Arduino making it capable of wireless communication. In this paper we chose XBee to implement indoor localization as illustrated in Fig. 3.2.

Fig. 3.2. XBee Series 1 module

3.2 Web 2.0 and REST

Web 2.0 is a term to describe the second generation of World Wide Web (WWW), which emphasize the ability for people to collaborate and share information online [23]. Web 2.0 services include video hosting services, wikis, blogs, social networking, and resource sharing environments (such as Flickr). Due to their platform independence and interoperability Web 2.0 and its REST API are the main data exchange protocols used in IoT. In Web 2.0, there are two formats to represent the resources, eXtensible Markup Language (XML) and JavaScript Object Notation (JSON). XML is a meta-language that can be used to define our communication language, while JSON is a lightweight alternative to XML format. JSON is a collection of key/value pairs, which belong to a subset of the object literal notation of JavaScript. Since these key/value pair structures can be adopted in any programming language, JSON is independent of the programming language used when exchanging data.

Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, for networked hyper-media applications. It is primarily used to build Web services that are lightweight, maintainable, and scalable. When Web services use REST architecture, they are called RESTful APIs (Application Programming Interfaces). REST is not dependent on any protocol, but almost every RESTful service uses Hypertext Transfer Protocol (HTTP) as its underlying protocol. In sum, RESTful systems typically, but not always, communicate over the HTTP with the same HTTP verbs (GET, POST, PUT, DELETE, etc.) used by web browsers to retrieve web pages and send data to remote servers.

The purpose of a REST service is to provide a window to its clients so that they can access the resources available on the server. REST does not put a restriction on the format of a representation of resource. In the smart chair system, we adopted JSON format. Moreover, RESTful API does not require the client to know anything about the structure of the API. Rather, the server needs to provide whatever information the client needs to interact with the service. For example, the server specifies the location of the resource, and the required fields. The client doesn't know in advance where to submit the information, and it doesn't know in advance what information to submit. Both forms of information are entirely supplied by the server. Hence, RESTful API is now popular in implementing web services and IoT backend.

3.3 Indoor localization

In this thesis, RSSI-based multilateration method [7] has been investigated and is used to determine position of chairs in order to provide the dynamic seat map in a smart classroom environment. The advantage of this method is that no additional equipment is needed; any radio frequency transceiver capable of providing RSSI measurement to users can be used.

The localization method uses XBee wireless transceivers based on ZigBee protocol [29]. ZigBee is low-power, low-bandwidth, lightweight, wireless solution, ideally suited for wireless sensor networks, home automation, remote control, industrial automation, agricultural automation and medical care [30]. XBee wireless transceivers can be connected to an Arduino microcontroller, which allows developers to implement a localization algorithm of choice in C programming language. The RSSI value can be extracted from the XBee node with Arduino library [31]. This value can then be used in the implementation of positioning algorithm in Arduino.

3.4 Preliminary proof-of-concept implementation

A proof-of-concept experimental system was implemented on a single chair. The system has multiple sensors and follows IoT paradigm. The collected information is uploaded to the cloud so that any client application can take advantage of data anywhere anytime when necessary. This initial proof-of-concept system will serve as a foundation for the proposed smart classroom system described in later chapters, which consists of multiple connected chairs with seat maps generated with indoor localization methods. The proof-of-concept system has three major components:

- The “thing”: the chair ID and its occupancy state collected and send via Arduino Yun.
- The cloud service backend: ThingSpeak [32].
- The mobile application: the chair occupied state and the user information can be displayed and monitored through mobile application.

The overall architecture of the smart chair system is described in Fig. 3.2. As shown, the RFID reader can automatically read the user’s identification (denoted by UID) and passes it along to the Arduino ATmega32u4 microcontroller. The custom program checks if the pressure was applied to the pressure resistor. If the pressure applied is more than twenty pounds, the program then passes the timestamp to the Atheros AR9331 microcontroller through the bridge library [33]. Finally, the bridge library issues a HTTP POST request to the ThingSpeak API [32] along with the UID of the scanned RFID card.

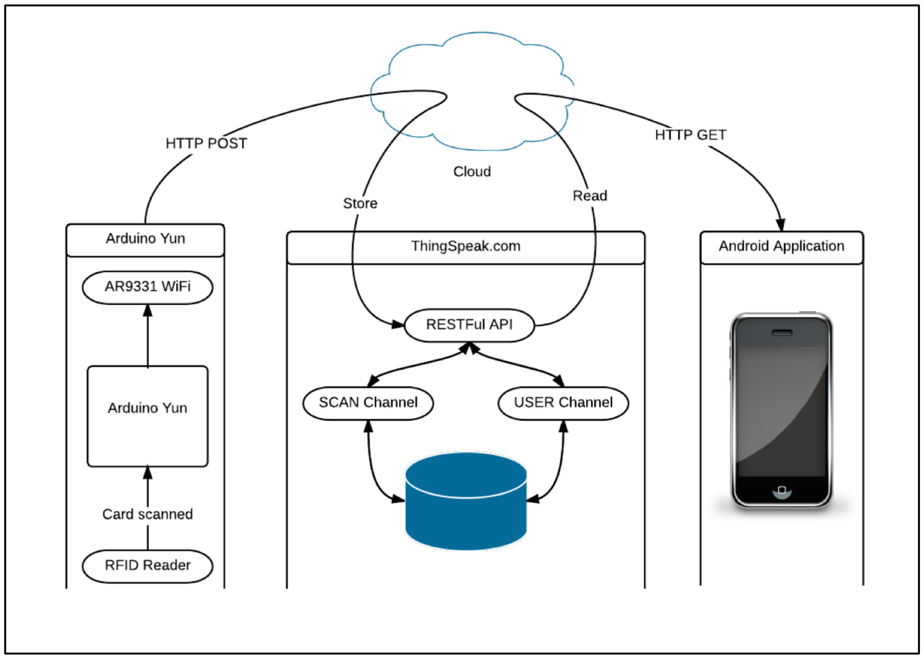


Fig. 3.2. Smart chair system architecture

The Android client application was developed to let administrators and other interested users monitor when or who has scanned his card and sat on the chair. The overview of the Android architecture is also illustrated in Fig. 3.2. The Android application issues an HTTP GET request to ThingSpeak's JSON feed. Each HTTP GET request is authenticated against an API key. When submitting a HTTP GET request, the API key must be included in the URL string. Finally, the retrieved data is displayed on the Android application. Table 3.1, illustrates the requirements of the initial implementation.

The distinct characteristic of the proposed system is that it has dynamic indoor localization built into the chairs. For instance, the position of the chairs can periodically change in a typical classroom for variety of reasons, e.g. student moves the chair or reorders the position, which may render the seat map inaccurate.

Table 3.1. Requirements of the proof-of-concept system.

Requirement	**Description**
Read RFID tag UID with reader	Write Arduino program which will read the UID off the RFID tag when in close proximity
Sense when pressure is applied	Connect pressure resistor and write corresponding program which detects when pressure is applied
Program indication lights	Implement LED indication lights in the program to know the system status
Find and configure IoT restful service to store data	ThingSpeak chosen as backend for the project. Corresponding USER and SCAN channels will be configured to store data.
Send the UID data to ThingSpeak	Write Arduino program which will send the UID data to ThingSpeak channel. Yun Bridge library is used for POST requests.
Develop Android Client	Android application will be developed to view the latest scans performed.
JSON Parser	A JSON parser will be written to parse the JSON data returned from ThingSpeak channels.
Implement Login screen	Login screen will be implemented on Android application to authenticate users

The proposed system uses trilateration technique, which dynamically updates the chair's position as described in the next chapters. Moreover, the system is based on Arduino which is open hardware platform, meaning the board can be custom engineered in the future to produce even more cost effective solution than Arduino by contracting manufacturer companies in Asia to assemble a custom tailored board.

CHAPTER 4

Proposed system

This chapter describes the proposed system beginning with system overview followed by proposed indoor localization method, web service, Android application and evaluation.

4.1 Proposed system overview

Based on initial proof-of-concept, smart classroom environment is proposed. The smart classroom system is able identify each student based on his or her RFID tag already embedded in each student id card. The system is planned to use XBee transceivers connected to Arduino microcontroller boards, making it perfect for indoor localization. High-level smart classroom system is illustrated in Fig. 4.1

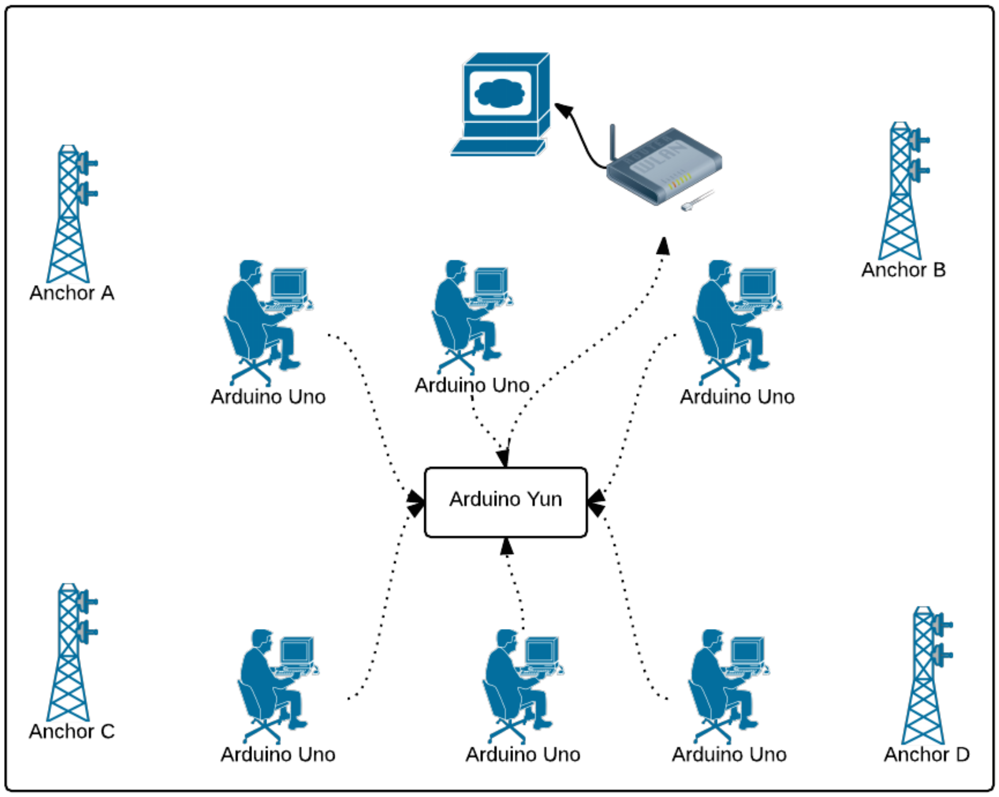


Fig. 4.1. Smart classroom illustration

As illustrated in the Fig. 4.1, the system consists of 4 anchor nodes, 1 master node and multiple mobile nodes. Each smart chair consists of Arduino Uno microcontroller board assembled with RFID, XBee transceiver and pressure sensor to detect seat occupancy. In addition, four anchor nodes are placed in four corners of the classroom, which will serve as reference nodes for RSSI multilateration. All the nodes will communicate with each other wirelessly through XBee. All the mobile nodes or chairs, can calculate their position by exploiting the RSSI indicator and applying triangulation calculation [34]. The master node is used to collect the data about chair occupancy, RSSI measurements from mobile nodes and to send the data to the custom web service. An android mobile application is developed to display chair occupancy data and generates dynamic seat map of students.

4.2 Proposed RSSI-based localization method

If we have three anchor nodes with known positions such that each anchor node's position is represented by $(x_{i,}y_i)$ for i=1…3. Then unknown node is represented by (x_u,y_u). By Pythagorean theorem we know that $a^2+b^2=c^2$. With this information we can set up three equations where "r" represents distance from mobile node to anchor node:

$$\begin{aligned}(x_1-x_u)^2+(y_1-y_u)^2&=r_1^2\\(x_2-x_u)^2+(y_2-y_u)^2&=r_2^2\\(x_3-x_u)^2+(y_3-y_u)^2&=r_3^2\end{aligned}\tag{1}$$

In order to solve the set of equations in 1, we have to convert them into linear equations by removing the quadratic terms x_u^2 and y_u^2. This can be done by subtracting the third equation from the previous two [6], which gives us two linear equations:

$$(x_1 - x_u)^2 - (x_3 - x_u)^2 + (y_1 - y_u)^2 - (y_3 - y_u)^2 = r_1^2 - r_3^2$$

$$(x_2 - x_u)^2 - (x_2 - x_u)^2 + (y_2 - y_u)^2 - (y_2 - y_u)^2 = r_2 - r_3^2.$$

Now the linear matrix equation can be constructed as in Equation (2).

$$2\begin{bmatrix} x_3 - x_1 & y_3 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{bmatrix}\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} (r_1^2 - r_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) \\ (r_2^2 - r_2^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2) \end{bmatrix} \quad (2)$$

The distance “r” can be calculated by ranging, where strength of received signal (RSSI) can be used to estimate distance. This is possible because electromagnetic waves have an inverse relationship between received power and distance traveled [34] as shown in Equation 3.

$$P_r = \frac{1}{d^2} \quad (3)$$

where P_r is the received power at a distance d from transmitter [34].

4.3 Web service

Custom web service is developed to handle back-end requests from master node and mobile application. Google App Engine is used to implement the IoT backend, which accepts POST requests from the master node and store the data in the cloud. The web service also provides GET requests which return the RSSI of the nodes in JSON format, which is used to display the results in the mobile application. In addition, the web service is used to store information about chair occupancy and the users of the system. The architecture and the flow of data of the web service are illustrated in Fig. 4.2.

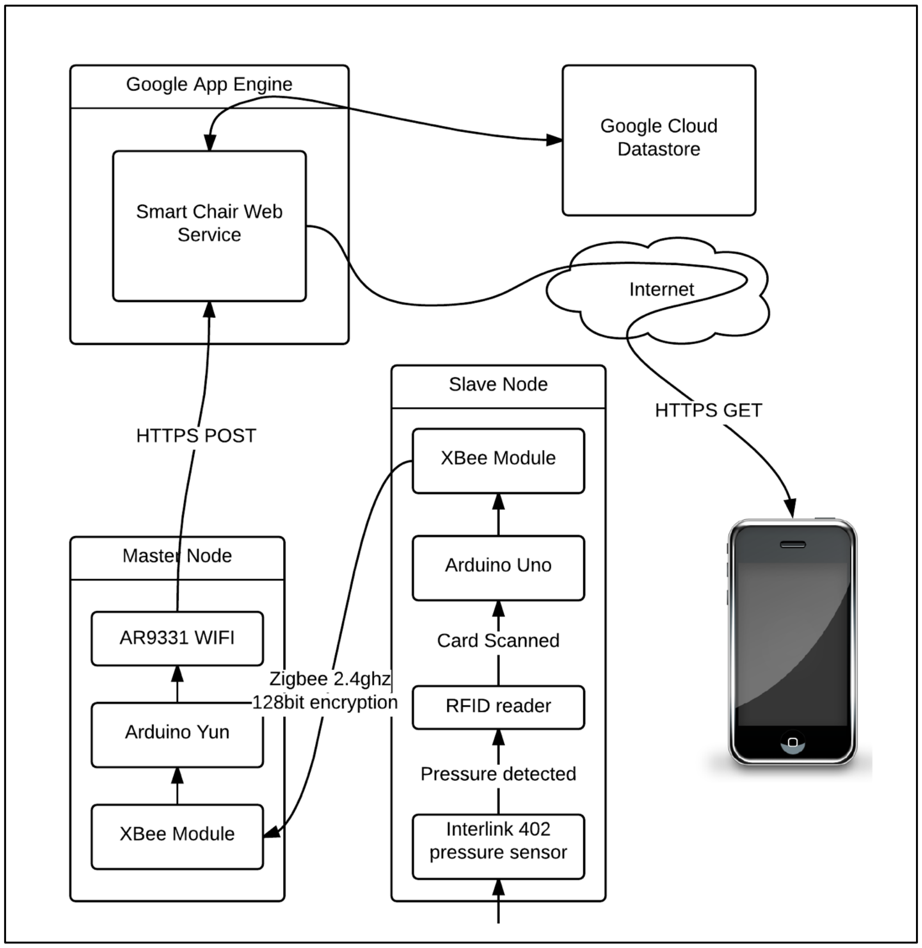


Fig. 4.2. Web service data flow

4.4 Android application

Android mobile application is developed to display the chair occupancy data and to generate the seat map. The mobile application interacts with the master node through the web service. The application retrieves the data from the web service with "GET" requests. Internally, the application performs parsing of the "json" data and uses the data to output the results in a user friendly format. The application can be used by instructors to display names of students, their arrival time and duration of their presence in the chair. Additionally, the application generates a seat map of students to help instructors visually identify them.

4.5 Evaluation

Every part of the system is evaluated for performance such as the smart chairs, web service, Android application and indoor localization. To verify the real world accuracy of localization, the multilaterated position is recorded and compared to the actual position and distance error is calculated. The advantage of using RSSI method is simplicity of implementation as well as ability to use existing transceivers as long as they provide RSSI value available to the user. The disadvantage of this method is possible inaccuracy due to nonlinear path loss indoor as well as shadowing [34]. The actual accuracy and performance will be evaluated in the later chapters.

CHAPTER 5

Technical challenges

This chapter describes the current technical challenges encountered during this work. The challenges include, packet loss, communication of master node with Android application, indoor localization accuracy, energy efficiency and dataflow protocol.

5.1 Packet loss

One of the most challenging parts of the system, is sending out multiple packets to the master node. The problem is that XBee module interacts with Arduino through serial communication. The limitation is that Arduino only supports 64bytes as its serial buffer. If new data is received by XBee module, then it needs to be consumed immediately, because if another packet arrives before the current one is read, the whole packet will be truncated. When sending request packet to slave nodes, each of them will request RSSI information from each of the anchor nodes almost simultaneously. Therefore, master node maybe receiving more packets than it can read, thus truncating or dropping them completely

5.2 Communication problem with Android application

The challenging part is to establish communication between Android application and the master node. Android application cannot communicate with master node directly in order to request localization or chair occupancy data. The master node has a Wi-Fi antenna, but without a static IP address communicating with it is difficult. Additionally, obtaining a static IP address for an IoT device is not always feasible. Conversely the master node cannot communicate with Android application directly. A solution has to be developed so that the Android application can request localization from the master node and so that master node can output it’s results back to the Android application.

5.3 Indoor localization inaccuracy

Implementation of indoor localization for this task is intricate and difficult, because indoor localization is inherently inaccurate. Path-loss and signal reflection loss can cause the chairs on the seat map to be aligned incorrectly.

5.4 Energy efficiency

Since the smart chairs are mobile and require a battery, it’s crucial that the system is energy efficient. The indoor localization can be implemented so that the calculation is performed either centrally or distributively. If localization is performed on the microcontroller of each smart chair individually, the battery could be depleted much faster than if the localization is performed on a master node or on an Android application. But the challenge is how the master node could calculate the position of another node. An algorithm has to be developed so that each mobile node only measures the RSSI from each of the anchor nodes, but sends the measurements to the master node without performing any calculation.

5.5 Dataflow protocol for localization

One of the main challenges is how will the localization requests work? When a user opens the Android application and presses “Display seat map”, what happens next? How will the initiation of localization request work? An algorithm has to be developed so that the master node knows when a user presses “Display seat map” button. When a master node receives the request for localization, what happens next?

A complete protocol has to be developed to handle the data flow from the moment user presses the button to the moment the seat map is displayed. The requests have to flow from Android application, to the master node, then to each of the slave nodes, then each of the slave nodes have to request RSSI measurements from each of the anchor nodes. Then each of the slave nodes have to send the results back to the master node, then master node has to send the result back to the Android application. This protocol is a difficult task which needs thorough consideration and development.

CHAPTER 6

System design

This chapter describes in detail, the system design of the smart classroom. It begins with high level design overview, following by description of the nodes, required hardware, node calibration, localization implementation, architecture of the web service and Android application. This chapter also provides solutions to the challenges described in the previous chapter. It provides solution to dataflow protocol, solution to inaccuracy, solution to packet loss problem and the solution for two-way communication with the Android application.

6.1 Design overview

In this section, the methodological design of the smart classroom environment is described in detail. The system consists of 4 anchor nodes and 4 slave nodes. Each node is composed of Arduino Uno and XBee Radio receiver attached to it. In addition, one master node is used. The master node collects all the data from mobile nodes (slaves). It is assembled out of Arduino Yun and XBee module. The XBee module is used for wireless data communication with slave nodes. The overview of the architecture is illustrated in Fig. 6.1. Each anchor node in Fig. 6.1 is assigned relative coordinates. Anchor A is assigned coordinates of (0,0), anchor b: (w,0), anchor c: (0,y) and anchor d: (w,y), where w and y are chosen based on the width and length of the room.

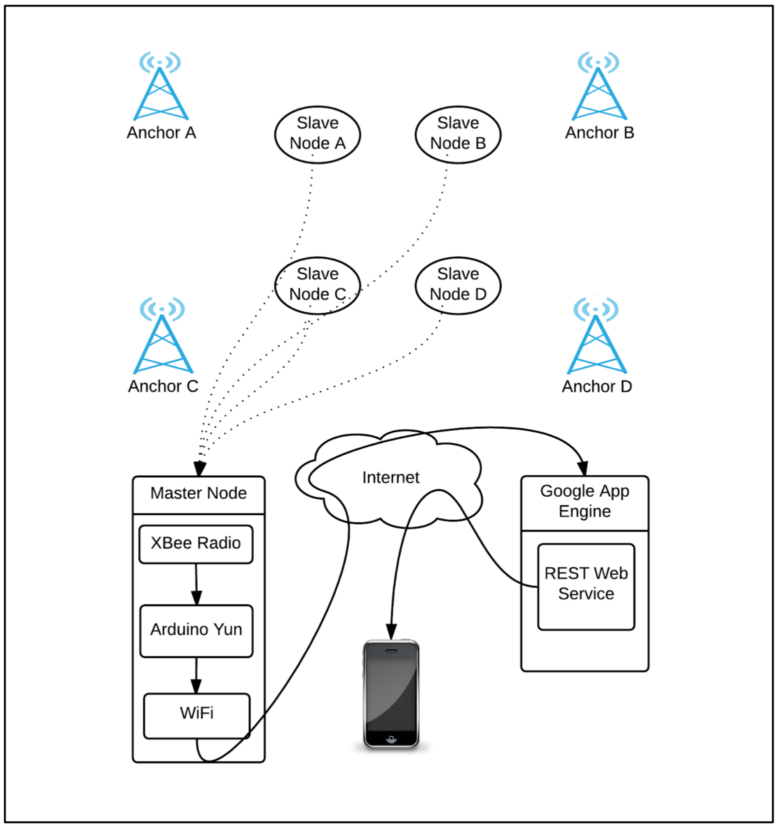


Fig. 6.1. System architecture

6.2 Slave node (Chair node)

Each smart chair (chair node), illustrated in Fig. 6.2, is assembled out of the following components:

- Arduino Uno
- RFID Reader: SainSmart RC522 RFID reader was chosen for reading UID off users' RFID cards. RC522 is easy to connect to Arduino, since a library [35] was already written to communicate with the reader through Arduino Serial Peripheral Interface (SPI) bus [36].
- Pressure Sensor: Interlink 402 pressure resistor is used to detect if a person is currently sitting on a chair.

- Miscellaneous: Two LEDs (Red and Blue) is used to indicate the status of card read and occupancy. Two 180 ohm resistors are used for LEDs and one 10k ohm resistor is used for pressure sensor.
- XBee module is used for wireless data communication and indoor positioning
- AmazonBasics 3000mAh re-chargeable battery pack.

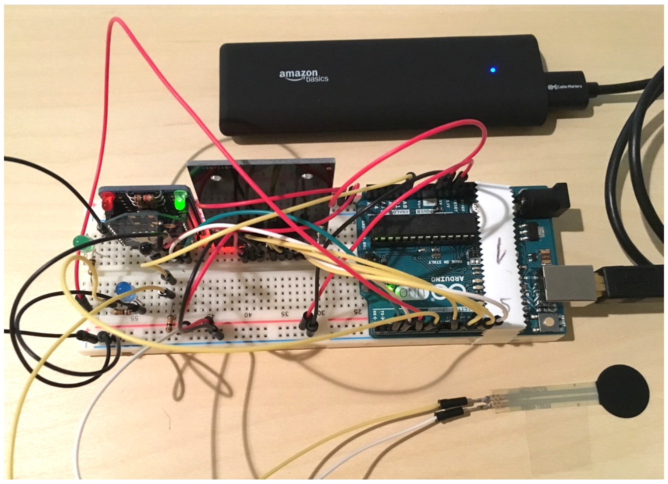


Fig. 6.2. Fully assembled chair (slave) node

6.3 Anchor node

Four anchor nodes are stationary and serve as reference nodes for multilateration. They are assembled out of the following components as illustrated in Fig 6.3:

- Arduino Uno
- XBee module used for wireless data communication and indoor positioning
- AmazonBasics 3000mAh re-chargeable battery pack.

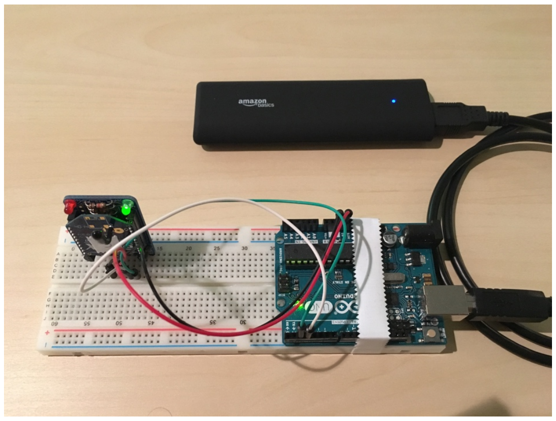


Fig. 6.3. Fully assembled anchor node

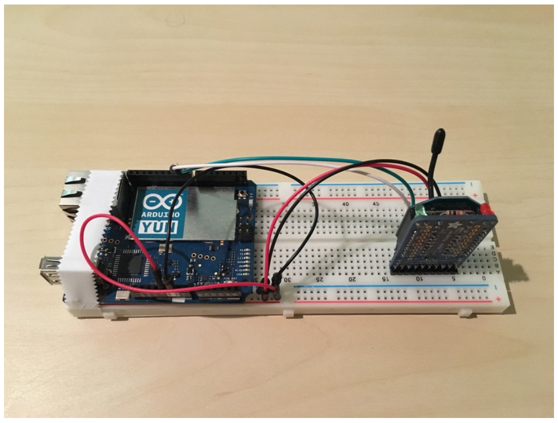


Fig. 6.4. Fully assembled master node

6.4 Master node

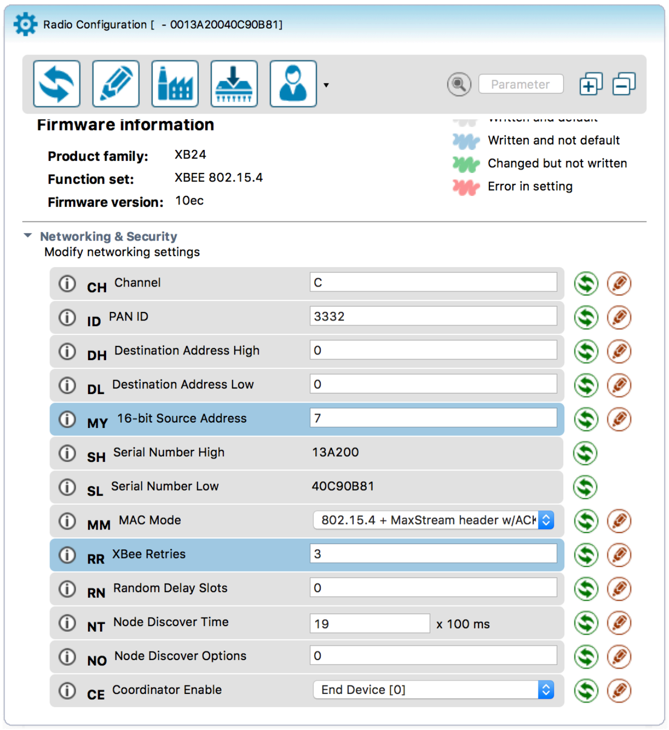
The master(sink) node collects data from the mobile nodes (chairs).It is assembled out of the following components as illustrated in Fig 6.4:

- Arduino Yun
- XBee module used for wireless data communication with mobile nodes

The only node that requires Arduino Yun is the master node, since Yun is the only model, which contains Wi-Fi antenna needed for internet communication in order to submit the localization data to the server.

6.5 Node calibration

Each XBee module's address is assigned sequentially by modifying the "MY" field of the module's firmware as illustrated in Fig. 6.5. Here, we use the terms node ID and address interchangeably. Node IDs from 1-4 are assigned to anchor nodes, while ID of 5 is assigned for master node and the rest of mobile (slave) nodes are assigned in sequential order from 6 to *n*. The reason we need to assign IDs to each node is to be able to send packages to specific nodes later in the program. For instance, as described later in this section, we will use 'lreq' packets and address them to anchor nodes only. In addition, the "AP" field has to be changed to “API enabled w/PPP” so that we are able to retrieve the RSSI values. The firmware settings can be changed by using XCTU software as illustrated in Fig 6.5



Firmware information

Product family:	XB24
Function set:	XBEE 802.15.4
Firmware version:	10ec

Written and not default
Changed but not written
Error in setting

Networking & Security
Modify networking settings

CH	Channel	C
ID	PAN ID	3332
DH	Destination Address High	0
DL	Destination Address Low	0
MY	16-bit Source Address	7
SH	Serial Number High	13A200
SL	Serial Number Low	40C90B81
MM	MAC Mode	802.15.4 + MaxStream header w/ACK
RR	XBee Retries	3
RN	Random Delay Slots	0
NT	Node Discover Time	19 x 100 ms
NO	Node Discover Options	0
CE	Coordinator Enable	End Device [0]

Fig. 6.5. XBee firmware configuration window in XCTU

6.6 Localization implementation

In this framework, RSSI-based multilateration method [18] is used to determine the position of the chairs. The advantage of this method is that no additional equipment is needed; any radio frequency transceiver capable of providing RSSI measurement to users can be used. In our case XBee module is attached to each Arduino Uno and communicates with each anchor node positioned in each corner in order to determine the nodes position. The following formula [37] is used to estimate the distance between mobile nodes and anchors:

$$d = 10^{\frac{A+rssi}{10*n}} \qquad (4)$$

In the formula above, *'d'* denotes distance between anchor node and slave node, *'A'* denotes RSSI at reference distance and *'n'* denotes path-loss exponent. The path loss is a value from 2 to 4, and is chosen experimentally based on characteristics of the environment. For indoor localization values ranging from 1.8 - 2.0 are usually used. In our case a value of 1.8 is used.

Once the distance is estimated, the node's position can be determined using equation (1), section 4.2, to compute the matrix (representing position of the chair) in the classroom. The matrix multiplication is illustrated using Java code in Fig. 6.6, to determine the location of the mobile node.

```
public  Node calcPosition(double[] x, double[] y, SlaveReply reply) {

    Matrix A=constructA(x,y);

    double[] r={ reply.anchor1,reply.anchor2,reply.anchor3,reply.anchor4};

    System.out.println("dist of node "+(reply.nodeId-5));
    r=convertToDist(r);
    Matrix b=constructB(x,y,r);
    A=A.times(2);
    Matrix X=A.solve(b);
    Node node=new Node(reply.nodeId,X.get(0,0),X.get(1,0));
    return node;
}

public  double[] convertToDist(double[] r){

    for(int i=0;i<r.length;i++){

        r[i]= Math.pow(10d, ((double)(C + r[i]) / (10 * n)));
        System.out.println("dist "+r[i]);
    }
    return r;
}

public  Matrix constructA(double[] x, double[] y){

    double[][] outAr=new double[x.length-1][2];
    for(int i=0;i<x.length-1;i++){
        int last=x.length-1;
        outAr[i][0]=x[last]-x[i];
        outAr[i][1]=y[last]-y[i];
    }
    return new Matrix(outAr);
}

public  Matrix constructB(double[] x, double[] y, double[] r){

    double[][] outAr=new double[x.length-1][1];
    for(int i=0;i<x.length-1;i++){
        int last=x.length-1;
        double rdiff= Math.pow(r[i],2)- Math.pow(r[last],2);
        double xdiff=Math.pow(x[i],2)-Math.pow(x[last],2);
        double ydiff=Math.pow(y[i],2)-Math.pow(y[last],2);
        outAr[i][0]=rdiff-xdiff-ydiff;
    }
    return new Matrix(outAr);
}
```

Fig. 6.6. Matrix multiplication

6.7 Inaccuracy in indoor localization

This section is dedicated to solve the inaccuracy challenge as described in section 5.3. Accuracy is a big problem in indoor localization, due to signal reflection and absorption loss which can occur in indoor environments. Fig. 6.7 illustrates the inaccuracy of chairs' position due to path loss. As we can see, the nodes are not aligned horizontally even though, they may be aligned in real life. However, to generate a seat map we only need relative position of each chair and not the actual position. Given the number of columns of chairs N, the following error correction algorithm can be applied:

```
Step 1. Calculate the estimated position with RSSI for each node
Step 2. Sort all nodes based on the Y coordinate
Step 3. Get the first N nodes, sort by X coordinate and assign
the average Y position for that row.
Step 4. Repeat Step 3, until no nodes are left, where N
represents the number of columns.
```

We have aligned each node by Y position, now we can apply a similar algorithm and align the nodes by X coordinates. The algorithm works by exploiting the fact that we need to know the chair's position relative to other chairs and not the absolute position.

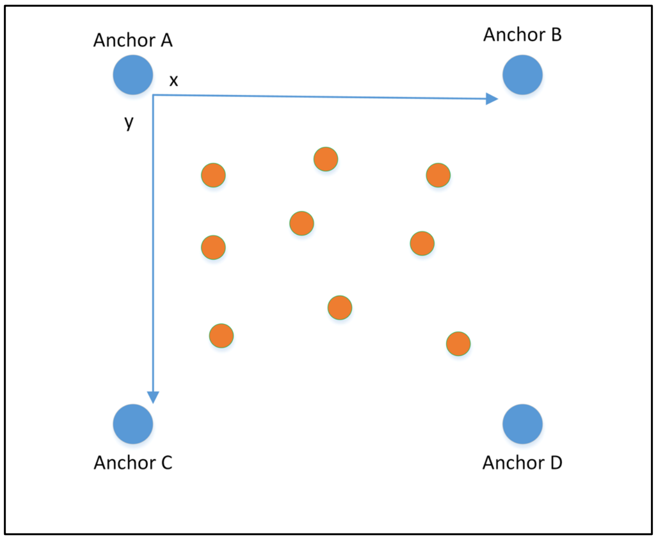


Fig. 6.7. Inaccuracy due to path and reflection loss

6.8 Dataflow protocol for localization

A complete protocol was developed to handle the data flow from the moment user presses the button to generate the seat map to the moment the seat map is displayed. The requests have to flow from Android application, to the master node, then to each of the slave nodes, then each of the slave nodes have to request RSSI measurements from each of the anchor nodes. Then after receiving the replies, each of the slave nodes have to send the results back to the master node, then master node has to send the result back to the Android application.

Table 6.1. Packet types and their description.

Packet type	**Description**
lreq	This packet is sent from slaves to anchor nodes to request RSSI packets.
creq	This packet is sent from master node to slave nodes to request initialization of localization
res	This packet is sent from anchor nodes to slave nodes which contain RSSI information.
creply	This packet is sent from each chair to master node which contain aggregated RSSI information for each anchor node for the chair.

The complete protocol works as follows. First, the android application posts a request for localization in our custom web service channel named “status”. Arduino Yun reads the channel, sees that there’s a pending request for localization, then sends a request called “creq” to slave nodes. Slave nodes receive the "*creq*" packet, then send "*lreq*" packet to anchor nodes. Upon received the "*lreq*" packet, each anchor node sends a reply packet called "res" back to the slave node. Slave node receives 4 "*res*" packets, which contains RSSI information from each of the anchor nodes. Upon receiving the "*res*" packets, the slave node sends aggregated "*RSSI*" information back to the master node in the form of "*creply*" packet. Master node receives the "*creply*" packet which contains the slave "node id" as well as the aggregated RSSI information from each anchor nodes, slave was able to receive. In the final step the master node sends the aggregated localization information to the custom web service channel for later retrieval which is described in the later sections. Packet types are described in Table 6.1.

The flow of packets is illustrated in Fig. 6.8. Each request is labeled in sequential order for illustration. Solid lines represent communication between master and slave nodes, while dashed lines represent communication between slave and anchor nodes.

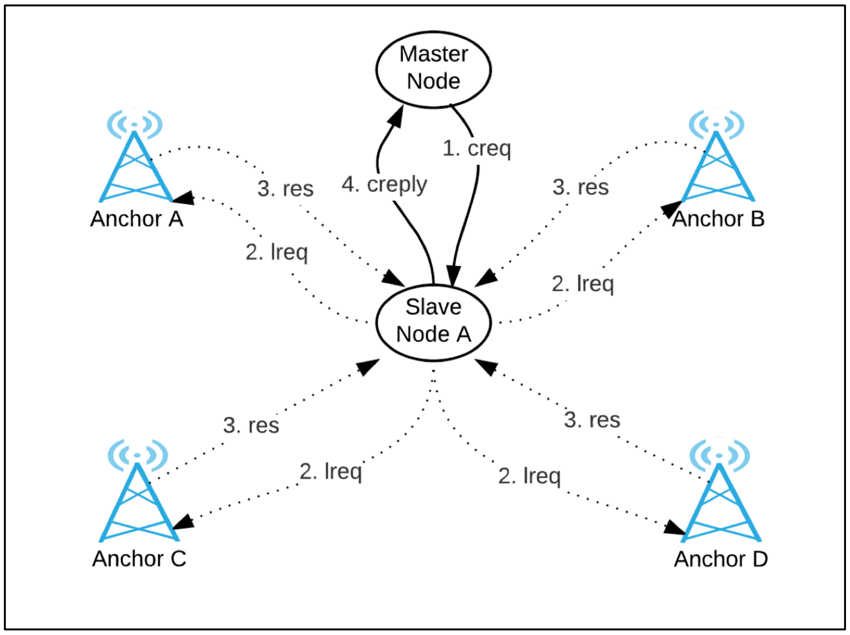


Fig. 6.8. Localization data flow

6.9 Packet loss problem

Previously we described the challenge of packet loss in section 5.1. One of the most challenging parts of the system, is sending out the “creq” packets. The problem is that XBee module interacts with Arduino through serial communication. The limitation is that

Arduino only supports 64bytes as its serial buffer. If new data is received by XBee module, then it needs to be consumed immediately, because if another packet arrives before the current one is read, the whole packet will be truncated. When sending "creq" packet to slave nodes, each of them will request RSSI information from each of the anchor nodes almost simultaneously. Therefore, master node maybe receiving more packets than it can read, thus truncating or dropping them completely.

The problem was solved by implementing delays based on the sequential id of the anchor node. Meaning the anchor node will receive the request, but will wait before sending out the reply based on a combination of slave_id and anchor_id. As mentioned in section IV, node ids from 1 to 4 are reserved for anchor nodes, and 6 to "*n*" for slave nodes. Based on this information the following delay formula is developed to avoid sending results simultaneously:

```
int del = 100 + ((anchor_id-1) * 100);
int final_delay = (500 * (slave_id - 1)) + del;
```

Each anchor node waits "final_delay" number of milliseconds before sending out the reply back. Table 6.2 illustrates the delays calculated for 4 anchor nodes and 4 slave nodes. We can observe that there's at least 100ms delay between different anchor nodes for the same slave_id and 200ms delay when switching to a different slave_id. This ensures that the master node has enough time to read the current packet before receiving the next one. After testing the system with the formula above, no packet loss has been detected so far, whereas before that, the packet loss was around 50%.

Table 6.2. Delays calculated for 4 anchor nodes and 4 slave nodes.

anchor_id	slave_id	delay (ms)
1	1	100
2	1	200
3	1	300
4	1	400
1	2	600
2	2	700
3	2	800
4	2	900
1	3	1100
2	3	1200
3	3	1300
4	3	1400
1	4	1600
2	4	1700
3	4	1800
4	4	1900

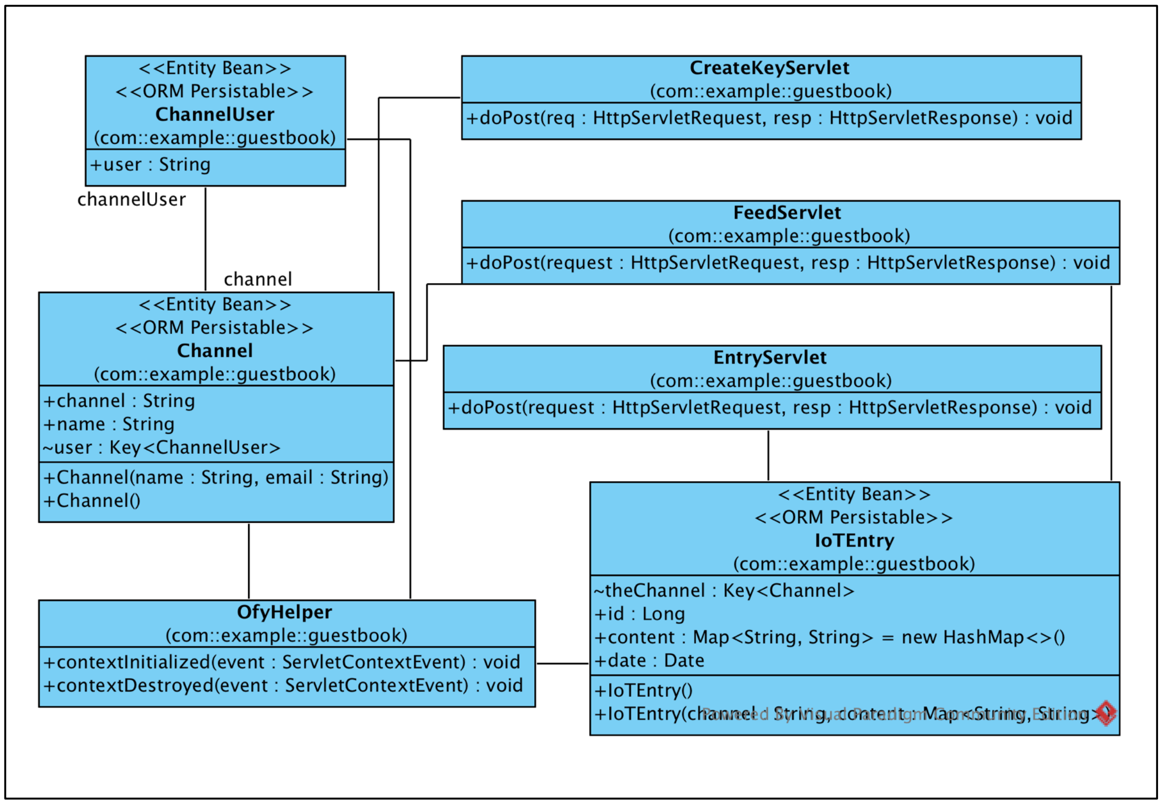
6.10 Web service

Web service running on Google App Engine was developed to store/retrieve data and to communicate between the master node and the Android application. The web service was written in Java. It runs on Google App Engine and uses Google Cloud Datastore for distributed data storage. The web service uses REST architecture and outputs the results on JSON format. The web service lets users create "channels" which can be tied to specific IoT devices for data storage. When a new channel is created, a unique key is generated for that specific channel. That key is used to "POST" or "GET" data from the channel. The web service has two servlets called "feedservlet" and "entryservlet", the former is used for retrieving data, while the latter is used for insertion. The following URLs illustrate the two servlets:

http://iot-service-1188.appspot.com/feedservlet?key=YOURKEY

http://iot-service-1188.appspot.com/entryservlet?key=YOURKEY

Note that each servlet requires “key” parameter. Once the “key” is provided, any number of parameters and values can follow. For example, to insert an entry for “name=John”, just append the URL with “&name=John”. This implementation makes it very easy to work with IoT devices, because no prior configuration or schema creation is required. Fig. 6.9 illustrates the UML diagram of the web service.



<<Entity Bean>>
<<ORM Persistable>>
ChannelUser
(com::example::guestbook)
+user : String
channelUser
channel
CreateKeyServlet
(com::example::guestbook)
+doPost(req : HttpServletRequest, resp : HttpServletResponse) : void
FeedServlet
(com::example::guestbook)
+doPost(request : HttpServletRequest, resp : HttpServletResponse) : void
<<Entity Bean>>
<<ORM Persistable>>
Channel
(com::example::guestbook)
+channel : String
+name : String
~user : Key<ChannelUser>
+Channel(name : String, email : String)
+Channel()
EntryServlet
(com::example::guestbook)
+doPost(request : HttpServletRequest, resp : HttpServletResponse) : void
<<Entity Bean>>
<<ORM Persistable>>
IoTEntry
(com::example::guestbook)
~theChannel : Key<Channel>
+id : Long
+content : Map<String, String> = new HashMap<>()
+date : Date
+IoTEntry()
+IoTEntry(channel : String, content : Map<String, String>)
OfyHelper
(com::example::guestbook)
+contextInitialized(event : ServletContextEvent) : void
+contextDestroyed(event : ServletContextEvent) : void
Powered by Visual Paradigm Community Edition

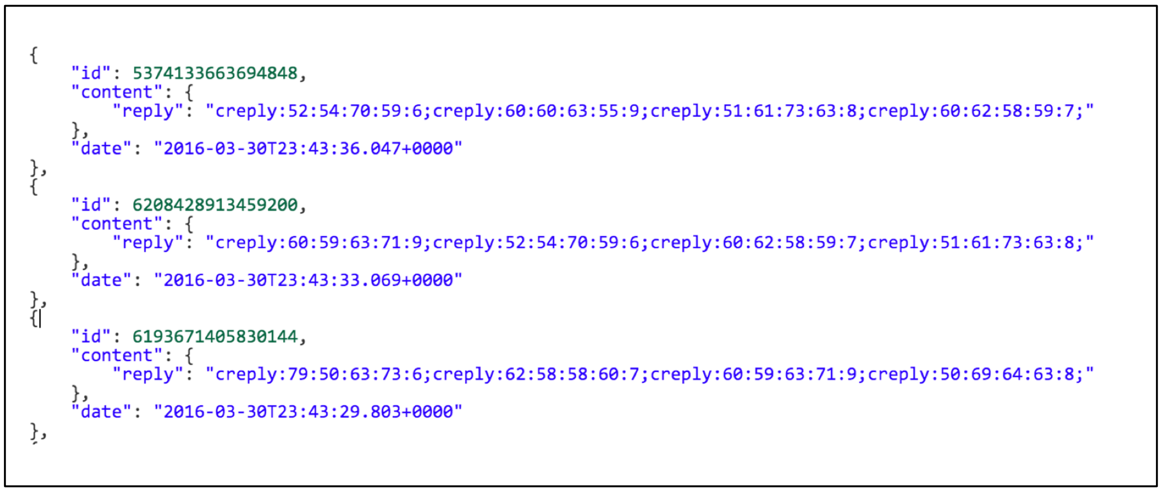
Fig. 6.9. UML of the back-end web service

Currently the smart classroom system has 5 channels, which are named “localization”, “status”, “scan”, “users” and “occupancy”. Channel named “localization” is used to store the the results of localization. Table 6.3 illustrates the fields of the channel.

Table 6.3. Localization channel.

Field	Description
id	The unique id of the channel entry
date	Stores the date when the entry was added
reply	Stores the aggregated reply string from the master node

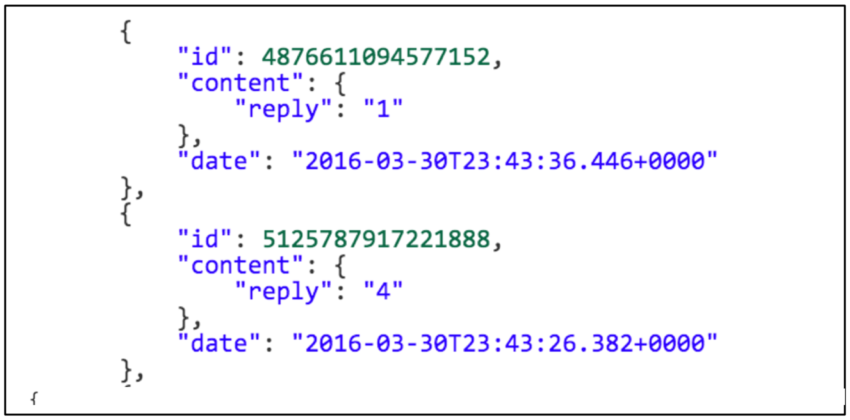
The actual JSON reply is illustrated in Fig 6.10. Note that each reply string contains 4 aggregated reply results called "creply" from each of the slave nodes. Each "creply" is delimited by semi colon. The first 4 integers following "creply" contain the RSSI values from each of the 4 anchor nodes sequentially. Last, in each "creply" comes the "node_id" of the slave node.



```
{
    "id": 5374133663694848,
    "content": {
        "reply": "creply:52:54:70:59:6;creply:60:60:63:55:9;creply:51:61:73:63:8;creply:60:62:58:59:7;"
    },
    "date": "2016-03-30T23:43:36.047+0000"
},
{
    "id": 6208428913459200,
    "content": {
        "reply": "creply:60:59:63:71:9;creply:52:54:70:59:6;creply:60:62:58:59:7;creply:51:61:73:63:8;"
    },
    "date": "2016-03-30T23:43:33.069+0000"
},
{
    "id": 6193671405830144,
    "content": {
        "reply": "creply:79:50:63:73:6;creply:62:58:58:60:7;creply:60:59:63:71:9;creply:50:69:64:63:8;"
    },
    "date": "2016-03-30T23:43:29.803+0000"
},
```

Fig. 6.10. JSON output from "localization" channel.

Next is the "status" channel to keep track of localization requests. This channel is monitored both by the master node and by the Android application for localization updates. When a button to generate a seat map is pressed, the Android application sends a request to "status" channel with reply of "4" indicating that a new localization is needed. When master node sees that the latest entry in the channel is named reply "4", it performs the RSSI measurement by sending "creq" to each slave node and sends the results to the "localization" channel. The last step is to update the "status" channel by inserting a new reply named "1", indicating that the localization request has been completed. Once Android application sees that the reply is "1", it retrieves the results from "localization" channel, completes the localization and generates the seat map. Fig. 6.11 illustrates the output of the "status" channel.



```
        {
            "id": 4876611094577152,
            "content": {
                "reply": "1"
            },
            "date": "2016-03-30T23:43:36.446+0000"
        },
        {
            "id": 5125787917221888,
            "content": {
                "reply": "4"
            },
            "date": "2016-03-30T23:43:26.382+0000"
        },
```

Fig. 6.11. Output of "status" channel.

Table 6.4 illustrates the "scan" channel used to store card scans. For example, when a user taps his card on the RFID reader, a new entry is added to this channel. Note that this channel can have multiple records with the same card_uid, because the same card can be scanned multiple times. A separate channel named "occupancy" stores the actual occupancy information for each card scan.

Table 6.4. Scan channel.

Field	**Description**
id	The unique id of the channel entry
date	Stores the date when the entry was added
card_uid	Stores the unique id of the RFID card that was scanned.

Channel named "users" stores information related to a specific user as described in Table 6.5. For example, it stores unique *id* of the card that belongs to the user. In fact, "card_uid" field is used as a foreign key in "scan" channel.

Table 6.5. User channel.

Field	Description
id	The unique id of the channel entry
date	Stores the date when the entry was added
user_type	Indicates what type of user e.g. "student", "Professor" and etc.
lastname	Last name of the user
firstname	First name of the user
card_uid	Stores the unique id of the RFID card that belongs to the user

Channel named "occupancy" stores the data each time a user sits down on the chair or stands up as described in Table 6.6. Field named "scan_id" can be treated as a foreign key to the "id" field of the "scan" channel.

Table 6.6. Occupancy channel.

Field	Description
id	The unique id of the channel entry
date	Stores the date when the entry was added
scan_id	This field can be treated as a foreign key to the "id" field of the "scan" channel.
sat_down	Records the timestamp when the user sat down
stood_up	Records the timestamp when the user stood up

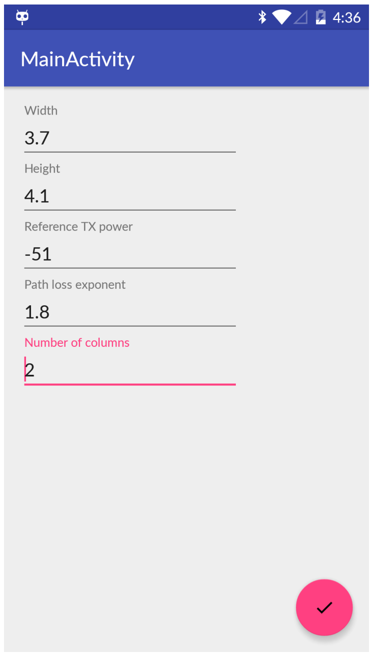
6.11 Establishing communication with master node

This section describes the solution to the communication challenge described in section 5.2. The Android application cannot communicate with the master node directly in order to request localization of chair occupancy data. The master node has a Wi-Fi antenna, but without a static IP address communicating with it is difficult. Additionally, obtaining a static IP address for an IoT device is not always feasible.

The problem was solved using a special channel named "status" in web service which is monitored both by the Android application and by the master node for status updates. In other words, the Android Application runs a separate thread in the background and keeps refreshing the “status” channel and looks if there are any updates. Additionally, when an Android Application needs to communicate with master node, it sends a request to the same “status” channel. The master node also keeps monitoring the “status” channel, when it sees that there’s an entry requesting data, it posts the information to the appropriate channel then updates the entry indicating to the Android application that the request has been completed.

6.12 Android application

The Android mobile application displays the attendance information, generates seat map, displays names of students and occupancy information. Fig. 6.12 illustrates the configuration screen where the width and height of the room is entered. In addition, path loss exponent, reference TX power and number of columns of chairs need to be specified. Once all the necessary information is entered, the application sends the localization request to the “status” channel, waits for reply, receives the reply, then generates the seat map as illustrated in Fig 6.13. All requests and data communication is performed in the background with no noticeable delay. When in “Seat Map” activity, each chair can be tapped and additional information will be displayed such as the name of the student, latest card scan and the duration of the chair occupancy as illustrated in Fig 6.14.


MainActivity

Width
3.7
Height
4.1
Reference TX power
-51
Path loss exponent
1.8
Number of columns
2

Fig. 6.12. Configuration screen

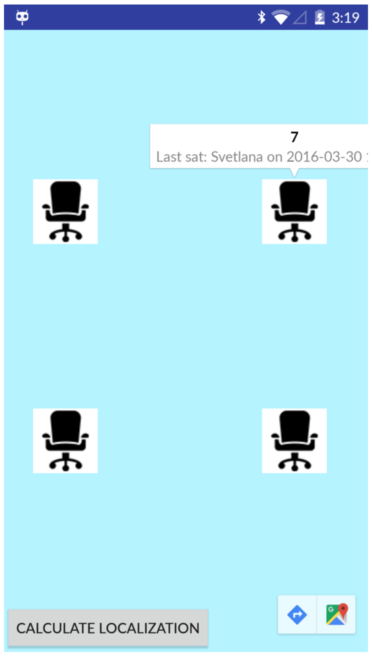
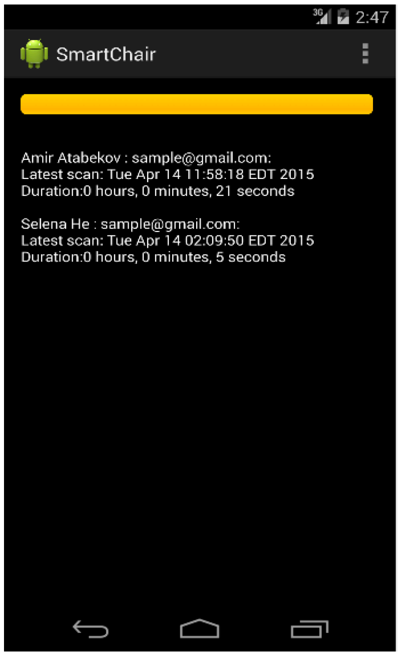


Fig. 6.13. Generated seat map


SmartChair

Amir Atabekov : sample@gmail.com:
Latest scan: Tue Apr 14 11:58:18 EDT 2015
Duration:0 hours, 0 minutes, 21 seconds

Selena He : sample@gmail.com:
Latest scan: Tue Apr 14 02:09:50 EDT 2015
Duration:0 hours, 0 minutes, 5 seconds

Fig. 6.14. Occupancy data

When "Calculate Localization" button is pressed, the Android application sends a post request with reply named "4", which indicates to the master node that localization is requested. Master node requests RSSI measurements from all slave nodes, then posts the results into "localization" channel. After doing so, the master node then sends a reply named "1" to the channel named "status". Meanwhile the Android application keeps monitoring the "status" channel for a reply of "1". As soon as it sees that a reply of "1" has been posted, it then proceeds to retrieve the posted data from "localization" channel. Simultaneously it also retrieves the latest data from the "scan", "users" and "occupancy" channels to display latest occupancy information about the chair as well as user information. Fig 6.15 illustrates the data flow from different channels.

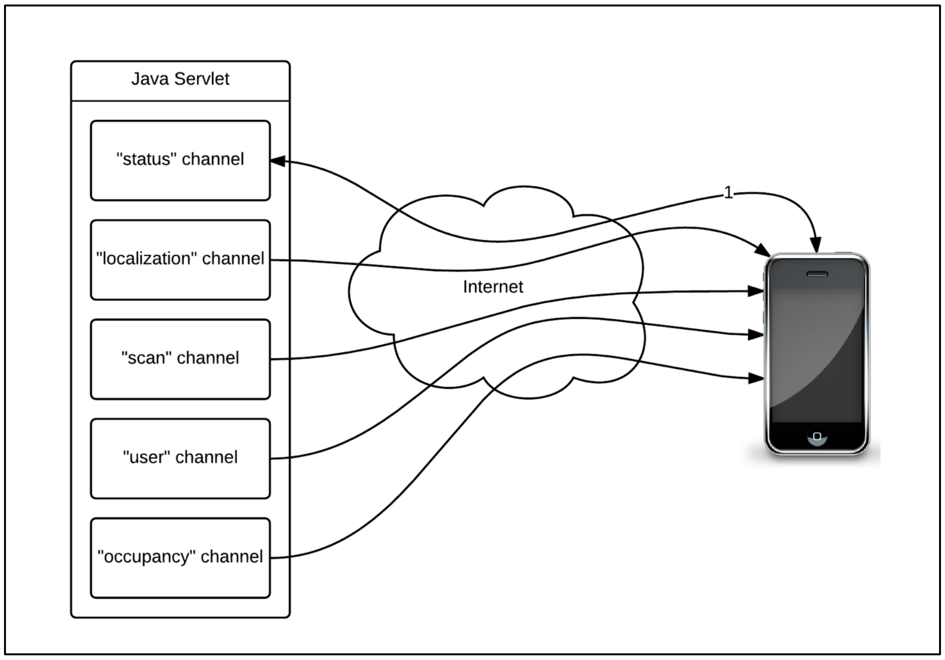


Fig. 6.15. Data flow from different channels

The data from “scan”, “user” and “occupancy” channel is aggregated to display easy to read occupancy information when a chair is tapped on the screen. UML Diagram of the application is illustrated in Fig. 6.16.

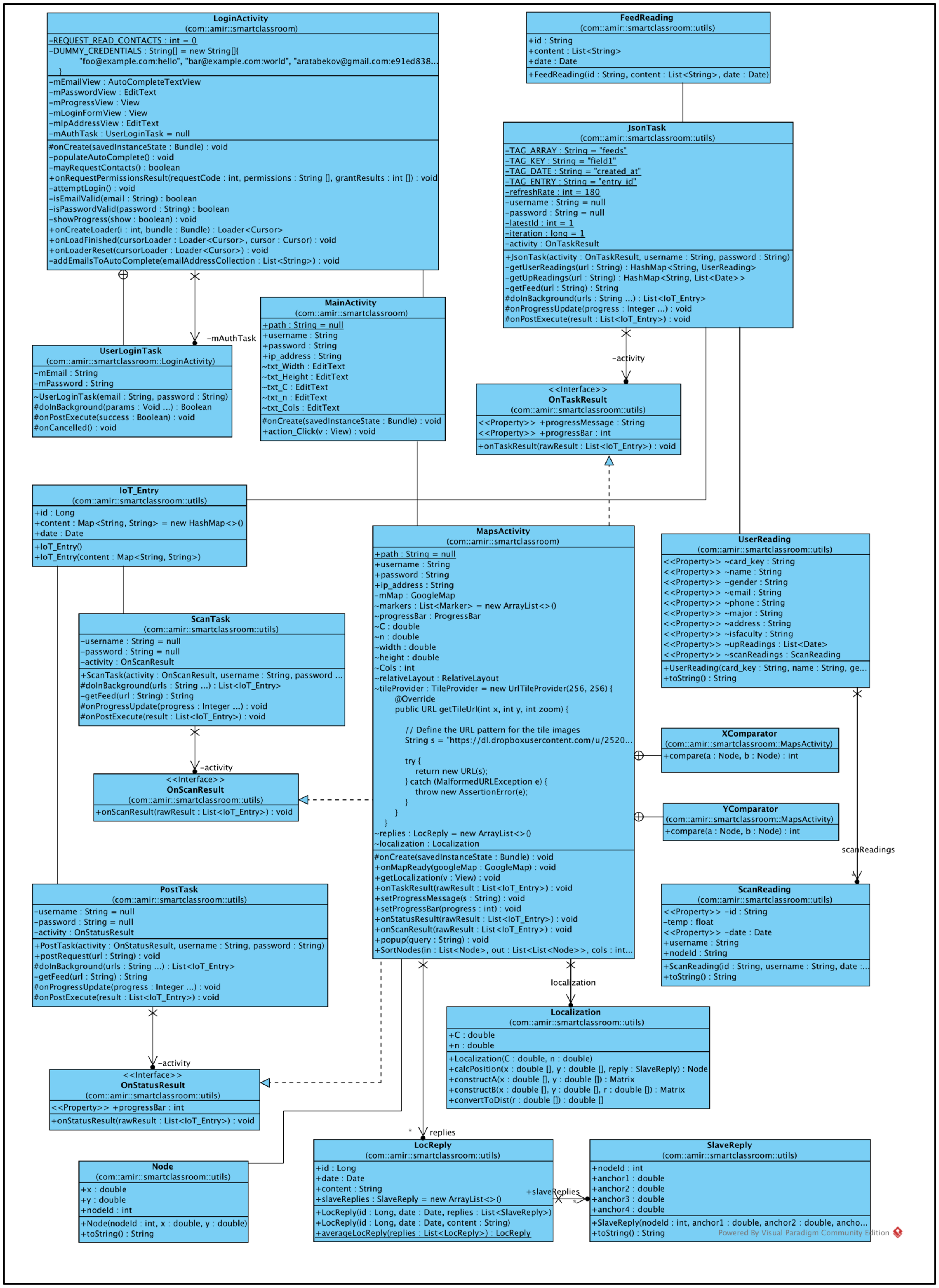

LoginActivity
(com::amir::smartclassroom)
-REQUEST_READ_CONTACTS : int = 0
-DUMMY_CREDENTIALS : String[] = new String[]{
"foo@example.com:hello", "bar@example.com:world", "aratabekov@gmail.com:e91ed838...
}
-mEmailView : AutoCompleteTextView
-mPasswordView : EditText
-mProgressView : View
-mLoginFormView : View
-mIpAddressView : EditText
-mAuthTask : UserLoginTask = null
#onCreate(savedInstanceState : Bundle) : void
-populateAutoComplete() : void
-mayRequestContacts() : boolean
+onRequestPermissionsResult(requestCode : int, permissions : String [], grantResults : int []) : void
-attemptLogin() : void
-isEmailValid(email : String) : boolean
-isPasswordValid(password : String) : boolean
-showProgress(show : boolean) : void
+onCreateLoader(i : int, bundle : Bundle) : Loader<Cursor>
+onLoadFinished(cursorLoader : Loader<Cursor>, cursor : Cursor) : void
+onLoaderReset(cursorLoader : Loader<Cursor>) : void
-addEmailsToAutoComplete(emailAddressCollection : List<String>) : void
FeedReading
(com::amir::smartclassroom::utils)
+id : String
+content : List<String>
+date : Date
+FeedReading(id : String, content : List<String>, date : Date)
JsonTask
(com::amir::smartclassroom::utils)
-TAG_ARRAY : String = "feeds"
-TAG_KEY : String = "field1"
-TAG_DATE : String = "created_at"
-TAG_ENTRY : String = "entry_id"
-refreshRate : int = 180
-username : String = null
-password : String = null
-latestId : int = 1
-iteration : long = 1
-activity : OnTaskResult
+JsonTask(activity : OnTaskResult, username : String, password : String)
-getUserReadings(url : String) : HashMap<String, UserReading>
-getUpReadings(url : String) : HashMap<String, List<Date>>
-getFeed(url : String) : String
#doInBackground(urls : String ...) : List<IoT_Entry>
#onProgressUpdate(progress : Integer ...) : void
#onPostExecute(result : List<IoT_Entry>) : void
-mAuthTask
UserLoginTask
(com::amir::smartclassroom::LoginActivity)
-mEmail : String
-mPassword : String
~UserLoginTask(email : String, password : String)
#doInBackground(params : Void ...) : Boolean
#onPostExecute(success : Boolean) : void
#onCancelled() : void
MainActivity
(com::amir::smartclassroom)
+path : String = null
+username : String
+password : String
+ip_address : String
~txt_Width : EditText
~txt_Height : EditText
~txt_C : EditText
~txt_n : EditText
~txt_Cols : EditText
#onCreate(savedInstanceState : Bundle) : void
+action_Click(v : View) : void
-activity
<<Interface>>
OnTaskResult
(com::amir::smartclassroom::utils)
<<Property>> +progressMessage : String
<<Property>> +progressBar : int
+onTaskResult(rawResult : List<IoT_Entry>) : void
IoT_Entry
(com::amir::smartclassroom::utils)
+id : Long
+content : Map<String, String> = new HashMap<>()
+date : Date
+IoT_Entry()
+IoT_Entry(content : Map<String, String>)
MapsActivity
(com::amir::smartclassroom)
+path : String = null
+username : String
+password : String
+ip_address : String
-mMap : GoogleMap
~markers : List<Marker> = new ArrayList<>()
~progressBar : ProgressBar
~C : double
~n : double
~width : double
~height : double
~Cols : int
~relativeLayout : RelativeLayout
~tileProvider : TileProvider = new UrlTileProvider(256, 256) {
@Override
public URL getTileUrl(int x, int y, int zoom) {
// Define the URL pattern for the tile images
String s = "https://dl.dropboxusercontent.com/u/2520...
try {
return new URL(s);
} catch (MalformedURLException e) {
throw new AssertionError(e);
}
}
}
~replies : LocReply = new ArrayList<>()
~localization : Localization
#onCreate(savedInstanceState : Bundle) : void
+onMapReady(googleMap : GoogleMap) : void
+getLocalization(v : View) : void
+onTaskResult(rawResult : List<IoT_Entry>) : void
+setProgressMessage(s : String) : void
+setProgressBar(progress : int) : void
+onStatusResult(rawResult : List<IoT_Entry>) : void
+onScanResult(rawResult : List<IoT_Entry>) : void
+popup(query : String) : void
+SortNodes(in : List<Node>, out : List<List<Node>>, cols : int...
UserReading
(com::amir::smartclassroom::utils)
<<Property>> ~card_key : String
<<Property>> ~name : String
<<Property>> ~gender : String
<<Property>> ~email : String
<<Property>> ~phone : String
<<Property>> ~major : String
<<Property>> ~address : String
<<Property>> ~isfaculty : String
<<Property>> ~upReadings : List<Date>
<<Property>> ~scanReadings : ScanReading
+UserReading(card_key : String, name : String, ge...
+toString() : String
ScanTask
(com::amir::smartclassroom::utils)
-username : String = null
-password : String = null
-activity : OnScanResult
+ScanTask(activity : OnScanResult, username : String, password ...
#doInBackground(urls : String ...) : List<IoT_Entry>
-getFeed(url : String) : String
#onProgressUpdate(progress : Integer ...) : void
#onPostExecute(result : List<IoT_Entry>) : void
-activity
<<Interface>>
OnScanResult
(com::amir::smartclassroom::utils)
+onScanResult(rawResult : List<IoT_Entry>) : void
XComparator
(com::amir::smartclassroom::MapsActivity)
+compare(a : Node, b : Node) : int
YComparator
(com::amir::smartclassroom::MapsActivity)
+compare(a : Node, b : Node) : int
scanReadings
*
PostTask
(com::amir::smartclassroom::utils)
-username : String = null
-password : String = null
-activity : OnStatusResult
+PostTask(activity : OnStatusResult, username : String, password : String)
+postRequest(url : String) : void
#doInBackground(urls : String ...) : List<IoT_Entry>
-getFeed(url : String) : String
#onProgressUpdate(progress : Integer ...) : void
#onPostExecute(result : List<IoT_Entry>) : void
ScanReading
(com::amir::smartclassroom::utils)
<<Property>> ~id : String
~temp : float
<<Property>> ~date : Date
+username : String
+nodeId : String
+ScanReading(id : String, username : String, date :...
+toString() : String
localization
Localization
(com::amir::smartclassroom::utils)
+C : double
+n : double
+Localization(C : double, n : double)
+calcPosition(x : double [], y : double [], reply : SlaveReply) : Node
+constructA(x : double [], y : double []) : Matrix
+constructB(x : double [], y : double [], r : double []) : Matrix
+convertToDist(r : double []) : double []
-activity
<<Interface>>
OnStatusResult
(com::amir::smartclassroom::utils)
<<Property>> +progressBar : int
+onStatusResult(rawResult : List<IoT_Entry>) : void
*
replies
Node
(com::amir::smartclassroom::utils)
+x : double
+y : double
+nodeId : int
+Node(nodeId : int, x : double, y : double)
+toString() : String
LocReply
(com::amir::smartclassroom::utils)
+id : Long
+date : Date
+content : String
+slaveReplies : SlaveReply = new ArrayList<>()
+LocReply(id : Long, date : Date, replies : List<SlaveReply>)
+LocReply(id : Long, date : Date, content : String)
+averageLocReply(replies : List<LocReply>) : LocReply
+slaveReplies
*
SlaveReply
(com::amir::smartclassroom::utils)
+nodeId : int
+anchor1 : double
+anchor2 : double
+anchor3 : double
+anchor4 : double
+SlaveReply(nodeId : int, anchor1 : double, anchor2 : double, ancho...
+toString() : String
Powered By Visual Paradigm Community Edition

Fig. 6.16. UML diagram

CHAPTER 7

Performance evaluation

This chapter provides the results of performance evaluation performed after the system was implemented. It contains evaluation of localization accuracy and power consumption.

7.1 Localization

The localization accuracy is evaluated by performing 30 localization requests, first without correction algorithm, then with it. The chairs are placed within 2 meters of each other and shuffled before each try. Without correction algorithm, sequential position of chairs was correct 49% of the time, with correction algorithm, the position of chairs was correct 80% of the time as illustrated in Fig. 7.1. Indoor localization in itself is very difficult subject and none of the current studies on RSSI-based algorithms are able to demonstrate a reliable accuracy less than 3 meters. The big advantage the smart classroom system has over other applications is that we don't need to know the absolute position of each chair. This fact makes it possible to apply the alignment algorithm and calculate the relative position of chairs.

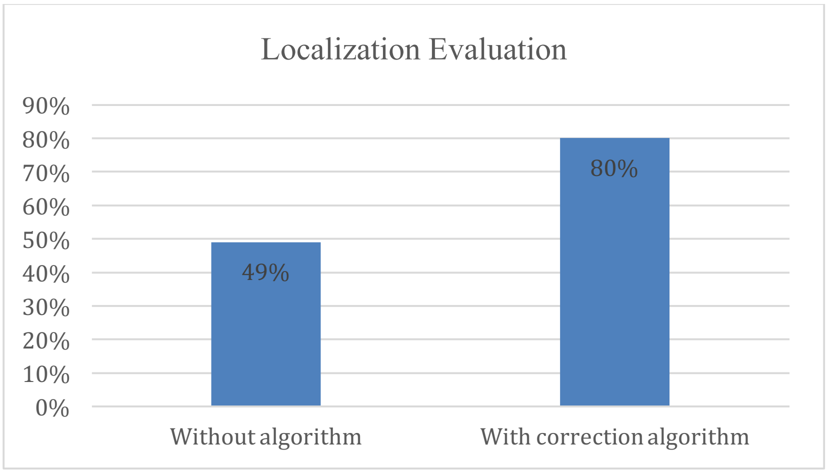


Fig. 7.1. Accuracy evaluation

This method is still inaccurate if it is applied in systems where absolute position is required, but that is out of scope of this work.

Average localization request takes 5.3 seconds to complete as measured by performing 30 random localization requests. This was timed starting from when "Calculate Localization" button is pressed until the seat map is generated.

7.2 Power consumption

Each slave node with 3000mAh battery lasts 12 hours in idle mode. In active mode, measured as sending data every 5 minutes, the battery lasts 10 hours as illustrated in Fig. 7.2. The test was performed with RFID card reader, pressure sensor and XBee module attached.

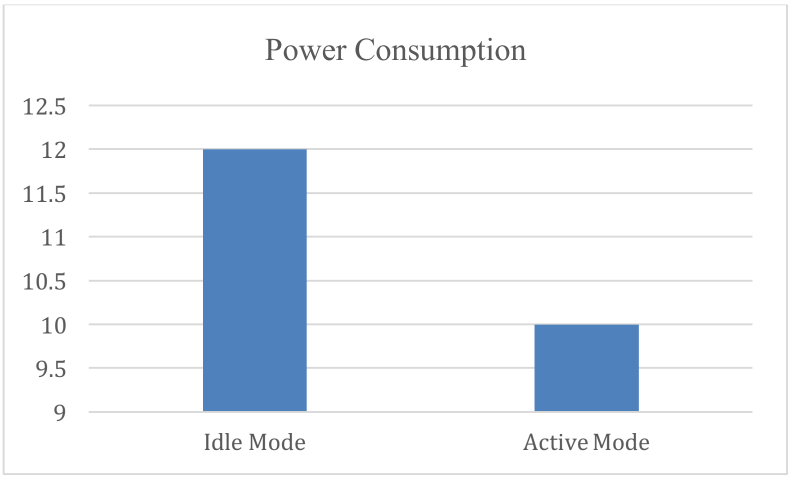


Fig. 7.2. Power consumption

The power consumption is more than adequate for this system. The 3000mAh battery a small battery which still managed to provide up to 12 hours of life. Such a low power consumption was possible because we implemented the master/slave architecture and off-loaded most of the CPU intensive calculations to master node and Android application.

Should we have chosen to perform localization on each chair's microcontroller, the battery life would have been much worse. For even better performance a battery with bigger capacity can be purchased to increase the battery life.

7.3 Reliability

Indoor localization accuracy can vary from room to room. It's crucial that the optimal path-loss exponent is chosen, which can be done based on the environment of the specific room. Usually a value from 2 to 4 is chosen, but it's best to choose the value experimentally based on characteristics of the specific room. This is a limitation of all RSSI-based algorithms, since different environments can introduce obstacles for wireless signals.

CHAPTER 8

Conclusion

The proof-of-concept smart chair system is developed based on Arduino microcontroller, demonstrates the viability of the proposed smart classroom system. The indoor localization implemented with this system can verify the accuracy of RSSI based multilateration method in a classroom setting. This smart classroom system not only offers the solution to a problem but also lays the foundation for series of future projects (such as intelligent parking system, dynamic ticketing system, *etc*.). Finally, the data collected through the system can be analyzed and used for additional purposes, such as resource management, attendance checking, or tutor time tracking management. Additionally, the localization technique described in this work can also be used to facilitate indoor localization education [42]

The smart classroom system can also greatly help students who are visually or hearing impaired, they can use the Android application to know which classmates showed up to class and if they are sitting in front or behind them so that they can communicate with them.

Since the system is based on Arduino, which is open hardware platform; the board can be further customized in the future to make it more tailored for the specific needs of smart classrooms. Furthermore, we will consider manufacturing an all-in-one board with all the components integrated, to cut costs and make it more affordable. This will make the system one step closer to commercial applications.

APPENDIX A: Arduino Source Code

Anchor Node Source Code

```
#include <SoftwareSerial.h>
#include <XBee.h>

int anchorID = 3;

SoftwareSerial mySerial =  SoftwareSerial(2, 3);
XBee xbee = XBee();
Rx16Response rx16 = Rx16Response();
int numChairs = 99;
int current = 0;

void setup()  {
  pinMode(13, OUTPUT);
  Serial.begin(9600);

  mySerial.begin(9600);
  xbee.setSerial(mySerial);
}

void loop()                    // run over and over again
{
  if (current == numChairs)
  {
    for (int i = 1; i <= numChairs; i++)
    {
      uint8_t payload[] = { 'r', 'e', 's' };
      //XBeeAddress64 addr64 = XBeeAddress64(0, 0x0000FFFF);
      int del = 100 + ((anchorID-1) * 100);
      int mydel = (500 * (i - 1)) + del;
      delay(mydel);

      int destID = i + 5;
      Serial.println("sending to " + String(destID) + " " + String(mydel));
      Tx16Request tx16 = Tx16Request(destID, payload, sizeof(payload));
      xbee.send( tx16 );
    }
    current = 0;
  }
  xbee.readPacket();
  if (xbee.getResponse().isAvailable())
  {
```

```
  if (xbee.getResponse().getApiId() == RX_16_RESPONSE)
  {
   xbee.getResponse().getRx16Response(rx16);

   String r = "";
   for (int i = 0; i < rx16.getDataLength(); i++)
   {
    r += (char)rx16.getData(i);
   }

   if (r.substring(0,4).equals("lreq")) {

    current++;
    String numChairs_str = r.substring(5, r.length());
    numChairs=numChairs_str.toInt();

   }
  }
 }
}
```

Master Node Source Code

```
#include <SoftwareSerial.h>
#include <Process.h>
#include <SPI.h>
#include <XBee.h>

String key_loc="e91ed838-2b52-4264-b807-305b9f045d92";
String key_status="a29f6f5e-4fe5-47a3-a94b-3b849ca2cac8";
SoftwareSerial mySerial =  SoftwareSerial(8, 9);
XBee xbee = XBee();
Rx16Response rx16 = Rx16Response();
int  numChairs = 4;
//This field will hold the current number or replies received up until now
int current = 0;
String replies;
bool requestPending = false;
int tries=0;
```

```
boolean mreq = false;
void setup()  {
  //while (!Serial);
  Bridge.begin();
  //pinMode(12, OUTPUT);
  Serial.begin(9600);
  // set the data rate for the SoftwareSerial port
  mySerial.begin(9600);

  xbee.setSerial(mySerial);
}

void loop()                    // run over and over again
{

  if(requestPending==false)
  {
    String reply=getRequestStatus(key_status);
    String req=reply.substring(9,10);
    if(req.equals("4"))
    {
      Serial.println(req);
      requestPending=true;
      mreq=true;
    }
    delay(500);
  }

  if (mreq == true) {
    for (int i = 1; i <= numChairs; i++)
    {
      Serial.println("sending creq");
      String reply = "creq:" + String(numChairs);
      int len = reply.length();
      uint8_t payload[len];
      for (int i = 0; i < len; i++)
      {
        payload[i] = reply[i];

      }
      Tx16Request tx16 = Tx16Request(i + 5, payload, sizeof(payload));
      xbee.send( tx16 );
    }
    mreq = false;
  }
```

```
if (current == numChairs)
{
 Serial.println(replies);
 postToLocate(key_loc, replies);

 if(tries==2){
  postToLocate(key_status,"1");
  tries=0;
 }
 else{
  tries++;
 }

 requestPending=false;
 current = 0;
 replies = "";

}
if (requestPending == true)
{
 xbee.readPacket();
 if (xbee.getResponse().isAvailable())
 {
  if (xbee.getResponse().getApiId() == RX_16_RESPONSE)
  {
   //Serial.println("available rx16 resp");
   xbee.getResponse().getRx16Response(rx16);
   //Serial.println("rssi is "+String(rx16.getRssi()));
   int P_LEN = rx16.getDataLength();
   String r = "" ;
   for (int i = 0; i < P_LEN; i++)
   {
    r += (char)rx16.getData(i);
    //Serial.print( rx16.getData(i), HEX );
   }
   //uncomment here to print the whole string

   String reply = String(r).substring(0, 6);

   if (reply.equals("creply"))
   {
    current++;
    replies += r + ";";
   }
  }
```

```
    }
  }
}

String getRequestStatus(String key)
{
  Process p;
  String cmd = "curl --data '' ";
  cmd = cmd + "'http://iot-service-1188.appspot.com/feedservlet?";
  cmd = cmd + "key=" + key + "' -s";
  cmd = cmd + " | grep -o '\"reply\":\"[0-9]\"' | head -n1";

  Serial.println(cmd);
  p.runShellCommand(cmd);

  String reply = "";
  while (p.running());
  while (p.available())
  {
    reply = p.readString();
  }
  Serial.flush();
  Console.println(cmd);
  p.close();
  return reply;
}

void postToLocate(String key, String value) {

  Process p;
  String cmd = "curl --data '' ";
  cmd = cmd + "'http://iot-service-1188.appspot.com/entryservlet?";
  cmd = cmd + "key=" + key + "&";
  cmd = cmd + "reply=" + value + "'";
  Serial.println(cmd);
  p.runShellCommand(cmd);

  while (p.running());
  while (p.available())
  {
    char c = p.read();
    Serial.print(c);
  }
  Serial.flush();
  Console.println(cmd);
  p.close();
}
```

Slave Node Source Code

```
#include <SoftwareSerial.h>
#include <XBee.h>
#include <MFRC522.h>
#include <SPI.h>

#define RST_PIN    9   //
#define SS_PIN    10   //

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

int nodeID = 6;
SoftwareSerial nss(2, 3);
XBee xbee = XBee();
Rx16Response rx16 = Rx16Response();
String uid = "";
int fsrPin = 0;     // the FSR and 10K pulldown are connected to a0
int fsrReading;

void setup()  {
  pinMode(SS_PIN, OUTPUT);
  pinMode(2, INPUT);
  pinMode(3, OUTPUT);
  Serial.begin(9600);
  while (!Serial);
  xbee.setSerial(nss);
  SPI.begin();
  // set the data rate for the SoftwareSerial port
  nss.begin(9600);

  mfrc522.PCD_Init();   // Init MFRC522
  ShowReaderDetails();  // Show details of PCD - MFRC522 Card Reader details
  Serial.println("Scan PICC to see UID, type, and data blocks...");
  //mySerial.println("Hello, world?");

}

bool request_completed = false;
bool r1 = false;
bool r2 = false;
bool r3 = false;
bool r4 = false;

String r1_rssi = "";
String r2_rssi = "";
String r3_rssi = "";
String r4_rssi = "";

bool reply_pending = false;
void loop()                    // run over and over again
{
 xbee.readPacket();
 if (xbee.getResponse().isAvailable())
 {
  //Serial.println("available");
  if (xbee.getResponse().getApiId() == RX_16_RESPONSE)
  {
   xbee.getResponse().getRx16Response(rx16);
   //Serial.println(rx16.getRssi());
   uint8_t str_rssi = rx16.getRssi();
   String r = "";

   for (int i = 0; i < rx16.getDataLength(); i++)
   {
    r += (char)rx16.getData(i);
    //Serial.print( rx16.getData(i), HEX );
   }
   if (r.equals("res"))
   {
    //Serial.println("hehehe");
    Serial.println(r + ", received from " + rx16.getRemoteAddress16() + ", rssi " +
str_rssi);
    if (rx16.getRemoteAddress16() == 1) {
     r1 = true;
     r1_rssi = str_rssi;
    }
    if (rx16.getRemoteAddress16() == 2) {
     r2 = true;
     r2_rssi = str_rssi;
    }
    if (rx16.getRemoteAddress16() == 3) {
     r3 = true;
     r3_rssi = str_rssi;
```

```
      }
      if (rx16.getRemoteAddress16() == 4) {
        r4 = true;
        r4_rssi = str_rssi;
      }

    }

    if (r.substring(0, 4).equals("creq"))
    {
      String numChairs = r.substring(5, r.length());
      reply_pending = true;;
      request_completed = false;
      Serial.println("Requesting RSSI:");
      //Sending a request to all anchor nodes
      //delay(100+(nodeID-6)*1000);
      String reply = "lreq:" + String(numChairs);
      int len = reply.length();
      uint8_t payload[len];
      for (int i = 0; i < len; i++)
      {
        payload[i] = reply[i];

      }
      for (int i = 1; i <= 4; i++)
      {
        Tx16Request tx16 = Tx16Request(i, payload, sizeof(payload));
        xbee.send( tx16 );
        //delay(1000);
      }
      //write the node specific delay here
    }
  }
 }

 if (r1 & r2 & r3 & r4 ) {
   request_completed = true;
 }
 if (request_completed == true && reply_pending == true)
 {
   String reply = "creply:" + r1_rssi + ":" + r2_rssi + ":" + r3_rssi + ":" + r4_rssi + ":"
+ nodeID;
   int len = reply.length();
   uint8_t payload[len];
   for (int i = 0; i < len; i++)
 {
   payload[i] = reply[i];

 }

 Tx16Request tx16 = Tx16Request(5, payload, sizeof(payload));
 xbee.send( tx16 );
 //requested = false;
 reply_pending = false;
 request_completed = false;
}

/*
WARNING: Starting the RFID code here
 */
fsrReading = analogRead(fsrPin);
// the FSR and 10K pulldown are connected to a0
if (fsrReading > 600 & !uid.equals(""))
{
 pinMode(6, OUTPUT);
 digitalWrite(6, HIGH);
}
else {
 pinMode(6, OUTPUT);
 digitalWrite(6, LOW);
}

// Check if new card is present
if ( ! mfrc522.PICC_IsNewCardPresent()) {
 return;
}

// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial()) {
 return;
}
String rfid = "";
for (byte i = 0; i < mfrc522.uid.size; i++) {
 rfid += mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ";
 rfid += String(mfrc522.uid.uidByte[i], HEX);
}

rfid.trim();
rfid.toUpperCase();
Serial.println("Uid is " + rfid);
```

```
 mfrc522.PICC_HaltA();

 if (uid.equals(rfid))
 {
  pinMode(5, OUTPUT);
  digitalWrite(5, LOW);
  Serial.println("Logging off");
  uid = "";
 }
 else
 {
  pinMode(5, OUTPUT);
  digitalWrite(5, HIGH);
  uid = rfid;
  Serial.println("Should be sending the data now");

  String reply = "scan:" + uid + ":" + nodeID;
  int len = reply.length();
  uint8_t payload[len];
  for (int i = 0; i < len; i++)
  {
   payload[i] = reply[i];
  }
  Tx16Request tx16 = Tx16Request(5, payload, sizeof(payload));
  xbee.send( tx16 );
 }
}

void ShowReaderDetails() {
 // Get the MFRC522 software version
 byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
 Serial.print("MFRC522 Software Version: 0x");
 Serial.print(v, HEX);
 if (v == 0x91)
  Serial.print(" = v1.0");
 else if (v == 0x92)
  Serial.print(" = v2.0");
 else
  Serial.print(" (unknown)");
 Serial.println("");
 // When 0x00 or 0xFF is returned, communication probably failed
 if ((v == 0x00) || (v == 0xFF)) {
  Serial.println("WARNING: Communication failure, is the MFRC522 properly
connected?");
 }
```

```
}
```

APPENDIX B: Android Source Code

"LoginActivity" Source Code

```
package com.amir.smartclassroom;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.annotation.TargetApi;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.app.LoaderManager.LoaderCallbacks;

import android.content.CursorLoader;
import android.content.Loader;
import android.database.Cursor;
import android.net.Uri;
import android.os.AsyncTask;

import android.os.Build;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.EditorInfo;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;

import static android.Manifest.permission.READ_CONTACTS;

/**
 * A login screen that offers login via email/password.
 */
public class LoginActivity extends AppCompatActivity implements
LoaderCallbacks<Cursor> {

    /**
     * Id to identity READ_CONTACTS permission request.
     */
    private static final int REQUEST_READ_CONTACTS = 0;

    /**
```

```
     * A dummy authentication store containing known user names and
passwords.
     * TODO: remove after connecting to a real authentication system.
     */
    private static final String[] DUMMY_CREDENTIALS = new String[]{
            "aratabekov@gmail.com:e91ed838-2b52-4264-b807-305b9f045d92"
    };
    /**
     * Keep track of the login task to ensure we can cancel it if requested.
     */
    private UserLoginTask mAuthTask = null;

    // UI references.
    private AutoCompleteTextView mEmailView;
    private EditText mPasswordView;
    private View mProgressView;
    private View mLoginFormView;
    private EditText mIpAddressView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // Set up the login form.
        mEmailView = (AutoCompleteTextView) findViewById(R.id.email);
        populateAutoComplete();

        mPasswordView = (EditText) findViewById(R.id.password);
        mPasswordView.setOnEditorActionListener(new
TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int id, KeyEvent
keyEvent) {
                if (id == R.id.login || id == EditorInfo.IME_NULL) {
                    attemptLogin();
                    return true;
                }
                return false;
            }
        });

        Button mEmailSignInButton = (Button)
findViewById(R.id.email_sign_in_button);
        mEmailSignInButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                attemptLogin();
            }
        });

        mLoginFormView = findViewById(R.id.login_form);
        mProgressView = findViewById(R.id.login_progress);
        this.mIpAddressView=(EditText)findViewById(R.id.ip_address);
    }

    private void populateAutoComplete() {
        if (!mayRequestContacts()) {
            return;
        }

        getLoaderManager().initLoader(0, null, this);
    }

    private boolean mayRequestContacts() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
            return true;
        }
        if (checkSelfPermission(READ_CONTACTS) ==
PackageManager.PERMISSION_GRANTED) {
            return true;
        }
        if (shouldShowRequestPermissionRationale(READ_CONTACTS)) {
            Snackbar.make(mEmailView, R.string.permission_rationale,
Snackbar.LENGTH_INDEFINITE)
                    .setAction(android.R.string.ok, new
View.OnClickListener() {
                        @Override
                        @TargetApi(Build.VERSION_CODES.M)
                        public void onClick(View v) {
                            requestPermissions(new String[]{READ_CONTACTS},
REQUEST_READ_CONTACTS);
                        }
                    });
        } else {
            requestPermissions(new String[]{READ_CONTACTS},
REQUEST_READ_CONTACTS);
        }
        return false;
    }

    /**
     * Callback received when a permissions request has been completed.
     */
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions,
                                           @NonNull int[] grantResults) {
        if (requestCode == REQUEST_READ_CONTACTS) {
            if (grantResults.length == 1 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                populateAutoComplete();
            }
        }
    }


    /**
     * Attempts to sign in or register the account specified by the login
form.
     * If there are form errors (invalid email, missing fields, etc.), the
     * errors are presented and no actual login attempt is made.
     */
    private void attemptLogin() {
        if (mAuthTask != null) {
            return;
        }

        // Reset errors.
        mEmailView.setError(null);
        mPasswordView.setError(null);

        // Store values at the time of the login attempt.
        String email = mEmailView.getText().toString();
        String password = mPasswordView.getText().toString();

        boolean cancel = false;
        View focusView = null;

        // Check for a valid password, if the user entered one.
        if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {

mPasswordView.setError(getString(R.string.error_invalid_password));
            focusView = mPasswordView;
            cancel = true;
        }

        // Check for a valid email address.
        if (TextUtils.isEmpty(email)) {
            mEmailView.setError(getString(R.string.error_field_required));
            focusView = mEmailView;
            cancel = true;
        } else if (!isEmailValid(email)) {
            mEmailView.setError(getString(R.string.error_invalid_email));
            focusView = mEmailView;
            cancel = true;
        }

        if (cancel) {
            // There was an error; don't attempt login and focus the first
            // form field with an error.
            focusView.requestFocus();
        } else {
            // Show a progress spinner, and kick off a background task to
            // perform the user login attempt.
            showProgress(true);
            mAuthTask = new UserLoginTask(email, password);
            mAuthTask.execute((Void) null);
        }
    }

    private boolean isEmailValid(String email) {
        //TODO: Replace this with your own logic
        return email.contains("@");
    }

    private boolean isPasswordValid(String password) {
        //TODO: Replace this with your own logic
        return password.length() > 4;
    }

    /**
     * Shows the progress UI and hides the login form.
     */
    @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
    private void showProgress(final boolean show) {
        // On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which
allow
        // for very easy animations. If available, use these APIs to fade-in
        // the progress spinner.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {
            int shortAnimTime =
getResources().getInteger(android.R.integer.config_shortAnimTime);
```

```
            mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
            mLoginFormView.animate().setDuration(shortAnimTime).alpha(
                    show ? 0 : 1).setListener(new AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    mLoginFormView.setVisibility(show ? View.GONE :
View.VISIBLE);
                }
            });

            mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            mProgressView.animate().setDuration(shortAnimTime).alpha(
                    show ? 1 : 0).setListener(new AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    mProgressView.setVisibility(show ? View.VISIBLE :
View.GONE);
                }
            });
        } else {
            // The ViewPropertyAnimator APIs are not available, so simply
show
            // and hide the relevant UI components.
            mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
        }
    }

    @Override
    public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
        return new CursorLoader(this,
                // Retrieve data rows for the device user's 'profile'
contact.
                Uri.withAppendedPath(ContactsContract.Profile.CONTENT_URI,
                        ContactsContract.Contacts.Data.CONTENT_DIRECTORY),
ProfileQuery.PROJECTION,

                // Select only email addresses.
                ContactsContract.Contacts.Data.MIMETYPE +
                        " = ?", new
String[]{ContactsContract.CommonDataKinds.Email
                .CONTENT_ITEM_TYPE},

                // Show primary email addresses first. Note that there won't
be
                // a primary email address if the user hasn't specified one.
                ContactsContract.Contacts.Data.IS_PRIMARY + " DESC");
    }

    @Override
    public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor) {
        List<String> emails = new ArrayList<>();
        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {
            emails.add(cursor.getString(ProfileQuery.ADDRESS));
            cursor.moveToNext();
        }

        addEmailsToAutoComplete(emails);
    }
    @Override
    public void onLoaderReset(Loader<Cursor> cursorLoader) {

    }

    private void addEmailsToAutoComplete(List<String> emailAddressCollection)
{
        //Create adapter to tell the AutoCompleteTextView what to show in its
dropdown list.
        ArrayAdapter<String> adapter =
                new ArrayAdapter<>(LoginActivity.this,
                        android.R.layout.simple_dropdown_item_1line,
emailAddressCollection);

        mEmailView.setAdapter(adapter);
    }


    private interface ProfileQuery {
        String[] PROJECTION = {
                ContactsContract.CommonDataKinds.Email.ADDRESS,
                ContactsContract.CommonDataKinds.Email.IS_PRIMARY,
        };

        int ADDRESS = 0;
        int IS_PRIMARY = 1;
    }

    /**
     * Represents an asynchronous login/registration task used to
authenticate
     * the user.
     */
    public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

        private final String mEmail;
        private final String mPassword;

        UserLoginTask(String email, String password) {
            mEmail = email;
            mPassword = password;
        }

        @Override
        protected Boolean doInBackground(Void... params) {
            // TODO: attempt authentication against a network service.

            try {
                // Simulate network access.
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                return false;
            }

            for (String credential : DUMMY_CREDENTIALS) {
                String[] pieces = credential.split(":");
                if (pieces[0].equals(mEmail)) {
                    // Account exists, return true if the password matches.
                    return pieces[1].equals(mPassword);
                }
            }
            // TODO: register the new account here.
            return true;
        }

        @Override
        protected void onPostExecute(final Boolean success) {
            mAuthTask = null;
            showProgress(false);

            if (success) {
                //finish();
                Intent intent = new
Intent(LoginActivity.this,MainActivity.class);

intent.putExtra("IP_ADDRESS",LoginActivity.this.mIpAddressView.getText().toSt
ring());
                intent.putExtra("USERNAME",mEmail);
                intent.putExtra("PASSWORD",mPassword);
                LoginActivity.this.startActivity(intent);
            } else {

mPasswordView.setError(getString(R.string.error_incorrect_password));
                mPasswordView.requestFocus();
            }
        }

        @Override
        protected void onCancelled() {
            mAuthTask = null;
            showProgress(false);
        }
    }
}
```

“MainActivity” Source Code

```
package com.amir.smartclassroom;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity  {

    public static String path = null;

    public String username;
    public String password;
    public String ip_address;

    EditText txt_Width;
    EditText txt_Height;
    EditText txt_C;
    EditText txt_n;
    EditText txt_Cols;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);


        this.txt_Width=(EditText)findViewById(R.id.editTxt_Width);
        this.txt_Height=(EditText)findViewById(R.id.editTxt_Height);

        this.txt_C=(EditText)findViewById(R.id.editTxt_C);
        this.txt_n=(EditText)findViewById(R.id.editTxt_N);

        this.txt_Cols=(EditText)findViewById(R.id.editTxt_Cols);
        //String ip_address;

        if (savedInstanceState == null) {
            Bundle extras = getIntent().getExtras();
            if(extras == null) {
                ip_address= null;
                username=null;
                password=null;
            } else {
                ip_address= extras.getString("IP_ADDRESS");
                username=extras.getString("USERNAME");
                password=extras.getString("PASSWORD");
                path =  "http://"+ip_address;
            }
        } else {
            ip_address= (String)
savedInstanceState.getSerializable("IP_ADDRESS");
            username=(String) savedInstanceState.getSerializable("USERNAME");
            password=(String) savedInstanceState.getSerializable("PASSWORD");
            path = "http://"+ip_address;
        }

        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);

    }

    public void action_Click(View v) {

        double  width=Double.valueOf(txt_Width.getText().toString());
        double  height=Double.valueOf(txt_Height.getText().toString());

        double  C=Double.valueOf(txt_C.getText().toString());
        double  n=Double.valueOf(txt_n.getText().toString());

        int cols=Integer.valueOf(txt_Cols.getText().toString());

        Intent intent = new Intent(MainActivity.this,MapsActivity.class);
        intent.putExtra("IP_ADDRESS",ip_address);
        intent.putExtra("USERNAME",username);
        intent.putExtra("PASSWORD",password);
        intent.putExtra("width",width);
        intent.putExtra("height",height);

        intent.putExtra("C",C);
        intent.putExtra("n",n);
        intent.putExtra("cols",cols);
        MainActivity.this.startActivity(intent);

    }
}
```

"MapsActivity" Source Code

```
package com.amir.smartclassroom;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.support.design.widget.Snackbar;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.RelativeLayout;

import com.amir.smartclassroom.utils.IoT_Entry;
import com.amir.smartclassroom.utils.JsonTask;
import com.amir.smartclassroom.utils.LocReply;
import com.amir.smartclassroom.utils.Localization;
import com.amir.smartclassroom.utils.Node;
import com.amir.smartclassroom.utils.OnScanResult;
import com.amir.smartclassroom.utils.OnStatusResult;
import com.amir.smartclassroom.utils.OnTaskResult;
import com.amir.smartclassroom.utils.PostTask;
import com.amir.smartclassroom.utils.ScanReading;
import com.amir.smartclassroom.utils.ScanTask;
import com.amir.smartclassroom.utils.SlaveReply;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.TileOverlayOptions;
import com.google.android.gms.maps.model.TileProvider;
import com.google.android.gms.maps.model.UrlTileProvider;

import java.net.MalformedURLException;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
```

```java
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;


public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback, OnTaskResult,OnStatusResult, OnScanResult {

    public static String path = null;
    public String username;
    public String password;
    public String ip_address;
    private GoogleMap mMap;
    List<Marker> markers=new ArrayList<>();
    List<LocReply> replies=new ArrayList<>();

    ProgressBar progressBar;

    double C; // Reference RSSI value
    double n ; // Path-loss exponent
    double width;
    double height;
    int Cols;

    RelativeLayout relativeLayout;
    Localization localization;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is
ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);


this.relativeLayout=(RelativeLayout)findViewById(R.id.relativeLayout);
        this.progressBar=(ProgressBar)(findViewById(R.id.progress_bar_maps));

        double w,h,C,n;
        int cols;
        if (savedInstanceState == null) {
            Bundle extras = getIntent().getExtras();
            if (extras == null) {
                ip_address = null;
                username = null;
                password = null;
                w=0;
                h=0;
                C=0;
                n=0;
                cols=0;
            } else {
                ip_address = extras.getString("IP_ADDRESS");
                username = extras.getString("USERNAME");
                password = extras.getString("PASSWORD");
                path = "http://" + ip_address;
```

```
                 w=extras.getDouble("width");
                 h=extras.getDouble("height");
                 C=extras.getDouble("C");
                 n=extras.getDouble("n");
                 cols=extras.getInt("cols");
            }
        } else {
            ip_address = (String)
savedInstanceState.getSerializable("IP_ADDRESS");
            username = (String)
savedInstanceState.getSerializable("USERNAME");
            password = (String)
savedInstanceState.getSerializable("PASSWORD");
            path = "http://" + ip_address;

             w=(double)savedInstanceState.getSerializable("width");
             h=(double)savedInstanceState.getSerializable("height");
             C=(double)savedInstanceState.getSerializable("C");
             n=(double)savedInstanceState.getSerializable("n");
             cols=(int)savedInstanceState.getSerializable("cols");
        }

        this.width=w;
        this.height=h;
        this.C=C;
        this.n=n;
        this.Cols=cols;

        this.localization=new Localization(C,n);

    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the
camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will
be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered
once the user has
     * installed Google Play services and returned to the app.
     */

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        // Add a marker in Sydney and move the camera
        LatLng latLng = new LatLng(0, 0);
        mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15));
        //mMap.addMarker(new MarkerOptions().position(sydney).title("Marker
in Sydney"));

        mMap.addTileOverlay(new
TileOverlayOptions().tileProvider(tileProvider));
        //mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }

    public void getLocalization(View v) {
        progressBar.setVisibility(View.VISIBLE);
        PostTask task = new PostTask(this, username, "a29f6f5e-4fe5-47a3-
a94b-3b849ca2cac8");
        task.execute(path);
    }

    public void onTaskResult(List<IoT_Entry> rawResult) {
        Log.d("debug", rawResult.toString());

        String resultStr = "\n\n";

        replies.clear();
        markers.clear();
        for(Marker marker:markers){
            marker.remove();
        }

        for (int i = 0; i < 3; i++) {
            IoT_Entry entry = rawResult.get(i);
            //resultStr+=entry.content.get("reply").toString()+"\n";
            LocReply reply = new LocReply(entry.id, entry.date,
entry.content.get("reply").toString());
            //popup(entry.content.get("reply").toString());
            replies.add(reply);

        }
        LocReply reply=LocReply.averageLocReply(replies);

        System.out.println();
        List<Node> res=new ArrayList<>();

        double[] x={0, this.width , 0, this.width};
        double[] y={0, 0, this.height, this.height};

        System.out.println("ontask");
        System.out.println(width);
        System.out.println(height);
        System.out.println(C);
        System.out.println(n);
        System.out.println(Cols);

        for(SlaveReply rep: reply.slaveReplies){

            System.out.println("rep "+rep.toString());
            Node X=localization.calcPosition(x,y,rep);
            res.add(X);
            System.out.println("loc of "+(rep.nodeId-5));
            System.out.println(X);
            System.out.println();
        }

        List<List< Node>> outList=new ArrayList<>();
        SortNodes(res, outList, Cols);

        System.out.println("sorted");

        int rows=0;
        for(List<Node> line:outList){

            int cols=0;
            for(Node col: line){
                System.out.print(col);
                //Marker marker=new Marker(new MarkerOptions().position(new
LatLng(rows, cols)).title(col.nodeId+""));
                Marker marker= mMap.addMarker(new
MarkerOptions().position(new LatLng(-rows, cols)).title(col.nodeId +
"").icon(BitmapDescriptorFactory.fromResource(R.drawable.chair)));
                markers.add(marker);
                cols++;
            }
            System.out.println();
            rows++;
        }

        LatLngBounds.Builder b = new LatLngBounds.Builder();
        for (Marker m : markers) {
            b.include(m.getPosition());
        }
        LatLngBounds bounds = b.build();
        //Change the padding as per needed
        int padding = 200; // offset from edges of the map in pixels
        CameraUpdate cu = CameraUpdateFactory.newLatLngBounds(bounds,
padding);
        mMap.animateCamera(cu);

        Snackbar.make(relativeLayout, "Request completed",
Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();

        progressBar.setVisibility(View.INVISIBLE);

        ScanTask task = new ScanTask(this, username, "924fe7bd-7fd9-4acd-
91cb-f9750c81e6dc");
        task.execute(path);

    }

    public void setProgressMessage(final String s) {

    }

    public void setProgressBar(int progress) {
            this.progressBar.setProgress(progress);
    }

    public void onStatusResult(List<IoT_Entry> rawResult){
        System.out.println("Request is ready");
        JsonTask task = new JsonTask(this, username, password);
        task.execute(path);
        //popup("completed");
    }

    public void onScanResult(List<IoT_Entry> rawResult){
        //System.out.println("Request is ready");
        //popup("completed");

        System.out.println("before reading");
        HashMap<String, ScanReading> readingHashMap=new HashMap<>();

        for (int i = 0; i < rawResult.size(); i++) {
            IoT_Entry entry = rawResult.get(i);
            String nodeID=entry.content.get("node");
            String user=entry.content.get("user");

            //popup(entry.content.get("reply").toString());
            System.out.println("Here is the reading ");
            System.out.println(entry.content.toString());
            System.out.println(nodeID);
            System.out.println(user);
            ScanReading reading= new
ScanReading(entry.id.toString(),user.replace("%20"," "),entry.date,nodeID);
            readingHashMap.put(nodeID,reading);
            //break;
        }

        for(Marker marker:markers) {

            SimpleDateFormat dt = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.S");
            marker.setSnippet("Last sat: " +
readingHashMap.get(marker.getTitle()).username + " on " +
dt.format(readingHashMap.get(marker.getTitle()).getDate()));

        }
    }

    public void popup(String query) {
        new AlertDialog.Builder(this)
                .setTitle("")
                .setMessage(query)
                .setPositiveButton(android.R.string.yes, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        // continue with delete
                    }
                })
                .setNegativeButton(android.R.string.no, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        // do nothing
                    }
                })
                .setIcon(android.R.drawable.ic_dialog_alert)
                .show();
    }

    TileProvider tileProvider = new UrlTileProvider(256, 256) {
        @Override
        public URL getTileUrl(int x, int y, int zoom) {

            // Define the URL pattern for the tile images
            String s =
"https://dl.dropboxusercontent.com/u/25207350/tile.png";

            try {
                return new URL(s);
            } catch (MalformedURLException e) {
                throw new AssertionError(e);
            }
        }
    };
    public  void SortNodes(List<Node> in, List<List<Node>> out , int cols){
        //in size 4, cols 3
        int added=0;
        if(!in.isEmpty()){
            Collections.sort(in, new YComparator());
            List<Node> subList=new ArrayList<>();

            for(int i=0;i<cols;i++) {
                //1,  0
                if(in.size()>i){
                    subList.add(in.get(i));
                    added++;
                }
            }
            Collections.sort(subList,new XComparator());
            System.out.println(subList);
            out.add(subList);
            SortNodes(in.subList(added, in.size()),out, cols);
        }
    }

    class YComparator implements Comparator<Node> {
        @Override
        public int compare(Node a, Node b) {
            return a.y < b.y ? -1 : a.y == b.y ? 0 : 1;
        }
    }

    class XComparator implements Comparator<Node> {
        @Override
        public int compare(Node a, Node b) {
            return a.x < b.x ? -1 : a.x == b.x ? 0 : 1;
        }
    }
}
```

“Localization” Source Code

```
package com.amir.smartclassroom.utils;

import Jama.Matrix;

/**
 * Created by Amir on 2/25/16.
 */
public class Localization {


    public double C;
    public double n;
    public Localization(double C, double n ){
        this.C=C;
        this.n=n;
    }

    public  Node calcPosition(double[] x, double[] y, SlaveReply reply) {
        Matrix A=constructA(x,y);
        double[] r={
reply.anchor1,reply.anchor2,reply.anchor3,reply.anchor4};

        System.out.println("dist of node "+(reply.nodeId-5));
        r=convertToDist(r);

        Matrix b=constructB(x,y,r);

        A=A.times(2);
        Matrix X=A.solve(b);

        Node node=new Node(reply.nodeId,X.get(0,0),X.get(1,0));
        //return X;
        return node;
    }
    public  Matrix constructA(double[] x, double[] y){

        double[][] outAr=new double[x.length-1][2];

        for(int i=0;i<x.length-1;i++){
            int last=x.length-1;
            outAr[i][0]=x[last]-x[i];
            outAr[i][1]=y[last]-y[i];
        }
        return new Matrix(outAr);
    }

    public  Matrix constructB(double[] x, double[] y, double[] r){

        double[][] outAr=new double[x.length-1][1];

        for(int i=0;i<x.length-1;i++){
            int last=x.length-1;

            double rdiff= Math.pow(r[i],2)- Math.pow(r[last],2);
            double xdiff=Math.pow(x[i],2)-Math.pow(x[last],2);
            double ydiff=Math.pow(y[i],2)-Math.pow(y[last],2);
            outAr[i][0]=rdiff-xdiff-ydiff;

        }
        return new Matrix(outAr);
    }

    public  double[] convertToDist(double[] r){
        for(int i=0;i<r.length;i++){

            r[i]= Math.pow(10d, ((double)(C + r[i]) / (10 * n)));
            System.out.println("dist "+r[i]);
        }
        return r;
    }
}
```

“FeedReading” Source Code

```java
package com.amir.smartclassroom.utils;

import java.util.Date;
import java.util.List;

/**
 * Created by Amir on 1/26/16.
 */
public class FeedReading {

    public String id;
    public List<String> content;
    public Date date;

    public FeedReading(String id, List<String> content, Date date){
        this.id=id;
        this.content=content;
        this.date=date;
    }
}
```

"IoT_Entry" Source Code

```java
package com.amir.smartclassroom.utils;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;

/**
 * Created by Amir on 1/26/16.
 */
public class IoT_Entry {
    public Long id;
    public Map<String, String> content=new HashMap<>();
    public Date date;
    public IoT_Entry() {
        date = new Date();
    }

    /**
     * A connivence constructor
     **/
    public IoT_Entry(Map<String, String> content) {
        this();
        this.content = content;
    }
}
```

"JsonTask" Source Code

```java
package com.amir.smartclassroom.utils;
```

```
/**
 * Created by Amir on 1/26/16.
 */
import android.os.AsyncTask;
import android.util.Log;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;

import org.codehaus.jackson.JsonFactory;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;
import org.json.JSONArray;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.TimeZone;

/**
 * Created by Amir on 10/22/14.
 */
public class JsonTask extends AsyncTask<String,Integer,List<IoT_Entry>>{

    private static final String TAG_ARRAY = "feeds";
    private static final String TAG_KEY = "field1";
    private static final String TAG_DATE = "created_at";
    private static final String TAG_ENTRY = "entry_id";
    private static final int refreshRate = 180; //in seconds

    private String username=null;
    private String password=null;

    private OnTaskResult activity;
    private static int latestId = 1;
    private static long iteration = 1;

    public JsonTask(OnTaskResult activity, String username,String password){
        this.activity = activity;
        this.username=username;
        this.password=password;
    }

    private HashMap<String,UserReading> getUserReadings(String url){

        HashMap<String,UserReading> userMap=new HashMap<String,
UserReading>();
        DefaultHttpClient httpclient = new DefaultHttpClient();
        HttpGet httpget = new HttpGet(url);

        InputStream inputStream = null;
        String result = null;
        try {

            HttpResponse response = httpclient.execute(httpget);
            HttpEntity entity = response.getEntity();
            inputStream = entity.getContent();
            // json is UTF-8 by default
            BufferedReader reader = new BufferedReader(new 
InputStreamReader(inputStream, "UTF-8"), 8);
            StringBuilder sb = new StringBuilder();

            String line = null;
            while ((line = reader.readLine()) != null)
            {
                sb.append(line + "\n");
            }
            result = sb.toString();

            Log.d("doing",result);
            JSONObject jsonObj = new JSONObject(result);
            JSONArray temp_Array = jsonObj.getJSONArray(TAG_ARRAY);

            this.activity.setProgressMessage("Retrieving data...");

            for (int i = 0; i < temp_Array.length(); i++) {

                publishProgress((int) ((i / (float) temp_Array.length()) * 
100));
                JSONObject reading = temp_Array.getJSONObject(i);
                String id = reading.getString(TAG_ENTRY);
                // String key = reading.getString(TAG_KEY);
                // float temp = Float.valueOf(reading.getString(TAG_TEMP));
                String str_date = reading.getString(TAG_DATE);
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss'Z'");
                //javax.xml.bind.DatatypeConverter.parseDateTime(str_date)
                Date utcdate = sdf.parse(str_date);
                Date date = new Date(utcdate.getTime() + 
TimeZone.getTimeZone("EST").getRawOffset());
                //Date date = sdf.parse(str_date);

                userMap.put(reading.getString("field1"),new 
UserReading(reading.getString("field1"),reading.getString("field2"),

reading.getString("field3"),reading.getString("field4"),reading.getString("fi
eld5"),reading.getString("field6"),

reading.getString("field7"),reading.getString("field8"),null));

            }
        } catch (Exception e) {
            // Oops
            Log.d("Displaying","something went wrong "+e.toString());
        }
        finally {
            try{if(inputStream != null)inputStream.close();}catch(Exception 
squish){}
        }
        return userMap;
    }
    private HashMap<String,List<Date>> getUpReadings(String url){


        HashMap<String,List<Date>> upMap=new HashMap<String, List<Date>>();
        //List<ScanReading> scanList=new ArrayList<ScanReading>();

        DefaultHttpClient httpclient = new DefaultHttpClient();
        HttpGet httpget = new HttpGet(url);


        InputStream inputStream = null;
        String result = null;
        try {

            HttpResponse response = httpclient.execute(httpget);
            HttpEntity entity = response.getEntity();
            inputStream = entity.getContent();
            // json is UTF-8 by default
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream, "UTF-8"), 8);
            StringBuilder sb = new StringBuilder();

            String line = null;
            while ((line = reader.readLine()) != null)
            {
                sb.append(line + "\n");
            }
            result = sb.toString();

            Log.d("doing",result);
            JSONObject jsonObj = new JSONObject(result);
            JSONArray temp_Array = jsonObj.getJSONArray(TAG_ARRAY);

            this.activity.setProgressMessage("Retrieving data...");

            for (int i = 0; i < temp_Array.length(); i++) {

                publishProgress((int) ((i / (float) temp_Array.length()) *
100));
                JSONObject reading = temp_Array.getJSONObject(i);
                String id = reading.getString(TAG_ENTRY);
                String key = reading.getString(TAG_KEY);
                // float temp = Float.valueOf(reading.getString(TAG_TEMP));
                String str_date = reading.getString(TAG_DATE);
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss'Z'");
                //javax.xml.bind.DatatypeConverter.parseDateTime(str_date)
                Date utcdate = sdf.parse(str_date);
                Date date = new Date(utcdate.getTime() +
TimeZone.getTimeZone("EST").getRawOffset());
                //Date date = sdf.parse(str_date);

                if(upMap.containsKey(key)){
                    upMap.get(key).add( date);
                }
                else {
                    List<Date> list=new ArrayList<Date>() ;
                    list.add(date);
                    upMap.put(key, list);
                }
            }
        } catch (Exception e) {
            // Oops
            Log.d("Displaying","something went wrong "+e.toString());
        }
        finally {
            try{if(inputStream != null)inputStream.close();}catch(Exception
squish){}
        }
        return upMap;
    }

    private String getFeed(String url){

        HashMap<String,FeedReading> feedMap=new HashMap<String,
FeedReading>();
        DefaultHttpClient httpclient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(url);

        InputStream inputStream = null;
        String result = null;
        try {

            HttpResponse response = httpclient.execute(httppost);
            HttpEntity entity = response.getEntity();
            inputStream = entity.getContent();
            // json is UTF-8 by default
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream, "UTF-8"), 8);
            StringBuilder sb = new StringBuilder();
            //sb.append("feed:");
            String line = null;
            while ((line = reader.readLine()) != null)
            {
                sb.append(line + "\n");
            }
            result = sb.toString();

        } catch (Exception e) {
            // Oops
            Log.d("Displaying","something went wrong "+e.toString());
        }
        finally {
            try{if(inputStream != null)inputStream.close();}catch(Exception
squish){}
        }
        return result;
    }

    protected List<IoT_Entry> doInBackground(String... urls)  {
        //if not first iteration, wait for a while to get new data
        publishProgress(50);
        if (iteration > 1) {
            try {
                Thread.sleep(refreshRate*1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        List<IoT_Entry> list=new ArrayList<>();
        try {
            String feed = getFeed(urls[0] + "/feedservlet?key="+password);
            JsonFactory f = new JsonFactory();
            JsonParser parser = f.createJsonParser(feed);
            ObjectMapper mapper = new ObjectMapper();
            List<IoT_Entry> entries = mapper.readValue(feed, new 
TypeReference<List<IoT_Entry>>(){});

            for(IoT_Entry entry: entries){
                list.add(entry);
            }
        }
        catch(Exception e){
            e.printStackTrace();
        }
        publishProgress(100);
        Log.d("Displaying","displaying the "+list);
        return list;
    }

    protected void onProgressUpdate(Integer... progress) {
        activity.setProgressBar(progress[0]);

    }

    //will pass only latest results to onTempResult. First time latestId is 0
    protected void onPostExecute(List<IoT_Entry> result) {
        this.activity.setProgressMessage("Processing data...");
        activity.onTaskResult(result);
    }

}
```

“LocReply” Source Code

```
package com.amir.smartclassroom.utils;

import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;

/**
 * Created by Amir on 2/24/16.
 */
public class LocReply {

    public Long id;
    public Date date;
    public String content;
    public List<SlaveReply> slaveReplies=new ArrayList<>();

    public LocReply(Long id, Date date, List<SlaveReply> replies){
        this.id=id;
        this.date=date;
        this.slaveReplies=replies;
    }
    public LocReply(Long id, Date date, String content){
        this.id=id;
        this.date=date;
        this.content=content;

        //starting the stuff to delimit the string
        content=content.substring(0,content.length()-1);
        System.out.println(content);

        String[] array=content.split(";");

        TreeMap<String,String> map=new TreeMap<>();

        for(String ar :array){
            System.out.println(ar);
            String[] items=ar.split(":");
            map.put(items[5],ar);
        }

        //System.out.println("outputting the sorted stuff");
        Iterator<Map.Entry<String,String>> it=map.entrySet().iterator();
        while(it.hasNext()){
            Map.Entry<String,String> entry=it.next();
            String[] items=entry.getValue().split(":");
            slaveReplies.add(new
SlaveReply(Integer.valueOf(items[5]),Integer.valueOf(items[1]),
Integer.valueOf(items[2]),
                    Integer.valueOf(items[3]), Integer.valueOf(items[4])));
        }
        // System.out.print("replies\n" +slaveReplies.toString());
    }

    public static  LocReply averageLocReply(List<LocReply> replies){

        List<SlaveReply> list=new ArrayList<>();

        //<SlaveID, RSSI>
        HashMap<Integer,Double> total_anchor1=new HashMap<>();
        HashMap<Integer,Double> total_anchor2=new HashMap<>();
        HashMap<Integer,Double> total_anchor3=new HashMap<>();
        HashMap<Integer,Double> total_anchor4=new HashMap<>();

        for(LocReply reply: replies){

            for(SlaveReply slave : reply.slaveReplies){
                //anchor 1
                if(total_anchor1.containsKey(slave.nodeId)){
                    double d=total_anchor1.get(slave.nodeId);
                    total_anchor1.put(slave.nodeId,slave.anchor1+d);
                }
                else
                    total_anchor1.put(slave.nodeId, (double)slave.anchor1);

                //anchor 2
                if(total_anchor2.containsKey(slave.nodeId)){
                    double d=total_anchor2.get(slave.nodeId);
                    total_anchor2.put(slave.nodeId,slave.anchor2+d);
                }
```

```
                 else
                     total_anchor2.put(slave.nodeId,(double)slave.anchor2);

                 //anchor 3
                 if(total_anchor3.containsKey(slave.nodeId)){
                     double d=total_anchor3.get(slave.nodeId);
                     total_anchor3.put(slave.nodeId,slave.anchor3+d);
                 }
                 else
                     total_anchor3.put(slave.nodeId,(double)slave.anchor3);

                 //anchor 4
                 if(total_anchor4.containsKey(slave.nodeId)){
                     double d=total_anchor4.get(slave.nodeId);
                     total_anchor4.put(slave.nodeId,slave.anchor4+d);
                 }
                 else
                     total_anchor4.put(slave.nodeId,(double)slave.anchor4);
             }
         }
         Set<Integer> keys=total_anchor1.keySet();
         for(int key :keys){

             double avg1=(total_anchor1.get(key)/replies.size());
             double avg2=(total_anchor2.get(key)/replies.size());
             double avg3=(total_anchor3.get(key)/replies.size());
             double avg4=(total_anchor4.get(key)/replies.size());
             list.add(new SlaveReply(key,avg1,avg2,avg3,avg4));
         }
         return new LocReply(1l,replies.get(0).date,list);
     }
}
```

“Node” Source Code

```
package com.amir.smartclassroom.utils;

import java.text.DecimalFormat;

/**
 * Created by Amir on 2/25/16.
 */
public class Node {

    public double x;
    public double y;
    public int nodeId;
    public Node(int nodeId,double x, double y){
        this.nodeId=nodeId;
        this.x=x;
        this.y=y;
    }
    public String toString(){
        DecimalFormat df = new DecimalFormat();
        df.setMaximumFractionDigits(2);
        return "["+nodeId+"="+df.format( x)+" : "+df.format( y)+"] ";
    }
}
```

“OnScanResult” Source Code

```
package com.amir.smartclassroom.utils;

import java.util.List;

/**
 * Created by Amir on 2/25/16.
 */
public interface OnScanResult {

    void onScanResult(List<IoT_Entry> rawResult);
}
```

“OnStatusResult” Source Code

```
package com.amir.smartclassroom.utils;

import java.util.List;

/**
 * Created by Amir on 2/25/16.
 */
public interface OnStatusResult {

    void onStatusResult(List<IoT_Entry> rawResult);
    void setProgressBar(int progress);
}
```

“OnTaskResult” Source Code

```
package com.amir.smartclassroom.utils;

import android.widget.ProgressBar;

import java.util.List;

/**
 * Created by Amir on 2/24/16.
 */
public interface OnTaskResult {
     void onTaskResult(List<IoT_Entry> rawResult);
     void setProgressMessage(final String s);
     void setProgressBar(int progress);
}
```

“PostTask” Source Code

```java
package com.amir.smartclassroom.utils;

import android.os.AsyncTask;
import android.util.Log;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.codehaus.jackson.JsonFactory;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;
import org.json.JSONArray;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.TimeZone;

/**
 * Created by Amir on 2/25/16.
 */
public class PostTask extends AsyncTask<String,Integer,List<IoT_Entry>> {

    private String username=null;
    private String password=null;

    private OnStatusResult activity;

    public PostTask(OnStatusResult activity, String username,String
password){
        this.activity = activity;
        this.username=username;
        this.password=password;
    }

    public void postRequest(String url){
        HashMap<String,FeedReading> feedMap=new HashMap<String,
FeedReading>();
        DefaultHttpClient httpclient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(url);

        InputStream inputStream = null;
        String result = null;
        try {

            HttpResponse response = httpclient.execute(httppost);
            HttpEntity entity = response.getEntity();
            inputStream = entity.getContent();
            // json is UTF-8 by default
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream, "UTF-8"), 8);
            StringBuilder sb = new StringBuilder();
            //sb.append("feed:");
            String line = null;
            while ((line = reader.readLine()) != null)
            {
                sb.append(line + "\n");
            }
            result = sb.toString();

        } catch (Exception e) {
            // Oops
            Log.d("Displaying","something went wrong "+e.toString());
        }
        finally {
            try{if(inputStream != null)inputStream.close();}catch(Exception
squish){}
        }
        //return result;
    }


    protected List<IoT_Entry> doInBackground(String... urls)  {

        postRequest(urls[0] + "/entryservlet?key="+password+"&reply=4");


        publishProgress(25);
        List<IoT_Entry> list=new ArrayList<>();
        try {
            boolean running=true;
            while(running) {


                list.clear();
                String feed = getFeed(urls[0] + "/feedservlet?key=" +
password);
                JsonFactory f = new JsonFactory();
                JsonParser parser = f.createJsonParser(feed);
                ObjectMapper mapper = new ObjectMapper();
                List<IoT_Entry> entries = mapper.readValue(feed, new
TypeReference<List<IoT_Entry>>() {
                });

                for (IoT_Entry entry : entries) {
                    list.add(entry);
                }
                String reply = list.get(0).content.get("reply").toString();
                System.out.println("here is "+list.get(0).id.toString());
                System.out.println(reply);
                if (reply.equals("1")) {
                    running=false;
                }
                else
                {
                    try {
                        Thread.sleep(500);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
        catch(Exception e){
            e.printStackTrace();
        }
        //publishProgress(100);
        Log.d("Displaying","displaying the "+list);
        return list;
    }

    private String getFeed(String url){

        HashMap<String,FeedReading> feedMap=new HashMap<String,
FeedReading>();
        DefaultHttpClient httpclient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(url);

        InputStream inputStream = null;
        String result = null;
        try {

            HttpResponse response = httpclient.execute(httppost);
            HttpEntity entity = response.getEntity();
            inputStream = entity.getContent();
            // json is UTF-8 by default
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream, "UTF-8"), 8);
            StringBuilder sb = new StringBuilder();
            //sb.append("feed:");
            String line = null;
            while ((line = reader.readLine()) != null)
            {
                sb.append(line + "\n");
            }
            result = sb.toString();

        } catch (Exception e) {
            // Oops
            Log.d("Displaying","something went wrong "+e.toString());
        }
        finally {
            try{if(inputStream != null)inputStream.close();}catch(Exception
squish){}
        }
        return result;
    }
    protected void onProgressUpdate(Integer... progress) {
        activity.setProgressBar(progress[0]);

    }

    //will pass only latest results to onTempResult. First time latestId is 0
    protected void onPostExecute(List<IoT_Entry> result) {
        //this.activity.setProgressMessage("Processing data...");
        activity.onStatusResult(result);
    }

}
```

“ScanReading” Source Code

```java
package com.amir.smartclassroom.utils;

import java.util.Date;

/**
 * Created by Amir on 1/26/16.
 */
public class ScanReading  {

    private String id;
    private float temp;
    private Date date;
    public String username;
    public String nodeId;
    public ScanReading(String id,String username,  Date date, String nodeId)
{
        this.id=id;
        this.temp=temp;
        this.date=date;
        this.username=username;
        this.nodeId=nodeId;
    }

    public String getId(){
        return this.id;
    }

    public Date getDate(){
        return this.date;
    }

    public String toString(){
        return getId()+" "+username+" "+getDate().toString()+"\n";
    }

}
```

“ScanTask” Source Code

```java
package com.amir.smartclassroom.utils;

import android.os.AsyncTask;
import android.util.Log;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.codehaus.jackson.JsonFactory;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 * Created by Amir on 2/25/16.
 */
public class ScanTask extends AsyncTask<String,Integer,List<IoT_Entry>> {

    private String username=null;
    private String password=null;
    private OnScanResult activity;


    public ScanTask(OnScanResult activity, String username,String password){
        this.activity = activity;
        this.username=username;
        this.password=password;
    }

    protected List<IoT_Entry> doInBackground(String... urls)  {

       // publishProgress(25);
        List<IoT_Entry> list=new ArrayList<>();
        try {

            list.clear();
            String feed = getFeed(urls[0] + "/feedservlet?key=" + password);
            JsonFactory f = new JsonFactory();
            JsonParser parser = f.createJsonParser(feed);
            ObjectMapper mapper = new ObjectMapper();
            List<IoT_Entry> entries = mapper.readValue(feed, new
TypeReference<List<IoT_Entry>>() {
            });

            for (IoT_Entry entry : entries) {
                list.add(entry);
            }

        }
        catch(Exception e){
            e.printStackTrace();
        }
        //publishProgress(100);
        Log.d("Displaying","displaying the "+list);
        return list;
    }

    private String getFeed(String url){

        HashMap<String,FeedReading> feedMap=new HashMap<String,
FeedReading>();
        DefaultHttpClient httpclient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(url);
        InputStream inputStream = null;
        String result = null;
        try {

            HttpResponse response = httpclient.execute(httppost);
            HttpEntity entity = response.getEntity();
            inputStream = entity.getContent();
            // json is UTF-8 by default
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream, "UTF-8"), 8);
            StringBuilder sb = new StringBuilder();
            //sb.append("feed:");
            String line = null;
            while ((line = reader.readLine()) != null)
            {
                sb.append(line + "\n");
            }
            result = sb.toString();

        } catch (Exception e) {
            // Oops
            Log.d("Displaying","something went wrong "+e.toString());
        }
        finally {
            try{if(inputStream != null)inputStream.close();}catch(Exception
squish){}
        }
        return result;
    }
    protected void onProgressUpdate(Integer... progress) {
       // activity.setProgressBar(progress[0]);

    }

    //will pass only latest results to onTempResult. First time latestId is 0
    protected void onPostExecute(List<IoT_Entry> result) {
        //this.activity.setProgressMessage("Processing data...");
        activity.onScanResult(result);
    }
}
```

“SlaveReply” Source Code

```
package com.amir.smartclassroom.utils;

/**
 * Created by Amir on 2/25/16.
 */
public class SlaveReply{
    public int nodeId;
    public double anchor1;
    public double anchor2;
    public double anchor3;
    public double anchor4;
    public SlaveReply(int nodeId,double anchor1,double anchor2,double
anchor3, double anchor4){
        this.nodeId=nodeId;
        this.anchor1=anchor1;
        this.anchor2=anchor2;
        this.anchor3=anchor3;
        this.anchor4=anchor4;
    }
    public String toString(){

        return this.nodeId+" "+Math.round(anchor1)+" "+Math.round(anchor2)+"
"+Math.round(anchor3)+" "+Math.round(anchor4);
    }
}
```

“UserReading” Source Code

```
package com.amir.smartclassroom.utils;

import java.util.Date;
import java.util.List;

/**
 * Created by Amir on 1/26/16.
 */
public class UserReading {

    String card_key;

    public String getCard_key() {
        return card_key;
    }

    public String getName() {
        return name;
    }

    public String getGender() {
        return gender;
    }

    public String getEmail() {
        return email;
    }

    public String getPhone() {
        return phone;
    }

    public String getMajor() {
        return major;
    }

    public String getAddress() {
        return address;
    }

    public String getIsfaculty() {
        return isfaculty;
    }

    String name;
    String gender;
    String email;
    String phone;
    String major;
    String address;
    String isfaculty;

    public void setScanReadings(List<ScanReading> scanReadings) {
        this.scanReadings = scanReadings;
    }

    public List<ScanReading> getScanReadings() {
        return scanReadings;
    }

    List<ScanReading> scanReadings;

    public List<Date> getUpReadings() {
        return upReadings;
    }

    public void setUpReadings(List<Date> upReadings) {
        this.upReadings = upReadings;
    }

    List<Date> upReadings;

    public UserReading(String card_key,String name,String gender,String
email,String phone,
                        String major,String address,String
isfaculty,List<ScanReading> list){
        this.card_key=card_key;
        this.name=name;
        this.gender=gender;
        this.email=email;
        this.phone=phone;
        this.major=major;
        this.address=address;
        this.isfaculty=isfaculty;
        this.scanReadings=list;
    }
    public String toString(){
        return name+" : "+email;
    }

}
```

APPENDIX C: Web Service Source Code

"Channel" Source Code

```
import com.googlecode.objectify.annotation.Entity;
import com.googlecode.objectify.annotation.Id;

import com.googlecode.objectify.annotation.Parent;
import com.googlecode.objectify.Key;
/**
 * The @Entity tells Objectify about our entity.  We also register it in
 * OfyHelper.java -- very important.
 *
 * This is never actually created, but gives a hint to Objectify about our
Ancestor key.
 */
@Entity
public class Channel {
    @Id public String channel;
    public String name;
    @Parent Key<ChannelUser> user;


    public Channel(String name, String email){
        this.name=name;
        this.channel = java.util.UUID.randomUUID().toString();
        this.user=Key.create(ChannelUser.class, email);

    }

    public Channel(){
        this.channel = java.util.UUID.randomUUID().toString();
    }

}
```

“CreateKeyServlet” Source Code

```
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

import java.io.IOException;
import java.util.Date;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import
com.google.appengine.repackaged.com.google.io.base.shell.ExecFailedException;
import com.googlecode.objectify.ObjectifyService;

public class CreateKeyServlet extends HttpServlet {

    // Process the http POST of the form
    @Override
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
throws IOException {
        Channel channel;

        UserService userService = UserServiceFactory.getUserService();
        User user = userService.getCurrentUser();  // Find out who the user
is.
```

```
        //String guestbookName = req.getParameter("guestbookName");
        String name = req.getParameter("channel_name");
        if (user != null) {
            channel = new Channel(name, user.getEmail());
            ObjectifyService.ofy().save().entity(channel).now();
            resp.sendRedirect("/main.jsp");

        } else {
            resp.sendRedirect("/error");
        }

    }
}
```

“EntryServlet” Source Code

```
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

import java.io.IOException;
import java.util.Date;
import java.util.List;
import java.util.*;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import
com.google.appengine.repackaged.com.google.io.base.shell.ExecFailedException;
import com.googlecode.objectify.ObjectifyService;

public class EntryServlet extends HttpServlet {

    // Process the http POST of the form
    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse resp)
throws IOException {

        Map<String, String> list=new HashMap<>();
       // String key=request.getParameter("key");
        String str=request.getQueryString();

        if(str.length()>0) {
            String[] str_array = str.split("&");
            for (int i = 1; i < str_array.length; i++) {

                String[] temp = str_array[i].split("=");
                String key = temp[0];
                String value = temp[1];
                list.put(key,value);

            }
            String channelKey = str_array[0].split("=")[1];
            IoTEntry entry = new IoTEntry(channelKey, list);
            ObjectifyService.ofy().save().entity(entry).now();

            ServletOutputStream output = resp.getOutputStream();
            output.println("success "+channelKey+" "+list.toString());

            try {
                if (output != null) output.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

"FeedServlet" Source Code

```
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

import java.io.IOException;
import java.util.Date;
import java.util.List;
import java.util.*;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import
com.google.appengine.repackaged.com.google.io.base.shell.ExecFailedException;
import com.googlecode.objectify.Key;
import com.googlecode.objectify.ObjectifyService;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.map.SerializationConfig;


public class FeedServlet extends HttpServlet {

    // Process the http POST of the form
    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse resp)
throws IOException {

        ObjectMapper mapper = new ObjectMapper();

mapper.configure(SerializationConfig.Feature.WRITE_DATES_AS_TIMESTAMPS,
false);

        List<String> list=new ArrayList<>();

        String key=request.getParameter("key");

        if(key!=null) {
            Key<Channel> channelKey = Key.create(Channel.class, key);


            List<IoTEntry> entries = ObjectifyService.ofy()
                    .load()
                    .type(IoTEntry.class)

                    .ancestor(channelKey)
                    .order("-date")
                    .list();


            resp.reset();
            resp.resetBuffer();
            resp.setContentType("application/json");

            ServletOutputStream output = resp.getOutputStream();

            String jsonInString = mapper.writeValueAsString(entries);
            output.println(jsonInString);


            try {
                if (output != null) output.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

    }
}
```

“IoT_Entry” Source Code

```
import com.googlecode.objectify.annotation.*;
import com.googlecode.objectify.Key;

import java.lang.String;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * The @Entity tells Objectify about our entity.  We also register it in
{@link OfyHelper}
 * Our primary key @Id is set automatically by the Google Datastore for us.
 *
 * We add a @Parent to tell the object about its ancestor. We are doing this
to support many
 * guestbooks.  Objectify, unlike the AppEngine library requires that you
specify the fields you
 * want to index using @Index.  Only indexing the fields you need can lead to
substantial gains in
 * performance -- though if not indexing your data from the start will
require indexing it later.
```

```
 *
 * NOTE - all the properties are PUBLIC so that can keep the code simple.
 **/
@Entity
public class IoTEntry {
    @Parent Key<Channel> theChannel;
    @Id public Long id;


    @EmbedMap
    public Map<String, String> content=new HashMap<>();


    @Index public Date date;

    /**
     * Simple constructor just sets the date
     **/
    public IoTEntry() {
        date = new Date();
    }

    /**
     * A connivence constructor
     **/
    public IoTEntry(String channel, Map<String, String> content) {
        this();
        if( channel != null ) {
            theChannel = Key.create(Channel.class, channel);  // Creating the
Ancestor key
        } else {
            theChannel = Key.create(Channel.class, "default");
        }
        this.content = content;
    }

}
```

REFERENCES

[1] Frank Palermo. InformationWeek. [Online]. http://www.informationweek.com/strategic-cio/executive-insights-and-innovation/internet-of-things-done-wrong-stifles-innovation/a/d-id/1279157 (accessed in March 2015).

[2] H. Chen, *Intelligent Classroom Attendance Checking System Based on RFID and GSM*, Advanced Materials Research, Vol. 989-994, pp. 5532- 5535, 2014.

[3] M. Zhi, and M. M. Singh, *RFID-Enabled Smart Attendance Management System*, Future Information Technology, Vol. 329, pp. 213-231, 2015.

[4] A. A. Kumbhar, K. S. Wanjara, D. H. Trivedi, A. U. Khairatkar, and D. Sharma, *Automated Attendance Monitoring System using Android Platform*, International Journal of Current Engineering and Technology, Vol. 4, No. 2, pp. 1096-1099, 2014.

[5] M. Choi, J.-H. Park, and G. Yi, *Attendance Check System and Implemen- tation for Wi-Fi Networks Supporting Unlimited Number of Concurrent Connections*, International Journal of Distributed Sensor Networks, pp. 1 - 10, 2014.

[6] Andreas Willig Holger Karl, *PROTOCOLS AND ARCHITECTURES FOR WIRELESS SENSOR NETWORKS*.: John Wiley & Sons Inc., 2005, p. 235.

[7] Xiuyan Zhu, Yuan Feng. *RSSI-based Algorithm for Indoor Localization*, Communications and Network, 2013, 5, 37-42.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, *Internet of Things (IoT): A vision, architectural elements, and future directions*, Future Generation Computing System, 29, pp. 1645 1660, 2013.

[9] K. Ashton, *That Internet of Things Thing*, RFiD Journal, Vol. 22, pp. 97 114, 2009.

[10] X. Jia, O. Feng, T. Fan, and Q. Lei, *RFID technology and its applications in internet of things (IoT)*, 2nd IEEE conference on Consumer Electronics, Communications and Networks (CECNet'12), pp. 1282 - 1285, 2012.

[11] C. Sun, *Application of RFID technology for logistics on internet of things*, AASRI Conference on Computational Intelligence and Bioinfor- matics, pp. 106 - 111, 2012.

[12] J. He, S. Ji, Y. Pan, and Y. Li, *Constructing Load-Balanced Data Aggre- gation Trees in Probabilistic Wireless Sensor Networks*, IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol. 25, No. 7, pp. 1681 - 1690, July, 2014.

[13] J. He , S. Ji, R. Beyah, Y. Xie, and Y. Li, *Constructing Load-Balanced Virtual Backbones in Probabilistic Wireless Sensor Networks via Multi- Objective Genetic Algorithm*, Transactions on Emerging Telecommunica- tions Technologies (ETT), Vol. 26, No. 2, pp. 147 - 163, February, 2015.

[14] T.G. Zimmerman, *Personal area networks: Near-field intrabody commu- nication*, IBM System Journal, Vol. 35, pp. 609 617, 1996.

[15] P. Baronti, P. Pillai, V.W. Chook, S. Chessa, A. Gotta, and Y.F. Hu, *Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards*, Computer Communication, Vol. 30, pp, 1655 1695, 2007.

[16] Bluetooth, *S.I.G. Specification of the Bluetooth System*, version 1.1. Available online: http://www.bluetooth.com (accessed in March 2015).

[17] G. Anstasi, M. Conti, E. Gregori, and A. Passarella, *802.11 power-saving mode for mobile computing in Wi-Fi hotspots: limitations, enhancements and open issues*, Wireless Networking, Vol. 14, pp. 745 768, 2008.

[18] M.K. Karakayali, G.J. Foschini, and R.A. Valenzuela, *Network coordina- tion for spectrally efficient communications in cellular systems*, Wireless Communication, Vol. 13, pp. 56 61, 2006.

[19] J. He, S. Ji, R. Beyah, and Z. Cai, *Minimum-sized Influential Node Set Selection for Social Networks under the Independent Cascade Model*, ACM MOBIHOC 2014, pp. 93 - 102, 2014.

[20] J. A. Gonzalez-Martnez, M. L. Bote-Lorenzo, E. Gomez-Sanchez, and R. Cano-Parra, *Cloud computing and education: A state-of-the-art survey*, Computers & Education, Vol. 80, pp. 132 - 151, 2015.

[21] Arduino Products, Available online: http://arduino.cc/en/Main/Products (accessed in March 2015).

[22] Arduino Yun Board, Available online: http://arduino.cc/en/Main/ArduinoBoardYun (accessed in March 2015).

[23] T. O'Reilly, *What is Web 2.0: Design patterns and business models for the next generation of software*, International Journal of Digital Economics, Vol. 65, pp. 17 - 37, 2007.

[24] Maxim Shchekotov, "Indoor Localization Method Based on Wi-Fi Trilateration Technique," in *PROCEEDING OF THE 16TH CONFERENCE OF FRUCT ASSOCIATION*.

[25] Nattapong Swangmuang and Prashant Krishnamurthy, "Location Fingerprint Analyses Toward Efficient Indoor Positioning," in *Sixth Annual IEEE International Conference on Pervasive Computing and Communications*.

[26] William J. O'Brien, Christine L. Julien Xiaowei Luo, "Comparative evaluation of Received Signal-Strength Index (RSSI) based indoor localization techniques for construction jobsites," *Advanced Engineering Informatics*, vol. 25, no. 2, pp. 355–363, April 2011.

[27] Zheng, Yang, Zhou Zimu, and Liu Yunhao, *From RSSI To CSI: Indoor Localization Via Channel Response,* ACM Computing Surveys 46.2 (2013): 25:1. Advanced Placement Source. Web. 24 Mar. 2015.

[28] Alhmiedat, Tareq, Ghassan Samara, and Amer O. Abu Salem, *An Indoor Fingerprinting Localization Approach For Zigbee Wireless Sensor Networks*, (2013): arXiv. Web. 3 Apr. 2015.

[29] XBee® 802.15.4, Available online: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module (accessed in March 2015).

[30] Y. Li and K. Zhang, *Research on Application of ZigBee Technology in Flammable and Explosive Environment*, Wireless Sensor Network, Vol. 2 No. 6, 2010, pp. 467-471**.**

[31] Arduino library for communicating with XBees in API mode. Available online: https://code.google.com/p/xbee-arduino/ (accessed in March 2015).

[32] ThingSpeak, Available online: https://thingspeak.com/ (accessed in March 2015).

[33] Arduino Bridge Library, Available online: http://arduino.cc/en/Reference/YunBridgeLibrary (accessed in March 2015).

[34] Chuan-Hsian Pu and Hoon-Jae Lee Chuan-Chin Pu, "Indoor Location Tracking Using Received Signal Strength Indicator," in *Emerging Communications for Wireless Sensor Networks*., 2011

[35] Arduino RFID Library for MFRC522, Available online: https://github.com/miguelbalboa/rfid (accessed in March 2015).

[36] Arduino Serial Peripheral Interface (SPI), Available online: http://arduino.cc/en/Reference/SPI (accessed in March 2015).

[37] Erin-Ee-Lin Lau; Wan-Young Chung, "Enhanced RSSI-Based Real-Time User Location Tracking System for Indoor and Outdoor Environments," in Convergence Information Technology, 2007. International Conference on , vol., no., pp.1213-1218, 21-23 Nov. 2007

[38] Palma, D.; Agudo, J.E.; Sánchez, H.; Macías, M.M. An Internet of Things Example: Classrooms Access Control over Near Field Communication. *Sensors* **2014**, *14*, 6998-7012.

[39] Ryu, M.; Kim, J.; Yun, J. Integrated Semantics Service Platform for the Internet of Things: A Case Study of a Smart Office. *Sensors* **2015**, *15*, 2137-2160.

[40] Choi, H.-S.; Rhee, W.-S. IoT-Based User-Driven Service Modeling Environment for a Smart Space Management System. *Sensors* **2014**, *14*, 22039-22064.

[41] Jing (Selena) He, Amir Atabekov, Yi Pan, "Implementing Internet of Things into Undergraduate Classes: Case Study a Capstone Research Project", Computer Education, 4(4):30-38 , April, 2016.

[42] Amir Atabekov, Jing (Selena) He, Patrick Otoo Bobbie, "Internet-of-Things-based Framework to Facilitate Indoor Localization Education", IEEE COMPSAC 2016, Atlanta, GA, June 10 -14, 2016.

[43] Jing (Selena) He, Amir Atabekov, Edward Mwangi, and Donald Toler, "Smart Chair: An Internet of Things Case Study for a Capstone Research Project", The 2015 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'15), July 27-30, Las Vegas, NV, 2015.

[44] Amir Atabekov, Marcel Starosielsky, Dan Chia-Tien Lo, and Jing (Selena) He, "Internet of Things-based Temperature Tracking System", IEEE COMPSAC 2015, Taichung, Taiwan, July 1 - 5, 2015.