MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

Parking Guidance System Solution

Group 1	
Group members	Trần Nguyễn Minh Trung – Team Leader – SE61496 Bùi Phú Hiệp – Team Member – SE61438 Nguyễn Đỗ Phương Huy – Team Member – SE61358
Supervisor	Nguyễn Đức Lợi
Ext. Supervisor	N/A
Capstone Project Code	PGSS

- Ho Chi Minh City, Jan, 2017

TRƯỜNG ĐẠI HỌC FPT

CAPSTONE PROJECT REGISTER

Class: Duration time: from 02/01/2017 To 26/04/2017

(*) Profession: <Software Engineer> Specialty: <ES> [x] <IS> [ ]

(*) Kinds of person make registers: Lecturer [x] Students [ ]

1. Register information for supervisor (if have)

	Full name	Phone	E-Mail	Title
Supervisor 1	Nguyễn Đức Lợi		loind@fpt.edu.vn	Mr.

2. Register information for students (if have)

	Full name	Student code	Phone	E-mail	Role in Group
Student 1	Trần Nguyễn Minh Trung	SE61496	0903954515	trungtnmse61496@fpt.edu.vn	Leader
Student 2	Bùi Phú Hiệp	SE61438	01275904362	hiepbpse61438@fpt.edu.vn	Member
Student 3	Nguyễn Đỗ Phương Huy	SE61358	0987081094	huyndpse61358@fpt.edu.vn	Member

3. Register content of Capstone Project

(*) 3.1. Capstone Project name:

English: Parking Guidance System Solution

Vietnamese: Giải pháp hệ thống chỉ dẫn đỗ xe

Abbreviation:

- PGSS
- PGSS provides indoor and outdoor parking solutions for parking garages and parking lots in shopping malls, at streets, at tourist attractions. Combined with mobile internet, the PGSS provides services of personalized parking reservation, parking guide, and online payment.

(*) 3.2. Main proposal content (including result and product):

a) Theory and practice (document):

Hardware:

- Research and implementation of the Main-broad controller.

  (Arduino, Raspberry Pi).
- Research about how to control stepper motor.
- Research about how to control RGB LED with PWM, Seven-Segment Display (SSD) and LCD Display.
- Research and implementation of the Ultrasonic Sensor.
- Research RF signal (nRF24L01).
- Design and implementation of the electrical power.

Software:

- Learn and implement Arduino language, C, C++ and Arduino Library.
- Research and implement Mesh Network with nRF24L01.
- Learn and develop an Android mobile application with Java.
- Learn and create a REST service using ASP.NET Web API and SQL Database in Azure App Service.

- Result:

Hardware:

1. Central Controller
2. Power Supply
3. LED Display Board
4. Vehicle Guidance Indicator
5. Single Space Detector

6. Single Space LED Indicator
7. Single Space Reservation Lock

Software:

1. Android application
2. Web server

b) Program:

- Java programming skill for Android.
- Python 3 programming skill for Raspberry Pi 3.
- C/C++/Arduino language for Arduino.
- ASP.NET and C# for Azure.
- Tool development: Android Studio, Visual Studio, Visual Micro, Python IDLE, Vim
- Graphic design: Inkscape, AutoCad, CoreRaw, Photoshop, GUI design studio, Proteus 8,…

c) Other products:

4. Other comment (propose all relative thing if have)

References

[1] Information on nRF24L01 datasheet :

https://www.nordicsemi.com/chi/content/download/2730/34105/file/nRF24L01_Product_Specification_v2_0.pdf

[2] Information on http://arduino.cc/en/Main/arduinoBoardUno

[3] Information on http://arduino.vn/

[4] Information on https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Tp.HCM, date 12/12/2016

Supervisor (If have)	**On behalf of Registers**
(Sign and full name)	*(Sign and full name)*
	Loi Nguyen Duc

This page is intentionally left blank

Table of Contents

List of Tables

List of Figures

Definitions, Acronyms and Abbreviations

Name	Definition
PGS	Parking Guidance System
Parking area	An area set aside for parking vehicles, aircraft, etc.
Parking lot	A place inside parking area that provide space for one vehicle
IoT	Internet of Things
CCU	Central Control Unit
ASIC	Application-specific integrated circuit
LCC	leadless chip carrier

Table 1: Definitions, Acronyms and Abbreviations

A. Introduction

1. Project Information

- Project name: Parking Guidance System Solution
- Project Code: PGSS
- Product Type: Internet of Things Application
- Start Date: 3-Jan-2017
- End Date: 24-Apr-2017

2. Introduction

Information and guidance system is designed the monitoring and provision of information on the occupancy of individual parking lots in the parking area. The system represents a solution to the current problem of a high proportion of a traffic generated by drivers seeking vacant parking spaces. The guidance system is able to provide drivers with the latest and dynamically changing information on the availability status of monitored parking lots. Using clear guidance signs, vehicles are guided directly to identified vacant parking lots that are the closest to vehicles' current positions.

With the help of the parking guidance system, drivers are able to find a vacant parking lot quickly and easily. The resulting benefits are the reduction of stop-start traffic, pleasant experience of parking, elimination of stressful situations and positive attitude towards the car park operator. The reduction of traffic minimizes the occurrence of traffic accidents. The positive mental state of drivers is important for all commercial subjects that need to stimulate required shopping behavior, repeated visits and the increase of customers' loyalty. In highly competitive environment, the parking guidance system may become a competitive advantage and generate additional profits for operators.

3. Current Situation

The current situation can be summarized through the following 3 categories:

3.1. Indoor parking area

Figure 1: Indoor parking area

Indoor parking areas are growing with the increasing number of vehicles in a developing economy and causing many problems due to multiple floors, followed by zones, distributed parking lots and absence of parking guidance to vehicles drivers. The traditional method of having to navigate around searching for an empty parking lot causes many troubles for drivers, as well as traffic jam in parking areas.

3.2. Outdoor parking area

Figure 2: Outdoor parking area

With the increasing number of vehicles, it creates lots of issues to build a parking building, or a basement plus some other kind of building on top, especially cost and planning structure. There is no other way except to utilize outdoor spaces of places like public parks, mall, hospital… as an outdoor parking areas.

3.3. Traditional PGS

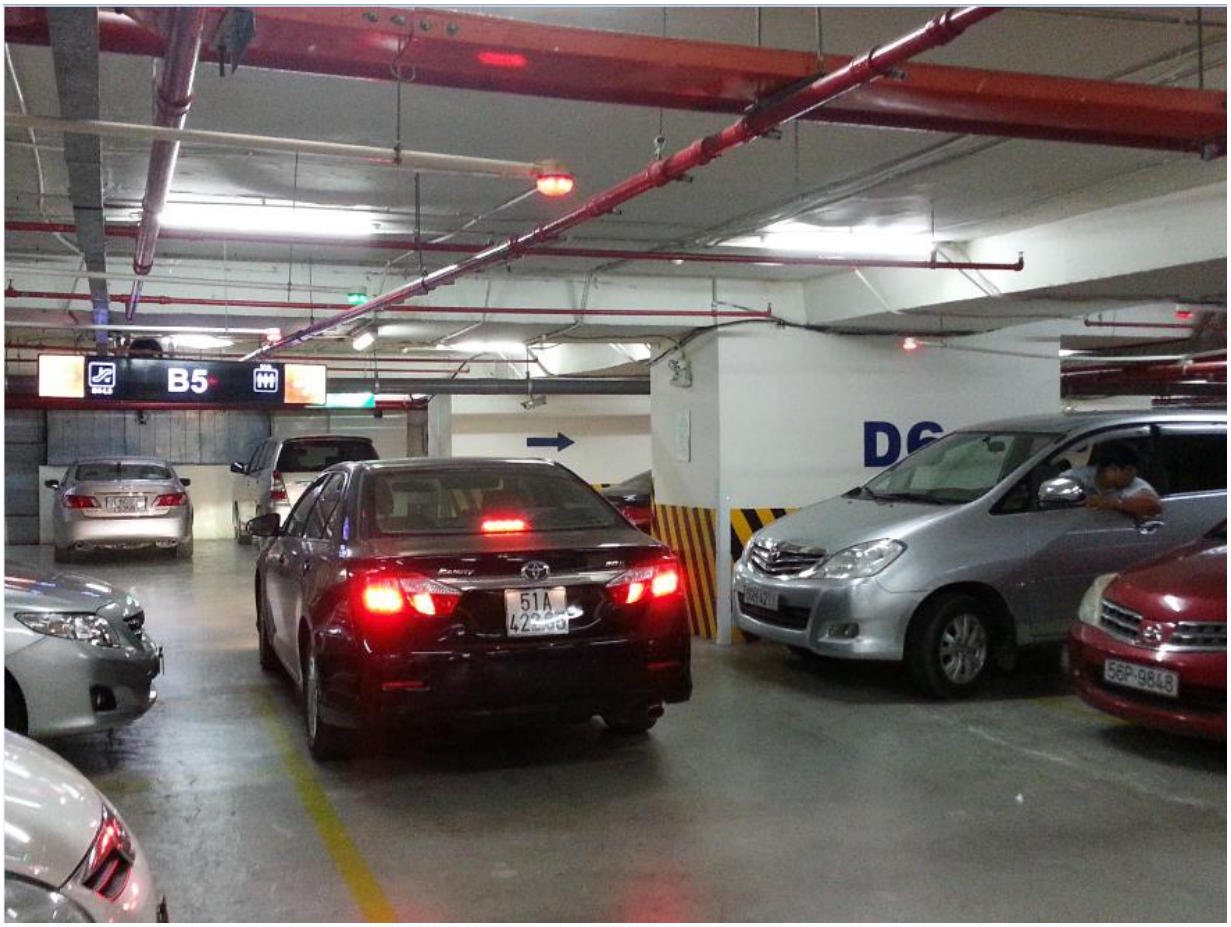


Figure 3: Parking area with PGS

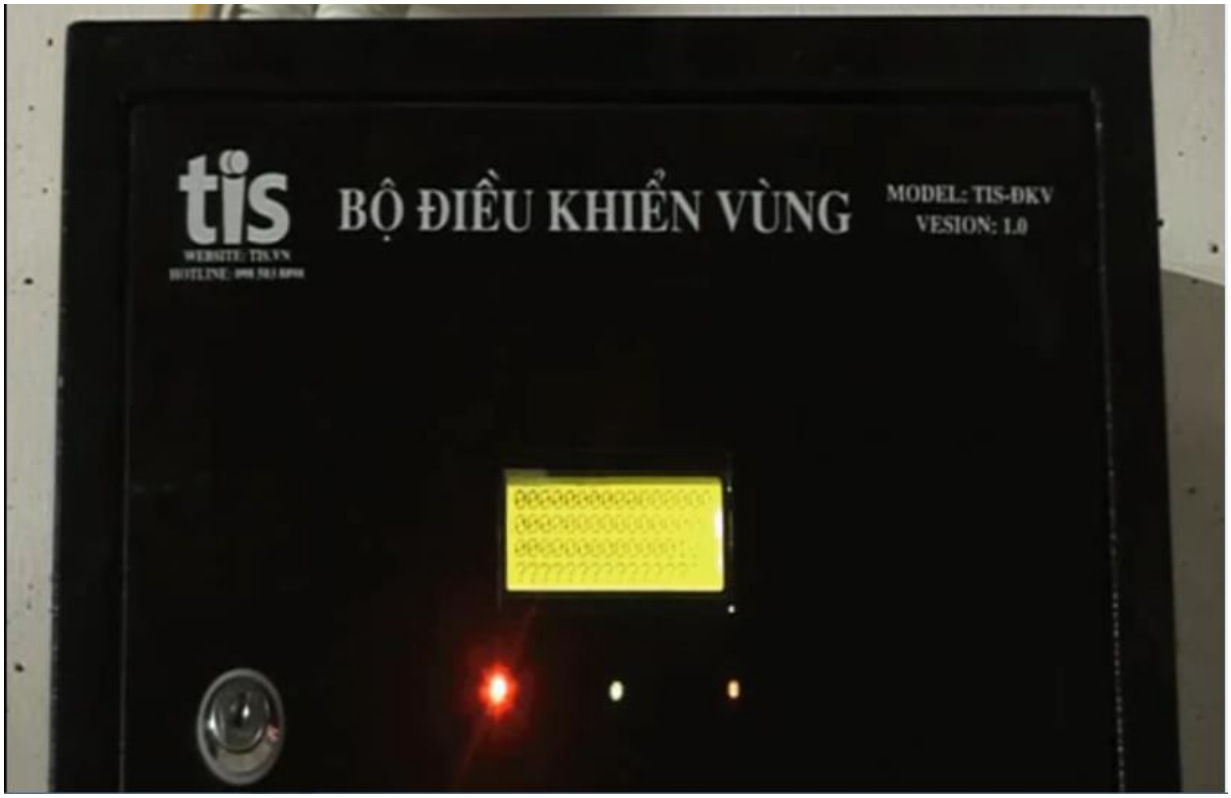


Figure 4: Zone Control Unit

As opposed to the traditional parking areas, the parking areas with PGS keep parking lots under systematic real-time monitoring so as drivers can see what parking spaces are available immediately and with minimal effort. By implementing this, operators also have the chance of increasing their revenues because of the increasing number of satisfied drivers.

The current version of PGS that implemented in a large number parking areas in Vietnam made use of RS485. Each parking lot is fitted with an ultrasonic detector and Indicator light, hence information displays at main entrance and at internal junction points are driven with real-time occupancy detected by ultrasonic detectors. All status sent to a Zone Control Unit on RS485, which in turn be sent to the Central Control Unit.

4. Problem Definition

The current version of PGS is working well but it still has some disadvantages:

- The system implements the RS485 so each Zone Control Unit can have maximum 8 loops of 32 Detectors hence supporting 256 parking lots with 8 Information LED Displays. This is fine for most of the current parking areas, but it provides complicated in a parking area with large number of parking lots like the 6 multi-story car parks with around 7000 parking lots each proposed by Ho Chi Minh City Transport Department.
- The Zone Control Units need to be wired with Detectors, Indicator LED, Information LED Displays and Central Control Unit hence the wiring is pretty

complicated and need careful planning in the construction stage. Therefore, the current version of PGS is hard to implement in most of the existed parking areas.
- The current version of PGS is difficult to use for outdoor parking areas because of the need of installation of the frame.
- Drivers can only get the information of available parking lots at the entrance of parking areas, so the issues of high proportion of traffic generated by drivers seeking vacant parking lots still remains.

5. Proposed Solution

The current version of PGS contains many flaws and proved to be unacceptable for a greater business. Therefore, our proposed solution is to build a parking guidance system with RF modules. The RF modules provide wireless communication directly between Central Control Unit and Detectors, Indicator LED, Information LED Displays so there is no need for the Zone Control Units.

5.1. Feature functions

5.1.1. Parking Guidance System

- Detectors sends out ultrasonic signals from the bottom upward and transmits the signals to the guidance units through RF.
- The Indicator LED, Information LED Displays also use RF communication so they are easier to install.
- The Central Control Unit connect with data stream network to provide real-time information to the app.

5.1.2. Mobile app

- Management portal for operators to setup and manage theirs parking area.
- Customer portal where drivers can view on maps the real time information of nearby parking areas or search for one.

5.2. Advantages

- Fast orientation of drivers when seeking vacant parking lots
- Minimizing the time needed for finding a vacant parking lot
- Improvement of safety, the increase of the traffic effectiveness and efficiency
- Decreases of exhaust fumes as well as the negative impact of traffic on the environment
- Maximum use of the entire car park capacity
- Easy to assemble

5.3. Disadvantages

- System does not provide car find feature
- The detector can only detect at the location above it so it can’t detect if there is anything around the corner of parking lot
- Management portal does not have web version

6. Functional Requirements

Function requirements of the system are listed as below:

- Detector component:
  - Sensor (ultrasonic, infrared, magnetic field, load cell…)
  - RF communication
- Indicator LED component:
  - RGB led controller
  - RF communication
- Information LED Displays component:
  - Main entrance LED Display
  - Internal junction points LED Display
  - RF communication
- Reservation Barrier component:
  - Servo motor controller
  - RF communication
- Central Control System component:
  - Data Stream Network
  - Web API communication
  - RF communication
- Management portal app component:
  - Parking Area Setup
  - Parking Area Analysis
  - Parking Lot Control
- Customer portal app component:
  - Parking Areas Search
  - Parking Lot Reservation
- Web API component:
  - Parking Areas Search
  - Parking Area Setup
  - Parking Area Analysis
  - Parking Lot Status

7. Roles and Responsibilities

No	Full name	Role	Position	Contact
1	Nguyễn Đức Lợi	Project manager	Supervisor	loind@fpt.edu.vn
2	Trần Nguyễn Minh Trung	Developer	Leader	trungtnmse61496@fpt.edu.vn
3	Bùi Phú Hiệp	Developer	Member	hiepbpse61438@fpt.edu.vn
4	Nguyễn Đỗ Phương Huy	Developer	Member	huyndpse61358@fpt.edu.vn

Table 2: Roles and Responsibilities

8. Conclusion

For this project, we will try to to reproduce the traditional Parking Guidance System with wireless technology, add a web server to manage information, a mobile app to provide UI for normal users and parking area operators to make this more like an IoT application. Therefore, we will need to:

- Research to determine and implement the appropriate MCU for the Central Control Unit and other nodes (**Arduino, Raspberry**, CC1310…)
- Research to determine and implement the appropriate sensor for the Detector (ultrasonic sensor, infrared sensor, **magnetic sensor**, load sensor…)
- Research to determine and implement the appropriate RF value and module to provide communication between nodes for the project (315Mhz, 433Mhz, **2.4Ghz**…)
- Research and implement LED RGB, seven-segment LED
- Research and implement real-time communication channel
- Research and host Web API on a cloud service
- Study and develop a mobile application (**Android**, iOS, Windows phone…)
- Study and develop program using embedded language (**Arduino**, C, **C++**, **Python**, Java Embedded, C#…)
- Study and create a Web API (Spring MVC, **ASP.NET**, Ruby…)
- Study and create a database (**SQL**, Oracle, MySQL, **SQLite** …)

B. Software – Hardware Project Management Plan

1. Problem Definition

1.1. Name of this Capstone Project

- Official name: Parking Guidance System Solution
- Vietnamese name: Giải pháp hệ thống chỉ dẫn đỗ xe
- Abbreviation: PGSS

1.2. Problem Abstract

As the economy of Vietnam growth, the number of personal cars also increasing, and that create a high proportion of traffic generated by drivers seeking vacant sparking lots. The current common Parking Guidance Systems in Vietnam are only suitable for a small number of indoor parking areas, and can't be implemented for outdoor parking areas, because of the need of complicated planning and wiring. Moreover, all of the current PGS parking areas are working separately in their own local area network so there is no way for drivers to know the current available parking lots except by coming to the entrance.

We provide a system which ease the complicated in set up a PGS for parking areas. Furthermore, we make the system in a way that each PGS parking area can connect to each other so we can provide more information to drivers to help them find a suitable parking

areas more quickly and easily.

1.3. Project Overview

1.3.1. Current Situation

In the market, we have some ways to manage car park:

- The guard check each car in each parking lot:
  - Advantages:
    - Can check empty or using slot exactly
  - Disadvantages:
    - Consume people's energy
    - Need much time to check all the parking lot
- Tradition PGS base on RS-485:
  - Advantages:
    - Can automatically check the empty slot in car park
  - Disadvantages:
    - Limited number of managed parking lots
    - Complicated wired system
    - Limited information provided to drivers

1.3.2. The Proposed System

Based on the result of our research, we propose the following solution: A Parking Guidance System based on Internet of Things that utilize the RF wireless technology to communicate between components. The system consists of Detectors, Indicator LED, Information LED Displays, Reservation Barrier, Central Control Unit, Web API Server and a Mobile Application.

After the Detector detect a car in the parking lot, it will send a signal to the Central Control Unit (CCU). The CCU will command the Indicator LED above that parking lot change to occupied color, update all related Information LED Displays, send a message to Web API Server to update information of parking lot on the server and Mobile Application.

In case of users want to reserve a parking lot in the parking area, they can use the Mobile Application to make a reservation. The Web API Server will update the database based on the reservation and the Reservation Barrier will lock the reserved parking lot.

1.3.2.1. Interaction Block

- This block will be place in each parking lot to check existed car and control signal light, barrier.
- Arduino is the main board to control the Interaction block, which show information to the end customer.
- Ultrasonic sensor is used to check the existed car in each parking lot
- DC Servo to monitor the barrier.
- 12A DC-to-DC step down module used to convert voltage high-to-low for other hardware.

1.3.2.2. Information Block

- This is the led panel in each area to show the number of empty slot in each area.
- Arduino is also the main board to control this block.

1.3.2.3. Central Control Unit

- This will be the central point of all Interaction Module.
- It will be control be Raspberry Pi 3.
- Send and receive data from server to analyze then monitor the Interaction Block.

1.3.2.4. Web API Server

- ASP.NET API to communicate between the mobile app, database and CCU.
- Get the position of each car park base on address to view on mobile app.

1.3.2.5. Mobile Application

- Our priority OS for Mobile Application is Android because of a higher market share than other mobile OS and a lower barrier of entry
- The App utilizes the Google Map API to provide an interactive map that show in real time the available parking lots in each parking areas.

1.3.3. Boundaries of the System

- System is available for both manager and end user.
- The language of system is English
- The input and output of system is car slot
- The boundaries of mechanical parts include:
  - End user need to park their car correctly in parking lot.
  - The information, which show number of empty slot may delay 5-10 seconds when the system start.
  - The system will run correctly when the weather condition is good.
- The boundaries of mobile application include:
  - End user only pay for the booking time, not pay the parking time on mobile app.
  - Need connect to internet to run smoothly.
- The complete product includes:
  - The entire PGSS system in car park.
  - The mobile application for manager and end user.
  - The database to store all needed information.
  - All the documents of the project.

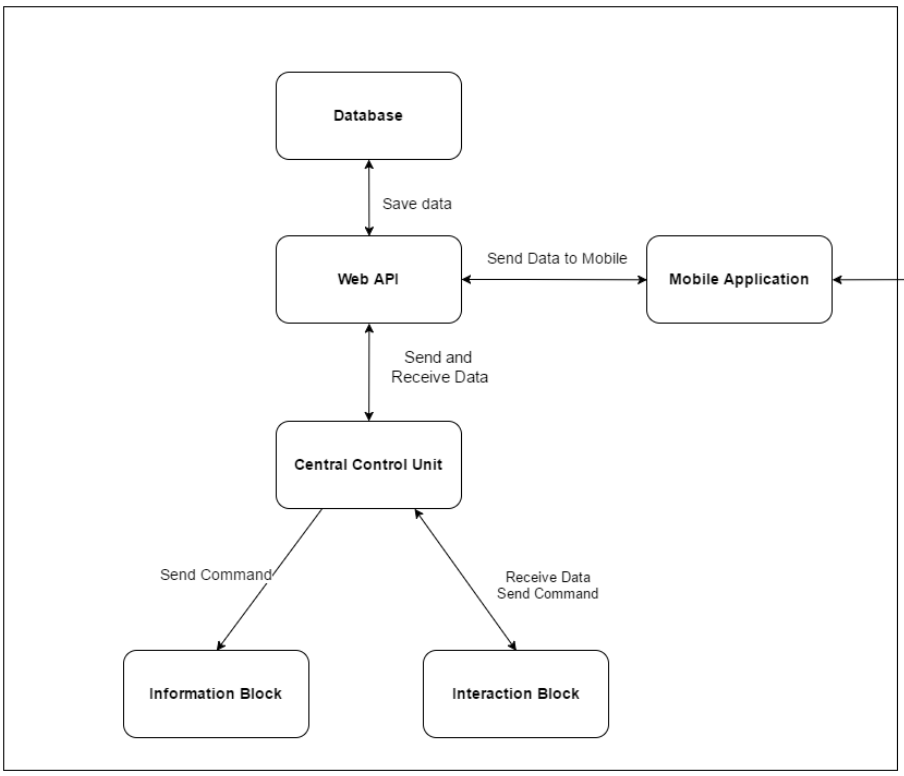


Figure 5: Project Block Diagram

1.3.4. Future Plans

There are no perfect solutions to problems, as well as there are no perfect systems. With the inexperience of our team members and the time constrains, our proposed solution and project contains many issues. Below are the problems encountered in this project:

- **Parking Operate Knowledge:** We are not experts in parking operating. All functions and features are developed in order to serve the needs which we had identified during 4 months of research.
- **Hardware Knowledge:** We are inexperienced with hardware. All the hardware components chosen to be used in this project is based on our familiar with them, or based on the shortest time we need to learn how to use them. So they are only the most appropriate, not the best choice for the project.
- **Single point of failure:** The communication of the PGS system and server is highly depended on the Central Control Unit in each parking area. So if the Central Control Unit crash, the PGS can no longer commute with server.
- **Server crash:** All the needed data for the app is stored in the server. So if server crash, all the devices cannot get parking area information.
- **Security:** Currently, there is few possible problems encountered with RF, as RF is vulnerable to replay attack.

Our future plan is try to solve these problems one by one. We design the system with separated modules in mind to make it easy to change one module without affect others and we also make it easy to scale to bigger models.

1.3.5. Development Environment

1.3.5.1. Hardware requirements

For Web API Server

Components	Requirement
DTU	10 DTU
Storage	250 GB

Table 3: Database requirement

Components	Requirement
Number of cores	1 core
RAM	1.75 GB Ram
Storage	10 GB

Table 4: API Service Requirement

For CCU

Components	Hardware
Mainboard	Raspberry Pi 3
Communication	USB Cable
Sensor Devices	Magnetometer
Motors	Servo
Power Source	

Table 5: Provide CCU Hardware

1.3.5.2. Software requirements

- Windows XP/7/8/10: operating system for developing and deploying.
- SQL Server Express 2012: used to create and manage database for PGSS.
- Visual Studio 2015: used to develop API.
- Arduino IDE: used to develop Arduino program.
- Proteus 8: used to drawing board with other hardware.
- Github & SourceTree: used for source control.
- StarUML: used to create models and diagrams.
- Slack: used for communication and meeting.
- C/C++: used for embedded module
- Python 3: used for Central Control Unit
- C#: used for web server
- Java: used for mobile application

2. Project organization

2.1. Software Process Model

This project is developed under Iterative and incremental development model. We apply customized Iterative and incremental development model to capable with current situation in our team. We choose this model because of the following reasons:

- We are still inexperienced and by develop the system through iterations (repeated cycles) and incrementally (in small portions of time), we can learn from our mistakes and apply that knowledge on the next iteration.
- We are researching and developing the system at the same time, so using this model allow us more flexibility to adapt to changes.
- Working with embedded system hides a lot of problems that are unknown in the planning phase until it is too late. With Iterative and incremental development model, we test the system in small portion at a time, therefore reduce risk and build a feature rich and robust system.

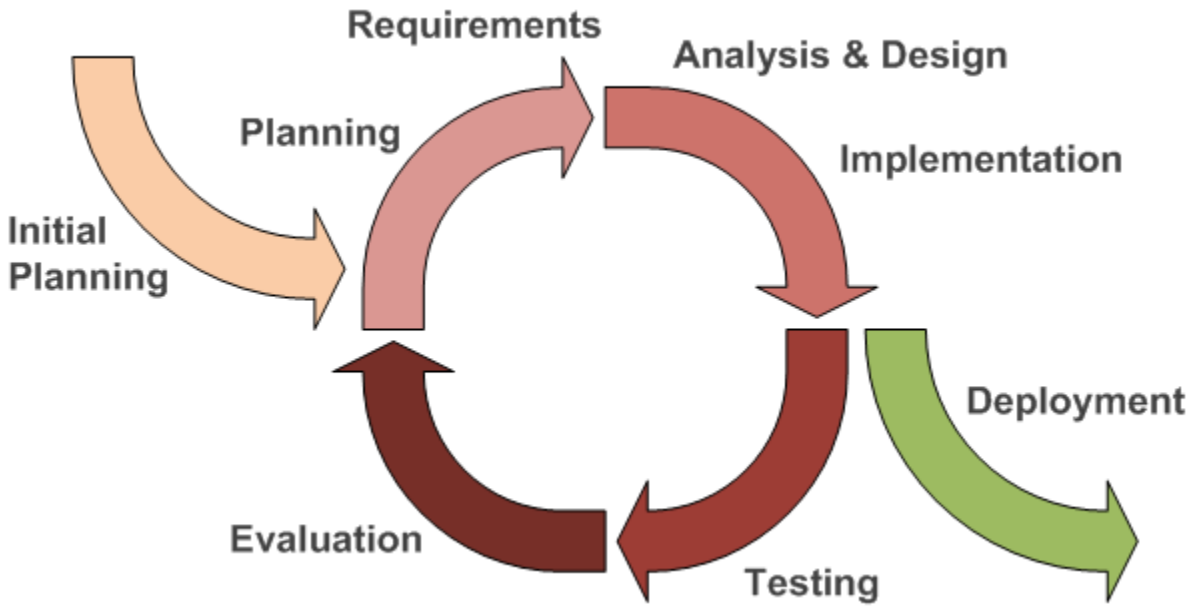


Figure 6: Iterative and Incremental development

2.2. Roles and responsibilities

No	Full name	Team Role	Responsibilities
1	Nguyễn Đức Lợi	Supervisor, Project Manager	• Specify user requirement • Advisor for ideas and solutions • Control the development process

			• Give out techniques and business analysis support
2	Trần Nguyễn Minh Trung	Team Leader, BA, Developer, Tester	• Managing process • Managing budget • Dividing tasks for team member • Create test plan • Clarifying requirements • Prepare document • Coding • Testing
3	Bùi Phú Hiệp	Team Member, Developer, Tester	• Create test plan • Clarifying requirements • Prepare document • Coding • Testing
4	Nguyễn Đỗ Phương Huy	Team Member, Developer, Tester	• Create test plan • Clarifying requirements • Prepare document • Coding • Testing

Table 6: Roles and Responsibilities Details

2.3. Tools and Techniques

Tools	
Operating System	Windows 7 Ultimate
	Raspbian Jessie

Developing tool	Android Studio
	Visual Studio 2015 Community
	IDLE 3
	Arduino IDE 1.6.12
Managing Database	SQL Server 2014 Management Studio
Source Control	Git 2.8.1 (Server https://github.com)
	SourceTree 1.9.10
Communication tool	Gmail
	Slack
	Trello
Models and Diagrams tool	https://www.draw.io/, StarUML

Table 7: Tools

Techniques	
Embedded System	C/C++
	Arduino
	Python 3
Mobile System	Android SDK
	Retrofit 2
	Google Map
Web Server System	Azure Cloud
	ASP.NET
Database Management System	SQL Server 2014
	SQLite 3.7

Table 8: Techniques

3. Project Management Plan

References to main document, Section B - 3. Project Management Plan

4. Coding Convention

References to main document, Section B – 4. Coding Convention

C. Software – Hardware Requirement Specification

1. Software Requirement Specification

1.1. Software Requirement

Manager can show the information of their car park to the end user, which will increase the interaction between car park provider and end user. The information includes:

- Address
- Contact info
- Number of empty parking lot

End user can find the nearest car park, which has empty parking lot.

Manager can manage their car park easily; make an automatic system to guide the end user base on the interaction panel, which show number of empty parking lot in each area and the status light on each parking lot.

Users can see empty slot and detail information about parking area by touching a marker on map.

User can reserve a parking slot.

1.2. GUI Requirement

User interface of mobile app must be simple, clearly and easy to use.

The color of mobile app must be elegant, not garish.

Each UI element must be arranged logically, allowing user access easily.

Meet all main function requirements.

2. Hardware Requirement Specification

2.1. Hardware Requirement

2.1.1. Hardware Interface

References to main document, Section C – 2.1.1 Hardware Interface

2.1.2. Communication Protocol

- We communicate between hardware component and board through GPIO pin.
- Arduino Nano board communicate with Raspberry Pi 3 via RF Module.

2.2. System Overview Use Case

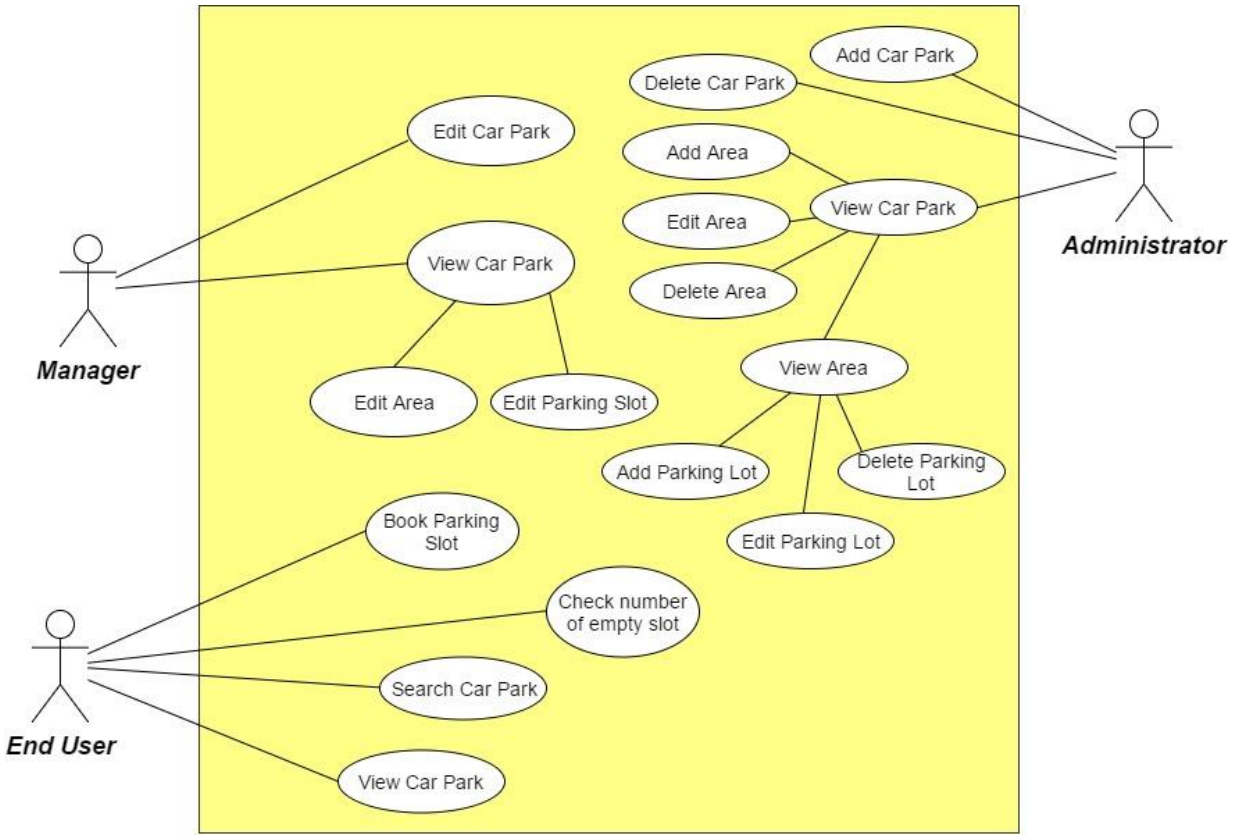


Figure 7: Overview Use Case Diagram

2.3. List of Use Case

References to main document, Section C – 2.3. List of Use Case

3. Software System Attribute

3.1. Usability

- User controls all system components via only mobile application.
- The system can install easily.
- User can learn how to use the system fast.

3.2. Reliability

3.3. Availability

- The mechanical component requires electrical system to work well.
- Hardware components are easy to find in the market.

3.4. Security

- Mobile application requires authentication and authorization implement well because manager and end user use the same application.

3.5. Maintainability

- Use plug and play component so we can easily replace it.

3.6. Portability

- Easy to construct.

3.7. Performance

- Detection car is fast, less then 50ms.
- The speed of server can scale base on the budget easily.

4. Conceptual Diagram

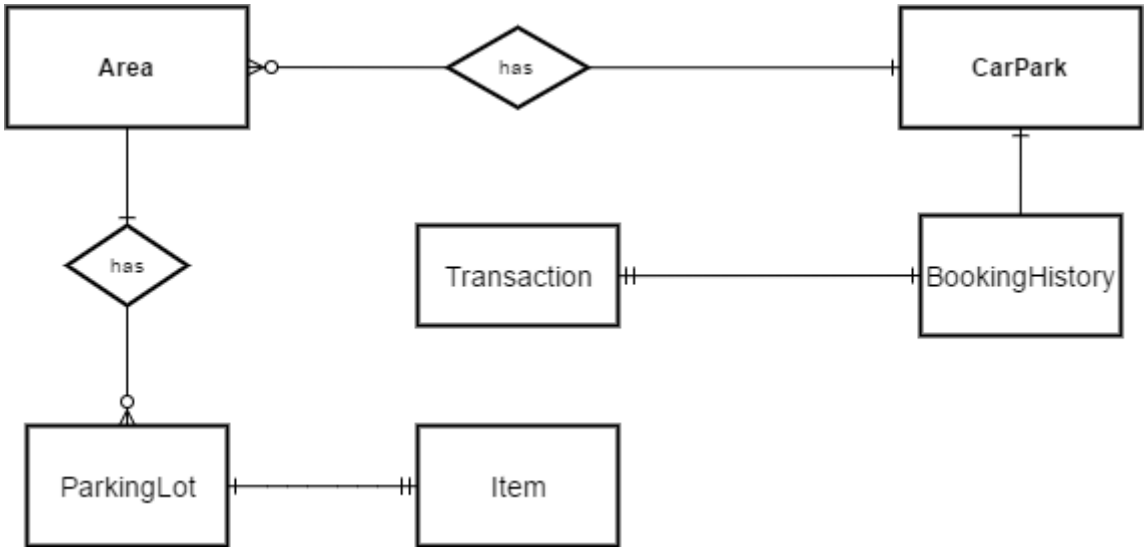


Figure 8: Conceptual Diagram

Data Dictionary

Entity Data dictionary: describe content of all entities	
Entity Name	**Description**
CarPark	Descript all car park information in the system
Area	Describe all area detail in car park
ParkingLot	Describe parking lot information in the area

Item	Describe hardware item in each parking lot
BookingHistory	Describe the booking history of the user
Transaction	Save the transaction of each booking

Table 9: Conceptual diagram data dictionary

D. Software – Hardware Design Description

1. Design Overview

This document describes the technical and user interface design of **PGSS.** It includes the architectural design, the detailed design of common functions and business functions and the design of database model.

The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.

The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use case.

The database design describes the relationships between entities and details of each entity.

Document overview:

- Section 2: gives an overall description of the system architecture design.
- Section 3: gives component diagrams that describe the connection and integration of the system.
- Section 4: gives the detail design description which includes class diagram, class explanation, and sequence diagram to details the application functions.
- Section 5: describe screen design.
- Section 6: describe a fully attributed ERD.
- Section 7: describe algorithms.

2. System Architectural Design

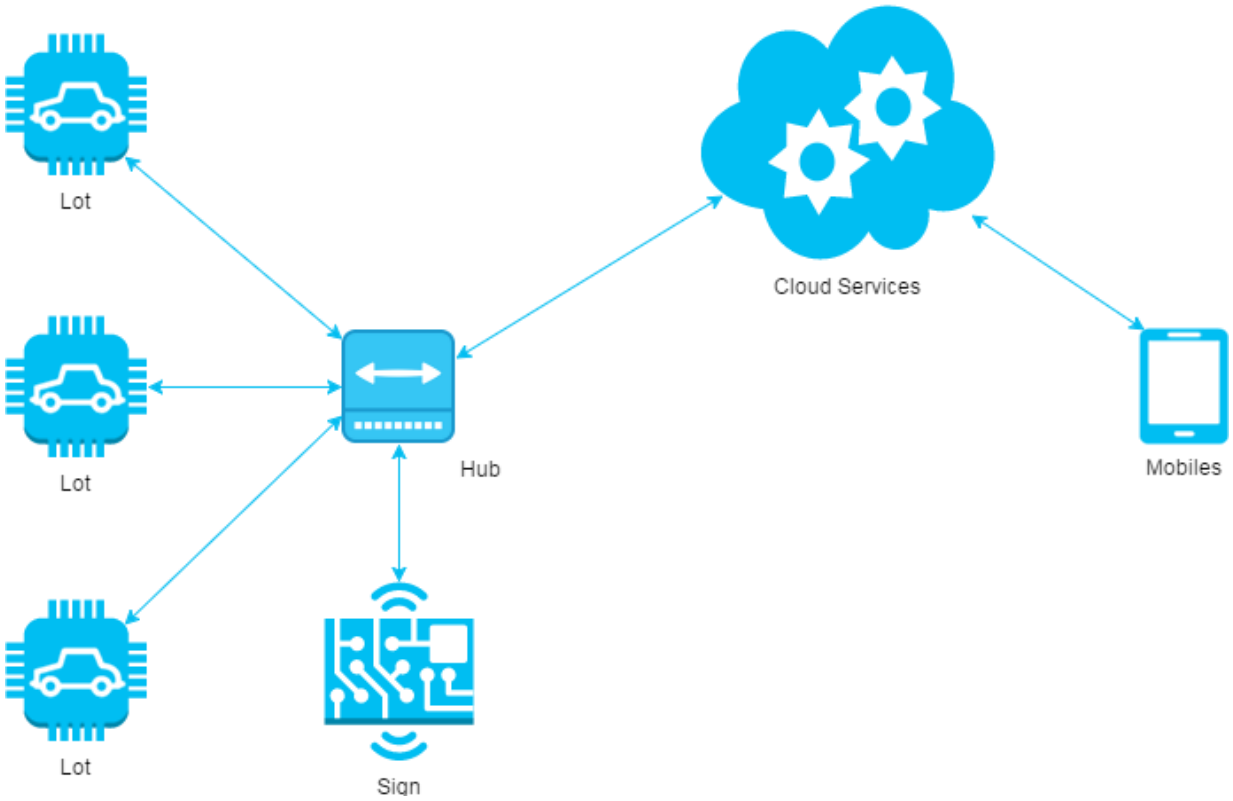


Figure 9: System architecture design

This diagram is referenced and modified from an original concept from: The Internet of Things: An Overview by The Internet Society (ISOC). In this device-to-gateway model, or more typically, the device-to-application-layer gateway (ALG) model, the IoT device connects through an ALG service as a conduit to reach a cloud service. The hub serves as a local gateway between individual IoT devices and a cloud service, but they can also bridge the interoperability gap between devices themselves.

2.1. Hardware Program Architecture Description

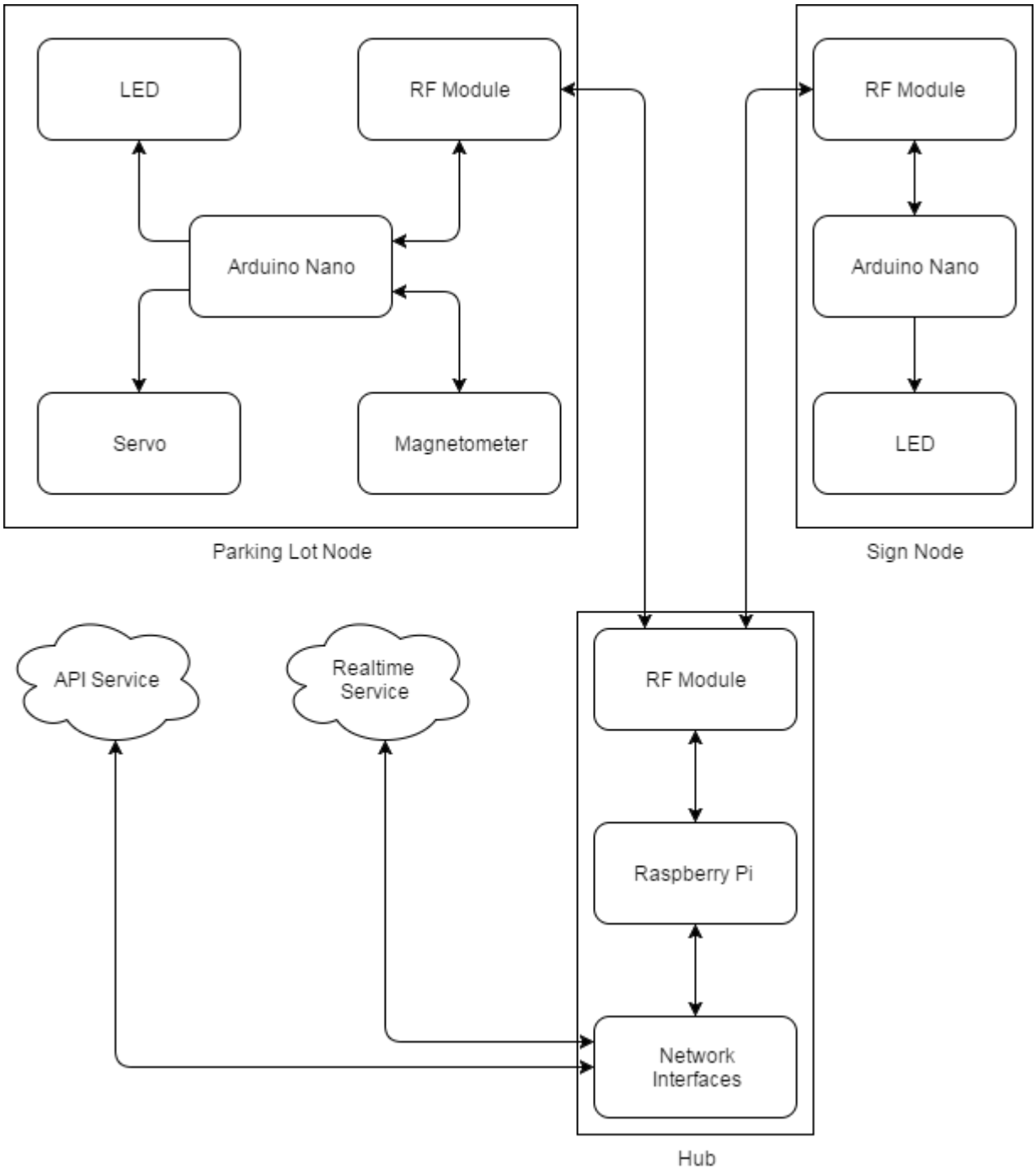


Figure 10: Hardware Program architectural

Parking Lot Node is a device placed at each lot in the car park. It plays the role as a passive IoT node and controls other components, which connects to it through several kinds of port to perform various works:

Arduino Nano: plays the roles as central processing unit for IoT node.

Magnetometer: an instrument that measures magnetism, which is used as metal detectors to detect ferrous metals in a car.

Servo: a linear actuator that allows for precise control of angular or linear position, velocity and acceleration. This will be used to control the barrier in each lot.

LED: a RGB LED used as an Indicator LED in each lot.

RF Module: a small device used to transmit/receive radio signals between two devices. This is used to create a network bridge for the whole system.

Sign Node is a device placed at each area and at the entrance of the car park. It also a passive node and used to display the available lots in each area:

Arduino Nano: plays the roles as central processing unit for IoT node.

RF Module: a small device used to transmit/receive radio signals between two devices. This is used to create a network bridge for the whole system.

LED: a large LED display screen

Hub is a stand-alone gateway device that has RF Module installed to communicate with all other IoT nodes. It also connects to cloud service, allowing the user to gain access to the devices using a smartphone app and an Internet connection:

Raspberry Pi: plays the roles as central processing unit for Hub node.

RF Module: a small device used to transmit/receive radio signals between two devices. This is used to create a network bridge for the whole system.

Network Interfaces: an interfaces to connect to the Internet.

API Service: is a cloud service that provides all necessary APIs and database.

Real time Service: a cloud service that provides real time communication between smartphone app and Hub.

2.2. Mobile Application Architecture Description

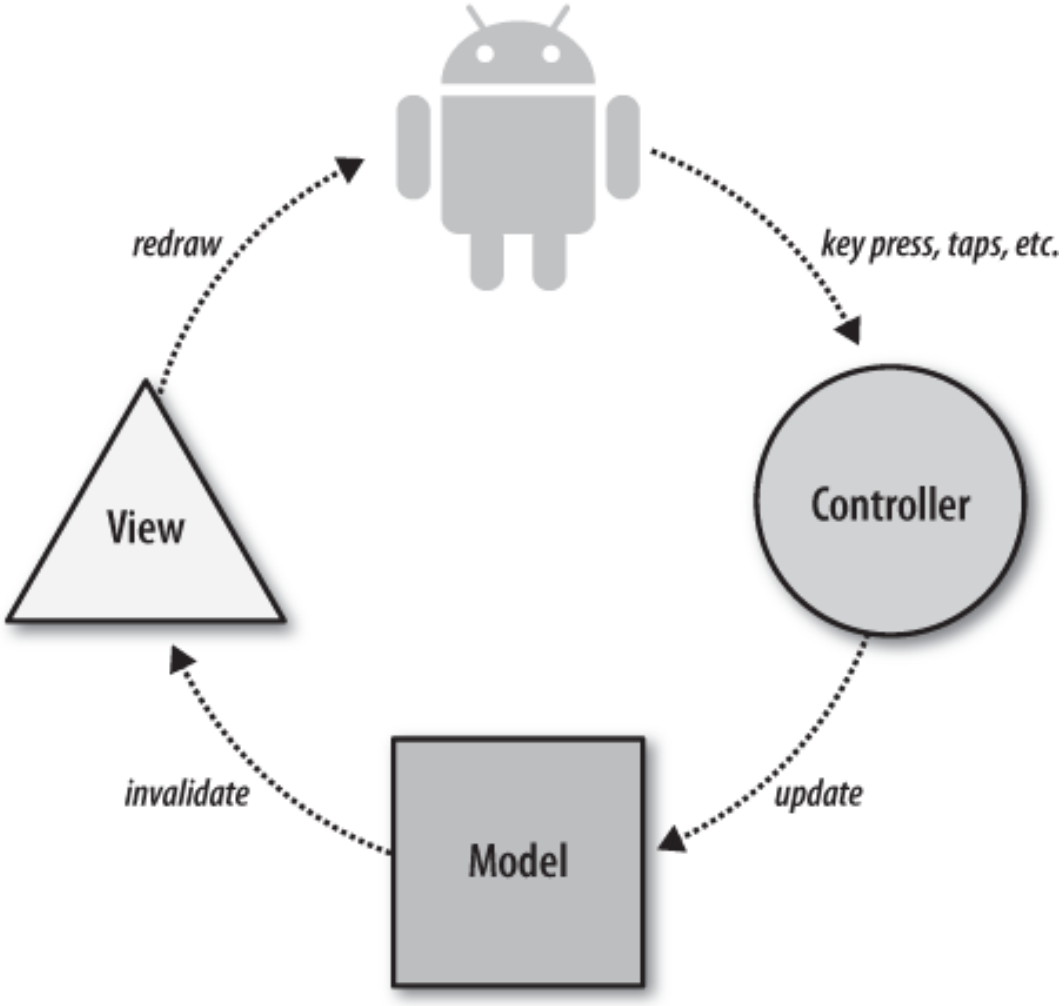


Figure 11: Mobile Application architecture

In Mobile Application, the system is developed under Standard Android Architecture. This is the default approach with layout files, Activities/Fragments acting as the controller and Models used for data and persistence. With this approach, Activities are in charge of processing the data and updating the views. Activities act like a controller in MVC but with some extra responsibilities that should be part of the view.

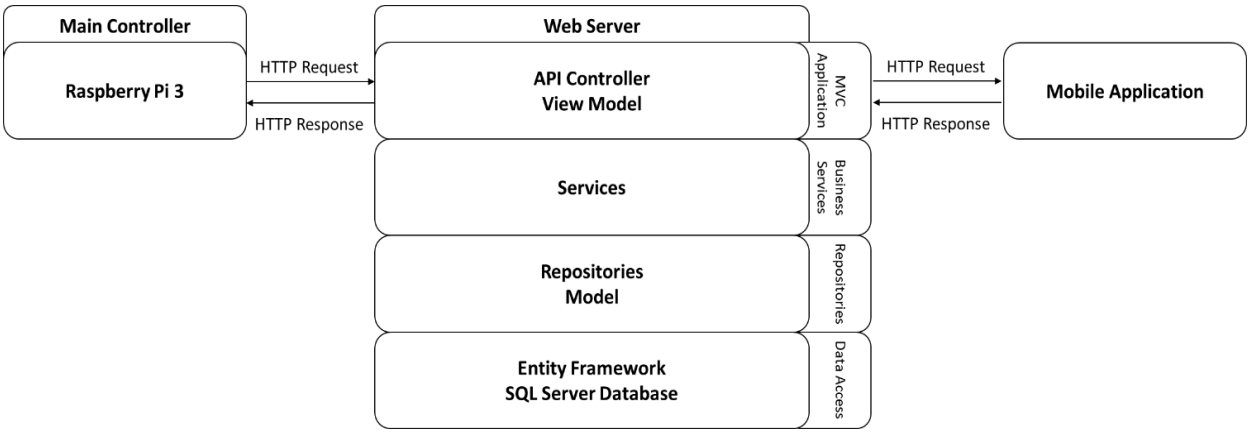
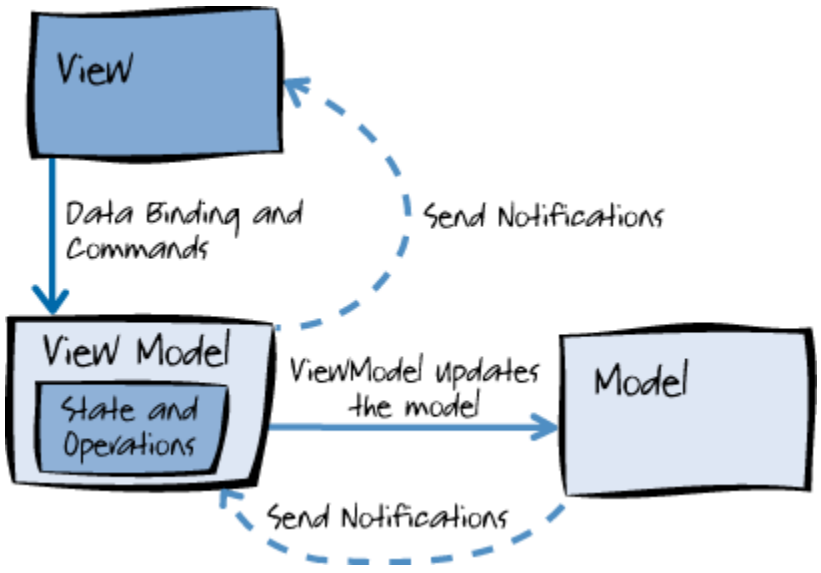


Figure 12: Hardware - Software Connection Architecture Description

2.3. Web API Architecture Description

In Web API, the server is design under MVVM Patern

There are three core components in the MVVM pattern: the model, the view, and the view model. Each serves a distinct and separate role. The following illustration shows the relationships between the three components.



3. Component Diagram

3.1. General Component Diagram

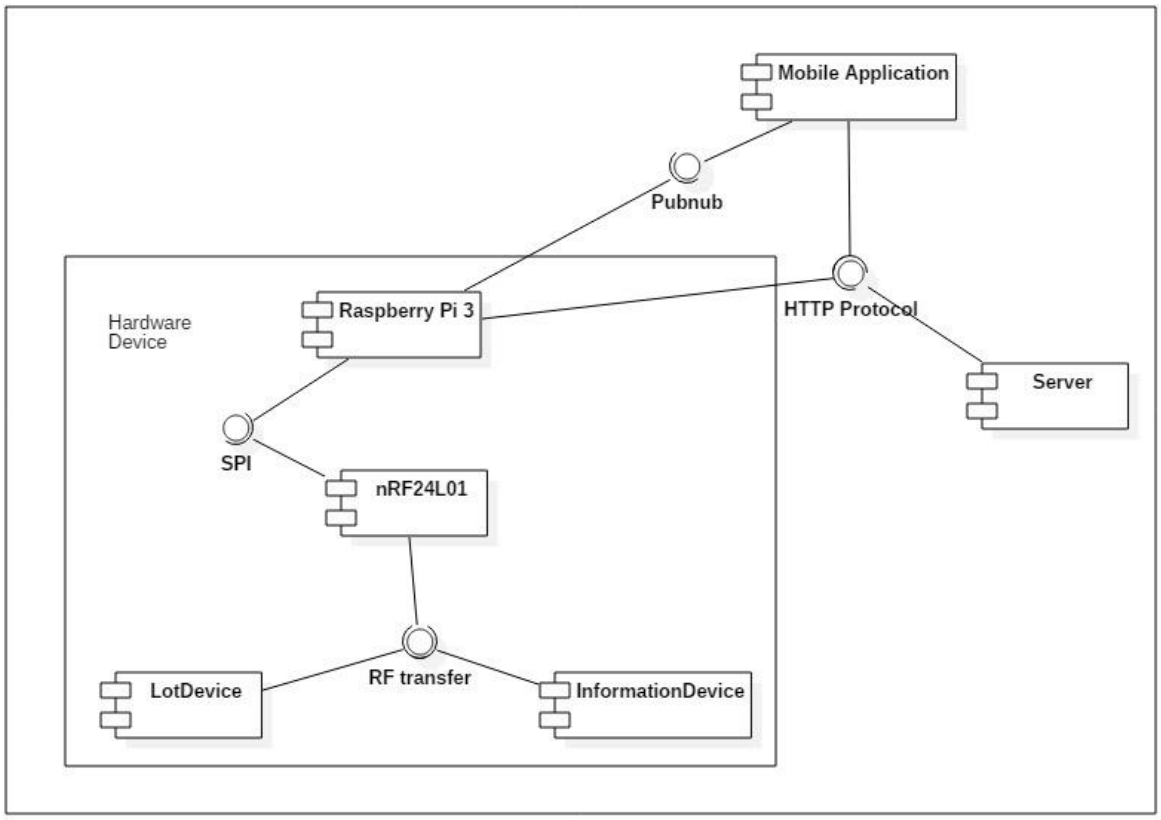


Figure 13: General Component Diagram

3.2. Lot Device Diagram

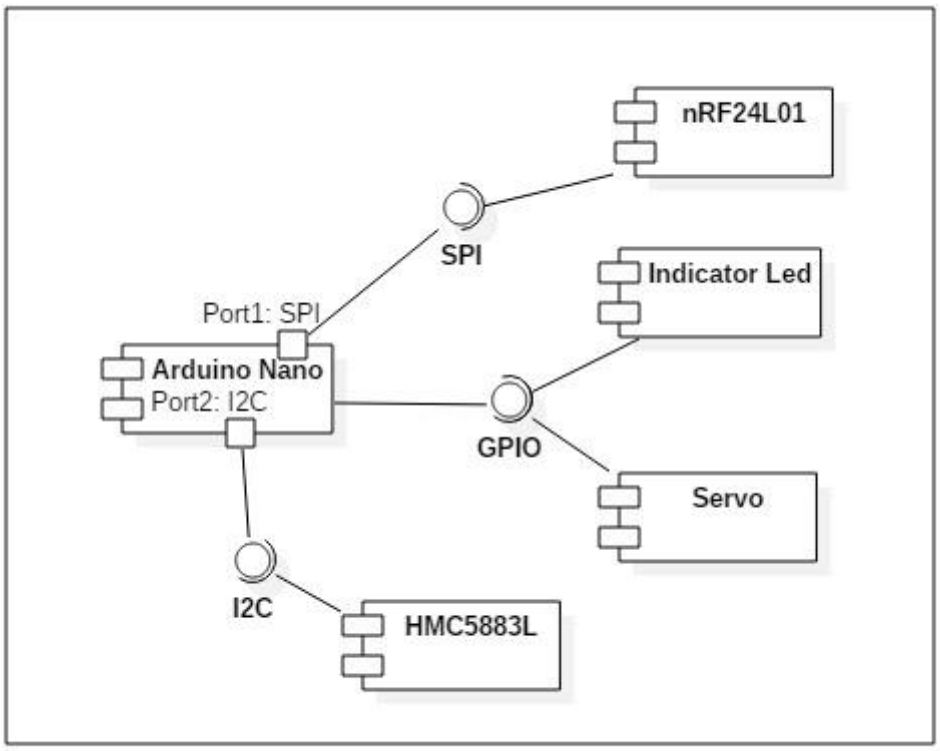


Figure 14: Lot Device Diagram

3.3. Information Device Diagram

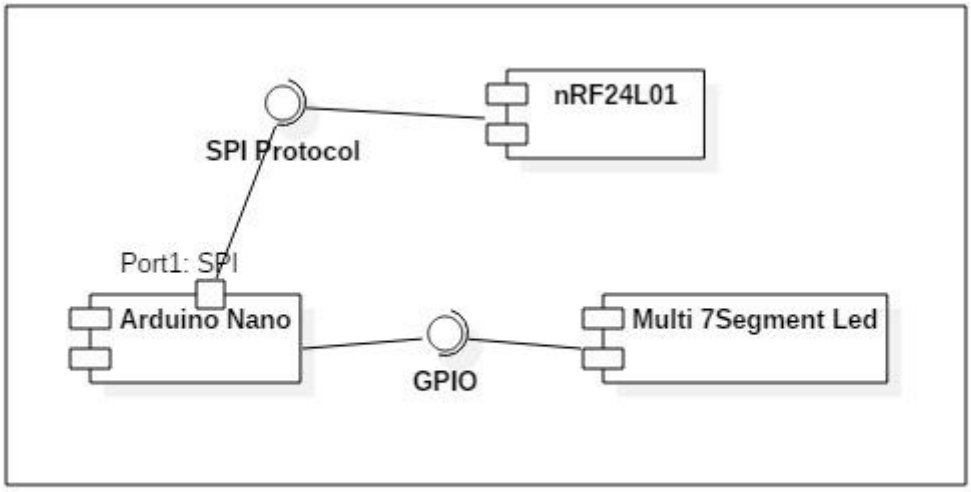


Figure 15: Information Device Diagram

3.4. Server Diagram

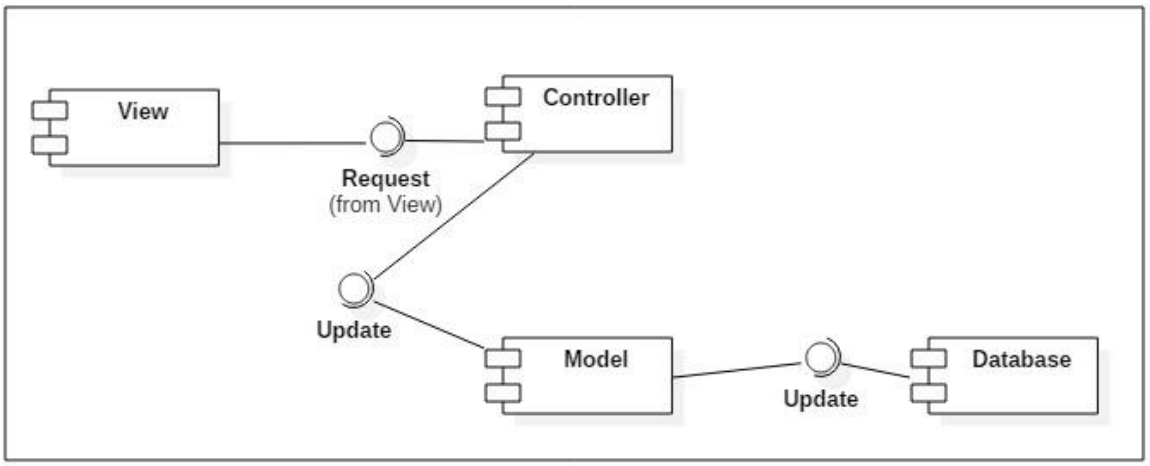


Figure 16: Server Diagram

3.5. Component Dictionary

Component Dictionary: Describes components	
Mobile Application	The application for user on android devices
Server	The Web API server, which make the interaction between the Raspberry Pi and the Mobile Application. It also save information to the database.
Raspberry Pi 3	The main controller in the hardware system that control all Lot Device and Information Device in each car park.
Lot Device	The device in each parking lot that control the indicator led, servo and use the magnetometer sensor to detect car.
Information Device	The number led board that show the number of empty lot in each area and number of empty lot in car park.
nRF24L01	The RF module, which help main controller, lot device and information device interact with each other.
Arduino Nano	The controller in lot device and information device.

Indicator Led	The RGB led to show the status of parking lot (empty, in used, reserved, disabled).
Servo	The barrier to prevent other car get in the parking lot when it is reserved.
HMC5883L	The magnetometer sensor to realize if the metal is near.
Pubnub	The third party API that help real time interaction.
HTTP Protocol	A transfer protocol to interact between server, mobile application and Raspberry Pi 3.
SPI	Synchronous serial communication interface used for short distance communication.
I²C	The inter-integrated circuit to communicate between arduino and magnetometer sensor.
GPIO	The communication between Arduino and led, servo, …
Multi 7-Segment Led	The 7 segment led to indicate number.
View	The JSON result of the API controller.
Controller	The API controller in the server.
Model	The entity model.
Database	The collection of data.

Table 10: Component dictionary

4. Detailed Description

References to main document, Section D – 4. Detailed Description

5. Interface

5.1. Component Interface

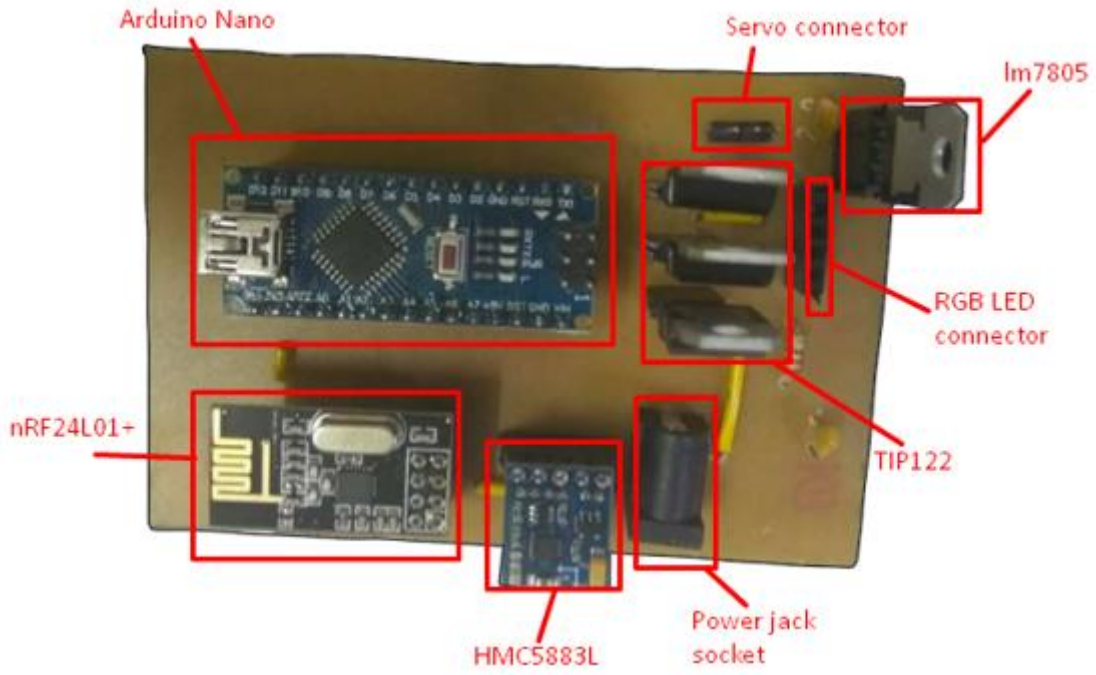


Figure 17: Lot node interface

This is the PCB of lot node. This PCB will be placed in the middle of parking lot so it can detect when a car is parked right above it. Because this PCB will be placed underground so the servo and RGB LED will connect with male and female header row respectively. In demo environment, we placed this directly under our car park model, but in product environment, this will be placed inside a protector.

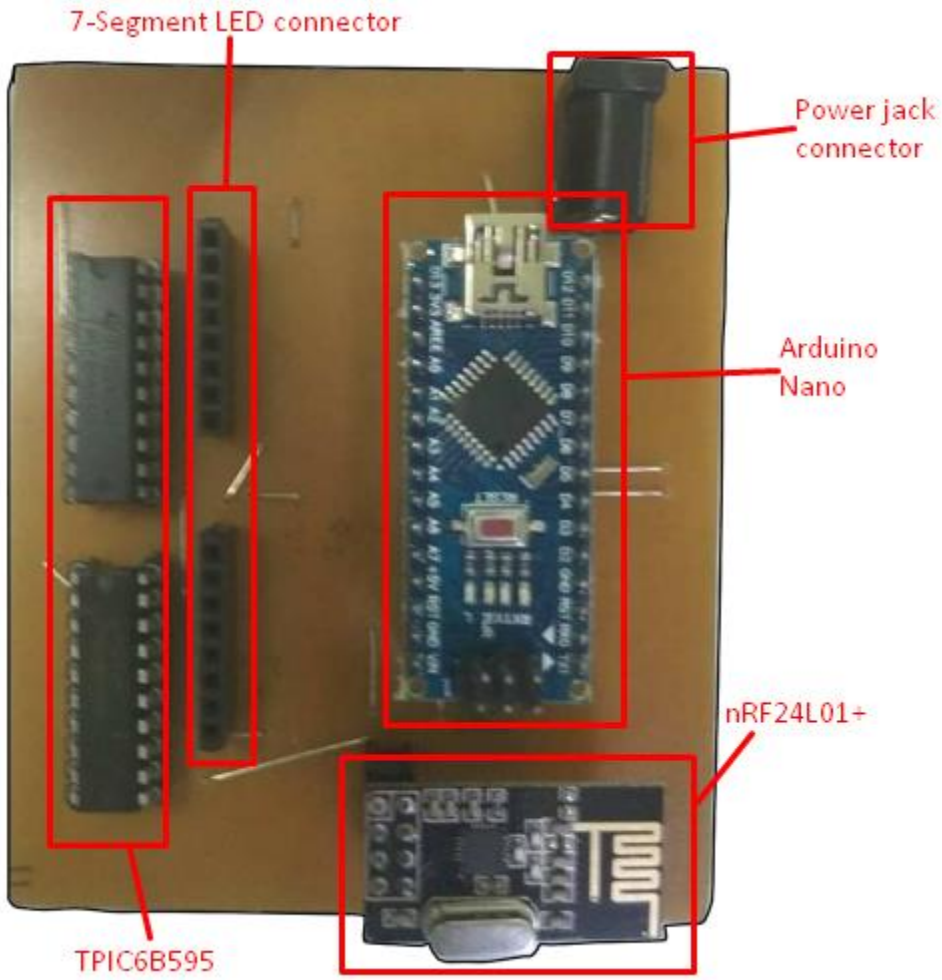


Figure 18: Sign node interface

This is the PCB of sign node. This PCB will connect to 7-segment led in our sign to display the number of available lots. In product environment, the placement of this PCB will be depended on its connected sign, but in demo environment, for simplicity's sake, we also placed this directly under our model, same as lot node.

5.2. User Interface Design

References to main document, Section D – 5.2. User Interface Design

6. Database Design

References to main document, Section D – 6. Database Design

7. Algorithms

7.1. GetDistance formula

To calculate the long distance with more accuracy, we must calculate the distance in the

circle by using Haversine formula

$$hav\left(\frac{d}{r}\right) = hav(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\, hav(\lambda_2 - \lambda_1)$$

hav is the haversine function

$$hav(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

d is the distance between the two points

r is the radius of the sphere,

φ_1, φ_2: latitude of point 1 and latitude of point 2, in radians

λ_1, λ_2: longitude of point 1 and longitude of point 2, in radians

By apply the inverse haversine or by using the arcsine, we can calculate d:

$$d = r.hav^{-1}(h) = 2r.\arcsin\left(\sqrt{h}\right)$$

$$d = 2r.\arcsin(\sqrt{hav(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\, hav(\lambda_2 - \lambda_1)})$$

$$d = 2r.\arcsin(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\, sin^2(\lambda_2 - \lambda_1)})$$

7.2. CRC Error Detection

7.2.1. Definition

The aim of an error detection technique is to enable the receiver of a message transmitted through a noisy (error-introducing) channel to determine whether the message has been corrupted. To do this, the transmitter constructs a value (called a checksum) that is a function of the message, and appends it to the message. The receiver can then use the same function to calculate the checksum of the received message and compare it with the appended checksum to see if the message was correctly received.

7.2.2. Define Problem

Because we do not require a license to operate radio equipment in the 2.4GHz band, as a result, other transmitter can broadcast on the same frequency that our devices use.

7.2.3. Solution theory

Two important aspects required to form a strong checksum function:

- **WIDTH:** a register width wide enough to provide a low a-priori probability of failure (e.g. 32-bits gives a 1/2^32 chance of failure).
- **CHAOS:** a formula that gives each input byte the potential to change any number of bits in the register.

For that reason, the CRC schemes use division with some changes so that if more bytes were added to the message, the checksum value could change radically again very quickly. All the arithmetic performed during CRC calculation is performed in binary with no carries, we will call it CRC arithmetic.

Adding two numbers in CRC arithmetic is the same as adding numbers in ordinary binary arithmetic except there is no carry. This means that each pair of corresponding bits determine the corresponding output bit without reference to any other bit positions. For example:

	1	0	0	1	1	0	1	1
+	1	1	0	0	1	0	1	0
	0	1	0	1	0	0	0	1

Figure 19: CRC arithmetic addition

Subtraction is identical:

	1	0	0	1	1	0	1	1
-	1	1	0	0	1	0	1	0
	0	1	0	1	0	0	0	1

Figure 20: CRC arithmetic subtraction

In fact, both addition and subtraction in CRC arithmetic is equivalent to the XOR operation, and the XOR operation is its own inverse. This effectively reduces the operations of the first level of power (addition, subtraction) to a single operation that is its own inverse. This is a very convenient property of the arithmetic.

Having defined addition, we can move to multiplication and division. Multiplication is absolutely straightforward, being the sum of the first number, shifted in accordance with the second number. (note: the sum uses CRC addition)

			1	1	0	1	
		x	1	0	1	1	
			1	1	0	1	
		1	1	0	1		
	0	0	0	0			
1	1	0	1				
1	1	1	1	1	1	1	

Figure 21: CRC arithmetic multiplication

Division is a little messier as we need to know when "a number goes into another number". To do this, we invoke the weak definition of magnitude defined earlier: that X is greater than or equal to Y if the position of the highest 1 bit of X is the same or greater than the position of the highest 1 bit of Y.

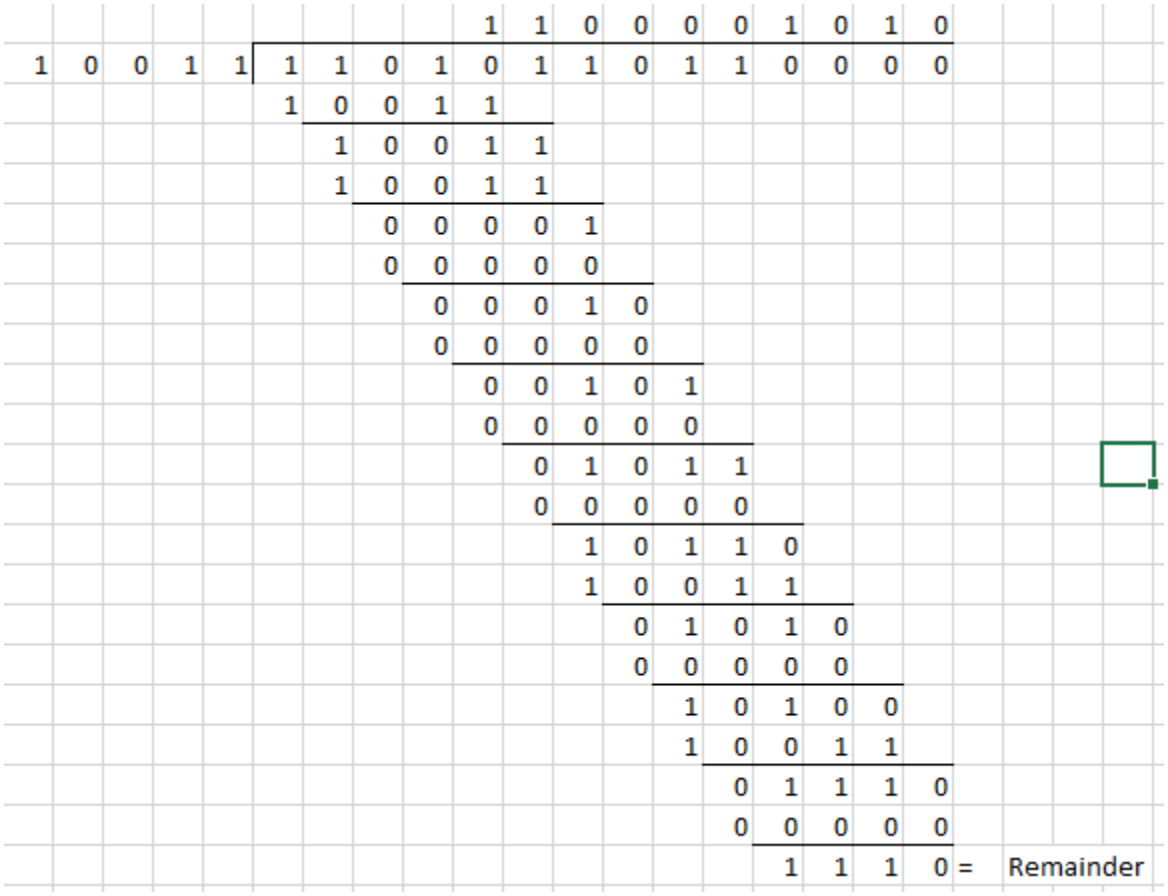


Figure 22: CRC arithmetic division

Having defined CRC arithmetic, we can now frame a CRC calculation as simply a division, because that's all it is. To perform a CRC calculation, we need to choose a divisor. In math marketing speak the divisor is called the "generator polynomial" or simply the "polynomial", and is a key parameter of any CRC algorithm. As a compromise, we will refer to the CRC polynomial as the "poly".

The width (position of the highest 1 bit) of the poly is very important as it dominates the whole calculation. Typically, widths of 16 or 32 are chosen so as to simplify implementation on modern computers. The width of a poly is the actual bit position of the highest bit. For example, the width of 10011 is 4, not 5. For the purposes of example, we will choose a poly of 10011 (of width W of 4).

Having chosen a poly, we can proceed with the calculation. This is simply a division (in CRC arithmetic) of the message by the poly. The only trick is that W zero bits are appended to the message before the CRC is calculated. Thus we have:

- Original message: 1101011011
- Poly: 10011
- Message after appending W zeros: 11010110110000

This is the same division as above figure, so we got the remainder, which is the calculated

checksum is 1110. Usually, the checksum is then appended to the message and the result transmitted. In this case the transmission would be: 11010110111110. This ends the calculation.

A summary of the operation of the class of CRC algorithms:

- Choose a width W, and a poly G (of width W).
- Append W zero bits to the message. Call this M'.
- Divide M' by G using CRC arithmetic. The remainder is the checksum.

7.2.4. Implementation

To implement a CRC algorithm all we have to do is implement CRC division. There are two reasons why we cannot simply use the divide instruction of whatever machine we are on. The first is that we have to do the divide in CRC arithmetic. The second is that the dividend might be ten megabytes long, and todays processors do not have registers that big.

So to implement CRC division, we have to feed the message through a division register. At this point, we have to be absolutely precise about the message data. In all the following examples the message will be considered to be a stream of bytes (each of 8 bits) with bit 7 of each byte being considered to be the most significant bit (MSB). The bit stream formed from these bytes will be the bit stream with the MSB (bit 7) of the first byte first, going down to bit 0 of the first byte, and then the MSB of the second byte and so on.

For the purposes of example, consider a poly with W=4 and the poly=10111. Then, the perform the division, we need to use a 4-bit register:

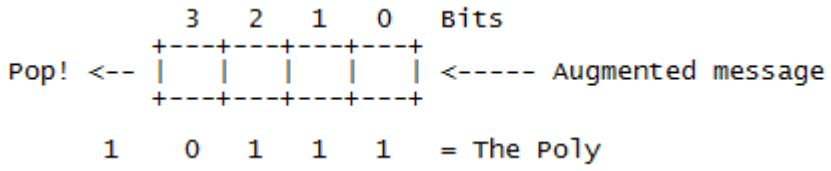


Figure 23: CRC implementation register

(Augmented message is the message followed by W zero bits)

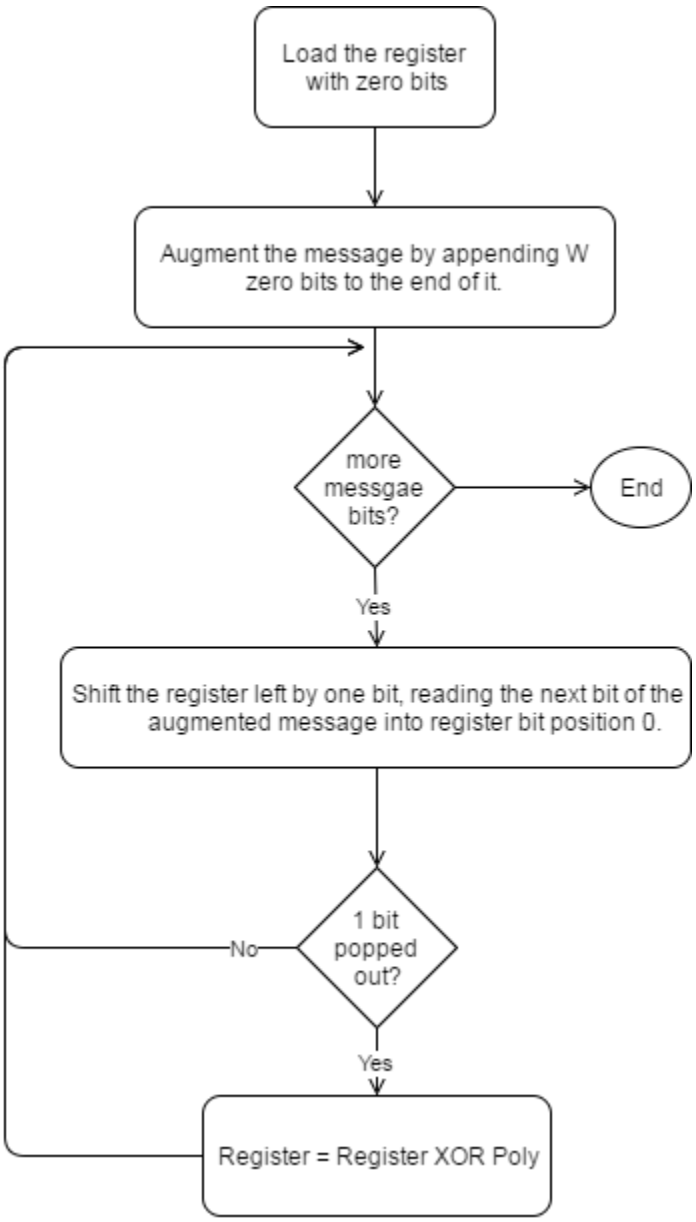


Figure 24: CRC implementation simple flow

7.2.5. Table-driven implementation

The implementation above is a good starting point because it corresponds directly to the theory presented so far, and because it is so simple. However, because it operates at the bit level, it is rather awkward to code (even in C), and inefficient to execute (it has to loop once for each bit). To speed it up, we need to find a way to enable the algorithm to process the message in units larger than one bit.

For the purposes of discussion, let us switch from a 4-bit poly to a 32-bit one. Our register looks much the same, except the boxes represent bytes instead of bits, and the Poly is 33 bits (one implicit 1 bit at the top and 32 "active" bits) (W=32).

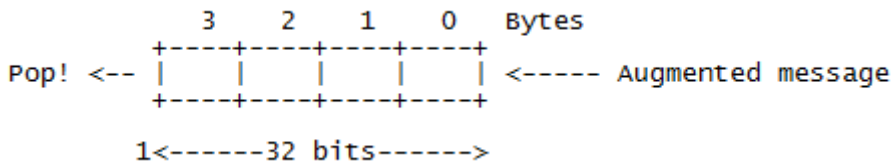


Figure 25: CRC 32-bits register

Consider for a moment that we use the top 8 bits of the register to calculate the value of the top bit of the register during the next 8 iterations. Suppose that we drive the next 8 iterations using the calculated values (which we could perhaps store in a single byte register and shift out to pick off each bit). Then we note three things:

- The top byte of the register now doesn't matter. No matter how many times and at what offset the poly is XORed to the top 8 bits, they will all be shifted out the right hand side during the next 8 iterations anyway.
- The remaining bits will be shifted left one position and the rightmost byte of the register will be shifted in the next byte.
- While all this is going on, the register will be subjected to a series of XOR's in accordance with the bits of the pre-calculated control byte.

Now consider the effect of XORing in a constant value at various offsets to a register. For example:

```
0100010  Register
...0110  XOR this
..0110.  XOR this
0110...  XOR this
-------
0011000
-------
```

The point of this is that you can XOR constant values into a register to your heart's delight, and in the end, there will exist a value which when XORed in with the original register will have the same effect as all the other XORs.

Putting all the pieces together we have an algorithm that goes like this:

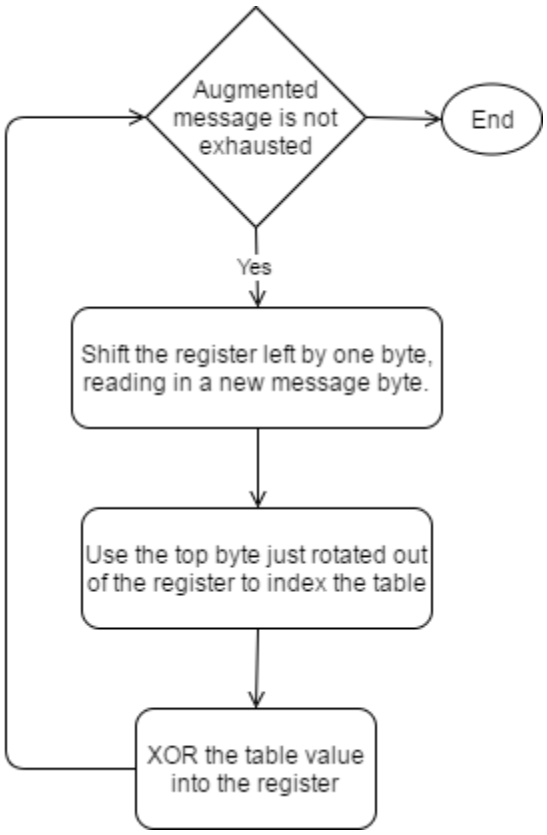


Figure 26: CRC implementation table-driven flow

For the actual implementation in project, we use 3 bytes in payload package as checksum, so we need to implement CRC24. Here is the table we use for CRC24 table-driven implementation:

```
0x00000000, 0x00864cfb, 0x008ad50d, 0x000c99f6, 0x0093e6e1, 0x0015aa1a, 0x001933ec,
0x009f7f17, 0x00a18139, 0x0027cdc2, 0x002b5434, 0x00ad18cf, 0x003267d8, 0x00b42b23,
0x00b8b2d5, 0x003efe2e, 0x00c54e89, 0x00430272, 0x004f9b84, 0x00c9d77f, 0x0056a868,
0x00d0e493, 0x00dc7d65, 0x005a319e, 0x0064cfb0, 0x00e2834b, 0x00ee1abd, 0x00685646,
0x00f72951, 0x007165aa, 0x007dfc5c, 0x00fbb0a7, 0x000cd1e9, 0x008a9d12, 0x008604e4,
0x0000481f, 0x009f3708, 0x00197bf3, 0x0015e205, 0x0093aefe, 0x00ad50d0, 0x002b1c2b,
0x002785dd, 0x00a1c926, 0x003eb631, 0x00b8faca, 0x00b4633c, 0x00322fc7, 0x00c99f60,
0x004fd39b, 0x00434a6d, 0x00c50696, 0x005a7981, 0x00dc357a, 0x00d0ac8c, 0x0056e077,
0x00681e59, 0x00ee52a2, 0x00e2cb54, 0x006487af, 0x00fbf8b8, 0x007db443, 0x00712db5,
0x00f7614e, 0x0019a3d2, 0x009fef29, 0x009376df, 0x00153a24, 0x008a4533, 0x000c09c8,
0x0000903e, 0x0086dcc5, 0x00b822eb, 0x003e6e10, 0x0032f7e6, 0x00b4bb1d, 0x002bc40a,
0x00ad88f1, 0x00a11107, 0x00275dfc, 0x00dced5b, 0x005aa1a0, 0x00563856, 0x00d074ad,
0x004f0bba, 0x00c94741, 0x00c5deb7, 0x0043924c, 0x007d6c62, 0x00fb2099, 0x00f7b96f,
0x0071f594, 0x00ee8a83, 0x0068c678, 0x00645f8e, 0x00e21375, 0x0015723b, 0x00933ec0,
0x009fa736, 0x0019ebcd, 0x008694da, 0x0000d821, 0x000c41d7, 0x008a0d2c, 0x00b4f302,
0x0032bff9, 0x003e260f, 0x00b86af4, 0x002715e3, 0x00a15918, 0x00adc0ee, 0x002b8c15,
0x00d03cb2, 0x00567049, 0x005ae9bf, 0x00dca544, 0x0043da53, 0x00c596a8, 0x00c90f5e,
0x004f43a5, 0x0071bd8b, 0x00f7f170, 0x00fb6886, 0x007d247d, 0x00e25b6a, 0x00641791,
0x00688e67, 0x00eec29c, 0x003347a4, 0x00b50b5f, 0x00b992a9, 0x003fde52, 0x00a0a145,
0x0026edbe, 0x002a7448, 0x00ac38b3, 0x0092c69d, 0x00148a66, 0x00181390, 0x009e5f6b,
0x0001207c, 0x00876c87, 0x008bf571, 0x000db98a, 0x00f6092d, 0x007045d6, 0x007cdc20,
0x00fa90db, 0x0065efcc, 0x00e3a337, 0x00ef3ac1, 0x0069763a, 0x00578814, 0x00d1c4ef,
0x00dd5d19, 0x005b11e2, 0x00c46ef5, 0x0042220e, 0x004ebbf8, 0x00c8f703, 0x003f964d,
0x00b9dab6, 0x00b54340, 0x00330fbb, 0x00ac70ac, 0x002a3c57, 0x0026a5a1, 0x00a0e95a,
0x009e1774, 0x00185b8f, 0x0014c279, 0x00928e82, 0x000df195, 0x008bbd6e, 0x00872498,
0x00016863, 0x00fad8c4, 0x007c943f, 0x00700dc9, 0x00f64132, 0x00693e25, 0x00ef72de,
0x00e3eb28, 0x0065a7d3, 0x005b59fd, 0x00dd1506, 0x00d18cf0, 0x0057c00b, 0x00c8bf1c,
0x004ef3e7, 0x00426a11, 0x00c426ea, 0x002ae476, 0x00aca88d, 0x00a0317b, 0x00267d80,
0x00b90297, 0x003f4e6c, 0x0033d79a, 0x00b59b61, 0x008b654f, 0x000d29b4, 0x0001b042,
0x0087fcb9, 0x001883ae, 0x009ecf55, 0x009256a3, 0x00141a58, 0x00efaaff, 0x0069e604,
0x00657ff2, 0x00e33309, 0x007c4c1e, 0x00fa00e5, 0x00f69913, 0x0070d5e8, 0x004e2bc6,
0x00c8673d, 0x00c4fecb, 0x0042b230, 0x00ddcd27, 0x005b81dc, 0x0057182a, 0x00d154d1,
0x0026359f, 0x00a07964, 0x00ace092, 0x002aac69, 0x00b5d37e, 0x00339f85, 0x003f0673,
0x00b94a88, 0x0087b4a6, 0x0001f85d, 0x000d61ab, 0x008b2d50, 0x00145247, 0x00921ebc,
0x009e874a, 0x0018cbb1, 0x00e37b16, 0x006537ed, 0x0069ae1b, 0x00efe2e0, 0x00709df7,
0x00f6d10c, 0x00fa48fa, 0x007c0401, 0x0042fa2f, 0x00c4b6d4, 0x00c82f22, 0x004e63d9,
0x00d11cce, 0x00575035, 0x005bc9c3, 0x00dd8538
```

Figure 27: CRC24 precomputed table

E. System Implementation & Test

1. Introduction

1.1. Overview

This section has all necessary information about test plan, test case and its result, the environment for testing and test pass/fail criteria

1.2. Test Approach

White-box: Developers self-test on code in which function they developed

In this section, Black-box testing will be used to test whether the whole system meets the following objectives

2. Database Relationship Diagram

References to main document, Section E – 2. Database Relationship Diagram

3. Performance Measures

The most important function in our project is the working of RF network, specifically the polling process of parking lots. The RF module we use is a 2.4GHz, which may contain lots of radio-frequency interference, so we perform a performance test with following conditions to check the reliability of the system:

- The location is in a personal home, so there is more interference than a real car park.
- We set the Hub to continuously polling 6 lot nodes place near each other. After the Hub polls data from all 6 lot nodes, it will count as 1 round.
- For simplicity's sake and easier to check logging data, we set the condition for the Hub to send a package and wait ACK package in 100ms. After 100ms, the Hub will resend the package. After 5 times resend and the Hub still not received the ACK package, the polling for that node will count as failed and move to the next node, and that round will also count as failed.

We let the system run continuously for around 16h (~11h PM to 3h PM the next day) and get the following results:

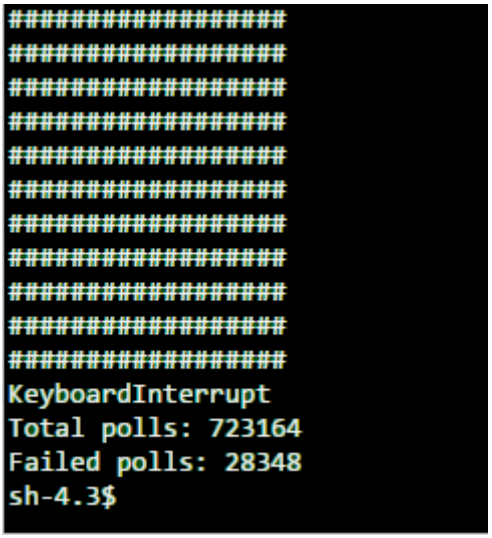


Figure 28: Performance test result

The system performed 723 164 rounds of polling and got 28 348 rounds that got at least 1 lot node didn't response after 500ms. After the test, we get that only **96.08%** of the package send from Hub is successfully received and executed. The test also shows that if the polling of lot in current round is failed, the next round will come quickly so the information Hub owns is reliable. In the current setting, with a large amount of lot nodes, when the system meets radio-frequency interference it may got a little lag.

4. Test Plan

The purpose of the test is to verify the functionality of the system.

Functions need to be tested. Functions to ensure technical requirements and system requirements of the user. Error will not happen after the trial

The next content will describe which function will be tested, which will not and plan for them.

4.1. Features to be tested

References to main document, Section E – 4.1 Features to be tested

4.2. Features not to be tested:

N/A

4.3. Test environment

Web server: Firefox 52.0.2(32-bit)

OS: Windows 10

Android application: Android 5.0

Hardware: Test on Raspberry Pi 3

4.4. Test Pass/Fail Criteria

For system testing, the criteria are:

- 90% of the test cases must pass
- 100% of test cases about hardware module must pass
- All test cases dealing with critical functionality must pass
- All medium and high severity defects must be fixed
- Test coverage must be at least 90%

5. System Testing Test Case

References to main document, Section E – 5. System Testing Test Case

F. User's Manual

1. Installation Guide

References to main document, Section F – 1. Installation Guide

2. User Guide

References to main document, Section F – 2. User Guide

G. Appendix

How to Connect a Voltage Regulator in a Circuit available at http://www.learningaboutelectronics.com/Articles/How-to-connect-a-voltage-regulator-in-a-circuit

Calculate distance from geo location available at https://msdn.microsoft.com/en-us/library/system.device.location.geocoordinate.getdistanceto(v=vs.110).aspx

Deriving the Haversine Formula available at http://mathforum.org/library/drmath/view/51879.html

Raspberry Pi 3 model B datasheet available at https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Arduino Board Nano information available at https://www.arduino.cc/en/Main/arduinoBoardNano

nRF24L01+ datasheet available at https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf

HMC5883L 3-Axis Digital Compass datasheet available at https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf

TPIC6b595 datasheet available at http://www.ti.com/lit/ds/symlink/tpic6b595.pdf

TIP122 datasheet available at https://www.onsemi.com/pub/Collateral/TIP120-D.PDF

Principles of communications networks and systems By Nevio Benvenuto and Michele Zorzi available at http://as.wiley.com/WileyCDA/WileyTitle/productCd-0470744316.html

A painless guide to CRC Error detection algorithms By Ross N. Williams available at http://www.zlib.net/crc_v3.txt

The Internet of Things: An Overview By Karen Rose, Scott Eldridge and Lyman Chapin available at https://www.internetsociety.org/doc/iot-overview

Task sheet

No.	Product Deliverables	Task	TrungTNM	HiepBP	HuyNDP	Size
1	Report 1 - Introduction	Project Information	O			
		Introduction	O			
		Current Situation	O			
		Problem Definition		O		
		Proposed Solution			O	
		Functional Requirements		O		
		Roles and Responsibilities	O			
		Conclusion			O	
		Review and merge document	O			
2	Report 2 – Software and Hardware Project Management Plan	Problem Definition			O	
		Project organization	O			
		Project management plan	O			
		Coding Convention		O		
		Review and merge document	O			
3	Report 3 – Software and Hardware Requirement Specification	Software Requirement Specification	O			
		Hardware Requirement Specification				
		Hardware requirement			O	
		System Overview Use Case		O		
		List of Use Case		O		
		Software System Attribute	O			
		Conceptual Diagram		O		
		Review and merge document	O			
4	Report 4 – Software and Hardware Design Description	Design Overview			O	
		System Architectural Design				
		Hardware Program Architecture Description	O	O	O	
		Mobile Application Architecture Description	O			
		Web API Architecture Description		O		
		Component Diagram			O	
		Detailed Description				

		Web API Detailed Description		O		
		Mobile Detailed Description	O			
		Embedded Program Detailed Description	O	O	O	
		Hardware Detailed Description	O	O	O	
		Interface				
		Component Interface			O	
		User Interface Design		O		
		Database Design		O		
		Algorithms	O			
		Review and merge document	O			
5	Report 5 – System Implementation and Test	Introduction		O		
		Database Relationship Diagram		O		
		Performance Measures	O			
		Test Plan			O	
		System Testing Test Case				
		Component Testing			O	
		API Web Service Testing		O		
		Mobile Testing	O			
		Review and merge document	O			
6	Report 6 – User's Manual	Installation Guide			O	
		User Guide			O	
		Review and merge document	O			