MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Report 2

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

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

Table 1: Definitions, Acronyms and Abbreviations

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.

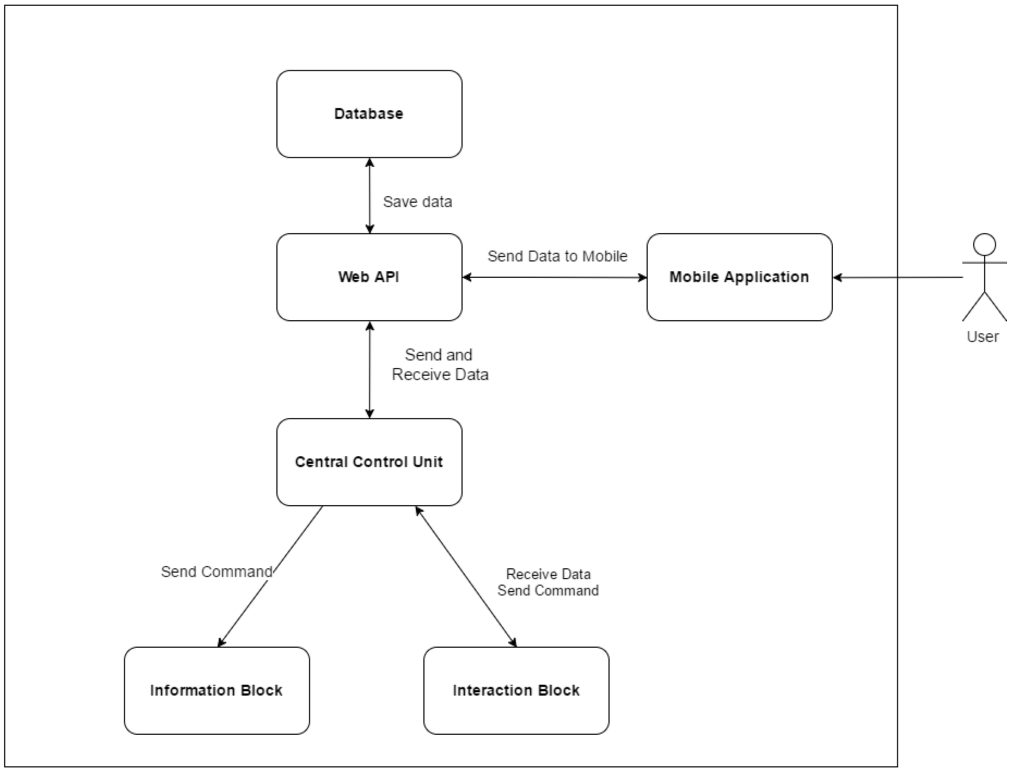

Figure 1: Project Block Diagram

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 2: Database requirement

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

Table 3: API Service Requirement

For CCU

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

Table 4: 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.

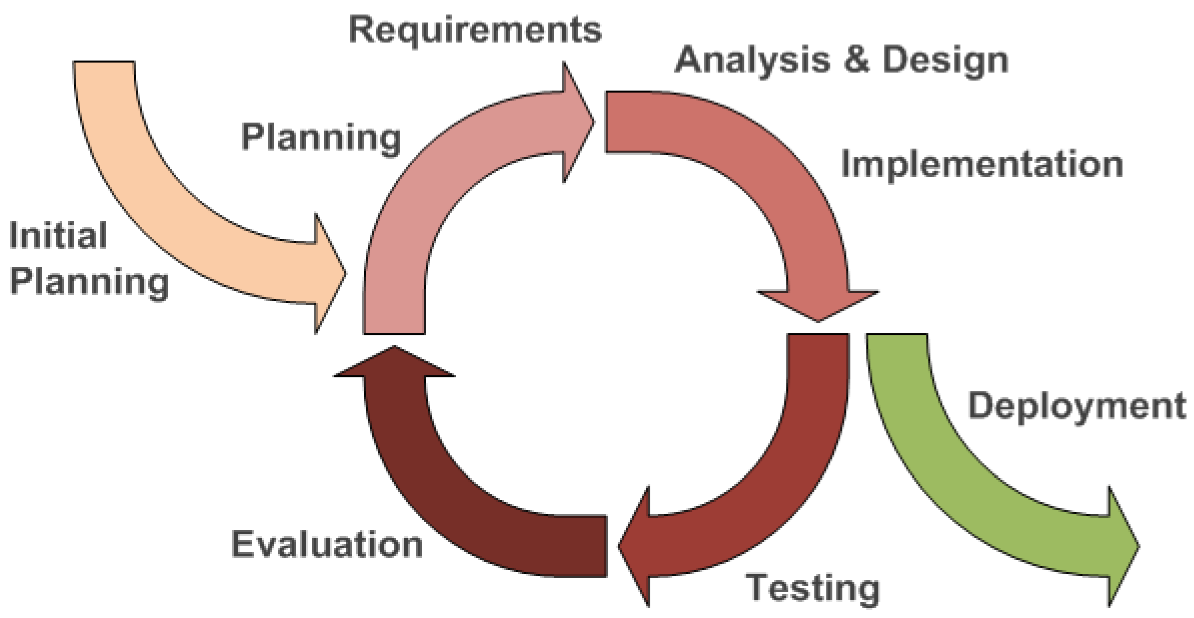

Figure 2: Iterative and Incremental development

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 5: 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 6: 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 7: Techniques

3. Project Management Plan

3.1. System development life cycle

Incremental development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross-discipline work, from the requirements to the deployment. The Unified Process groups increments/iterations into phases:

Phase	Description	Deliverables	Risks
Inception	In this phase, we will identify project scope, requirements (functional and non-functional) and risks at a high level but in enough detail that work can be estimated	• Introduction of proposed system • Software and Hardware requirement specification • Project Task Plan and Risk	• The lack of knowledge may lead to misunderstand of the requirement • The inexperienced of team may lead to deficient in Task Plan and Risk
Elaboration	Delivers a working architecture that mitigates the top risks and fulfills the non-functional requirements	• Software and Hardware design document	• System architecture or design issues may arise because not all requirements are gathered
Construction	Incrementally fills-in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements	• Completed and fully tested system • Implementation and Test document	• The lack of knowledge of team member about hardware • The inexperienced of team may lead to missing test cases • There may be hidden High to Critical bugs in the system
Transition	Delivers the system into the production operating environment	• User manual document • Installation guide document • The final and full version document of the system	• The documents may not fully describe the system

Table 8: System Development Life Cycle

Each of the phases may be divided into 1 or more iterations, which are usually time-boxed rather than feature-boxed.

3.2. Plan Detail

Iteration	Scope	Evaluation	Activities	Estimated Duration	Assign Responsibilities
Initial Iteration	Initial team workplace and identify project scope	A working team environment	• Set up Git Repository with Gitflow • Set up Slack • Set up Trello	5 days	TrungTNM, HiepBP, HuyNĐP
Iteration 1	Identify boundaries of the system, planning software and hardware. Create a proof-of-concept prototype.	Report 1, Report 2 and a proof-of-concept prototype	• Introduction document • Software and Hardware Project Management Plan document • Proof-of-concept prototype	15 days	TrungTNM, HiepBP, HuyNĐP
Iteration 2	Produce an architectural prototype	Report 3, Report 4 and an architectural prototype	• Software and Hardware Requirement Specification document • Software and Hardware Design Description document • Architectural prototype	15 days	TrungTNM, HiepBP, HuyNĐP

Iteration 3	Build the product (up to beta release)	Report 5 and a working product (beta release)	• System Implementation and Test document • PCB • Mobile Application • Web API Server	15 days	TrungTNM, HiepBP, HuyNĐP
Iteration 4	Finish the product (full product release)	Report 6 and the completed product	• Software and Hardware User's Manual document • Product demonstration model	15 days	TrungTNM, HiepBP, HuyNĐP
Final Iteration	Prepare for Demo Day	Final Documentation, Presentation Slide	• Final Document • Mini Document • CD contains all source code • Presentation Slide	5 days	TrungTNM, HiepBP, HuyNĐP

Table 9: System Development Detail Plan

3.3. All Meeting Minutes

All meeting minutes are saved at:

https://github.com/Hinaka/-FPT-CAPSTONE-PGSS/tree/master/Common

4. Coding Convention

4.1. C/C++ Convention

C/C++: Using to develop program and solve algorithm on hardware.

Summary:

- Naming Convention:
  - Using Pascal case for class name.
  - Using Camel case for function, variable's name.
  - The #define and global variable's name must uppercase and separate by underscore. Ex: GLOBAL_VARIABLE.
- Commenting Convention:
  - Place the comment on the separate line with function.
  - Place the comment at the end of the line, which has calculation formula.

More details about coding conventions for C/C++ language by Google:

https://google-styleguide.googlecode.com/svn/trunk/cppguide.html

4.2. C#, ASP.NET Convention

C#: Using to develop Web API

Summary:

- Naming Convention:
  - Use Camel case for variable's name.
  - Use Pascal case for class's name, function's name.
  - Global variable's name must uppercase and separate by underscore.

More detail about code conventions for C# language by Microsoft:

https://msdn.microsoft.com/en-us/library/ff926074.aspx

4.3. Python Convention

Python: Using to develop program on Raspberry Pi

Summary:

- Naming Convention:
  - "Internal" means internal to a module or protected or private within a class.

- Prepending a single underscore (_) has some support for protecting module variables and functions (not included with import * from). Prepending a double underscore (__) to an instance variable or method effectively serves to make the variable or method private to its class (using name mangling).
- Place related classes and top-level functions together in a module. Unlike Java, there is no need to limit yourself to one class per module.
- Use Pascal Case for class names, but lower_with_under.py for module names.

More detail about code conventions for Python by Google:

https://google.github.io/styleguide/pyguide.html

4.4. Android Convention

Android use Java and HTML to develop mobile application

Summary:

- Naming convention:
  - Follow basic principle of <WHAT>_<WHERE>_<DESCRIPTION>_<SIZE> for resource names
  - Follow basic principle of <WHAT>_<WHERE>.XML for layout
  - Follow basic principle of <WHERE>_<DESCRIPTION> for string resources
  - Follow basic principle of <WHERE>_<DESCRIPTION>_<SIZE> for drawable resource
  - Follow basic principle of <WHAT>_<WHERE>_<DESCRIPTION> for IDs

More detail about code conventions for Android by Google and our team:

https://source.android.com/source/code-style.html

https://github.com/Hinaka/-FPT-CAPSTONE-PGSS/tree/master/Common