|  |  |
| --- | --- |
|  | MINISTRY OF EDUCATION AND TRAINING |

FPT UNIVERSITY

Capstone Project Document

**DESIGN AND CONSTRUCTION SUN DRYING WET CLOTHES SYSTEM**

|  |  |
| --- | --- |
| **Group 2** | |
| **Group members** | **Hoàng Phi Long – SE62021**  **Nguyễn Đình Phong – SE**  **Trịnh Bình – SE** |
| **Supervisor** | **Nguyễn Đức Lợi** |
| **Ext. Supervisor** | **N/A** |
| **Capstone Project code** | **DCDCS** |

-Ho Chi Minh City, **June 26th 2018**

*This page is intentionally left blank*

# Table of Contents

[Table of Contents 2](#_Toc522291107)

[List of Tables 5](#_Toc522291108)

[List of Figures 6](#_Toc522291109)

[Definitions, Acronyms and Abbreviations 7](#_Toc522291110)

[A. Introduction 8](#_Toc522291111)

[1. Project Information 8](#_Toc522291112)

[2. Introduction 8](#_Toc522291113)

[3. Current Situation 8](#_Toc522291114)

[4. Problem Definition 9](#_Toc522291115)

[5. Proposed Solution 9](#_Toc522291116)

[5.1 Feature Functions 9](#_Toc522291117)

[5.2 Advantages and Disadvantages 9](#_Toc522291118)

[6. Functional Requirements 10](#_Toc522291119)

[7. Role and Responsibility 11](#_Toc522291120)

[B. Software Project Management Plan 11](#_Toc522291121)

[1. Problem Definition 11](#_Toc522291122)

[1.1 Name of this Capstone project 11](#_Toc522291123)

[1.2 Problem Abstract 12](#_Toc522291124)

[1.3 Project Overview 12](#_Toc522291125)

[2. Project Organization 14](#_Toc522291126)

[2.1 Software Process Model 14](#_Toc522291127)

[2.2 Roles and Responsibility 15](#_Toc522291128)

[2.3 Tools and Techniques 16](#_Toc522291129)

[3. Project Management Plan 18](#_Toc522291130)

[3.1 System Development Life-cycle 18](#_Toc522291131)

[3.2 Plan Detail 20](#_Toc522291132)

[C. Software & Hardware Requirement Specification 23](#_Toc522291133)

[1. User Requirement Specification 23](#_Toc522291134)

[2. System Requirement Specification 23](#_Toc522291135)

[2.1 External Interface Requirement 23](#_Toc522291136)

[2.2 System Overview Usecase 25](#_Toc522291137)

[3. Hardware Requirement Specification 27](#_Toc522291138)

[3.1 Hardware Interface 27](#_Toc522291139)

[4. Conceptual Diagram 28](#_Toc522291140)

[D. Software & Hardware Design Description 29](#_Toc522291141)

[1. Design Overview 29](#_Toc522291142)

[2. System Architectural Design 30](#_Toc522291143)

[2.1 API Web Server Architectural Design 31](#_Toc522291144)

[2.2 Android Application Architectural Design 33](#_Toc522291145)

[2.3 Hardware System Architecture 35](#_Toc522291146)

[3. Component Diagram 37](#_Toc522291147)

[4. Detailed Description 39](#_Toc522291148)

[3.3 Class Diagram 39](#_Toc522291149)

[3.4 Interaction Diagram 46](#_Toc522291150)

[4. Database Design 51](#_Toc522291151)

[4.1 Entity Relational Database (ERD) 51](#_Toc522291152)

[4.2 Data Dictionary 51](#_Toc522291153)

[5. Algorithms 52](#_Toc522291154)

[5.1 System Control 52](#_Toc522291155)

[E. Task Sheet 54](#_Toc522291156)

[F. Appendix 55](#_Toc522291157)

# List of Tables

[Table 1: General Roles and Responsibilities of Member 11](#_Toc522291158)

[Table 2: Hardware development environment requirement for DCDCS System 14](#_Toc522291159)

[Table 3: Software development environment requirement for DCDCS System 14](#_Toc522291160)

[Table 4: Roles and responsibilities 16](#_Toc522291161)

[Table 5: Tools and techniques 17](#_Toc522291162)

[Table 6: Project task planning 19](#_Toc522291163)

[Table 7: Plain Detail - Requirement Analysis 20](#_Toc522291164)

[Table 8: Plain Detail - Design 21](#_Toc522291165)

[Table 9: Plain Detail – Implementation 22](#_Toc522291166)

[Table 10: Plain Detail –Testing 22](#_Toc522291167)

[Table 11: Plain Detail –Maintenance 22](#_Toc522291168)

[Table 12: Data dictionary for conceptual diagram 28](#_Toc522291169)

[Table 13: Component diagram dictionary 38](#_Toc522291170)

[Table 14: API Web server class diagram dictionary 42](#_Toc522291171)

[Table 15: Hardware controller class diagram dictionary 45](#_Toc522291172)

[Table 16: Entity diagram data dictionary 51](#_Toc522291173)

# List of Figures

[Figure 1: Waterfall methodology 15](file:///D:\Capstone%20Document\Summary.docx#_Toc522291174)

[Figure 2: Hardware system overview usecase diagram 25](file:///D:\Capstone%20Document\Summary.docx#_Toc522291175)

[Figure 3: Android application overview usecase diagram 26](#_Toc522291176)

[Figure 4: System block diagram 27](file:///D:\Capstone%20Document\Summary.docx#_Toc522291177)

[Figure 5: Conceptual diagram 28](file:///D:\Capstone%20Document\Summary.docx#_Toc522291178)

[Figure 6: System overview architecture 30](file:///D:\Capstone%20Document\Summary.docx#_Toc522291179)

[Figure 7: API Web server architecture 31](file:///D:\Capstone%20Document\Summary.docx#_Toc522291180)

[Figure 8: Android application internal architecture 33](file:///D:\Capstone%20Document\Summary.docx#_Toc522291181)

[Figure 9: Hardware system architecture 35](file:///D:\Capstone%20Document\Summary.docx#_Toc522291182)

[Figure 10: Component diagram 37](file:///D:\Capstone%20Document\Summary.docx#_Toc522291183)

[Figure 11: API Web Server Class Diagram Part 1 39](#_Toc522291184)

[Figure 12: API Web Server Class Diagram Part 1 39](file:///D:\Capstone%20Document\Summary.docx#_Toc522291185)

[Figure 13: API Web Server Class Diagram Part 2 40](file:///D:\Capstone%20Document\Summary.docx#_Toc522291186)

[Figure 14: API Web Server Class Diagram Part 3 41](file:///D:\Capstone%20Document\Summary.docx#_Toc522291187)

[Figure 15: Hardware system controller class diagram part 1 43](file:///D:\Capstone%20Document\Summary.docx#_Toc522291188)

[Figure 16: Hardware system controller class diagram part 2 44](file:///D:\Capstone%20Document\Summary.docx#_Toc522291189)

[Figure 17: Control system with android app sequence diagram 46](#_Toc522291190)

[Figure 18: Update system information sequence diagram 47](file:///D:\Capstone%20Document\Summary.docx#_Toc522291191)

[Figure 19: Control DC activity diagarm 48](file:///D:\Capstone%20Document\Summary.docx#_Toc522291192)

[Figure 20: Control Dryer activity diagram 49](file:///D:\Capstone%20Document\Summary.docx#_Toc522291193)

[Figure 21: Auto control activity diagram 50](#_Toc522291194)

[Figure 22: Entity Relational Database 51](file:///D:\Capstone%20Document\Summary.docx#_Toc522291195)

[Figure 23: System Control overview flowchart 53](file:///D:\Capstone%20Document\Summary.docx#_Toc522291196)

# Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| Name | Definition |
| DCDCS | **D**esign and **C**onstruction sun **D**rying wet **C**lothes **S**ystem |
| RF | Radio frequency |
| HTTP | Hypertext transfer protocol |
| I2C | Inter-Integrated Circuit |
| UART | Universal asynchronous receiver-transmitter |
| DIY | Do it yourself |
| REST | Representational state transfer |
| API | Application programming interface |
| GPIO | General-purpose input/output |
| I/O | Input/Output |

# Introduction

## Project Information

* **Project name:** DESIGN AND CONSTRUCTION SUN DRYING WET CLOTHES SYSTEM
* **Project Code:** DCDCS
* **Product Type:** Embedded Device, Android Application, API Web Server
* **Start Date:** 14/06/2018
* **End Date:** 31/08/2018

## Introduction

In this document, we introduce a solution for automatic clothes drying system. We build a system, which use rain sensor to detect rain, ESP8266 for communication between Android application and embedded device.

This document also describes our working process in 4 months includes our perspective in the system, component designs and detailed core workflows. We hope the system will help resolve some aspects of the problem that the current face recognition systems are facing today.

## Current Situation

Vietnam is a rainy country, with 6 months of sunshine and 6 months of rainy. Vietnamese people prefer drying their clothes under sunshine, wind over using clothes dryer or another dryer machines. When the rainy season comes, Vietnamese people tend to worry about their clothes at home being wet by rains. There are a few solutions to solve this problem as known as “Smart Clothesline Rigs”. This device really expensive and not really that smart. With 13.000.000 VND, you can have controllable system with UV light, build-in dryer and remote control within 30 meters. However, this system is not really solve the core problem: Automatically collecting clothes. Therefore, we come to this solution, helping Vietnamese people not to worry about their clothes during rainy season.

## Problem Definition

With systems currently available on market

Advantage of their system:

* UV disinfection
* Built-in dryer
* Strong structure can lift up to 25kg of clothes
* Below are disadvantages of current situation:
* Current systems have high production costs
* Hard to extend
* Control manually when the electricity is down
* Cannot automatically collecting clothes when rain

## Proposed Solution

Our proposed solution is designing and construction automatic clothes drying system called DCDCS to solve missing feature of current “Smart Clothesline Rigs”. Our system will help users automatically collect clothes when it is a rain. It is much cheaper, easy to install and mobile and extendable.

DCDCS system includes a mobile app and an embedded device with following functions:

### Feature Functions

* **Mobile App:**
  + Control the system through wireless
  + Check weather information
  + Check system status
* **Embedded Device:**
  + Check system status
  + Control system through hard buttons

### Advantages and Disadvantages

* **Advantages:**
  + Low costs
  + Can detect rain very fast
  + Can control with mobile app
  + Use solar energy and have battery to storage unused energy
* **Disadvantages:**
  + Cannot detect whether the clothes is dry or not
  + Cannot detect whether rain is over or not

## Functional Requirements

Functional requirements of the system are listed as below:

* Embedded system component:
  + RESTful API communication through wireless
* Use Arduino Mega 2560 as a central circuit unit
* Show information about the system
* Time
* Temperature
* Humidity
* Control dryer
* Control clothesline
* Power supply component:
* Power supply operates for the entire system
* Distributed voltage 5V and 12V
* Auto charging
* Storing energy
* User component:
* Control the system from Android application through wireless
* Turning on/off build-in dryer
* Set timer for dryer
* Control the clothesline
* Check system status and weather
* Edit user information
* Name
* Address
* Mobile phone
* Etc
* Mobile Application component:
* Communicate with system through wireless and by REST API
* Show information about the system
* Time
* Temperature
* Humidity
* Weather (Rain or not)
* System status

## Role and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Full name | Role | Position | Contact |
| 1 | Nguyễn Đức Lợi | Project Manager | Supervisor | loinnd@fpt.edu.vn |
| 2 | Hoàng Phi Long | Developer | Leader | longhpse62021@fpt.edu.vn |
| 3 | Nguyễn Đình Phong | Developer | Member | phongndse@fpt.edu.vn |
| 4 | Trịnh Bình | Developer | Member | binhtse@fpt.edu.vn |

Table 1: General Roles and Responsibilities of Member

# Software Project Management Plan

## Problem Definition

### Name of this Capstone project

* Official name: Design and construction sun drying wet clothes system
* Vietnamese name: Thiết kế và xây dựng hệ thống phơi đồ tự động
* Abbreviation: DCDCS

### Problem Abstract

Vietnamese people work all day long. They spend time at evening and night to do their housework. One of the housework that is washing clothes then drying them. However, Vietnam is a rainy country. During rain season, everybody very worry about their drying clothes at home getting wet.

### Project Overview

#### Current Situation

Below are the problems encountered in the project:

* **Hard to improve the system:** Our system is a very simple system. However, to improve the system is a hard mission. Our system currently cannot detect when the clothes are dry, when the rain is stopped for auto collecting clothes or continue drying wet clothes. To do so, it requires mathematics model called Hidden Markov Models. However, due to the lack of knowledge in statistics and linear algebra; we are currently unable to implement this model.
* **Lack of knowledge in telecommunication:** While using ESP8266, we found out that there are some interferences during transmission. Without telecommunication, we do have hard time to detect the problem.

#### The Purposed System

According to the technology researches, we found that the simple rain sensor and ESP8266 Wi-Fi module is capable in solving the problem. We can use rain sensor detect raining and ESP8266 for wireless communication.

We assign task responsibility vertically to make sure if any member in this project fail in our team, harm would be minimized for the project.

We also build a mobile application for real-time demonstration.

#### The Boundaries of the system

Our system provides these functions:

* Automatically control clothesline when there is a rain or at night.
* Dryer system so that user can dry their clothes on rainy days
* Control system via RF Remote control
* Control system via Button on the system
* Check system status and control system via Mobile application

#### Future plans

* Implement Hidden Markov Models (HMM) for rain forecasting
* Implement the system can determine when the rain has stopped using HMM
* Build a website for user to check their account information and control the system along with mobile application
* Build a system that can detect whenever the clothes is dry or wet

#### Development Environments

##### Hardware Development Environment Requirement

For CCU clothes drying system

|  |  |
| --- | --- |
| Component | Hardware |
| Mainboard | Arduino Mega 2560 |
| Communication | Wire and cable |
| Devices | - Module real-time clock DS1307  - Rain sensor  - Humidity and Temperature sensor DHT11  - Light sensor BH1750  - DC Motor  - Nokia 5110 LCD  - 4x4 Matrix keypad  - Limit switches  - Solar Panel  - Battery  - … |
| Power source | 5V – 12V |
| Android Device | Any android mobile phone has 3G/4G or Wi-Fi connection |

Table 2: Hardware development environment requirement for DCDCS System

##### Software Development Environment Requirement

|  |  |
| --- | --- |
| Software | Name / Version |
| Operating System | Windows 7 or above |
| Environment/Run-time | Adruino Mega 2560  NodeJS |
| Modeling tool | Draw.io for UML  Proteus 8 for PCB Board |
| IDE | Visual Studio Code  Arduino IDE |
| DBMS | MongoDB |
| Source control | Git-scm and Github |
| Communication tools | Facebook Messenger  Gmail |

Table 3: Software development environment requirement for DCDCS System

## Project Organization

### Software Process Model

This project is developed using modified waterfall model. We apply modified waterfall model because it suitable with current situation in our team. We choose this model because of the following reasons:

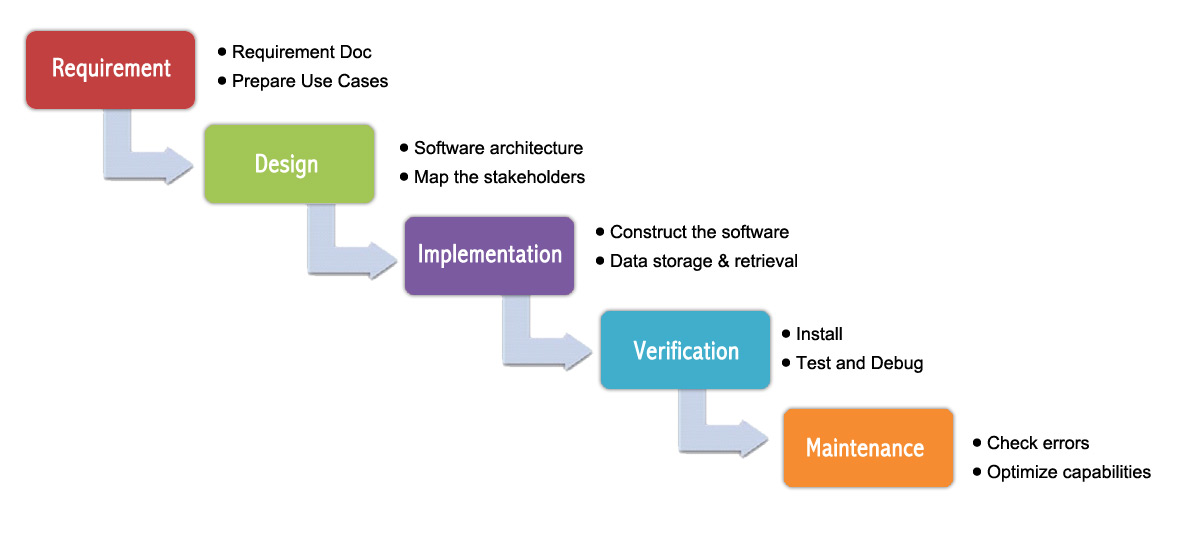
* This project is 4-months long due to the FPT University Capstone Project timeline, which can be consider a short project.
* Based on researches and current clarified face recognition system, the requirements of this project are stable, clear, fixed and well-understood by all team members.
* The Modified Waterfall Model involves verification and validation between the phases, so any deviations can be corrected immediately, providing the customer satisfaction, so this is preferred.

Figure 1: Waterfall methodology

### Roles and Responsibility

|  |  |  |  |
| --- | --- | --- | --- |
| No | Fullname | Role | Responsibilities |
| 1 | Nguyễn Đức Lợi | Supervisor, Project Manager | * Specify user requirement * Advisor for ideas and solutions * Give out techniques and business analysis support |
| 2 | Hoàng Phi Long | Team leader, developer, tester | * Managing process * Dividing tasks for team member * Create test plan * Clarifying requirements * Coding * Testing * Verify document * Managing budget * Database design |
| 3 | Nguyễn Đình Phong | Team member, developer, tester | * Create test plan * Database design * Clarifying requirements * Prepare document * Coding * Testing * GUI Design |
| 4 | Trịnh Bình | Team member, developer, tester | * Create test plan * GUI Design * Database design * Clarifying requirements * Prepare document * Coding * Testing |

Table 4: Roles and responsibilities

### Tools and Techniques

|  |  |
| --- | --- |
| Tools | |
| Developing tools | Visual Studio Code  Arduino IDE |
| Database system management | MongoDB |
| Source Control | Git-scm and Github |
| Models and Diagrams tool | Draw.io |
| Techniques | |
| Embedded System | C/C++ , Arduino SDK |
| API Web Server System | ExpressJS & NodeJS |
| Mobile Application | React Native, Javascript |

Table 5: Tools and techniques

## Project Management Plan

### System Development Life-cycle

Below are all the major tasks that need to be performed sequentially during the development of the system.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Description | Deliverables | Resource needed | Dependencies and Constraints | Risks |
| Requirement Analysis | - Identify and clarify main functions.  - Prepare task plan.  - Research mechanics of collecting clothes system  - Research solar energy circuit | - Report No. 1 Introduction.  - Project Management Plan  - Task sheet  - Prototypes | 14 man-days | N/A | - Missing requirement.  - Unclear project’s scope.  - Lack of member share of understand. |
| Design | - Identify hardware and software requirements.  - Decide software architecture.  - GUI design using top-down break down.  - Design database. | - Report No. 2 Software Project Management Plan.  - Report No. 3 Software Requirement Specification.  - Report No. 4 Software Design Description. | 20 man-days | Depend on “Requirement Analysis”. | - Misunderstood or unclear system’s requirement.  - Lack of practical experience leading to unreasonable design. |
| Implementation | - Collect temperature, humidity datasets.  - Build hardware system  - Implement embedded software system  - Implement Android GUI.  - Build REST API | - Demonstration application.  - Report No.5 System Implementation & Test. | 50 man-days | Depend on “Design”. | - Lack of practical experience and knowledge.  - Human mistake.  - Broken hardwares due to wrong implementation  - Interference signal while ESP8266 communicate with Http Protocal |
| Testing | - Prepare test plan and test case.  - Test all functions and results. | Report No.5 System Implementation & Test. | 20 man-days | Depend on “Implementation”. | - Lack of experience.  - Not enough time for performing test.  - Missing bugs.  - Human resource. |
| Maintenance | - Deploy the system.  - Create the user’s manuals. | Report No.6 Software User’s Manual. | 10 man-days | Depend on “Testing”. | - Lack of experience and knowledge.  - Human mistake.  - User’s manual may be difficult for user to understand and confuse. |

Table 6: Project task planning

### Plan Detail

#### Phrase 1: Requirement Analysis

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Research mechanics of collecting clothes system | - Research on current systems, their strengths and weakness. | Hoàng Phi Long  Nguyễn Đình Phong |
| 1. Research solar energy | - Research on current systems, their strengths and weakness.  - Research how to convert solar to electricity and charge into batter | Nguyễn Đình Phong  Trịnh Bình |
| 3. Identify and clarify main functions | Define main and needed functions the system must include. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |
| 4. Create system introduction | Complete Introduction Report. | Hoàng Phi Long |
| 5. Software Project Management Plan | Prepare Project Management Plan. | Hoàng Phi Long |
| 6. Prototype | Build a prototype of system and mobile application. | Nguyễn Đình Phong  Trịnh Bình |
| 7. SRS | Create SRS document. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |

Table 7: Plain Detail - Requirement Analysis

#### Phrase 2: Design

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Identify hardware and software detail design | Find out the suitable hardware and software for the system, as well as its minimum and recommended requirements. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |
| 2. Decide software architecture | - Define the major software components and interfaces.  - Draw core flow diagram, use case diagram, prototype…  - Group meeting to review and modify. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |
| 3. Decide Android App GUI | - UX/UI Design for Android Application | Nguyễn Đình Phong  Trịnh Bình |
| 4. Design database | - Design database for the system. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |

Table 8: Plain Detail - Design

#### Phrase 3: Implementation

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Collect temperature, humidity datasets | Program a small embedded program to collect data from sensors | Nguyễn Đình Phong  Trịnh Bình |
| 2. Construct hardware system | Build system from hardware components  Draw and print PCB board | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |
| 3. Implement embedded software system | Develop embedded program to control the system. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |
| 4. Implement Android GUI | Using React Native and Expo to implement Android Application GUI with fake datas | Hoàng Phi Long  Nguyễn Đình Phong |
| 5. Build REST API | Using NodeJS & ExpressJS building REST API for Mobile app and the system | Hoàng Phi Long  Trịnh Bình |

Table 9: Plain Detail – Implementation

#### Phrase 4: Testing

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Integration testing | Write test case and testing system. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |
| 2. Alpha testing | Do alpha test with customer. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |

Table 10: Plain Detail –Testing

#### Phrase 5: Maintenance

|  |  |  |
| --- | --- | --- |
| Task | Description | Author |
| 1. Installation guide | Write installation guide. | Hoàng Phi Long |
| 2. User Manual | Write user manual. | Hoàng Phi Long  Nguyễn Đình Phong  Trịnh Bình |

Table 11: Plain Detail –Maintenance

# Software & Hardware Requirement Specification

## User Requirement Specification

User is a person who use our device and mobile application. These are functions that user can use:

* Login to mobile application
* Control system to collecting or drying clothes by RF Remote control
* Control system to collecting or drying clothes by button on hardware
* Control system to collecting or drying clothes by android application
* Check information of the system
* Setup and control dryer to dry their clothes when there is a rain
* Manage/edit contact or account information (Name, Address, Mobile phone, Username, Password, …)

## System Requirement Specification

### External Interface Requirement

#### User Interface

The user interface uses English language for mobile application, hardware display interface. General requirement for graphics user interface should be simple, clear, intuitive, and reminiscent. The User interface should design with the following rules:

* User interface is created by using model top-down, left-right design.
* The interface design is an iterate process includes: design, sketching, prototyping, user assessment.
* Some design principles will be taken into consideration:
  + How To Design A Great User Interface – WDD Staff

#### Hardware Interface

Server:

* RAM: 512MB
* CPU: Intel Xeon X5550 @ 2.67GHz
* Disk Storage:
  + Operating System: Minimum 512MB (depends on Operating system)
  + Runtime Environment: 55MB
  + Application server: 60MB
  + Total: 615 MB

Android Phone:

* RAM: Minimum 512MB
* Operating System: Android 4.4 or later
* Network connection: Wi-Fi 802.11 a/b/g/n/ac, 3G, 4G/LTE
* Disk Storage: Minimum 16MB

### System Overview Usecase

#### Hardware System Usecase

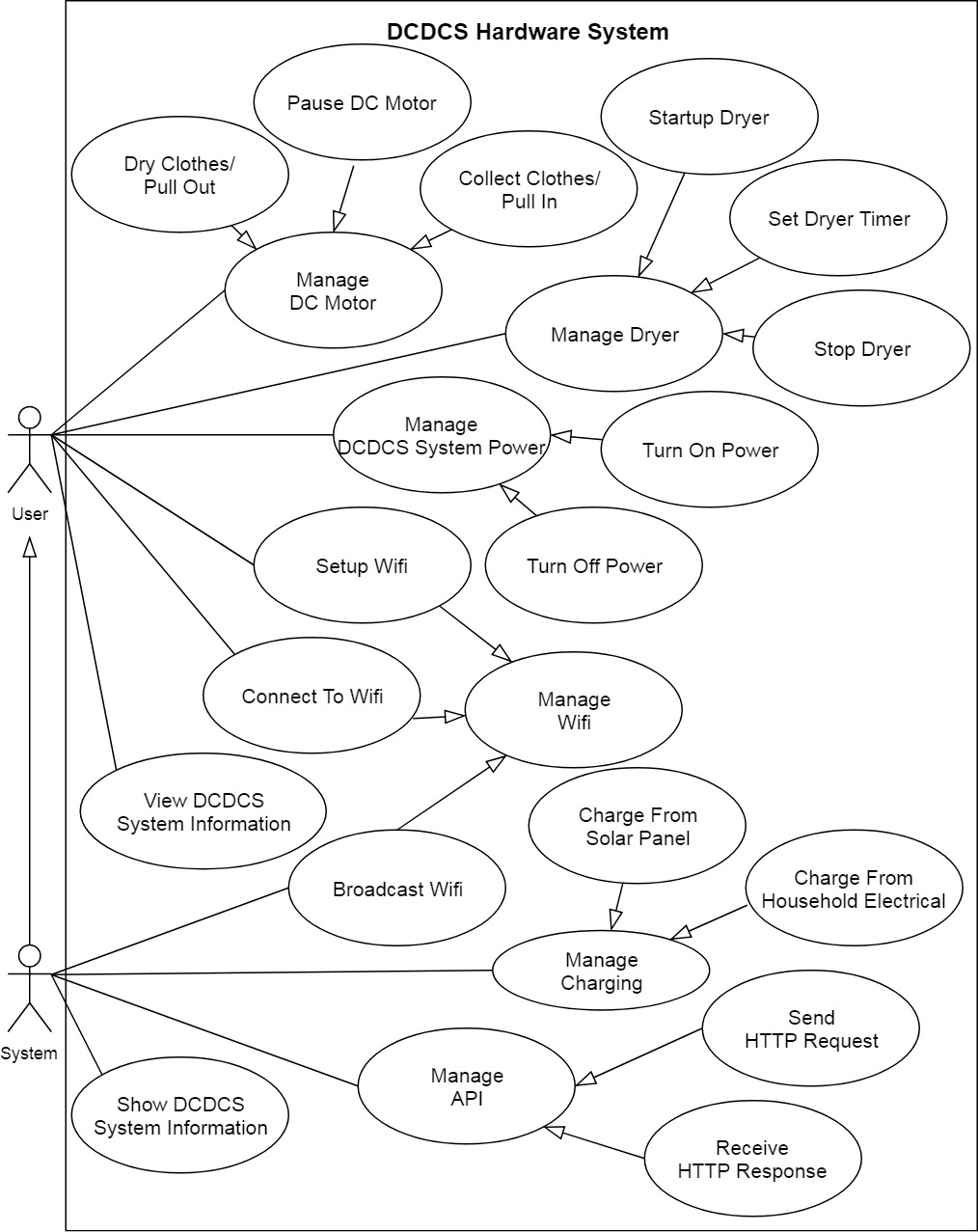


Figure 2: Hardware system overview usecase diagram

#### Android Application Usecase

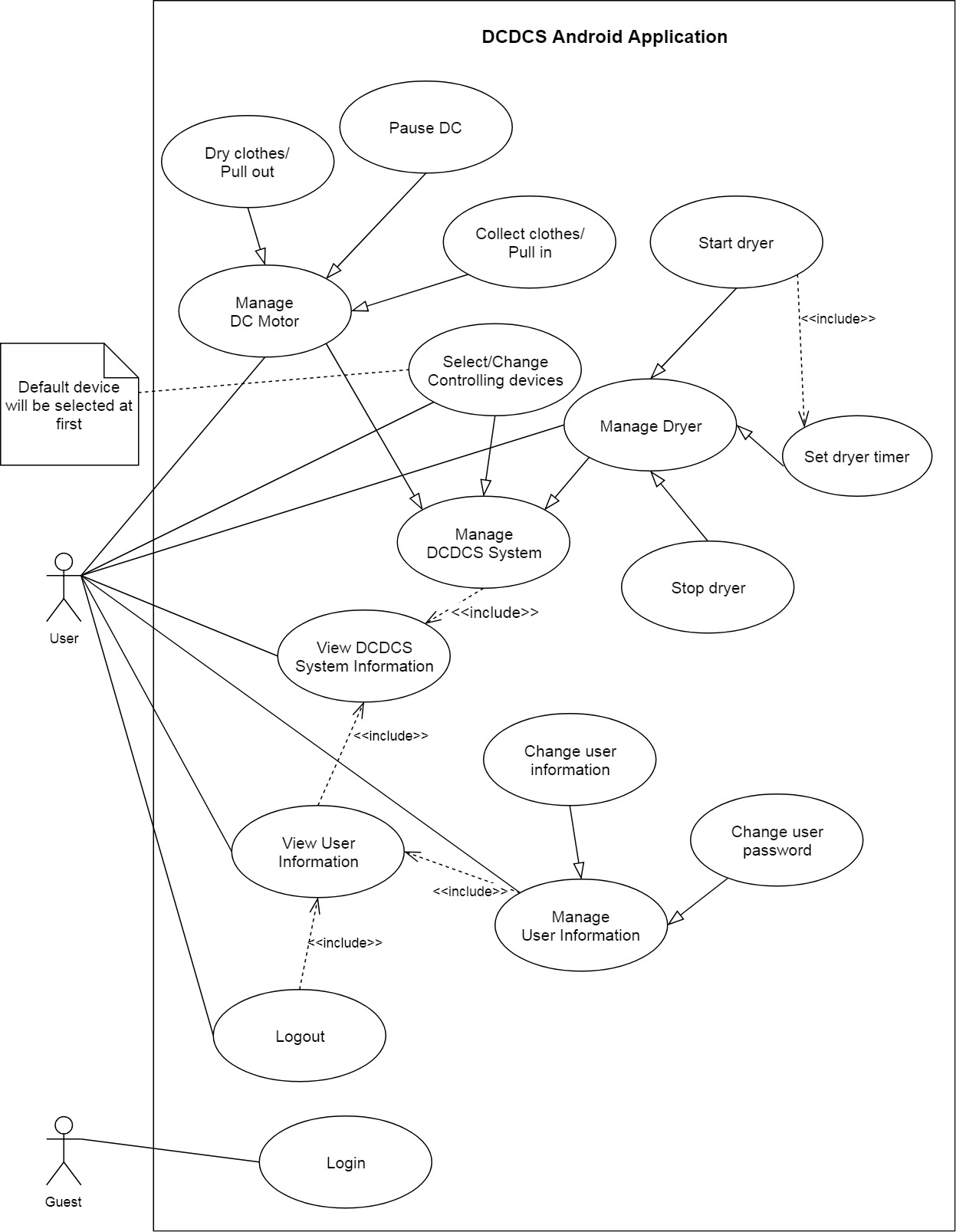


Figure 3: Android application overview usecase diagram

## Hardware Requirement Specification

### Hardware Interface

The hardware interface must have satisfied the following requirements:

* Easy to replace
* Low-cost module
* Easy to implement

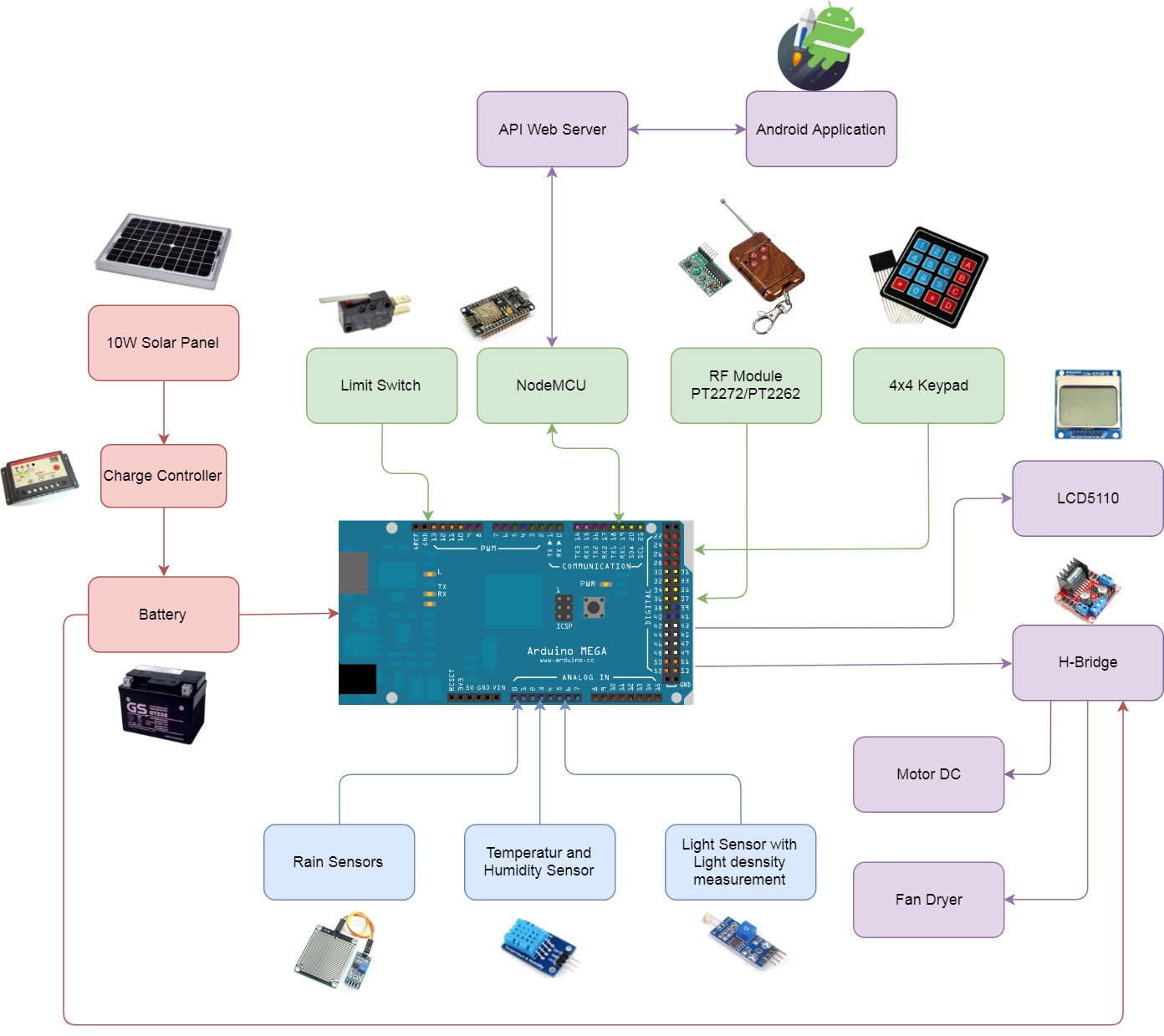
Based on project requirement we have choose following hardware components

Figure 4: System block diagram

## D:\Capstone Document\Diagrams\Conceptual Diagram.jpgConceptual Diagram

Figure 5: Conceptual diagram

**Data Dictionary**

|  |  |
| --- | --- |
| Entity Name | Description |
| Customer | Contains information of customer who brought our product |
| Product | Contains information about product |
| Model | Contains information about product’s model |
| User | Contains information about account of the system |
| Message | A message queue, contains a message to communicate with hardware system |

Table 12: Data dictionary for conceptual diagram

# Software & Hardware Design Description

## Design Overview

This document describes the technical and user interface design of DCDCS System. 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 cases.

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

* Section 1: Introduction
* 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 a fully attributed ERD
* Section 6: Describe algorithms

## D:\Capstone Document\Diagrams\System Architecture Diagram.jpgSystem Architectural Design

Figure 6: System overview architecture

### 2.1 API Web Server Architectural DesignD:\Capstone Document\Diagrams\Web API Architecture Diagram.jpg

Figure 7: API Web server architecture

In API development, the system is developed under MVC architecture style. We choose this architecture for API because of following advantages:

* With MVC architecture, we can separate business code with Controller and View, so we can use the business code in API web server without repeat the code.
* It can eliminate the creation of the singleton and factory classes and well defined interface to business layer
* By separating concerns into 3 distinct pieces, we can perform unit testing easily. Our Presentation layer can be tested free of the Model or Controller, and vice-a-versa
* It supports all aspects of application development, business aspects, persistence aspects, etc., so we can develop a complete application.

This project follows MVC architecture with following components:

* Controller: is the parts of the application that acts like event handler to handles user interaction. Typically, controller reads data from a request and calls appropriate business’s method then selects view to return to user.
* View: The view renders the contents of a model. It gets data from the model and specifies how that data should be presented. It updates data presentation when the model changes. A view also forwards user input to a controller. Depending on the task being performed by the user the model can be looked at from different perspectives.
* Model: Represents the business data and any business logic that govern access to and modification of the data. The model notifies views when it changes and lets the view query the model about its state. It also lets the controller access application functionality encapsulated by the model. Typically, when a change in the model is to be reflected from user, it should be reflected in all the model’s views.

### Android Application Architectural DesignD:\Capstone Document\Diagrams\Android Internal System Architecture.jpg

Figure 8: Android application internal architecture

In Android application, the system is developed under Flux architecture. We choose this architecture for Android Application because of following advantages:

* Flux is all about controlling the flow inside the app — and making it as simple to understand as possible.
* Easy to implement and understand. Hence it makes source code easier to maintain and reduce time to develop application
* Having supported library (Redux)
* Suitable for React Native codebase

Android Application follows Flux architecture with following components:

* **Actions:** Helpers that pass data to the Dispatcher. Are simple objects with a type property and some data. For example, an action could be:

{“type”: “IncreaseCount”, “payload”: {“delta”: 1}}

* **Dispatcher:** Receives these Actions and broadcast payloads to registered callbacks. Acts as a central hub. The dispatcher processes actions (for example, user interactions) and invokes callbacks that the stores have registered with it. The dispatcher isn’t the same as controllers in the MVC pattern — usually the dispatcher does not have much logic inside it and you can reuse the same dispatcher across projects
* **Stores**: Contain the application’s state and logic. The best abstraction is to think of stores as managing a particular domain of the application. They aren’t the same as models in MVC since models usually try to model single objects, while stores in Flux can store anything. The real work in the application is done in the Stores. The Stores registered to listen in on the actions of the Dispatcher will do accordingly and update the Views.
* **Views**: are **controller-views**, also very common in most GUI MVC patterns. They listen for changes from the stores and re-render themselves appropriately. Views can also add new actions to the dispatcher, for example, on user interactions. The view are usually coded in React, but it’s not necessary to use React with Flux.

### D:\Capstone Document\Diagrams\Hardware System Architecture.jpgHardware System Architecture

Figure 9: Hardware system architecture

In Embedded Hardware control application, the system is developed under Internet of Things architecture style. We choose this architecture for Embedded Hardware control application because of following advantages:

* Highly scalable and available out of the box due to the nature of each selected component.
* Minimal knowledge required to start.
* It’s scalable and fault tolerant by design.
* Reduces the development and deployment costs and timeframes

The system follows IoT architecture with following components:

* **Sensors and Actuators:** this part measures a physical quantity such as sound, temperature, moisture etc. and converts it into electrical quantity to make the system understand and act accordingly
* **Connectivity (NodeMCU):** The received signals are to be uploaded on the network using different communication medium such as Wi-Fi, Bluetooth or BLE, LoPAN etc.
* **People and Processes:** Networked inputs are then combined into bidirectional system that integrate data, people and processes for better decision making.

## D:\Capstone Document\Diagrams\Component Diagram.jpgComponent Diagram

Figure 10: Component diagram

**COMPONENT DIAGRAM DICTIONARY: DESCRIBE COMPONENTS**

|  |  |
| --- | --- |
| Component Name | Description |
| RF Component | Component to handle RF Remote |
| Rain Sensor Component | Component to handle Rain sensor |
| Keypad Component | Component to handle Keypad |
| NodeMCU Component | Component to handle Wifi, API Request/Response |
| Processing Component | Component to control the system |
| Light Sensor Component | Component to handle Light sensor |
| Dryer Component | Component to handle dryer |
| Display Component | Component to display system’s information |
| DC Motor Component | Component to handle DC motor |
| API Handler | Component to handle API Request/Response on Android |
| Controllers | A group of components that help control android app |
| (View) Login | Login screen |
| (View) Home | Home screen |
| (View) User Profile | User profile screen |
| System Database | Component to handle with database |
| Mongoose | Component to handle request/response and mapping document to Javascript object |
| Controllers | A group of components that help handling API request |
| Express Web Server | A component help build a API server |

Table 13: Component diagram dictionary

## Detailed Description

### Class Diagram

#### D:\Download\Chrome\API Web Server Class Diagram Part 1.jpgAPI Web Server

Figure 11: API Web Server Class Diagram Part 1

Figure 12: API Web Server Class Diagram Part 1

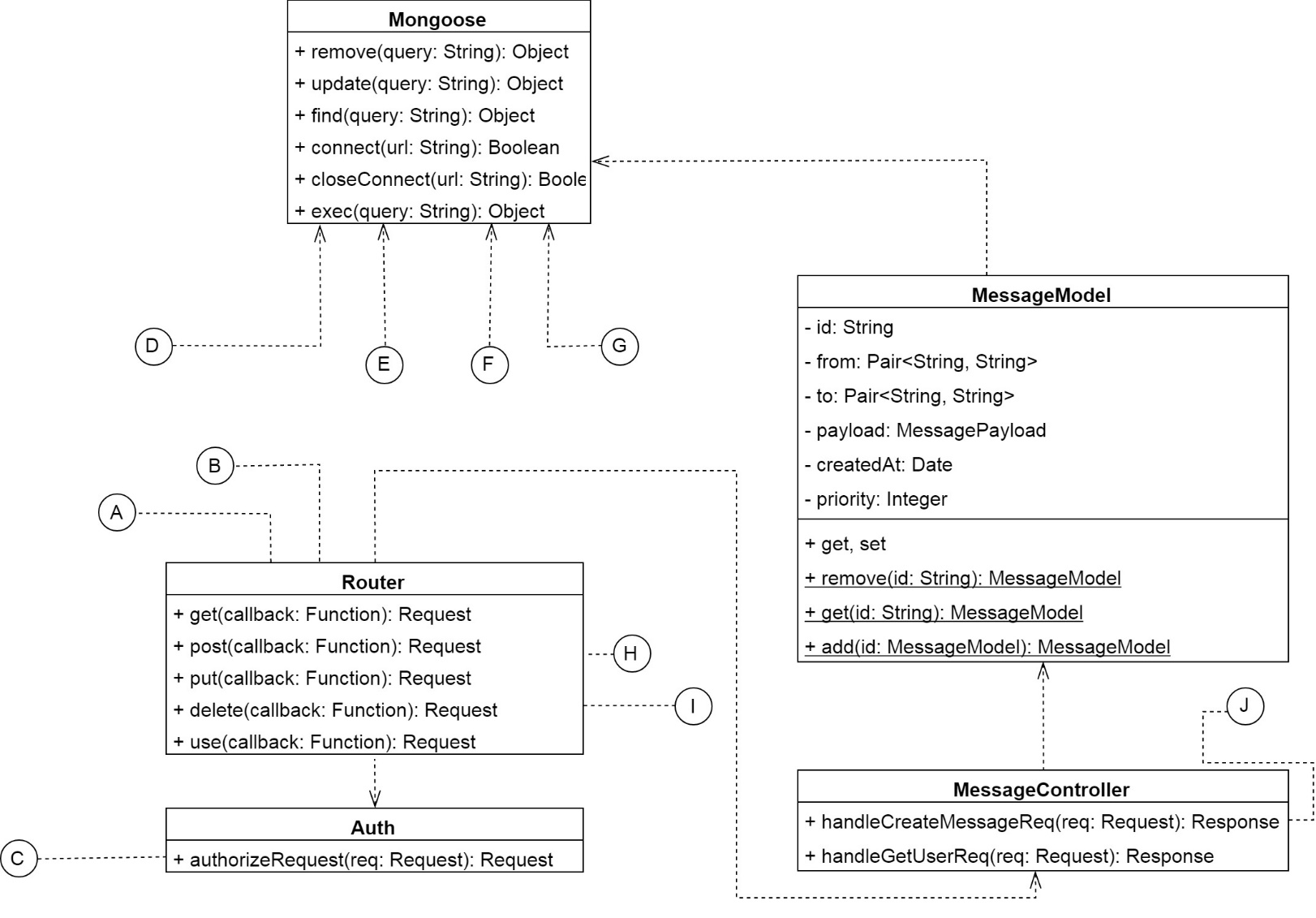


Figure 13: API Web Server Class Diagram Part 2

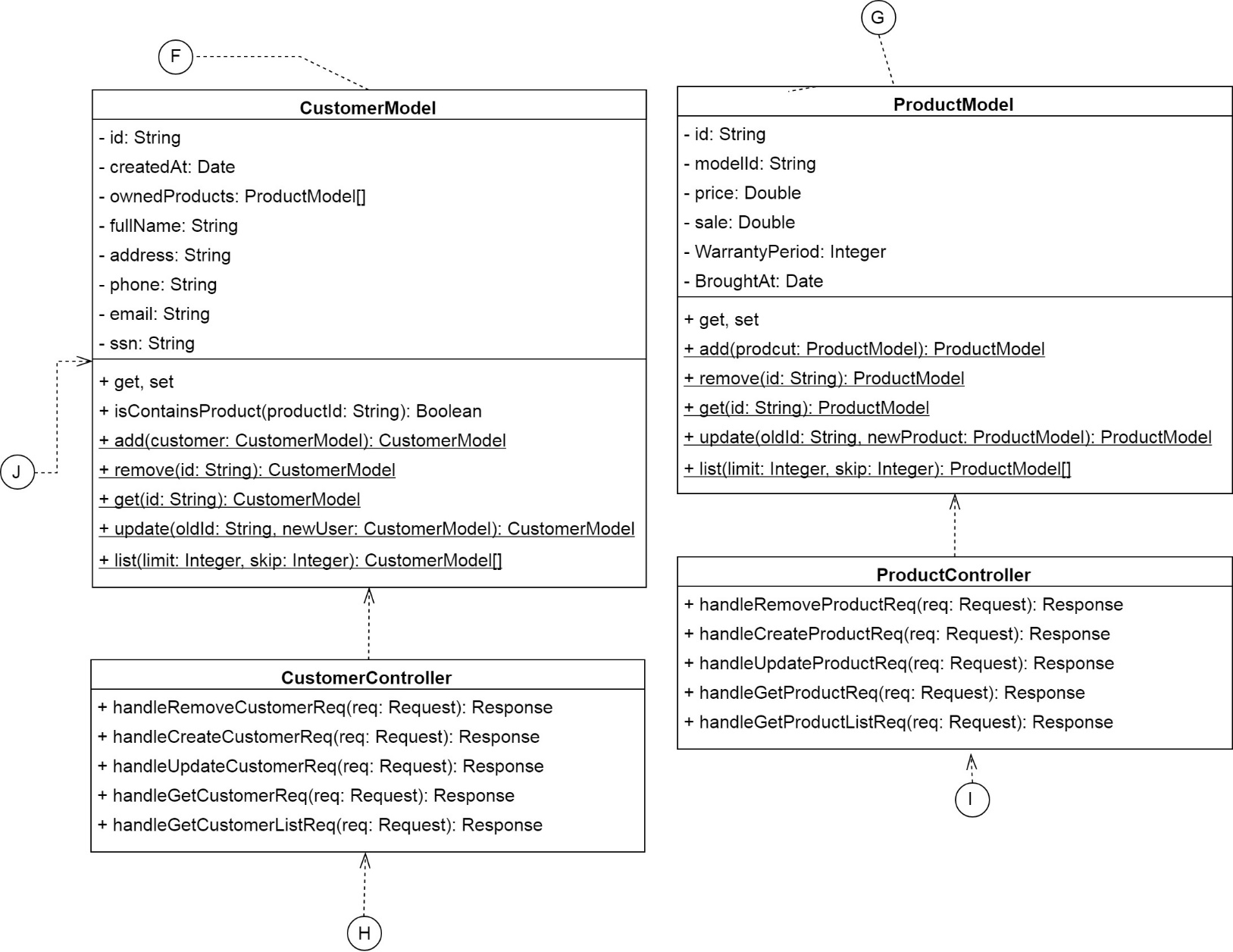


Figure 14: API Web Server Class Diagram Part 3

|  |  |  |
| --- | --- | --- |
| Class Name | Mapped Column on Conceptual Diagram | Description |
| CustomerModel | Customer | Contains customer information |
| ProductModel | Product | Contains product information |
| UserModel | User | Contains user account information |
| ModelModel | Model | Contains model of product information |
| MessageModel | Message | Contains message which used to communicate with hardware system |
| CustomerController | N/A | This class has functions that will handle any request about customer |
| ProductController | N/A | Contains functions that will handle any request about product |
| UserController | N/A | A class with functions that will handle any request about user, login, change password, etc. |
| ModelController | N/A | Contains functions that will handle any request about product model |
| MessageController | N/A | A class has functions that will allow user to publish and get action message |
| Auth | N/A | Authorize request based on access token |
| Router | N/A | A class that listen to request so that the server can call the correct controller |
| Mongoose | N/A | A class help connect and communicate, handle request/response from MongoDB |

Table 14: API Web server class diagram dictionary

#### D:\Capstone Document\Diagrams\System Controller Class Diagram Part 1.jpgHardware System

Figure 15: Hardware system controller class diagram part 1

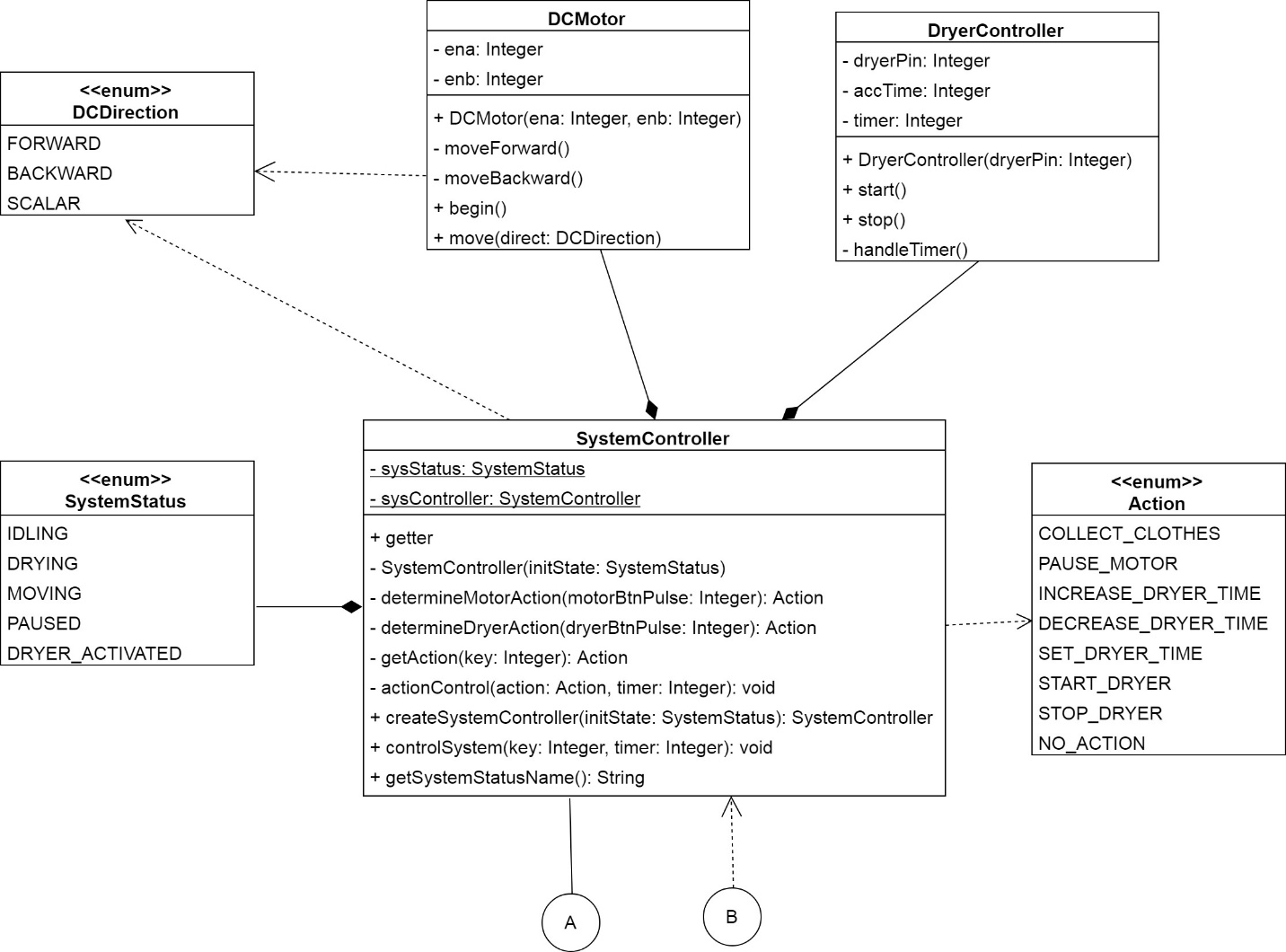


Figure 16: Hardware system controller class diagram part 2

|  |  |
| --- | --- |
| Class Name | Description |
| CentralController | The class that receive data from another class and tell SystemController class to control the system correctly |
| SystemController | This class will determine and control system with given action key |
| SwitchHandler | Handler class for limit switch |
| RFHandler | Handler class for limit switch |
| KeypadHandler | Handler class for 4x4 matrix keypad |
| LightSensorHandler | Handler class for light sensor to read light density and determine it is night or day |
| RainSensorHandler | Handler class for rain sensor. |
| WifiHandler | Handle event from NodeMCU that send through I2C Protocol |
| LCDHandler | A class that help print to LCD more easier |
| Wire | External library that help communicate with another device via I2C Protocol |
| DHT | An external library that help reading data from DHT Module |
| BHT1750 | An external library that help reading data from Light Sensor Module |
| DCMotor | This class help controlling dc motor to collect or dry clothes |
| DryerController | This class help controlling dryer fan |
| Action | This is an enum that descriptions the control action of the system |
| SystemStatus | This is an enum that descriptions the status of the system |
| DCDirection | This is an enum that descriptions the status of the dc motor |

Table 15: Hardware controller class diagram dictionary

### Interaction Diagram

#### Sequence Diagrams

##### Control system from android application

**Summary:** This diagrams show how android application and hardware system can communicate with each other. [ACTION] can be DRY\_CLOTHES, COLLECT\_CLOTHES, START\_DRYER, STOP\_DRYER

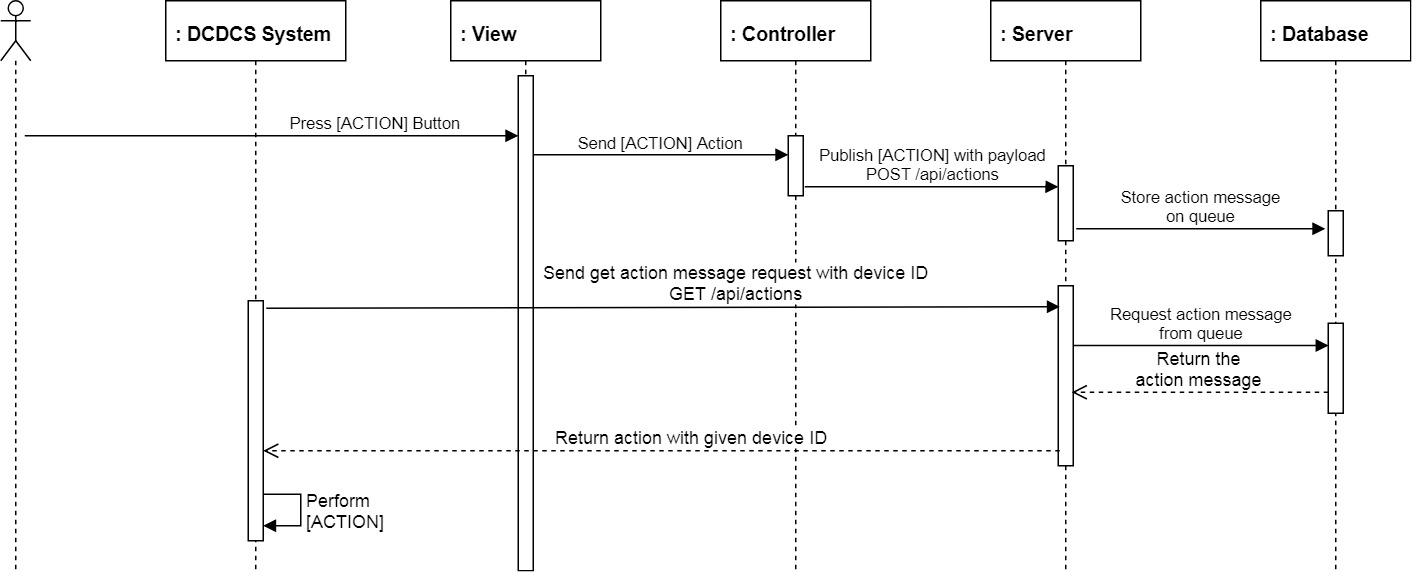


Figure 17: Control system with android app sequence diagram

##### Update system information

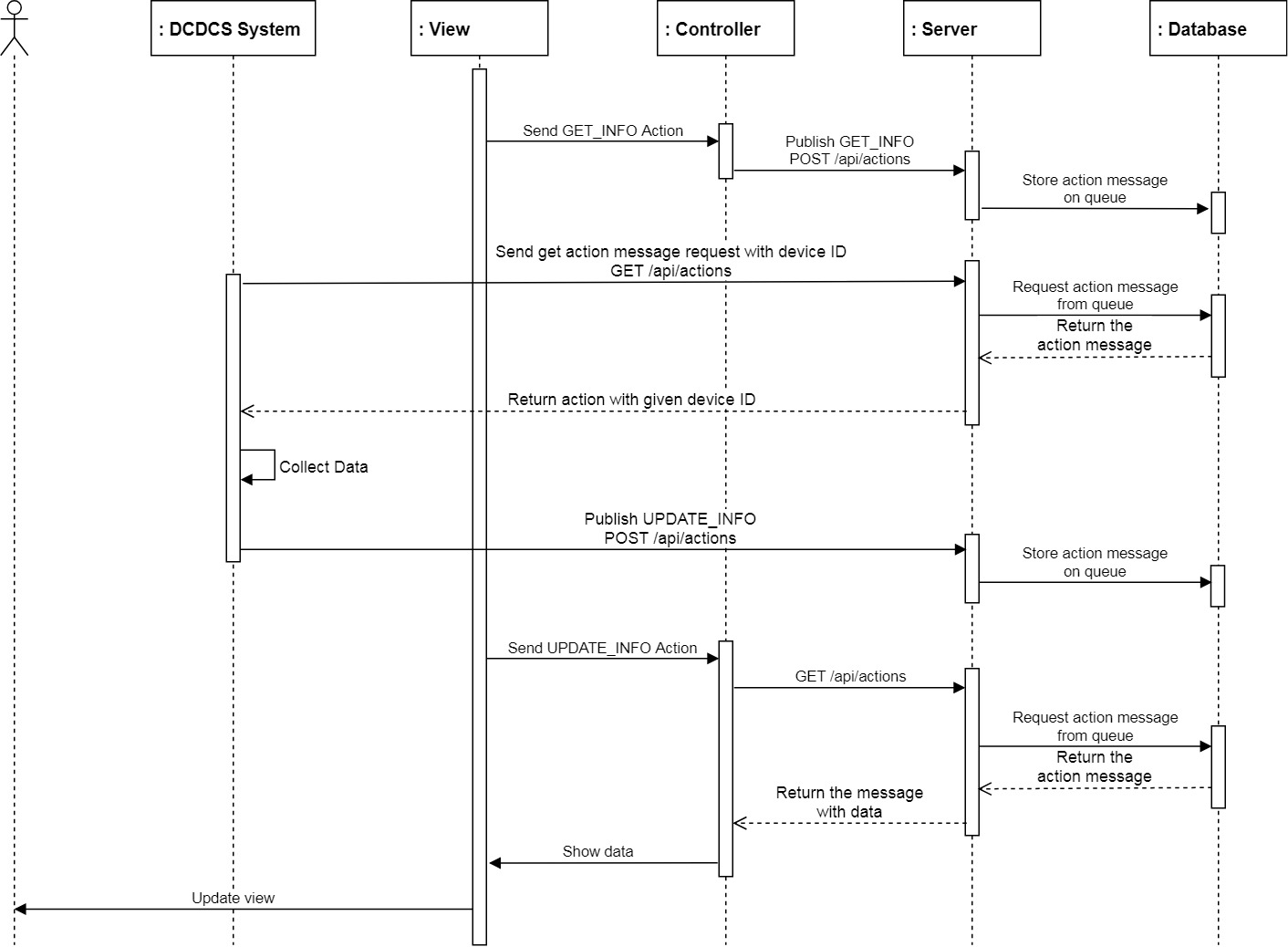
**Summary:** This diagrams show how android application gathers information from hardware system

Figure 18: Update system information sequence diagram

#### Activity Diagrams

##### Control DC

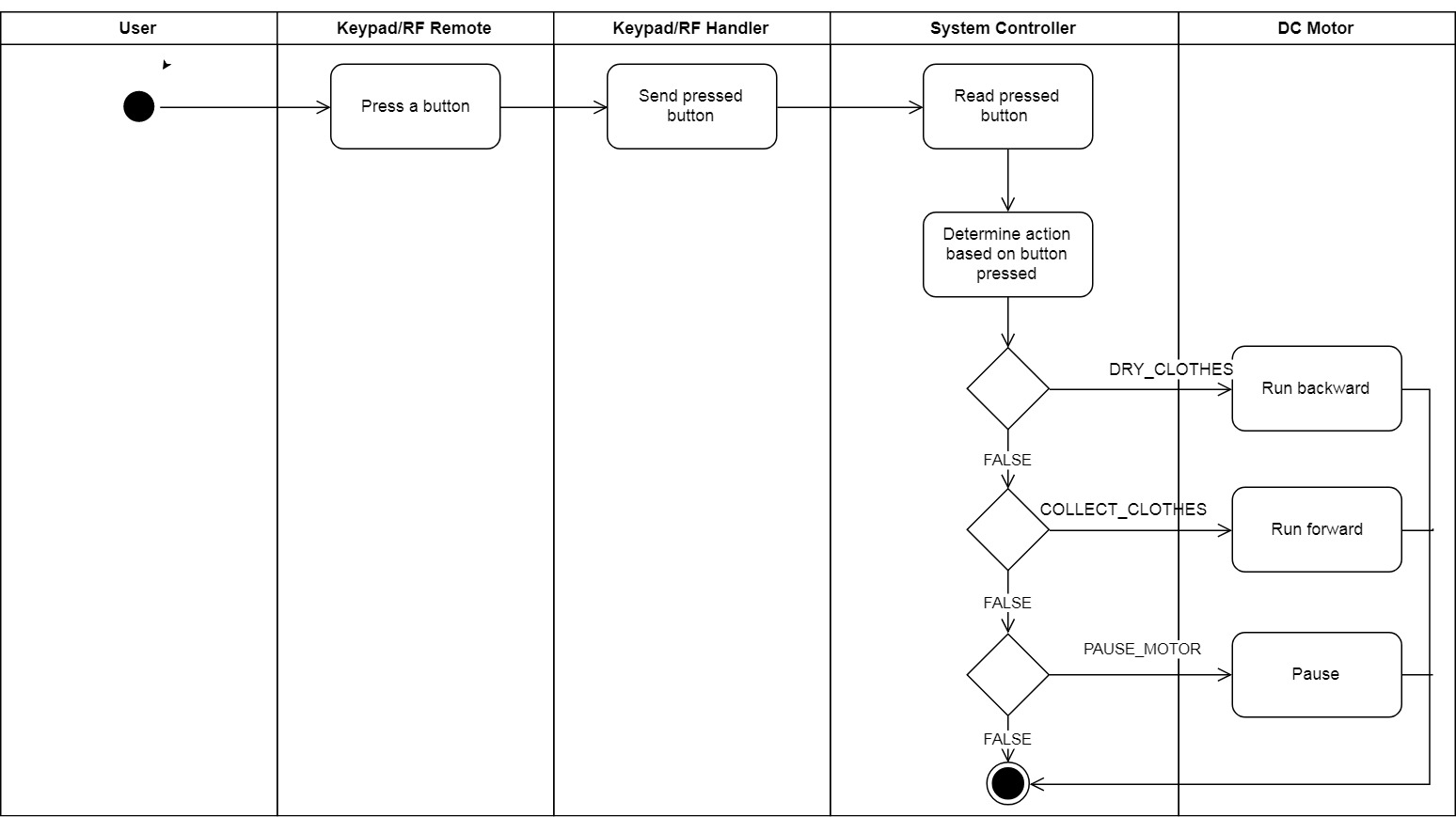
**Summary**: This diagrams show how user can control the DC

Figure 19: Control DC activity diagarm

##### Control Dryer

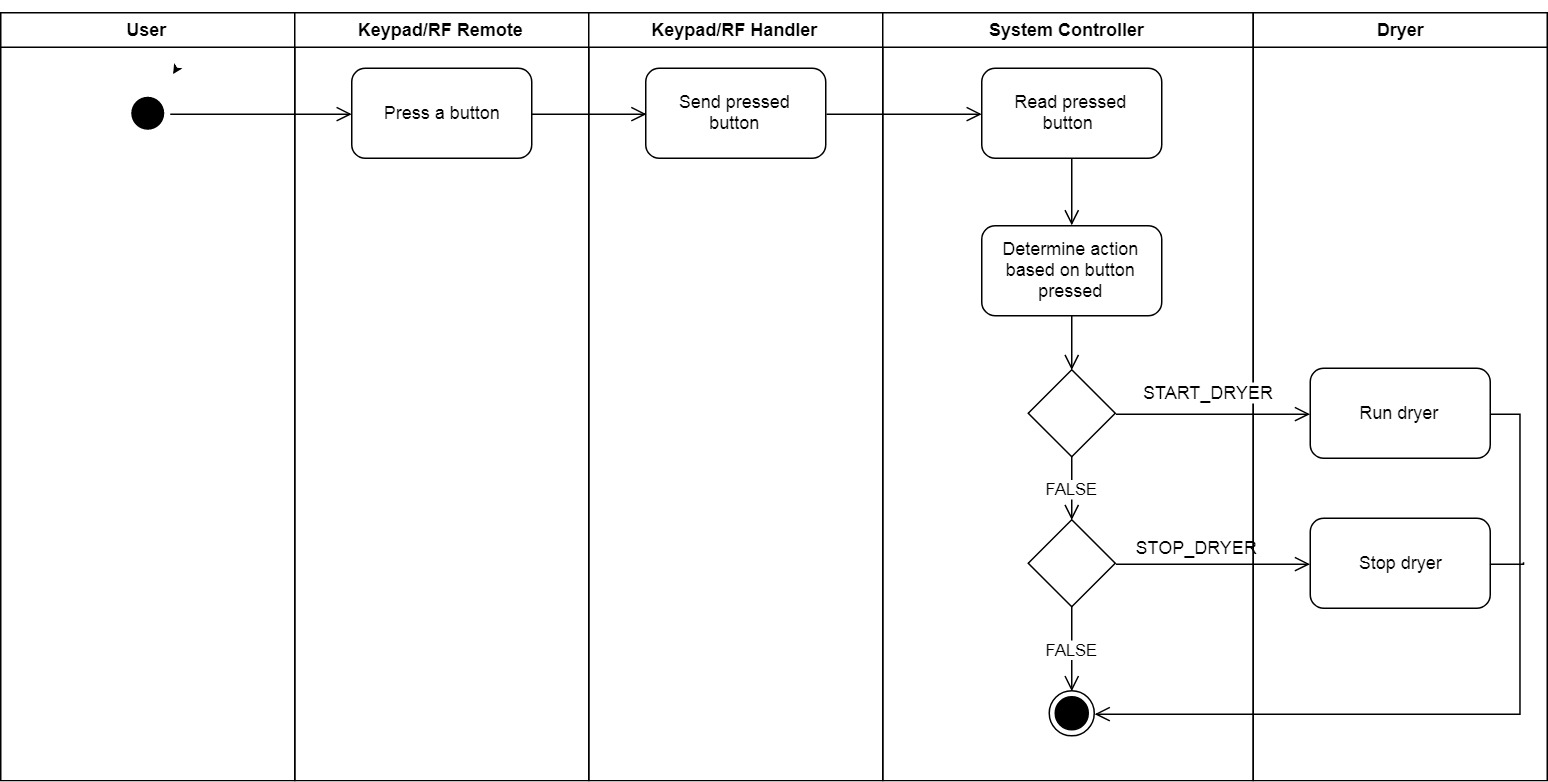
**Summary:** This diagrams show how user can control the dryer

Figure 20: Control Dryer activity diagram

##### Auto control

**Summary:** This diagrams show how system itself control.

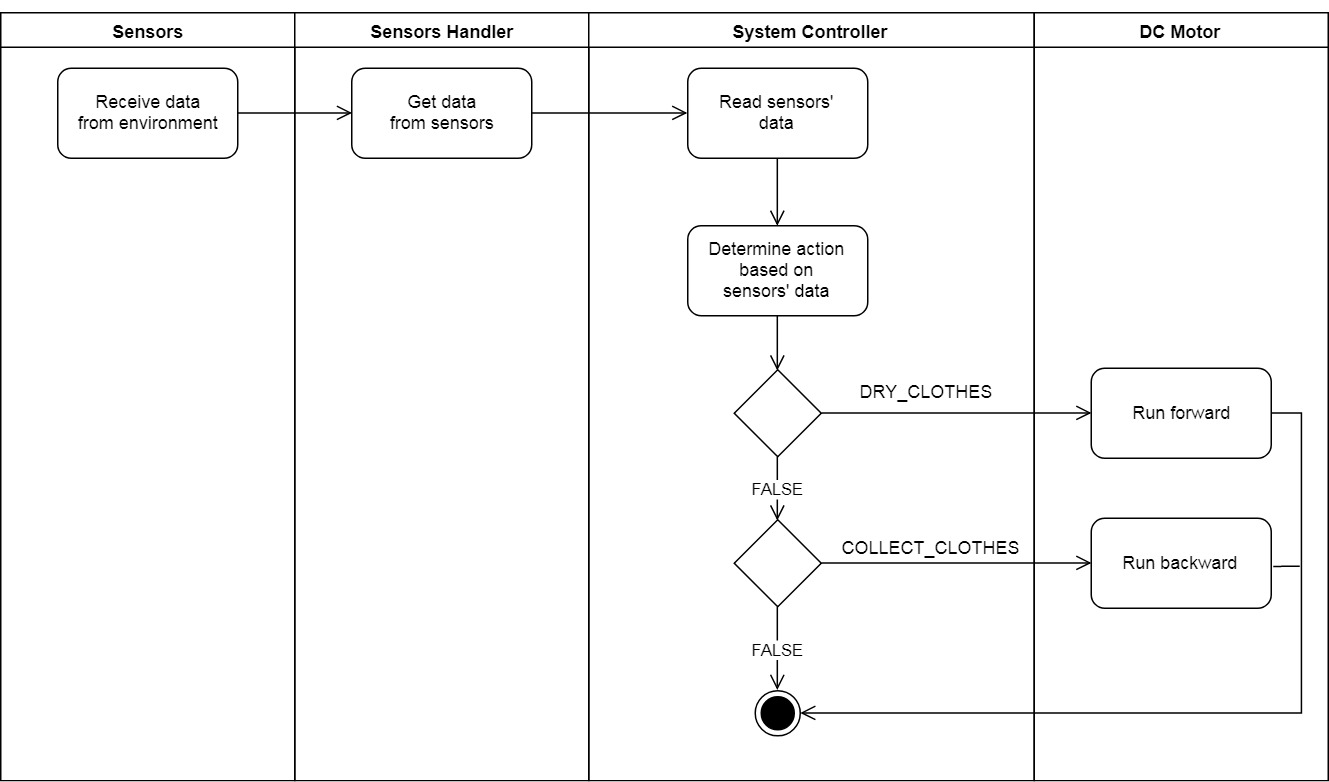


Figure 21: Auto control activity diagram

## Database Design

### Entity Relational Database (ERD)

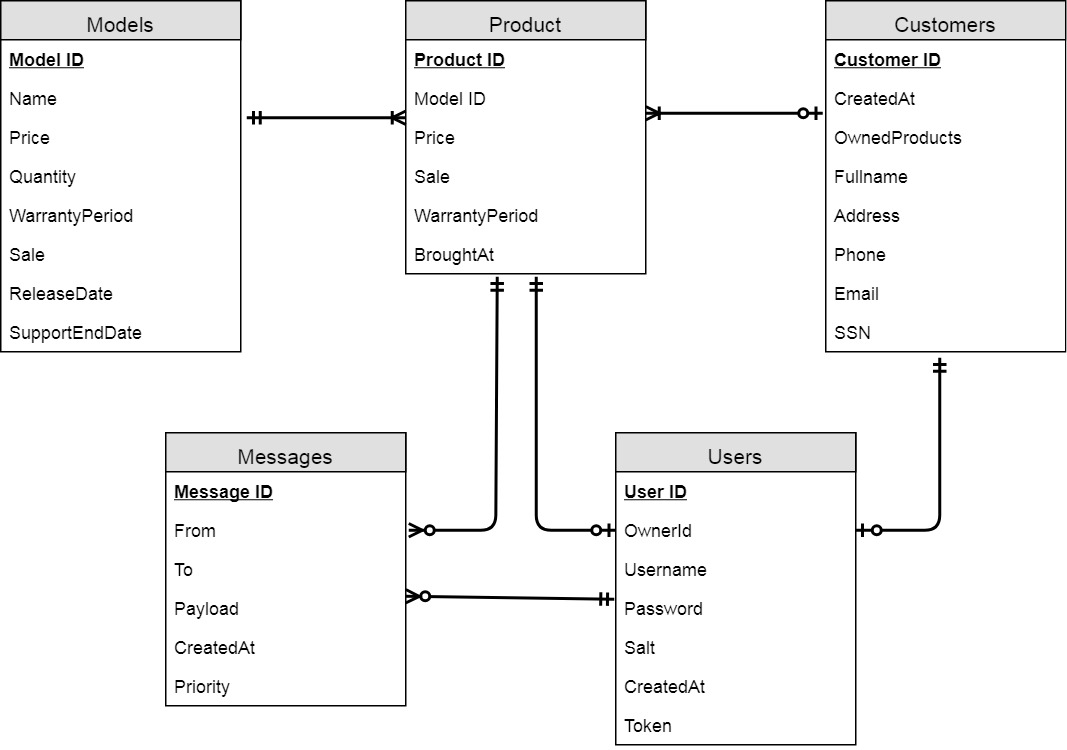


Figure 22: Entity Relational Database

### Data Dictionary

|  |  |
| --- | --- |
| Entity Name | Description |
| Customer | Contains information of customer who brought our product |
| Product | Contains information about product |
| Model | Contains information about product’s model |
| User | Contains information about account of the system |
| Messages | A message queue, contains a message to communicate with hardware system |

Table 16: Entity diagram data dictionary

## Algorithms

### System Control

#### Definition

System has many ways to control the system; i.e. RF Remote, Android application, hardware button. From these controllers, they can control many another devices like DC Motor to collect or dry clothes.

#### Define Problem

While using multiple controller at the same time. It causes a collision that leading to the system doesn’t work correctly.

#### Solution

We use one thread and blocking I/O to sequentially reading each controller. Therefore, when we’re handling a single controller. Another controller will be ignored.

#### Pros & Cons

* Pros:
  + No more collisions
  + Easy to control because the system now works on priority of the controller
  + Easy to extends when there are new controller
  + Memory reduced due to using only a single thread
* Cons:
  + An action takes longer time than user to complete (due to the priority)

#### Algorithm Complexity

* Time: O(n) with n is the number of controller
* Space: O(1) because we don’t use any additional spaces

#### Overview Flowchart

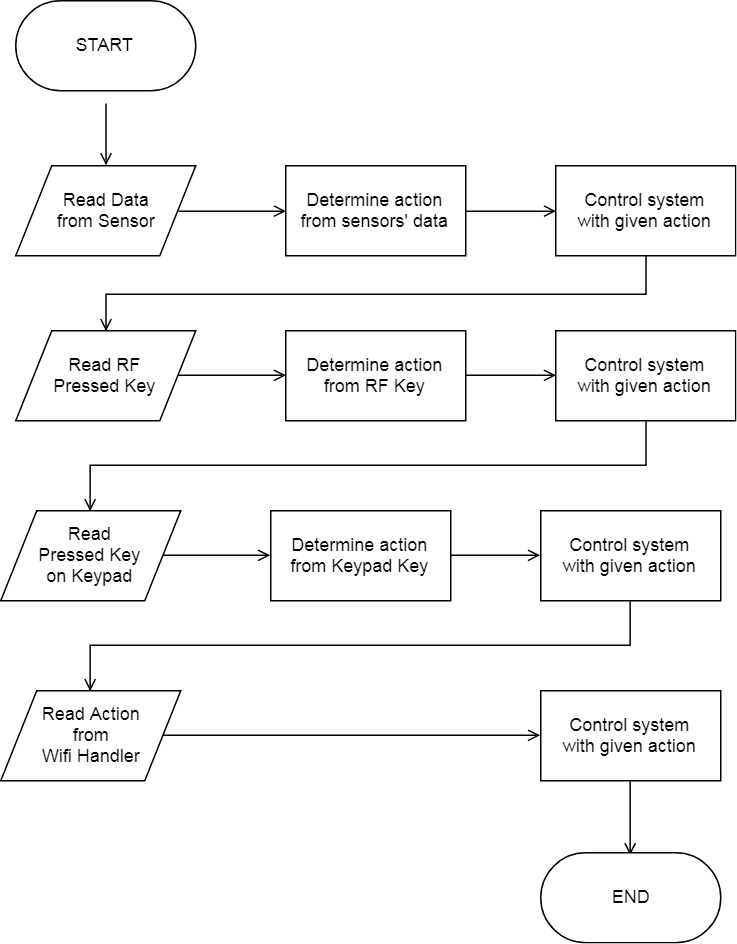


Figure 23: System Control overview flowchart

# Task Sheet

# Appendix

* Flux Architecture: <https://facebook.github.io/flux/>
* Bit Twiddling Hacks: <https://graphics.stanford.edu/~seander/bithacks.html>