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 – SE61968** **Trịnh Bình – SE61780**
Supervisor	**Nguyễn Đức Lợi**
Ext. Supervisor	**N/A**
Capstone Project code	**DCDCS**

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

Table of Contents

List of Tables

List of Figures

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

A. Introduction

1. 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

2. Introduction

An automatic clothes drying system, which uses rain sensor to detect rain and ESP8266 for communications between Android application and embedded device, was developed to allow consumers to effectively manage their chores. This document will explain the foundations and the processes of this innovative system.

Furthermore, this document will describe 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.

3. Current Situation

Vietnam is a tropical country with a long rainy season accounting for almost half of the year. Vietnamese have a habit and are also much more used to drying their clothes naturally under the sun rather than using the dryer. This habit always pose a problem of inefficient clothes drying process during the rainy months. That is, Vietnamese are constantly worried about not being able to collect their clothes which leaves the clothes potentially getting wet under the rains. Thus, it is necessary to develop a system that helps people to manage their laundry chores better. There has been a few solutions given such as the “Smart Clothesline Rigs”. However, this system is relatively expensive and ineffective. At 13.000.000 VND, the “Smart Clothesline Rigs” is a controllable system with UV light, build-in dryer and remote control (only works within 30 meters). Nevertheless, the system does not solve the core problem of allowing the automation of clothes collecting. Thus, the Sun Drying Wet Clothes

System, which integrates the automatic clothes collecting function, would better suit the needs of Vietnamese.

4. Problem Definition

Advantage of existing system on the market

- UV disinfection
- Built-in dryer
- Strong structure which can lift up to 25kg of clothes
- Drawbacks of existing system on the market
- High production costs which lead to relatively unaffordable selling prices
- Hard to extend
- Automation fully dependent on electricity
- Cannot automatically collecting clothes

5. Proposed Solution

Our proposed solution is to design and construct an automatic clothes drying system called DCDCS to solve missing feature of the current “Smart Clothesline Rigs”. Our system will allow the automation of laundry-colleting in the case of rains. Our system will also be competitively priced, easier installation, more compact and mobile, and extendable compare to the existing system.

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

5.1 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

5.2 Advantages and Disadvantages

- **Advantages:**
  - Low costs which allow more affordable prices
  - Fast rain detection
  - Can control using mobile app
  - Use solar energy and store extra energy as battery for use under adverse conditions
- **Disadvantages:**
  - Cannot detect whether the clothes is dry or not
  - Cannot detect whether rain is over or not

6. 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

7. 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

B. Software Project Management Plan

1. Problem Definition

1.1 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

1.2 Problem Abstract

Vietnamese have long working hours which means they spend time at evening and night to do their chores. The chores include washing and drying clothes. However, Vietnam also has long rainy season which indicate a persistent problem of inefficient clothes drying process.

1.3 Project Overview

1.3.1 Current Situation

Below are the problems encountered in the project:

- **Lack of robust statistic and mathematical knowledge:** Our system currently cannot detect when the clothes are dry or when the rain stop 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 unable to implement this function.
- **Lack of telecommunication knowledge:** While using ESP8266, we found that there would be interferences during transmission. Nonetheless, we were experiencing difficulties in detecting problems due to our lack of telecommunication knowledge.

1.3.2 The Proposed 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.

1.3.3 The Boundaries of the system

Our system provides these functions:

- Automatically control clothesline at nighttime and at raining periods
- 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

1.3.4 Future plans

- Implement Hidden Markov Models (HMM) for rain forecasting
- Implement the ability to detect when the rain stop 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

1.3.5 Development Environments

1.3.5.1 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

1.3.5.2 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

2. Project Organization

2.1 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 clothes drying 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.

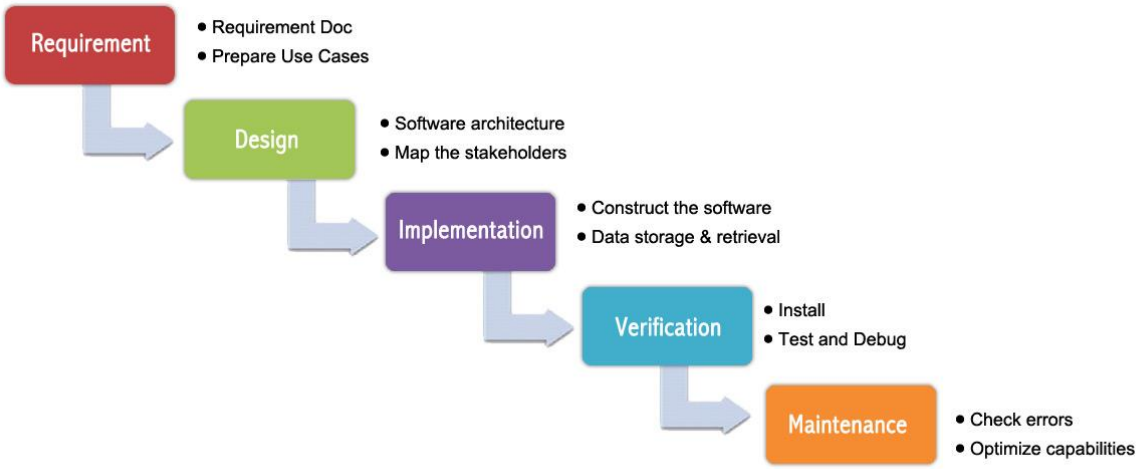

Figure 1: Waterfall methodology

2.2 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

2.3 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

3. Project Management Plan

3.1 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

3.2 Plan Detail

3.2.1 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

3.2.2 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

3.2.3 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

3.2.4 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

3.2.5 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

C. Software & Hardware Requirement Specification

1. 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, …)

2. System Requirement Specification

2.1 External Interface Requirement

2.1.1 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

2.1.2 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

2.2 System Overview Usecase

2.2.1 Hardware System Usecase

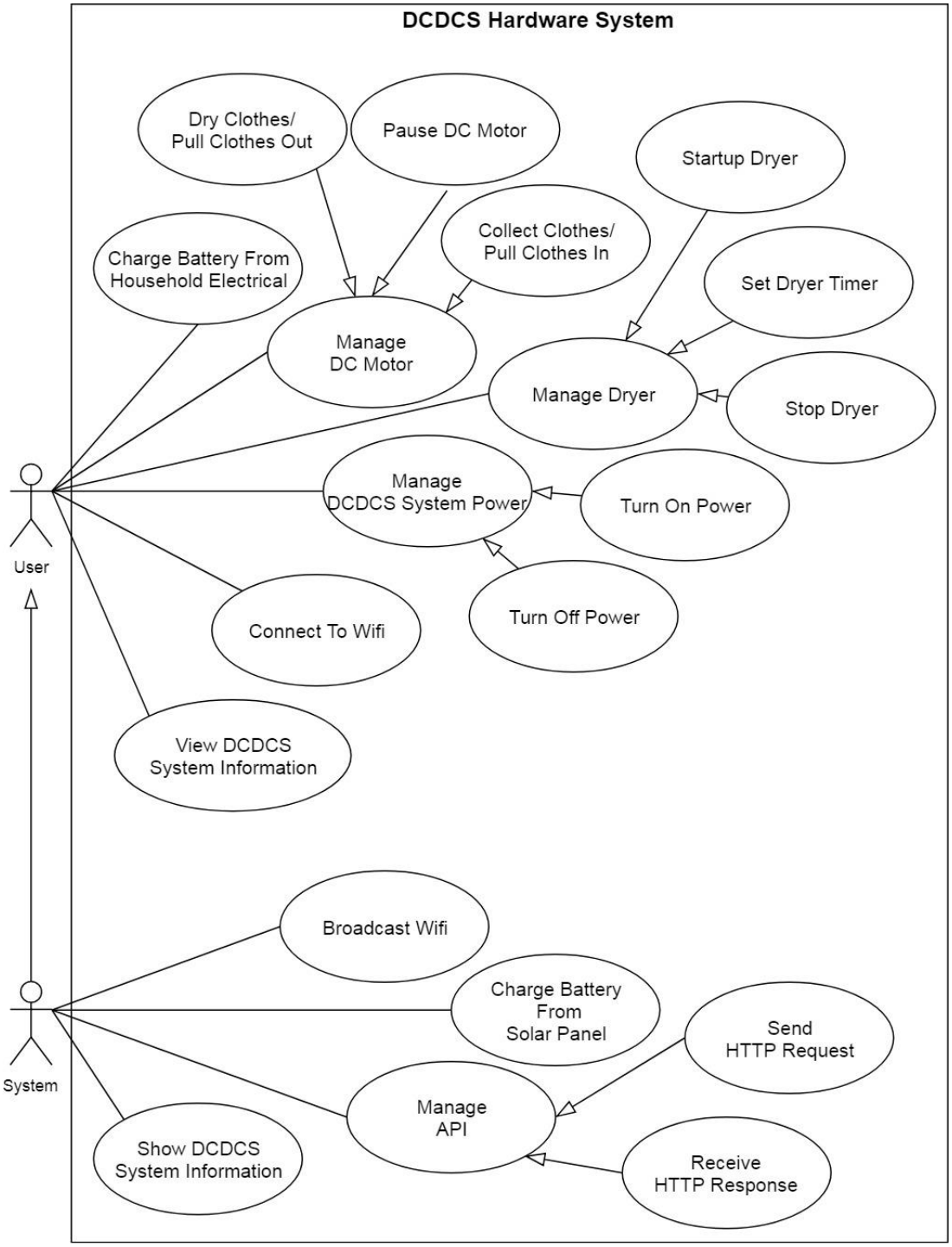

Figure 2: Hardware system overview usecase diagram

2.2.2 Android Application Usecase

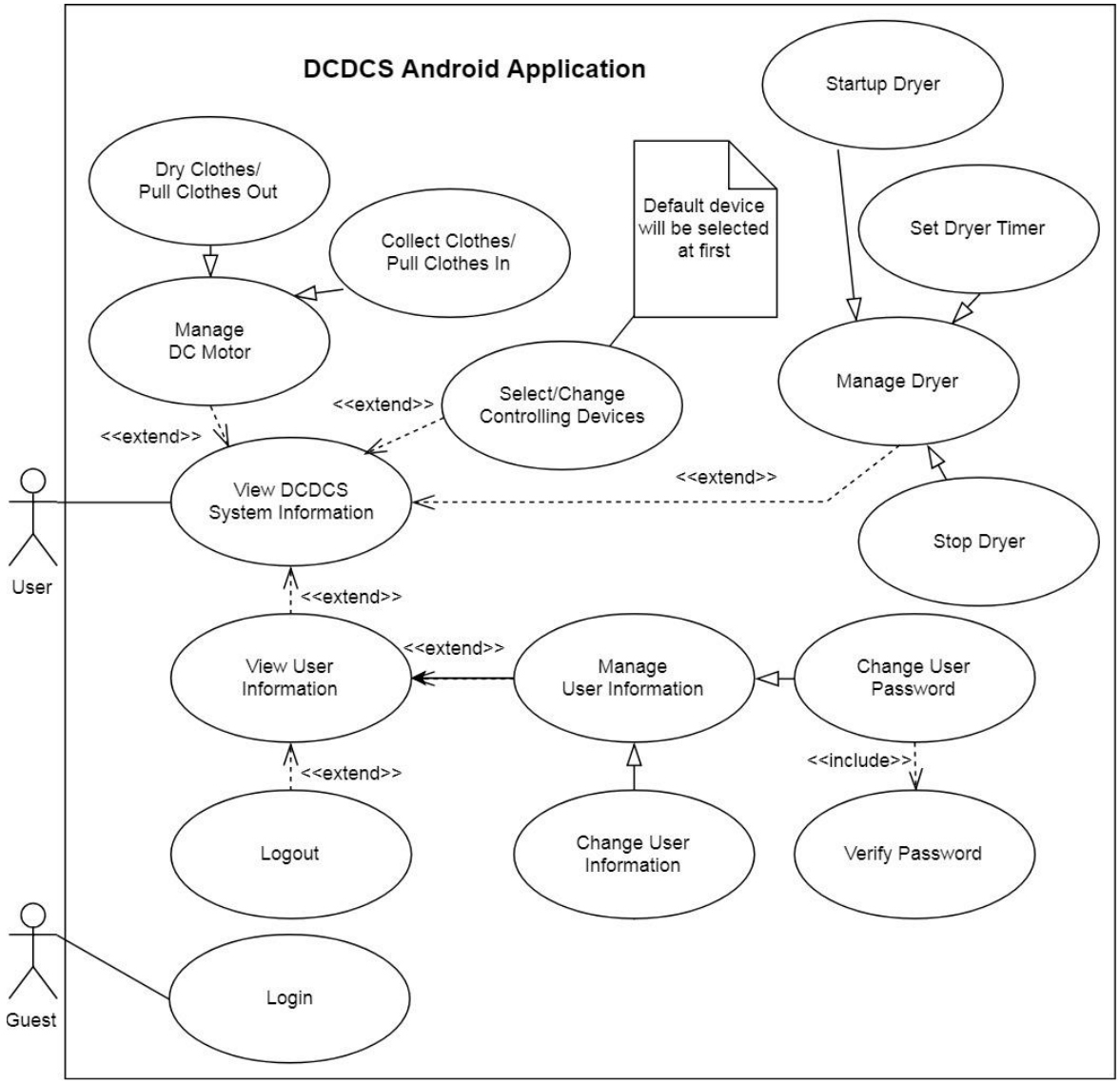

Figure 3: Android application overview usecase diagram

3. Hardware Requirement Specification

3.1 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

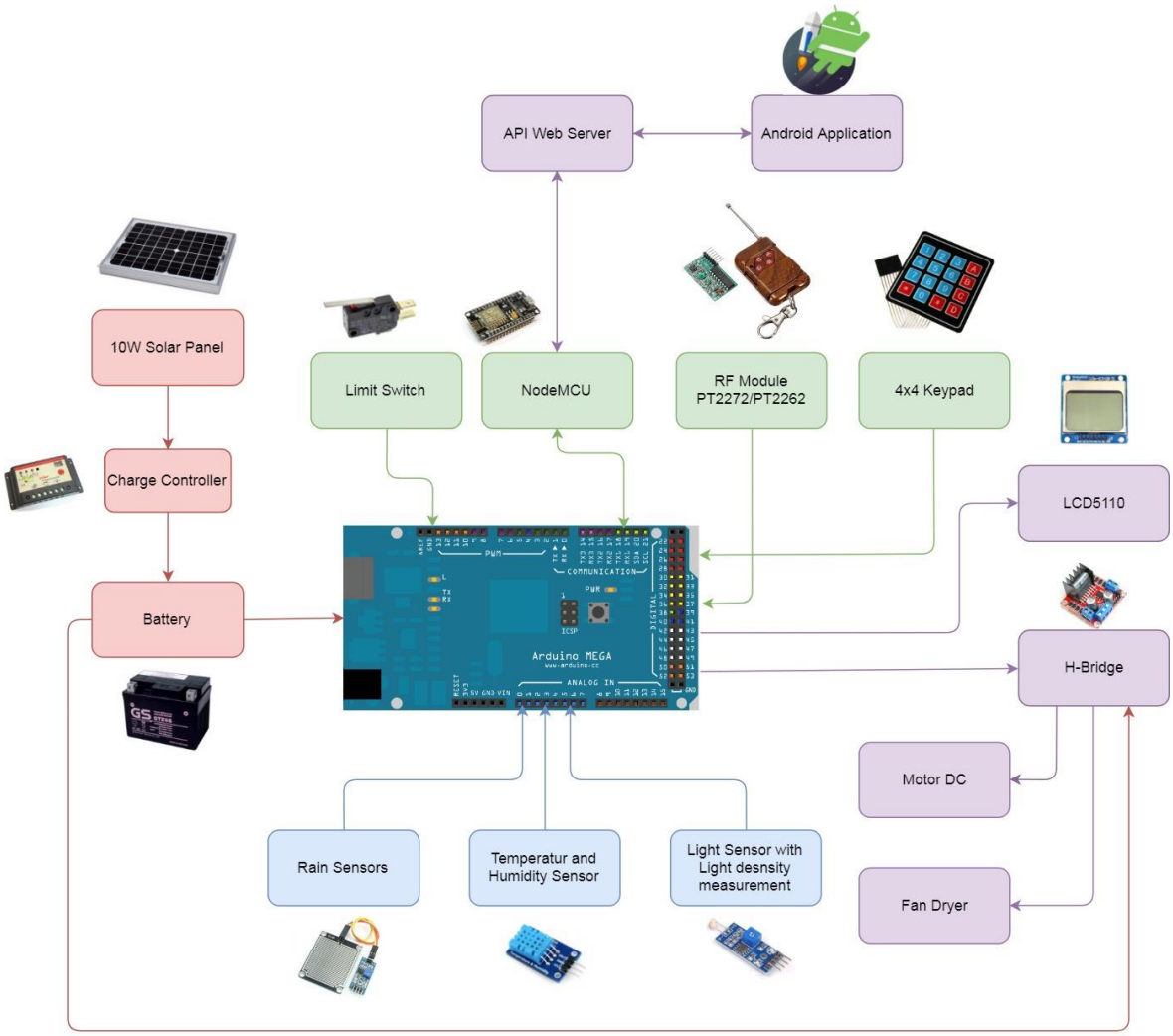

Figure 4: System block diagram

4. Conceptual Diagram

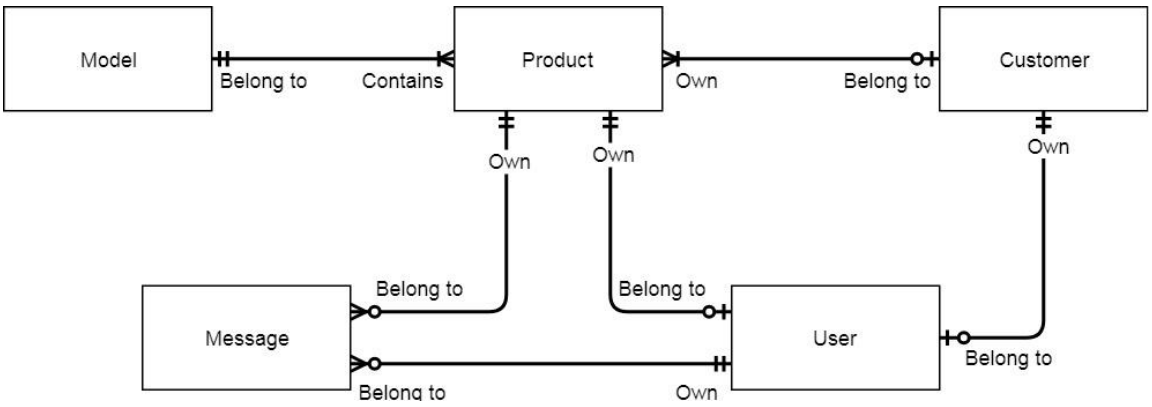

Figure 5: Conceptual diagram

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

D. Software & Hardware Design Description

1. 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

2. System Architectural Design

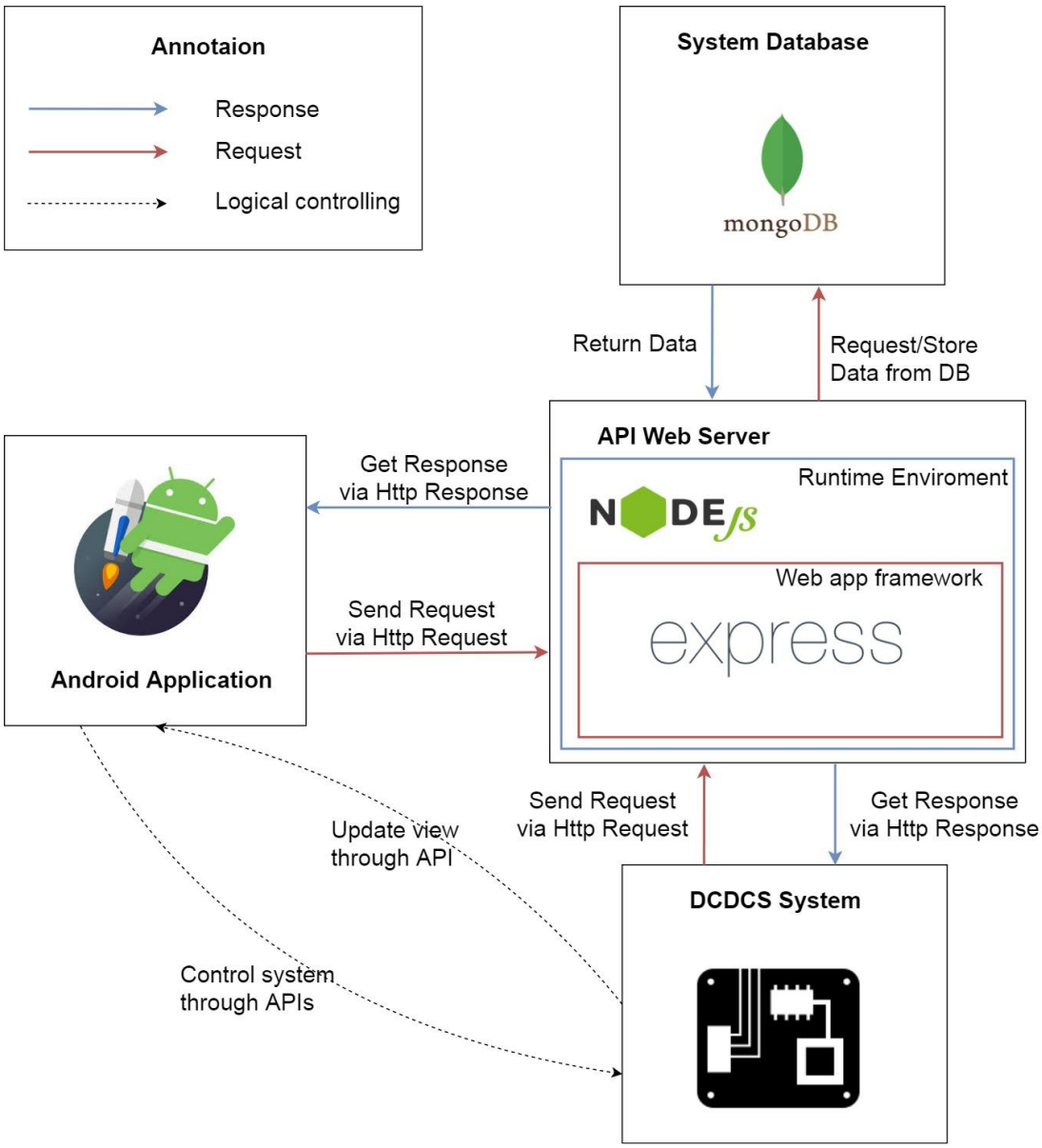

Figure 6: System overview architecture

2.1 API Web Server Architectural Design

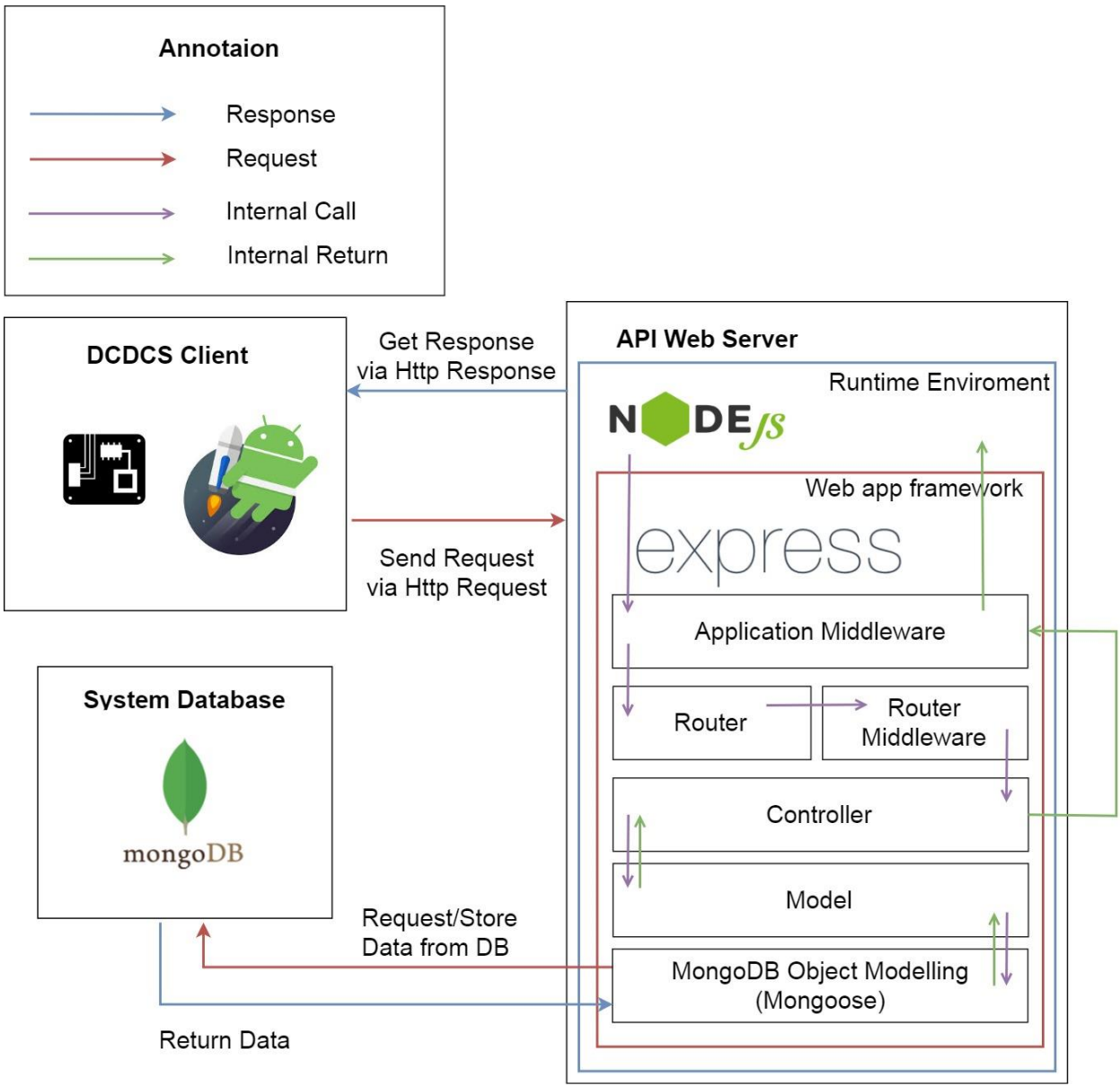

Figure 7: API Web server architecture

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.

2.2 Android Application Architectural Design

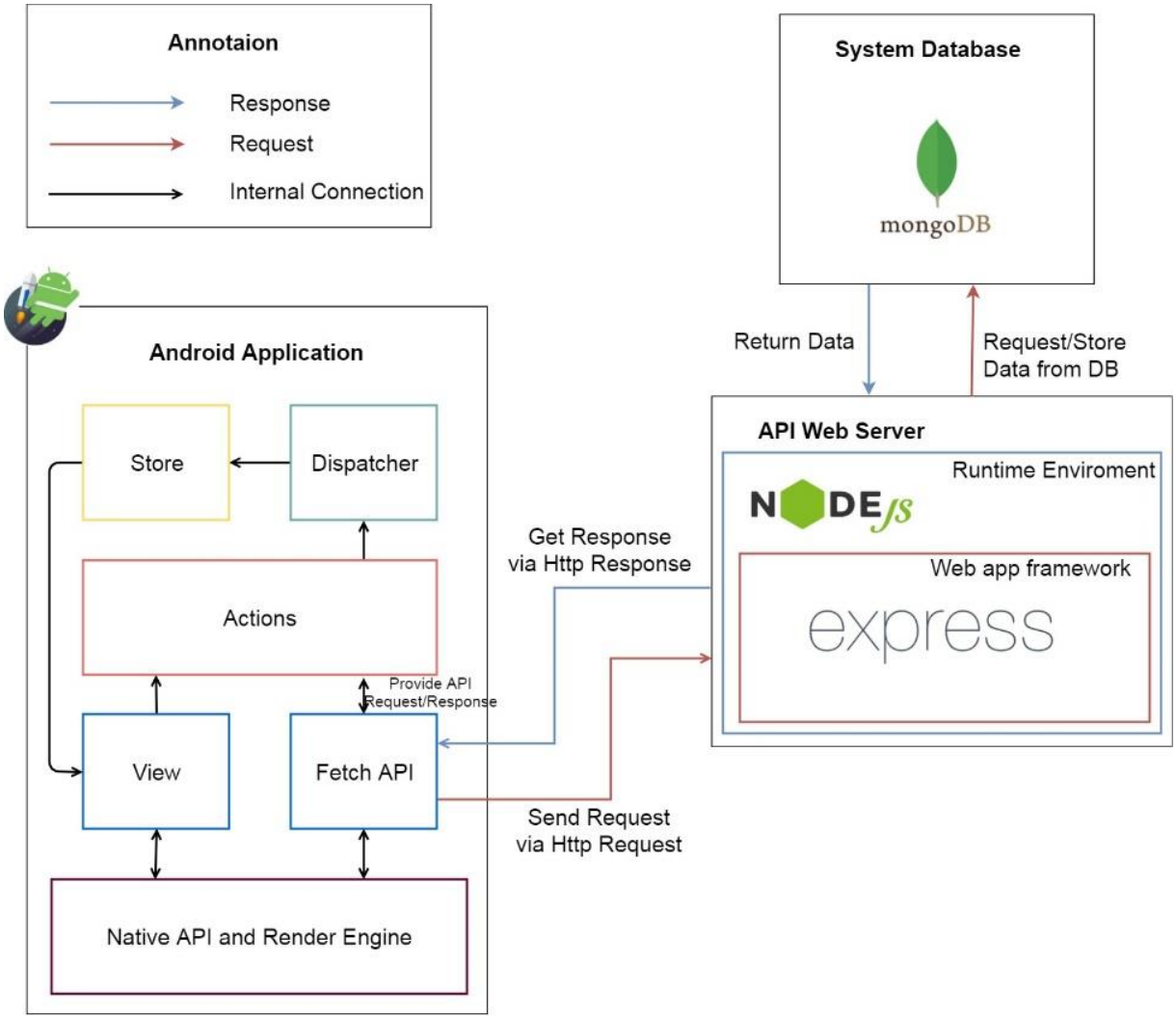

Figure 8: Android application internal architecture

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.

2.3 Hardware System Architecture

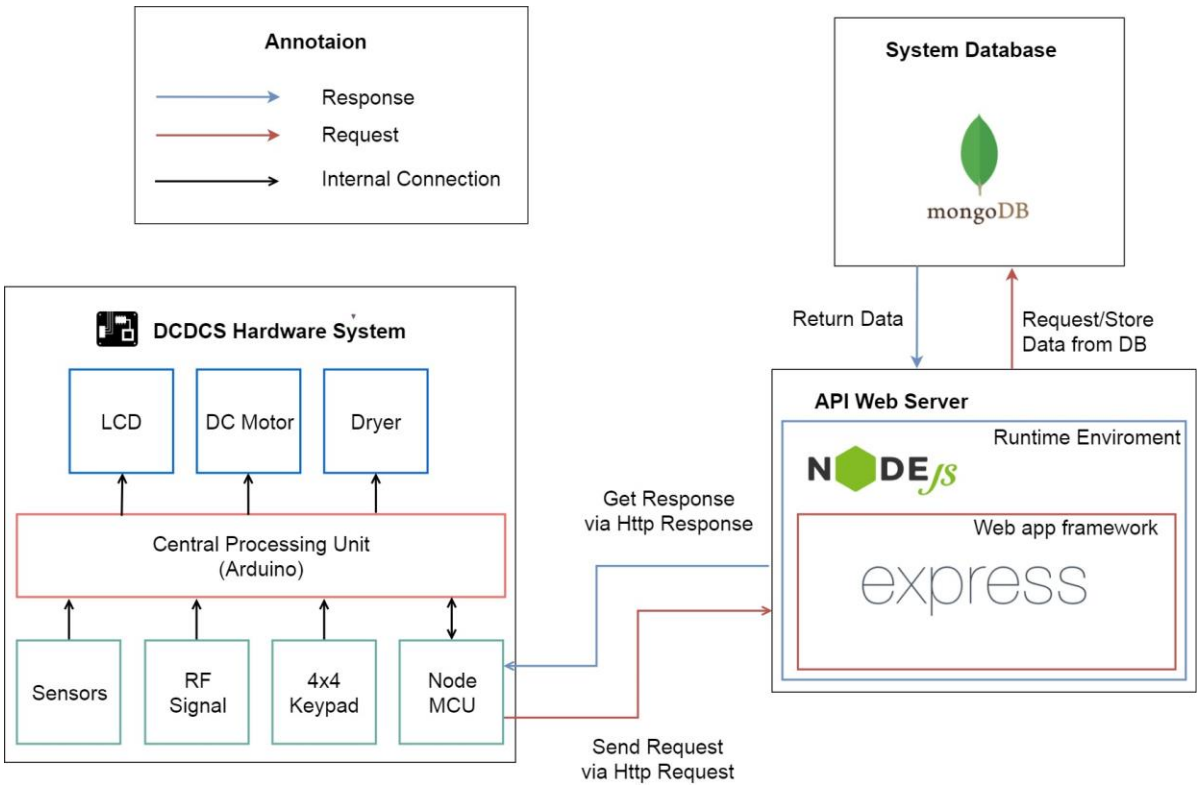

Figure 9: Hardware system architecture

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.

3. Component Diagram

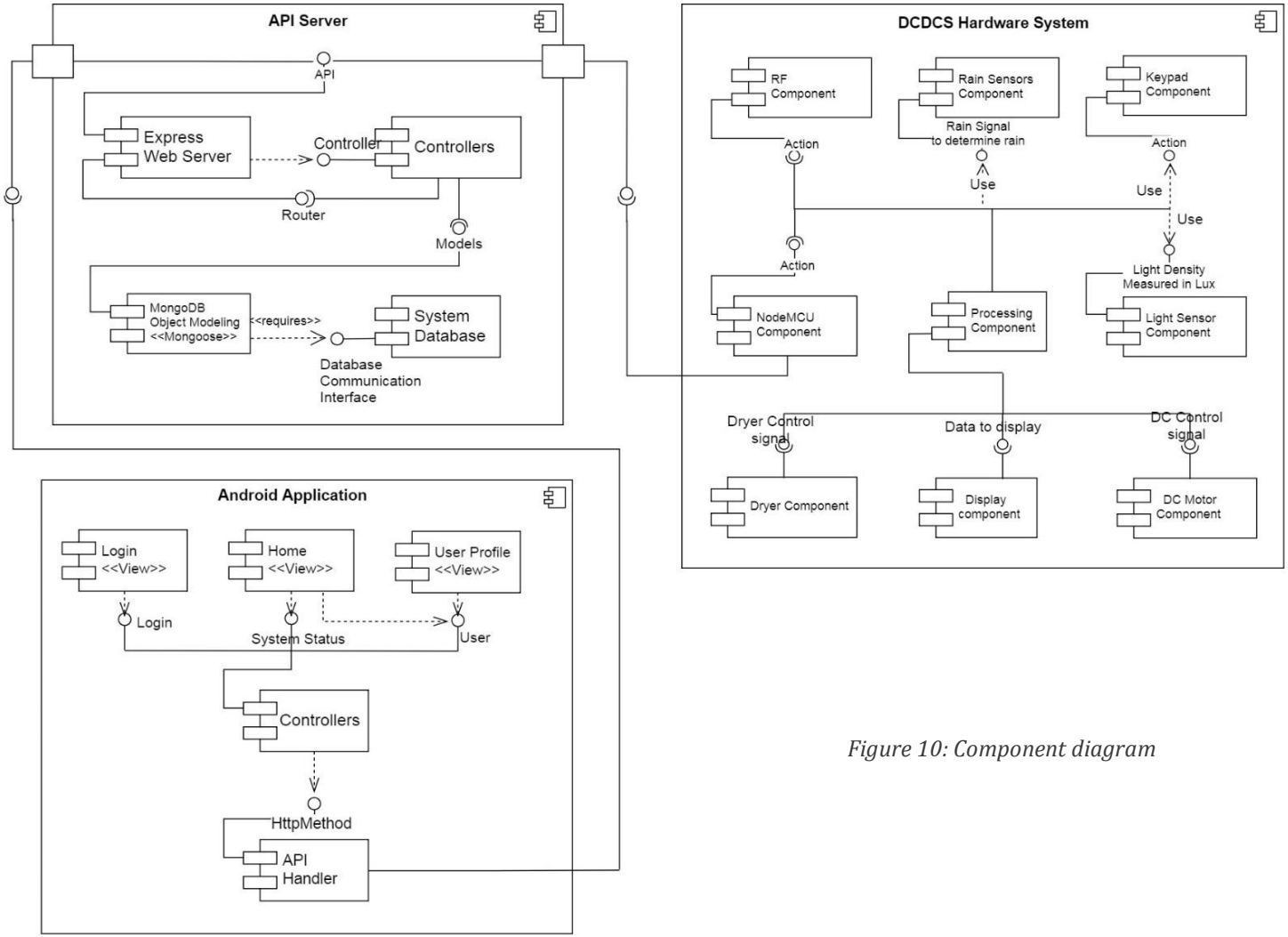

Figure 10: Component diagram

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

4. Detailed Description

3.3 Class Diagram

3.3.1 API Web Server

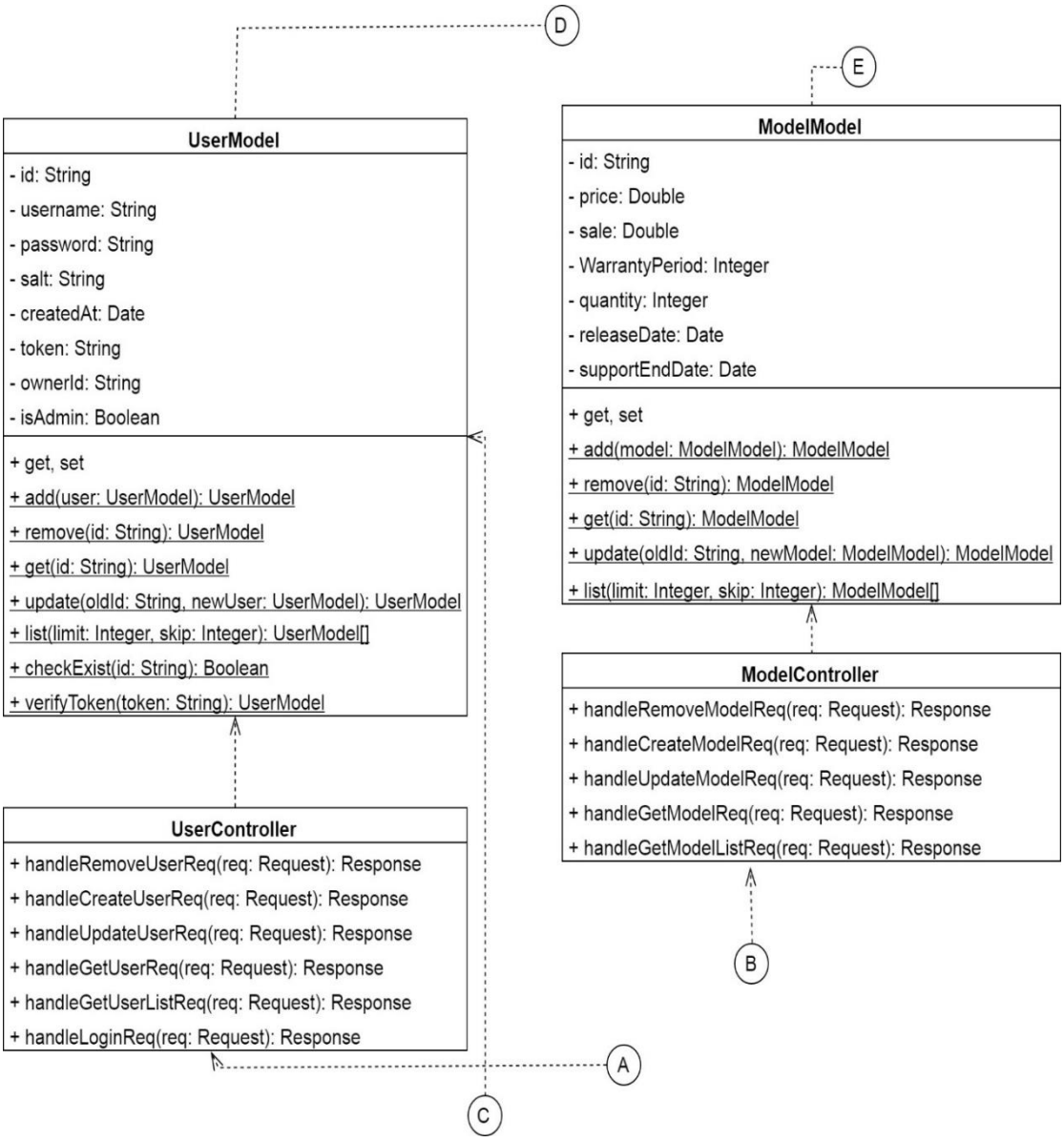

Figure 11: API Web Server Class Diagram Part 1

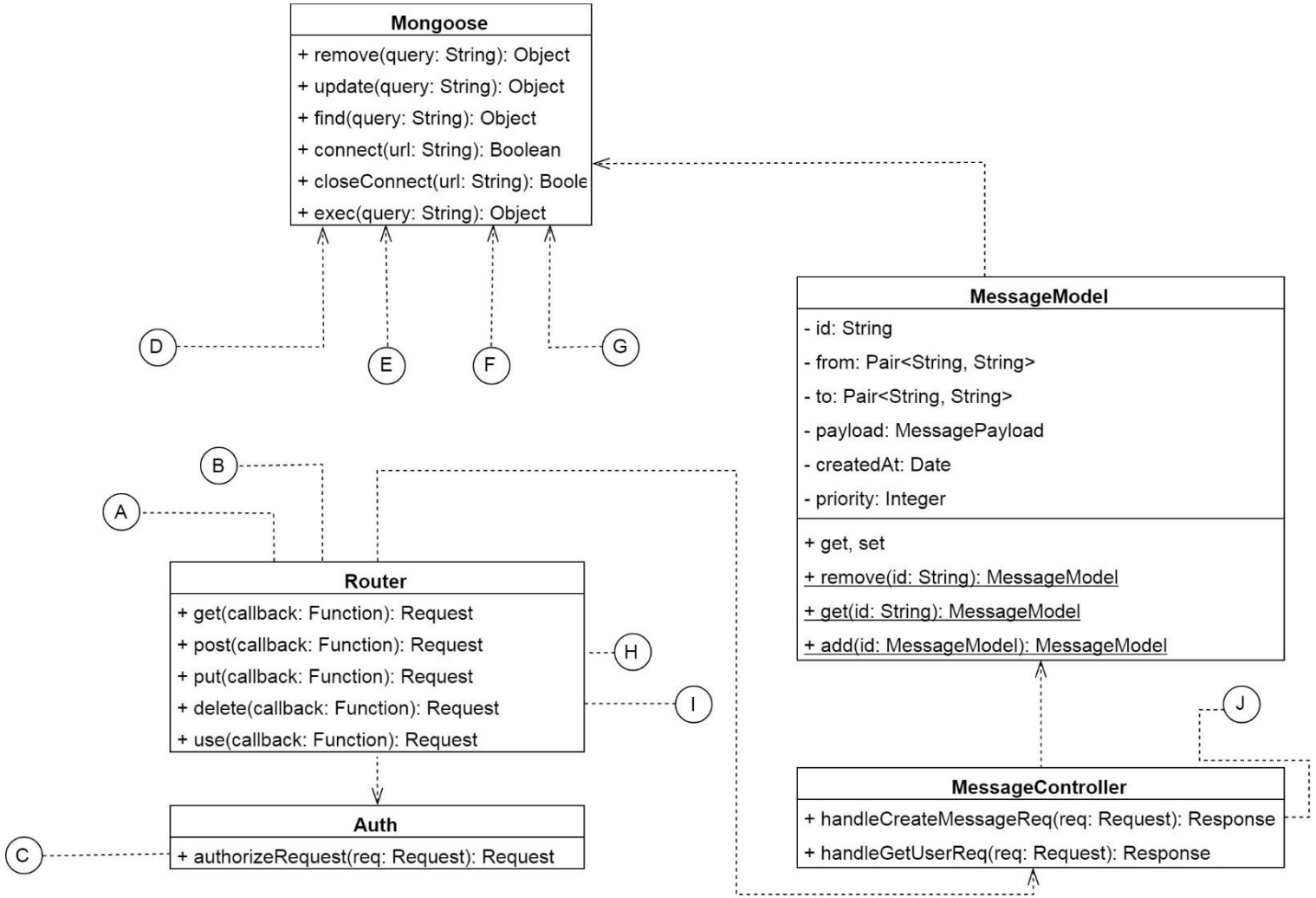

Figure 13: API Web Server Class Diagram Part 2

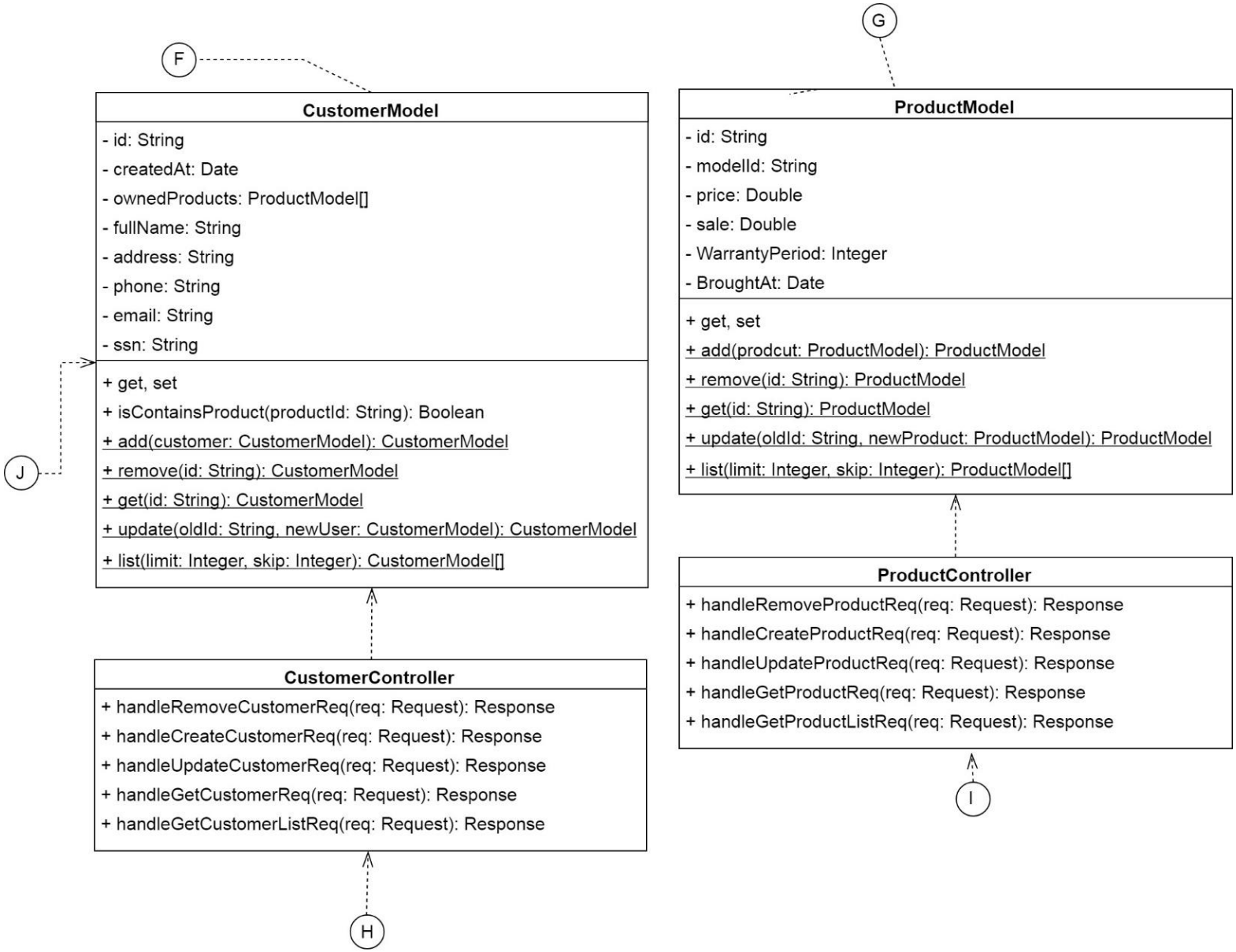

Figure 14: API Web Server Class Diagram Part 3

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

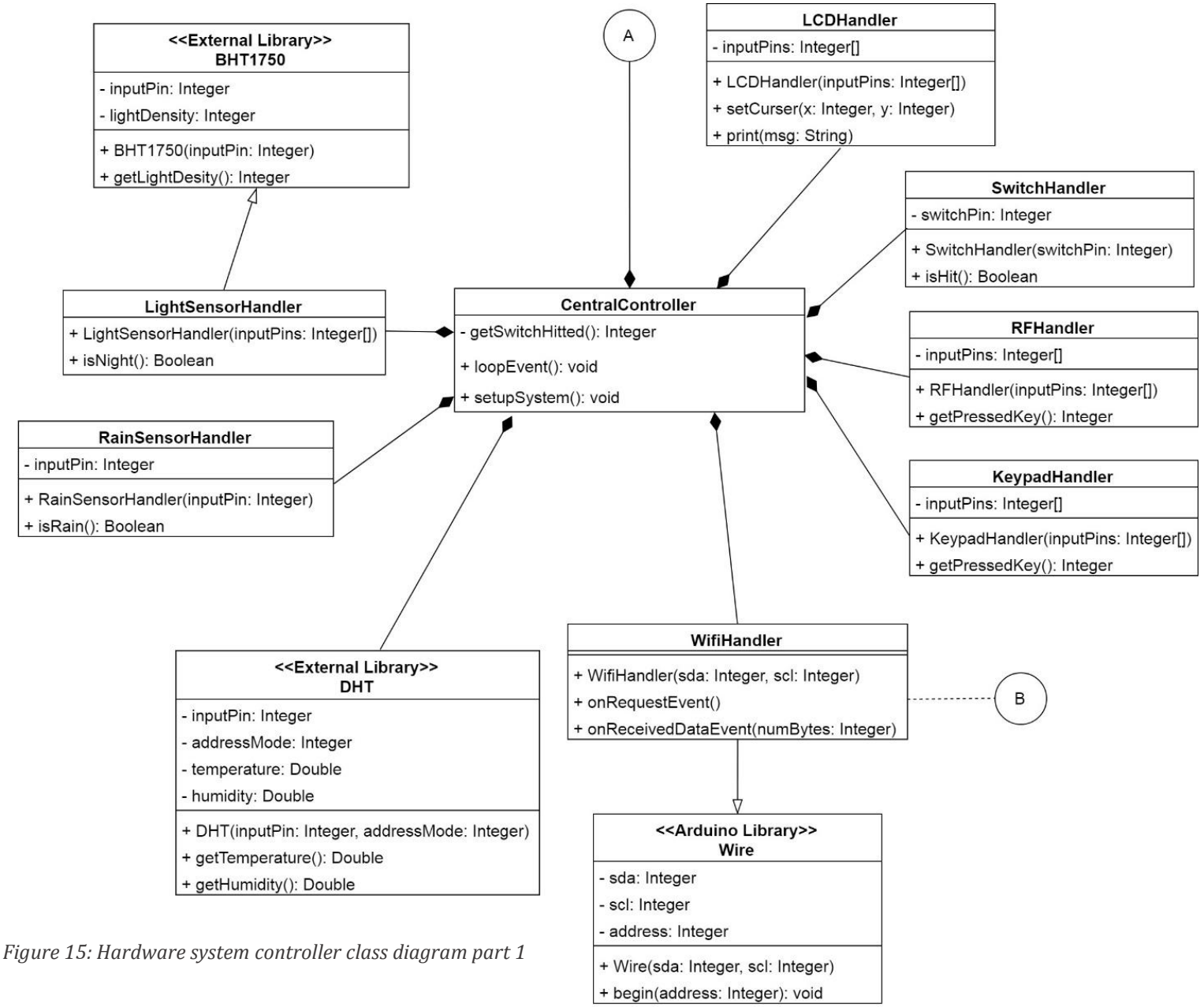

Figure 15: Hardware system controller class diagram part 1

3.3.3 Hardware System

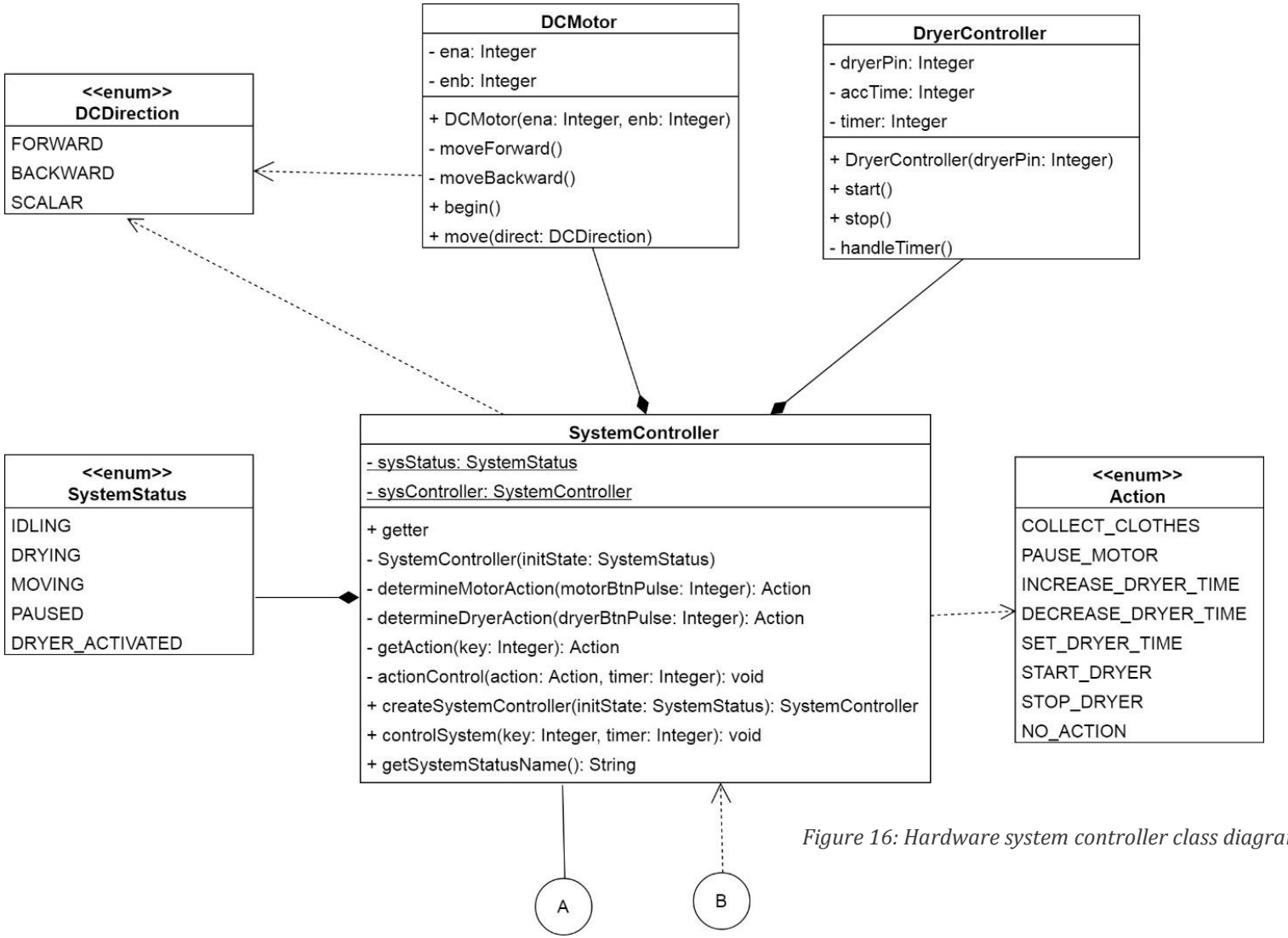

Figure 16: Hardware system controller class diagram part 2

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

3.4 Interaction Diagram

3.4.1 Sequence Diagrams

3.4.1.1 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

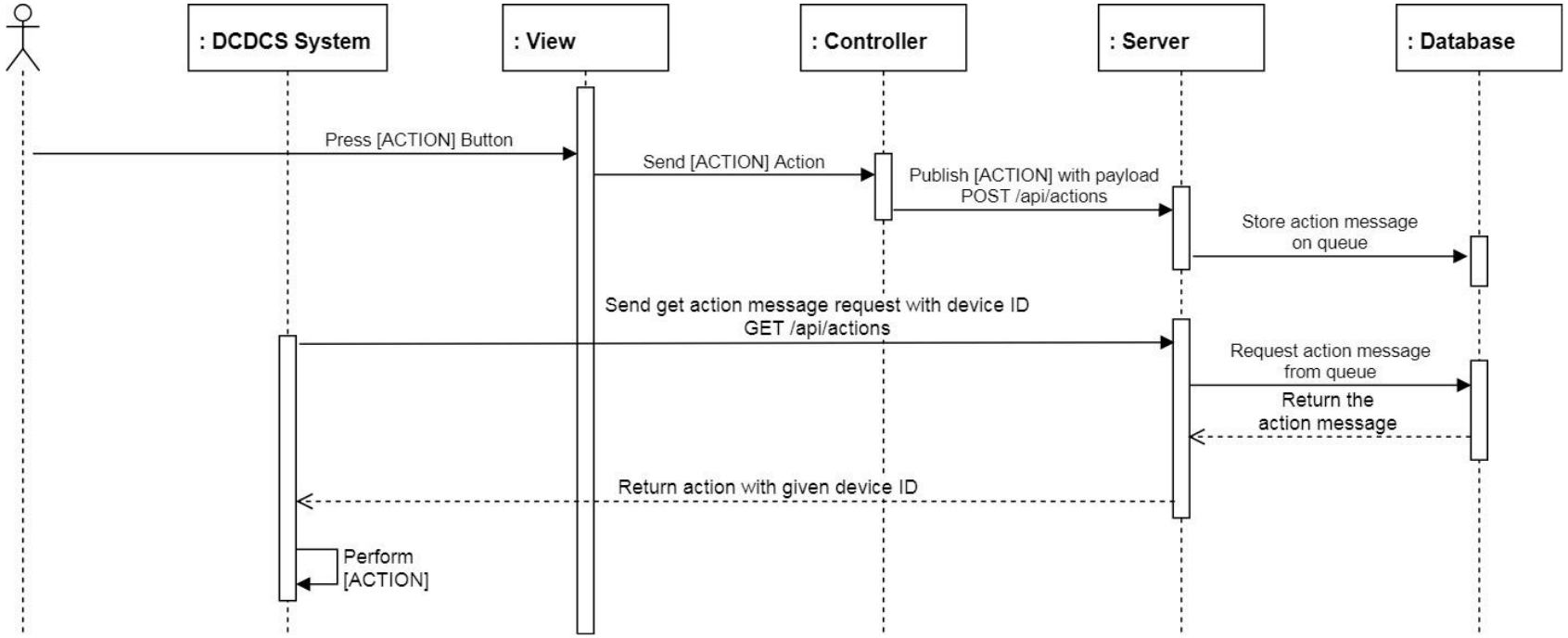

Figure 17: Control system with android app sequence diagram

3.4.1.2 Update system information

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

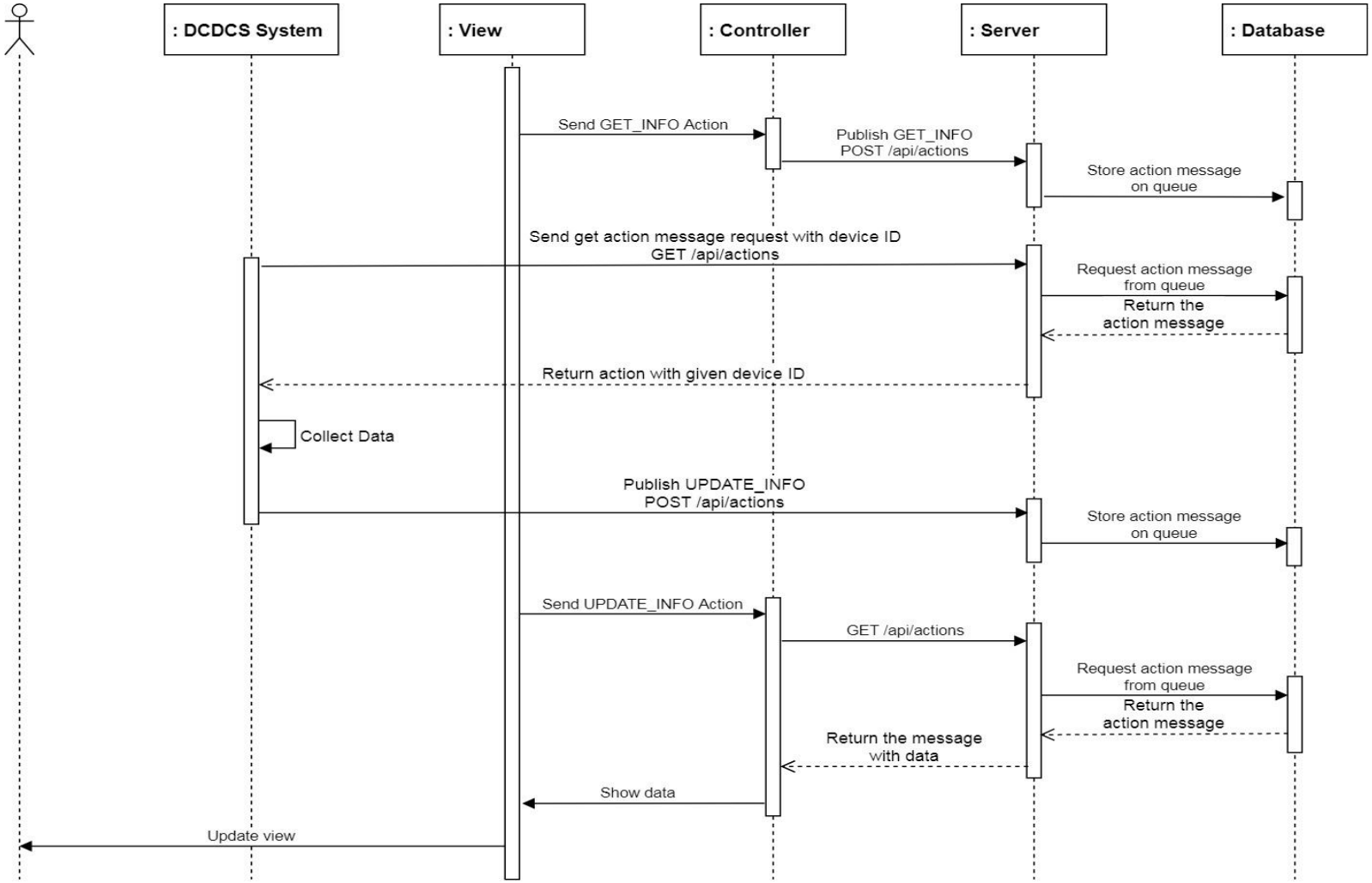

Figure 18: Update system information sequence diagram

3.4.2 Activity Diagrams

3.4.2.1 Control DC

Summary: This diagrams show how user can control the DC

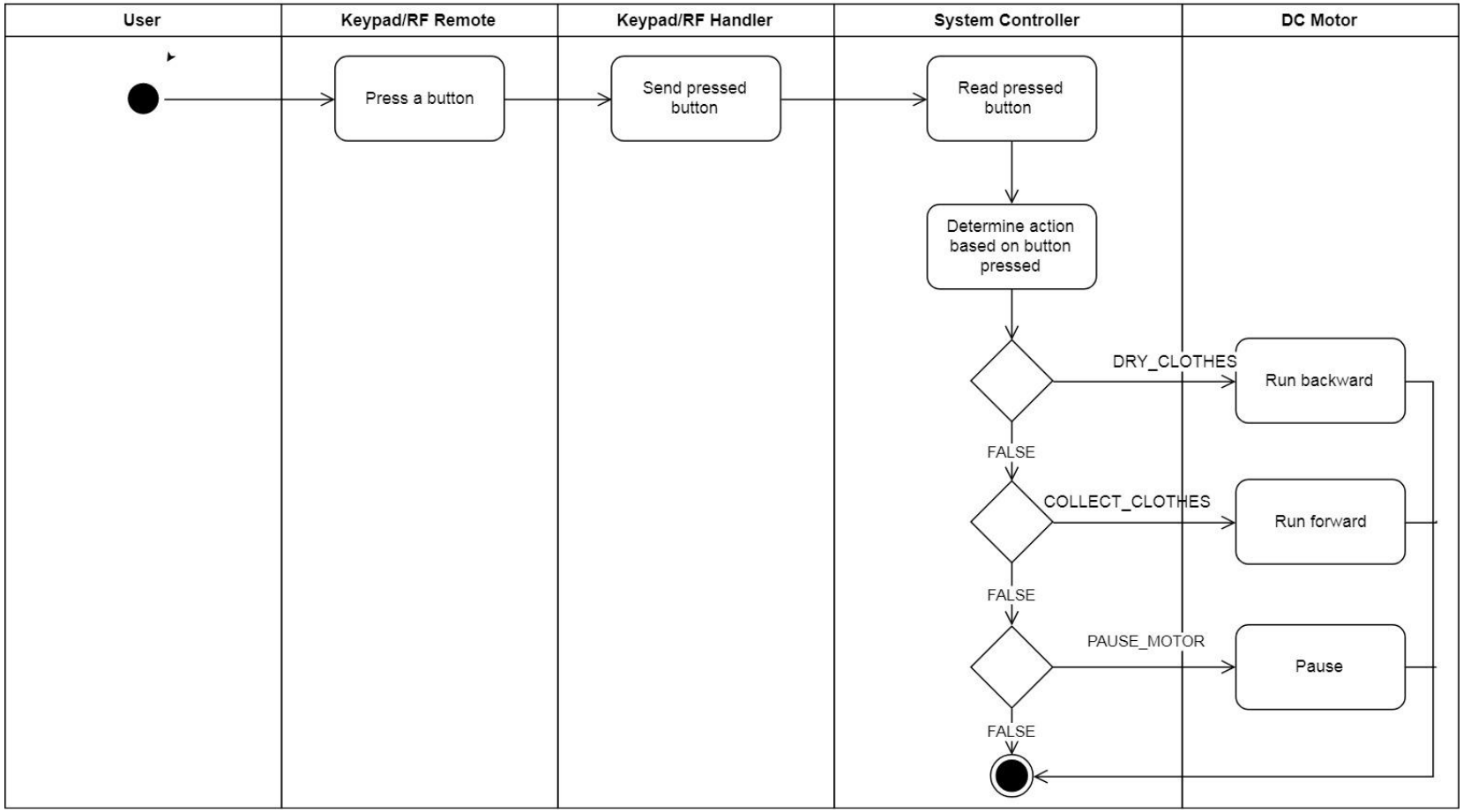

Figure 19: Control DC activity diagarm

3.4.2.2 Control Dryer

Summary: This diagrams show how user can control the dryer

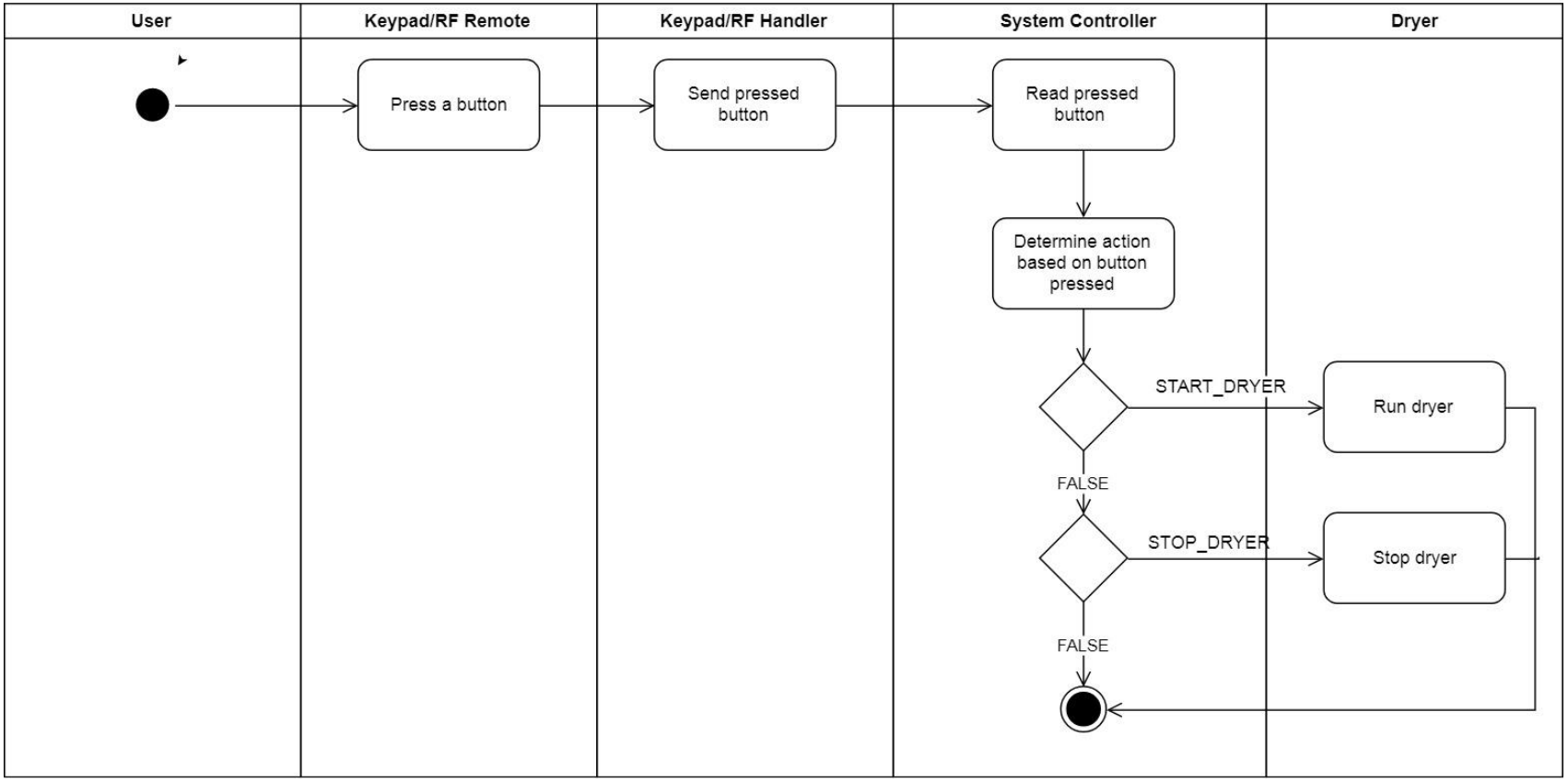

Figure 20: Control Dryer activity diagram

3.4.2.3 Auto control

Summary: This diagrams show how system itself control.

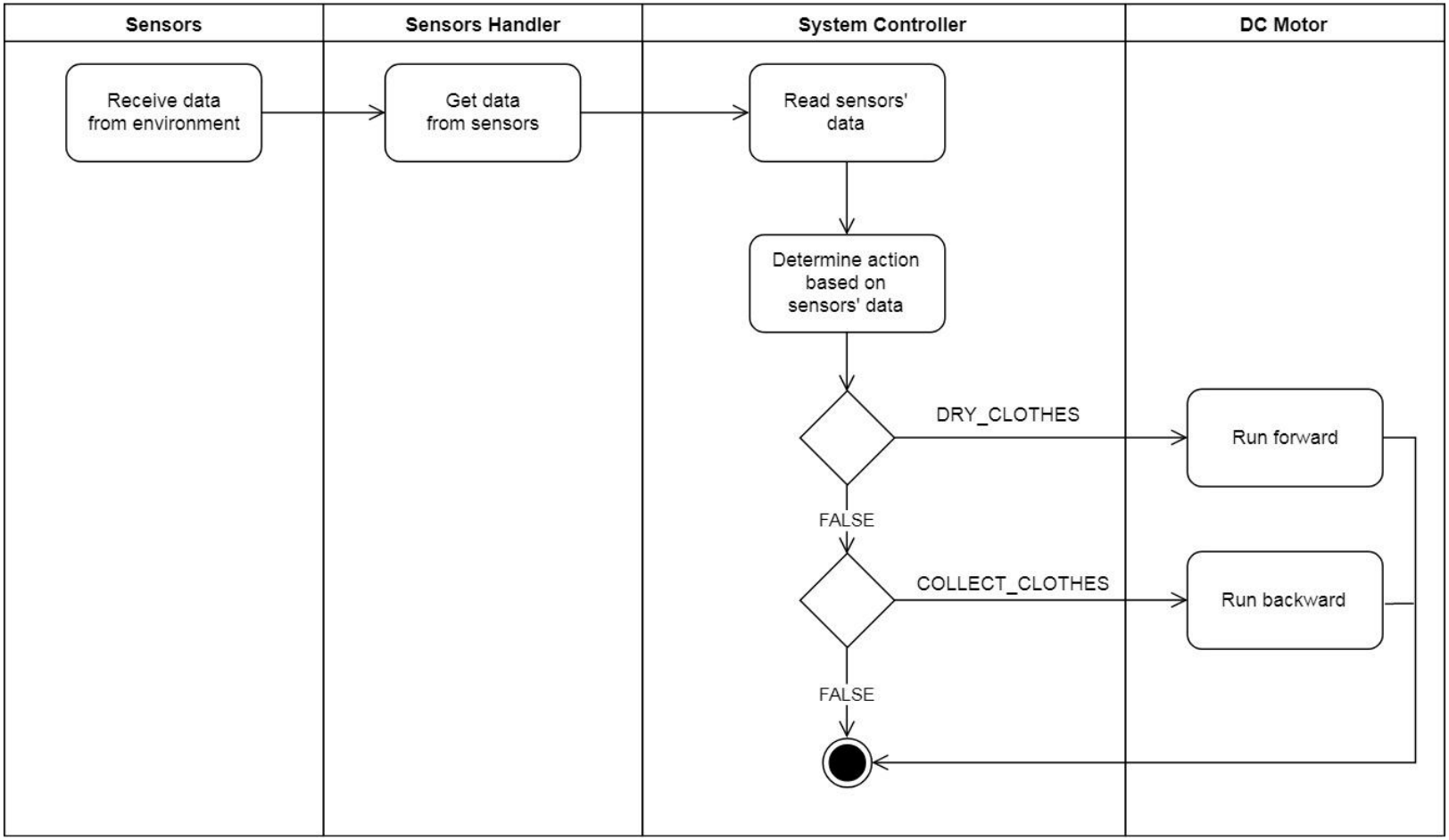

Figure 21: Auto control activity diagram

4. Database Design

4.1 Entity Relational Database (ERD)

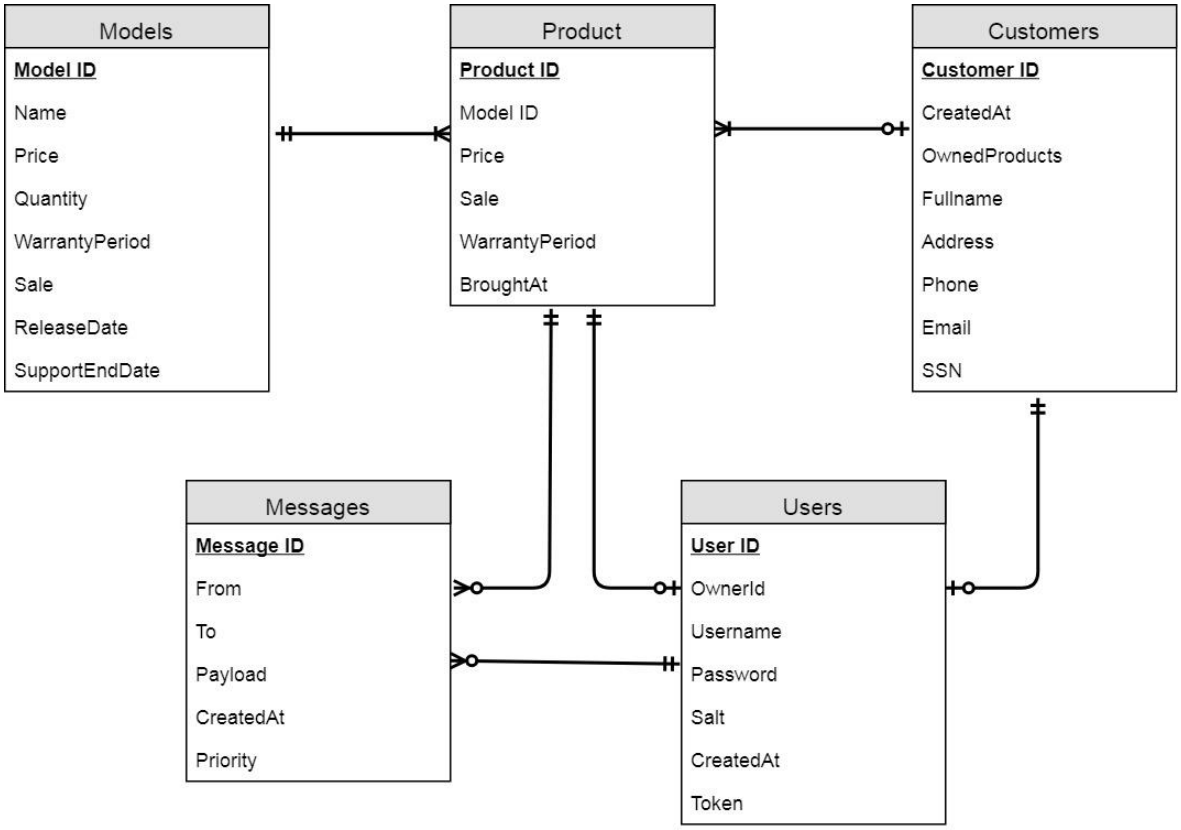

Figure 22: Entity Relational Database

4.2 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

5. Algorithms

5.1 System Control

5.1.1 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.

5.1.2 Define Problem

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

5.1.3 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.

5.1.4 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)

5.1.5 Algorithm Complexity

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

5.1.6 Overview Flowchart

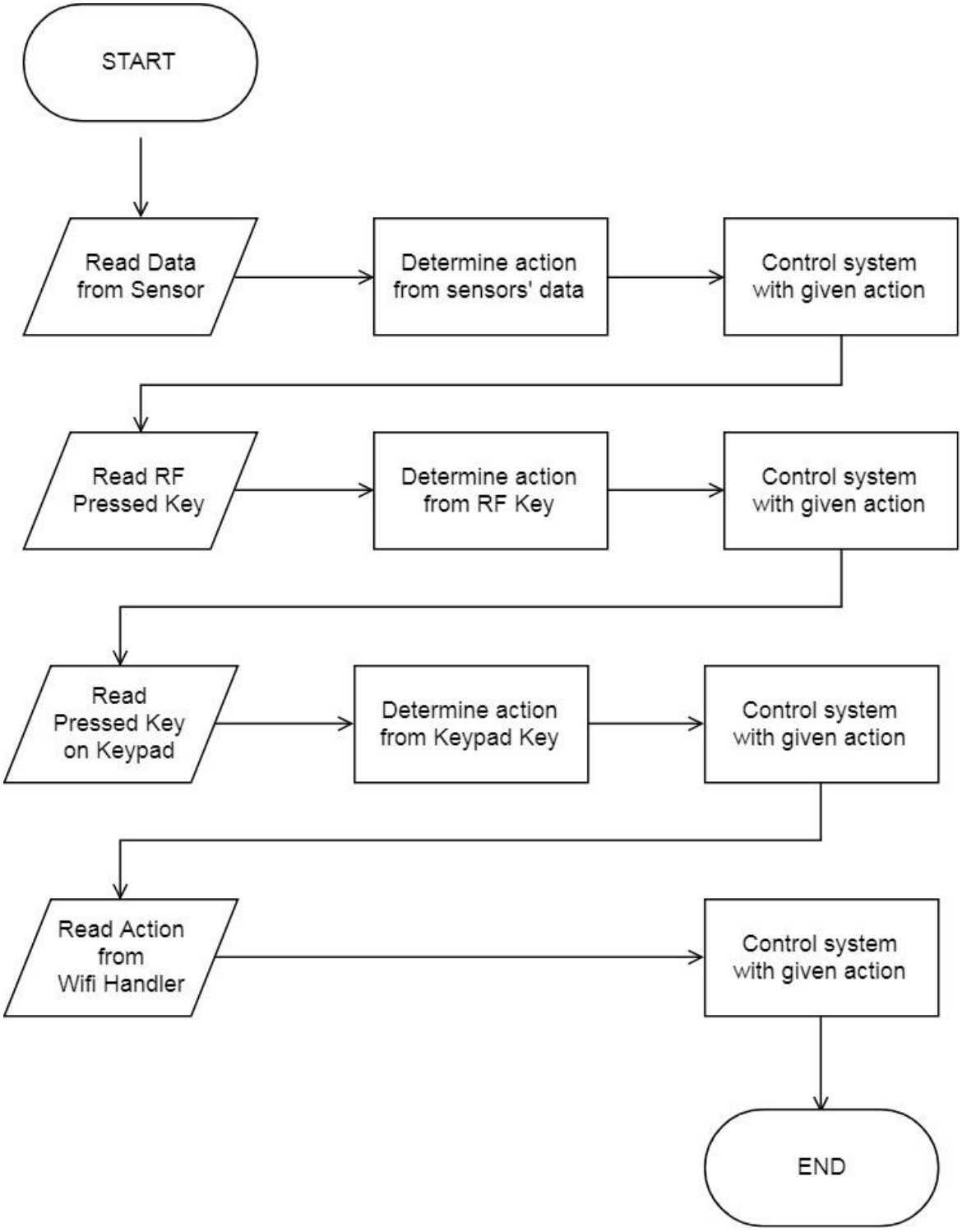

Figure 23: System Control overview flowchart

E. Task Sheet

No	Product Deliverables	Task	LongHP	PhongND	BinhT	Unit	Size
1	Report 1 - Introduction	Project Information	**O**				1
		Introduction	**O**				1
		Current Situation	**O**				1
		Problem Definition	**O**				1
		Proposed Solution					
		Feature Functions	**O**				1
		Advantages and Disadvantages	**O**				1
		Functional Requirement	**O**				1
		Roles and Responsibility	**O**				1
		Conclusion	**O**				1
2	Report 2 - Software Project Management Plan	**Problem Definition**					
		Name of this Capstone Project			**O**		1
		Problem Abstract			**O**		1
		Problem Overview					
		Current Situation		**O**			1
		The Proposed System		**O**			1
		Boundaries of the system	**O**				1
		Future Plan	**O**				1
		Development Environment					
		Hardware Development Environment Requirement	**O**				1
		Software Development Environment Requirement	**O**				1
		Project Organization					
		Software Process Model	**O**				1
		Roles and Responsibility	**O**				1
		Tools and Techniques		**O**			1
		Project Management Plan					
		Software development life cycle	**O**				1
		Phase Detail	**O**				1
		Task sheet	**O**				1
		All Meeting Minutes		**O**	**O**		1
		Coding Convention					1
		C++	**O**				1
		Javascript	**O**				1
	Report 3 - Software Requirement Specification	**Software & Hardware Requirement Specification**					
		User Requirement Specification	**O**				1
		System Requirement Specification					

		External Interface Requirements					1
		User Interface	**0**				1
		Hardware Interface	**0**				1
		System Overview Use Case					
		Hardware System Usecase	**0**	**0**			3
		Android Application Usecase	**0**	**0**			3
		List of Use cases					
		<Guest> Overview Usecase		**0**			1
		<Android User> Overview Usecase					
		<Android User> Stop dryer		**0**			1
		<Android User> Start dryer		**0**			1
		<Android User> Setup dryer timer		**0**			1
		<Android User> Control DC to dry clothes		**0**			1
		<Android User> Control DC to collect clothes		**0**			1
		<Android User> Change controlling device		**0**			1
		<Android User> View system information		**0**			1
		<Android User> View user information		**0**			1
		<Android User> Change user information		**0**			1
		<Android User> Change user password		**0**			1
		<Android User> Logout		**0**			1
		<System User> Overview Usecase					3
		<System User> Turn on power		**0**			
		<System User> Turn off power		**0**			
		<System User> Stop dryer		**0**			
		<System User> Start dryer		**0**			
		<System User> Set dryer timer		**0**			
		<System User> View system information		**0**			
		<System User> Pause DC Motor		**0**			
		<System User> Control DC to dry clothes		**0**			
		<System User> Control DC to collect clothes		**0**			

		<System User> Connect to Wi-Fi		**O**			
		<System> Overview Usecase					1
		<System> Control DC to collect clothes	**O**				
		<System> Show information	**O**				
		<System> Broadcast Wi-Fi	**O**				
		<System> Send HTTP Request	**O**				
		<System> Get HTTP Response	**O**				
		Hardware Requirement Specification					
		Hardware Interface					
		Rain Sensor			**O**		1
		Arduino Mega 2560 R3			**O**		1
		Humidity & Temperature DHT11			**O**		1
		Light Sensor BH1750			**O**		1
		LCD Nokia 5110			**O**		1
		Matrix Keypad (4x4)			**O**		1
		Limit Switches			**O**		1
		DC Motor GA37 125RPM			**O**		1
		Solar Panel			**O**		1
		Solar Charge Controller			**O**		1
		GTZ5S-E Battery			**O**		1
		NodeMCU			**O**		1
		Communication Protocol					
		I2C Protocol			**O**		1
		SPI Protocol			**O**		1
		UART Protocol			**O**		1
		HTTP Protocol			**O**		1
		System Attribute					
		Usability	**O**				1
		Reliability	**O**				1
		Availability	**O**				1
		Maintainability	**O**				1
		Portability	**O**				1
		Performance	**O**				1
		Security	**O**				1
		Conceptual Diagram	**O**				1
	Report 4 - Software	Design Overview	**O**				
		System Architectural Design					

	Design Description	API Web Server Architectural Design	**0**				1
		Android Application Architectural Design					
		Android Application Overview Architecture	**0**				1
		Android Application Internal Architecture	**0**				1
		Hardware System Architecture	**0**				1
		Component Diagram	**0**		**0**		1
		Detailed Description					
		Class diagram					
		API Web Server	**0**				2
		Hardware System	**0**				2
		Class Diagram Explanation					
		UserModel	**0**		**0**		1
		ModelModel	**0**		**0**		1
		MessageModel	**0**		**0**		1
		CustomerModel	**0**		**0**		1
		ProductModel	**0**		**0**		1
		Mongoose	**0**		**0**		1
		UserController	**0**		**0**		1
		ModelController	**0**		**0**		1
		MessageController	**0**		**0**		1
		CustomerController	**0**		**0**		1
		ProductController	**0**		**0**		1
		Router	**0**		**0**		1
		Auth	**0**		**0**		1
		SystemController	**0**		**0**		1
		DryerController	**0**		**0**		1
		WifiHandler	**0**		**0**		1
		DCMotor	**0**		**0**		1
		LightSensorHandler	**0**		**0**		1
		LCDHandler	**0**		**0**		1
		SwitchHandler	**0**		**0**		1
		RFHandler	**0**		**0**		1
		KeypadHandler	**0**		**0**		1
		RainSensorHandler	**0**		**0**		1
		CentralController	**0**		**0**		1
		BHT1750	**0**		**0**		1

		Wire	**0**		**0**		1
		DHT	**0**		**0**		1
		Interaction Diagram					
		Sequence Diagrams	**0**				1
		Control system from android application	**0**				1
		Update system information	**0**				1
		Activity Diagrams					
		Control DC	**0**				1
		Control Dryer	**0**				1
		Auto control	**0**				1
		Determine action based on sensors' data	**0**				1
		Determine DC Motor action	**0**				1
		Determine dryer action	**0**				1
		Interface					
		Guest Interface	**0**				1
		User Interface					
		Home Screen	**0**				1
		Control DC Motor	**0**				1
		Control Dryer	**0**				1
		Select Product	**0**				1
		User Profile	**0**				1
		Database Design					
		Entity Relational Database (ERD)			**0**		1
		Data Dictionary		**0**			1
		Algorithms					
		System Control					
		Definition	**0**				1
		Define Problem	**0**				1
		Solution	**0**				1
		Pros & Cons	**0**				1
		Algorithm Complexity	**0**				1
		Overview Flowchart	**0**				1
	Report 5 - System Implementation and Test	**Introduction**					
		Overview	**0**				1
		Test approach	**0**				1
		Database Relationship Diagram					

		Physical Diagram	**0**				1
		Data Dictionary	**0**				1
		Performance Measures					
		Control System from Android App Performances		**0**			1
		Control System from RF Remote/Keypad		**0**			1
		Test plan					
		Features to be tested					1
		Android Application	**0**				
		Hardware System	**0**				
		API Web Server	**0**				
		Features not to be tested	**0**				1
		System Testing Test Case					
		State Machine Diagram					
		Control DC Motor			**0**		1
		Control Dryer			**0**		1
		Control System			**0**		1
		Android Application Test Case					
		Gather System Information		**0**	**0**		1
		Gather Customer Information		**0**	**0**		1
		Control DC Motor		**0**	**0**		1
		Control Dryer		**0**	**0**		1
		Change Control Device		**0**	**0**		1
		Chang user/customer information		**0**	**0**		1
		Hardware System Test Case					
		Control from RF Remote/Keypad		**0**	**0**		1
		Auto control		**0**	**0**		1
		Connect to Wi-Fi		**0**	**0**		1
		API Web Server Test Case	**0**				1
6	Report 6 - Software User's Manual	**Installation Guide**					
		Setup environment at server side					
		Hardware Requirements					
		Hardware System Requirement	**0**				1
		Server Hardware Requirement	**0**				1
		Android Hardware Requirement	**0**				1
		Software Requirement	**0**				1
		API Web Server Deployment Process		**0**			1

		Android Application Deployment Process		**O**			1
		System Controller Deployment Process			**O**		1
		System API Handler Deployment Process			**O**		1
		User's Guide					
		Hardware Configuration					
		Connect to local Wi-Fi	**O**				1
		Android Application					
		Control DC					
		Dryer Clothes	**O**				1
		Collect Clothes	**O**				1
		Control Dryer					
		Setup timer and start dryer		**O**			1
		Stop the dryer		**O**			1
		Change Control Device			**O**		1
		Change User Information			**O**		1

Table 17: Task sheet

F. Appendix

- Flux Architecture: https://facebook.github.io/flux/
- How Expo Works: https://docs.expo.io/versions/latest/workflow/how-expo-works
- React Native Mechanism Explanation: https://wetalkit.xyz/react-native-what-it-is-and-how-it-works-e2182d008f5e
- Bit Twiddling Hacks: https://graphics.stanford.edu/~seander/bithacks.html
- Understanding Node.js & Express.js: https://medium.com/@LindaVivah/the-beginners-guide-understanding-node-js-express-js-fundamentals-e15493462be1
- Understand Express: https://evanhahn.com/understanding-express/
- Visual Diagram Guide: https://www.visual-paradigm.com/guide/
- The 4 stages of an IoT architecture: https://techbeacon.com/4-stages-iot-architecture