|  |  |
| --- | --- |
|  | 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 3](#_Toc521504602)

[List of Tables 5](#_Toc521504603)

[List of Figures 6](#_Toc521504604)

[Definitions, Acronyms and Abbreviations 7](#_Toc521504605)

[A. Introduction 8](#_Toc521504606)

[1. Project Information 8](#_Toc521504607)

[2. Introduction 8](#_Toc521504608)

[3. Current Situation 8](#_Toc521504609)

[4. Problem Definition 9](#_Toc521504610)

[5. Proposed Solution 9](#_Toc521504611)

[5.1 Feature Functions 9](#_Toc521504612)

[5.2 Advantages and Disadvantages 9](#_Toc521504613)

[6. Functional Requirements 10](#_Toc521504614)

[7. Role and Responsibility 11](#_Toc521504615)

[8. Conclusion 11](#_Toc521504616)

[B. Software Project Management Plan 12](#_Toc521504617)

[1. Problem Definition 12](#_Toc521504618)

[1.1 Name of this Capstone project 12](#_Toc521504619)

[1.2 Problem Abstract 12](#_Toc521504620)

[1.3 Project Overview 12](#_Toc521504621)

[2. Project Organization 15](#_Toc521504622)

[2.1 Software Process Model 15](#_Toc521504623)

[2.2 Roles and Responsibility 16](#_Toc521504624)

[2.3 Tools and Techniques 17](#_Toc521504625)

[3. Project Management Plan 18](#_Toc521504626)

[3.1 System Development Life-cycle 18](#_Toc521504627)

[3.2 Plan Detail 20](#_Toc521504628)

[4. Coding Convention 23](#_Toc521504629)

[C. Software & Hardware Requirement Specification 25](#_Toc521504630)

[1. User Requirement Specification 25](#_Toc521504631)

[2. System Requirement Specification 25](#_Toc521504632)

[2.1 External Interface Requirement 25](#_Toc521504633)

[2.2 System Overview Usecase 27](#_Toc521504634)

[2.3 Usecase Details 29](#_Toc521504635)

[3. Hardware Requirement Specification 30](#_Toc521504636)

[3.1 Hardware Interface 30](#_Toc521504637)

[3.2 Communication Protocol 43](#_Toc521504638)

[4. System Attributes 45](#_Toc521504639)

[4.1 Usability 45](#_Toc521504640)

[4.2 Reliability 46](#_Toc521504641)

[4.3 Availability 46](#_Toc521504642)

[4.4 Maintainability 46](#_Toc521504643)

[4.5 Portability 46](#_Toc521504644)

[4.6 Performance 46](#_Toc521504645)

[D. Software & Hardware Design Description 47](#_Toc521504646)

[E. System Implementation & Testing 48](#_Toc521504647)

[F. Software & Hardware User’s Manual 49](#_Toc521504648)

[G. Acknowledgement 50](#_Toc521504649)

[H. Reference 51](#_Toc521504650)

# List of Tables

[Table 1: General Roles and Responsibilities of Member 12](#_Toc521505547)

[Table 2: Hardware development environment requirement for DCDCS System 15](#_Toc521505548)

[Table 3: Software development environment requirement for DCDCS System 16](#_Toc521505549)

[Table 4: Roles and responsibilities 18](#_Toc521505550)

[Table 5: Tools and techniques 18](#_Toc521505551)

[Table 6: Project task planning 20](#_Toc521505552)

[Table 7: Plain Detail - Requirement Analysis 21](#_Toc521505553)

[Table 8: Plain Detail - Design 22](#_Toc521505554)

[Table 9: Plain Detail – Implementation 23](#_Toc521505555)

[Table 10: Plain Detail –Testing 23](#_Toc521505556)

[Table 11: Plain Detail –Maintenance 23](#_Toc521505557)

[Table 12: Specifications for rain sensor 32](#_Toc521505558)

[Table 13: Specifications for Arduino mega 2560 34](#_Toc521505559)

[Table 14: Specifications for DHT11 35](#_Toc521505560)

[Table 15: Specifications for BH1750 35](#_Toc521505561)

[Table 16: Specifications for LCD5110 36](#_Toc521505562)

[Table 17: Specifications for 4x4 matrix keypad 37](#_Toc521505563)

[Table 18: Specifications for limit switch 38](#_Toc521505564)

[Table 19: Specifications for DC Motor GA37 39](#_Toc521505565)

[Table 20: Specifications for solar panel 40](#_Toc521505566)

[Table 21: Specifications for charge controller 42](#_Toc521505567)

[Table 22: Specifications for GTZ5S-E battery 42](#_Toc521505568)

[Table 23: Specifications for NodeMCU 43](#_Toc521505569)

# List of Figures

[Figure 1: Waterfall methodology 16](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505570)

[Figure 2: Hardware system overview usecase diagram 28](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505571)

[Figure 3: Android application overview usecase diagram 29](#_Toc521505572)

[Figure 4: System block diagram 31](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505573)

[Figure 5: Rain sensor 32](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505574)

[Figure 6: Arduino Mega 2650 R3 33](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505575)

[Figure 7: Module DHT11 34](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505576)

[Figure 8: Module Light sensor BH1750 35](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505577)

[Figure 9: Module LCD5110 36](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505578)

[Figure 10: 4x4 Matrix keypad 37](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505579)

[Figure 11: Limit switch 38](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505580)

[Figure 12: DC Motor GA37 39](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505581)

[Figure 13: 10W Solar Panel 40](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505582)

[Figure 14: Solar Charge Controller 41](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505583)

[Figure 15: GTZ5S-E Battery 42](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505584)

[Figure 16: NodeMCU v1.0 43](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505585)

[Figure 17: I2C Protocol 44](#_Toc521505586)

[Figure 18: SPI Protocol 45](#_Toc521505587)

[Figure 19: UART Protocol 45](#_Toc521505588)

[Figure 20: Arduino communicates with server through HTTP Protocol 46](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc521505589)

# Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| Name | Definition |

# Introduction

## Project Information

* **Project name:** DESIGN AND CONSTRUCTION SUN DRYING WET CLOTHES SYSTEM
* **Project Code:** DCDCS
* **Product Type:** Embedded Device, Android Application, RESTful Web Service
* **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

## Conclusion

* Research to determine and implement the appropriate MCU for the Central Control Unit and other nodes
* Design and implement integrate PCB board.
* Research and implement NoSQL Database, RESTful API, Mobile Application.
* C, C++ embedded into Arduino.
* Use software in design PCB, Schematic such as OrCAD, Proteus
* Communication technique: TCP, HTTP

# 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:

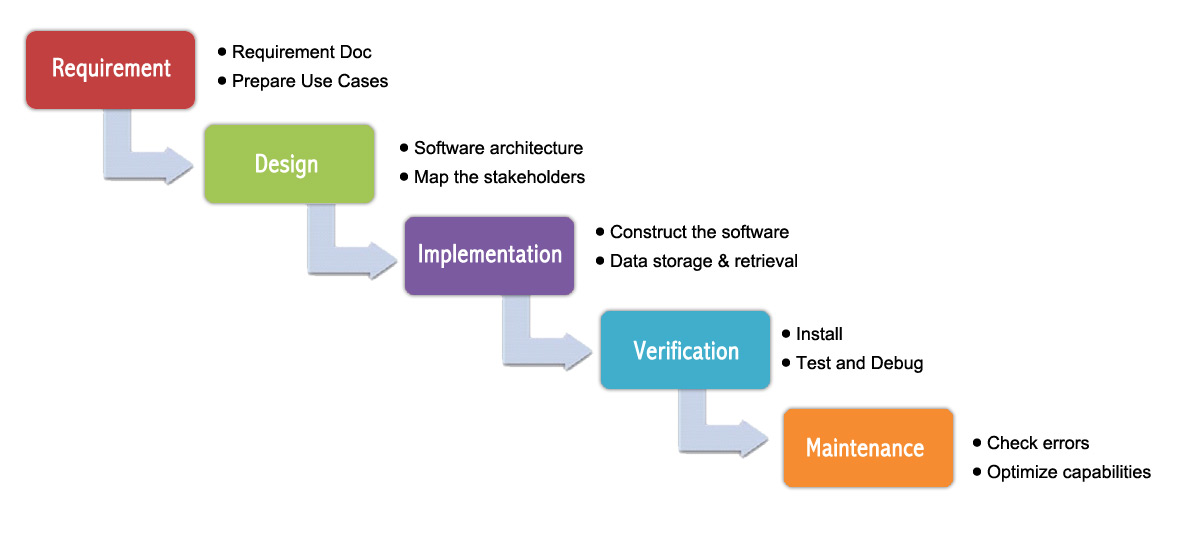
* This project is 4-months long due to the FPT University Capstone Project timeline, which can be consider a short project.
* Based on researches and current clarified face recognition system, the requirements of this project are stable, clear, fixed and well-understood by all team members.
* The Modified 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 |
| Web Service 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

## Coding Convention

* **C/C++ Convention**: Using to develop program and solve algorithm on hardware.
* Name convention:
  + Names should be descriptive; avoid abbreviation
  + Type names start with a capital letter and have a capital letter for each new word, with no underscores
  + The names of variables (including function parameters) and data members are all lowercase, with underscores between words. Data members of classes (but not structs) additionally have trailing underscores.
  + Variables declared constexpr or const, and whose value is fixed for the duration of the program, are named with a leading "k" followed by mixed case
  + Regular functions have mixed case; accessors and mutators may be named like variables.
* Comments:
  + Use either the // or /\* \*/ syntax, as long as you are consistent.
* Indentation:
  + Use only spaces, and indent 2 spaces at a time.
* Line length:
  + Each line of text in your code should be at most 80 characters long.
* More details about coding conventions for C/C++ language by Google:
  + <https://google.github.io/styleguide/cppguide.html>
* **Javascript Convention**: Using to develop web service and mobile application.
* Naming Convension:
  + Avoid single letter names. Be descriptive with your naming
  + Use camelCase when naming objects, functions, and instances
  + Use PascalCase only when naming constructors or classes
  + Do not use trailing or leading underscores
  + A base filename should exactly match the name of its default export
  + Use camelCase when you export-default a function. Your filename should be identical to your function’s name
  + Use PascalCase when you export a constructor / class / singleton / function library / bare object.
* Indentation:
  + Convert 1 tab to 2 spaces
* Comments:
  + Use /\*\* ... \*/ for multi-line comments.
  + Use // for single line comments. Place single line comments on a newline above the subject of the comment. Put an empty line before the comment unless it’s on the first line of a block.
  + Prefixing your comments with FIXME or TODO helps other developers quickly understand if you're pointing out a problem that needs to be revisited, or if you're suggesting a solution to the problem that needs to be implemented. These are different than regular comments because they are actionable. The actions are FIXME: -- need to figure this out or TODO: -- need to implement.
* References:
  + Using *const* and let instead of *var*
* More detail about code conventions for Javascript language by Airbnb:
  + <https://github.com/airbnb/javascript>

# 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

#### D:\Capstone Document\Diagrams\DCDCS System Use Case.jpgHardware System Usecase

Figure 2: Hardware system overview usecase diagram

#### Android Application Usecase

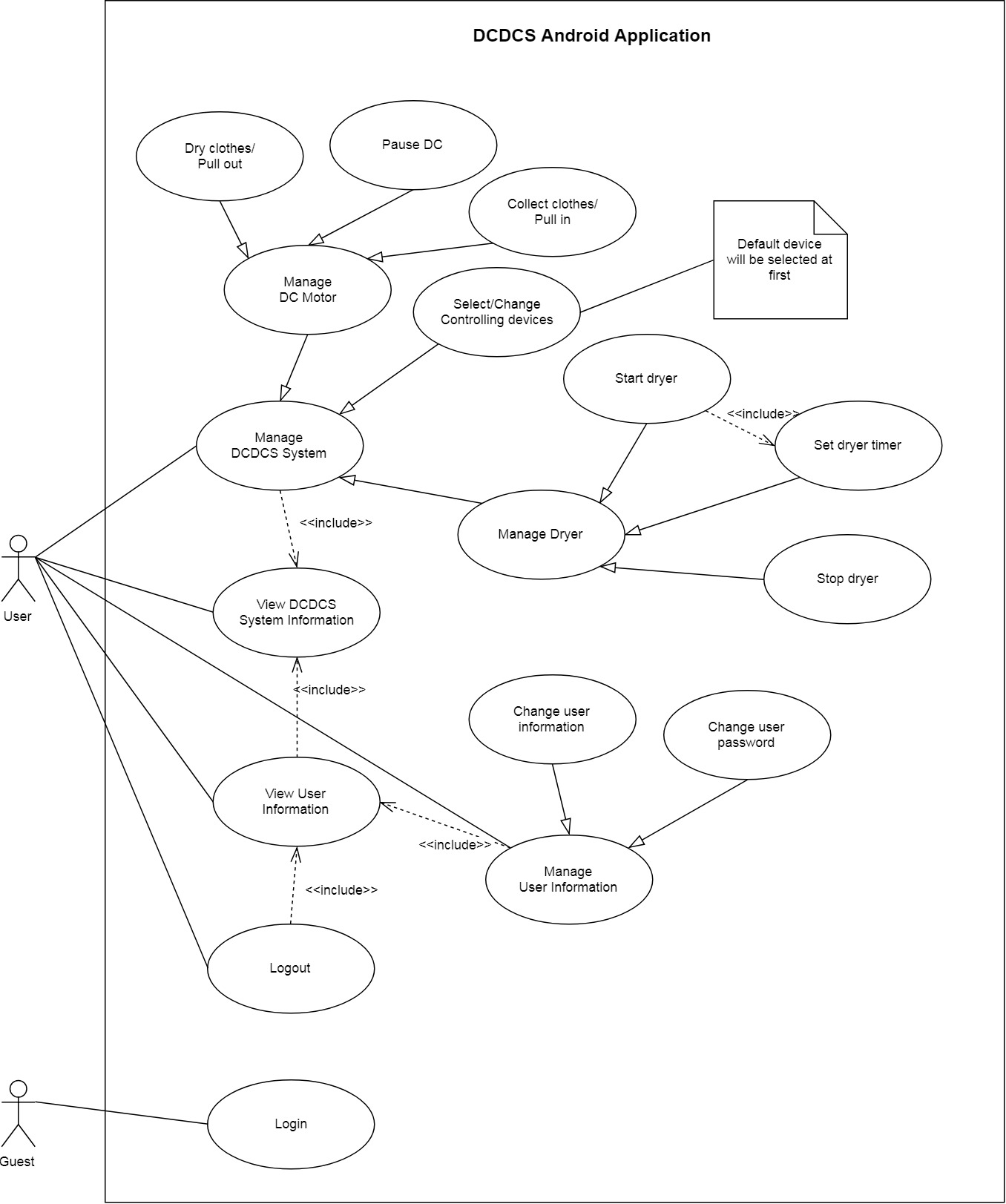


Figure 3: Android application overview usecase diagram

### Usecase Details

## 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

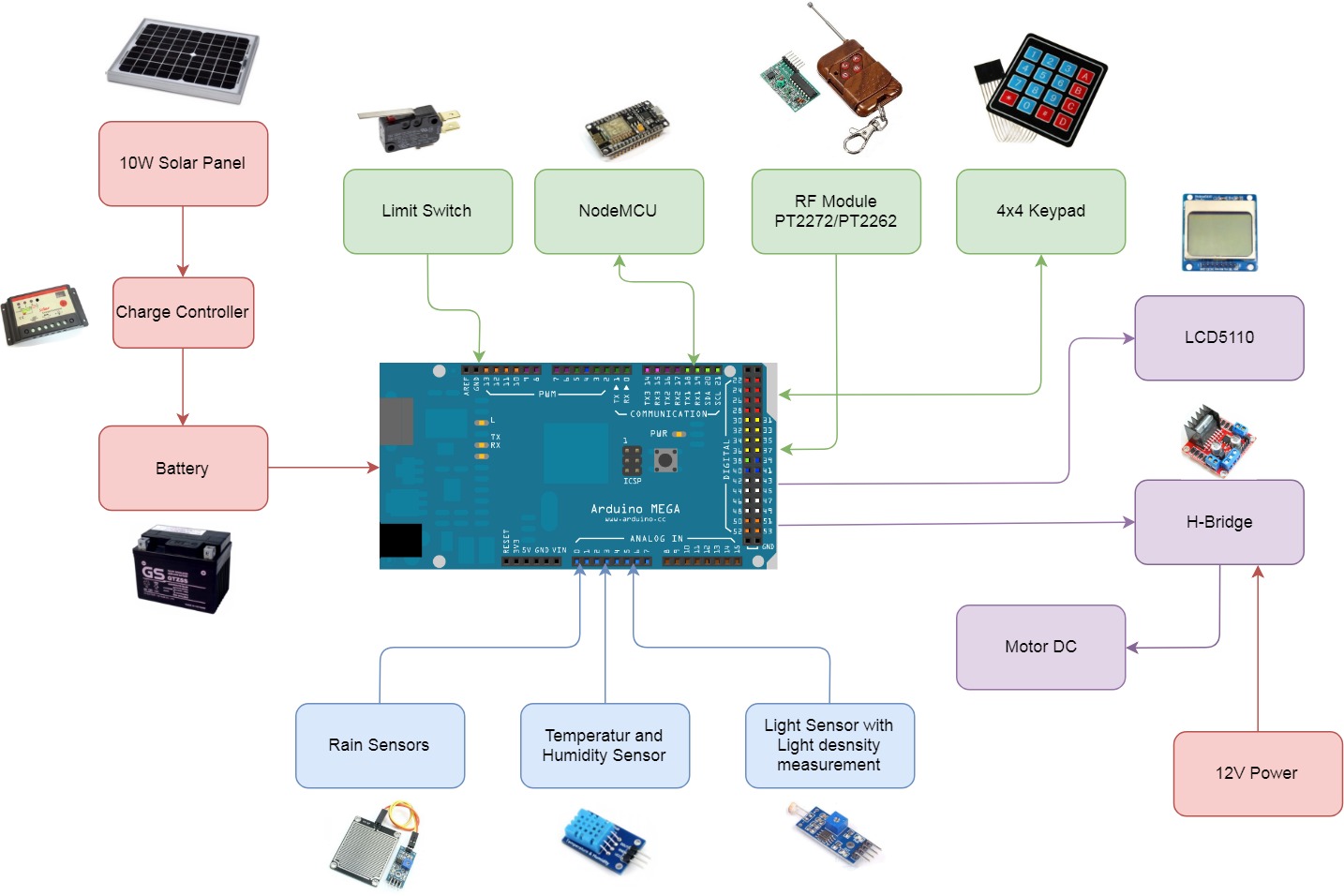


Figure 4: System block diagram

#### IMG_256Rain Sensor

Figure 5: Rain sensor

**Overview**

Rain Sensor is used to detect water levels, rain, or watery environments. The rain sensor is placed outdoors to test the rain, thereby transmitting the control signal to the relay.

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Rain sensor plate size | 54 x 40 mm |
| Board PCB size | 30 x 16 mm |
| Voltage | 5V |
| Led | Power (green) Rain (red) |
| Signal | Analog(AO) Digital(DO) |
| LM 358 | convert AO to DO |

Table 12: Specifications for rain sensor

#### [IMG_256](https://www.google.com.vn/url?sa=i%26rct=j%26q=%26esrc=s%26source=images%26cd=%26cad=rja%26uact=8%26ved=2ahUKEwifk-XW2fPbAhXFMo8KHfqUAgwQjRx6BAgBEAU%26url=https:/www.researchgate.net/figure/Arduino-MEGA-2560_fig1_281538436%26psig=AOvVaw1kQWriB-UurUx9QZDhHp1d%26ust=1530183907204330)Arduino Mega 2560 R3

Figure 6: Arduino Mega 2650 R3

**Overview**

Arduino Mega 2560 R3 is a microcontroller board based on the ATmega2560. This board has 54 I / O pins (14 pins PWM), 16 analog inputs, 4 UARTs (serial port hardware), 16 MHz quartz, USB port, and a reset button. The board has everything needed to support microcontrollers.

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Microcontroller | ATmega2560 |
| Operating voltage | 5 V |
| Input voltage (recommend) | 7-12 V |
| DC current per I/O pin | 40 mA |
| DC current for 3.3V pin | 50 mA |
| Flash memory | 256 KB of which 8 KB used by bootloader. |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock speed | 16 HZ |

Table 13: Specifications for Arduino mega 2560

#### [IMG_256](https://www.google.com.vn/url?sa=i%26rct=j%26q=%26esrc=s%26source=images%26cd=%26cad=rja%26uact=8%26ved=2ahUKEwjRyqvp4fPbAhVJuo8KHf0gDqwQjRx6BAgBEAU%26url=https:/www.smart-prototyping.com/DHT11-Humidity-and-Temperature-Sensor-Module%26psig=AOvVaw09lUI_I06gUL9HmbYYFbbd%26ust=1530186102692603)Humidity & Temperature DHT11

Figure 7: Module DHT11

**Overview**

Temperature and humidity sensors The DHT11 is a basic and cheap sensor, which is suitable for basic data acquisition applications. The DHT11 sensor has two parts, a humidity sensor capacitance and a thermal resistance. Output data of the DHT sensor is digital signal.

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Size | 28x12x10 mm |
| Operating voltage | 5 VDC |
| Temperature | Range of measure: 0 - 50°C  Error: ± 2°C |
| Humidity | Range of measure: 20 - 80%RH  Error: ± 5% |
| Maximum sampling frequency | 1Hz |

Table 14: Specifications for DHT11

#### [IMG_256](https://www.google.com.vn/url?sa=i%26rct=j%26q=%26esrc=s%26source=images%26cd=%26cad=rja%26uact=8%26ved=2ahUKEwic8oaE6fPbAhUVTn0KHRNJAXsQjRx6BAgBEAU%26url=http:/shop.wtihk.com/index.php?route=product/product%26product_id=111%26psig=AOvVaw0jl3rLQJ_IMq9mloF7jYbA%26ust=1530188066901462)Light Sensor BH1750

Figure 8: Module Light sensor BH1750

**Overview**

The BH1750 light sensor is a light sensor with a 16-bit integrated AD converter in the chip and can output data directly in digital form. BH1750 is much more accurate than the use of optical sensor to measure the intensity of light with variable data on voltage resulting in high error. BH1750 gives out direct data with the unit form is LUX, which does not need to calculate conversion through the transfer I2C.

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Size | 21x16x3.3 mm |
| Input voltage | 3 - 5 VDC |
| Range of measure | 1 - 65535 lux |
| Protocol | I2C |

Table 15: Specifications for BH1750

#### [IMG_256](https://www.google.com.vn/url?sa=i%26rct=j%26q=%26esrc=s%26source=images%26cd=%26cad=rja%26uact=8%26ved=2ahUKEwjboKb49fPbAhXBq48KHTO3APoQjRx6BAgBEAU%26url=http:/www.electrodragon.com/product/nokia-5110-lcd-pcd8544-driver-chip/%26psig=AOvVaw1RSx28rfCMX2l2jXmnp7uF%26ust=1530191552224541)LCD Nokia 5110

Figure 9: Module LCD5110

**Overview**

The Nokia 5110 is a basic graphic LCD screen for lots of applications. It was originally intended to be used as a cell phone screen. This one is mounted on an easy to solder PCB. It uses the PCD8544 controller, which is the same used in the Nokia 3310 LCD. The PCD8544 is a low power CMOS LCD controller/driver, designed to drive a graphic display of 48 rows and 84 columns. All necessary functions for the display are provided in a single chip, including on-chip generation of LCD supply and bias voltages, resulting in a minimum of external components and low power consumption. The PCD8544 interfaces to microcontrollers through a serial bus interface

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Resolution | 84x48 pixels |
| Operating voltage | 3 - 5 VDC |
| Potentiometer | 1K |
| Resistor (2) | 51 Ohm |
| Diode Zener (2) | 3 V |

Table 16: Specifications for LCD5110

#### [IMG_256](https://www.google.com.vn/url?sa=i%26rct=j%26q=%26esrc=s%26source=images%26cd=%26cad=rja%26uact=8%26ved=2ahUKEwjCgMSL__PbAhWBo48KHcHsD0kQjRx6BAgBEAU%26url=https:/hobbytronics.pk/product/membrane-4x4-matrix-keypad/%26psig=AOvVaw2tq59Bhbmd_BBqNj7vqfST%26ust=1530193999287857)Matrix Keypad (4x4)

Figure 10: 4x4 Matrix keypad

**Overview**

* Ultra-thin design
* Adhesive backing
* Excellent price/performance ratio
* Easy interface to any microcontroller

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Maximum Rating | 24 VDC, 30 mA |
| Interface | 8-pin access to 4x4 matrix |
| Operating temperature | 0 to 50°C |
| Size | Keypad: 6.9 x 7.6 cm  Cable: 2.0 x 8.8 cm |

Table 17: Specifications for 4x4 matrix keypad

#### Limit Switches

Figure 11: Limit switch

**Overview**

Limit switches commonly found in computer mouse connectors. With the advantages of small size, high mechanical efficiency and low price, it should be used widely in 3D printers, CNC machines, computer mouse buttons.

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Maximum Rating | 125V, 3A |
| Mechanical life | 100 000 times |
| Number of legs | 3 |
| Contact | NO - COM - NC |

Table 18: Specifications for limit switch

#### D:\Download\Chrome\dcMotor.pngDC Motor GA37 125RPM

Figure 12: DC Motor GA37

**Overview**

DC motor is the most commonly type which is used for the simple robot setup. It has medium quality and price with the easy accessible feature, which brings cost savings and convenience to the user

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Operating voltage | 12VDC |
| Frequency | 125 rpm |
| Moment | 20 kg.cm |

Table 19: Specifications for DC Motor GA37

#### Solar Panel

Figure 13: 10W Solar Panel

**Overview**

* Solar panel convert optical energy from the sun into electricity. The 25-year performance guarantee ensures maximum performance for the highest system efficiency.
* This panel meets strict quality standards of IEC, UL, CE, TUV, ETL, PV Cycle, MCS, BBA, Safety class II.
* Longevity Solar panels 30 years to 50 years.
* Aluminum frame shape, easy to install and beautiful design, modern appearance

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Maximum Power | 10 W |
| Maximum Power Voltage | 17.07 V |
| Maximum Power Current | 0.58 A |
| Open Circuit Voltage | 21.24 V |
| Short Circuit Current | 0.63 A |
| Size of Module | 357 ×247 × 25 mm |
| Weight | 1.2 Kg |

Table 20: Specifications for solar panel

#### Solar Charge Controller

Figure 14: Solar Charge Controller

**Overview**

* It is a device that regulates the charging of the battery, protects the battery against overcharging and discharges too deeply to:
  + Increase the life of the battery.
  + Helps the solar cell system to be effective and lasting..
* The controller also shows the charging status of the solar panel to the battery to help users control the load. It also performs over-voltage protection (> 13.8V) or low voltage (<10.5v).
* Only use with 12/24V battery and solar panel

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Maximum Power | 240 W |
| Maximum Power Current | 10 A |
| Size | 133 x 70 mm |
| Weight | 150 g |
| Led SUN | OFF: not charging ON: charging FLASH: fully charged |
| Led BAT | GREEN: full battery ORANGE: average battery RED: low battery |
| Led LOAD | OFF: no current ON: have current |

Table 21: Specifications for charge controller

#### [IMG_256](https://www.google.com.vn/url?sa=i%26rct=j%26q=%26esrc=s%26source=images%26cd=%26cad=rja%26uact=8%26ved=2ahUKEwiQ5fXMmfTbAhUQUI8KHTN3BxgQjRx6BAgBEAU%26url=http:/acquydaithang.com.vn/san-pham/ac-quy-khoi-dong--binh-kho-38/%26psig=AOvVaw0mu-N0bGRB-zOLUOOwRpi0%26ust=1530201100125795)GTZ5S-E Battery

Figure 15: GTZ5S-E Battery

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Capacity | 20 V - 3.5Ah |
| Size | 112 x 70 x 85 mm |
| Charging method | Standard: 0.4A x 5-10h Quick: 3.0A x 0.5h |

Table 22: Specifications for GTZ5S-E battery

#### Image result for nodemcuNodeMCU

Figure 16: NodeMCU v1.0

**Overview**

Development KIT contains Wi-Fi SoC ESP8266. Using for Wi-Fi Access Point or Wi-Fi Station to broadcast/receive Wi-Fi signal. NodeMCU is widely used in IoT.

**Specifications**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Operating voltage | 5VDC |
| Main IC | ESP8266 Wi-Fi SoC |
| Communication Protocol | UART, I2C, SPI  TPC/IP |
| Memory | 128KB |
| Storage | 4MB |
| CPU | 80MHz to 160MHz |
| Wi-Fi Standard | IEEE 802.11 b/g/n |

Table 23: Specifications for NodeMCU

### Communication Protocol

#### I2C Protocol

I2C (Inter-Integrated Circuit) is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems. It was invented by Philips and now it is used by almost all major IC manufacturers. Each I2C slave device needs an address – they must still be obtained from NXP (formerly Philips semiconductors).

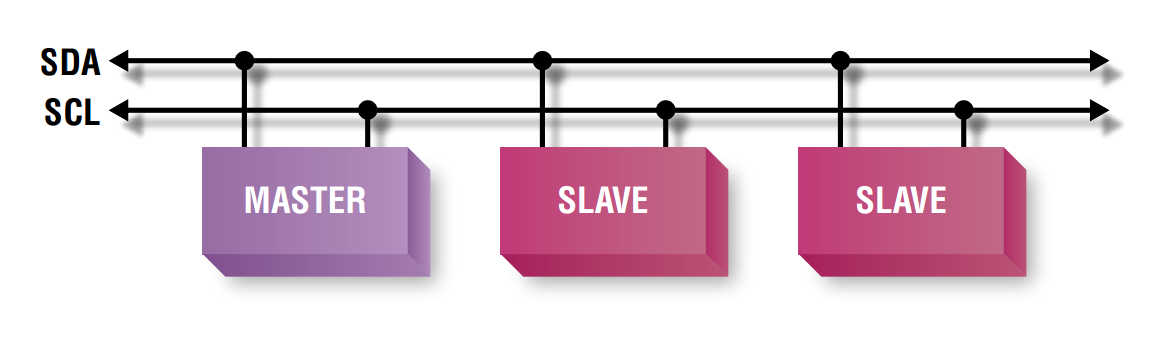


Figure 17: I2C Protocol

I2C uses only two wires: SCL (serial clock) and SDA (serial data). Both need to be pulled up with a resistor to + Vdd. There are also I2C level shifters which can be used to connect to two I2C buses with different voltages.

#### SPI Protocol

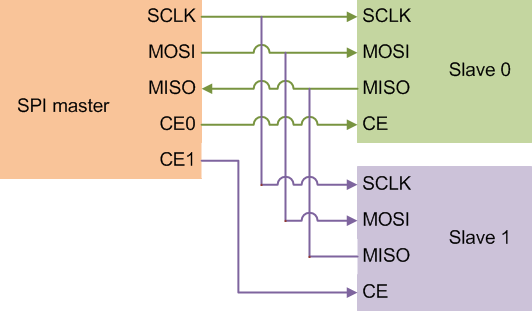
SPI (**Serial Peripheral Interface)** is a [synchronous](https://en.wikipedia.org/wiki/Synchronous_circuit) [serial communication](https://en.wikipedia.org/wiki/Serial_communication) interface specification used for short distance communication, primarily in [embedded systems](https://en.wikipedia.org/wiki/Embedded_systems). The interface was developed by [Motorola](https://en.wikipedia.org/wiki/Motorola) in the late 1980s and has become a [de facto standard](https://en.wikipedia.org/wiki/De_facto_standard). Typical applications include [Secure Digital](https://en.wikipedia.org/wiki/Secure_Digital) cards and [liquid crystal displays](https://en.wikipedia.org/wiki/Liquid_crystal_display). 

Figure 18: SPI Protocol

#### UART Protocol

UART or Universal Asynchronous Receiver Transmitter is a serial communication device that performs parallel – to – serial data conversion at the transmitter side and serial – to – parallel data conversion at the receiver side. It is universal because the parameters like transfer speed, data speed, etc. are configurable.

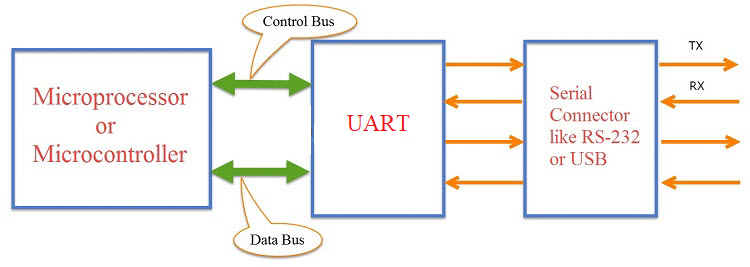


Figure 19: UART Protocol

#### HTTP Protocol

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

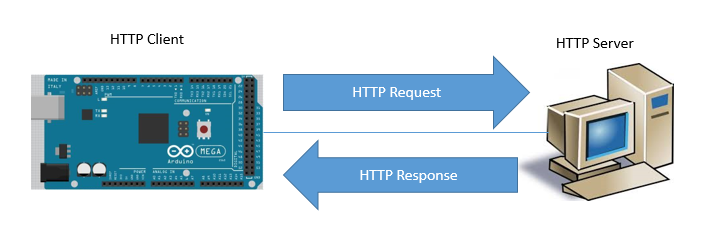
HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a website may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

Figure 20: Arduino communicates with server through HTTP Protocol

## System Attributes

### Usability

* Users can install the system easily.
* Users can learn how to use the system quickly.
* The system is divided into several components to facilitate the installation into maintenance

### Reliability

* Control signal from mobile application controls correctly the system 95% accuracy
* The system must detect the rain correctly with 99% accuracy
* Received data from temperature and humidity sensors with 90% accuracy
* Response the action from RF Remote with 90% accuracy
* Response the action from buttons with 95% accuracy

### Availability

* The mechanical component requires electrical system to work well.
* System must reply within 3 seconds from Wi-Fi command and less than 1 second from RF Remote or buttons onboard.
* System must work in cloudy day and night
* Wi-Fi broadcast range is maximum 50 meters

### Maintainability

* The system is subdivided into modules so it is easy to replace.
* Hardware components are easy to find in the market or DIY.

### Portability

* Easy to construct
* Mobility
* User can use the mobile application on devices running Android 4.0 or later

### Performance

* Fast signal response, less than 3000ms.
* Fast system response, less than 1000ms
* System can handle 1000 requests at one time

### Security

* Each role of user has a specific permission to interact with the system.
* System always checks for authorization and authentication before doing anything.
* Only Administrator can grant permission to other roles.
* System owner can only control their own systems
* Only system owner can only control their systems

# 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 screens design
* Section 6: Describe a fully attributed ERD
* Section 7: Describe algorithms

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

### D:\Capstone Document\Diagrams\Web API Architecture Diagram.jpg2.1 API Web Server Architectural Design

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 web service 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

#### D:\Capstone Document\Diagrams\Android App Overview Architecture Diagram.jpgAndroid Application Overview Architecture

From the overview we can see how the React Native Application works. First from Android Application, React Native Javascript Library and Fetch API is wrapper by ReactJS Library. Therefore, you can code React Native like ReactJS with Web API support (Fetch API). However, to run ReactJS we need a Javascript runtime/engine. So that we need a Javascript engine wraps ReactJS to actually run Javascript code and React. Due to the power that React Native can use native library written in Java, Kotlin or Objective-C, Swift, the Javascript engine will be wrapped by React Native Library to handle those native libraries. Node that, we have two very similar library that is React Native and React Native Javascript. To know the difference, the React Native written in native code while React Native Javascript is written in Javascript and run on Javascript Engine with ReactJS.

In React Native App there are 3 main threads:

* UI Thread – As known as Main Thread. This is used for native Android or iOS UI rendering.
* Javascript Thread - JS thread is the thread where the logic will run. For e.g., this is the thread where the application’s Javascript code is executed, API calls are made, touch events are processed and many other. Updates to native views are batched and sent over native side at the end of each event loop in the JS thread (and are executed eventually in UI thread).
* Native Module Thread: Sometimes an app needs access to platform API, and this happens as part of native module thread.

To communicate between UI Thread and JS Thread efficiently. React Native make up something called Bridge. The bridge is the concept that provides a way for bidirectional and asynchronous communications between these two universes. What’s important here is that they are completely written in different technologies, but they are able to communicate.

#### D:\Capstone Document\Diagrams\Android Internal System Architecture.jpgAndroid 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.jpg2.3 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

# System Implementation & Testing

# Software & Hardware User’s Manual

# Acknowledgement

First and foremost, we would like to express our greatest gratitude to Mr. Nguyen Duc Loi, our supervisor, for his support and guidance during this project, especially in  
fulfilling project requirements and improving project’s quality.

In addition, we would like to send our special thanks to all FPT University Lecturers who have taught and guided us from the very beginning. This has been a great instruction in this strenuous journey.

Last, we want to say thank you to our family and friends who supported us mentally, helped us to complete this project.

During the project, we can’t avoid making mistakes. Hopefully, people can show them to us. Therefore, we can improve our product better and gain new experiences.

Thank you, once again, for all your support to my team.

With great appreciate, sincerely.

# Appendix

* Flux Architecture: <https://facebook.github.io/flux/>