|  |  |
| --- | --- |
|  | 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**

*This page is intentionally left blank*

# Table of Contents

[Table of Contents 3](#_Toc522392854)

[List of Tables 7](#_Toc522392855)

[List of Figures 12](#_Toc522392856)

[Definitions, Acronyms and Abbreviations 16](#_Toc522392857)

[A. Introduction 17](#_Toc522392858)

[1. Project Information 17](#_Toc522392859)

[2. Introduction 17](#_Toc522392860)

[3. Current Situation 17](#_Toc522392861)

[4. Problem Definition 18](#_Toc522392862)

[5. Proposed Solution 18](#_Toc522392863)

[5.1 Feature Functions 18](#_Toc522392864)

[5.2 Advantages and Disadvantages 19](#_Toc522392865)

[6. Functional Requirements 19](#_Toc522392866)

[7. Role and Responsibility 20](#_Toc522392867)

[8. Conclusion 20](#_Toc522392868)

[B. Software Project Management Plan 21](#_Toc522392869)

[1. Problem Definition 21](#_Toc522392870)

[1.1 Name of this Capstone project 21](#_Toc522392871)

[1.2 Problem Abstract 21](#_Toc522392872)

[1.3 Project Overview 21](#_Toc522392873)

[2. Project Organization 24](#_Toc522392874)

[2.1 Software Process Model 24](#_Toc522392875)

[2.2 Roles and Responsibility 25](#_Toc522392876)

[2.3 Tools and Techniques 26](#_Toc522392877)

[3. Project Management Plan 27](#_Toc522392878)

[3.1 System Development Life-cycle 27](#_Toc522392879)

[3.2 Plan Detail 29](#_Toc522392880)

[4. Coding Convention 32](#_Toc522392881)

[C. Software & Hardware Requirement Specification 34](#_Toc522392882)

[1. User Requirement Specification 34](#_Toc522392883)

[2. System Requirement Specification 34](#_Toc522392884)

[2.1 External Interface Requirement 34](#_Toc522392885)

[2.2 System Overview Usecase 36](#_Toc522392886)

[2.3 List of Usecases 38](#_Toc522392887)

[3. Hardware Requirement Specification 78](#_Toc522392888)

[3.1 Hardware Interface 78](#_Toc522392889)

[3.2 Communication Protocol 91](#_Toc522392890)

[4. System Attributes 94](#_Toc522392891)

[4.1 Usability 94](#_Toc522392892)

[4.2 Reliability 94](#_Toc522392893)

[4.3 Availability 94](#_Toc522392894)

[4.4 Maintainability 94](#_Toc522392895)

[4.5 Portability 94](#_Toc522392896)

[4.6 Performance 94](#_Toc522392897)

[4.7 Security 95](#_Toc522392898)

[5. Conceptual Diagram 96](#_Toc522392899)

[D. Software & Hardware Design Description 97](#_Toc522392900)

[1. Design Overview 97](#_Toc522392901)

[2. System Architectural Design 98](#_Toc522392902)

[2.1 API Web Server Architectural Design 98](#_Toc522392903)

[2.2 Android Application Architectural Design 101](#_Toc522392904)

[2.3 Hardware System Architecture 105](#_Toc522392905)

[3. Component Diagram 107](#_Toc522392906)

[4. Detailed Description 109](#_Toc522392907)

[4.1 Class Diagram 109](#_Toc522392908)

[4.2 Class Diagram Explanation 116](#_Toc522392909)

[4.3 Interaction Diagram 133](#_Toc522392910)

[5. Interface 141](#_Toc522392911)

[5.1 Guest Interface 141](#_Toc522392912)

[5.2 User Interface 143](#_Toc522392913)

[6. Database Design 154](#_Toc522392914)

[6.1 Entity Relational Database (ERD) 154](#_Toc522392915)

[6.2 Data Dictionary 154](#_Toc522392916)

[7. Algorithms 155](#_Toc522392917)

[7.1 System Control 155](#_Toc522392918)

[E. System Implementation & Testing 157](#_Toc522392919)

[1. Introduction 157](#_Toc522392920)

[1.1 Overview 157](#_Toc522392921)

[1.2 Test Approach 157](#_Toc522392922)

[2. Database Relationship Diagram 158](#_Toc522392923)

[2.1 Physical Diagram 158](#_Toc522392924)

[2.2 Data Dictionary 161](#_Toc522392925)

[3. Performance Measures 163](#_Toc522392926)

[3.1 Control System from Android App Performances 163](#_Toc522392927)

[3.2 Control System from RF Remote/Keypad Performance 164](#_Toc522392928)

[4. Test Plan 165](#_Toc522392929)

[4.1 Features to be tested 165](#_Toc522392930)

[4.2 Feature not to be tested 165](#_Toc522392931)

[5. System Testing Test Case 166](#_Toc522392932)

[5.1 State Machine Diagram 166](#_Toc522392933)

[5.2 Android Application Test Case 169](#_Toc522392934)

[5.3 Hardware System Test Case 173](#_Toc522392935)

[5.4 API Web Server Test Case 176](#_Toc522392936)

[F. Software & Hardware User’s Manual 177](#_Toc522392937)

[1. Installation Guide 177](#_Toc522392938)

[1.1 Setup environment 177](#_Toc522392939)

[1.2 API Web Server Deployment Process 179](#_Toc522392940)

[1.3 Android Application Deployment Process 179](#_Toc522392941)

[1.4 System Controller Deployment Process 180](#_Toc522392942)

[1.5 System API Handler Deployment Process 180](#_Toc522392943)

[2. User Guide 181](#_Toc522392944)

[2.1 Hardware Configuration 181](#_Toc522392945)

[2.2 Android Application 184](#_Toc522392946)

[G. Acknowledgement 196](#_Toc522392947)

[H. Appendix 197](#_Toc522392948)

# List of Tables

[Table 1: General Roles and Responsibilities of Member 20](#_Toc522392949)

[Table 2: Hardware development environment requirement for DCDCS System 23](#_Toc522392950)

[Table 3: Software development environment requirement for DCDCS System 23](#_Toc522392951)

[Table 4: Roles and responsibilities 25](#_Toc522392952)

[Table 5: Tools and techniques 26](#_Toc522392953)

[Table 6: Project task planning 28](#_Toc522392954)

[Table 7: Plain Detail - Requirement Analysis 29](#_Toc522392955)

[Table 8: Plain Detail - Design 30](#_Toc522392956)

[Table 9: Plain Detail – Implementation 31](#_Toc522392957)

[Table 10: Plain Detail –Testing 31](#_Toc522392958)

[Table 11: Plain Detail –Maintenance 31](#_Toc522392959)

[Table 12: USE CASE – DCDCS\_GU - Login 39](#_Toc522392960)

[Table 13: USE CASE – DCDCS\_US01 – Stop dryer 42](#_Toc522392961)

[Table 14: USE CASE – DCDCS\_US02 - Start dryer 43](#_Toc522392962)

[Table 15: USE CASE – DCDCS\_US03 - Setup dryer timer 45](#_Toc522392963)

[Table 16: USE CASE – DCDCS\_US04 - Control DC to dry clothes 46](#_Toc522392964)

[Table 17: USE CASE – DCDCS\_US05 - Control DC to collect clothes 47](#_Toc522392965)

[Table 18: USE CASE – DCDCS\_US06 - Change controlling device 49](#_Toc522392966)

[Table 19: USE CASE – DCDCS\_US07 - View system information 50](#_Toc522392967)

[Table 20: USE CASE – DCDCS\_US08 - View user information 52](#_Toc522392968)

[Table 21: USE CASE – DCDCS\_US09 - Change user information 53](#_Toc522392969)

[Table 22: USE CASE – DCDCS\_US10 - Change user password 55](#_Toc522392970)

[Table 23: USE CASE – DCDCS\_US11 - Logout 57](#_Toc522392971)

[Table 24: USE CASE – DCDCS\_SU01 - Turn on Power 59](#_Toc522392972)

[Table 25: USE CASE – DCDCS\_SU02 - Turn off power 60](#_Toc522392973)

[Table 26: USE CASE – DCDCS\_SU03 - Stop dryer 62](#_Toc522392974)

[Table 27: USE CASE – DCDCS\_SU04 - Start dryer 63](#_Toc522392975)

[Table 28: USE CASE – DCDCS\_SU05 - Set dryer timer 64](#_Toc522392976)

[Table 29: USE CASE – DCDCS\_SU06 - View system information 65](#_Toc522392977)

[Table 30: USE CASE – DCDCS\_SU07 - Pause DC 67](#_Toc522392978)

[Table 31: USE CASE – DCDCS\_SU08 - Control dc to dry clothes 68](#_Toc522392979)

[Table 32: USE CASE – DCDCS\_SU09 - Control dc to collect clothes 69](#_Toc522392980)

[Table 33: USE CASE – DCDCS\_SU10 - Connect to Wi-Fi 71](#_Toc522392981)

[Table 34: USE CASE – DCDCS\_SYS01 - Control DC to collect clothes 72](#_Toc522392982)

[Table 35: USE CASE – DCDCS\_SYS02 - Show information 74](#_Toc522392983)

[Table 36: USE CASE – DCDCS\_SYS03 - Broadcast Wi-Fi 75](#_Toc522392984)

[Table 37: USE CASE – DCDCS\_SYS04 - Send HTTP request 76](#_Toc522392985)

[Table 38: USE CASE – DCDCS\_SYS05 - Get HTTP response 77](#_Toc522392986)

[Table 39: Specifications for rain sensor 79](#_Toc522392987)

[Table 40: Specifications for Arduino mega 2560 81](#_Toc522392988)

[Table 41: Specifications for DHT11 81](#_Toc522392989)

[Table 42: Specifications for BH1750 82](#_Toc522392990)

[Table 43: Specifications for LCD5110 83](#_Toc522392991)

[Table 44: Specifications for 4x4 matrix keypad 84](#_Toc522392992)

[Table 45: Specifications for limit switch 85](#_Toc522392993)

[Table 46: Specifications for DC Motor GA37 86](#_Toc522392994)

[Table 47: Specifications for solar panel 87](#_Toc522392995)

[Table 48: Specifications for charge controller 88](#_Toc522392996)

[Table 49: Specifications for GTZ5S-E battery 89](#_Toc522392997)

[Table 50: Specifications for NodeMCU 90](#_Toc522392998)

[Table 51: Data dictionary for conceptual diagram 96](#_Toc522392999)

[Table 52: Component diagram dictionary 108](#_Toc522393000)

[Table 53: API Web server class diagram dictionary 112](#_Toc522393001)

[Table 54: Hardware controller class diagram dictionary 115](#_Toc522393002)

[Table 55: Attribute dictionary for UserModel 116](#_Toc522393003)

[Table 56: Method dictionary for UserModel 116](#_Toc522393004)

[Table 57: Attribute dictionary for ModelModel 117](#_Toc522393005)

[Table 58: Method dictionary for ModelModel 117](#_Toc522393006)

[Table 59: Attribute dictionary for MessageModel 118](#_Toc522393007)

[Table 60: Method dictionary for MessageModel 118](#_Toc522393008)

[Table 61: Attribute dictionary for CustomerModel 119](#_Toc522393009)

[Table 62: Method dictionary for CustomerModel 119](#_Toc522393010)

[Table 63: Attribute dictionary for ProductModel 120](#_Toc522393011)

[Table 64: Method dictionary for ProductModel 120](#_Toc522393012)

[Table 65: Method dictionary for Mongoose 121](#_Toc522393013)

[Table 66: Method dictionary for UserController 121](#_Toc522393014)

[Table 67: Method dictionary for ModelController 122](#_Toc522393015)

[Table 68: Method dictionary for MessageController 122](#_Toc522393016)

[Table 69: Method dictionary for CustomerController 123](#_Toc522393017)

[Table 70: Method dictionary for ProductController 123](#_Toc522393018)

[Table 71: Method dictionary for Router 124](#_Toc522393019)

[Table 72: Method dictionary for Auth 124](#_Toc522393020)

[Table 73: Attribute dictionary for SystemController 125](#_Toc522393021)

[Table 74: Method dictionary for SystemController 125](#_Toc522393022)

[Table 75: Attribute dictionary for DryerController 126](#_Toc522393023)

[Table 76: Method dictionary for DryerController 126](#_Toc522393024)

[Table 77: Method dictionary for WifiHandler 126](#_Toc522393025)

[Table 78: Attribute dictionary for DCMotor 127](#_Toc522393026)

[Table 79: Method dictionary for DCMotor 127](#_Toc522393027)

[Table 80: Method dictionary for LightSensorHandler 127](#_Toc522393028)

[Table 81: Attribute dictionary for LCDHandler 128](#_Toc522393029)

[Table 82: Method dictionary for LCDHandler 128](#_Toc522393030)

[Table 83: Attribute dictionary for SwitchHandler 128](#_Toc522393031)

[Table 84: Method dictionary for SwitchHandler 128](#_Toc522393032)

[Table 85: Attribute dictionary for RFHandler 129](#_Toc522393033)

[Table 86: Method dictionary for RFHandler 129](#_Toc522393034)

[Table 87: Attribute dictionary for KeypadHandler 129](#_Toc522393035)

[Table 88: Method dictionary for KeypadHandler 129](#_Toc522393036)

[Table 89: Attribute dictionary for RainSensorHandler 130](#_Toc522393037)

[Table 90: Method dictionary for RainSensorHandler 130](#_Toc522393038)

[Table 91: Method dictionary for Central Controller 130](#_Toc522393039)

[Table 92: Attribute dictionary for BHT1750 131](#_Toc522393040)

[Table 93: Method dictionary for BHT1750 131](#_Toc522393041)

[Table 94: Attribute dictionary for Wire 131](#_Toc522393042)

[Table 95: Method dictionary for Wire 131](#_Toc522393043)

[Table 96: Attribute dictionary for DHT 132](#_Toc522393044)

[Table 97: Method dictionary for DHT 132](#_Toc522393045)

[Table 98: <Guess> Login screen fields table 142](#_Toc522393046)

[Table 99: <Guess> Login screen buttons/hyperlinks table 142](#_Toc522393047)

[Table 100: <User> Home screen fields table 144](#_Toc522393048)

[Table 101: <User> Home screen buttons/hyperlinks 144](#_Toc522393049)

[Table 102: <User> DC Motor control screen fields 146](#_Toc522393050)

[Table 103: <User> DC Motor control screen buttons/hyperlinks 146](#_Toc522393051)

[Table 104: <User> Setup and control dryer screen fields 148](#_Toc522393052)

[Table 105: <User> Setup and control dryer screen buttons/hyperlinks 148](#_Toc522393053)

[Table 106: <User> Select controlling product screen fields 150](#_Toc522393054)

[Table 107: <User> Select controlling product screen buttons/hyperlinks 150](#_Toc522393055)

[Table 108: <User> User's profile screen fields 152](#_Toc522393056)

[Table 109: <User> User's profile buttons/hyperlinks 153](#_Toc522393057)

[Table 110: Entity diagram data dictionary 154](#_Toc522393058)

[Table 111: Description of physical database 161](#_Toc522393059)

[Table 112: Attribute description for physical database 163](#_Toc522393060)

[Table 113: API request performance 163](#_Toc522393061)

[Table 114: API response performance 163](#_Toc522393062)

[Table 115: RF Remote/Keypad performance 164](#_Toc522393063)

[Table 116: Get system information test cases 169](#_Toc522393064)

[Table 117: Get customer information test cases 170](#_Toc522393065)

[Table 118: Control dc motor from mobile app test cases 170](#_Toc522393066)

[Table 119: Control dryer from mobile app test cases 170](#_Toc522393067)

[Table 120: Change control device test cases 171](#_Toc522393068)

[Table 121: Update user/customer information test cases 172](#_Toc522393069)

[Table 122: RF Remote and Keypad control test cases 174](#_Toc522393070)

[Table 123: System auto control test cases 175](#_Toc522393071)

[Table 124: Wi-Fi configuration test cases 175](#_Toc522393072)

[Table 125: API Web Server test cases 176](#_Toc522393073)

[Table 126: Hardware system requirement 177](#_Toc522393074)

[Table 127: Server hardware requirement 177](#_Toc522393075)

[Table 128: Android hardware requirement 178](#_Toc522393076)

[Table 129: System requirement 178](#_Toc522393077)

[Table 130: Connect to local Wi-Fi part 1 - Description 181](#_Toc522393078)

[Table 131: Connect to local Wi-Fi part 2 - Description 182](#_Toc522393079)

[Table 132: Connect to local Wi-Fi part 3 183](#_Toc522393080)

[Table 133: Dry clothes part 1 - Description 184](#_Toc522393081)

[Table 134: Dry clothes part 2 - Description 185](#_Toc522393082)

[Table 135: Collect clothes part 1 - Description 186](#_Toc522393083)

[Table 136: Collect clothes part 2 - Description 187](#_Toc522393084)

[Table 137: Setup timer and start dryer part 1 - Description 188](#_Toc522393085)

[Table 138: Setup timer and start dryer part 2 - Description 189](#_Toc522393086)

[Table 139: Stop dryer part 1 - Description 190](#_Toc522393087)

[Table 140: Stop dryer part 2 - Description 191](#_Toc522393088)

[Table 141: Change control device part 1 - Description 192](#_Toc522393089)

[Table 142: Change control device part 2 - Description 193](#_Toc522393090)

[Table 143: Change user information part 1 - Description 194](#_Toc522393091)

[Table 144: Change user information part 2 – Description 195](#_Toc522393092)

# List of Figures

[Figure 1: Waterfall methodology 24](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393093)

[Figure 2: Hardware system overview usecase diagram 36](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393094)

[Figure 3: Android application overview usecase diagram 37](#_Toc522393095)

[Figure 4: <Guest> Overview Usecase 38](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393096)

[Figure 5: <Android User> Overview Usecase 40](#_Toc522393097)

[Figure 6: <Android User> Stop dryer 41](#_Toc522393098)

[Figure 7: <Android User> Start dryer 42](#_Toc522393099)

[Figure 8: <Android User> Setup dryer timer 44](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393100)

[Figure 9: <Android User> Control DC to dry clothes 45](#_Toc522393101)

[Figure 10: <Android User> Control DC to collect clothes 46](#_Toc522393102)

[Figure 11: <Android User> Change controlling device 48](#_Toc522393103)

[Figure 12: <Android User> View system information 49](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393104)

[Figure 13: <Android User> View user information 50](#_Toc522393105)

[Figure 14: <Android User> Change user information 52](#_Toc522393106)

[Figure 15: <Android User> Change user password 53](#_Toc522393107)

[Figure 16: <Android User> Logout 55](#_Toc522393108)

[Figure 17: <System User> Overview Usecase 58](#_Toc522393109)

[Figure 18: <System User> Turn on power 59](#_Toc522393110)

[Figure 19: <System User> Turn off power 60](#_Toc522393111)

[Figure 20: <System User> Stop drying 61](#_Toc522393112)

[Figure 21: <System User> Start dryer 62](#_Toc522393113)

[Figure 22: <System User> Set dryer timer 63](#_Toc522393114)

[Figure 23: <System User> View system information 64](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393115)

[Figure 24: <System User> Pause DC Motor 66](#_Toc522393116)

[Figure 25: <System User> Control DC to dry clothes 67](#_Toc522393117)

[Figure 26: <System User> Control DC to collect clothes 68](#_Toc522393118)

[Figure 27: <System User> Connect to Wi-Fi 70](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393119)

[Figure 28: <System> Overview Usecase 71](#_Toc522393120)

[Figure 29: <System> Control DC to collect clothes 71](#_Toc522393121)

[Figure 30: <System> Show information 73](#_Toc522393122)

[Figure 31: <System> Broadcast Wi-Fi 74](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393123)

[Figure 32: <System> Send HTTP Request 75](#_Toc522393124)

[Figure 33: <System> Get HTTP Response 76](#_Toc522393125)

[Figure 34: System block diagram 78](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393126)

[Figure 35: Rain sensor 79](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393127)

[Figure 36: Arduino Mega 2650 R3 80](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393128)

[Figure 37: Module DHT11 81](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393129)

[Figure 38: Module Light sensor BH1750 82](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393130)

[Figure 39: Module LCD5110 83](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393131)

[Figure 40: 4x4 Matrix keypad 84](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393132)

[Figure 41: Limit switch 85](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393133)

[Figure 42: DC Motor GA37 86](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393134)

[Figure 43: 10W Solar Panel 87](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393135)

[Figure 44: Solar Charge Controller 88](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393136)

[Figure 45: GTZ5S-E Battery 89](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393137)

[Figure 46: NodeMCU v1.0 90](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393138)

[Figure 47: I2C Protocol 91](#_Toc522393139)

[Figure 48: SPI Protocol 92](#_Toc522393140)

[Figure 49: UART Protocol 92](#_Toc522393141)

[Figure 50: Arduino communicates with server through HTTP Protocol 93](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393142)

[Figure 51: Conceptual diagram 96](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393143)

[Figure 52: System overview architecture 98](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393144)

[Figure 53: API Web server architecture 99](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393145)

[Figure 54: Android application overview architecture 101](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393146)

[Figure 55: Android application internal architecture 103](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393147)

[Figure 56: Hardware system architecture 105](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393148)

[Figure 57: Component diagram 107](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393149)

[Figure 58: API Web Server Class Diagram Part 1 109](#_Toc522393150)

[Figure 59: API Web Server Class Diagram Part 1 109](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393151)

[Figure 60: API Web Server Class Diagram Part 2 110](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393152)

[Figure 61: API Web Server Class Diagram Part 3 111](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393153)

[Figure 62: Hardware system controller class diagram part 1 113](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393154)

[Figure 63: Hardware system controller class diagram part 2 114](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393155)

[Figure 64: Control system with android app sequence diagram 133](#_Toc522393156)

[Figure 65: Update system information sequence diagram 134](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393157)

[Figure 66: Control DC activity diagram 135](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393158)

[Figure 67: Control Dryer activity diagram 136](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393159)

[Figure 68: Auto control activity diagram 137](#_Toc522393160)

[Figure 69: Determine action based on sensors’ data activity diagram 138](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393161)

[Figure 70: Determine dc motor action activity diagram 139](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393162)

[Figure 71: Determine dryer action activity diagram 140](#_Toc522393163)

[Figure 72: <Guess> Login screen 141](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393164)

[Figure 73: <User> Home screen 143](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393165)

[Figure 74: <User> Control DC Motor screen 145](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393166)

[Figure 75: <User> Setup and control dryer screen 147](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393167)

[Figure 76: <User> Select controlling product screen 149](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393168)

[Figure 77: <User> User's profile screen 151](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393169)

[Figure 78: Entity Relational Database 154](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393170)

[Figure 79: System Control overview flowchart 156](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393171)

[Figure 80: Manage message to communicate between clients 158](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393172)

[Figure 81: Manage user 159](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393173)

[Figure 82: Manage customer and product information 160](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393174)

[Figure 83: Control DC Motor State machine diagram 166](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393175)

[Figure 84: Control dryer State machine diagram 167](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393176)

[Figure 85: Control system State machine diagram 168](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393177)

[Figure 86: Connect to local Wi-Fi part 1 181](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393178)

[Figure 87: Connect to local Wi-Fi part 2 182](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393179)

[Figure 88: Connect to local Wi-Fi part 3 183](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393180)

[Figure 89: Dry clothes part 1 184](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393181)

[Figure 90: Dry clothes part 2 185](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393182)

[Figure 91: Collect clothes part 1 186](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393183)

[Figure 92: Collect clothes part 2 187](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393184)

[Figure 93: Setup timer and start dryer part 1 188](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393185)

[Figure 94: Setup timer and start dryer part 2 189](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393186)

[Figure 95: Stop dryer part 1 190](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393187)

[Figure 96: Stop dryer part 2 191](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393188)

[Figure 97: Change control device part 1 192](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393189)

[Figure 98: Change control device part 2 193](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393190)

[Figure 99: Change user information part 1 194](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393191)

[Figure 100: Change user information part 2 195](file:///D:\Capstone%20Document\Capstone%20Project%20Document.docx#_Toc522393192)

# Definitions, Acronyms and Abbreviations

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

# Introduction

## Project Information

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

## Introduction

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.

## 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.

## 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

## 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:

### 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 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

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

### Project Overview

#### 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.

#### The Proposed System

According to the technology researches, we found that the simple rain sensor and Wi-Fi module ESP8266 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 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

#### 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

#### 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:

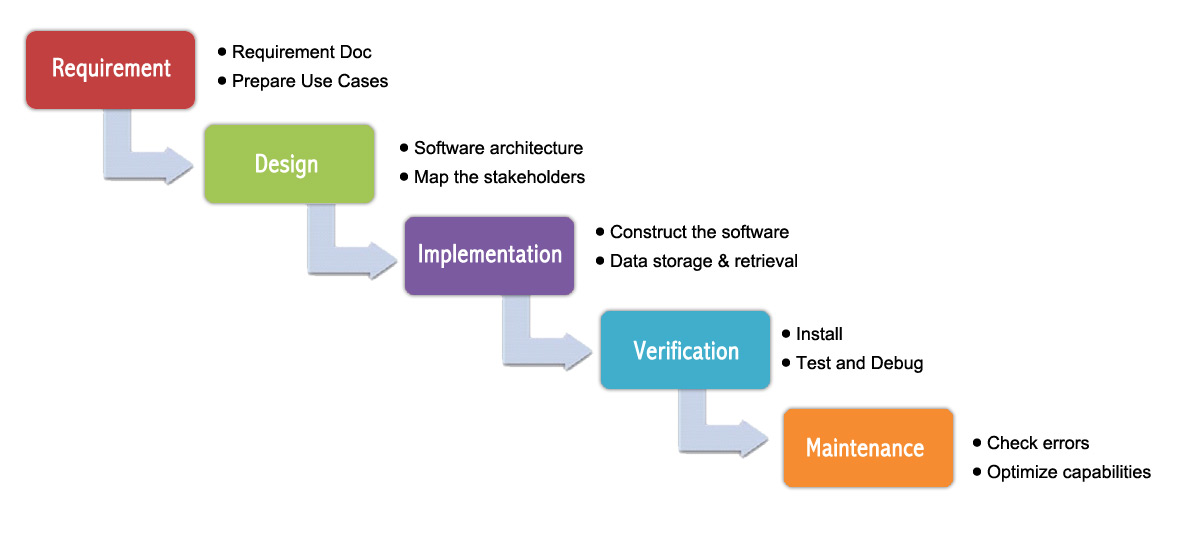
* This project is 4-months long due to the FPT University Capstone Project timeline, which can be consider a short project.
* Based on researches and current clarified clothes drying system, the requirements of this project are stable, clear, fixed and well-understood by all team members.
* The Modified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

### Roles and Responsibility

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

Table 4: Roles and responsibilities

### Tools and Techniques

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

Table 5: Tools and techniques

## Project Management Plan

### System Development Life-cycle

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

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

Table 6: Project task planning

### Plan Detail

#### Phrase 1: Requirement Analysis

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

Table 7: Plain Detail - Requirement Analysis

#### Phrase 2: Design

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

Table 8: Plain Detail - Design

#### Phrase 3: Implementation

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

Table 9: Plain Detail – Implementation

#### Phrase 4: Testing

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

Table 10: Plain Detail –Testing

#### Phrase 5: Maintenance

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

Table 11: Plain Detail –Maintenance

## 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 API web server 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\image\Usecase Diagram\DCDCS System Use Case.jpgHardware System Usecase

Figure 2: Hardware system overview usecase diagram

#### Android Application Usecase

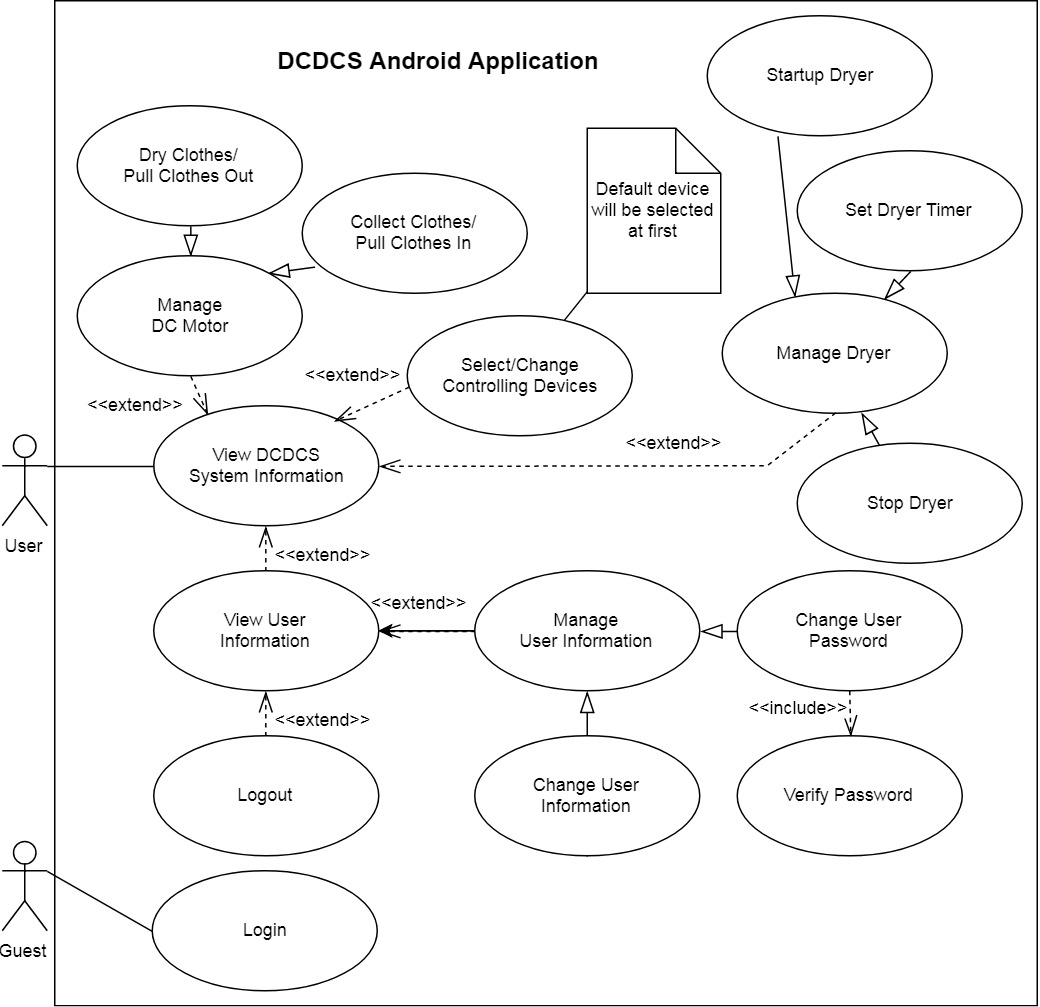


Figure 3: Android application overview usecase diagram

### List of Usecases

#### login<Guest> Overview Usecase

Figure 4: <Guest> Overview Usecase

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_GU | | | |
| Use Case No. | DCDCS\_ | **Use Case Version** | 1.0 |
| Use Case Name | Login | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * Guest   Summary:   * This use case allows actor login into application   Goal:   * Actor must be authenticated and authorized before access the system   Triggers:   * Actor tab on “LOGIN” button   Pre-conditions: N/A  Post-conditions:   * Success: System redirect to home screen * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor run application  [Exception 1] | System redirect to welcome screen include:  + DCDCS App: Label  + Tab to continue: TouchableView | | 2 | Actor tab on “Tab to continue” button  [Exception 1] | System show login screen include:  + Username: TextBox  + Password: TextBox  + LOGIN: Button  + Need a help: Button | | 3 | Actor type username and password then tab on “LOGIN” button  [Exception 1, 2, 3] | System redirect to home screen include:  + System status: Label  + System temperature: Label  + Product name: Label  + System time: Label  + System greeting: Label  + Weather: Image  + Username: TouchableView  + Dry clothes: Button or Collect clothes: Button, it bases on status of system  + Dryer setting: Button  + Change device: Button |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” | | 2 | Actor enter wrong username or wrong password | System show error message: “Invalid username or password!” | | 3 | Actor enter empty text | System show error message: “Invalid username or password!” |   Relationships: N/A  Business Rules: N/A | | | |

Table 12: USE CASE – DCDCS\_GU - Login

#### <Android User> Overview Usecase

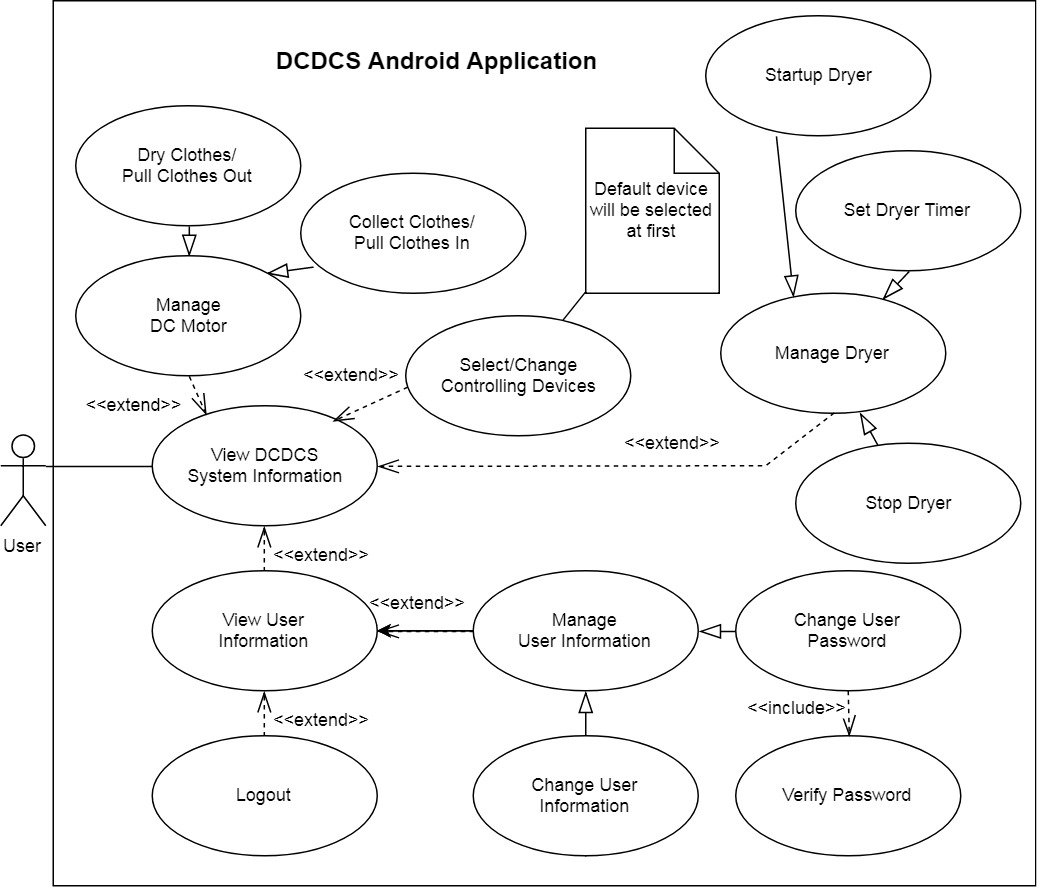


Figure 5: <Android User> Overview Usecase

##### <Android User> Stop dryer

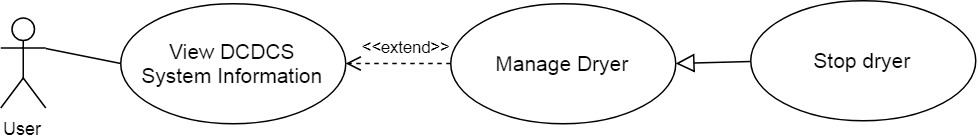


Figure 6: <Android User> Stop dryer

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US01 | | | |
| Use Case No. | DCDCS\_US01 | **Use Case Version** | 1.0 |
| Use Case Name | Stop Dryer | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor stop dryer   Goal:   * Actor can stop dryer by smart phone   Triggers:   * Actor tab on “Stop dryer” button   Pre-conditions:   * Actor must login into application * Status of system is DRYER\_ACTIVED   Post-conditions:   * Success: “Stop dryer” button change to “Start dryer” button, system show success message * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor go to home screen [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button  + Dryer setting: button  + Change device: button | | 2 | Actor tab on “Dryer setting” button  [Exception 1] | System show “Dryer setting” dialog include:  + Dryer time: textview  + Choosing timer: slider  + “Please set up the time you want to dry your clothes”: textview  + Stop dryer: button  + Cancel: button | | 3 | Actor tab on “Stop dryer” button  [Exception 1] | “Stop dryer” button change to “Start dryer” button System show success message |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” |   Relationships: Manage Dryer (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone * If system is connecting to smartphone, it will disconnect to android app. | | | |

Table 13: USE CASE – DCDCS\_US01 – Stop dryer

##### <Android User> Start dryer



Figure 7: <Android User> Start dryer

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US02 | | | |
| Use Case No. | DCDCS\_US02 | **Use Case Version** | 1.0 |
| Use Case Name | Startup Dryer | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Medium |
| Actor:   * User   Summary:   * This use case allows Actor startup dryer   Goal:   * Actor can dry clothes on raining days   Triggers:   * Actor tap on “Start dryer” button   Pre-conditions:   * Actor must login into application * Status of system is IDLING   Post-conditions:   * Success: “Start dryer” button change to “Stop dryer” button, system show success * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor go to home screen [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button  + Dryer setting: button  + Change device: button | | 2 | Actor tab on “Dryer setting” button  [Exception 1] | System show “Dryer setting” dialog include:  + Dryer time: textview  + Choosing timer: slider  + “Please set up the time you want to dry your clothes”: textview  + Start dryer: button  + Cancel: button | | 3 | Actor tab on “Start dryer” button  [Exception 1] | “Start dryer” button change to “Stop dryer” button System show success message |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” |   Relationships: Manage Dryer (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone * If system is connecting to smartphone, it will disconnect to android app. | | | |

Table 14: USE CASE – DCDCS\_US02 - Start dryer

##### android_set_dryer_timer<Android User> Setup dryer timer

Figure 8: <Android User> Setup dryer timer

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US03 | | | |
| Use Case No. | DCDCS\_US03 | **Use Case Version** | 1.0 |
| Use Case Name | Setup Dryer Timer | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Medium |
| Actor:   * Actor   Summary:   * This use case allows actor setup dryer timer appropriately   Goal:   * Actor can setup dryer timer   Triggers:   * Actor presses tab on slider and pull it   Pre-conditions:   * Actor must login into application   Post-conditions:   * Success: Value of “Dryer timer” textview changed * Fail: Value of “Dryer timer” textview is not change   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor go to home screen [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button  + Dryer setting: button  + Change device: button | | 2 | Actor tab on “Dryer setting” button  [Exception 1] | System show “Dryer setting” dialog include:  + Dryer time: textview  + Choosing timer: slider  + “Please set up the time you want to dry your clothes”: textview  + Start dryer: button  + Cancel: button | | 3 | Actor presses tab on slider and pull it | “Start dryer” button change to “Stop dryer” button System show success message |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” |   Relationships: Manage Dryer (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone * If system is connecting to smartphone, it will disconnect to android app. | | | |

Table 15: USE CASE – DCDCS\_US03 - Setup dryer timer

##### <Android User> Control DC to dry clothes



Figure 9: <Android User> Control DC to dry clothes

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US04 | | | |
| Use Case No. | DCDCS\_US04 | **Use Case Version** | 1.0 |
| Use Case Name | Dry Clothes / Pull Clothes Out | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor dry clothes by smart phone   Goal:   * Actor can dry clothes   Triggers:   * Actor tab on “Dry clothes” button   Pre-conditions:   * Actor must login to application * Status of system is IDLING   Post-conditions:   * Success: System status on application changed to DRYING * Fail: System status on application does not change   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor go to home screen | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button  + Dryer setting: button  + Change device: button | | 2 | Actor tap on “Dry clothes” button  [Exception 1] | System pull clothes out  Status of system change to DRYING  “Dry clothes” button change to “Collect clothes” button  [Exception 2, 3] |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” | | 2 | Weather is raining | System show error message: “Request failed!” | | 3 | Time is night | System show error message: “Request failed!” |   Relationships: Manage DC Motor (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone * If system is connecting to smartphone, it will disconnect to android app. | | | |

Table 16: USE CASE – DCDCS\_US04 - Control DC to dry clothes

##### <Android User> Control DC to collect clothes

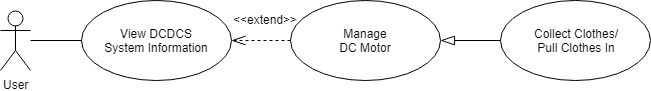


Figure 10: <Android User> Control DC to collect clothes

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US05 | | | |
| Use Case No. | DCDCS\_US05 | **Use Case Version** | 1.0 |
| Use Case Name | Collect Clothes / Pull Clothes In | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor collect clothes (pull clothes in) by smart phone   Goal:   * Actor can collect clothes (pull clothes in)   Triggers:   * Actor presses on “M button”   Pre-conditions:   * System is running stability * Motor DC is stopped * Actor must login to application   Post-conditions:   * Success: System status on application changed to IDLING * Fail: System status on application does not change   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor goes to login screen |  | | 2 | Actor presses on “M button”  [Exception 1, 2] | System pulls clothes in  [Exception 1] |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery runs out | Whole system stop working | | 2 | “M button” is broken | Motor DC is not running |   Relationships: Dry Clothes (Pull Out), Pause Motor DC  Business Rules:   * When actor use this use case, system will not receive signal from smartphone * If system is connecting to smartphone, it will disconnect to android app. | | | |

Table 17: USE CASE – DCDCS\_US05 - Control DC to collect clothes

##### android_select_change<Android User> Change controlling device

Figure 11: <Android User> Change controlling device

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US06 | | | |
| Use Case No. | DCDCS\_US06 | **Use Case Version** | 1.0 |
| Use Case Name | Select/Change Controlling Devices | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Low |
| Actor:   * User   Summary:   * This use case allows actor change user interface of each product   Goal:   * Actor can control a lot of products   Triggers:   * Actor tab on name of product   Pre-conditions:   * Actor must login into system   Post-conditions:   * Success: System redirect to home screen of new product * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor run application  [Exception 1] | System redirect to welcome screen include:  + DCDCS App: textview  + Tab to continue: buttonview | | 2 | Actor tab on “Tab to continue” button  [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button or Collect clothes: button, it bases on status of system  + Dryer setting: button  + Change device: button | | 3 | Actor tap on “Change device” button  [Exception 1] | System show “Product List” dialog include: list name of products: textview | | 4 | Actor tap on name of product  [Exception 1] | System redirect to home screen of new product |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” |   Relationships: Extend from “View DCDCS System Information” Usecase  Business Rules: N/A | | | |

Table 18: USE CASE – DCDCS\_US06 - Change controlling device

##### view<Android User> View system information

Figure 12: <Android User> View system information

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US07 | | | |
| Use Case No. | DCDCS\_US07 | **Use Case Version** | 1.0 |
| Use Case Name | View DCDCS System Information | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor view system information   Goal:   * Actor can follow operating state of system   Triggers:   * Actor tab on “Tap to continue” button in welcome screen   Pre-conditions:   * Actor must login into system   Post-conditions:   * Success: System redirect to home screen * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor run application | System redirect to welcome screen include:  + DCDCS App: textview  + Tab to continue: buttonview | | 2 | Actor tab on “Tab to continue” button  [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button or Collect clothes: button, it bases on status of system  + Dryer setting: button  + Change device: button |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” |   Relationships: Extend to use cases: +Manage DC Motor (Abstract Use Case)  +Manage Dryer (Abstract Use Case)  +View User Information  +Select/Change Controlling Devices  Business Rules: N/A | | | |

Table 19: USE CASE – DCDCS\_US07 - View system information

##### view_user_information<Android User> View user information

Figure 13: <Android User> View user information

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US08 | | | |
| Use Case No. | DCDCS\_US08 | **Use Case Version** | 1.0 |
| Use Case Name | View User Information | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Medium |
| Actor:   * User   Summary:   * This use case allows actor view their information   Goal:   * Actor can view their information   Triggers:   * Actor tab on their username in home screen   Pre-conditions:   * Actor must login into system   Post-conditions:   * Success: System redirect to profile screen * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor run application  [Exception 1] | System redirect to welcome screen include:  + DCDCS App: textview  + Tab to continue: buttonview | | 2 | Actor tab on “Tab to continue” button  [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button or Collect clothes: button, it bases on status of system  + Dryer setting: button  + Change device: button | | 3 | Actor tap on “Username” buttonview  [Exception 1] | System redirect to profile screen include:  +Username: textbox  +Password: textbox  +Name: textbox  +Address: textbox  +Phone: textbox  +Email: textbox  +Save Changes: button  +Logout: button |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” |   Relationships: Extend from “View DCDCS System Information” Usecase  Business Rules: N/A | | | |

Table 20: USE CASE – DCDCS\_US08 - View user information

##### <Android User> Change user information

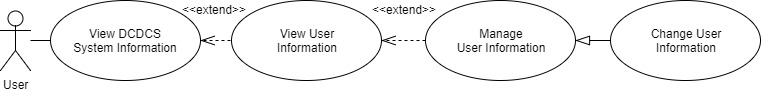


Figure 14: <Android User> Change user information

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US09 | | | |
| Use Case No. | DCDCS\_US09 | **Use Case Version** | 1.0 |
| Use Case Name | Change User Information | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Medium |
| Actor:   * User   Summary:   * This use case allows actor change their information   Goal:   * Actor can modify their information   Triggers:   * Actor tab on “Save Changes” button   Pre-conditions:   * Actor must login into system   Post-conditions:   * Success: System show success message * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor run application  [Exception 1] | System redirect to welcome screen include:  + DCDCS App: textview  + Tab to continue: buttonview | | 2 | Actor tab on “Tab to continue” button  [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button or Collect clothes: button, it bases on status of system  + Dryer setting: button  + Change device: button | | 3 | Actor tap on “Username” buttonview  [Exception 1, 2, 3, 4, 5] | System redirect to profile screen include:  +Username: textview  +Password: password regrex: “”  +Change: button  +Name: textbox  +Address: textbox  +Phone: textbox regrex: “”  +Email: textbox regrex: “”  +Save Changes: button  +Logout: button | | 4 | Actor enter their information then tab on “Save Changes’ | System show success message |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” | | 2 | Actor input empty “Phone” textbox | System show error message: “Invalid phonenumber! Phone must be: “” ” | | 3 | Actor input empty “Email” textbox | System show error message: “Invalid email! Email must be: “” ” | | 4 | Actor input empty “Name” textbox | System show error message: “Name can’t be empty” | | 5 | Actor input empty “Address” textbox | System show error message: “Address can’t be empty” |   Relationships: Extend from “View User Information”, Manage User Information (Abstract Use case)  Business Rules: N/A | | | |

Table 21: USE CASE – DCDCS\_US09 - Change user information

##### <Android User> Change user password

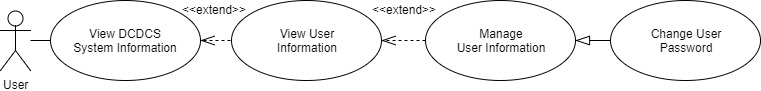


Figure 15: <Android User> Change user password

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US10 | | | |
| Use Case No. | DCDCS\_US10 | **Use Case Version** | 1.0 |
| Use Case Name | Change Password | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Medium |
| Actor:   * User   Summary:   * This use case allows actor change their password   Goal:   * Actor can improve more security   Triggers:   * Actor tab on “Change password” button   Pre-conditions:   * Actor must login into system   Post-conditions:   * Success: System show success message * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor run application  [Exception 1] | System redirect to welcome screen include:  + DCDCS App: textview  + Tab to continue: buttonview | | 2 | Actor tab on “Tab to continue” button  [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button or Collect clothes: button, it bases on status of system  + Dryer setting: button  + Change device: button | | 3 | Actor tap on “Username” buttonview  [Exception 1] | System redirect to profile screen include:  +Username: textview  +Password: password regrex: “”  +Change: button  +Name: textbox  +Address: textbox  +Phone: textbox regrex: “”  +Email: textbox regrex: “”  +Save Changes: button  +Logout: button | | 4 | Actor tab on “Change” button | System show “Change Password” dialog include:  +Current password: password  +New password: password regrex: “”  +Confirm password: password regrex: “” +Change password: button  +Cancel: button | | 5 | Actor type value then tab on “Change password” button | System show success message |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” | | 2 | Actor enter empty value to “Current password” | System show error message: “Invalid current password! Password must be: “” ” | | 3 | Actor enter empty value to “New password” or “Confirm password” | System show error message: “Invalid current password! Password must be: “” ” | | 4 | “New password” is not match to “Confirm password” | System show error message: “Password is not match” |   Relationships: Extend from “View User Information”, Manage User Information (Abstract Use case)  Business Rules: N/A | | | |

Table 22: USE CASE – DCDCS\_US10 - Change user password

##### <Android User> Logout

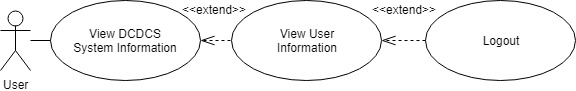


Figure 16: <Android User> Logout

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_US11 | | | |
| Use Case No. | DCDCS\_US11 | **Use Case Version** | 1.0 |
| Use Case Name | Logout | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Medium |
| Actor:   * User   Summary:   * This use case allows actor change their password   Goal:   * Actor can improve more security   Triggers:   * Actor tab on “Change password” button   Pre-conditions:   * Actor must login into system   Post-conditions:   * Success: System show success message * Fail: System show error message   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor run application  [Exception 1] | System redirect to welcome screen include:  + DCDCS App: textview  + Tab to continue: buttonview | | 2 | Actor tab on “Tab to continue” button  [Exception 1] | System redirect to home screen include:  + System status: textview  + System temperature: textview  + Product name: textview  + System time: textview  + System greeting: textview  + Weather: image  + Username: buttonview  + Dry clothes: button or Collect clothes: button, it bases on status of system  + Dryer setting: button  + Change device: button | | 3 | Actor tap on “Username” buttonview  [Exception 1] | System redirect to profile screen include:  +Username: textview  +Password: password regrex: “”  +Change: button  +Name: textbox  +Address: textbox  +Phone: textbox regrex: “”  +Email: textbox regrex: “”  +Save Changes: button  +Logout: button | | 4 | Actor tap on “Logout” button | System show confirm dialog include:  + “Are you sure to logout?” textview  + Yes: button  + Cancel: button | | 5 | Actor tap on “Yes” button | System redirect to login screen |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Lost internet connection | System show error message: “Network request failed!” |   Relationships: Extend from “View User Information”  Business Rules: N/A | | | |

Table 23: USE CASE – DCDCS\_US11 - Logout

#### <System User> Overview Usecase

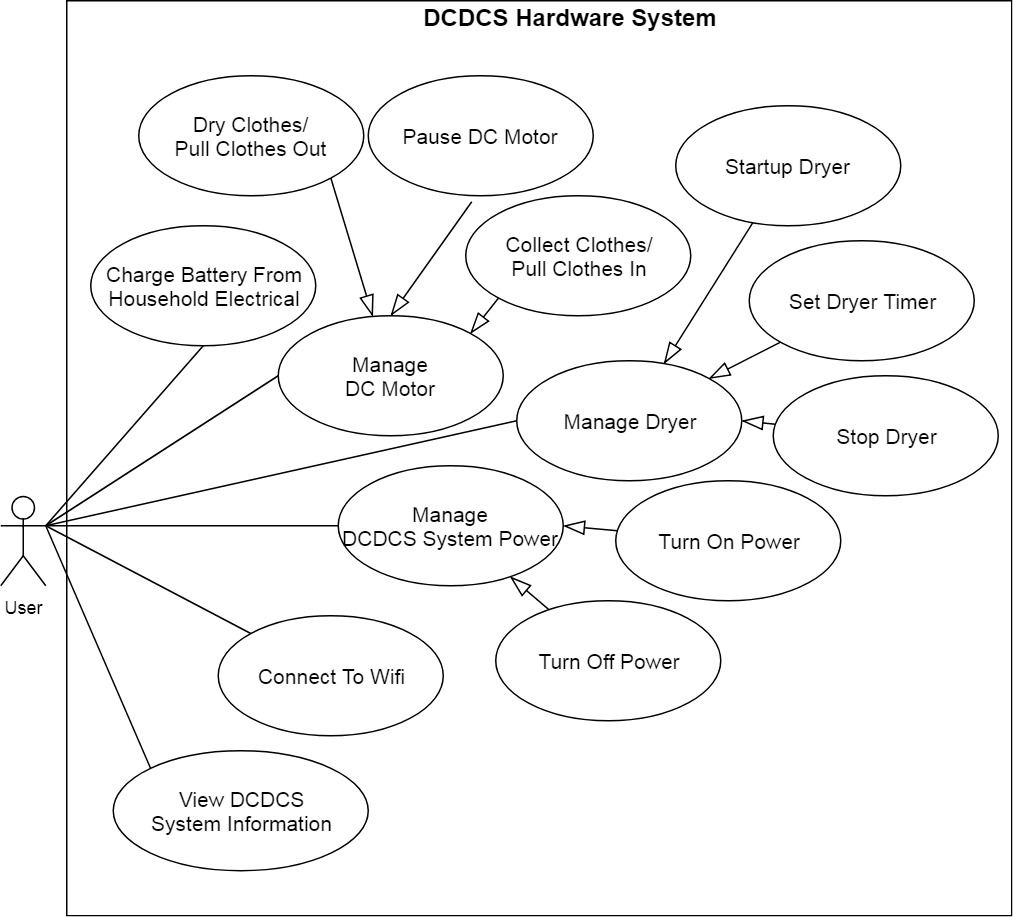


Figure 17: <System User> Overview Usecase

##### D:\Download\Chrome\a.jpg<System User> Turn on power

Figure 18: <System User> Turn on power

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU01 | | | |
| Use Case No. | DCDCS\_SU01 | **Use Case Version** | 1.0 |
| Use Case Name | Turn On Power | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Low |
| Actor:   * User   Summary:   * This use case allows actor turn on power of system   Goal:   * Actor can startup system   Triggers:   * Actor press on power switch   Pre-conditions:   * Battery of system still has energy * System is turned off   Post-conditions:   * Success: System is startup, LCD is turned on * Fail: System fails to startup   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on power switch  [Exception 1, 2] | System boot and turn on LCD  [Exception 1] |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | System can’t startup | | 2 | Power switch is broken | System can’t startup |   Relationships: Manage DCDCS Power (Abstract Use Case)  Business Rules: N/A | | | |

Table 24: USE CASE – DCDCS\_SU01 - Turn on Power

##### D:\Download\Chrome\DCDCS System Use Case 2.jpg<System User> Turn off power

Figure 19: <System User> Turn off power

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU02 | | | |
| Use Case No. | DCDCS\_SU02 | **Use Case Version** | 1.0 |
| Use Case Name | Turn Off Power | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | Low |
| Actor:   * User   Summary:   * This use case allows actor turn off power of system   Goal:   * Actor can shutdown system   Triggers:   * Actor press on power switch   Pre-conditions:   * System is running   Post-conditions:   * Success: System is turned off, LCD turn off * Fail: System still running , LCD still working   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on switch power  [Exception 1] | System shutdown and turn off LCD |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Power switch is broken | System still running |   Relationships: Manage DCDCS Power (Abstract Use Case)  Business Rules: N/A | | | |

Table 25: USE CASE – DCDCS\_SU02 - Turn off power

##### <System User> Stop dryer

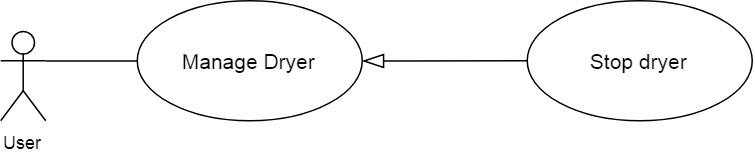


Figure 20: <System User> Stop drying

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU03 | | | |
| Use Case No. | DCDCS\_SU03 | **Use Case Version** | 1.0 |
| Use Case Name | Stop Dryer | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor stop dryer   Goal:   * Actor can stop dryer   Triggers:   * Actor press on “D” button   Pre-conditions:   * System is running stability * Dryer is running   Post-conditions:   * Success: Dryer is stopped * Fail: Dryer can’t stop   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on “D” button on keypad  [Exception 1, 2] | System stop dryer  [Exception 1] |   Alternative Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on “D” button on RF remote  [Exception 1, 3, 4] | System stop dryer  [Exception 1] |   Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | “D” button on keypad is broken | Dryer is not stopping | | 3 | “D” button on RF remote is broken | Dryer is not stopping | | 4 | Battery of RF remote run out | Dryer is not stopping |   Relationships: Manage Dryer (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone | | | |

Table 26: USE CASE – DCDCS\_SU03 - Stop dryer

##### <System User> Start dryer

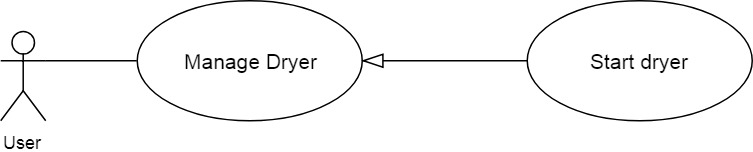


Figure 21: <System User> Start dryer

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU04 | | | |
| Use Case No. | DCDCS\_SU04 | **Use Case Version** | 1.0 |
| Use Case Name | Startup Dryer | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor startup dryer   Goal:   * Actor can startup dryer   Triggers:   * Actor press on “D” button   Pre-conditions:   * System is running stability * Dryer is not running   Post-conditions:   * Success: Dryer is running * Fail: Dryer is not running   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on “D” button on keypad  [Exception 1, 2] | System startup dryer  [Exception 1] |   Alternative Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on “D” button on keypad  [Exception 1, 3, 4] | System startup dryer  [Exception 1] |   Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | “D” button on keypad is broken | Dryer is not running | | 3 | “D” button on RF remote is broken | Dryer is not running | | 4 | Battery of RF remote run out | Dryer is not running |   Relationships: Manage Dryer (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone | | | |

Table 27: USE CASE – DCDCS\_SU04 - Start dryer

##### <System User> Set dryer timer

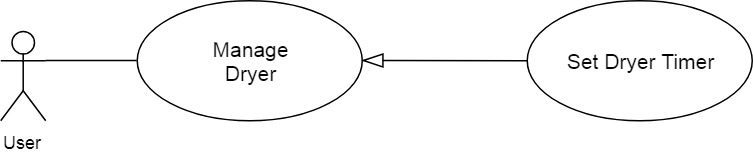


Figure 22: <System User> Set dryer timer

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU05 | | | |
| Use Case No. | DCDCS\_SU05 | **Use Case Version** | 1.0 |
| Use Case Name | Setup Dryer Timer | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor setup dryer timer   Goal:   * Actor can set timer for dryer   Triggers:   * Actor press on “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png” button or “2” button   Pre-conditions:   * System is running stability   Post-conditions:   * Success: System responses timer value in LCD correctly * Fail: The timer value in LCD is not change   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png” button or “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png” button on keypad  [Exception 1, 2] | System responses timer value in LCD  Dryer timer: 30, 60, 90, 120, 150, 180. Unit: minutes  [Exception 1] |   Alternative Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png” button or “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png” button on RF remote  [Exception 1, 3, 4] | System responses timer value in LCD  Dryer timer: 30, 60, 90, 120, 150, 180. Unit: minutes  [Exception 1] |   Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png” button or “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png” button on keypad is broken | The timer value in LCD is not change | | 3 | “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png” button or “C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png” button on RF remote is broken | The timer value in LCD is not change | | 4 | Battery of RF remote run out | The timer value in LCD is not change |   Relationships: Manage Dryer (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone | | | |

Table 28: USE CASE – DCDCS\_SU05 - Set dryer timer

##### D:\Download\Chrome\DCDCS System Use Case 2.jpg<System User> View system information

Figure 23: <System User> View system information

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU06 | | | |
| Use Case No. | DCDCS\_SU06 | **Use Case Version** | 1.0 |
| Use Case Name | View DCDCS System Information | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor view information of system   Goal:   * Actor can follow and control system easily   Triggers:   * Actor press on power button   Pre-conditions: N/A  Post-conditions:   * Success: LCD show information * Fail: LCD can’t startup   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press power button  [Exception 1] | System startup, LCD show information include:  + IP: text  + Temperature: text  + Humidity: text  + System status: text  + Dryer timer: text  [Exception 2] |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Power button is broken | System can’t startup | | 2 | Battery of system run out | Whole system stop working immediately |   Relationships: N/A  Business Rules: N/A | | | |

Table 29: USE CASE – DCDCS\_SU06 - View system information

##### <System User> Pause DC Motor

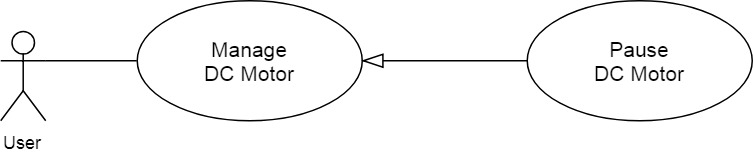


Figure 24: <System User> Pause DC Motor

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU07 | | | |
| Use Case No. | DCDCS\_SU07 | **Use Case Version** | 1.0 |
| Use Case Name | Pause DC Motor | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor pause DC motor   Goal:   * Actor can pause DC motor   Triggers:   * Actor press on “M” button   Pre-conditions:   * System is running stability * DC motor is running   Post-conditions:   * Success: DC motor is stopped * Fail: DC motor doesn’t stop   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on “M” button on keypad  [Exception 1, 2] | System stop DC motor  [Exception 1] |   Alternative Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press on “M” button on RF remote  [Exception 1, 3, 4] | System stop DC motor  [Exception 1] |   Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | “M” button on keypad is broken | DC motor doesn’t stop | | 3 | “M” button on RF remote is broken | DC motor doesn’t stop | | 4 | Battery of RF remote run out | DC motor doesn’t stop |   Relationships: Manage DC motor (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone | | | |

Table 30: USE CASE – DCDCS\_SU07 - Pause DC

##### <System User> Control DC to dry clothes

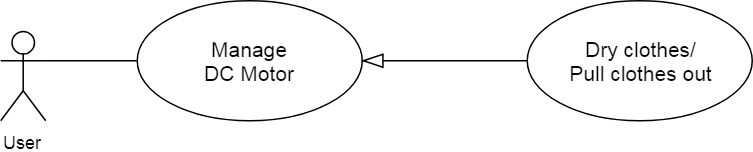


Figure 25: <System User> Control DC to dry clothes

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU08 | | | |
| Use Case No. | DCDCS\_SU08 | **Use Case Version** | 1.0 |
| Use Case Name | Dry Clothes / Pull Clothes Out | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor dry clothes   Goal:   * User can dry clothes   Triggers:   * User press on “M” button   Pre-conditions:   * System is running stability * DC motor is stopped * Weather is not raining * Time is not night   Post-conditions:   * Success: DC motor pulls clothes out successful * Fail: DC motor is not running   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User press on “M” button on keypad  [Exception 1, 2] | System pulls clothes out  [Exception 1] |   Alternative Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User press on “M” button on RF Remote  [Exception 1, 3, 4] | System pull clothes in  [Exception 1] |   Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | “M” button on keypad is broken | DC motor doesn’t run | | 3 | “M” button on RF remote is broken | DC motor doesn’t run | | 4 | Battery of RF remote run out | DC motor doesn’t run |   Relationships: Manage DC motor (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone * If system is connecting to smartphone, it will disconnect to smartphone | | | |

Table 31: USE CASE – DCDCS\_SU08 - Control dc to dry clothes

##### <System User> Control DC to collect clothes



Figure 26: <System User> Control DC to collect clothes

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU09 | | | |
| Use Case No. | DCDCS\_SU09 | **Use Case Version** | 1.0 |
| Use Case Name | Collect Clothes / Pull Clothes In | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor collect clothes   Goal:   * User can collect clothes   Triggers:   * User press on “M” button   Pre-conditions:   * System is running stability * DC motor is not running   Post-conditions:   * Success: DC motor pull clothes in successful * Fail: DC motor doesn’t run   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User press on “M” button on keypad  [Exception 1, 2] | System pull clothes in  [Exception 1] |   Alternative Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User press on “M” button on RF Remote  [Exception 1, 3, 4] | System pull clothes in  [Exception 1] |   Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | “M” button on keypad is broken | DC motor doesn’t run | | 3 | “M” button on RF remote is broken | DC motor doesn’t run | | 4 | Battery of RF remote run out | DC motor doesn’t run |   Relationships: Manage DC motor (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone * If system is connecting to smartphone, it will disconnect to smartphone | | | |

Table 32: USE CASE – DCDCS\_SU09 - Control dc to collect clothes

##### D:\Download\Chrome\Usecase detail.jpg<System User> Connect to Wi-Fi

Figure 27: <System User> Connect to Wi-Fi

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SU10 | | | |
| Use Case No. | DCDCS\_SU10 | **Use Case Version** | 1.0 |
| Use Case Name | Connect to Wi-Fi | | |
| Author | LongHP | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * User   Summary:   * This use case allows actor to connect system to a Wi-Fi   Goal:   * System connects to Wi-Fi   Triggers:   * User enter Wi-Fi name and password then press “Connect”   Pre-conditions:   * System is not connected to Wi-Fi   Post-conditions:   * Success: System connects to given Wi-Fi * Fail: System is failed to connect to given Wi-Fi   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User connect to system Wi-Fi station  [Exception 1] | N/A | | 2 | User goto address 10.0.1.1  [Exception 1] | System return website contains Wi-Fi configurations | | 3 | User click “Configure WiFi”  [Exception 1] | System scan Wi-Fi station and return return all found stations to user | | 4 | User choose a Wi-Fi station | N/A | | 5 | User enter Wi-Fi station password | N/A | | 6 | User click “Save”  [Exception 1, 2] | System Wi-Fi station will turn off |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | User disconnects from system Wi-Fi | | 2 | Wrong SSID or Password | System Wi-Fi station is not turning off |   Relationships: N/A  Business Rules: N/A | | | |

Table 33: USE CASE – DCDCS\_SU10 - Connect to Wi-Fi

#### <System> Overview Usecase

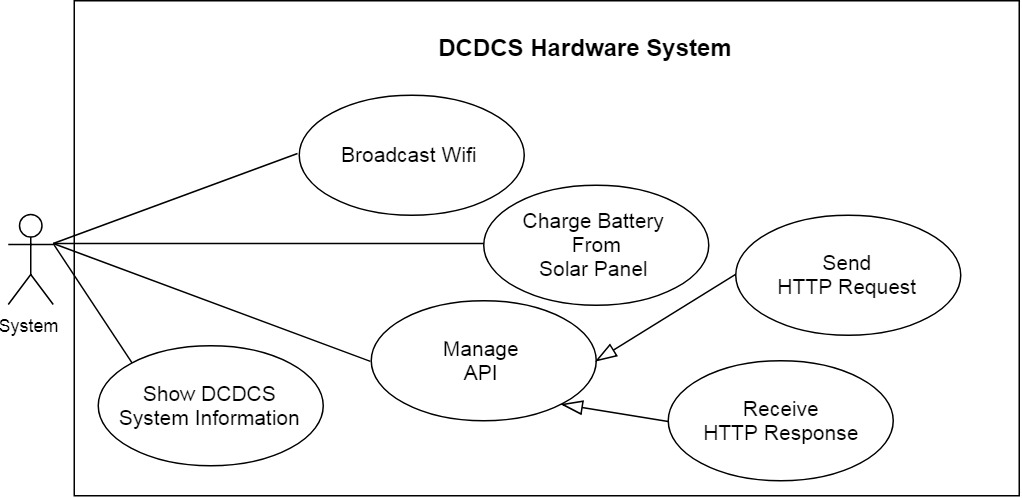


Figure 28: <System> Overview Usecase

##### <System> Control DC to collect clothes



Figure 29: <System> Control DC to collect clothes

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SYS01 | | | |
| Use Case No. | DCDCS\_SYS01 | **Use Case Version** | 1.0 |
| Use Case Name | Collect Clothes / Pull Clothes In | | |
| Author | PhongND | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * System   Summary:   * This use case allows system collect clothes automatically   Goal:   * Controller control system to collect clothes   Triggers:   * Controller send signal to H-bridge module   Pre-conditions:   * System is running stability * DC motor is not running   Post-conditions:   * Success: DC motor pull clothes in successful * Fail: DC motor doesn’t run   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Controller receive rain signal from rain sensor  Controller send signal to H-bridge module  [Exception 1, 2] | System pull clothes in  [Exception 1] |   Alternative Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Controller receive lux signal from light sensor  Controller send signal to H-bridge module  [Exception 1, 3] | System pull clothes in  [Exception 1] |   Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | Rain sensor is broken | DC motor doesn’t run | | 3 | Light sensor is broken | DC motor doesn’t run | | 4 | H-bridge module is broken | DC motor doesn’t run |   Relationships: Manage DC motor (Abstract Use Case)  Business Rules:   * When actor use this use case, system will not receive signal from smartphone | | | |

Table 34: USE CASE – DCDCS\_SYS01 - Control DC to collect clothes

##### D:\Download\Chrome\Usecase detail.jpg<System> Show information

Figure 30: <System> Show information

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SYS02 | | | |
| Use Case No. | DCDCS\_SYS02 | **Use Case Version** | 1.0 |
| Use Case Name | Show DCDCS System Information | | |
| Author | LongHP | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * System   Summary:   * This use case allows actor show information of system   Goal:   * Actor can follow and control system easily   Triggers:   * Actor press on power button   Pre-conditions: N/A  Post-conditions:   * Success: LCD show information * Fail: LCD can’t startup   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press power button  [Exception 1] | System startup  System collect data from sensors  LCD show information include:  + IP: text  + Temperature: text  + Humidity: text  + System status: text  + Dryer timer: text  [Exception 2, 3] |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Power button is broken | System can’t startup | | 2 | Battery of system run out | Whole system stop working immediately | | 3 | Temperature and Humidity sensor is broken | System show wrong information |   Relationships: N/A  Business Rules: N/A | | | |

Table 35: USE CASE – DCDCS\_SYS02 - Show information

##### D:\Download\Chrome\Usecase detail (1).jpg<System> Broadcast Wi-Fi

Figure 31: <System> Broadcast Wi-Fi

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SYS03 | | | |
| Use Case No. | DCDCS\_SYS03 | **Use Case Version** | 1.0 |
| Use Case Name | Broadcast Wi-Fi | | |
| Author | LongHP | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * System   Summary:   * This use case allows actor broadcast Wi-Fi   Goal:   * Actor can broadcast Wi-Fi to allow user connect   Triggers:   * Actor press on power button * System Wi-Fi disconnected or cannot connect to Wi-Fi   Pre-conditions:   * System is not connected to any Wi-Fi stations   Post-conditions:   * Success: System Wi-Fi show up on Wi-Fi list * Fail: System Wi-Fi doesn’t show up on Wi-Fi list   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor press power button  [Exception 1] | System startup  System starting to broadcast Wi-Fi  [Exception 2, 3] |   Alternative Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Wi-Fi is disconnected | System starting to broadcast Wi-Fi  [Exception 2, 3] |   Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Power button is broken | System can’t startup | | 2 | Battery of system run out | Whole system stop working immediately | | 3 | System stack run-out | System will freeze and requires to reboot |   Relationships: N/A  Business Rules: N/A | | | |

Table 36: USE CASE – DCDCS\_SYS03 - Broadcast Wi-Fi

##### <System> Send HTTP Request

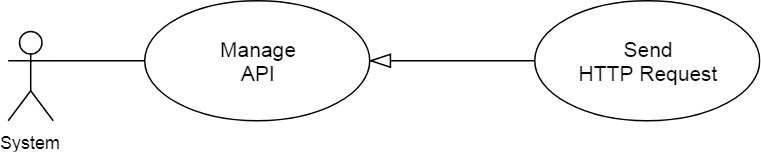


Figure 32: <System> Send HTTP Request

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SYS04 | | | |
| Use Case No. | DCDCS\_SYS04 | **Use Case Version** | 1.0 |
| Use Case Name | Send a request to server | | |
| Author | LongHP | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * System   Summary:   * This use case allows actor to send a request to server via HTTP(s)   Goal:   * Actor can send a request to server   Triggers:   * Wi-Fi is connected   Pre-conditions:   * System is connected to any Wi-Fi stations   Post-conditions:   * Success: Server receives a request from system * Fail: Server doesn’t receive a request from system   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | System is connected to Wi-Fi  [Exception 1] | N/A  [Exception 2, 3] |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | SSL fingerprint is expired | System cannot send a request to server. Require to update firmware to update SSL fingerprint | | 3 | Unable to send due to slow internet connection | System will try again until request is sent |   Relationships: Manage API (Abstract Use case)  Business Rules: N/A | | | |

Table 37: USE CASE – DCDCS\_SYS04 - Send HTTP request

##### <System> Get HTTP Response

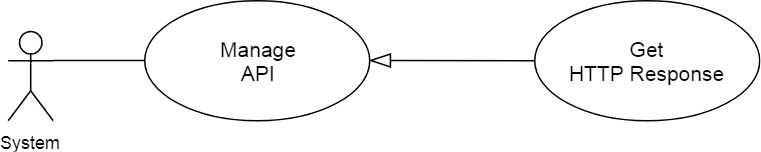


Figure 33: <System> Get HTTP Response

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE – DCDCS\_SYS05 | | | |
| Use Case No. | DCDCS\_SYS05 | **Use Case Version** | 1.0 |
| Use Case Name | Get a response from server | | |
| Author | LongHP | | |
| Date | 14/07/2018 | **Priority** | High |
| Actor:   * System   Summary:   * This use case allows actor to get a response from server via HTTP(s)   Goal:   * Actor can get a response to server   Triggers:   * Wi-Fi is connected   Pre-conditions:   * System is connected to any Wi-Fi stations   Post-conditions:   * Success: System receives a request from server * Fail: System doesn’t receive a request from server   Main Success Scenario:   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | System is connected to Wi-Fi  [Exception 1] | N/A  [Exception 2, 3] |   Alternative Scenario: N/A  Exceptions:   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Battery of system run out | Whole system stop working immediately | | 2 | SSL fingerprint is expired | System cannot send a request to server. Require to update firmware to update SSL fingerprint | | 3 | Unable to get response due to slow internet connection | System will try again until request is sent |   Relationships: Manage API (Abstract Use case)  Business Rules: N/A | | | |

Table 38: USE CASE – DCDCS\_SYS05 - Get HTTP response

## Hardware Requirement Specification

### Hardware Interface

The hardware interface must have satisfied the following requirements:

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

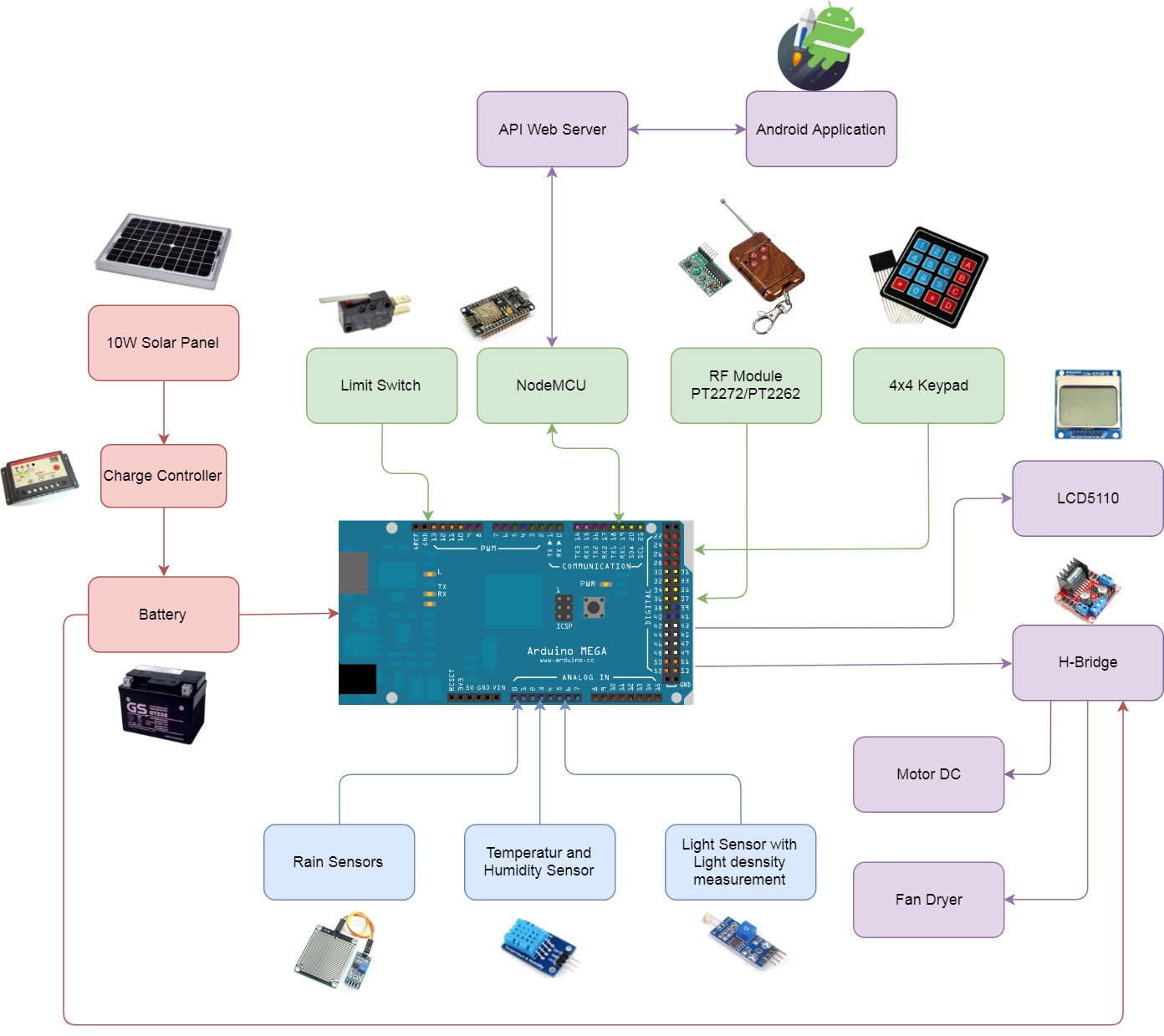
Based on project requirement we have choose following hardware components

Figure 34: System block diagram

#### IMG_256Rain Sensor

Figure 35: 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 39: 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 36: 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 40: 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 37: 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 41: 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 38: 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 42: 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 39: 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 43: 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 40: 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 44: Specifications for 4x4 matrix keypad

#### Limit Switches

Figure 41: 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 45: Specifications for limit switch

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

Figure 42: 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 46: Specifications for DC Motor GA37

#### Solar Panel

Figure 43: 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 47: Specifications for solar panel

#### Solar Charge Controller

Figure 44: 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

Table 48: Specifications for charge controller

**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 |

#### [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 45: 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 49: Specifications for GTZ5S-E battery

#### Image result for nodemcuNodeMCU

Figure 46: 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 50: 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).

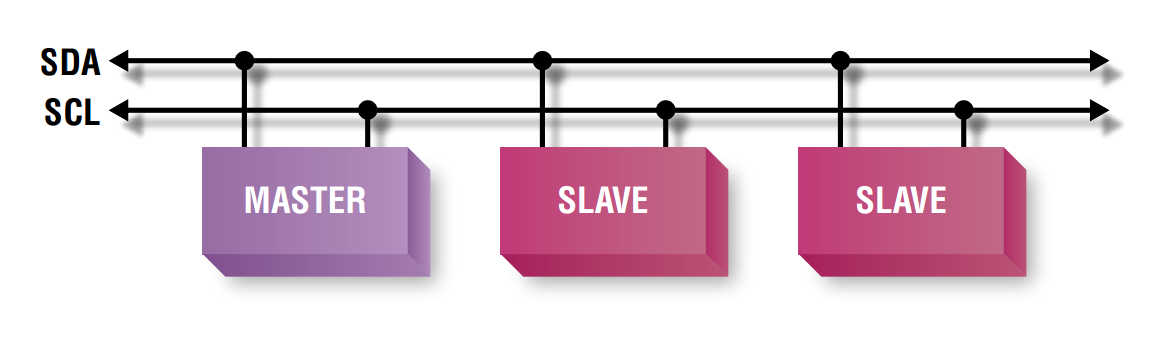


Figure 47: I2C Protocol

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

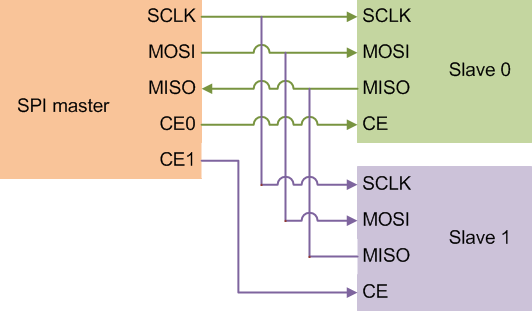
SPI (**Serial Peripheral Interface)** is a [synchronous](https://en.wikipedia.org/wiki/Synchronous_circuit) [serial communication](https://en.wikipedia.org/wiki/Serial_communication) interface specification used for short distance communication, primarily in [embedded systems](https://en.wikipedia.org/wiki/Embedded_systems). 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 48: 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 49: 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.

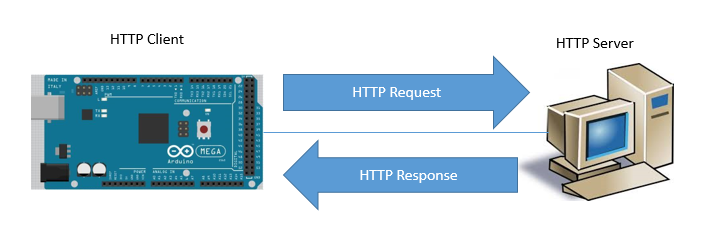
HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a website may be the server. The client submits an HTTP request message to the server. The server, which provides resources 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 50: 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

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

Figure 51: 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 51: Data dictionary for conceptual diagram

# Software & Hardware Design Description

## Design Overview

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

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

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

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

* Section 1: Introduction
* Section 2: Gives an overall description of the system architecture design
* Section 3: Gives component diagrams that describe the connection and  
  integration of the system
* Section 4: Gives the detail design description which includes class diagram,  
  class explanation, and sequence diagram to details the application functions
* Section 5: Describe screens design
* Section 6: Describe a fully attributed ERD
* Section 7: Describe algorithms

## System Architectural Design

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

Figure 52: System overview architecture

Figure 53: API Web server architecture

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

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

This project follows MVC architecture with following components:

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

### Android Application Architectural Design

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

Figure 54: Android 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

Figure 55: Android application internal architecture

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

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

Android Application follows Flux architecture with following components:

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

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

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

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

Figure 56: Hardware system architecture

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

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

The system follows IoT architecture with following components:

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

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

Figure 57: 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 52: Component diagram dictionary

## Detailed Description

### Class Diagram

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

Figure 58: API Web Server Class Diagram Part 1

Figure 59: API Web Server Class Diagram Part 1

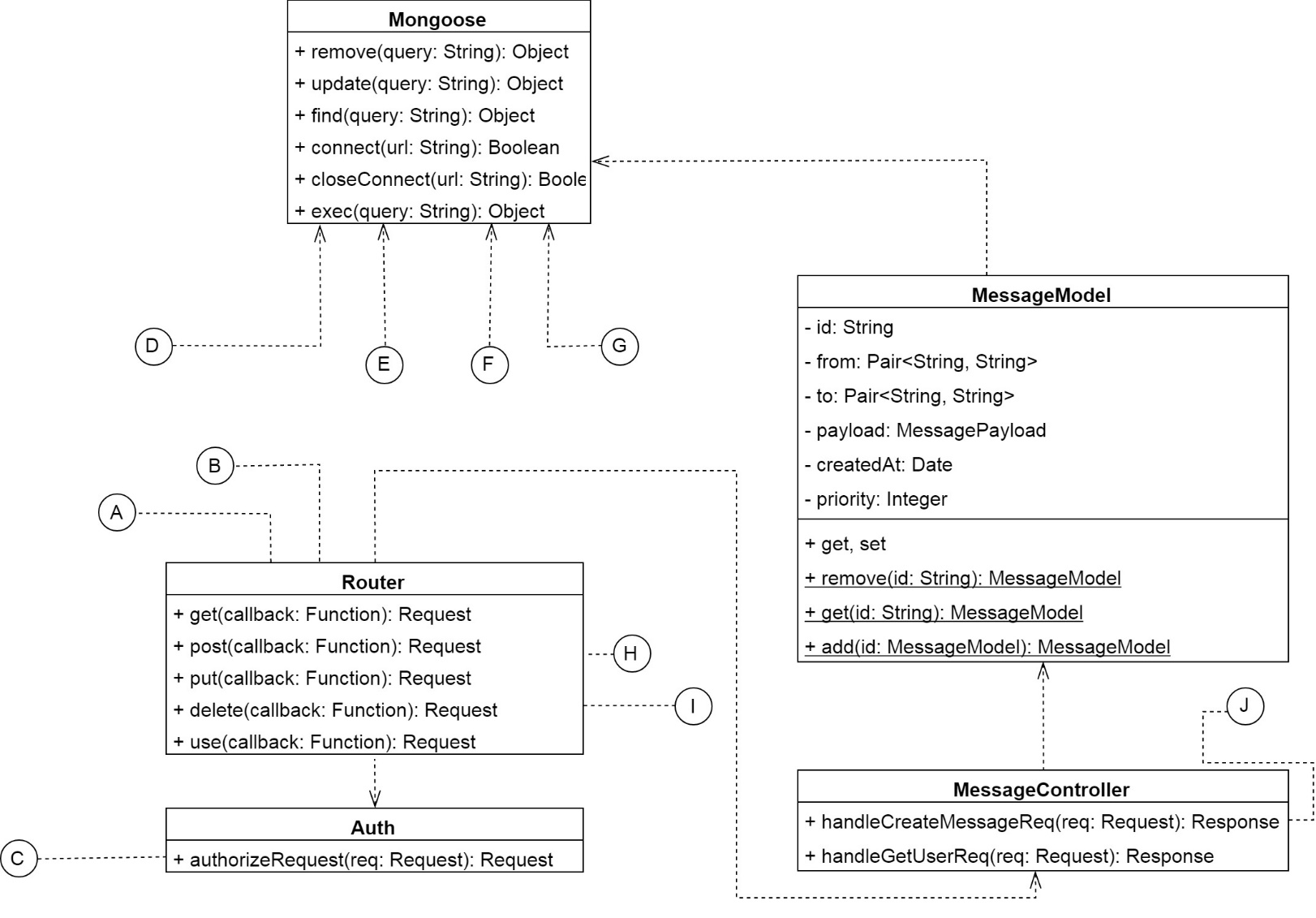


Figure 60: API Web Server Class Diagram Part 2

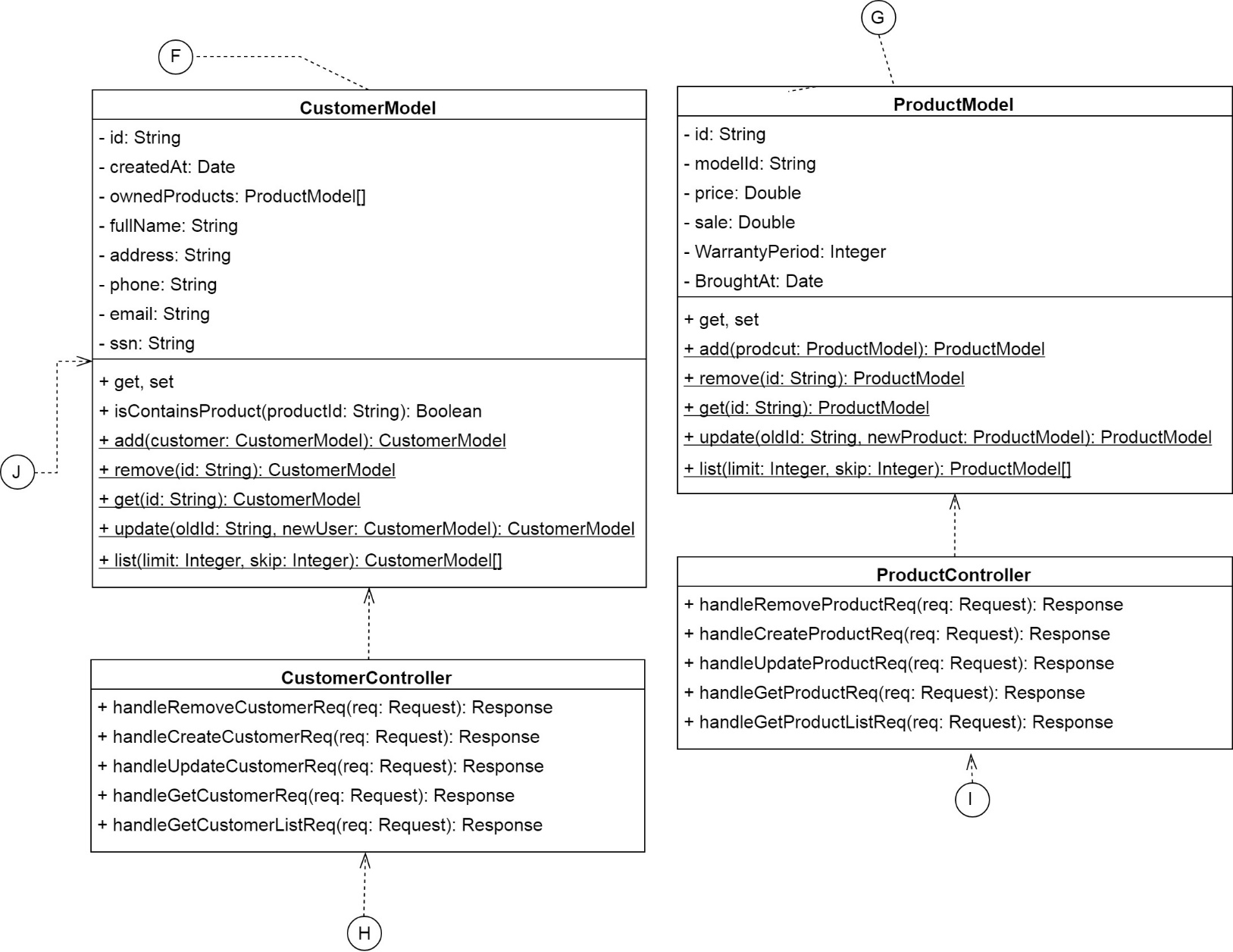


Figure 61: API Web Server Class Diagram Part 3

|  |  |  |
| --- | --- | --- |
| 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 53: API Web server class diagram dictionary

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

Figure 62: Hardware system controller class diagram part 1



Figure 63: 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 54: Hardware controller class diagram dictionary

### Class Diagram Explanation

#### UserModel

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| id | String | Private | User’s unique identifier |
| username | String | Private | Name of user |
| password | String | Private | Password of user |
| salt | String | Private | Security salt |
| createdAt | Date | Private | Date that user was created |
| token | String | Private | Access token of user |
| ownerID | String | Private | The owner of the single user |
| isAdmin | Boolean | Private | Is user administrator? |

Table 55: Attribute dictionary for UserModel

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| Getter | Attribute type | Public | Get attribute value |
| Setter | Void | Public | Set attribute value |
| add(user: UserModel) | UserModel | Public, Static | Add a user to DB |
| remove(id: String) | UserModel | Public, Static | Remove a user from DB |
| get(id: String) | UserModel | Public, Static | Get single user with given Id from DB |
| update(oldId: String, newUser: UserModel) | UserModel | Public, Static | Update user info to DB |
| list(limit: Integer, skip: Integer) | UserModel[ ] | Public, Static | Get a list of users |
| checkExist(id: String) | Boolean | Public, Static | Check if user with given Id exists |
| verifyToken(token: String) | UserModel | Public, Static | Verify user token |

Table 56: Method dictionary for UserModel

#### ModelModel

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| id | String | Private | Unique identifier for a model |
| price | Double | Private | A price for a product that belongs to a model |
| sale | Double | Private | A percent of sale off |
| quantity | Integer | Private | A number of products belong to a model |
| warrantyPeriod | Integer | Private | Warranty period of a product that belongs to a model |
| releaseDate | Date | Private | The first day release a model |
| supportEndDate | Date | Private | Support end date of a model |

Table 57: Attribute dictionary for ModelModel

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| Getter | Attribute type | Public | Get attribute value |
| Setter | Void | Public | Set attribute value |
| add(model: ModelModel) | ModelModel | Public, Static | Add a model to DB |
| remove(id: String) | ModelModel | Public, Static | Remove a model from DB |
| get(id: String) | ModelModel | Public, Static | Get single model with given Id from DB |
| update(oldId: String, newModel: ModelModel) | ModelModel | Public, Static | Update model info to DB |
| list(limit: Integer, skip: Integer) | ModelModel[] | Public, Static | Get a list of models |

Table 58: Method dictionary for ModelModel

#### MessageModel

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| id | String | Private | Unique identifier for message |
| from | Pair<String, String> | Private | Contains userId and deviceId of where message is sent |
| to | Pair<String, String> | Private | Contains userId and deviceId of where message is received |
| payload | MesseagePayload | Private | An object contains action and data |
| createAt | Date | Private | Message created time |
| priority | Integer | Private | Priority of the message |

Table 59: Attribute dictionary for MessageModel

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| Getter | Attribute type | Public | Get attribute value |
| Setter | Void | Public | Set attribute value |
| add(model: MessageModel) | MessageModel | Public, Static | Add a message to DB |
| remove(id: String) | MessageModel | Public, Static | Remove a message from DB |
| get(id: String) | MessageModel | Public, Static | Get a message from DB |

Table 60: Method dictionary for MessageModel

#### CustomerModel

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| id | String | Private | Unique identifier for a customer |
| createAt | Date | Private | Customer created time |
| ownedProducts | ProductModel[ ] | Private | List of products owned |
| fullname | String | Private | Customer’s fullname |
| address | String | Private | Customer’s address |
| phone | String | Private | Customer’s phone |
| Email | String | Private | Customer’s email |
| SSN | String | Private | Customer’s ssn |

Table 61: Attribute dictionary for CustomerModel

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| Getter | Attribute type | Public | Get attribute value |
| Setter | Void | Public | Set attribute value |
| isContainsProduct(productId: String) | Boolean | Public | Check if a customer contains a product |
| add(customer: CustomerModel) | CustomerModel | Public, Static | Add a customer to DB |
| remove(id: String) | CustomerModel | Public, Static | Remove a customer from DB |
| get(id: String) | CustomerModel | Public, Static | Get a single customer info from DB |
| update(oldId: String, newUser: CustomerModel) | CustomerModel | Public, Static | Edit/Update customer info to DB |
| list(limit: Integer, skip: Integer) | CustomerModel | Public, Static | List customers |

Table 62: Method dictionary for CustomerModel

#### ProductModel

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| id | String | Private | Unique identifier for a product |
| modelID | String | Private | Model of a product |
| price | Double | Private | Price of a product |
| sale | Double | Private | Percent sale off of a product |
| WarrantyPeriod | Integer | Private | Warranty period of a product |
| BroughtAt | Date | Private | Product’s brought date |

Table 63: Attribute dictionary for ProductModel

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| Getter | Attribute type | Public | Get attribute value |
| Setter | Void | Public | Set attribute value |
| add(prodcut: ProductModel) | ProductModel | Public, Static | Add a product to DB |
| remove(id: String) | ProductModel | Public, Static | Remove a product from DB |
| get(id: String) | ProductModel | Public, Static | Get a single product info from DB |
| update(oldId: String, newProduct: ProductModel) | ProductModel | Public, Static | Edit/Update a product info to DB |
| list(limit: Integer, skip: Integer) | ProductModel[ ] | Public, Static | List customers |

Table 64: Method dictionary for ProductModel

#### Mongoose

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| remove(query: String) | Object | Public | Perform a remove query (an) items from DB |
| update(query: String) | Object | Public | Perform an update query (an) items from DB |
| find(query: String) | Object | Public | Perform a query to find (an) items from DB |
| connect(url: String) | Boolean | Public | Connect to DB |
| closeConnect(url: String) | Boolean | Public | Close connection to DB |
| exec(query: String) | Object | Public | Execute a query |

Table 65: Method dictionary for Mongoose

#### UserController

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| handleRemoveUserReq(req: Request) | Response | Public | Handle an API call to remove a user |
| handleCreateUserReq(req: Request) | Response | Public | Handle an API call to create a new user |
| handleUpdateUserReq(req: Request) | Response | Public | Handle an API call to update a customer user |
| handleGetUserReq(req: Request) | Response | Public | Handle an API call to get a single user |
| handleGetUserListReq(req: Request) | Response | Public | Handle an API call to list user |
| handleLoginReq(req: Request) | Response | Public | Handle an API call to login |

Table 66: Method dictionary for UserController

#### ModelController

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| handleRemoveModelReq(req: Request) | Response | Public | Handle an API call to remove a customer |
| handleCreateModelReq(req: Request) | Response | Public | Handle an API call to create a new customer |
| handleUpdateModelReq(req: Request) | Response | Public | Handle an API call to update a customer info |
| handleGetModelReq(req: Request) | Response | Public | Handle an API call to get a single customer |
| handleGetModelListReq(req: Request) | Response | Public | Handle an API call to list customer |

Table 67: Method dictionary for ModelController

#### MessageController

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| handleCreateMessageReq(req: Request) | Response | Public | Handle an API call to create a message |
| handleGetUserReq(req: Request) | Response | Public | Handle an API call to get a message and then remove it |

Table 68: Method dictionary for MessageController

#### CustomerController

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| handleRemoveCustomerReq(req: Request) | Response | Public | Handle an API call to remove a customer |
| handleCreateCustomerReq(req: Request) | Response | Public | Handle an API call to create a new customer |
| handleUpdateCustomerReq(req: Request) | Response | Public | Handle an API call to update a customer info |
| handleGetCustomerReq(req: Request) | Response | Public | Handle an API call to get a single customer |
| handleGetCustomerListReq(req: Request) | Response | Public | Handle an API call to list customer |

Table 69: Method dictionary for CustomerController

#### ProductController

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| handleRemoveProductReq(req: Request) | Response | Public | Handle an API call to remove a product |
| handleCreateProductReq(req: Request) | Response | Public | Handle an API call to create a new product |
| handleUpdateProductReq(req: Request) | Response | Public | Handle an API call to update a product info |
| handleGetProductReq(req: Request) | Response | Public | Handle an API call to get a single product |
| handleGetProductListReq(req: Request) | Response | Public | Handle an API call to list products |

Table 70: Method dictionary for ProductController

#### Router

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| get(callback: Function) | Request | Public | Register an handler to handle any GET requests with given URI |
| post(callback: Function) | Request | Public | Register an handler to handle any POST requests with given URI |
| put(callback: Function) | Request | Public | Register an handler to handle any PUT requests with given URI |
| delete(callback: Function) | Request | Public | Register an handler to handle any DELETE requests with given URI |
| use(callback: Function) | Request | Public | Use a router level middleware |

Table 71: Method dictionary for Router

#### Auth

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| authorizeRequest(req: Request) | Request | Public | Check if user has permission to perform this API call |

Table 72: Method dictionary for Auth

#### SystemController

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| sysStatus | SystemStatus | Private | Save the current system status |
| sysController | SystemController | Private | The instance of this class |

Table 73: Attribute dictionary for SystemController

**Method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Return type | Visibility | | Description |
| Gettter | Attribute type | Public | Get attribute value | |
| SystemController(initState: SystemStatus) | N/A | Private | SystemController’s construction | |
| determineMotorAction(motorBtnPulse: Integer) | Action | Private | A function to help determine what action should system do with DC Motor | |
| determineDryerAction(dryerBtnPulse: Integer) | Action | Private | A function to help determine what action should system do with Dryer | |
| getAction(key: Integer) | Action | Private | Get action from action key | |
| actionControl(action: Action, timer: Integer) | Void | Private | Control system based on action | |
| createSystemController(initState: SystemStatus) | SystemController | Public | Create system controller | |
| controlSystem(key: Integer, timer: Integer) | Void | Public | Control system based on action key | |
| getSystemStatusName() | String | Public | Get name of current system status | |

Table 74: Method dictionary for SystemController

#### DryerController

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| dryerPin | Integer | Private | A pin number to control dryer |
| accTime | Integer | Private | Accumulated timer to determine a minute has passed |
| timer | Integer | Private | Dryer timer. Dryer will stop when timer goes to 0 |

Table 75: Attribute dictionary for DryerController

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| DryerController(dryerPin: Integer) | N/A | Public | Constructor |
| start() | Void | Public | Start the dryer |
| stop() | Void | Public | Stop the dryer |

Table 76: Method dictionary for DryerController

#### WifiHandler

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| WifiHandler(sda: Integer, scl: Integer) | N/A | Public | Constructor |
| onRequestEvent() | void | Public | Listen request from master (I2C Protocol) |
| onReceivedDataEvent(numBytes: Integer) | void | Public | Listen the receiving data event from master |

Table 77: Method dictionary for WifiHandler

#### DCMotor

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| ena | Integer | Private | Enable Pin A to control DC Motor |
| enb | Integer | Private | Enable Pin B to control DC Motor |

Table 78: Attribute dictionary for DCMotor

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| DCMotor(ena: Integer,enb: Integer) | N/A | Public | Constructor |
| moveForward() | Void | Private | Control DC to move forward |
| moveBackward() | Void | Private | Control DC to move backware |
| begin() | Void | Public | Setup pin for DC Motor |
| move(direct: DCDirection) | Void | Public | Move DC with given direction |

Table 79: Method dictionary for DCMotor

#### LightSensorHandler

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| LightSensorHandler(inputPins: Integer[]) | N/A | Public | Constructor |
| isNight() | Boolean | Public | Determine whenever it’s night or day |

Table 80: Method dictionary for LightSensorHandler

#### LCDHandler

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| inputPins | Integer[] | Private | A series of input pins to control LCD |

Table 81: Attribute dictionary for LCDHandler

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| LCDHandler(inputPins: Integer[]) | N/A | Public | Constructor |
| setCurser(x: Integer, y: Integer) | Void | Public | Put a cursor at point (x, y) on LCD screen |
| print(msg: String) | Void | Public | Print a message |

Table 82: Method dictionary for LCDHandler

#### SwitchHandler

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| switchPin | Integer | Private | A pin number to read data from limit switch |

Table 83: Attribute dictionary for SwitchHandler

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| SwitchHandler(switchPin: Integer) | N/A | Public | Constructor |
| isHit() | Boolean | Public | Check if limit switch is hitted/pressed |

Table 84: Method dictionary for SwitchHandler

#### RFHandler

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| inputPins | Integer[] | Private | A series of input pins to control RF |

Table 85: Attribute dictionary for RFHandler

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| RFHandler(inputPins: Integer[]) | N/A | Public | Constructor |
| getPressedKey() | Integer | Public | Get the key pressed from RF Remote |

Table 86: Method dictionary for RFHandler

#### KeypadHandler

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| inputPins | Integer[ ] | Private | A series of input pins to control Keypad |

Table 87: Attribute dictionary for KeypadHandler

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| KeypadHandler(inputPins: Integer[]) | N/A | Public | Constructor |
| getPressedKey() | Integer | Public | Get the key pressed from Keypad |

Table 88: Method dictionary for KeypadHandler

#### RainSensorHandler

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| inputPin | Integer | Private | An input pin to read data from Rain Sensor |

Table 89: Attribute dictionary for RainSensorHandler

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| RainSensorHandler(inputPin: Integer) | N/A | Public | Constructor |
| isRain() | Boolean | Public | Check if it is raining or not |

Table 90: Method dictionary for RainSensorHandler

#### CentralController

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| getSwitchHitted() | Integer | Public | Determine which limit switch is pressed/hitted |
| loopEvent() | Void | Public | The main thread of the system. Loop infinite |
| setupSystem() | Void | Public | The function that called before loopEvent to help setup system |

Table 91: Method dictionary for Central Controller

#### BHT1750

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| inputPin | Integer | Private | An input pin to read data from BHT1750 |
| lightDensity | Integer | Private | Measured light density in lux |

Table 92: Attribute dictionary for BHT1750

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| BHT1750(inputPin: Integer) | N/A | Public | Constructor |
| getLightDesity() | Integer | Public | Get measured light density in lux |

Table 93: Method dictionary for BHT1750

#### Wire

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| sda | Integer | Private | SDA pin for I2C |
| scl | Integer | Private | SCL pin for I2C |
| address | Integer | Private | Device address to communicate with another device |

Table 94: Attribute dictionary for Wire

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| Wire(sda: Integer, scl: Integer) | N/A | Public | Constructor |
| begin(address: Integer) | Void | Public | Init and setup pin mode for I2C |

Table 95: Method dictionary for Wire

#### DHT

**Attribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Description |
| inputPin | Integer | Private | Input pin to control/read data from DHT |
| addressMode | Integer | Private | Address mode for DHT |
| temperature | Double | Private | Read temperature in Celsius |
| humidity | Double | Private | Read humidity in percent |

Table 96: Attribute dictionary for DHT

**Method**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Return type | Visibility | Description |
| DHT(inputPin: Integer, addressMode: Integer) | N/A | Public | Constructor |
| getTemperature() | Double | Public | Return temperature in Celsius |
| getHumidity() | Double | Public | Return humidity in percent |

Table 97: Method dictionary for DHT

### Interaction Diagram

#### Sequence Diagrams

##### Control system from android application

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

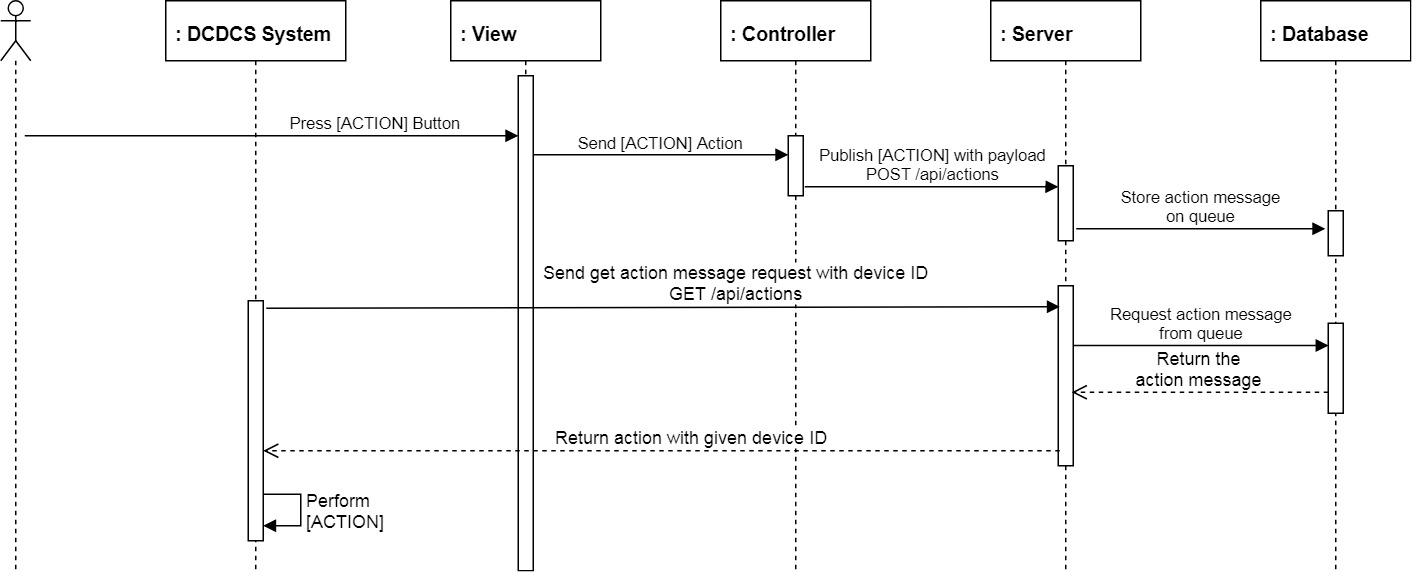


Figure 64: Control system with android app sequence diagram

##### Update system information

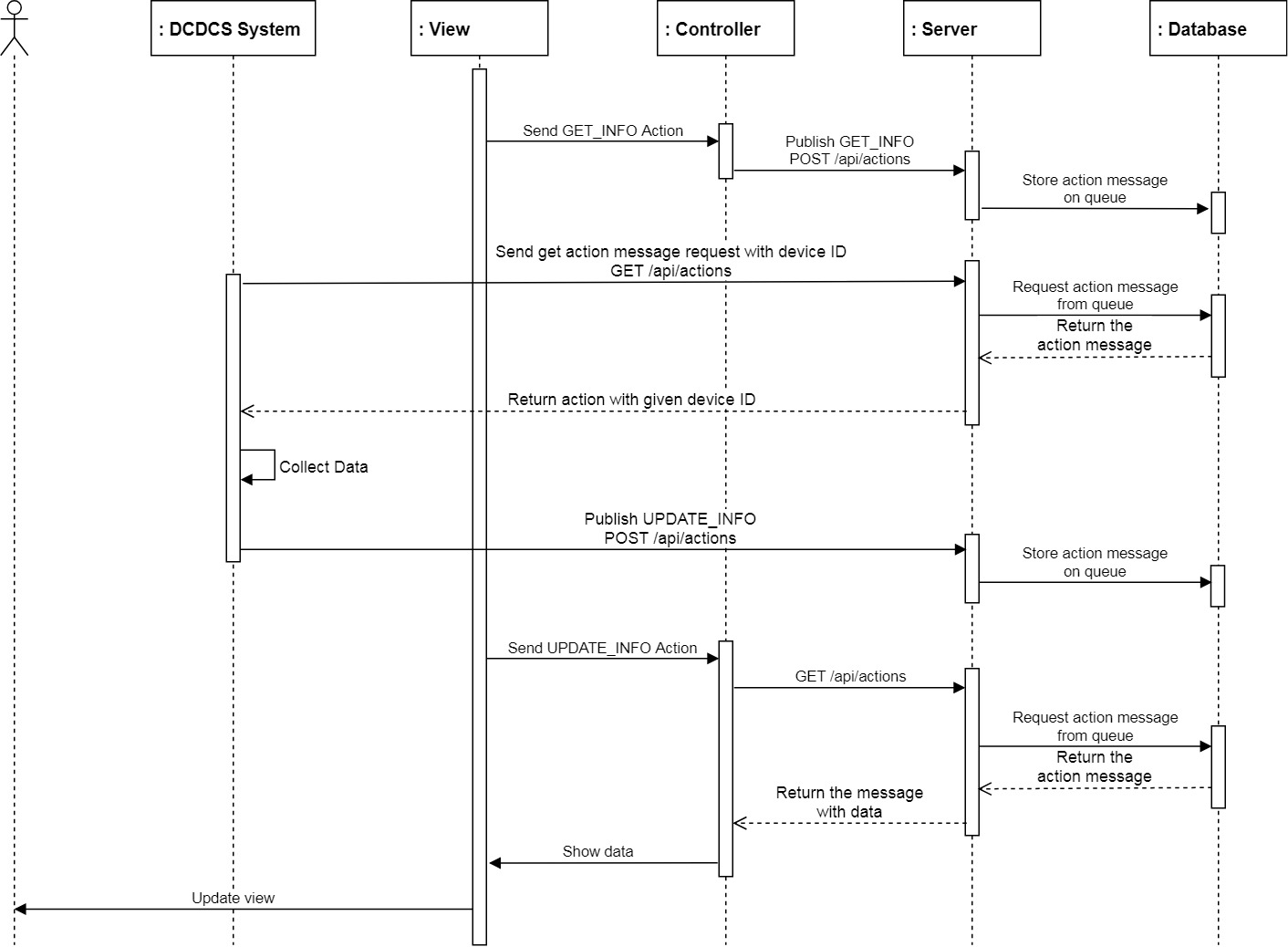
**Summary:** This diagrams show how android application gathers information from hardware system

Figure 65: Update system information sequence diagram

#### Activity Diagrams

##### Control DC

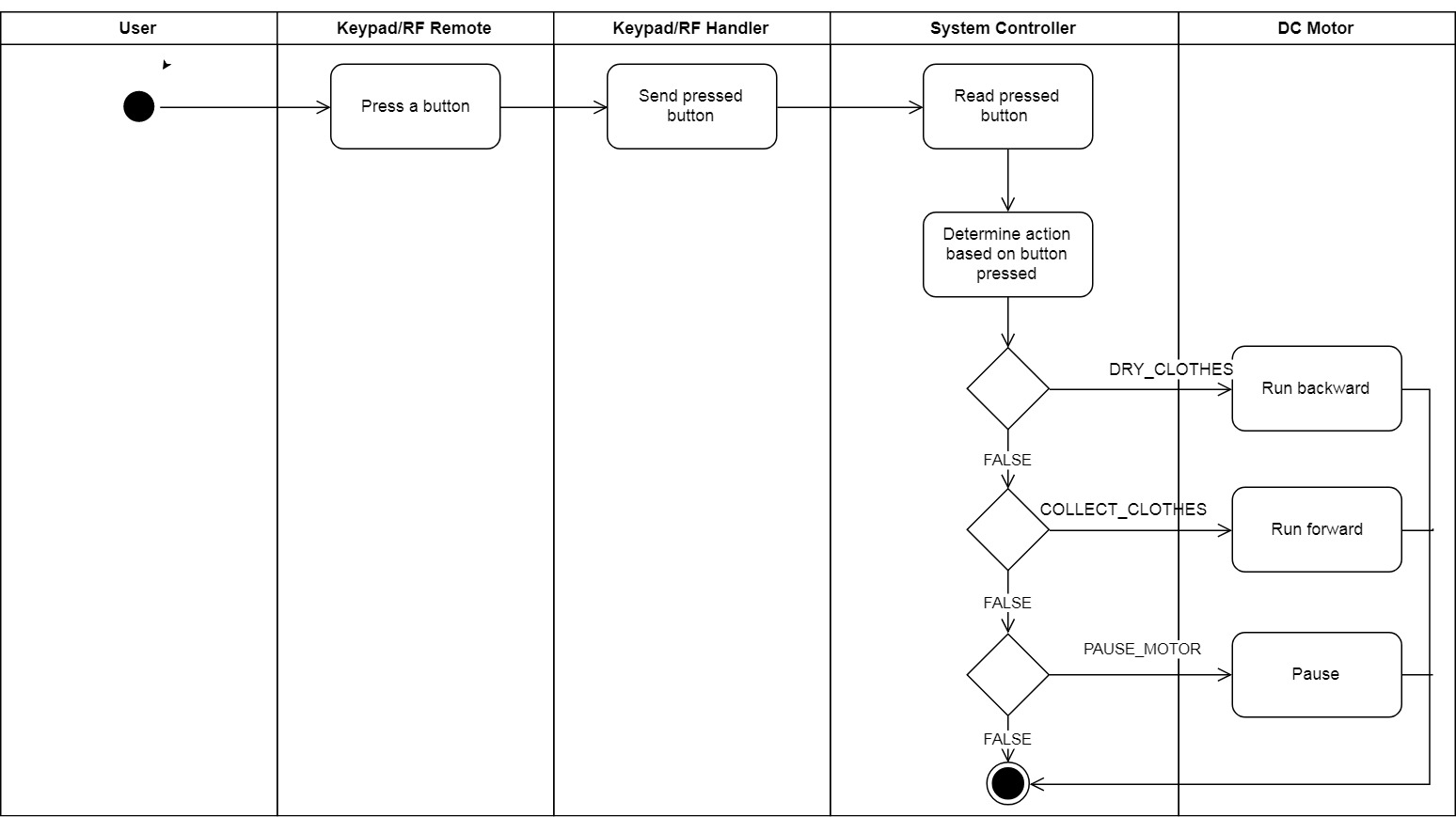
**Summary**: This diagrams show how user can control the DC

Figure 66: Control DC activity diagram

##### Control Dryer

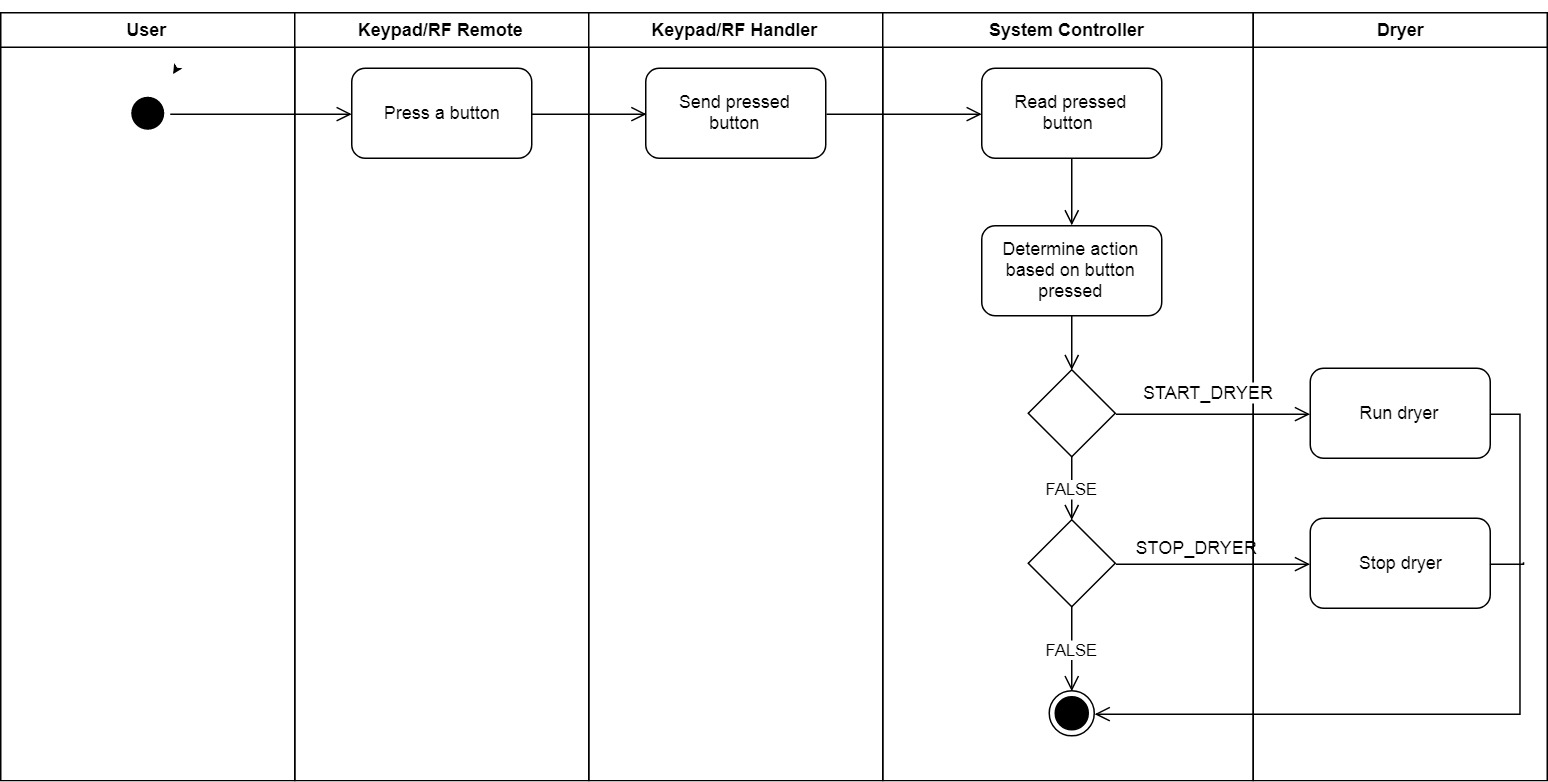
**Summary:** This diagrams show how user can control the dryer

Figure 67: Control Dryer activity diagram

##### Auto control

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

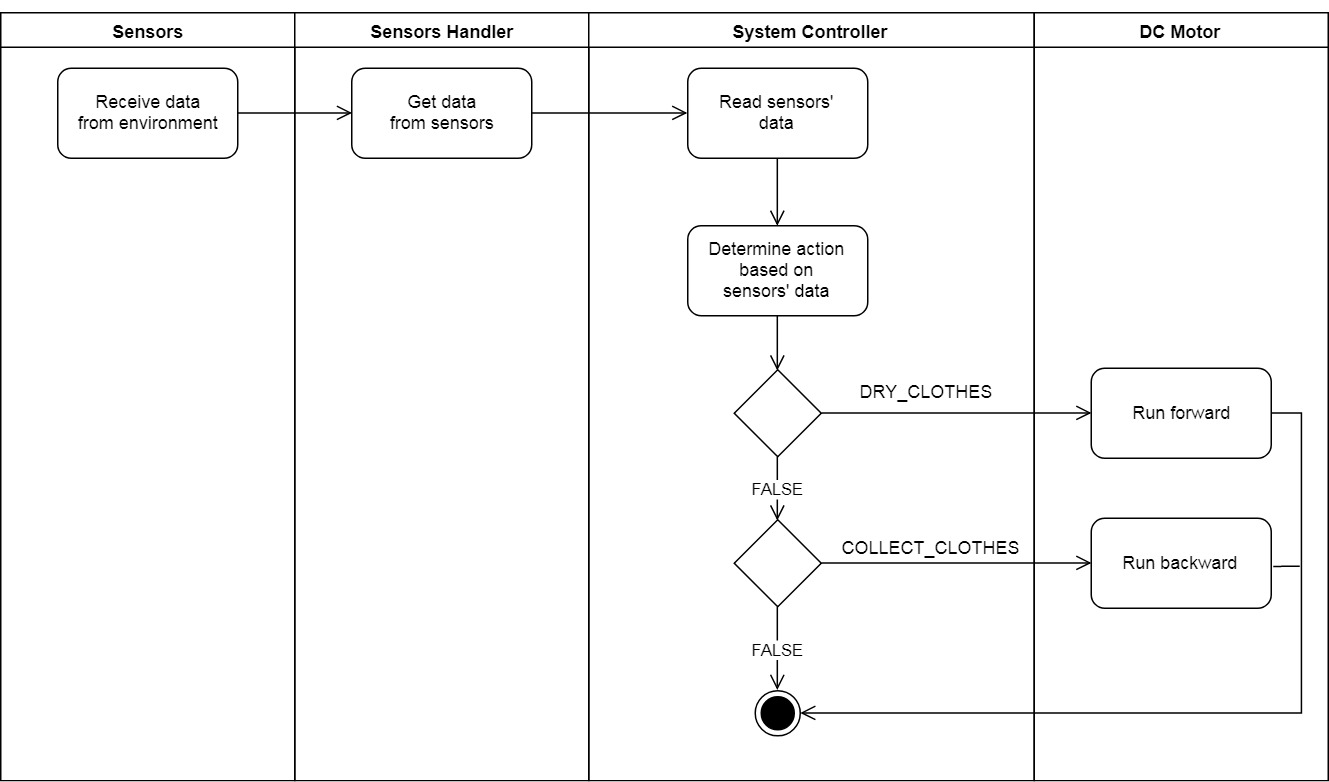


Figure 68: Auto control activity diagram

##### Determine action based on sensors’ data

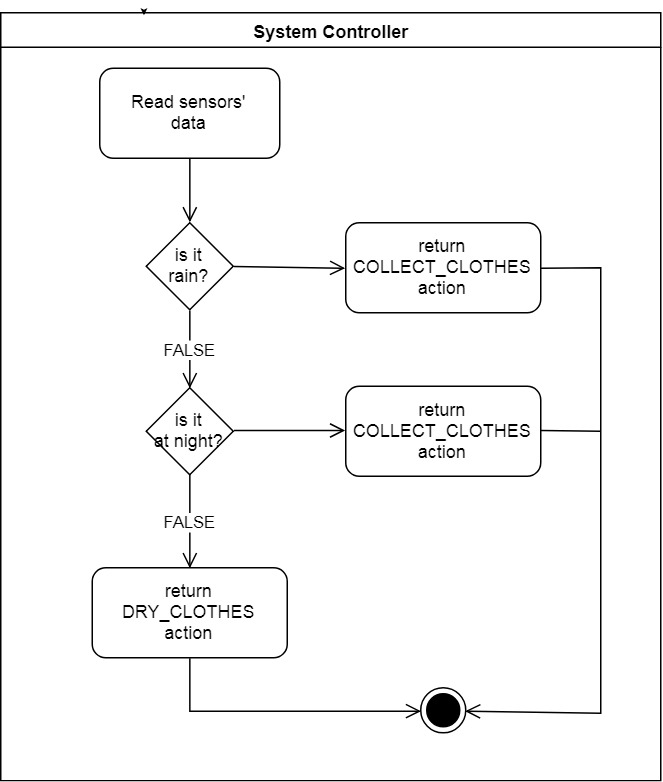
**Summary:** This diagrams show how system determine to perform an action from sensors

Figure 69: Determine action based on sensors’ data activity diagram

##### Determine DC Motor action

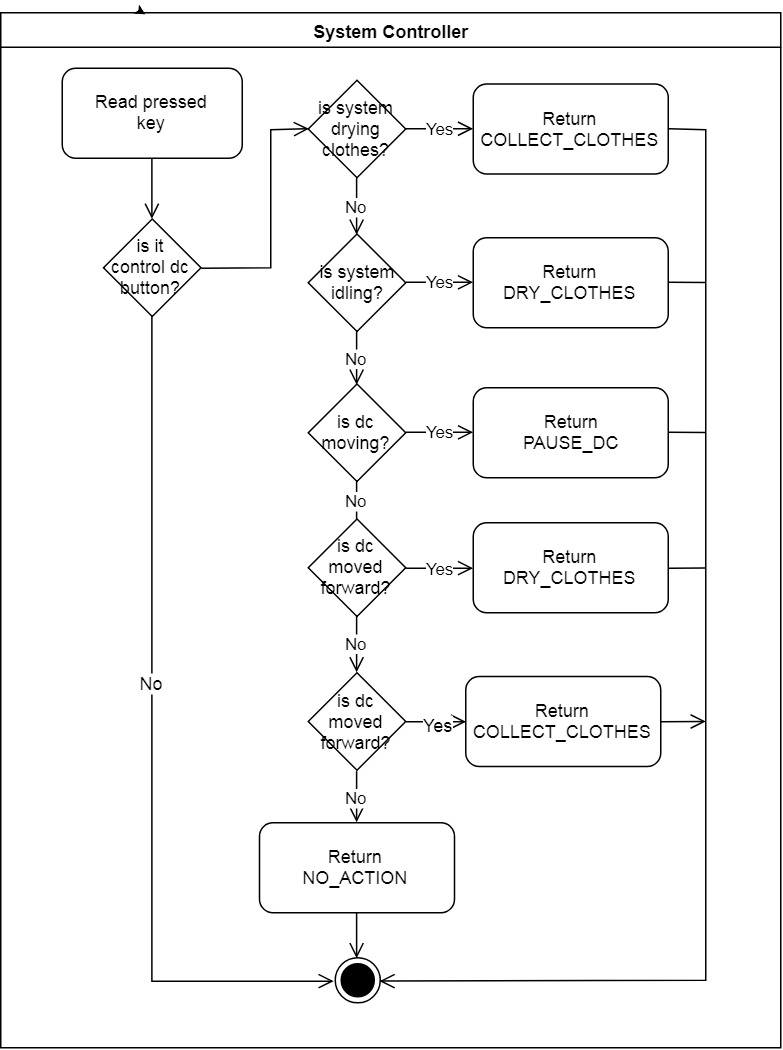
**Summary:** This diagrams show how system determine what action to control dc motor

Figure 70: Determine dc motor action activity diagram

##### Determine dryer action

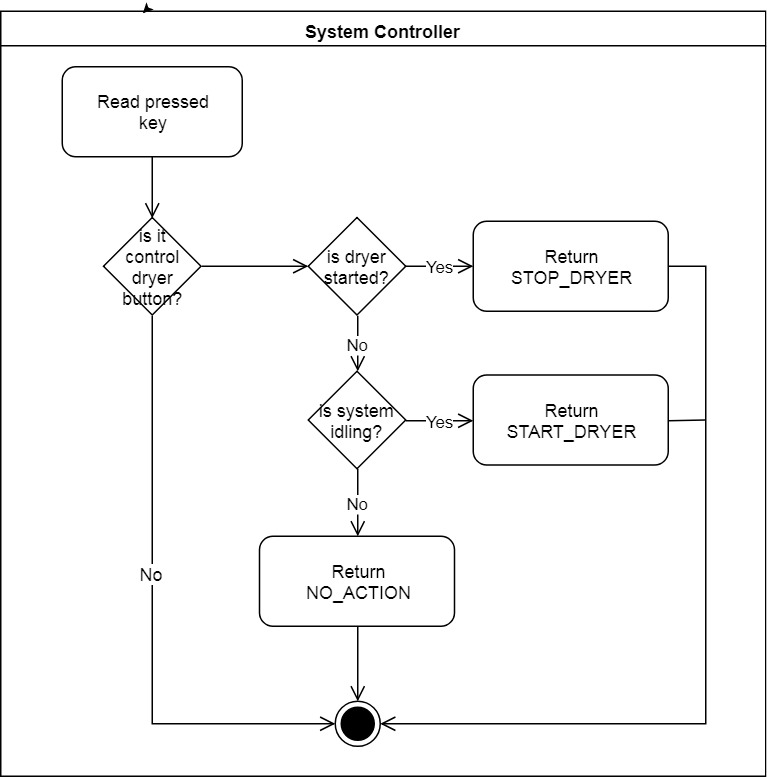
**Summary:** This diagrams show how system control dryer.

Figure 71: Determine dryer action activity diagram

## Interface

### D:\Capstone Document\User Interface\Screen - Login.PNGGuest Interface

Figure 72: <Guess> Login screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Field Name | Description | Read Only | Mandatory | Control Type | Date Type | Length |
| 1 | txtUsername | Username | No | Yes | TextBox | String | 64 |
| 2 | txtPassword | Password | No | Yes | TextBox | String | 128 |
| 4 | lblHeader | Screen Header | N/A | Yes | Label | String | N/A |

Table 98: <Guess> Login screen fields table

**Button/Hyperlinks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Description | Validation | Outcome |
| 3 | btnLogin | Login Button | N/A | Post user info to login |
| 5 | btnHelp | Help button | N/A | Navigate to help screen |

Table 99: <Guess> Login screen buttons/hyperlinks table

### User Interface

#### Home Screen

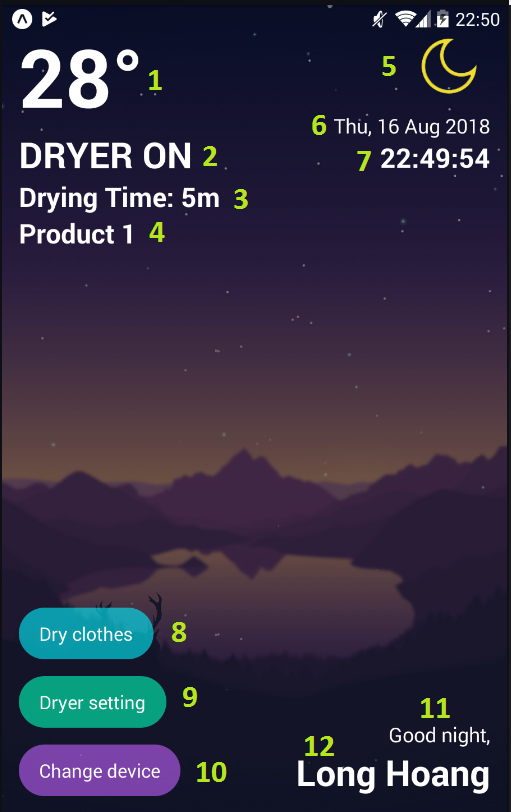


Figure 73: <User> Home screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Field Name | Description | Read Only | Mandatory | Control Type | Date Type | Length |
| 1 | lblTemperature | Temperature | Yes | Yes | Label | String | N/A |
| 2 | lblSysStatus | System status | Yes | Yes | Label | String | N/A |
| 3 | lblDryingTime | Dryer timer | Yes | No | Label | String | N/A |
| 4 | lblSelectedPrd | Selected Product | Yes | Yes | Label | String | N/A |
| 5 | imgWeather | Weather | Yes | Yes | Image View | Image | N/A |
| 6 | lblDate | Current date | Yes | No | Label | String | N/A |
| 7 | lblTime | Current time | Yes | No | Label | String | N/A |
| 11 | lblGreeting | Greeting | Yes | No | Label | String | N/A |

Table 100: <User> Home screen fields table

**Buttons/Hyperlinks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Description | Validation | Outcome |
| 8 | btnDCControl | Control Dc | N/A | Open control dc dialog |
| 9 | btnDryerControl | Control Dryer | N/A | Open control dryer dialog |
| 10 | btnChangePrd | Change control product | N/A | Open change product dialog |
| 12 | btnUser | Show user’s fullname | N/A | Navigate to user profile |

Table 101: <User> Home screen buttons/hyperlinks

#### D:\Capstone Document\User Interface\Home - Dry Clothes.PNGControl DC Motor

Figure 74: <User> Control DC Motor screen

**Field**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Field Name | Description | Read Only | Mandatory | Control Type | Date Type | Length |
| 1 | lblDialogHdr | Dialog Header | Yes | Yes | Label | String | N/A |
| 2 | lblMessage | Dialog message | Yes | Yes | Label | String | N/A |

Table 102: <User> DC Motor control screen fields

**Buttons/Hyperlinks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Description | Validation | Outcome |
| 3 | btnConfirm | Confirm Button | N/A | Send action to control dc motor and close dialog |
| 4 | btnDecline | Decline button | N/A | Close dialog |

Table 103: <User> DC Motor control screen buttons/hyperlinks

#### Control Dryer

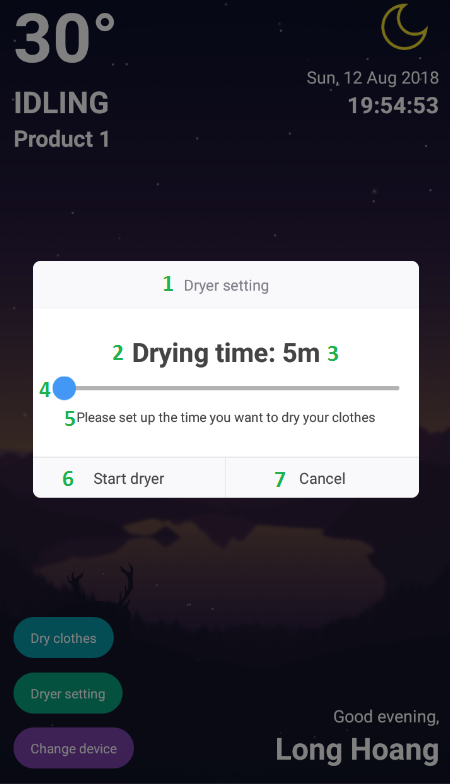


Figure 75: <User> Setup and control dryer screen

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Field Name | Description | Read Only | Mandatory | Control Type | Date Type | Length |
| 1 | lblDialogHdr | Dialog Header | Yes | Yes | Label | String | N/A |
| 2 | lblTimerTitle | Label for timer | Yes | Yes | Label | String | N/A |
| 3 | lblTimer | The timer | Yes | Yes | Label | String | N/A |
| 4 | sldTimer | Timer slider | N/A | Yes | Slider | Number | N/A |
| 5 | lblNote | Note | N/A | Yes | Label | String | N/A |

Table 104: <User> Setup and control dryer screen fields

**Buttons/Hyperlinks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Description | Validation | Outcome |
| 6 | btnConfirm | Confirm Button | N/A | Send action to control dryer and close dialog |
| 7 | btnDecline | Decline button | N/A | Close dialog |

Table 105: <User> Setup and control dryer screen buttons/hyperlinks

#### D:\Capstone Document\User Interface\Home - Select Control Product.PNGSelect Product

Figure 76: <User> Select controlling product screen

**Field**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Field Name | Description | Read Only | Mandatory | Control Type | Date Type | Length |
| 1 | lblDialogHdr | Dialog Header | Yes | Yes | Label | String | N/A |
| 2 | lblProductName | Product Name | Yes | Yes | Label | String | N/A |
| 3 | imgCheck | Check icon | N/A | No | ImageView | Image | N/A |

Table 106: <User> Select controlling product screen fields

**Buttons/Hyperlinks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Description | Validation | Outcome |
| 4 | item | A single item contains product name | N/A | Update item selected and close dialog |

Table 107: <User> Select controlling product screen buttons/hyperlinks

#### D:\Capstone Document\User Interface\Screen - Profile.PNGUser Profile

Figure 77: <User> User's profile screen

**Field**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Field Name | Description | Read Only | Mandatory | Control Type | Date Type | Length |
| 1 | lblHeader | Screen Title | Yes | Yes | Label | String | N/A |
| 2 | lblUsername | Username input title | Yes | Yes | Label | String | N/A |
| 3 | txtUsername | Username | No | Yes | TextBox | String | 64 |
| 4 | lblPassword | Password input title | Yes | Yes | Label | String | N/A |
| 5 | txtPassword | Password | No | Yes | TextBox | String | 128 |
| 6 | lblName | Name input title | Yes | Yes | Label | String | N/A |
| 7 | txtName | User’s name | No | Yes | TextBox | String | 255 |
| 8 | lblAddress | Address input title | Yes | Yes | Label | String | N/A |
| 9 | txtAddress | User’s address | No | Yes | TextBox | String | 1024 |
| 10 | lblPhone | Phone input title | Yes | Yes | Label | String | N/A |
| 11 | txtPhone | User’s phone | No | Yes | TextBox | String | 32 |
| 12 | lblEmail | Email input title | Yes | Yes | Label | String | N/A |
| 13 | txtEmail | User’s email | No | Yes | TextBox | String | 255 |

Table 108: <User> User's profile screen fields

**Buttons/Hyperlinks**

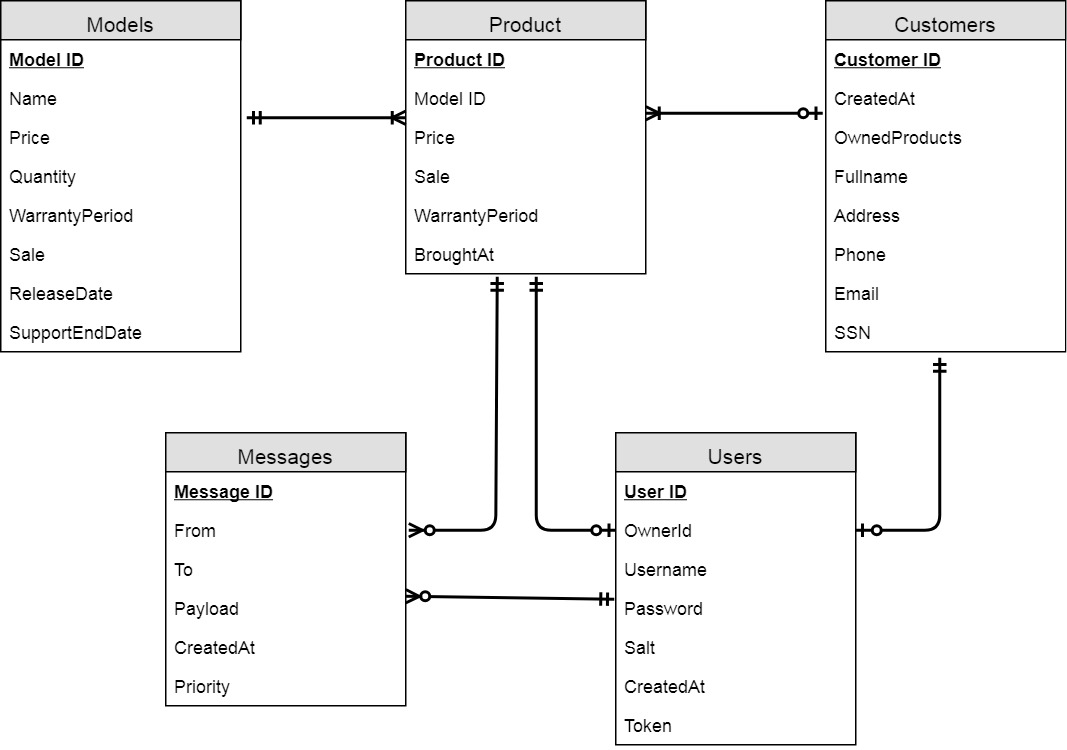
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Field Name | Description | Validation | Outcome |
| 14 | btnSave | Save button | N/A | Save user data and navigate to home screen |
| 15 | btnLogout | Logout button | N/A | Logout user and navigate to login screen |

Table 109: <User> User's profile buttons/hyperlinks

## Database Design

### Entity Relational Database (ERD)

Figure 78: Entity Relational Database



### Data Dictionary

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

Table 110: Entity diagram data dictionary

## Algorithms

### System Control

#### Definition

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

#### Define Problem

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

#### Solution

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

#### Pros & Cons

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

#### Algorithm Complexity

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

#### Overview Flowchart

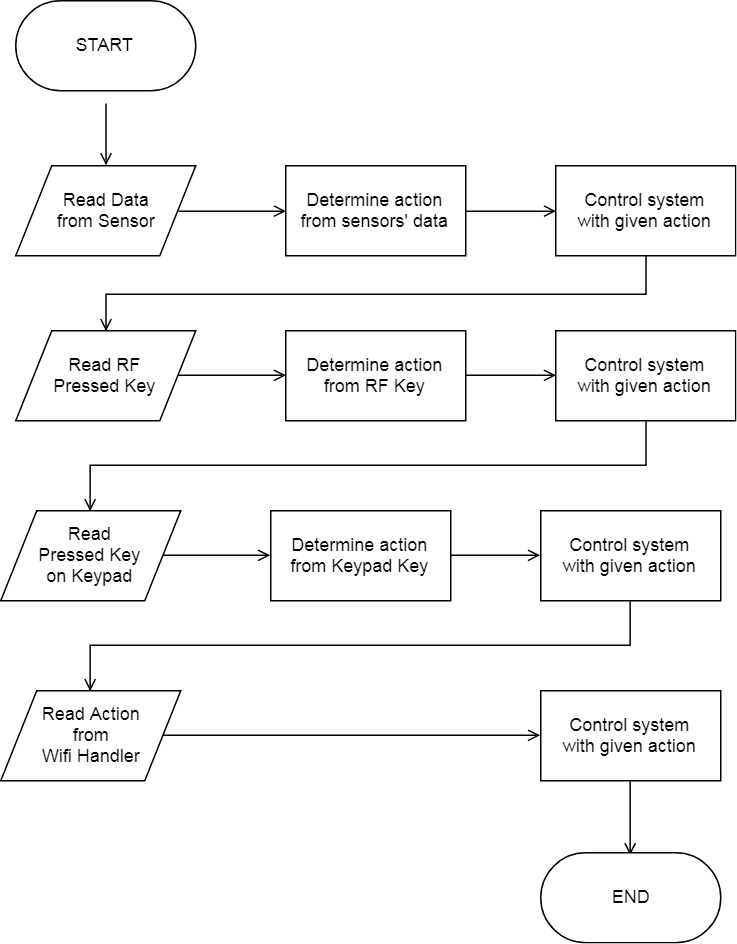


Figure 79: System Control overview flowchart

# System Implementation & Testing

## Introduction

### Overview

This section describes the approach and methodologies used by our team to plan, organize and manage the testing of DCDCS System.

It provides detail all necessary information about the implementation and testing procedure included test plans, test case description, test case procedure, expected output and test result pass/fail criteria as well as a testing flow to cover all possible cases.

### Test Approach

* **Goal:** Test all features in the whole DCDCS based on core flow.
* **Method:** black-box testing.
* **Technique:** boundary value analysis, pair testing, error guessing, user case testing.

The testing for this project will consists of Integration System test level.

System testing is focused on assessing the system’s reliability and the process of testing an integrated system to verify that it meets specified requirements.

This testing will determine if the results generated by information systems and their components are accurate and that the system performs according to specifications.

* Testing will cover functionality testing for DCDCS changes through the use of the test interface. This will validate base functions of the new code as it relates to the standard of presentation for data and user entered data.

This process is concerned with finding errors that result from unanticipated interactions between components and component interface problems.

## Database Relationship Diagram

### D:\Capstone Document\Diagrams\image\Physical Database\Message Queue.PNGPhysical Diagram

Figure 80: Manage message to communicate between clients

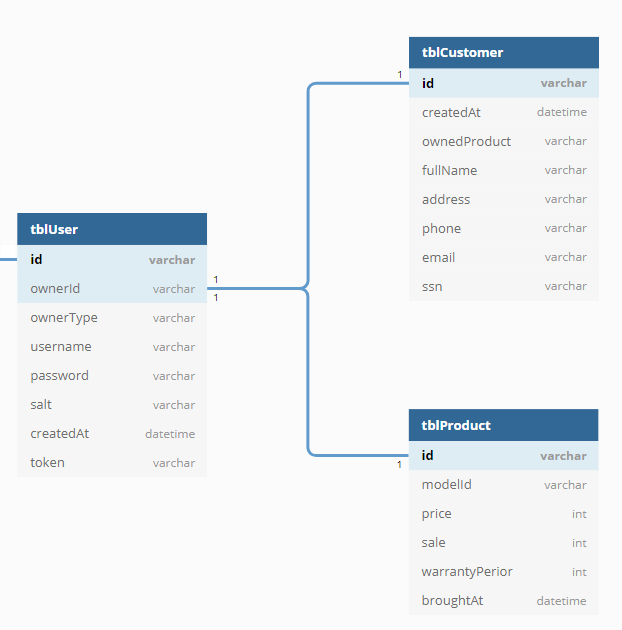


Figure 81: Manage user

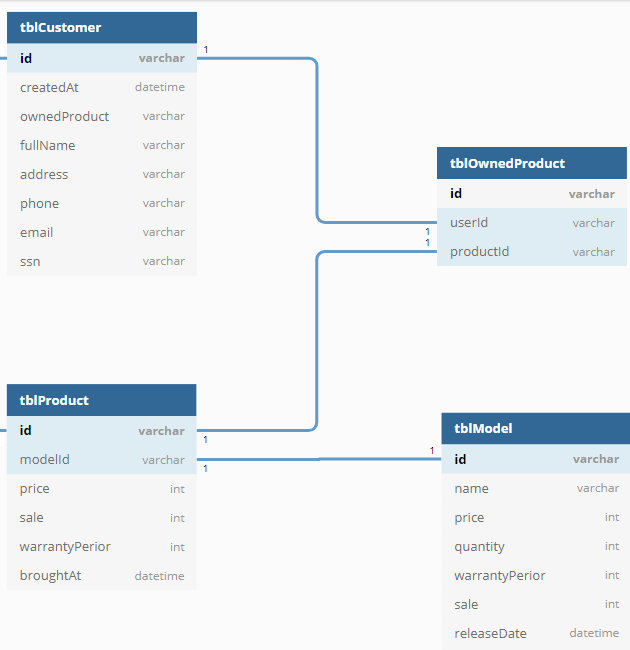


Figure 82: Manage customer and product information

### Data Dictionary

|  |  |
| --- | --- |
| Table Name | Description |
| tblUser | Contains information of a system’s user |
| tblCustomer | Contains information of a customer |
| tblProduct | Contains information of a product |
| tblOwnedProduct | Contains which customer owns which product |
| tblModel | Contains model of a product |
| tblMessage | Contains a message to communicate with another clients |
| tblPayload | Contains payload of a message |
| tblMessageUser | Contains information of sender and receiver of a message |

Table 111: Description of physical database

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table Name | Attributes | Description | Domain | Allow Null? |
| tblUser | id (PK) | Random generated id for each user | varchar(32) | No |
| ownerId | Owner of this user | varchar(32) | No |
| ownerType | Type of the owner. Can be product or customer | varchar(32) | No |
| username | Unique name for user | varchar(64) | No |
| password | Password of user | varchar(128) | No |
| salt | Security salt to secure password | varchar(128) | Yes |
| createdAt | Date that user was created | Datetime | No |
| token | User access token | varchar(8192) | No |
| tblCustomer | Id (PK) | Random generated id for each customer | varchar(32) | No |
| createdAt | Date that customer was created | Datetime | No |
| ownedProduct | Product that customer owns | varchar(32) | No |
| fullname | Full name of customer | varchar(255) | No |
| address | Address of customer | varchar(255) | No |
| phone | Phone of customer | varchar(32) | No |
| email | Email of customer | varchar(255) | Yes |
| SSN | Social Security Number of customer | varchar(255) | No |
| tblProduct | Id (PK) | Random generated id for each customer | varchar(32) | No |
| modelId | Model ID references to model of a product | varchar(32) | No |
| price | Price of a single product | int | No |
| sale | Sale percent of a product | int | Yes |
| warrantyPeriod | Warranty period in month | int | Yes |
| broughtAt | Date that product was brought | Datetime | No |
| tblOwnedProduct | Id (PK) | Random generated id | varchar(32) | No |
| userId | User Id | varchar(32) | No |
| productId | Product Id belongs to User Id | varchar(32) | No |
| tblModel | Id (PK) | Random generated id | varchar(32) | No |
| name | Name of the model | varchar(255) | No |
| price | Price of all product belongs to this model | Int | No |
| quantity | Quantity of products of a model | Int | No |
| warrantyPeriod | Warranty period of all product of a model | Int | No |
| sale | Sale percent of a model | Int | No |
| releaseDate | Model’s release date | Datetime | No |
| endSupportDate | The date that model will not receive support from manufacturer | Datetime | Yes |
| tblMessage | Id (PK) | Random generated id | varchar(32) | No |
| from | Sender information | varchar(32) | No |
| to | Receiver information | varchar(32) | No |
| payload | Payload to send | varchar(32) | No |
| priority | Priortity of message | int | No |
| createdAt | Date that customer was created | Datetime | No |
| tblPayload | Id (PK) | Random generated id | varchar(32) | No |
| action | Action to perform | varchar(32) | No |
| data | Additional data goes along with action | varchar(1024) | No |
| tblMessageUser | Id (PK) | Random generated id | varchar(32) | No |
| userId | User ID | varchar(32) | No |
| deviceId | ID of using device of current userId | varchar(128) | No |

Table 112: Attribute description for physical database

## Performance Measures

### Control System from Android App Performances

**Request time performance**

|  |  |  |
| --- | --- | --- |
| Network Type | System | Request Time |
| Wi-Fi  (Ping 2, Down: 20Mb, Up: 15Mb) | Android Application | 143ms |
| Wi-Fi  (Ping 2, Down: 20Mb, Up: 15Mb) | DCDCS System | 252ms |
| 3G | Android Application | 471ms |

Table 113: API request performance

**Response time performance**

|  |  |  |
| --- | --- | --- |
| Network Type | System | Response Time |
| Wi-Fi  (Ping 2, Down: 20Mb, Up: 15Mb) | Android Application | 352ms |
| Wi-Fi  (Ping 2, Down: 20Mb, Up: 15Mb) | DCDCS System | 461ms |
| 3G | Android Application | 621ms |

Table 114: API response performance

### Control System from RF Remote/Keypad Performance

|  |  |  |
| --- | --- | --- |
| Control Type | Distance | Response Time |
| RF Remote | 0m | 5ms |
| RF Remote | 50m | 32ms |
| RF Remote | 100m | 182ms |
| Keypad | 0m | 5ms |

Table 115: RF Remote/Keypad performance

## Test Plan

The purpose of this section is to verify and ensure that DCDCS meets its design specification and other requirements from user. The following part describe features which will be tested and which won’t.

### Features to be tested

#### Android Application

* Gather system information
* Gather customer information
* Control DC Motor
* Control dryer
* Change control device
* Change user/customer information

#### Hardware System

* Control from Keypad
* Control from RF Keypad
* Auto control
* Connect to Wi-Fi
* Display information

#### API Web Server

* Authorization

### Feature not to be tested

Functions not listed above and manager functional won’t be test

## System Testing Test Case

### State Machine Diagram

#### D:\Capstone Document\Diagrams\image\State Machine Diagram\DCMotor State Machine.jpgControl DC Motor

Figure 83: Control DC Motor State machine diagram

#### Control Dryer

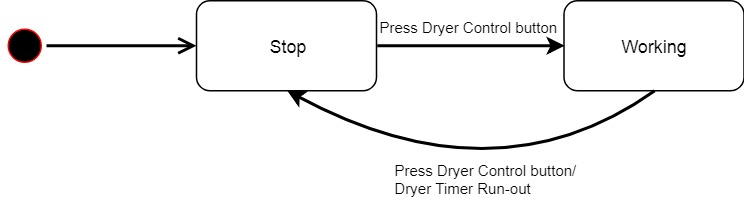


Figure 84: Control dryer State machine diagram

#### Control System

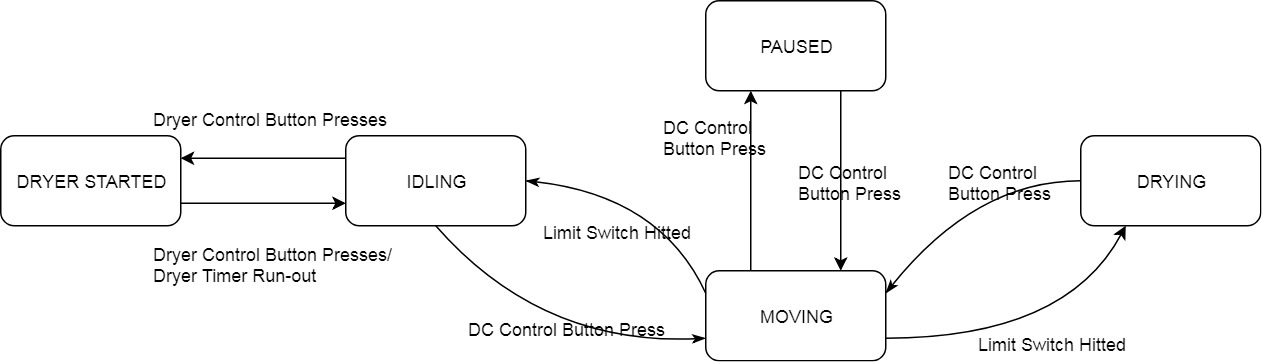


Figure 85: Control system State machine diagram

### Android Application Test Case

#### Gather System Information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| GSI01 | Gather System Information: temperature, humidity, state of system with android app | 1. Login success | * Move to Home screen. * Show correct temperature, status of system | Pass: 8  Fail: 2 | From 15/08/2018  to  18/08/2018 |
| GSI02 | Gather System Information: temperature, humidity, status of system with android app but the hardware is turned off | 1. Login success | * Move to Home screen. * temperature, humidity, status of system must be N/A | Pass: 9  Fail: 1 | From 15/08/2018  to  18/08/2018 |

Table 116: Get system information test cases

#### Gather Customer Information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| GCI01 | Gather customer Information: temperature, humidity, state of system | 1. Login success 2. Click “Trinh Binh” (name of user is showed beside welcome). | * Show information of user: * Name: Trinh Binh * Address: HCMC * Phone:0907569080 * Email: abc@gmail.com | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |

Table 117: Get customer information test cases

#### Control DC Motor

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| CDCM01 | Control motor  to dry clothes with android app | 1.Login success  2.Click Dry Clothes | * Pull out clothes * Status of system changes to DRYING. | Pass: 9  Fail: 1 | From 15/08/2018  to  18/08/2018 |
| CDCM02 | Control motor  to collect clothes with android app | 1.Login success  2.Click Collect Clothes | * Pull in clothes * Status of system changes to IDLING | Pass: 8  Fail: 2 | From 15/08/2018  to  18/08/2018 |

Table 118: Control dc motor from mobile app test cases

#### Control Dryer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| CD01 | Start dryer  to dry clothes with android app | 1. Login success  2. Click Dryer settings  3. Slide the slider to set time  4. Click Start Dryer | * Dry clothes with given time. * System status changed to DRYER ACTIVATED | Pass: 8  Fail: 2 | From 15/08/2018  to  18/08/2018 |
| CD02 | Stop dryer | 1. Start dryer 2. Click Dryer settings 3. Press Stop Dryer | * Dryer working * Dryer stop * System status changed to IDLING | Pass: 8  Fail: 2 | From 15/08/2018  to  18/08/2018 |

Table 119: Control dryer from mobile app test cases

#### Change Control Device

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| CCD01 | Change control Product | 1.Login success  2.Click Change device is DRYING  3. Select Product 1 in Product list | Product 1 will be showed in home screen and controlled device change to Product 1 | Pass: 10  Fail: 0 | From 15/08/2018  to  18/08/2018 |

Table 120: Change control device test cases

#### Chang user/customer information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| CI01 | Update user/customer information | 1.Login success  2. Click “Trinh Binh” (name of user is showed beside welcome).  3.Enter update information:  Name: Binh Trinh | Home screen will show Binh Trinh and name in profile changes to Binh Trinh. | Pass: 9  Fail: 1 | From 15/08/2018  to  18/08/2018 |
| CI02 | Update user/customer with invalid username | 1.Login success  2.Click “Trinh Binh” (name of user is showed beside welcome).  3.Enter username: “!@#$%#$^$” | Show error that username is invalid | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CI03 | Update user/customer with empty information (except Password) | 1.Login success  2. Click “Trinh Binh” (name of user is showed beside welcome).  3.Remove username | Show error that username cannot be empty | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CI04 | Update username with existed one | 1.Login success  2. Click “Trinh Binh” (name of user is showed beside welcome).  3.Enter username with “crabbycrab” | Show error that username cannot is taken | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CI05 | Update user with invalid password | 1.Login success  2. Click “Trinh Binh” (name of user is showed beside welcome).  3.Enter password with: “1” | Show error that password is invalid | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |

Table 121: Update user/customer information test cases

### Hardware System Test Case

#### Control from RF Remote/Keypad

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| CRFK01 | Control DC motor to dry clothes | 1.Press M | * + - Pull out clothes until limit switch is hit.     - Status of system will change from IDLING to MOVING and DRYING (when limit switch is hit) | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CRFK02 | Control DC motor to collect clothes | 1.Press M | * + - Collect clothes until the inside limit switch is hit.     - Status of system will change from DRYING to MOVING and then IDLING | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CRFK03 | Pause DC motor | 1.Press M  2.Press M before max range  3.Press M | * + - DC Motor should move and then pause for a few seconds and then move again     - System status changed to MOVING, PAUSED, MOVING | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CRFK04 | Start Dryer when system is IDLING | 1.Press D | * + - Start dryer and dry in 30 minutes | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CRFK05 | Start Dryer when system is not IDLING | 1.Press D | Do nothing | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CRFK06 | Set drying time and then start dryer | 1.Press C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png  2.Press C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png  3.Press C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png  4.Press C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png  5.Press D | * + - Increase 10 minutes     - Start Dryer and dry in 40 minutes | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CRFK07 | Set drying time while dyer is running | 1.Press D  2.Press C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png  3.Press C:\Users\Phong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tải xuống.png | * + - Drying time increases 10 minutes | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| CRFK08 | Stop dryer | 1.Start dryer  2.Press D | * + - Dryer working     - Dryer stop working     - Timer set to 30 minutes * System status changes to IDLING | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |

Table 122: RF Remote and Keypad control test cases

#### Auto control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| AC01 | Auto Control when rain | Receive rain signal | Pull in drying clothes | Pass: 9  Fail: 1 | From 15/08/2018  to  18/08/2018 |
| AC02 | Auto Control when night | Receive light density smaller than 5 | Pull in drying clothes | Pass: 8  Fail: 2 | From 15/08/2018  to  18/08/2018 |

Table 123: System auto control test cases

#### Connect to Wi-Fi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| WIFI01 | Connect to Wi-Fi with wrong SSID or Password | 1.Enter SSID  2.Enter Wi-Fi Password | DCDCS Wifi System (Wi-Fi Access Point) is not turned off. | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| WIFI02 | Connect to Wi-Fi with correct SSID and Password | 1.Enter SSID  2.Enter Wi-Fi Password | Wi-Fi connected and access point is turned off | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| WIFI03 | Wi-Fi disconnected | Turn off Wi-Fi | DCDCS Wifi System is turned on | Pass: 5  Fail: 0 | From 15/08/2018  to  18/08/2018 |

Table 124: Wi-Fi configuration test cases

### API Web Server Test Case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Case Procedure | Expected output | Result | Test Date |
| API01 | Authorization without token | Guest request to get user information | * Return 403 * Message: “You don’t allow to perform this action” | Pass: 10  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| API02 | Authorization with fake token | Guest fakes user access token and request to get user information | * Return 403 * Message: “You don’t allow to perform this action” | Pass: 10  Fail: 0 | From 15/08/2018  to  18/08/2018 |
| API03 | Authorization with invalid token | User use his/her token and request to get another user information | * Return 403 * Message: “You don’t allow to perform this action” | Pass: 10  Fail: 0 | From 15/08/2018  to  18/08/2018 |

Table 125: API Web Server test cases

# Software & Hardware User’s Manual

## Installation Guide

### Setup environment

There are some recommend about specifications after testing performance according to recommend development environment from previous section of this document.

#### Hardware Requirements

##### Hardware System Requirement

|  |  |
| --- | --- |
| Hardware | Specification |
| MCU | ATmega2560 |
| Communication | I2C, UART, SPI |
| Power Supply | 12VDC |
| Modules | * RF PT2272/PT2262 * 4x4 Matrix Keypad * Light Sensor BHT1750 * Temperature & Humidity Sensor DHT11 * Rain Sensors * NodeMCU or ESP8266 * DC Motor * H-Bridge * Fan dryer |

Table 126: Hardware system requirement

##### Server Hardware Requirement

|  |  |
| --- | --- |
| Hardware | Specification |
| Ram | 512MB |
| CPU | Intel Xeon X5550 @ 2.67GHz |
| Hard Disk | Minimum 5GB |
| Operating System | CentOS or Debian |

Table 127: Server hardware requirement

##### Android Hardware Requirement

|  |  |
| --- | --- |
| Hardware | Specification |
| Ram | Minimum 512MB |
| Network Connection | Wi-Fi 802.11 a/b/g/n/ac, 3G, 4G/LTE |
| Hard Disk | 16MB free space is required |
| Operating System | Android 4.4 or later |

Table 128: Android hardware requirement

#### Software Requirement

|  |  |  |
| --- | --- | --- |
| Software | Name/Version | Description |
| Environment | NodeJS 10 | Specification for developing android application and API web server |
| Application | DCDCS System | Manage the system |
| DBMS | MongoDB | Used to create & manage the database for system |
| File Archiver | 7zip 18.05 | Unzip and zip files or folders |
| Package Manager | Yarn 1.7 | Package manager for NodeJS |
| Source Control | Git-scm 2.17 | Control, cloning source code |
| IDE | Arduino IDE | Deploying code to hardware |

Table 129: System requirement

### API Web Server Deployment Process

All steps below requires cmd or terminal to complete

**Step 1**: Cloning source code with git

git clone <https://github.com/longhoang0304/DCDCS_Server>

**Step 2**: Change directory to DCDCS\_Server

cd DCDSC\_Server

**Step 3**: Install packages and dependencies with yarn

yarn install

**Step 4**: Start server

yarn start

### Android Application Deployment Process

In a real product environment, user can go to play store / app store on their respective phone and search for our app to install. But for demo sake, in our current environment, user can install our application by follow these steps:

**Step 1**: Go to <https://goo.gl/iQSL2g>

**Step 2**: Go to folder APK Build and download APK that available

**Step 3:** Enable install Third-party application

**Step 4:** Tap to downloaded application so that Android can install it automatically

**Step 5:** Open application and accept all permission required

### System Controller Deployment Process

Some steps below requires cmd or terminal to complete

**Step 1**: Cloning source code with git

git clone <https://github.com/longhoang0304/DCDCS_Arduino>

**Step 2**: Change directory to DCDCS\_Arduino and open in explorer (Windows)

cd DCDSC\_Arduino && explorer .

**Step 3**: Open **SystemController.ino** with Arduino IDE

**Step 4**: Build and upload code to Arduino by pressing **Ctrl + U**

### System API Handler Deployment Process

Some steps below requires cmd or terminal to complete

**Step 1**: Cloning source code with git

git clone <https://github.com/longhoang0304/DCDCS_Wifi>

**Step 2**: Change directory to DCDCS\_Wifi and open in explorer (Windows)

cd DCDCS\_Wifi && explorer .

**Step 3**: Open **WifiController.ino** with Arduino IDE

**Step 4**: Build and upload code to Arduino by pressing **Ctrl + U**

## User Guide

### Hardware Configuration

#### D:\Capstone Document\User Interface\Hardware\Connect To Wifi AP.pngConnect to local Wi-Fi

Figure 86: Connect to local Wi-Fi part 1

|  |  |
| --- | --- |
| Step | Description |
| 1 | Search in your Wi-Fi settings **DCDCS System Wifi** then connect to it with the given password |

Table 130: Connect to local Wi-Fi part 1 - Description

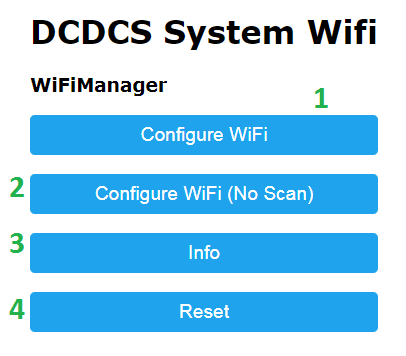


Figure 87: Connect to local Wi-Fi part 2

|  |  |
| --- | --- |
| Step | Description |
| 0 | After connected. Type 10.0.1.1 to go to the main Wi-Fi manager page |
| 1 | Click “Configure WiFi” to go to Wi-Fi selection page |
| 2 | Click “Configure WiFi (No Scan)” to go to manual Wi-Fi configuration without Scan |
| 3 | Click “Info” to check system’s Wi-Fi information |
| 4 | Click “Reset” to reset system’s Wi-Fi configuration |

Table 131: Connect to local Wi-Fi part 2 - Description

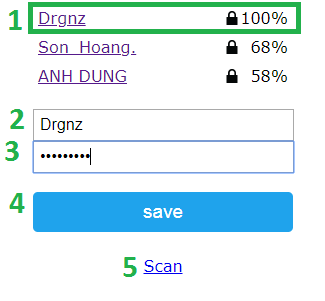


Figure 88: Connect to local Wi-Fi part 3

|  |  |
| --- | --- |
| Step | Description |
| 1 | Select the Wi-Fi station you want to connect |
| 2 | Or you can manually enter the Wi-Fi name or SSID in the first box |
| 3 | Enter the password for the Wi-Fi station |
| 4 | Click “Save” to connect to the given Wi-Fi |
| 5 | If you Wi-Fi doesn’t appear in the list above. Click “Scan” for re-scan the Wi-Fi station |

Table 132: Connect to local Wi-Fi part 3

### Android Application

#### Control DC

##### D:\Capstone Document\User Interface\Manual\Home Dry Clothes - Main.PNGDryer Clothes

Figure 89: Dry clothes part 1

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Dry clothes” button to open dc control dialog |

Table 133: Dry clothes part 1 - Description

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Yes, do it” to tell the system dry clothes and close the dialog |
| 2 | Touch “No” will close the dialog and do nothing else |

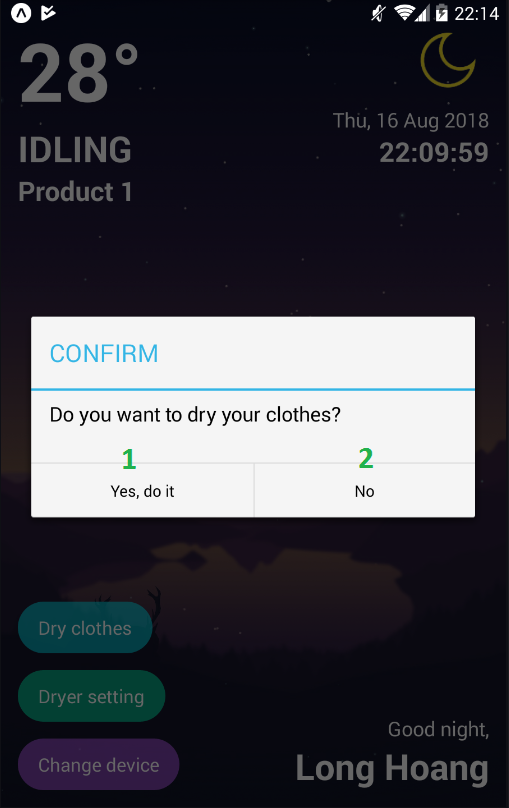
Table 134: Dry clothes part 2 - Description

Figure 90: Dry clothes part 2

##### D:\Capstone Document\User Interface\Manual\Home Collect Clothes - Main.PNGCollect Clothes

Figure 91: Collect clothes part 1

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Collect clothes” button to open dc control dialog |

Table 135: Collect clothes part 1 - Description

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Yes, do it” to tell the system collect clothes and close the dialog |
| 2 | Touch “No” will close the dialog and do nothing else |

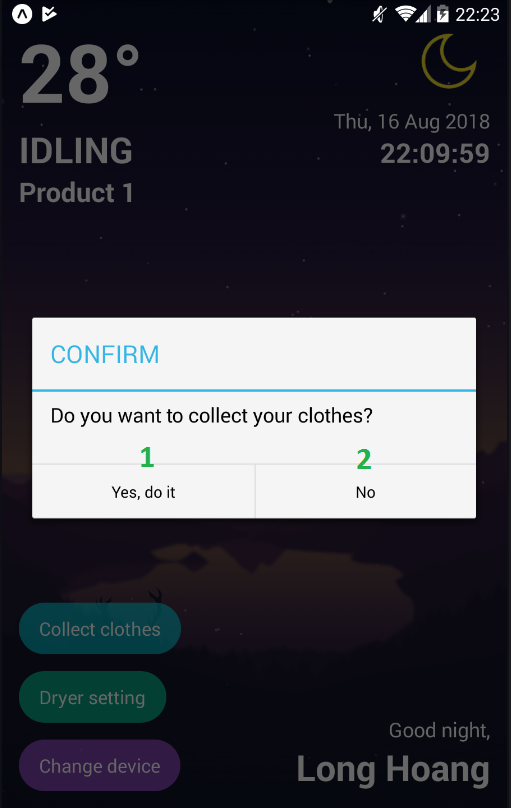
Table 136: Collect clothes part 2 - Description

Figure 92: Collect clothes part 2

#### Control Dryer

##### D:\Capstone Document\User Interface\Manual\Home Dryer Off.PNGSetup timer and start dryer

Figure 93: Setup timer and start dryer part 1

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Dryer setting” button to open dryer control dialog |

Table 137: Setup timer and start dryer part 1 - Description

|  |  |
| --- | --- |
| Step | Description |
| 1 | Slide the slider to setup dryer timer. |
| 2 | Touch “Start dryer” to tell the system start the dryer with the given timer and then close the dialog |
| 3 | Touch “Cancel” to close the dialog without doing anything |

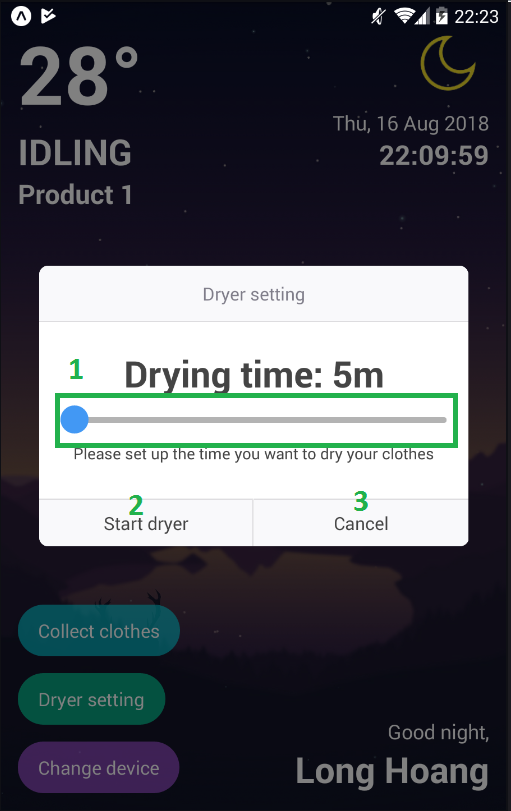
Table 138: Setup timer and start dryer part 2 - Description

Figure 94: Setup timer and start dryer part 2

##### D:\Capstone Document\User Interface\Manual\Home Dryer On.PNGStop the dryer

Figure 95: Stop dryer part 1

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Dryer setting” button to open dryer control dialog |

Table 139: Stop dryer part 1 - Description

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Stop dryer” to tell the system stop drying |
| 2 | Touch “Cancel” will close the dialog and do nothing else |

Table 140: Stop dryer part 2 - Description

Figure 96: Stop dryer part 2

#### D:\Capstone Document\User Interface\Manual\Home Change Product - Main.PNGChange Control Device

Figure 97: Change control device part 1

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Change device” button to open device control dialog |

Table 141: Change control device part 1 - Description

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch “Product 1” item to select device named Product 1. The dialog will close automatically for you |

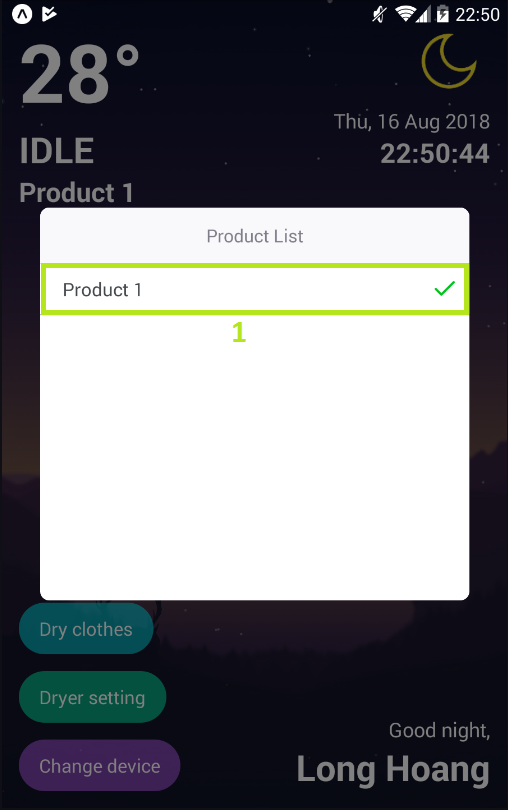
Table 142: Change control device part 2 - Description

Figure 98: Change control device part 2

#### D:\Capstone Document\User Interface\Manual\Home Navigate To Profile.PNGChange User Information

Figure 99: Change user information part 1

|  |  |
| --- | --- |
| Step | Description |
| 1 | Touch the green rectangle area go to Profile screen |

Table 143: Change user information part 1 - Description

|  |  |
| --- | --- |
| Step | Description |
| 1 | Input new username (optional) |
| 2 | Input new password (optional) |
| 3 | Input new full name (optional) |
| 4 | Input new address (optional) |
| 5 | Input new phone (optional) |
| 6 | Input new email (optional) |
| 7 | Click Save Changes to update new information |

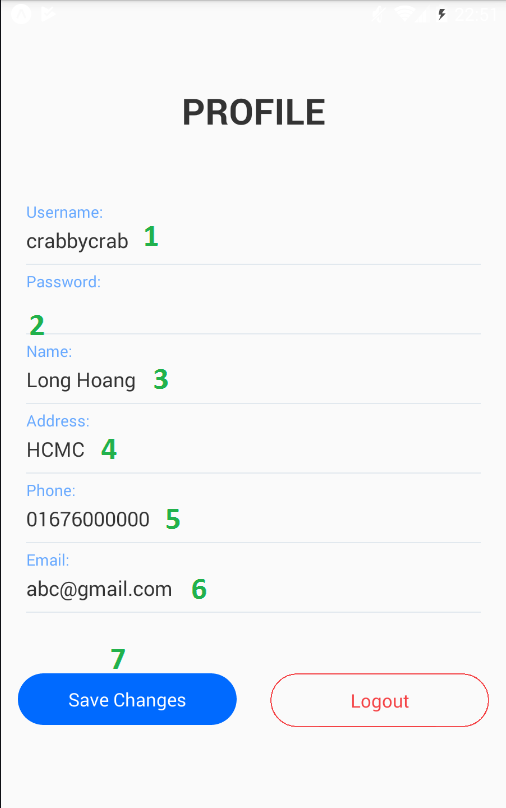
Table 144: Change user information part 2 – Description

Figure 100: Change user information part 2

# 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/>
* 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>