

November 2, 2014

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A1S6

Re: ENSC 305W/440W Design Specification for Smart Irrigation System

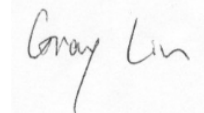
Dear Dr. Rawicz:

Please find the enclosed paper, "Design Specification for Smart Irrigation System", which discusses the smart control box for automated garden irrigation system designed specifically for the Internet of Things (IOT).

This document provides the design specification throughout the entire developing period, but mainly focuses on the proof-of-concept which will be implemented on the first prototype. Moreover, test plans will be explained to measure the product functionality and project process.

There are five experienced engineering students in our team: Team Chase Technologies (TCT). If you have any questions or concerns, please contact me by phone at (778)881-5322 or by email at yuhengl@sfu.ca. I will serve as the contact person for our team. We look forward to your comments and suggestions.

Sincerely,



Gray (Yu Heng) Lin
Chief Executive Officer
Team Chase Technologies

Enclosure: Design Specification for Smart Irrigation System



Team Chase Technologies

Design Specification:

Smart Irrigation System

Project Team: Yu Heng Lin
Chase (Youdao) Wen
Yolanda Wu
Abel Lin
Yuchen Wang

Main Contact: Yu Heng Lin
778-881-5322
yuhengl@sfu.ca

Submitted to: Dr. Andrew Rawicz – ENSC440
Steve Whitmore – ENSC305
School of Engineering Science
Simon Fraser University

Issued date: Nov 06, 2014

Revision: 4.4

Executive Summary

This document provides the details of the design specification for the smart irrigation system throughout the entire developing period. The design specification mainly focuses on the proof-of-concept stage which will be implemented on the first prototype, but possible improvements for the final product will also be discussed.

There are three major components of our product,

- Hardware, including control unit and sensors
- Software, including embedded system and networking
- Mobile App, which allows user has easy access to the system on their phone

The detailed design specification of each component will be presented to meet the functional requirement in the previous document “Functional Specification of Smart Irrigation System” [1]. Justification for the design approaches and description of choice will be fully explained.

The last section of this document provides the test plans on both system and individual components, to examine the product functionality on the first prototype. The details of the test plans can also be found in the appendix section.

Table of contents

Executive Summary.....	ii
1. Introduction.....	1
1.1 Scope	1
1.2 Intended Audience	1
1.3 Requirement Classification.....	2
2. System Specification.....	2
3. Overall System Design	3
3.1 Top Level Design.....	4
3.2 Function Justification	4
4. Irrigation Control Unit	5
4.1 Main Logic Application	6
4.1.1 Measurement Thread	6
4.1.1.1 PCIe Handshaking Agreement	7
4.1.2 Scheduling AI Thread	9
4.1.2.1 Irrigation Time Calculation.....	10
4.1.3 Hardware Listener Thread	13
4.1.4 UI Thread.....	13
4.1.5 Server Connection Thread	14
4.2 Digital Circuit Module.....	16
4.2.1 Module Overview.....	16
4.2.2 PCIe Controller	18
4.2.3 Sensor Controller	19
4.2.4 Temperature Data Capture Logic Block.....	20
4.2.5 Soil Moisture Data Capture Logic Block.....	21
4.2.6 Time Controller	23
4.2.6.1 Control Unit (CU)	24
4.2.6.2 Datapath (DP)	25
4.2.7 LCD Display.....	25
4.3 Enclosure	25

5. Sensor Circuitry.....	27
6. Server (Cloud Service).....	28
6.1 Database server	28
6.2 Server Application	30
6.3 Connection Agent Thread	31
6.4 Android Mobile Connection API.....	33
7. Mobile Application	34
7.1 Mobile App Design	34
7.2 Login Page and Signup Page.....	35
7.3 Operation Page.....	36
7.4 Setting Page.....	37
7.5 Historical Page	38
8. Test Plan	39
8.1 Hardware Test Plan	39
8.1.1 Soil Moisture Sensor Test	39
8.1.2 Temperature Sensor Test.....	39
8.1.3 Valve Test.....	39
8.1.4 Control Box Test.....	39
8.2 Server Test Plan	40
8.2.1 Data Transmission.....	40
8.2.2 Privacy	40
8.3 Mobile App Test Plan	40
8.3.1 Login Test Page	40
8.3.2 Setting Page Test.....	41
8.3.3 Operation Page Test	41
8.3.4 History Page Test	41
8.4 Operation System Test Plan	41
8.4.1 Data Analyze	42
8.4.2 Watering Schedule.....	42
8.4.3 Server Connection.....	42
8.5 Integrated Test Plan	42

8.5.1	Soil Moisture Factor Test	42
8.5.2	Weather Factor Test	43
8.5.3	Self-operation Test.....	43
8.5.4	User Command Test.....	43
8.5.5	Overall Stability	43
9.	Conclusion	43
10.	Reference.....	44
Appendix – Test Plan Sheets		45
	Hardware Test Sheet.....	45
	Server Test Sheet.....	46
	Mobile App Test Sheet	47
	Operation System Test Sheet.....	48
	Integrated Test Sheet.....	49

List of Figures

Figure 1: System Overview.....	3
Figure 2: Block Diagram of DE2i-150	5
Figure 3: Flowchart of Measurement Thread	6
Figure 4: Handshaking Process	8
Figure 5: Flowchart of Scheduling AI Thread	9
Figure 6: Water Pattern.....	11
Figure 7: Flowchart of Hardware Listener Thread	13
Figure 8: Flowchart of Server Connect Thread Part 1.....	14
Figure 9: Flowchart of Server Connect Thread Part 2.....	15
Figure 10: Flowchart of Server Connect Thread Part 3.....	15
Figure 11: Block Diagram of Module Overview	17
Figure 12: Block Diagram of PCIe Controller.....	18
Figure 13: FSM of PCIe Controller.....	19
Figure 14: Block Diagram of Sensor Controller.....	20
Figure 15: AM2303 Output Timing	20
Figure 16: FSM of Temperature Data Capture	21
Figure 17: MCP3008 ADC Output Timing	22
Figure 18: FSM of Soil Moisture Data Capture	22
Figure 19: Block Diagram of Time Controller.....	23
Figure 20: FSM of CU in Timer Controller	24
Figure 21: FSM of DP in Time Controller.....	25
Figure 22: Enclosure for Irrigation Control Unit.....	26
Figure 23: Dimension of the Device Enclosure	26
Figure 24: Schematic of Sensor Circuitry	27
Figure 25: PCB Layout for Sensor Circuitry	28
Figure 26: Database Design Table	29
Figure 27: Database Relation Schema	29
Figure 28: Flowchart of Server Application	30
Figure 29: Flowchart of Connection Agent Thread (Case 1).....	31
Figure 30: Flowchart of Connection Agent Thread (Case 2).....	32
Figure 31: Flowchart of Connection Agent Thread (Case 3).....	33
Figure 32: The Overall App Structure	34

Figure 33: Login Page and Signup Page	35
Figure 34: The Operation Page	36
Figure 35: The Setting Page	37
Figure 36: The Historical Page.....	38

List of Tables

Table 1: PCIe Transmission Content Summary	7
Table 2: Available Water Hold Capacity.....	10
Table 3: Root Depth of Certain Plants.....	10
Table 4: MAD% for Various Plants	11
Table 5: Information and Position on LCD display	25

Glossary

APP	– Application
AWC	– Available Water Hold Capacity
CU	– Control Unit
DP	– Datapath
IOT	– Internet of Things, IOT is a scenario that objects are capable transferring data to other objects over a network without human interaction [2]
LCD	– Liquid-Crystal Display
LED	– Light-Emitting Diode
MAD	– Management Allowable Deficit
OTA	– Over The Air
PCB	– Printed Circuit Board
RAM	– Random-Access Memory
UI	– User Interface

1. Introduction

The project, “C-sprinkler”, is a smart irrigation system which has independent AI, remote control ability and cloud service. The goal of design is to help users to free their hands and still maintain their garden well even when they are away from their homes. The C-Sprinkler will still retain some functions of a tradition irrigation system, such as timers and changing directions of the sprinkler. The design specifications of each component for the C-Sprinkler, as proposed by Team Chase Technologies, are described in this design specification document.

1.1 Scope

This document states that design requirements must be implemented into the design of C-Sprinkler to meet the Functional Specification for Smart Irrigation System [1]. The design specification requirements are included for each developing stage from proof-of-concept to the first prototype to the final product. These technical details and test plans will be used as references throughout the entire developing period.

1.2 Intended Audience

This document is intended to be used by all members of the Team Chase Technologies, to ensure that design specification meets expectations throughout the entire project. The CEO shall refer to the design requirements to measure the developing progress. Hardware/Software engineers shall refer to the specification while implementing designs. Test engineers shall refer to the testing section in the debugging process from small components to the whole system.

1.3 Requirement Classification

The reference requirements in this document are taken from the function specification [1]. The following conventions have been used to denote the function requirements:

[R#-P#] A function requirement

R# - Denotes the section number

P# - Denotes the priority of the function requirement

- Priority I – Feature must be implemented to the proof-of-concept system
- Priority II – Feature must be implemented to the first prototype
- Priority III – Feature must be implemented to the final production versions

2. System Specification

The C-Sprinkler will use various information to irrigate gardens automatically without requiring human inputs. The system collects data from sensor and online open source, and calculate on the cloud server to determine the best timing for irrigation and the watering amount.

The user can also manually change settings to their preference, either through the mobile app or on actual control box. All settings and history records will be uploaded to cloud servers securely. The system will also give the user warning when there is a sudden change in the environment to ensure the plant's health.

3. Overall System Design

C-Sprinkler has three main components that allow user to remotely control the system on the mobile app or on the control box itself. **Figure 1** below shows how user can interact with the system. Solid lines indicate direct access from user and dotted lines indicate the connection behind the UI.

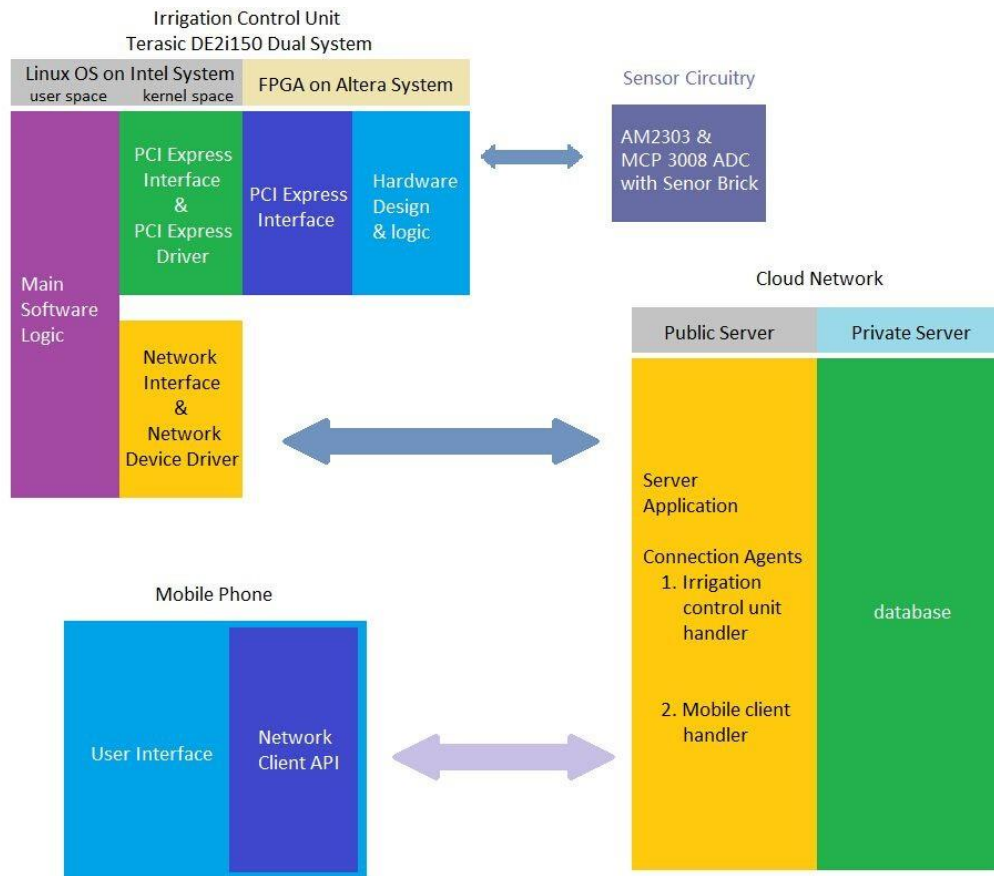


Figure 1: System Overview

The mobile app and control box provides the interface between the user and system. User can use mobile app to remote control the sprinkler and check the current status as long as the mobile phone is connected to the internet. The user may send the command from the phone to the server and server will push the command to the control box. Some functions will be provided, such as irrigation scheduling, turning on/off of the valve and environment status checks. The user can also directly interact with the control box. There is a LCD display on the front of the box which provides various information and the control box is connected with the server to receive commands.

3.1 Top Level Design

The system design consist of three major sections

- Hardware development aimed to develop a customized control box and sensors to monitor the environment
- Software development aimed to develop an embedded system on control box and set up the proper cloud service on server
- Mobile App development aimed to develop an mobile app on Android platform which is capable to send/receive command to/from server

3.2 Function Justification

The goal of the design is to propose an affordable, user friendly, and reliable irrigation system to replace traditional sprinklers. Therefore, functionality should be chosen to meet user's demands and extra features to draw the attention of potential buyers.

The C-Sprinkler can turn the vale on or off to adjust the water flow and can also control spraying direction, which is an important function of traditional sprinklers. The large LCD with a few buttons that will be assembled onto the control box provides easy access for people such as elderly who prefer to use it instead of the app.

For the majority of our customers, which is the younger generation, an app will be deployed to control the system and can be used on any smart phone. Nowadays, the concept of IOT is widely used and people use their mobile phones more than ever before, therefore the development of a mobile application that grants remote access to the C-Sprinkler will be necessary. Currently the mobile app is only offered on the Android platform due to open source, but the mobile app on other platforms will be developed to expand the customer base on the final product.

The other main feature of the C-Sprinkler is its automated irrigation. The system will collect information to determine when to irrigate and how much water should be used. To achieve this, the cloud service is implemented into the design. The system will ask the user to provide vegetation information and it will search the vegetation's characteristics in the pre-loaded database on the server. Then the server will use environment data from sensors, weather forecast from online open sources, and vegetation characteristics to determine the schedule of the irrigations. Moreover, some areas may have local regulations regarding to watering or different pricing of water usage during the day, the system will consider such information in determining the schedule.

By choosing cloud service on server, user data will be stored on the server in an encrypted format. The user therefore check the history of irrigations and upcoming schedules on their phone. Furthermore, if there is a future firmware update for the control box, the system can push the update through over the air (OTA) which is more convenient compared to traditional methods such as using a SD card or a CD drive.

4. Irrigation Control Unit

The development of the irrigation control unit is done on De2i-150 FPGA development kit for the proof-of-concept and prototype version. The following block diagram shows the configuration of De2i-150 FPGA. Besides the FPGA chipset, the board also contains Intel processor chipset which possesses with general computer functionalities. Two chipsets are able to communicate with each other through PCIe bus. For the irrigation control unit design, software applications and OS will be implemented by the Intel chipset in the right half of the block diagram. Digital circuits will be designed in Altera chipset represented in the left half of the block diagram.

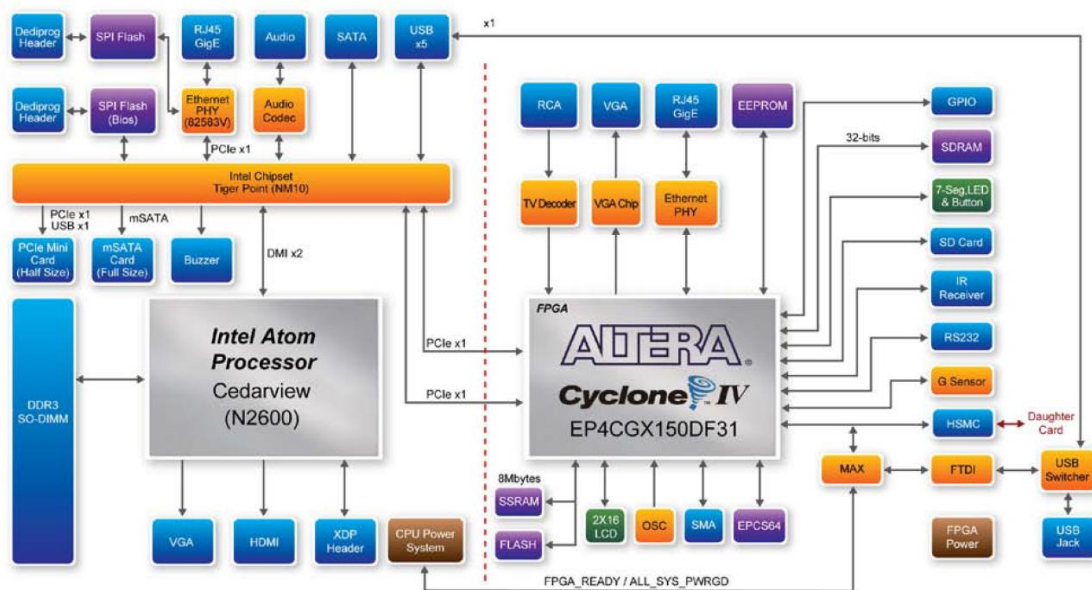


Figure 2: Block Diagram of DE2i-150 [3]

The reasons of using De2i-150 FPGA as the development kit is that De2i-150 FPGA enable developers to design a product involved both embedded software applications and the digital logic circuit on the same platform. Compared to Raspberry PI and Arduino whose hardware components and layout are fixed, developers can customize and reconfigure the digital circuits between drivers and physical components on De2i-150. Such availability not only optimizes the product performance based on desired functionalities, but also ease any design modifications during developing phase.

All components which will be introduced in the following sections are assemble to meet general requirements **[R 1-PI] to [R 7-PII]** from Functional Specification. For a final product version, components will be moved onto a PCB with minimal clearance.

4.1 Main Logic Application

The main logic application contains a main thread and the following child threads:

1. Measurement thread
2. Scheduling AI thread
3. Server connection thread
4. Hardware listener thread
5. UI thread

The main thread is responsible for initializing PCIE driver and child threads, and then it becomes an infinite loop to ensure the child threads are up and running all the time.

4.1.1 Measurement Thread

In order to correctly obtain data from hardware **[R 24-PI]**, the measurement thread is an infinite loop for measurements. It measures the hardware data every 60 minutes and saves the data on the server. It also contains a buffer which is used when internet connection is not available.

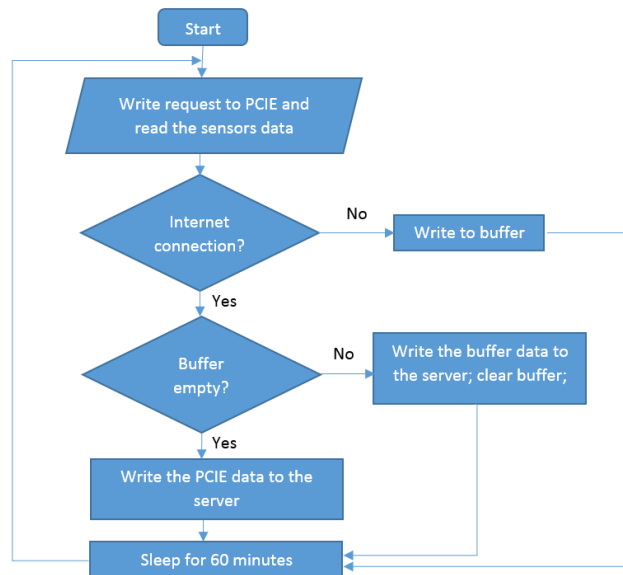


Figure 3: Flowchart of Measurement Thread

4.1.1.1 PCIe Handshaking Agreement

PCIe Handshaking agreement is designed to improve the stability of PCIe communication with hardware to meet [R 23-PI] and [R 24-PI] requirement in Functional Specification. In order for two systems, Altera FPGA and Intel CPU, communicate properly via PCIE, a handshaking algorithm has been created. There are two buses connected to two 32bits memories in our PCIE on each side, and these two memories are for the following two purpose: read for Intel CPU system and write for Intel CPU system, which also means write for Altera FPGA system and read for Altera FPGA system, respectively.

Note: Our software design, the Main Logic Application is reside on Intel CPU system, and our Hardware design is reside on Altera FPGA system.

The following tables shows the design for the handshaking agreement.

System\bits	Read: Intel Write: Altera	System\bits	Read: Altera Write: Intel
0-15	general_data	0-7	time_data
16	Notation 0 Plus 1 Minus	8-9	Notation 00 Off 01 Plus 10 Minus
17 – 18	Type of service 00 OFF 01 Temperature 10 Humidity 11 UserTime	10	ACK for UserTime in general_data
19	ACK for time_data	11	ACK for temperature in general_data
20-31	Not used	12	ACK for humidity in general_data
		13	Request general_data
		14-31	Not used

Table 1: PCIe Transmission Content Summary

There are three types of data that the software will read from hardware.

1. Temperature
2. Humidity
3. UserTime (user adjusted time from push buttons interface)

There is one type of data that the hardware will read from software.

1. Valve activation time

The data can only be sent one followed by another, not concurrently. The transactions are in the following sequence:

1. Sender: sends the data; waiting for ACK from
2. Receiver: gets the data; sends the ACK to sender
3. Sender: received the ACK, and clear the data

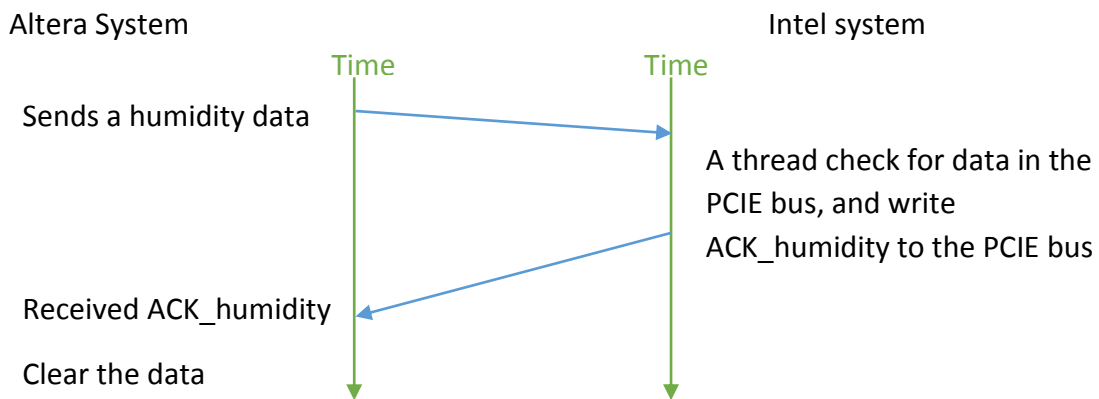


Figure 4: Handshaking Process

4.1.2 Scheduling AI Thread

Scheduling AI thread is a loop that calculates and schedules the time for irrigation to meet the requirement [R 29-PIII]. Scheduling is based on three aspects: user-set timing policy and preferences, measurements of current environmental conditions, and forecasting data from server. At the end of the cycle, the thread goes to sleep until the next available time slot from timing policy.

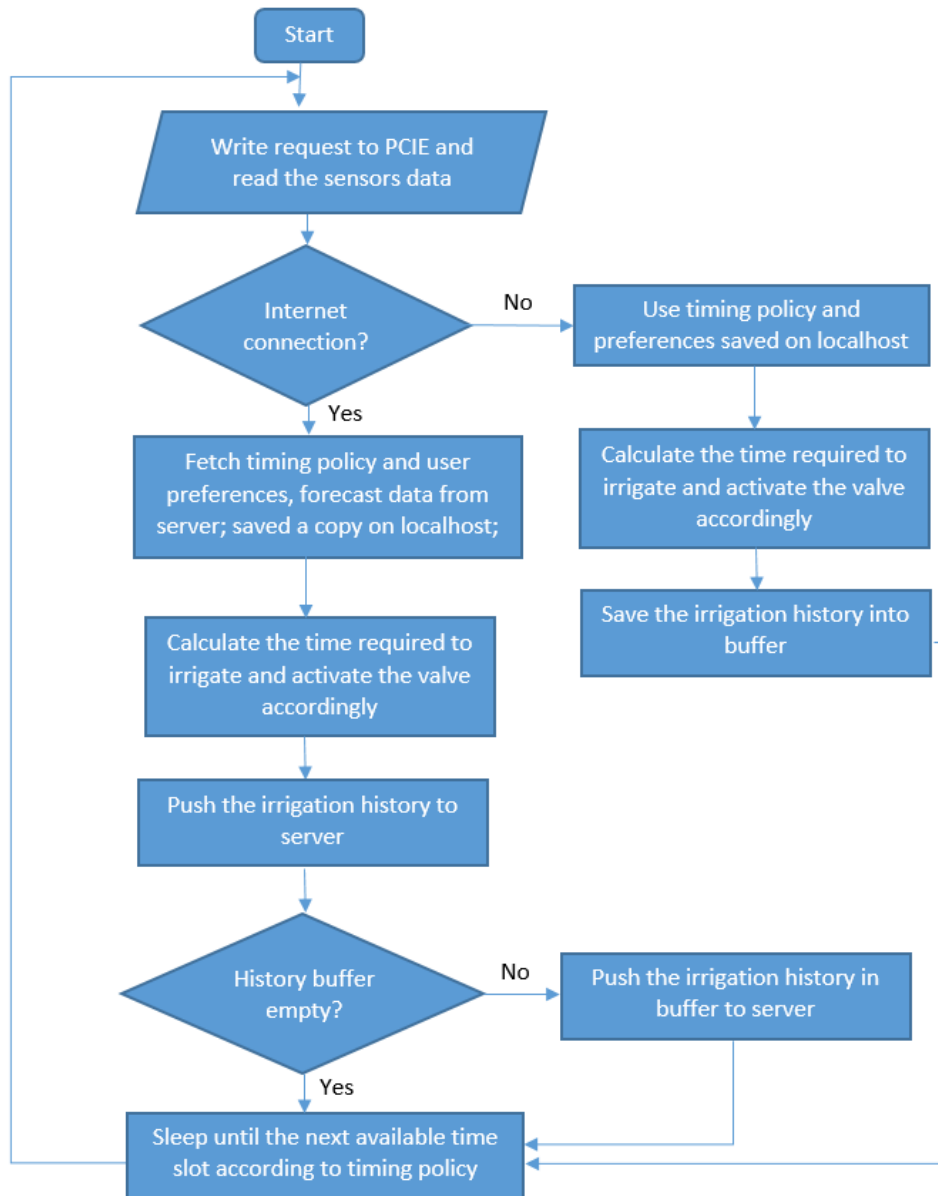


Figure 5: Flowchart of Scheduling AI Thread

4.1.2.1 Irrigation Time Calculation

There are two factors determine how much water the plant needs and when to water. They are the field capacity for water, and the water stress point of the plant. The field capacity of the soil is set as an upper limit and the water stress point is set as the lower limit for watering. Ideally the water content would bounce between these two boundaries. Watering beyond the upper limit would cause waste of water and when water content is below the lower limit, the plant would be thirst for water.

The field capacity for water can be gathered using moisture sensor. Place the moisture sensor at the root depth of selected plant and take the reading after the soil is well watered. Then the formula [6] below would give us the field capacity using the provided root depth in **Table 3**.

$$\text{Probe Measurement} \times \text{Depth of Root} = \text{Field Capacity} \quad (1)$$

The water stress point is determined by the field capacity, the available water hold capacity (AWC) of soil, root depth, and the management allowable deficit (MAD) of the selected plant. The water hold in the soil can be found base on the AWC through the following formula [6]:

$$\text{AWC} \times \text{Depth of Root} = \text{Volume of Water Hold in Soil} \quad (2)$$

Soil Texture	Water Holding Capacity (in water/ft soil)
silt loam	2.5
silt	2.4
silty clay loam	2.3
loam	2.0
very fine sandy loam	1.8
fine sandy loam	1.6
sandy loam	1.4
loamy sand	0.9
sand	0.6

Table 2: Available Water Hold Capacity [5]

Crop	Rooting Depth (ft)
Row Crops	
corn	3
cotton	3
soybean	2
spring grain	3
winter wheat	3
Vegetables	
cole	1.5
pepper	2.0
tomato	2.5
cucurbit	2.5
potato	1.5
Fruit	
strawberry	0.75
berry & bramble	3
wine grape	4
table grape	4
apply, cherry, & pear	4
stone fruit	4
Forages	
alfalfa	4
bemuda grass	4
fescue & orchard grass	2.5
Other	
tobacco	2.5

Table 3: Root Depth of Certain Plants [5]

Then the maximum depletion can be calculated base on the MAD using the following formula [6]:

$$AWC \times \text{Depth of Root} \times MAD = \text{MAX Depletion} \quad (3)$$

Crop	MAD (%)
Row Crops	
corn	55
cotton	65
soybean	50
spring grain	55
winter wheat	55
Vegetables	
cole	45
pepper	30
tomato	40
cucurbit	40
potato	35
Fruit	
strawberry	20
berry & bramble	50
wine grape	65
table grape	30
apply, cherry, & pear	50
stone fruit	50
Forages	
alfalfa	55
bemuda grass	55
fescue & orchard grass	60
Other	
tobacco	60

Table 4: MAD% for Various Plants

Now find out the water stress point by the following formula [6]:

$$\text{Field Capacity} - \text{MAX depletion} = \text{Water Stress Point} \quad (4)$$

The idea watering activity would behave as **Figure 6** which have a field capacity of 14.7 and MAD being 10.5 inch.

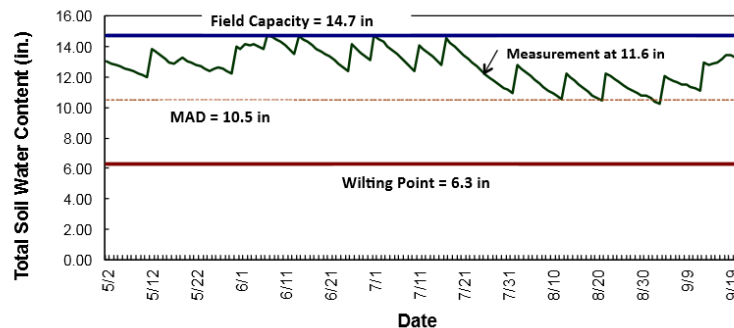


Figure 6: Water Pattern [6]

Due to some areas have time restrictions for irrigation, the user might not able to water at any time. To keep the plant at its best condition, watering in advance should be consider. Through the next formula [4] water consumption can be predicted for the following days.

$$\begin{aligned} \text{amount of water needed for the day } \left(\frac{\text{mm}}{\text{day}} \right) \\ = -0.028 \text{ temp}^2 + 1.7608 \text{ temp} - 22.932 \end{aligned} \quad (5)$$

Where Temp represents temperature in Celsius.

If the remaining water content does not satisfy the water need of the plant until the next available watering date, the water request will be added for today to maintain the humidity of the soil.

4.1.3 Hardware Listener Thread

In order to read sensor data [R 24-PI], the thread check if there is any incoming data from PCIE driver in every 30 seconds.

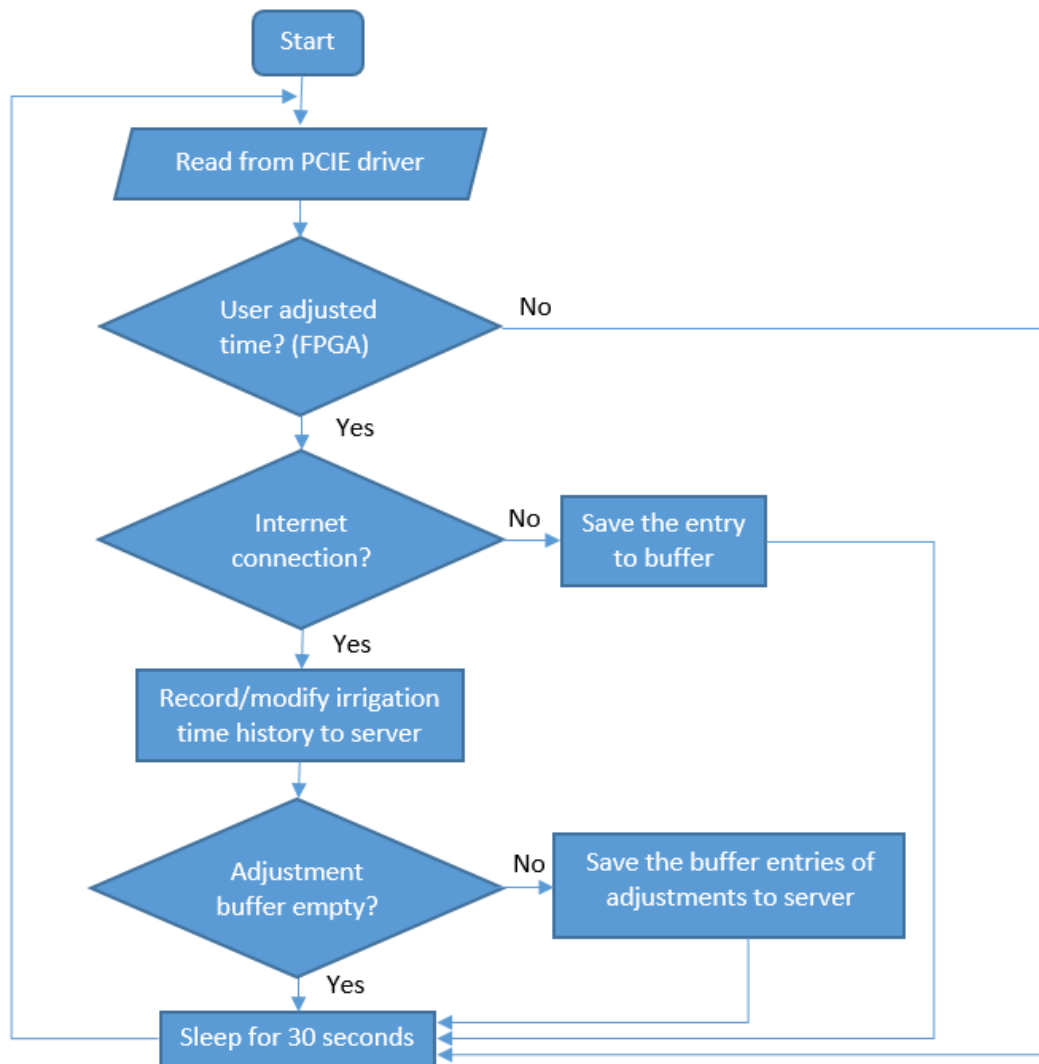


Figure 7: Flowchart of Hardware Listener Thread

4.1.4 UI Thread

UI thread is a loop that will display current status of the program to terminal in real time. The thread will access some shared memory and output to terminal in real time. This is mainly for experimental and debugging purpose.

4.1.5 Server Connection Thread

Server connection thread will satisfy server requirement [R 26-PII] to [R 28-PII] in Function specification. It is responsible for maintaining the connection between the server and itself. The server can send instructions via this connection. To ensure the connection is valid at most times, the design expects server to send ACK every 30 minutes. If data does not arrive on time (check on every 32 minutes), it resets connection to server. If data arrives on time, it resets the timer. If any instruction arrives at any time, it will also reset the timer because the connection is valid.

By checking the Internet availability from other site (e.g. Google.com) in a more frequent interval (i.e. every 1 minute), the system may achieve a better detection of dropping connection. If internet connection is down, drop the current connection and attempt to reconnect to server every 20 seconds

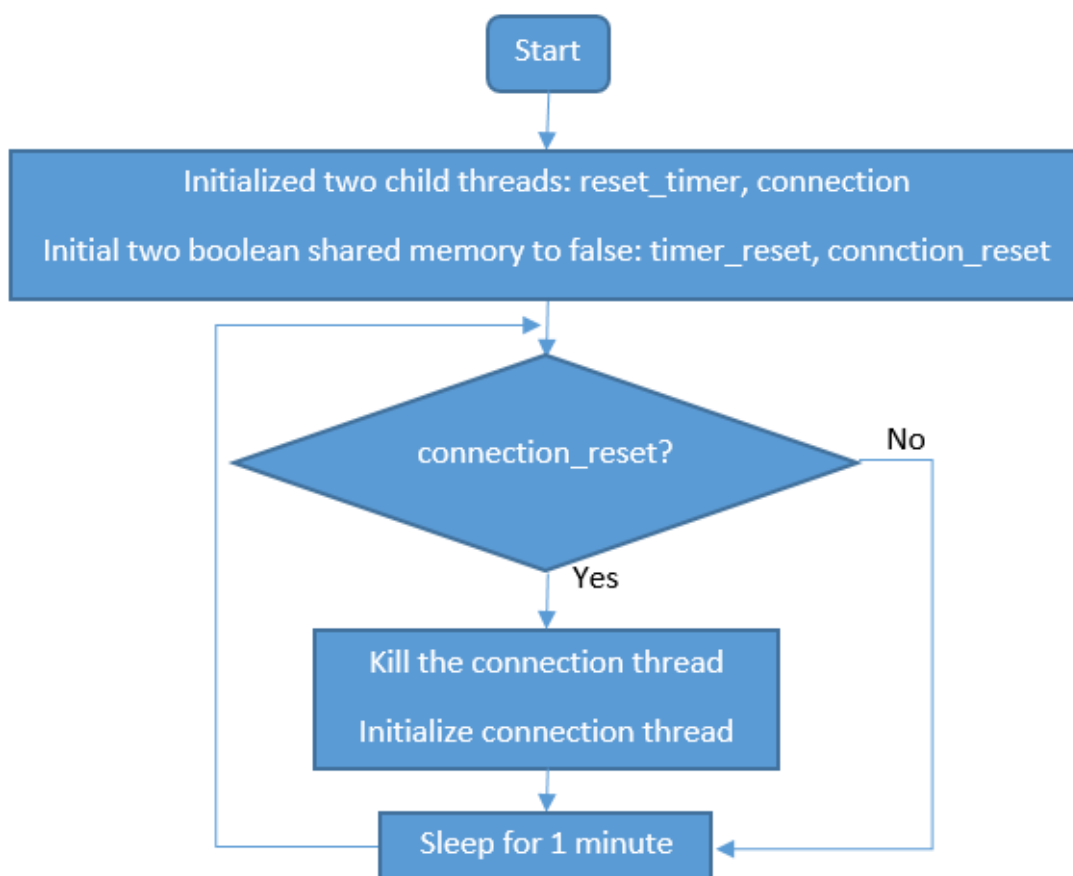


Figure 8: Flowchart of Server Connect Thread Part 1

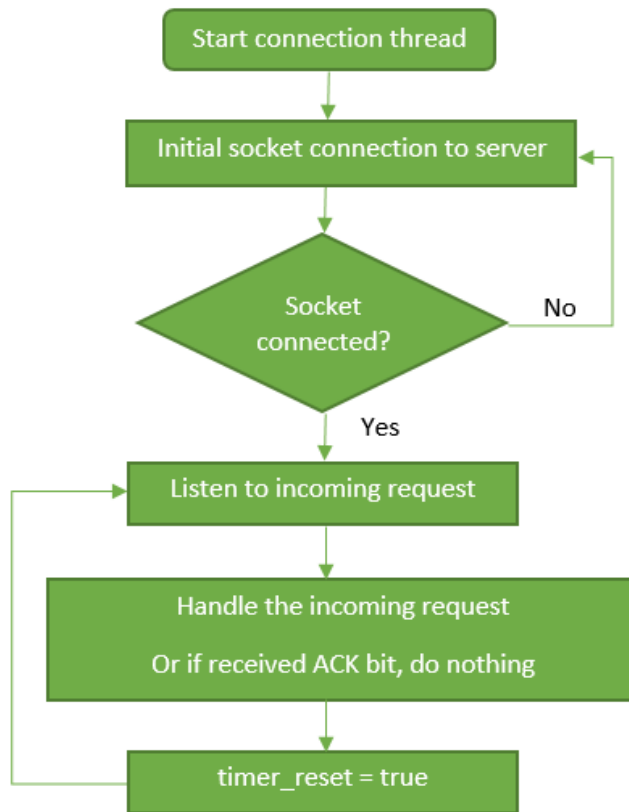


Figure 9: Flowchart of Server Connect Thread Part 2

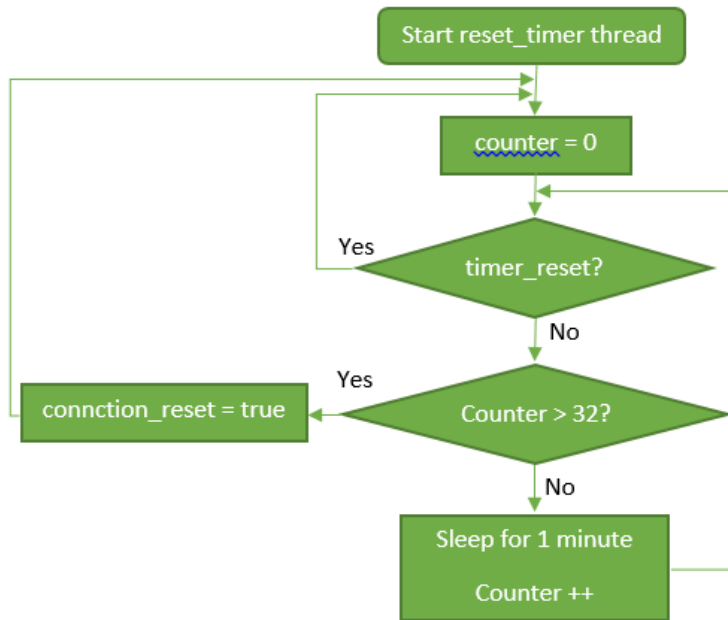


Figure 10: Flowchart of Server Connect Thread Part 3

4.2 Digital Circuit Module

4.2.1 Module Overview

The digital circuit module is a subcomponent within the irrigation control unit. Development is done on Altera FPGA by VHDL programming. Main functionalities of this module include

- Control the valve power either by push buttons or request from Intel processor
- Display sensor results and remaining time of irrigation
- Data capture logic for temperature sensor and soil moisture sensor
- Synchronize the latest changes to the Intel processor through PCI Express interface

Theoretically, expected functionalities can be realized on a single Intel processor, digital circuits only need to direct required signals. One of the reasons of implementing some functionalities by means of digital logic circuit in the presence of the Intel processor is that the Intel processor is relatively unstable due to multithreading, application deficiency, Internet traffic and the Intel processor selection which may all cause uncertainty in performance. However, operations such as the sensor communication process and push button control must be completed in real-time. Any unexpected delays during the sensor communication process will result in a failure in data collection. Delays on push button control while operating the controller will cause a poor user experience.

In contrast, digital logic circuitry is an appropriate option for real time operations including irrigation timer, LCD display, sensor communication and push button control due to its reliability and responsive operating time. The **Figure 11** in the next page demonstrates the overall design for the digital circuit component.

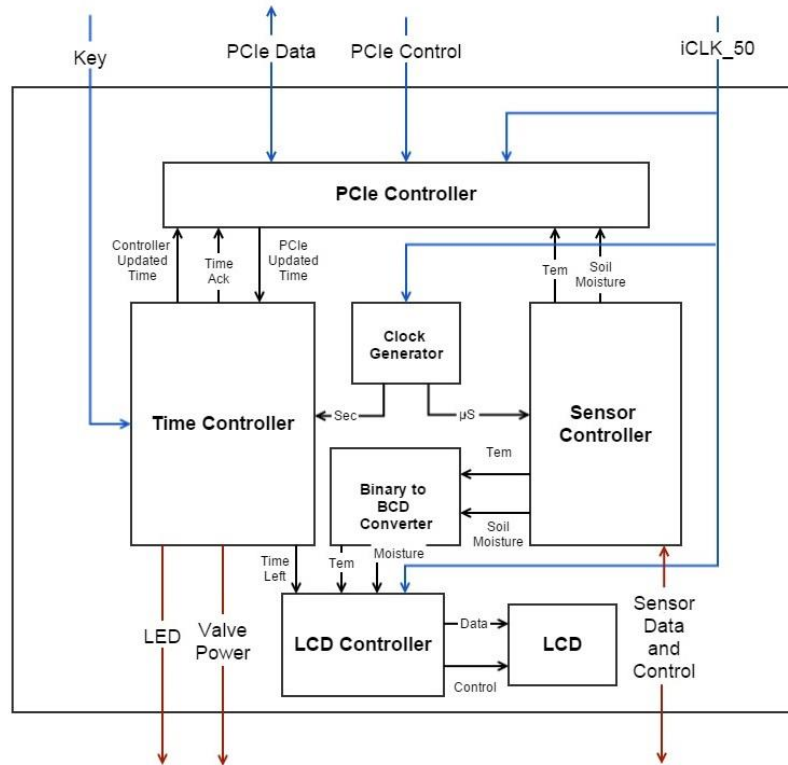


Figure 11: Block Diagram of Module Overview

Most of the functionalities are achieved in the following three major logic blocks: the PCIe controller, the sensor controller and the time controller. The PCIe controller is designed to communicate with Intel processor through PCIe interface in order to receive commands from Intel processor and synchronize latest changes to mobile [R 23-PI]. Aside from the data transmission between the digital circuit component and the Intel processor, the PCIe controller possesses a handshaking process to prevent data loss.

The objective of the sensor controller is to control external sensors, capture sensor signals and store them to registers [R 24-PI]. Both the temperature sensor and the soil moisture sensor apply SPI to transfer output data. Therefore, a precise timing logic for data capture is required based on sensor specification. Resulting data will then be transferred to PCIe controller for AI analysis and LCD display.

After receiving a 'start' request with a non-zero irrigation time, time controller will start to count down in seconds. The irrigation time can be adjusted by push buttons and PCIe under system idle or runtime condition. The controller is also responsible for data synchronization if the irrigation information is updated [R 25-PI]. For instance, the remaining time will be output to LCD and PCIe in real time so that users can acquire latest status. The valve will be powered until remaining time is decreased to zero.

Aside from major logic components, clock generator is developed to convert the input clock rate to the required clock rate referred to the destination component. LCD controller connected to the LCD is used to display time and surrounding information for users. Binary-to-BCD converter converts the binary string into binary-coded-decimal format which is needed for LCD display.

4.2.2 PCIe Controller

Compared to the older PCI, PCIe possesses faster speed, less required I/O pin and smaller physical footprint. The following block diagram shows the architecture of PCIe Controller,

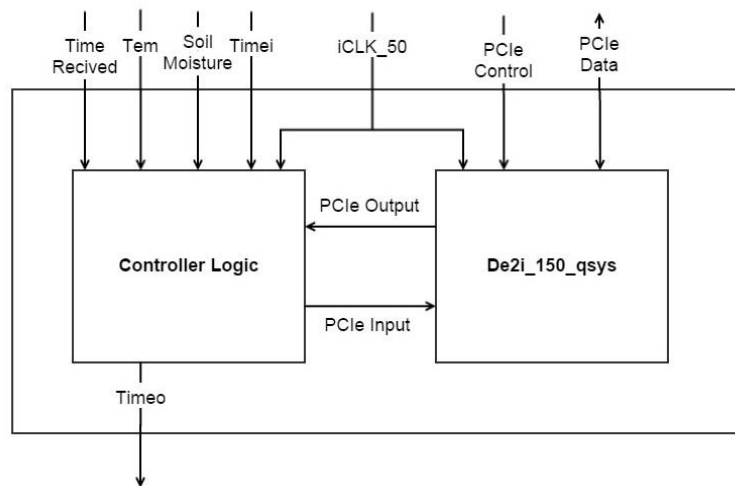


Figure 12: Block Diagram of PCIe Controller

The controller contains two major blocks: the controller logic block and De2i_150_qsys. The purpose of the controller logic block is to receive or output data from or to other digital components, determine the sequence of sending data to PCIe, as well as ensure signals are successfully transferred to the Intel processor by using the acknowledge bit.

De2i_150_qsys is a Qsys module edited from Altera PCIe demonstration code [7] which contains the protocol for data collection and transmission from/to Intel processor through PCIe bus. Using PCIe Qsys module from Altera ease the development process and reduce the length of the development cycle which are good for proof of concept and prototype. However, not all functions in De2i_150_qsys are used and interconnect components of Qsys dramatically increase the quantity of logic gates. According to functional specification requirement **[R 59-PIII]** for the final product, PCIe Qsys needs to be customized to replace De2i_150_qsys or implement other transfer method such as USB and PCI to minimize logic components.

The circuit behavior is summarized in the following FSM,

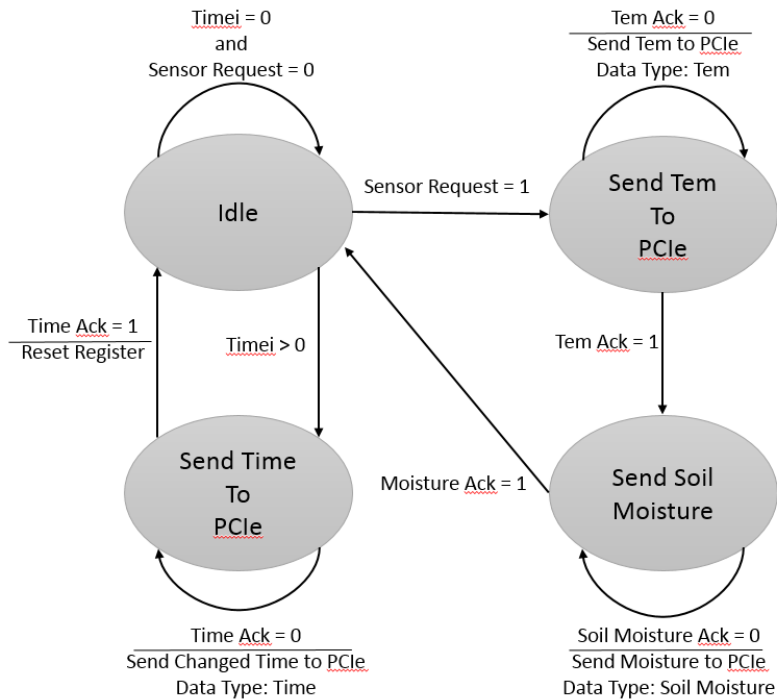


Figure 13: FSM of PCIe Controller

The PCIe controller will send data to Intel processor if one of the following condition comes true,

- Intel processor sends a sensor request to inquire the sensor data
- Irrigation time is set or modified by push button so that latest time needs to be synchronized ($Timei > 0$)

Each sending state will not proceed to the next state (register data will not be erased) until an acknowledge signal is sent by Intel processor. The handshaking process and the description of each individual bit within the PCIe interface are introduced in Software development section and therefore will not be repeated in this section.

4.2.3 Sensor Controller

The Sensor controller's purpose is to activate external sensors and capture sensor data through SPI serial data pin. Due to the different transfer mechanism between sensors, two separated logic block are designed to handle data capturing. Once the sensor data is collected and stored into the temperature and soil moisture registers, results will be output to other digital logic components such as LCD and PCIe controller for further implementation.

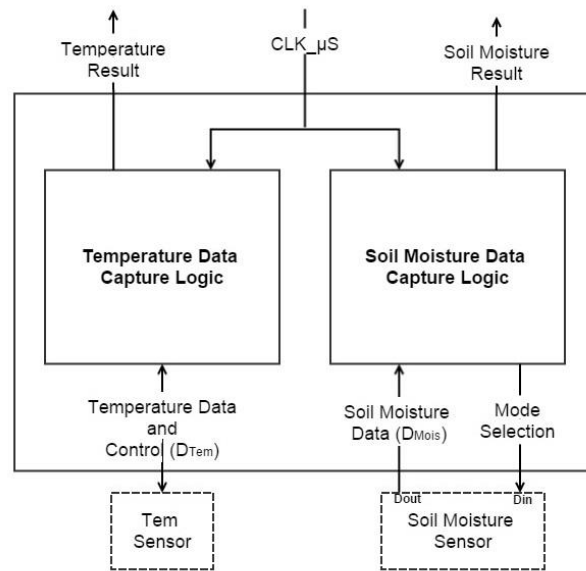


Figure 14: Block Diagram of Sensor Controller

The circuit design of temperature data capture logic is based on AM2303 temperature humidity sensor. Soil moisture sensor consists of a soil moisture sensor brick and a MCP 3008 ADC, the design of soil moisture data capture logic is based on the specification of MCP 3008 ADC which physically connects to the controller. The following two sections describes the methodology of these two logic blocks.

4.2.4 Temperature Data Capture Logic Block

The design employs the AM2303 sensor as the temperature sensor. The sensor outputs calibrated digital signal through SPI bus. During the transmission to sensor controller, the first 16 bits represent temperature data and following 16 bits represent humidity (not used in the project). The last 8 bits are check sum bits for verification. The following diagram demonstrates the transmission mechanism and timing for AM2303.

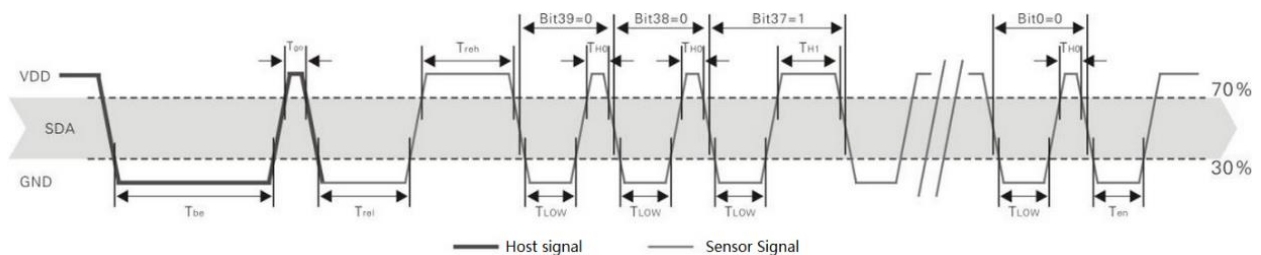


Figure 15: AM2303 Output Timing [8]

The sensor is activated by a start signal created by the sensor controller (host) followed by a pull up signal. Then controller releases the data line to indicate the host is ready to receive data. After the responding signal is arrived from the sensor, the data transmission starts.

Based on the specification on the sensor transmission, the design of the temperature data capture logic is shown in the FSM diagram below. The system remains idle unless a start bit is presented. After handshaking between the controller and sensor is completed, the transmission begins. The length of high-voltage-level signal in **Data Capture High** state will determine if the signal is logic high '1' or logic low '0'. Data capturing state will repeat forty times until all data are stored in the controller.

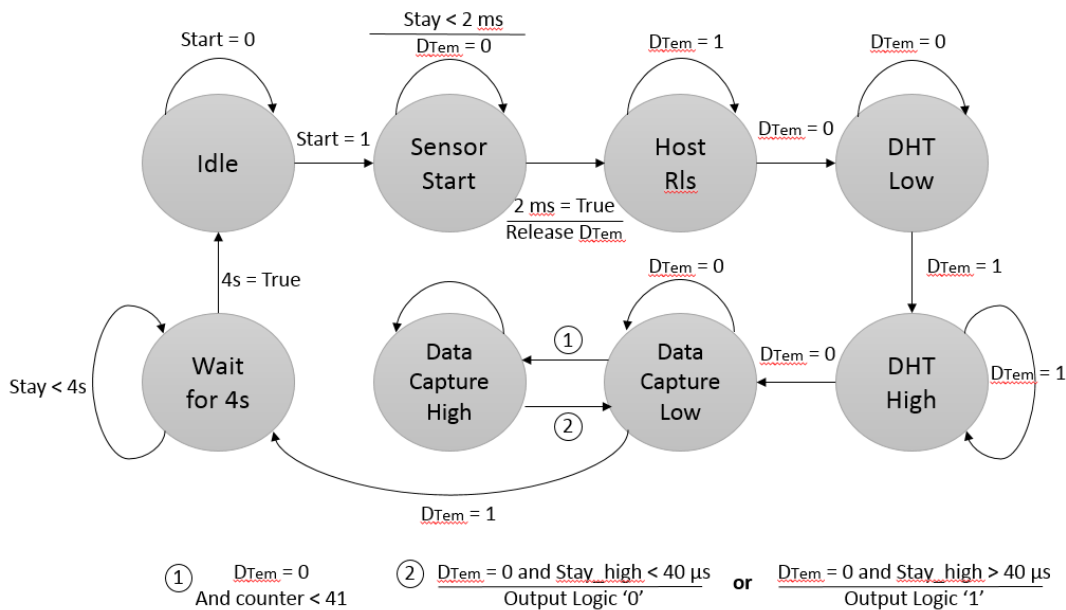


Figure 16: FSM of Temperature Data Capture

In the FSM, **DHT** represents sensor response state. **DTem** is bi-direction GPIO port connected to the SPI data line. To prevent the sensor from data collision, the sensor will idle four seconds after each full capture cycle completes.

4.2.5 Soil Moisture Data Capture Logic Block

Unlike the AM2303 temperature sensor, the output of soil moisture brick is an analog signal with a range between 0 ~ 5V. Therefore, MCP 3008 ADC is needed to convert the analog signal to digital signal so that FPGA can store and process. The data capture logic is based on the specification of MCP 3008 ADC which is shown in the following diagram.

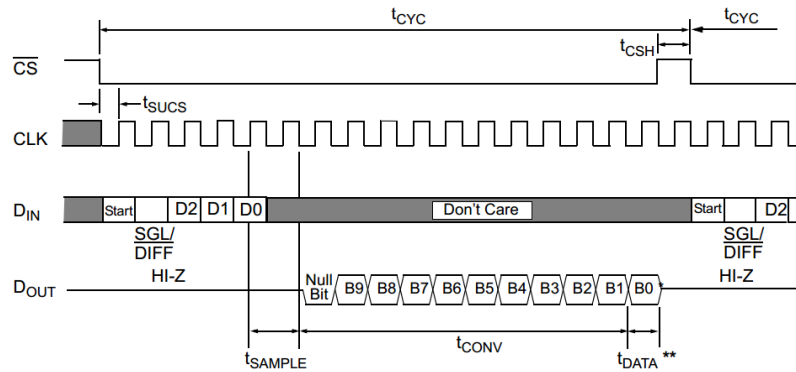


Figure 17: MCP3008 ADC Output Timing [9]

\overline{CS} /SHDN bit indicates the current status of the ADC, the ADC is idle when $\overline{CS} = 1$ and is on when $\overline{CS} = 0$. After ADC is activate, the controller needs to select the operating mode for the ADC by 5 sequential binary string through Din. For single analog signal conversion, the ADC only needs to operate under single-ended mode. Therefore, the binary string to Din should be 11100. After the mode is selected, the ADC will start to convert signal from soil moisture sensor and transmit the data to controller. Based on the characteristics above, the FSM of data capture logic for soil moisture sensor is shown as following diagram.

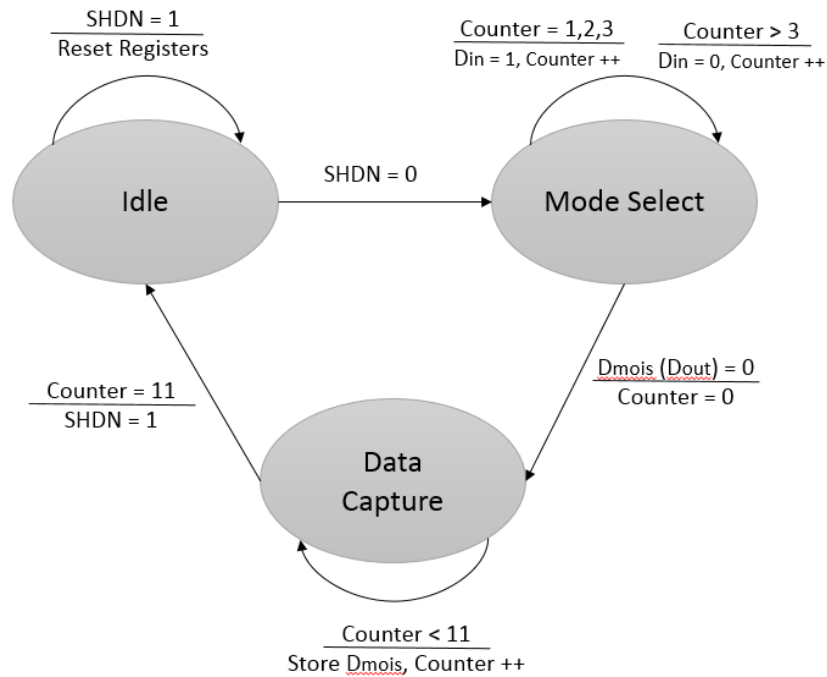


Figure 18: FSM of Soil Moisture Data Capture

4.2.6 Time Controller

Time controller is the last major logic block in the design, the main function of the component is to decrease the time duration which is set by either push button (traditional method) or PCIe data (external request). The valve power will be on as long as the remaining time is greater than zero. During the runtime, the length of time can be adjusted by either push button or external request from PCIe. In case of changing time through push button, the changed time will be also synchronized Intel processor by PCIe. The following block diagram summarized the configuration introduced above.

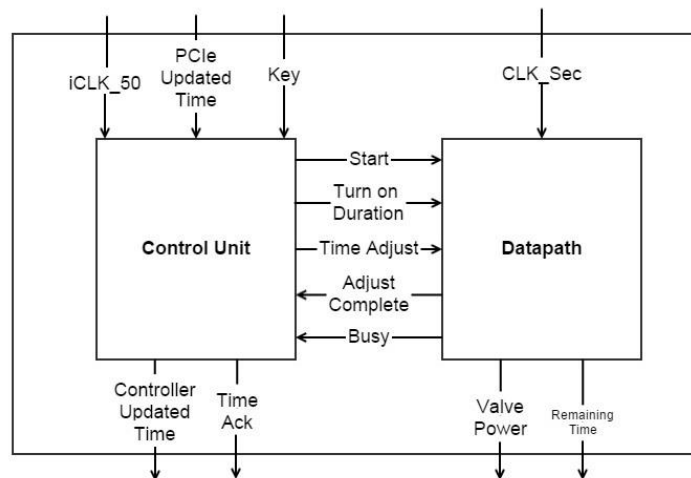


Figure 19: Block Diagram of Time Controller

The Time Controller consists of two logic blocks, control unit and datapath. Control unit is mainly focus on state change based on input signal which is implemented in 20 ns clock. In contrast, the datapath will be in charge of calculation of the time which utilize second clock. However, when the user adjust the time, the responding time is expected to be less than 10 ms. The delay time will be notice if entire calculations are assemble into Datapath. To improve the performance and simplify the algorism, time accumulation before the decrement starts will be implemented in control unit.

4.2.6.1 Control Unit (CU)

The following FSM summarizes the behaviour of the logic circuit for control unit.

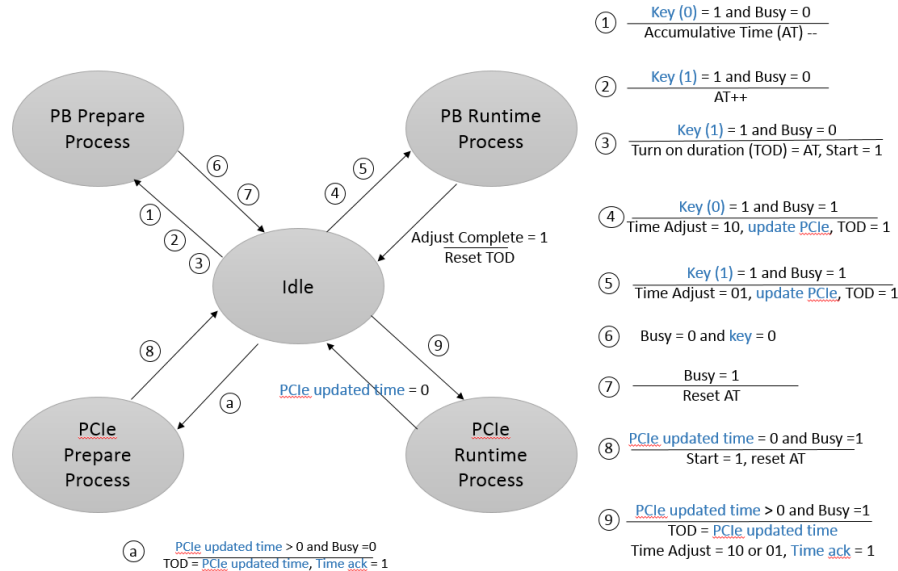


Figure 20: FSM of CU in Timer Controller

If the system is idle ($\text{Busy} = 0$), the status changes in push buttons or PCIe bus will result in **Prepare Process** state. The purpose of **Prepare Process** states are to ensure data and request from push button/PCIe will only be fetched once if the PB/PCIe status remains the same. For instance, if a user keep pressing the '+' button, the time will only increase by once instead of accumulating. To accumulate the time, users need to first release push button and press again to re-enter the prepare process state, the countdown will not start until a start button is pressed.

If the system is running ($\text{Busy} = 1$), pressing the push button or fetching PCIe data will lead the system to **Runtime Process**. **Runtime Process** is used to adjust the countdown time during the runtime, meanwhile sending out the acknowledge bit to prevent the controller from duplicating operations. The major difference between prepare process and runtime process is that time changes in prepare process will not activate the system until 'start' button is pressed, time changes in runtime process will take effect immediately.

Signals in blue color in the FSM indicates these signal are connected to external digital component. More details of input/output signals are introduced in block diagram. Other signals are internal signals which are connected to Datapath. A 'start' signal will inform the DP to start counting down. 'Time Adjust' tells DP what operations will be for the time adjustment. 'Adjust complete' indicates the adjust operation is done by DP and CU can proceed to the next state.

4.2.6.2 Datapath (DP)

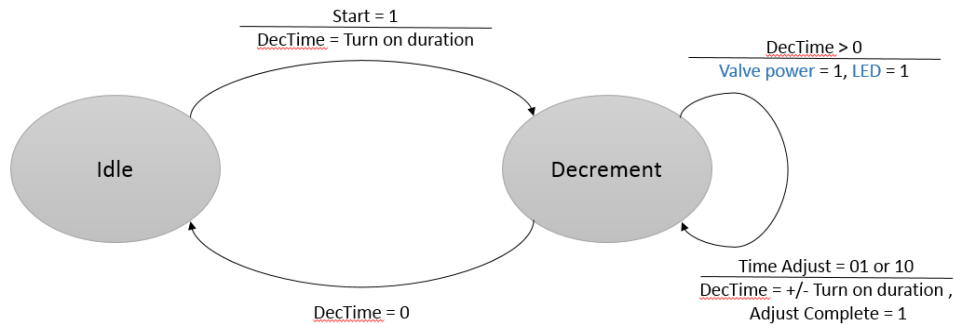


Figure 21: FSM of DP in Time Controller

Datapath of the time controller is to control the valve power based on the remaining time. The valve switch will be on as long as the remaining time is greater than zero. During the decrement process, the time can be adjusted according to instructions from CU. LED is applied to indicate the valve status.

4.2.7 LCD Display

Development of the HD44780 LCD module is based on the open source code from eewiki [10]. It contains the initialization process and display algorithm. In this section, only the display format will be introduced. The following table summarizes the content for each character space in two line mode.

T	I	M	E		L	E	F	T	:		X	X	:	X	X
T	E	M	:	X	X	.	X		M	O	I	:	X	X	%

Table 5: Information and Position on LCD display

For the proof of concept and prototype, due to a limit space for LCD display, only remaining time, Temperature and moisture will be displayed. For the final product, larger LCD will be used so that more information such as current zone, humidity and forecast can be provided to users.

4.3 Enclosure

Proof-of-concept model and first prototype mainly focus on functionalities such as valve automated control, push button control and LCD display which will be all implemented on developing board. For a final product version, PCB with an enclosure will be required

to prevent damaging circuitry and protect users from electric shock. The following list demonstrates the design for the controller enclosure. The drawing below shows the appearance of a final product.

- Controller will have thermal-insulated-plastic enclosure to protect the circuitry
- Enclosure will have 5 push buttons and LCD with proper position
- Device must have at least LCD with 20 characters width and 40 Characters length
- Buttons and LCD will be embedded on the enclosure
- Weight of the device must not exceed 5 kg
- Device will have Ethernet cable port, sensor data port and valve power control port

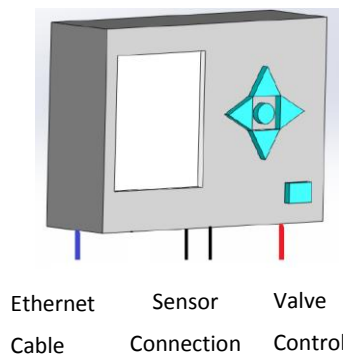
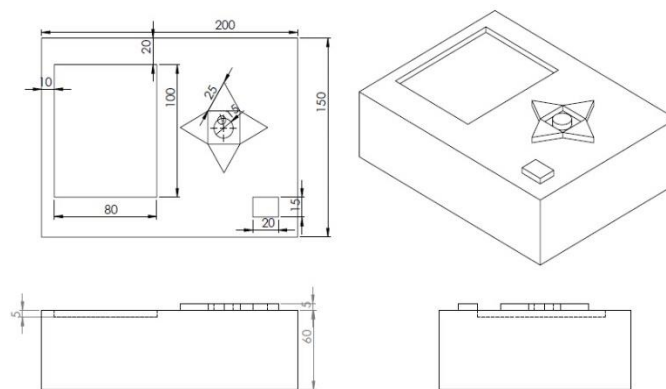


Figure 22: Enclosure for Irrigation Control Unit

The dimension of the device is shown in the following drawing in unit mm. Based on the size of the PCB, dimension of the enclosure and component position can vary ± 10 mm. In order to make the product competitive to current irrigation controller in the market, the size and weight of the enclosure should be under average.



Dimension

Figure 23: Dimension of the Device Enclosure

5. Sensor Circuitry

Environment circumstance is a crucial element that affects plant’s health condition. Temperature and soil moisture thus are chosen to be the input factor for AI analysis. As previous sections introduced, the chosen temperature sensor and soil moisture sensor for the design are AM2303 and four-line soil moisture brick connected to MCP 3008 ADC.

The reason of choosing AM2303 as the temperature sensor for the design is that the sensor output SPI data so that the output pins to the controller can be minimized. Besides the transmission advantage, AM2303 also provides functionality to measure humidity which will be useful in system upgrade in the future. In terms of physical characteristics, AM2303 and MCP 3008 both are functional with long transmission distance and low power consumption.

For the proof-of-concept and prototype, all electronic components will be soldered onto a perfboard. A final product will embed all components onto PCB for productivity and stability requirements. Following schematic shows the PCB design of the sensor circuit. Pin headers in the schematic are used to connect sensors and the controller.

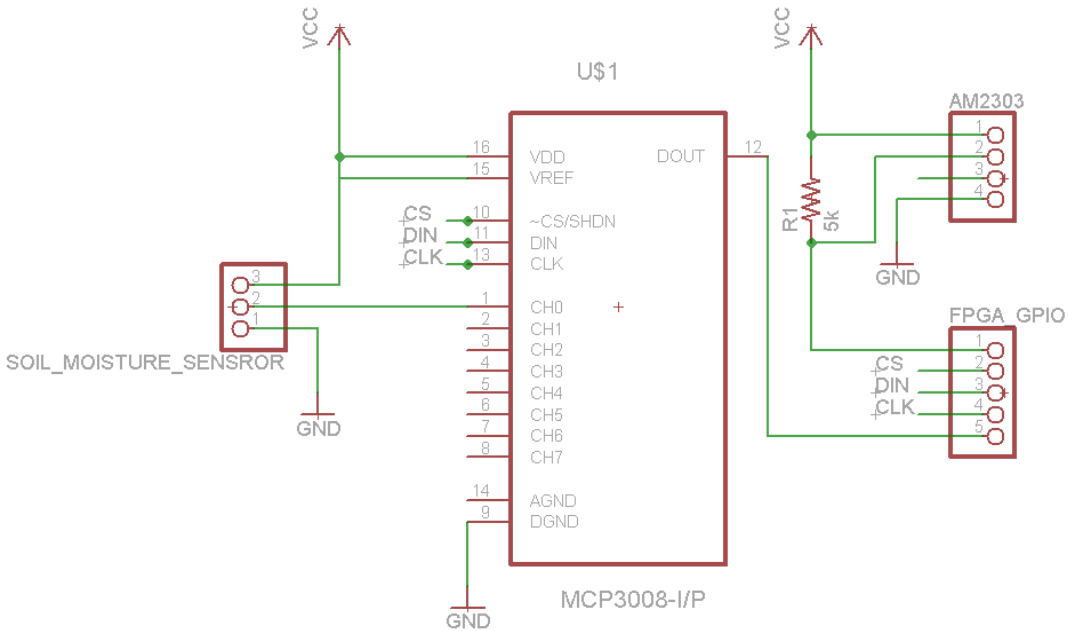


Figure 24: Schematic of Sensor Circuitry

The following schematic shows the board layout generated from the circuit schematic above. Only two layers are used due to the simplicity of the circuit structure.

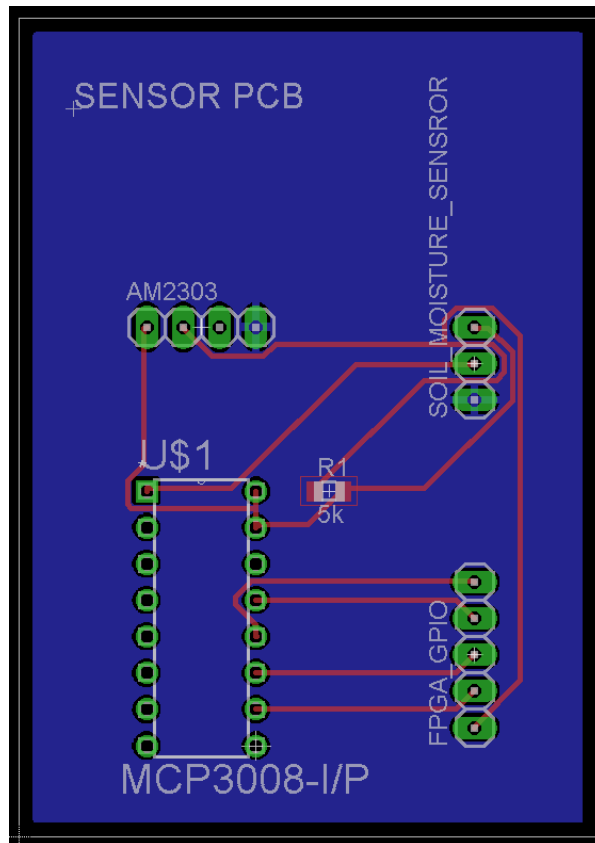


Figure 25: PCB Layout for Sensor Circuitry

6. Server (Cloud Service)

6.1 Database server

The database includes the following design table and relation schema. Database server can only accept connections from the Server application in a private network, and this is to reinforce the database security.

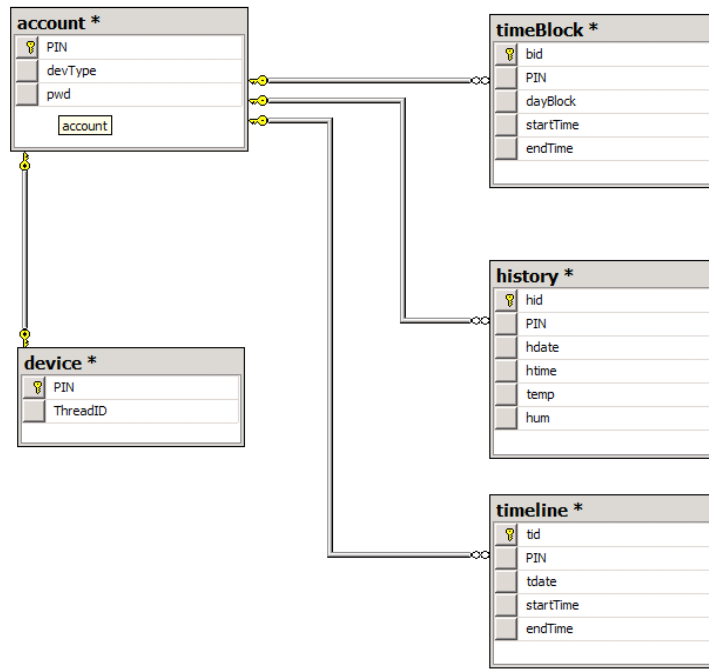


Figure 26: Database Design Table

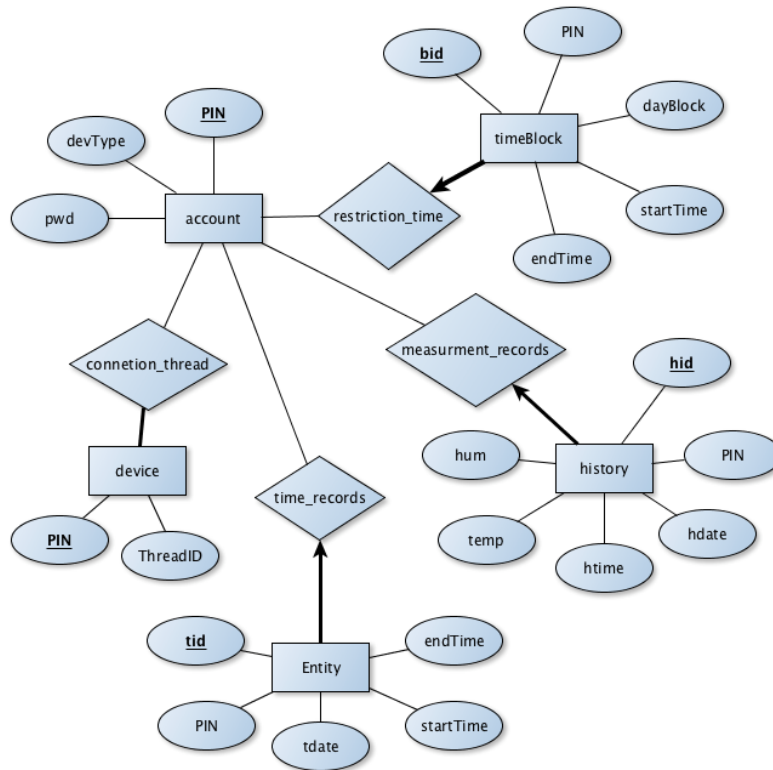


Figure 27: Database Relation Schema

6.2 Server Application

The server application is responsible for accepting all the requests from Main Logic Application on control units and mobile phones. The server application acts as a relay between control units and mobile phones, and also a bridge to database server. The server application contains a main thread and connection agent threads. The server application thread is an infinite loop for creating connection agent threads. The connection agents are responsible for accepting the incoming connections. Once the connection agent has accepted a connection from a client, the server application main thread creates a new connection agent thread.

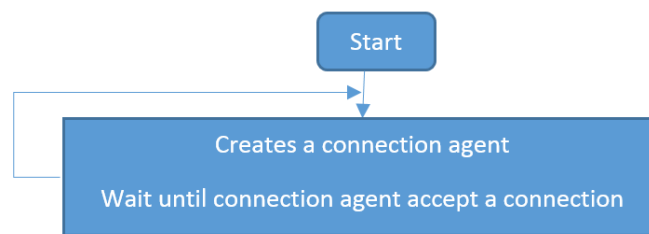


Figure 28: Flowchart of Server Application

The connection will be established one right after another, instead of waiting for the connected connection to finish the inquiries, and this meets the concurrency requirement.

6.3 Connection Agent Thread

Depending on the inquiries from clients, the connection agent thread generally handles connections in the following three cases:

1. read/write request to database
2. instruction tunnel to control units (FPGA)
3. instruction from mobile

Case 1: For read/write requests to database, the thread will create a query statement base on its inquiries, and it will send the query to the database server, and finally reply the result to the client.

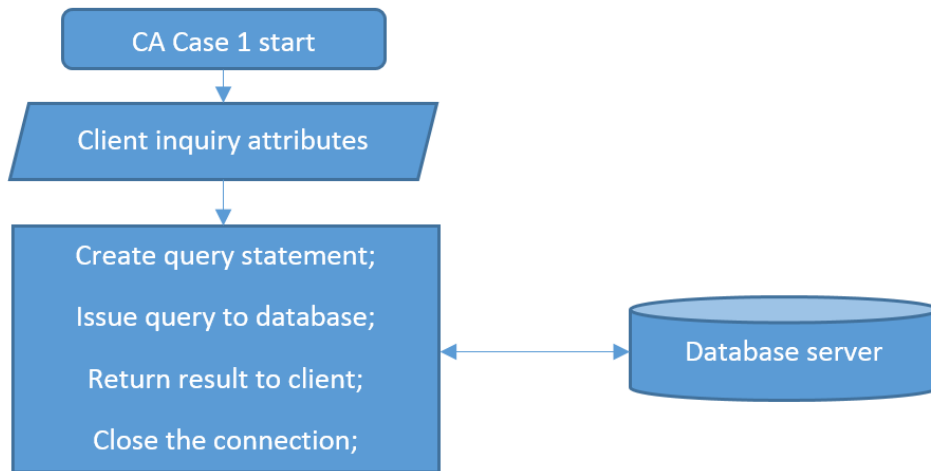


Figure 29: Flowchart of Connection Agent Thread (Case 1)

Case 2: Instruction tunnel to control units (FPGA) is a connection for server to control the irrigation system remotely. There will be only exact one connection for each control unit, so it will do a duplication check to close the old connection. After the thread has accepted a connection from the client, it saves the thread ID to database, and puts the connection into a timed wait state and the timer is set to 30 minutes. Under standby, the thread goes to sleep and wakes up in any of the following two conditions: interruption from another thread, or expiration of the timer. If the timer expires, it sends an ACK to client. If it is interrupted, it sends instructions to its client according to instruction buffer created by the Case 3 by other connections. After the either of the takes is finished, it resets the timer and back to the timed wait state. If the thread fails to connect to its client at any time, it will clear its ID record in database, and terminate the thread.

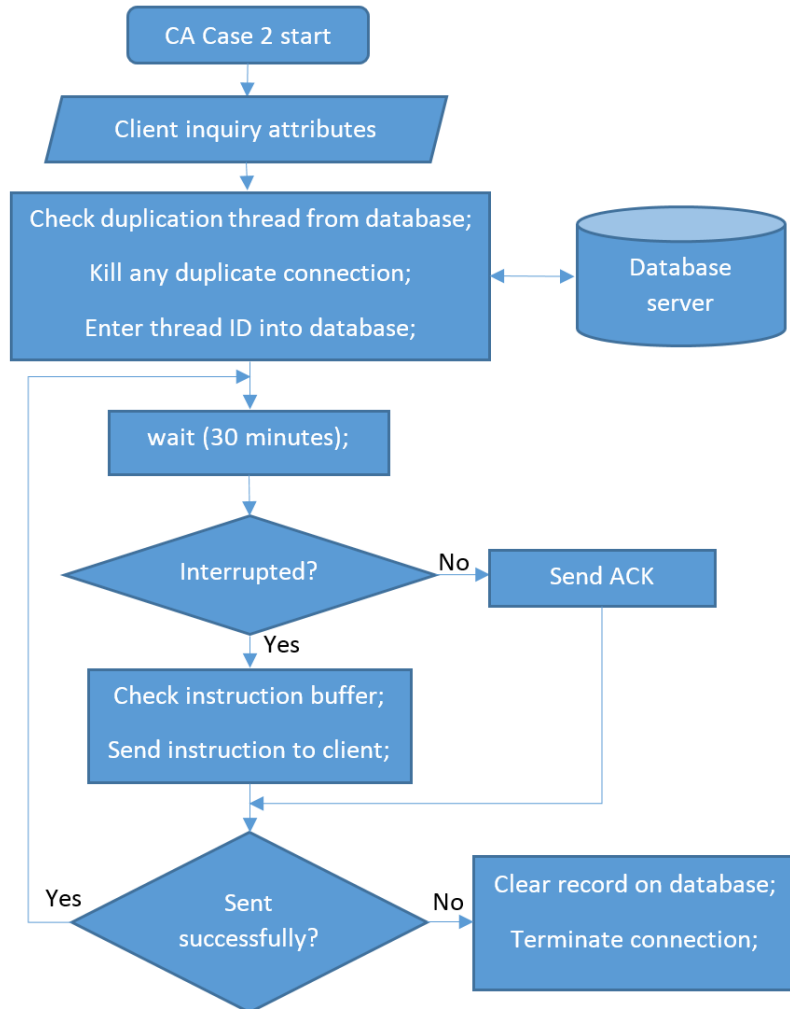


Figure 30: Flowchart of Connection Agent Thread (Case 2)

Case 3: Instruction from mobile is a connection type for mobile to communicate to server and let the server pass the instruction to Control unit (FPGA) remotely. It first checks if the FPGA is has a Case 2 connection from database. If it is not available, mobile phone gets the offline notification. If it is online, it saves the instruction to the instruction buffer and sends an interruptions to the Case 2 connection.

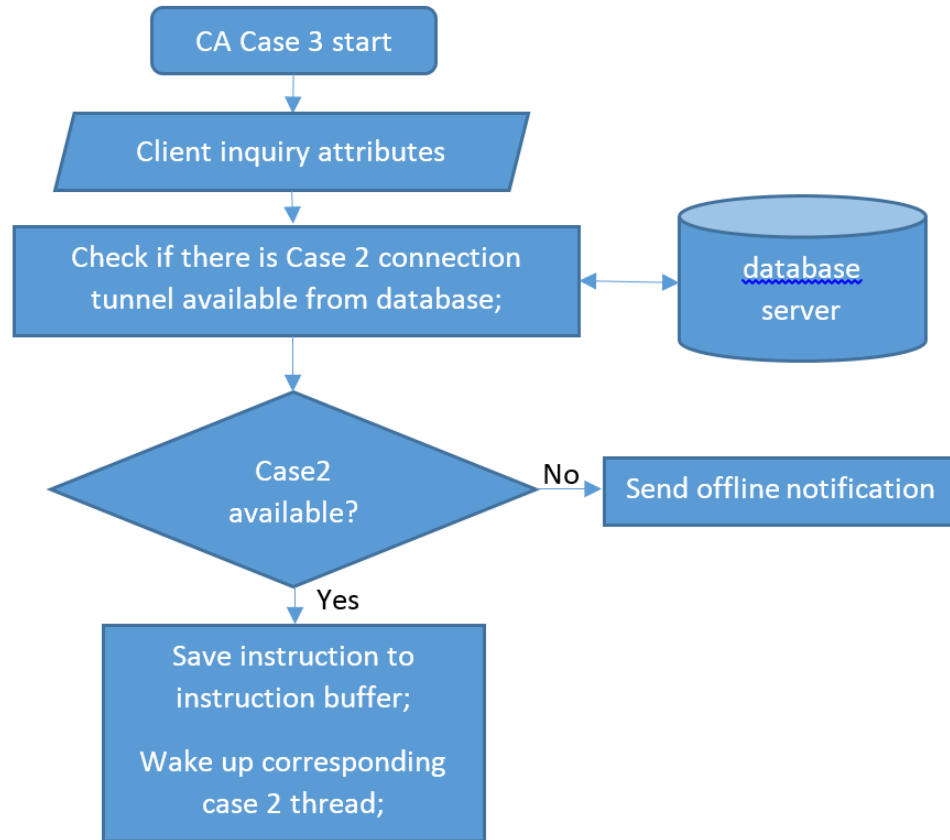


Figure 31: Flowchart of Connection Agent Thread (Case 3)

6.4 Android Mobile Connection API

Android Mobile Connection API is a Java class that is designed for deployment to any Java platform. It is designed specifically for communicating to the connection agents in the server application. It serves as the client of case 1 and case 3 connections in Connection Agent.

For communications using case 1, it is generally used for pulling the device status and history data.

For communications using case 3, it is generally used for users to control the irrigation system remotely.

7. Mobile Application

7.1 Mobile App Design

The mobile app functions as a user interface therefore the layout must be user friendly. It must be able to clearly display data received from the server and allow users to make some modifications and send it back to the server. The app can be started by clicking the icon of the app and can be terminated any time with the hard key on the smartphone. The app contains 5 pages: Login page, Signup page, Operation page, History page and Settings page. There is an action bar sitting on top of all pages with a menu drop down list to go to the operation page, the history page or the settings page and a logout button to go back to the login page. The company logo will be used as the app icon. The float diagram for the app is shown below:

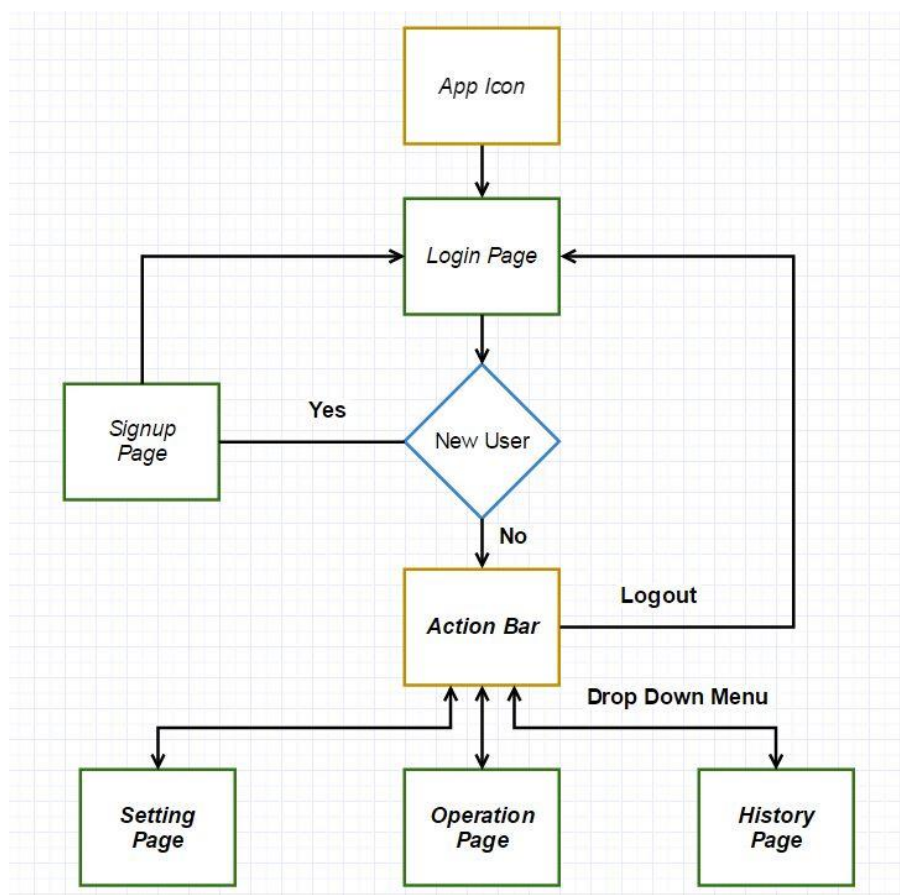


Figure 32: The Overall App Structure

7.2 Login Page and Signup Page

The layout of the login page and the signup page is shown below:



Figure 33: Login Page (left) and Signup Page (right)

To satisfy [R 125-PII], for the login page, user name and password are required to identify users. After filling out the two fields, by pressing “Login” button, the app will send information to the server to check if the users name and password are correct. If they are correct, it will go into the operation page with the action bar on top. If they are not correct, it will stay on the login page unless user inserted the correct user name and password. If the user doesn’t have an account, then go to signup page by clicking “Signup”.

For the signup page, user name, password, confirm password, address and a checkbox are required for basic operations. If the address is residential address, then the checkbox should be checked. If the checkbox is not checked, it means non-residential by default. The pressed “Cancel” button will discard all the information that the user already filled in and go back to the login page. When the “Submit” button is clicked, all the information that the user filled in will be sent to the server. It will go to the login page if the server sent information to show that the new account has been created otherwise it will stay on the signup page.

After signed in, the app will go to the operation page by default and users are able to select the history page and the settings page by clicking the “Menu” button on the action bar.

7.3 Operation Page

The layout of the operation page is shown below:

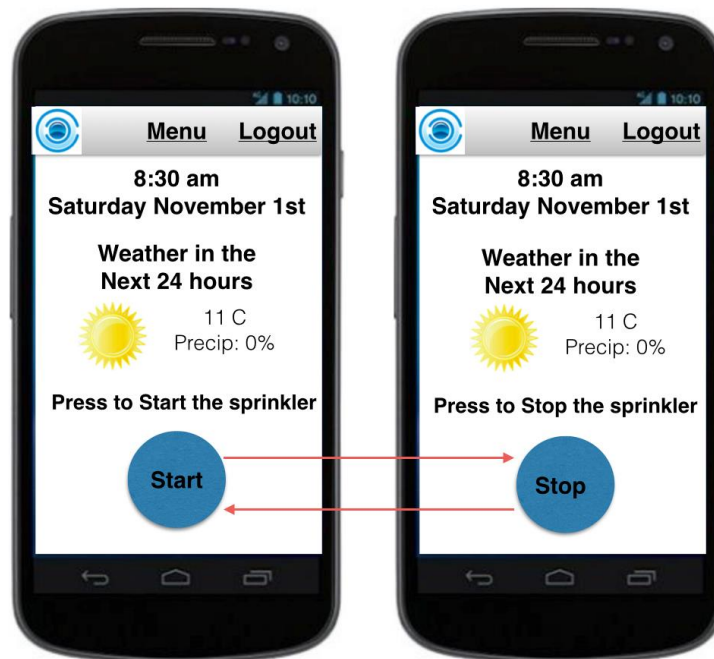


Figure 34: The Operation Page

The layout of the operation page contains only one button to start/stop the sprinkler with a note information above the button. To satisfy requirement [R 126-PIII], when the start button is clicked, signals will be sent to the server and check if it is permitted for sprinkling now then the sprinkler will start operation immediately unless the default period of sprinkling time has passed. To satisfy requirement [R 119-P111] and [R 129-PII], the date and time and the weather forecast will be also on the top of the button. The weather forecast will be received from the server.

7.4 Setting Page

The layout of the settings page is shown below:



Figure 35: The Setting Page

The settings page allows user to make modifications to the settings so that the sprinkling schedule will be modified automatically based on the settings. To satisfy requirement [R 118-PI], the “Address” and the “Residential Address” decided which local sprinkling policy to follow. To satisfy [R 124-PIII], since different kinds of plants has different requirement on soil humidity, the “Plant Type” decided which kind of plant will be used as watering standard, such as “grass”, “butterfly orchid” and “jasmine”. To satisfy [R 127-PII] and [R 121-PIII], in case of some specific time that it is not convenient for sprinkling, the settings of “No Sprinkling From To” will help users to avoid sprinkling at that period of time. The once the no sprinkling time has been set, it will apply to every day. When the “Done” button is clicked, the modified settings will be sent to the server for updating.

7.5 Historical Page

The layout of the settings page is shown below:

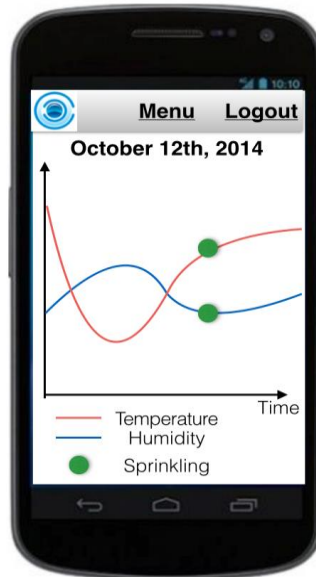


Figure 36: The Historical Page

To satisfy requirement **[R 130-PIII]**, when specific day in the history is selected, the history page shows the daily temperature and humidity change respect to time. The sprinkling recorded will also be on the line chart. By clicking on the date, a calendar will pop up and allows user to change another day to display the recorded information received from the server. An empty chart will be shown if there is no record on the selected time.

8. Test Plan

The test plan will take place after prototype has been integrated and should fulfill the purpose of examining the function of each individual component as well as the corporation between each components. The test process will follow logic steps from simpler examine to more complex examine with the purpose to test whether our product have met the expectations from our function specification.

8.1 Hardware Test Plan

8.1.1 Soil Moisture Sensor Test

The sensor should provide consistent reading according to the water content of the soil and the reading should correctly appears on the mobile application.

Test process: place soil moisture sensors in soil with different water content. Then check the reading on mobile application

8.1.2 Temperature Sensor Test

The temperature sensor should provide responsive and correct reading on the mobile application

Test process: Manually create a temperature change to test the temperatures sensor. Then check the reading on mobile application

8.1.3 Valve Test

The valve controller should open or close the valve according to the command sent from operation system within half second delay.

Test process: Send virtual command to valve controller and observer the behavior of the water valve.

8.1.4 Control Box Test

The LCD display should display correct and accurate reading gathered from sensors. The four push buttons should work as noted.

Test process: send virtual read of soil moisture sensor and temperature sensor and check the reading on the LCD display. Then operate the control box with the four push button

to test their functionality.

8.2 Server Test Plan

The server is the transit point for mobile device and valve controller. It should be able to receive, send and store data for each pair of mobile device and its matching valve controller.

8.2.1 Data Transmission

The server should be able to receive, send and store all data without any loss.

Test process: Send virtual data to server from operation system, then send the stored virtual data to mobile device. Then send and receive at the same time. Compare the received data with the sent data at each stage and check the stored virtual data.

8.2.2 Privacy

The data transmission is setup for only matched mobile device and valve controller. The command send by mobile device is only send to corresponding valve controller. The gathered data send by valve controller will also send only to its matched mobile device.

Test process: Send command from mobile device and look for the command at its corresponding valve controller. Then send virtual data using valve controller and look for the data at corresponding mobile device.

8.3 Mobile App Test Plan

The application should be able to install on the simulation device. After clicking on the application icon, the application should direct to the login page without crashing. In general, the application should run without frequent crashing. All buttons should work as how they were described. The general requirements will be tested in the process of tests for detailed functions.

8.3.1 Login Test Page

At the login page, only matched username and password would be approved, remain at login page otherwise.

Test process: Login in with matched username and password, as well as unmatched pair to test the login function of the application.

Signup button should direct user to signup page. Return to login page if signup is successful, remain at signup page otherwise.

Test process: Tap signup button. After being directed, Signup at login page with virtual information. User will be denied when using existing username, otherwise will return to login page if successful.

8.3.2 Setting Page Test

The setting page should be accessible without crashing. All information user filled in should be correctly updated to the server.

Test process: Tap setting button to access setting page. Fill in virtual setting to examine function the setting page. Fill virtual information should be found at the server.

8.3.3 Operation Page Test

The operation page should be accessible without crashing. Necessary information are correctly displayed. Hardware correctly respond to operation command with delay no longer than 1 second.

Test process: Tap operation button to access operation page. Observe information displayed at operation page. After taping the start/stop button, the valve correctly respond to the command.

8.3.4 History Page Test

The history page should correctly display operation history sent from server.

Test process: Store virtual operation history in server. Tap history button, to access history page then check if the operation history data is correctly presented by the mobile application.

8.4 Operation System Test Plan

The operation system should be able to install in available space provided by hardware. It should be capable of analyze all gather data without crashing. The following tests will be performed to ensure its functionality.

8.4.1 Data Analyze

Hardware data should be correctly analyzed and saved locally when internet connection is unavailable or to the server when internet connection is available.

Test process: Sent virtual data to operation system. Check the processed data in internet connected scenario and no internet connection scenario

8.4.2 Watering Schedule

The watering timetable should be scheduled according to local policy, environmental condition, and weather forecast. The result schedule should be send to the server and correctly shows on the mobile application.

Test process: Sent virtual data to operation system. Check the processed watering schedule in internet connected scenario and no internet connection scenario

8.4.3 Server Connection

All analyzed data and processed watering schedule should be send to server when internet connection is available.

Test process: Check the server data base after tested data analyze and watering schedule in internet connected scenario.

8.5 Integrated Test Plan

Integrated test plan is used to test the coordination of all part after they are assembled. It would examine the functionality of the final prototype.

8.5.1 Soil Moisture Factor Test

The valve controller should irrigate according to the soil condition.

Test process: place the soil moisture sensor in dry soil condition and wet soil condition and observer the products behavior. In dry soil scenario a watering request should be updated for the next available schedule. In wet soil scenario, the valve controller should stop watering or cancel the next watering request if the watering has not started yet.

8.5.2 Weather Factor Test

The valve controller should operate according to the forecasted weather.

Test process: sent virtual weather forecast to operation system and check the water schedule. The raining day water scheme should replace the regular water scheme.

8.5.3 Self-operation Test

The valve controller would work following the preset watering schedule when no outside interference is applied. The valve should open or close according to the schedule within one second delay.

Test process: Create a virtual watering schedule with rapid watering frequency at the operation system. Check the response of the watering valve.

8.5.4 User Command Test

The valve controller should irrigate following the command send from mobile application within one second delay without considering the delay caused by wireless transmission.

Test process: use the mobile application to send virtual command and observer the behavior of the valve.

8.5.5 Overall Stability

The final prototype should work consistently without bug or crash.

Test process: monitor and examine the behavior of the prototype during the test phase.

9. Conclusion

This documentation has briefly explained the possible design solutions to meet the functional specification of the C-Sprinkler. This smart irrigation system consists of three major components: hardware, software and mobile app. Details of each component's design specification are presented in the previous sections. The required functional specifications are listed in the corresponding section. Also the test plans are provided to examine the functionality of the first prototype for both individual components and integrated system. The detail test sheets can be found in the appendix section. Overall, this design specification document clearly states the functionality for the first prototype and final production of the C-Sprinkler.

10. Reference

- [1] Functional Specification for the Smart Irrigation System (2015) , Team Chase Technologies
- [2] Internet of Things (IOT) (2014) TechTarget. Retrieved from <http://whatis.techtarget.com/definition/Internet-of-Things>[Accessed: Nov 6, 2014]
- [3] DE2i-150 FPGA Development Kit. (n.d.). Retrieved from <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=11&No=529> [Accessed: Nov 6, 2014]
- [4] PreeyaphornKosa (2011). The Effect of Temperature on Actual Evapotranspiration based on Landsat 5 TM Satellite Imagery, Evapotranspiration, Prof. LeszekLabeledzki (Ed.), ISBN: 978-953-307-251-7, InTech, Retrieved from <http://cdn.intechopen.com/pdfs-wm/14187.pdf> [Accessed: Nov 6, 2014]
- [5] Water Holding Capacity and Plant Available Water. (n.d.)Retrieved from http://www.utcrops.com/irrigation/irr_mgmt_waterholdingsoils.html [Accessed: Nov 6, 2014]
- [6] R. Troy Peters, PE, Ph.D., Extension Specialist; Kefyalew Desta, Ph.D., Soil and Crop Management; and Leigh Nelson, PE, WSU Prosser Irrigated Agriculture Research & Extension Center. (2013). Practical Use of Soil Moisture Sensors and Their Data for Irrigation Scheduling. Retrieved from <http://cru.cahe.wsu.edu/CEPublications/FS083E/FS083E.pdf> [Accessed: Nov 6, 2014]
- [7] DE2i-150 FPGA Development Kit. How to distinguish Version A and Version B board? (n.d.). Retrieved from <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=11&No=529&PartNo=5> [Accessed: Nov 6, 2014]
- [8] Aosong electronics Co. (n.d.). Digital-output relative humidity & temperature sensor/module Retrieved from <http://www.adafruit.com/datasheets/DHT22.pdf> [Accessed: Nov 6, 2014]
- [9] 2008 Microchip Technology inc. (n.d.). 2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface. Retrieved from <https://www.adafruit.com/datasheets/MCP3008.pdf> [Accessed: Nov 6, 2014]
- [10] Larson, S. (2013, Aug). Character LCD Module Controller (VHDL). Retrieved from <http://www.eewiki.net/pages/viewpage.action?pageId=4096079> [Accessed: Nov 6, 2014]

Appendix – Test Plan Sheets

Hardware Test Sheet

Tests Subject	Result(P/F)	Comments
Soil Moisture Sensor		
Soil moisture sensor provides correct and consistent reading		
Temperature Sensor		
Temperature sensor provides correct and consistent reading		
Valve Controller		
Valve controller correctly operate according to virtual commands		
Valve controller operates within delay tolerance		
Control Box		
Correct data on LCD display		
Proper function of push button right		
Proper function of push button up		
Proper function of push button left		
Proper function of push button down		

Server Test Sheet

Tests Subject	Result(P/F)	Comments
Data Transmission		
Successfully received data		
Data stored without loss		
Successfully send data		
Simultaneous test		
Privacy		
Valve controller received command from corresponding mobile device		
Mobile device received data from corresponding valve controller		

Mobile App Test Sheet

Tests Subject	Result(P/F)	Comments
Opening Application		
Successfully opened the application without crashing		
Login Page		
Wrong match of username and password are denied		
Correct match of username and password are approved		
Successfully signup using signup page		
Setting Page		
Page access successful		
Information entered without bug		
Information correctly stored locally when internet is unavailable		
Information correctly stored in server when internet is available		
Operation Page		
Page access successful		
Correct information display		
Operation commend executed within delay tolerance		
History Page		
Page access successful		
Correct operation history data display		
General Requirements		
Run without frequent crash		
All buttons work as expected		

Operation System Test Sheet

Tests Subject	Result(P/F)	Comments
System Operation		
System could operate without bug or frequency crash		
Data Analyze		
Correct data analyze according preset formulas		
Successfully saved locally when internet connection is not available		
Successfully saved in server when internet connection is available		
Watering Schedule		
Correctly create watering schedule according to requirements		
Successfully saved locally when internet connection is not available		
Successfully saved in server when internet connection is available		
Server Connection		
All data sent are saved at the server data base without any lose		

Integrated Test Sheet

Tests Subject	Result(P/F)	Comments
Soil Moisture Factor		
Correct schedule updated on mobile application		
Weather Factor		
Correct schedule updated on mobile application		
User Command Test		
The valve operates according to users command		
The valve respond within delay tolerance		
Self-operation Test		
Valve turn on as schedule watering period within one second delay		
Valve turn off after watering is over within one second delay		
Overall Stability		
Remain stable during entire test phrase		