Smart Fruit Drying System

Project submitted in fulfilment of the requirements for the

Software Engineering Project

Diploma: Engineering: Electrical in the department of Computer Systems

Faculty of Engineering and Technology

B SELLO

218507615

Dip: Engineering: Computer systems

A NEKHAVHAMBE

216099331

Dip: Engineering: Computer systems

TM MABOKELA

218485883

Dip: Engineering: Computer systems

N NKAMBULE

218515383

Dip: Engineering: Computer systems

Vaal University of Technology

Vanderbijlpark

South Africa

Date: 23/11/2022

Declaration

I **Boitumelo Sello** declare that this report is my own, unaided work. It is being submitted for Software Engineering 1 to the Department of Electrical Engineering at the Vaal University of Technology, Vanderbijlpark. It has not been submitted before for any subject or evaluation to any educational institution.

Bsello

Boitumelo Sello
23/11/2022

I **Aubrey Nekhavhambe** declare that this report is my own, unaided work. It is being submitted for Software Engineering to the Department of Electrical Engineering at the Vaal University of Technology, Vanderbijlpark. It has not been submitted before for any subject or evaluation to any educational institution.

A Nekhavhambe
Aubrey Nekhavhambe
23/11/2022

I **Tabudi Mabokela** declare that this report is my own, unaided work. It is being submitted for Software Engineering to the Department of Electrical Engineering at the Vaal University of Technology, Vanderbijlpark. It has not been submitted before for any subject or evaluation to any educational institution.

T Mabokela
Tabudi Mabokela
23/11/2022

I **Nokuthula Nkambule** declare that this report is my own, unaided work. It is being submitted for Software Engineering to the Department of Electrical Engineering at the Vaal University of Technology, Vanderbijlpark. It has not been submitted before for any subject or evaluation to any educational institution.

N Nkambule
Nokuthula Nkambule
23/11/2022

Table of Contents

List of figures

List of tables

If you right click on the tables and then on update field it should automatically update the tables.

List of abbreviations

Do not remove the section break below

Chapter 1

1.1 Introduction

Smart fruit drying system for fruit drying factory

1.1.1 Background

Fruits are one of the most nutritious and very important food sources in the human diet. Besides reducing the risk of stroke and heart diseases, a diet rich in fruits can reduce the risk of eye and digestive problems. Because of their low saturation of fat, sugar and salt, they help lower cholesterol and blood pressure levels (Popeck, 2020). However fruits are highly perishable because of their high water content, abundant in season and very limited out of season. Due to the fast growing global population food security and safety has become a major concern. Key fact made by the world health organization suggest that an estimate 600 million people fall ill after consuming contaminated food and 420 000 of those die every year (WHO, 2022). Food preservation methods play a very huge role in providing food security and safety.

An ancient and very popular preservation method used is drying. Dried fruit is defined as fruit that has had almost all water content removed through different drying methods (Bjarnadottir, 2017). Apart from being less hygienic the conventional methods of fruit drying such as solar drying are not able to successfully dry all different types of fruits and are very slow. Moreover it is often difficult to maintain and control the drying parameters in natural environments.

For successful fruit drying, most fruits require a minimum temperature of 30 degrees Celsius to a maximum of 60 degrees Celsius. A relative humidity of 60% is also an important factor. Another non-quantitative parameter that is essential for drying is good air circulation (Johncheff, 2022). A smart fruit drying should be able to effectively deploy these conditions hence maximize the output, with little or no negative health impacts and in the shortest period of time as possible.

Figure 1 fruit & vegetable dehydrator (Anon., n.d.)

1.2 **Problem Statement**

Varying Temperature and humidity because inconsistent drying of fruits and this negatively affects not only the yield of dried fruits but also the quality of the dried fruits.

1.3 **Sub-problems**

View drying status

This sub-problem allows the user to see the current drying status.

Dry fruits

This sub-problem performs the main work in the system of drying fruits. It stops when the system is stopped and starts when the system is started.

Start/stop/pause drying

This sub-problem allows the user to start, pause or stop the drying cycle.

View current temperature & humidity

This sub-problem allow the user to view the current temperature and humidity inside the smart fruit drying system.

Maintain drying parameters

This sub-problem allows the user to maintain drying parameters. This includes factors air circulation and the time for drying. The user should be able to set and adjust these elements.

Maintain drying temperature & humidity

This sub-problem allows the user to maintain the drying temperature & humidity, the system should include an interface where the user will be able to set and adjust the desired drying humidity and temperature.

1.4 Delimitations

The development and design of humidity and temperature probes are beyond the scope of this project.

1.5 Definitions of terms

A list of terms and associated definitions should be provided.

1.6 Importance of the project

Since the system being designed is for a fruit drying factory, this system aims to reduce operational costs while giving the highest yield possible.

The prevailing technological advancements show that with proper management of drying parameters such as humidity and temperature the drying process can be sped up. Popular drying systems are also small-scale dryers as air circulation can easily be controlled however with the need of high yield a large scale dehydrating system will help preserve fruits and in turn combat the global food shortage.

1.7 Overview of the project and system to be developed

The aim of this project is to develop a smart fruit drying system for a fruit drying factory which will allow the drying of different kinds of fruits in large proportions at desired temperature and humidity levels. The designed system will be based on an existing fruit dehydrators

Chapter 2 will present the requirements models for the system.

Chapter 3 will present the analysis models for the system.

Chapter 4 will present the design model for the system

Chapter 5 will present the implementation model for the system.

Chapter 6 will present the conclusion and recommendation.

1.8 Summary

In this chapter, the importance of the sub-problems of the smart fruit drying system were introduced. The use case diagram introduces the use case and sub-problems which will be covered in order to develop the smart fruit drying system. The next chapter provides the requirements model for the smart hydroponic system. The requirements model covers the outline & detail use case specification and activity diagram for one of the use cases.

Chapter 2 Requirements model

2.1 Introduction

This chapter will expand on the start/stop/pause drying sub-problem for the smart hydroponic system. The requirements, object oriented analysis and design models will be provided.

2.2 Literature review

The health benefits of fruits to the human system are unquestionable. Climate change has negatively affected the overall yield of fruits across the globe not only in quality and quantity, but also in lifespan as temperatures are no longer consistent. The major problem faced by the producers of fruits is the ability to prevent these products from spoilage.

The drying of fruits has proven to be very vital in combating this problem. Recent technological advancements have simplified sensing and controlling technologies, making it easier to design and implement smart drying for multiple products (ya su, 2014). These advancements have proven to be cost-effective in their ability to monitor and detect various drying parameters in order to produce food of good health standards. The preservation of fruits has been in existence for years and has proven to be successful in preserving the nutritional value of fruits while also improving their life span, however conventional are deemed slow

Needless to say the aim of drying is to prolong shelf-life however other factors such as flavor aroma and texture are as equally important. Drying technologies have not only made it possible to prolong shelf life but also the quality, texture, flavor appearance and the chemical properties of dried products.

Drying technologies are normally judged upon their ability to reduce the amount of water in fruits at the least amount of energy consumption whilst preserving the good sensorial value of the fruits. Their intelligence compared to conventional drying methods is very conspicuous as they have a significantly high efficiency.

They also provide users with the ability to fully manage and control important factors that affect drying. Among these factors are time which under traditional methods depended on natural sources such as sunlight. Two other related factors include Temperature and Humidity.

There are also many other complex factors such as fruit mass or size which are not easy to manage or control.

2.3 **Requirements model**

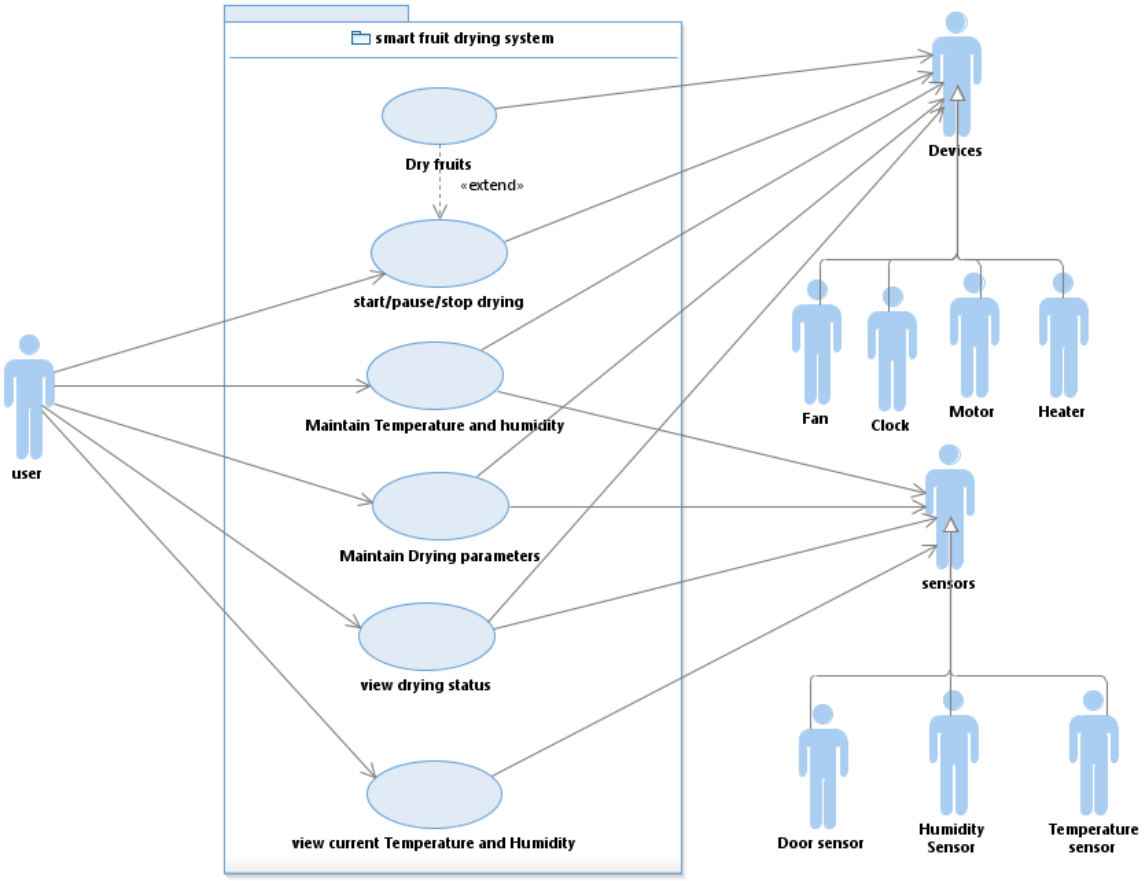

Figure 2 smart fruit drying use case diagram

2.4 **Activity diagrams**

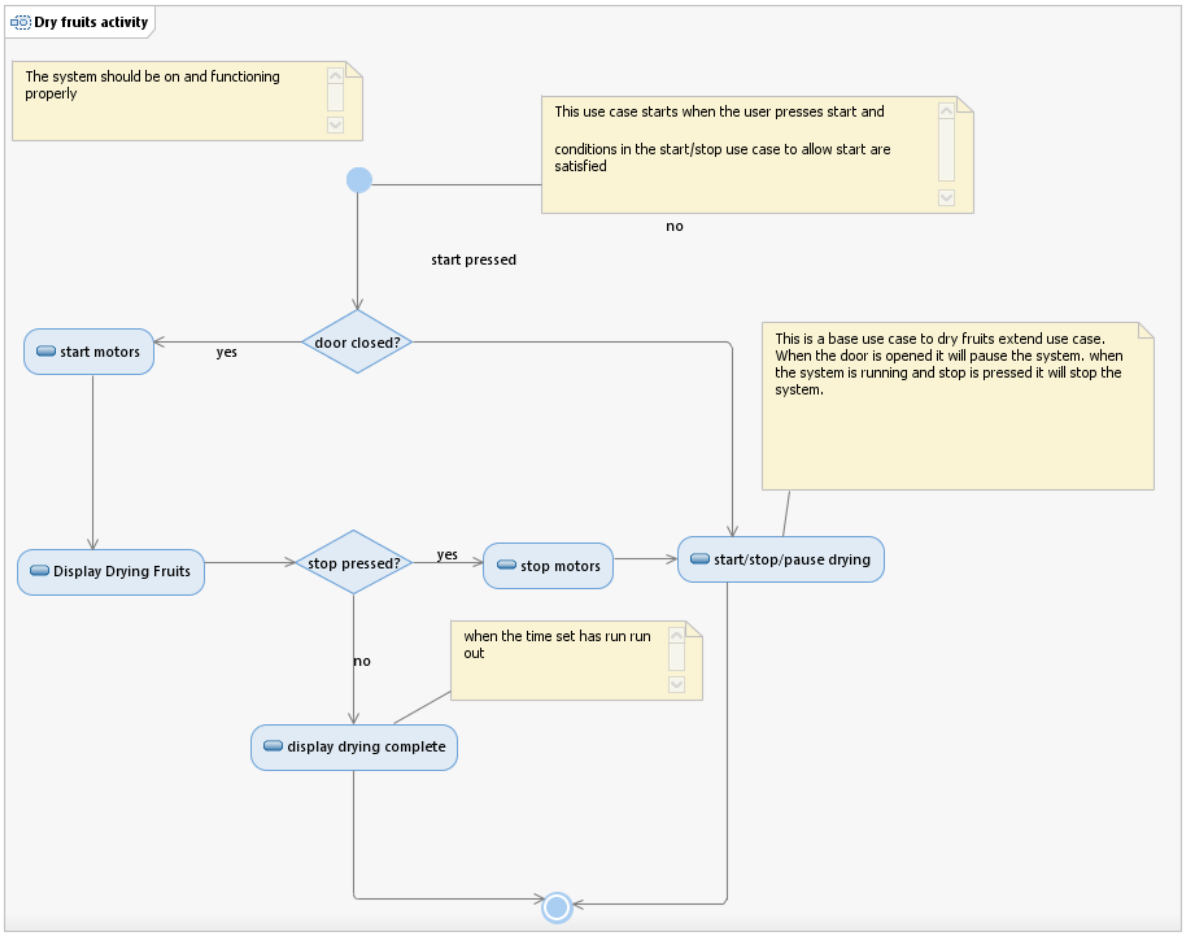

Figure 3 dry fruits activity diagram

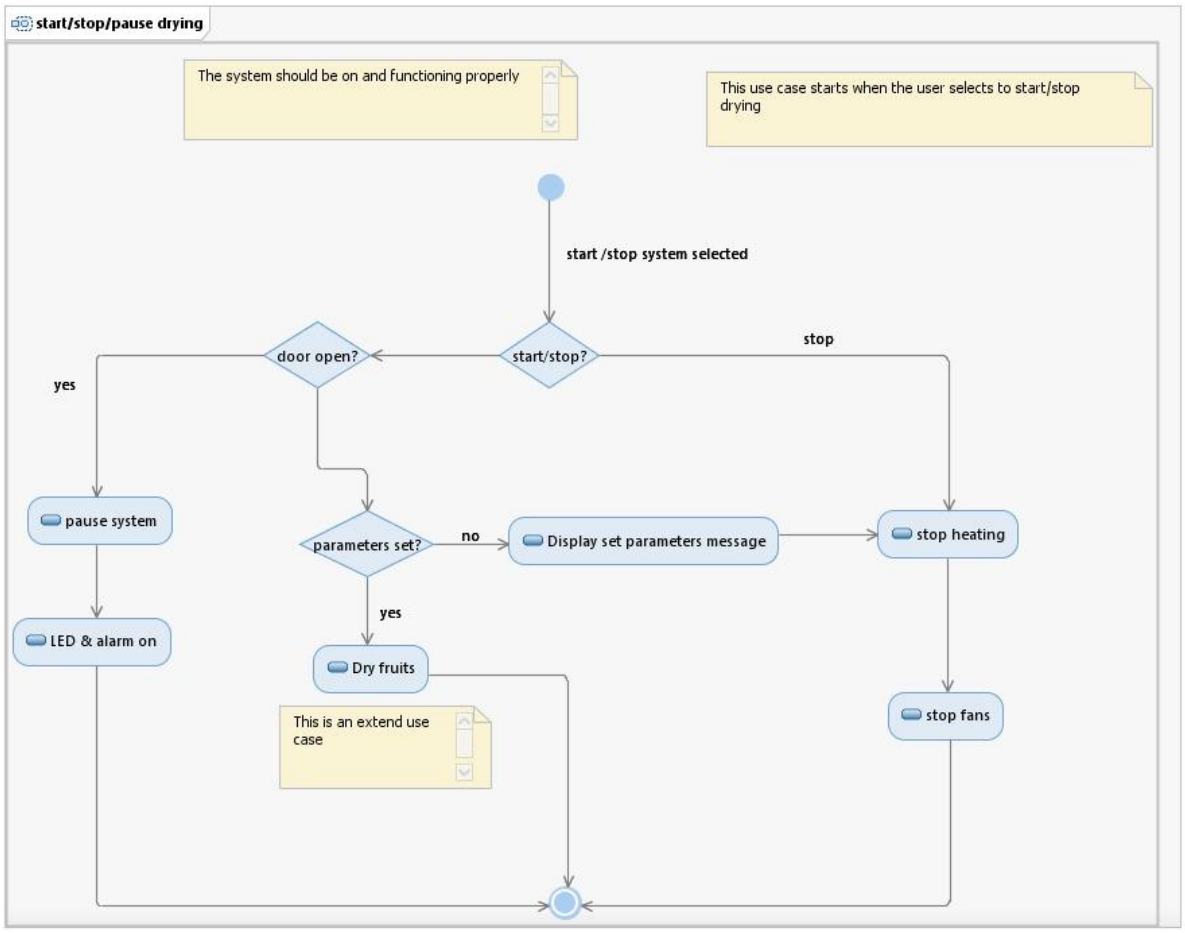

Figure 4 start/stop drying Activity diagram

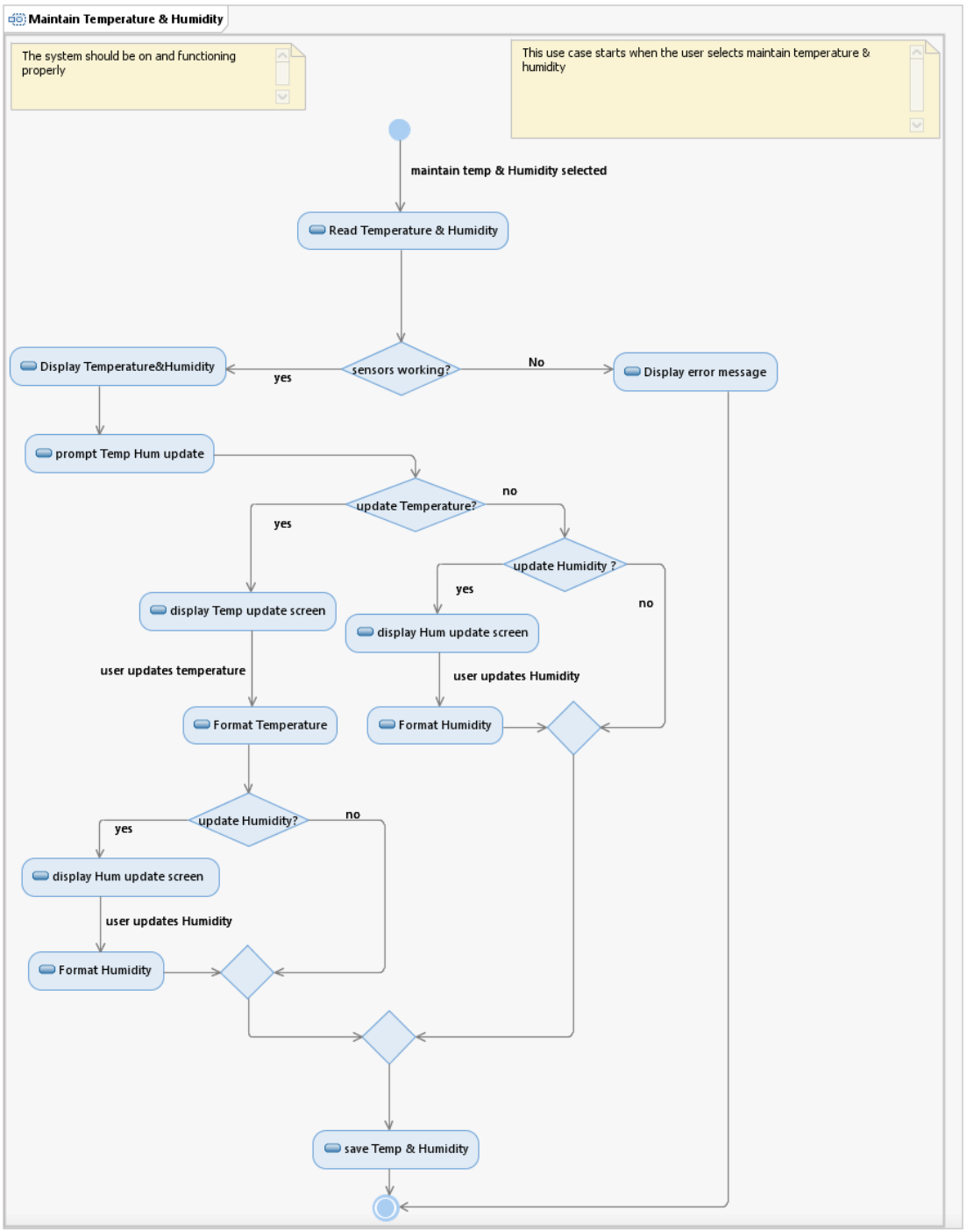

Figure 5 Maintain Temperature & Humidity activity diagram

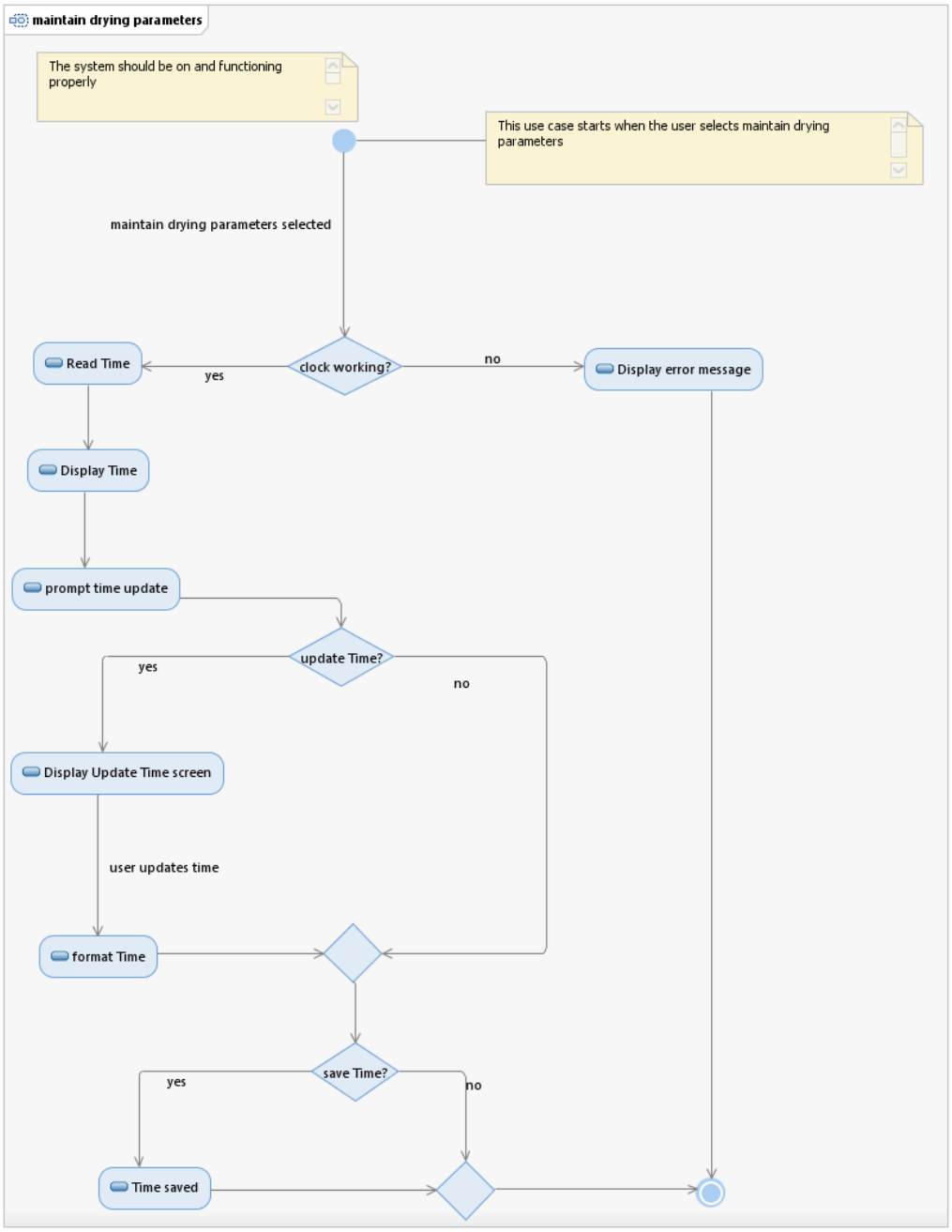

Figure 6 maintain drying parameters activity diagram

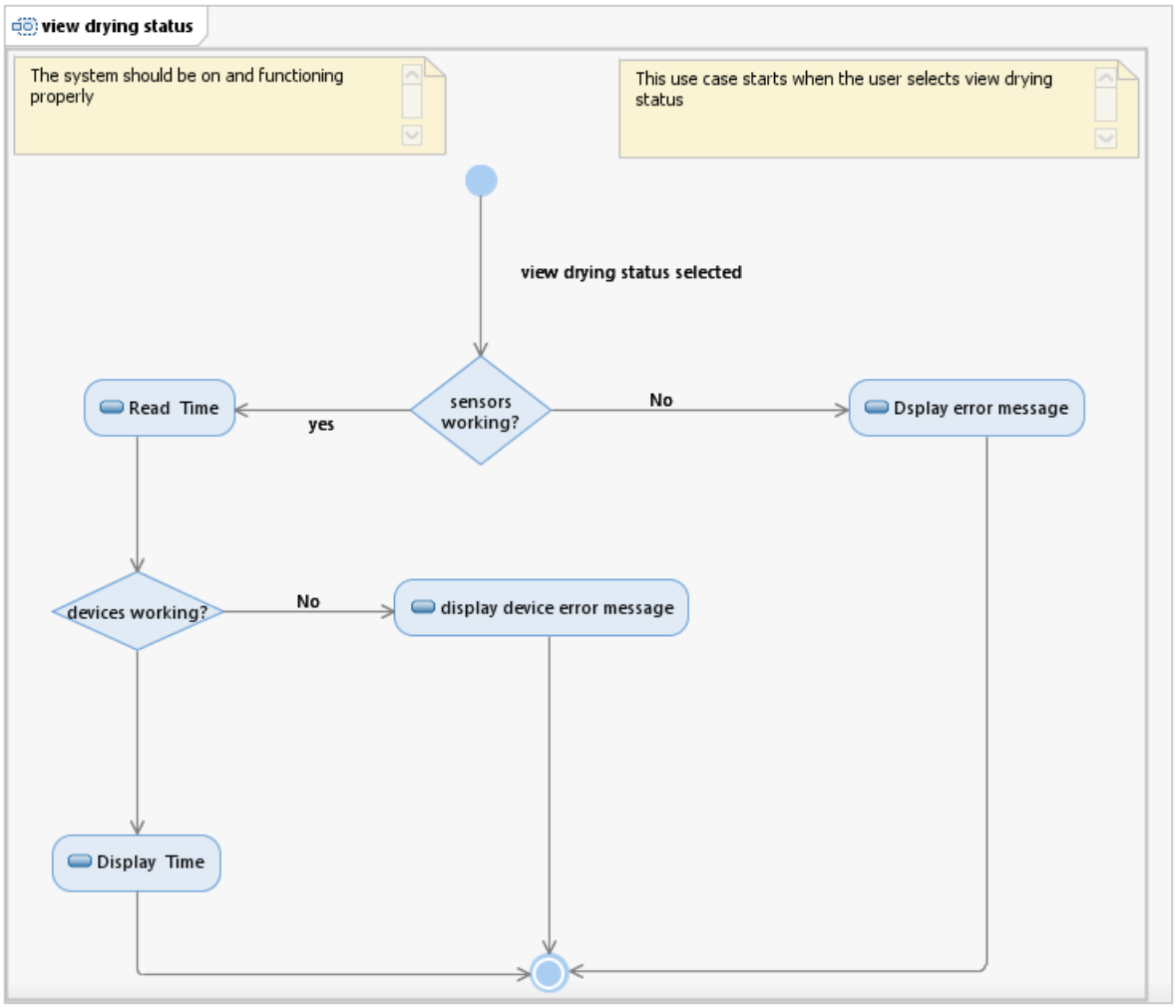

Figure 7 view drying status activity diagram

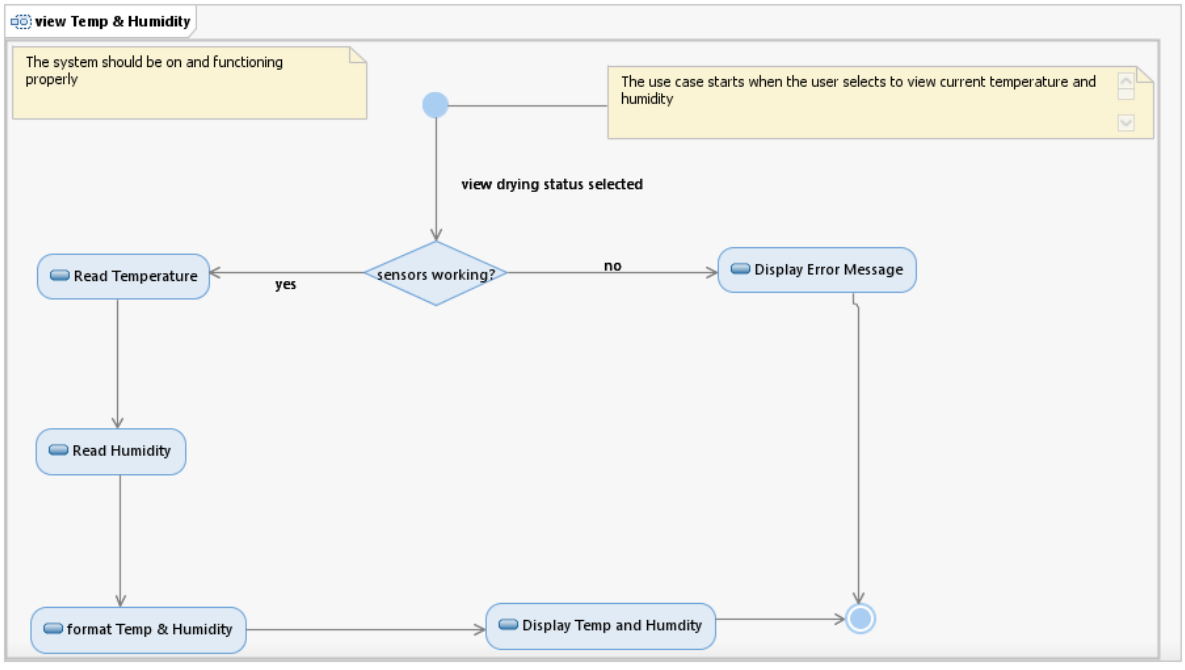

Figure 8 view Temp & Humidity activity diagram

2.5 Detail use case specifications

Use-Case Specification: dry fruits

1. Brief Description

This use case allows the user to dry fruits, it is an extend use case to the base use case start/stop/pause drying. .

2. Use Case Diagram (Participating Actors) User, motors.

3. Flow of Events

This use case starts when the User selects to start the system.

3.1 Basic Flow

3.1.1 Start pressed

3.1.1.1 The system detects that the start was selected. .

3.1.2 Door closed

3.1.2.1 The system detects from the door sensor that the door is closed.

3.1.3 Start motors

3.1.3.1 The system starts motors.

3.1.4 Display drying

3.1.4.1 The system should display to the user that it is drying.

3.1.5 Stop not pressed

3.1.5.1 The system detects that stop was not pressed and continues drying.

3.1.6 Drying complete

3.1.6.1 The system displays to the user that the drying is complete..

3.1.7 Use case ends

3.2 Alternative Flow

3.2.1 Door open.

In step 3.1.2 the system detected that the door was open.

The use case goes to the base use case and the use case ends.

3.2.2 Stop pressed

In step 3.1.5 the system detected that stop was pressed the use case goes to the base use case and the use case ends.

3.3 Sub Flows *None*

4. Key Scenarios

4.1 Success Scenario's

4.1.1 Display drying: Basic flow

4.2 Failure Scenario's

4.2.1 Pause drying: Basic flow, door opened.

4.2.2 Stop drying: Basic flow, stop pressed.

5. Preconditions

5.1 The system should be on and functioning properly.

6. Post condition

6.1 The system should have started drying or paused if the door is open or stopped drying when stopped was pressed,.

7. Extension Points None

8. Special Requirements None

9. Additional Information

On the outside of the dryer there should be a button which the User can use to start the use case.

Use-Case Specification: start/stop/pause drying

1. Brief Description

This use case allows the user to start/stop or pause the system.

2. Use Case Diagram (Participating Actors) User, door sensors, heaters and fans.

3. Flow of Events

This use case starts when the User selects to start the system.

3.1 Basic Flow

3.1.1 Start pressed

3.1.1.1 The system detects that the start was selected. .

3.1.2 Door closed

3.1.2.1 The system detects from the door sensor that the door is closed.

3.1.3 Parameters set.

3.1.3.1 The system detects that all parameters are set.

3.1.4 Start drying

3.1.4.1 The system jumps to the dry fruits extend use case.

3.1.5 Use case ends

3.2 Alternative Flow

3.2.1 Stop pressed

In step 3.1.1 the system detected that system was stop was pressed.

The system stops fans and heaters and the use case ends.

3.2.2 Door open

In step 3.1.2 the system detected that the door was open and the system could not be started the system pauses the system and turns led & alarm on the use case ends.

3.2.3 Parameters not set

In step 3.1.3 the system detected that some parameters were not set by the user. The system stops the devices the use case ends.

3.3 Sub Flows *None*

4. Key Scenarios

4.1 Success Scenario's

4.1.1 Start drying: Basic flow

4.2 Failure Scenario's

4.2.1 Stop drying: Basic flow, parameters not set.

5. Preconditions

5.1 The system should be on and functioning properly.

6. Post condition

6.1 The system should have started drying if it was stopped or stop if it was started. If the door was open the system should have paused the system. If the parameters were not set a parameters not set error message should be displayed on the LCD.

7. Extension Points None

8. Special Requirements None

9. Additional Information

On the outside of the dryer there should be a button which the User can use to start the use case.

Use-Case Specification: Maintain Temperature and Humidity

1. Brief Description

This use case allows the user to maintain the current temperature and humidity inside the drying unit.

2. Use Case Diagram (Participating Actors)

User, Temperature and Humidity sensors.

3. Flow of Events

This use case starts when the User selects to maintain the current drying parameters on the outside of the dryer unit.

3.1 Basic Flow

3.1.1 Read Temperature and Humidity

3.1.1.1 The system reads the current humidity and Temperature.

3.1.2 Sensors working properly

3.1.2.1 The system detects when drying to read the temperature and humidity that the sensors are working properly.

3.1.3 Display temperature and Humidity

3.1.3.1 The system displays the current temperature and humidity on the LCD.

3.1.4 Prompt user update

3.1.4.1 The system displays an update Temperature and humidity option to the user

3.1.5 Display Temperature update screen

3.1.5.1 The system detects that the user has opted to update the temperature and displays the update screen to allow the user to input values.

3.1.6 Format Temperature

3.1.6.1 The system formats the user updated Temperature.

3.1.7 update humidity option

3.1.7.1 The system gives the user an option to update the humidity option to the user

3.1.8 Display Humidity update screen

3.1.8.1 The system detects that the user has opted to update the Humidity and displays the update screen to allow the user to input values.

3.1.9 Format Humidity

3.1.9.1 The system formats the user updated Humidity.

3.1.10 Save temp and humidity

3.1.10.1 The system saves the temperature and humidity.

3.1.11 Use case ends

3.2 Alternative Flow

3.2.1 Sensors not working

In step 3.1.2 the system detected that the clock does not work or cannot be reached. The system displays an error message to the User on the LCD. The use case ends.

3.2.2 No temperature update

In step 3.1.4 the system detected that the user opted not to update the temperature, the use case goes to the humidity option, saves and ends.

3.2.3 Humidity not updated

In step 3.1.7 the system detects that the Humidity was not updated by the user. The use case goes to save temperature and humidity and ends.

3.3 Sub Flows *None*

4. Key Scenarios

4.1 Success Scenario's

4.1.1 Display Temperature and humidity: Basic flow

4.1.2 Update Temperature and humidity: Basic flow

4.1.3 Save temperature and humidity : Basic flow

4.2 Failure Scenario's

4.2.1 device not working: Basic flow, Sensors not working

5. Preconditions

5.1 The system should be on and functioning properly.

6. Post condition

6.1 The system should have displayed the Temperature and humidity to the User on the LCD, allowed the user to update the temperature and humidity. If the temperature and humidity sensors are not working an error message should be displayed on the LCD.

7. Extension Points None

8. Special Requirements None

9. Additional Information

On the outside of the dryer there should be a button which the User can use to start the use case. There should also be an LCD display to display the current temperature and humidity in degrees and percentage.

Use-Case Specification: Maintain drying parameters

1. Brief Description

This use case allows the user to view, update and save the drying parameters inside the drying unit

2. Use Case Diagram (Participating Actors)

User, clock, sensors.

3. Flow of Events

This use case starts when the User selects to maintain the current drying parameters on the outside of the dryer unit.

3.1 Basic Flow

3.1.1 Clock working

3.1.1.1 The system detected that the clock is working correctly and can be accessed

3.1.2 Read Time

3.1.2.1 The system reads the Time.

3.1.3 Display Time

3.1.3.1 The system displays the read time on the LCD.

3.1.4 Prompt user update

3.1.4.1 The system displays an update Time option to the user

3.1.5 Display time update screen

3.1.5.1 The system detects that the user has opted to update the time and displays the update screen to allow the user to input values.

3.1.6 Format Time

3.1.6.1 The system formats the user updated time.

3.1.7 save time option

3.1.7.1 The system gives the user the option to select to save time or not.

3.1.8 Time saved

3.1.8.1 The system detects that the user has saved the time.

3.1.9 Use case ends

3.2 Alternative Flow

3.2.1 **clock not working**

In step 3.1.1 the system detected that the clock does not work or cannot be reached. The system displays an error message to the User on the LCD. The use case ends.

3.2.2 **No time update**

In step 3.1.4.1 the system detected that the user opted not to update the Time, the use case goes to the save option.

3.2.3 **Time not saved**

In step 3.1.7 the system detects that the temperature was not saved. The use case ends.

3.3 Sub Flows *None*

4. Key Scenarios

4.1 Success Scenario's

4.1.1 Display Time: Basic flow

4.1.2 Update Time: Basic flow

4.1.3 Save time : Basic flow

4.2 Failure Scenario's

4.2.1 device not working: Basic flow, Sensors not working

5. Preconditions

5.1 The system should be on and functioning properly.

6. Post condition

6.1 The system should have displayed the Time to the User on the LCD, allowed the user to update the time and save the time they want to extend drying or if the devices are not working an error message should be displayed on the LCD.

7. Extension Points None

8. Special Requirements None

9. Additional Information

On the outside of the dryer there should be a button which the User can use to start the use case. There should also be an LCD display to display the current time, in minutes and seconds.

Use-Case Specification: View drying status

1. Brief Description

This use case allows the user to view the drying status inside the dryer unit

2. Use Case Diagram (Participating Actors) User, clock, sensors

3. Flow of Events

This use case starts when the User selects to view drying status on the outside of the dryer unit.

3.1 Basic Flow

3.1.1 Sensors working correctly

3.1.1.1 The system detected that the sensors can be accessed and are working correctly.

3.1.2 Read Time

3.1.2.1 The system reads the Time.

3.1.3 Clock working properly

3.1.3.1 The system detected that the clock can be accessed and is working correctly.

3.1.4 Display Time

3.1.4.1 The system displays the Time on the LCD.

3.1.5 Use case ends

3.2 Alternative Flow

3.2.1 **Sensors not working**

In step 3.1.1 the system detected that the sensors does not work or cannot be reached. The system displays an error message to the User on the LCD. The use case ends.

3.2.2 **Time device not working**

In step 3.1.3 the system detected that the clock does not work or cannot be reached. The system displays an error message to the User on the LCD. The use case ends.

3.3 Sub Flows *None*

4. Key Scenarios

4.1 Success Scenario's

4.1.1 Display time: Basic flow

4.2 Failure Scenario's

4.2.1 Sensors not working: Basic flow, Sensors not working

4.2.2 Devices not working: alternative flow, devices not working

5. Preconditions

5.1 The system should be on and functioning properly.

6. Post condition

6.1 The system should have displayed the Time elapsed and time left to the User on the LCD, or if the sensors are not working or if the devices are not working an error message should be displayed on the LCD.

7. Extension Points None

8. Special Requirements None

9. Additional Information

On the outside of the dryer there should be a button which the User can use to start the use case. There should also be an LCD display to display the time left and elapsed time.

Use-Case Specification: View current temperature & humidity

1. Brief Description

This use case allows the user to view the current temperature and humidity inside the dryer unit

2. Use Case Diagram (Participating Actors) User, Temperature sensor, Humidity sensor

3. Flow of Events

This use case starts when the User selects to view the current temperature & humidity on the outside of the dryer unit.

3.1 Basic Flow

3.1.1 Sensors working correctly

3.1.1.1 The system detected that the sensors can be access and are working correctly.

3.1.2 Read temperature

3.1.2.1 The system reads the current temperature.

3.1.3 Read humidity

3.1.3.1 The system reads the current humidity.

3.1.4 Format temperature & humidity

3.1.4.1 The system formats the temperature and humidity to be displayed.

3.1.5 Display temperature & humidity

3.1.5.1 The system displays the formatted temperature & humidity on the LCD.

3.1.6 Use case ends

3.2 Alternative Flow

3.2.1 **Sensors not working**

In step 3.1.1 the system detected that the sensors does not work or cannot be reached. The system displays an error message to the User on the LCD. The use case ends.

3.3 Sub Flows *None*

4. Key Scenarios

4.1 Success Scenario's

4.1.1 Display temperature & humidity: Basic flow

4.2 Failure Scenario's

4.2.1 Sensors not working: Basic flow, Sensors not working

5. Preconditions

5.1 The system should be on and functioning properly.

6. Post condition

6.1 The system should have displayed the current temperature and humidity to the User on the LCD, or if the sensors are not working an error message should be displayed on the LCD.

7. Extension Points None

8. Special Requirements None

9. Additional Information

On the outside of the dryer there should be a button which the User can use to start the use case. There should also be an LCD display to display the current temperature and humidity.

2.6 **Summary**

This chapter introduced the requirements artefacts for the smart food drying system. The next chapter provides the analysis model for the system.

Chapter 3 Analysis model

3.1 Introduction

This chapter will expand on the sub-problems of the smart fruit drying system. The Analysis model related to all sub-problem is provided.

3.2 Use case realizations

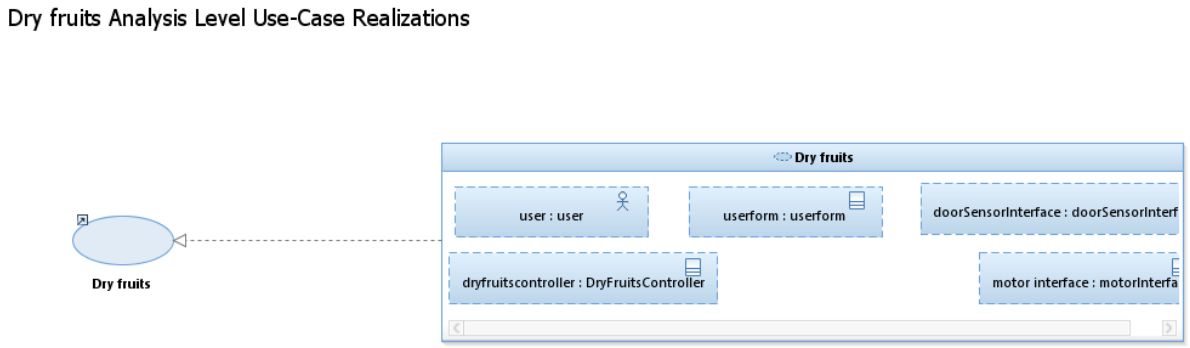

Figure 9 dry fruits use case realization

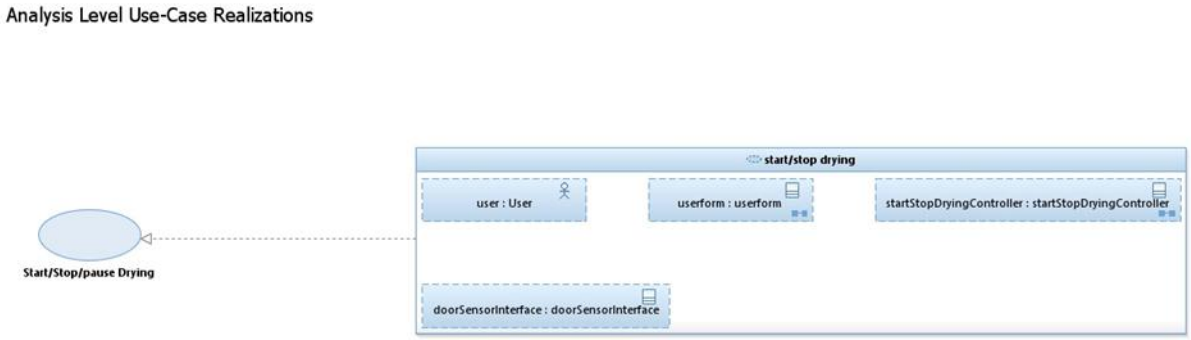

Figure 10 start/stop drying use realization

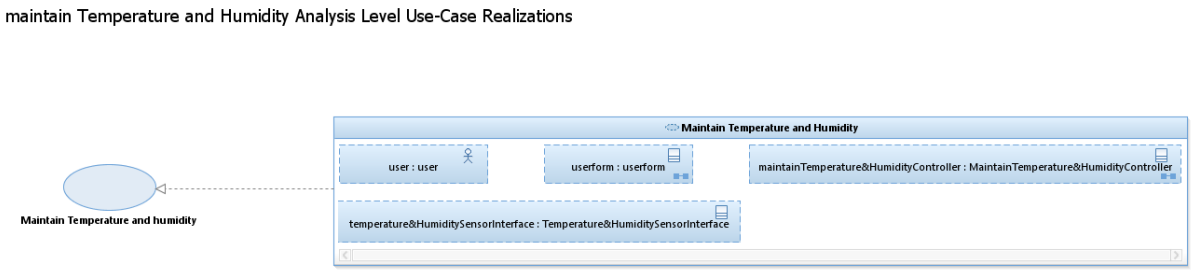

Figure 11 Maintain Temperature & Humidity use case realization

maintain drying parameters Analysis Level Use-Case Realizations

Figure 12 maintain drying parameters use case realization

view drying status Analysis Level Use-Case Realizations

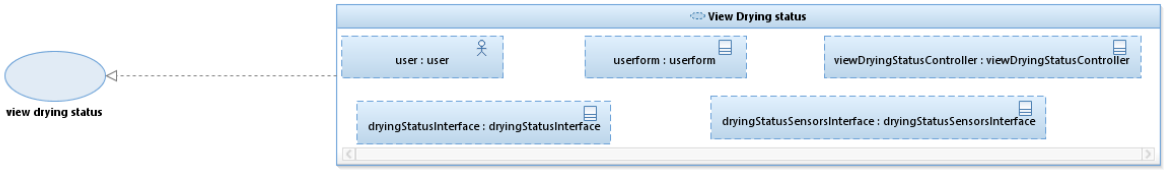

Figure 13 view drying use case realization

view Temperature and Humidity Analysis Level Use-Case Realizations

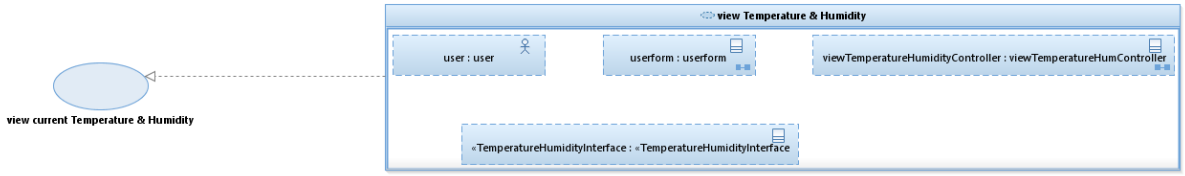

Figure 14 View TempHumidity use realization

3.3 **Analysis classes**

Table 1 Analysis classes for maintain temperature use case

Table 1 Dry Fruits

Boundary	userform doorsensorinterface motorinterface	
control	Dryfruitscontroller	
entity	motor	

Table 2 start/stop/pause drying

Boundary	userform TemperaturesensorInterface DoorSensorinterface FanInterface heaterInterface	
control	startStopDryingController	
Entity	Temperature Temperaturesensor Fan Heater Alarm LED	

Table 3 maintain temp & Hum

Boundary	userform Temperature&HumiditysensorInterface	
control	Maintaintemperaturecontroller	
Entity	Temperature humidity Fan heater	

Table 4 maintain drying parameters

Boundary	Userform ClockInterface	
control	MaintainDryingParameterscontroller	
Entity	Clock	

Table 5 view drying status

Boundary	userform dryStatusInterface dryingStatusSensorinterface	
control	viewDryingStatuscontroller	
Entity		

Table 6 view current Temp & Humidity

Boundary	userform Temperrature&HumiditysensorInterface TempHuminterface	
control	MaintainTemperatureHumiditycontroller	
Entity	Temperature Temperaturesensor	

3.3.1 **Noun extraction and CRC cards**

Table 7 Noun extraction table

Candidate Noun	Is this candidate inside our system boundary?	Is this candidate an Entity?	Does this candidate have identifiable behaviour for our problem domain?	Does this candidate have relationships with any other candidates?	Is the candidate a Class?
user	NO	NO	NO	NO	NO
Temperature	YES	YES	YES	YES	YES
Humidity	YES	YES	YES	YES	yes
sensors	YES	YES	YES	YES	YES
system	NO	NO	YES	NO	NO
fan	YES	YES	NO	NO	NO
heater	YES	YES	NO	NO	NO
motor	YES	YES	NO	NO	NO
clock	YES	YES	NO	YES	YES

Table 8 Class responsibility collaboration card for class

Class Responsibility Collaboration Cards *<<Class name>>*	
RESPONSIBILITY	COLLABORATION
Knows sensor's name	maintanTempHumController TemperatureSensorsInterface MaintainTemperatureHumidityController MaintainDryingParametersController startStopDryingController viewDryingStatusController DryFruitsController
Knows sensor's	MaintainTemperatureHumidityController MaintainDryingParametersController startStopDryingController viewDryingStatusController DryFruitsController
Know sensor description	MaintainTemperatureHumidityController MaintainDryingParametersController startStopDryingController viewDryingStatusController DryFruitsController
Get sensors	MaintainTemperatureHumidityController MaintainDryingParametersController startStopDryingController viewDryingStatusController DryFruitsController

3.4 Class diagrams

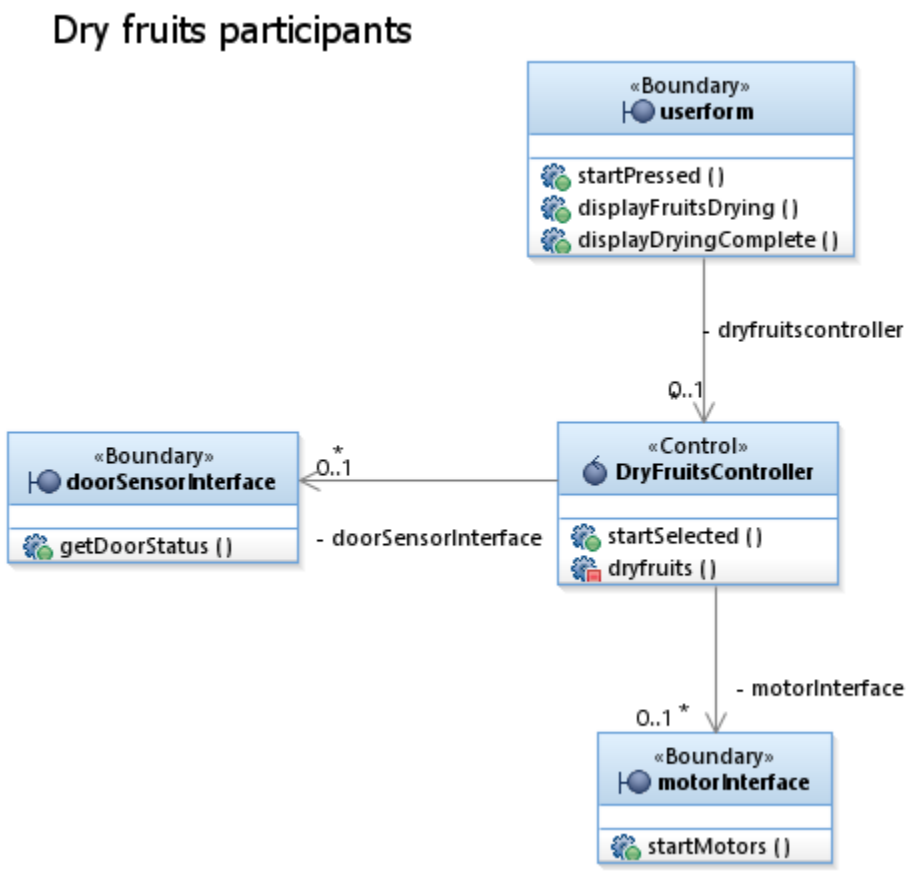

Figure 15 dry fruits analysis class diagram

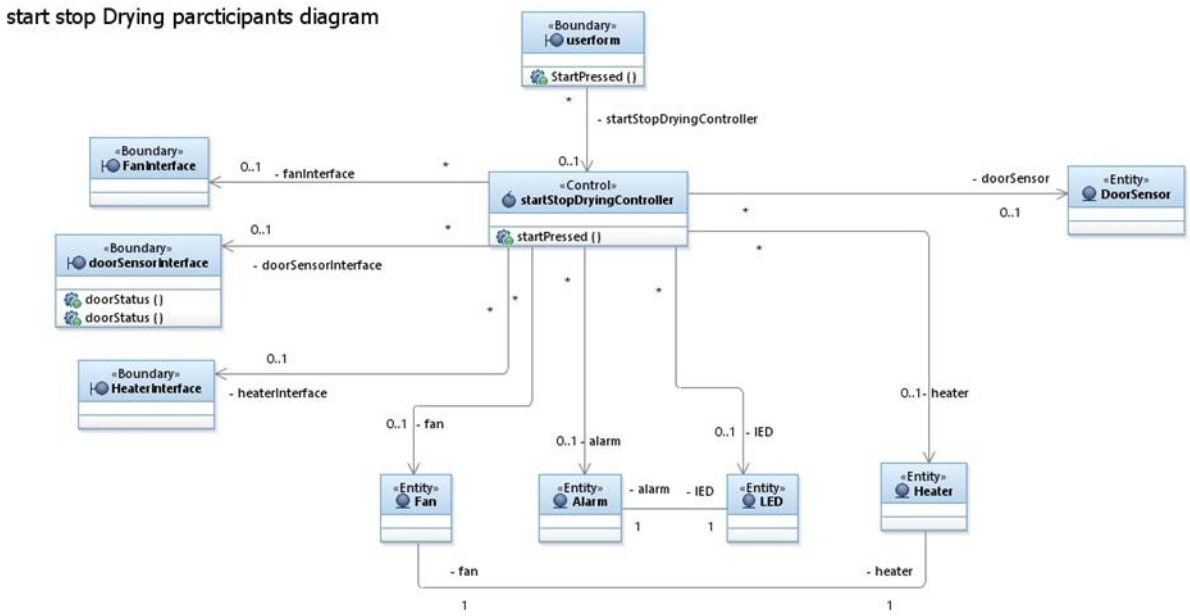

Figure 16 start/stop drying analysis class diagram

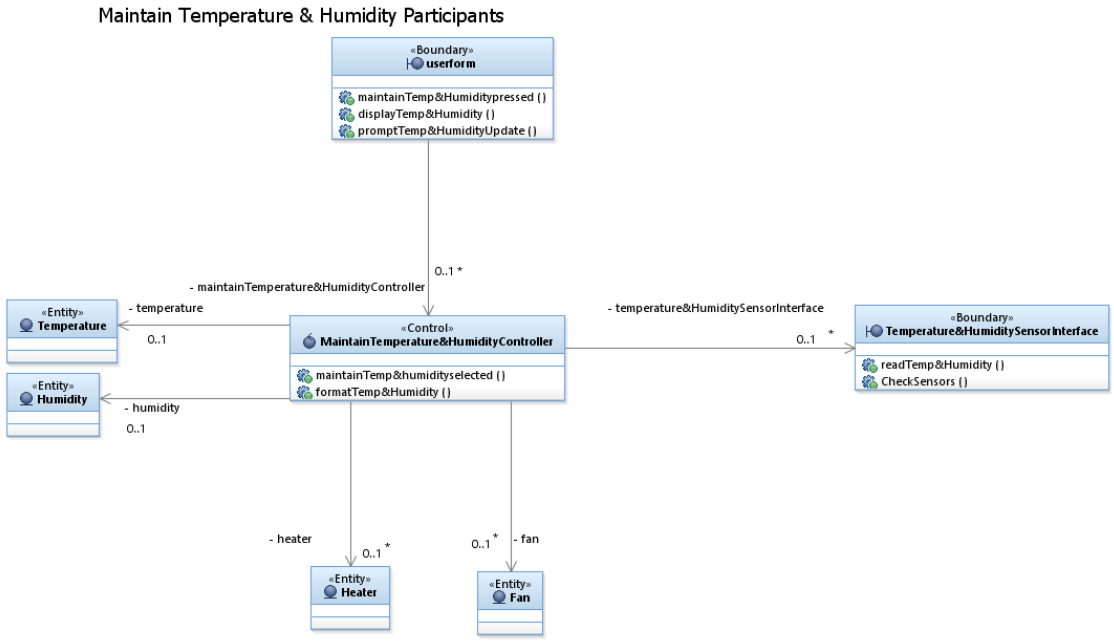

Figure 17 maintain Temp & Humidity analysis level class diagram

maintain drying parameters Participants

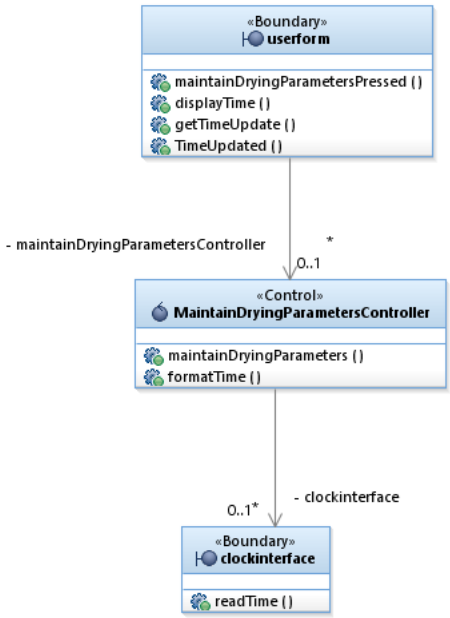

Figure 18 Maintain drying parameters analysis level class diagram

${use.case} Participants

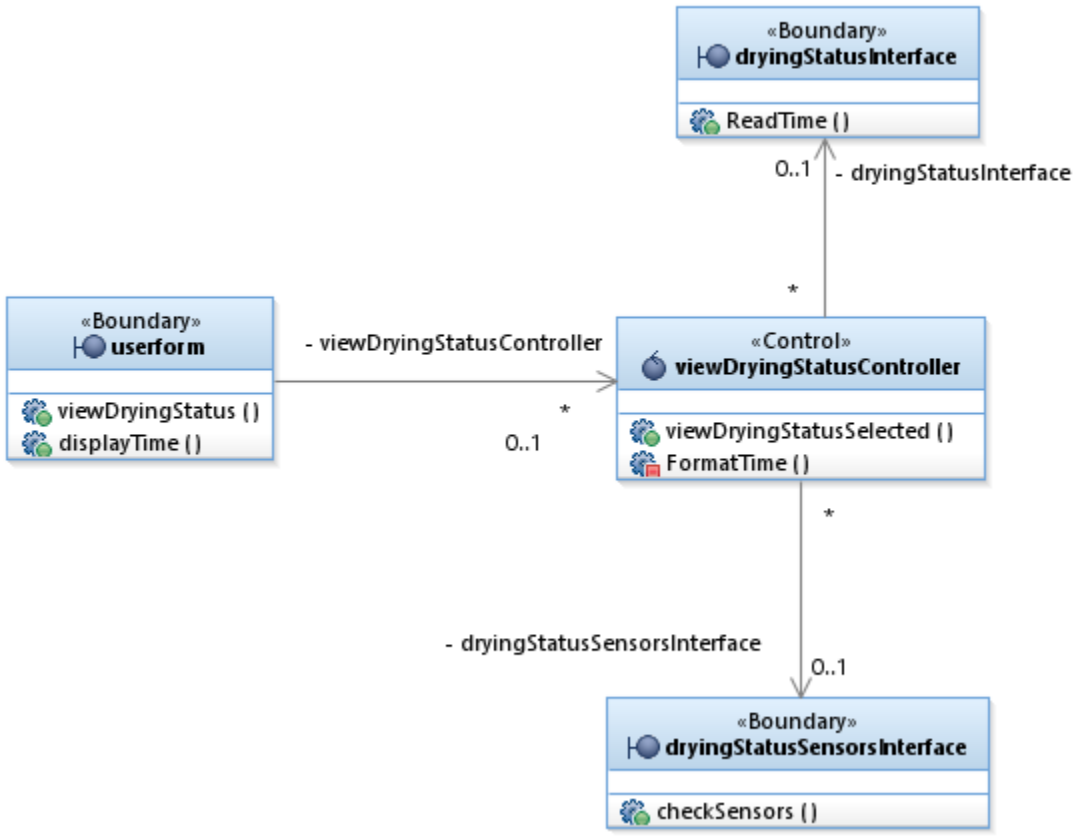

Figure 19view drying status analysis class diagram

View Temperature and Humidity Participants

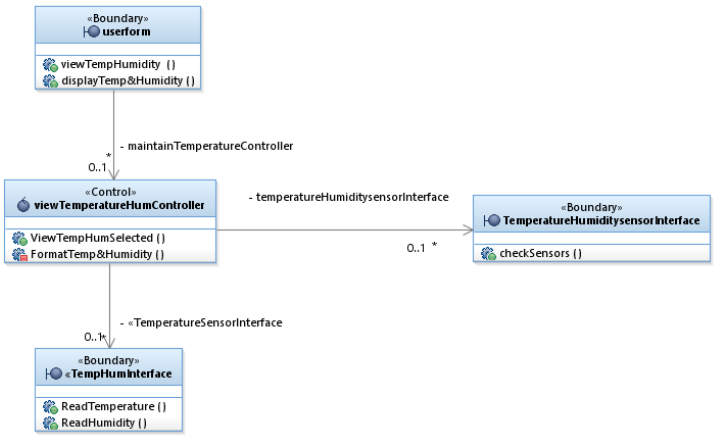

Figure 20 view Temp & Humidity class diagram

3.5 Sequence diagrams

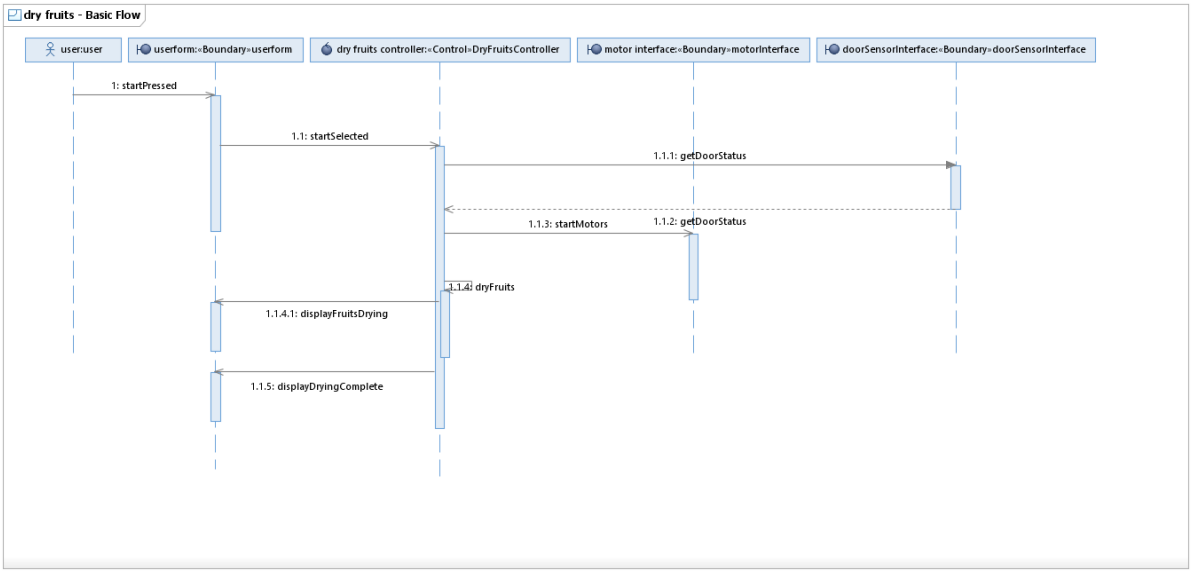

Figure 21 dry fruits sequence diagram

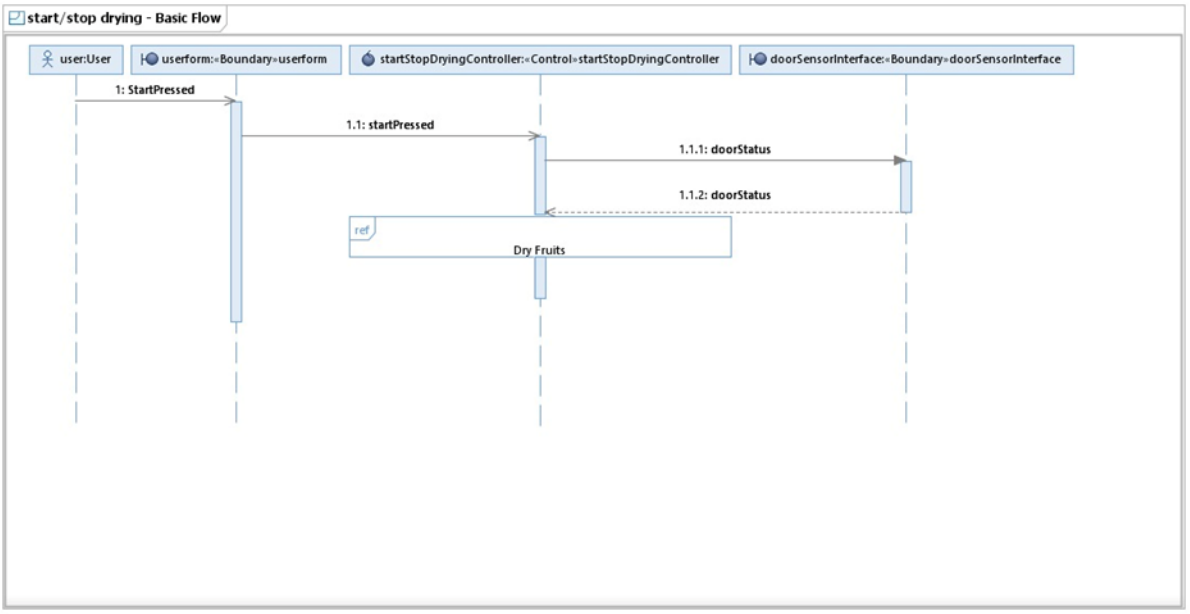

Figure 22 start/stop drying sequence diagram

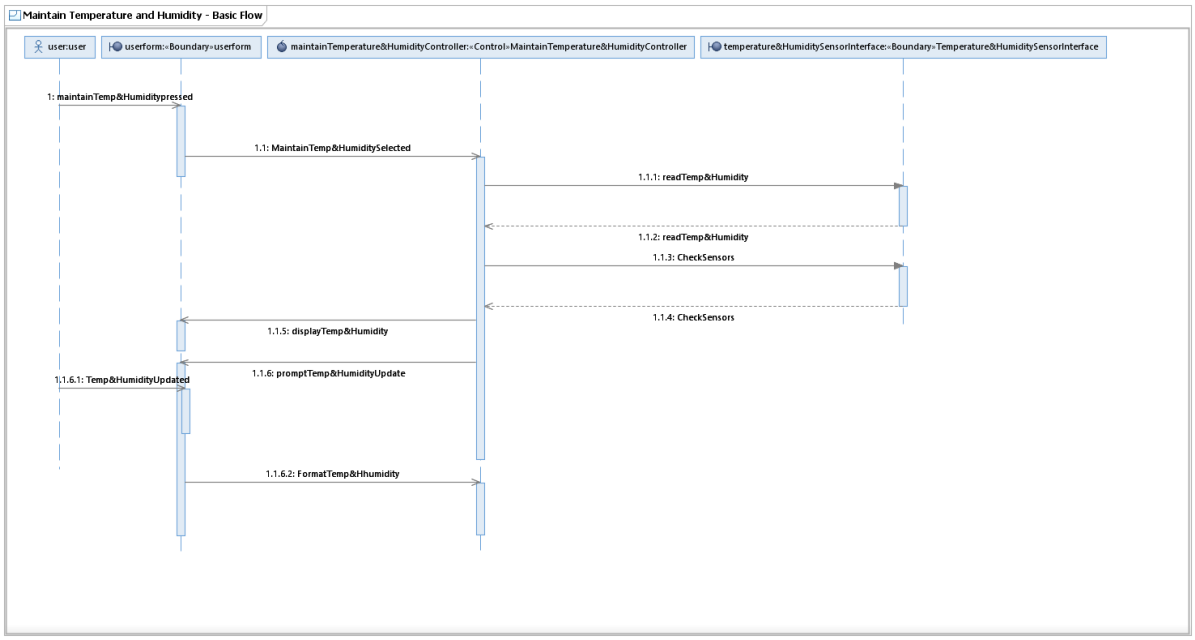

Figure 23 Maintain Temp & Humidity analysis level sequence diagram

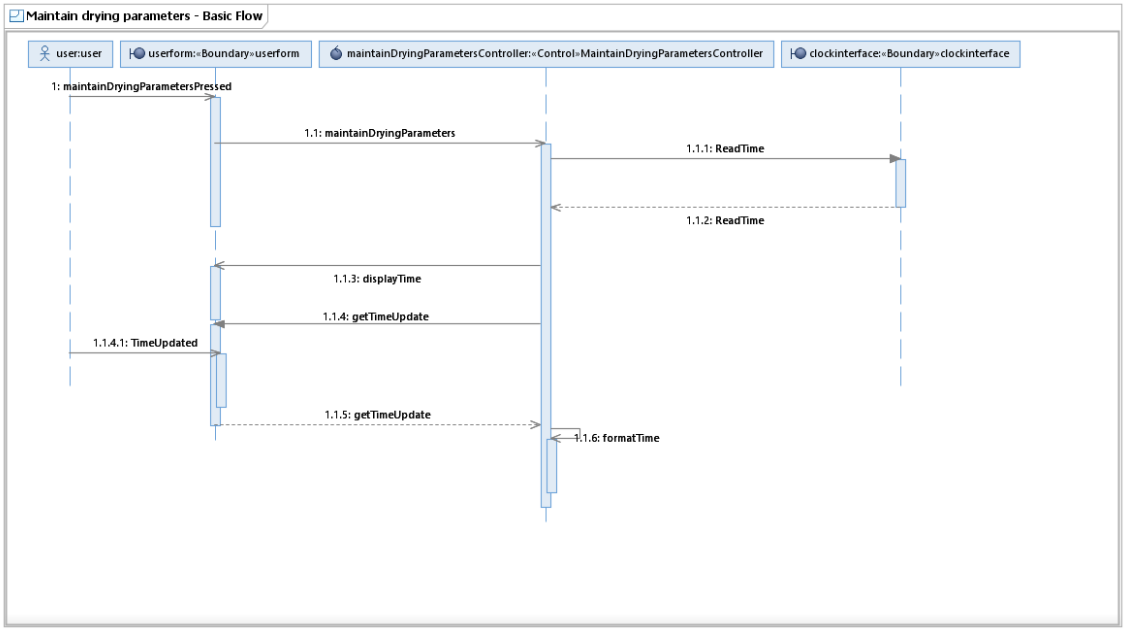

Figure 24 maintain drying parameters analysis level sequence diagram

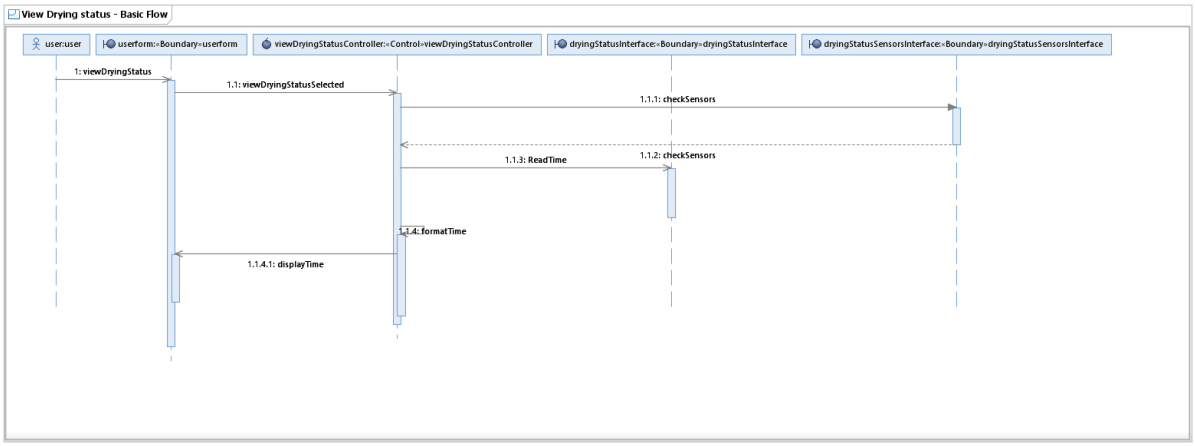

Figure 25 view drying status analysis sequence diagram

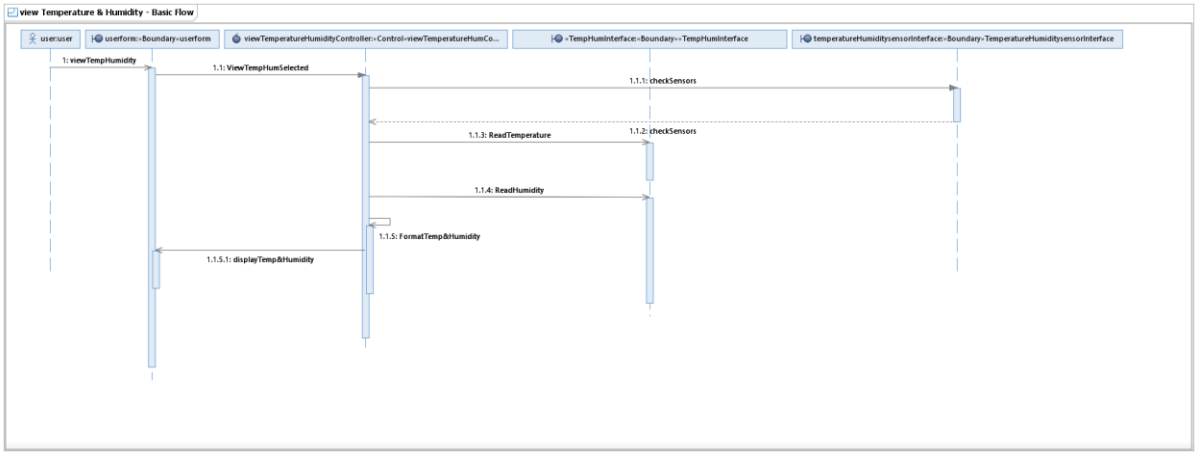

Figure 26 view Temp & Humidity analysis level sequence diagram

3.6 State-charts

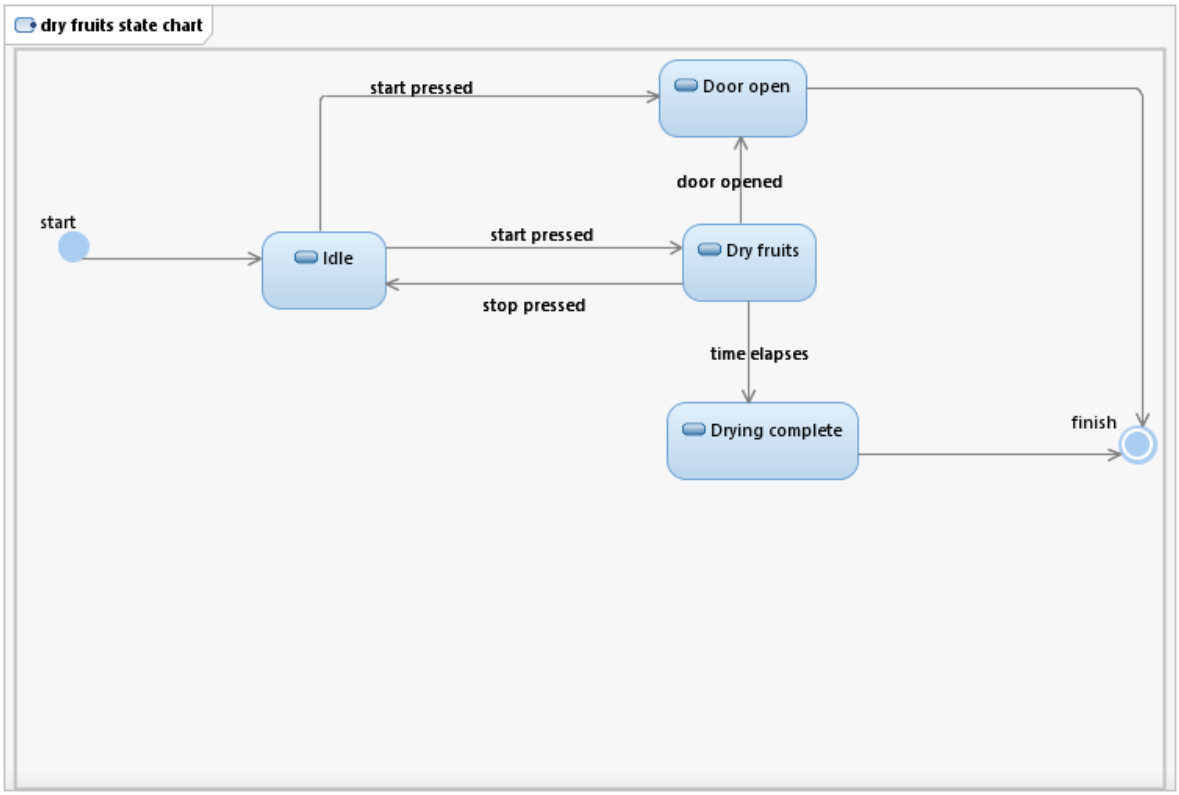

Figure 27 dry fruits state chart

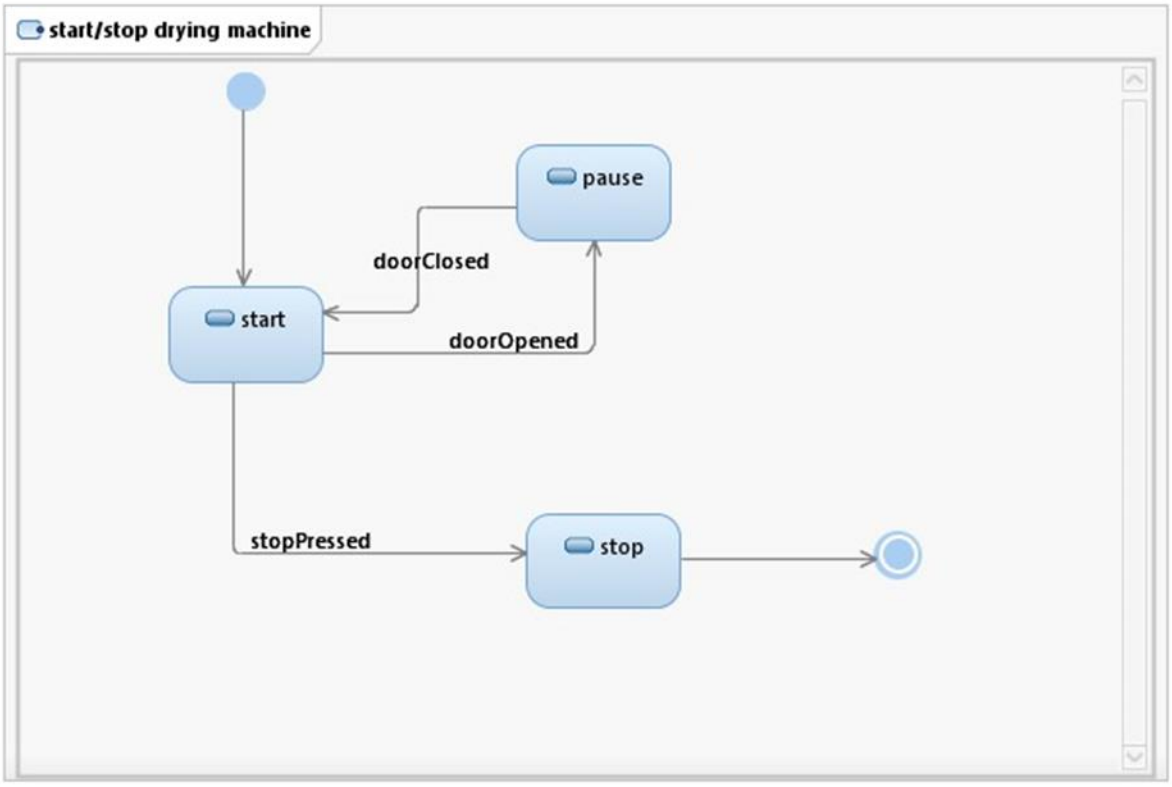

Figure 28 start/stop state machine

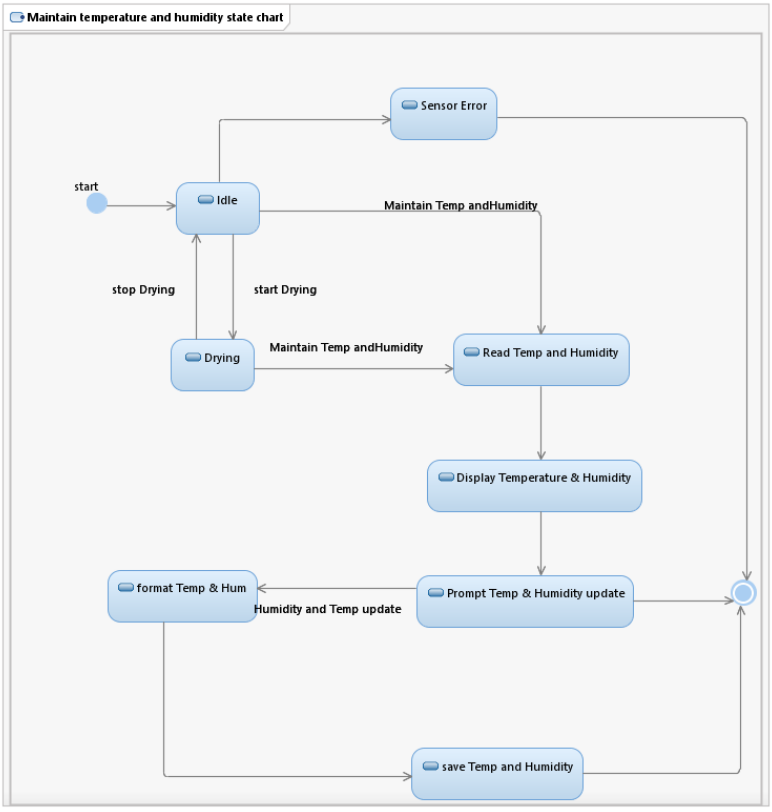

Figure 29 maintain Temp and Humidity state chart

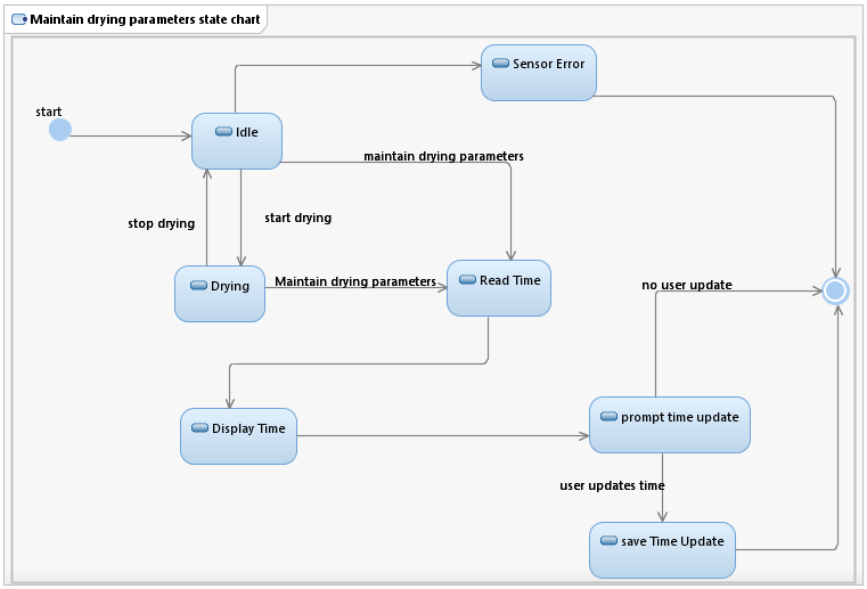

Figure 30 Maintain drying parameters state chart

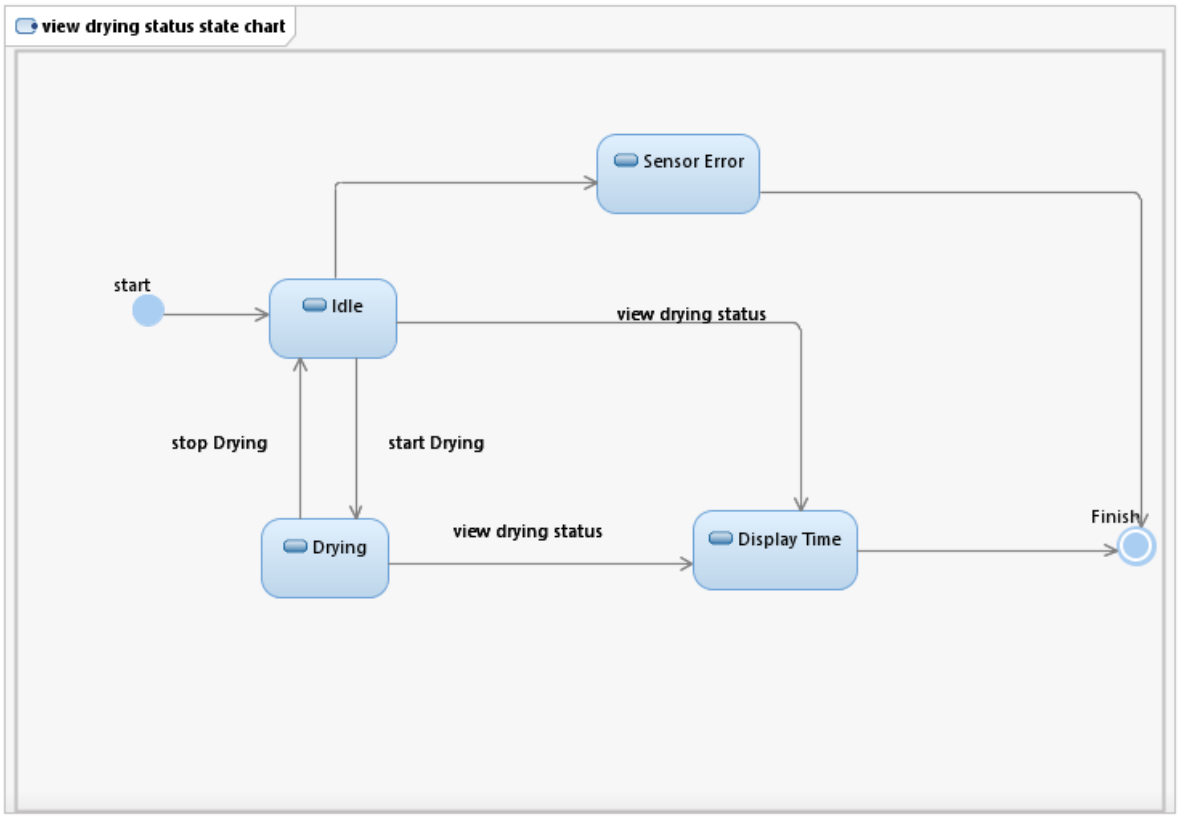

Figure 31 view drying status state chart

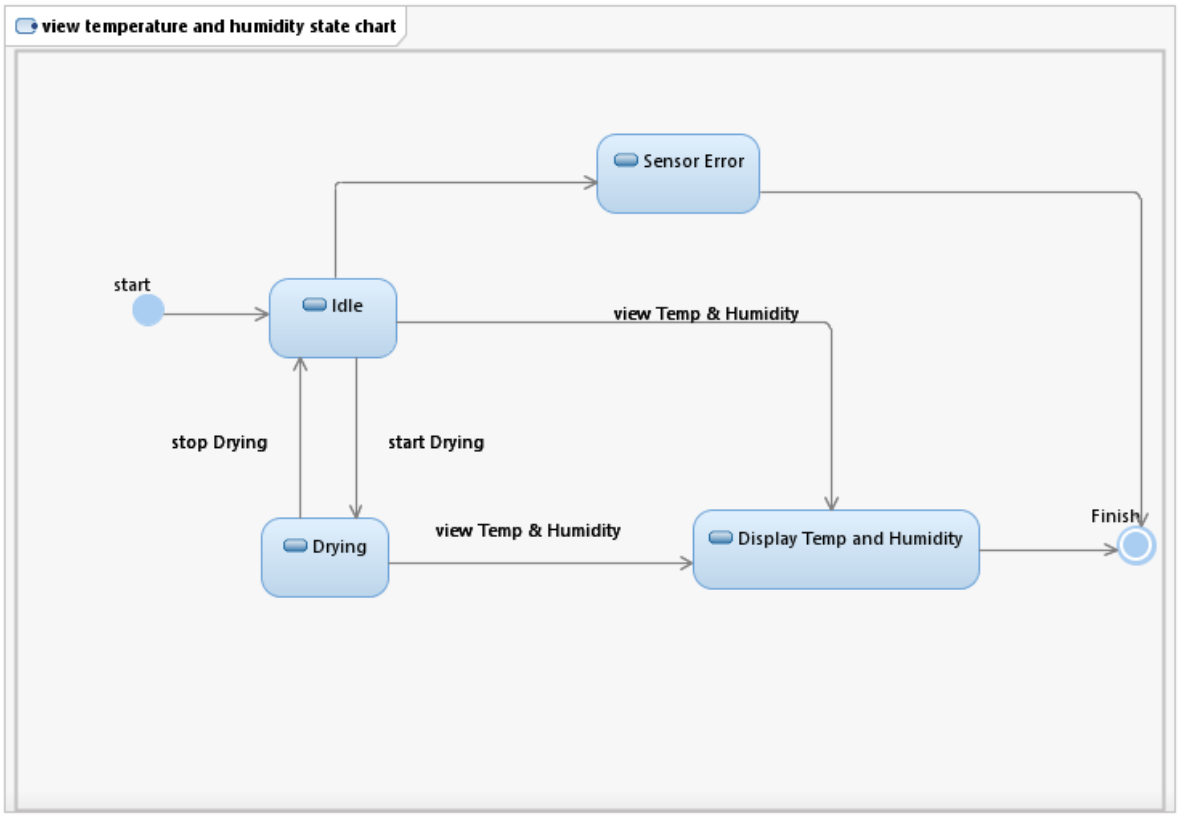

Figure 32 view Temp & Humidity state chart

3.7 **Summary**

This chapter provided the analysis artefacts for the smart fruit drying sub-problems. The next chapter provides the Design artefacts for the smart fruit drying sub problems.

Chapter 4 Design model

4.1 Introduction

This chapter will expand on the sub-problems of the smart fruit drying system. The Design model related to all sub-problem is provided.

4.2 Architectural overview diagram

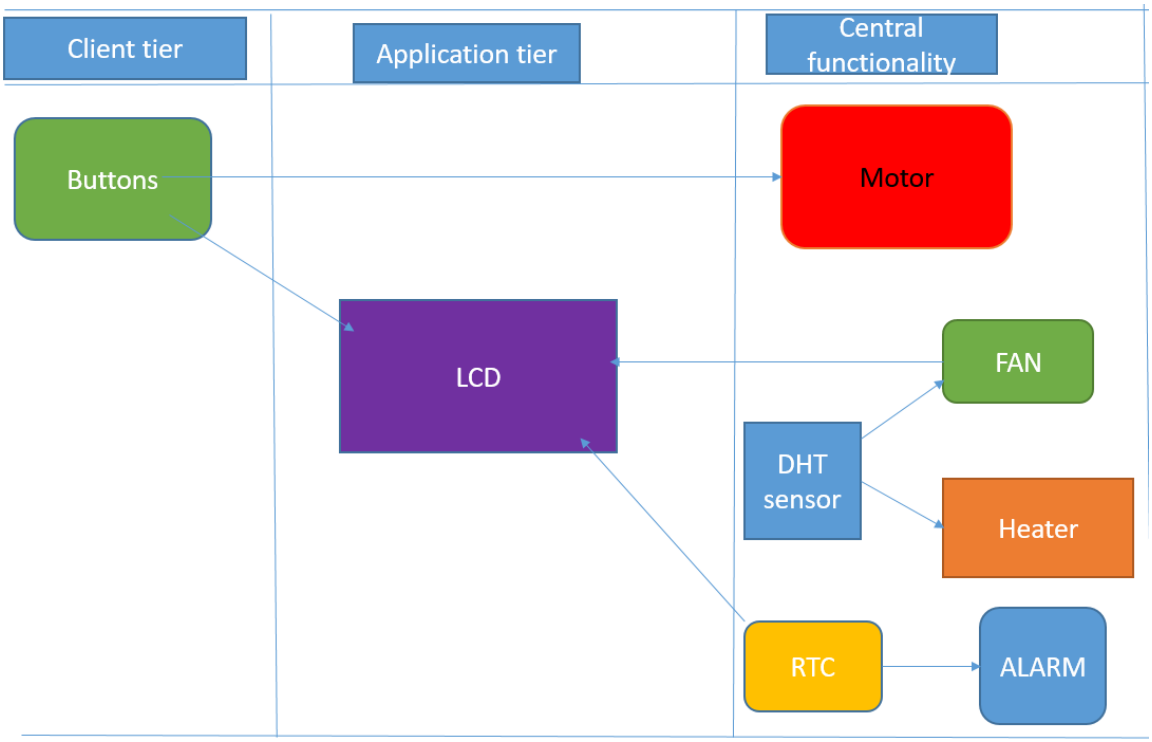

Figure 33 smart fruit drying architectural diagram

4.3 User interface prototypes

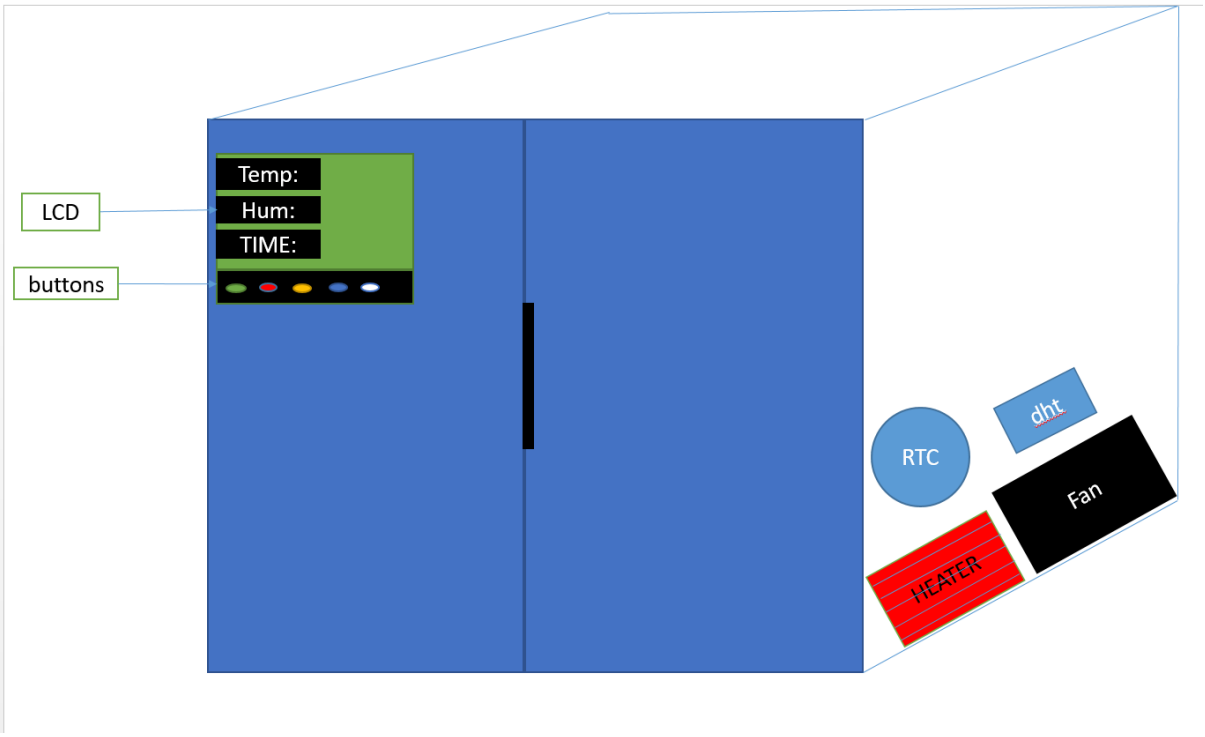

Figure 34 user interface prototype

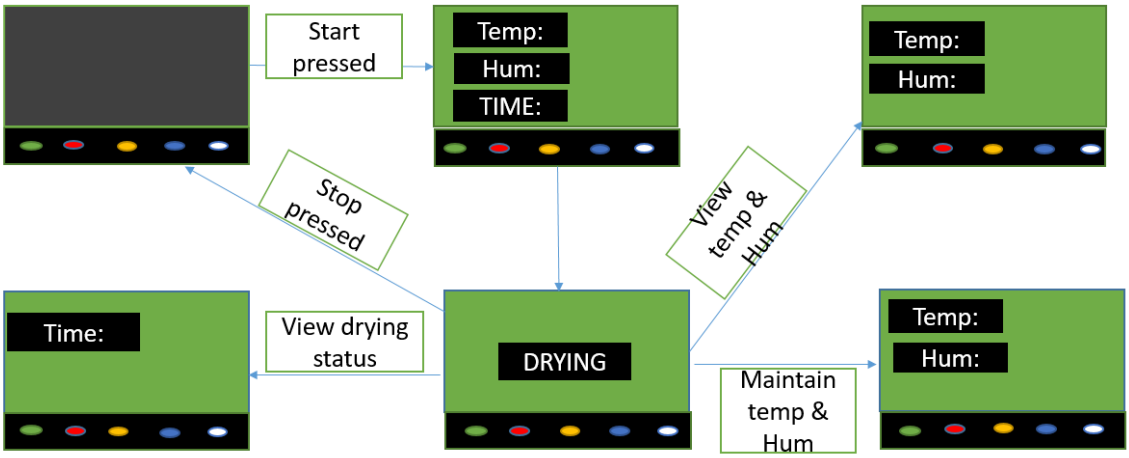

Figure 35 user interaction

4.4 Design sequence diagrams

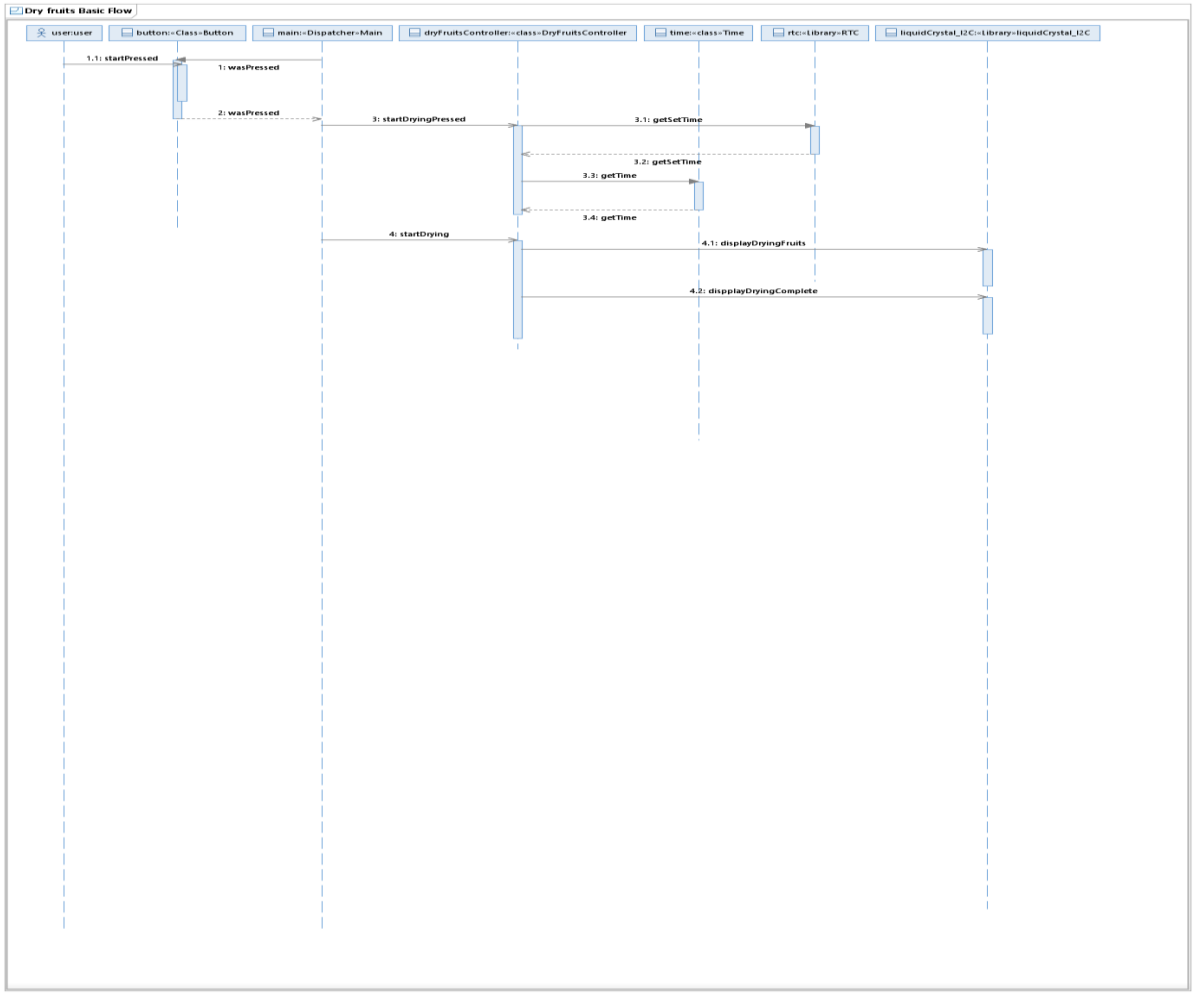

Figure 36 dry fruits sequence diagram

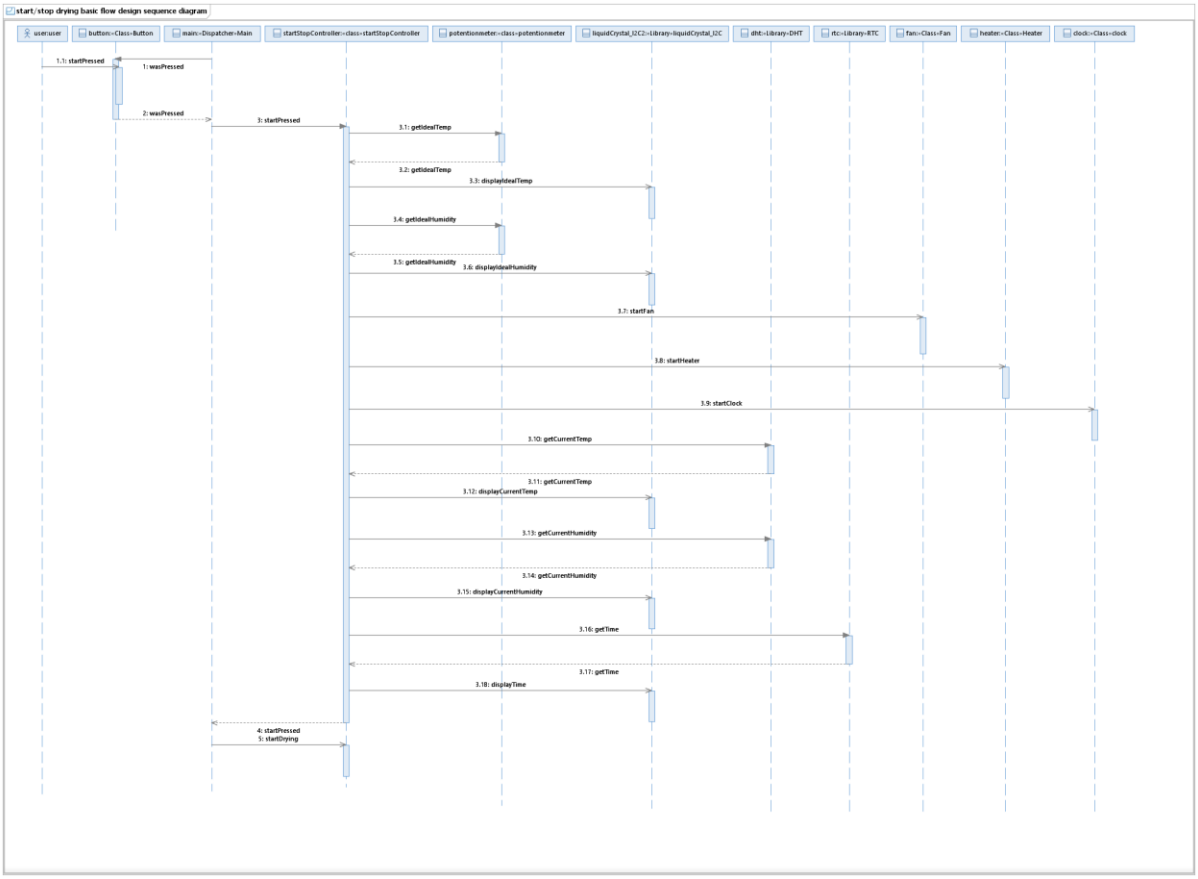

Figure 37 start/stop drying design sequence diagram

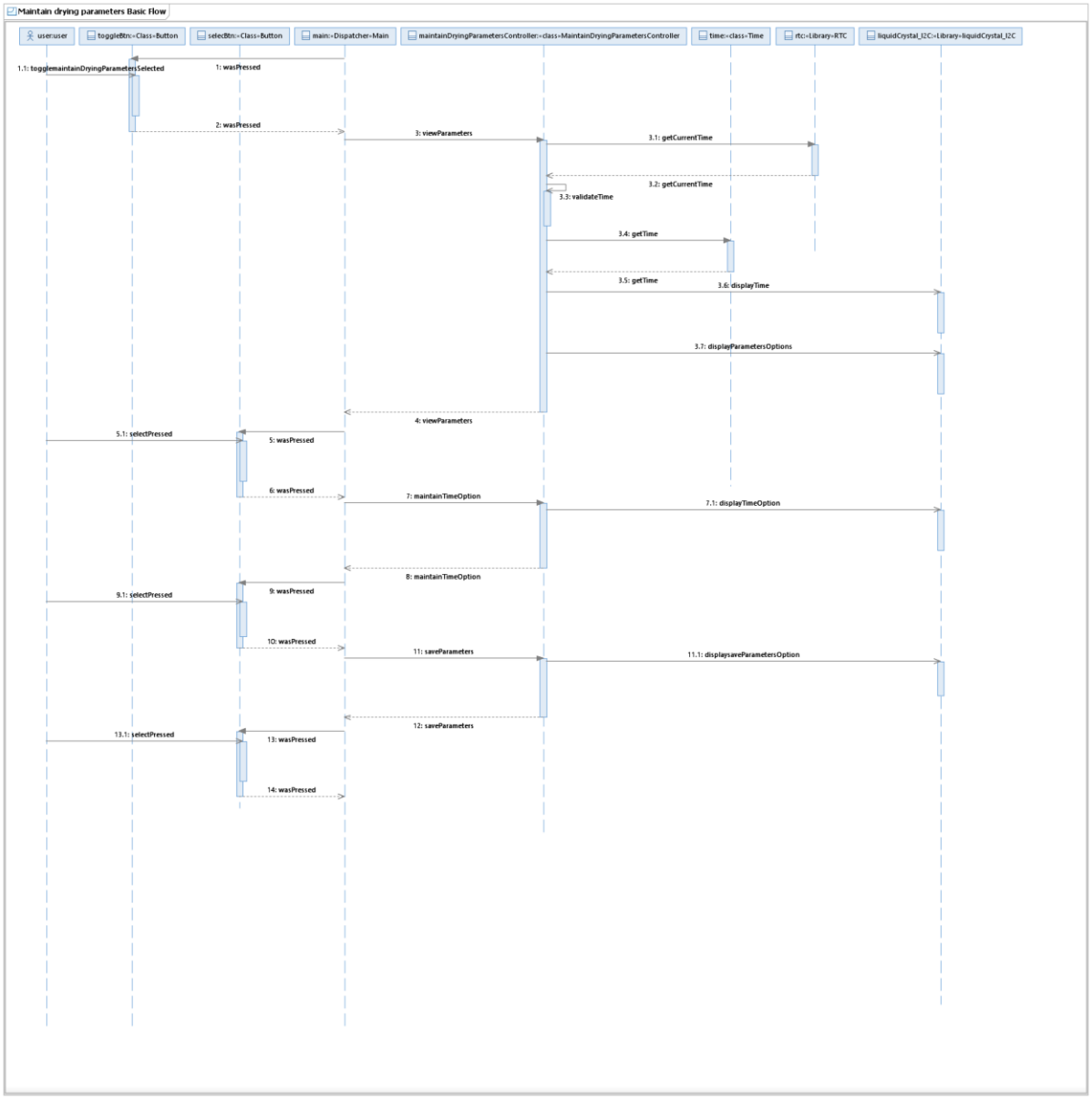

Figure 38 maintain Temp & Humidity design sequence diagram

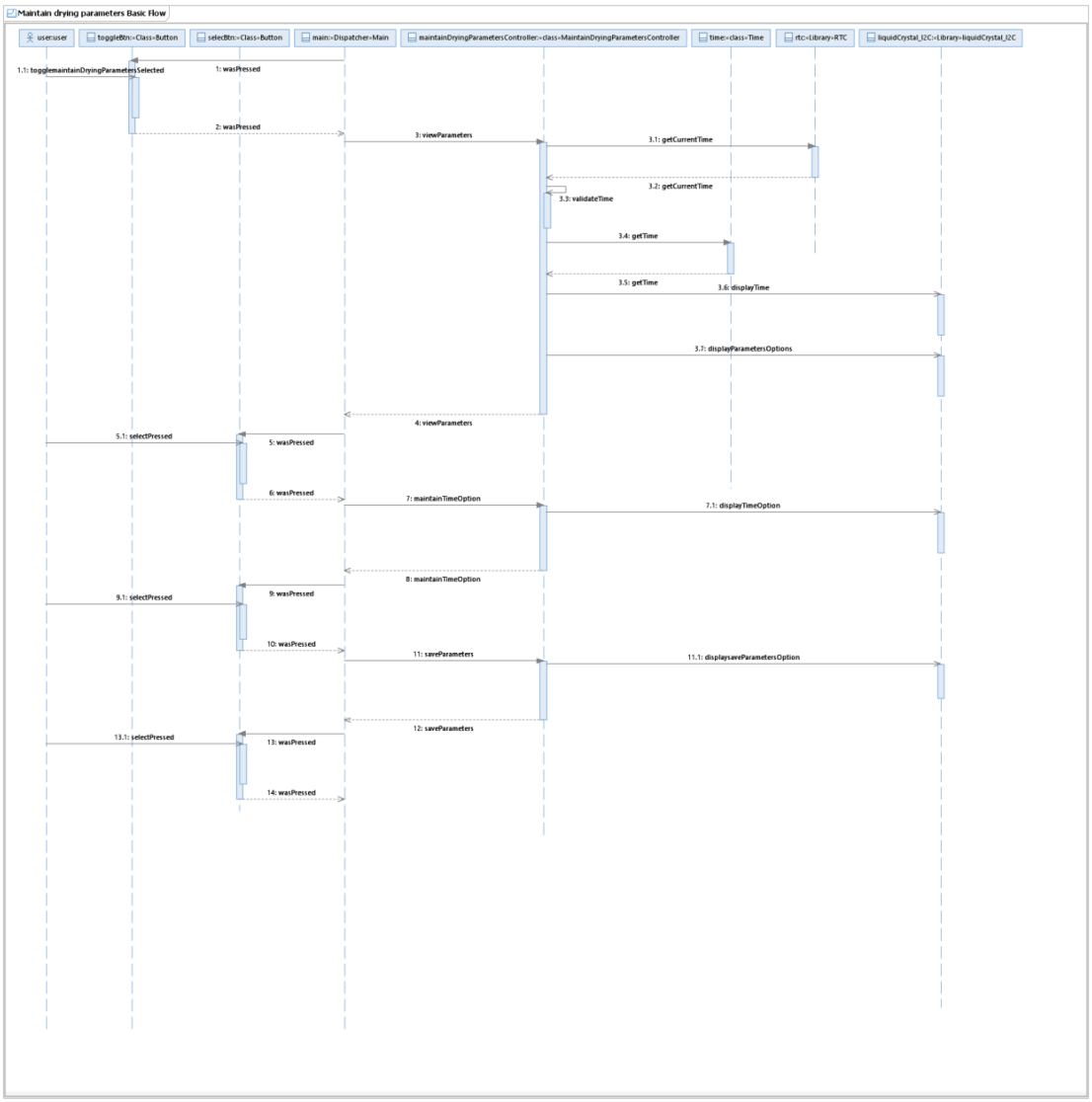

Figure 39 maintain drying parameters design sequence diagram

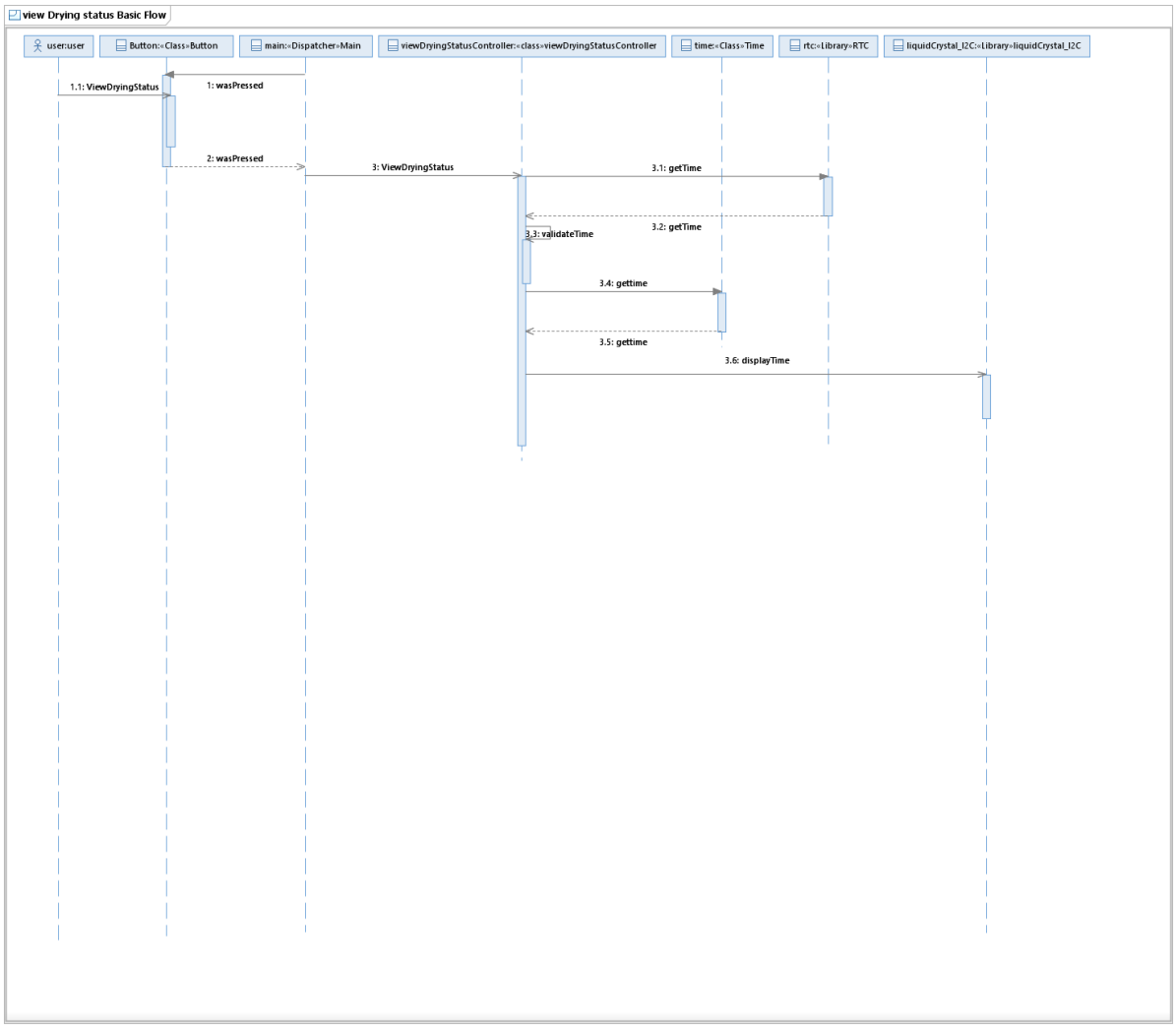

Figure 40 view drying status design sequence diagram

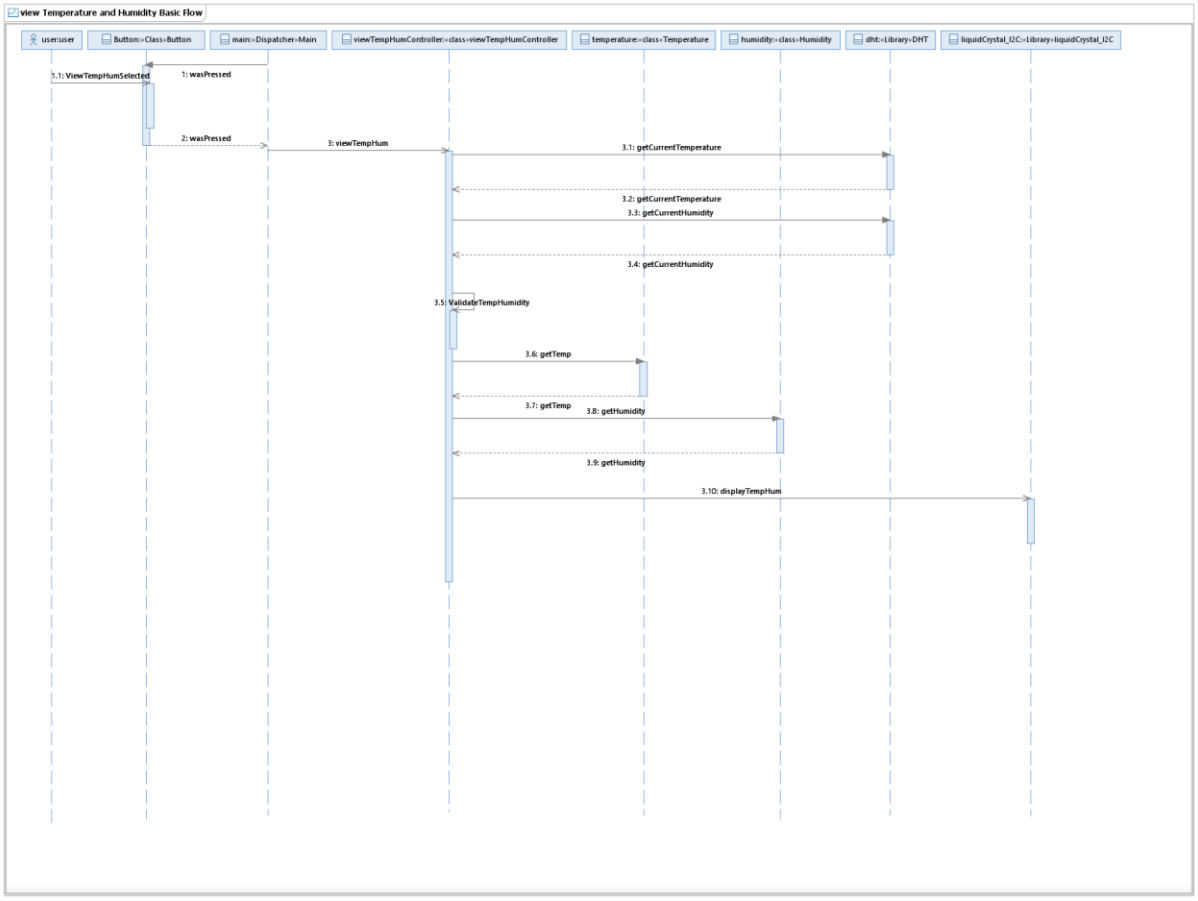

Figure 41 view Temp & humidity design level sequence diagram

4.5 Design class diagrams

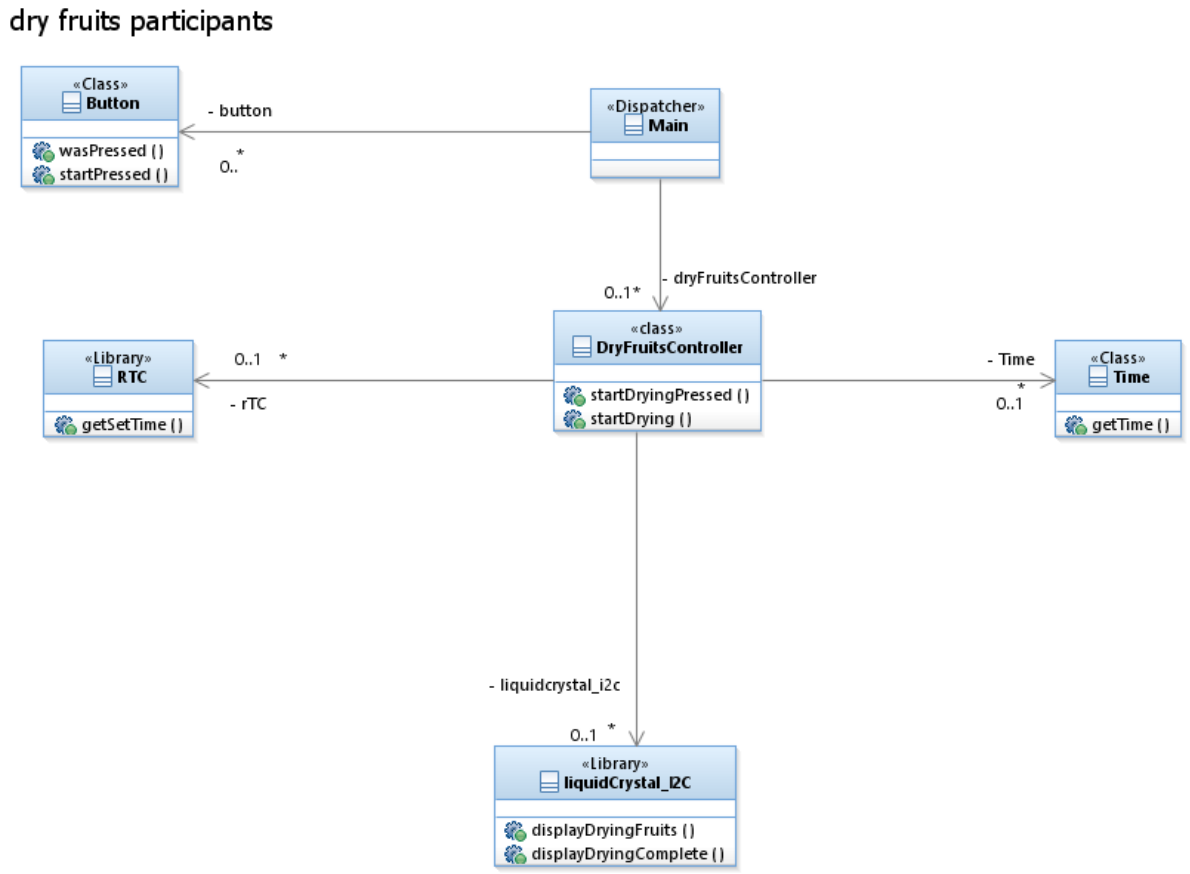

Figure 42 dry fruits class diagram

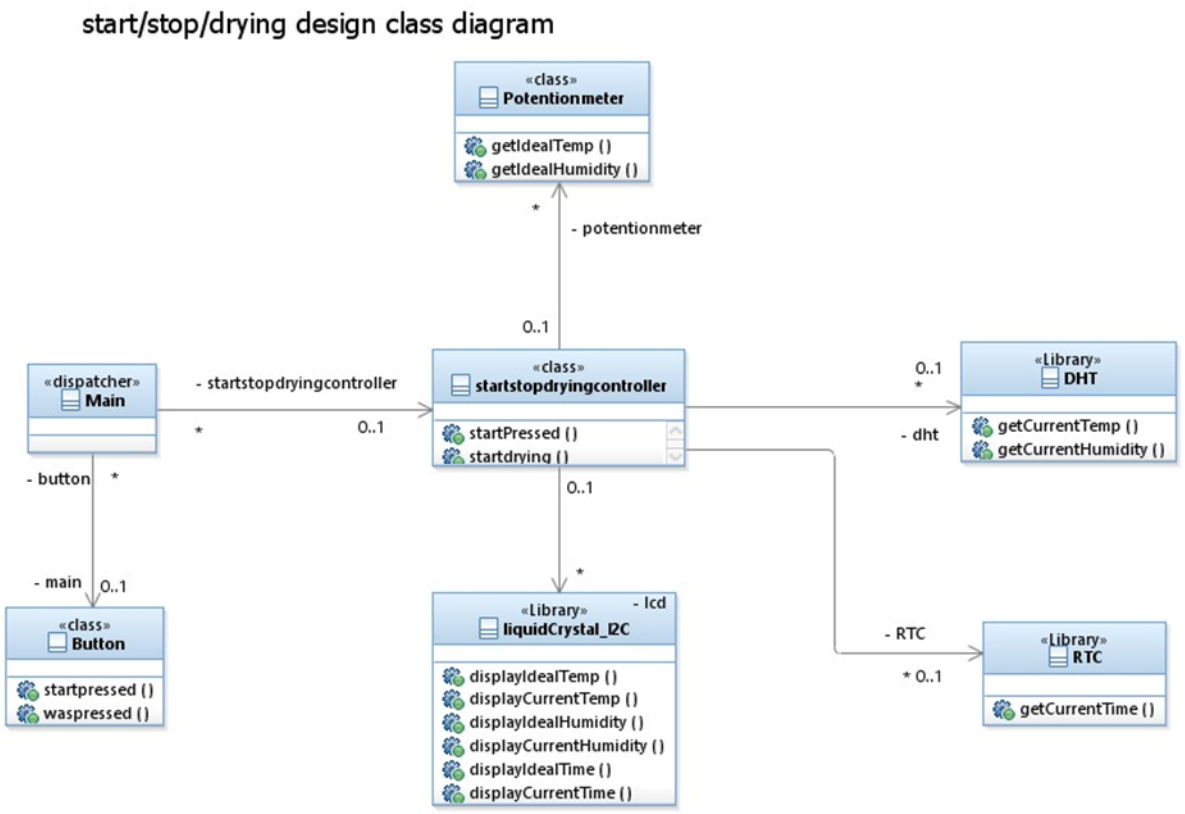

Figure 43 start/stop drying design class diagram

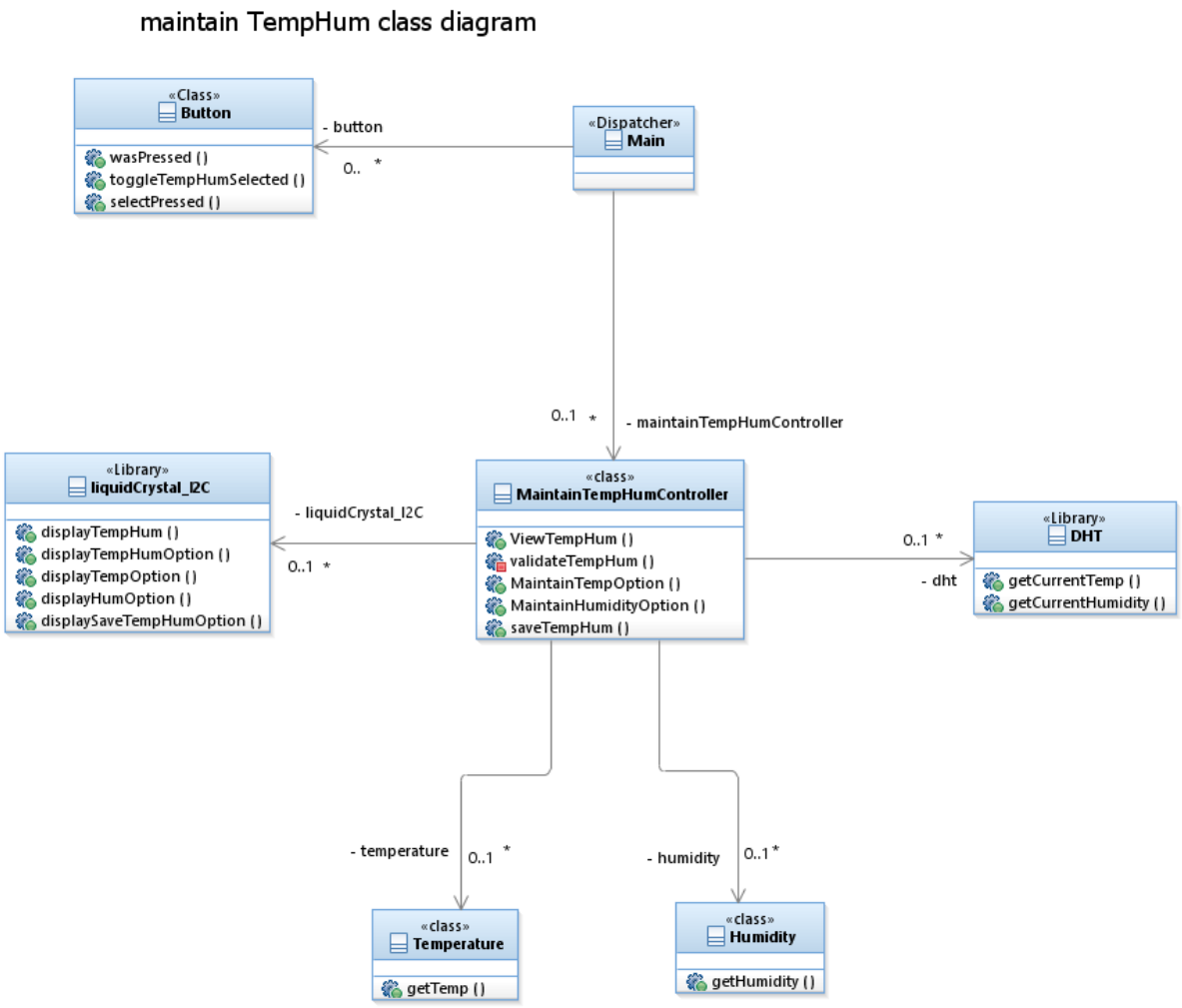

Figure 44 Maintain temp and humidity design class diagram

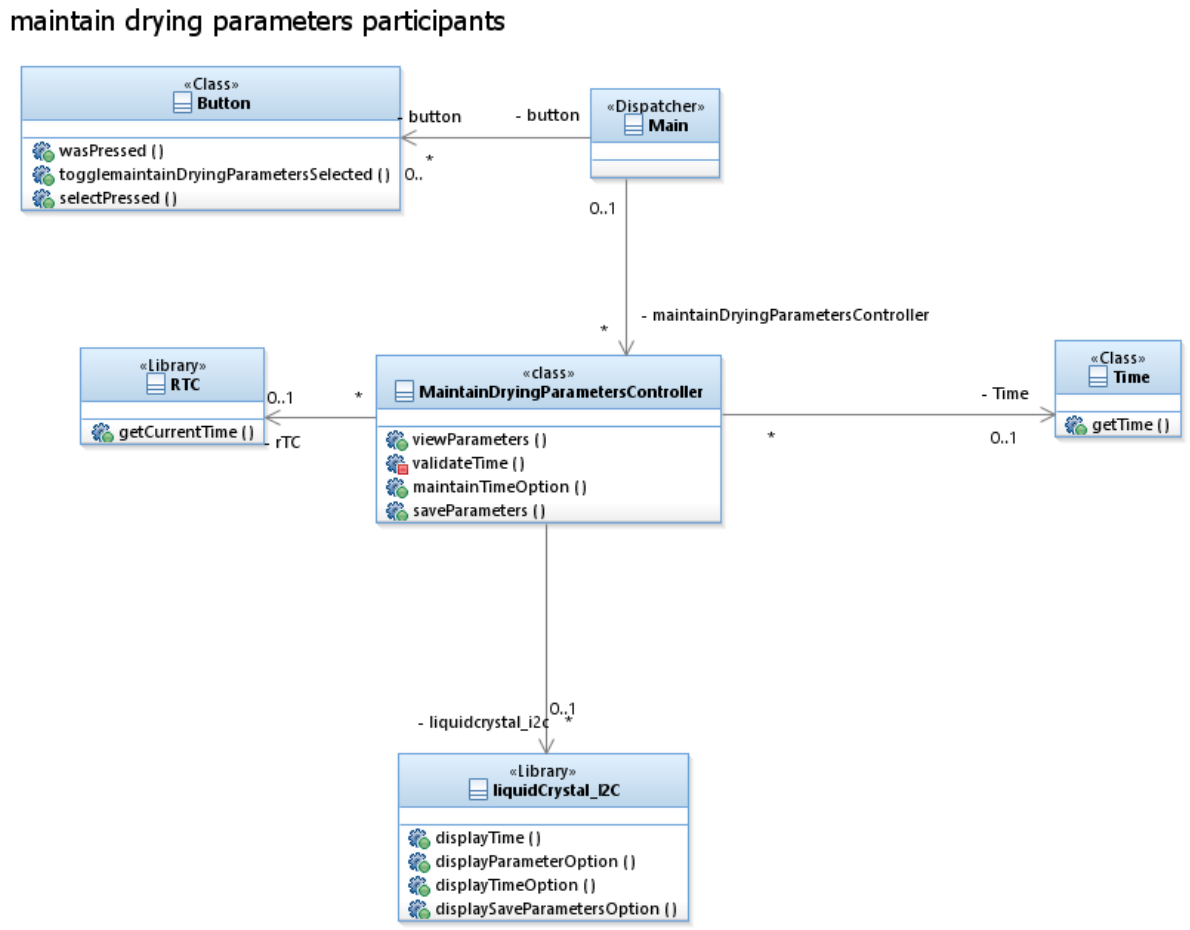

Figure 45 maintain drying parameters class diagram

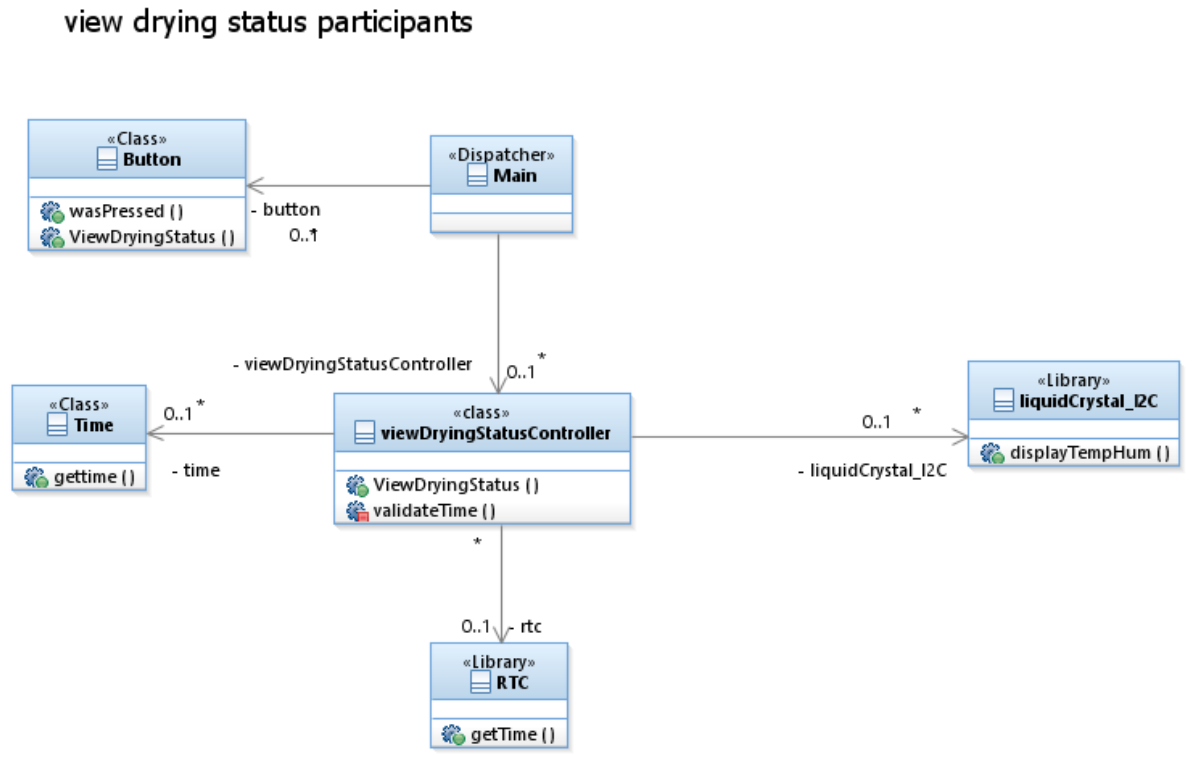

Figure 46 view drying status design class diagram

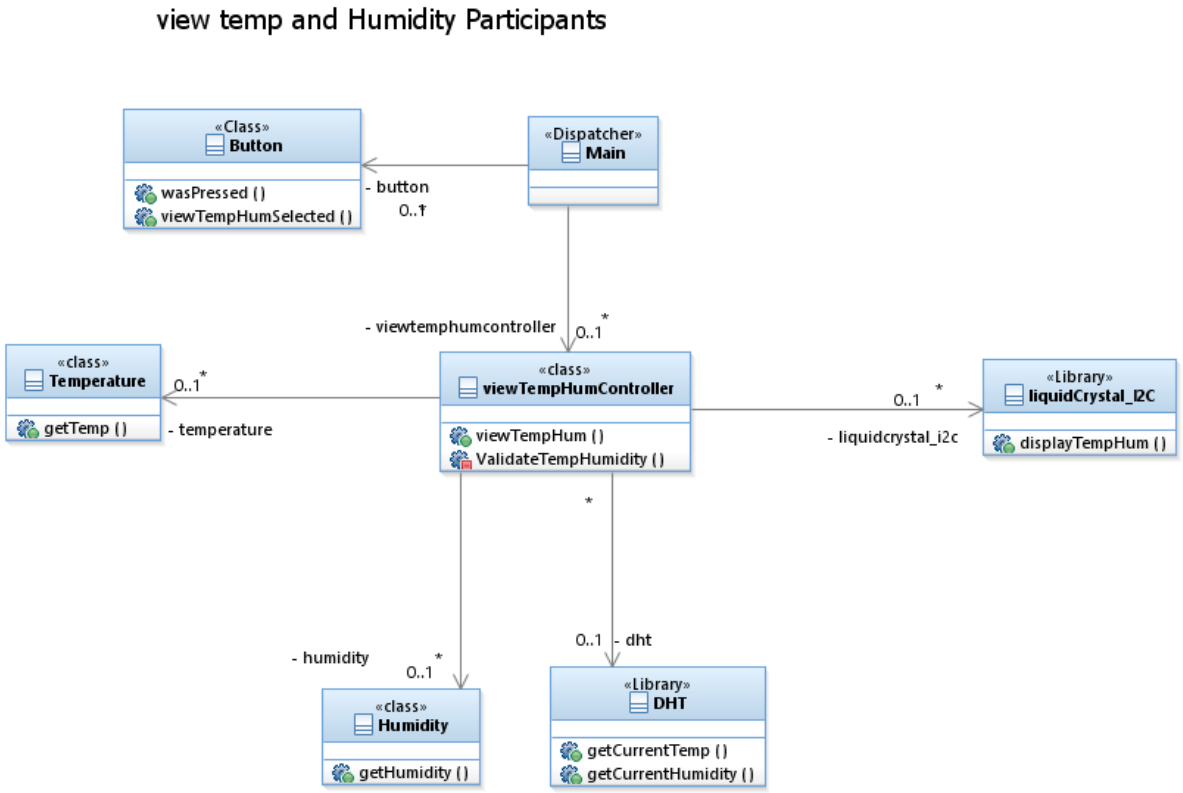

Figure 47 view Temp & Humidity design level class diagram

4.6 Summary

This chapter focused on the Design model related to the smart fruit drying system. The next chapter will focus on the implementation model.

Chapter 5 Implementation model

5.1 Introduction

This chapter will expand on the sub-problems of the smart fruit drying system. The Implementation model related to all sub-problem is provided.

5.2 Software artifacts

```
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
DHT;
#define I2C_ADDR    0x27
#define LCD_COLUMNS 20
#define LCD_LINES   4
#define DHT22_PIN 2
#define DHTTYPE DHT22
int run;
int buttonPin;



DHT dht(DHT22_PIN, DHTTYPE);

LiquidCrystal_I2C lcd(I2C_ADDR, LCD_COLUMNS, LCD_LINES);


class Temperature
{
    int idealtemp=30;
     public:
  int getTemp() {
      return idealtemp;
    }
  void setTemperature(int t) {
    idealtemp = t;
    }

};

class Humidity {
    int idealHum = 50;
  public:
  int getHumidity() {
      return idealHum;
    }

  void setHumidity(int h){
    idealHum = h;
```

```
    }
};
class startStopControl {
    String status;
    int tempOptions[9] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
    int humOptions[9] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
    int defualtHum = 3;
    int defualtTemp = 3;
    Temperature temp;
    Humidity hum;
    float currentTemp, currentHum;
    float idealtemp, idealHum;

    public:
    void setStatus(String s){
      status = s;
    }

    String getStatus() {
      return status;
    }

        int getTemp(){
      return tempOptions[defualtTemp];
    }

    int getHum(){
      return humOptions[defualtHum];
    }
      bool viewTempHum() {
      setStatus("viewTempHum");

      currentTemp = dht.readTemperature();
      currentHum = dht.readHumidity();

      if(validate(currentTemp, currentHum)) {
        idealtemp = temp.getTemp();
        idealHum = hum.getHumidity();

        displayTempHum(idealtemp, idealHum, currentTemp, currentHum);
        return true;
      }
      else {
        return false;
        }
    }

    bool validate(float t, float h) {
```

```
      if (isnan(t) || isnan(h)) {
        Serial.println(F("Failed to read from DHT sensor!"));
        return false;
      }
      else return true;
    }

    void displayTempHum(float st, float sh, float ct, float ch) {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Temp: " + String(ct) + "/("+ String(st) + ")");
      lcd.setCursor(0,1);
      lcd.print("Hum:  " + String(ch) + "/("+ String(sh) + ")");
    }
};
  class Button {
    const byte buttonPin;
    static constexpr byte debounceDelay = 30;
    const bool active;
    bool lastButtonState = HIGH;
    byte lastDebounceTime = 0;
    public:


    Button(byte attachTo, bool active = LOW) : buttonPin(attachTo),
active(active) {}

      void begin() {
      if (active == LOW)
        pinMode(buttonPin, INPUT_PULLUP);
      else
        pinMode(buttonPin, INPUT);
    }

        bool wasPressed() {
      bool buttonState = LOW;                                    // the
current reading from the input pin
      byte reading = LOW;                                        //
"translated" state of button LOW = released, HIGH = pressed, despite the
electrical state of the input pint
      if (digitalRead(buttonPin) == active) reading = HIGH;      // if we
are using INPUT_PULLUP we are checking invers to LOW Pin
      if (((millis() & 0xFF ) - lastDebounceTime) > debounceDelay)   // If the
switch changed, AFTER any pressing or noise
      {
        if (reading != lastButtonState && lastButtonState == LOW)    // If
there was a change and and last state was LOW (= released)
        {
          buttonState = HIGH;
```

```
        }
        lastDebounceTime = millis() & 0xFF;
        lastButtonState = reading;
      }
      return buttonState;
    }

};
String status;
Button startBtn{A0};
Button stopBtn{A3};
startStopControl startStopControl;

void setup() {


  Serial.begin(115200);
  startBtn.begin();
  stopBtn.begin();

  dht.begin();


}


void loop() {
   status = startStopControl.getStatus();

  if (startBtn.wasPressed()) {
     lcd.init();  // initialize the lcd

  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("start pressed");
  lcd.setCursor(0,1);
  lcd.print("smart fruit drying");
  delay(2500);
  Serial.println(F("start"));

  if(startStopControl.viewTempHum()) {

      }
    }
  else if (stopBtn.wasPressed()) {
  lcd.clear();

  lcd.setCursor(0,0);
  lcd.print("system stopped");
```

```
  delay(2500);
  lcd.clear();
  delay(2500);
  lcd.noBacklight();
  Serial.println(F("stop"));}

}
```

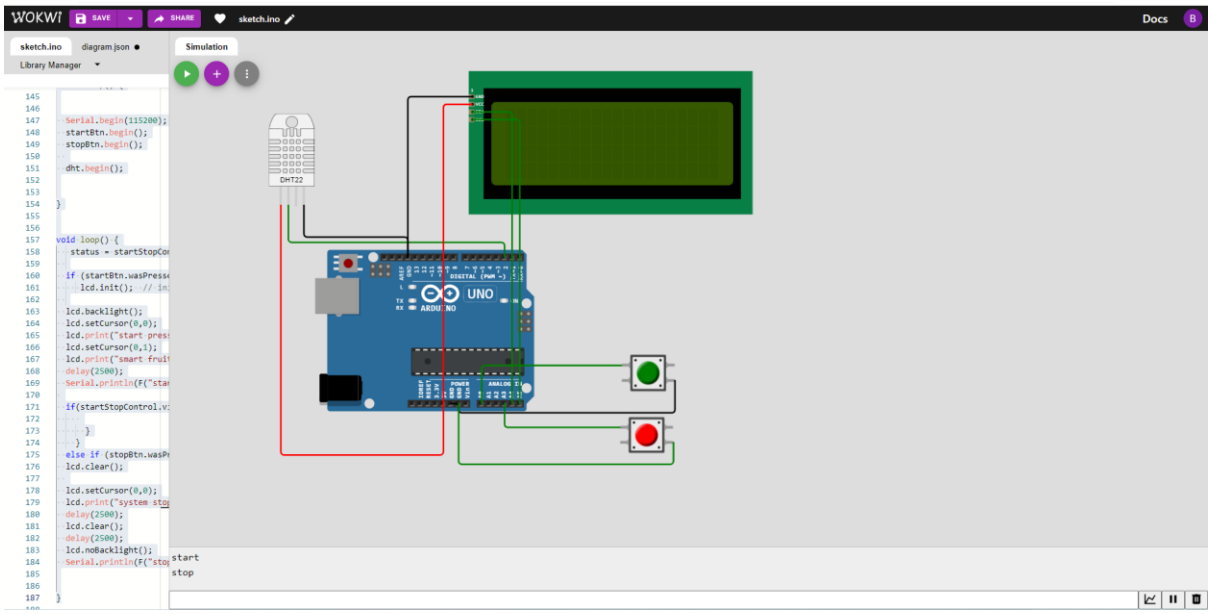

Figure 48 start/stop drying wokwi diagram

```
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
DHT;
#define I2C_ADDR    0x27
#define LCD_COLUMNS 20
#define LCD_LINES   4
#define DHT22_PIN 2
#define DHTTYPE DHT22
int run;
int buttonPin;



DHT dht(DHT22_PIN, DHTTYPE);

LiquidCrystal_I2C lcd(I2C_ADDR, LCD_COLUMNS, LCD_LINES);


class Temperature
{
    int idealtemp=30;
     public:
  int getTemp() {
```

```
      return idealtemp;
    }
  void setTemperature(int t) {
    idealtemp = t;
    }

};

class Humidity {
    int idealHum = 50;
  public:
  int getHumidity() {
      return idealHum;
    }

  void setHumidity(int h){
    idealHum = h;
    }
};
class viewTempHumControl {
    String status;
    int tempOptions[9] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
    int humOptions[9] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
    int defualtHum = 3;
    int defualtTemp = 3;
    Temperature temp;
    Humidity hum;
    float currentTemp, currentHum;
    float idealtemp, idealHum;

    public:
    void setStatus(String s){
      status = s;
    }

    String getStatus() {
      return status;
    }

        int getTemp(){
      return tempOptions[defualtTemp];
    }

    int getHum(){
      return humOptions[defualtHum];
    }
      bool viewTempHum() {
      setStatus("viewTempHum");
```

```
      currentTemp = dht.readTemperature();
      currentHum = dht.readHumidity();

      if(validate(currentTemp, currentHum)) {
        idealtemp = temp.getTemp();
        idealHum = hum.getHumidity();

        displayTempHum(idealtemp, idealHum, currentTemp, currentHum);
        return true;
      }
      else {
        return false;
        }
    }

    bool validate(float t, float h) {
      if (isnan(t) || isnan(h)) {
        Serial.println(F("Failed to read from DHT sensor!"));
        return false;
      }
      else return true;
    }

    void displayTempHum(float st, float sh, float ct, float ch) {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Temp: " + String(ct) + "/("+ String(st) + ")");
      lcd.setCursor(0,1);
      lcd.print("Hum:  " + String(ch) + "/("+ String(sh) + ")");
    }
};
  class Button {
    const byte buttonPin;
    static constexpr byte debounceDelay = 30;
    const bool active;
    bool lastButtonState = HIGH;
    byte lastDebounceTime = 0;
    public:


    Button(byte attachTo, bool active = LOW) : buttonPin(attachTo),
active(active) {}

      void begin() {
      if (active == LOW)
        pinMode(buttonPin, INPUT_PULLUP);
      else
        pinMode(buttonPin, INPUT);
```

```
    }

        bool wasPressed() {
      bool buttonState = LOW;                                    // the
current reading from the input pin
      byte reading = LOW;                                        //
"translated" state of button LOW = released, HIGH = pressed, despite the
electrical state of the input pint
      if (digitalRead(buttonPin) == active) reading = HIGH;      // if we
are using INPUT_PULLUP we are checking invers to LOW Pin
      if (((millis() & 0xFF ) - lastDebounceTime) > debounceDelay)   // If the
switch changed, AFTER any pressing or noise
      {
        if (reading != lastButtonState && lastButtonState == LOW)    // If
there was a change and and last state was LOW (= released)
        {
          buttonState = HIGH;
        }
        lastDebounceTime = millis() & 0xFF;
        lastButtonState = reading;
      }
      return buttonState;
    }

};
String status;
Button viewTempHum{A0};
viewTempHumControl viewTempHumControl;

void setup() {


  Serial.begin(115200);
  viewTempHum.begin();
  lcd.init();
  lcd.backlight();
  lcd.print("Drying");


  dht.begin();


}


void loop() {
   status = viewTempHumControl.getStatus();
  if (viewTempHum.wasPressed()) {
      // initialize the lcd
```

```
  lcd.clear();
  lcd.print("view Temp & HUmidity");
  lcd.setCursor(0,1);
  delay(2500);
  Serial.println(F("view temp"));
  if(viewTempHumControl.viewTempHum()) {

      }
    }


}
```

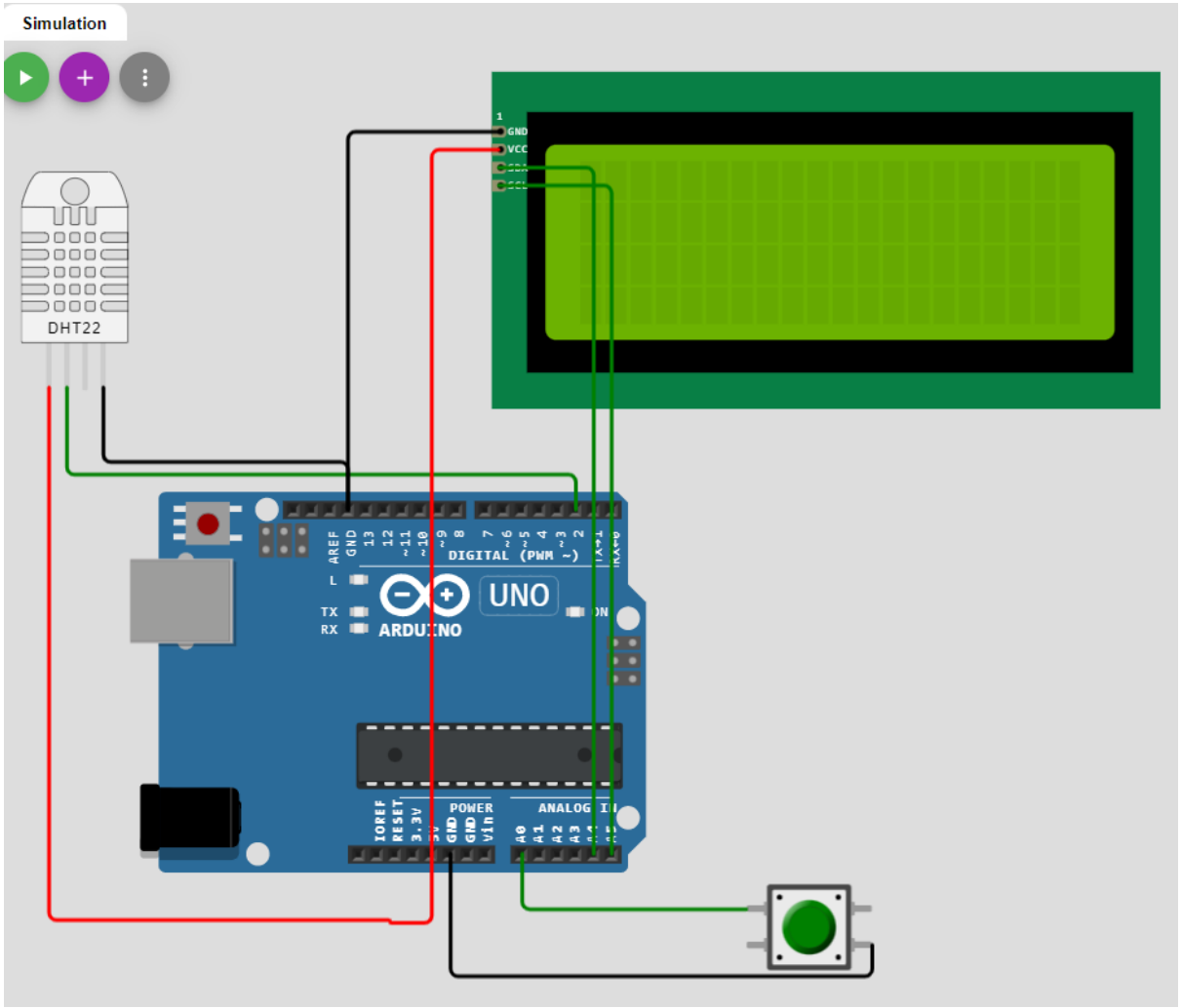

Figure 49 view temp Hum

```
#include "DHT.h"
#include <LiquidCrystal_I2C.h>
#define DHTPIN 2

#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
DHT dht(DHTPIN, DHTTYPE);
```

```
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16
chars and 2 line display

// Temperature class with a ideal temperature of 30C
class Temperature {
    int storedTemp = 30;
  public:
  int getTemperature() {
      return storedTemp;
    }
  void setTemperature(int t) {
    storedTemp = t;
    }
};

// Humidity class with a ideal humidity of 50%
class Humidity {
    int storedHum = 50;
  public:
  int getHumidity() {
      return storedHum;
    }

  void setHumidity(int h){
    storedHum = h;
    }
};

//This is the MaintainTempHumControl class
//It contains the logic related to the use case
class MaintainTempHumControl {

    String status;
    int tempOptions[9] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
    int humOptions[9] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
    int defualtHum = 3;
    int defualtTemp = 3;
    Temperature temp;
    Humidity hum;
    float currentTemp, currentHum;
    float storedTemp, storedHum;

  public:
    void setStatus(String s){
      status = s;
    }
```

```
String getStatus() {
  return status;
}

void increaseTemp(){
  if (defualtTemp < 9) {
    defualtTemp++;
  } else {
    defualtTemp = 1;
    }
  displayTempOption();
}

void decreaseTemp(){
  if (defualtTemp < 9) {
    defualtTemp--;
  } else {
    defualtTemp = 1;
    }
  displayTempOption();
}

void increaseHum(){
  if (defualtHum < 9) {
    defualtHum++;
  } else {
    defualtHum = 1;
    }
  displayHumOption();
}

void decreaseHum(){
  if (defualtHum < 9) {
    defualtHum--;
  } else {
    defualtHum = 1;
    }
  displayHumOption();
}

int getTemp(){
  return tempOptions[defualtTemp];
}

int getHum(){
  return humOptions[defualtHum];
}
```

```
bool viewTempHum() {
  setStatus("viewTempHum");

  currentTemp = dht.readTemperature();
  currentHum = dht.readHumidity();

  if(validate(currentTemp, currentHum)) {
    storedTemp = temp.getTemperature();
    storedHum = hum.getHumidity();

    displayTempHum(storedTemp, storedHum, currentTemp, currentHum);
    displayTempHumOption();
    return true;
  }
  else {
    return false;
    }
}

bool validate(float t, float h) {
  if (isnan(t) || isnan(h)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return false;
  }
  else return true;
}

void displayTempHum(float st, float sh, float ct, float ch) {
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp: " + String(ct) + "/("+ String(st) + ")");
  lcd.setCursor(0,1);
  lcd.print("Hum:  " + String(ch) + "/("+ String(sh) + ")");
}

void displayTempHumOption(){
  lcd.setCursor(0,2);
  lcd.print("Change: Y/N");
}

//function to maintain the temperature
bool maintainTempOption(){
  setStatus("maintainTemp");
  displayTempOption();
  return true;
}

//function to display temperature options to the user
```

```
    void displayTempOption(){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Maintain temperature");
      lcd.setCursor(0,1);
      lcd.print("Temp option:" + String(getTemp()));
    }

    //function to maintain the humidity
    bool maintainHumOption(){
      setStatus("maintainHum");
      displayHumOption();
      return true;
    }

    //function to display humidity options to the user
    void displayHumOption(){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Maintain humidity");
      lcd.setCursor(0,1);
      lcd.print("Hum option:" + String(getHum()));
    }

    //function to set the temperature and humitidy
    bool saveTempHum(){
      setStatus("saveTempHum");
      displaySaveOption();
      temp.setTemperature(getTemp());
      hum.setHumidity(getHum());
      return true;
    }

    //function to ask the user if they want to save
    void displaySaveOption(){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Save Temp & Hum: Y/N");
    }

};

class Button {                          // a simple class for buttons based on
the "Debounce" example
    const byte buttonPin;               // the GPIO / pin for the button
    static constexpr byte debounceDelay = 30;    // the debounce time;
increase if the output flickers. Static because we only need one value for all
buttons
```

```
    const bool active;                // is the pin active HIGH or active LOW
(will also activate the pullups!)
    bool lastButtonState = HIGH;      // the previous reading from the input
pin
    byte lastDebounceTime = 0;        // the last time the output pin was
toggled - we check only ONE byte, so I didn't mess around with unsigned long

  public:
    /**
       \brief constructor for a button

       The constructor takes the GPIO as parameter.
       If you omit the second parameter, the library will activate the
internal pullup resistor
       and the button should connect to GND.
       If you set the second parameter to HIGH, the button is active HIGH.
       The button should connect to VCC.
       The internal pullups will not be used but you will need an external
pulldown resistor.

       \param attachTo the GPIO for the button
       \param active LOW (default) - if button connects to GND, HIGH if button
connects to VCC
    */
    Button(byte attachTo, bool active = LOW) : buttonPin(attachTo),
active(active) {}

    /**
       \brief set the pin to the proper state

       Call this function in your setup().
       The pinMode will be set according to your constructor.
    */
    void begin() {
      if (active == LOW)
        pinMode(buttonPin, INPUT_PULLUP);
      else
        pinMode(buttonPin, INPUT);
    }

    /**
        \brief indicate if button was pressed since last call

        @return HIGH if button was pressed since last call - debounce
    */
    bool wasPressed() {
      bool buttonState = LOW;                                    // the
current reading from the input pin
```

```
      byte reading = LOW;                                                  //
"translated" state of button LOW = released, HIGH = pressed, despite the
electrical state of the input pint
      if (digitalRead(buttonPin) == active) reading = HIGH;          // if we
are using INPUT_PULLUP we are checking invers to LOW Pin
      if (((millis() & 0xFF ) - lastDebounceTime) > debounceDelay)    // If the
switch changed, AFTER any pressing or noise
      {
        if (reading != lastButtonState && lastButtonState == LOW)    // If
there was a change and and last state was LOW (= released)
        {
          buttonState = HIGH;
        }
        lastDebounceTime = millis() & 0xFF;
        lastButtonState = reading;
      }
      return buttonState;
    }
};


String status;
Button toggleBtn{A0};
Button selectBtn{A1};
Button nextBtn{A2};
Button backBtn{A3};

MaintainTempHumControl maintainTempHumControl;


void setup()
{
  Serial.begin(115200);
  toggleBtn.begin();
  selectBtn.begin();
  nextBtn.begin();
  backBtn.begin();

  lcd.init();  // initialize the lcd
  // Print a message to the LCD.
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Fruit drying system");
  lcd.setCursor(0,1);
  lcd.print("Press toggle to");
  lcd.setCursor(0,2);
  lcd.print("continue");

  dht.begin();
```

```
}

void loop()
{
  status = maintainTempHumControl.getStatus();

  if (toggleBtn.wasPressed()) {
    Serial.println(F("Toggle"));
    if(maintainTempHumControl.viewTempHum()) {
      Serial.println(F("View Temp Hum"));
      }
    }
  if (selectBtn.wasPressed()) {
    Serial.println(F("Select"));
    if(status.equals("viewTempHum")){
      if(maintainTempHumControl.maintainTempOption()) {
        Serial.println(F("Maintain temp options"));
        }
        else {
          Serial.println(F("ERROR - Maintain temp option"));
        }
      }
    if(status.equals("maintainTemp")){
      if(maintainTempHumControl.maintainHumOption()) {
        Serial.println(F("Maintain humidity options"));
        }
        else {
          Serial.println(F("ERROR - Maintain hum option"));
        }
    }
    if(status.equals("maintainHum")){
        Serial.println(F("Save option"));
      if(maintainTempHumControl.saveTempHum()) {
        Serial.println(F("Save Temp & Hum"));
        if(maintainTempHumControl.viewTempHum()){
          Serial.println(F("Display temp & humidity again"));
        };
        }
        else {
          Serial.println(F("ERROR - Maintain saving"));
        }
    }
  }

  if (nextBtn.wasPressed()) {
    Serial.println(F("Next/Y"));
    if(status.equals("maintainTemp")){
```

```
      Serial.println(F("Maintain Temp - Increase"));
      maintainTempHumControl.increaseTemp();
      }
    if(status.equals("maintainHum")){
      Serial.println(F("Maintain Hum - Increase"));
      maintainTempHumControl.increaseHum();
      }
    }

  if (backBtn.wasPressed()) {
    Serial.println(F("Back/N"));
    if(status.equals("maintainTemp")){
      Serial.println(F("Maintain Temp - Decrease"));
      maintainTempHumControl.decreaseTemp();
      }

    if(status.equals("maintainHum")){
      Serial.println(F("Maintain Hum - Decrease"));
      maintainTempHumControl.decreaseHum();
      }
    }

}
```

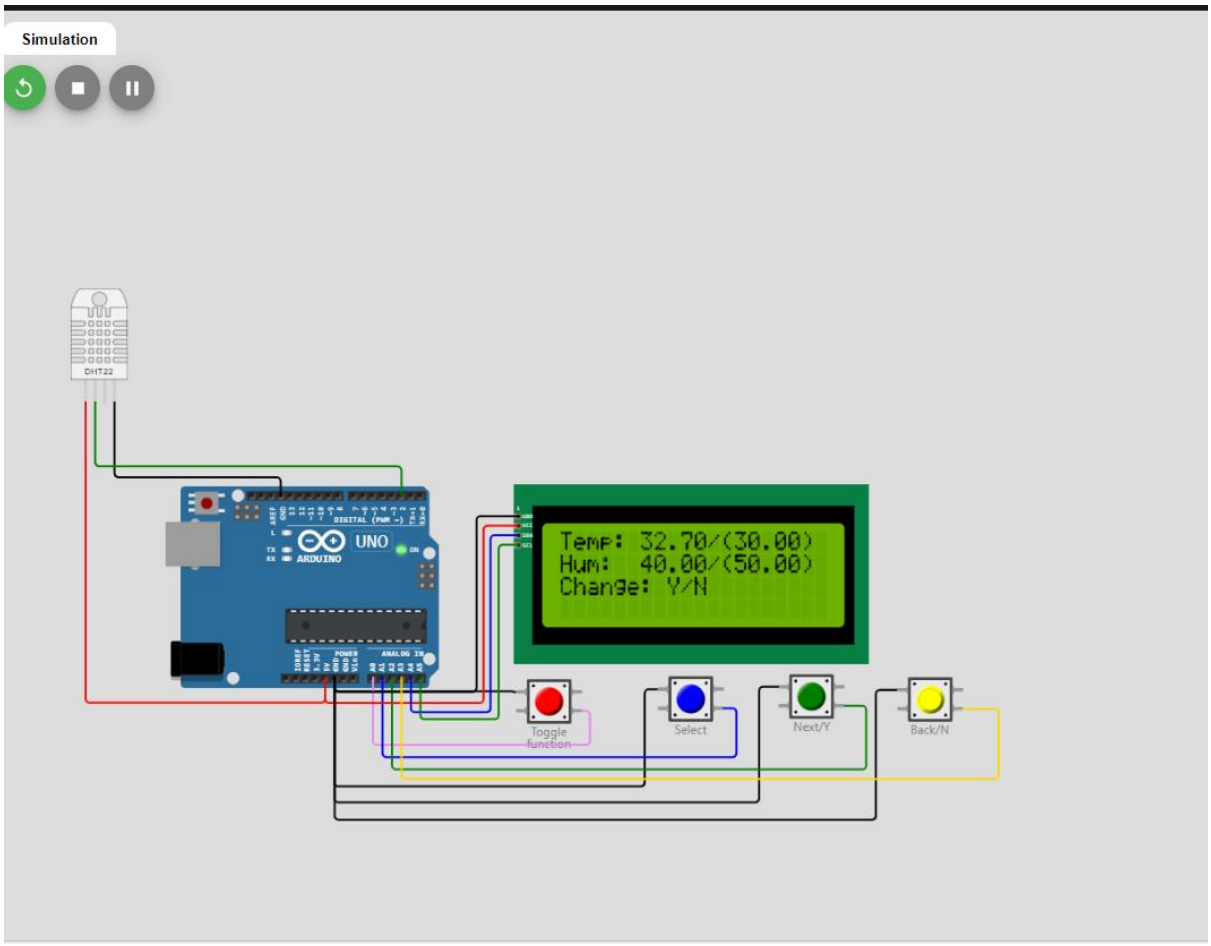

Figure 50 maintain Temp Hum

5.3 Summary

This chapter provided the implementation model codes written using wokwi for start/stop drying , view temp and humidity , and maintain temp and humidity sub-problems. The next chapter will focus on the results and conclusions of the project.

Chapter 6 Results & Conclusions

6.1 Introduction

This chapter will focus on the reassessment of original problems, recommendations which include limitations and suggestions and fields for further study.

6.2 Reassessment of original problem(s)

Varying Temperature and humidity causes inconsistent drying of fruits and this negatively affects not only the yield of dried fruits but also the quality of the dried fruits.

View drying status

This sub-problem allows the user to see the current drying status.

Dry fruits

This sub-problem performs the main work in the system of drying fruits. It stops when the system is stopped and starts when the system is started.

Start/stop/pause drying

This sub-problem allows the user to start, pause or stop the drying cycle.

View current temperature & humidity

This sub-problem allow the user to view the current temperature and humidity inside the smart fruit drying system.

Maintain drying parameters

This sub-problem allows the user to maintain drying parameters. This includes factors air circulation and the time for drying. The user should be able to set and adjust these elements.

Maintain drying temperature & humidity

This sub-problem allows the user to maintain the drying temperature & humidity, the system should include an interface where the user will be able to set and adjust the desired drying humidity and temperature.

6.3 Recommendations

This implementation platform limited the project's reach as some components like motors are not available.

RTC library provides the real time clock and made it difficult to implement a clock countdown on the lcd not the systems RTC

6.4 **Fields for further study**

The requirements, analysis, design and implementation models have all been provided for the smart fruit drying system. The project can be improved in the future by design of automatic mass detectors and fruit slicers, to decrease or ease human effort in productivity as much as possible.

6.5 **Summary**

This chapter focused on the reassessment of original problems the limitations of the project and fields of further for further study.

Bibliography

Anon., n.d. [Online].

Bjarnadottir, A., 2017. *Healthline.* [Online] Available at: https://www.healthline.com/nutrition/dried-fruit-good-or-bad#TOC_TITLE_HDR_2 [Accessed 15 august 2022].

Blog, L. S., n.d. *Research Methods.* [Online] Available at: https://lovestats.wordpress.com/dman/ [Accessed 4 Oct 2018].

Editage Insights, 2016. *How do I find reliable scientific literature on the Internet?.* [Online] Available at: https://www.editage.com/insights/how-do-i-find-reliable-scientific-literature-on-the-internet [Accessed 28 July 2018].

Editage Insights, 2017. *What is the best way of stating the background of a study?.* [Online] Available at: https://www.editage.com/insights/what-is-the-best-way-of-stating-the-background-of-a-study [Accessed 3 August 2018].

Johncheff, K., 2022. *commercialdehydrators.* [Online] Available at: https://www.commercialdehydrators.co.uk/post/best-practice-for-drying-fruit-and-vegetables-commercial-dehydrators [Accessed 16 august 2022].

Popeck, L., 2020. *orlando health.* [Online] Available at: https://www.orlandohealth.com/content-hub/top-10-reasons-to-eat-more-fruits-and-vegetables [Accessed 12 august 2022].

WHO, 2022. *world health organization.* [Online] Available at: https://www.who.int/news-room/fact-sheets/detail/food-safety [Accessed 16 august 2022].

ya su, m. z. A. S. m., 2014. recent developments in smart technology. *drying technology,* 33(3), p. 276.

Appendices

8.1 Appendix A

Start each appendix on a new page. Place appendices in the same order as they are referred to in the body of the thesis. That is, the first appendix referred to should be Appendix A, the second appendix referred to should be Appendix B, and so on. Appendix formatting can be different to the main document.