# RePitile System

Design Document

Alex Bioni Seth Law Wesley Norris Scott Sheppard

California University of Pennsylvania
CSC 490 – Senior Project
Due: 12/06/17

# Instructor Comments/Evaluation

# Table of Contents


Instructor Comments/Evaluation ..... 1

Abstract ..... 3

Description of this Document ..... 4

I. Purpose and Use ..... 4

II. Ties to Specification Document ..... 4

III. Intended Audience ..... 4

Project Block Diagram ..... 5

Design Details ..... 6

I. System Modules and Responsibilities ..... 6

a. Architectural Diagram ..... 6

II. Design Analysis ..... 7

a. Data Flow / Transaction Analysis ..... 7

III. Design Organization ..... 8

a. Detailed Tabular Description of Classes / Objects ..... 8

b. Module Cohesion ..... 13

c. Module Coupling ..... 14

IV. Functional Description ..... 15

a. Input / Output / Return Parameters / Types ..... 15

b. Modules Used ..... 23

c. Files Accessed ..... 23

d. Real-Time Requirements ..... 23

V. Reuse / Portability ..... 23

VI. Design Workflow CASE Diagrams ..... 24

VII. Design Testing ..... 24

Appendix A: Schematic & Bill of Materials ..... 25

Appendix B: References ..... 27

Appendix C: Team Details ..... 28

A. Alex Bioni ..... 28

B. Seth Law ..... 28

C. Scott Sheppard ..... 28

D. Wesley Norris ..... 28

Appendix D: Workflow Authentication ..... 29

# Abstract

The objective of this document is to layout the design for the RePitile, a modular, automatic reptile tank care system. Design analysis and documentation for the organization of the various modules in the RePitile system is described here. The Design Details section contains information about each of the specific modules and what is contained within them, including items such as data flow analysis, cohesion and coupling levels for each module, and how each of the modules use each other (if any). This document also provides testing for the design modules to ensure that they meet the requirements to create a functional system.

# Description of this Document

## I. Purpose and Use

This software design document will provide in great detail the implementation process for the Repitile System. This will provide insight into the structure and design of each component while providing the details of how the software will be built. In this document, graphical documentation will be provided. We will see how all the components interact with each other. In summary, the purpose of this document is to provide the reader with a strong understanding of how the Repitile System will be implemented.

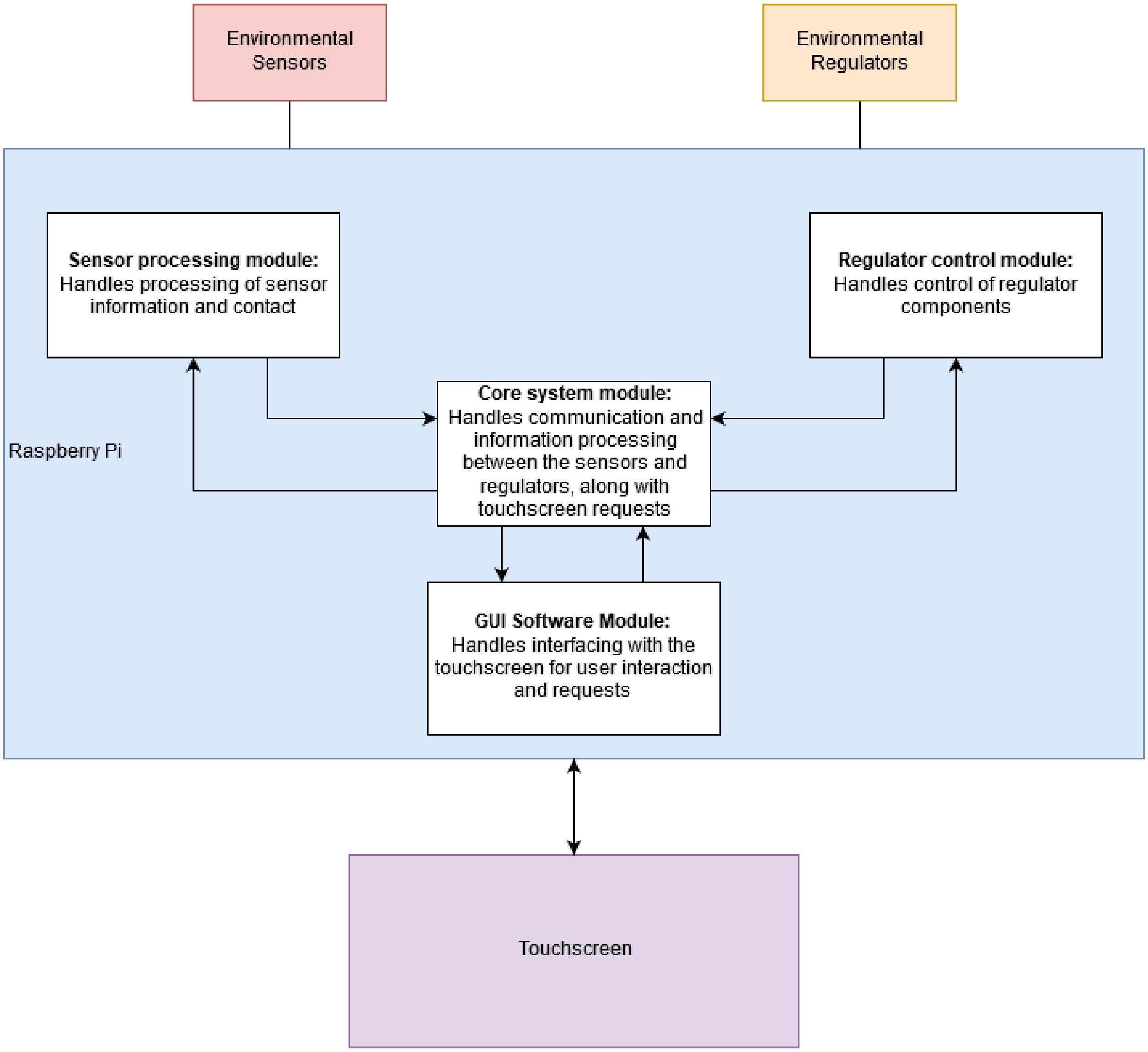
## II. Ties to Specification Document

This design document provides clarification and more concrete information of the classes provided in the specification document. This document also takes relevant goals and requirements from the specification document and improves or adds upon in this document.

## III. Intended Audience

This document has been written for hardware and software engineers implementing the system designs. It provides the information to integrate the software and hardware components together to create a working system. The hardware that is required and constraints on how the software is built to control the system is also detailed. This goes into what each of the classes is responsible for and how it communicates with the rest of the modules.

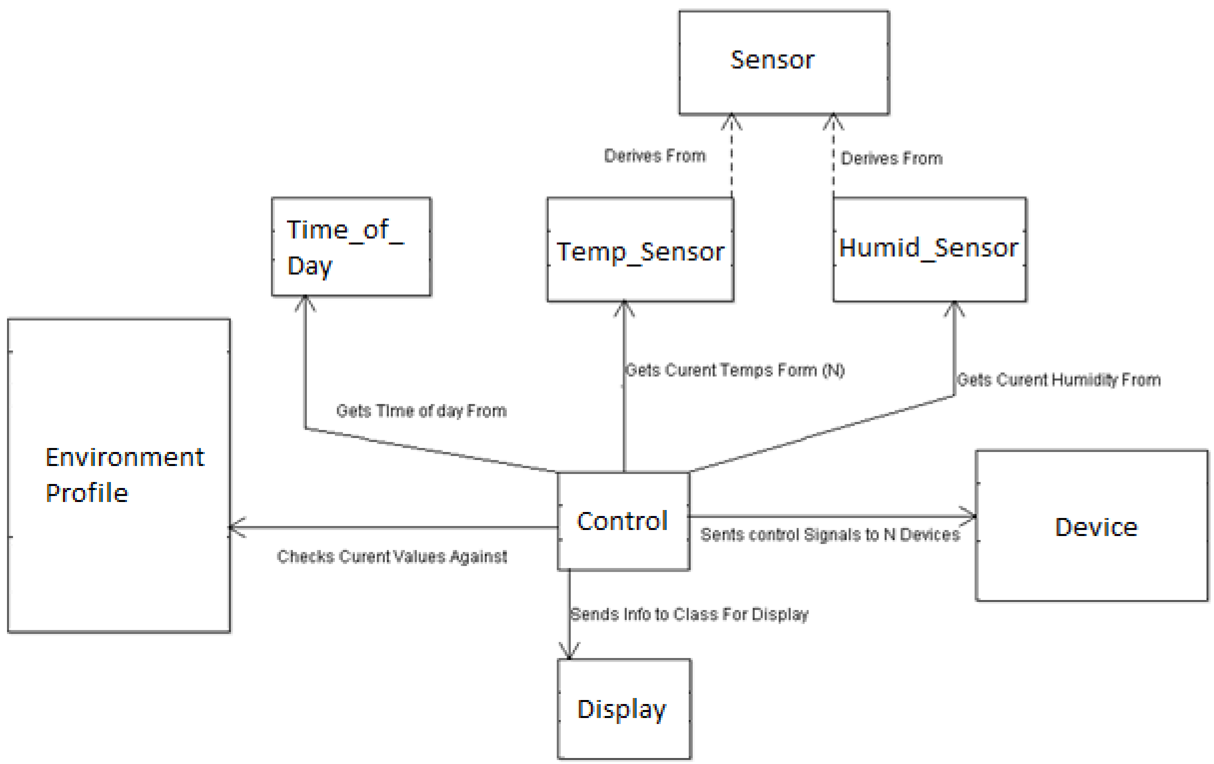
# Project Block Diagram



# Design Details

## I. System Modules and Responsibilities

### a. Architectural Diagram




In our architectural diagram, the control module serves as a go-between for each of the other devices that require knowledge about other devices. Each of the other classes manages the particular device(s) that it is named after.

# II. Design Analysis

## a. Data Flow / Transaction Analysis

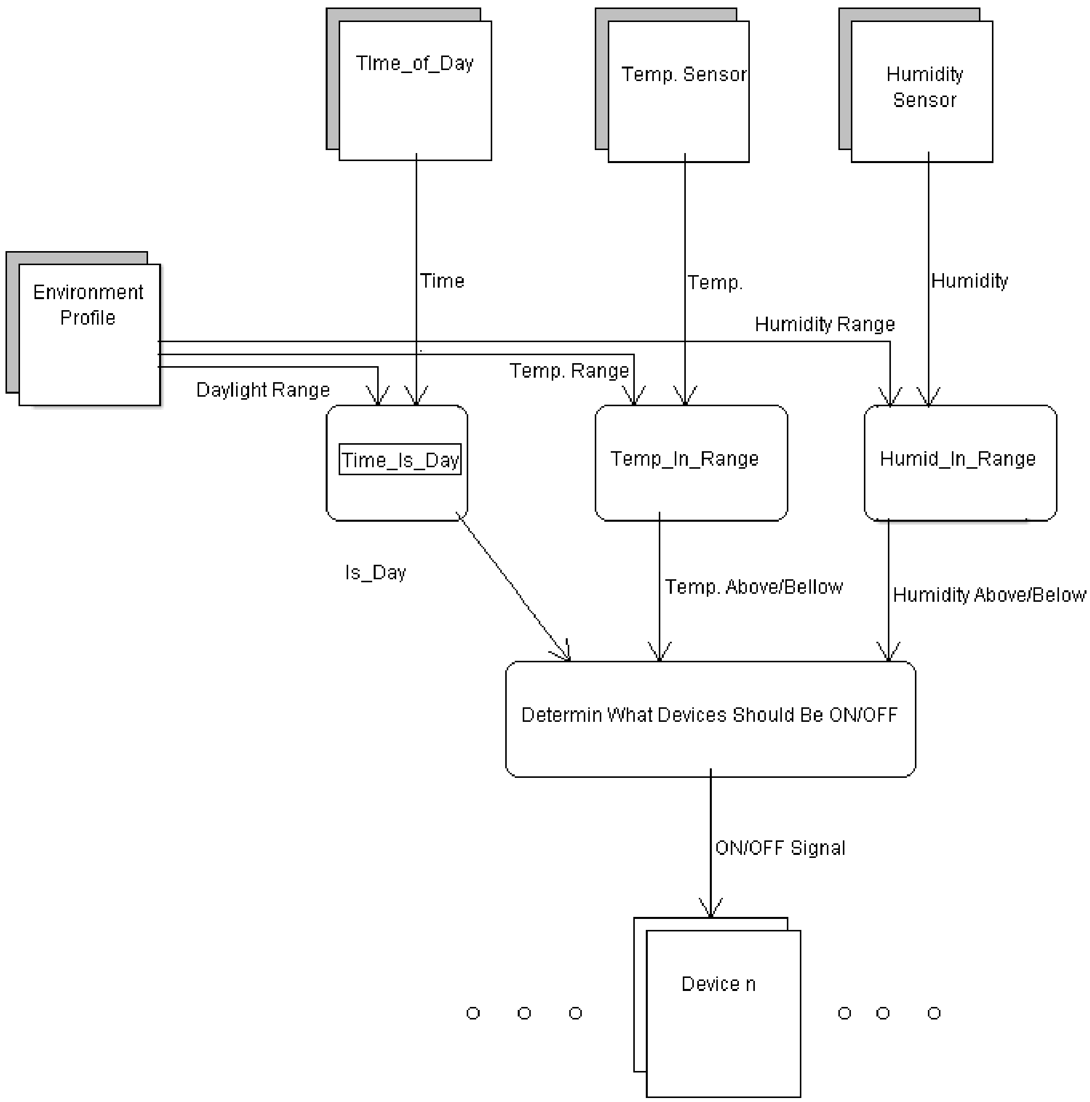



*Figure 1: Data Flow Analysis (Bioni, Law and Norris)*

This diagram illustrates how the data is taken from the various sensors and interior data flows through the system to turning on and off the different components that will be in the system.

# III. Design Organization

## a. Detailed Tabular Description of Classes / Objects

| | | | |
|---|---|---|---|
| **Class** | Sensor | | |
| | A generalized edge class for the sensors to measure the conditions of the environment. This class is not meant to be referenced directly, and is only intended to be a templet for other classes. | | |
| **Data** | **Name:** | **Type:** | **Constraints:** |
| | Pin | integer | Must be a Viable pin on the microcontroller |
| | Current_Value | Temp | Template to hold the Last read value |
| **Functio** | **Name:** | **Description** | |
| | Sample() | Returns the current value of the sensor | |

| | | | |
|---|---|---|---|
| **Class** | Temp_Sensor | | |
| | An edge class that acts as the software interface for the temperature sensor. Derived from the Sensor Class | | |
| **Data** | **Name:** | **Type:** | **Constraints:** |
| | Current_Temp | Integer | Value in Degrees Fahrenheit |
| | Pin | integer | Must be a Viable pin on the microcontroller |
| **Function** | **Name:** | **Description** | |
| | Sample_Temp() | Retrieves current temperature from the sensor | |

| | | | |
|---|---|---|---|
| **Class** | Humid_Sensor | | |
| | An edge class that acts as the software interface for the humidity sensor. Derived from the Sensor Class | | |
| **Data** | **Name:** | **Type:** | **Constraints:** |
| | Current_Humid | Integer | Value in percent (0 – 100 Inclusive) |
| | Pin | integer | Must be a Viable pin on the microcontroller |
| **Function** | **Name:** | **Description** | |
| | Sample_Humid() | Retrieves current Humidity from the sensor | |

| | | | |
|---|---|---|---|
| **Class** | Time_of_Day | | |
| | An entity class that that allows for the retrieval for the current time | | |
| **Data** | **Name:** | **Type:** | **Constraints:** |
| | | | |
| **Function** | **Name:** | **Description** | |
| | Get_Current_Time() | Returns the Number of seconds since Midnight | |

| Class | |
|---|---|
| Class | Environment_Profile |
| | An entity class that holds data for what should be considered an ideal environment for the inhabitants of the enclosure. |

| Data | Name: | Type: | Constraints: |
|---|---|---|---|
| Data | Name | String | 64 Character Max Length |
| | Temp_Min | Integer | Must be less than Temp_Max<br>Value in Degrees Fahrenheit |
| | Temp_Max | Integer | Must be greater than Temp_Min<br>Value in Degrees Fahrenheit |
| | Humid_Min | Integer | Must be non-negative<br>Must be less than Humid_Max<br>Value in Percent (0 – 100 inclusive) |
| | Humid_Max | Integer | Must be non-negative<br>Must be greater than Humid_Min<br>Value in Percent (0 – 100 inclusive) |
| | Daylight_On | Integer | Note: “Sunrise”<br>Must be between 0 and 86402<br>Value in Seconds after Midnight (00:00:00) |
| | Daylight_Off | Integer | Note: “Sunset”<br>Must be between 0 and 86402<br>Value in Seconds after Midnight (00:00:00) |

| Functions | Name: | Description |
|---|---|---|
| Functions | Environment_Profile() | Constructor Class for the Environment_Profile class |
| | Get_Name() | Returns the name of the Profile |
| | Temp_In_Range() | Checks if the current Temperature is in acceptable range |
| | Humid_In_Range() | Checks if the current Humidity is in acceptable range |
| | Time_Is_day() | Checks if the Daylight Should be ON or OFF |

| | | | |
|---|---|---|---|
| **Class** | Device | | |
| | An edge class is intended to be used to control a connected device through a circuit. There are intended to be multiple instances of these classes, one for each device the system will be controlling. | | |
| **Data** | **Name:** | **Type:** | **Constraints:** |
| | `Power` | Boolean | |
| | `Device_Type` | Enum(Int) | {Light, Humid, Heat_Bulb, Heat_Pad} |
| | `Pin` | Integer | Must be a Viable pin on the microcontroller |
| **Functions** | **Name:** | **Description** | |
| | `Device()` | Constructor function for the `Device` class | |
| | `Power_On()` | Powers on the Connected Device | |
| | `Power_Off()` | Powers off the Connected Device | |
| | `Get_Status()` | Returns the value of `Power` | |
| | `Get_Type()` | Returns the value of `Device_Type` | |

| | | | |
|---|---|---|---|
| **Class** | Control_Class | | |
| | This controller class is the main class in this system, collecting information from various other classes and using the information given to determine what devices need to be powered on or off. | | |
| **Data** | **Name:** | **Type:** | **Constraints:** |
| | `Profile_Name` | String | 64 Character Max Length |
| | | | |
| **Function** | **Name:** | **Description** | |
| | `Run_System()` | The main function of the system | |

| | | | |
|---|---|---|---|
| **Class** | Display_Class | | |
| | A sub-class that acts as a base for the Window_Class. Defines the entire layout of the window we created. | | |
| **Data** | **Name:** | **Type:** | **Constraints:** |
| | Grid_Layout | Integer | Must be passed GridLayout.cols and GridLayout.rows |
| | | | |
| **Functions** | **Name:** | **Description** | |
| | Initialize | Sets up the graphical user interface in a matrix fashion | |
| | Create_File_Button | Creates a button for create file option | |
| | Load_File_Button | Creates a button for load file option | |
| | Quit_Button | Creates a button that exits the program | |
| | Title_Label | A title label for the program | |
| | Time_Label | Shows current time | |
| | Temperature_Graphic | A pictorial display of the current temperature | |
| | Humidity_Graphic | A pictorial display of the current relative humidity | |
| | Temperature_Label | A label for temperature graphic | |
| | Humidity_Label | A label for humidity graphic | |
| | Current_File_Label | Displays the current configuration file | |
| | TempNum_Label | Numerical label for the temperature | |
| | HumNum_Label | Numerical label for the relative humidity | |
| | Day_On_Label | A pictorial display for the day on setting | |
| | Day_Off_Label | A pictorial display for the day off setting | |

| | | | |
|---|---|---|---|
| **Class** | `Window_Class` | | |
| | A base class of our Python graphical user interface. It is used to build our graphical user interface. | | |
| **Data** | **Name:** | **Type:** | **Constraints:** |
| | `Self` | Widget | Must be of type Widget |
| | | | |
| **Function** | **Name:** | **Description** | |
| | `Build()` | Initializes the app via graphical user interface library | |
| | | | |

## b. Module Cohesion

- `Environment_Profile`

The `Environment_Profile` class is a data class whose sole functionality is to store data that is collected from the sensors and devices. It does have some functions to manipulate the data collected, but overall this class simply stores data. It exhibits functional cohesion.

- `Sensor`

The functionality of the `Sensor` class will largely be hardware dependent. However, the core functionality of a sensor is universal. Since the class depends on implemented hardware, the `Sensor` class will exhibit functional to informational cohesion.

- `Device`

The functionality of the `Device` class will also largely be hardware dependent. However, each device will function as anticipated according to its purpose in the system. Since the class depends on implemented hardware, the `Device` class will exhibit functional to informational cohesion.

- `Control_Class`

  The Control_Class has power over all the other modules in our system

- `Time_of_Day`

  The `Time_of_Day` class simply retrieves the time of day from a device. Time is handled differently depending on the hardware and software implemented, but the core functionality of this class remains the same. It exhibits functional cohesion.

- `Display_Class`

  The functionality of the `Display_Class` will be done based on the software implemented to construct the graphical user interface. While it does depend on information from the `Control_Class`, it largely acts independently since its core functionality is to generate a working graphical user interface. This class exhibits functional cohesion.

## c. Module Coupling

Due to the nature of our system, the `Control_Class` is where all the coupling will occur. Our system is focused around a main control module that gathers information from all other modules and acts accordingly. All devices utilized in the system act independently, but the microcontroller brings those individual components together and exerts control over them. Therefore, our system exhibits control coupling as a worst-case scenario.

# IV. Functional Description

## a. Input / Output / Return Parameters / Types

<table>
<tr><td rowspan="3">Function</td><td colspan="3"><code>Run_System()</code></td></tr>
<tr><td>Class:</td><td colspan="2"><code>Control_Class</code></td></tr>
<tr><td colspan="3">The main function of the system<br><br>Creates all Supporting Classes<br>Loops the Following:<br>- Retrieves environmental data form the Various sensors and the current time of day from the <code>Time_Of_Day</code> Module.<br>- Compares send the collected data to the current <code>Enviroment_Profile</code>, being returned info of how the current conditions compare to the desired conditions<br>- Humidity<br>  - If Humidity is low, turn on humidity device<br>  - If Humidity is High, Turn of Humidity Device<br>  - Else, (Humidity is in range) Do Nothing<br>- Daylight<br>  - If Daytime and Daylight Device is off, turn on Daylight Device<br>  - If Nighttime and Daylight Device is on, turn off Daylight Device<br>  - Else, Do nothing<br>- Temperature<br>  - If Temperature is low and Heat Bulb is off, Turn on Heat bulb<br>  - Else If Temperature is low, and Heat Pad is off, Turn on Heat pad<br>  - Else if Temperature is high and Heat pad is on, Turn off Heat pad<br>  - Else if Temperature is high and Heat bulb is on, Turn off Heat bulb<br>  - Else, (Temp is in Range) Do Nothing<br>- Rest for 30 Seconds<br>- Loop</td></tr>
<tr><td rowspan="2">Input</td><td>Name:</td><td>Type:</td><td>Constraints:</td></tr>
<tr><td colspan="3">None</td></tr>
<tr><td rowspan="2">Return</td><td>Type</td><td colspan="2">Description</td></tr>
<tr><td colspan="3"><code>Void</code></td></tr>
</table>

<table>
<tr><td rowspan="3">Function</td><td colspan="3"><code>Environment_Profile()</code></td></tr>
<tr><td>Class:</td><td colspan="2"><code>Device</code></td></tr>
<tr><td colspan="3">Constructor function for the <code>Environment_Profile</code> class</td></tr>
<tr><td rowspan="7">Inputs</td><td>Name:</td><td>Type:</td><td>Constraints:</td></tr>
<tr><td><code>Temp_Min</code></td><td>Integer</td><td>Must be less than <code>Temp_Max</code><br>Value in Degrees Fahrenheit</td></tr>
<tr><td><code>Temp_Max</code></td><td>Integer</td><td>Must be greater than <code>Temp_Min</code><br>Value in Degrees Fahrenheit</td></tr>
<tr><td><code>Humid_Min</code></td><td>Integer</td><td>Must be non-negative<br>Must be less than <code>Humid_Max</code><br>Value in Percent (0 – 100 inclusive)</td></tr>
<tr><td><code>Humid_Max</code></td><td>Integer</td><td>Must be non-negative<br>Must be greater than <code>Humid_Min</code><br>Value in Percent (0 – 100 inclusive)</td></tr>
<tr><td><code>Daylight_On</code></td><td>Integer</td><td>Note: “Sunrise”<br>Must be between 0 and 86402<br>Value in Seconds after Midnight (00:00:00)</td></tr>
<tr><td><code>Daylight_Off</code></td><td>Integer</td><td>Note: “Sunset”<br>Must be between 0 and 86402<br>Value in Seconds after Midnight (00:00:00)</td></tr>
<tr><td rowspan="2">Return</td><td>Type</td><td colspan="2">Description</td></tr>
<tr><td colspan="3"><code>Void</code></td></tr>
</table>

| | | | |
|---|---|---|---|
| **Function** | `Get_Name()` | | |
| | Class: | `Environment_Profile` | |
| | Returns the name of the Profile | | |
| **Input** | **Name:** | **Type:** | **Constraints:** |
| | None | | |
| **Retur** | **Type** | **Description** | |
| | `String` | 64 Character Max Length | |

| | | | |
|---|---|---|---|
| **Function** | `Temp_In_Range()` | | |
| | Class: | `Environment_Profile` | |
| | Checks if the current Temperature is in acceptable range | | |
| **Inputs** | **Name:** | **Type:** | **Constraints:** |
| | `Current_Temp` | Integer | Value in Degrees Fahrenheit |
| **Return** | **Type** | **Description** | |
| | `Integer` | If `Current_Temp` is below `Temp_Min`, Return<br>`Current_temp - Temp_Min`<br>If Current_Temp is above Temp_Max, Return:<br>`Current_temp - Temp_Max`<br>Else (If within range), Return: 0 | |

<table>
<tr><td rowspan="3">Function</td><td colspan="3"><code>Humid_In_Range()</code></td></tr>
<tr><td>Class:</td><td colspan="2"><code>Environment_Profile</code></td></tr>
<tr><td colspan="3">Checks if the current Humidity is in acceptable range</td></tr>
<tr><td rowspan="2">Inputs</td><td>Name:</td><td>Type:</td><td>Constraints:</td></tr>
<tr><td><code>Current_Humid</code></td><td>Integer</td><td>Must be non-negative<br>Value in Percent (0 – 100 inclusive)</td></tr>
<tr><td rowspan="2">Return</td><td>Type</td><td colspan="2">Description</td></tr>
<tr><td><code>Integer</code></td><td colspan="2">If <code>Current_Humid</code> is below <code>Humid_Min</code>, Return<br><code>Current_Humid - Humid_Min</code><br>If Current_Humid is above Humid_Max, Return:<br><code>Current_Humid - Humid_Max</code><br>Else (If within range), Return: 0</td></tr>
</table>

<table>
<tr><td rowspan="3">Function</td><td colspan="3"><code>Time_Is_Day()</code></td></tr>
<tr><td>Class:</td><td colspan="2"><code>Environment_Profile</code></td></tr>
<tr><td colspan="3">Checks if the current time is within Daylight hours</td></tr>
<tr><td rowspan="2">Inputs</td><td>Name:</td><td>Type:</td><td>Constraints:</td></tr>
<tr><td><code>Current_Time</code></td><td>Integer</td><td>Must be between 0 and 86402<br>Value in Seconds after Midnight (00:00:00)</td></tr>
<tr><td rowspan="2">Return</td><td>Type</td><td colspan="2">Description</td></tr>
<tr><td><code>Boolean</code></td><td colspan="2">Return True if <code>Current_Time</code> is between <code>Daylight_On</code> and<br><code>Daylight_Off</code><br>Else, Return False</td></tr>
</table>

<table>
<tr><th rowspan="3">Function</th><td colspan="3" align="center"><code>Sample_Temp()</code></td></tr>
<tr><td>Class:</td><td colspan="2"><code>Temp_Sensor</code></td></tr>
<tr><td colspan="3">Retrieves current temperature from the sensor</td></tr>
<tr><th rowspan="2">Input</th><th>Name:</th><th>Type:</th><th>Constraints:</th></tr>
<tr><td colspan="3" align="center">None</td></tr>
<tr><th rowspan="2">Return</th><th>Type</th><th colspan="2">Description</th></tr>
<tr><td><code>Integer</code></td><td colspan="2">Returns the temp value in degrees Fahrenheit</td></tr>
</table>

<table>
<tr><th rowspan="3">Function</th><td colspan="3" align="center"><code>Sample_Humid()</code></td></tr>
<tr><td>Class:</td><td colspan="2"><code>Humid_Sensor</code></td></tr>
<tr><td colspan="3">Retrieves current humidity from the sensor</td></tr>
<tr><th rowspan="2">Input</th><th>Name:</th><th>Type:</th><th>Constraints:</th></tr>
<tr><td colspan="3" align="center">None</td></tr>
<tr><th rowspan="2">Return</th><th>Type</th><th colspan="2">Description</th></tr>
<tr><td><code>Integer</code></td><td colspan="2">Value in percent (0 – 100 Inclusive)</td></tr>
</table>

| | | | |
|---|---|---|---|
| **Function** | `Get_Current_Time()` | | |
| | Class: | `Time_of_Day` | |
| | Returns the Number of seconds since Midnight | | |
| **Input** | **Name:** | **Type:** | **Constraints:** |
| | None | | |
| **Return** | **Type** | **Description** | |
| | `Integer` | Must be between 0 and 86402<br>Value in Seconds after Midnight (00:00:00) | |

| | | | |
|---|---|---|---|
| **Function** | `Device()` | | |
| | Class: | `Device` | |
| | Constructor function for the `Device` class | | |
| **Inputs** | **Name:** | **Type:** | **Constraints:** |
| | Device_Type | Boolean(Int) | {Light, Humid, Heat_Bulb, Heat_Pad} |
| | Pin | Integer | Must be a viable pin on the microcontroller |
| **Return** | **Type** | **Description** | |
| | `Void` | | |

| | | | |
|---|---|---|---|
| **Function** | `Power_On()` | | |
| | Class: | `Device` | |
| | Powers on the Connected Device<br>Have the GPIO pin identified by `Pin`, Output “ON” | | |
| **Input** | **Name:** | **Type:** | **Constraints:** |
| | None | | |
| **Return** | **Type** | **Description** | |
| | `Void` | | |

| | | | |
|---|---|---|---|
| **Function** | `Power_Off()` | | |
| | Class: | `Device` | |
| | Powers off the Connected Device<br>Have the GPIO pin identified by `Pin`, Output “OFF” | | |
| **Input** | **Name:** | **Type:** | **Constraints:** |
| | None | | |
| **Return** | **Type** | **Description** | |
| | `Void` | | |

<table>
<tr><td rowspan="3">Function</td><td colspan="3">Get_Status()</td></tr>
<tr><td>Class:</td><td colspan="2">Device</td></tr>
<tr><td colspan="3">Returns the value of Power</td></tr>
<tr><td rowspan="2">Input</td><td>Name:</td><td>Type:</td><td>Constraints:</td></tr>
<tr><td colspan="3">None</td></tr>
<tr><td rowspan="2">Return</td><td>Type</td><td colspan="2">Description</td></tr>
<tr><td>Bool</td><td colspan="2">Return True if Power is True<br>Else, Return False</td></tr>
</table>

<table>
<tr><td rowspan="3">Function</td><td colspan="3">Get_Type()</td></tr>
<tr><td>Class:</td><td colspan="2">Device</td></tr>
<tr><td colspan="3">Returns the value of Device_Type</td></tr>
<tr><td rowspan="2">Input</td><td>Name:</td><td>Type:</td><td>Constraints:</td></tr>
<tr><td colspan="3">None</td></tr>
<tr><td rowspan="2">Return</td><td>Type</td><td colspan="2">Description</td></tr>
<tr><td>Enum(integer)</td><td colspan="2">Returns the value of Device_Type</td></tr>
</table>

### b. Modules Used

The control class accesses each of the other individual classes, which are independent of each other. This is required so we can properly regulate each component.

### c. Files Accessed

The only module which access files is the display class to load the configuration settings which are then passed into the Control class to be handled.

### d. Real-Time Requirements

None of the above modules have specific real-time requirements. The system gradually adjusts the environment to fit the desired configuration.

## V. Reuse / Portability

The design of our system allows for reuse in various types of scenarios, including using different sensors and regulators. It can also be ported to any system that has support for GPIO pins and the language we choose.

# VI. Design Workflow CASE Diagrams

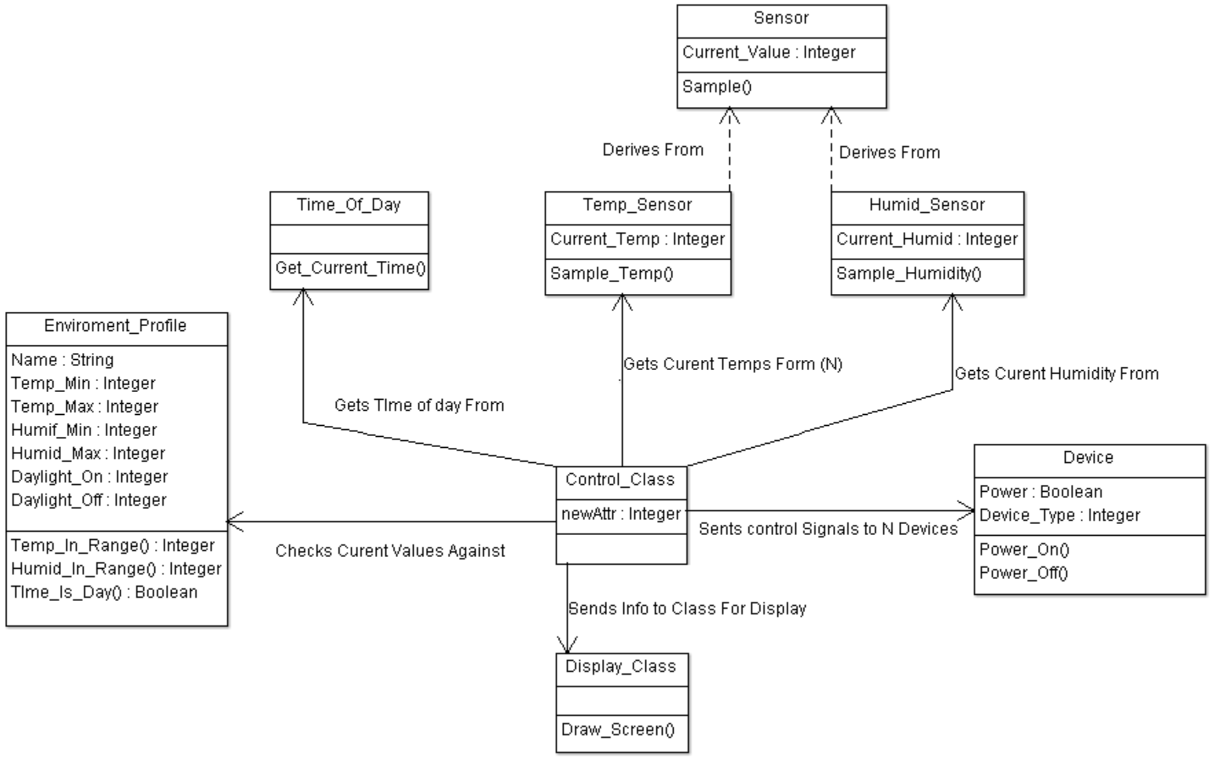



*Figure 2: Detailed Class Diagram (Bioni, Law and Norris)*

# VII. Design Testing

- Added more detailed descriptions for more complex functions/modules.
- Reanalyzed cohesion and coupling and revised.
- Expanded Display class.

# Appendix A: Schematic & Bill of Materials

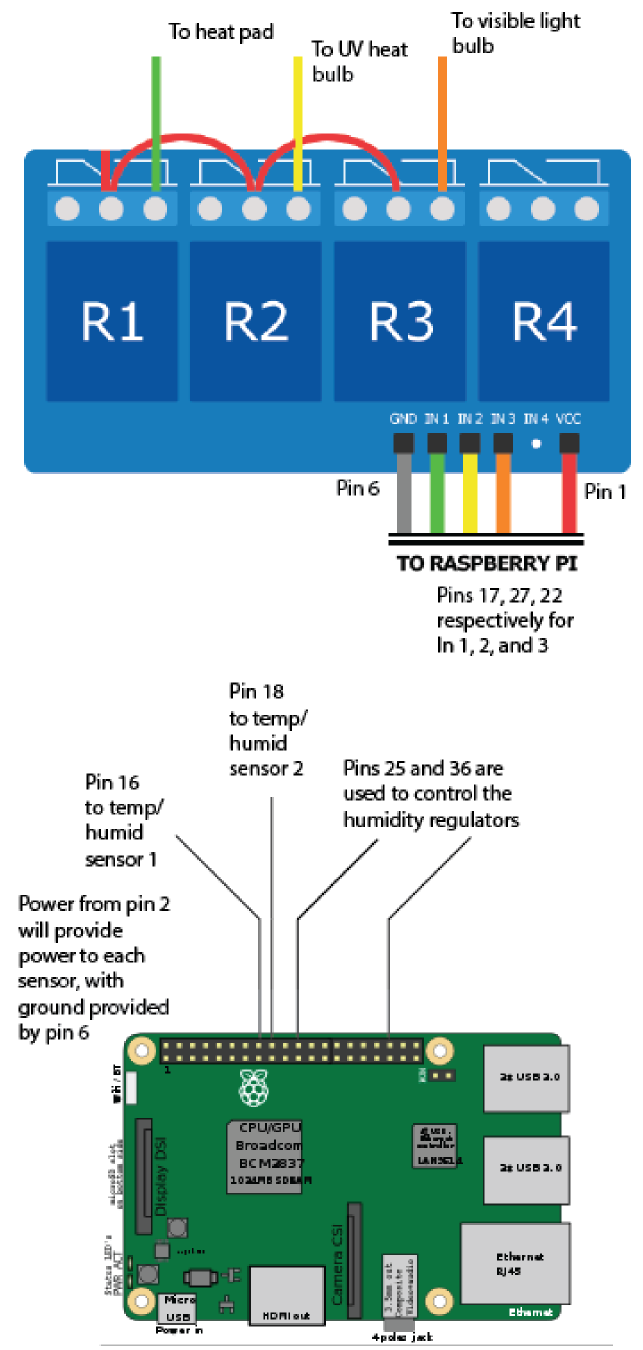



*Figure 3: Wire-Diagram*

| Bill of Materials | | | | |
|---|---|---|---|---|
| Item | Description | Vendor | Product Code | Price |
| Humidity and Temperature Sensor - RHT03 (x2) | Accurately measures the current temperature and relative humidity | Sparkfun | 10167 | \$9.95 x 2 = \$19.90 |
| Kuman 4 Channel DC 5V Relay Module | A 4-channel relay controllable by a Raspberry Pi | Amazon | B01BACQF1Y | \$7.99 |
| Zilla 10-Gallon Reptile Tank Starter Kit | Contains the basic materials needed for a reptile setup | Amazon | B005E7Q9VS | \$60.00 |
| Zoo Med Eco Earth Coconut Fiber Substrate | Required for proper testing of humidity and temperature control | Amazon | B00BUFSX7G | \$11.69 |
| | | | Total: | \$99.58 |

# Appendix B: References

Bioni, Alex, et al. "RePitile System - Specifications Document." 2017.

Sheppard, Scott T, et al. "RePitile System - Requierments Document." 2017.

Willseph. *RaspberryPi Thermostat*. n.d. <https://github.com/Willseph/RaspberryPiThermostat>.

# Appendix C: Team Details

## A. Alex Bioni

Analysis workflow supervisor. Gathered all the specific intel to create document. Provided a detailed description on the purpose and intended audience of the Repitile System design document. Obtained a broad understanding of the requirements needed in the design document. Also obtained the ties from the specification document to the design document.

## B. Seth Law

Responsible for the requirements workflow. Made sure all the requirements made it into the design document. Designed the display diagram. He is mostly focusing on the implementation for the touch screen. Discussed cohesion and coupling for the system.

## C. Scott Sheppard

The leader of the design workflow. Provided detailed graphs for the CRC functions, functional description, and design organization diagrams. He also made sure every team member provided their work to the document.

## D. Wesley Norris

Finally, the implementation supervisor. Responsible for the wire diagram and block diagram. Wes made sure everything was included into the design document so when it’s his time for the implementation phase, everything goes smoothly.

# Appendix D: Workflow Authentications

Alex Bioni: ____________________________________________________________

Date: ____________________

Seth Law: ____________________________________________________________

Date: ____________________

Wesley Norris: ________________________________________________________

Date: ____________________

Scott Sheppard: _______________________________________________________

Date: ____________________