RePitile System

Specification Document

Alex Bioni Seth Law Wesley Norris Scott Sheppard

California University of Pennsylvania

CSC 490 – Senior Project 1

Due 11/6/17

Instructors Comments

Table of Contents

Abstract

The objective of this document is to determine the specifications for the RePitile system; a modular, automatic reptile tank care system. In order to understand this document please refer back to the requirements document. The specifications provided in this document will describe the way that the RePitile system is constructed and operated. The System Description section describes the constraints and requirements for how the system is to be developed to function properly. The System Description will also provide in detail what kind of hardware needs to be used to create the system, what languages can be used to code, how much time it will take to create the RePitile system, and how much funding will be needed to create the system. This document will also provide the testing criteria. At this stage, tests should be done to make sure the RePitile systems requirements are complete and up to date. This document differs from the previous requirements document by going into detail on what materials are required, constraints on software and hardware, and how the various separate components are connected.

Description of the Document

I. Purpose and Use

The purpose of this document is to describe the required system constraints, requirements, modeling for the hardware and software, and act as a binding agreement between the client and developers of the RePitile system. This document should be consulted during the creation of the RePitile system to ensure the system works as described. The engineers who will be building the RePitile system will find all specifications in this document on how to create the system: what hardware is required to have a functioning product, and what language constraints must be considered to program the system. This document will also provide a budget that is required to obtain all the parts and hardware needed to build the system. Once complete, tests will be conducted to determine if the requirements from the requirements document are met. If all the requirements and specifications are met the next phase is to start the design phase.

II. 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 type of hardware that is required and constraints on how the software can be built to control the system is also detailed.

System Description

I. Overview

The RePitile system is designed to monitor and adjust the environmental conditions for a reptile pet to ensure a healthy living environment. Conditions that the RePitile system is designed to control are: temperature, humidity, and light. The components that are configured to adjust conditions should be easily replaceable and maintainable by the product owner without intimate knowledge of the underlying system implementation. The user interface should be simple and easy to use, while providing accurate and relevant information for the owner. The system will use various sensors to monitor the conditions and engage environment regulators (such as a heat lamp) to meet the desired settings specified in the selected configuration file. Once the RePitile system is created it will be able to fit on any tank the consumer selects.

II. Block Diagram of System

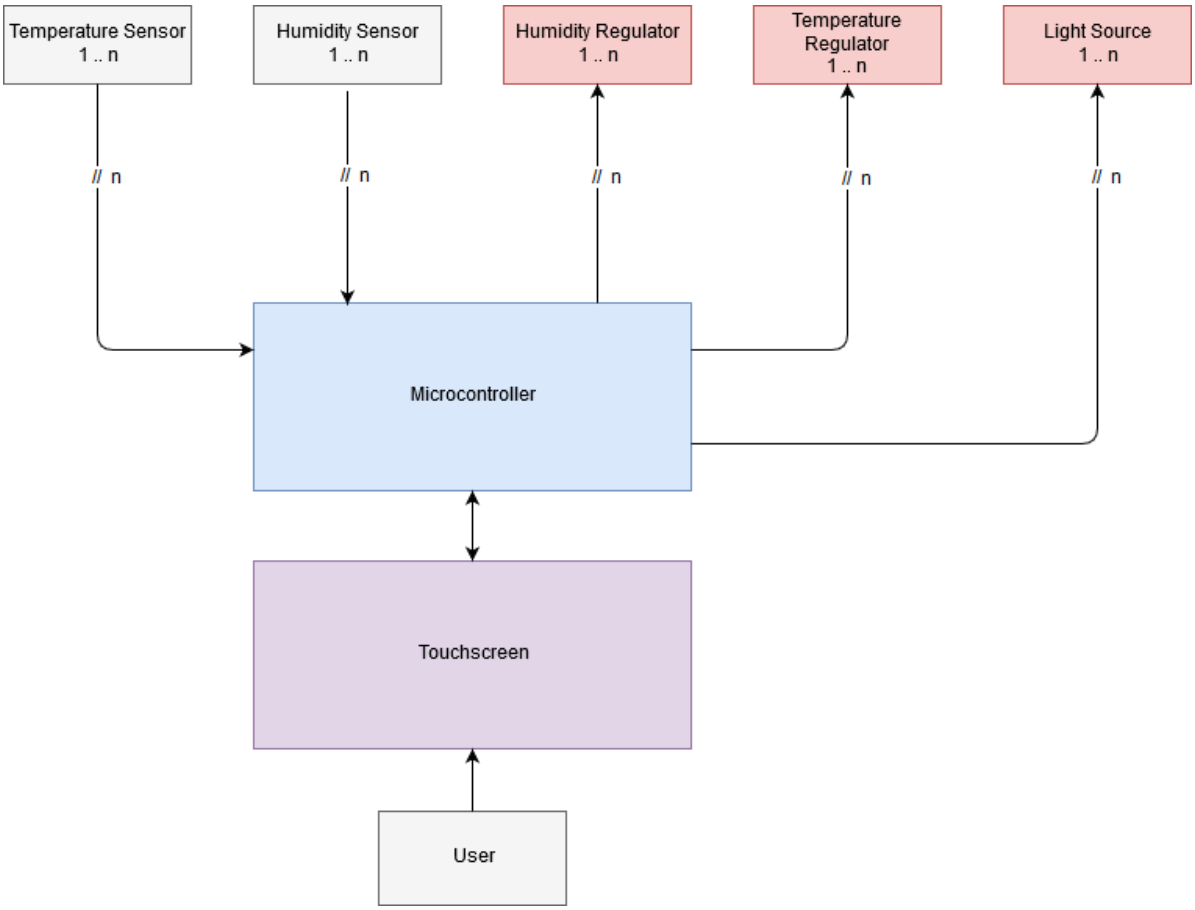

Figure 1: Block Diagram of System

This block diagram shows how each part of the system is connected. There are four sources of input: temperature sensors, humidity sensors, the touchscreen, and user input. These provide the microcontroller with information about tank conditions or about what actions to perform. The output receivers are: humidity regulators, temperature regulators, light sources, and the touchscreen. Each of these components allow the microcontroller to regulate the conditions in the tank to provide a consistent environment for the reptile. The user will be able to interact with the system via the touchscreen and control triggering of the regulators.

III. Environment and Constraints

The RePitile system is designed to be run in an indoor environment at normal conditions (average room temperature, humidity) without needing regular maintenance. It will also be active 24 hours a day, 7 days a week for continuous monitoring. Constraints are based on making the RePitile system as accurate, effective, and inexpensive as possible.

A. End User Profile

Reptile owners will primarily be the end user of the RePitile system. Before using our system, users should already have a vivarium to house their pet(s), an understanding of the kind of environment their pet requires in order to live comfortably, and some general familiarity with properly caring for their pet. Users should also have a basic understanding of how to operate a touchscreen interface similar to that of a smartphone or tablet. No formal training is required to operate the RePitile system. Everything can be configured and managed via the touchscreen interface or for more advanced users, connecting to the embedded system via networking if available.

B. User Interaction

Users will interact directly with the microcontroller via a touchscreen and graphical user interface (GUI). This makes using the RePitile system much easier than using a text interface. The touchscreen will consist of graphical displays of the current conditions inside the environment as well as two interactable icons. The first icon will allow the users to select a configuration file that the system will use to maintain the environment conditions. The second icon will allow users to create their own configuration file if they aren’t satisfied with the ones provided.

The GUI is to be simple and sleek, providing the necessary components in an effective yet visually appealing manner. All interfacing will be done through the touchscreen attached to the microcontroller so the functionality of the GUI is essential.

C. Hardware Constraints

In order to achieve the goals of having an accurate, long lasting system that is also inexpensive, we can immediately discard a large amount of higher end embedded systems. Power consumption must also be low which lends itself to systems built on the ARM architecture. Possible considerations are the Raspberry Pi, ST Microelectronics, and Atmel brand microcontrollers. However, any consumer grade ARM-based microcontroller with the following requirements would be sufficient: high degree of quality to ensure that the microcontroller will be able to function for long periods of time, abundance of connections to interface other hardware with the microcontroller, and is not very expensive (covered in the Cost Constraints section). Also, as previously stated, the user interface will be based on a touchscreen, so a microcontroller that is able to control the touchscreen is a requirement.

Other hardware requirements are for the regulation of the environment itself. The system needs at minimum: at least one temperature sensor, at least one humidity sensor, at least one controllable light fixture, at least one controllable humidity source, and at least one controllable heat source. Each of these components is vital to the system operation and must be functional with a minimal degree of error. The system may also need a relay circuit to control parts of the system (such as lighting) that require a higher voltage or amperage than the microcontroller can offer by itself.

D. Software Constraints

The software for the RePitile system is essential for proper operation. Since the software is the core of the system as a whole, we need software that will work correctly on the target microcontroller and operating system that we choose. The operating system is the basis for communicating with the other hardware and thus extremely important for proper operation of the system as a whole. The operating system needs to be able to allow software running on it to interface with the general-purpose input-output that the microcontroller board provides and also allow for multiple pieces of software to run at once. We will also need a compiler and linker that is able to target the architecture of the chosen microcontroller so that we can build the core software for the RePitile system. This is extremely important when moving into the analysis phase, as inability to properly develop software for the target hardware will become a large issue. Likewise, the language that the compiler supports must be able to interface through the operating system to communicate with the other pieces of hardware. It should also have the proper structures to represent the problem at hand and implement the features in a timely fashion. The RePitile software also needs to be able to effectively interface with the touchscreen and user interface, changing the environment or configurations as needed.

E. Time Constraints

The main timing constraint for the RePitile system is with reading the environment conditions from the sensors. These readings must be done over a specific timing interval, which can be set by the user. This is important for accurately tracking changes in the environment which can allow the user to adjust the configuration as such. Other timing constraints are not as strict, for example in situations such as when asked to update the

current condition readings from the user interface, there shouldn't be a significant delay (more than 15 to 20 seconds) in displaying the new information to the user. The user interface should be as responsive as possible during operation and not degrade in performance over time.

F. Cost Constraints

The RePitile system is designed to be a fairly inexpensive automated reptile environment regulation system. No part should be priced above $50 individually if possible, while ensuring that each component meets the standards for quality and accuracy required for the system. The pricing constraints should make it possible for this system to be affordable for the consumer.

G. Other Concerns

The system should be able to fit on any tank the user selects. One major concern with the RePitile System is that the system might not fit on all tanks. During the design phase we need to make sure the RePitile System is designed to be universal for all tanks without sacrificing too much visual appeal. This specification will impact how the components are connected together.

IV. Acceptance Test Criteria

A. Testers

Testers of the RePitile system will be owners of reptiles/amphibians who do not have a system in place that allows their pet(s) to properly maintain internal temperature or who are looking to upgrade their current system. Testing should be conducted over the period of

about six hours to ensure all the sensors are working properly and that adjustments to heat and humidity are being done correctly.

B. Criteria for User Acceptance

This list defined what tests need to pass to be accepted as a successful project:

1. The system keeps temperatures within +/- 2 degrees for at least 80% of time active.
2. The system keeps the humidity within +/- 5% for at least 80% of the time active.
3. The system transitions the light sources from day to night cycle within one half of an hour of the set transition time.
4. The system responds properly to different system configurations specified by the user.
5. The system displays accurate information about past and current environment conditions.
6. The system configuration should be customizable from inside of the graphical user interface.

V. Integration of Separate Parts and Installation

There are multiple different hardware and software components for the RePitile system. Each of the hardware components (sensors, regulators, touchscreen) need to be connected in the proper area of the microcontroller, or in the case of some of the components, a controller that is connected to the microcontroller. The sensors and regulators will mainly be connected to the general-purpose input-output pins of the microcontroller so the microcontroller can directly read/write from each of them. The sensors and regulators will also need to be set up

inside or on top of the reptile's enclosure in the proper positions to monitor conditions accurately. Light sources and possibly some heat sources may need to be connected to a relay which will be controlled by the microprocessor system to ensure proper operating voltages. Each part of the system that requires power not supplied by the microcontroller will also need to be connected to a power source.

System Modeling

I. Functional

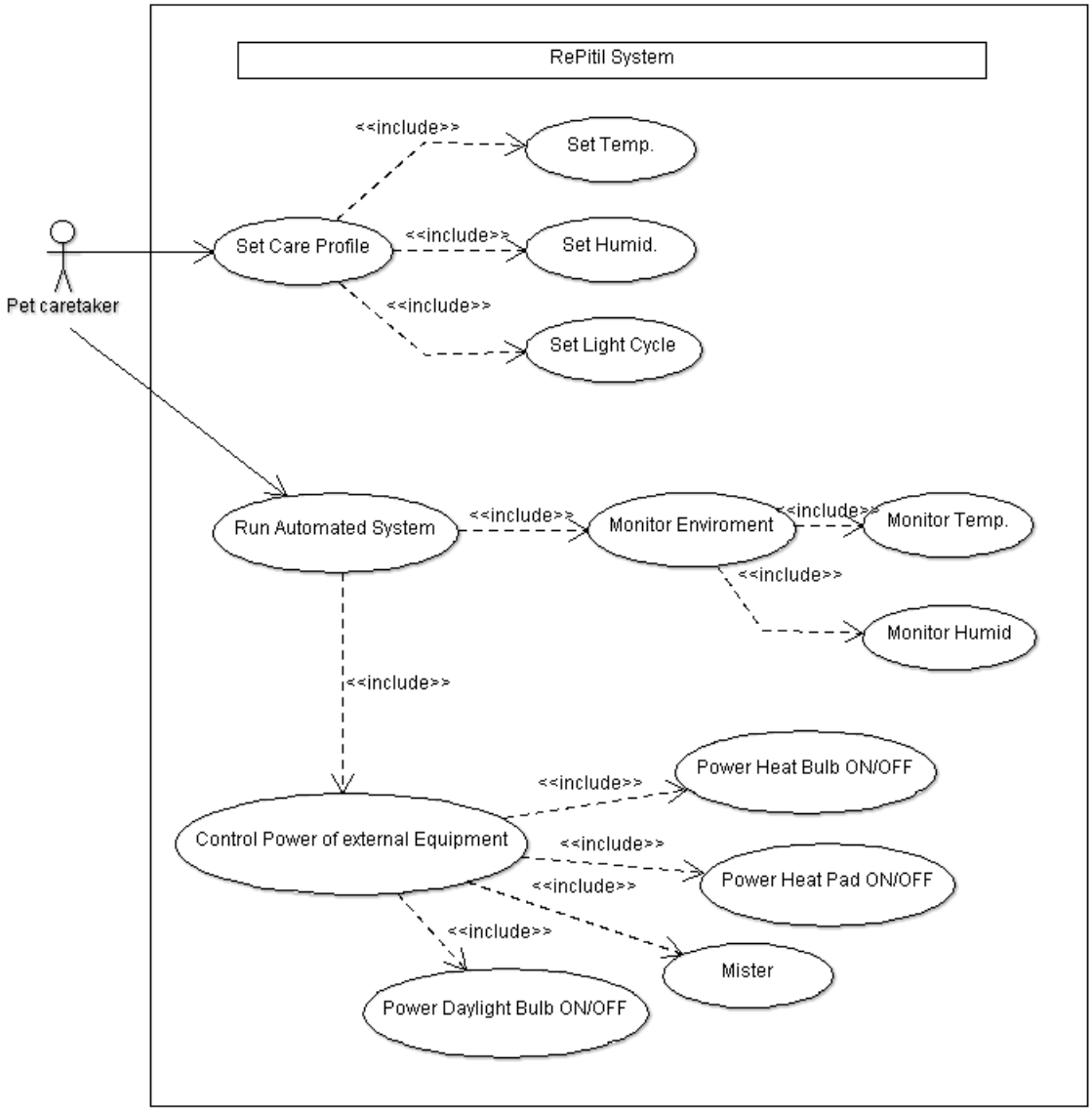

Figure 2: Uses Case Diagram, from Requirements Document [1]

This is a use case diagram that describes the general overall function of the RePitile System. The user will mainly be interacting with the Set Care Profile part since our system is

automatic. The most common and basic scenario the user will experience with the RePitile system is loading the proper configuration profile and monitoring values via the touchscreen.

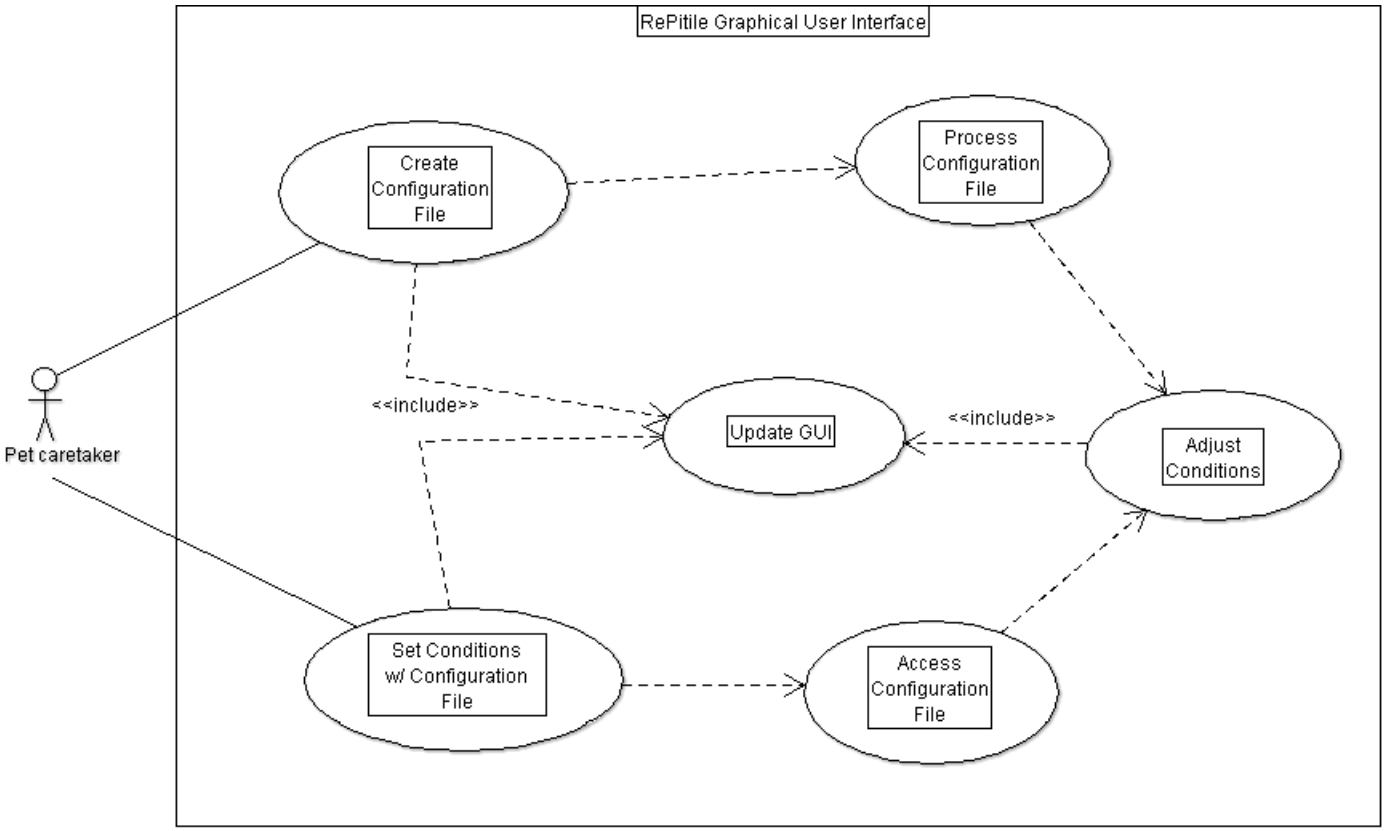

Figure 3: GUI Uses Case Diagram

The Graphical User Interface is a fundamental part of the RePitile System. This use case diagram demonstrates the basic functionality of the GUI. Users will be able to select a configuration file from the ones provided or have the option to make a configuration file of their own if none of the ones provided match their needs. In this scenario the user will either choose to pick new values for the conditions in the current configuration profile, or create a new profile. After that, the user will then fill in the new values, and save the changes. This event will cause the system to update the required conditions. One by one, for each sensor, the system will then

check the current conditions against the new requirements and attempt to adjust them as necessary. For any conditions that do require adjustment, it will trigger the regulators to turn on or change the operating value. For example, if it is not warm enough, it may cause a heat pad to increase its temperature by staying on for longer periods of time.

II. Entity

A. Class Diagram

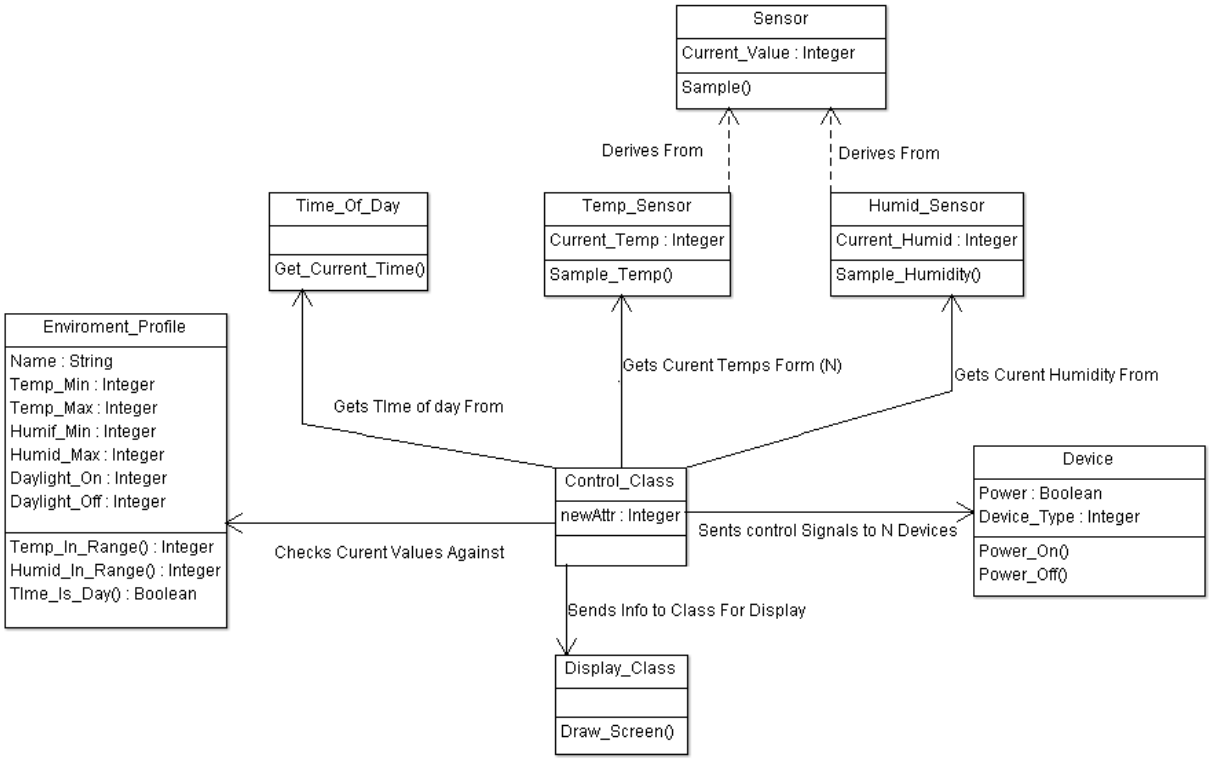

Figure 4: Automated Environment Control System Class Diagram

B. Class Name / Description / Type

Environment_Profile:

This entity class holds data for what should be considered an ideal environment for the inhabitants of the tank.

Device:

This edge class is intended to be used to control a connected device through a circuit. There are intended to be multiple of these classes, one for each device the system will be controlling.

Display_Class:

An edge class responsible for painting and outputting. This class is sent data of the current conditions, then uses the information to output a graphic display to the screen for the User to View and monitor.

Time_of_Day:

An entity class that keeps track of the time of day.

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.

Temp_Sensor:

An edge class for the temperature sensors, derived from the generic Sensor class. Used to measure the temperature of the environment.

Humid_Sensor:

An edge class for the humidity sensor, derived from the generic Sensor class. Used to measure the humidity of the environment.

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.

C. Attributes

Environment_Profile:

The data in this class includes an acceptable range for both temperature and humidity, represented by integer values for the minimum and maximum boundaries. It also contains a pair of integers to represent the number of seconds after Midnight should the light turn on (Daylight_On) or off (Daylight_Off) to simulate a 24-hour day/night cycle. The Name field is a string containing the common name of the species the environment is meant for. Most of the data is intended to be protected, using instead the public functions to return information. The Temp_In_Range() and Humid_In_Range() functions both work by comparing a passed in value to the appropriate maximum and minimum values. The system then returns a positive value if the value is above the range, a negative value if below the range, and a zero value if it is within the range. The Time_Is_Day() function is similar, but instead returns ‘true’ if the value is between the Daylight_On and Daylight_Off values, and false otherwise.

Device:

Each iteration of the Device class contains two pieces of data. The first is a Boolean value indicating whether the current device is on or off, using ‘true’ to represent being on. The other value is the Device_Type field. This indicates what kind of device it is. For example, 0 is a daylight bulb, 1 is the heat bulb, 2 would be the optional heat pad, and 4

for humidifiers/misters. The two functions are public to provide eternal classes the ability to send an “on” or “off” signal to the device.

Display_Class:

This class is not currently intended to hold any data pertinent to the operation of the system. The Draw_Screen() function will use a series of input parameters to create a graphic to draw on the screen of the user interface.

Time_of_Day:

This class is currently does not list any local data for its processing, but it is possible this may need to change depending on the method of implementation. The function Get_Current_Time() retrieves the current time from the system, and then converts it to the number of seconds after midnight for return to the calling function.

Sensor:

This class is a template for other classes, so the stored data, Current_Value, will be the value of the last measured data stored as an integer. The Sample function is a place holder for how to sample the appropriate sensor and return usable data.

Temp_Sensor:

Derived from the Sensor class, this class stores the current temperature in the Current_Temp as an integer. The Sample_Temp() function doesn’t require any input parameters and instructs this class to record the current temp and return it in degrees Fahrenheit.

Humid_Sensor:

Derived from the Sensor class, this class stores the current temperature in the

Current_Humid as an integer. The Sample_Humid () function doesn't require any input parameters and instructs this class to record the current humidity and return it as an integer percent out of 100.

Control_Class:

This class holds only a small amount of data, this includes the current level "Heat Level," so when there are multiple heat sources it knows which ones should be tuned on or off, allowing it to send the appropriate "on" or "off" message to appropriate Device.

III. Dynamic

A. State Charts

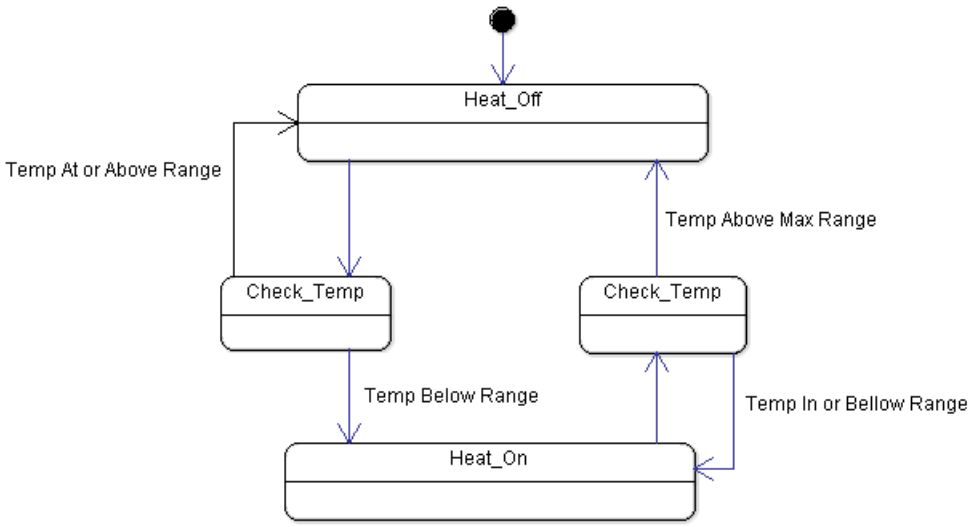

Figure 5: State Chart of The Temperature Control Functionality, 1st Iteration

This State chart diagram shows the initial concept for the thermal regulatory functionality of the system. This sub system turns or the heating elements when it gets too cool, and back off when the temperature gets too high. Note, because this state chart diagram has been expanded upon in the next diagram, the states, events and transitions are not elaborated on further in this document.

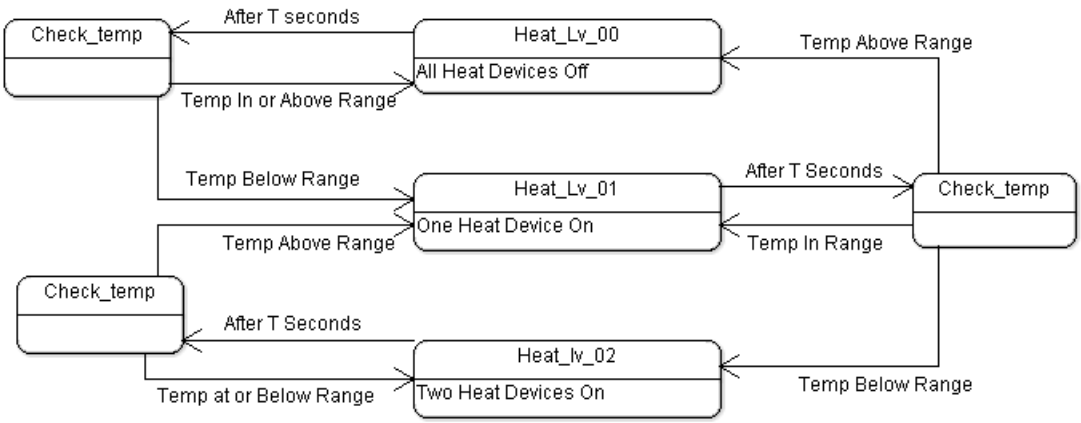

Figure 6: State Chart of The Temperature Control Functionality, 2nd Iteration

Here the state diagram for the regulating the temperature has been expanded accommodate two heat sources that don't need to both be on simultaneously, such as a lamp and a heat pad. So when the temperature drops it will continue to turn on heating elements, one at a time, and do the inverse if the heat is too high.

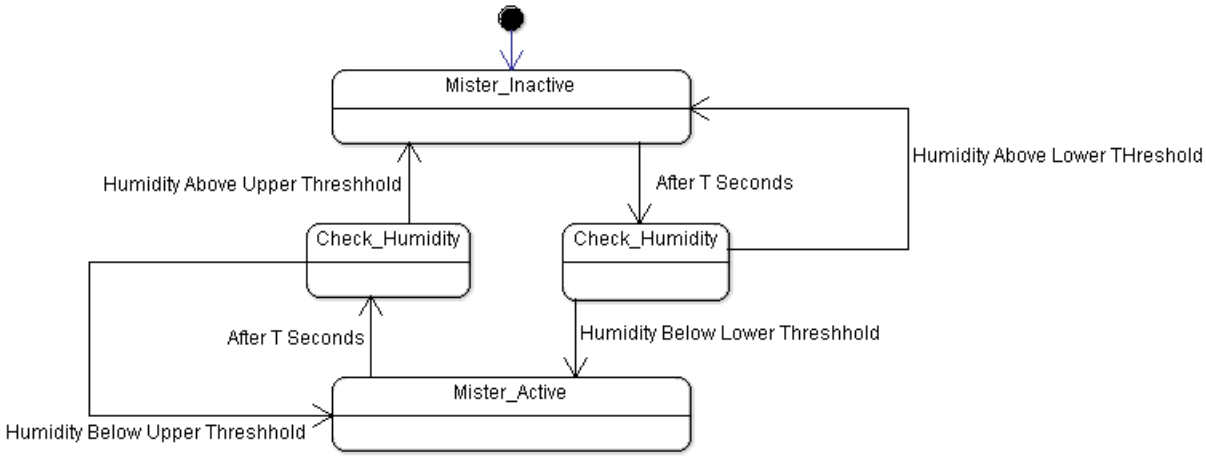

Figure 7: State Chart of The Humidity Control Functionality

Very similar to the initial state diagram for the thermals, this state diagram shows how the humidity is regulated, similarly turning on when the humidity is getting low, and turning off

once the upper threshold is reached. The level of humidity is allowed to fall passively until it once again is falls out of the threshold.

B. States

Heat_Lv_00:

A state in the Temperature control state diagram indicating when all heat devices are turned off.

Heat_Lv_01:

A state in the Temperature control state diagram indicating when only “level 01” heat devices (i.e. Heat Lamps) are turned on, with all others being on.

Heat_Lv_02:

A state in the Temperature control state diagram indicating when all heat devices are turned on.

Check_Temp:

These are transition state in the temperature state chart diagram, used as a decision point to move determine what state to transition to next.

Mister_Active:

A state in the Humidity control state diagram indicating when the mister, or other humidity increasing device is powered on.

Mister_Inactive:

A state in the Humidity control state diagram indicating when the mister, or other humidity increasing device is powered off.

Check_Humidity

These are transition state in the humidity state chart diagram, used as a decision point to move determine what state to transition to next.

C. Events

The only trigger event in any of the state chart diagrams is the event “After T seconds” to move from some of the “resting” states (i.e. Mister_Inactive, Mister_Active, Heat_Lv_01, ect.) to a “check” state (i.e. Check_Temp, Check_Humidity). This allows some time to be spent at each “resting” state as not have the system be caught in a situation where a device is constantly being turned on and off to maintain a certain value, to hopefully prolong the life of the attached equipment.

D. Transitions

The transitions in the Thermal state chart diagram not caused by the Timed event mentioned in the previous section move from a “check” state to a “resting” state. When then temperature is higher than desired, it will transition to a “lower resting state,” a state with less heat devices on, compared to the resting state it was on previously, if one is available. When the temperature is lower than desired, it will transition to a “higher resting state” then the one it was previously on, if available. And when the temperature is within the specified range, or there is no higher/lower state available for it to move to, it will return to the resting state it was on previously.

The transitions for the Humidity state chart diagram are simpler. Coming from the Mister_Inactive state, if the humidity is too low it moves to the Mister_Active state, else it returns to the Mister_Inactive state. In contrast, from the Mister_Active state, if the humidity is too high it moves to the Mister_Inactive state, else it returns to the Mister_Active state.

IV. Dataflow Diagrams

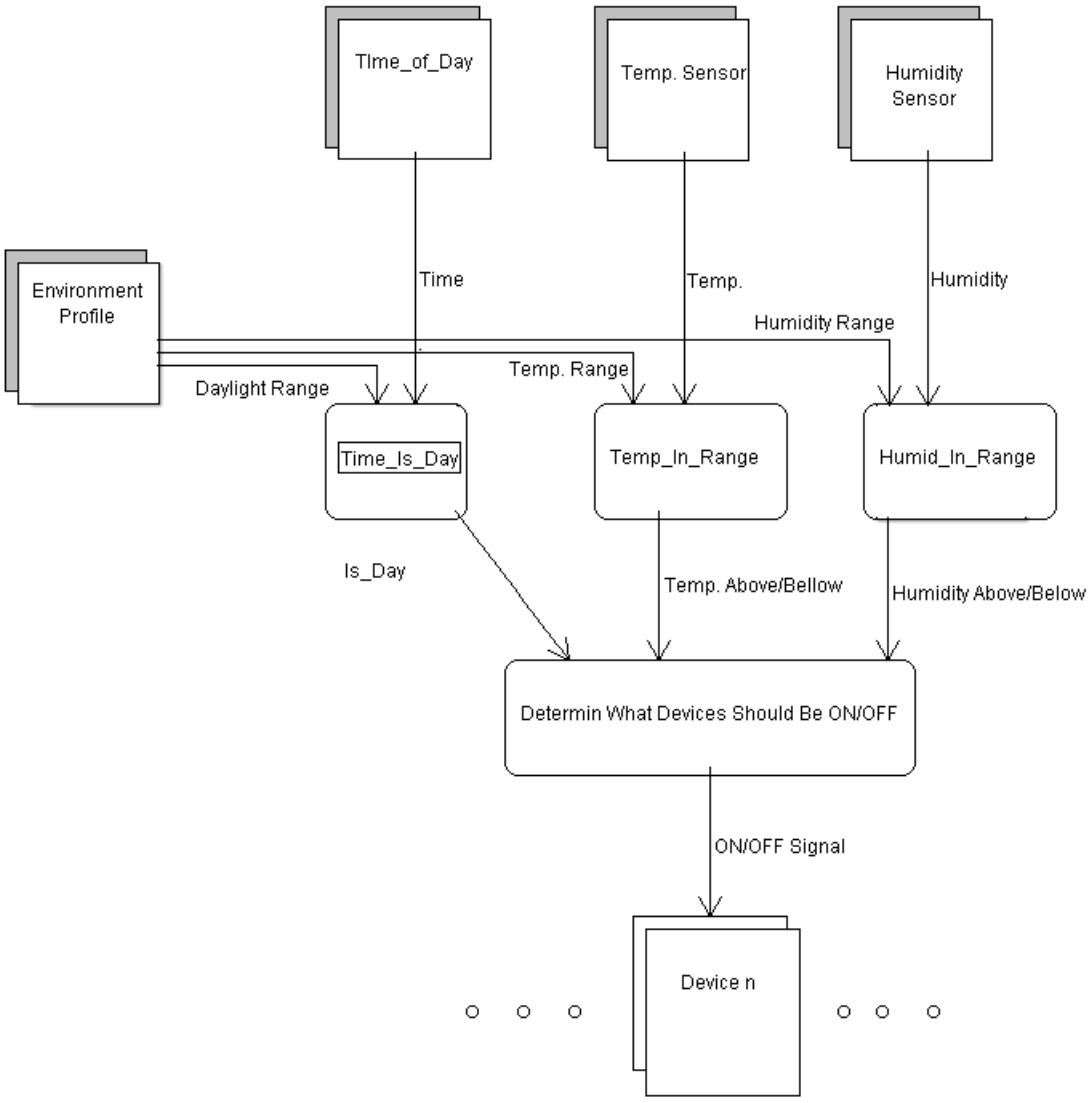

Figure 8: Automated Environment Control Data Flow Diagram

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.

Components / Tools Needed

- A microcontroller for the basis of the project, possible considerations include some version of the Raspberry Pi, products from ST Microelectronics, or Atmel.
- A device to control the power to multiple AC power circuits using DC current, such as the Kuman 4 Channel DC 5V Relay Module [2].
- A Surge Protector, to Plug the system into for safer testing.
- A touch screen apparatus for user input.
- A mister to increase humidity of the environment.
- Humidity sensors to measure the humidity of the environment.
- Temperature sensors to measure the temperature of the environment.
- A (micro) SD card to hold the Program and Data for the microcontroller to read.
- A Bread board for rapid prototyping of circuit layouts.
- A length Wire for connecting various components.
- Heat shrink for insulating the bare wire ends.
- Possibly a 3D printer to create a shell
- Wire cutters for cutting different components.
- Small Fan, preferably one with PWM capabilities

Appendix A: Technical Glossary

Ambient temperature – the typical conditions inside the environment

Ancillary product – provides appropriate support to necessary elements i.e. dirt, plants, food etc.

Bad shed – a shedding issue that arises in reptiles that can be caused by improper care

Configuration file – a template that the RePitile can read in and use to adjust environment

Ectothermic – the scientific term for a cold-blooded animal

Heat bulbs – a specialty UV/Infrared lightbulb that emits a controllable amount of heat

Heat pad – a device which attaches to the environment that enables the reptile to thermo-regulate

Microcontroller – a integrated circuit that uses a microprocessor to control and interface with other devices

Sub-tropical Regions – the region bordering the tropical region. Also called the temperate region.

Thermoregulation – the process by which core internal body temperature is maintained

Tropical Regions - the region of the Earth close to the equator between the tropics of Cancer and Capricorn

Vivarium – the specific environment that houses the reptile/amphibian

Appendix B: Team Details

I. Description of Team Member Contributions

A. Alex Bioni

Analysis workflow supervisor. Gathered all the specific intel to create document. Provided a detailed description on the purpose and overview of the Repitile System. Obtained a broad understanding of the requirements document in order to find specifics needed for the system. Described the specifications that will result is a successful implementation of our design.

B. Seth Law

Responsible for the requirements workflow. Gained an understanding on the domain of the Repitile System. Gave a in depth description on the user interaction. A touchscreen and graphical user interface will be used to allow the user to interact directly with the system. Also gave a description on the acceptance test criteria, describing how the user themselves will be the testers of the product.

C. Scott Sheppard

The leader of the design workflow. Contributed on the Dynamic section. Created the state charts. The state chart diagrams show how the thermal, temperature, and heat regulators will be implemented. Also, contributed to the attributes by describing in detail about the different sensors, classes, and the environment profile.

D. Wesley Norris

Finally, the implementation supervisor. Described the constraints of the hardware and software. Determined what kind of sensors will be needed, what microcontroller is needed and what power source is needed to build on the ARM. Also determined what software language will be implemented on the Repitile System. Finally, determined what criteria is needed for the user acceptance criteria.

Appendix C: References

[1] S. T. Sheppard, W. Norris, S. Law and A. Bioni, "RePitile System - Requierments Document," 2017.

[2] "Kuman 4 Channel DC 5V Relay Module for Arduino Raspberry Pi DSP AVR PIC ARM K49," kuman Ltd., 2017. [Online]. Available: http://www.kumantech.com/kuman-4-channel-dc-5v-relay-module-for-arduino-raspberry-pi-dsp-avr-pic-arm-k49_p0048.html. [Accessed 1 November 2017].

Appendix D: Workflow Authentication

Alex Bioni: _____

Date: _____

Seth Law: _____

Date: _____

Wesley Norris: _____

Date: _____

Scott Sheppard: _____

Date: _____