SYSC 3010
Design Proposal

Strawberi E-Tank

Due Date:
November 1st 2019

Group:
M2 (Stawberi E-Bois)

Written By:
Chase Badalato
Chase Fridgen
Haydn Brown

Problem Statement (1.0):

The objective of this project is to make caring for an animal easier through the use of automation, remote controlled systems, data collection, and visualization. The remote systems allow a user to care for the animals needs at any time and any place by giving it food, water, and changing its habitat temperature. The data collection and visualization allow the user to track habitat conditions and have a record of information if needed. It also allows the system to regulate itself in an event such as overheating. This design has practicality in a commercial setting such as PetSmart. Each animal would have its own E-Tank, with its own set of sensors. The database, GUI, and the Android App have the ability to have multiple E-Tanks connected at once, allowing for an easy way to view any desired E-Tank from one place.

All Functional Features:

- Give your pet food or water remotely
- Keep track of your pets movements and sleep schedule
- Set the temperature to a temperature of your liking, or set the temperature to be the current live temperature of anywhere in the world
- Change the intensity of the heat lamp / LED from within your app

System Architecture (2.0):

Figure 2.1 below depicts a high abstraction level of our proposed design with colors representing the following items.

Green: Databases. The (local) sensor log database stores sensor data in the form of JSON files. The (external) weather database gathers weather information from the internet to be used to heat the tank.
Pink: Hardware for user interaction. The user can interact with the system from either the Raspberry Pi's GUI, or the Android App.
Blue: Microcontrollers. The two raspberry Pi's and the Arduino are the microcontrollers that are programmed and make all of the decisions.
Orange: Sensors, LED, Heater, and Motor. These are how the system will interact with the terrarium and the pet.

Figure 2.2 shows the general layout that the system will have. All of the sensors are required per tank to have a fully functional semi-automated terrarium.

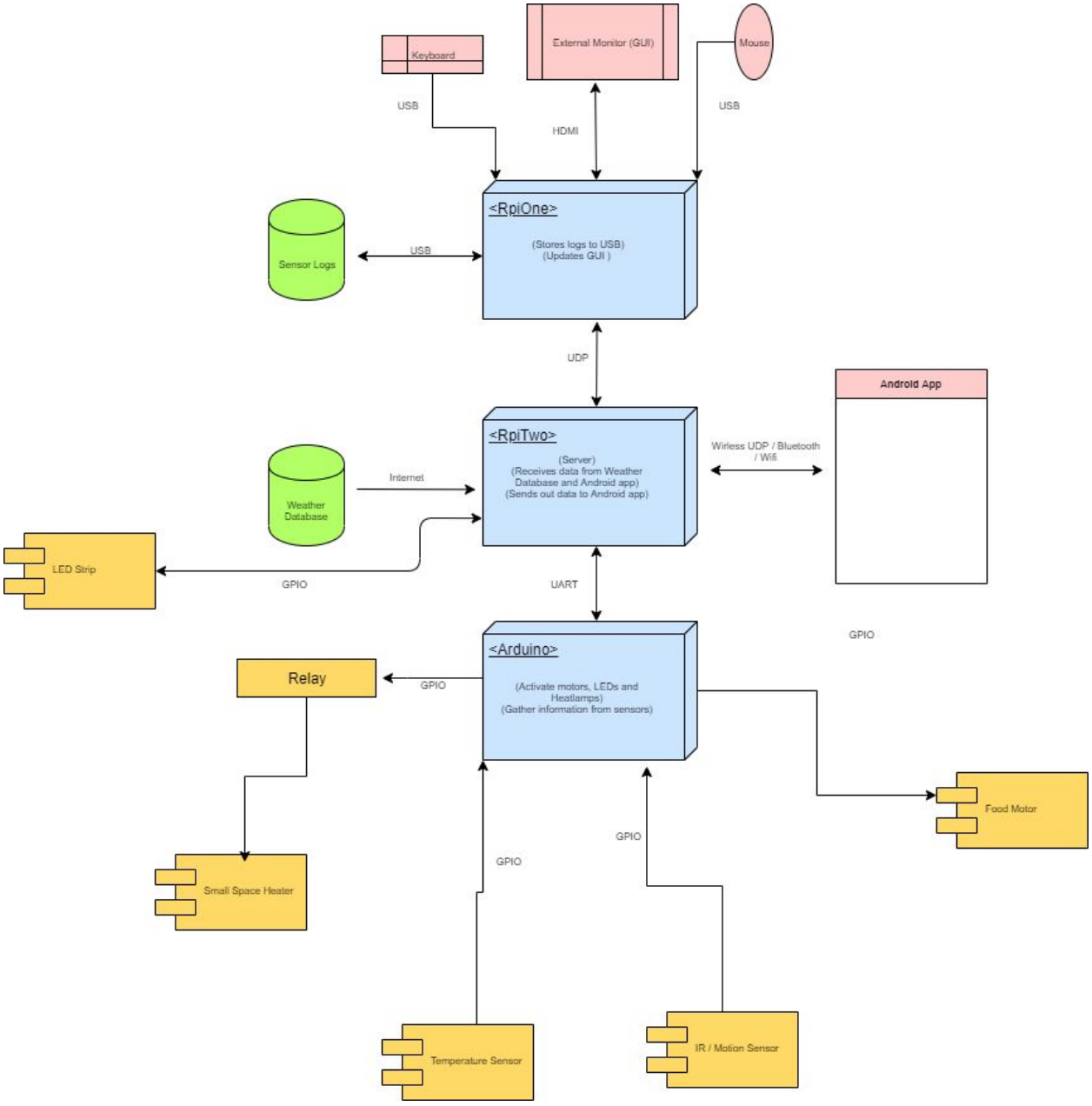


Figure (2.1: UML Deployment Diagram (ABOVE))

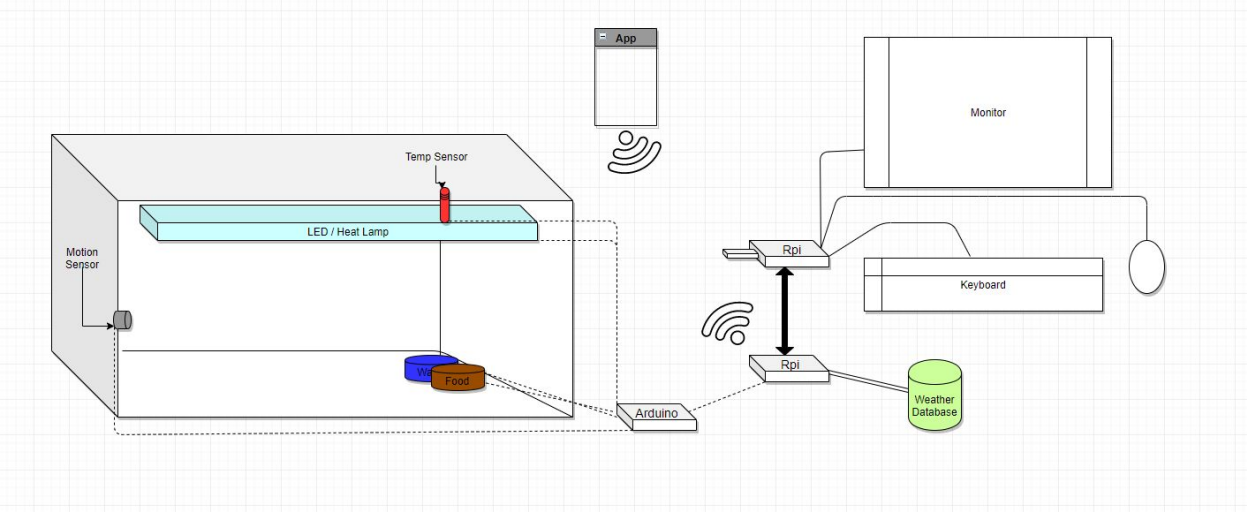


Figure (2.2: System Architecture Diagram (BELOW))

Communication Protocols (3.0):

Figure 3.1 is a Sequence Diagram for communication between the android app, external weather database, and Rpi2. The Rpi2 is the microcontroller that is connected wirelessly to the Android so it is the first line of contact. The user can check the temperature of any location in the world and it will be displayed in the app. If the user chooses they can set that as the temperature, or they can set it using a manual number. Each time a user completes an action the system returns a message confirming that their selection is complete because the user may not be in the same room as the tank and need to know if their action was completed.

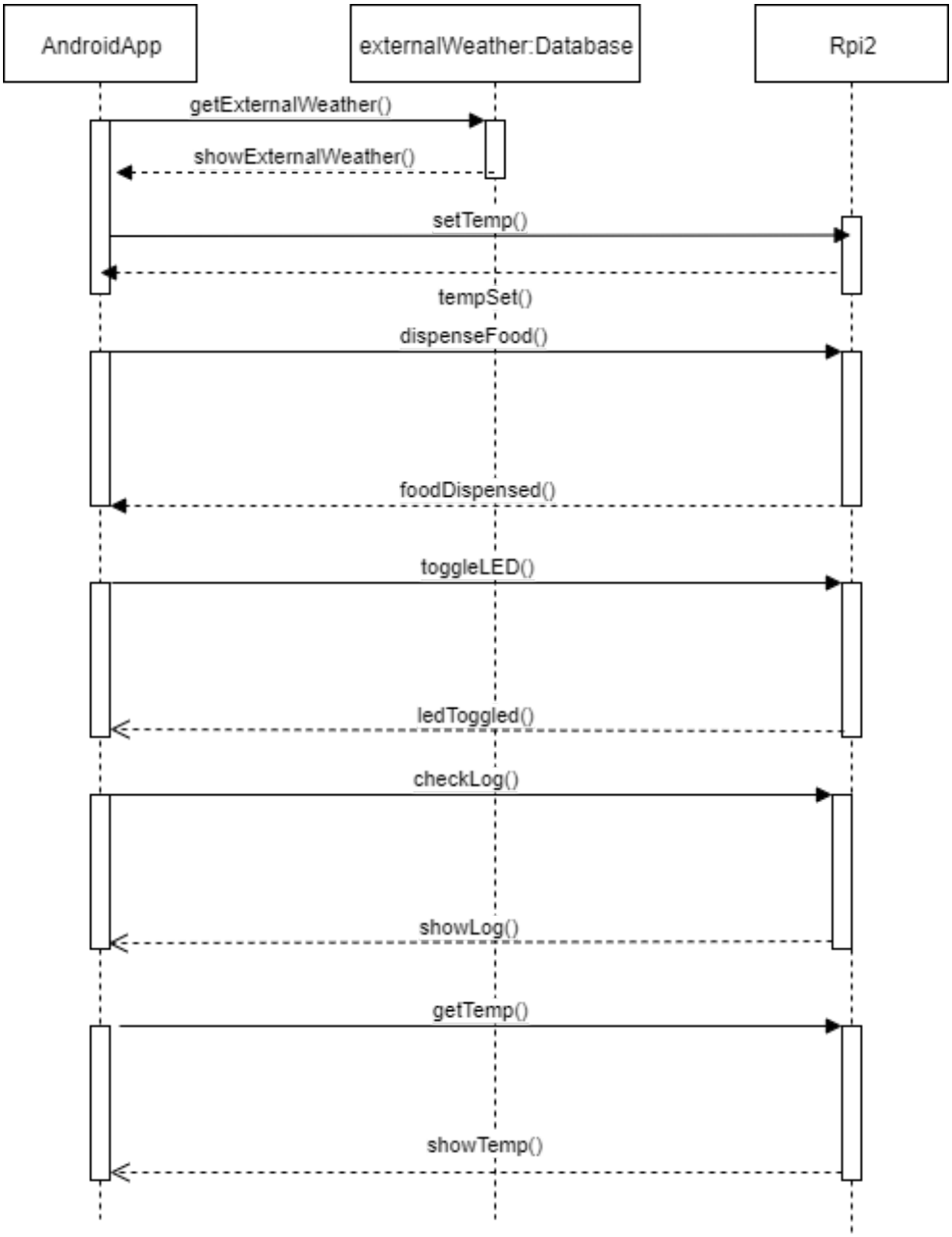


Figure (3.1: Sequence Diagram for App and Rpi2)

Method Name	Sender	Receiver	Message	Format
getExternalWeather	AndroidApp	Database	Request	SQL Query
showExternalWeather	Database	AndroidApp	Response	JSON file
setTemp	AndroidApp	Rpi2	Request	int
tempSet	Rpi2	AndroidApp	Response	Boolean
dispenseFood	AndroidApp	Rpi2	Request	int
foodDispensed	Rpi2	AndroidApp	Response	Boolean
toggleLED	AndroidApp	Rpi2	Request	Int (null?)
ledToggled	Rpi2	AndroidApp	Response	Boolean
checkLog	AndroidApp	Rpi2	Request	(null?)
showLog	Rpi2	AndroidApp	Response	Boolean
getTemp	AndroidApp	Rpi2	Request	(null?)
showTemp	Rpi2	AndroidApp	Response	Int
getMotion	AndroidApp	Rpi2	Request	(null?)
showMotion	Rpi2	AndroidApp	Response	Boolean

Table (3.1: Sequence Table for App and Rpi2)

Figure and table 3.2 show the communication between Rpi1, Rpi2, and the database.Rpi1 is connected to both the GUI and the USB Drive database. This means that it will need to take input as well as log values simultaneously. Whenever a person changes the temperature or feeds their pet it is sent into the database. The IR sensor logs its values every minute. The temperature sensor also logs its values every minute. Much like the android app, the GUI gets confirmation every time that a requested action is completed.

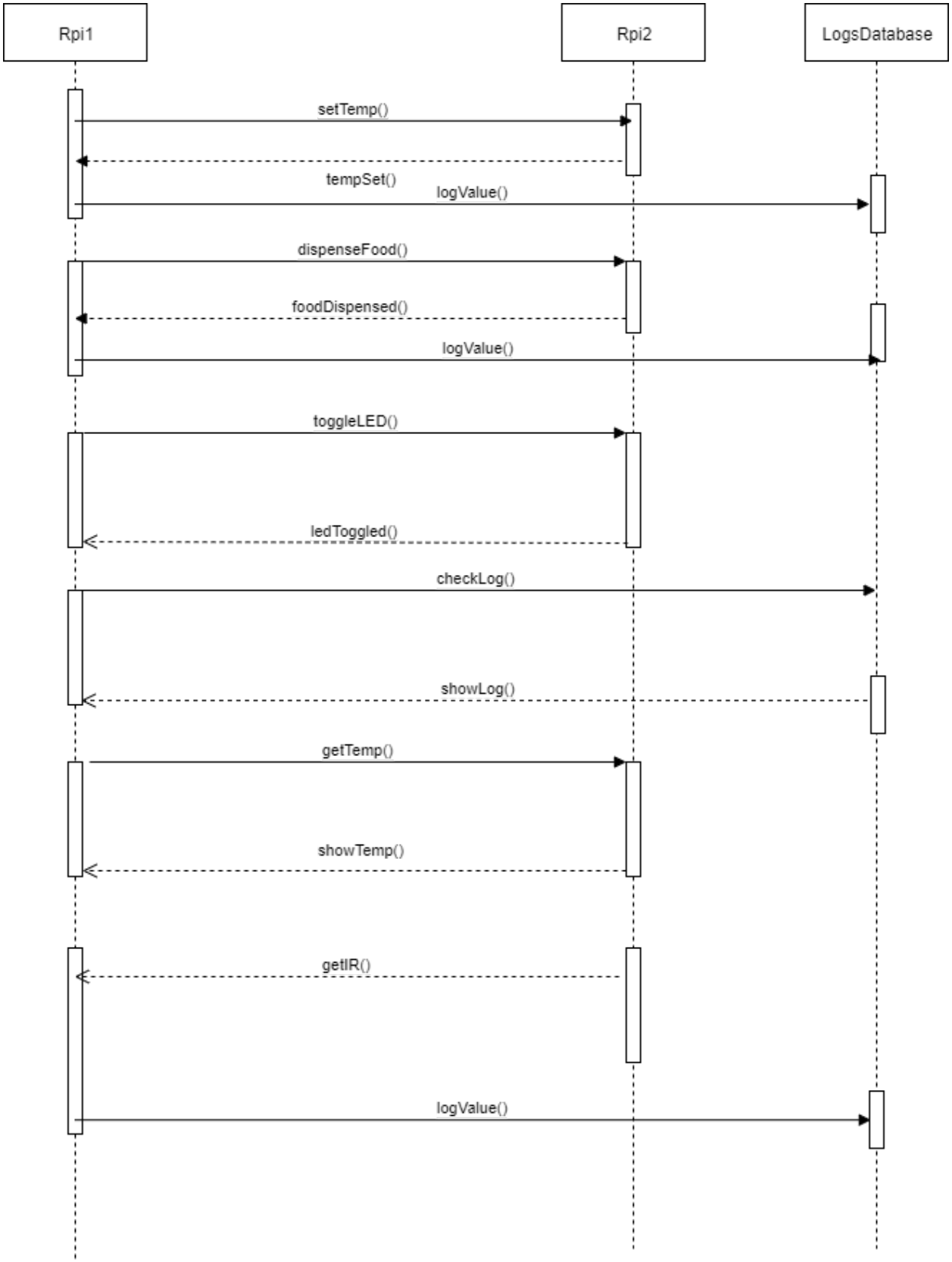
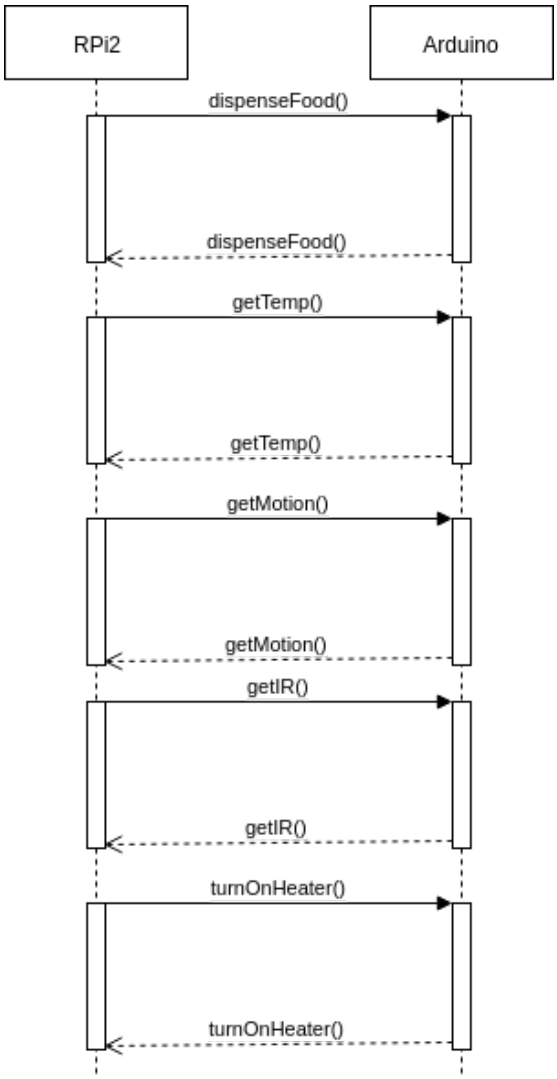


Figure (3.2: Sequence Diagram for Rpi1, Database, and Rpi2)

Method Name	Sender	Receiver	Message	Format
checkLog	RPi1	Database	Request	null
showLog	RPi2	Database	Response	[int, int, Time]
logValue	RPi1	Database	Request	[int, int, Time]
getTemp	RPi1	RPi2	Request	null
showTemp	RPi2	RPi1	Response	int
getMotion	RPi1	RPi2	Request	null
getIR	RPi1	RPi2	Request	null
dispenseFood	RPi1	RPi2	Request	int
foodDispensed	RPi2	RPi1	Response	Boolean
toggleLED	AndroidApp	Rpi2	Request	Int (null?)
ledToggled	Rpi2	AndroidApp	Response	Boolean

Table(3.2: Sequence Table for Rpi1, Database, and Rpi2)

Figure and table 3.3 deal with the methods that will be going through the rpi2 and arduino. All user inputs funnel through the Rpi2 and are relayed to the Arduino. The Ardunio must also send log values to the Rpi2 to send to other parts of the system. The Arduino must communicate with the sensors and motors to function when requested.



Figure(3.3: Sequence Diagram for Rpi2 and Arduino)

Method Name	Sender	Receiver	Message	Format
dispenseFood	RPi2	Arduino	Request	(null)
dispenseFood	Arduino	RPi2	Response	boolean
getTemp	RPi2	Arduino	Request	(null)
getTemp	Arduino	RPi2	Response	float
getMotion	RPi2	Arduino	Request	(null)
getMotion	Arduino	RPi2	Response	float
getIR	RPi2	Arduino	Request	float
getIR	Arduino	RPi2	Response	float
turnOnHeater	RPi2	Arduino	Request	boolean
turnOnHeater	Arduino	RPi2	Response	boolean

Table(3.3: Sequence Table for Rpi2 and Arduino)

Possible errors:

-If a message is sent (method called) while a process is already executing, the receiving node may never get the method call and will not execute it, leaving the node that originally called it with nothing returned.

-A method may be called and executed, but interrupted before being able to return a value, causing the node that called the method originally believing that it was never executed.

-UDP related issues

- Wrong Packet was sent
- Packet was never received
- Packet is empty

Each of our components that receives a packet will always send back a confirmation packet. So if the sender never get a confirmation packet from the receiver then something must be wrong and should send another one. Since UDP doesn't have to be sent in a specific order there is no need to worry about latency issues and packets being received in the wrong order. If the packet was empty or the wrong packet was received an error packet will be sent back to the sender and request another packet to be sent. This will be easy to implement because within both the GUI and the Android App, a method is always called and sent back to the user (which will either be the Rpi1 or Rpi2) to confirm that the action was completed. We can have our error/confirmation packets that are to be sent back to the sender in these methods.

-There is potential for race conditions if the user is inputting into the GUI and the Android App.

Database (4.0):

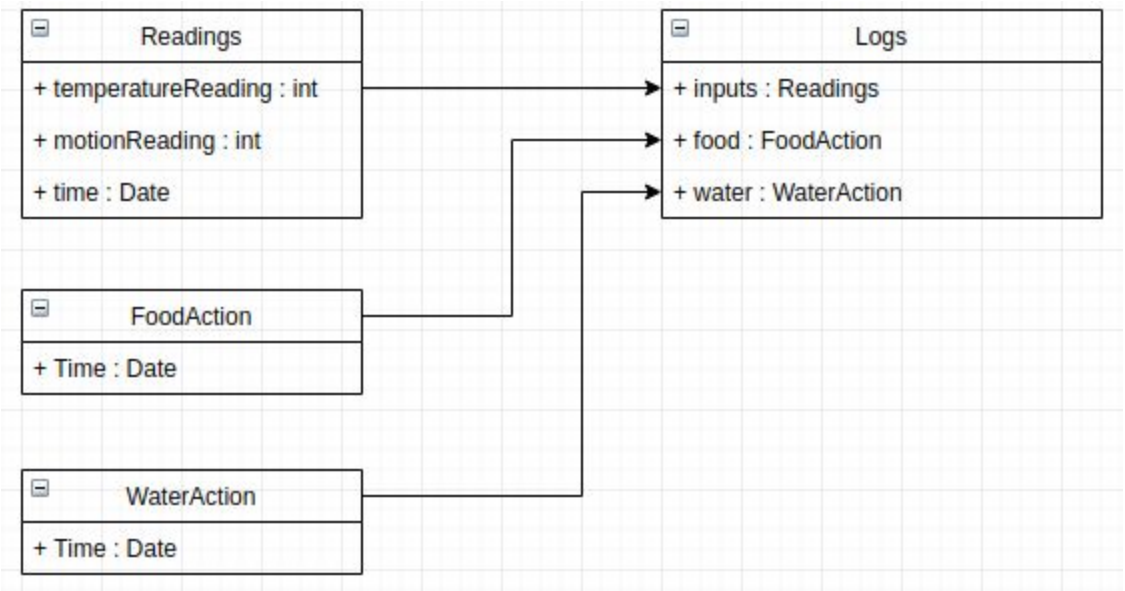


Figure (4.1: Database UML Diagram of the Local Database)

The Database schema pictured above consists of four tables. The main table that contains a record of all information is the Logs table. Its contents come from the 3 other tables. The readings table is used to store data obtained from the sensors which gather input, such as the temperature and motion sensor. The Arduino and RPi2 will take the inputs from these sensors every 60 seconds and record each sensor value along with the time they were obtained and store them into JSON files. The Rpi2 will send the files, via UDP, to the RPi1 which will store the JSON files into the readings section of the log table. The log will then be saved onto a connected USB drive. The FoodAction and WaterAction tables represent a record of when then actuators for dispensing food and water are used. When either is used, either manually by the user or automatically at a set time, the time of use is recorded in the appropriate table. The second database is a weather website that contains information on the current weather anywhere in the world. We will use the Raspberry Pi to communicate to the weather server to return packets of JSON files that contains the desired weather at the specified location.

Hardware Design (5.0):

The 5 circuits below depicts the main physical circuit that will interact with the tank. We concluded that the Arduino may not be able to handle all of the power and constant polling that is required for these circuits to function properly. Therefore we will have to split the load with one of the Pi's.

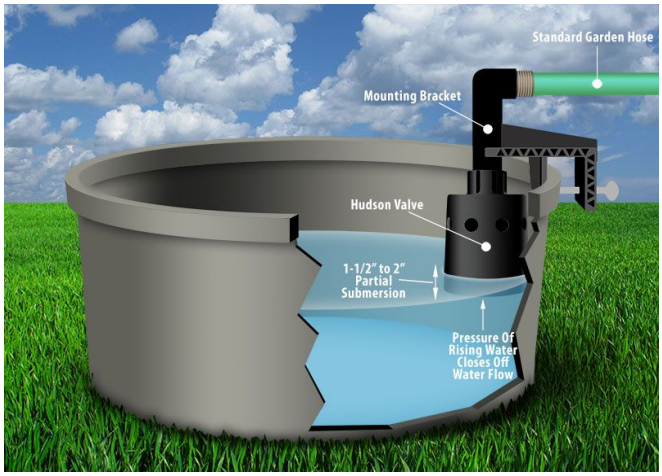


Figure (5.1, 5.2: Automatic Water Dispenser)

The water dispenser can be created using the same technique pictured above. If the proper level of submersion is found the pressure from the accessible water will cause the water to enter equilibrium state where no more water will come out of the bottle. We do not need a hudson valve because we will manually pour water into a large container so the constant pressure of a garden hose is not an issue. There is no need to put a restriction on how much water the pet drinks so there is no need to control it automatically.

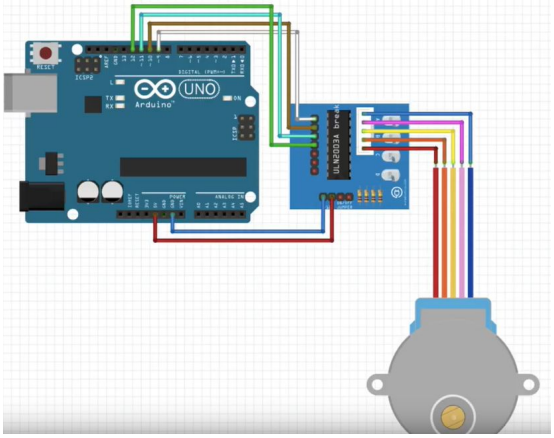


Figure (5.3: DC Motor for Food Dispenser)

The motor circuit is for dispensing food. The motor circuit may be instead connected to one of the Pi's if required. The current design idea for the food dispensing motor is as follows. The motor will be connected to a threaded object (like a bolt). This will be threaded through another cylindrical threaded object (like a nut). This will be connected to a platform that will act as the bottom portion of a container. The container will be filled with pet food. As the motor is activated it will rotate the threaded object its connected to slightly. This will raise the platform causing food to spill over the top of the container and into the food bowl for the pet to eat. The amount of food can be altered based on how long the motor is activated. The motor and motor driver being used is from the Kuman Kits supplied in SYSC 3010.

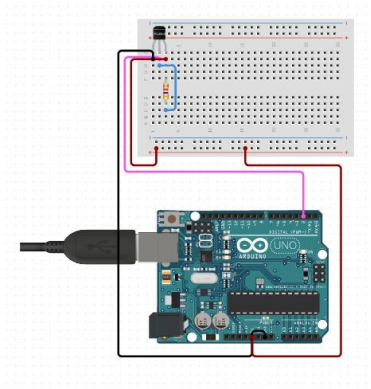


Figure (5.4: Temperature Sensor Circuit)

The temperature circuit will read the temperature from within the tank. The sensor being used is the DHT11.

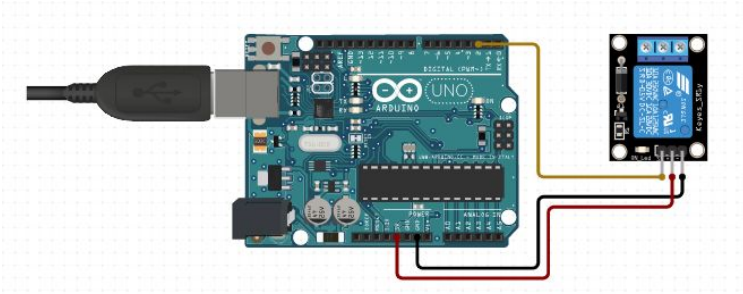


Figure (5.5: Heater Circuit)

The relay is a small circuit that can act like an external switch for regular electronics that use the 120V outlets. The relay and Arduino cannot directly control the 120 volts that are coming in however it can change the current which can then turn the electronic on / off. A very small space heater will be used in our system design to

control the heat within the tank. If the temperature sensor reads that it is too hot, (by checking the JSON logs from the last minute from within the database) the relay will cut the current to the space heater. Likewise, if the tank is too cold the relay will allow the space heater to turn on.

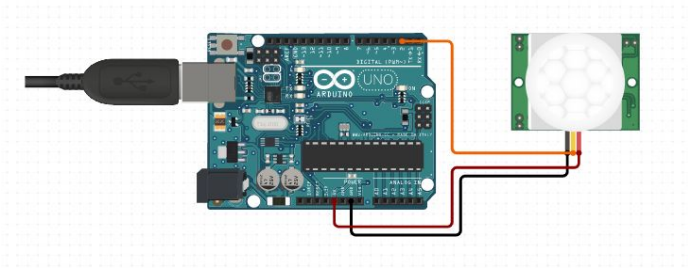


Figure (5.6: IR Sensor)

The IR sensor will detect the motion of the pet. The sensor used is the OSOYOO purchased from amazon.

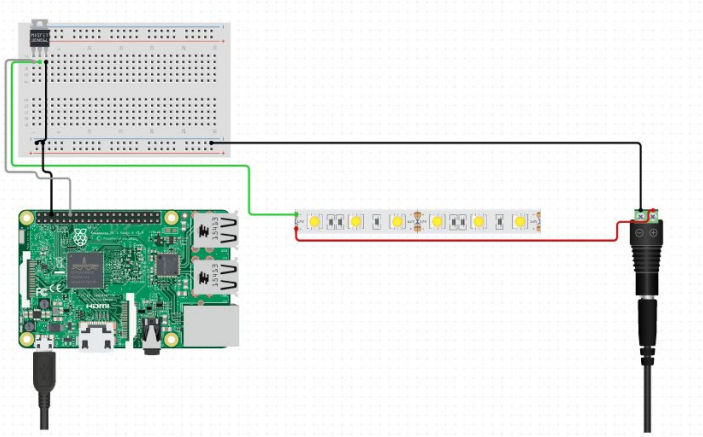


Figure (5.7:LED Circuit)

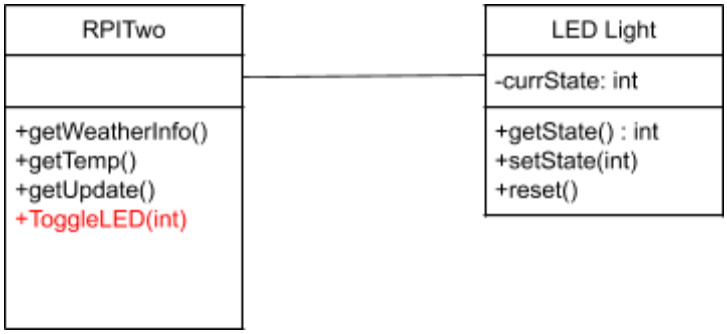
The LED strip will provide light for the pet and will be connected to a Raspberry Pi. In an ideal case the space heater and LED strip could be replaced with a heat lamp. Reptiles require light that gives off UVA and UVB lighting.

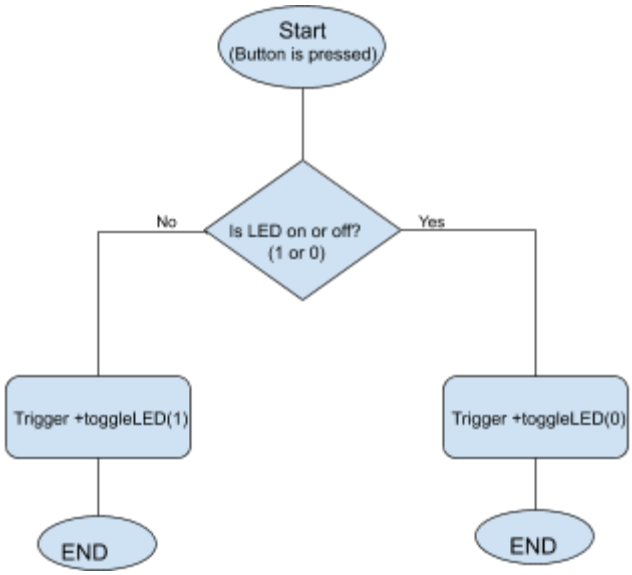
Each circuit shown below has a unique testing code. The website www.circuito.io has a tool that creates test programs based on the hardware that you add to the simulated board. This made it easy to create the basic test code that will be used each time the system is booted up.

Software Design (6.0):

LED Strip:

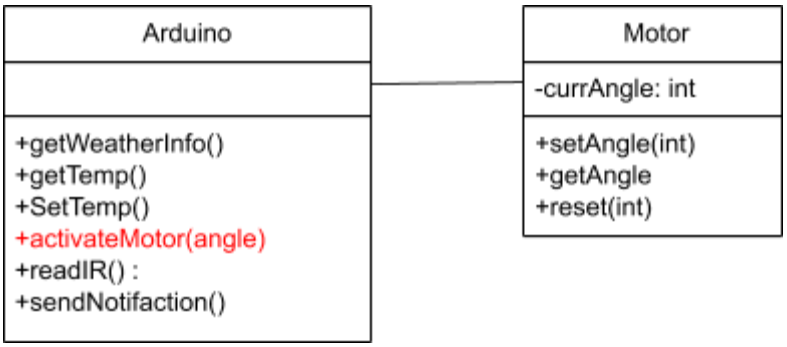
The LED Strip will be controlled by a Button on RpiTwo also controlled from the android app/GUI where it will be remotely turned on/off.

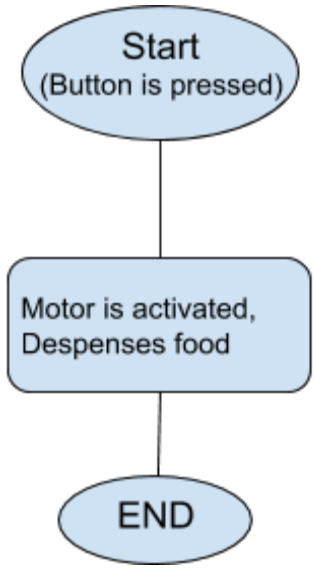




Food Motor:

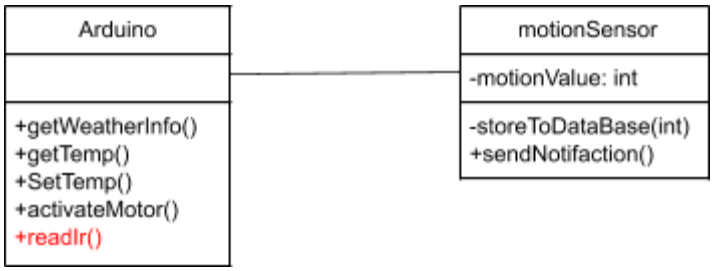
The motor used to dispense the food the is activated by the arduino and dispenses food through embedded Software (C) and the hardware implementation.

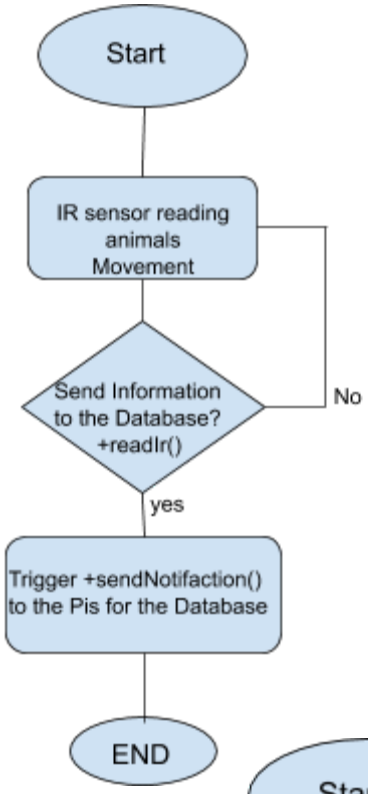




IR/ Motion Sensor

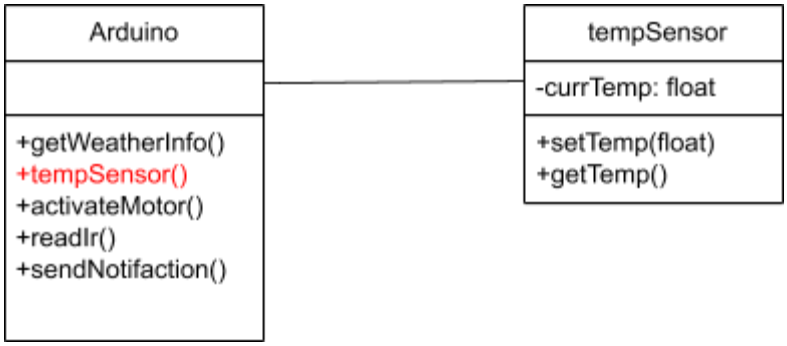
The IR Sensor is implemented by the Arduino and polls every minute to track the movement of the animal in the enclosure. The arduino then sends the information to the Pis to input the movement into the databases.

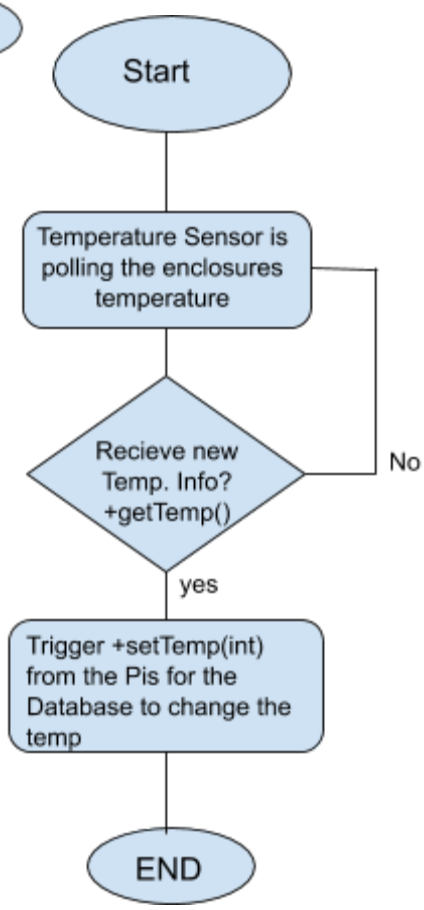




Temperature Sensor

The temperature sensor will poll the temperature inside the enclosure every minute and depending on if the user wants to change the temp of the enclosure it will take the information of the current temperature in the desired location stored in the databases and will adjust the temperature on the enclosure to the specified temperature with +setTemp(int).





References (7.0)

Figure (5.1)
https://i.pinimg.com/originals/1c/30/1c/1c301c7c94673cb9f1ab5eea70f420fc.jpg

How to create pet food feeder
https://mad-science.wonderhowto.com/how-to/build-vacation-pet-feeder-with-diy-linear-actuator-0134695/

Figure (5.2)
https://www.plumbingsupply.com/images/how-to-install-an-automatic-pet-watering-system.jpg

DC Motor Test Program
https://github.com/NikodemBartnik/ArduinoTutorials/blob/master/28BYJ-48/28BYJ-48.ino

Other section 5 figures made by ChaseB on Circuito
https://circuito.io