Wireless Backyard Chicken Coop

Team: sdmay20-55

Client/Advisor: Dr. Andrew Bolstad

Team: Alex Goldyuk, Amine Hocine, Lexi Reicks

Ryan Toepfer, Alexander Lewozcko, Armando Vila-Colon

Contact: [sdmay20-55@iastate.edu](mailto:sdmay20-55@iastate.edu)

Website: <https://sdmay20-55.sd.ece.iastate.edu/>

# Executive Summary

## Engineering Standards & Practices Used

P16085 - ISO/IEC/IEEE International Draft Standard - Systems and Software Engineering - Life Cycle Processes - Risk Management.

IEEE 2430-2019 - IEEE Trial-Use Standard for Software Non-Functional Sizing Measurements.

## Summary of Requirements

#### Economical Requirements

* Total budget for the project shall fall within the allotted $200 basic budget.

#### Environmental Requirements

* Device shall be able to operate at temperature ranges between -10 degrees Fahrenheit to 120 degrees Fahrenheit.
* Device shall be shielded by a food-grade composite in the event of chicken absorption.
* Device shall be able to withstand interference from the chickens.

#### UI Requirements

* User Interface shall allow for unit personalization on weight, volume, and temperature readings.
* User Interface shall allow for notification personalization when sensor measurement thresholds are attained.
* User Interface shall display accurate real-time data to within 30 minutes of server updates.
* User Interface shall employ user authentication to verify login credentials.

#### Software Requirements

* Software shall employ user authentication to verify login credentials.
* Software shall be updated by the sensors’ data at a period of 30 minutes.
* Software shall be available for access at all times.
* Software shall be able to push notifications when personalized thresholds have been reached.

#### Hardware Requirements

* Device shall be able to operate at temperature ranges between -10 degrees Fahrenheit to 120 degrees Fahrenheit.
* Battery life shall be able to sustain the device for a period of one week.
* Device shall be able to accurately (to within .50 lb) read the weight sensor data and update the database accordingly.
* Device shall be able to accurately (to within 1 degree) read the temperature sensor data and update the database accordingly.
* Device shall be able to wirelessly communicate with the database, using the client’s home wifi connection.

## Applicable Courses from Iowa State University Curriculum

E E 201: Electric Circuits, E E 230: Electronic Circuits and Systems  
COM S 309: Software Development Practices,   
COM S 227: Object-oriented Programming,   
CPRE 185: Intro to Problem Solving, CPRE 288: Embedded Systems

## 

## Table of Contents

[**Executive Summary**](#_tc0ujm4nz88r) **1**

[Engineering Standards & Practices Used](#_acvjldcwnhf4) 1

[Summary of Requirements](#_uqawfotr1lfl) 1

[Applicable Courses from Iowa State University Curriculum](#_4mn3toedfpyv) 2

[Table of Contents](#_ydovw4ctcsap) 3

[**1 Introduction**](#_c4i2psealxw3) **7**

1.1 [Acknowledgement](#_tyjcwt) 7

1.2 [Problem and Project Statement](#_3dy6vkm) 7

1.3 [Operational Environment](#_1t3h5sf) 8

1.4 [Requirements](#_4d34og8) 8

1.5 [Intended Users and Uses](#_2s8eyo1) 9

1.6 [Assumptions and Limitations](#_17dp8vu) 9

1.7 [Expected End Product and Deliverables](#_3rdcrjn) 10

1.7.1 [Stretch Deliverables](#_yefxvzdatmjh) 11

[**2. Specifications and Analysis**](#_nmvtangkf91x) **11**

2.1 [Proposed Design](#_lnxbz9) 11

2.2 [Design Analysis](#_m34yv9ccpt) 12

2.3 [Development Process](#_pxlcp65w5w7b) 13

2.4 [Design Plan](#_44sinio) 13

2.4.1 [Use Cases](#_qmge5vw1sulm) 13

[**3. Statement of Work**](#_3uv5i1slzmib) **14**

[3.1 Previous Work And Literature](#_yzueu52mehha) 14

[3.2 Technology Considerations](#_8d8m47c0wlxr) 15

[3.3 Task Decomposition](#_6zl6b0dcyxui) 17

3.3.1 [Administrative and Organizational Duties](#_xys20d2tii8w) 18

3.3.2 [Device Selection](#_5w2h35k9vp02) 18

3.3.3 [Weight Sensor](#_rkwc47g6qv4h) 18

3.3.4 [Temperature Sensor](#_d0khbv1wtizn) 19

3.3.5 [Client Application](#_ik4smqqep4es) 20

3.3.6 [Server/Database](#_u3z673j7bogo) 20

3.3.7 [Power](#_xjh9e1fjhf5x) 20

3.3.8 [Physical Housing](#_jtl3i2vvmzk7) 21

[3.4 Possible Risks and Risk Management](#_xc1ngg8mton) 22

3.4.1 [Weight Sensor](#_3hd6ekqbcsmz) 22

3.4.2 [Anomaly Tracking](#_oiotitbjyybo) 22

[3.5 Project Proposed Milestones and Evaluation Criteria](#_io4o7w7edycj) 23

[3.6 Project Tracking Procedures](#_bh9szkbiofsp) 24

[3.7 Expected Results and Validation](#_ra3wsho2qgxw) 25

[**4. Project Timeline, Estimated Resources, and Challenges**](#_s6see7egtw3x) **26**

[4.1 Project Timeline](#_49x2ik5) 26

[4.2 Feasibility Assessment](#_5j7y33ijnmqx) 27

[4.3 Personnel Effort Requirements](#_147n2zr) 28

[4.4 Financial Requirements](#_23ckvvd) 30

[**5. Testing and Implementation**](#_bu0dvgij9cqa) **30**

5.1 [Interface Specification](#_32hioqz) 31

5.2 [Hardware and Software](#_1hmsyys) 31

5.3 [Functional Testing](#_41mghml) 31

5.4 [Non-Functional Testing](#_2grqrue) 32

5.5 [Process](#_vx1227) 32

5.6 [Software Results](#_3fwokq0) 33

[5.6.1 Web Application](#_qdg7qlibqr4m) 33

[5.6.2 Notifications](#_n0wg5wspbwyp) 33

[5.6.3 Server](#_gw2yopske69p) 34

[5.6.4 Embedded System](#_nwf69dc6mi86) 34

5.7 [Hardware Results](#_pmd04g8o705w) 34

[**6. Closing Material**](#_8xxzta698610) **37**

[6.1 Conclusion](#_masivvgyf756) 37

[6.2 Appendices](#_z864vsn6xpwr) 38

[Figure 6.2.1 Measurement Collection](#_4poay4tmumwl) 38

[Figure 6.2.2 Users Collection](#_vtq2gp3jax0z) 39

[Figure 6.2.3 Windowed Measurement Collection](#_adcl2vdvm0pq) 39

# 

# 1 Introduction

## Acknowledgement

This report is compiled from the development completed for the Wireless Backyard Chicken Coop project. We are grateful for the number of contributors, colleagues, and all third parties who collaborated in this project’s development. We’d also like to sincerely thank our client Dr. Andrew Bolstad for the encouragement he extended to us throughout this implementation. We’d like to further acknowledge the support that was extended from our Iowa State University’s Department of Engineering staff members.

## Problem and Project Statement

The hobbyist chicken raising market has deficiencies in terms of monitoring chicken behavior and resource usage. Though there are a multitude of devices to aide in the raising of chickens, these devices are typically aimed at more commercial endeavors, with monitoring technology not being available at the hobbyist level. Furthermore, the available commercial devices typically have a rather complex and expensive design that does not support modular functionality to monitor only specifically desired statistics. As such, our client approached our team with the aim to address this gap in functionality and create a set of devices that would allow a hobbyist to monitor the resource usage of their chickens.

The primary goals for the device we have fabricated was to monitor three specific attributes of the client’s chicken coop and allow for a user friendly monitoring system.   
The statistics that the client desired to monitor are as follows:

* The remaining feed available
* The remaining water available
* The temperature of the water

Furthermore, the client desires a cost-effective solution to the problem that must fall within the allocated senior design budget of 200 dollars. An additional key focal point of the solution was to provide the client with a very simple interface to wirelessly check the aforementioned levels within the coop and personalize monitoring statistics.

Though we are currently intending to design this product with solely the client’s functionality in mind, we do believe that the modular design of the device could be applied at a market level for various chicken hobbyists.

## Operational Environment

The final product will be deployed within the client’s chicken coop in Ames, Iowa. Due to the enclosure of the chicken coop, the device will not be directly exposed to precipitation, though it will need to sustain extreme temperature ranges. The device will also need to withstand interference from the chickens within the coop as well as being shielded by a food-grade composite in case of chicken absorption.

The device is also being designed without a wired power source in mind and will need to operate solely on battery power. This period of operation will need to be at least a week to minimize the client’s direct involvement with the final product.

## Requirements

#### Economical Requirements

* Total budget for the project shall fall within the allocated $200 basic budget.

#### Environmental Requirements

* Device shall be able to operate at temperature ranges between -10 degrees Fahrenheit to 120 degrees Fahrenheit.
* Device shall be shielded by a food-grade composite in the event of chicken absorption.
* Device shall be able to withstand interference from the chickens.

#### UI Requirements

* User Interface shall allow for unit personalization on weight, volume, and temperature readings.
* User Interface shall allow for notification personalization when sensor measurement thresholds are attained.
* User Interface shall display accurate real-time data to within 30 minutes of server updates.
* User Interface shall employ user authentication to verify login credentials.

#### Software Requirements

* Software shall employ user authentication to verify login credentials.
* Software shall be updated by the sensors’ data at a period of 30 minutes.
* Software shall be available for access at all times.
* Software shall be able to push notifications when personalized thresholds have been reached.

#### Hardware Requirements

* Device shall be able to operate at temperature ranges between -10 degrees Fahrenheit to 120 degrees Fahrenheit.
* Battery life shall be able to sustain the device for a period of one week.
* Device shall be able to accurately (to within .50 lb) read the weight sensor data and update the database accordingly.
* Device shall be able to accurately (to within 1 degree) read the temperature sensor data and update the database accordingly.
* Device shall be able to wirelessly communicate with the database, using the client’s home wifi connection.

## Intended Users and Uses

The key demographic of our product is the home chicken hobbyist who wishes to better monitor the resource usage of their chickens. Our aim is to provide a very simple interface for this monitoring as we do not expect the user to have any solution-specific software or hardware experience. This product is being developed in a modular fashion though our release version will only be able to monitor the weight of the feed and water, as well as the temperature of the water itself.

## Assumptions and Limitations

|  |  |
| --- | --- |
| Assumptions and Limitations | |
| Assumptions | Limitations |
| The project shall be completed at the end of March. | Hardware and software functionality may have limited peer support. |
| Every team member must perform throughout the entirety of the development process. | Individual workload on features may be over-allocated. |
| Some hardware components may be expensive. | Cost on certain components may have to be mitigated to fall within budgets. |
| UI should be readily comprehensible by all users. | Certain statistics may have to be simplified. |
| Hardware should be able to withstand weather extremes. | More fragile components may need to be shielded to achieve durability. |
| Client application should display data in relative real-time. | Battery drain may become an issue due to frequent updating. |
| The end product will be used by chicken hobbyists. | Final system will focus on one group of livestock; chickens. |
| Hardware should be able to support wireless communication. | Sharing data wirelessly consumes battery. |

*Table 1. Assumptions and Limitations*

## Expected End Product and Deliverables

#### Temperature Sensor Reading

The temperature sensor system will monitor the temperature of the water inside the chicken coop. This temperature data will be read in from the SparkFun microcontroller using the associated temperature sensor. Once the data has been read, it will be communicated to the central database at an interval of 30 minutes over the available WIFI connection. This dataset can then be fetched by the user application to display to the client, or further analyzed to create use statistics and graphs.

#### Weight Reading

The weight sensor system will monitor the weight of the water and feed inside the chicken coop. This weight data will be read in from the SparkFun microcontroller using the associated weight sensors. Once the data has been read, it will be communicated to the central database at an interval of 30 minutes over the available WIFI connection. This dataset can then be fetched by the user application to display to the client, or further analyzed to create use statistics and graphs.

#### Web Application

The web application will be used to fetch sensor data from the database running on the server and either display this data directly to the client, or perform analysis to create graphs and projections of the data. This web application must be always available to properly monitor the chicken coop. It must also be user configurable to allow for the personalization of certain monitoring metrics.

#### Remote Connection to Server from Microcontroller

All microcontrollers must be able to send the read sensor data to the server. Each sensor within the chicken coop will have a dedicated task to fulfill being; monitoring the weight of the feed, monitoring the weight of the water, and monitoring the temperature of the water. The populated server data will then be communicated to the web application, which will allow the client to view the data acquired from the microcontroller sensors.

### 1.7.1 Stretch Deliverables

#### Supplemental Solar Power

A major risk of the project is that the batteries will not provide enough power to sustain the device for a period of at least one week. A stretch feature we have identified to mitigate this risk would be to install a supplemental solar panel to trickle charge the device while in use. In the event of periods with greater available sun exposure, the client would also not need to change the batteries of the device. This solar panel would be installed within the client’s backyard, in an area with the least foliage coverage. A key block to completing this stretch feature would be the price of the solar panel itself, which may be prohibitively expensive in terms of the project's budget.

#### Image Processing Implementation

An advanced stretch feature of the project would be to install a low-light camera to visually monitor various statistics within the chicken coop. This camera, in tandem with image processing, could be used to identify the number of chickens within the coop as well as the number of eggs laid. One such tool that we could use to run this image processing would be OpenCV, running directly on the microcontroller, to recognize the chickens and the eggs. A major risk to this stretch features completion would be the battery power necessary to run this image processing, as well as the available computing power on the microcontroller itself.

# 2. Specifications and Analysis

## Proposed Design

Our proposed design consists of hardware to read sensor data from the deployed microcontroller and send this data to a database running on the server. A web application will also be created to allow the client to view this data in relative real-time. To satisfy the base requirements, there will be three sensors in total, consisting of two weight sensors and one temperature sensor. Each weight sensor would be respectively used to monitor the weight of the water and the weight of the feed. The temperature sensor would then be used to specifically monitor the temperature of the water, ensuring that it is above the freezing point. The weight sensors will be attached to the feed and water containers and hang from the ceiling, indicating how much feed and water is available and if the containers need to be refilled. This information will be used to display graphs and make predictions about the feed and water use. The web application will mainly need to send alerts if the feed or water falls below a certain configurable threshold, or if the water is reaching the freezing point.

These sensors connect to a microcontroller that will periodically send sensor data to the database. Before choosing a microcontroller or any hardware for that matter, a few key requirements must be kept in consideration. Because this device is going to be in a predominately unheated chicken coop, it is essential that it will be able to handle the wide range of temperatures in Iowa. The hardware will also need to withstand potential interference from the chickens. Since this hardware is in contact with livestock, it must also be shielded by a food-grade composite.

The front-end of the device needs the following basic functionalities:

* Accessing sensor data in a human-readable format.
* Querying the sensor data for a specific timespan or sensor type.
* Graphs depicting the usage of feed and water.

This web application must also have basic security, utilizing both user registration and authentication. This application will also be accessible from mobile devices by going to the website directly.

## Design Analysis

The majority of the first semester was spent researching and planning. Throughout our development process, we have made some moderate changes to our initial project plan. The hardware has taken priority as it will go on to involve the other elements of the design. One of the biggest changes to the initial project plan was the requirement to have the hardware battery powered. This requirement brought many considerations to light:

* How much battery life do we need?
* Do we want the battery to be rechargeable?
* How often to send data?
* How can we conserve battery power without sacrificing performance?

With these requirements in mind, we started choosing our hardware.

After performing research on several different microcontrollers, we selected the SparkFun “ESP32 Thing” for the microcontroller that will be used in our device. This model was selected for multiple reasons which include support of the existing Arduino IDE, on-board wireless capabilities, on-board battery charging capabilities, low power consumption, small footprint, and low cost.

For our sensor suite, we evaluated multiple different methods of measuring weight, but determined that Degraw 40 was the only economically feasible solution, as all of the other options significantly exceeded the project budget. We are still evaluating multiple choices of temperature sensors, which include the sensor built into the microcontroller itself, and a DS18B20 sensor package, which allows for remote measurement of temperature of up to 6 feet away.

For our database, we decided on MongoDB. MongoDB is a document database that requires little set up, has an easy schema design, and is suitable for projects that require little to no relations. After choosing our database we wanted to choose a backend language/framework that is readily compatible with MongoDB. Flask is very lightweight and easy to set up, being an ideal choice for our purposes. Choosing python will also allow us to add our stretch goals in a modular and straightforward fashion.

## Development Process

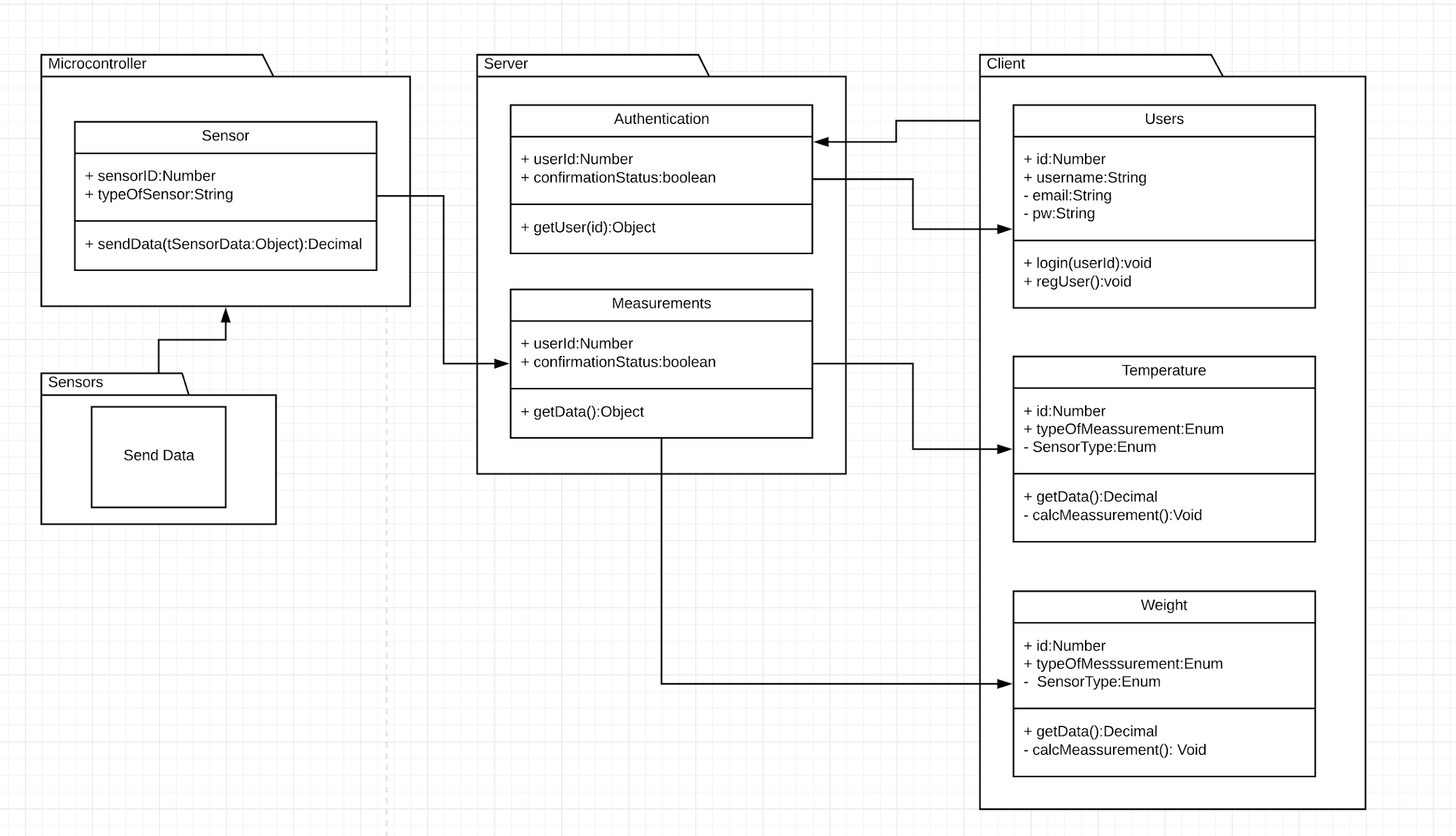
For our project, we are following the Waterfall development process. We decided on this development process because the requirements are fairly set in stone, so once the hardware is done there won’t be any new features we need to add. Our project is fairly dependent on having all of the parts done at the same pace. For the Waterfall process to be successful we believe that the requirements and design phases need to be completed thoroughly. If the design is solid and all of the requirements are met, then building our project and testing should go smoothly.

## Design Plan

### 2.4.1 Use Cases

1. As a user, I need to receive alerts if feed or water falls below a certain configurable threshold.
2. As a user, I need to receive alerts if the water reaches the freezing point.
3. As a user, I need to be able to view current feed and water levels.
4. As a user, I need to be able to view the current temperature of water.
5. As a user, I need to be able to view historical data about feed and water usage.
6. As a user, I need the system to be battery powered.
7. As a user, I would like to see predictions about how often feed and water need to be refilled.

Our modules are discussed in the diagram below.



*Figure 1. Module Decomposition*

# 3. Statement of Work

## 3.1 Previous Work And Literature

Throughout our cross examination to identify existing commercial off-the-shelf (COTS) technologies that fulfill the requirements of this project, we were unable to find any comparable products that can satisfy the problem statement. There are currently no products available on the market that effectively serve as a monitoring system for a hobbyist chicken coop. While there are clear examples of consumers satisfying this need for themselves, the market as a whole has a deficiency for a product of this function.

While we are developing a product that cannot pull from existing chicken coop solutions, there are more general products that we believe can be applied to the project. These technologies would be used for subsystem functionality that can be integrated to the holistic design.

We will be seeking to integrate the following existing technologies in our final product:

* high-density battery packs
* hanging weight sensors
* temperature sensors
* microcontrollers
* solar panels

Though commercial and readily available, these technologies fulfill such basic functions that virtually no considerations had to be made for project-specific applicability. Instead, considerations were largely made to satisfy both environmental and power requirements. This played its most significant role when choosing the microcontroller board, as we had to guarantee it would endure the most extreme ranges of Iowa weather.

As a result of the low-level technologies that we are working to integrate into our design, our final product should be unlike any commercially available hobbyist chicken coop solution. Not only does this necessitate a higher degree of ingenuity than projects that have a solid baseline to start with, it may serve to satisfy a legitimate consumer need that is not currently met.

Cross-examination research was conducted with the following resources:

* Popular online shopping outlets
  + Amazon
  + Sparkfun
* Chicken enthusiast websites
  + thehappychickencoop
  + raising-happy-chickens
  + thegardencoop

## 3.2 Technology Considerations

As stated in Section 3.1, the largest considerations our team made in terms of integratable technologies were in regards to the environmental and power requirements that we are under.

All COTS technologies that we select for our inclusion in the project must adhere to the following specifications:

* Must sustain a temperature down to -10 degrees fahrenheit
* Must sustain a temperature up to 120 degrees fahrenheit
* Exposed materials must be food-safe
* Must be partially moisture resistant
* Must be resistant to chicken interference

These restrictions played a huge role in selecting our microcontroller board as it is the most sensitive COTS component that we must integrate into our product.

We were then able to use this list of criteria to make the selection of our board from the following table of options:

|  |  |  |
| --- | --- | --- |
| Microcontroller Board Comparison | | |
|  | Advantages | Disadvantages |
| SparkFun ESP32 Thing | Built in WIFI and Bluetooth  Built in LiPo Battery Chip  Temp range of -13F to +140F  Arduino IDE  $21.95 | No battery pack included |
| Arduino Nano 33 IoT | Arduino Product  Built in WIFI  Arduino IDE  SMT compatible  $18.00 | No built in power regulator  No built in charger  Only accepts 3.3 volts |
| Raspberry Pi Zero W | Built in WIFI  Highest CPU power  Highest storage  Display driver  $10.00 | Very high power draw  Sensor compatibility  More closed platform |

*Table 2. Microcontroller Board Comparison*

We then made the decision to select the SparkFun board as it satisfied all of our basic requirements on the product. The most essential of these requirements being the clearly listed temperature thresholds of the SparkFun board.

An additional technical consideration our team had to go through was the housing for our final design, of which we identified no existing COTS products that could be reused for our purpose. Due to the unique nature of our design, we would need to create a custom housing that would provide the moisture and chicken resistance necessary.

After exploring the potential routes we could go down to create a custom product housing, we decided on creating a CAD model and using the university’s CNC mill to create our design. This solution was selected as it will be free of cost to our client and be able to conform to all of our physical requirements. By using a milled stainless steel housing, we are providing an impact and moisture resistant solution that is food safe.

The advantages we identified of using the CNC milled design over a traditional 3D printed design are product longevity and moisture retention. A 3D printed housing will not be as durable as a milled steel housing, especially when considering extreme temperature fluctuation. Seeing as we will not be continuing support of this project post-graduation, the longevity of the design is tantamount to our client. Another byproduct of using standard high-grade printing plastic, is the amount of moisture the material retains. As plastic provides a semi-porous surface, it may retain moisture for a far longer time period than steel. This has the potential to damage our internal components within the housing and essentially ending the project implementation.

Below is a prototype model of the CNC housing we will use in our final product.

  
*Figure 2: Prototype CNC Housing*

## 

## 3.3 Task Decomposition

We decided to decompose the problem statement by deliverables on a timeline basis, rather than on a per-feature basis. This was done to leverage the amount of parallel work we can have ongoing at any given time as well as ensure that we would adhere to the final deadline. Our team has a strong variety among members that correlates well with the diverse range of implementations necessary for the project. Therefore our major organizational goal we had when we approached this problem was to allow for hardware, software, and systems work to be ongoing simultaneously throughout the project. As such, stated deliverables (as can be seen in the schedule) offer progress in each individual feature of the project.

Despite decomposition being done on a scheduled basis, we have identified separate aspects of the project that must be completed by final implementation:

* Administrative and Organizational Duties
* Device Selection
* Weight Sensor
* Temperature Sensor
* Client Application
* Server/Database
* Power
* Physical Housing

### 3.3.1 Administrative and Organizational Duties

The administrative and organizational duties of the project cover the following sub-tasks:

* Documents required for 491
* Project Documentation
* Meeting Minutes
* Requirements
* Scheduling
* Planning

These tasks are an ongoing aspect to the class and are a shared responsibility across all team members. Most of the organizational duties to the project have been completed now that the project structure is firmly in place, with future work being done as it is assigned.

### 3.3.2 Device Selection

This was the initial phase to starting the product implementation and required us to research potential COTS technologies that we can integrate with our product. This work has largely concluded as we have made all major device selections and have tested for compatibility.

### 3.3.3 Weight Sensor

The weight sensor is one of the major features to our project. In our current design, we will be able to use two identical weight sensors in order to track both the feed and water capacity in the chicken coop. This feature involves all major prototyping and implementation of an embedded system weight sensor including both hardware and software development.

The weight sensor feature can be decomposed into the following tasks:

* Create a weight sensing demo on microcontroller board
* Test accuracy and capacity of weight sensor
* Integrate embedded code to transfer readings over network
* Test stability of network and sensor readings
* Create a physical housing for the weight sensor
* Test housing in environmental conditions
* Install weight sensor in chicken coop

Each separate task has independent testing criteria that must be fulfilled before advancing to the next task. While each of the overarching features can be worked on simultaneously, the individual tasks in each feature do have to be followed iteratively for the most part. There are opportunities to do some of these tasks in parallel however, by developing the weight sensing code and housing together.

### 3.3.4 Temperature Sensor

The temperature sensor is another specific feature of the project, though it is a stretch feature and of less importance than the weight sensor. We will use this sensor in largely the same capacity as it will monitor the water for freezing and alerts the client code if it detects the freezing point.

The temperature sensor feature can be decomposed into the following tasks:

* Create a temperature sensing demo on microcontroller board
* Test accuracy of temperature sensor
* Integrate embedded code to transfer readings over network
* Test stability of network and sensor readings
* Create a physical housing for the temperature sensor
* Test accuracy of temperature sensor with housing
* Install temperature sensor in chicken coop

#### 

### 3.3.5 Client Application

The client application will be a mobile application that the client can use to query for chicken coop data. This application will provide real-time data on feed/water levels and coop temperature, it will provide history analytics. These analytics will be able to historically track the data in the coop and potentially make inferences based on these heuristics. Furthermore, the client application will provide notifications if certain levels in the coop fall below client-specified thresholds.

The client application feature can be decomposed into the following tasks:

* Create basic app skeleton
* Implement authorization on application
* Utilize APIs to query and receive data from server
* Test app-server communication
* Create analytics for server data
* Test analytics on the app
* Polish and refine app
* Release app to client

#### 

### 3.3.6 Server/Database

The server/database effort for this project will be developing the back-end to communicate with both the on-site device as well as the client application. This hosting is being provided by the ETG and will be supported at their discretion for the client.

The server/database development can be decomposed into the following tasks:

* Create database on server
* Test capacity of database
* Test queries on database
* Integrate device data with the server
* Test stability of network and sensor readings

#### 

### 3.3.7 Power

This essentially covers the efforts to provide a low power device as well as battery pack development, custom power supplies, and renewable energy. The requirements of this project necessitate that the device can be run solely on battery power and only require changing battery packs every 1 week. As the device will have to constantly be on, power is a huge consideration and limitation of the project. We are exploring unconventional solutions that might satisfy these requirements while allowing us to use a higher powered device. Due to the connections of one of our team members, we may be able to provide a solar panel for use in the project at no cost to the client. This would serve to greatly relieve some of our power concerns and allow us to aim for high powered stretch features, such as machine vision.

The power development can be decomposed into the following tasks:

* Test battery capacity
* Create a battery pack design
* Test battery pack design
* Create custom power supply
* Test custom power supply
* Explore solar panel integration
* Minimize microcontroller power consumption
* Test battery life with full-functioning device

### 3.3.8 Physical Housing

The physical housing component to the project covers only the large scale CNC housing for the device, as well as wiring. Each individual component, such as the sensors, will require its own unique housing that is covered under its respective features.

The physical housing efforts can be decomposed into the following tasks:

* Create CAD model of housing
* 3D print prototype housing
* Test prototype housing
* CNC mill the finalized housing
* Test the finalized housing

## 3.4 Possible Risks and Risk Management

Throughout the initial planning and execution processes of this project we have identified multiple potential risks, as well as management strategies for each risk. The most imperative risks we have identified to date are the feasibility aspects of the weight sensor as well as concerns over anomaly tracking.

### 3.4.1 Weight Sensor

The weight sensor feasibility is purely a concern for water level sensing applications. In the current chicken coop layout, using a hanging weight sensor for a canister of feed should not prove to be an issue. However, due to the fluid nature of the water, it may be an inefficient solution to hang the container for sensing.

If this proves to be the case we have identified two auxiliary technologies that could be used as follows:

* standing weight sensor
* buoyancy level sensor

We would be able to use a standing weight sensor, similar to a traditional scale, to gauge the water level. This may not be the most desirable solution however, as it would occupy significant floor space within the coop.

Our other option would be to use a buoyancy level sensor, placed within the water canister, to monitor the water levels. This would be most comparable to a fuel gauge sensor, though does require its own unique considerations. Seeing as this sensor would be placed directly within the water canister, it is crucial that the composite materials are food safe. Hazardous materials could potentially be consumed by the chickens and passed to the client through their eggs. To mitigate this risk, if we do choose to use a buoyancy level sensor, we would select an aquarium sensor that has been tested for non-toxicity.

### 3.4.2 Anomaly Tracking

A function of our device-hosted sensing algorithms is that they must be able to account for anomalies in the environment. These anomalies are largely based on potential activity by the chickens, as well as potential environmental anomalies.

Our strategy to handle these issues is to use historical tracking to monitor sensing levels with a degree of context. For example, if a chicken went to roost on top of one of these canisters, we would identify this sudden and drastic increase of weight as an anomaly. This would be done by setting thresholds of weight deviation as well as periodically storing sensor data. We are also preparing for potential damage to the device that the chickens may cause through the use of a rugged physical case and wire insulation.

As for handling environmental concerns, we are taking extra care to ensure that our device is exposed as little as possible. Furthermore, we plan to elevate the device installation within the coop, as well as provide a water resistant casing to the device. This would aid in the mitigation of an event such as an extreme storm or flooding.

## 3.5 Project Proposed Milestones and Evaluation Criteria

As stated in Section 3.3, our project can be decomposed into the following primary features:

* Weight Sensor
* Temperature Sensor
* Client Application
* Server/Database
* Power
* Physical Housing

Each of these features have their own sub-tasks that have also been listed in Section 3.3, with individual feature milestones being functional demos and their subsequent testing events.

In addition to demo and testing milestones, per individual feature, we also have several overall project milestones that deal primarily with integration, including their own evaluation criteria. These milestones are covered in table 3.

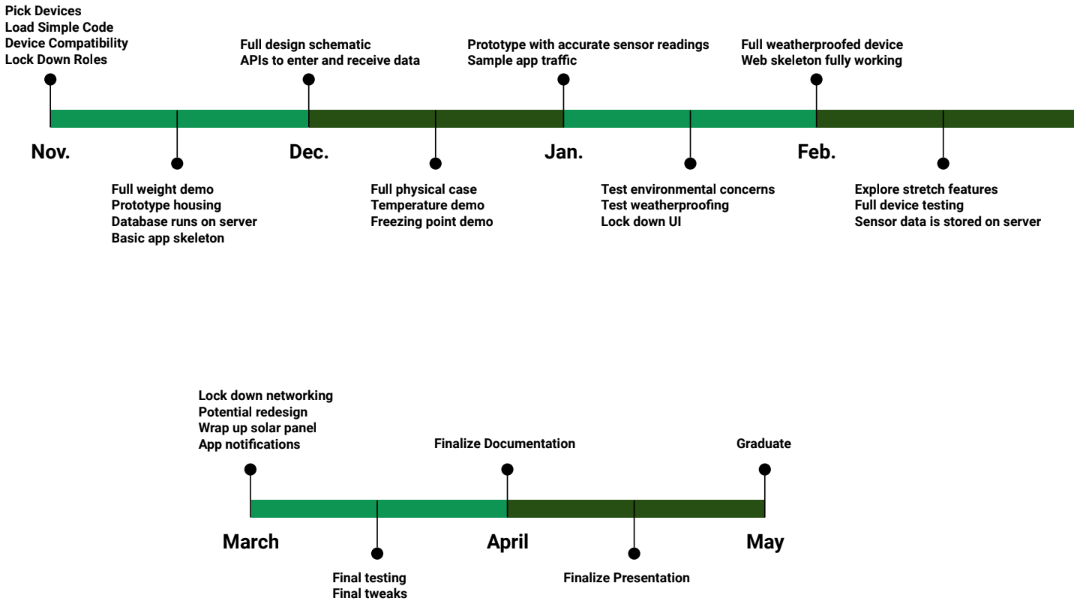
|  |  |  |
| --- | --- | --- |
| Integration Milestones and Evaluation Criteria | | |
|  | Description | Evaluation Criteria |
| Milestone #1 | Successfully collect accurate data from all sensors on the microcontroller. | Accurate sensor data is continuously aggregated by the microcontroller for a period of 24 hours. |
| Milestone #2 | Integrate embedded systems code on the microcontroller to continuously send sensor data to the server. | Accurate sensor data is continuously stored by the server for a period of 24 hours. |
| Milestone #3 | Integrate client application code to query and perform analytics on server-side data. | Client can successfully access server data from the mobile application. |
| Milestone #4 | Verify battery longevity and ease of use of the finalized product. | Full device implementation sustains a battery life of 1 week before recharging.  The client is satisfied with the ease of use. |

*Table 3: Integration Milestones and Evaluation Criteria*

## 3.6 Project Tracking Procedures

Due to the structure of the timeline that our team has generated, we have elected not to use an AGILE workflow structure to track project progress. Instead we have divided key features by team member, according to the project role that they have assumed.

Furthermore, our schedule is aligned for biweekly deliverables coming at the 15th and 30th of each month. These deliverables are what is being used to monitor project progress with a fixed schedule calling for even intervals of work on separate features.



*Figure 3: Fixed Deliverable Schedule*

In addition to tracking efforts through biweekly deliverables, we also have a weekly meeting with our client. These weekly meetings are used as an opportunity to track both team progress and deliverable qualities. The client can then verify the fidelity of implemented demos and the overall quality of the implementation.

## 3.7 Expected Results and Validation

The desired outcome of our project is to provide the client with a fully-functioning and stable system that meets his specific needs.

These needs have been expanded on in Section 2.1 though cover the following general requirements:

* Device can be powered on battery for 1 week
* Device is water-resistant
* Device is chicken-resistant
* Weight sensors accurately monitor feed and water levels
* Temperature sensor accurately monitors water freezing point
* Device will account for sensor anomalies
* Device reports sensor data to server on a regular basis
* Server stores all sensor data in an accessible database
* Client application can support login functionality
* Client application can query the server for data
* Client application can provide analytics of data
* Client application will provide alerts based on level thresholds

Each feature has its own tasks with independent testing events as well as the project having multiple milestones with their own evaluation criteria.

Separately, to validate high level functionality we’ll be primarily running scale and stability tests on the system. This would include running the full-scale system for long periods of time and testing for any efficiency or connectivity issues. Our goal is to validate each independent feature as a whole using these stress tests.

Furthermore, there are some aspects to this project that can not be evaluated through component testing. Client satisfaction is one such aspect and to ensure that all resulting designs meet his expectations, final implementations will be jointly decided at the prototype phase. This covers elements from aesthetic choices to device weight or application accessibility.

Overall we believe that our statement of work is well-planned and will allow for an efficient implementation of our project, that satisfies our client’s needs.

# 4. Project Timeline, Estimated Resources, and Challenges

## 4.1 Project Timeline

The timeline is based on the tasks needed to accomplish all of the deliverables in a timely manner. The current timeline is set, as the project continues we have and will update the tasks based on the work speed. We have already seen that some deliverables are quickly accomplished while others have been taking some extra time.

When the group missed or finalized any deadlines, the entirety of the project schedule was updated to reflect as such.

|  |  |  |  |
| --- | --- | --- | --- |
| Project Timeline | | | |
| sdmay20-55 | Project Start: | Sat, 9/14/2019 | |
| Task | | Start | End |
| Hardware Implementation | |  | |
| Research project | | 9/14/19 | 9/24/19 |
| Research hardware solutions | | 9/24/19 | 10/6/19 |
| Order devices | | 10/6/19 | 10/20/19 |
| Test devices/calibration and initialization | | 10/20/19 | 10/25/19 |
| Re-order new parts | | 10/26/19 | 11/9/19 |
| Create a connection between hardware and software | | 10/20/19 | 10/25/19 |
| Software Implementation | |  |  |
| Choose back-end and front-end services | | 10/21/19 | 10/25/19 |
| Setup front-end | | 10/23/19 | 10/30/19 |
| Acquire and setup server | | 10/30/19 | 11/2/19 |
| Setup database | | 10/30/19 | 11/6/19 |
| Authentication | | 10/30/19 | 11/6/19 |
| Front-end utilizes data from back-end to display sensor readings | | 11/6/19 | 11/11/19 |
| Running full application from server | | 11/11/19 | 11/14/19 |
| Full Implementation | |  |  |
| Graphical representations from live data | | 1/20/20 | 1/27/20 |
| Text and email notifications sent based on user preferences | | 1/20/20 | 1/27/20 |
| Weight and temperature sensors sending consistent and accurate data to the server | | 1/27/20 | 2/7/20 |
| Final prototype installed and testing | | 2/10/20 | 2/28/20 |
| Prepare final presentation | | 3/2/20 | 3/27/20 |

*Table 3. Project Timeline*

## 4.2 Feasibility Assessment

By the end of the fall semester the project should be at a functional prototype status. Factors that can affect the timeline are any issues in our design or under-allocating time in development. Since most of our work will be conducted in the winter, we will be testing in a harsh environment with difficulties.

By the beginning of the second semester, we will have a fully functional hardware prototype that can be used for software testing, as well as field testing. This includes the final microcontroller selection, load cells, and temperature sensors, as well as the first revision of the project enclosure, to allow for portability and battery use for field testing. The first two months of the second semester will allow for any changes to the enclosure, to allow for better mounting on-site as well as ease of use for battery replacement.

For the second semester, we are planning on jumping in quickly and getting into a fully functional and feature complete status by the start of March or early April. The unexpected difficulties that can be found with that are the physical restraints in building in the real world. We will be deploying the device within the chicken coop, meaning in theory that the chickens may be able to interact with the device. We are planning around this possible difficulty however, without real world data of how the chickens will act we can not be fully prepared for all possibilities.

Our front-end will be completed as well by the March/April time mark. We are hoping to create some stretch functionality within our web application, depending on the pace of development.

## 4.3 Personnel Effort Requirements

|  |  |  |
| --- | --- | --- |
| Personal Effort Requirements | | |
| Hardware Implementation | | Personal Effort Requirements |
| Research project | | This task took us 25 hours of overall work because we wanted to start with a well researched problem design. |
| Research hardware solutions | | Going through our possible solutions and researching the most cost-effective and reliable technologies took us 20 hours. |
| Order devices | | Not a time intensive task probably under an hour however waiting for the delivery took approximately 2 weeks of work. |
| Test devices/calibration and initialization | | We spent a fair amount of time testing and setting up devices which summed up to 10 hours of personal effort on members. |
| Re-order new parts | | The microcontroller we originally ordered did not arrive and we had gotten an incorrect unit. |
| Create a connection between hardware and software | | Our team members spent 5 hours of time each creating the communication abilities of our hardware and our backend. |
| Software Implementation | |  |
| Choose back-end and front-end services | | Going through our possible solutions and researching the tools we were all the most comfortable with and reliable technologies took us 5 hours |
| Setup front-end | | Creating the base front-end application took about 10 hours. This includes implementing creating the application, implementing some base functionality with mock data. |
| Acquire and setup server | | Initial server setup took approximately 5 hours. At this time we were able to make requests from the hardware device to the server to input data. |
| Setup database | | Database setup took very little time, but we are constantly adding to it as our application evolves. |
| Authentication | | After the server was set up and we had a basic front-end, we were able to implement basic authentication setup including registering a user, login, and request authorization. |
| Front-end utilizes data from back-end to display sensor readings | | We are able to view current sensor data as well as some basic statistics about the data. After the initial steps, this took very little time. This is still in progress as there can be many more visualizations added as needed. |
| Running full application from server | | This was the most time consuming task, it required a lot of additional server setup. The hardest part was making the website accessible to the public. |
| Full Implementation | |  |
| Graphical representations of live data | | Our front end currently displays live scale data in a graph form. This task took us about 10 hours of work though some of it was linked in the work of setting up our backend. |
| Text and email notifications sent based on user preferences | | Sending email notifications took around 2 hours to implement. Due to cost constraints, implementing text notifications took slightly longer. We had to implement our own way of sending texts without a paid third party service. The next constraint was sending these notifications when we received measurements that were outside the threshold set by the user. In all, this task took around 15 hours. |
| Weight and temperature sensors sending consistent and accurate data to the server | | We expect to need about 10 hours of personal effort for this task though it is highly variable because this is not something that currently is a known task. |
| Final prototype installed and testing | | Placing an hour count on this task is currently not possible as it involves a lot of unknowns. |
| Prepare final presentation | | Final presentation work will take 30-40 hours of personal effort requirements. |

*Table 4. Personal Effort Requirements*

## 4.4 Financial Requirements

|  |  |  |
| --- | --- | --- |
| Financial Requirements | | |
|  | Resources | Budget |
| Client | Water and feed buckets | - |
| ETG | Weight Sensors, Microcontrollers, Temp Sensors, Server Hosting | $200 |

# 5. Testing and Implementation

Ensuring that the sensors, microcontrollers (MCU’s), embedded software, server-side software, and web software all work correctly is crucial to this project. The primary way that we can accomplish this is through the best feasible testing practices. Each of these categories will require different testing standards and tactics.

## Interface Specification

In a networked system such as this, communication APIs are a clear failure point. Specifically this happens when systems disagree on what the expected API is. One measure that we’ve taken to keep versions in sync is to have the web sources and the server sources in the same repository. This ensures that they are versioned together. We also specify the intended design in a single-source-of-truth (SSOT) document under version control. This is especially necessary since Python, the language we use for our backend, does not strictly enforce types.

We also need to consider hardware interfaces for this project. In order to ensure compatibility, we have heavily relied on standard Arduino interfaces. Arduino is a board manufacturer with a reputation for producing easy-to-use hardware. We have selected a board that is produced by SparkFun Electronics that is programmed using the arduino IDE and uses arduino libraries for interacting with pins. This allows us to buy arduino specific sensors so that the manufacturer has the responsibility to test for compatibility.

## Hardware and Software

Our software testing strategy involves integration testing over unit testing. We made this decision as a web team since the internal application logic is not algorithmically complicated. Because of this, individual testing methods are unlikely to be helpful. Instead we have opted for a broader scope, such as externally triggered integration tests, manual test scripts, and a test database that contains fake data. An example of such a test would be a script that sends a fake measurement to the server, waits 1 second and then checks that the data has been inserted in the database.

Testing of hardware will cover three main topics; battery life, sensor accuracy, and signal propagation. First, we must determine what our expected operating time will be on a standard set of batteries and see if it falls within the 1 week specification outlined previously. If this goal is not met, changes will need to be made on a software level to take advantage of built-in power saving modes and other power reduction strategies in order to increase operating time. Next, we must test to make sure that the readings from our load cells and temperature sensors give reliable and repeatable data. Lastly, we must ensure that the WiFi signal produced by our module is capable of connecting to the network in the device’s intended location.

## Functional Testing

Battery life will be one of the key requirements that will be the subject of functional testing. In order to test battery life, the device will be run in a lab environment simulating the conditions that the device will operate in. The device should operate with all of the software and firmware that it will be used in the final implementation, in order to ensure accurate power draw. The device will be monitored to see when it stops responding, to establish the stop-point for our run time testing. Because this method of testing would ideally take 1 to 2 weeks, we can prepare for this testing in advance by powering the device with a laboratory power supply and noting current consumption while the device is in operation. With these measurements, and the published capacity of our batteries, we can calculate our estimated runtime in advance and make software changes as needed before starting lengthy functional trials.

An additional hardware area that requires functional testing is in signal integrity. Because this device will be located in a harsh environment, it is important that the WiFi performance of our device be testing in the exact conditions it is expected to perform in. In order to do this, the device will be run on-site for an extended length of time while monitoring the network integrity between the device and the router, checking for dropped connections, packet loss, and other negative performance markers. In addition to this, an external signal monitor can be used to measure the strength of the signal of each device in the location where it will be mounted.

## Non-Functional Testing

One of the most important goals for non-functional testing is to establish proof that our suite of sensors can provide accurate and reliable data for our software team to use. In order to do this, there are two different methods that will be used. In order to establish performance characteristics for our load cell, we can utilize an electronic load tester to run a series of dynamic loads in order to see how the load cells will respond to changing conditions. This test is important as it will establish that the load cells report the same load for a given weight despite different patterns of usage. To establish performance characteristics for the temperature sensors, we can utilize the temperature and humidity chambers on campus. These devices will allow us to create a computer-controlled temperature profile that will subject the device and sensors to varying conditions, to ensure that all temperatures are reported accurately and consistently regardless of weather patterns.

## Process

Using the numbering from Section 2.4, each use case in the design process is tested in the following way:

1. Testing this use case (for alerts) is done via the integration testing done with the software system. As far as measurement accuracy, we only have the resources to trust the manufacturer.
2. The network component to this use caseid done the same way as #1. The sensor is different so we will need to test that separately.
3. For the purposes of this section I will interpret this use case as only the front end component. I.e. it is assumed that the data is correct in the database. This is a visual component so our usability testing will consist of showing it to people not involved in the project.
4. This use case has the same testing standards and assumptions as #3.
5. This use case has the same testing standards and assumptions as #3.
6. This is implicitly guaranteed by the system design.
7. Due to the inherent inaccuracies in predictions it is hard to give concrete test cases. In lieu of that we will be setting up mock datasets and asserting that the predictions fall within a reasonable range. For example, the system should never predict that there will be negative feed.

## Software Results

### 5.6.1 Web Application

The web application consists of a dashboard, which allows you to monitor your chicken coop at a glance. Here you can add new sensors, see events that have happened in your coop, and get more information about individual sensors. There is also a page for individual sensors which gives more in depth coverage of the sensor.

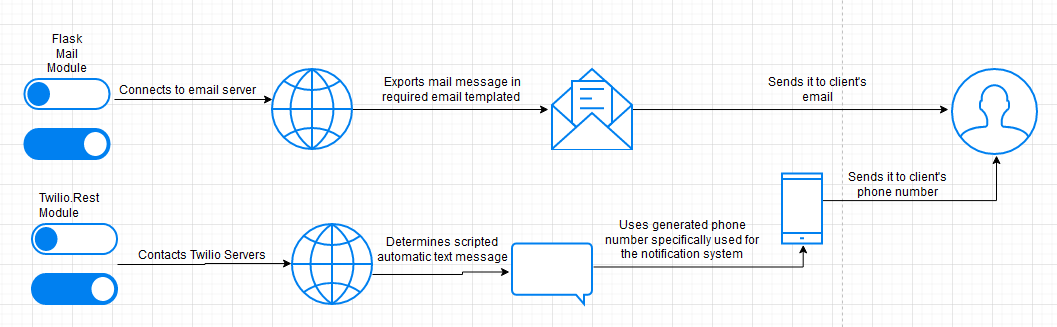
In order to see any data, you need to create an account and register your sensors by their ids. After you register your sensors, no one but you can view the data for that sensor. The application also allows users to change their notification preferences, view preferences, and other user information.

### 5.6.2 Notifications

For notifications, a hard requirement was the ability to send emails, and a soft requirement was ability to send text messages or push notifications. Whether they receive emails, texts, or both should be easily configurable. The user should also be able to specify the thresholds at which they recieve notifications. For example, if the user wishes to be notified when the food level reaches 10% or a certain weight, we should be able to use that setting in order to determine if we should send them a notification.

We decided that the best solution for sending notifications would be running a background task whenever a measurement is sent from the device. We opted for the background task because it can be run in a separate thread so we could close the connection from the request that the device made. This will save battery life for the hardware. This background task determines if we should notify the user.

If we determine that we should send notifications, depending on the user’s preferences we will send a text message, an email, or both. We are sending these messages using python’s built in smtplib library. Sending emails with this is very simple. Sending text messages was a little more complicated. Without using a paid third-party library, we had to use the phone number in order to determine the phone carrier’s sms server. Once we have this information, we can send the text message using the same library for sending email over smtp.



### 5.6.3 Server

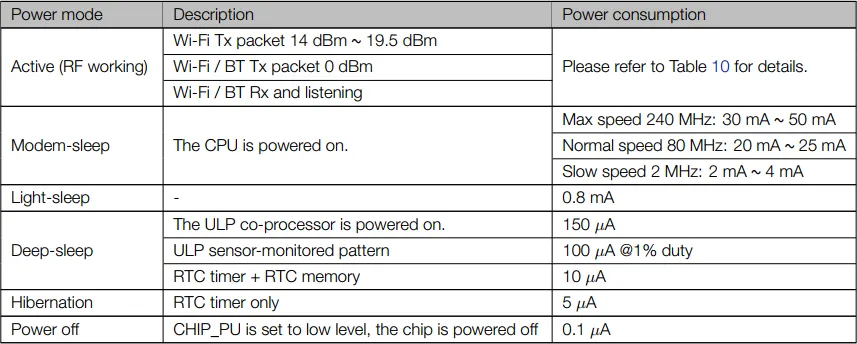
The server results are hard to measure and discuss since they work indirectly on other goals. An example of this is that if the server responded to data entry slowly, the microcontroller might waste battery while waiting. Similarly, if the server gave incorrect data to the web application then all the graphs would be incorrect.

### 5.6.4 Embedded System

The embedded system was able to monitor a sensor periodically, and on a different cycle send the cached results to the server correctly. Each microcontroller successfully works with 2 sensors and has a battery life of approximately 1 week (using AA batteries). This was a little below what we expected, but is sufficient for the project’s requirements.

## Hardware Results

After multiple deployments in the field, we have identified multiple conclusions regarding the hardware designs and choices for our project. First, we’ve established the ruggedness and reliability of our project enclosure. The project enclosure has maintained its physical integrity after multiple cycles of temperature extremes and the silicon covering around the project enclosure has prevented dust and water intrusion inside of the enclosure. One potential area for improvement that we had previously identified is securing the PCB within the enclosure by using the PCB standoffs available in the case. It was originally intended that we would use a carrier PCB that would screw in to the four PCB standoffs, which the microcontroller and ports would be mechanically and electrically soldered to, but the vast majority of PCB fabrication houses were shut down as a direct result of COVID-19. By using a carrier PCB, instead of allowing the microcontroller to float within the case, we could minimize the potential for solder joints cracking over time from vibrations or other movement.



*Figure 4. SparkFun ESP32 Thing Power Consumption*

We have also learned that our battery life has been effectively stretched to one week when using typical, retail-grade nickel metal hydride or nickel cadmium AA batteries. This was achieved by the software team being able to take advantage of some power saving features built in to the ESP-32 microcontroller that were previously identified by the hardware team. The ESP-32 has a secondary, low-power processor built in that can be enabled during idle periods, allowing one to disable both the primary processor and the communications modem as well. In theory, this brings idle current draw down from 100-200 mA to as low as 10uA, which amounts to a significant difference in idle current draw. By implementing some of these features we extended our battery life to one week. With further refinement of the code and lowering the number of polls taken, I believe it would have been possible to extend battery life to the initial goal of two weeks. However, our deliverable was impacted by COVID-19, and it was decided that the battery life of 1 week was acceptable given the circumstances.

Lastly, we learned that one of our 3.5mm plugs had frequent issues with soldered wires breaking off from the connector. The hardware team investigated this issue and determined that a new way of routing wires within the connector assembly itself could introduce a strain relieving effect on the soldered connections if the wires were ever tugged. Additionally, it was suggested that a set of sacrificial wires be used between the sensor wires and the connector itself, to prevent the need to shorten the non-replaceable wires that are installed on the sensor in case the wires ever become detached again. These strategies proved to be very effective and no further instances of wires breaking off have occured since the design change.



*Figure 5. Wire issues with 3.5mm plug*

# 6. Closing Material

## 6.1 Conclusion

Consumer automation is an ongoing trend that will be found on various levels of products in the future. The application of remote automation and monitoring for those at the hobbyist level will have wide reaching effects on the quality of life for the chickens and their owners. Though the implementation of our senior design project was done solely with our client’s needs in mind, we believe that this solution has potential to satisfy needs at the market level. Due to the modular design of the product, this device leaves room for expandability for advanced features such as image processing or supplemental solar power. Additional documentation is also being fabricated to allow future senior design groups to build on our initial design and further the space of hobbyist chicken coop autonomy.

Overall, we believe that we executed our original design plan in an industry-appropriate standard, rigidly adhering to our initial timeline whenever possible. Along the way, our team was faced with several significant hurdles, such as component acquisition and complications due to COVID-19. As we sourced our parts from various online outlets, the time delay in product arrival drastically hindered our ability to uniformly test joint functionality. The work-around that we identified to mitigate this newly arisen risk was to emphasize independent testing when possible, to ensure that the intended component functionality was operating as expected. Additionally, the isolation of COVID-19 impacted our project for the worse, causing our team members to be unable to physically collaborate with each other on the remaining product features. Thankfully, prior to the shelter-in-place order, our team had already deployed a near final prototype on our client’s site, allowing us to have all major proof of concepts in place for testing. Unfortunately, this prototype was missing a secondary weight sensor, blocking our team from being able to concurrently measure the weight of the feed and water containers. Though this was an undesirable conclusion to our physical implementation, it did leave us with the ability to test both the temperature and weight sensing components of the project.

Lastly, we would like to take one final opportunity to thank our client, Dr. Andrew Bolstad, and the on-site Iowa State engineering staff for their work and consideration to our goals of the project. This was an excellent learning opportunity for all members of the team and significantly furthered our readiness for completing engineering projects in our respective fields.

## 6.2 Appendices

## 

## Figure 6.2.1 Measurement Collection

Example documents for the measurement collection which stores the results of sensor readings.



### Figure 6.2.2 Users Collection

Example documents for the measurement collection which stores the end users of the system.This includes a secure hash of the user’s password.



### 

### Figure 6.2.3 Windowed Measurement Collection

Example object within a collection to keep windowed averages for analytics, such as displaying graphs of measurements over time on the website.

### 