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

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

Version 3

Executive Summary

Development 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

- Spending should be limited to our budget ($200)

Environmental Requirements

- Maintain weather consistency (i.e. high temperatures, low temperatures, and rain)

UI Requirements

- Easy to navigate and utilize
- All required real time data displayed

Software Requirements

- Consistent hardware support
- Contain data tools in order to maintain required data
- Real time analytics
- Image processing support

Hardware Requirements

- Able to withstand harsh weather
- Consistent software support
- Able to fetch sensor data
- Able to fetch weight data
- Able to send data wirelessly

Applicable Courses from Iowa State University Curriculum

E E 201: Electric Circuits, COM S 227: Object-oriented Programming, CPRE 288: Embedded Systems

Table of Contents

1 Introduction

1.1 Acknowledgement

This project is based on the development conducted on the Wireless Backyard Chicken Coop project. We are grateful for the number of contributors, friends, colleagues and any future third party who took part in this project's development. We'd also like to sincerely appreciate the encouragement given to us by our client Dr Andrew Bolstad. We'd like to further acknowledge the support that was extended from our Iowa State University's Department of Engineering staff member. As we could not have the opportunity to embark on this project without them.

1.2 Problem and Project Statement

In day to day Farmers are having issues in maintaining and monitoring their livestock. This causes the issue of losing livestock. Which can lead to the hulking issue of overspending on resources to maintain livestock. Furthermore, the loss of livestock unknowingly can lead to the individual farmer to never make the assumption of random variables such as weather or predators to be the source of the issue. This is usually to the faulty monitoring techniques used by farmers. It is not common for farmers to use real time monitoring software systems to monitor their needs. In the current software market surprisingly the amount of livestock software monitoring tools is limited. This does not consider the amount of user-friendly software, which does not allow the inclusion of more users to be more frequent.

Our vision is to optimize the livestock software environment by providing an efficient livestock monitoring software system. We will start by focusing on chickens since chickens are one of the most common animals used as livestock. Users will be able to monitor their chickens by using our web application from the comfort of their home. The user will be given a simple to navigate interface where the chicken coop data will display multiple essential variables such as the amount of eggs that the chickens have recently laid, the current food and water available, and the current amount of chickens that reside within the coop. The software will embed hardware that will allow the user to open and close the chicken coop from their web application. Furthermore, the web application will provide real time data, optimizing the user's experience.

1.3 Operational Environment

The product will result in a sustainable system that can withstand most of the climate changes associated with the Midwestern area of the United States. The system should be able to conduct its task throughout low temperatures (range

from -20 to 0 degrees), high temperatures (range from 100 to 150 degrees), heavy snow, and heavy rain. The main concern for these types of situations is the system's hardware. The product will result in consisting hardware capable to sustain any weather. In order to do this the hardware being used must feature support for any necessary add-ons that can potentially be added in order to disregard some weather conditions.

1.4 Requirements

Economical Requirements

- Spending should be limited to our budget ($200)

Environmental Requirements

- Maintain weather consistency (i.e. high temperatures, low temperatures, and rain)

UI Requirements

- Easy to navigate and utilize
- All required real time data displayed

Software Requirements

- Micro Controllers and Sensors are sending data into the server.
- Back-end fetches and transfers data to client.
- Client is able to get all required data from back-end and display it to the user.
- Image Processing tools, such like OpenCV, are supported for potential implementation.
- Complete Hardware to Server to Client relationship.

Hardware Requirements

- Able to withstand harsh weather
- Consistent software support
- Able to fetch sensor data
- Able to fetch weight data
- Able to send data wirelessly

1.5 Intended Users and Uses

The focus on this project will heavily rely on allowing our users to easily be able to operate our system. That being said the project's user demographic consist of

farmers that have little to no knowledge of operating in software environments. The user demographic does not only range in farmers but should be accessible to any kind of user searching for an efficient chicken livestock monitoring system. The project's development process is in consideration of all types of users and should be able to provide optimal functionality towards the users. Throughout the development process our client will be our prioritized user.

1.6 Assumptions and Limitations

Assumptions and Limitations	
Assumptions	Limitations
The project should be concluding completion towards the end of March	Hardware and software capabilities may have limited peer to peer support
Every team member must perform throughout the entirety of the development process	Individual workload may be over excessive
Some hardware may be expensive	Spending should not exceed budget
UI should appeal to all types of users	Some users may have zero to none experience
Hardware should be able to withstand harsh weather	Fragile hardware might not be able to withstand certain conditions.
Web application should display data in real time	Some data might require time lapses in order to be updated correctly
The end product should be used for hobbyist chicken owners	Final system will focus on one group of livestock; Chickens
Hardware should be able to support WIFI and Bluetooth communication	Communication between hardware may be limited due to WIFI (or Bluetooth) range

1.7 Expected End Product and Deliverables

Temperature Sensor Reading (February 20, 2020):

The temperature sensor system will administrate the temperature inside the chicken coop. The temperature data will be administrated by the microcontrollers. The temperature sensor readings will allow them microcontrollers to send the data to the client via WIFI connection. This dataset will be used for hazards like

extreme temperatures, that could potentially hurt the chickens. The temperature sensors will also allow the user to be aware that the temperature within the chicken coop is currently stable.

Weight Reading (February 25, 2020):

The weight reading sensors will be used to observe the weight data of certain utilities inside the chicken coop. Utilities like water and the amount of eggs that were laid, will be tracked by the weight sensors inside the chicken coop. We will use the weight of water using hangers connected to the microcontrollers in order to keep track of the current amount of water that the chickens hold. The user will be able to view the amount of drinkable water that the chickens contain within the coop through the web application.

Web Application (March 15, 2020):

The web application will be the central hub for the entire system. Essentially all microcontrollers and hardware systems will be communicating through WIFI in order to connect to the server. The server will provide the required data that the microcontrollers fetched, the web application will then display the data. The user will be able to observe all of the data of the Chicken Coop in real-time.

Remote Connection to Server from Microcontroller (March 20, 2020):

All microcontrollers are able to send the required data to the server. The server will then communicate to the client side which will allow the user to view the data acquired from the microcontroller. The microcontrollers reside inside the chicken coop each with different tasks; Tracking water capacity, temperature, etc. The final product will have a network of microcontrollers speaking to the server.

Stretch Deliverables

Autonomous Chicken Coop Doors (April 14, 2020):

The implementation of microcontrollers could be expanded to perform useful operations that the user can take advantage of. The user will be able to close and open the chicken coop for the client web application. Thu, allowing the user to have more control of the chicken coop than he normally could. The autonomous doors will operate by using the gained data from the other microcontrollers to determine whether all of the amount of chickens are inside the coop, and the doors will close autonomously, or open manually from the web application.

Image Processing Implementation (April 20, 2020):

The integration of image processing will be used to determine certain attributes inside the chicken coop that the user might find useful. Attributes like determining whether the chickens are inside the coop, and the amount of eggs they have laid can be tracked by using image processing. The camera will be connected to a microcontroller inside the chicken coop placed in an angle where it can easily fetch the required data. Using tools like OpenCV it will be possible to

determine whether there are chickens and eggs inside the coop. Adding this will provide a more efficient method to gather the data of the two attributes discussed, without having to rely on two separate microcontrollers.

2. Specifications and Analysis

2.1 Proposed Design

Our proposed design consists of hardware to receive and send data from our sensors, and our web application for viewing pertinent information about said data. For our base requirements, we need to have four sensors in all, two weight sensors and two temperature sensors, one of each for the food and water. Our weight sensors will be attached to food and water containers and hang from the ceiling, this will tell us how much food and water we have left and if they need to be refilled. We will use this information to show graphs in order to make predictions about the food and water use. The temperature sensors will mainly need to alert us if the food or water is frozen.

These sensors connect to a microcontroller that will periodically insert sensor data into our database. Before choosing a microcontroller or any hardware for that matter, we needed to keep a few things in consideration. Because this hardware is going to be outdoors, we need to ensure that it will be able to handle the wide range of temperatures and the rain, snow, and sun that we have in Iowa. We also need to be aware that our hardware will be exposed to chickens, not only do we need to make sure they can't eat through the wires or hardware components, but we need to make sure all of our sensors are food safe.

With the hardware considerations in mind, we began designing the web application. We wanted the application to have a few basic functionalities: viewing current sensor data in a friendly way, the ability to query for sensor data for a specific timespan or sensor type, and graphs depicting the usage of food and water. We also wanted this app to have basic security, so user registration and authentication. Ideally we would like the web application to be portable to a mobile platform as well.

2.2 Design Analysis

The majority of this semester was spent researching and planning what we were going to need going forward. Throughout our development process, we have made some changes to our initial project plan. The hardware side of the project has taken most of the attention as it has the most requirements. One of the biggest changes 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 in to the microcontroller itself, and a DS18B20 sensor package, which allows for remote measurement of temperature of up to 6 feet away.

Once our hardware was chosen and we ordered one of the sensors and we were able to receive data, we began work on our web application. For our database, we decided on MongoDB. MongoDB is a document database that requires little set up, easy schema design, and is perfect for projects that require little to no relations. After choosing our database we wanted to choose a backend language/framework that works nicely with MongoDB. We decided on python and Flask as our framework. Flask is very lightweight and easy to set up, ideal for our purposes. Choosing python will allow us to add our stretch goals easily as well.

2.3 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.

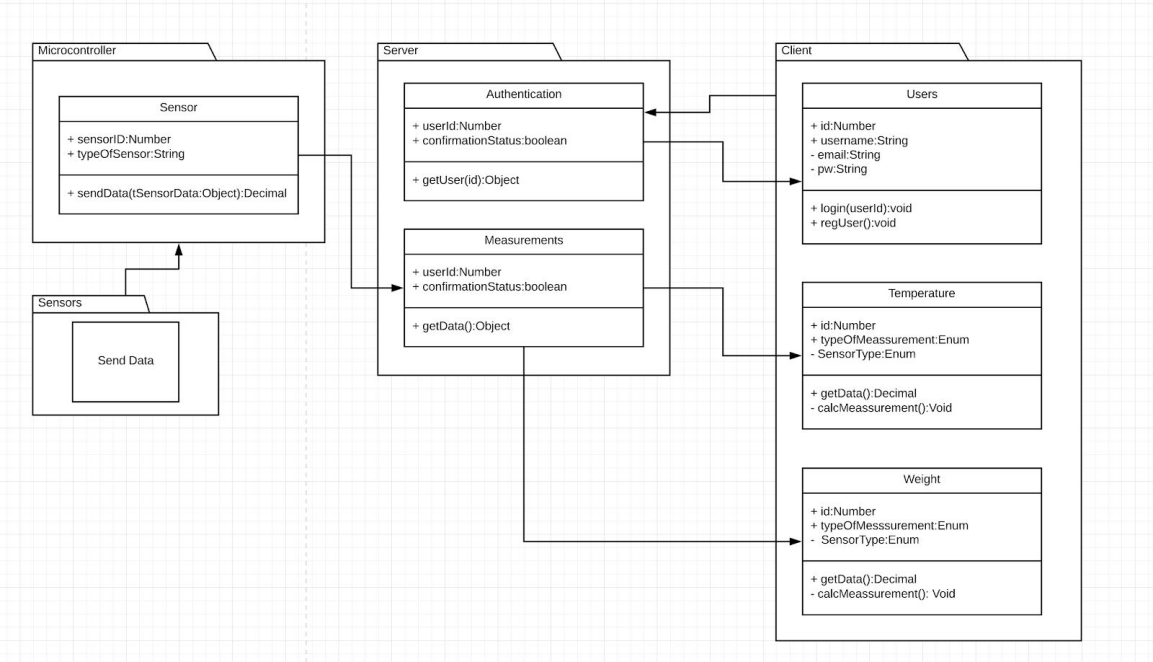
2.4 Design Plan

Use Cases

1. As a user, I need to receive alerts if food or water goes below a certain level.
2. As a user, I need to receive alerts if food or water freezes.
3. As a user, I need to be able to view current food and water levels.
4. As a user, I need to be able to view current temperature of food and water.
5. As a user, I need to be able to view historical data about food and water usage.
6. As a user, I need system to be wireless (battery powered).

7. As a user, I would like to see predictions about how often food and water needs to be refilled.

Our modules are discussed in the diagram below.



Figure

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 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 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 110 degrees fahrenheit
- Exposed materials must be food-safe
- Must be partially moisture resistant
- Must be resistant to chickens

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 Comparision		
	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

Fig 3.2.1: 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 clearer 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 can use in our final product.

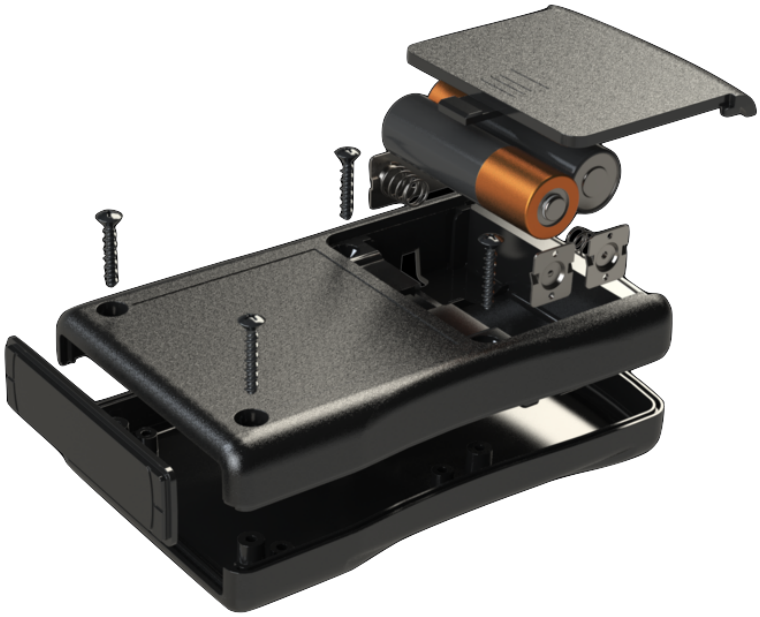


Fig 3.2.2: Prototype CNC Housing

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/Organizational Duties
- Device Selection
- Weight Sensor
- Temperature Sensor
- Client Application
- Server/Database
- Power
- Physical Housing

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.

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.

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.

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

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
- Test login functionality on skeleton
- 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

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

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 2 weeks. 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

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
- Insulate wiring within chicken coop

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.

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.

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.

Integration Milestones and Evaluation Criteria

The integration milestones and evaluation criteria are identified as follows:

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 2 weeks before recharging. Client is satisfied with the ease of use.

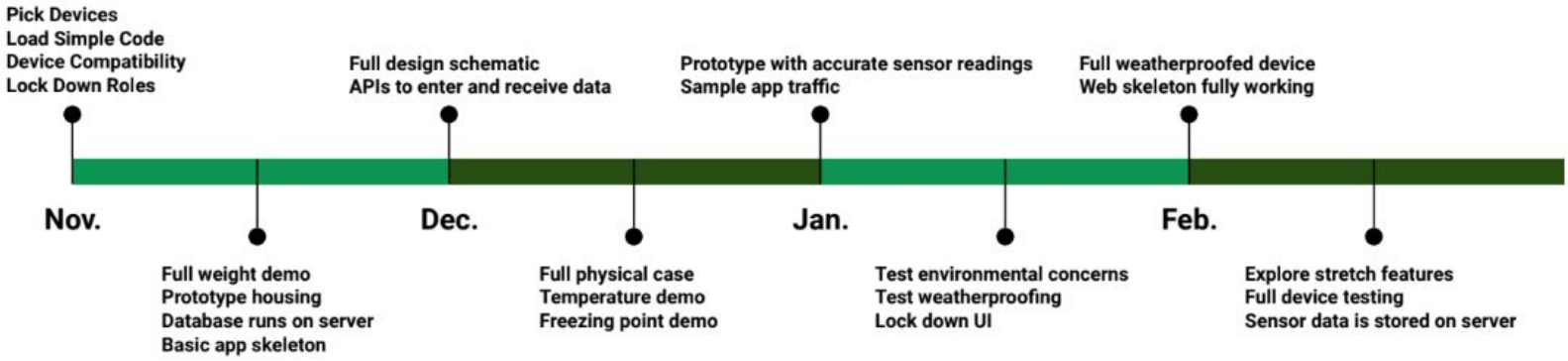
Fig 3.5: Integration Milestones and Evaluation Criteria

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.

This fixed deliverable schedule can be seen below:



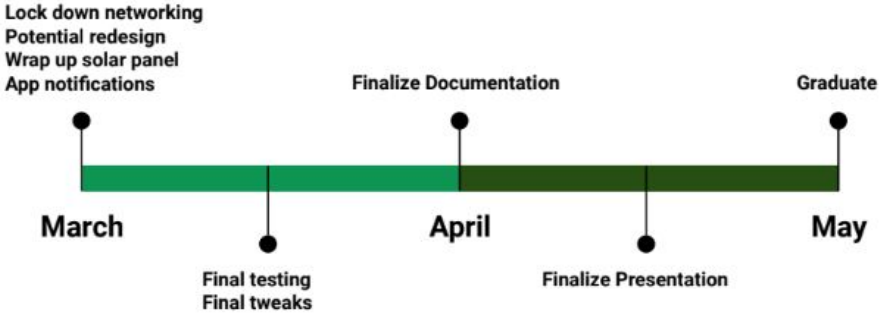


Fig 3.6: Fixed Deliverable Schedule

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 2 weeks
- 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 once 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

Our timeline is based on all the tasks we have decided need to be accomplished to finish all of our deliverables in a timely manner. Our current timeline is set however as the project continues we have and will update the tasks based on our work speed. We have already seen that some deliverables are quickly accomplished while others have been taking some extra time.

In our original timeline we believed that the web skeleton would take us until the end of the fall semester. We found that we had extra time in waiting for our hardware to arrive and were able to complete the web skeleton a whole month early. Including having it be able to run on the server.

Wireless Backyard Chicken Coop		
sdmay20-55	Project Start:	Sat, 9/14/2019
Task	Start	End
Hardware Implementation		
Research Project	9/14/19	9/24/19
Research Possible Hardware	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 backend and frontend services	10/21/19	10/25/19

Create web skeleton	10/23/19	10/30/19
Acquire and set-up server	10/30/19	11/2/19
Set-up Database	10/30/19	11/6/19
Set-up Front-end	10/30/19	11/6/19
Front-end and Back-end communicate	11/6/19	11/11/19
Front-end displays data from back-end to visualize sensor	11/11/19	11/14/19
Full Implementation		
Graphical Representations from live data	1/20/20	1/27/20
Message alerts from freezing temperatures	1/20/20	1/27/20
Detection of chicken weight on sensors	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

4.2 Feasibility Assessment

By the end of the fall semester the project should be a functional prototype status. To accomplish this we'll need to receive all of our required parts and casings. The difficulties to achieve this are in weather and sensor accuracy. Since most of our work will be conducted in the winter we will be testing in a harsh environment with difficulties.

Our web elements are well taken care of with our team and we are expecting the web element to be functional by the end of the fall semester as well. We were originally just planning on having a web skeleton only however as our team started working we found that it was the most accessible place to put our extra time while we were waiting for our parts to arrive so we have reached our milestone in web development early.

By the beginning of the spring 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 spring 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 our spring 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 having the device within the chicken coop and that means in theory that the chickens may be able to interact with the device. We are building 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 web elements will be complete as well by the March/April time mark. We are hoping to create some stretch functionality within our website and 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 Possible Hardware	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 backend and frontend services	Going through our possible solutions and researching the tools we were all the most comfortable with and reliable technologies took us 5 hours
Create web skeleton	Creating the web skeleton that was properly implemented took our software team up too 30 hours.
Acquire and set-up server	This was a surprisingly quicker task then expected. It took 5 hours work hours.
Set-up Database	Database is still in progress although it is functional.
Set-up Front-end	Front-end setup took us 5 hours in initial design and 10 hours in implementation
Front-end and Back-end communicate	Getting the Django backend and the Angular front end to communicate had some complications but because of sufficient documentation we were able to get it working within 10 hours.
Front-end displays data from back-end to visualize sensor	We have our initial graphing working in 5 hours however this task is still in-progress and we have about 10 more hours left on this task.
Full Implementation	
Graphical Representations from 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.
Message alerts from freezing temperatures	We expect to need about 15 hours of personal effort from inception to

	completion. Though we are placing that under consideration should any major difficulties arise.
Detection of chicken weight on sensors	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.

4.4 Financial Requirements

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

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 do that is through the best feasible testing practices. Each of these categories will require different testing standards and tactics.

5.1 Interface Specification

In a networked system like 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 what 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.

5.2 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, testing methods individually is unlikely to be helpful. Instead we have opted for a broader scope such as externally triggered integration tests and 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 to 2 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.

5.3 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.

5.4 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.

5.5 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 food.

5.6 Results

Because of the way the project is structured there are none of the tests have been written yet.

6. Closing Material

6.1 Conclusion

By combining the skills and experience of engineers from multiple disciplines, our team has been able to implement a design plan on path to finish a feature complete project by the end of the year. We also will have proper documentation on how we implemented our features so that any possible future redesigns or problems can be solved quickly and effectively.

6.2 Appendices

Figure 6.2.1 Measurement Collection

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

```
{
    "_id" : ObjectId("5dbdcb02af4cc21bbab2e1d3"),
    "sensorId" : 3,
    "timestamp" : ISODate("2019-11-02T18:29:22.192Z"),
    "measure" : "WEIGHT_LBS",
    "data" : {
        "value" : 50.9
    }
}
{
    "_id" : ObjectId("5dbdcb02af4cc21bbab2e1d3"),
    "sensorId" : 3,
    "timestamp" : ISODate("2019-11-02T18:29:50.987.Z"),
    "measure" : "WEIGHT_LBS",
    "data" : {
        "value" : 50.6
    }
}
{
    "Id" : ObjectId("5dbdcb02af4cc21bbab2e1d4"),
    "sensor_id" : 4,
    "timestamp" : ISODate("2019-11-02T18:29:22.192Z"),
    "measure" : "TEMP_F",
    "data" : {
        "value" : 57.6
    }
},
{
    "_id" : ObjectId("5dbdcb02af4cc21bbab2e1d4"),
    "sensorId" : 4,
    "timestamp" : ISODate("2019-11-02T18:29:22.192Z"),
    "measure" : "IMAGE",
    "data" : {
        "IMAGENAME": "AVCBDHDHDJD",
        "size": [96,96],
        "type": "mono 8"
    }
}
```

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.

```
{
    "_id" : ObjectId("5dbdcb02af4cc21bbab2e1d4"),
    "fName": "Andrew",
    "lName": "Bolstad",
    "nickname": "Andrew Bolstad",
    "password": "$2b$04$oYlc0YzdoRM00MsajVIGx.mcZYukPPF0jzN3v1lVH1zb2d.xxxxxx",
    "sensors": [
        {
            "sensorId": 3,
            "since": ISODate("2019-11-02T18:29:22.192Z"),
            "measure" : "WEIGHT_LBS",
            "type": 0,
            "name": "Food Weight Sensor",
            "alerts": [
                {"value": 3, "threshold": 30, "thresholdUnits": "MINUTE"}
            ]
        },
        {
            "sensorId": 4,
            "since": ISODate("2019-11-02T18:29:22.192Z"),
            "measure" : "TEMP_F",
            "type": 1,
            "name": "Food Temp Sensor"
        }
    ]
}
```

Figure 6.2.3 Windowed Measurement Collection

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

```
{
    "_id" : ObjectId("5dbdcb02af4cc21bbab2e1d4"),
    "sensorId" : 4,
    "measure_count": 5,
    "measure_average": 49.9,
    "period": "HOUR",
    "start": ISODate("2019-11-02T17:00:00.000Z")
}
```