Weather-triggered Wireless Telemetry System

DESIGN DOCUMENT

sdmay25-18

Client: Daji Qiao

Advisor: Sarath Babu

Alex Chambers: Individual Component Designer

Alexander Christie: Client Interaction

Adam Fields: Data Formatting

Nisha Raj: Team Lead

Aidan Gull: Component Integration

Colin Kempf: Documentation

Team Email: sdmay25-18@iastate.edu

Team Website: https://sdmay25-18.sd.ece.iastate.edu/

Revised: 12/9/24 / Version 2

Executive Summary

ARA is an advanced wireless research platform covering Iowa State University, Ames, and nearby rural areas. The team is tasked with creating a weather-triggered telemetry system that measures the performance of wireless technologies developed on ARA before, during, and after weather events of interest. The team will be utilizing multiple external forecast API's in order to automatically have the ARA equipment recognize when a weather event is going to occur. This system will eventually allow researchers to determine how the performance from the ARA framework differs during different weather events.

The current problem we are looking to solve is determining how weather events affect experiments conducted on the ARA testbed. Researchers want to know how the results of their experiments will be affected due to weather events occurring. The requirements of our design include utilizing forecast API's to predict future weather events, triggering our software to collect and store weather and wireless data 30 minutes before, during and after weather events. This data must then be stored in a ZIP file hierarchy, which then ARA researchers would be able to query for this data using a user-friendly interactive UI.

The team currently has a design that includes using three unique forecast APIs that would predict when a weather event would occur, and then based on that information, we would trigger the wireless signal data collection and store that data in the ZIP hierarchy. During this semester, the team initially developed a Python program that queried these three APIs, pulling their forecasts for a future predicted time every hour. The team then visualized this data using graphs, comparing the different APIs against true ARA data. The graphs gave the team, along with the client, a visualization to see the predicted weather from the API compared to the actual weather measured at the ARA weather stations. This would serve as an accuracy test to see how well the weather prediction model would work in terms of predicting a weather event and then begin collecting wireless signal data.

The next steps to be taken for this project is to work on improving our prediction model in order to increase the accuracy of our system.

Learning Summary

Development Standards & Practices Used

- IEEE 1413-2010: IEEE Standard Framework for Reliability Prediction of Hardware
- IEEE 1063-2001: IEEE Standard for Software User Documentation
- IEEE 1012-2016: IEEE Standard for Software Verification and Validation
- ISO/IEC/IEEE 14764-2021: Standard for Software Engineering Maintenance
- IEEE 1448a-1996: Standard for Information Technology Life Cycle
- Software version control utilizing Git

Summary of Requirements

Functional

- Must trigger collection based on abnormal weather conditions
- Must trigger data collection when rain is detected within desired lead-in data collection time.
- Must trigger data collection when snow is detected within desired lead-in data collection time.
- Must trigger data collection when winds above 15 mph are detected within desired lead-in data collection time.
- The software shall have a specified lead-in time where data is being collected a specified amount of time before the weather event.
- The software shall have a specified lead-out time where data is being collected a specified amount of time after the weather event.
- The software shall use three weather forecasting APIs to predict when a weather event will occur.
- The software shall incorporate a UI that allows users to query weather events on specific days.
- Must utilize local weather data for validation.
- While the software detects there is a weather event it shall keep collecting data until the weather event or the lead-out time is complete.
- When the software first starts it shall query the ARA weather API and the weather forecasting APIs.
- When the software detects a weather event it shall begin lead-in data collection before a specified amount of time before the weather event.

- When data collection for a weather event is complete the software shall send data to be processed.
- If there is no weather event then the software shall wait for the next weather event prediction.
- If there is a weather event but not within the specified time to start lead-in time data collection then the software shall wait to begin collection until the specified lead-in time.
- The software shall store weather collection data in a ZIP file hierarchy.

<u>Resource</u>

- Uses the ARA framework to collect, store, and provide access to weather data.
- Shall have access to external weather forecasting APIs
- Weather data shall be stored in the storage space provided by the ARA framework
- Shall run on the server space provided by the ARA framework

<u>Physical</u>

- Must use ARA's Disdrometer when collecting live weather data from the ARA framework

<u>Aesthetic</u>

- The software shall utilize data visualization tools such as graphs and histograms
- The software shall graph the predicted weather at specified intervals and plot the difference between the predicted and actual weather.
- The graph shall utilize multiple colors to see the difference between actual and predicted weather.

<u>User Experiential</u>

- GUI shall be accessible from the ARA systems
- Shall be easy to use for someone who has experience running scripts using the command line.
- The GUI shall be intuitive to query data for users.
- The GUI shall be easy to navigate for new users on the first attempt of using it.

<u>UI</u>

- The software shall incorporate a UI that uses the ARA color scheme requirements.
- The UI shall execute queries in a timely manner such as depending on the size of the dataset under 2 seconds.
- The UI shall allow users to write queries or select pre-determined queries.

Applicable Courses from Iowa State University Curriculum

- COMS 2270 (Introduction to Object-Oriented Programming)
- COMS 2280 (Introduction to Data Structures)
- COMS 3090 (Software Development Practices)
- COMS 3110 (Introduction to Algorithms)
- CPRE 2810 (Digital Logic)
- CPRE 3080 (Operating Systems: Principles and Practice)
- SE 3170 (Introduction to Software Testing)

New Skills/Knowledge acquired that was not taught in courses

- Python scripting development
- Linux server experience
- Data parsing and formatting
- Model predictions

Table of Contents

List of figures/tables/symbols/definitions

Important Definitions and Terms

ARA: Acronym for Agriculture and Rural Communities is an at-scale platform for advanced wireless research across Iowa State University and the city of Ames spanning a rural diameter over 60km.

Disdrometer: Optical device situated on ground station platforms that measure precipitation.

GUI and UI: Graphical user interface and user interface which are both used interchangeably through the document. These are visual ways for users to interact with digital backend devices.

API: Acronym for application programming interface which in our project is used for external weather sources. This is a software intermediary that allows applications to communicate with each other.

Feedback Loop: A process that uses the output of a system to modify its input, creating a chain of cause and effect that forms a loop.

SQL Query: An inquiry about a set of data using the SQL (structured query language) to retrieve relevant information from databases.

ZIP File Structure: ARA dataset guidelines for organizing the data generated from weather and wireless signal experiments. Allow for a standard naming convention for experimental dataset across ARA.

1 Introduction

1.1 Problem Statement

Our clients are researchers for ARA (Agriculture and Rural Communities), an advanced wireless research platform in Ames, Iowa and the surrounding area. Part of their research involves collecting weather data from disdrometers set up at points throughout the region. These disdrometers can record data from all types of weather to be used for research. However, recording all data at all times is inefficient, taking up more space than needed, gathering data when weather conditions are normal. But how can the disdrometers know when a weather event is occurring to begin recording data?

This problem is what our group aims to solve. Our project is tasked with creating a system that will recognize and predict when a weather event is occurring. This trigger signals data collection before a given weather event has begun and allows us to continue collecting data until the weather event has passed. The collected weather data will eventually allow researchers to determine how the performance from the ARA framework differs during different weather events. The weather data will be able to be queried once gathered, allowing for easy visualization and analysis.

As our group works through this problem, we want to intelligently collect data on a wide range of network data during a variety of weather events. We also want to use forecast data to predict future weather events to gather data only when weather events we want to record are going to occur. Lastly, we need to store collected data and allow for user queries to access and format selected data.

Allowing for this data to be gathered and analyzed will allow for better research into ARA's primary mission; creating a wireless lab for the research and development of wireless technology that is both affordable and connective for rural communities and industries such as agriculture. Both internal researchers at ARA and external researchers from around the world will be able to utilize the data our project collects to further their studies and create new devices.

In order to predict the weather events, our project looks to weather forecasting APIs (Application Programming Interfaces). These will give our project data on weather events that are forecasted in the near future. But this can create an issue for us as we need access to other programs not owned by ARA. This is an important issue because without these forecast APIs, our predictions won't be as accurate. We want to have access to multiple forecast APIs so that we can rely on many predictions rather than just one. In order to address this issue we are requesting access to APIs we find that suit our data needs through keys.

The data from these external APIs can also prove problematic even once we have access to them. Just because they have the data we need doesn't mean that all of it is useful or formatted in a usable way. This is another important issue as we need to be able to read the data to inform our disdrometers the predicted time to begin recording the live weather data. To fix this, we are working to clean the data from the APIs as it is received, only keeping the important parts for prediction, and formatting it to a uniform standard. (Appendix 1)

1.2 Intended Users

1. Internal Users for ARA typically from Iowa State (All have same needs)

   Examples: Professors, Graduate students, Undergraduate students

2. External Users of ARA platform (All have same needs)

   Examples: Outside Universities, Industry Professional, Individual Users

From speaking with our client we determined that there are only two users for our product. Our client described to us that users using our product will either be internal users or external users. They described that the only difference between the external and internal users would be that internal users would be from Iowa State institutions, however, these users will not have different needs from the external client. Our client described that the internal users will only have admin access to the product. The external users will be outside of Iowa State but this can include other universities, industry and even individual users. Through this process we determined that our product only has two categories of users: internal and external.

The product that we create will be used by internal ARA users which will include groups of people from Iowa State such as professors, graduate students, and undergraduate students. The product will also be used by external users who will have the same needs as the internal users. Some of the external users include outside university faculty, industry professionals, and individual users.

The people who will benefit from our product are the internal and external users since they will be able to learn when weather events are occurring and how those weather events affect the experiments they are running of the ARA framework. The internal users are local to Iowa State and some of their characteristics is they want to query for certain weather data that occurred on a specific day. They will want to do this in order to see how their experiments running on the ARA framework might have been affected by weather events. The external users will want to query for certain weather data that occurred on a specific day. They will want to do this in order to see how their experiments running on the ARA framework might have been affected by weather events. These characteristics are the same for both internal and external users.

The user needs for the internal users are the ARA researcher needs weather data to be collected when a weather event occurs because they want to collect, store, and publish this information.(Appendix 3) Another need is the researcher needs weather data because they want to analyze how the ARA equipment was affected by weather events. For external users their needs are similar; The researchers need a way to know when weather events occurred because they will analyze how the weather affected their test results on the ARA framework. (Appendix 2) Another need is the researchers want an easy way to query specific weather events data because they want to access weather event data efficiently.

The characteristics of each of our user groups are described in the empathy maps in the appendix below. However, a brief description and characteristics of our internal and external users are as follows. The internal users are any users that are internal to Iowa State University. These can include undergraduate and graduate students, professors, postdoc researchers as well. These researchers have various goals, some can include trying to advance 5G or massive MIMO technology. For example, our persona David is a 25 year old PhD student who is working on experimenting with ARA containers to provide high-speed connectivity in rural areas. Our external user persona, Bob, is a long-time researcher at Bell Laboratories. He recently started working on exploring 5G connectivity and he hopes to find large data sets online that will help his investigation on how weather events can affect connectivity.

The users will benefit from our product because they will have an easy and intuitive way to query weather data so they can easily identify when a weather event occurred so they know if their experiments were affected by weather events. Our problem statement includes storing and collecting data for users to query and access in a formatted way. The proposed benefits we have directly connect to our problem statement as users will have an easy and intuitive way to query data.

2 Requirements, Constraints, And Standards

2.1 Requirements and Constraints

Functional

- Must trigger collection based on abnormal weather conditions
- Must trigger data collection when rain is detected within desired lead-in data collection time.
- Must trigger data collection when snow is detected within desired lead-in data collection time.
- Must trigger data collection when winds above 15 mph are detected within desired lead-in data collection time.
- The software shall have a specified lead-in time where data is being collected a specified amount of time before the weather event.
- The software shall have a specified lead-out time where data is being collected a specified amount of time after the weather event.
- The software shall use three weather forecasting APIs to predict when a weather event will occur.
- The software shall incorporate a UI that allows users to query weather events on specific days.
- Must utilize local weather data for validation.
- While the software detects there is a weather event it shall keep collecting data until the weather event or the lead-out time is complete.
- When the software first starts it shall query the ARA weather API and the weather forecasting APIs.
- When the software detects a weather event it shall begin lead-in data collection before a specified amount of time before the weather event.
- When data collection for a weather event is complete the software shall send data to be processed.
- If there is no weather event then the software shall wait for the next weather event prediction.
- If there is a weather event but not within the specified time to start lead-in time data collection then the software shall wait to begin collection until the specified lead-in time.
- The software shall store weather collection data in a ZIP file hierarchy.

Resource

- Uses the ARA framework to collect, store, and provide access to weather data.

- Shall have access to external weather forecasting APIs
- Weather data shall be stored in the storage space provided by the ARA framework
- Shall run on the server space provided by the ARA framework

Physical

- Must use ARA's Disdrometer when collecting live weather data from the ARA framework

Aesthetic

- The software shall utilize data visualization tools such as graphs and histograms
- The software shall graph the predicted weather at specified intervals and plot the difference between the predicted and actual weather.
- The graph shall utilize multiple colors to see the difference between actual and predicted weather.

User Experiential

- GUI shall be accessible from the ARA systems
- Shall be easy to use for someone who has experience running scripts using the command line.
- The GUI shall be intuitive to query data for users.
- The GUI shall be easy to navigate for new users on the first attempt of using it.

UI

- The software shall incorporate a UI that uses the ARA color scheme requirements.
- The UI shall execute queries in a timely manner such as depending on the size of the dataset under 2 seconds.
- The UI shall allow users to write queries or select pre-determined queries.

2.2 Engineering Standards

Engineering standards are important since they often provide a guideline for different processes and components of a project. These guidelines are commonly used in a variety of projects which makes every project that follows these standards easier to understand. It also makes the process of creating a project easier since anyone making a project can find various standards that have been set by other projects and apply them to their own.

The first engineering standard that we had picked was standard 1413 IEEE Standard Framework for Reliability Prediction of Hardware. This standard goes over a framework for a reliability prediction of hardware. It specifies any required elements needed to form a reliability prediction as well as information needed to determine if collected data meets requirements. Each reliability prediction needs to have a specified list of inputs, assumptions, data sources, methodologies, and uncertainties. All of these are used when calculating risks when we decide to use the results gathered from our hardware. This standard does not provide any way to evaluate various methodologies the group might pick when deciding which one is best for our reliability prediction. The standard also does not provide any instructions on how to perform the reliability prediction. The standard's goal is to allow an ease of defining and creating reliability predictions for hardware so that development teams and users of the application can be assured that the data gathered by hardware components meets the acceptable ranges set in the requirements.

The second engineering standard that we had picked was standard 1063 IEEE Standard for Software User Documentation. This standard details the minimum requirements for each format, structure, and content for various pieces of user documentation. It states that each of these minimum requirements are important for user documentation since many software applications can be made far more difficult to use if the documentation surrounding them are unclear, disorganized, and difficult to read. The standard breaks down each of the minimum requirements into their own sections while stating that the order of each section does not imply a correct step by step process to follow. For the format of each document should remain consistent throughout not only one document but all associated with the project. Each document should also have a file format that is available for most modern computers and have the ability to be modified for users with special accommodations. For the structure, each document should be broken down and divided into topics. Each section should only contain relevant information regarding the main topic with some reference to other topics covered by the document. Lastly the standard covers how the content in the user documents should be laid out in a way that facilitates an easy understanding of topics covered in user documents. It goes into more detail about how the content should be as accurate as possible without being too full of technical definitions that would make the content more difficult to understand. This standard’s goal is to provide a baseline for how any documents created for a project should be made. This baseline can be used to evaluate different documents created for this project.

The third engineering standard that we had chosen was standard 1012 IEEE Standard for Software Verification and Validation. This standard goes over the software validation and verification process. These processes help determine whether a piece of software satisfies requirements and user needs. The standard lists four different levels called software integrity levels that help quantify if the software meets requirements. These levels are in decoding order, high, major, moderate, and low. These processes may have different steps such as analysis, evaluation, testing, review, and assessment. The standard also states that any evaluations or assessments of software must be done within the context of the system which includes any hardware, users, environment, and interface software associated with the tested software. The goal of this standard is to provide a common framework for validation and verification of pieces of software. It also wants to define what the validation and verification process is and the various parts that go into the validation and verification process.

The first standard we had picked is relevant to our project. Since our project uses many different hardware components to measure and record data points we need to ensure that each one of these components is measuring and recording data points correctly. The standard goes over a process on each item that is picked for the reliability prediction. The first part of this process is to define any elements needed for the reliability prediction. This includes a description of the item, intended prediction results, methodologies used for the prediction, inputs for the prediction, metrics used, and any uncertainties that might be in the prediction. Each one of these steps can be done on the hardware components we are going to use. These pieces of hardware are WS100 Radar Precipitation Sensor/Smart Disdrometer which is located at ARA’s Agronomy Hall site.

The other site has a different distometer, OTT Parsivel² Disdrometer that is located at Wilson Hall. Each of these sites also has a weather station that provides other measuring tools.

The second standard covers user documentation standards. ARA already has various user manuals and documents that would need to be updated with our new features. We would need to be confident that our changes to ARA's user documents match with previously set formatting and structural design decisions. Currently our plan is to create an API that can predict weather events so we can begin collecting data points during those weather events. The details of this API will need to be included in ARA's user manual under the ARA API section. In this section instructions on how to use the API will be included as well as details such as the formatting of data, parameters, and outputs for the API. All of this will be structured similarly to that of ARA's current API documents. Based on the many changes that will need to be made to the ARA user documentation this standard is very relevant to our project since it details how user documentation should be structured, formatted, and how the content is presented in each document.

The third standard details the process for validation and verification. Our project has many requirements that are specified above in the requirements section and our API needs to satisfy them. In order to determine if our developed pieces of software meet specific requirements we need some process that can be used as an evaluation tool. This standard provides us with such a process. It helps us figure out how and where to test our API's components and to what degree do those components meet requirements. Due to all of the helpful planning and evaluation information in the standard it is important for our project.

The standards picked out will only make small changes to our project. They are mostly providing some structure and guidance on how we should go about certain parts of our project. Each of the standards is adding something new that we need to the project but they are not necessarily changing our current approaches to developing the API. The first standard will require us to add some kind of hardware testing to the project to ensure that the hardware components are gathering data points correctly. The second standard will give us a guideline on how we should go about updating the user documentation on the ARA user manual. The third will be adding a verification and validation process to the end of our development cycles for each piece of software to make sure that the software satisfies the requirements for the project.

3 Project Plan

3.1 Project Management/Tracking Procedures

For this project we are going to be adopting an agile project management style. The reason we had picked this project management style is that it follows our current structure for weekly meetings. Every week we meet with our client and advisor to discuss what we have done and what we will work on for the next week. Due to this meeting structure it makes sense to use an iterative development style like agile. Our team's progress is currently being tracked using a Github where we list out tasks that need to be completed by the end of the next week. We also are using Discord for communication and have a shared Google Drive for our documentation.

3.2 Task Decomposition

First Semester

The tasks for the first semester focus more so on the planning of the project which is reflected in the tasks that need to be completed.

- Project Charter
- Conceptual Design
- Initial Research
- ARA Experiments
- Formulating Users
- Creating Requirements
- Finalize Requirements and Users
- Task Decomposition
- Detailed Design
- Weather Forecast API Prototype
- Analyze Weather API Data
- Forecast Data Visualization
- Weather Prediction Prototype

Second Semester

For the second semester we created a conceptual design of our project to better understand how it would work and what we would need to do to implement it. For task decomposition we broke up our design into four different tasks to better understand them and how we would approach solving them.

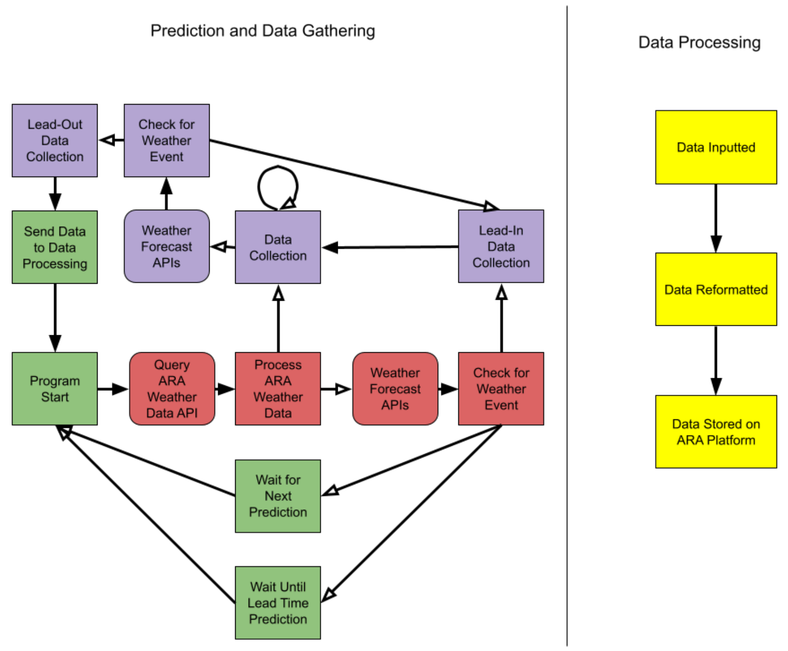


Figure 3.2.1: Component Breakdown

The red sections handle the querying and formatting of weather data. First we query ARA's API and process the data to validate that there is or is not a current weather event. If there is a weather event currently going on, we move to Consume and Record Active Weather; otherwise, we query 3rd party weather APIs and find out when the next weather event is occurring. If it starts soon, we move to Consume and Record Active Weather Data; otherwise, we move to Program Scheduling.

The purple sections in the block diagram specify the way we will record recognized weather events data. Either we will enter this logic block when a weather event is currently ongoing, or we will enter this logic block just before a weather event, specifically, the lead time out. Either we will keep recording data if there are continued weather events occurring. If there are no longer weather events occurring then it will enter the lead out data collection. During the weather data collection if there is another weather event in the forecast then we will just keep on collecting the weather data.

The green sections act as gatekeepers to starting the program again and beginning the consolidating weather data phase. When the program reaches one of the green logic blocks, it means that it needs to wait a certain amount of time before triggering the program start again. There are three ways in which this happens. One, it has finished collecting weather data and needs to predict when the next event is right away. Two, there is no predicted weather event within a given interval, and the program should wait that interval until predicting again. Three, the next predicted weather event is within the given interval, and the program needs to begin predicting more over time at intervals smaller than the initial given one.

The yellow sections handle the output data and formats it in a queryable format. When the weather event is finished and the data from the event is all gathered, it will be reformatted and stored in the ARA platform's database. This data can be queried by users from the ARA platform which can be used for visualization and analysis.

The last part of the project, separate from the backend component design, is the GUI that users will use to query and view the data. The GUI will allow users to query specific data with ease using SQL. The GUI will pull from the data our program has collected and present the information that the user specifies to them. Users should be able to filter for the exact data they want and the GUI should present it in a readable visualized form.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

<u>Consolidate Weather Data</u>

- **Milestone:** Create a system to predict weather events utilizing data from various forecasting APIs and ARA Weather Station data. The model should be collecting data for an hour before the weather event starts

- Metric 1: The model should be able to accurately predict weather conditions 1 hour out for various data points, those currently being:
    - Humidity: within ±5%
    - Temperature: within ±2°C
    - Wind Speed: within ±3 mph

  number of data points we will be examining is liable to change going forward

- Metric 2: The model should collect the 1 hour of weather data leading into weather events (which we refer to as lead in time). We consider the system to be passing this metric if it collects 80% of all expected lead in time.
- Metric 3: The model should tell the <u>Consume and Record Active Weather Data</u> system to record 100% of the time when weather events are starting.

<u>Consume and Record Active Weather Data</u>

- **Milestone:** Create a system to collect current ARA Weather Station data along with current ARA Wireless Connection data. Additionally, this system should measure when on-going weather events are ending and continue collecting data until it has detected that the current weather event has ended for more than 1 hour.

- Metric 1: The system should collect 100% of the data collected by ARA Weather Stations and 100% of the data collected on performance of various ARA Wireless Devices during and 1 hour leading out of weather events.
- Metric 2: The system should detect when a current weather event has ended 100% of the time, and should stop collecting data 1 hour after the event ended.
- Metric 3: The system should detect if a new weather event is occurring within 1 hour of a different weather event ending, and should concatenate the two weather events and continue data collection 100% of the time.

<u>Program Overhead</u>

- **Milestone:** Full Prototype with Weather Prediction/Collection Loop and Scheduling
- Metric: The program runs continuously with no issue.

<u>Data Processing</u>

- **Milestone:** Initial Prototype to feed in some sample test datasets.
- Metric: The program formats the sample datasets correctly.

<u>ARA Platform Querying</u>

- **Milestone:** Initial Prototype to query sample datasets.
- Metric 1: We are able to query the data from the backend correctly.
- Metric 2: Any user is able to query the data using our GUI.

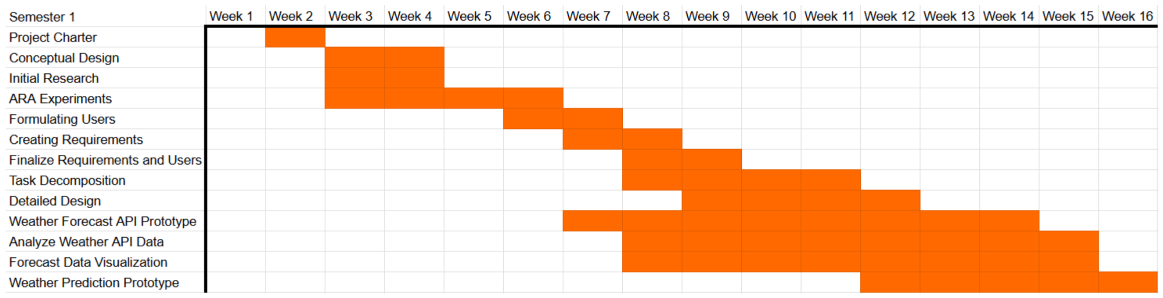
<u>Final Deliverable</u>

- **Milestone:** Putting all the pieces together.
- Metric 1: The accuracy of the model to predict actual weather events. The accuracy of such should predict all actual weather events.
- Metric 2: The accuracy of the model to recognize the true end of a weather event. The model shall accurately predict the end of a weather event 90% of the time.
- Metric 3: The program runs continuously with no issue.
- Metric 4: The program formats data 100% correctly.
- Metric 5: A user is able to query the data using our GUI.

- Metric 6: The final product is completely ready (meets all metrics) by the demo in April.

3.4 Project Timeline/Schedule

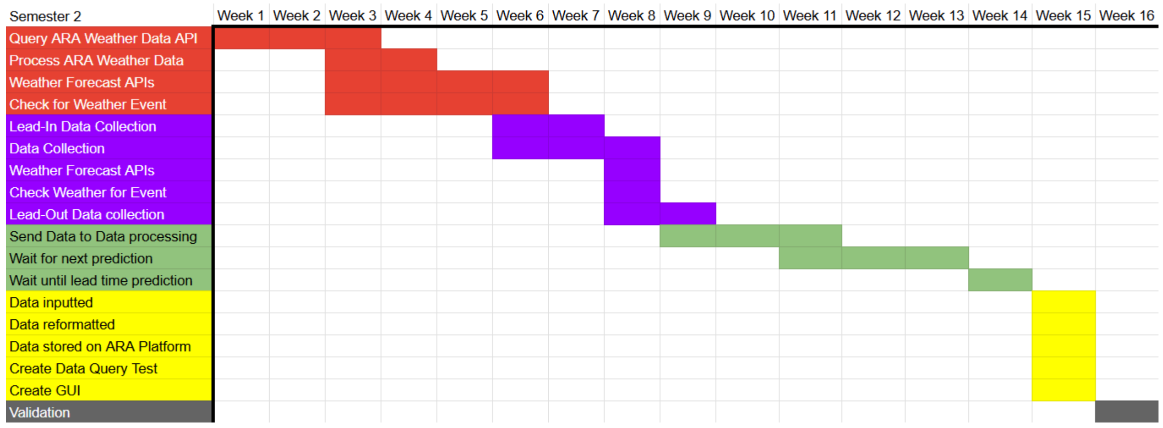
First Semester



Semester 1	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
Project Charter		🟧														
Conceptual Design			🟧	🟧												
Initial Research			🟧	🟧												
ARA Experiments			🟧	🟧	🟧	🟧										
Formulating Users						🟧	🟧									
Creating Requirements							🟧	🟧								
Finalize Requirements and Users								🟧	🟧							
Task Decomposition								🟧	🟧	🟧	🟧					
Detailed Design									🟧	🟧	🟧	🟧				
Weather Forecast API Prototype							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Analyze Weather API Data								🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	
Forecast Data Visualization								🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	
Weather Prediction Prototype												🟧	🟧	🟧	🟧	🟧

Figure 3.4.1: First Semester Schedule

Second Semester



Semester 2	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
Query ARA Weather Data API	🟥	🟥	🟥													
Process ARA Weather Data			🟥	🟥												
Weather Forecast APIs			🟥	🟥	🟥	🟥										
Check for Weather Event			🟥	🟥	🟥	🟥										
Lead-In Data Collection						🟪	🟪									
Data Collection						🟪	🟪	🟪								
Weather Forecast APIs								🟪								
Check Weather for Event								🟪								
Lead-Out Data collection								🟪	🟪							
Send Data to Data processing									🟩	🟩	🟩					
Wait for next prediction											🟩	🟩	🟩			
Wait until lead time prediction														🟩		
Data inputted															🟨	
Data reformatted															🟨	
Data stored on ARA Platform															🟨	
Create Data Query Test															🟨	
Create GUI															🟨	
Validation																⬛

Figure 3.4.2: Second Semester Schedule

3.5 Risks And Risk Management/Mitigation

For each task, identify all salient risks (certain performance targets may not be met; certain tools may not work as expected) and assign an educated guess of probability for that risk. For any risk factor with a probability exceeding 0.5 and each high severity risk, develop a risk mitigation plan. Can you eliminate that task and add another task or set of tasks that might cost more? Can you buy something off-the-shelf from the market to achieve that functionality? Can you try an alternative tool, technology, algorithm, or board?

Agile projects can associate risks and risk mitigation with each sprint.

Consolidate Weather Data

- Risk 1: External API goes down for extended time
  - Probability: 0.10
  - Severity of Risk: High

    - Mitigation Plan: By utilizing multiple API we can mitigate the severity of this risk as a single API becoming unavailable will not break the system. Will need to find an alternative API to replace the unavailable one.
- Risk 2: External API call fails a small amount of time
    - Probability: 0.05
    - Severity of Risk: Mild
    - Mitigation Plan: Once again since we are utilizing multiple APIs and using separate calls for each API this reduces the risk of all the API calls failing at the same time.
- Risk 3: External API changes data output format
    - Probability: 0.2
    - Severity of Risk: Low
    - Mitigation Plan: The plan in case this happens is to keep monitoring the API websites we are using and stay up-to-date on any updates from those sites mentioning a change in their data formatting.

<u>Consume and Record Active Weather Data</u>

- Risk 1: Unable to collect data from weather station
    - Probability: 0.1
    - Severity: Moderate
    - Mitigation Plan: Because this project is intended to be on the ARA Framework, any issues with the weather stations not outputting data would be an issue across most of the ARA Framework. This issue would be mediated by the ARA admins and requires no mitigation plan from our team.
- Risk 2: ARA API changes data format
    - Probability: 0.2
    - Severity: Low
    - Mitigation Plan: The probability of this happening is extremely low. Since we are directly working with the researchers who control the ARA equipment, our plan is to make them aware of how we are using the data currently and inform them how a change in formatting can affect our model.

<u>Program Overhead</u>

- Risk 1: Server goes down
    - Probability: 0.1
    - Severity: High
    - Mitigation Plan: In terms of our code there will always be a copy of the code we are running on our GitHub. In terms of the program itself going down and not being able to predict weather events we will have to temporarily pause weather event data collection in order to get the server up and running again. Another solution could be to request an additional server and have scripts running in parallel on both the servers.

<u>Data Processing</u>

- Risk 1: Data has been corrupted or input is not expected
    - Probability: 0.3

    - Severity: Moderate
    - Mitigation Plan: We plan to have a check in data processing script that will check that the data has not been corrupted and is in the correct expected format.
- Risk 2: ARA platform unreachable for data storage
    - Probability: 0.1
    - Mitigation Plan: We are planning on researching possible external storage for the data as well. This way even if the ARA platform is unreachable then the data will still be stored somewhere such as a cloud server.

<u>ARA Platform Querying</u>

- Risk 1: SQL Injection
    - Probability: 0.3
    - Mitigation Plan: We will need to make sure we are correctly validating and sanitizing the data the users are inputting to query the data. This means we will have to have checks that will ensure the correct formatting is used when the user is inputting the query.
- Risk 2: GUI fails to visualize data
    - Probability: 0.3
    - Mitigation Plan: The team will need to get user validation on if the data was correctly visualized in graphical form. If the data was not correctly visualized then there will be a button for the user to regenerate the graphical visualizations.
- Risk 3: GUI fails to pull correct data
    - Probability: 0.1
    - Mitigation Plan: The team plans to implement multiple tests in order to prevent this failure from occurring. However, if the GUI did fail to pull correct data then the user can have the option to report this issue to the developers which is the team.

<u>Data Collection, Storage and Hardware</u>

- Risk 1: Increase Storage Space & Costs
    - Mitigation Plan: Decrease excess lead-in time, gathering unnecessary data, by having accurate predictions.
- Risk 2: Inaccurate Data Collection
    - Mitigation Plan: By using local devices to measure weather data and incorporating methods for false positives and negatives.
- Risk 3: Creating Hardware Malfunctions
    - Mitigation Plan: Ensure our program is thoroughly tested before merging it with the ARA servers and currently existing applications on their hardware.

3.6 Personnel Effort Requirements

Tasks	Description	Hours
Project Charter	Begin learning about our project and exploring ideas.	2
Conceptual Design	Start brainstorming initial ideas based on client description.	6
Initial Research	Research ideas on how to begin implementing initial parts of our project.	4
ARA Experiments	Start experimenting with the ARA framework and get familiar with the reserving resources.	4
Formulating Users	Work with client to identify potential future users of our project.	2
Creating Requirements	Based on the users of the project identify requirements needed for the project.	4
Finalize Requirements and Users	Confirm with the client the requirements needed to meet the user's needs.	2
Task Decomposition	Breakdown the different tasks needed to complete the project and how all the components fit together.	6
Detailed Design	Create component design of predicting and gathering weather data and breaking down the chart based on different phases.	10
Weather Forecast API Prototype	Begin creating a prototype that uses external APIs to predict and gather weather event data.	12

Forecast Data Visualization	Use the weather forecast data output and create graphs to visualize the difference in predicted weather forecast at a certain time out.	6
Weather Prediction Prototype	Create a prototype that will use external APIs to predict when a weather event will occur.	20
	End of First Semester, going into Second Semester	
Query ARA Weather Data API	Integrate the ARA weather instruments to prototype to query when a weather event is occurring at a weather	6
Process ARA Weather Data	Prototype will process the collected ARA data that has been collected from the weather instruments.	3
Weather Forecast APIs	Integrate using both the externals APIs and the ARA weather instruments to the prototype.	3
Check for Weather Event	Use both the external APIs and the ARA instruments to check when a weather event will occur.	10
Lead-in Data Collection	Implement the lead-in collection time before a weather event when data should begin being collected.	2
Data Collection	Identify the methods and implement the data collection and processing	1

	for collecting the weather data during a weather event. If another weather event occurs while collecting data then data will keep being collected.	
Weather Forecast API	Query weather APIs during data collection to see if there will be ongoing weather events during data collection.	1
Check Weather for Event	Checks for a weather event during the lead-out time after a previous weather event supposedly ends.	6
Lead-Out Data Collection	Collects data for a specified interval after the weather event finishes.	4
Send data to data processing	Sends all collected weather data and wireless data to the data formatting block.	1
Wait for next prediction	End collection and keep checking for a new weather event at the normal rate.	1
Wait until lead time prediction	Waiting for a predicted event to occur.	2
Data inputted	Receives collected data from collection loop.	1
Data reformatted	Reformats the data into a queryable format.	8
Data stored on ARA platform	Store the reformatted data in the ARA database.	3
Data Query Test	Create a test on the backend to query the data from the ARA database.	20
Create GUI	Create the GUI that users	20

	will use to query the data on the ARA platform	
Validation	Verifying that the final product meets all requirements and client expectations.	>25

Table 1: Personnel Hours Breakdown

4 Design

4.1 Design Context

4.1.1 Broader Context

The broader context of our design problem involves the main focus of ARA's goals. ARA focuses on research for communicating in rural communities. Our project is specifically involved with research into wireless signal strength. Our designs will help the ARA researchers better understand what weather conditions could worsen the wireless communication to rural areas. Not only does this affect the ARA researchers or external researchers who use our project, but it could also affect people living in rural areas as researchers use this data to improve communication for their communities. Through this work, our project addresses the societal need for better, faster communication, even in areas of the United States where technology is less prevalent.

Relevant considerations in areas of importance, related to our project:

Area	Description
Public Health, safety, and welfare	The project research will lead to a better understanding of how weather affects wireless data. In turn, ARA can discover ways to help keep internet uptime at a maximum.
Global, cultural, and social	Our project reflects the ideals, values, goals, and practices of the ARA community and workspace. Our project helps further their research and achieve their goals. Our project also affects rural communities through ARA. Our project looks to reflect their needs by assisting in the creation of better wireless communication for those communities.
Environmental	The software we are developing should have essentially zero impact on the environment.
Economic	The only costs would be for ARA to

	maintain and manage the data storage after we have completed the project. Otherwise, it is of no cost to us or other stakeholders.

Table 2: Areas of Ethical Concern

4.1.2 Prior Work/Solutions

According to our thorough product research, there is weather forecasting software out there. In fact, in the previous iteration of our design, we are using several APIs to predict when weather events would be happening. Specifically, we are using Tomorrow.io API [1], Open-Meteo API [2], and National Weather Service API [3] for this purpose. We are using a feedback loop [4] of data to help determine weather events.

4.1.3 Technical Complexity

Our project is of sufficient technical complexity since the design consists of multiple components and subsystems that each utilize distinct scientific and engineering principles. Our design requires the use of external APIs to predict weather events and the use of weather stations to collect data. The design process has been quite challenging and requires scientific accuracy to determine what is considered inclement weather and not simply a muggy day. We also need to be accurate in our software to make sure that it is precise and not buggy.

Our project also matches and exceeds the technical complexity of other projects in the market. Our design features the use of weather stations to determine when inclement weather occurs and collects wireless signal data during those weather events. In our thorough product research, there has not been anything that matches our project in terms of what we are trying to do. Overall, our design meets the technical complexity requirements.

4.2 Design Exploration

4.2.1 Design Decisions

Key Design Decisions:

1. We will use multiple forecasting API’s to predict future weather conditions in Ames, Iowa. This is important to the success of our project because by using these data points, we can determined when to start collecting data in advance in order to collect data during any performance changes
2. We will only be predicting when the next weather event occurs at any point in time. This is important because it determines how far ahead with the forecast data we need to look. Additionally, it simplifies the logic for when to start and stop data collection
3. We will use ARA operated weather devices to measure current weather conditions. This is important to the success of our project as it will be how we validate our forecasting algorithm to determine when we should collect wireless data.

4.2.2 Ideation

When it comes to the first design decision we made it made sense to have multiple forecast API's. This was done due to the variety of the data samples that each API provided. We also found that many of the API's used recorded some data points with better accuracy compared to other APIs. The end goal is to use all of the APIs to have a more accurate prediction model for when we want to start the collection process of performance data for wireless data. Here are 5 API's that we had looked at,

- Open-Meteo
- Tomorrow Weather API
- National Weather Service
- WeatherBit
- AccuWeather

Currently we are using 3 API's in our final project, Open-Meteo, Tomorrow Weather API, and National Weather Service. How we determined these 3 is that we had prioritized free API's and ones that we can get data frequently. Accuracy was also a large part of determining which API's that we wanted in our project. Below is a list of pros and cons for each one of these API's.

4.2.3 Decision-Making and Trade-Off

We discussed as a group what the best decisions would benefit us the most throughout our project. We were primarily concerned about gathering accurate forecast data from forecast APIs, so we can predict weather events. We looked at many different APIs and weighed the options about what benefits and drawbacks they would each have. There were 5 primary APIs that we considered. Through our analysis and testing, we narrowed the list down to 3 APIs (Open-Meteo, National Weather Service, and Tomorrow). After making a decision as a group, we presented our findings with our client and advisor, and they approved our decisions.

API	**Pros**	**Cons**
Open-Meteo	● No API key ● 10000 requests per day ● Pulling weather data from NOAA ● Hourly weather data for a 7 day forecast ● Weather models are updated every hour ● Incorporate real time measurements from airplane data, buoys, rain radar and satellite observations ● Historical weather data from the past 80 years	● When tried the API for Ames it didn't seem to be that accurate but could also be user error ● Not as intuitive as the other APIs ● Outputs a lot of information that will need to be parsed into

	● Provide predictions for Temperature, Clouds, Rain, Solar radiation, Winds at higher altitudes, Transpiration, and Air quality ● Geocoding API → find coordinates of locations ● Seems relatively easy to use and integrate ● API can respond with a chart of whatever data points user wants plotted	more readable format to make decisions from
Tomorrow Weather API	● Free version ● Provides hyper localized weather data ○ Can deliver weather data that's very localized, down to specific city blocks ● Provides real time updates ● Advanced weather layers ○ Can choose between different parameters to include in your forecast such as precipitation, temperature, humidity, wind speed, cloud cover ● Integrates multiple weather models ○ Uses weather prediction models from NOAA ● Can get access to historical weather data ○ Can analyze the trends and tailor our application to specific weather events ● Well documented on how to integrate the API with projects ● Has 60 different weather fields to choose from ● Easy to personalize the location to take weather from	● Output of the API is a lot of information ○ Looks like it needs some work to be parsed and formatted correctly ● Allows for 25 requests an hour ○ Returns too many calls error until next hour ● Requires an API key ○ Need to generate a new key every hour
National Weather Service	● Completely free and open source ○ Everything found on github	● The JSON format seems to be a little

	• Retrieve data for a location using latitude and longitude • Data available ○ Temperature, Dewpoint, maxTemp, minTemp, Relative humidity, Feel like temperature, Heat index, Wind chill, Sky cover, Wind direction, Wind speed, Wind gust, Weather conditions array, Hazards, Probability of precipitation, and Quantitative precipitation • Has even more weather data layers such as snowfall amount and ice accumulation • grid points endpoint fives access to raw numerical data ○ Formatted in JSON document ○ Each data point has valid time which is the interval that the value applies to ○ 2019-07-04T18:00:00+00:00/PT3H ■ Interval starting on 7/4/19 at 1800 and going for 3 hours ○ Value which is the actual value that applies in the valid Time interval ○ Include last-modified header • Caching the result of the points requests to avoid having to do same request multiple time	nicer to read and parse • Need to do 2 API requests to get the weather data ○ 1 for location ○ Another for the actual data user wants • User's have said the cache part is a little complicated • The model grid point also seems a little complicated, need to first get the grid point then will create the table of weather data • Not super reliable, goes down often ○ Was down at the beginning of this year in January • Integration is not as smooth
WeatherBit	• Has multiple API endpoints ○ Current weather ○ Severe weather ○ Forecasts ○ Historical weather ○ Ag-weather	• Doesn't provide an individual endpoint for wind speed ○ Combines it with cooling and heating

	○ Air quality ● Has many methods to look up weather ○ latitude/longitude ○ City name ○ Weather station ID ○ Zip code ○ City id ● Seems to have good documentation for integration ● Uses swagger UI ● Precipitation forecast returns 1 minutes interval forecast for precipitation rate and snowfall rate	degree day information ● Requires an API ● Need a subscription ● Only 50 requests per day with free version ● Lighting data, climate normals, air quality, degree days not included in free version ● Can only have 1 API key with free version ● Standard plan is $40 a month ● Accuracy depends on location ● Semmes to be pulling weather from nearest weather station
AccuWeather	● There is a key for specifically getting forecast data every hour ● Has great documentation with a lot of information about what the requests will give and how we can access the information	● Needs an account. Free accounts can only have one API key ● Free accounts only get 50 calls a day, which could be a problem eventually ● Keys get the weather forecast for a specific area. If we want to get the forecast data for somewhere outside of Ames down the line, we would need additional API keys for those areas

		requiring a paid account or multiple accounts.

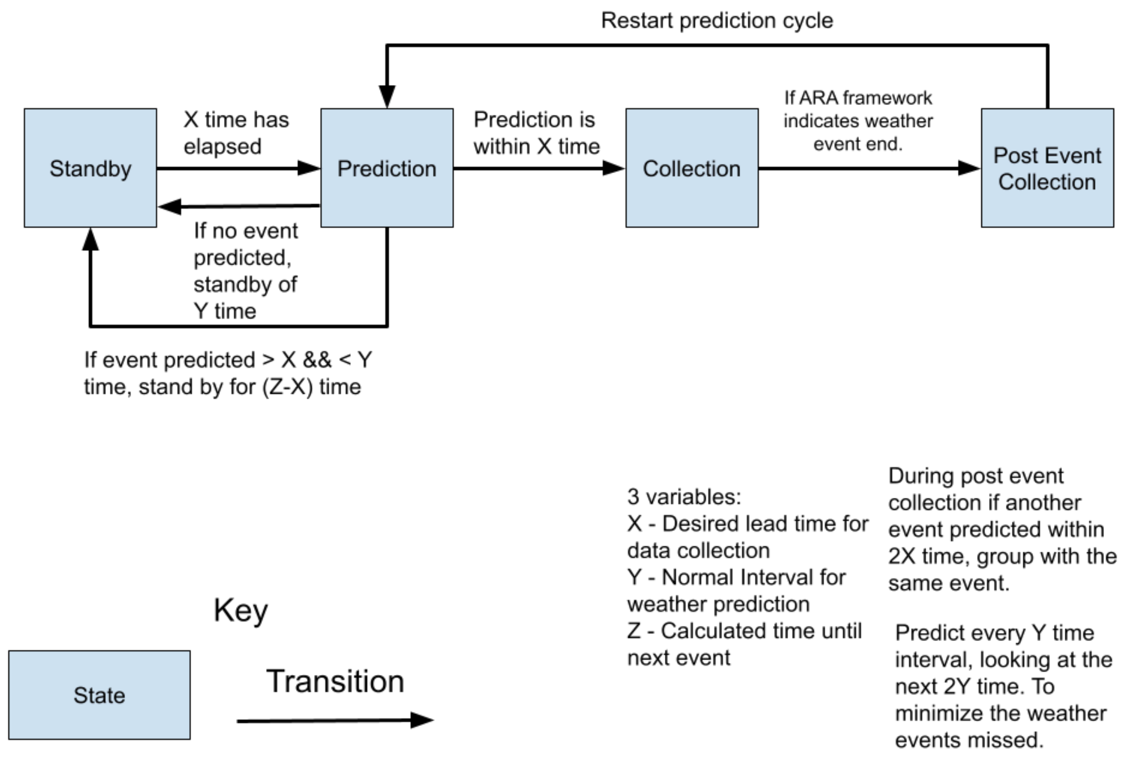
Table 3: Analysis of Implementation

After discussing our options with our client and advisor, they were interested in us pursuing using APIs to predict upcoming events or gathering all data for periods at a time and deleting the unneeded data. We first did research into possible APIs we could use that would gather the forecast data we would need and would be free for us to use. The APIs we decided to pursue were Mateo, Weather.gov, and Tomorrow API. We create a basic implementation (described further in section 6) that would gather the forecast predictions from those APIs. We then compared those predictions to actual historical data for those predicted times. Overall, we found that these forecast predictions could never meet the desired accuracy. Additionally, the APIs couldn't gather specific enough forecast data for the locations where the weather would be recorded, instead encompassing a larger area. This led us to decide to go with the other option, gathering all data for set periods of time and deleting the unneeded data.

4.3 Proposed Design

4.3.1 Overview

Our current design focuses on using APIs to gather forecast data, and use this data to inform when to begin gathering live weather data during weather events. The state diagram below gives a clear visualization of the phase our design will go through as it gathers, predicts, and collects data. It also takes into account certain outlier cases such as back to back weather events.


Restart prediction cycle
Standby
X time has elapsed
Prediction
Prediction is within X time
Collection
If ARA framework indicates weather event end.
Post Event Collection
If no event predicted, standby of Y time
If event predicted > X && < Y time, stand by for (Z-X) time
3 variables:
X - Desired lead time for data collection
Y - Normal Interval for weather prediction
Z - Calculated time until next event
During post event collection if another event predicted within 2X time, group with the same event.
Predict every Y time interval, looking at the next 2Y time. To minimize the weather events missed.
Key
State
Transition

The state diagram shows the different states of our system relative to our scheduling system. The four main states of our software are:

1. Standby: The passive state of the software.
2. Prediction: Where the software uses forecast API to look for future weather events.
3. Collection: The state the software is in when it gets the live weather data from the ARA Framework for the lead-in time before and during a weather event.
4. Post Event Collection: Where the software gathers data for the lead-out time, configures the collected data from a weather event and stores it in a designated location.

There are also three different variables which are used when transitioning from one state to another. These are:

1. X: The desired time before a weather event (lead-in time) when we wish to begin gathering data from the event. For example if our desired lead-in time is 1 hour, we want to begin gathering data 1 hour before an upcoming weather event.
2. Y: The normal interval between weather predictions. This is how much time our program waits between making predictions.
3. Z: How much time there is before the next predicted event.

4. ST: Standby time, how long the program will wait before predicting. This is inferred in the diagram but is elaborated on here. ST is initially set equal to X when the program starts.

These variables allow the program to switch between the states of the system creating the following transitions:

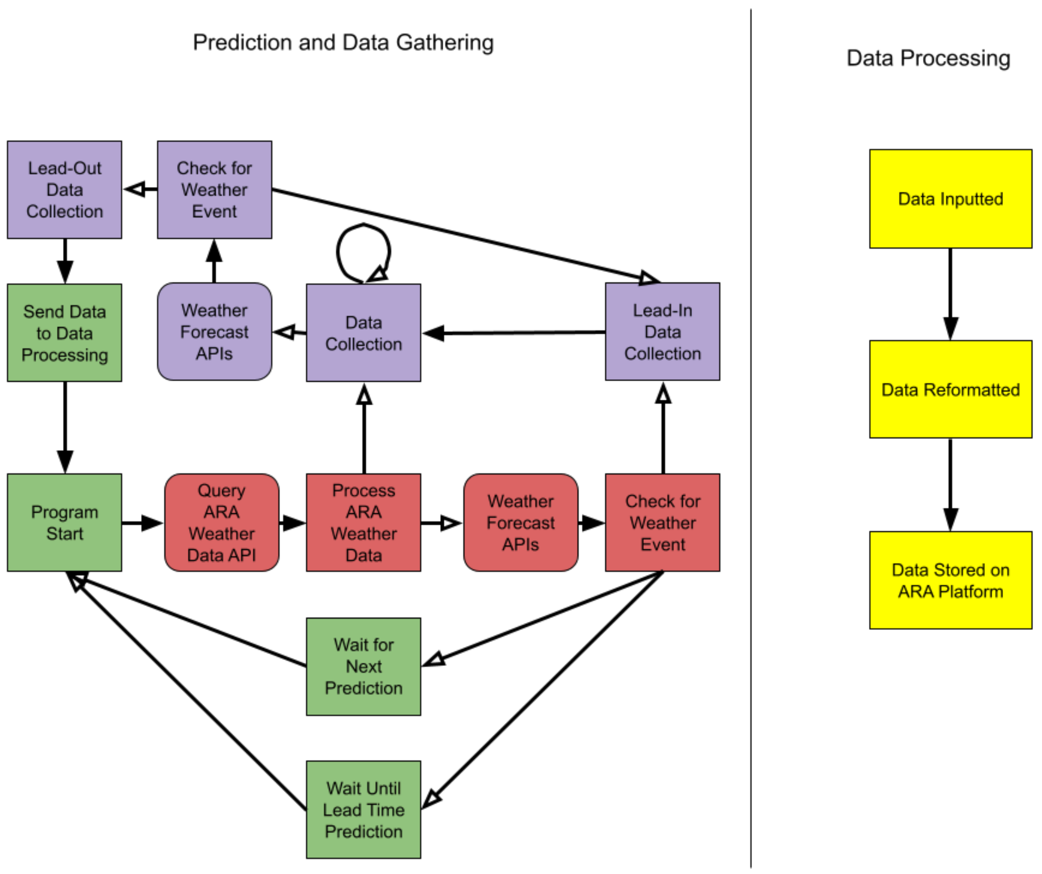
1. When the program is on standby, when ST time has elapsed, it transitions to the prediction state.
2. When the program is on prediction, if there is no event predicted, it transitions to the standby state, and sets ST equal to Y.
3. When the program is on prediction, and it predicts an event, if the time until the next predicted event is further than X but less than Y, it transitions to the standby state and sets ST equal to Z - X.
4. When the program is on prediction, and it predicts an event, if the time until the next predicted event is within X time, it transitions to the collection state.
5. When the program is on collection, and the ARA Framework data indicates that the weather event is over, it transitions to the post event collection state.
6. When the program is on post event collection, it will transition back to the prediction state without a trigger.

These variables also indicate how far to be looking ahead when predicting and how the states should react to predicted events. The two instances this design highlights are:

1. Based on our prediction time interval Y, we should look ahead 2Y when predicting to minimize missed events.
2. If another event is predicted within 2X time when transitioning from post event collection to prediction, the data from both events should be grouped as one event.

4.3.2 Detailed Design and Visual(s)

To help ourselves and others better grasp our design, we created a component diagram, and separated different components into small groups. Ideal phases for us to create the programming of the design.



The diagram above breaks down the overall system into 4 logic blocks:

1. Consolidate Weather Data (Red)
2. Consume and Record Active Weather Data (Purple)
3. Program Overhead(Green)
4. Data Processing (Yellow)

Consolidate Weather Data (Red):

This logic block handles the querying and formatting of weather data. First we query ARA's API and process the data to validate that there is or is not a current weather event. If there is a weather event currently going on, we move to Consume and Record Active Weather; otherwise, we query 3rd party weather APIs and find out when the next weather event is occurring. If it starts soon, we move to Consume and Record Active Weather Data; otherwise, we move to Program Scheduling.

Consume and Record Active Weather Data (Purple):

The purple sections in the block diagram specify the way we will record recognized weather events data. Either we will enter this logic block when a weather event is currently ongoing, or we will enter this logic block just before a weather event, specifically, the lead time out. Either we will keep recording data if there are continued weather events occurring. If there are no longer weather events occurring then it will enter the lead out data collection. During the weather data collection if there is another weather event in the forecast then we will just keep on collecting the weather data.

Program Overhead (Green):

The green blocks act as gatekeepers to starting the program again and beginning the consolidating weather data phase. When the program reaches one of the green logic blocks, it means that it needs to wait a certain amount of time before triggering the program start again. There are three ways in which this happens. One, it has finished collecting weather data and needs to predict when the next event is right away. Two, there is no predicted weather event within a given interval, and the program should wait that interval until predicting again. Three, the next predicted weather event is within the given interval, and the program needs to begin predicting more over time at intervals smaller than the initial given one.

Data Processing (Yellow):

The yellow blocks handle the output data and formats it in a queryable format. This module will put the output data into a hierarchical zip file as well. This zip file structure is the format the client would like so that users would be able to access these data files. When the weather event is finished and the data from the event is all gathered, it will be reformatted and stored in the ARA platform's database. This data can be queried by users from the ARA platform which can be used for visualization and analysis.

4.3.3 Functionality

In the real-world the way we envision our product working as a UI that would allow users to query for different sets of weather data and wireless signal data. Our design would work such that the actual script that's collecting weather data at the different ARA weather stations will be constantly running. This should be started by an ARA maintainer because this data should constantly be collected and then should be formatted into the correct zip file hierarchical structure. The next step in our project will be to begin incorporating the attribute of the wireless signals and create a correlation matrix that would visualize how certain weather events affect the wireless signals being transmitted by the ARA infrastructure. Then the user would interface with the UI itself. In order for the user to do this there will need to be another script that puts data into a database and a queryable format. The user will be able to query the weather data that is being collected by the script. This UI that we are planning on developing will allow users to interface and query for certain data sets specifying which date of weather data they are searching for. The users could launch this UI or it would be integrated into a website and then query the desired data.The UI could also include a query bank where some common queries will already be located and users could just click on these items.

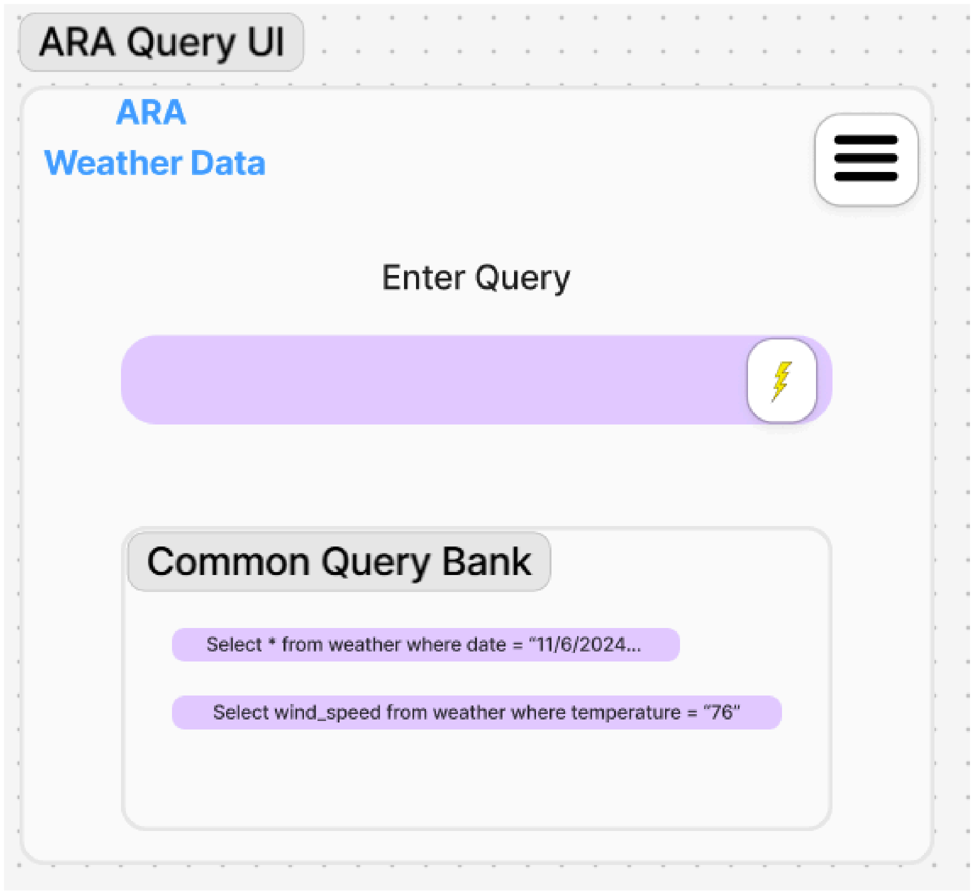

ARA Query UI
ARA
Weather Data
Enter Query
Common Query Bank
Select * from weather where date = "11/6/2024...
Select wind_speed from weather where temperature = "76"

Figure 4.3.3: Query GUI Mockup

4.3.4 Areas of Concern and Development

Our current solution addresses the needs of the internal and external users of the ARA framework users. As a team, we initially had to identify the users and then figure out what exactly they needed from our project. Through this process we derived many requirements that we plan on implementing in our design. The current solution addresses the user needs well since we have directly derived our requirements from the needs the users wanted from the solution. We plan to implement functionality that will allow the users to query for weather data given a certain date that would allow the ARA users to determine if any weather conditions affected their experiments that were running on the ARA framework. Some changes that could improve the solution is providing either an application or an integrated UI with the ARA framework website that would allow users to query weather data straight from the ARA website instead of an outside UI.

With our current design some of our concerns include determining which lead-in time metrics will work best for our prediction calculations. This would include making sure the amount of lead-in time we have allocated is satisfactory for our design. Additionally, we will need to make sure the metrics that we are taking during that lead-in time will accurately give us the predictions of when a weather event will occur. For example, how well does wind-speed or temperature as a lead-in time metric work for predicting a weather event. In terms of addressing this concern we plan to have different trial runs and see which metrics work best for predicting weather events. Another area of concern is determining which features from the forecast APIs will allow us to best make accurate predictions of when a weather event will occur. To address once again it will take some trial and error to correctly determine which metrics and features would best aid us correctly predicting when a weather event will occur. We could also read research articles about which metrics best predict when a certain weather event will occur. For example, will the temperature drop right before a rainstorm so is this one way to even more accurately predict the exact time a rainstorm will occur.

4.4 Technology Considerations

When designing our project, we had to consider the abilities and limitations of the technology at our disposal:

ARA Framework & Servers
Strength: Client owned and created, with through documentation and resources.
Weakness: Must fit within the bounds of the existing ARA framework, both for gathering live data and storing data on their servers.
Trade-offs: We get inside access to the ARA backend, but are limited by how much storage space we have. Our program must also fit in with the existing ARA systems without causing problems

Third-party Forecast APIs
Strength: Are pre existing, professional resources that allow us to easily get the data we need.
Weakness: Must find free, public, APIs. The data collected from APIs will have a range of accuracy.
Trade-offs: We have no control over the code, but we get a much wider range of data for prediction than we could otherwise.
Solutions: Use multiple APIs to get multiple opinions on predicted weather events.

4.5 Design Analysis

So far, we have developed the logic to query, collect, and format forecast weather data pulled from our selected forecast weather APIs. In order to accurately use these APIs to predict future weather events, we need to determine the accuracy of the gathered data. To determine the accuracy of the data, the team has developed 3d visualizations comparing the data to ground truths. Our prototype is being updated to pull real-time weather data from the ARA Platform to act as our ground truth for data analysis. Our analysis of the forecast weather APIs will directly influence our approach to Program Scheduling. Preliminary analysis without an ARA platform ground truth has already shown that some forecast data may be more reliable than others.

5 Testing

5.1 Unit Testing

Unit Testing

- **ZIP File Data Packaging Structure:** Making sure the data collected is correctly packaged into the given ZIP file structure.
- **APIs:** Making sure the feedback loop is working correctly such that the weather patterns can be recognized when looking at both external weather API's and ARA's own weather API's. This is important so we can correlate that weather data to the wireless signal data during that same time frame a weather event occurred.
- **Correct Corresponding Time Frames:** Need to test the functionality to make sure the corresponding data during a given time frame when a weather event is occurring the wireless signal data is also corresponding to that same time frame.
- **User Interface:** Testing the user interface to make sure queries to get specified weather data and wireless signal data are correctly output to users. Essentially, the team will ensure all user functionalities are tested.

Approach

- For each unit, we will first go through the different paths the code could take using control flow graphs.
- Next, we will write test cases for each unit and check to see if the output is as expected.
- We will use requirements to derive our test cases to make sure that each requirement is correctly covered in our units.
- Tests will be automated so that tests can be repeated and this also ensures consistency.
- The team will work to write test cases that cover functionality of all of the units including edge cases.

5.2 Interface Testing

Interfaces

- **Python Scripts and Server:** Interaction between the python scripts for the feedback loop and running that script on the server so that the script can access both the weather and wireless data signals. Ensuring the APIs and the script correctly correspond so that data prediction works as expected.
- **Collecting weather data and wireless signal:** Interaction between the weather equipment and the wireless signal equipment and the scripts that are used to collect that data.
- **Query UI:** Interaction of the weather and wireless signal data and the UI used to query this data.

Interface Testing

- In order to test the python scripts and server the team plans to run the python scripts on the server and verify the outputs of the unit test are the same as when we run it locally. Some test tools we will utilize are Pytest and Behave.
- To test the interaction between the weather data and wireless signal data we will need to manually check that the wireless signal data times grabbed correctly correspond to when a weather event is occurring. This will be used to ensure that the data collection is working properly.
- Testing the interaction between the UI for the queries and the data will be done through unit tests that will ensure that the data is being grabbed from the database correctly and is giving the required output for the specified query. Some tools the team is planning to utilize are JUnits tests and SQL Server.

5.3 Integration Testing

Feedback Loop Integration

- The feedback loop will need to be integrated with the server and work with the external weather data and ARA's weather data. This will be used to determine what corrections need to be made to the prediction model when a weather event occurs. The feedback loop will be used to recognize the patterns in weather data when a weather event is occurring and then pull that corresponding wireless signal data.

- Integration tests will simulate sample external API weather data and ARA weather data. Test cases will be used to verify that a weather event is correctly being identified and that the correct wireless signal data is being pulled when a weather event is recognized.

UI Integration

- Since users will need to use a UI to query weather data and wireless signal data, the testing of the UI is critical to ensure the query interface correctly pulls the correct data from the database.
- Integration tests will ensure the functionality of the query and the data is working as expected. Test cases will validate that correct data for a given query is being correctly found when queried.

Data Storage and Formatting

- The approach of our system is to use external weather APIs and ARA's weather collection devices to match wireless signals data at a time and store this locally on the ARA equipment. Once when a weather event occurs and the corresponding wireless signal data for that is matched. The data will need to be stored in a given ZIP file hierarchy structure.
- Integration tests will ensure that only up to two days of relevant data will be collected. This can be done by setting a 48 hour time limit for storing data.
- In order to validate the ZIP file hierarchy structure is followed. Integration tests will check that hierarchy parameters are met.

5.4 System Testing

The system level testing strategy for the system involves an approach that will include unit testing, interface testing and integration testing. When we conduct the unit tests those will be used to make sure all the individual components such as the python scripts for the feedback loop and weather and data collections are functioning as expected. Additionally, that the APIs are functioning and correctly simulating lead-in and lead-out data collection times. Unit tests will also be used to ensure that the data formatting into the ZIP structure is working as expected using arbitrary data. For the UI the unit tests will make sure the the queries and other user functionalities within the UI are tested for expected functionality. In terms of interface tests needed for the system the team will need to ensure the weather data and wireless signal data are correctly collected in their respective data storage mediums. Along with this we will need to ensure the database storing the data is correctly interfacing with UI to allow for querying. Additionally, the feedback loop will need to interface with the weather systems. For system level testing the integration tests needed include ensuring the UI, data formatting and data collection all work together in parallel and avoid any conflicts between these components.

5.5 Regression Testing

While integrating new functionality to our system we will ensure old functionality will not break by conducting regression testing. The first functionality to be implemented is collecting ARA weather data for use later in the project. After implementing additional functionality we will go back and run unit tests on this component again to make sure the added functionality did not cause any problems with this initial component. Next, we will need to make sure the data is formatted correctly in the ZIP file hierarchy. This will be tested each time functionality is added to ensure that the ZIP file hierarchy is still followed as expected. Once the feedback loop is implemented we will need to ensure that any additional functionality after this does not affect the weather data pattern recognition. This will be done by once again running unit tests after implementation to ensure that weather events are correctly recognized.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

- **Client Involvement:** In order to ensure the system meets the standards and all the requirements of the client we have the client review the product. These reviews will happen regularly such that we receive timely feedback and can make any changes to better fit the client's needs and requirements. Additionally, we will have a steady stream of feedback from our client. Thus, we will ensure that we modify any testing based on client feedback.
- **Requirements Traceability:** We will incorporate a type of requirements traceability table that will map each requirement-functional and non-functional-to a test case. This will help ensure that all requirements have a test case and no requirements are not implemented or not being tested.
- **Fit Criteria:** In addition to the requirements traceability, for non-functional requirements we will ensure that each requirement has a fit criteria. This will make deriving test cases for non-functional requirements much simpler since it will give the team a criteria to test the requirements against.

5.8 Results

Currently, the team is working on finishing the gathering of ARA weather and wireless data. The results from current API tests show inaccuracies in the data of the external API's which we plan to make up for using ARA's own data collection in a feedback loop. Therefore, testing will begin once preliminary scripts are completed with the feedback loop. The results of which will be included here.

6 Implementation

Our main implementation that we had worked on was focused on the usage of external API's for our prediction model for when to start collecting data. We chose to use external API's due to concerns about costs for gathering and storing data on ARA's servers. The external API's can provide us with weather data for our prediction model without needing to be stored and gathered on ARA's own systems. We created a state diagram to go over the main states of the implementation. The state diagram has four states, the first being a standby where we wait for the prediction model. The second is the prediction state where we evaluate if we need to begin collecting. Then we have the collection state where we begin collecting data. Finally we have the post event collection where we process the data.

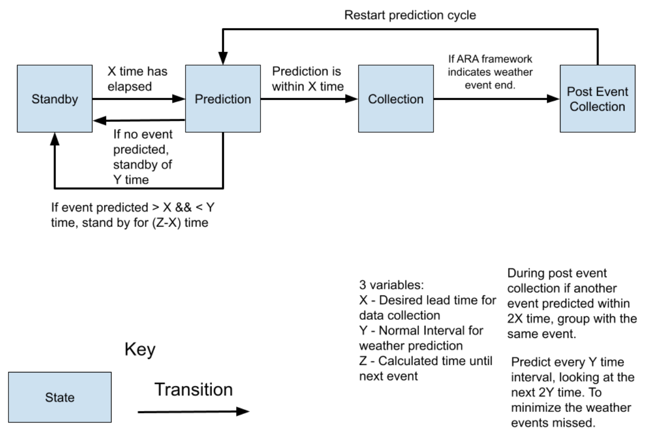

Restart prediction cycle
Standby
X time has elapsed
Prediction
Prediction is within X time
Collection
If ARA framework indicates weather event end.
Post Event Collection
If no event predicted, standby of Y time
If event predicted > X && < Y time, stand by for (Z-X) time
3 variables:
X - Desired lead time for data collection
Y - Normal Interval for weather prediction
Z - Calculated time until next event
During post event collection if another event predicted within 2X time, group with the same event.
Predict every Y time interval, looking at the next 2Y time. To minimize the weather events missed.
Key
State
Transition

Figure 6.1: State Diagram

We also created a task description of the different components of the initial design. The first section of components are the red components which The red sections handle the querying and formatting of weather data. We first query the ARA weather data API then process that data for use in determining if we need to begin collecting the wireless data. The next section is purple which records and formats the data for use in the prediction model for if we need to keep collecting both weather and wireless data. The next section is green which acts as the gatekeepers that start or end various processes for the program such as when to start collecting weather data or if we need to wait. The last section is yellow which processes the data into readable formats or graphs to show correlation between weather events and wireless data performance.

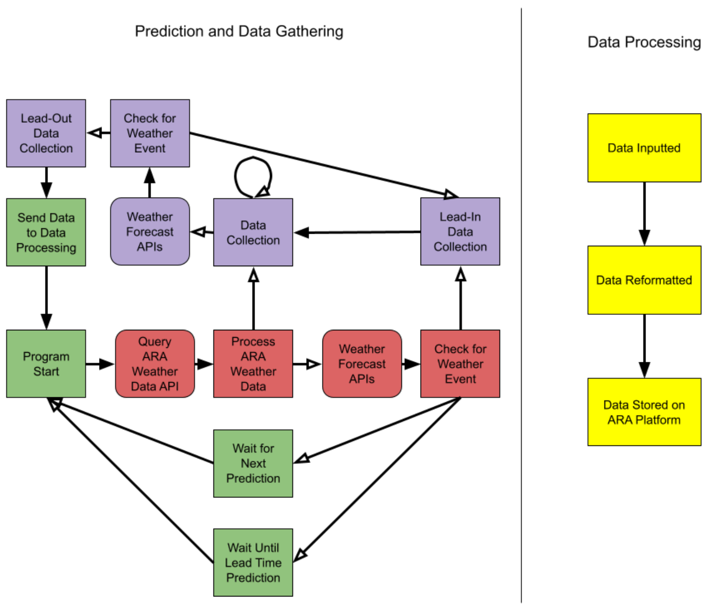


Figure 6.2: Task Decomposition

For each of the API's that we wanted to use in the final project we created scripts that gathered different data values such as temperature, wind speed, humidity, and wind direction. Each of these data points were stored away into text files that would be sent to the prediction model for use in that system. These text files were then used to create graphs that showed absolute error of the graph.

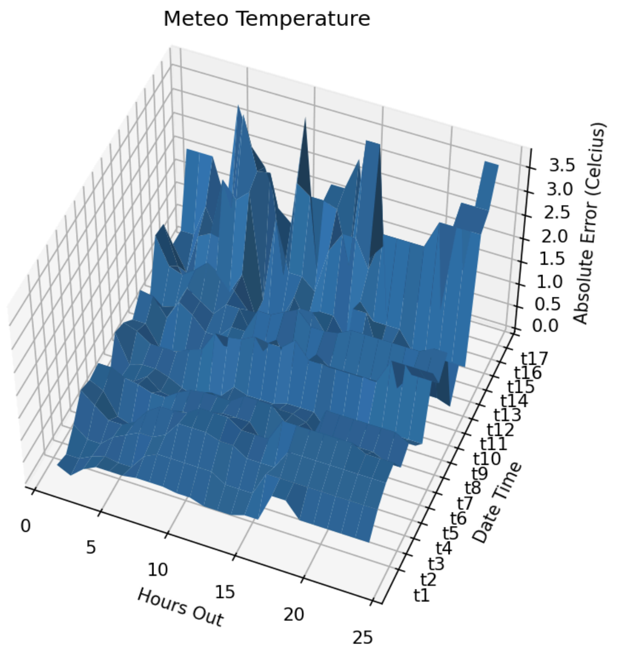


Figure 6.3: 3D Meteo Temperature Graph

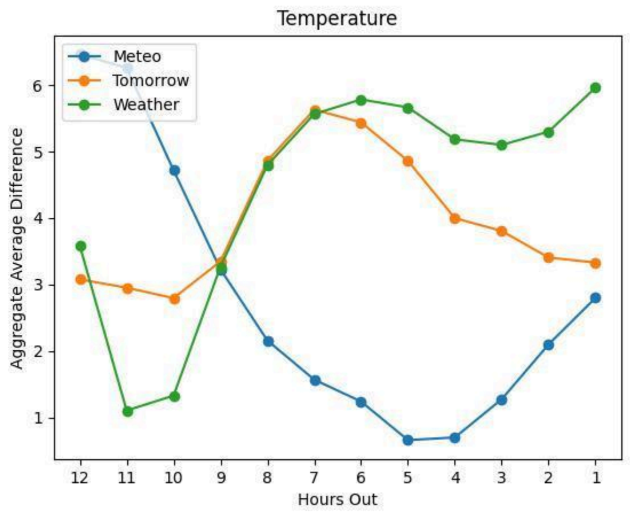


Figure 6.4: Temperature Graph for 3 Different APIs

When analyzing the data from the graphs our team had noticed that there were going to be moments of high inaccuracy. Our team also had other issues with the APIs such as them not being specific enough to the locations of ARA base stations. The external weather APIs could be gathering their data miles away from the ARA base station that we want to turn on. This can lead to moments where the base station is being hit with rain so we want to collect data but the external API weather station is not being hit with rain so the system never kicks on. From this we recognize that a reassessment of the API's implementation is needed to improve the accuracy of prediction.

7 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	**Definition**	**Code of Ethics**	**Adherence to the Code**
Public	Create products and services that benefit the public.	1.01. Accept full responsibility for their own work.	Our team has adhered to this code throughout our project. Everyone is assigned tasks and are responsible for completing that task at a reasonable time and quality.
Client and Employer	Perform work that	2.03. Use the	Our team has

	aligns with the interests of the client and employers.	property of a client or employer only in ways properly authorized, and with the client's or employer's knowledge and consent.	contacted and signed up for various accounts for third party APIs to use them for parts of the project. We have also talked to our client to ensure that these third party APIs are alright to use. We have also contacted our client to gain access to their own systems so we can run our project's prototypes on them.
Product	Products created meet specified requirements and standards.	3.07. Strive to fully understand the specifications for software on which they work.	Our team has contacted our clients many times to better understand their systems so our project doesn't interfere with the other processes that are on the system. We wanted to ensure that our code would not prevent any other users from gathering weather data and wireless performance data
Judgement	Maintain integrity about any assumptions and judgments made in the project.	4.03. Maintain professional objectivity with respect to any software or related documents they are asked to evaluate.	Our team has ensured that our documents have been as accurate as possible when describing our progress on our project.
Management	Managers and leaders ensure an ethical approach to the management of software.	5.02. Ensure that software engineers are informed of standards before being held to them.	While our team doesn't necessarily have a manager we as a team still define the standards set out for each other when completing tasks.

Profession	Advance the integrity and reputation of the profession with the interests of the public.	6.08. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.	Our team reviews documents and code that are created before we finalize them. If there are any inaccuracies or parts that need to be changed then requests for change are created and carried out to ensure that the product meets the end requirements.
Colleagues	Be fair and supportive towards colleagues.	7.04. Review the work of others in an objective, candid, and properly- documented way.	Whenever our team reviews another member's work, if there are any changes that need to be made we state them in an objective and constructive way. Oftentimes we offer to help or give some suggestions.
Self	Participate in learning more about the profession of software engineering and an ethical approach to the profession.	8.03. Improve their ability to produce accurate, informative, and well-written documentation.	Our team has used senior design I as a learning tool to create and design well-written documentation. These skills are invaluable in the professional work force and are always needed for any job so being able to make mistakes and learn from them without too much consequence is very important.

Table 4: Areas of Professional Responsibility

<u>Areas Done Well</u>

One area that our team is doing well in is the product area. Our team has been able to produce high quality products for our client in a reasonable amount of time. Any suggestions the client has in regards to our current deliverables is incorporated in by the next weekly meeting. We understand what needs to be done for our client and when it needs to be done. This shows that our team is motivated when it comes to doing work and is also diligent when it comes to ensuring that our deliverables meet our clients needs.

Areas for Improvement

One area that our team can improve on is the client and employer area. There were several times where there was miscommunication between our team and the client as to the scope and limitations of our project. Our team could have done a better job in reaching out to our client more often to ensure that we fully understood the scope and limitations of the project. In the future we plan to have more meetings and go to our clients office hours if we have questions about certain system requirements.

7.2 Four Principles

	Beneficence	**Nonmaleficence**	**Respect for Autonomy**	**Justice**
Public Health, Safety, and welfare	Design helps researchers gather and analyze both weather data as well as wireless device data.	Design works on existing ARA systems and can be easily expanded with ARA's deployment of other base stations.	Design allows for users to provide feedback. Users can also tailor the project to fit their local area.	Design promotes access to all users with permissions.
Global, cultural, and social	Design helps gather data to improve transmission of wireless data in rural areas.	Design allows a variety of users to use our project. Different cultures and groups of people can use the project.	Design allows users to use the project for specific purposes. They can gather data for their local area.	Implementation is focused on access for educational purposes.
Environmental	Design will help research in rural environments.	Implementation will slightly disrupt the environment. The only impact would be the base stations that need to be set up in that local area.	Design can be tailored to the individual environment where the base station is set up.	Design does not disturb one type of habit over others.
Economic	Design could help to speed up research regarding solutions to rural broadband connections.	Design has not a large impact on the economy. It could affect certain companies that offer products for broadband connection.	Design has some costs for the client currently. These costs are mostly around computation costs for gathering and parsing wireless data.	Design would be usable for all. As long as users follow ARA's user agreements they are allowed access to ARA's system.

Table 5: Broader Context and Four Principles

One context principle pair that is important to our project is the global, cultural, and social beneficence pair. This pair is important because our project will be used by a variety of other users. For now the users in our current scope are internal users in ARAs system and external users. When it comes to the principle pair we are more focused on the external users since they will have different needs and uses for our project. We are working hard to ensure that our project will be useful for these different needs and uses by making sure that our project is easy to use and provides reliable and accurate information.

One context pair that our project is lacking is economic respect for autonomy. Our current project has some cost for our current client. The costs of our project comes from the computations needed to gather and parse the wireless data, and the storage of some of that data to use in our prediction models. These costs can quickly grow as more users are on ARA's system as there will be many calls to not only our project but other experiments on the system. To mitigate these costs our team is only storing enough data to reliably predict weather events and are ensuring that we are only gathering wireless data whenever we are certain that a weather event is occurring.

7.3 Virtues

Team Virtues

1. **Commitment to quality:** This virtue is all about creating quality work which is something that our team strives for. We all value our work and want to ensure that it meets the requirements set out by our clients as well as our own personal standards. We as a team review and critique each other's work to find any areas for improvement.

2. **Responsibility:** This virtue is about each team member taking responsibility for the work that they have done. As a team we make sure that work is divided evenly and each person knows what work they need to do. If one person is struggling with their work we offer help to them.

3. **Integrity:** This virtue is about being honest with the work that each person has done. Each person in the group knows what work they must do and if they are having issues completing their work then we appreciate honesty about how they are struggling with the work. This integrity is important since we want to ensure that the work still gets done at an acceptable quality.

Individual Virtues

Aidan Gull

I have demonstrated friendliness. It is important to me because I want to have a solid team dynamic as well as a friendly work environment. I have demonstrated this by offering assistance whenever needed.

I have not demonstrated industriousness. It is important to me because I would like to be more focused and get more work done for the team. I can demonstrate this by removing any possible distractions from my work areas to help me stay on task.

Adam Fields

I have demonstrated responsiveness. It is important to me because the team contract requires it of me. I have demonstrated this by paying close attention to the team discord and helping clarify things as soon as possible.

I have not demonstrated resolution in a few cases. It is important to me because the team expects me to do what I have said that I would do. I can demonstrate this by doing what I have said I'm going to do without fail.

Alex Chambers

I have demonstrated resolution. It is important to me because when someone commits to working on something, there is a certain expectation that they will complete that work. I have demonstrated this by generally working to completion on various code components of our project.

I have not demonstrated timeliness. It is important to me because when working on a task, it is important to complete it at a convenient time so as not to interfere with others' progress. I can demonstrate this by striving to be more productive during normal working hours rather than doing a majority of my work later into the night.

Alexander Christie

I have demonstrated communication. It is important to me because communication is the key to success in a project like this. With so many moving parts and pieces that need to be completed, strong communication with our team, client, and advisor is necessary. I have demonstrated this by responding promptly to team messages and announcements, alerting my team members to progress I've made on different tasks, and serving as the head of client communications.

I have not demonstrated competence. It is important to me because being able to complete work at an adequate pace is important, especially if I've previously made assurances a deadline could be met. I can demonstrate this by being more aware of my limitations and working to improve my skills.

Colin Kempf

I have demonstrated order. It is important to me because I want to keep things organized and easy to find so the team won't get caught up on where things are or problems arriving from clarity. I have demonstrated this by keeping our documentations structured and by making sure that we follow similar formats throughout our work.

I have not demonstrated silence. It is important to me because our team often needs to focus and remain on task to get work done. I can demonstrate this by talking less with others in distracting conversations and staying on task.

Nisha Raj

I have demonstrated cooperativeness. It is important to me because in order to work well within a team all members need to cooperate with each other. I have demonstrated this by making sure all voices within our team are heard and by ensuring that all team members have a chance to express their opinions and views on team matters.

I have not demonstrated resourcefulness. This virtue is important to me and the team because we need to be able to find innovative solutions for any problems that might arise. I can demonstrate this by working to think of more out of the box and unique solutions to problems that come up while we are working on our project.

8 Closing Material

8.1 Conclusion

The goal of our project is to create a system that would allow ARA researchers to see how certain weather events affect wireless signal data. This information can be used by users of the ARA testbed to determine if a weather event occurred while their experiment was running and, if so, how those events could have impacted the performance of their wireless tests. The plan of action to achieve this goal is first to collect and store a specified number of days worth of wireless signal data and weather data on the respective equipment. Then, initially, we will manually go through and see when a weather event occurred by looking at the weather data. Then, we will correlate those events to what the wireless signal data looked at during those times. To help aid these correlations we are going to construct a feedback loop. The goal of this feedback loop is so that, eventually, it will be able to recognize the patterns from the external weather API's data and ARA weather data to help predict when a weather event has occurred. We then can recognize the weather event occurring and then pull the corresponding times in the wireless signal data. Initially, the team, along with our advisor and client, researched ways to predict when a weather event would occur using multiple weather APIs and ARA weather equipment. Once a weather event was predicted we would have a lead-in and lead-out time of collecting the wireless signal data. This would then similarly allow ARA users to see how certain weather events affected their wireless signal tests.

8.2 References

[1] Tomorrow.io APIs, Tomorrow, https://www.tomorrow.io/weather-api/ (Accessed Sept. 25, 2024).

[2] Open-Meteo.com Free Open-Source Weather API, Open-Meteo, https://open-meteo.com/ (Accessed Sept. 25, 2024).

[3] API Web Service, National Weather Service, https://weather-gov.github.io/api/ (Accessed Sept. 25, 2024).

[4] "How to Create a Feedback Loop: Step-By-Step Guide With Best Practices", Userpilot, https://userpilot.com/blog/how-to-create-a-feedback-loop/ (Accessed Dec. 2, 2024).

[5] "The Ultimate Guide to Building a Functioning Feedback Loop Model", Fibery, https://fibery.io/blog/product-management/feedback-loop-model-guide/ (Accessed Dec. 2, 2024).

9 Team

9.1 Team Members

1) Aidan Gull
2) Colin Kempf
3) Adam Fields
4) Nisha Raj
5) Alexander Christie
6) Alexander Chambers

9.2 Required Skill Sets for Your Project

Front-End development: Developing the front end so that users can query weather data and the wireless signal data.

Linux Experience: Running processes in the background and piping output into a file that can be parsed.

Python Development: The scripts that we will be creating in order to correlate weather data and wireless signal data will be in Python. Many ARA APIs already utilize Python for scripting so this was the preferred language.

Experience with ARA: Familiarity, with the ARA platform and different experiments that can be run on the testbed.

Data Analysis: Understanding how to assess and manipulate gathered data to inform our development, create an algorithmic feedback loop, and format for the final deliverable to the end user.

9.3 Skill Sets covered by the Team

Front-End Development: Adam, Colin, Alex Cha., Aidan

Linux Experience: Adam, Nisha, Alex Chr., Alex Cha., Aidan

Python Development: Nisha, Alex Chr., Colin, Aidan

Experience with ARA: Nisha, Alex Chr.

Data Analysis: Colin, Adam

9.4 Project Management Style Adopted by the team

Our team uses an Agile methodology since we have team meetings weekly where we collaborate as a team. We also speak with our advisors on a weekly basis where we work and discuss breaking down tasks into smaller components. The weekly meetings help facilitate consistent feedback from our client and help us stay on track with our project.

9.5 Initial Project Management Roles

Alex Chambers: Individual Component Designer

Alexander Christie: Client Interaction

Adam Fields: Data Formatting

Nisha Raj: Team Lead

Aidan Gull: Component Integration

Colin Kempf: Documentation

9.6 Team Contract

<u>Team Procedures</u>

1. Day, time, and location (face-to-face or virtual) for regular team meetings: Monday and Fridays from 2-4pm at the library (face-to-face)
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face): Discord
3. Decision-making policy (e.g., consensus, majority vote): Majority Vote
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived): Team Recorder: Colin Kempf, Archived: Stored in documents in a team shared Google Drive.

<u>Participation Expectations</u>

1. Expected individual attendance, punctuality, and participation at all team meetings: We expect all members to attend all meetings unless some unforeseen circumstances arise or they communicate a reasonable reason for not being able to attend. We also expect all members to contribute equally to the project and try to split up tasks depending on the member's strengths.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines: Everyone should roughly have equal responsibilities. Do what you commit to within the timeline specified, if issues arise, communicate with team members and ask for help.
3. Expected level of communication with other team members: We expect steady communication of schedule conflicts, development progress, and ideas among the team through our discord.
4. Expected level of commitment to team decisions and tasks: Strong commitment to the team, decisions, and tasks. Communicate with the team during the decision making process and talk about any disagreements and differences in opinions.
5. Strategies for supporting and guiding the work of all team members: We will use consistent communication to ensure that the team is organized and support is extended to individuals who need additional help on their tasks.
6. Strategies for recognizing the contributions of all team members: Give positive feedback when team members do a good job on a project. Give team members the correct credit for their contributions.

<u>Collaboration and Inclusion</u>

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

   Alexander Christie: I have industry experience utilizing Python scripts to automate complex tasks. Also, through my experience with acquiring a Technical Communications minor, I've gained valuable skills in technical document writing and professional communication.

   Alexander Chambers: I have 3 semesters of research experience utilizing python scripts. Additionally, I have had 3 summer internships giving me a lot of experience working on and producing commercially viable code.

   Nisha Raj: I have done previous research with the ARA team so I have some background experience in their framework and their ARA portal. I have industry experience working in wireless communications. I also have a minor in cybersecurity so I have experience in Linux and security measures the team could take to keep the data we collect secure.

   Aidan Gull: I have some industry experience working on the backend of various systems. I have also worked with python and frontend development that would be useful for this project.

   Colin Kempf: I have industry experience with gps devices from my time doing an internship with Samasung. I also am getting a Data Science minor along with my major so I have experience handling and analyzing data.

   Adam Fields: I have industry experience with monitoring wireless connections. I am also minoring in Cyber Security.

2. Strategies for encouraging and supporting contributions and ideas from all team members:

   Give positive feedback when team members do a good job on a project. Give team members the correct credit for their contributions.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

   The best way to inform the team of this issue is to communicate in person with the team during our meetings on Fridays. Members can also bring it up in the Discord chat. The team can address the issue and analyze what is causing it and how to go about resolving the issue. The goal of the team is to make sure every team member is heard and has an equal say in all the decisions and projects.

<u>Goal-Setting, Planning, and Execution</u>

1. Team goals for this semester: Complete deliverables as they arise, meet expectations of our client, set ourselves up for success next semester. Make sure we are making progress in terms of getting the requirements of the project and then towards the end of the semester start implementing the software components.
2. Strategies for planning and assigning individual and team work: Assign tasks based on individual skills and team members schedules. Work together to determine who is best

suited to approach what and who may need more help based on difficulty and time. Additionally, try to work together on larger tasks that may be more involved and need full team collaboration.

3. Strategies for keeping on task:

   Make sure the team doesn't get off track by talking about unrelated topics. This can be achieved by keeping each other accountable when someone goes off track. The team needs to work together to bring everyone back to working on the related task at hand.

<u>Consequences for Not Adhering to Team Contract</u>

1. How will you handle infractions of any of the obligations of this team contract?

   First, try to determine the cause of the issue and resolve things internally by communicating. Speak with team members and see what is going on and why the infractions are happening. Make it clear with team members that this is a team project and everyone needs to evenly contribute in order to achieve the end goal, and let them know that by participating in these infractions they are letting the team down and even jeopardizing the project.

2. What will your team do if the infractions continue?

   If the infractions continue then the team will speak with the advisors and the professors about the team members' continued infraction and how it is negatively affecting the productivity of the team.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Aidan Gull	DATE 9/13/2024
2) Colin Kempf	DATE 9/13/2024
3) Adam Fields	DATE 9/13/2024
4) Alexander Christie	DATE 9/13/2024
5) Alexander Chambers	DATE 9/13/2024
6) Nisha Raj	DATE 9/13/2024