Executive Summary

Smart Irrigation Controller Project

Man Ha, ham@oregonstate.edu

Maxwell Okazaki, okazamax@oregonstate.edu

Joshua Barringer, barrinjo@oregonstate.edu

Overview

Originally, our project's goal was to create a system that provides recommendations and automatically controls a user's irrigation system to increase the garden's outputs based on the size of a user's garden, what specific plants are in their garden, and the forecasted weather conditions. Our project aimed to make the task of figuring out how much to water and fertilize a garden easier, giving a user more control over their crops.

This system was designed for a home garden that already has an irrigation system set up. We wanted to create a simple and friendly web interface that could report the current garden status, which includes humidity, temperature, soil moisture, and light information.The data would then be used to create the watering schedule and make a task list for the user in order to make caring for the garden as easy as possible, while also ensuring that the garden is being properly cared for. Furthermore, the user can manually control the system and make schedule changes as needed.

Our approach to this project was to split into individual work and work by block. There were three main code blocks that we were developing, the database, back-end server, and web interface. A team member would work on a section of code for a repository and upload it. After testing from other team members, that code would be updated or merged into the standard code branch for that repository.

In terms of the task management process we used Trello to track the workload and work assignments. This was helpful to track which team members are working on which tasks for the project. We met weekly created tasks, and assigned them. We updated regularly with task progress and communicated early if a task needed to be changed or reassigned. We also used git and github as our version control software. Early code was developed in separate branches and merged into the main branch when complete.

The initiation phase for this project will be to develop a list of minimum viable product (MVP) requirements that the project will have, determining the project goal. The initiation and planning phase will be to research how to best create and design the necessary components of the MVP. Next, we will begin creating and implementing the components in project execution phases. The project planning and control phase will consist of debugging and testing to ensure that the final product meets the standards that were outlined by our project partner.

During development, we will be designing the hardware and software in the context of the project partner's back yard. However, we will be designing the software in a way that would make it applicable to other irrigation projects as well. The goal is to design a general software that is compatible with different irrigation systems. The project partner has very flexible project guidelines, and encouraged refining the project to fit our needs for a project. The initial project that we have decided reflects the conversation with our project partner.

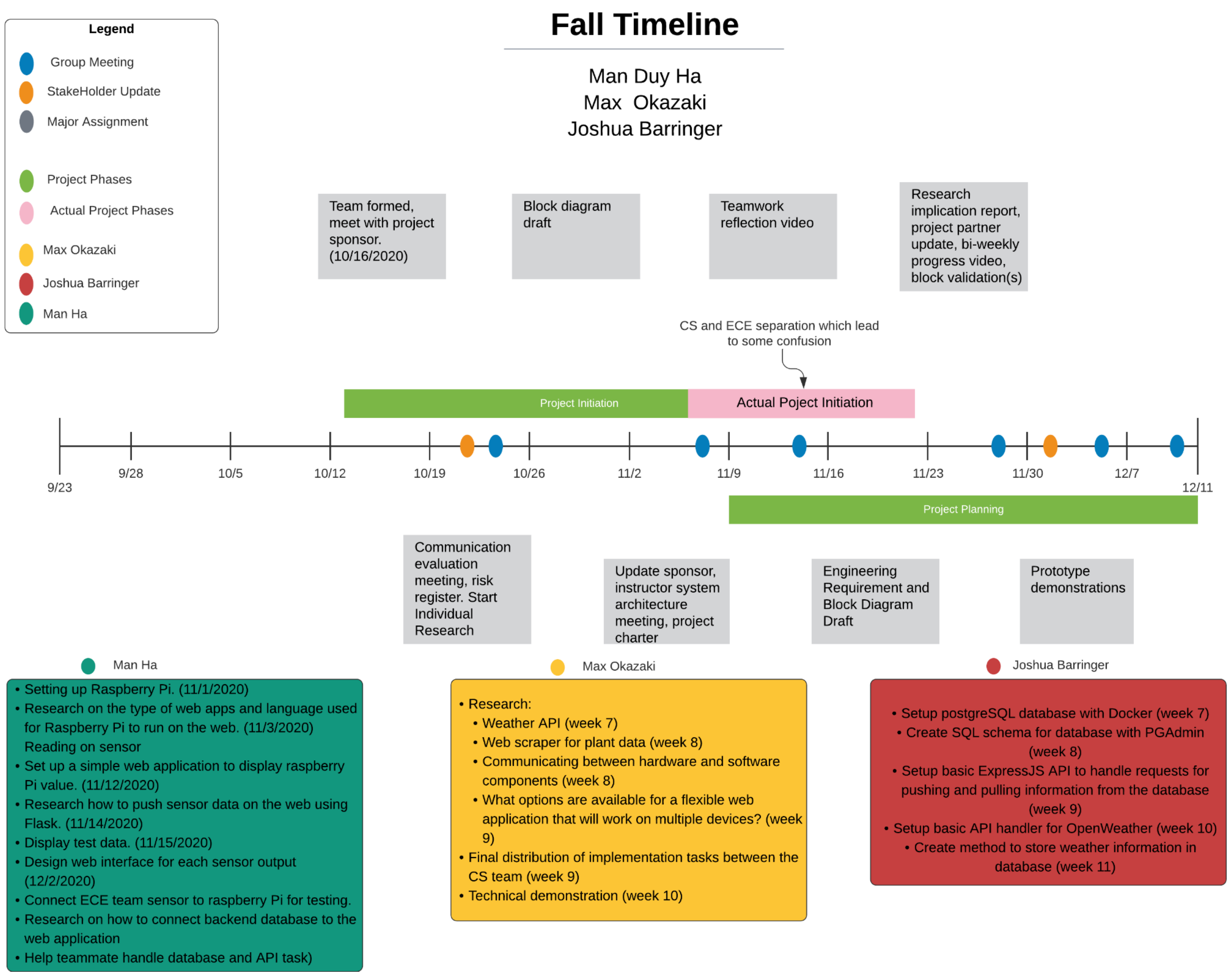
Originally with our project partner plan, we wanted to create a smart irrigation system that will provide feedback and recommendations to the user based on sensed data and available weather data. As the project progressed and we were moving closely to the execution phases, we decided to narrow down the possible things that we can implement in the time period between Winter and Spring along with focusing on our CS-side specifically. Integrating with the ECE team became harder due to the divisive nature of the assignment and one teammate that did not make it in the second term during execution phases. Due to a more definitive split between the software and hardware team, we decided to alter the scope and simply export a CSV file into a network folder. In the future, we can try to publish the website then using raspberry Pi to connect to the hosting services. Originally, we planned to make a weekly task scheduler, but we decided to opt for a more basic scheduling method due to time limitations. This is a good recommendation as we will take this into our features project.

One of the key lessons that we learned through project management is communication. We should consider meeting more often in the Winter, when the project's scope is being addressed. The way we handle this in Spring is to take a table to map out each team member's availability during the first week of the month, and use the table to schedule weekly meetings. During our meeting, we try to state what we want to do and learn, as well as what features are feasible to introduce in the two-month span between Winter and Spring. Working in ECE and CS teams can be a good way to have team members that are able to help with common problems among group members. However, be wary of poor communication between teams. Staying up-to-date with your other team is a good way to reduce the amount of work necessary to complete the project. Keeping up with the other team is a nice way to cut down on the amount of time needed to finish the job. We should have met our project partner more, set up a meeting with the project partner so we can share updates and get feedback. This will be critical to staying on track and the project partner should help you work out kinks.

Other than that, some key technical takeaways include: looking into various hosting websites and services as we weren't able to post our app publicly. Meet with your team early and understand what each team member's development environments are to ensure that the languages and packages being used are compatible with all team members. The earlier we do it the more time for bug testing and visual improvements. We encountered some errors during installation and setup and it was due to different OS systems and how we set things up. One way to do this is to make sure team members are fluent in standard version control management. Creating CSS and animations for web apps can be a more time-consuming process than one expects. In a time-crunch, visual design of a product gets pushed back. When there are close deadlines, the final product can work but look rough.

Timeline

On the following pages are images of the timeline we worked with during the Fall, Winter, and Spring terms of this project.



Fall Timeline

Man Duy Ha
Max Okazaki
Joshua Barringer

Team formed, meet with project sponsor. (10/16/2020)

Block diagram draft

Teamwork reflection video

Research implication report, project partner update, bi-weekly progress video, block validation(s)

Communication evaluation meeting, risk register. Start Individual Research

Update sponsor, instructor system architecture meeting, project charter

Engineering Requirement and Block Diagram Draft

Prototype demonstrations

● Man Ha

- Setting up Raspberry Pi. (11/1/2020)
- Research on the type of web apps and language used for Raspberry Pi to run on the web. (11/3/2020) Reading on sensor
- Set up a simple web application to display raspberry Pi value. (11/12/2020)
- Research how to push sensor data on the web using Flask. (11/14/2020)
- Display test data. (11/15/2020)
- Design web interface for each sensor output (12/2/2020)
- Connect ECE team sensor to raspberry Pi for testing.
- Research on how to connect backend database to the web application
- Help teammate handle database and API task)

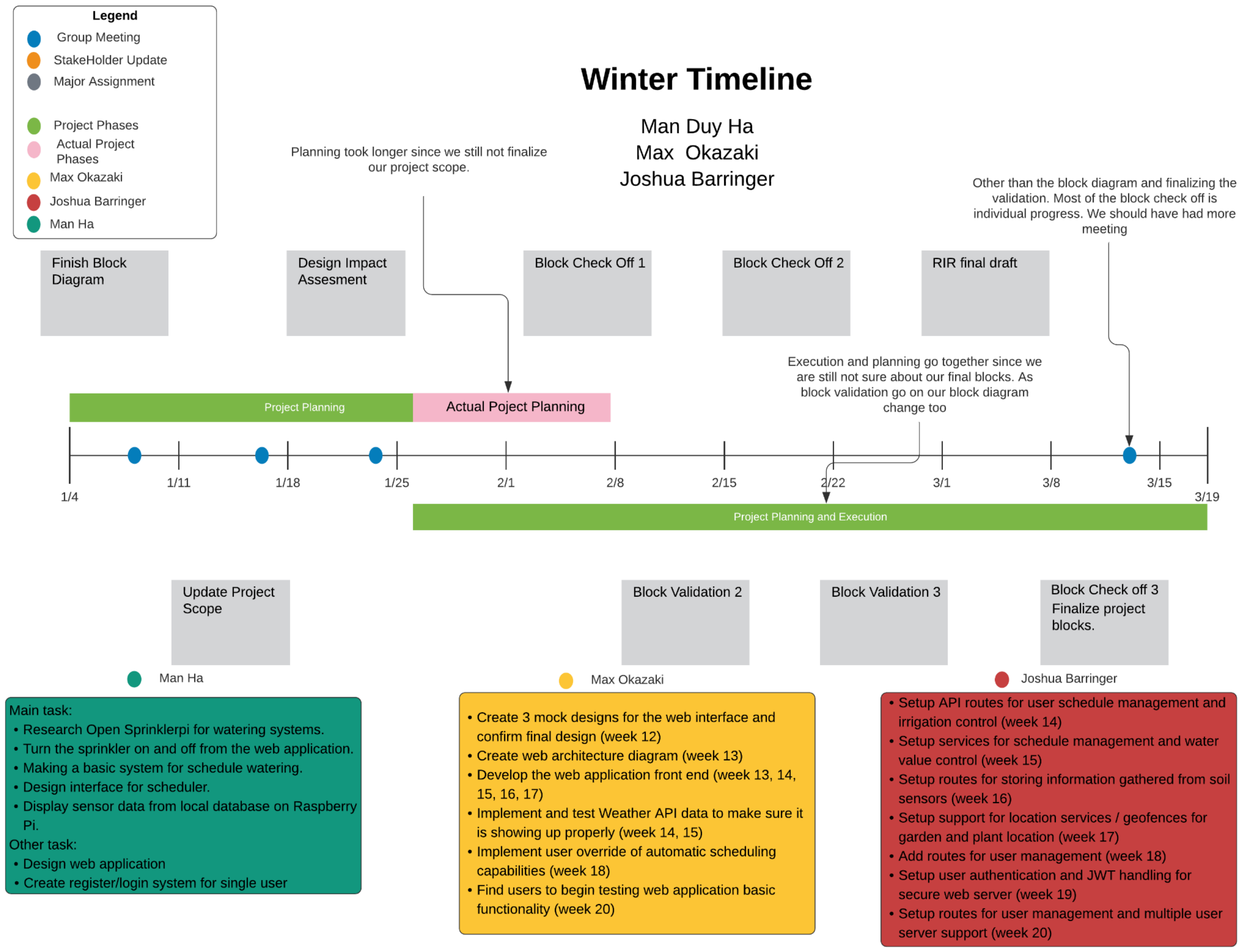
● Max Okazaki

- Research:
  - Weather API (week 7)
  - Web scraper for plant data (week 8)
  - Communicating between hardware and software components (week 8)
  - What options are available for a flexible web application that will work on multiple devices? (week 9)
- Final distribution of implementation tasks between the CS team (week 9)
- Technical demonstration (week 10)

● Joshua Barringer

- Setup postgreSQL database with Docker (week 7)
- Create SQL schema for database with PGAdmin (week 8)
- Setup basic ExpressJS API to handle requests for pushing and pulling information from the database (week 9)
- Setup basic API handler for OpenWeather (week 10)
- Create method to store weather information in database (week 11)

Figure 1: Fall Timeline



Planning took longer since we still not finalize our project scope.

Other than the block diagram and finalizing the validation. Most of the block check off is individual progress. We should have had more meeting

Execution and planning go together since we are still not sure about our final blocks. As block validation go on our block diagram change too

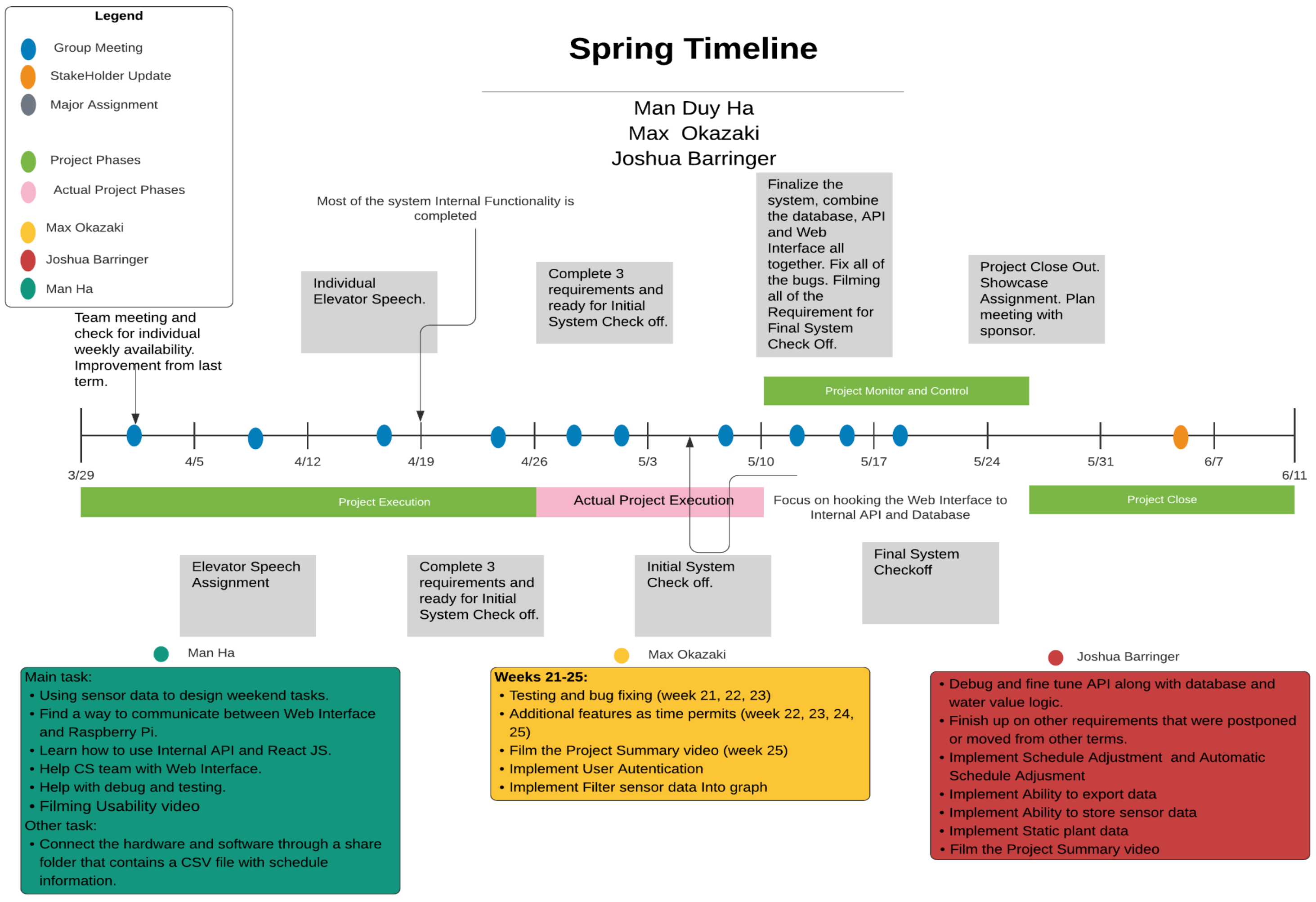
Main task:
- Research Open Sprinklerpi for watering systems.
- Turn the sprinkler on and off from the web application.
- Making a basic system for schedule watering.
- Design interface for scheduler.
- Display sensor data from local database on Raspberry Pi.

Other task:
- Design web application
- Create register/login system for single user

- Create 3 mock designs for the web interface and confirm final design (week 12)
- Create web architecture diagram (week 13)
- Develop the web application front end (week 13, 14, 15, 16, 17)
- Implement and test Weather API data to make sure it is showing up properly (week 14, 15)
- Implement user override of automatic scheduling capabilities (week 18)
- Find users to begin testing web application basic functionality (week 20)

- Setup API routes for user schedule management and irrigation control (week 14)
- Setup services for schedule management and water value control (week 15)
- Setup routes for storing information gathered from soil sensors (week 16)
- Setup support for location services / geofences for garden and plant location (week 17)
- Add routes for user management (week 18)
- Setup user authentication and JWT handling for secure web server (week 19)
- Setup routes for user management and multiple user server support (week 20)

Figure 2: Winter Timeline


Legend
● Group Meeting
● StakeHolder Update
● Major Assignment
● Project Phases
● Actual Project Phases
● Max Okazaki
● Joshua Barringer
● Man Ha
Spring Timeline
Man Duy Ha
Max Okazaki
Joshua Barringer
Most of the system Internal Functionality is completed
Individual Elevator Speech.
Complete 3 requirements and ready for Initial System Check off.
Finalize the system, combine the database, API and Web Interface all together. Fix all of the bugs. Filming all of the Requirement for Final System Check Off.
Project Close Out. Showcase Assignment. Plan meeting with sponsor.
Team meeting and check for individual weekly availability. Improvement from last term.
Project Monitor and Control
3/29
4/5
4/12
4/19
4/26
5/3
5/10
5/17
5/24
5/31
6/7
6/11
Project Execution
Actual Project Execution
Focus on hooking the Web Interface to Internal API and Database
Project Close
Elevator Speech Assignment
Complete 3 requirements and ready for Initial System Check off.
Initial System Check off.
Final System Checkoff
Man Ha
Max Okazaki
Joshua Barringer

Main task:
- Using sensor data to design weekend tasks.
- Find a way to communicate between Web Interface and Raspberry Pi.
- Learn how to use Internal API and React JS.
- Help CS team with Web Interface.
- Help with debug and testing.
- Filming Usability video

Other task:
- Connect the hardware and software through a share folder that contains a CSV file with schedule information.

Weeks 21-25:
- Testing and bug fixing (week 21, 22, 23)
- Additional features as time permits (week 22, 23, 24, 25)
- Film the Project Summary video (week 25)
- Implement User Autentication
- Implement Filter sensor data Into graph

- Debug and fine tune API along with database and water value logic.
- Finish up on other requirements that were postponed or moved from other terms.
- Implement Schedule Adjustment and Automatic Schedule Adjusment
- Implement Ability to export data
- Implement Ability to store sensor data
- Implement Static plant data
- Film the Project Summary video

Figure 3: Spring Timeline