Farmbot Project Home

Project Overview:

Project Overview

- The overall gaol of our part of the project is integrating a thermal camrea to a robot named farmbot, so that the users could collect thermal data of plants remotely. Users should access the camera on farmbot web-app and can take thermal pictures if they wish. Once there is an emergency (for example threatening temperature) the system could send alarms to user.
- ***click here for detailed project information***

Goals of the Whole Project

- Integration of an e-nose sensor within the web-app controlling the robot
- **Integration of an thermal camera within the web-app controlling the robot (our part)**
- Integration of AI solvers that automatically control the robot.

Short Cuts:

Confluence Shortcuts:

Requirements for courses

Timeline

Testings

Developments and Deployments

Products of the project

meeting minutes

code reviews

Short Cuts for Requirements

PROJECT INTROUCTION

DEVELOPMENT ENVIRONMENT

BACKLOGS AND ACEPTANCE CRITERIA

PERSONAS AND USER STORIES

SCOPE AND REQUIREMENTS

PROJECT PLANNING

Short Cuts for Other Tools

Project Github Repository

Project Trello board

Deployed Server

Project Goodle drive

Project Farmbot

Client and Supervisor

Nir Lipovetzky Client Nir Lipovetzky	Lin Li Supervisor Lin Li

Team Members

				
Dikai ZHU Scrum Master Dikai ZHU	Chengtian Jiang Software Developer Chengtian JIANG	Yuxi He Product Owner YUXI HE	Tingzheng Ren Test Leader Tingzheng Ren	Zhiyu Chen Software Developer Zhiyu Chen

Resource Links:

Team work space:

github repository

trello board

conference home

goodle drive

Other Resources:

Farmbot Control

Farmbot Training

Farmbot Official Github

Farmbot Previous Project

RoadMaps and Plans:

Project Plans:

- Our goal is to finalize the project development during Sprint 2 and Sprint 3. To achieve that, we have reorganized the user stories into user cases, meanwhile taking into account their priorities of the user cases. Our team has created a well-defined development plan, which can be viewed on Trello: FA-Boxjelly development plan.
- Click here for detailed project plan

Road Map:

Sprint 1	Friday, March 31st 2023 Team Building and Project Design
Sprint 2	Friday, April 28th 2023 Finish Half Stories and First Release
Sprint 3	Friday, May 26th 2023 Finish Project Release and Final Present
Sprint 4	Friday, June 9th 2023 Review and Self-Reflection

Recent updates:

Recent space activity

YUXI HE

Sprint 3 Demo 7 •

Chengtian JIANG

Accessing the resource 11:57 •

Zhiyu Chen

Sprint 2 Retro 26 , 2023 •

Sprint 2 Planning 26 , 2023 •

Sprint 4 Planning 26 , 2023 •

Space contributors

No contributors found for: authors on selected page(s)

REQUIREMENTS

This section contains all the information required by the project check list

Farmbot Project Home

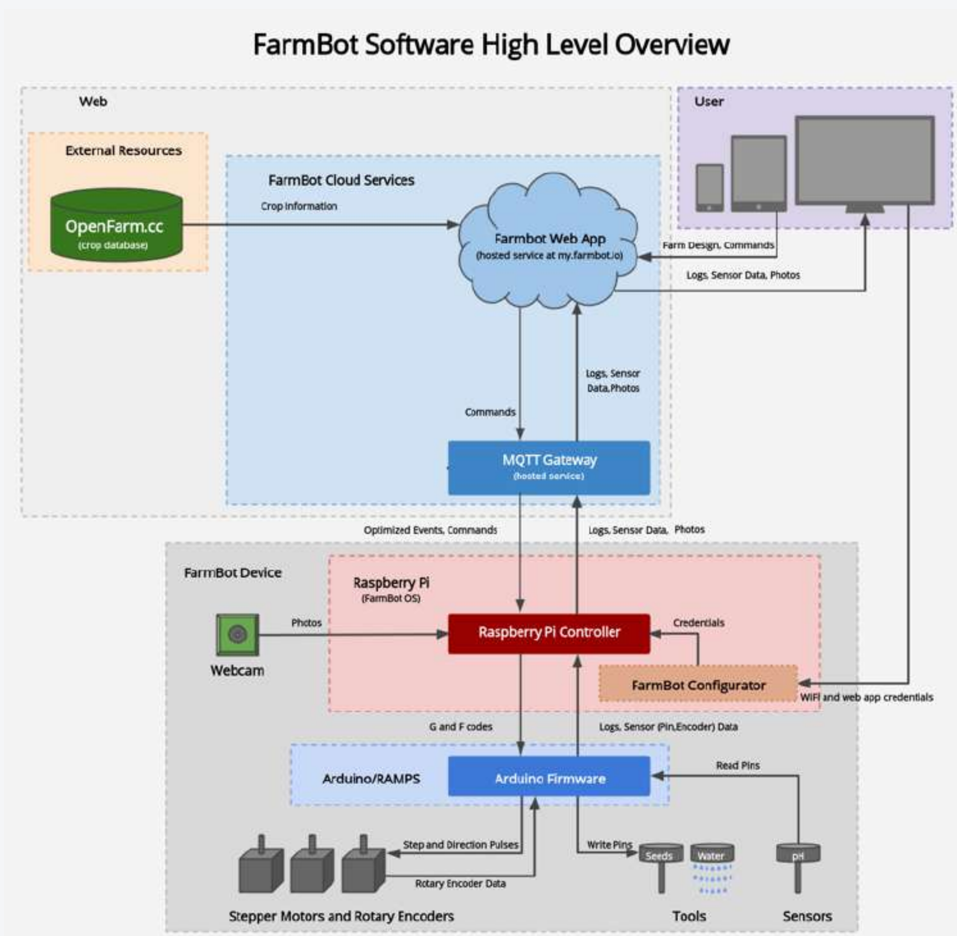
PROJECT INFORMATION

Backgound Project: Farmbot (which is our project-based

Introduction to the Farmbot Project

- The FarmBot project, initiated in 2011 by Rory Aronson, is an innovative open-source endeavor aimed at revolutionizing farming through automation. By combining robotics, web-based control, and open-source customization, FarmBot offers a unique solution to streamline farming processes and make them more efficient and sustainable.
- The FarmBot system consists of a robot that is installed on a raised bed or within a greenhouse, and it is controlled by a user-friendly web-based software application. This software, known as the web app, acts as the central interface for users to interact with the robot and manage their farming operations.
- Using the web app, users can instruct the FarmBot robot to perform a wide range of tasks. For example, they can specify the planting of seeds at precise locations, the removal of weeds, the watering of plants, and other essential farming processes. The flexibility and customization options of the web app allow farmers and researchers to adapt the FarmBot system to suit their specific needs and objectives.
- One of the key advantages of the FarmBot project is its open-source nature. The entire system, including the hardware designs, software code, and documentation, is freely available to the public. This openness fosters collaboration and empowers researchers, developers, and farmers worldwide to contribute to the project and further enhance its capabilities.
- The FarmBot system operates through a well-defined process, as illustrated in the accompanying diagram. When users initiate requests from their terminals through the web app, these requests are dispatched to an MQTT server, which acts as a message broker. The MQTT server then sorts and forwards the messages to the Raspberry Pi, a small and powerful single-board computer.
- The Raspberry Pi plays a crucial role in the FarmBot system. It receives the requests from the MQTT server and analyzes them to determine the appropriate actions to take. If the request involves interacting with sensors, such as checking soil moisture or monitoring temperature, the Raspberry Pi communicates with the Farmduino board.
- The Farmduino, running Arduino Firmware, serves as an interface between the Raspberry Pi and the physical sensors and actuators of the FarmBot robot. It receives signals from the Raspberry Pi through pin nodes and sends commands to the end sensors to carry out the requested tasks. This seamless interaction between the Farmduino board and the sensors ensures precise and accurate execution of farming operations.

  In conclusion, the FarmBot project represents a transformative approach to farming automation. By harnessing the power of robotics, web-based control, and open-source collaboration, FarmBot empowers farmers and researchers to create customized and sustainable farming solutions. With its ability to perform tasks with precision and scalability, FarmBot has the potential to revolutionize agriculture and contribute to a more efficient and environmentally friendly food production system.



Farmbot Project Structure Diagram

Software of the Farmbot Project

- Web-app:
    - The FarmBot web app serves as a user-friendly interface, providing a centralized platform for users to control and monitor their FarmBot robot remotely. It offers a range of features and functionalities to enhance the farming experience.
    - Through the web app, users have the ability to configure various settings for their FarmBot robot. This includes specifying planting patterns, defining watering schedules, and setting up weed removal procedures. By utilizing an intuitive interface, users can easily customize these parameters according to their specific farming requirements.
    - One notable feature of the web app is the integration of a thermal camera. Users can access this camera through the interface, enabling them to capture thermal images of their plants. These thermal images can be valuable for analyzing plant health and identifying any potential issues or irregularities.
- Farmbot OS:
    - FarmBot OS plays a critical role in the software architecture of the FarmBot system. It is a self-contained disk image file that incorporates a Linux system core, an Erlang OTP, and compiled Elixir scripts. This comprehensive package is constructed as a Nerves image, ensuring efficient and reliable performance.
    - As the operating system for the FarmBot robot, FarmBot OS includes all the necessary software components and drivers required to control the hardware components of the robot. This encompasses the Raspberry Pi, Farmduino (which interfaces with the sensors and actuators), and the robotic arm. The integration of these components within FarmBot OS ensures seamless communication and synchronization between the software and hardware elements of the system.
    - It's important to note that the current version of FarmBot OS does not support connections through SSH or HDMI. This means that reproducing a customized version of FarmBot OS may require substantial effort and technical expertise. However, the open-source nature of the FarmBot project, including the availability of the official source code on GitHub (https://github.com/FarmBot/farmbot_os.git), allows developers to explore and contribute to the continuous improvement and customization of the FarmBot OS.
    - To obtain the FarmBot OS image file, users can visit the official image file download page (https://my.farm.bot/os). This page provides a convenient and centralized location for accessing the latest version of the FarmBot OS image file, ensuring users have the most up-to-date software for their FarmBot system.
    - By combining the capabilities of the web app and the robustness of FarmBot OS, users can leverage the full potential of their FarmBot robot, enabling efficient and automated farming processes while enjoying a user-friendly and customizable software experience.

- 

Official Web App Page (register needed)

Hardware of the Farmbot Project

- Arms of the Farmbot:

    The FarmBot system features a versatile arm that plays a crucial role in performing various actions required to take care of plants in the field. This arm is mounted on a 3-axis platform, which allows it to move freely across the entire plant field and stop at any desired location. This flexibility enables the FarmBot robot to efficiently tend to plants at different positions within the field.

- Raspberry Pi of the bot:

    The Raspberry Pi serves as a central component of the FarmBot system, acting as a small yet powerful single-board computer. It has been customized with a specialized operating system specifically tailored for FarmBot. The Raspberry Pi plays several essential roles within the system:

    - Connection Management: It manages the connections between the FarmBot robot and the web app. It receives commands and instructions from the web app and translates them into signals that can be understood by the other hardware components of the system.
    - Data Upload: The Raspberry Pi receives sensor data from the Farmduino and uploads it to the web app. This allows users to remotely monitor and track the conditions of their crops, such as soil moisture levels, temperature readings, and ambient light levels.
- Farmduino:

The Farmduino is a customized Arduino board that interfaces with the Raspberry Pi. It serves as a crucial control unit for the FarmBot system, responsible for coordinating various hardware components and sensors. Key functions of the Farmduino include:

- Sensor Control: The Farmduino controls most of the sensors used by the FarmBot robot. This includes sensors that monitor soil moisture, temperature, ambient light levels, and other relevant environmental factors. It receives data from these sensors and communicates it to the Raspberry Pi for further processing and transmission.
- Hardware Interface: The Farmduino acts as an interface between the Raspberry Pi and other components of the FarmBot system. It receives commands and instructions from the Raspberry Pi and translates them into physical actions that the robot needs to perform. These actions can include movements of the robotic arm, activation of watering systems, and other operations required for plant care.

By combining the capabilities of the FarmBot arms, Raspberry Pi, and Farmduino, the FarmBot system provides a comprehensive hardware infrastructure to automate farming processes. The arms offer flexibility and precise positioning, while the Raspberry Pi and Farmduino facilitate seamless communication and control between the software and hardware elements of the system.

	
Raspberry Pi (green board) and Farmduino (blue board)	Arm of the Farmbot

Project Goals (what we are going to achieve)

Introduction

- Our project aims to integrate a thermal camera into the FarmBot robot to allow users to remotely collect thermal data of plants. This will enable users to access the camera through the FarmBot web-app and take thermal pictures either by manually clicking a button on the web page or by configuring an auto-collection feature. The thermal camera will capture the image and upload it to the web-app, providing thermal data for future use. The automatically collected thermal data will be used in agricultural research conducted by Professor Sigfredo Fuentes.

  Therefore:

- Project goal: Our project aims to integrate a thermal camera into agricultural management through a self-established Ethernet connection, utilizing its unique features to enhance agricultural management and decision-making capabilities.

Reasons for integrate thermal camera

- Temperature monitoring and thermal mapping: Real-time monitoring and recording of temperatures in crops, soil, and the environment. Generate thermal maps to visualize temperature distributions.
- Plant health detection: Analyze thermal maps to identify abnormal temperature areas in plants, enabling early detection of diseases, pests, or growth issues. Facilitate timely control and management.
- Irrigation optimization: Monitor temperature variations on the soil surface to determine plant water requirements. Adjust irrigation based on temperature differentials for precise optimization and water resource efficiency.

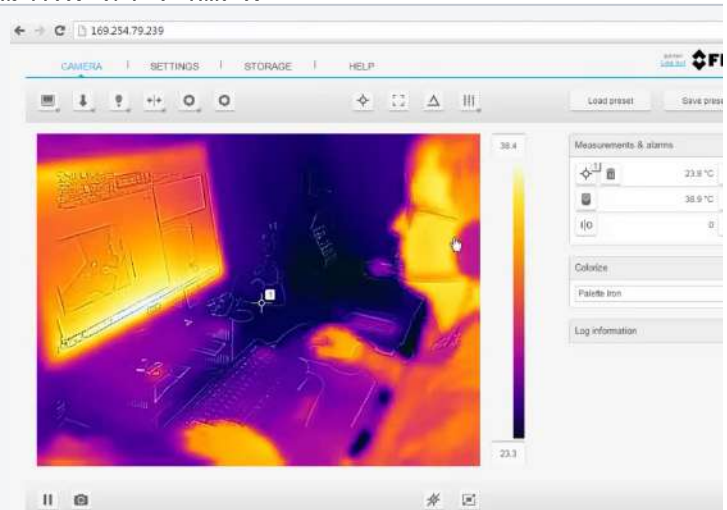
- Environmental monitoring and control: Monitor temperature distributions in greenhouses or fields to understand heat conditions and differentials. Optimize the growing environment by adjusting ventilation, shading, and heating systems.
- Data integration and decision support: Integrate and analyze temperature data with other agricultural data to provide decision support. Optimize cultivation strategies and improve yield and quality through data-driven decision-making.

  By integrating the thermal camera, we aim to enhance the accuracy and efficiency of agricultural management, striving for sustainable, productive, and high-quality agricultural production.

	
Prof. Sigfredo Fuentes	Picture for Farmbot

Thermal Camera

- The client has chosen the FLIR_Ax8 camera to be integrated into the FarmBot. This camera can automatically establish a connection to the host via Ethernet when the IP and subnet mask are the same. Although FLIR does not provide any API support for this camera, it can be configured and captures can be taken through its Ethernet website, accessible via its IP address: 80.
- The FLIR Ax8 camera offers three capture modes, including capturing a normal picture, a thermal picture, or overlaying two pictures together. The camera captures the image and stores it within the camera. Users must download the picture manually from the website.
- It is important to note that the camera requires external power as it does not run on batteries.

	
Picture of FLIR Ax8	webpage for FLIR Ax8 Ethernet

Integration Design

- A new thermal-cam page will be introduced to the FarmBot web-app. This page will allow users to control the thermal camera and view the pictures and collected thermal data.
- A control script will be developed to automatically configure and capture images with the thermal camera.
- To ensure that the control script can be run effectively, the FarmBot OS will be updated to support the new control script. By doing this, users can easily use the thermal camera to collect thermal data and automate the process of capturing images.

- To enable the thermal camera to move with the arm of the Farmbot, a holder needs to be designed and implemented. This holder will ensure that the camera remains attached to the arm while it moves, allowing it to capture thermal data from variouspositions.

Client Goals

1. Integrate the thermal imaging camera into the FarmBot system:

a. Research and select a suitable thermal imaging camera that is compatible with FarmBot.

b. Install the camera and ensure proper connection and wiring of the thermal imaging camera to the FarmBot system.

2. Develop a software module to collect and store the thermal data from the camera:

a. Create a software module to interface with the thermal imaging camera and retrieve the thermal data.

b. Implement a data storage solution to record thermal data with timestamps and plant information.

c. Develop user-friendly interfaces for customers to easily access and view the collected thermal data.

3. Analyze thermal data to monitor plant health and growth by

a. Research methods for interpreting thermal data for specific species of plants.

b. Implement algorithms to analyze thermal data and gain insights on plant health and growth.

c. Integrate the analysis module with the FarmBot system to provide real-time feedback on plant condition.

4. Use thermal data to detect pests and diseases by

a. Investigate thermal signatures of common pests and diseases that affect plants being grown.

b. Develop algorithms to identify these thermal signatures in the collected data.

c. Incorporate pest and disease detection features into the FarmBot system.

5. Implement an alarm and notification system for pest and disease detection:

a. Design a notification system that alerts customers in real time of detected pests or diseases.

b. Implement customizable alarm thresholds for different types of pests and diseases.

c. Ensure integration with the FarmBot system and the customer's preferred method of communication (e.g., email, SMS).

6. Optimize FarmBot performance based on thermal data insights:

a. Use the analyzed thermal data to adjust FarmBot's watering, fertilization and light control strategies for optimal plant growth.

b. Monitor the effectiveness of adjustments and refine algorithms for continuous improvement.

c. Ensure that the FarmBot system remains user-friendly and efficient while performing its tasks.

Technique Details

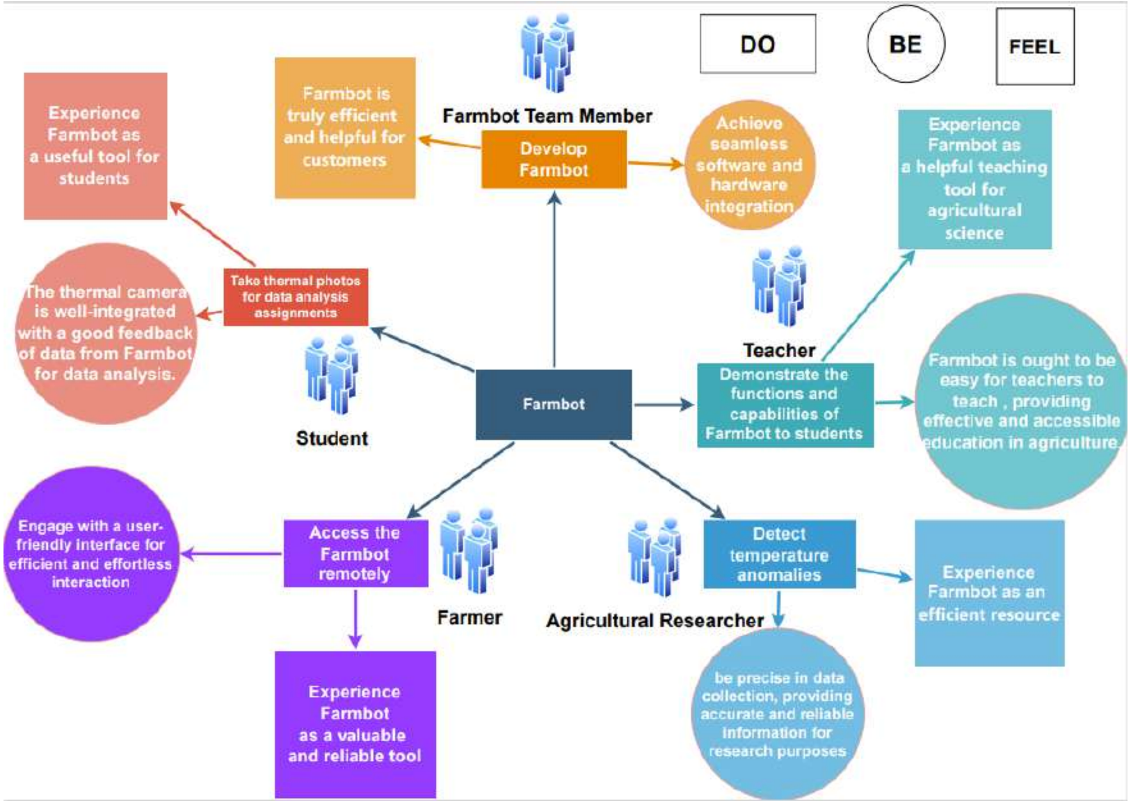
Component	Tool /Framework	Description
Farmbot Web App Deployment	Azure	Azure is a service that provides virtual servers. The farmbot web app is deployed at our own Azure server.
Web Framework / Back End Framework	Flask	Flask is a lightweight web framework for Python. It is used for the back-end development of farmbot web app.
Front End UI Components	ReactJS and TypeScript	ReactJS is a JavaScript library for building user interfaces. TypeScript is a superset of Javascript, which is widely used for large-scale web applications.
Database / Store	MongoDB	The current database used is MongoDB. Classified as a NoSQL database program, MongoDB supports a wide range of programming languages, including Python, Java, JavaScript, Ruby, and many others.
Thermal	Lua	Lua is a versatile and widely-used scripting language that can be embedded in various applications, including

Camera Controller		embedded systems. It is used in the thermal camera controller to efficiently handle tasks such as capturing, downloading, and transmitting photos from the Farmbot to the web application in this project.
Code Management	GitHub	It provides distributed version control and source code management (SCM) capabilities along with additional features of its own.
Farmbot OS Reproduce	Nerves	The process involves taking the modified Farmbot OS source code and converting it into a nerves image file.
Code Integration	Elixir and Erlang	Our integrated script needs to run on the Lua sandbox, which is supported by the system.
Thermal Camera Connection	Ethernet	The camera is limited to Ethernet connectivity and does not support any other type of connection

Do-Be-Feel List

WHO (Roles)	DO (Functional Goal)	BE (Quality Goal)	FEEL (Emotional Goal)
Farmer	Access the Farmbot remotely to manage and monitor their crops.	Engage with a user-friendly interface for efficient and effortless interaction.	Experience Farmbot as a **valuable** and **reliable** tool for improving agricultural productivity and crop management.
Agricultural Researcher	Utilize Farmbot to detect temperature anomalies for better understanding of plant health and growth conditions.	Farmbot should be precise in data collection, providing accurate and reliable information for research purposes.	Experience Farmbot as an **efficient** resource for advancing agricultural research and supporting data-driven insights.
Teacher	Allow teacher to demonstrate the functions and capabilities of Farmbot to students, enhancing their learning experience in agricultural science.	Farmbot is ought to be easy for teachers to teach and students to understand, providing effective and accessible education in agriculture.	Experience Farmbot as a **helpful** teaching tool for agricultural science, making student engaged and interested in the subject.
Student	Use Farmbot to take thermal photos for data analysis assignments, enhancing their practical skills in agricultural studies.	The thermal camera is well-integrated with a good feedback of data from Farmbot for data analysis.	Experience Farmbot as a **useful** tool for students, providing help within their agricultural studies.
Farmbot Team Member	Develop Farmbot using a high-quality coding style to ensure efficient performance and maintainability.	Achieve seamless software and hardware integration within Farmbot, delivering a robust and reliable system.	Create a Farmbot that is truly **efficient** and **helpful** for customers, contributing to their needs and satisfaction.

Motivational Model



Farmbot Project Home
Parent Page: Requirement

DEVELOPMENT ENVIRONMENT

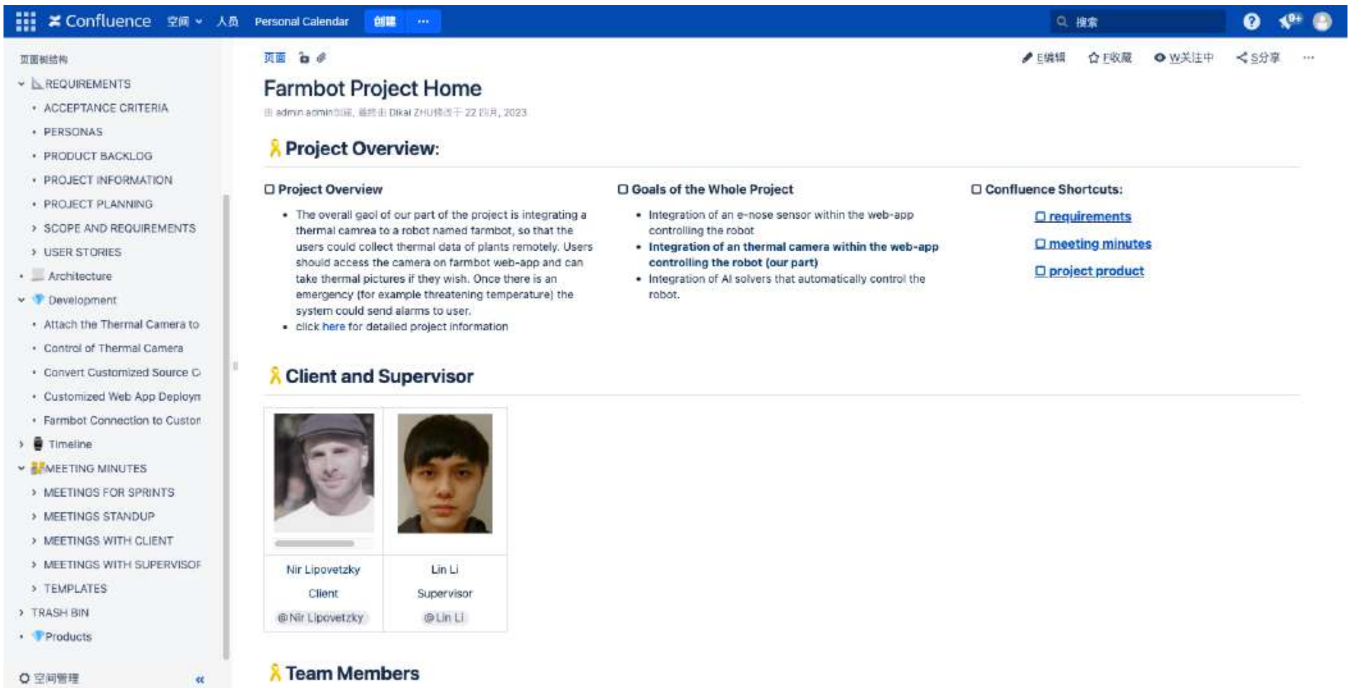
This page is about the overall development environment of the project, which includes the following:

1. Project documentation
2. Project progress tracking
3. Project code hosting
4. Project deployment URL

Project documentation:

All development-related documents are recorded on Confluence, which basically includes 6 aspects: REQUIREMENTS, Architecture, Development, Timeline, MEETING MINUTES, and Products.

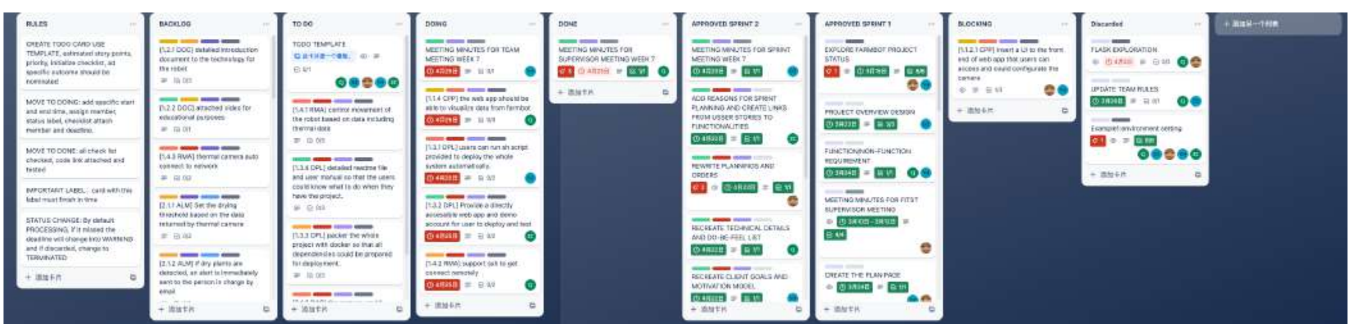
Farmbot Project Home



Project progress tracking:

All development progress tracking and task assignments are recorded on Trello.

Farmbot Trello Dashboard



Project code hosting:

All development-related code is hosted on GitHub.

FA-Boxjelly Github Page

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

COMP90082-2023-SM1 / **FA-Boxjelly** Private

Watch 1 · Fork 0 · Star 1

Code · Issues · Pull requests · Actions · Projects · Security · Insights · Settings

main · 8 branches · 7 tags · Go to file · Add file · Code

Your main branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. Learn more
Protect this branch

ecolismith reupload of the same file on the root page · 22f1644 last week · 29 commits

data samples	Create sample_data.md	last month
docs	reupload of the same file on the root page	last week
prototypes	Create sample_high_fidelity.py	last month
src	Create sample_code.py	last month
tests	Create sample_test.py	last month
ui	Create sample_ui.png	last month
COMP900822023SM1FABoxJelly-...	Add files via upload	last month
README.md	Update README.md	last month

FA-Boxjelly

Project overview

The overall gaol of our part of the project is integrating a thermal camrea to a robot named farmbot, so that the users could collect thermal data of plants remotely. Users should access the camera on farmbot web-app and can take thermal pictures if they wish. Once there is an emergency (for example threatening temperature) the system could send alarms to user.

More information :
https://confluence.cis.unimelb.edu.au:8443/display/COMP900822023SM1FABoxJelly/Farmbot+Project+Home

Goals of the Whole Project

- Integration of an e-nose sensor within the web-app controlling the robot
- **Integration of an thermal camera within the web-app controlling the robot (our part)**
- Integration of AI solvers that automatically control the robot.

Client : Nir Lipovetzky

Supervisor : Lin Li

Team Members

Name	Role	E-mail
Dikai ZHU	Scrum Master	dikai.zhu@student.unimelb.edu.au
Chengtian Jiang	Software Developer	chengtian.jiang@student.unimelb.edu.au
Yuxi He	Product Owner	yuxi.he2@student.unimelb.edu.au
Tingzheng Ren	Test Leader	tingzheng.ren@student.unimelb.edu.au
Zhiyu Chen	Software Developer	zhiyu.chen3@student.unimelb.edu.au

Repository structure and navigation

├── **docs/** # Documentation files (you can create subfolders here to organize your requirements)
├── **src/** # src code
├── **tests/** # User/system tests
├── **prototypes/low fidelity/** # low fidelity files (screens, mockups and so on)
├── **prototypes/high fidelity/** # high fidelity files (screens, source files and so on)
├── **ui/** # All the images created for the prototypes (icons, fonts, backgrounds... should be here. This is different from the prototypes' folders. These are the graphical elements that goes into the prototypes)
├── **data samples/** # Documents you need to generate with all the data (inputs) necessary to simulate/demonstrate your prototype (whatever can be provided as an input in your prototype)
└── **README.md** #(this file must be updated at all times. please, make sure you explain your github structure here and generate changelogs for each sprint before you tag it)

Branch details:

- main: The place to release the final version.
- vanilla_farmbot_app: Default deployment code, project starting point

Baseline tag convention: COMP90082_2023_SM1_<TwoDigits>_<team>_BL_<sprint>
- BL means BASELINE. A baseline is a reference point in the software development life cycle marked by the completion and formal approval of a set of predefined work products
- Example of TAG in this subject: COMP90082_2023_SM1_CM_Wombat_BL_SPRINT1

Current Status:
- Just started and without any implementations

About

our boxjelly team for farmbot project

Readme · 1 star · 1 watching · 0 forks

Releases 4

Sprint 1 Latest on Mar 24

+ 3 releases

Packages

No packages published
Publish your first package

Contributors 2

ecolismith Dikai Zhu
TingzhengRen TingzhengRen

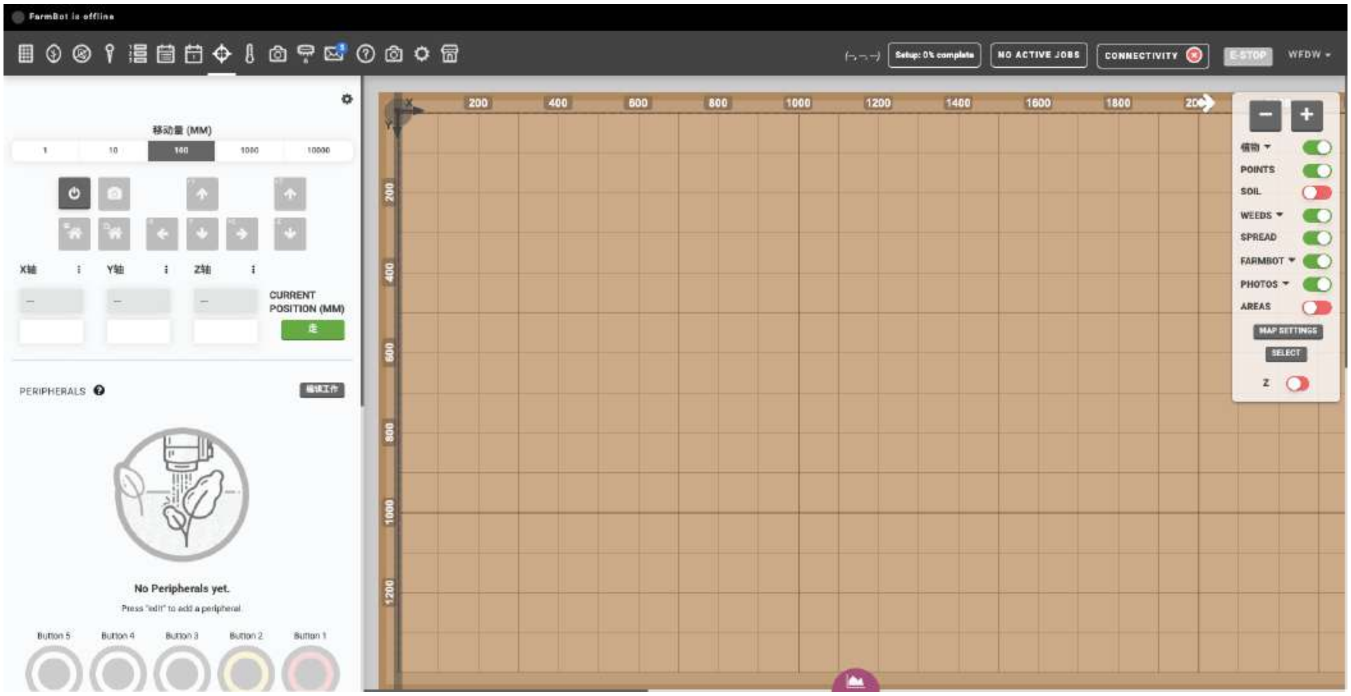
Languages

Python 100.0%

Project deployment URL

The links to all developed pages are as follows:

Project Entrance



Farmbot Project Home

Parent Page: Requirement

BACKLOGS

- PRODUCT BACKLOG
- SPRINT BACKLOG
- ACCEPTANCE CRITERIA

PRODUCT BACKLOG

User Story ID	User Stories	Story Priority	Related to functinalities.	Estimate Story Points In Total	
1.1	As a teacher/researcher/farmer, I want to use the camera to capture pictures of different stages of crops, so that the students can learn the lifecycle of crops.	MEDIUM	Capture pictures, CPP the functionality of capturing pictures is essential to achieve the desired outcome of documenting different stages of crops. It outlines the requirements for a camera feature that enables the user	21 (determined by summing up story points of all divided tasks see sprint backlog for details)	
1.2	As a teacher, I want to show the students the automated process of planting, watering, and harvesting crops, so that they can understand how technology works in agricultural production.	MEDIUM	Documentations, DOC The functionality of documentation is that documentation serves as a means to capture and present information about the automated process of planting, watering, and harvesting crops.	5	
1.3	As a teacher, I want to deploy the Farmbot in class so that students can gain hands-on experience in high-tech farming.	HIGH	Deployment, DPL To effectively demonstrate hands-on experience to others, it is essential to deploy the system and have access to a test account. Only through this process can users truly showcase their practical understanding of the system.	16	
1.4	As a farmer/researcher/teacher, I want to farm remotely so that I can reduce costs and get more time.	HIGH	Remote Access, RMA By providing remote access to the integrated thermal camera, the farmbot empowers users to engage in remote farming activities. This feature allows individuals to oversee and manage their farms from a distance, leveraging the capabilities of the thermal camera to monitor crucial aspects such as temperature and crop health.	20	
2.1	As a teacher/researcher, I want to receive alarms when the crops are dry so that I can remind students that they forgot to water the plants.	LOW	Alarms, ALM The implemented alarm system serves as a valuable tool for notifying users of any drought-affected plants. Through this feature, users are promptly alerted to plants experiencing drought conditions, enabling them to take immediate action and provide necessary interventions to ensure the plants' well-being.	8	
2.2	As a sustainable farm owner, I want to use the thermal camera to detect whether the crops are infected with pests and diseases so that I can timely detect and treatment of pests and diseases.	LOW	Disease Detection, DTC Analysis with thermal data and extract disease, then can the user access timely detect of pests and disease of the plants.	20	
2.3	As a sustainable farm owner, I want to use thermal cameras to help me accurately control the irrigation water quantity so that I can accurately control irrigation volume, thereby improving crop growth efficiency.	MEDIUM	Thermal Based Controls, TBC Thermal data has to be integrated with the control system of the robot, so that the robot can do the plating and irrigation according to the timely thermal data, and only in this way, precise farming can be achieved.	20	
2.4	As a farmer/researcher, I want to use the infrared imaging function to detect the temperature of the environment including the plant and the soil so that I can improve the quality and nutritional value of crops through precision agriculture techniques	HIGH	Thermal Picture Analysis, TPA To conduct an analysis, simply relying on a thermal picture itself is insufficient. Instead, it is necessary to extract thermal data from the pictures in order to perform a comprehensive analysis of the plants.	17	
3.1	As a teacher/researcher, I want to collect thermal data through farmbot so that it could be used for further study.	HIGH	Data Auto Collection, DAC By utilizing a farmbot to automatically collect thermal data, the teacher/researcher can efficiently gather the necessary information for their study. The automation feature of data collection eliminates the need for manual measurements, saving time and effort.	12	
3.2	As an agricultural researcher, I want to manage my database related to the plants so that I have a clean and reliable dataset for my study.	HIGH	Database Management, DBM	10	

			By effectively managing the database related to the plants, the researcher can ensure that they have a clean and reliable dataset for their study. Database management allows the researcher to organize, store, and retrieve plant-related information in a structured manner, facilitating easy access and analysis of the data.		

Farmbot Project Home

Parent Page: Requirement

SPRINT BACKLOG

Versions

1.1	Rename column user case ID into task ID, rename column user description to task description	2023-03-24
1.0	Initialize user cases based on user stories version 2	2023-03-22

Some rules for this table:

1 Story point equals around 0.5 days of work (3 hours)

Small-sized user case is less than 1 day. (1 to 2 points)

Medium-sized user case is 2 to 3 days (3 to 6 points)

Large-sized user case is 4 to 5 days (7 to 10 points)

Sprint ID	User Story ID	User Stories	Related to functinalities.	Estimate Story Points In Total	Task ID	Task Description	Supplementary notes	Story Points	Size	Status
SPRINT 2	1.1	As a teacher /researcher /farmer, I want to use the camera to capture pictures of different stages of crops, so that the students can learn the lifecycle of crops.	Capture pictures, CPP the functionality of capturing pictures is essential to achieve the desired outcome of documenting different stages of crops. It outlines the requirements for a camera feature that enables the user	21 (determined by summing up story points of all divided tasks)	1.1.1	The camera should be attached to the robot so that the camera could move with the robot	By moving the robot arm. we can adjust the position of the camera	2	SMALL	*DONE*
					1.1.2	Insert a UI to the front end of a web app that users can access and could configure the camera	Configuration includes starting and stopping, configuring camera settings, and taking pictures with the camera.	8	LARGE	*DONE*
					1.1.3	After clicking the take picture button, the system could set the camera to real picture mode and take a picture		4	MEDIUM	*DONE*
					1.1.4	The web app should be able to visualize data from farmbot		8	LARGE	*DONE*
	3.2	As an agricultural researcher, I want to manage my database related to the plants so that I have a clean and reliable dataset for my study.	Database Management, DBM By effectively managing the database related to the plants, the researcher can ensure that they have a clean and reliable dataset for their study. Database management allows the researcher to organize, store, and retrieve plant-related information in a structured manner, facilitating easy access and analysis of the data.	10	3.2.1	The data collected from farmbot should be stored in a database	A database that could hold all data including the raw image data.	4	MEDIUM	*DOING*
					3.2.2	The data, including the results of any analysis and raw images, should be well-organized and interconnected.	The design of the data structure should be clear and accepted by the client.	2	SMALL	*TO DO*
					3.2.3	The web application should have the capability to select, create, update, and delete any data that has been stored.	This should be accessed through a simple UI on the front, do not need to support multiple users at the same time (maybe in the future)	4	MEDIUM	*TO DO*
	1.3	As a teacher, I want to deploy the Farmbot in class so that students can gain hands-on experience in high-tech farming.	Deployment, DPL To effectively demonstrate hands-on experience to others, it is essential to deploy the system and have access to a test account. Only through this process can users truly showcase their practical understanding of the system.	16	1.3.1	Users can run scripts provided to deploy the whole system automatically.		8	LARGE	*DONE*
					1.3.2	Provide a directly accessible web app and demo account for users to deploy and test		2	SMALL	*DONE*
					1.3.3	packer the whole project with docker so that all dependencies		5	MEDIUM	*TO DO*

						could be prepared for deployment.				
					1.3.4	Detailed ReadMe file and user manual so that the users could know what to do when they have the project.		2	SMALL	*DONE*
	1.4	As a farmer /researcher /teacher, I want to farm remotely so that I can reduce costs and get more time.	Remote Access, RMA By providing remote access to the integrated thermal camera, the farmbot empowers users to engage in remote farming activities. This feature allows individuals to oversee and manage their farms from a distance, leveraging the capabilities of the thermal camera to monitor crucial aspects such as temperature and crop health.	20	1.4.1	Control movement of the robot based on data including thermal data	Keep detecting; ai planning for abnormal status	8	LARGE	*TO DO*
					1.4.2	Support ssh to get connect remotely		4	MEDIUM	*DONE*
					1.4.3	Thermal camera auto-connects to the network	Plug and play	8	LARGE	*TO DO*
S P R I N T 3	1.2	As a teacher, I want to show the students the automated process of planting, watering, and harvesting crops, so that they can understand how technology works in agricultural production.	Documentations, DOC The functionality of documentation is that documentation serves as a means to capture and present information about the automated process of planting, watering, and harvesting crops.	5	1.2.1	Detailed introduction document to the technology for the robot		4	MEDIUM	*DONE*
					1.2.2	Attach to video for educational purposes		1	SMALL	*TO DO*
	2.1	As a teacher /researcher, I want to receive alarms when the crops are dry so that I can remind students that they forgot to water the plants.	Alarms, ALM The implemented alarm system serves as a valuable tool for notifying users of any drought-affected plants. Through this feature, users are promptly alerted to plants experiencing drought conditions, enabling them to take immediate action and provide necessary interventions to ensure the plants' well-being.	8	2.1.1	Set the drying threshold based on the data returned by a thermal camera		4	MEDIUM	*TO DO*
					2.1.2	If dry plants are detected, an alert is immediately sent to the person in charge by email		4	MEDIUM	*TO DO*
ALSO Sprint 3	2.4	As a farmer /researcher, I want to use the infrared imaging function to detect the temperature of the environment including the plant and the soil so that I can improve the quality and nutritional value of crops through precision agriculture techniques	Thermal Picture Analysis, TPA To conduct an analysis, simply relying on a thermal picture itself is insufficient. Instead, it is necessary to extract thermal data from the pictures in order to perform a comprehensive analysis of the plants.	17	2.4.1	Continuous monitoring of soil conditions with stream data from a thermal camera.		4	MEDIUM	*TO DO*
					2.4.2	Identify each plant on the thermal picture with computer vision algorithms.	Ensure that each plant is individually recognized through image-processing techniques	4	MEDIUM	*TO DO*
					2.4.3	Analyzing the thermal energy of each plant's leaves using a thermal imaging camera.	Help to assess the environmental stress on the plants.	8	LARGE	*TO DO*
					2.4.4	Calculate the environmental stress on each plant and save the data.	This data can be used to guide other processes and research.	1	SMALL	*TO DO*
	3.1	As a teacher /researcher, I want to collect thermal data through farmbot so that it could be used for further study.	Data Auto Collection, DAC By utilizing a farmbot to automatically collect thermal data, the teacher /researcher can efficiently gather the necessary information for their study. The automation feature of data collection eliminates the need for manual	12	3.1.1	The thermal camera should be able to take thermal pictures automatically.		2	SMALL	*TO DO*
					3.1.2	The user should have access to control the frequency of auto-collection.		4	MEDIUM	*TO DO*
ALSO					3.1.3	The camera would categorize		8	LARGE	*DONE*

Sprint 3 **ALSO Sprint 3**			measurements, saving time and effort.			data and send the data to the web app so that it could be saved and the user could download it.				
S P R I N T 4	2.3	As a sustainable farm owner, I want to use thermal cameras to help me accurately control the irrigation water quantity so that I can accurately control irrigation volume, thereby improving crop growth efficiency.	Thermal Based Controls, TBC Thermal data has to be integrated with the control system of the robot, so that the robot can do the plating and irrigation according to the timely thermal data, and only in this way, precise farming can be achieved.	20	2.3.1	Use Farmbot software to set up the irrigation program, including irrigation time and water volume.		4	MEDIUM	*TO DO*
					2.3.2	Set up the camera program in Farmbot software to automatically capture thermal images of plants during irrigation.		4	MEDIUM	*TO DO*
					2.3.3	Analyze the thermal images of plants captured automatically during irrigation to determine their water status. Thermal images can display the surface temperature of plants, and based on the temperature changes, it is possible to determine whether the plants need irrigation.		8	LARGE	*TO DO*
					2.3.4	Based on the thermal images and the irrigation program settings, Farmbot can automatically control the irrigation water amount. If the plant needs more water, Farmbot will increase the irrigation water amount, and vice versa		4	MEDIUM	*TO DO*
	2.2	As a sustainable farm owner, I want to use the thermal camera to detect whether the crops are infected with pests and diseases so that I can timely detect and treatment of pests and diseases.	Disease Detection, DTC Analysis with thermal data and extract disease, then can the user access timely detect of pests and disease of the plants.	20	2.2.1	Data pre-processing of data on plant body surface temperature distribution		8	LARGE	*TO DO*
					2.2.2	Analysis of changes in plant body surface temperature to determine the presence and extent of pests and diseases		8	LARGE	*TO DO*
					2.2.3	If any pests and diseases are detected, an alert is immediately sent to the farm owner by email		4	MEDIUM	*TO DO*

Farmbot Project Home

Parent Page: Requirement

ACCEPTANCE CRITERIA

Versions

VersionID	Description	Date
2.0	1. Update test cases for following features 2. Modify user story add test cases	2022-04-24
1.0	Created first version	2022-03-15

User Story Global ID	User Story	Acceptance Criteria ID	Given	When	Then	Test Result	Comments
1	As a teacher/researcher/farmer, I want to use the camera to capture pictures of different stages of crops, so that the students can learn the lifecycle of crops.	1.1	I have a camera on the farmbot	I click the capture button	The teacher /researcher/farmer can see the photo of the crops on farmbot web app	PASS	
2	As a teacher, I want to show the students the automated process of planting, watering, and harvesting crops, so that they can understand how technology works in agricultural production.	2.1	I have logged in the web app	I click the planting /watering /harvesting button	The farmbot will execute the planting /watering/harvesting actions	PASS	
3	As a teacher, I want to deploy the Farmbot in class so that students can gain hands-on experience in high-tech farming.	3.1	I have farmbot	I deploy the web app	I can teach students about how to use farmbot with the web app interface	PASS	
4	As a farmer/researcher/teacher, I want to farm remotely so that I can reduce costs and get more time.	4.1	I have access to farmbot remotely	I click the buttons on the web app	I can control the farmbot by different buttons	PASS	
5	As a teacher/researcher, I want to receive alarms when the crops are dry so that I can remind students that they forgot to water the plants.	5.1	I have connect the web app to the farmbot	The crop needs water	The farmbot will detect the needs and send alarm to the web app		
6	As a sustainable farm owner, I want to use the thermal camera to detect whether the crops are infected with pests and diseases so that I can timely detect and treatment of pests and diseases.	6.1	I have logged in the web app	Pests or diseases on crops	The farmbot can detect the situation and give me feedbacks		
7	As a sustainable farm owner, I want to use thermal cameras to help me accurately control the irrigation water quantity so that I can accurately control irrigation volume, thereby improving crop growth efficiency.	7.1	I have logged in the web app	I click the irrigation button	I can control the irrigation volume of farmbot by the button of irrigation controller on the farmbot web app	PASS	
8	As a farmer/researcher, I want to use the infrared imaging function to detect the temperature of the environment including the plant and the soil so that I can improve the quality and nutritional value of crops through precision agriculture techniques	8.1	I have logged in the web app	I start the infrared imaging function by clicking the button	The farmbot can detect the temperature of environment and return the data back.		
9	As a teacher/researcher, I want to collect thermal data through farmbot so that it could be used for further study.	9.1	I have logged in the web app	I click the button of thermal camera	The farmbot will take a phtot of crops and send back the data of the photo		
10	As an agricultural researcher, I want to manage my database related to the plants so that I have a clean and reliable dataset for my study.	10.1	I have deployed my farmbot	I want to do research about the plants	The farmbot database can provide a clean and reliable dataset collected from farmbot		

Farmbot Project Home

Parent Page: Requirement

PERSONAS AND USER STORIES

PERSONAS

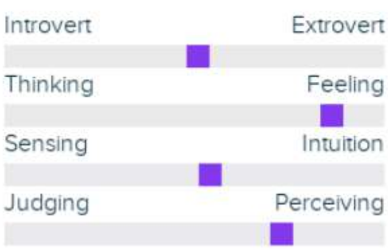
Type	Bio	Goals	Frustrations
Sustainable Farm Owner	Samantha is 36-year-old lady, and owns a small-scale sustainable farm that grows a variety of fruits and vegetables. She is passionate about using environmentally friendly methods and wants to ensure that her crops are free from pests and diseases without the use of harmful chemicals.	1. Avoid using harmful chemicals 2. Reduce the amount of time spent monitoring farm manually 3. Quickly detect any pests or diseases	1. Focus on Cost and Time 2. Competing against larger farms with more resources and funding 3. Provide healthy and chemical-free produce
High School Teacher	Yamanaka Sawako is a 35-year-old high school teacher in Tokyo who devoted to teaching agriculture. Grown up in coutryside, she worked at sakura high school as a teacher after complete her degree, . She is responsible for leading the school's agriculture program, which provides students with hands-on learning experiences by allowing them to plant and harvest real crops.	1. Ensure the crops are well-cared 2. Give student better understanding of farming and agriculture 3. Enable students stay update to latest agricultural technologies.	1. Large class size to manage 2. Student not proper care the crops 3. Not familiar with robots
Agrcultural Researcher	Dr. John Smith is 39-year-old agricultural researcher based in Australia. Growing up on a family farm in rural New South Wales, John developed an early interest in the natural world. After completing a degree in agricultural science at the University of Sydney, John went on to earn his Ph.D. in plant genetics from the University of Melbourne. His research has focused on developing new plant varieties that are resistant to pests and diseases, as well as improving crop yields through precision agriculture techniques.	1. Develop new plant varieties that are resistant to pests and diseases 2. Improve the quality and nutritional value of crops through precision agriculture techniques 3. Increase the efficiency and productivity of farming practices through the use of innovative technologies	1. Limited time and resource to collect data from crops. 2. Lack of effective alarm that can quickly notify crop emergencies. 3. Currently equipments are not precise enough.

Yamanaka Sawako

"Education is not about filling a bucket, it's about lighting a fire."

Age: **35**
Work: **High School Teacher**
Family: **Married**
Location: **Japan**

Personality



Friendly | Hardworking

Goals

- Ensure the crops are well-cared.
- Give student better understanding of farming and agriculture.
- Enable students stay update to latest agricultural technologies.

Frustrations

- Large class size to manage.
- Student not proper care the crops.
- Not familiar with robots.

Bio

Yamanaka Sawako is a 35-year-old high school teacher in Tokyo who devoted to teaching agriculture. Grown up in coutryside, she worked at sakura high school as a teacher after complete her degree, . She is responsible for leading the school's agriculture program, which provides students with hands-on learning experiences by allowing them to plant and harvest real crops.

Samantha

"We have not inherited the earth from our ancestors, we have borrowed it from our children."

Age: **36**
Work: **Sustainable Farm Owner**
Family: **Married**
Location: **India**

Personality

Introvert — Extrovert
Thinking — Feeling
Sensing — Intuition
Judging — Perceiving

Hardworking | Enthusiastic

Goals

- Avoid using harmful chemicals
- Reduce the amount of time spent monitoring farm manually
- Quickly detect any pests or diseases

Frustrations

- Focus on Cost and Time
- Competing against larger farms with more resources and funding
- Provide healthy and chemical-free produce

Bio

Samantha is 36-year-old lady, and owns a small-scale sustainable farm that grows a variety of fruits and vegetables. She is passionate about using environmentally friendly methods and wants to ensure that her crops are free from pests and diseases without the use of harmful chemicals.

John Smith

Professional | Enthusiastic

"The future of agriculture is not about better recipes, it's about a better understanding of ecology."

Age: **39**
Work: **Agricultural researcher**
Family: **Married**
Location: **Australia**

Personality

Introvert — Extrovert
Thinking — Feeling
Sensing — Intuition
Judging — Perceiving

Goals

- Develop new plant varieties that are resistant to pests and diseases.
- Improve the quality and nutritional value of crops through precision agriculture techniques.
- Increase the efficiency and productivity of farming practices through the use of innovative technologies.

Frustrations

- Limited time and resource to collect data from crops.
- Lack of effective alarm that can quickly notify crop emergencies.
- Currently equipments are not precise enough.

Bio

Dr. John Smith is 39-year-old agricultural researcher based in Australia. Growing up on a family farm in rural New South Wales, John developed an early interest in the natural world. After completing a degree in agricultural science at the University of Sydney, John went on to earn his Ph.D. in plant genetics from the University of Melbourne. His research has focused on developing new plant varieties that are resistant to pests and diseases, as well as improving crop yields through precision agriculture techniques.

Farmbot Project Home
Parent Page: Requirement

USER STORIES VERSION 2

Versions

2.3	Added documents for what user story is generated with chatGPT	2023-02-24
2.2	Reset story points according to the product backlog and story themes	2023-02-24
2.1	Reset Priorities and the color of priority and status	2023-02-23
2.0	Merge user stories according to guidance from the client, rewrite some user stories, reorder the IDs	2023-03-22
1.0	Initialize a user story table based on the current understanding of requirements, goal model, and persona.	2023-03-21

- The user stories were first created in March 2023 after the meeting with the client. As the team got started on coding partial functions and we followed the agile developing approach, some user stories were updated and optimized. Also, new user stories were created after a discussion with the client.

Feature	Global ID	Local ID	User Stories	User Story Themes	Priority	Story Points	Remarks	Update	Version	Status	Generated With ChatGPT?
Control	1	1.1	As a teacher/researcher /farmer, I want to use the camera to capture pictures of different stages of crops, so that the students can learn the lifecycle of crops.	Capture pictures, CPP	MEDIUM	21		2023.03.22	2.1	*TO DO*	YES
	2	1.2	As a teacher, I want to show the students the automated process of planting, watering, and harvesting crops, so that they can understand how technology works in agricultural production.	Documentations, DOC	MEDIUM	5		2023.03.22	2.1	*TO DO*	YES
	3	1.3	As a teacher, I want to deploy the Farmbot in class so that students can gain hands-on experience in high-tech farming.	Deployment, DPL	HIGH	16		2023.03.22	2.1	*TO DO*	YES
	4	1.4	As a farmer/researcher /teacher, I want to farm remotely so that I can reduce costs and get more time.	Remote Access, RMA	HIGH	20		2023.03.22	2.1	*TO DO*	NO
Monitor	5	2.1	As a teacher/researcher, I want to receive alarms when the crops are dry so that I can remind students that they forgot to water the plants.	Alarms, ALM	LOW	8		2023.03.22	2.1	*TO DO*	NO
	6	2.2	As a sustainable farm owner, I want to use the thermal camera to detect whether the crops are infected with pests and diseases so that I can timely detect and treatment of pests and diseases.	Disease Detection, DTC	LOW	20		2023.03.22	2.1	*TO DO*	NO
	7	2.3	As a sustainable farm owner, I want to use thermal cameras to help me accurately control the irrigation water quantity so that I can accurately control irrigation volume, thereby improving crop growth efficiency.	Thermal Based Controls, TBC	MEDIUM	20		2023.03.22	2.1	*TO DO*	NO
	8	2.4	As a farmer/researcher, I want to use the infrared imaging function to detect the temperature of the environment including the plant and the soil so that I can improve the quality and nutritional value of crops through precision agriculture techniques	Thermal Picture Analysis, TPA	HIGH	17		2023.03.22	2.1	*TO DO*	NO
Data	9	3.1	As a teacher/researcher, I want to collect thermal data through farmbot so that it could be used for further study.	Data Auto Collection, DAC	HIGH	12		2023.3.22	2.1	*TO DO*	NO
	10	3.2	As an agricultural researcher, I want to manage my database related to the plants	Database Management, DBM	HIGH	10		2023.3.22	2.1	*TO DO*	NO

			so that I have a clean and reliable dataset for my study.								

Farmbot Project Home

SCOPE AND REQUIREMENTS

Farmbot Project Home

Parent Page: Requirement

FUNCTIONAL REQUIREMENTS

Version1

Epic	Functions	User stories	Story Points
Integration of Thermal Camera in Farmbot	The user should be able to control the thermal on the web app with interface operations	1.3 As a teacher, I want to deploy the Farmbot in class so that students can gain hands-on experience in high-tech farming.	16
	The photos taken by the thermal camera should be able to transmit to the web app	1.1 As a teacher/researcher/farmer, I want to use the camera to capture pictures of different stages of crops, so that the students can learn the lifecycle of crops.	21
	The web app should have a user interface for camera data visualization	2.4 As a farmer/researcher, I want to use the infrared imaging function to detect the temperature of the environment including the plant and the soil so that I can improve the quality and nutritional value of crops through precision agriculture techniques	17
Camera Monitoring	The web app should have an interface for plant monitor and can control the movement accordingly	2.3 As a sustainable farm owner, I want to use thermal cameras to help me accurately control the irrigation water quantity so that I can accurately control irrigation volume, thereby improving crop growth efficiency.	20
	The web app should be able to control the farmbot by sending instructions remotely	1.4 As a farmer/researcher/teacher, I want to farm remotely so that I can reduce costs and get more time.	20
	An alarm should be able to send from farmbot to the web app if an emergency show up	2.1 As a teacher/researcher, I want to receive alarms when crops are dry, so that I can remind students that they forgot to water the plants.	8
Data Management	The data collected from farmbot should be able to transmit to the database	3.1 As a teacher/researcher, I want to collect thermal data through farmbot so that it could be used for further study.	12
	The users should have an interface to search for the data stored	3.2 As an agricultural researcher, I want to manage my database related to the plants so that I have a clean and reliable dataset for my study.	10
	The web app should have an interface for data processing or data visualization	2.2 As a sustainable farm owner, I want to use the thermal camera to detect whether the crops are infected with pests and diseases so that I can timely detect and treatment of pests and diseases.	20
	The web app should have a module to display a detailed introduction document of the technology for the robot and an educational video	1.2 As a teacher, I want to show the students the automated process of planting, watering, and harvesting crops, so that they can understand how technology works in agricultural production.	5

Farmbot Project Home

Parent Page: SCOPE AND REQUIREMENTS

NONFUNCTIONAL REQUIREMENTS

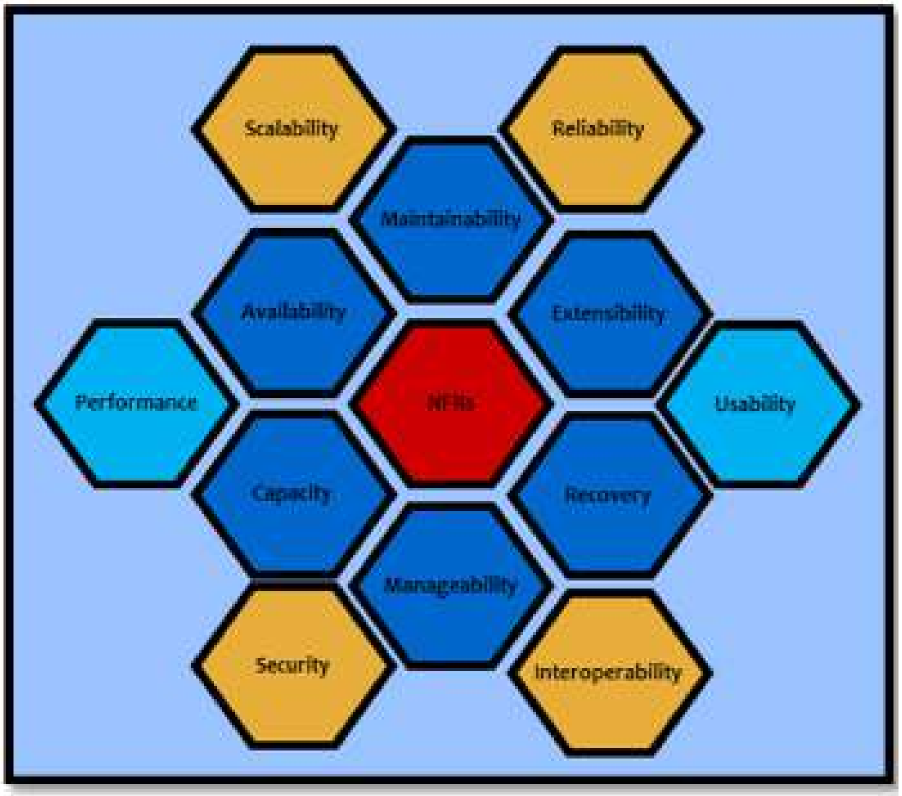


Figure 1: Key non-functional requirements (Paradkar. 2017)

For non-functional requirements, we referenced the book *Mastering Non-Functional Requirements:*

Performance

- The data obtained by the thermal camera should be clear, accurate, and credible.
- Farmbot's remote access should be accurate.

Scalability

- The thermal camera should be able to monitor data from multiple plants at the same time.

Availability

- The farmbot web app should be able to be used by users at any time.

Security

- Users' private data should be protected by encryption.
- Regulate the use mechanism of the farmbot to avoid unnecessary loss of personnel and property.

Maintainability

- Thermal imaging cameras must be regularly tuned and maintained to ensure their stability and accuracy.

Manageability

- The web app interface should be able to manage the farmbot, including actions control, camera monitoring, data analysis, and so on.

- The database is supposed to be managed by the users for data management.
- Users should have access to download any data from the database.

Reliability

- The operating system and interface of the web app should be reliable for farmbot control
- The database should be reliable for data storage and operations

Extensibility

- The web app structure should be flexible for extensions in the future
- The database should be scalable for future usage.

Recovery

- The database should have the ability to recover after crashes
- The web app should be able to recover from network interruption

Interoperability

- The thermal camera should be able to seamlessly integrate with the Farmbot web app, ensuring a seamless and efficient experience for users.

Usability

- Instead of a complex command terminal, the web app should feature a user-friendly and intuitive interface to enhance users' clarity and ease of use.

References:

- Paradkar, S. (2017). Mastering Non-Functional Requirements [electronic resource] / Paradkar, Sameer. Packt Publishing.

Farmbot Project Home

Parent Page: SCOPE AND REQUIREMENTS

SCOPE

Requirements in scope

Function	Description
Camera management	• Users can access and could configure the camera • The system could set the camera to real picture mode and take a picture when the user clicks the "take a picture" button
Data management	• Data are visible on the web app for users • Users can access to select, create, update, and delete any data that has been stored. interconnected.
Plant drying detection	• Users will hear an alarm if the plant drying data is lower than the threshold.
Receiving email	• If dry plants or pests and diseases are detected, an alert is immediately sent to the user in charge by email
pests and diseases detection	• If any pests and diseases are detected, an alert is immediately sent to the user by email
irrigation program	• Users can set up irrigation time • Users can set up irrigation volume

Requirements not in scope

Function	Description
Multiple users operate a component at the same time	• Concurrency issues are addressed in future development
Uploading customized avatar	• we do not develop customized uploaded avatars.
Fix the previous team's bug	• The current step sometimes cannot work as expected in the create task which is designed by the previous team. We have not fixed this bug.

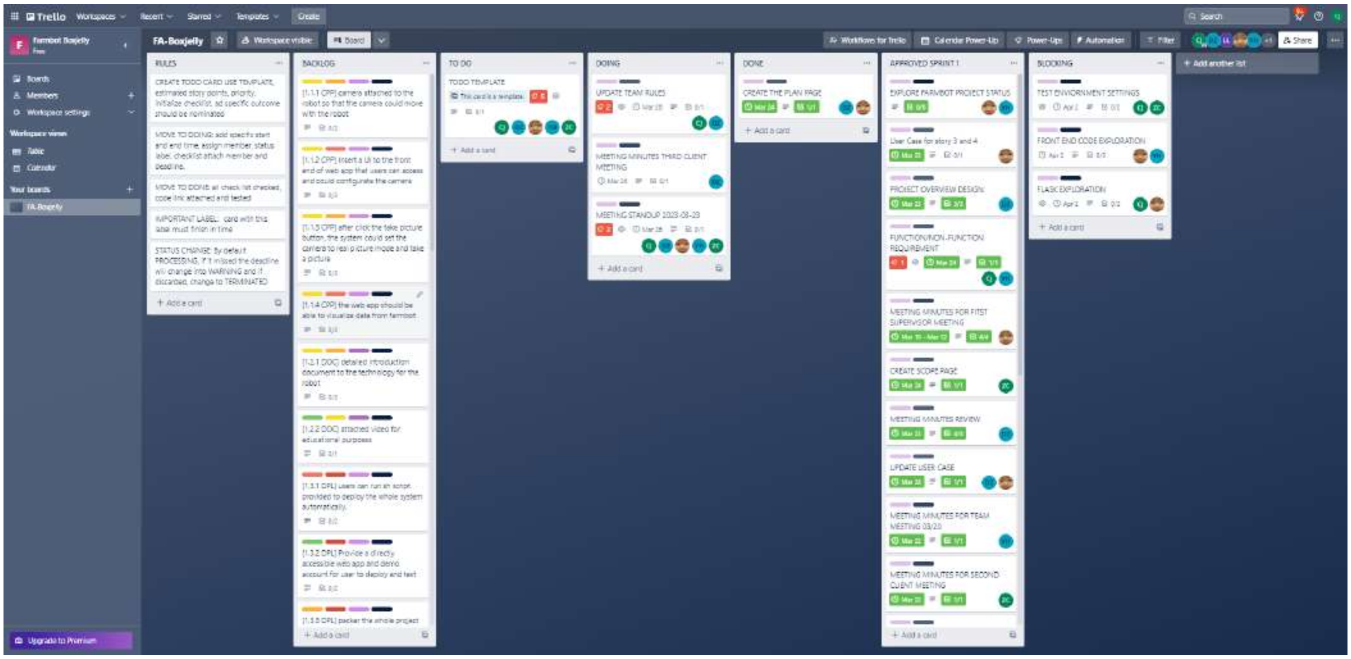
Farmbot Project Home

Parent Page: SCOPE AND REQUIREMENTS

PROJECT PLANNING

Introduction

- Our goal is to finalize the project development during Sprint 2 and Sprint 3. To achieve that, we have reorganized the user stories into user cases, meanwhile taking into account their priorities of the user cases. Our team has created a well-defined development plan, which can be viewed on Trello: FA-Boxjelly development plan.



Sprints

- Sprint 2:

Time Period	Plans	Reasons
week 5	1. Familiar with the source code from web-app and find a way to test the web-app without compose 2. Control the camera with python script on PC 3. Design and pruchase tools to attach camera to the robot 4. Find a way to register farmbot to self-host web-app	1. Divide the entire project into two parts, software and hardware, to facilitate division of labor and initiation. 2. Testing the deploy of web application would take a lot of time for compilation and different environment settings may lead to different issues. 3. As our project is related to thermal imaging cameras, installing and controlling the cameras is something we must do.
week 6	1. Insert code into web-app to create our own UI on the web-app 2. Design and set up database for the thermal data and connect the database to web-app 3. Find a way to attach the camera to the farmbot 4. Find a way to connect the camera to the raspberry pi and control it with python code	1. We need a front-end page to control and communicate with our thermal imaging camera. 2. When using the camera, we also want to save the data for later viewing or processing. 3. Everything relies on an installed, functional, and controllable thermal imaging camera.
week 7	1. By clicking the UI on the frontend, run python code implemented inside the webapp to control camera 2. Set up python code to fully control the camera (set up configuration, caputring and real-time data streaming) 3. Try save thermal data to database	1. After the front-end page is completed, we need it to send requests to the back-end and display the data provided by the back-end on the front-end page. 2. The data is stored in a database so that the back-end can provide it to the front-end.
week 8		

	1. Test deployment and transfer data into shell and pack in docker. 2. Write and test the system with designed test cases 3. Documentation review 4. Release the product version 1.0.	1. After the back-end is completed, we need to test the program to ensure smooth operation. 2. After the program testing is completed, we need to test the deployment to ensure that it can be deployed on different machines. 3. Reviewing our documentation, and once completed, we can release the first version of our program.

- Sprint 3:

Time Period	Plans	Reasons
week 9	1. Bug fix for previous release 2. Preprocess thermal image data 3. Familiar with computer vision algorithms	1. Fix the bugs that occurred in the previous version. 2. To expand the use of other technologies, we may need to start from the aspect of computer vision.
week 10	1. Set up controls for auto-picture taking 2. Set up auto picture process and save analyzed data to database 3. Distinguish each plants on the picture with computer vision algorithms	1. We want to free up productivity through automated processing, and scheduled automatic acquisition of photos and plant status is essential data for subsequent processing. 2. By using computer vision to recognize plants, we can independently monitor and record different plants.
week 11	1. Set up dry soil definition according to thermal data and set up alarms 2. Set up plant stress definition 3. Set up movement mode to auto-water dried plants	After obtaining information about different plants, we need to continuously monitor and set thresholds to ensure that the plants can continue to grow.
week 12	1. Test deployment and transfer data into shell and pack in docker. 2. Write and test the system with designed test cases 3. Documentations review 4. Release the product version 2.0.	After adding new features, continue testing the overall code, and then test the deployment to ensure that the new features are fully and correctly integrated into our system. Then, review the documents and publish our new version.

Infrastructure to deploy the project

Deployment plan process

Scope	4 days	Reason
Establishing project's scope	1 day	A quick overview of the project's overall goals and functions should not take too much time.
Identifying main resources	3 days	Establishing the main direction and conducting preliminary code learning would require a moderate amount of time.

Scope	4 days	Reasons
Camera check and review	1 day	Installing and checking the camera should not take too much time.
Web-app check and review	2 days	Reviewing and inspecting the front-end and back-end code would require a reasonable amount of time.

Deployment check and review	1 day	Checking the project deployment should not take too much time.

Scope	5 days	Reason
Camera evaluation and improvement	2 days	Evaluating the camera and checking for areas of improvement would require a reasonable amount of time.
Web-app evaluation and improvement	2 days	Evaluating the web application and checking for areas of improvement would require a reasonable amount of time.
Deployment evaluation and improvement	1 day	Evaluating the project deployment and checking for areas of improvement should not take too much time.

Farmbot Project Home
Parent Page: Requirement

CODE REVIEWS

- ChatGPT Code Review
- PEER TO PEER CODE REVIEW

ChatGPT Code Review

ChatGPT Prompt:

```
import time
import subprocess
from subprocess import Popen, PIPE
camera_IP = "169.254.15.202"

TIME_OUT = 15
TIME_WAIT = 5

# uses ping to check if camera ethernet is connected
def ping_camera():
    cmdPing = 'ping -c 1 {} | grep "bytes from"'.format(camera_IP)
    try:
        res = subprocess.check_output(cmdPing, shell=True)
        if res:
            return True
        else:
            return False
    except subprocess.CalledProcessError:
        return False

# download a jpg file from the camera ethernet website
# pic_id is the id of the picture
# pic name is the name of the picture, you have to know id and the name of the picture to download that.
def jpg_download(pic_id,pic_name, target_path):
    sftp_cmd = "sshpass -p 3vlig sftp -oStrictHostKeyChecking=no fliruser@" + camera_IP + ":/FLIR/images/"+ pic_name +".{fmt} {name}.{fmtN}"
    try:
        full_name = "{}cap{:04d}_{}".format(target_path, pic_id, pic_name)
        cmd = subprocess.call(bytes(sftp_cmd.format(fmt="jpg", name=full_name, fmtN="jpg"),encoding='utf-8'), shell=True)
        print("download successful")
    except Exception as e:
        print("[SFTP] Download Error: {}".format(e))
        result = False

# this is used for report when time out
# popen is the sesstion we currently used
# error_message is the message to print when time out.
def time_out_error(popen, error_message):
    timer = time.time()
    while not popen.stdout.read(1):
        time.sleep(TIME_WAIT)
        if time.time > timer + TIME_OUT:
            popen.kill()
            print(error_message)
        return True
    return False

# this function sends a command to the cameara through popen.
# cmd string is the command to send
# success_message is the information to print when request successful
# print lines is the number of lines to print when feedback received
def send_command(cmd_str,success_message,print_lines):
    ssh_cmd = "sshpass -p 3vlig ssh -oStrictHostKeyChecking=no -t -t fliruser@{}".format(cameraIP)
    # open interactive session to camera
    try:
        popen = Popen(ssh_cmd, shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE)
        # check for a response
        if not popen.stdout.read(1):
            popen.kill()
            print("popen error: no command input")
            return False,[]
        popen.stdout.flush()
        print("popen session set up")
        # send command to camera
        popen.stdin.write(bytes("{}\n".format(cmd_str), encoding='utf-8'))
        popen.stdin.flush()
        print("popen command sent")
        if time_out_error(popen,"ERROR, time out: no response from device"):
            return False,[]

        # read two lines
        # read lines
        result = []
        for i in range(0,print_lines):
            result = result + popen.stdout.readline().split()
```

```
        print("popen result received")
        #end popen session
        popen.stdin.write(bytes("exit\n", encoding='utf-8'))
        popen.stdin.flush()
        if time_out_error(popen,"ERROR, time out: when exit"):
            return False,[]
        print(success_message)
        return True, result

    except Exception as e:
        print("[SSH ERROR] {}".format(e))
    return False, []

# this functin used popen to send a command to the cameara to take a picture
# the file_name is the name of the picture to be downloaded,
# pic id is hard-coded with 2, we are not gonna use it.
def camera_ssh_trigger_single(file_name):
    ssh_cmd_capjpg = "/FLIR/usr/bin/store -v /FLIR/images/" + file_name + ".jpg"
    success_message = "capture successful"
    flag, result = send_command(ssh_cmd_capjpg,success_message,2)
    if flag:
        print(result)

# gets camera version info over SSH
def camera_ssh_version():
    ssh_cmd_version = "/FLIR/usr/bin/version"
    success_message = "ssh version got"
    flag, result = send_command(ssh_cmd_version,success_message,2)
    if flag:
        print(result)

# the main function first ping the camera to check the status
# and then the capture and downloa the picture to the hard-coded path.
if __name__ == '__main__':
    res = ping_camera()
    if res:
        print("[PWR] [FLIR] Camera detected, Wait for camera OS boot")
        camera_ssh_version()
        camera_ssh_trigger_single("test.jpg")
        jpg_download(11,"01","/home/dikaiz/Downloads/")
```

Please evaluate the code above:

Use the following checklist to guide your analysis:

1. Documentation Defects:
   a. Naming: Assess the quality of software element names.
   b. Comment: Analyze the quality and accuracy of code comments.
2. Visual Representation Defects:
   a. Bracket Usage: Identify any issues with incorrect or missing brackets.
   b. Indentation: Check for incorrect indentation that affects readability.
   c. Long Line: Point out any long code statements that hinder readability.
3. Structure Defects:
   a. Dead Code: Find any code statements that serve no meaningful purpose.
   b. Duplication: Identify duplicate code statements that can be refactored.
4. New Functionality:
   a. Use Standard Method: Determine if a standardized approach should be used for single-purpose code statements.
5. Resource Defects:
   a. Variable Initialization: Identify variables that are uninitialized or incorrectly initialized.
   b. Memory Management: Evaluate the program's memory usage and management.
6. Check Defects:
   a. Check User Input: Analyze the validity of user input and its handling.
7. Interface Defects:
   a. Parameter: Detect incorrect or missing parameters when calling functions or libraries.
8. Logic Defects:
   a. Compute: Identify incorrect logic during system execution.
   b. Performance: Evaluate the efficiency of the algorithm used.

Provide your feedback in a numbered list for each category. At the end of your answer, summarize the recommended changes to improve the quality of the code provided.

Feedback From ChatGPT

Based on the checklist provided, here is an evaluation of the code:

Documentation Defects:
a. Naming: The naming of variables, functions, and parameters is generally clear and descriptive, with a few exceptions:

camera_IP should be cameraIP to maintain consistency with other variable names.
TIME_OUT and TIME_WAIT should follow the snake_case naming convention, such as timeout and time_wait.
b. Comment: The code includes some comments that provide explanations for certain sections of code. However, there are areas where comments could be improved or added to enhance code comprehension. For example:

The purpose of the jpg_download function could be clarified in a comment.
The purpose of the time_out_error function and how it is used could be explained in a comment.
More comments could be added to describe the purpose of each function and its parameters.
Visual Representation Defects:
a. Bracket Usage: The code appears to have correct bracket usage, with opening and closing brackets matched appropriately.

b. Indentation: The code indentation is generally consistent and readable. However, there are a few areas where the indentation could be improved for better readability.

c. Long Line: The code contains a few long lines that may hinder readability. Breaking them into multiple lines or applying appropriate line wrapping would improve code clarity.

Structure Defects:
a. Dead Code: There doesn't appear to be any dead code in the provided code snippet.

b. Duplication: The code snippet does not contain any obvious duplicate code.

New Functionality:
a. Use Standard Method: There are no specific instances where a standardized approach should be used for single-purpose code statements.

Resource Defects:
a. Variable Initialization: All the variables used in the code have proper initialization.

b. Memory Management: The code does not involve complex memory management or manipulation. However, it uses subprocesses, and it's important to ensure proper handling and cleanup of resources.

Check Defects:
a. Check User Input: The code does not involve direct user input. However, it relies on camera_IP and other hard-coded values, which should be validated or configurable externally.

Interface Defects:
a. Parameter: The parameters used in function definitions and function calls appear to be correct and consistent.

Logic Defects:
a. Compute: No obvious logical issues were identified during a static analysis of the code.

b. Performance: The performance aspects of the code cannot be accurately assessed without additional context or information about the expected workload.

Recommended changes to improve the quality of the code:

Review and update variable names for consistency and adherence to naming conventions.
Add more comments to clarify the purpose and functionality of functions, parameters, and sections of code.
Improve code indentation for better readability.
Break long lines into multiple lines or apply appropriate line wrapping.
Consider refactoring sections of code that appear to be similar to avoid duplication.
Validate or make the camera_IP and other hard-coded values configurable externally.
Ensure proper handling and cleanup of subprocesses and resources.
Consider implementing error handling and exception catching for subprocess calls to provide more informative error messages.
Conduct thorough testing and profiling for performance evaluation and optimization if required.
Please note that this assessment is based on a static analysis of the code provided, and a more comprehensive review and testing might be necessary to uncover additional issues or improvements.

PEER TO PEER CODE REVIEW

DATE

2023-3-23

PARTICIPANTS

MEMBER GROUP	MEMBER
Team Members	☑ Dikai ZHU ☑ Chengtian JIANG ☑ Tingzheng Ren ☑ YUXI HE ☑ Zhiyu Chen

GOALS

- Review the code of web app front-end and thermal camera control script
- Find issues and fix the bugs

CODE REVIEW

Link to the original file: https://github.com/COMP90082-2023-SM1/FA-Boxjelly/blob/main/docs/COMP90082_FA_Boxjelly_CodeReview.xlsx

ORGANISING CODE REVIEW								
Name and local of artifact (on Github) to be reviewed:		https://github.com/COMP90082-2023-SM1/FA-Boxjelly/tree/main/src						
What to be reviewed?		web app interface/ thermal camera control script						
When's the code review meeting happening:		Read artifact: April 25th / Code review meeting: April 26th						
Reviewers:		Dikai Zhu, Chengtian Jiang, Yuxi He, Tingzheng Ren, Zhiyu Chen						
CODE REVIEW MEETING (STARTING THE MEETING)								
Team:	<FA>-<Boxjelly>							
Date:	<26/04/2023>							
Time:	<14:30:00>							
Facilitator:	Dikai Zhu							
Reviewers:	Chengtian Jiang, Yuxi He, Tingzheng Ren, Zhiyu Chen							
CODE REVIEW (DURING MEETING TIME)								
Item	Artifact (on GitHub)	Location (where the issue was found in the reviewed artifact?)	Severity	Type	Defects Category	Description	Fixed by the author?	Verified by the Moderator?
1	FA-Boxjelly/src/python_script/control_script_version_2.1.py	general	High	Improvement	Documentation Defects (Comment)	Code should have a better documentation describing each functionality and usage.	Yes	Yes
2	FA-Boxjelly/src/python_script/control_script_version_2.1.py	function send_command	Trivial	Issue	Structure Defects (Dead Code)	Improve the exit condition for the loop	Yes	Yes
3	FA-Boxjelly/src/python_script/control_script_version_2.1.py	function camera_ssh_trigger_single	Trivial	Improvement	Resource Defects (Variable Initialization)	Variable sshCamCapJPG is not the same format as other variables	Yes	Yes
4	FA-Boxjelly/src/python_script/control_script_version_2.1.py	function jpg_download	Medium	Investigate	Interface Defects (Parameter)	The function should accept parameters instead of having values hard-coded within it	Yes	Yes
5	FA-Boxjelly/src/python_script/control_script_version_2.1.py	general	Medium	Improvement	New Functionality (Use Standard Method)	Organize the code into functions or modules	Rejected	Rejected
6	FA-Boxjelly/src/python_script/control_script_version_2.1.py	function camera_ssh_trigger_single	Trivial	Improvement	Structure Defects (Duplication)	Use a function to wrap repeated code	Yes	No

7	FA-Boxjelly/src/front_end /frontend/camera/photos.tsx	general	High	Improvement	Documentation Defects (Comment)	Code should have a better documentation describing each functionality and usage.	No	No
8	FA-Boxjelly/src/front_end /frontend/camera/photos.tsx	general	Trivial	Improvement	Structure Defects (Dead Code)	Delete unused packages	Rejected	Yes
9	FA-Boxjelly/src/front_end /frontend/camera/photos.tsx	function render	High	Improvement	New Functionality (Use Standard Method)	Lark of implementation of button logic	No	Yes
10	FA-Boxjelly/src/front_end /frontend/camera/photos.tsx	general	Medium	Improvement	New Functionality (Use Standard Method)	Organize the code into functions or modules	No	No
11	FA-Boxjelly/src/front_end /frontend/camera/photos.tsx	componentDidMount	High	Issue	Resource Defects (Variable Initialization)	Parameter undefined	No	No
19								
END OF CODE REVIEW MEETING								
Number of severe/critical errors:		<1>						
Number of medium errors:		<N>						
Number of trivial errors:		<1>						
Total inspection time (hs):		<150 mins>						

Development

This section contains details of all the implemented features

- Attach Thermal Camera in Web App
- Attach the Thermal Camera to Farmbot
- Control of Thermal Camera
- Convert Customized Source Code to Nerves Image
- Customized Web App Deployment
- Farmbot Connection to Customized Web App
- Information Flow of the Picture Capture Process in farmbot
- Thermal Camera Connection to Raspberry Pi Server

Farmbot Project Home

Attach Thermal Camera in Web App

Introduction

- To implement the functionality of our thermal camera, we have decided to add a new button in the navigation bar that will direct users to our temperature camera feature.
- After successfully connecting to the FarmBot, the thermal camera page will display a default image or video.

Step 1: Simply insert a button in the navigation bar.

- Firstly, we should insert a button into the navigation bar as the entry point for our page.
  - **Insert button**

```
...
export enum Panel {
  ...
  Sensors = "Sensors",
  Photos = "Photos",
  Camera = "Camera",
  Farmware = "Farmware",
  Tools = "Tools",
  Messages = "Messages",
  ...
}
...
export const PANEL_SLUG: Record<Panel, string> = {
  ...
  [Panel.Logs]: "logs",
  [Panel.Help]: "help",
  [Panel.Camera]: "camera",
  [Panel.Settings]: "settings",
  [Panel.Shop]: "shop",
}
...
export const getLinks = (): Panel[] => [
  ...
  Panel.Photos,
  ...(showFarmware() ? [Panel.Farmware] : []),
  Panel.Tools,
  Panel.Messages,
  Panel.Help,
  Panel.Camera,
  Panel.Settings,
]
...
```

Please note that the above code is not in the same file and consists of multiple code blocks from different files, mainly to demonstrate the link of the button code.

Step 2: Implementing the camera page.

- After the entry button is clicked, we want to display our camera page.
  - **Main camera page**

```
import React from "react";
...

export class RawDesignerPhotos
  extends React.Component<DesignerPhotosProps> {

  componentDidMount = () => this.props.dispatch(maybeOpenPanel("photos"));

  toggle = (key: keyof PhotosPanelState) => () =>
```

```
    this.props.dispatch({ type: Actions.TOGGLE_PHOTOS_PANEL_OPTION, payload: key });

  get imageShowFlags(): ImageShowFlags {
    return getImageShownStatusFlags({
      getConfigValue: this.props.getConfigValue,
      env: this.props.env,
      image: this.props.currentImage,
      designer: this.props.designer,
      size: this.props.currentImageSize,
    });
  }

  render() {
    const wDEnvGet = (key: WDENVKey) => envGet(key, this.props.wDEnv);
    const { syncStatus, botToMqttStatus, photosPanelState } = this.props;
    const botOnline = isBotOnline(syncStatus, botToMqttStatus);
    const common = {
      syncStatus,
      botToMqttStatus,
      timeSettings: this.props.timeSettings,
      dispatch: this.props.dispatch,
      images: this.props.images,
      env: this.props.env,
      currentImage: this.props.currentImage,
    };
    const imageCommon = {
      flags: this.imageShowFlags,
      designer: this.props.designer,
      getConfigValue: this.props.getConfigValue,
    };
    const farmwareNames = Object.keys(this.props.farmwares);
    return <DesignerPanel panelName={"camera"} panel={Panel.Camera}>
      <DesignerNavTabs />
      <DesignerPanelContent panelName={"camera"}>
        <label>{t("Camera")}</label>
        <button>Click</button>
        <Photos {...common} {...imageCommon}
          currentBotLocation={this.props.currentBotLocation}
          movementState={this.props.movementState}
          arduinoBusy={this.props.arduinoBusy}
          currentImageSize={this.props.currentImageSize}
          imageJobs={this.props.imageJobs} />

      </DesignerPanelContent>
    </DesignerPanel>;
  }
}

export const DesignerPhotos = connect(mapStateToProps)(RawDesignerPhotos);

export interface UpdateImagingPackageProps {
  farmwareName: string;
  version: string | undefined;
  botOnline: boolean;
}

export const UpdateImagingPackage = (props: UpdateImagingPackageProps) =>
  props.version
    ? <div className={"update"}>
      <p>v{props.version}</p>
      <i className={"fa fa-refresh"}
        onClick={requestFarmwareUpdate(props.farmwareName, props.botOnline)} />
    </div>
    : <div className={"update"} />;
```

Step 3: Creating link paths.

- When we have finished creating the button and designing the main page, the final step is to connect them together.
  - **Route link**

```
...
route({
    children: true,
    $: Path.route(Path.camera()),
    getModule,
    key,
    getChild: () => import("./camera/photos"),
    childKey: "DesignerPhotos"
  })
...
```

Farmbot Project Home

Parent Page: Development

Attach the Thermal Camera to Farmbot

Introduction

- To ensure that the camera moves in sync with the robot arm, it is necessary to find a way to attach it to the arm. This can be achieved by using a mount or bracket that is designed to hold the camera securely in place while the arm is in motion. By attaching the camera to the arm in this way, you can capture footage or images of the area where the arm is working, allowing you to monitor the progress of the robot's activities and make any necessary adjustments in real-time.

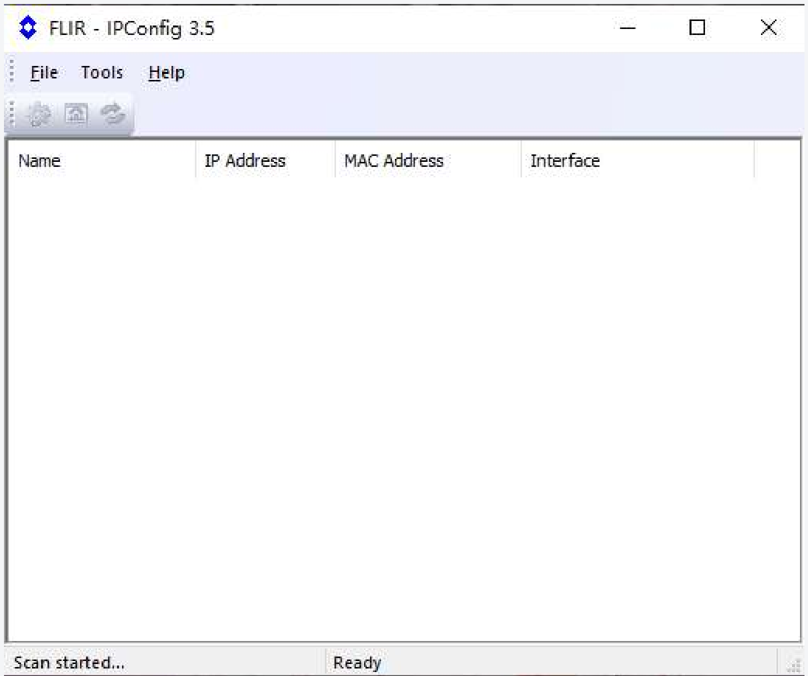
Step 1

- We make use of a mobile phone holder with a sucker to attach the camera to the robot
- 

Physical attach the camera to the arm

Step 2

- To enable the camera to connect to the robot via Ethernet, it is necessary to use a cable to connect the Ethernet ports on both devices. However, changing the camera's default IP is also required to ensure that it can communicate with the robot on the same subnet and subnet mask.
- FLIR provides an ipconfig tool that can be used to change the camera's default IP address. This tool allows you to set the camera's IP address, subnet mask, and gateway so that it can communicate with the robot on the same network. By using this tool, you can configure the camera to operate within the same network as the robot and ensure that they can communicate with each other effectively.
- ipconfig download: https://support.flir.com/SwDownload/Assets/FLIR%20IP%20Config/FLIR_IP_Config_3_0.zip



User API for ipconfig

Step 3: Ping the camera from the robot

- Since we still not find a way to send commands from the robot, this test was done with a Raspberry Pi 3, and it was successful.

Farmbot Project Home

Parent Page: Development

Control of Thermal Camera

Introduction

- Since the FLIR-Ax8 camera only supports Ethernet connections and requires configuration and capture through an Ethernet website, it is not feasible for the camera to be directly controlled by a robot. Additional software may be required for the robot to interface with the camera.
- The objective of this development is to produce a software solution that enables the FLIR-Ax8 camera to be controlled automatically, without the need for manual capture via the Ethernet website. The goal is to streamline the capture process and enable the camera to be integrated into automated systems.

Solution Version 1: (Deprecated)

- After capturing all the packets sent from the local browser to the Ethernet website, we were able to identify key packets that send commands to the FLIR-Ax8 camera. Using Java, we were able to reproduce these packets under our control and send them to the camera through Farmbot, allowing us to manage the camera automatically. This solution streamlines the capture process and enables the camera to be integrated into automated systems.
- source code path: https://github.com/COMP90082-2023-SM1/FA-Boxjelly/blob/thermal_control/src/Java_script/captureAndDownload

Solution Version 2:

- The initial version of our solution involved sending requests through a Servlet to the Ethernet website's front end, but this approach was slow and did not leverage the capabilities of the Raspberry Pi system. To address this issue, we collaborated with Bryce from the Engineer Department at the university to develop a new Python script. This script accesses the backend via SSH and sends commands more efficiently, improving the overall performance of our solution.
- The script used Popen to connect to the Ethernet website, we can directly connect using the 'sshpass' command. This allows us to force-execute PHP scripts that are hard-coded inside the website. In this way, we can create a script that captures and configures everything automatically.
- This version of code is tested on a Raspberry Pi and passed.
- source code path: https://github.com/COMP90082-2023-SM1/FA-Boxjelly/blob/thermal_control/src/python_script/thermal_control.py

Solution Version 3: (To Be Done)

- Unfortunately, there is some bad news. The Farmbot OS only accepts Lua as a valid programming language, which means we'll have to rewrite our Solution Version 2 using Lua. We'll need to ensure that our code can run on the Lua sandbox included in the Farmbot OS.
- source code path: TO BE DONE

Farmbot Project Home

Parent Page: Development

Convert Customized Source Code to Nerves Image

Introduction

- Farmbot is a software system that controls farming machines, consisting of a Raspberry Pi and an Arduino board called Farmduino. The system runs on Farmbot OS, which is made up of three different parts (Linux core, compiled Elixir scripts, Erlang OTP) and requires burning a Nerves image file onto an SD card for installation. However, it's worth noting that Farmbot OS doesn't support HDMI or SSH connections, making it difficult to modify the running OS directly. Additionally, the nerves image file only contains compiled scripts and configuration files, making it impossible to directly insert code into the image file to customize the system. To customize the system and integrate a script, one needs to update the source code and create a modified version of the Nerves Image File that can be run through the web app. While Farmbot is an open-source project, it doesn't provide an SDK for developers, as the primary aim of the project is to simplify farming rather than support further development.
- Note: Unfortunately, the official website provides very few tutorials and instructional resources.

Step 1: Dependency preparation

- The developer should first prepare a virtual machine server to run the project, ubuntu is recommended.

Dependency Required for Nerves Project

```
sudo apt-get update
sudo apt-get install -y ncurse-dev
sudo apt-get install -y gcc
sudo apt-get install -y openssl-dev
sudo apt-get install -y erlang-ssl
sudo apt-get install -y unixodbc unixodbc-dev
sudo apt-get install -y xsltproc
sudo apt-get install -y aop
sudo apt-get install -y libxml2-utils
sudo apt-get install -y libwxgtk3.0-webview-dev
sudo apt-get install -y libssl-dev
sudo apt-get install -y libncurses5-dev
sudo apt-get install -y autoconf
sudo apt-get install -y automake
sudo apt-get install -y openjdk-8-jdk
sudo apt-get install -y fop
sudo apt-get install -y rebar3
rebar3 deps update inets
sudo apt-get install -y libwxgtk-webview3.0-gtk3-dev
sudo apt install -y libgtk-3-dev
sudo apt-get install -y libusb-1.0-0-dev
sudo apt-get install -y libcurl4-openssl-dev
sudo apt-get install -y libarchive-dev
wget https://github.com/fhunleth/fwup/releases/download/v1.9.1/fwup_1.9.1_amd64.deb
sudo dpkg -i fwup_1.9.1_amd64.deb
```

Step 2: Install asdf package manager

- asdf is used as the manager for the Nerves Project.

Install asdf Package Manager

```
git clone https://github.com/asdf-vm/asdf.git ~/.asdf --branch v0.11.3
echo '. $HOME/.asdf/asdf.sh' >> ~/.bashrc
echo '. $HOME/.asdf/completions/asdf.bash' >> ~/.bashrc
```

Step 3: Install the specific version of Erlang and Elixir through the asdf package manager

- The Nerves Project specifies particular versions of Erlang and Elixir in its config file. This means that the versions of these languages must match the hard-coded versions specified in both the Farmbot OS '***.tool-versions***' file and '***mix.exs***' file. In addition, the corresponding RPi branch also specifies the versions of the language used, which means that a specific version of Farmbot OS will only support certain types of RPi, depending on the specified language versions.
- Farmbot OS version is farmbot os 15.4
- RPi versions are releases v.18.1
- Erlang version is erlang 24.2
- Elixir version is elixir 1.13.2-otp-24

Install Erlang and Elixir though asdf

```
asdf plugin-add erlang
asdf install erlang 24.2
asdf global erlang 24.2
asdf plugin-add elixir
asdf install elixir 1.13.2-otp-24
asdf global elixir 1.13.2-otp-24
```

Step 4: Install Nerves Framework

- Nerves Framework will convert the source code into nerves image files,
- It is important to note that the version of the Nerves Project must correspond to the version of Elixir being used

Install Nevers Framework

```
sudo apt install build-essential automake autoconf git squashfs-tools ssh-askpass pkg-config curl
libmnl-dev
mix local.hex -y
mix local.rebar -y
mix archive.install hex nerves_bootstrap
```

Step 5: Clone source code and apply Nerves Framework

- To customize the source code, a developer must clone the three RPi branches and make a release of them. Additionally, the developer must modify the source code inside the '***mix.exs'*** file.
- Note: root access is needed to start a Nerves Project

Clone source code and start Nervs Project

```
git clone https://CloneRepoToken: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@github.com/COMP90082-2023-
SM1/FA-Boxjelly --depth=1 --branch=farmbot-os
cd FA-BOXJELLY
mix archive.install github hexpm/hex branch latest
```

Step 6: Conduct the conversion

- The default script includes many tests, and it consumes lots of time. if the developer only wants a Nerves image file for RPi3, he can execute '***run_rpi3.sh***' instead.
- The exported information is used for test on a real web app we deployed.

Conduct Conversion

```
export FARMBOT_EMAIL=dikaiz@student.unimelb.edu.au
export FARMBOT_PASSWORD=12345678
export FARMBOT_SERVER=http://20.213.164.146:3000/
chmod +x run_all.sh
./run_all.sh
```

- When the script above finished executing, a new folder named _build would appear and the updated version of the system is inside.

Step 7: Install your OS to SD card:

- Farmbot support nerves image file and fw files, any image file reader can burn the file to SD card.
- We used balenaEtcher as an example:
- download link of balenaEtcher: https://www.balena.io/etcher

UI page of balenaEther

Farmbot Project Home
Parent Page: Development

Customized Web App Deployment

Introduction

- Since the official web app cannot be modified, the only option for adding a new page to thermal camera management is to create a self-hosted web app.
- Our steps represent a modified version of the instructions provided on the official GitHub page for self-hosting: https://github.com/FarmBot/Farmbot-Web-App/blob/staging/README.md

Step 1: Prepare the environment

- To prepare for self-hosting, you will need to set up a server and enable the following ports: 8883, 8080, 80, and 443.
- Install docker inside your server.
  - **install docker**

```
# Remove old (possibly broke) docker versions
sudo apt remove docker-engine
sudo apt remove docker docker.io containerd runc

# Install docker
sudo apt update
sudo apt install ca-certificates curl gnupg lsb-release -y
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings
/docker-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.
gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.
list.d/docker.list > /dev/null
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin -y
sudo docker run hello-world # Should run!
# Install docker-compose
sudo mkdir -p /usr/local/lib/docker/cli-plugins
sudo curl -SL "https://github.com/docker/compose/releases/download/v2.15.1/docker-compose-$(uname -s)
-$(uname -m)" -o /usr/local/lib/docker/cli-plugins/docker-compose
sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose
sudo docker compose version # test installation
```

Step 2: clone the web app source code

- the code should be cloned from a self-hold GitHub repository, not an official one.
  - **clone github**

```
git clone https://CloneRepoToken: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@github.com/COMP90082-2023-
SM1/FA-Boxjelly --depth=1 --branch=web_app_front
cd FA-Boxjelly
```

Step 3: Configurate Web App

- we prepared a tom.env file for all the self-hold configurations, client could use this file or change some information inside the file.
  - **configurate web app**

```
cp tom.env .env #change our prepared file into the default configurate file.
# nano .env # client can change code inside if he wants.
```

Step 4: Compose and Initialize the Web App

- **compose and execute web app**

```
# Install the correct version of bundler for the project
sudo docker compose run web gem install bundler
# Install application specific Ruby dependencies
sudo docker compose run web bundle install
# Install application specific Javascript deps
sudo docker compose run web npm install
# Create a database in PostgreSQL
sudo docker compose run web bundle exec rails db:create db:migrate
# Generate a set of *.pem files for data encryption
sudo docker compose run web rake keys:generate #  SKIP THIS STEP IF UPGRADING!
# Build the UI assets via ParcelJS
sudo docker compose run web rake assets:precompile
# Run the server! (^^)
# NOTE: DONT TRY TO LOGIN until you see a message similar to this:
#   "  Built in 44.92s"
# THIS MAY TAKE A VERY LONG TIME ON SLOW MACHINES (~3 minutes on DigitalOcean)
# You will just get an empty screen otherwise.
# This only happens during initialization
# configurator -d is added to official code to backup the web app
sudo docker compose up -d

# Create the database for the app to use:
sudo docker compose run -e RAILS_ENV=test web bundle exec rails db:setup
```

Farmbot Project Home

Parent Page: Development

Farmbot Connection to Customized Web App

Introduction

- To personalize the FarmBot interface, it's necessary to link it to your own web application because the official web app is not modifiable.
- There is a detailed tutorial on the official website of the Farmbot: http://software.farm.bot/v15/app/intro

Step 1:

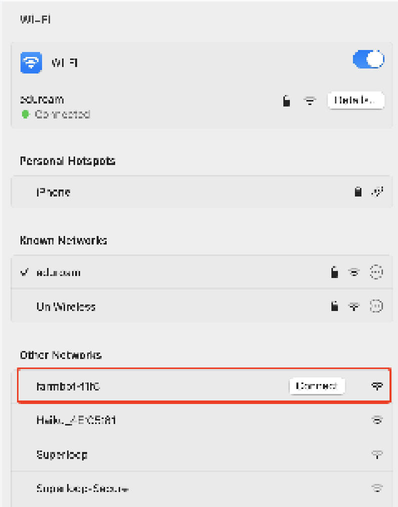
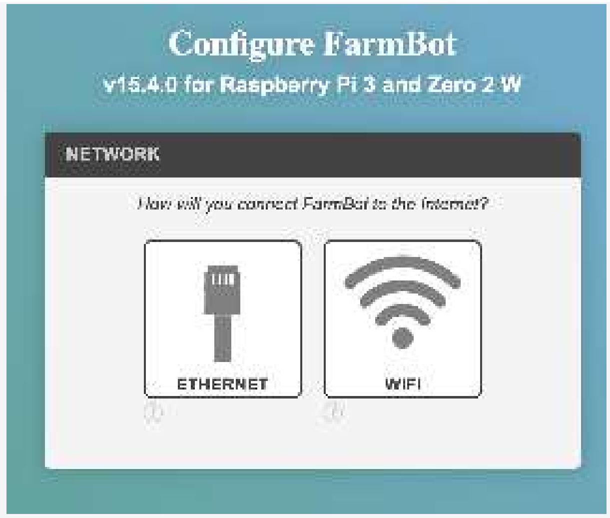
- Initially, the developer must have a self-hosted Farmbot web application (click here for how to deploy a Farmbot web app), and ensure that the correct version of Farmbot OS is produced and installed onto the SD card. (click here for how to produce customed Farmbot OS)
- Note that you have to open all the ports in the following picture to make the robot work properly

Port
22
443
3000
80
4000
8080
8883
3002
5672
1883

All ports needed

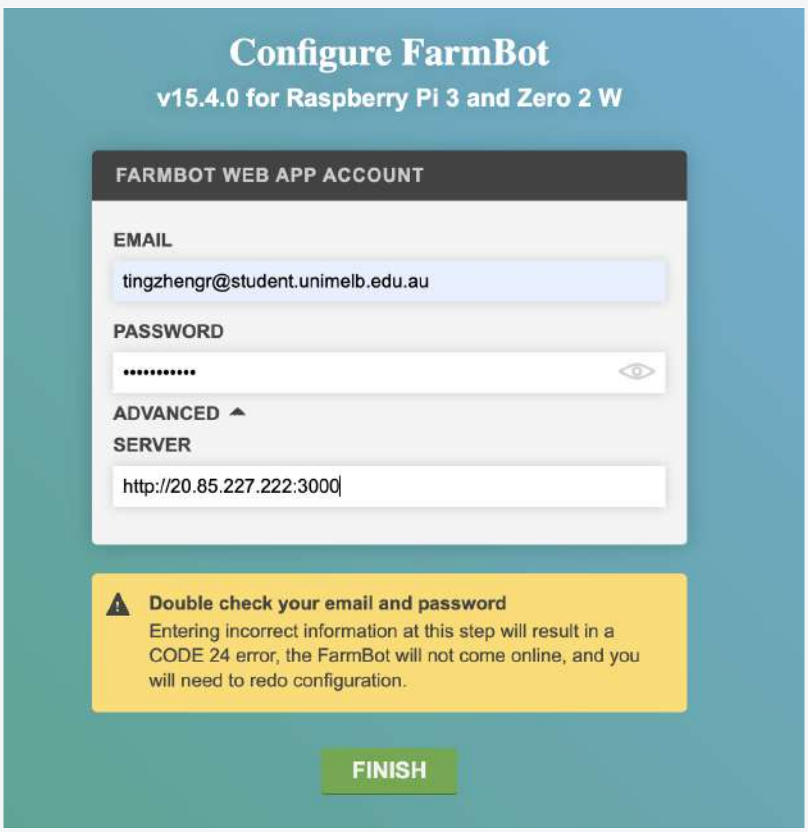
Step 2:

- Connect to the WiFi produced by the robot (ensuring that ports 80, 443, 8883, and 3000 of the server are accessible), and the robot will direct you to the connection configuration page.

	
WIFI produced by the robot	Farmbot Connect Configuration

Step 3:

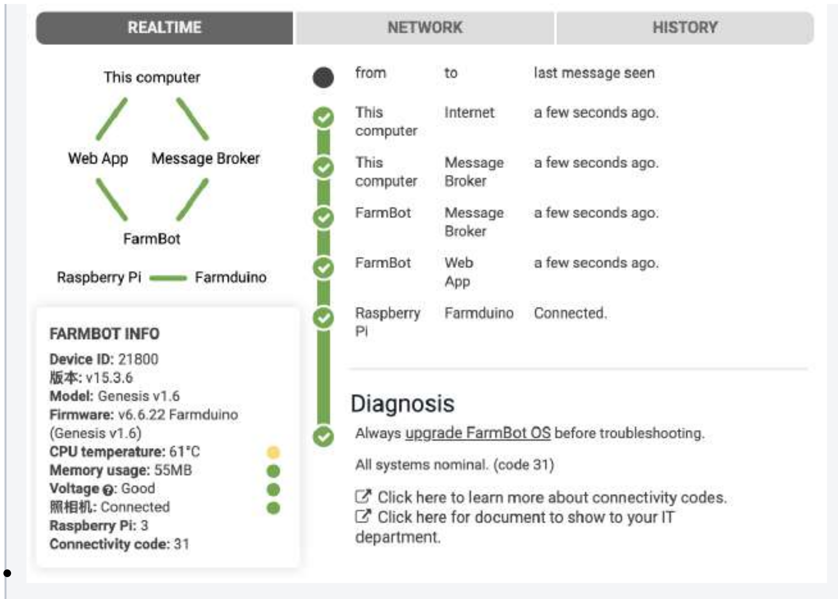
- Next, you'll need to enter your registered account information. The advanced server refers to the public address of your deployed web app.



Type in your account information and server address

Step 4:

- After the robot establishes the connection, you can view the status of the robot on the web app.

- 

Connected Status Page for the Robot

[Farmbot Project Home]

[Parent Page: Development]

Information Flow of the Picture Capture Process in farmbot

Introduction

- This page explains how a capture command is created from the front end of the web app to how it executed inside the farmbot.

Step 1: execute function in farmbot class

1. When a user clicks capture button on the front end, the system would call a function named take-photo from a class named farmbot.
2. Class farmbot is defined inside the farmbot-js repository and this is a public code package. This is farmbot-js package location.

```
/** Snap a photo and send to the API for post processing. */
takePhoto =
  (args = {}) => this.send(rpcRequest([{ kind: "take_photo", args }]))
```

take-photo function inside farmbot class in farmbot-js/corpus.ts (written in typescript)

Step 2: MQTT send take_photo command to farmbot OS

1. The take-photo function send a pre-defined RPC request to the MQTT server.
2. the request is already registered in MQTT as take_photo and would be send to farmbot OS as another rpc command

```
def _do_cs(self, kind, args, body=[]):
    """
    This is a private helper that wraps CeleryScript in
    an `rpc` node and sends it to the device over MQTT.
    """
    return self._connection.send_rpc({
        "kind": kind,
        "args": args,
        "body": body
    })
```

do_cs function in MQTT wrapper (written in python)

```
def take_photo(self)
    """
    Snap a photo and
    """
    return self._do_
```

take_photo

Step 3: farmbot OS preprocess take_photo command

1. MQTT server received the command, and then create a AST node named take_photo and add it to the CeleryScript AST
2. node inside CeleryScript AST will wait to join Elixir AST, where tasks actually done with specially designed firmware commands.

```
@doc """
iex> (new() |> rpc_request("x") |> take_photo()).body
[%FarmbotOS.Celery.AST{
  body: [],
  comment: nil,
  meta: nil,
  args: %{},
  kind: :take_photo
}]
"""
def take_photo(%AST{} = ast) do
  ast |> add_body_node(new(:take_photo))
end
```

definition of take_photo AST node in
farmbot_os/lib/celery/ast/factory.ex
(written in elixir)

Step 4: convert CeleryScript AST node to Elixir AST node

1. convert CeleryScript AST node to Elixir AST node

```
def take_photo(%{body: params}, cs_scope) do
  execute_script(%{args: %{label: "take-photo"}, body: params}, cs_scope)
end
```

Elixir tree node constructor inside
farmbot-os/lib/celery/compilers/farmware_compiler.ex (written in elixir)

```
@firmware_cmds %{
  "noop" => "noop.py",
  "camera-calibration" =>
  "plant-detection" => "qu
  "historical-camera-calib
  "historical-plant-detect
    "quickscripts/download
  "take-photo" => "take-ph
  "Measure Soil Height" =>
}
# Default configs from the
```

Elixir tree node definitionr

Step 5: execute embedded cature scripts

1. As we notice from the previous definition of Elixir tree nodes, it actually triggers the execution of a piece of python code, what we can do is change the script inside the take-photo.py to do extra capture.

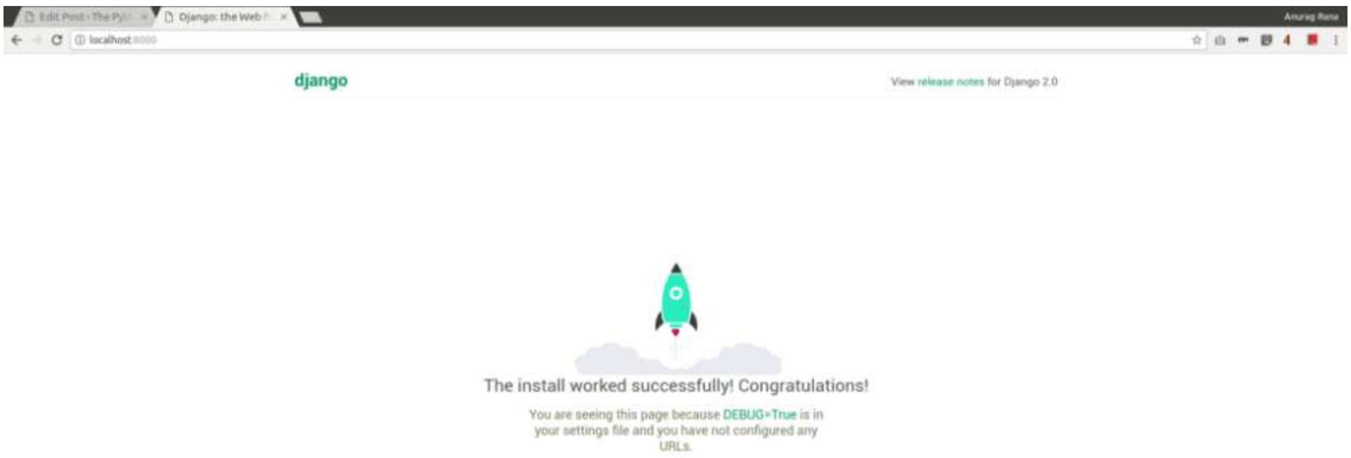
Thermal Camera Connection to Raspberry Pi Server

Introduction

- In our development project, we aim to establish a connection between a Raspberry Pi server, built with a Django structure, and a thermal camera. The Raspberry Pi server acts as the central control unit, receiving commands from a web app within the same Wi-Fi network. It then sends instructions to the thermal camera to capture an image. The captured image is stored on a local server connected to the Raspberry Pi. The server retrieves the captured image and sends it back to the web app for display and further processing.

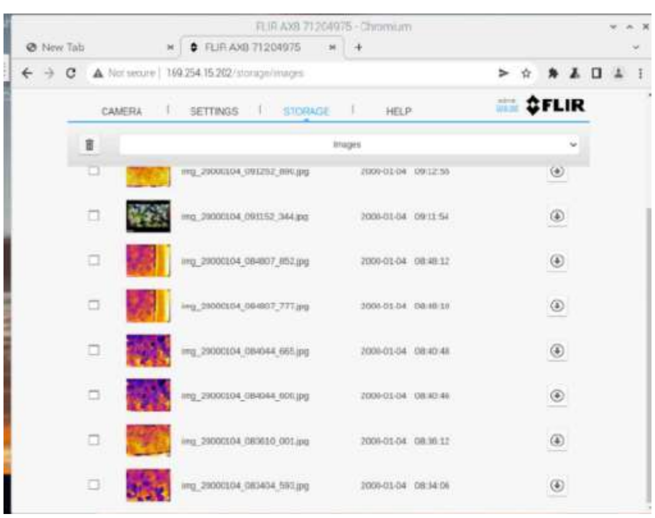
Step 1: Set up the Raspberry Pi Server:

- Install the Raspberry Pi OS
- Configure the network settings to connect to the Wi-Fi network.
- Install Django and set up the project structure.
- Create the required Django server



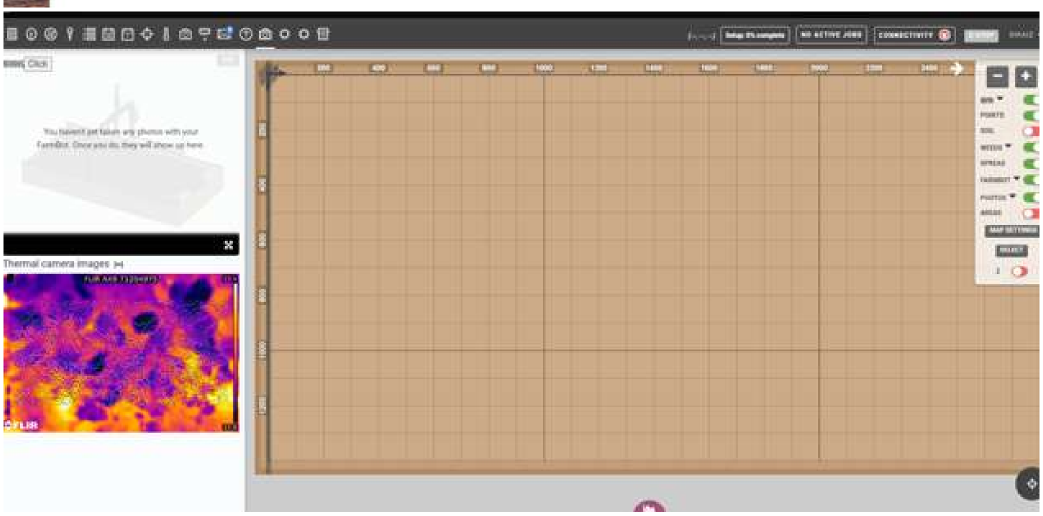
Step 2:Prepare the Thermal Camera and Local Server:

- Connect the thermal camera to the Raspberry Pi using USB.
- Install required drivers and libraries for the thermal camera to function properly with the Raspberry Pi.
- Set up a local server on the Raspberry Pi to store the captured images from the thermal camera.



Step 3: Develop Communication Logic:

- Establish communication channels between the Raspberry Pi server and the thermal camera.
- We use the https address for the server to receive commands from the web app via the Wi-Fi network. For example, http://server/take_photo will send a take photo command and http://server/photo_1 will require a photo from the server



Farmbot Project Home

Parent Page: Development

Project Timeline

This section contains the timeline of the project sprints

- Sprint 1 Planning
- Sprint 1 Feedback and Review
- Sprint 1 Retro
- Sprint 2 Planning
- Sprint 2 Feedback and Review
- Sprint 2 Retro
- Sprint 3 Feedback and Review
- Sprint 3 Planning
- Sprint 3 Retro
- Sprint 4 Planning

Sprint 1 Planning

Team

Name	Role
Yuxi He	Product Owner
Dikai Zhu	Scrum Master
Tingzheng Ren	Testing leader
Zhiyu Chen	Software Developer
Chengtian Jiang	Software Developer

Capacity planning

	Current sprint
Total days	18 days
Total hours fully working on the projects	48 hours/ppl 240 hours for whole team
Team capacity	0%
Total story points capacity	153
Total story points planned to complete	0

Sprint Planning

Goal	Plan
• Complete project design • Complete physical	• Complete design and planning for each sprints

Sprint 1 Feedback and Review

Feedback from Nir:

- Happy with the progress and understand there are so many challenges.
- Have a short discussion with professors from the Agriculture department to learn the user stories.
- The integration failed in this meeting because the SD card does not work for the farmbot and we failed to build a connection
- Discussed how to power the camera with Arduino board, the voltage the camera needs is 1030 v
- Discussed how to control the camera with Python code, how to get the IP of the camera

Feedback from grader

Review from Feedbacks	To-Dos
GitHub tag does not contain the Confluence space exported for assessment. It is very difficult to follow the structure of Confluence space, hence difficult to understand what the product is, how it links to client goals and the motivational model.	A major rework on how our Confluence space is presented to anyone is in order.
Personas are not very clear about the users in interests, needs, goals, and other requirements.	Analysis of Personas including potential users involved with the product and have a range of interests, frustrations, and goals that are helpful in driving the analysis of the product
There are two versions of user stories—which one should be used for assessment? - Without knowledge of the product (FarmBot) it is difficult to understand how the user stories are related to its functionality.	Create detailed FarmBot user stories and relate user stories to their functions. Conduct a meeting to select the most suitable user stories version for assessment. Implement and gather feedback to ensure improved comprehension and relevance.
There is a very short description of the product, which then flows to technique details (what does this mean?) and a very short Do-Be-Feel table.	Expand the product description for enhanced understanding. Break down technical details into easily comprehensible segments. Enrich the Do-Be-Feel table with more details, correlating it with the product's features and benefits. Implement changes and gather user feedback for continuous improvement.
The structure of the Confluence space needs a serious review.	Review the text to remove spelling mistakes, typos, capitalization, and grammar
The page tree starts with Trash Bin, followed by Meeting Minutes, then Project Product (what does this mean?), etc. This is not a logical order for someone to follow and get an adequate idea of what your product is and what the project is.	Revise the page tree to start with an introduction to the Project Product. This should be followed by an overview of the project, Meeting Minutes, and other related information. The Trash Bin should be moved to the end. The goal is to create a logical and intuitive flow for the users to better understand the product and the project.

Sprint 1 Retro

Date

2023-3-23

Team members

- Dikai ZHU
- Zhiyu Chen
- @Chengtian Jiang
- @Yuxi He
- @Tingzheng Ren

Goals

- *Sprint 1 review*
- *Sprint 1 retrospective*
- *Sprint 2 planning*

Sprint 1 Review

Item	Discussion
Review the design	• All team members do the design review for the project
Review the sprint 1 assignment docs	• All team members take participate in the review of documents on the Confluence and make the final updates: Architecture Design, user story
Review the architecture	▪ Update diagrams in Architecture Design

Sprint 1 Retro

What we did well	What we didn't do well
Good communication and teamwork based on the agile process	We can not compete all tasks we planned
Good Confluence structure.	Trello daily update not in time
We are clear about what other team members have done in this sprint.	We cost more time learning the code language of the existing system
We got feedback from the client and we organized what need to be done in the next sprint.	
Every team member complete their tasks on time	

Sprint 2 Planning

Epic	Rank (Timeline)
Complete Physical connection	1
Deployed the custom web app	2
Complete a front page for the thermal cameral	3
Complete a script that controls the camera	4

Sprint 2 Planning

Team

Name	Role
Yuxi He	Product Owner
Dikai Zhu	Scrum Master
Tingzheng Ren	Testing leader
Zhiyu Chen	Software Developer
Chengtian Jiang	Software Developer

Capacity planning

	Current sprint
Total days	35 days
Total hours fully working on the projects	48 hours/ppl 240 hours for whole team
Team capacity	60%
Total story points capacity	153
Total story points planned to complete	59

Sprint Planning

<table>
<tr><th>Goal</th><th>Plan</th></tr>
<tr><td>
Goal 1: Familiarize with the web app's source code and find a way to test the web app without using compose.

Analyze the structure and components of the web app's source code.
Understand how the web app works and its dependencies.
Research and select suitable testing tools or frameworks for testing the web app in a local environment.
Configure and run the necessary components and services of the web app for testing.
Write test cases and perform tests to validate the functionality and performance of the web app.

Goal 2: Control the camera with a Python script on a PC.

Research and select suitable Python libraries or APIs for interacting with the camera.
Write a Python script to control the camera, such as setting configurations and capturing images.
Test the Python script on a PC to ensure the camera works correctly.
Integrate the Python script with other necessary functionalities to achieve full control of the camera.

Goal 3: Design and purchase tools to attach the camera to the robot.

Analyze the robot's structure and available mounting positions to determine suitable tools and accessories.
Design or find appropriate brackets, fixtures, or interfaces to attach the camera to the robot.
Purchase the required tools, accessories, and materials and perform the necessary installation and adjustments.

Goal 4: Find a way to register Farmbot to a self-hosted web app.

Research the registration process and requirements of the self-hosted web app.
Configure the network connection and authentication information of Farmbot to meet the requirements of the self-hosted web app.
Test whether Farmbot can successfully register with the self-hosted web app.
Document any issues encountered during the registration process and their solutions.

Goal 5: Insert code into the web app to create a custom user interface.

Analyze the code structure and interface components of the web app.
Write or modify code to insert a custom user interface into the frontend of the web app.

</td><td>

Complete Physical connection
Successfully deployed the custom web app
Complete a front page for the thermal cameral
Complete a script that controls the camera

</td></tr>
</table>

- Ensure the custom user interface can interact and communicate correctly with the backend code.
- Test the functionality and responsiveness of the user interface and make necessary fixes and adjustments.

Goal 6: Design and set up a database for storing thermal imaging data and connect it to the web app.

- Choose a suitable database system for storing thermal imaging data, such as MySQL or PostgreSQL.
- Design the table structure of the database to store thermal images and related data.
- Configure the database connection and ensure the web app can read from and write to the database correctly.
- Write and execute test cases to validate the functionality and data integrity of the database.

Goal 7: Find a way to attach the camera to the Farmbot.

- Research the hardware structure and interfaces of Farmbot to determine how to connect the camera to Farmbot.
- Research and select suitable camera connection methods and adapters.
- Purchase the necessary connectors, cables, and other required accessories and perform installation and debugging.

Goal 8: Find a way to connect the camera to the Raspberry Pi and control it with Python code.

- Research the hardware and interfaces of the Raspberry Pi to determine how to connect the camera to the Raspberry Pi.
- Research and select a suitable camera module and relevant drivers.
- Write Python code to control the camera, such as setting configurations and capturing images.
- Test the Python code on the Raspberry Pi to ensure the camera works correctly.

Goal 9: Run Python code implemented inside the web app to control the camera by clicking the UI on the frontend.

- Design and implement interactive functionalities with the frontend interface, such as buttons or links.
- Map the frontend interface operations to the backend Python code for controlling the camera.
- Test the functionality and reliability of the interface operations and make adjustments and fixes as needed.

Goal 10: Set up Python code to fully control the camera (set up configuration, capturing, and real-time data streaming).

- Integrate the previously written Python code to achieve full control of the camera.
- Ensure the Python code can handle camera configuration settings, image capturing, and real-time data streaming.
- Test the functionality and performance of the Python code and make necessary fixes and optimizations.

Goal 11: Attempt to save thermal imaging data to the database.

- Create appropriate tables and fields in the previously designed database to store thermal imaging data.
- Modify the Python code to enable saving thermal imaging data to the database.
- Run test scenarios to validate whether thermal imaging data is correctly saved to the database.
- Check and resolve any issues related to data saving or database connections.

Goal 12: Test deployment and transfer data to the shell and pack it in a Docker container.

- Configure a deployment environment to simulate the target production environment.
- Transfer the codebase and necessary dependencies to the deployment environment.
- Test the deployment process, including code execution and data transfer.
- Package the application and its dependencies into a Docker container.
- Verify the correct operation of the Docker container and ensure it has access to the required resources.

Goal 13: Write and execute designed test cases to test system functionality.

- Write test scripts based on the designed test cases or use appropriate testing tools.
- Execute the test cases to validate the functionality and performance of the system under various scenarios.
- Check the test results, document and resolve any issues discovered.
- Modify and supplement test cases as needed and rerun the tests.

Goal 14: Documentation review.

- Carefully review existing documentation to ensure its completeness and accuracy.
- Update the documentation based on project changes or additions, ensuring alignment with the actual situation.
- Ensure the documentation provides clear instructions for system setup, configuration, and usage.
- Adhere to required formatting or style guidelines to ensure consistency in the documentation.
- Validate the accuracy and readability of the documentation based on feedback from team members or stakeholders.

Goal 15: Release product version 1.0.

- Ensure all features and goals have been completed and successfully tested.

• Deploy the system to the target environment and ensure its smooth operation. • Write release notes and documentation to inform users about the system's features and usage. • Ensure the stability and reliability of the released version. • Mark the product version 1.0 as an official release and carry out promotion and marketing activities.	

Sprint 2 Feedback and Review

Feedback from Nir:

- Happy with the progress and understand there are so many challenges.
- Have a short discussion with professors from the Agriculture department to learn the user stories.
- The integration failed in this meeting because the SD card does not work for the farmbot and we failed to build a connection.

Feedback from grader:

Reviews from feedbacks	To-Dos
There is a short description of the product. Without a more in-depth knowledge of the product (FarmBot), it is difficult to understand how the user stories are related to its functionality. There is confusion between the product (FarmBot) and the project that will create it	Develop a comprehensive and clear product description for FarmBot, detailing its functionalities, benefits, and potential use cases. Additionally, clearly differentiate between the FarmBot product and the project undertaken to create it.
How are decisions made during meetings allocated to team members to be implemented? Stand-up meetings are spaced two weeks apart? How is project progress maintained? How do you know what is happening?	Establish a systematic process for decision-making and allocation of tasks during meetings. Create a protocol for regular progress reports and updates in-between stand-up meetings.
The Trello board is used to keep track of project progress. But where is the “well-defined development plan” that the team created? There does not seem to be sufficient detail in the Trello board as it is now. How does one link the tasks on the Trello board to user stories?	The Trello board should be updated to include more detailed tasks, milestones, and timelines that align directly with the user stories. Integrate user stories into each task, linking them back to the project's objectives.
The Sprint planning page has team roles (though no deputies), a short presentation of capacity planning, and very high-level major sprint activities. How were team members allocated to activities?	The Sprint Planning page should include a detailed section that outlines the assignment of team members to specific activities based on their roles, skills, and capacity. This system should take into consideration the estimated effort for each task and the individual capacities of team members.
Short sprint review. Is this the full sprint review? Not all items on the retrospective have a discussion of issues and how to address them. How will the learning from the issues in this sprint be used to improve the performance of the team in the coming sprints?	For each item, there should be a detailed discussion on the issue encountered and the measures to fix it. Furthermore, a section should be dedicated to outlining how learnings from the current sprint's issues will be applied to improve performance in future sprints.

Sprint 2 Retro

Date

2023-4-30

Team members

- Dikai ZHU
- Zhiyu Chen
- @Chengtian Jiang
- @Yuxi He
- @Tingzheng Ren

Goals

- *Sprint 2 review*
- *Sprint 2 retrospective*
- *Sprint 3 planning*

Sprint 2 Review

Item	Discussion
Review the code	• All team members do the final code review for others' code
Review the sprint 2 assignment docs	• All team members take participate in the review of documents on the Confluence and make the final updates: Testing, Architecture Design, user story • Merge codes (dev and others) into the main branch
Review the architecture	▪ Update diagrams in Architecture Design
Review the testing	• All team members do the testing on the features developed by others

Sprint 2 Retro

What we did well	What we didn't do well
Good communication and teamwork based on the agile process	We can not compete all tasks we planned
Good code quality: produce reusable function components.	We cost more time learning the code language of the existing system
Good Confluence structure.	We face challenges on insert UI to the frontend and visualize data.
We are clear about what other team members have done in this sprint.	Trello daily update not in time
We release a product that is of a standard and form demonstratable to the client.	
We got feedback from the client and we organized what need to be done in the next sprint.	
Every team member complete their tasks on time	

Sprint 3 Feedback and Review

Feedback from Nir:

- Happy with the new way of solving the connection problem between raspberry pi and the thermal camera.
- A complete development process with a demo of the product.
- Happy with the answer: the solution is useful for the next group in the next semeste

Review:

We are delighted to receive such positive feedback from our valued user regarding the new solution we developed for the connection problem between the Raspberry Pi and the thermal camera. Our team is thrilled to know that the solution has met and exceeded your expectations.

Throughout the development process, our team was committed to delivering a comprehensive and reliable solution. We carefully analyzed the connection problem and devised a strategic plan to address it effectively. Our team's technical expertise and dedication were instrumental in developing a solution that seamlessly connects the Raspberry Pi and the thermal camera.

One of the key highlights of our development process was the inclusion of a complete demo of the product. We understand the importance of practical demonstrations in showcasing the capabilities and usability of a solution. By providing a thorough demo, we aimed to present the solution in a tangible way, ensuring its potential applications and benefits were clearly conveyed.

We are pleased to hear that you found the demo helpful and informative. It is our intention to ensure that the solution we provide not only solves the existing problem but also proves to be valuable for future users. Knowing that you believe our solution will be useful for the next group in the upcoming semester reinforces our confidence in its effectiveness and reliability.

We would like to express our gratitude for your trust in our team and our solution. Your satisfaction is a testament to our team's hard work, dedication, and commitment to delivering high-quality results. We are constantly striving to provide innovative solutions that meet the needs of our users, and your positive feedback reinforces our drive to continue pushing the boundaries of excellence.

Thank you once again for your feedback, and we look forward to serving you and future users with more cutting-edge solutions in the future.

Sprint 3 Planning

Team

Name	Role
Yuxi He	Product Owner
Dikai Zhu	Scrum Master
Tingzheng Ren	Testing leader
Zhiyu Chen	Software Developer
Chengtian Jiang	Software Developer

Capacity planning

	Current sprint
Total days	24 days
Total hours fully working on the projects	48 hours/ppl 240 hours for whole team
Team capacity	61%
Total story points capacity	153
Total story points planned to complete	94

Sprint Planning

Goal	Plan
• Bug fix for previous release 1. Review bug reports and identify the specific issues that need to be fixed. 2. Analyze the codebase to understand the cause of the bugs. 3. Implement necessary code changes to fix the identified bugs. 4. Test the bug fixes thoroughly to ensure they resolve the issues. 5. Document the bug fixes and update the release notes. • Set up controls for auto-picture taking 1. Identify the hardware requirements for auto-picture taking, such as cameras and triggering mechanisms. 2. Research and select appropriate software libraries or APIs to interface with the hardware. 3. Implement code to control the camera, including triggering the capture process. 4. Test the camera controls to ensure proper functioning. 5. Document the setup process and any configuration options available. • Set up auto picture process and save analyzed data to database 1. Define the desired image analysis tasks, such as object detection or temperature measurement. 2. Implement the necessary computer vision algorithms to analyze the captured images. 3. Store the analyzed data, along with the corresponding images, in a database. 4. Test the auto picture process and data storage to ensure correctness. 5. Document the image analysis algorithms used and the database structure for storing the data. • Test deployment and transfer data into shell and pack in docker. 1. Set up a deployment environment to simulate the target production environment.	• Complete project integration • The robot can send command and move to a defied destination • Insert a UI to the front end of web app that users can access and could configurate the camera • Collect data from bot and visualize data • user manual with informations about steps, readme file about farmbot and web app

2. Transfer the codebase and required dependencies to the deployment environment. 3. Test the deployment process, including code execution and data transfer. 4. Package the application and its dependencies into a Docker container. 5. Verify that the Docker container runs correctly and can access the required resources. 6. Document the deployment process and provide instructions for deploying the application using Docker. • Documentations review 1. Review the existing documentation for completeness and accuracy. 2. Update the documentation to reflect any changes or additions made during the project. 3. Ensure that the documentation provides clear instructions for setting up, configuring, and using the system. 4. Verify that the documentation adheres to any established formatting or style guidelines. 5. Document any known limitations or issues that users may encounter. 6. Seek feedback from team members or stakeholders to validate the documentation.	

Sprint 3 Retro

Date

2023-4-25

Team members

- Dikai ZHU
- Zhiyu Chen
- @Chengtian Jiang
- @Yuxi He
- @Tingzheng Ren

Goals

- *Sprint 3 review*
- *Sprint 3 retrospective*

Sprint 3 Review

Item	Discussion
Review the code	• All team members do the final code review for others' code
Review the sprint 3 assignment docs	• All team members take participate in the review of documents on the Confluence and make the final updates: Testing, Architecture Design, user story • Merge codes (dev and others) into the main branch
Review the architecture	• Update diagrams in Architecture Design
Review the testing	• All team members do the testing on the features developed by others
Review user manual with informations about steps	• All team members do the readme file and user manual review

Sprint 3 Retro

What we did well	What we didn't do well
Good communication and teamwork based on the agile process	We faced challenges in completing all the tasks that were planned for the sprint. It is important for us to analyze the reasons behind this and identify ways to improve our task estimation and allocation processes.
Good code quality: produce reusable function components.	We encountered difficulties and delays in understanding the code language of the existing system. This impacted our productivity and the speed at which we could implement changes. Going forward, we need to allocate time for learning and familiarizing ourselves with the codebase before diving into development tasks.
Good Confluence structure.	Inserting UI elements into the frontend and effectively visualizing data posed challenges for us. It is crucial to identify the specific obstacles we faced and explore potential solutions or alternative approaches to address these challenges more efficiently in future sprints.
We are clear about what other team members have done in this sprint.	We experienced delays in updating our Trello board with daily progress and task statuses. This lack of timely updates hindered transparency and coordination within the team. To improve communication and collaboration, we need to ensure that daily updates on Trello are consistently maintained and kept up to date.
We release a product that is of a standard and form demonstratable to the client.	

We got feedback from the client and we organized what need to be done in the next sprint.	
Every team member complete their tasks on time	

Sprint 4 Planning

Team

Name	Role
Yuxi He	Product Owner
Dikai Zhu	Scrum Master
Tingzheng Ren	Testing leader
Zhiyu Chen	Software Developer
Chengtian Jiang	Software Developer

Sprint Planning

Goal	Plan
1. Database Analysis: While some progress has been made, further analysis is required for the database. Additionally, we have successfully implemented automatic picture uploads to a local server. However, this server is separate from the web app's server. As a next step, we should work on connecting the backend local server to the web app. 2. Integration of Capture Process: The button for initiating the capture process has been transformed into a URL link. To complete the integration, we need to integrate this URL link into the web app. 3. Testing and Development of Customized Farmbot OS: The customized Farmbot OS offers great potential for further development. To ensure its effectiveness, rigorous testing is necessary.	• More analysis is needed for the database. • Implemented successful automatic picture uploads to a separate local server. • The capture process button has been transformed into a URL link and needs to be integrated into the web app. • The customized Farmbot OS has great potential for development and requires rigorous testing.

Testing

- Server Testing

Farmbot Project Home

Server Testing

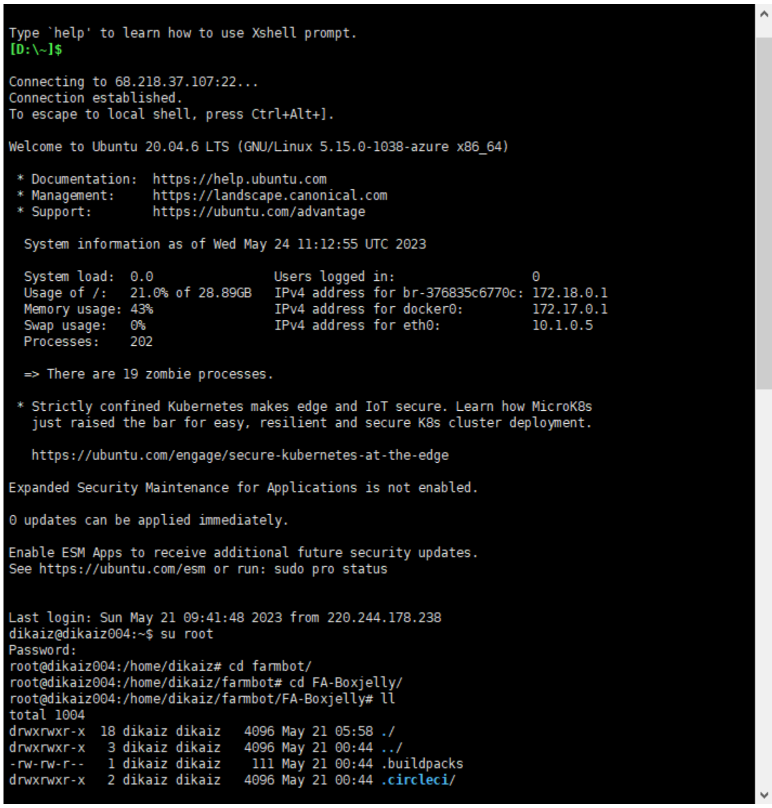
Versions

VersionID	Description	Creator	Date
2.1	Raspberry server initialization	Jianeng	2022-05-17
2.0	Webapp Server Update	Dikai	2022-04-27
1.0	Initialise Webapp Server	Dikai	2022-04-20

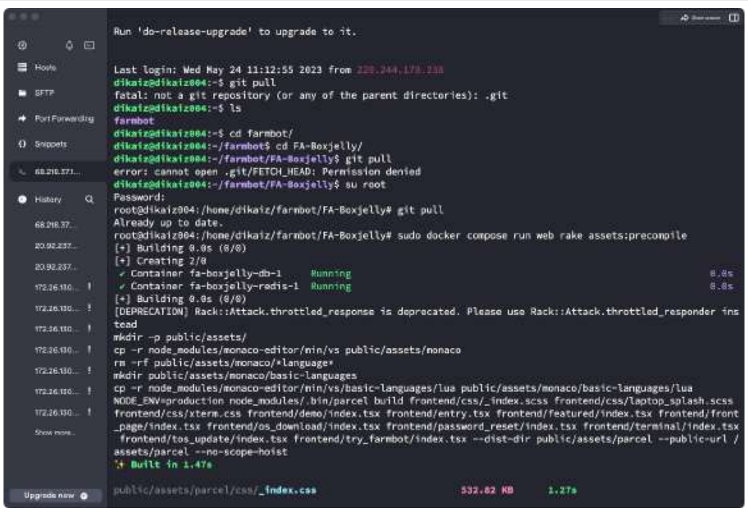
Webapp Server Testing

<table>
<tr><th>ID</th><th>Description</th><th>Result</th><th>Status</th><th>Actor</th></tr>
<tr><td>1</td><td>To verify the successful deployment of the web application on our own server in the Azure cloud platform.

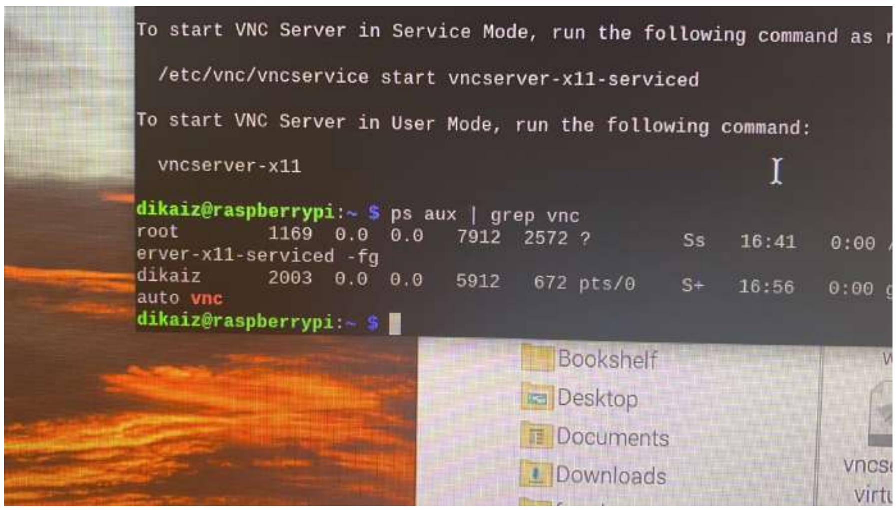
Preconditions:

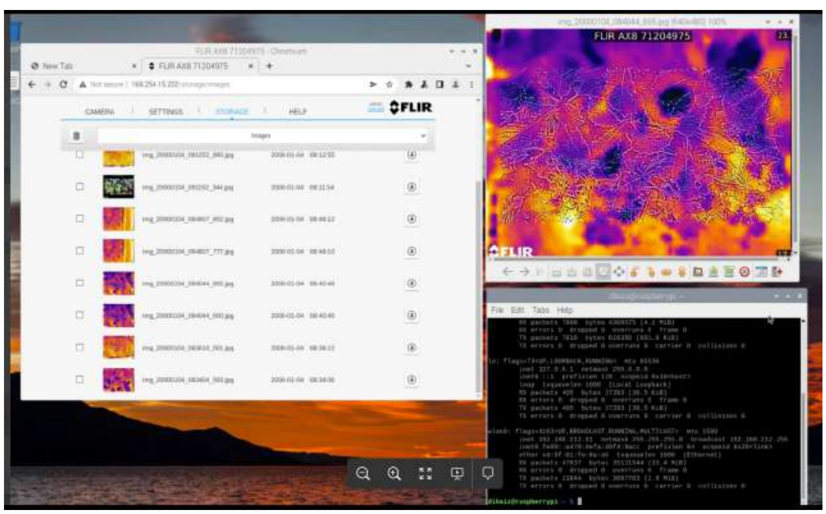
1. Access to the Azure portal with necessary permissions.
2. The web application and its deployment package are available.</td><td>
</td><td>Success</td><td>Dikai Zhu</td></tr>
<tr><td>2</td><td>To verify the successful update of the web application on our own server in the Azure cloud platform.

Preconditions:

1. Access to the Azure portal with necessary permissions.
2. The web application is already deployed and running on the Azure server.</td><td>
</td><td>Success</td><td>Tingzheng Reng</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
</td><td></td><td></td></tr>
<tr><td>3</td><td>To verify the successful connection and communication between the Raspberry server and a thermal camera, ensuring data retrieval and functionality.

Preconditions:
Raspberry server with appropriate hardware and operating system.
The thermal camera is compatible with the Raspberry server.
Necessary libraries or drivers installed on the Raspberry server for thermal camera integration.</td><td>


</td><td>Success</td><td>Chengtian Jiang</td></tr>
</table>

Products

This section Contains demos for the sprints releases

Sprint 2 Demo

Sprint 3 Demo

Farmbot Project Home

Sprint 2 Demo

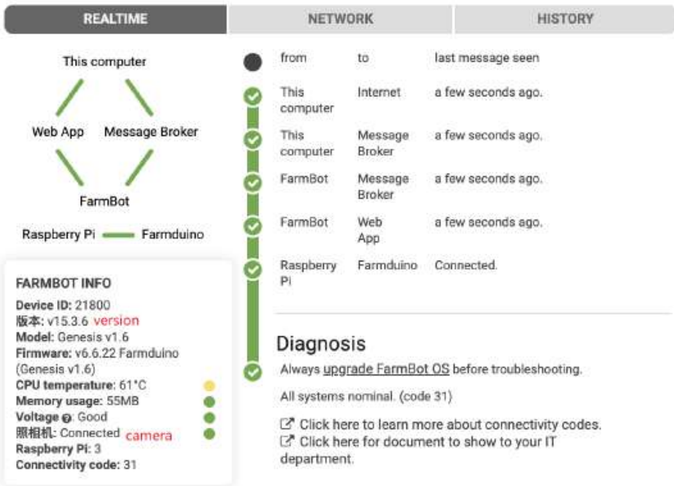
This page shows the functionality we have implemented in sprint 2.

Thermal Camera Connection

We connected the thermal camera to the farmbot robot with data cables. On the top left of the picture below, there is the Raspberry Pi, whereas there are two splitters in the middle and a thermal imaging camera at the bottom right of the picture.

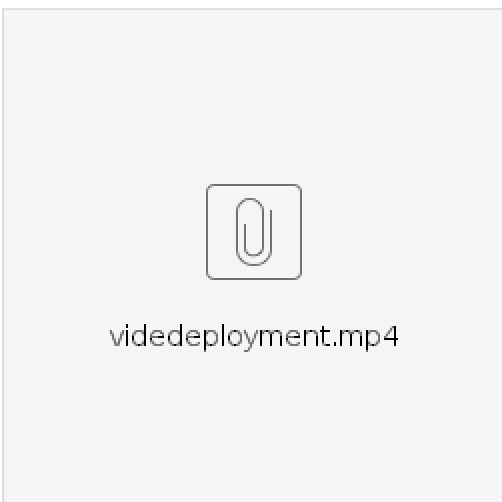
By testing the connection offline, we found that the thermal imaging camera could be used successfully and take pictures of the specified plants.

After our adjustments, the farmbot robot can successfully connect to various important modules such as web app and computer.

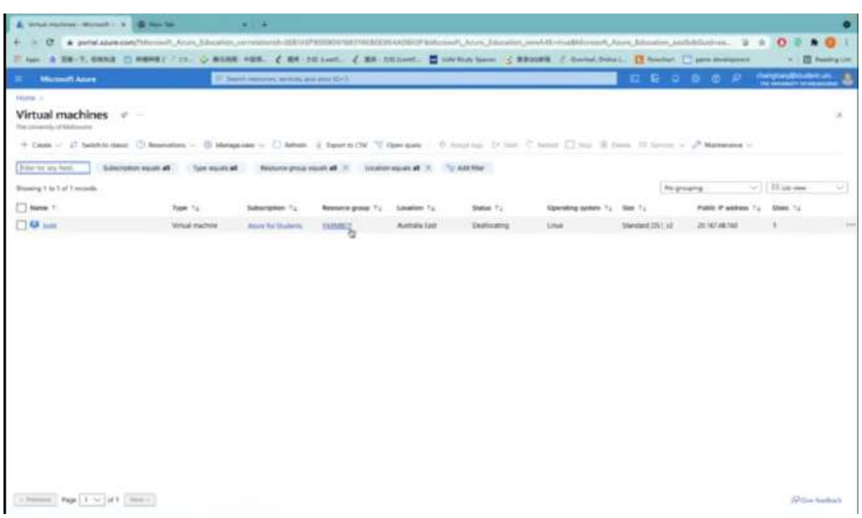


Web app deployment

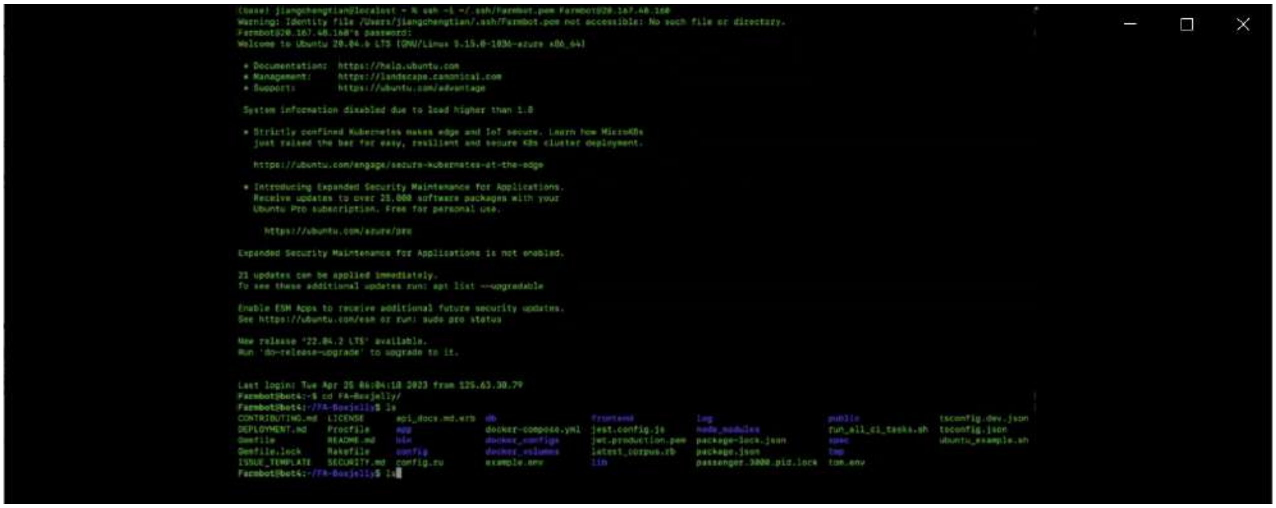
The video below shows the process of how we deploy the web app and successfully access the login page.



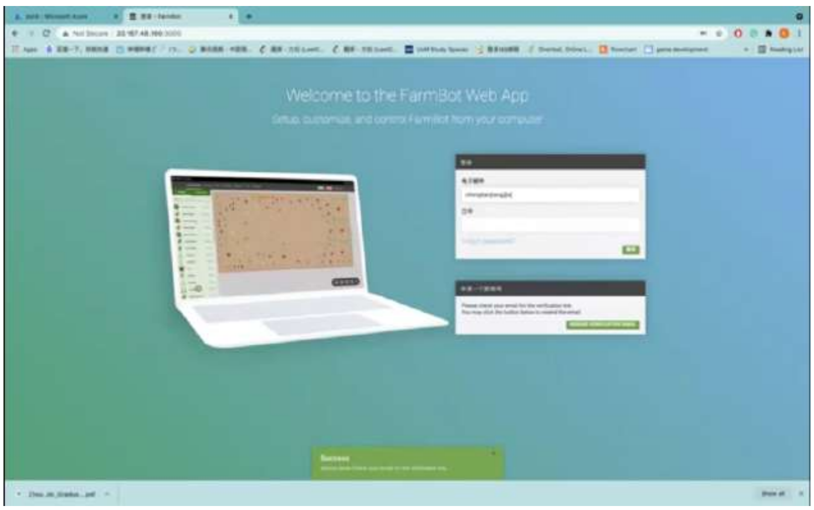
We rent a cloud server of Azure to deploy the web app of farmbot.



We connected the virtue machine on Azure to deploy the web app.

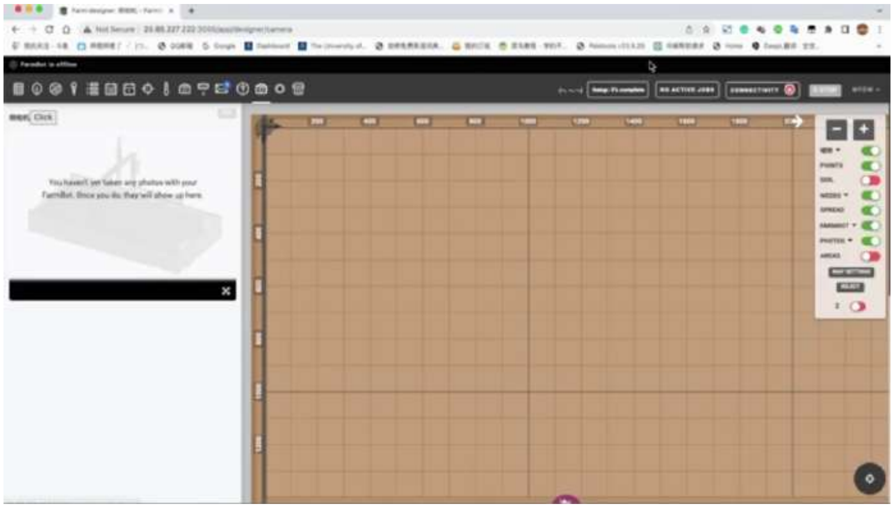


We logged in the web app.

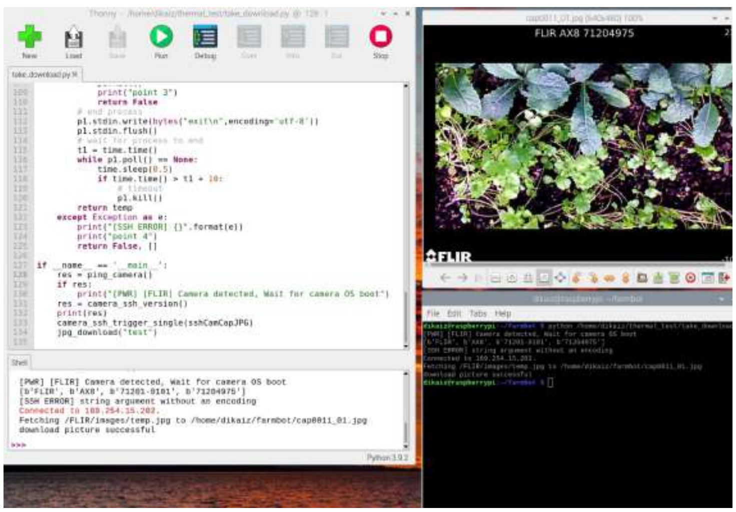


Thermal camera Module

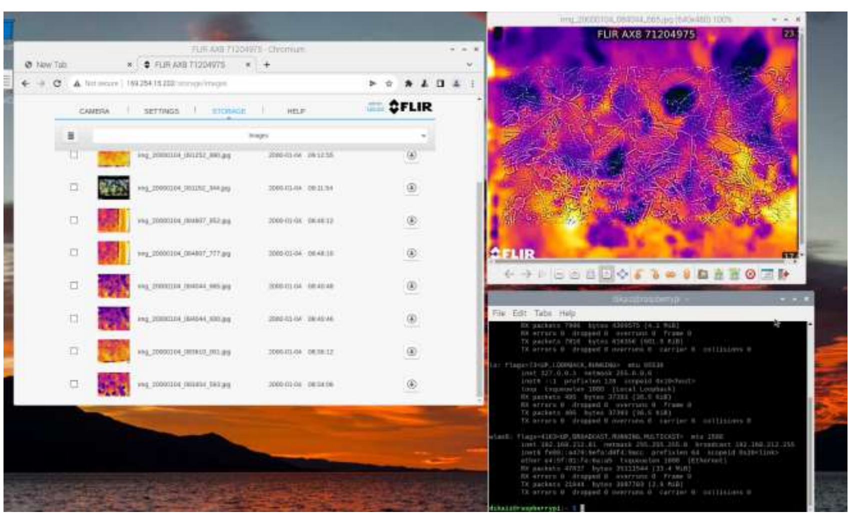
We created a new module to implement the operations related to the thermal camera. For instance, we can choose a specific plant to take a picture of and display the picture on the left side of the module. Then we can get the temperature data of this plant after analysis. For sprint 2, we can display the images taken by the thermal camera.



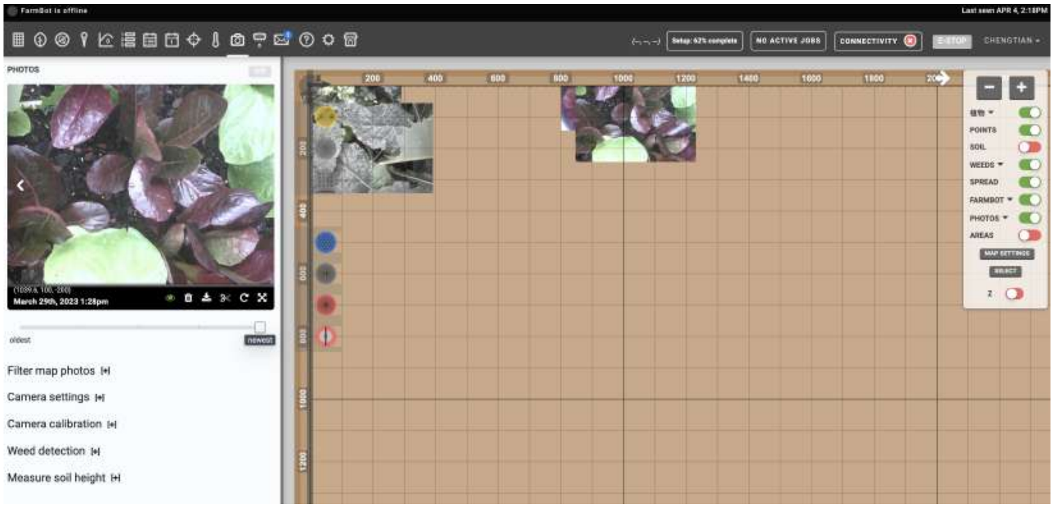
We wrote a python script to download the pictures taken by the thermal camera.



Also the thermal radiation images.



Then we can upload these images to display them on the module.



FarmBot is offline
Last seen APR 4, 2:18PM
NO ACTIVE JOBS
CONNECTIVITY
E-STOP
CHENGTIAN
PHOTOS
March 29th, 2023 1:28pm
oldest
newest
Filter map photos
Camera settings
Camera calibration
Weed detection
Measure soil height
POINTS
SOIL
WEEDS
SPREAD
FARMBOT
PHOTOS
AREAS
MAP SETTINGS
SELECT

Sprint 3 Demo

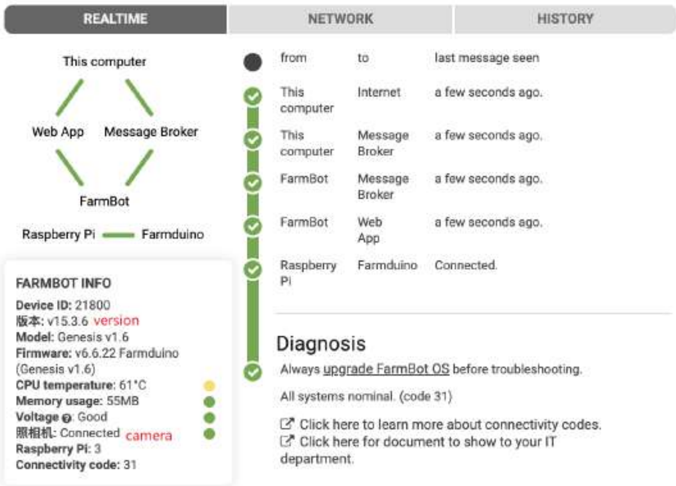
This page shows the functionality we have implemented in sprint 3.

Thermal Camera Connection

We connected the thermal camera to the farmbot robot with data cables. On the top left of the picture below, there is the Raspberry Pi, whereas there are two splitters in the middle and a thermal imaging camera at the bottom right of the picture.

By testing the connection offline, we found that the thermal imaging camera could be used successfully and take pictures of the specified plants.

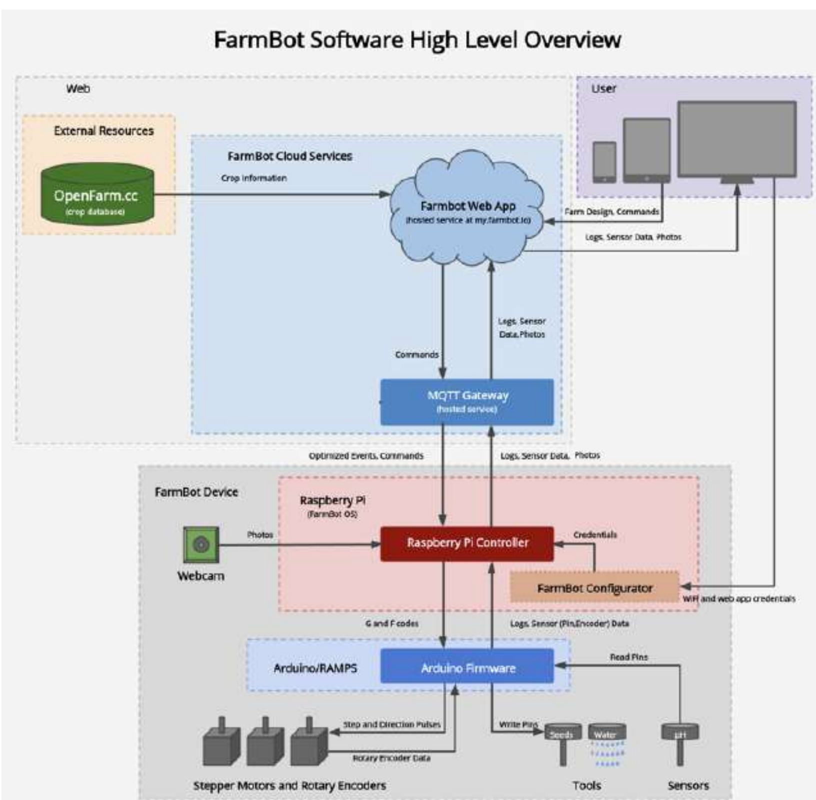
After our adjustments, the farmbot robot can successfully connect to various important modules such as web app and computer.



Technology Framework

What composed the Farmbot Project?

- Farmbot arm Raspberry Pi and Farmduino
- Front-end: typescript
- Database: postgresql
- Back-end: flask
- OS: erlang, nerves and elixir



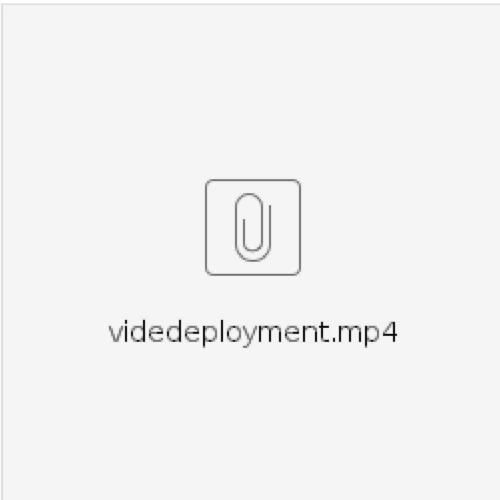
Information Flow

- click button on front end
- request created
- message broker dispatch request
- OS receives request

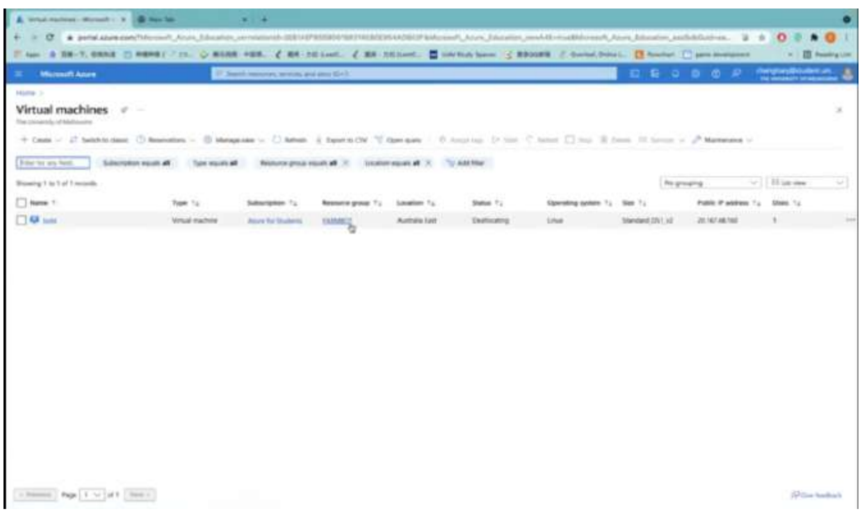
- OS create a node on AST
- AST execute to the node
- Action taken

Web App Deployment

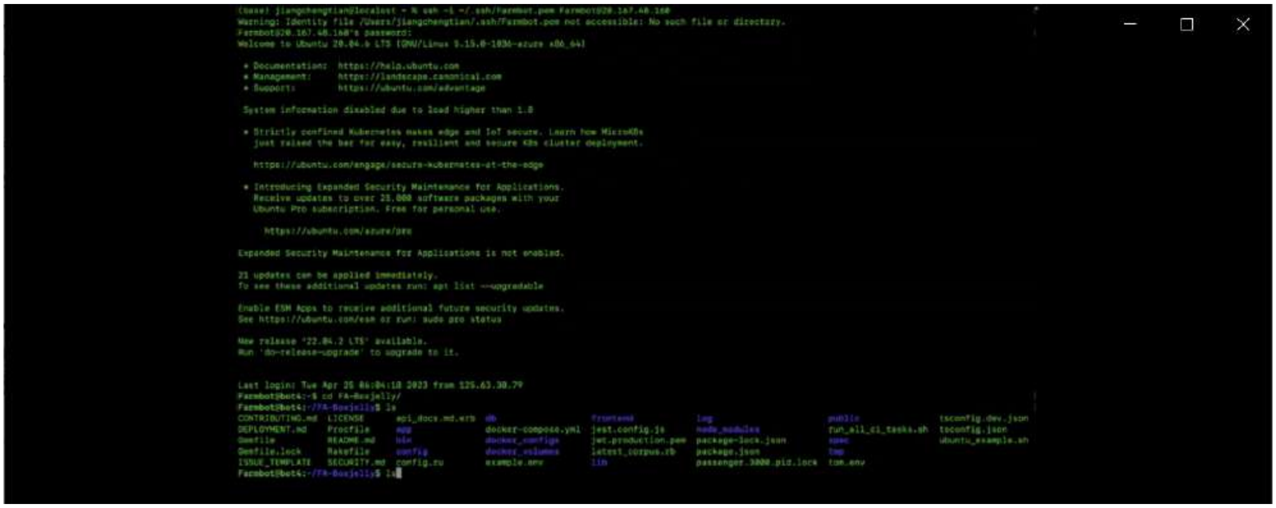
The video below shows the process of how we deploy the web app and successfully access the login page.



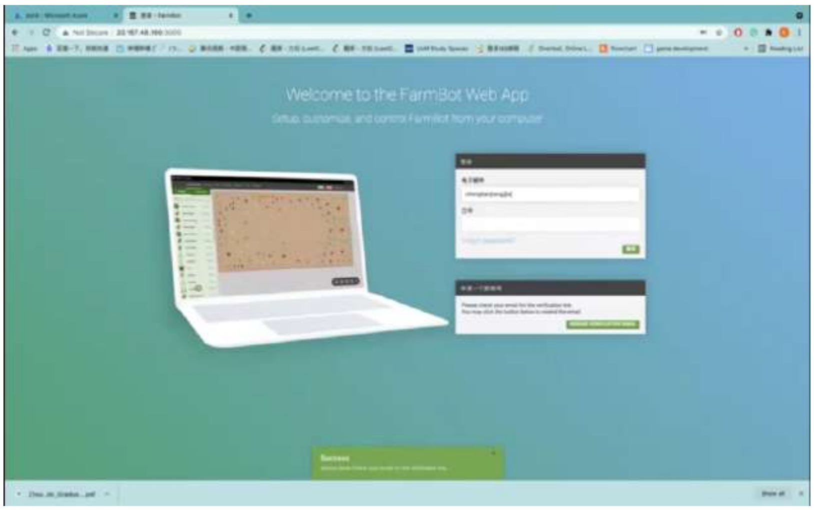
We rent a cloud server of Azure to deploy the web app of Farmbot.



We connected the virtue machine on Azure to deploy the web app.



We logged in the web app.



Web-app URL

URL: http://68.218.37.107:3000/

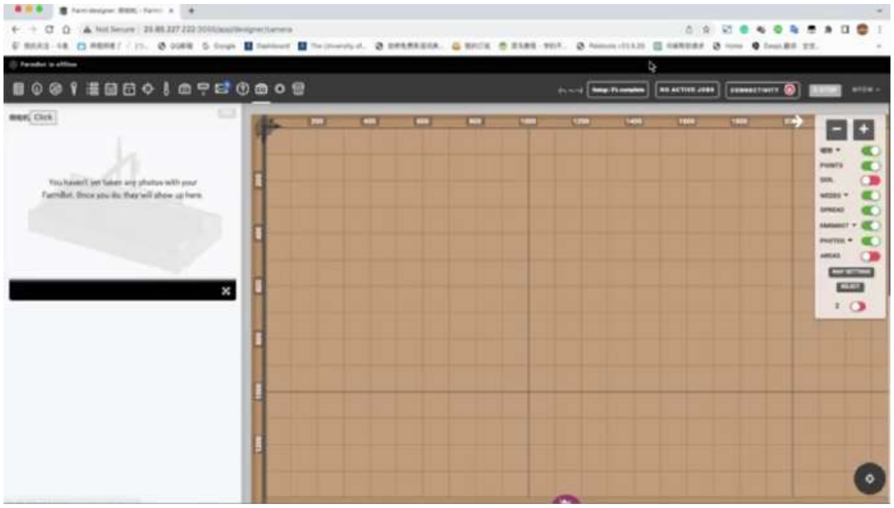
Deployed Web App Server IP: 68.218.37.107

Web-app Username: dikaiz@student.unimelb.edu.au

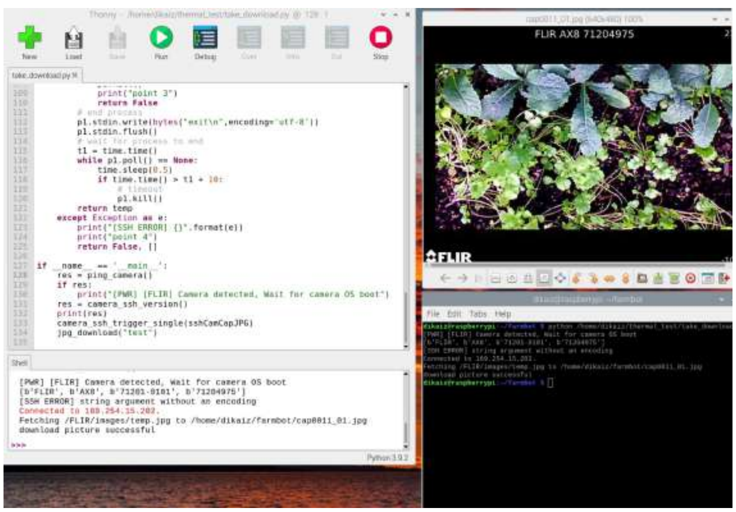
Web-app Password: 12345678

Thermal Camera Module

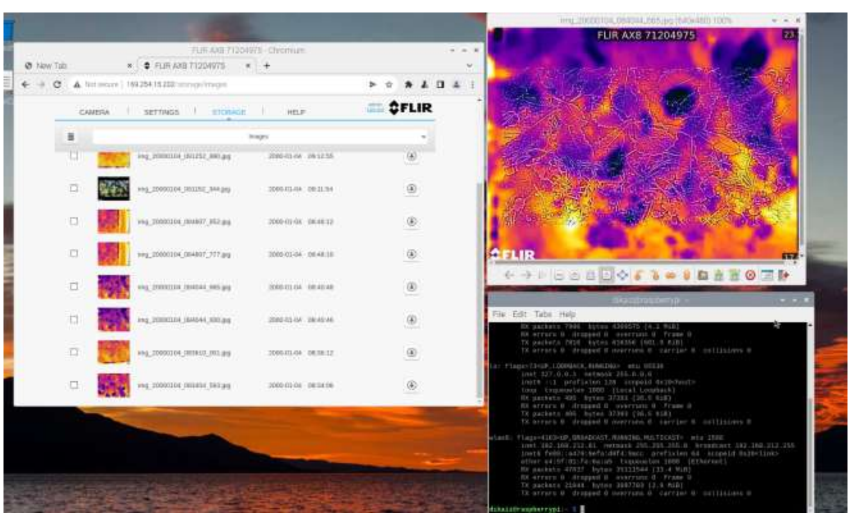
We created a new module to implement the operations related to the thermal camera. For instance, we can choose a specific plant to take a picture of and display the picture on the left side of the module. Then we can get the temperature data of this plant after analysis. For sprint 3, we can display the images taken by the thermal camera.



We wrote a python script to download the pictures taken by the thermal camera.

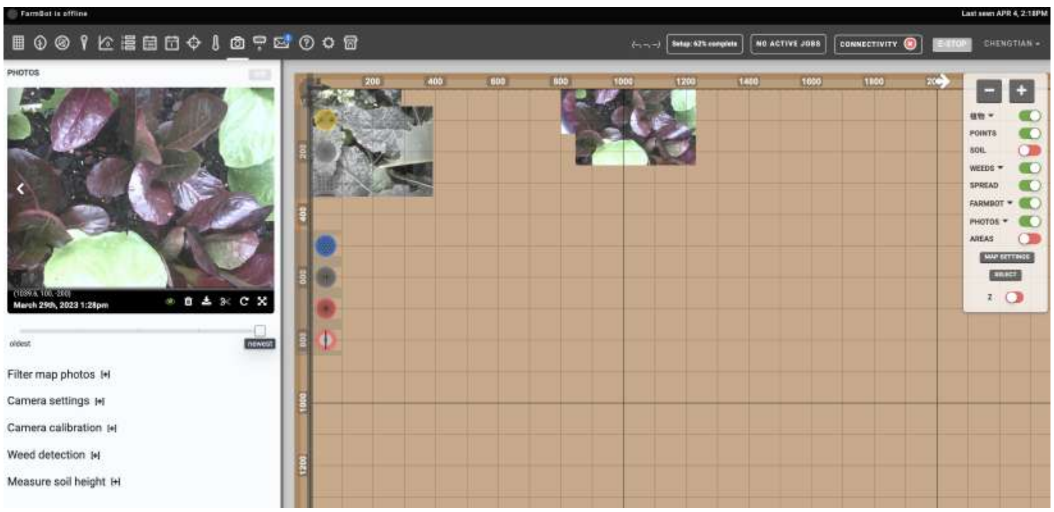


Also the thermal radiation images.

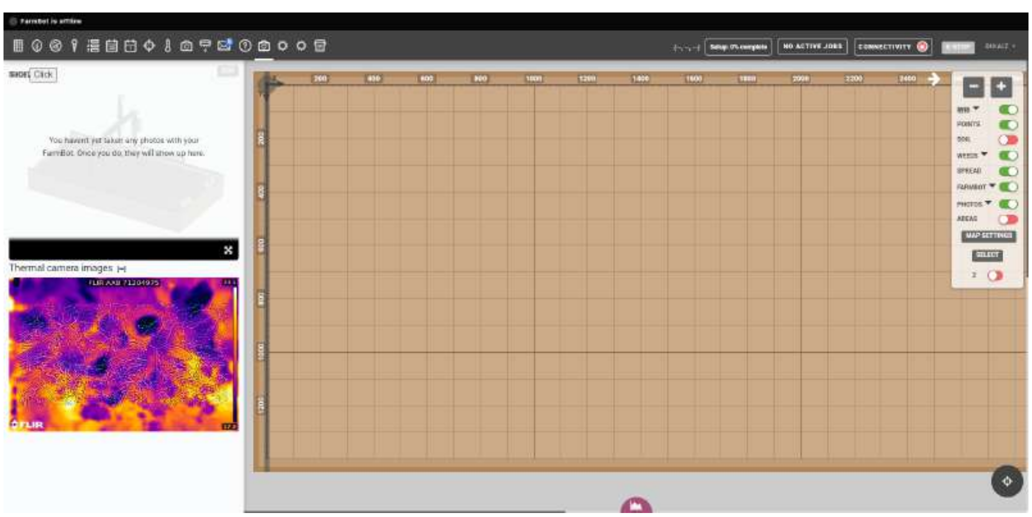


This video below shows how we can connect Raspberrypi to take thermal images.

Then we can upload these images to display them on the module.

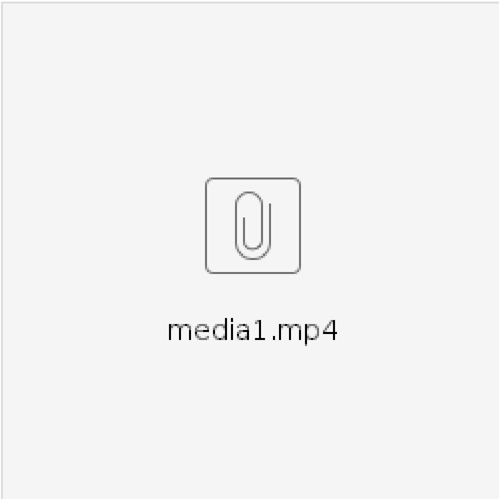


The thermal image can be shown on our webpage.



Farmbot Control

The video below shows the process how we can control the Farmbot to move to a certain place.



Farmbot OS reconstruction

Details in here