RELEASE NOTE

1. Introduction to the Project

1.1. Background Project

The FarmBot project, initiated by Rory Aronson in 2011, aims to evolutionize traditional farming methods using open-source technology. By combining robotics, web-based control systems, and customizable open-source software, FarmBot presents a unique approach to enhancing agricultural processes for improved efficiency and sustainability.

One of the key attractions of the FarmBot project is its open-source nature, which encourages collaboration among global researchers, developers, and farmers. This collective effort has propelled FarmBot from a mere concept to a powerful tool with the potential to transform agriculture. With its open-source foundation, FarmBot is poised to revolutionize food production by providing a more efficient and environmentally friendly approach.

The adaptability of FarmBot is another notable aspect, offering a range of flexible and user-friendly software that can be customized to suit diverse needs and objectives. This versatility makes FarmBot suitable for various users, including small-scale farmers, large agribusinesses, and research institutions.

1.2. Project Goal

The aim of the project is integrate thermal imaging cameras into the FarmBot system. By incorporating thermal imaging technology, this integration intends to provide valuable insights into plant health and productivity for farmers, researchers, and other users. This knowledge will empower them to effectively manage crops and make informed decisions.

The project has two specific goals. Firstly, it enables users to remotely collect thermal data from plants. Users can access the FarmBot web application to trigger thermal imaging manually or set the system to automatically collect data at specific intervals. The captured thermal images are then uploaded to the web application, creating a comprehensive dataset for reviewing plant health, identifying issues, and devising intervention strategies.

Secondly, the project aims to support agricultural research. The automatically collected thermal data serves as a valuable asset for researchers, including Professor Sigfredo Fuentes. By utilizing thermal imaging, researchers gain insights into how environmental factors influence plant health and productivity. This knowledge can contribute to the development of resilient and sustainable agricultural practices.

2. Organization of Resources on GitHub

2.1. Repository

GitHub is a web-based hosting service for version control using git. It provides a platform for developers to collaborate on different projects and share their work with others. For our project, we use GitHub as a important tool and deployment staff.

Here are a few tips for organizing our resources on GitHub:

The first step towards organizing our resources on GitHub is to create a repository. A repository is like a folder that contains all our project files. When we create a repository, we can add files, folders, and sub folders to it. We can also add collaborators to your repository, so they can contribute to your project.

Repository link:

https://github.com/COMP90082-2023-SM1/FA-Boxjelly

2.2. Descriptive Name

When we creating files and folders, use descriptive names that are easy to understand. For our project, a folder named "src" could contain source code files, while a folder named "docs" could contain documentation files. Using descriptive names will make it easier for others to find and navigate our repository.

2.3. Readme File

A readme file is a file that provides information about your project. It can include a description of the project, instructions on how to use it, and other important information. By including a readme file in your repository, you can help others understand your project and how to use it.

2.4. Branches

Branches are copies of your repository that you can make changes to without affecting the original repository. In our repository, we have several branches include os, rpi, frontend and thermal control related.

2.5. Pull Requests

Pull requests allow you to propose changes to the main project and collaborate with others. In our project, we used pull requests extensively, but we also communicated about them through WeChat to avoid creating too many pull requests. This helped us keep our project organized and well-maintained.

2.6. Tags and Releases

Tags and releases are used to mark specific points in your project's history. To meet the project requirements, we created some tags and releases to help keep track of changes to our project and make it easier for others to download and use it. This will also ensure that we can receive marks for meeting the requirements.

2.7. Repository Structures

Here are some of the key files and directories in our repository:

- **docs/:** This directory contains documentation files for our project.
- **src/:** This directory contains the source code for our project.
- **tests/:** This directory contains user/system tests for our project.
- **prototypes/low fidelity/:** This directory contains low fidelity files such as screens and mockups.
- **prototypes/high fidelity/:** This directory contains high fidelity files such as screens and source files.
- **ui/:** This directory contains all the images created for the prototypes - icons, fonts, backgrounds, etc. This is different from the prototypes' folders. These are the graphical elements that go into the prototypes.
- **data samples/:** This directory contains documents you need to generate with all the data inputs necessary to simulate/demonstrate your prototype.
- **.env:** The .env file is a document that contains environment variables used in the project.
- **README.md:** The README file is a document that contains useful information in the project and it must be updated at all times.

2.8. Key Files and Directories

Here is some key changes in our GitHub(include code and document), also mentioned it in README file as Sprint change Log:

Sprint 1(Finished: 2023-03-26):

- Finish the project plan and the project overview. (2023-03-26)
- Project review and Sprint 1 release. (2023-03-26)
- ***All files are not changed in this sprint.***

Sprint 2(Finished: 2023-04-30):

Code:

- Web app page: Add a new page for thermal camera. (2023-04-25)

Boxjelly/tree/3b7c5c4d4c465f7100e7d8c42c1ebdd8241d1fb0/src/front_end/frontend/camera

- Thermal camera: Add a new thermal camera control python script. (2023-04-25)

Boxjelly/tree/e32b4bd674f53096204b8d2fdd65d27c1874a65c/src/thermal_comtrol_scripts

- ***All changed codes in src folder***

Document:

- Code review: Add code review document. (2023-04-25)
- Confluence ducument: Add confluence document. (2023-04-25)
- ***All changed documents in docs folder***

Sprint 3(Finished: 2023-05-25):

Code:

- Web app page: Add Thermal Camera images in our panel. (2023-05-25)

Boxjelly/tree/789d38f3e2c01152238359ec3b8ca6f29ff4973d/src/front_end

- Resberry pie: Add python code for our own resberry pie server (2023-05-25)

Boxjelly/tree/789d38f3e2c01152238359ec3b8ca6f29ff4973d/src/Rasp%20server

- rpi3: Add python code for rpi3 (2023-05-25)

Boxjelly/tree/789d38f3e2c01152238359ec3b8ca6f29ff4973d/src/rpi3

- Farmbot OS: Add python code for farmbot os (2023-05-25)

Boxjelly/tree/789d38f3e2c01152238359ec3b8ca6f29ff4973d/src/Farmbot_os

- ***All changed codes in src folder***

Document:

- Confluence ducument: Add confluence document. (2023-05-25)
- ***All changed documents in docs folder***

Sprint 4(Finished: 2023-06-8):

- Finish group reflection. (2023-06-8)
- Project review and Sprint 4 release. (2023-06-8)
- ***All files are not changed in this sprint.***

3. Project Prerequisite

3.1. Resource name: Web App Server

For: Web app custom deployment. The web app server can be put here.

Where: Azure(used), Google Cloud, etc.

How:

1. Sign in and create Azure server: Visit the Azure Portal website (portal.azure.com) and sign in using your Azure account. In the Azure Portal, navigate to the "Virtual Machines" section and click on "Create" to start the process of creating a new virtual machine.
2. Configure VM Settings: Choose the operating system you want to use for your server. In this case, we use Linux for our web app server. Then, specify the details for your virtual machine, such as the name, region, resource group, and virtual machine size. These will be used for server deployment.
3. Configure Networking: Set up the networking options for your server. This includes selecting a virtual network, subnet, and configuring network security groups and public IP settings.
4. Set up Authentication: Specify the authentication method for accessing your server. Azure offers options such as SSH keys or username/password-based authentication. You can choose any of these options for server connections.
5. Review and Create: Review all the settings, especially the ports. Click on "Create" to start the deployment process. The server is used to hold the web app.

Technology:

- Server deployment pipeline

3.2. Resource name: Server codes

For: Web app front-end and back-end

Where: Github: https://github.com/COMP90082-2023-SM1/FA-Boxjelly

How: Follow the link and readme file on Github

Technology:

- Typescript for front-end development

3.3. Resource name: Raspberry Pi and server for thermal camera

For:

1. Raspberry Pi is a device for holding the server. It can communicate with the camera through cables and wireless networks for information and data transmission.

2. The server is a bridge between the web app and thermal for communications and photo storage.

Where: Online shopping platforms, shopping centers, stores

How:

1. Set up a Raspberry Pi. https://www.raspberrypi.com/
2. Build a server for data transmission and storage including instructions from the web app and photos from the thermal camera. We choose Django as our server: https://www.djangoproject.com/
3. The codes for the server can be found on Github: https://github.com/COMP90082-2023-SM1/FA-Boxjelly/tree/main/src/Rasp%20server

Technology:

- Django for raspberry server
- Python for the original codes on the thermal camera

3.4. Resource name: Thermal Camera

For: Taking photos and sending data to the web app

Where: Documentation for the camera: https://thermal-imager.com.au/?gclid=Cj0KCQjw4NujBhC5ARIsAF4Iv6c6_sOEsbgwgG6Am5oxfUWdHrY_DidS-_BDmIi0CPzz-NqTZS0lSabkaAtuPEALw_wcB

How:

1. The camera can be controlled by codes. We developed a camera controller script on Github, the script can be put in the Raspberry Pi server, and the camera can store the photos on its own local server: https://github.com/COMP90082-2023-SM1/FA-Boxjelly/tree/main/src/thermal_comtrol_scripts

Technology:

- Lua for thermal camera controller
- Database management system

3.5. Resource name: Web app Guidance and Farmbot setup

For:

1. Web app development and customization
2. Farmbot setup

Where: https://developer.farm.bot/v15/docs/farmbot-software-development

How:

The official farmbot development documentation, including:

1. Guidance to deploy the web app server.
2. Configuration for farmbot and web app connection.
3. Web app server set up advice

4. Project Running

NOTE: Because we can not keep the farmbot open after the finish of development, so the user could not test the web app.

4.1. Attach the camera to Farmbot

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

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.

download: https://support.flir.com/SwDownload/Assets/FLIR%20IP%20Config/FLIR_IP_Config_3_0.zip

4.2. Customized Web App Deployment

To prepare for self-hosting, you will need to set up a server and enable the following ports: 8883, 8080, 80, and 443. Docker is needed for successful deployment

Clone the source code to your repository with following script:

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

Then to the FA-BOXJELLY file and rename tom.env to .env, and change the IP address and MQTT IP address in the file to your server IP

Execute the following script to deploy the webapp

```
# 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! ٩(^‿^)۶
# 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
```

4.3. Install customized farmbot-OS

Install the dependencies for farmbot-OS

```
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
```

Install asdf

```
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
```

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

```
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
```

Install Nerves 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
```

Clone source code and apply Nerves Framework

```
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
```

Produce the customized farmbot-OS

```
export FARMBOT_EMAIL=dikaiz@student.unimelb.edu.au
export FARMBOT_PASSWORD=12345678
export FARMBOT_SERVER=http://[your IP address]:[your port]/
chmod +x run_all.sh
./run_all.sh
```

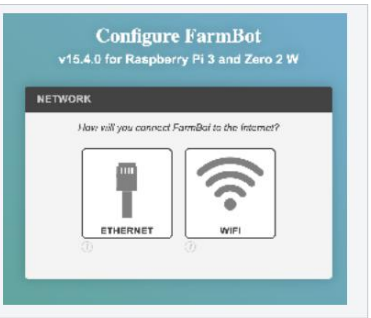
Install your OS to SD card, Pull out SD card from raspberry Pi, burn your img file to your SD card and insert it back to your farmbot

4.4. connect the farmbot to webapp

Supply power to farmbot

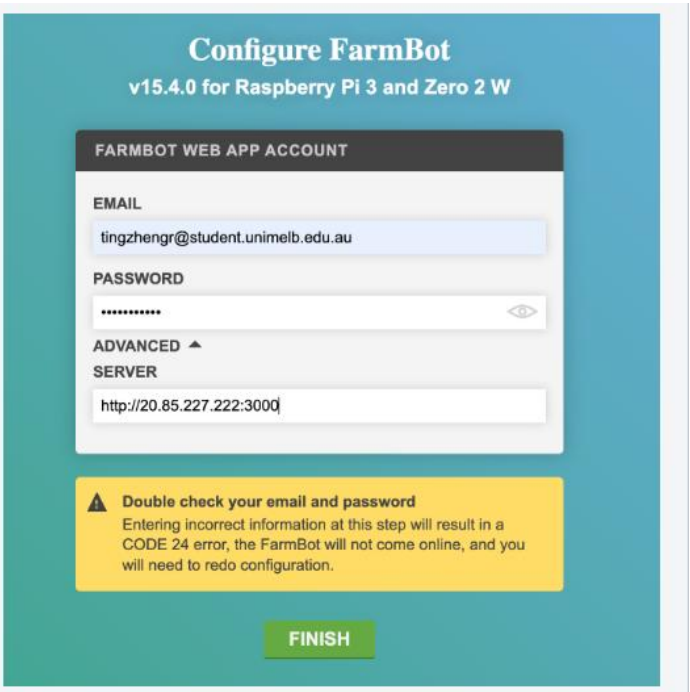
Connect to the WiFi produced by the robot (ensuring that ports 80, 443, 8883, and 3000 of the server are accessible), and the robot will direct you to the connection configuration page as shown below.

WIFI produced by the robot



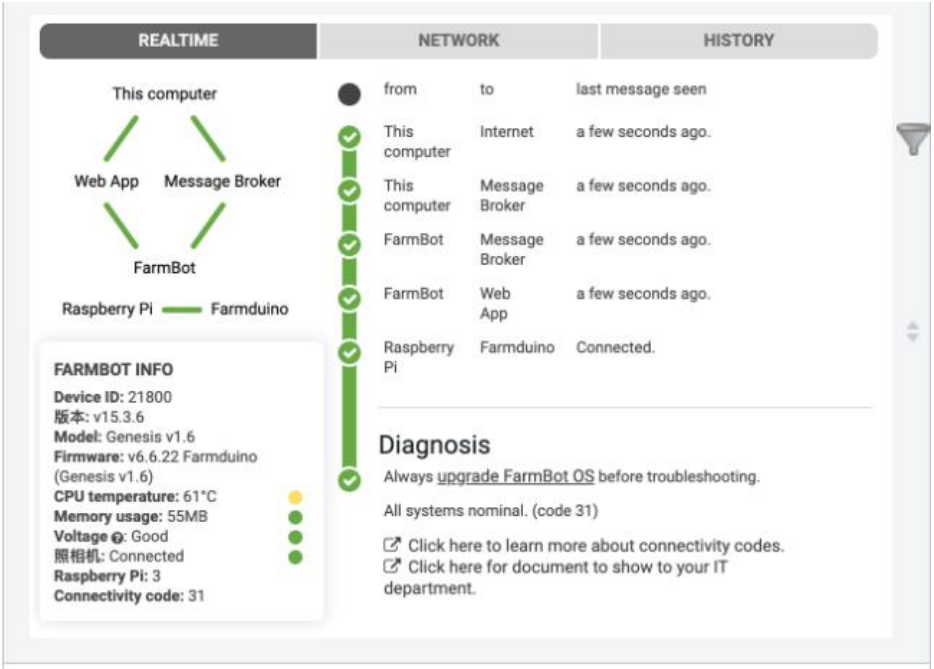
Farmbot Connect Configuration

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

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



Connected Status Page for the Robot