Farmbot Presentation

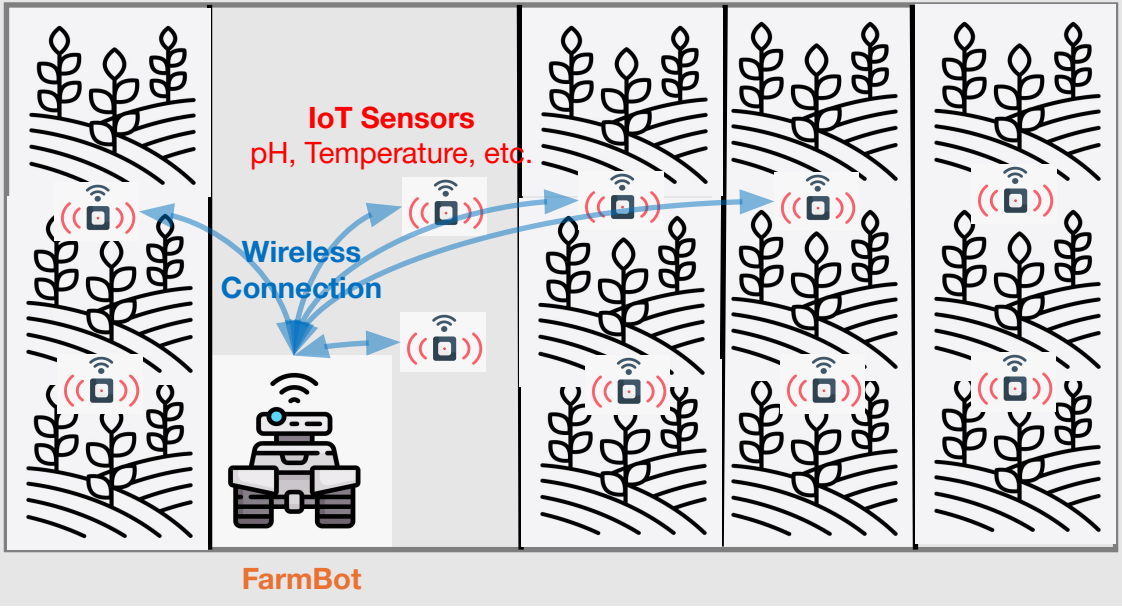
Sawyer Brundage and Joshuah Chen

August 28th, 2024

Stanford University

Farmbot - Motivation and Background

- Key idea: improve farming with IoT
  - Limited wireless infrastructure -> Use mobile base station



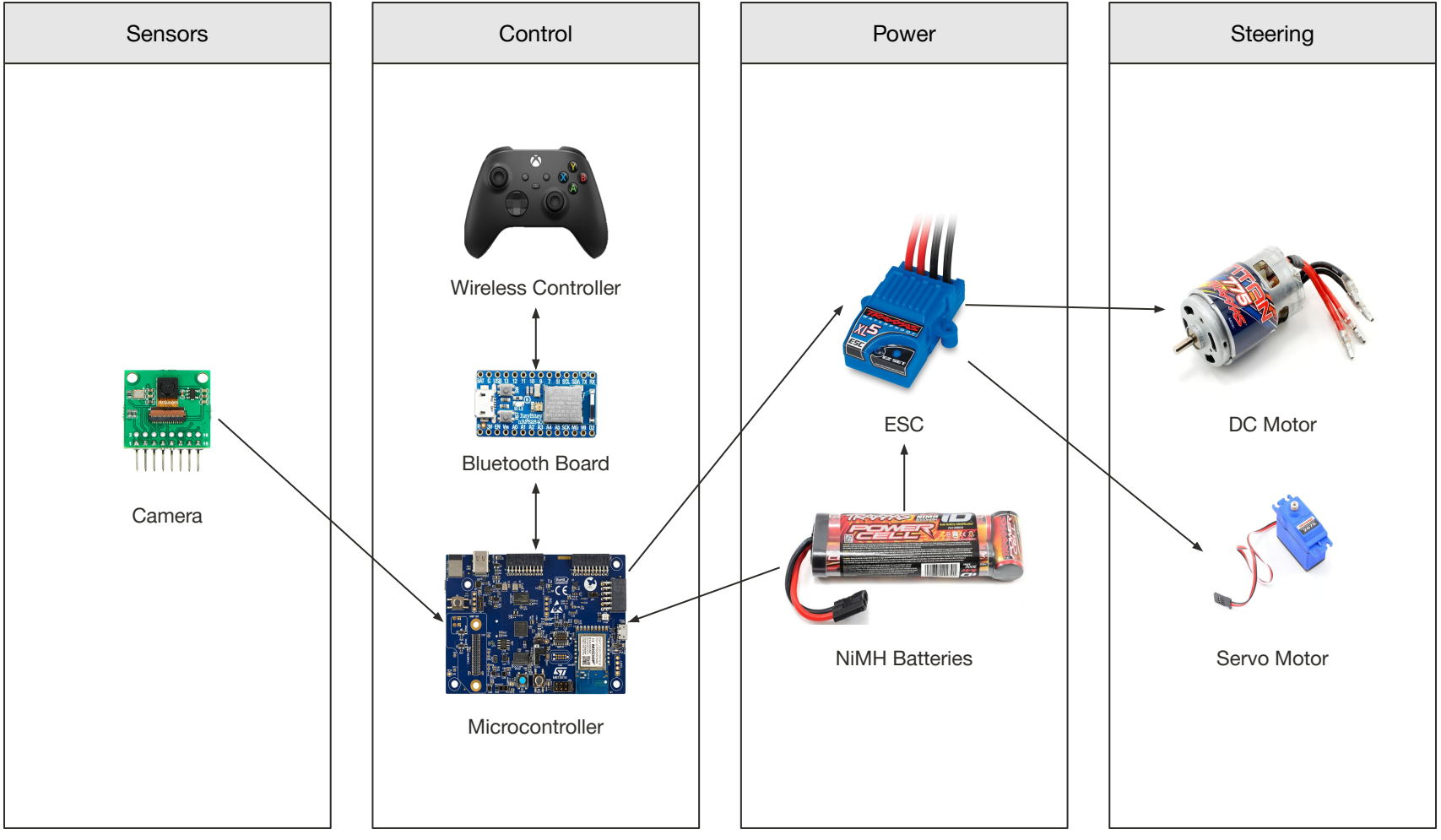
Farmbot - Motivation and Background

- Want a self-sustaining, autonomous robot
  - Operate for a full day without intervention
  - Low power
  - Long battery life
- Begin with optimizing MCU, firmware, and navigation
  - STM32U5
  - Off-the-shelf locomotion

EarthSense

Development Overview

- Locomotion
  - Motor control
  - Bluetooth remote connection
- Data Storage & Communication
  - On-Board Storage
  - Bluetooth
  - Satellite
- Navigation
  - Perception sensors
  - Path planning
- Power
  - Batteries
  - Solar Panel


Sensors
Control
Power
Steering
Wireless Controller
Bluetooth Board
Camera
Microcontroller
ESC
NiMH Batteries
DC Motor
Servo Motor

Stanford University

Locomotion: Servo, ESC, and DC Motor

Servo Motor

- Supports ~60° of rotation when attached to wheels
- 50 Hz PWM control signal

Angle	Direction	Duty Cycle
0°	Right	6.54%
30°	Straight	8.0%
60°	Left	9.45%

ESC and Brushed DC Motor

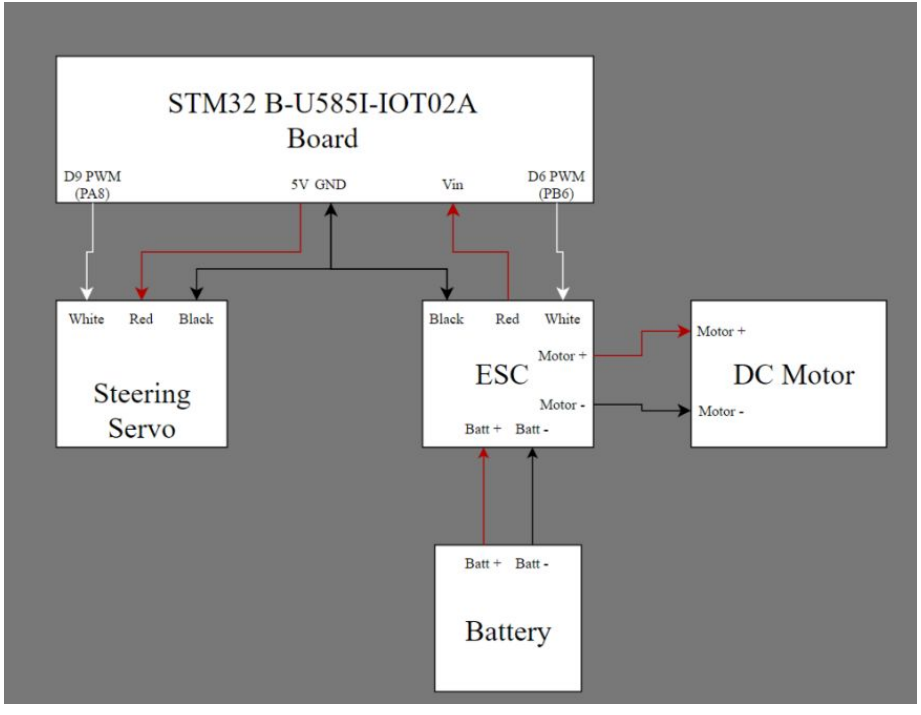
- 50 Hz PWM control signal
- Up to 8.4 V DC voltage

Throttle	Duty Cycle
Full Reverse/Brake	5.0%
Idle	7.5%
Full Forward	10.0%

Power Consumption

Battery Voltage: 8.4 V

	Current (A)	Power (W)
Idle	0.25	2.09
Steering	0.66	5.5
Half Throttle	2	17
Half Throttle + Steering	2.5	20

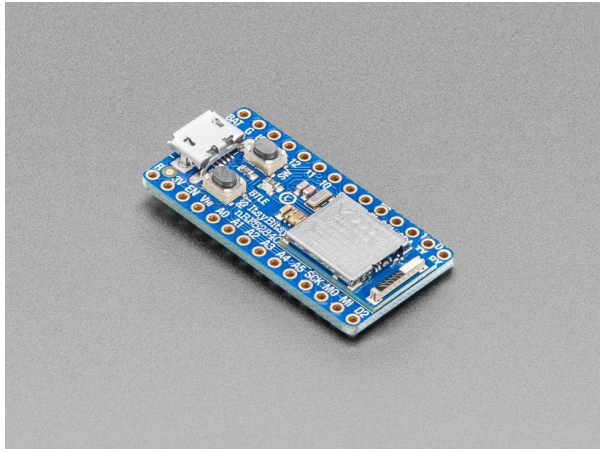


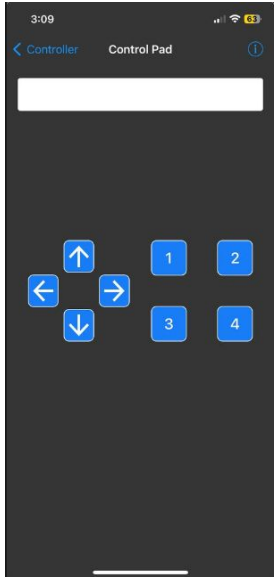
Stanford University

Locomotion: Bluetooth Control with ItsyBitsy nRF52840

First Iteration: Controller App

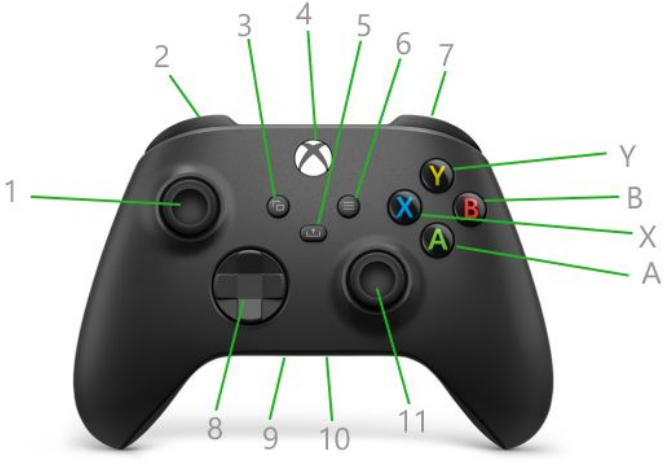
- Example code used with pre-built app
- MCU reads serial buffer using interrupts
- Works, but not ideal:
  - Imprecise steering and throttle
  - Limited functionality





Second Iteration: Xbox Controller

- Uses (tweaked) Bluefruit library
- MCU reads serial buffer…
  - …using interrupts (STM32U5)
  - …using polling (Pico)
- Improved controls:
  - Easier steering with analog sticks
  - Analog triggers for throttle
  - Button mapping

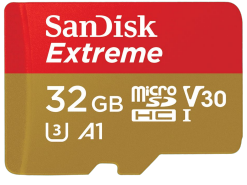


Stanford University

Data Storage & Communication

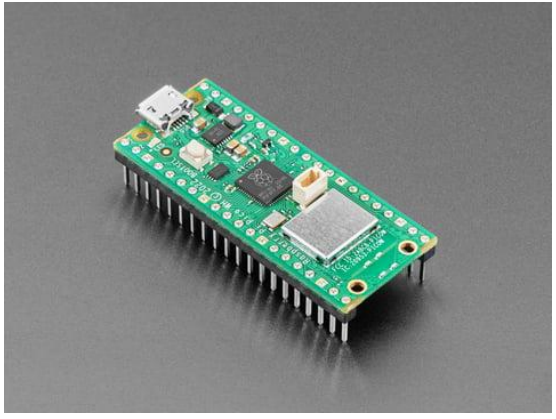
MicroSD Integration

- Pictures taken are stored on microSD card
- Uses FatFS library
- Connected to MCU via SPI:
  - Works with the Pico…
  - …but not with the STM32U5
    - Neither MMC or SPI works
    - Problem with initialization



Going Forward

- For now, we can use Pico for data collection
- Still need microSD integration with STM32:
  - Write our own driver
  - Or use a different board

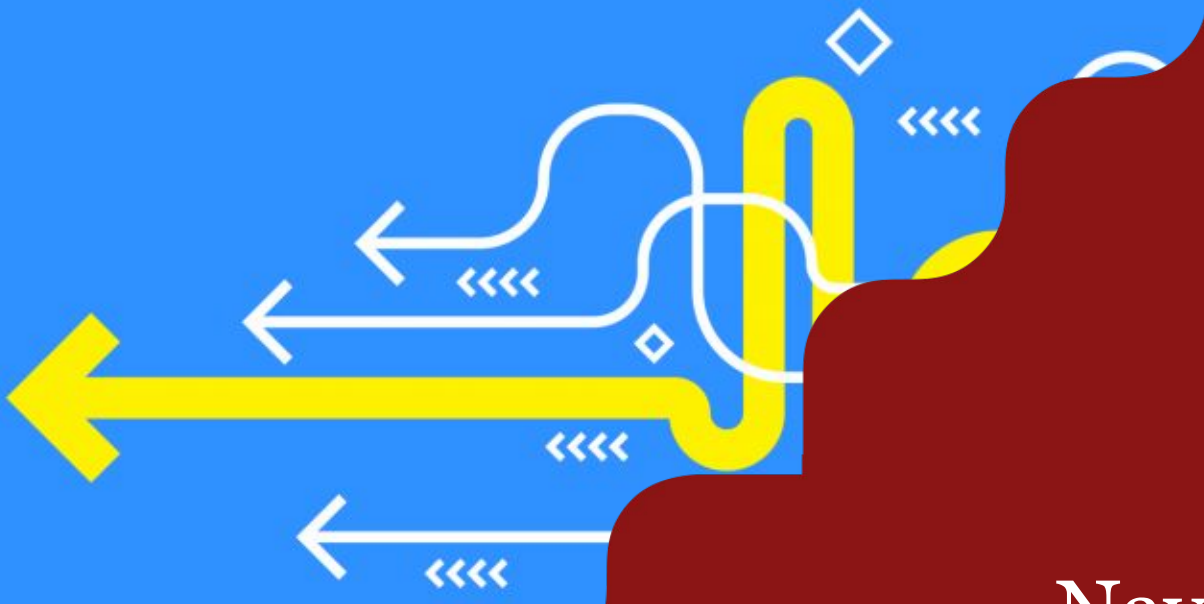


Going Forward

- Establish satellite communication (e.g. Swarm)
- Install ultra-low-power IoT tags that collect farm data (e.g. soil moisture, pH)

Figure 2: GreenTag deployment in a greenhouse, where two commodity RFID tags are attached to each pot.

Stanford University



Navigation

Sensors

- LiDAR
- Camera
- Radar
- GPS
- IMU
- Ultrasonic Sensors

LiDAR

- Able to accurately detect the distance and angle of surrounding objects with respect to heading angle
- Makes it easier for us to analyze the data and identify objects in the path
- Visibility during night-time

- Downside is that it consumes a lot of power (compared to the other sensors)
- Under certain environmental conditions (such as foggy weather), Lidar isn't very accurate (or need to change laser frequency)

Camera

- Can use ML to identify objects
- Low-power options available

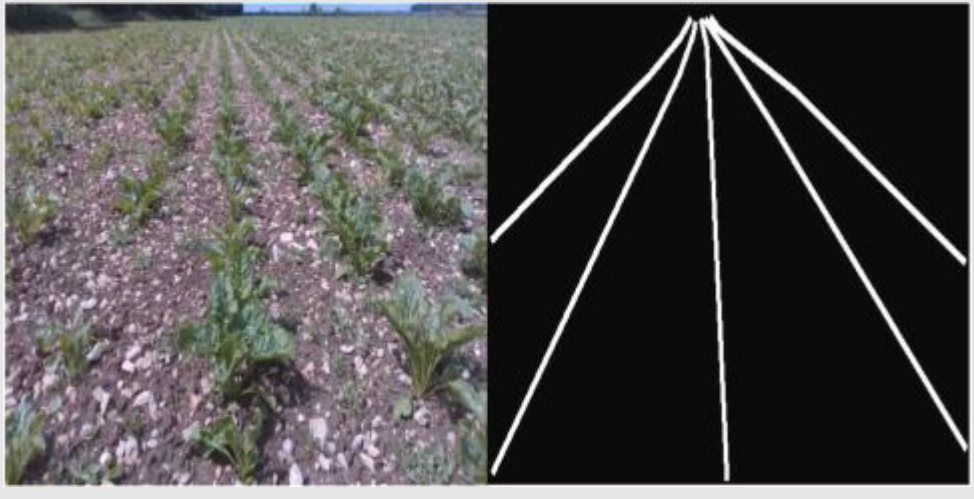
- Downside is that performance heavily depends on the quality of the ML model
- May also need multiple cameras

Stanford University

RPLidar A1M8

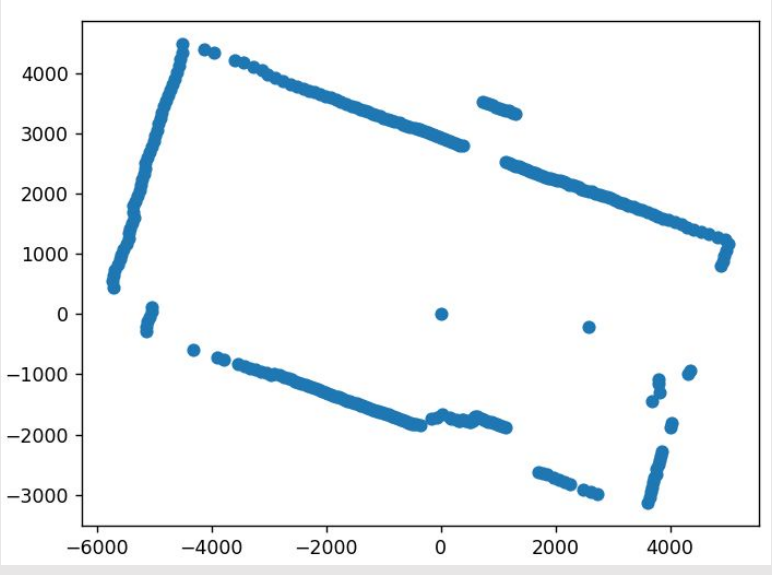
Purpose and Goal

- Use low-power Lidar to map crop field
- Identify crop rows from the data points
- Use crop rows to determine how the RC car will navigate through the field

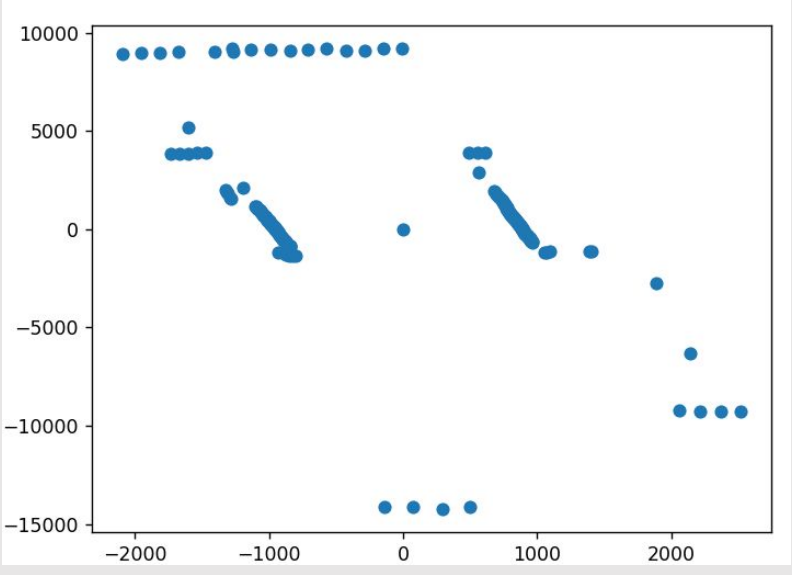


Indoor Performance Test

Inside the Office



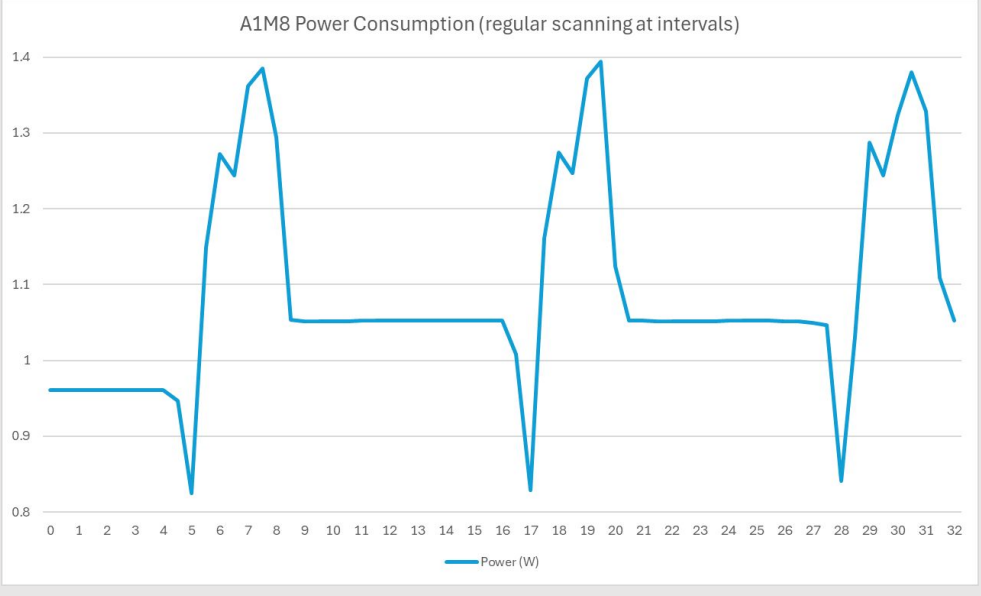
In the Hallway



Power Consumption

Idle: ~1.05W

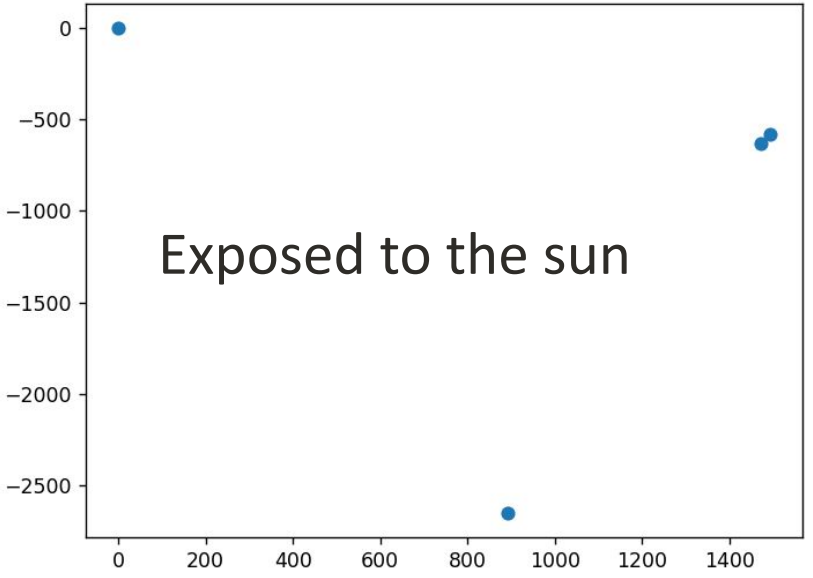
Scanning: Up to ~1.4W

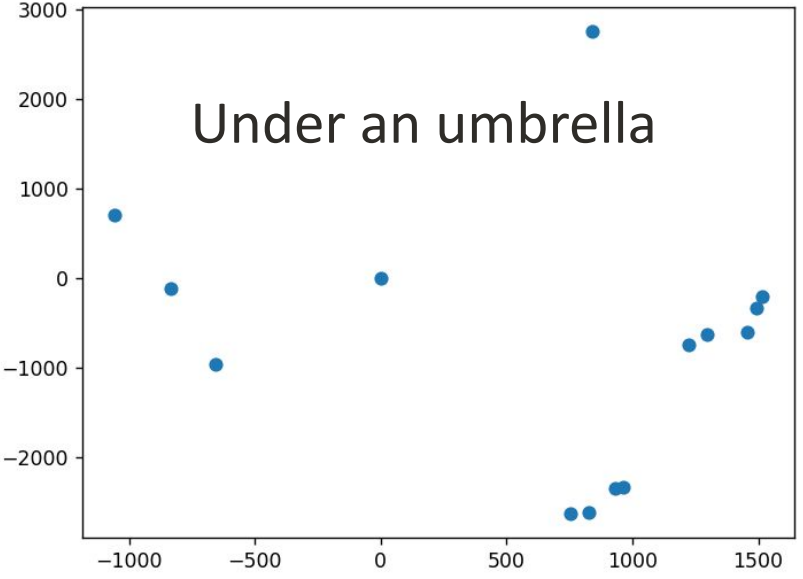


Outdoor Testing

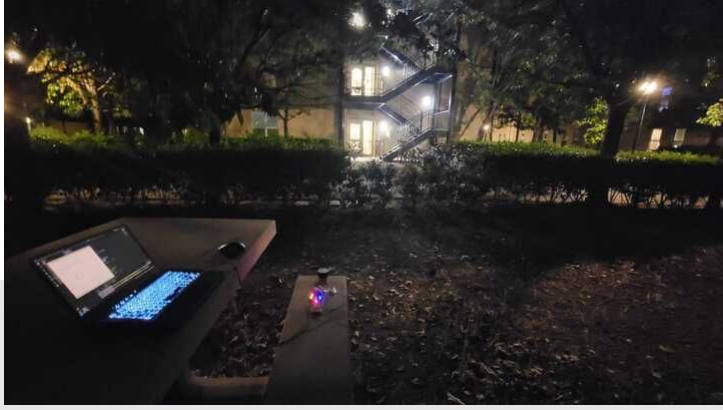
Barely any points were scanned

- Leaves and branches too thin?
- Is the laser wavelength not suitable for plants?
- The LIDAR doesn't work well outdoors?



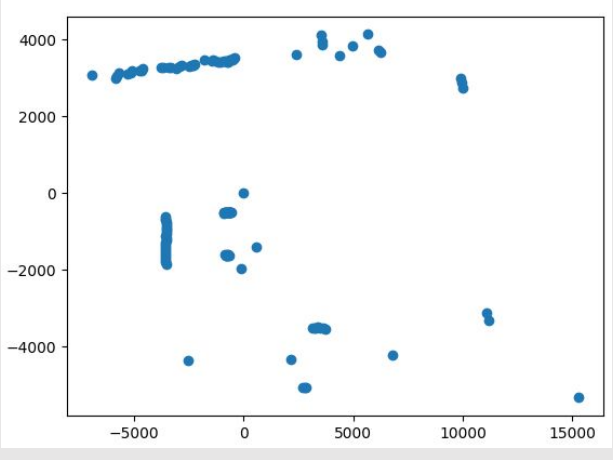


Outdoor Testing (day/night)

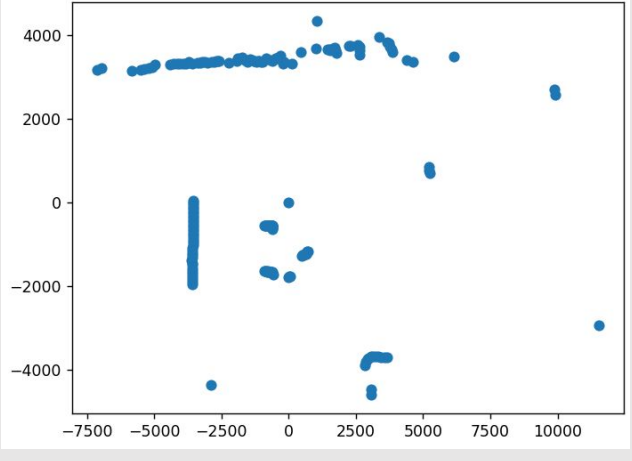


Between studio 3 and studio 4

Day (~11:20am)



Night



Problem Conclusion

RPLidar A1M8 (although not explicitly said on their datasheet) cannot work outdoors under direct sunlight exposure. Thus we had to order a different Lidar!

For Model S2M1-R2 Only

Item	Detail
Application Scenarios	Ideal for both outdoor and indoor environments with reliable resistance to daylight

Datasheet Link

Row-identification algorithm (indoor navigation)

Goal:

- Identify two proper rows
- Determine the heading angle

Algorithm:

- Using DBSCAN, group sets of points and create their lines of best fit
- Calculate a weighted MSE for each line and their points and rank them from least to greatest
- Select two lines that satisfy the following criteria:
  - Both lines must be on different sides of the origin
  - Both lines must have a similar slopes (we want parallel lines)

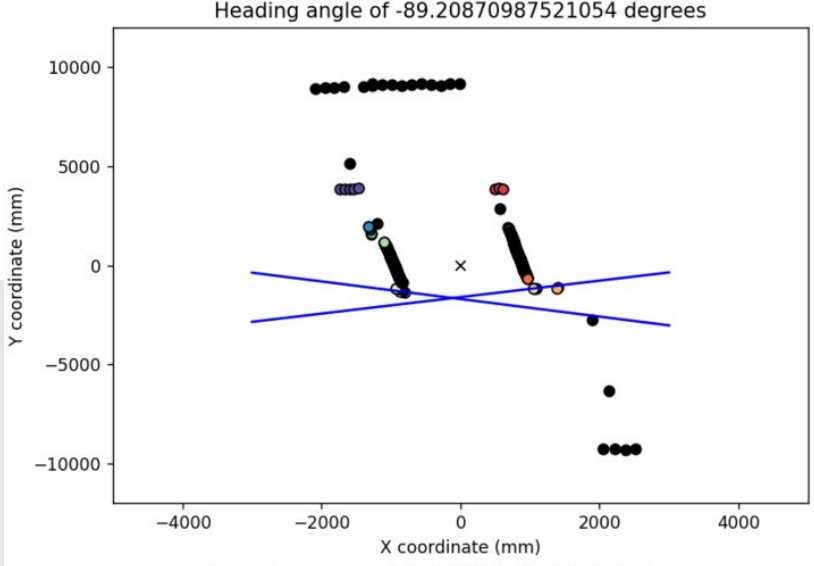
Why a weighted MSE?

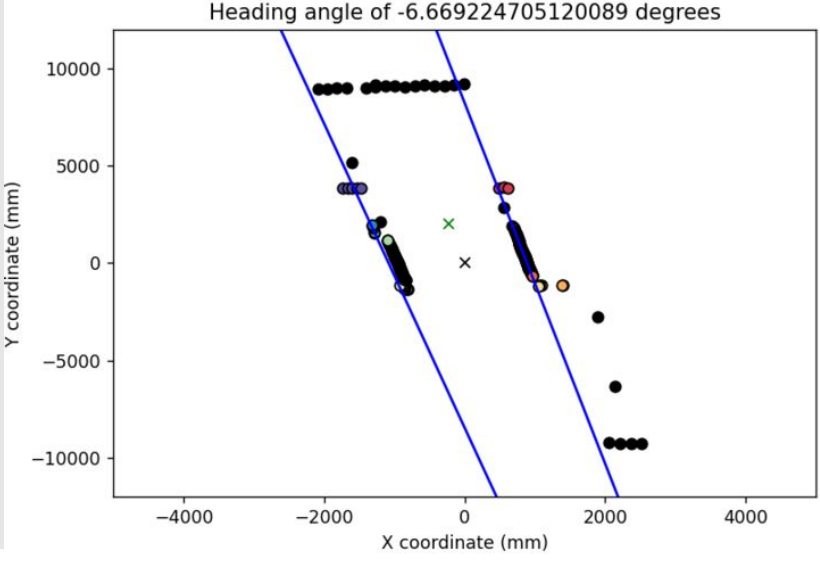
Could have clusters with more points (thus most likely a better representation of the row)

But these will more likely result in greater MSE compared to other smaller clusters that may have a smaller MSE

Weighted MSE is calculated by:

MSE / Num of points in cluster





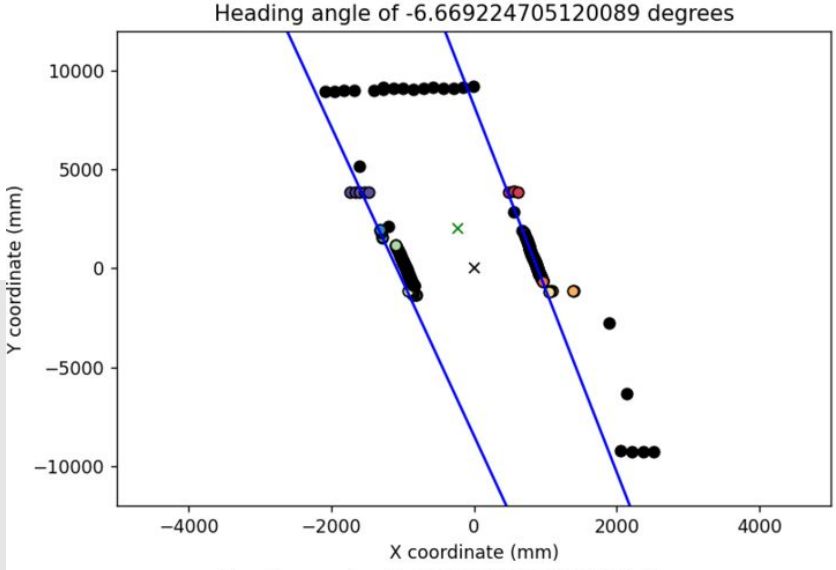
How is the row identified?

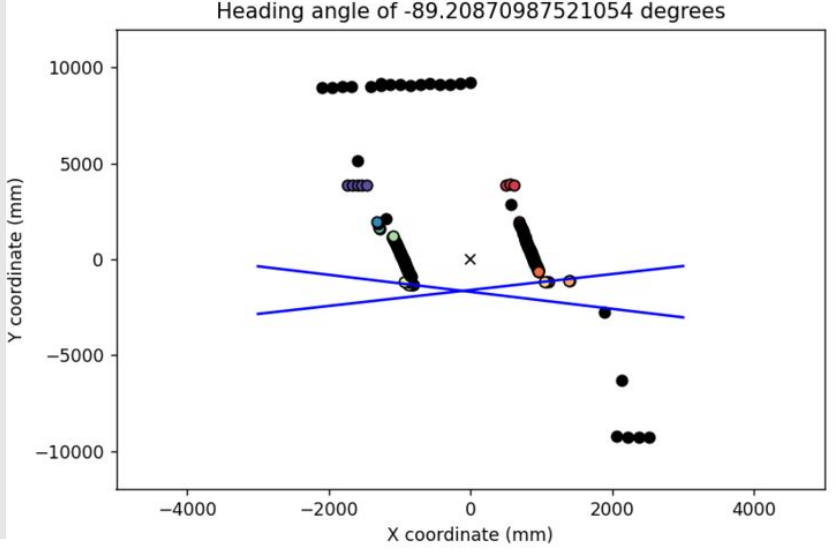
Both lines to not be on the same side:

- Multiplying their x-intercept will result in a negative number

Slope comparison:

- |1 - (|slope1|/|slope2|)| < thresh
- Threshold is pre-set





TerraSentia's Algorithm

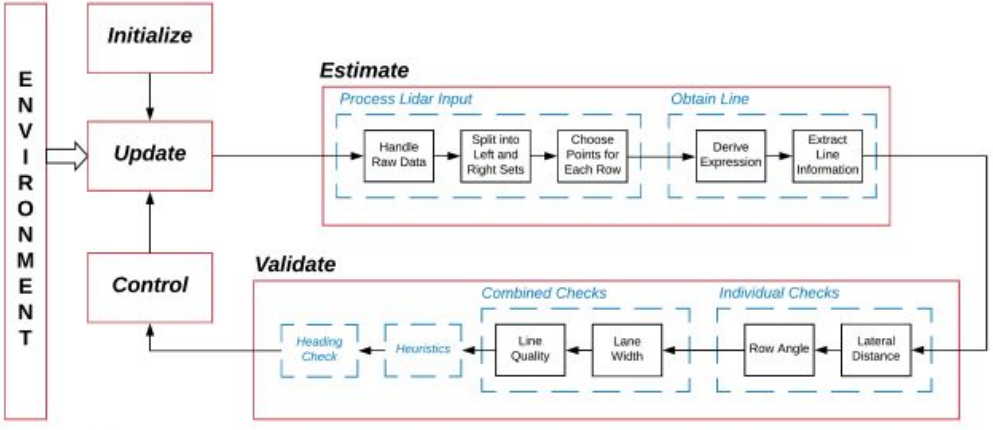


Figure 5: Diagram of embedded robot code. Configuration files are loaded, devices are started and initial values set in the ***Initialize*** step. Sensor readings are refreshed in ***Update***. The ***Estimate*** and the ***Validate*** stages compose the proposed perception subsystem. First, ***Estimate*** processes LiDAR input and obtain the lines corresponding to lateral rows. Then, ***Validate*** ensures correct estimates were obtained. Finally, using the perception outputs, ***Control*** uses a PID controller to generate steering commands.

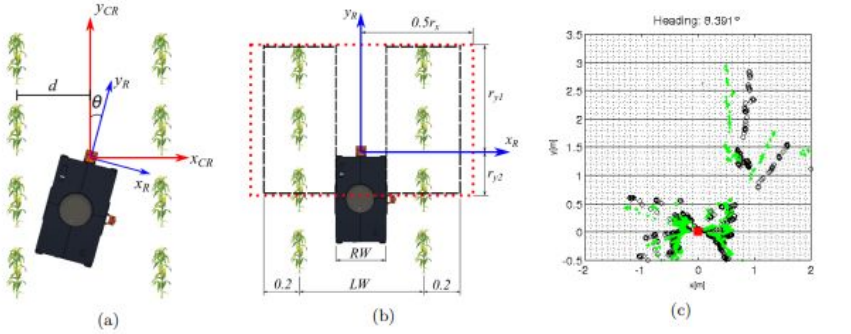
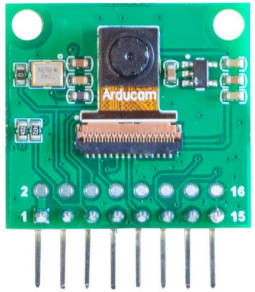


Figure 6: (a) Crop (red) and robot (blue) reference frames. Their origins are fixed at LiDAR, but while robot y-axis (y_R) is aligned with robot's longitudinal axis, crop y-axis (y_{CR}) is parallel to rows. The angle between these two y-axis is defined as heading θ and the lateral distance d is the orthogonal distance between the center of sensor and an adjacent row. (b) Limits for LiDAR readings and robot reference frame. In **Process LiDAR input**, readings outside red dotted rectangle (defined by r_x, r_{y1} and r_{y2}) are filtered out. we define r_x to read only the immediate lateral row. r_{y1} and r_{y2} weights the importance between the readings ahead with those behind the sensor. Later, only readings inside black dashed rectangles are considered as possible distance measurements for each lateral row. LW is the lane width and RW is the robot width (c) Black circles are original LiDAR readings and green dots are rotated ones using estimated robot's heading θ. Red square marks the sensor position.

Will require our PID control to be implemented first so we can have checks for validation, as well as predicting what our current heading angle was before the scan.

ArduCam

Stanford University



Arducam HM01B0

Goal

Use with RPI Pico to take Grasycale pictures

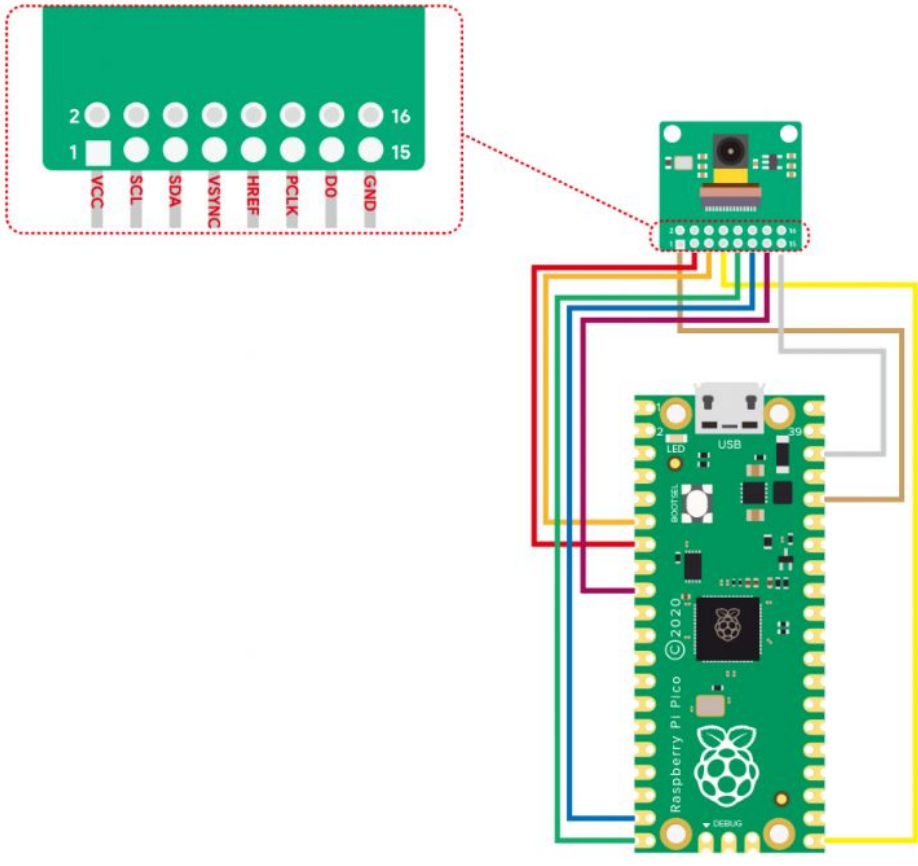
Mount this on RC car and take pictures in the farm to validate Chaeyoung's model (or even train one with our own dataset)

Allow us to navigate through the farm-field using vision-based navigation

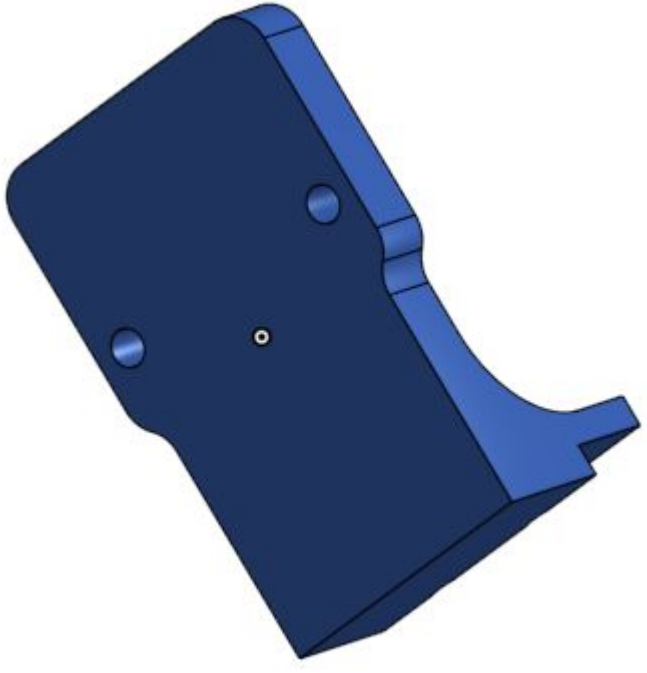
Firmware

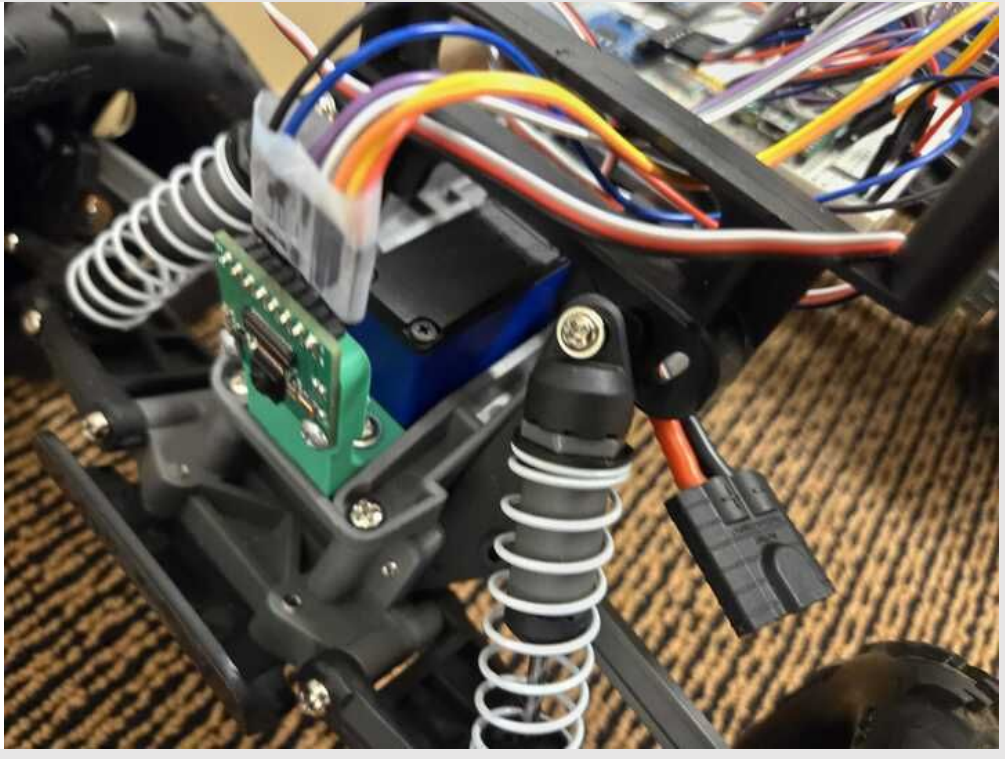
There is already an open-sourced example could out there using RPI-Pico to take pictures with the camera

We quickly used it and was able to successfully take pictures



3D-Printed mounting structure





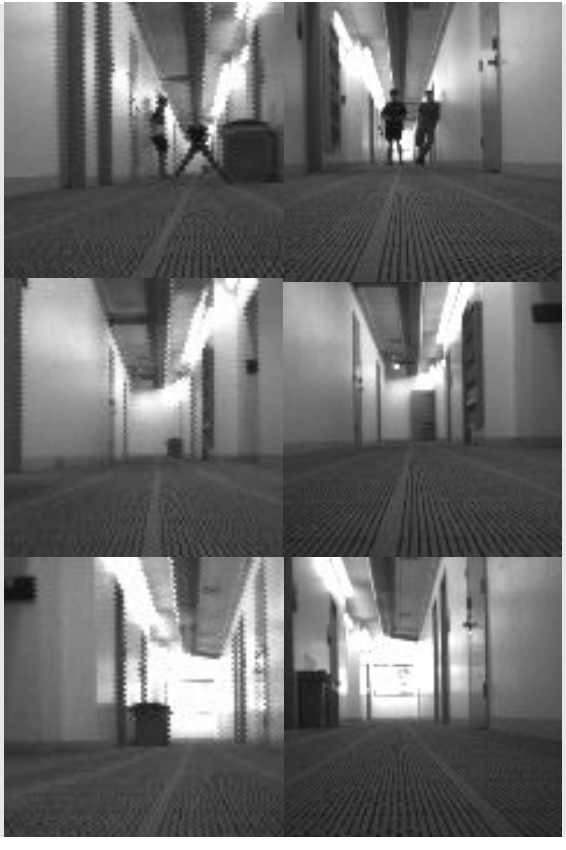
Picture resolution problem

The example code would only take 96x96 resolution pictures, whereas the camera can go up to 320x240 pixels.

The maximum resolution we were able to achieve was 120x120 pixels

We tried looking into the firmware but couldn't really conclude what exactly was causing the issue for us to further increase the resolution without causing the frames to glitch.

Picture resolution problem



Lagging issue

As we took more pictures, it seems like it takes longer for the whole system to process and write to the SD Card

When it is taking a picture, any command sent to the RC Car will not be processed until the writing to the SD Card is finished

This is something that needs to be looked into in the future

Future Work

See if we could increase the picture resolutions to 160x120, or even 320x240 if possible

Try to resolve our lagging issue

Go into the farm and take pictures!

Check whether Chaeyoung's model works with these pictures or not

If we have enough pictures, see if we could annotate them and train a model ourselves!

Thank You