2026 ASABE Student Robotics Challenge
Standard Division
Robot Design Report

Department of Biological and Agricultural Engineering

University of California, Davis

Faculty Advisor: Dr. Vivian Vuong
Post-Doctoral Advisor: Dr. Heesup Yun
Graduate Advisors: Earl Ranario, Willian Klippel Huber, Zhian Li
Undergraduate Members: Kaileen Navas, Madeleine Kim, Nico Puente, Serenity Pico

I. The Need and Benefits to Agriculture

Corn is one of the world's most important crops and plays a crucial role in food production, livestock feed, and fuel. According to the USDA, not only is corn the primary feed grain in the U.S., but it also accounts "for more than 95 percent of total feed grain production and use" [1]. Agriculture is increasingly adopting autonomous technologies to improve efficiency, increase productivity, and ease the physical demands of workers. Modern agriculture relies on achieving successful plant populations to maximize crop yield and does so by stand counting. Stand counting assesses field plant populations, such as corn, to determine if they were successfully planted. Traditionally, stand counting is completed manually through various methods such as the 1/1000th acre method, stick method, or hoop method [2]. These three methods are not necessarily ideal for large crop populations due to their labor-intensive, time-consuming nature and susceptibility to human error.

Autonomous agricultural robots enable automation of tasks to improve consistency, rapidly assess plant populations, and assist with planting decisions. Robots equipped with sensors, cameras, and color recognition systems can allow rapid assessment of corn populations. These technologies allow farmers to make time-efficient decisions to improve productivity. Specifically in this project, the robot's ability to distinguish between field conditions enables more precise decision-making, where healthy plants are preserved, unhealthy ones are removed, and empty spots are located. Technology such as this allows farmers to make faster, informed planting decisions to improve productivity and crop uniformity.

II. Approach and Originality

Our autonomous stand assessment and actuation system was designed to automate the process of evaluating corn stands and performing corrective actions in agricultural environments. The main design is based on its ability to accurately identify the condition of each corn stand, maintain a live stand count, and remove unhealthy yellow plants from double stands without human intervention. The robot integrates navigation, vision, sensing, and actuation into a single system that is able to complete tasks without human intervention.

The system is designed with four primary subsystems: navigation, vision, sensing, and actuation. Each system works independently while communicating through the micro:bit to allow individual components to be tested and developed before being added to the complete system. Our modular design simplifies troubleshooting and allows for future possible upgrades or revisions that would not require major changes to the current overall system.

The system build is designed to operate on the 8 ft x 8 ft competition board following the area's black navigation lines. The system's navigation system allows autonomous movement to specified locations within the given 5-minute time limit. At each stop the Pixy2 vision sensor classifies the stand as either a single plant, double plant, or an empty stand. The micro:bit will then process the results and determine whether actuation commands need to occur. If a yellow

plant is sensed in a double plant stand, the robot will independently remove the unhealthy yellow plant.

III. Definition of Design Objectives and Criteria

- **Objective 1:** Design a robot that can autonomously navigate the competition board while following the designated path. The navigation system should be able to follow straight and curved lines while maintaining position, as well as stop at each planting location.
- **Objective 2:** Develop a computer vision system capable of detecting and recognizing different colors. It should be able to identify the three categories: green healthy plant, empty planting spot, and double plant with 1 healthy green plant and 1 unhealthy yellow plant. The model should provide accurate
- **Objective 3:** Design an effective mechanism to be able to remove the yellow plant from the double stand, without disturbing the healthy green plants.
- **Objective 4:** Create an integrated control algorithm that combines autonomous navigation, plant classification, and plant actuation in an accurate and efficient system.

IV. Parts List & Table

Table 1: Parts list for the robot

#	Component	Quantity	MSRP ($)	Purchase Price ($)	Total ($)
1	Hosyond I2C LCD 1602	1	12.99	12.99	12.99
2	Parallax cyber:bot Robot Kit - with micro:bit	1	299.00	299.00	299.00
3	Parallax QTI Line Follower AppKit for the Small Robot	1	44.95	44.95	44.95
4	Parallax 3x Feedback 360 High Speed Servo	1	27.99	27.99	27.99
5	Paralax Ping))) Ultrasonic Distance Sensor	2	37.95	37.95	75.90
6	Pixy2 CMUcam5	1	129.95	129.95	129.95
				Total MSRP ($):	590.78
				Total Purchase Cost ($):	590.78

V. Hardware Description

A. Overall System

The robot was built using the Parallax cyber:bot platform, which provided a compact, modular base for autonomous navigation, shown in Figure 1. The system combines sensing and actuation components to enable the robot to navigate the competition course, identify corn stand conditions, and perform necessary removals of plants. The primary components include the mobile robot chassis, line followers, vision camera, distance sensors, and servo actuator.

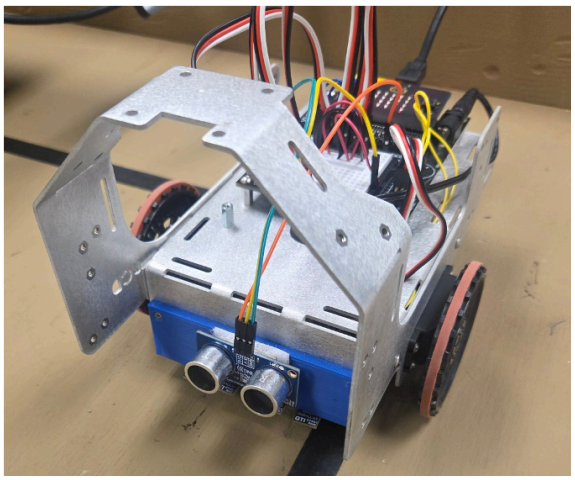

Figure 1: Overview of the robot hardware

B. cyber:bot

The foundation of the robot is the Parallax cyber:bot Robot Kit - with micro:bit. The cyber:bot acts as the primary platform supporting all electrical components. The robot uses a differential drive system where two wheels are powered independently to allow forward movement and turning. By varying the speed of each wheel, the robot can accurately turn and position itself through the course to get to each corn stand.

C. Line Following System

For navigation, we utilized the Parallax QTI Line Follower. This line-following system uses infrared emitters to detect black lines on the competition course from other lighter surfaces. As the robot moves, the QTI sensors will continue to detect the position of the black guideline. The micro:bit processes the information and adjusts the wheel speeds to allow the robot to stay centered on the course path. This system allows the robot to roam between planting locations autonomously.

D. Vision System

The robot uses a Pixy2 CMUcam5 camera to identify the condition and color of each corn stand. The Pixy2 uses a color-based filtering algorithm to detect objects. This allows the robot to distinguish among the three competition scenarios: (1) A single healthy green plant, (2) a double standing of one healthy green plant and one unhealthy yellow plant, and (3) an empty plant location. The camera processes colors in real time to send the detected object information to the

micro:bit microcontroller. Depending on the identified stand condition, the controller decides if no action is required or if a yellow plant should be removed. An example of plant condition detection using the vision system is shown in Figure 2.

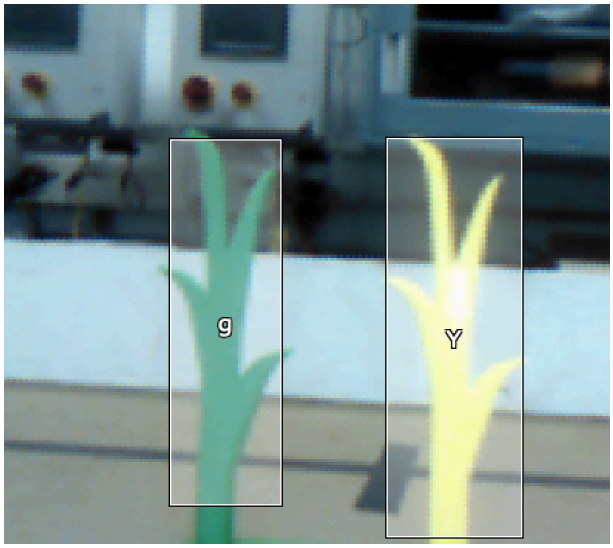

Figure 2: Detection of unhealthy and unhealthy plants using Vision System

E. Distance Sensing

In order to measure the distance between the robot and nearby objects, the Parallax Ping))) Ultrasonic Distance Sensors are mounted onto the robot. The ultrasonic sensors dispense high-frequency sound waves to calculate the distance to an object. Such measurements help the robot maintain a proper distance at each plant location.

F. Actuation System

The robot uses a servo motor connected to a geared mechanism to extend and retract an actuation rod, shown in Figure 3. This rod is used to physically knock down unhealthy yellow plants when removal is required. The robot first uses the vision system to determine whether actuation is needed based on the detected corn stand condition. If a yellow plant is identified, the Pixy2 camera helps determine the plant's position so the robot can align itself for removal.

During the actuation process, the servo motor drives the gear mechanism to extend the rod outward. Once the rod is extended, the robot moves either forward or backward, depending on the plant's location, so that the rod makes contact with the unhealthy plant and knocks it down. After the removal action is completed, the rod can be retracted, allowing the robot to continue navigating to the next plant location.

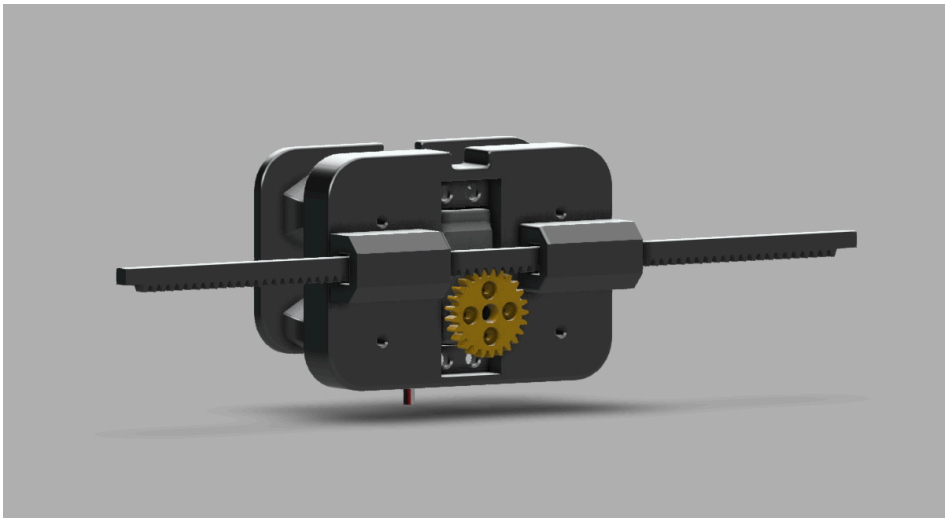

Figure 3: Servo-driven geared mechanism used to extend and retract the actuation rod

VI. Software Developments and Simulation

A. Software Simulation Overview

To develop the robot's control, navigation, and counting tasks, we developed a software simulator based on the Python programming language. Figure 4 shows the overview of the robot simulation program. The Tkinter library was used to build the GUI of the software. The arena environment, robot dimensions, and sensor configurations were adapted from a real robot and environments and tuned to achieve the best simulation results. Based on the updated robot dimensions, sensor configuration, and tuning parameters, the real robot was updated and tested.

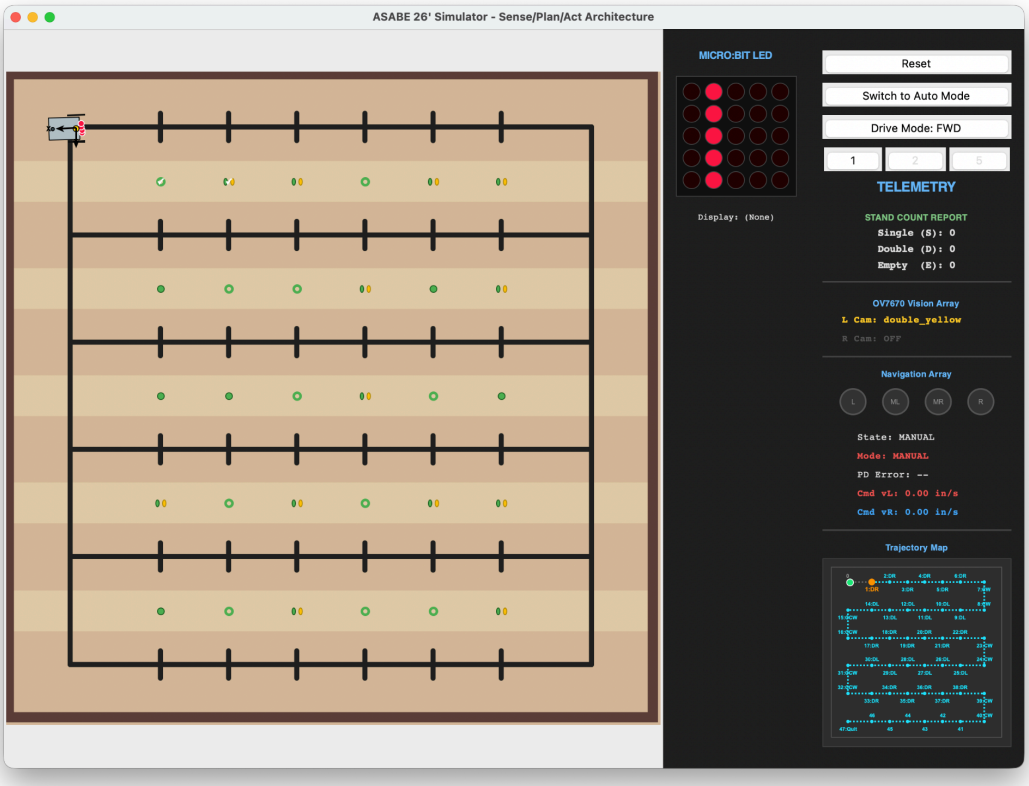

Figure 4: Overview of the robot simulation program.

B. QTI Sensor Processing

The QTI sensor stands for charge (Q), transfer (T), and infrared (I). The QTI line follower detects black lines on the board using a close-proximity infrared emitter and receiver. In the robot software, the QTI readings are processed as binary sensor inputs to determine whether each sensor is detecting the line. Figure 5 shows the QTI sensor placement.

Since the manufacturer's example code assumes a high-contrast black-and-white surface, additional calibration was tested for the competition board, where the black line is placed on a brown surface. The calibrated QTI readings were then used by the line-following controller, waypoint detector, and turning logic.

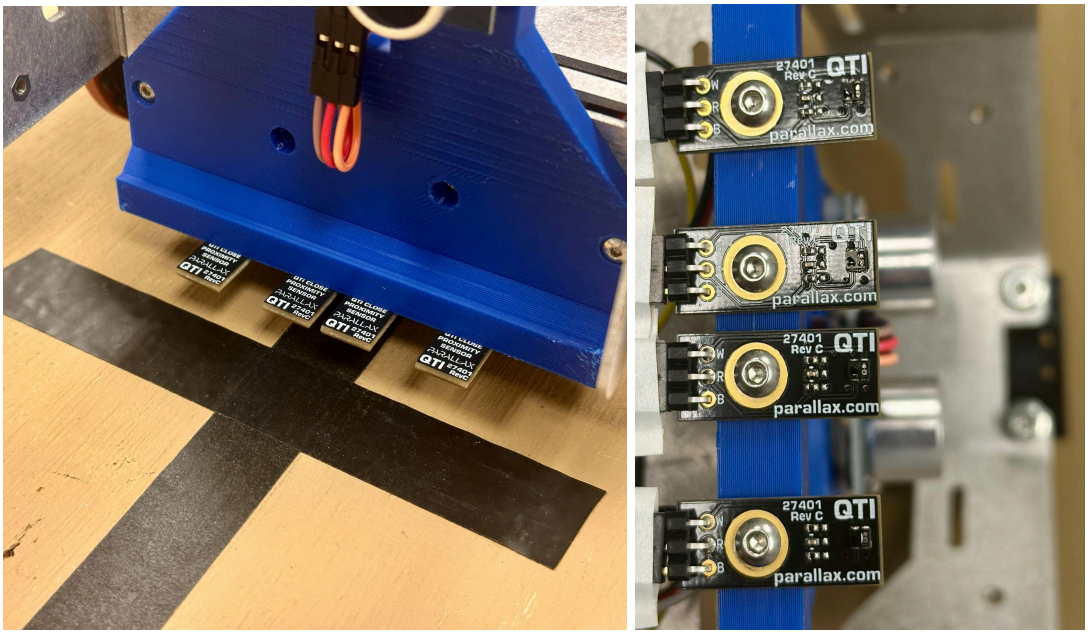

Figure 5: QTI sensor placement

C. EKF for Robot State Estimation

An extended Kalman filter (EKF) was implemented in the robot software to estimate the robot's position and heading during navigation. The EKF combines odometry-based motion prediction with heading updates from the QTI line-following system. This allows the robot to maintain an estimate of its location as it moves between rows, approaches plant locations, and performs turns. The state vector xk at time step k is represented:

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}$$

where:

- xk, yk are the 2D Cartesian coordinates
- θk is the heading angle (in radians)

The state covariance matrix Pk representing the estimation uncertainty is defined as:

$$P_k = \begin{bmatrix} P_{00} & P_{01} & P_{02} \\ P_{01} & P_{11} & P_{12} \\ P_{02} & P_{12} & P_{22} \end{bmatrix}$$

The prediction step propagates the state and covariance matrix using the open-loop odometry model, given the robot wheelbase (W) and the left and right wheel velocities (vL, vR). The linear velocity vk and angular velocity wk are calculated as:

$$v_k = \frac{v_L + v_R}{2}, \quad \omega_k = \frac{v_L - v_R}{W}$$

The predicted state x_k^- at time step k is calculated from the previous state x_{k-1} by:

$$\mathbf{x}_k^- = f(\mathbf{x}_{k-1}, \mathbf{u}_k) = \mathbf{x}_{k-1} + \begin{bmatrix} v_k \cos(\theta_{k-1})\Delta t \\ v_k \sin(\theta_{k-1})\Delta t \\ \omega_k \Delta t \end{bmatrix}$$

where Δt is the control loop time step dt, which is 50ms in our implementation.

The predicted error covariance P_k^- is updated using the Jacobian matrix of the transition function:

$$P_k^- = F_k P_{k-1} F_k^T + Q$$

Fk is the Jacobian matrix calculated at the estimate x_{k-1}:

$$F_k = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & -v_k \sin(\theta_{k-1})\Delta t \\ 0 & 1 & v_k \cos(\theta_{k-1})\Delta t \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & f_{02} \\ 0 & 1 & f_{12} \\ 0 & 0 & 1 \end{bmatrix}$$

Q is the process noise covariance matrix:

$$Q = \begin{bmatrix} Q_{00} & 0 & 0 \\ 0 & Q_{11} & 0 \\ 0 & 0 & q_{22} \end{bmatrix}$$

The heading process noise is manually adjusted during the turn to place more trust in odometry.

The updated elements for P_k^- are:

$$P_{00} \leftarrow \min\left(64.0,\ m_{00} + m_{02} f_{02} + Q_{00}\right)$$

$$P_{01} \leftarrow m_{01} + m_{02} f_{12}$$

$$P_{02} \leftarrow m_{02}$$

$$P_{11} \leftarrow \min\left(64.0,\ m_{11} + m_{12} f_{12} + Q_{11}\right)$$

$$P_{12} \leftarrow m_{12}$$

$$P_{22} \leftarrow \min\left(0.5,\ P_{22} + q_{22}\right)$$

Where P00, P11, and P22 are capped at the maximum covariance values.

When the robot follows the line and the QTI sensors detect that the robot is aligned to the line, EKF updates the heading angle to 0, 90, 180, or -90 degrees, assuming it's measured from the QTI sensors.

The measurement zk represents the measured heading θmeasured:

$$z_k = h(\mathbf{x}_k) = \theta_{\text{measured}}$$

The measurement Jacobian matrix is:

$$H_k = \frac{\partial h}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

The innovation $\tilde{y}_k$ accounts for the angle bounded to -90 and 90

$$\tilde{y}_k = \text{wrap}\left(\theta_{\text{measured}} - \theta_k^-\right)$$

The innovation covariance Sk is computed as:

$$S_k = H_k P_k^- H_k^T + R = P_{22} + R$$

where R is the measurement noise variance.

The Kalman gain vector Kk is:

$$K_k = P_k^- H_k^T S_k^{-1} = \frac{1}{P_{22} + R} \begin{bmatrix} P_{02} \\ P_{12} \\ P_{22} \end{bmatrix} = \begin{bmatrix} K_0 \\ K_1 \\ K_2 \end{bmatrix}$$

Combining robot odometry and heading estimate, the updated state vector xk is:

$$\mathbf{x}_k = \mathbf{x}_k^- + K_k \tilde{y}_k \implies \begin{cases} x_k \leftarrow x_k^- + K_0 \tilde{y}_k \\ y_k \leftarrow y_k^- + K_1 \tilde{y}_k \\ \theta_k \leftarrow \theta_k^- + K_2 \tilde{y}_k \end{cases}$$

The updated covariance matrix Pk is:

$$P_k = (I - K_k H_k) P_k^-$$

When it arrives at the waypoints, we can update the robot's position with known positional information. The robot detects the waypoint arrival by detecting four QTI sensors that are activated at the vertical tick marks for the plant locations. The robot detects the corner waypoints by combining Ping))) sensor distance, EKF position estimates, and QTI readings of the turn directions.

When the robot detects the target coordinates (Xwp, Ywp), the estimated robot position is

$$x_{\text{est}} \leftarrow X_{\text{wp}} - d_{\text{sensor}} \cos(\theta_{\text{est}})$$

$$y_{\text{est}} \leftarrow Y_{\text{wp}} - d_{\text{sensor}} \sin(\theta_{\text{est}})$$

Where d_{sensor} is the sensor distance from the robot center.

Since the absolute position is known from the map with high precision, the uncertainty covariance matrix is reset to small values:

$$P = \begin{bmatrix} 0.001 & 0.0 & 0.0 \\ 0.0 & 0.001 & 0.0 \\ 0.0 & 0.0 & 0.001 \end{bmatrix}$$

D. Line-Following Control (PD controller)

During straight-line navigation, the robot uses a PD controller to stay centered on the black guideline. The line-tracking error is calculated from the four QTI sensor readings using weighted sensor positions. The proportional term corrects the robot

based on the current line error, while the derivative term helps reduce overshoot and oscillation. The sensor detections are multiplied with spatial weights w = [-3 -1 1 3].

Let $s_{i,k} \in \{0, 1\}$ be the binary state of QTI sensor $i \in \{0, 1, 2, 3\}$ at step k. The error e_k is the weighted average of all QTI sensor binaries:

$$e_k = \frac{\sum_{i=0}^{3} s_{i,k} \cdot w_i}{\sum_{i=0}^{3} s_{i,k}}$$

The derivation of the error ek is defined:

$$\Delta e_k = \frac{e_k - e_{k-1}}{\Delta t}$$

where Δt is the control loop time step dt=50ms.
Since the raw derivative of error is too sensitive to the noise, a first-order numerical low-pass filter was implemented:

$$d_k = \alpha \cdot d_{k-1} + (1 - \alpha) \cdot \Delta e_k$$

where $\alpha \in [0, 1]$ is the smoothing factor. The cut-off frequency is defined as:

$$f_c \approx \frac{f_s(1-\alpha)}{2\pi\alpha}$$

where f_s is the sampling frequency (1/50ms= 20hz).

The PD controller uses proportional gain K_p and derivative gain K_d to calculate the output based on the error and filtered error derivatives:

$$u_k = K_p e_k + K_d d_k$$

To make u_k within the robot control input range and to prevent excessive steering commands that make the robot overshoot, the steering correction is saturated at the 80% of the maximum wheel velocity.

$$u_k^{\text{sat}} = \text{clamp}\,(u_k,\, -u_{\max},\, u_{\max})$$

where $u_{max} = 0.8\ v_{max}$.

The final robot wheel velocity is mixed with the base speed v_{follow} to differentiate the right and left wheel velocities (v_L, v_R).

$$v_L = v_{\text{follow}} + u_k^{\text{sat}}$$

$$v_R = v_{\text{follow}} - u_k^{\text{sat}}$$

E. Software Robustness Testing

To make the simulation close to a real-world situation, random noise to the sensor, motor bias, and external turbulence were added. For the QTI sensor, sensor positional jitter was added to simulate the robot's random motion from the area surface. Also, signal bit-flips were added to simulate the 2% false readings of the sensors.

To simulate real-world servo motor performance difference bias, 2% of static motor bias from the left and the right side of the motor was simulated. To simulate the signal and servo speed noise, the Gaussian noise was added to the wheel velocity.

To simulate the external turbulence and robot slip, Gaussian noise was added to the robot's x and y coordinates and heading value.

F. Waypoint-based Navigation

The robot's mission path is controlled using a waypoint-based navigation algorithm. Waypoints are generated based on the arena dimensions and the locations of the plant rows. At each waypoint, the software assigns a specific command, such as detect right (DR), detect left (DL), turn right (CW), turn left (CCW), or quit (Q). Figure 6 shows the defined mission waypoints.

The robot adjusts its behavior based on the waypoint command. For plant detection waypoints, the robot slows down, stops, and activates the vision system. For turning waypoints, the robot slows before reaching the corner to reduce overshoot and improve turning accuracy. This waypoint-based structure makes the mission sequence easier to organize, test, and modify.

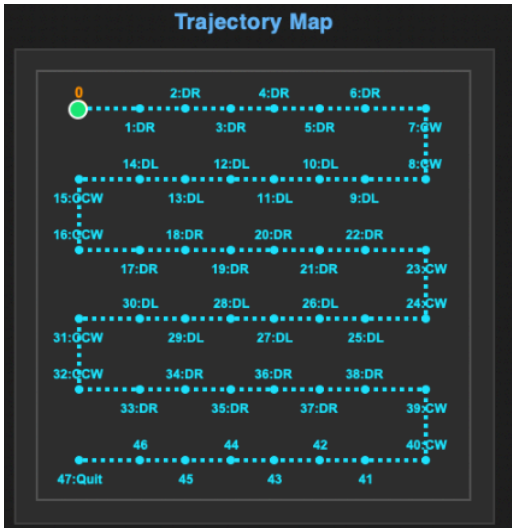

Figure 6: Visualization of the defined waypoints.

G. Detection and Actuation Logic

The detection task workflow is shown in Figure 7. The task begins when the robot follows the line and navigates to the closest corn stand. Once the robot detects that it has reached a corn stand position, it stops and captures an image using the Pixy2 vision camera. The captured image is classified using a thresholding method based on the detected plant color, allowing the system to determine whether the stand contains a single green plant, an empty stand, or a double stand with both a green and yellow plant. If a single green plant or an empty stand is detected, no actuation is required, and the robot continues to the next stand. If a double stand with an unhealthy yellow plant is detected, the robot activates the removal mechanism to knock down the yellow plant.

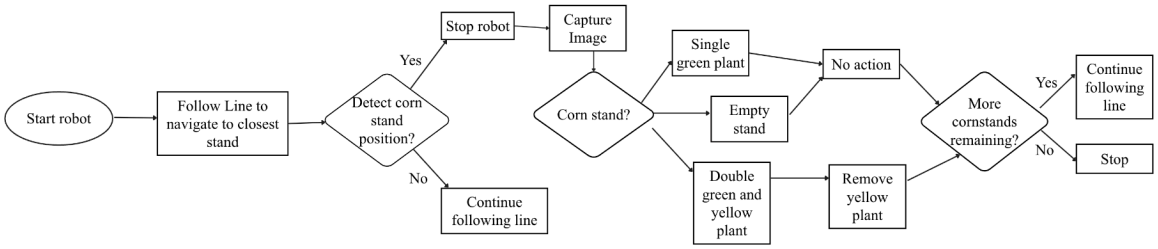

Figure 7: Detection and actuation algorithm

H. Simulation Results

Because of the sensor nose and turbulence, the robot sometimes failed line following and turning, as shown in Figure 8 (a)-(d). Based on the multiple iterations and tuning,

our robot simulation managed to get 100% success rate for the various noise random seed scenarios.

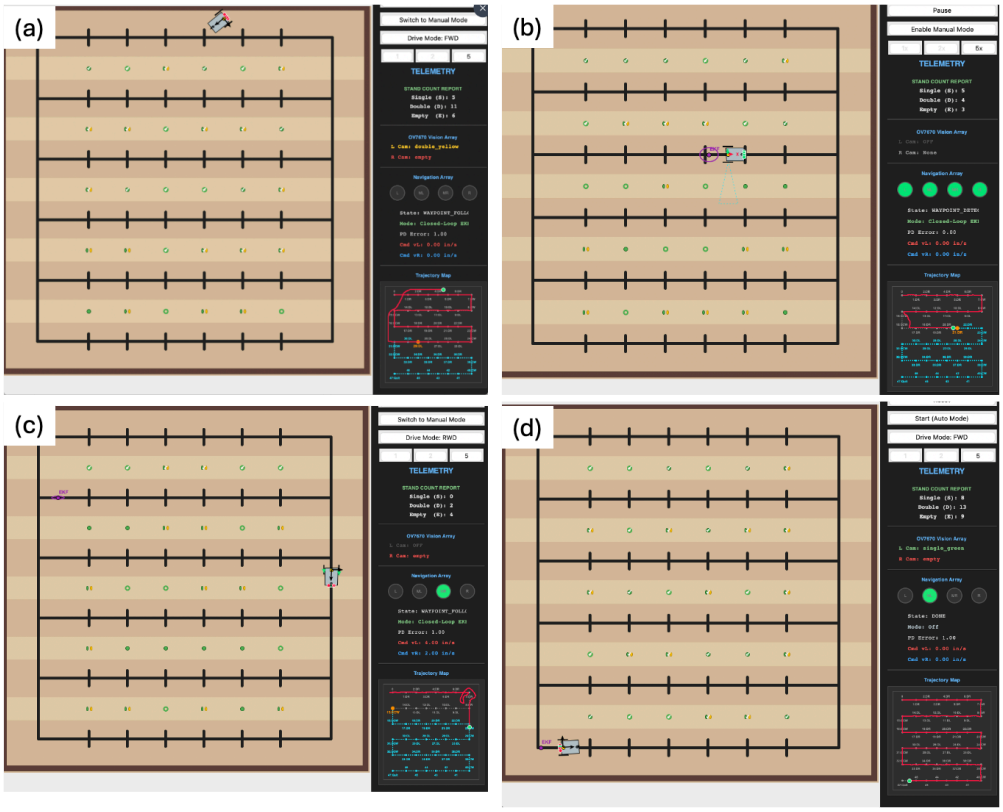

Figure 8: Various failure cases during the robot driving. (a) the robot failed to turn at the third row of the area (b) the robot overshoots turning when entering the third row of the area, making EKF position estimate wrong. (c) the robot fails the first turn of the first row, but EKF position estimates keep going forward even if the robot went in the other direction. (d) the robot finished the missions but stopped early before the final waypoint.

Figure 9 shows the end screen of the simulation program. As shown in Figure 9, the robot arrived and stopped at the final waypoint, detected 7 single plant locations, 12 double plant locations, and 11 empty plant locations. In the simulation, the counted plant locations were displayed in the GUI and in the 5x5 LED matrices in the real robot code, with scrolling actions.
Since robot autonomous driving is heavily affected by random sensor noise, we ran the simulation 100 times, calculated the mission success rate, and averaged the navigation task. Table 2 shows the navigation task result summaries.

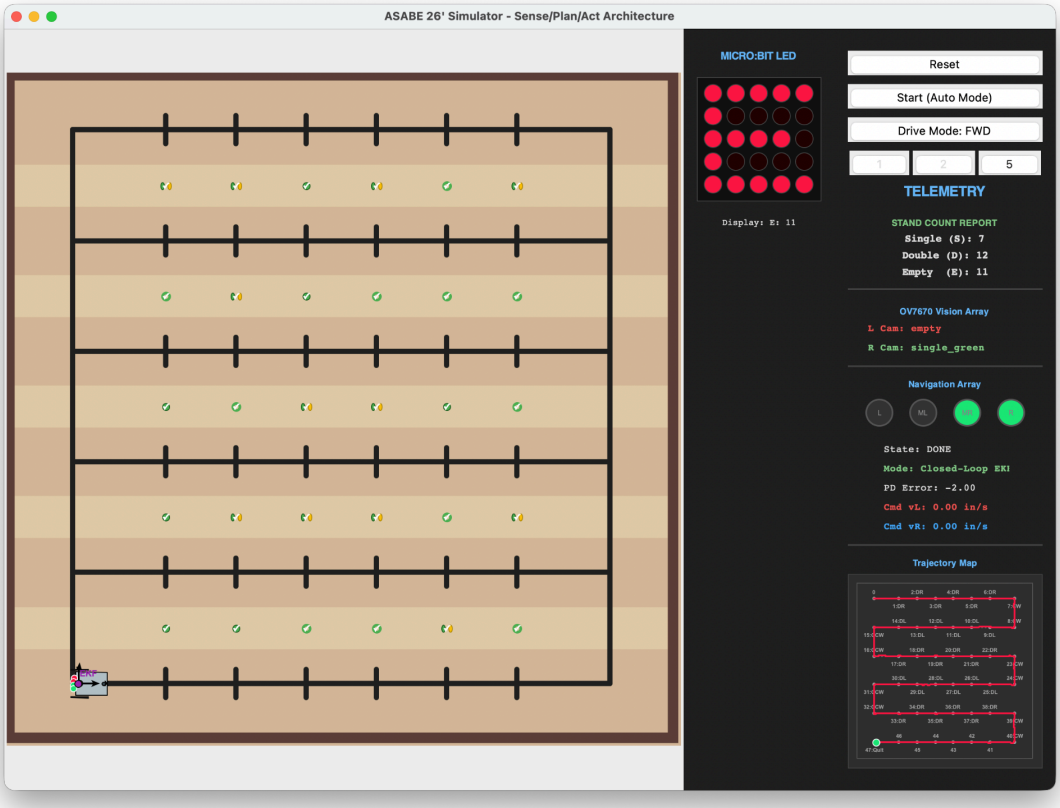

Figure 9: Visualization of simulation results. The GUI simulated the micro:bit's 5x5 LED matrix to show the final detection results

Table 2: Navigation task simulation results

`random.seed(seed) (N = 100)`	2000 - 2099
Navigation Success (Success / Total)	100 / 100
Completed Times (seconds)	min=140.60s, avg= 145.75s, max=165.35s
Max Path Error (inches)	min=3.69 in, avg=6.10 in, max=8.00 in
Average $NavigationScore = \frac{DistanceTraveled^2}{ElapsedTime}$	2054.42

VII. Achievement of Objectives

The robot design achieved the main objectives by integrating autonomous navigation, plant-condition detection, distance sensing, and selective actuation into one complete system. Using the cyber platform, QTI line followers, ultrasonic sensors, Pixy2 vision camera, and servo-driven actuation rod, the robot was designed to follow the competition path, identify healthy plants, unhealthy yellow plants, double stands, and empty locations, and remove unhealthy plants when necessary. Figure 9 shows one successful robot run, including the starting position, navigation through the third row, and arrival at the final waypoint. Simulation results also demonstrated strong navigation performance, with the robot completing 100 out of 100 randomized trials successfully.

Future improvements could focus on increasing the robustness of the vision and actuation systems. The Pixy2 camera could be further tested under different lighting conditions and plant positions to improve detection accuracy, while the actuation mechanism could be refined to improve removal precision and avoid disturbing healthy plants. Additional sensors, wheel encoders, and more physical testing could also improve turning accuracy, position tracking, and overall reliability in real-world agricultural environments.

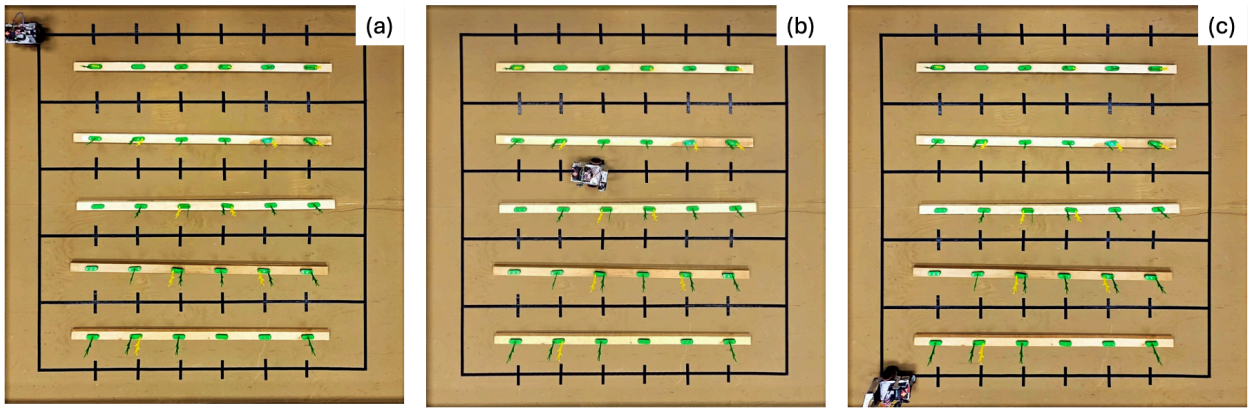

Figure 9: Top-view of the area and the robot. (a) the robot is at the starting position. (b) the robot is driving in the third row of the field. (c) the robot arrived at the final waypoint of the field.

References

[1] Bond, Jennifer K, et al. “Corn and Other Feed Grains - Feed Grains Sector at a Glance | Economic Research Service.” *USDA Economic Research Service*, 7 May 2026, www.ers.usda.gov/topics/crops/corn-and-other-feed-grains/feed-grains-sector-at-a-glance.

[2] Ahmad, Muhammad, et al. “Timely Field Stand Counts for Corn, Soybean and Sorghum |.” *CropWatch*, University of Nebraska, 4 June 2026, cropwatch.unl.edu/timely-field-stand-counts-corn-soybean-and-sorghum/.