MIAMI UNIVERSITY

Robot Final Project - Ece 314

Joystick Controlled Plant Spraying Robot

Spring 2024

College of Engineering and Computing

Department of Electrical and Computer Engineering

ECE 314: Elements of Robotics

Professor - Veena Chidurala, PhD

Professor – James Leonard

Frederick Levins

Parker Murphy

5/10/2024

Description of Project and Goal

Our project entailed successfully planning, building, implementing, and programming a theoretical plant spraying robot with drive speeds and turning controlled by a joystick. The project's goal is to create an interactive and interesting robot application to fulfill the requirements of the final project for ECE 314 Elements of Robotics.

Strategy and Implementation

Components Involved (not including already built robot components from Labs 1-5):

- Raspberry PI 5
- Two Arduino Unos
- Two servo motors
- Sprayer (Nozzle, Tubing, and Pump)
- Various wires
- Joystick
- IR transmitter/receiver
- Resistors
- 9V battery
- Raspberry PI Power Supply

Strategy:
Use serial communication between the raspberry pi and arduino to enable image detection, coordinate location of image, and movement of servo of sprayer.

Implementation:

- Serial connection between arduino and pi
- Duct tape on top of other components to fit structure
- Tube to and from sprayer into cup of water to enable spraying function
- Camera implemented on front and higher than rest of components for clear view
- Sprayer at utmost front to not hit any electrical components
- LCD portrays static wording with name of project and our intials

Difficulty and Procedural Adjustments

(1) Voltage providing to servo motor

- The servos weren't receiving enough voltage to properly operate from the arduino itself. This led to continual crashing of the python script. The error portrayed was "termios.error: (5, 'Input/output error')". We solved this by direct providing voltage to the servo from an external battery source.

(2) Working with libraries on PI

- We initially struggled to get various packages installed onto the pi. We tried to use increased permissions and different OS set ups. Eventually we came across a post that explained we were simply missing -python3 at the end of every installation. This solved the problem.

(3) Weight of robot

- The weight of the robot drastically increased which threw off self driving and encoder usage capabilities. We solved this by manually setting speeds based on the type of movement required within the driving code.

(4) Sprayer

- The sprayer had issues with current wiring and needed some sort of water tank system. We solved the wiring difficulties by resoldering new grounds and voltage wires. The water tank was solved by putting the end of a tube into a cup that would be moved with the robot.

(5) Boundary Box Over Creation

- The initial set up of over image processing led to various boundary boxes being created constantly and all over an image. We solved this through using temp variables and a check for the area of boxes. The temp variables allowed for the pi to recognize boxes being created in the same location as the previous box, eliminating over creation in likewise positions. The area check allowed for eliminating tiny boxes being created constantly all over the object.

Code Explanation (See attached files for scripts and comments)

MainRobotCode.ino

This is the main code for control of motor power and driving functionality. It is degraded from the last lab to early labs to enable manual drive and motor control, rather than self driving with encoders. This is due to the inability of the encoders to proper function, combined with the absurd weight addition of the new components. It is in Arduino C.

On the car side of things we decided it would be best to not use the encoders. There were too many different cases the car could be in and we really wanted the car to react to how the joystick was pressed. The issue with this is that even with the same PWM value set to each motor the car has a hard time going straight. To remedy this we simply went through each of our three speeds for going straight, slow, regular, and fast,

and manually adjusted the PWM values for each side so that the car would drive mostly straight. After that any issues with moving straight can be adjusted with a simple movement of the joystick.

Remote.ino

This is the code that allows for the remote and joystick control of the robot. It is in Arduino C.

We wanted to be able to control the movement of the robot via an infrared transmitter and receiver, but did not think that the standard IR remote was a good fit. So, we decided to build a simple controller with a joystick that could move the robot in whichever direction you moved it and at faster speeds the further you moved it from the center. To achieve this a simple joystick module was wired to an arduino's analog pins so that the x and y positions of the joystick could be easily tracked. This arduino was powered with a 9V battery and a barrel jack connector and was also connected to an IR transmitter via a breadboard. This gave us a very simple controller that could transmit different IR signals based off of the different x and y values. For example when the joystick is not touched a signal that tells the car to stop is sent, and when it is pushed all the way forward it sends a signal to tell the car to move forward quickly. Overall there ended up being 48 different cases that could be transmitted to the robot car.All with varying speeds and directions.

Sprayer.ino

This is the code that communicates with the raspberry pi and receives coordinates. It then converts the coordinates to readable positions for the servo motor to enable movement of the sprayer. It is in Arduino C.

sprayerG.py

This is the code that drives the image processing of the robot. It uses opencv RGB library to create boundary boxes around green objects. It then serially communicates with the arduino and sends these coordinates separated by an X and Y. See figures One and Two. It is in python.

See videos for interoperability of code.

Future Possible Steps and Goals

To improve the project and functionally of the project one would improve the image sensing algorithm, implement and support self driving, a true water tank system, cleaning up of system build and components, and integrate a true machine learning data set that could actually detect plants. The image sensing could be improved by a

more intense algorithm that fixes shades and disregards various image noise. Self driving would need improved arduino code and integration of encoders to control motor speed changes. The robot currently only sees plants theoretically, improving the algorithm to look for specific plants would be much more useful in production. A true water tank system would make it more useful, applicable, and autonomous. This could be as simple as a 3D printed box that fits weight and location requirements. For fixing the structure, this is obvious duct tape and grit are not true techniques for a proper robot. This was done simply due to time constraints, things have to be improvised for demos. For component simplification, the circuits could be optimized, and we also strongly believe the Raspberry PI5 is capable of completely running all the code itself, removing the need for more microcontrollers and serial connection between them.

Images and Videos

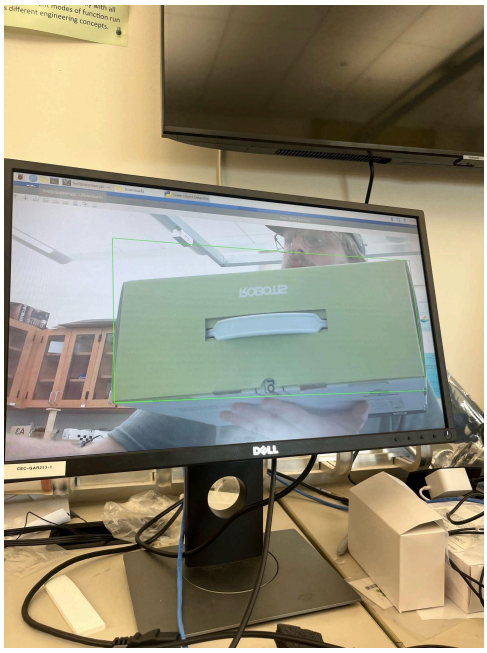

Figure One: Live feed of camera with drawn boundary box around green image, detected

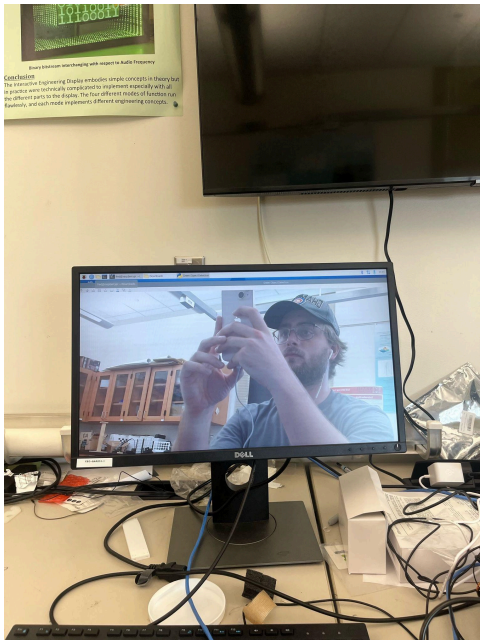

Figure Two: Live feed of camera with no boundary boxes, no image detected

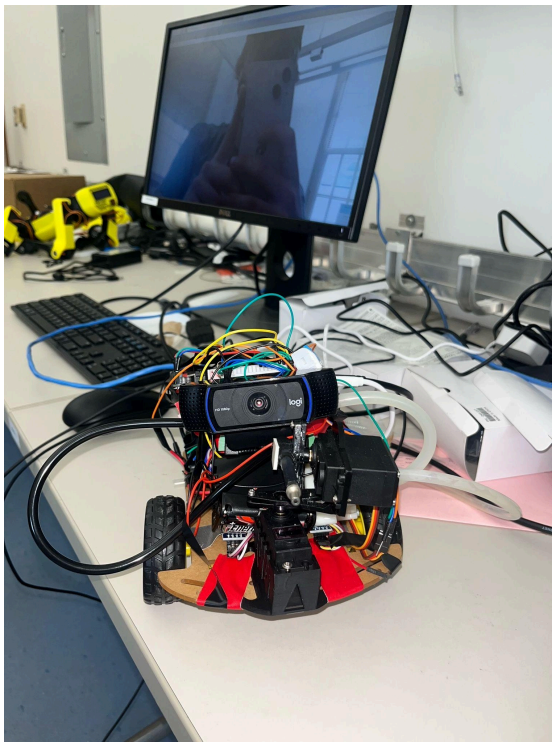

Figure Three: Front image of robot

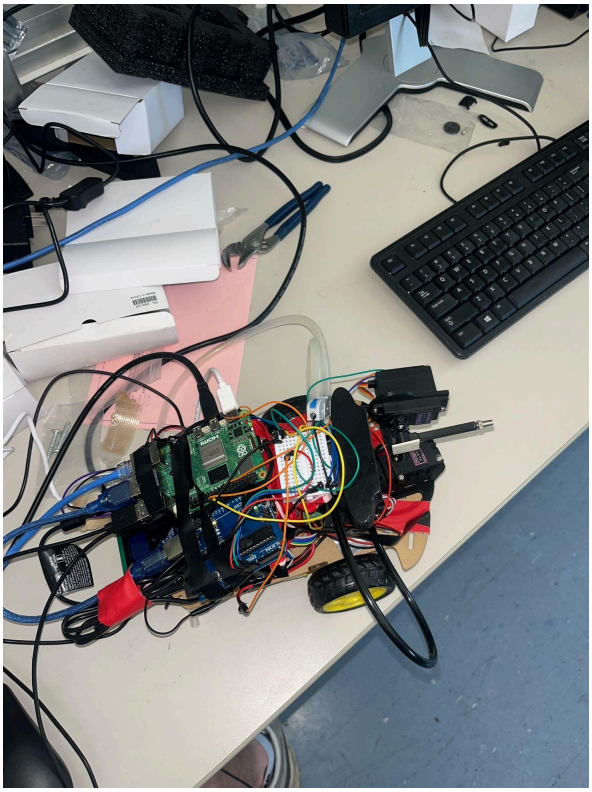

Figure Four: Top view of board

https://youtube.com/shorts/KVK7w-ZC1uE?feature=share
Figure Five: Video of Robot tracking the movement of a green image, this shows the way the sprayer connects to the image processing
https://youtube.com/shorts/-qLuGfuuEyc?si=rMoXRXaishPhtsQi
Figure Six: Video of the live demo showing the robot only triggering and detecting green images, as well as the controls of the joystick
https://youtube.com/shorts/YTvkW8MgSLI?si=RCR4vUJPCwdWhDzq
Figure Seven: Live video of the sprayer spraying water when it detects a green image