Delaware Technical Community College

ITN 290 – IT Capstone

(Fall 2020)

Autonomous Vehicle Project

By:

Domenico Zaffora

Bethany Cacayorin

Jamie Laning

Joshua Gornik

TABLE OF CONTENTS

I. Overview

II. Software Requirements and Specifications

III. Hardware Requirements and Specifications

IV. Comparison of similar projects and materials used

V. Weekly Progress Report – Errors and Solutions

VI. Team Roles

VII. References and links

**Overview**

The autonomous car project is modeled on the Donkey car project. An open source Artificial Intelligence self-driving car that models its driving on a human driver. The software is written in Python and runs on the Raspberry Pi 3 in conjunction with an x86 based PC.

To train the AI we first record a human driving the car

around a track. The donkey car software on the Pi that we have attached to the

RC car records, through the use on the installed camera, the images of the

track, along with the speed and turning information. All that data is stored

into what is called the “tub”. The tub is a directory on the micro SD card

installed on the Raspberry Pi. Once the data is collected into the tub, we then

transfer it to a PC running the donkey car software. The Raspberry Pi does not

have the processing power to train the AI, so we offload the data to a more

powerful PC that can do the work. Then we transfer the results back to the

software running on the Raspberry Pi and let the AI drive the track without

human intervention.

By training the car on multiple random tracks, the AI

learns to drive within the lines and not just to simply drive the same

formation repeatedly. The goal of this project was to train the AI to also

avoid obstacles in the way and stop at stop signs, but we were unable to

complete our stretch goals due to the COVID-19 pandemic cutting our project

Short.

**Software Requirements and Specifications**

Donkey car – is a modular self-driving library for Python that was developed for hobbyists with a focus on allowing fast experimentation and easy community contributions. (Documentation [link](https://www.donkeycar.com/))

Raspain (Stretch) – Raspberry OS

Windows 10 pro – Host PC

**Hardware Requirements and Specifications:**

Host PC

Autonomous Vehicle:

· Raspberry Pi3 – robot platform

· Sombrero v1 board – Servo driver replacement

· RC car

- Aggressive all terrain treaded tires and off-road style wheels

- 12mm standard wheel hexes

- 6061-T6 aluminum shock towers

- Brushed 380 Motor

- Li-Po compatible Brushed ESC

- 4WD shaft drivetrain

· 7.2V 1100mAh NiMH battery

· 400GB Micro SD Card

· Camera – OV5647 for the Raspberry Pi

Xbox Controller

Track

· Cube cardboards

· Colorful painters’ tape

Battery power bank with Micro USB cable

**Comparison of similar projects and materials used:**

Autonomous RC Car using Arduino

A high school student decided to create an autonomous RC car for his capstone project in Engineering Design, Development and Robotics course. It ended up winning an award for best autonomous vehicle at a high school STEM expo.

Instead of creating a vehicle from scratch he decided to use a car that was already paired with an Arduino Uno RedBoard. The Arduino was selected specifically because it is easy to program and use. He also mentions that for the motor he was using a Redcat Racing 03061 Splash-Resistant ESC with a brushed motor. The ESC was already working with a controller that came with the vehicle. However, he did not test this with a brushless motor since the school did not have one available.

The car collects data from 5 HC-SR04 Ultrasonic sensors which then send the data back to the Arduino, where it makes decisions on how to move. The Arduino then controls the steering servo and motor accordingly. The program only uses the standard Arduino servo library and no other additional libraries were involved.

Materials used

· (1) Arduino Uno Redboard Cost: $19.95

· (1) Breadboard - for this project, he took the +/- rail from one breadboard and used another, smaller breadboard. Any size will do. Cost: $5.90

· (5) HC-SR04 Ultrasonic Sensors Cost: $11.98

· (1) Potentiometer - used to control the speed of the car Cost: $12.98

· (20) Female-Male Dupont wires - I highly recommend having more to use as extenders for other wires if needed Cost: $10.99

· Soldering Iron with solder Cost: Cheapest one I saw on Amazon for a full kit $29.99

· Arduino Power Supply - in this case, he used (6) 1.2v AA batteries wired in series. External phone and tablet power banks will also work well when plugged into the USB port. Cost $6.49

· Tape, hot glue, and/or any other items used to fasten items together Cost in total: around $10

· (1) Toggle Switch (optional -- to turn the Arduino on and off) Cost: $8.03

ROI/Breakeven Cost: N/A was used primarily for a capstone project for a high school student.

MIT RACECAR Project

MIT participated in creating their own version of an autonomous vehicle although they are utilizing NVIDIA’s Jetson platform. The course description stated that the students will design and implement perception and planning algorithms for cars that can navigate quickly through complex environments. Individuals will be assigned into 6 groups with each team given one RC race car which is powered by an NVIDIA Jetson embedded supercomputer, an inertial measurement unit, a visual odometer, a laser scanner, and a camera. The class also taught the students how to use the Robot Operating System (ROS) on the NVIDIA platform while interfacing with sensors and actuators.

Materials Used

· The R/C Car – Traxxas Rally 7407

· On board computer – NVIDIA Jetson TK1

· 2D LIDAR – Hokuyo UST-10LX

· Camera – Point Grey Firefly MV

· Battery for electronics – Energizer XP8000AB

A couple of the electronic components come from Sparkfun, specifically an opto-isolator board and Razor 9DOF IMU.

The structure of the vehicle is augmented by acrylic platforms to mount the sensors and electronics, along with some 3D printed parts for the overall structure itself.

An optical flow visual odometer, a PX4FLOW, is mounted on the top platform. However, in practice the device was not used very much because it did not provide enough resolution for the environment where the cars were operating.

There is one custom built electronic part on the vehicle. A custom circuit board connects to the Jetson J3 header which adds access to the Jetson GPIO signals, adds a real time clock, and an opto-isolator. The GPIO access is used to send PWM signals to the vehicle’s servos and motors.

**Weekly Progress Report – Errors and Solutions:**

· Completed work

- Errors encountered and solutions made

January 30, 2020

· Hardware Parts were attained

February 6, 2020

· Circuit Boards were attached

· Donkey car kit assembled

· Host PC’s software is installed

February 14, 2020

· Raspberry Pi OS installed

· Network configuration were completed

- The pi is not connecting to the network. Changed the network interface to static.

- The wpa\_supplicant.conf is not detected on boot.

- ERROR: “Package 'mock' requires a different python: 3.5.3 not in '>=3.6'” SOLUTION: installing a different version of tensorflow.

· Base board is attached to the car chassis

· Cables were connected to the control box for throttle and steering

February 21, 2020

· The network problem was solved.

February 27, 2020

- Connecting the pi to the Host PC through ssh is unsuccessful.

· Default user were configured

· Donkey software is downloaded to the pi.

- Router stopped working. SOLUTION: Resetting the router.

· Hardware pieces were prepared for soldering.

February 28, 2020

· Parts were soldered together

- The configuration file in the donkey software is missing.

- ERROR: “ImportError: No module named ‘numpy.testing.decorators’”

March 2-6, 2020

- Due to multiple errors, installment from scratch is necessary.

- ERROR: “Package 'mock' requires a different python: 3.5.3 not in '>=3.6’“ SOLUTION: installing a different version of tensorflow.

- ERROR: “ImportError: No module named ‘numpy.testing.decorators’” SOLUTION: Uninstalled the old version and installed numpy 1.17.0

· OS and Donkey software are perfectly working

· Able to make changes on the configuration file of the car.

- The overall look of the car was requested to be changed.

- The car is not detecting the configuration file.

- Not able to control the car with the web controller.

March 9-14, 2020

· Change of the team roles were discussed, and the team decided to switch team members around.

- SOLUTION: Connecting the throttle and steering directly to the pi, instead of the control box of the RC car.

· Throttle and steering values were set through the calibration process.

- The drive function is not detecting the configuration file. SOLUTION: 1. Set the default configuration to our own configuration file. 2. Set the Servo drive to Sombrero.

- The car kept stopping. SOLUTION: We ordered another battery because we figured that it does not have enough battery life.

· The car is driving using the drive function which is the default program of the car.

· Connect an Xbox controller to drive the car via joystick.

· Redesigning the car and decided to use the old frame and case.

· Mapped the track.

- The track that was created is not enough space for the car.

March 16-20, 2020

- SOLUTION: We ordered more cardboard cubes for the track.

· Using the school’s Tennis court as our training track.

- ERROR: std::bad\_alloc SOLUTION: We ordered another micro SD card with bigger space.

· The case of the car was painted.

· Everything is attached together.

March 22-present

The project was put on hold due to the pandemic.

**Team Roles:**

I. Team Leader – is responsible for overseeing the project, running the team, and assigning responsibilities.

· Domenico Zaffora & Bethany Cacayorin

II. Programming Group – are responsible for the creation and application of the car software.

· Bethany Cacayorin

· Mark Schmit

· Domenico Zaffora

III. Engineering and Hardware Group – are responsible for the construction, assembly, and the maintenance of the car hardware.

· Charlie Liston

· Jamie Laning

· Bethany Cacayorin

· Joshua Gornik

IV. Car Testing and AI Training Group –are responsible for training the AI, applying the trained AI, and testing the car.

· Joshua Gornik

Domenico Zaffora·

Jamie Laning·

Mark Schmit·

Bethany Cacayorin

V. Notion and Presentation Group – are responsible for recording the changelog, taking notes, recording video and pictures, and editing the presentation video.

· Jamie Laning

· Domenico Zaffora

· Charlie Liston

· Bethany Cacayorin

**References and Links:**

Donkey Car Documentation:

<http://docs.donkeycar.com/>

Project folder:

<https://drive.google.com/open?id=1R6dONdkmNTzrD05lXTHBc28SU2yIADK_>

Project Video Presentation:

<https://www.youtube.com/watch?v=oKydJvL5bA4>