ECE 110 / 120 Honors Lab Project Proposal

Spring 2023

Project Report

First Project Proposal

Automated Driving Robot

Jason Mei (jasonm5, UIN 674870012) - ECE110, Stephen Cao (cao48, 678503888) - ECE120

1 Introduction

Autonomous vehicles have received significant attention in recent years due to advances in electric vehicles and the growing need for safer and more efficient modes of transportation. Self-driving systems, an application of machine learning, will revolutionize how we commute and travel thanks to the ability to use sensors, cameras, and software algorithms to navigate roads and make decisions. In addition, autonomous vehicles are also applied in service delivery. For example, Amazon and UPS are experimenting with ground robots for deliveries. The application of self-driving cars reduces traffic jams and improves delivery efficiency, thus improving social productivity. Therefore, for our project, we will build an automatic vehicle with functions like identifying roadblocks and judging directions. In our scenario, our robot will be required to drive up to specific brightly colored obstacles, and use its bottom-mounted spinner to knock them over.

1.1 Background Research

The difference between our project and previous or reference projects is that we use different sensors and different algorithms to improve the accuracy of autonomous driving. For example, in previous projects, such as the "Autonomous Car" by Raghav Pramod Murthy and Frank Lu, students used computer webcams, like the Logitech C920s webcam for vision, as well as advanced CPUs, like the Nvidia Jetson 2. On the other hand, we will be primarily focusing on the durability of our robot, so in our project, we will use both an Arducam camera specifically made for microcomputers, as well as other sensors for vision. In order to utilize the automation of the car, the main challenge is how to use the data obtained by multiple sensors to use data algorithms to make the car make decisions. We used three sensors in this project: MPU 6050 Module 3 Axis Accelerometers, HC-SR04 Ultrasonic sensors, and Arducam HM01B0.

The first sensor, an MPU 6050 Module 3 Axis Accelerometer, is used to collect acceleration data. There are two kinds of data provided by MPU 6050, linear acceleration (acceleration due to movement) and rotational acceleration (using the principle of Coriolis acceleration). In a linear accelerator, the change in capacitance is then converted into an electrical signal, which is proportional to the acceleration. In a gyroscope, moving in a circular path experiences a force perpendicular to its direction of motion. This force is sensed by a spinning mass and converted into an electrical signal that is proportional to the rotational acceleration. We will use an I2C interface to read the data because the accelerometer will provide acceleration measurements along the X, Y, and Z axes. We pass a simple integration method to get the vehicle's position and orientation. The linear acceleration measurements can be integrated to obtain the object's velocity, which can be further integrated to obtain the position of our vehicle. The data from the gyroscope will be integrated over time to give angular velocity, and the angular velocity can be used to estimate the orientation of the device. However, the method of integration will bring drift errors.The numerical integration process used to estimate the velocity and position can also introduce errors, especially if the time step (Δt) used in the integration process is very large. Therefore, to combine the data provided by the MPU-6050 sensor, we will use the method of sensor fusion. There are several algorithms that can be used for sensor fusion, and we choose the Kalman filtering approach. The Kalman filter could be used to estimate the orientation and position of the device based on the outputs of the accelerometer and gyroscope. The filter would model the orientation and position of the device as a set of states, and use a mathematical model to predict the orientation and position based on the current measurement. The filter would then update the prediction using the measurement from the accelerometer and gyroscope, taking into account the measurement noise and other uncertainty factors.Firstly, initialize the Kalman filter by defining the mathematical model describing the modeled system and the measurement noise, defining the state transition matrix, measurement matrix, and noise covariance matrices. Then, collect sensor data from the MPU-6050 sensor (the outputs from the accelerometer and gyroscope). Then, use the mathematical model and the previous state estimate to predict the system's current state, which involves multiplying the state transition matrix by the previous state estimate and adding the predicted process noise. Finally, update the state estimate using the Kalman gain to weigh the measurement and the prediction, then update the state estimate using the weighted measurement and prediction. Furthermore, the sensitivity of the accelerometer and gyroscope in an MPU-6050 sensor can vary depending on the range and configuration of the sensor. The accelerometer used by the MPU-6050 is typically sensitive to acceleration in the range of $\pm$ 2g, $\pm$ 4g, or $\pm$ 8g, where g is the acceleration due to gravity. The sensitivity in units of LSB(Least Significant Bit)/g. For example, if the accelerometer has a sensitivity of 16384 LSB/g, a change in acceleration of 1 g will result in a change of 16384 LSB in the sensor output.The gyroscope in an MPU-6050 is sensitive to angular velocity and is usually specified in units of LSB per degree per second (dps). The range of the gyroscope can vary, but common ranges are $\pm$ 250 dps, $\pm$ 500 dps, or $\pm$ 2000 dps. Same as the accelerometer, the sensitivity of the gyroscope is specified in LSB/dps.

The second sensor is a VL53L4CX Time of Flight Distance sensor used to sense walls and obstacles. It uses IO to trigger distance measurement. The emitter sends out a 940 nm laser invisible to the human eye alongside starting a timer, which then bounces off the target. Some of the light emitted is reflected back into the sensor. The timer stops, and the measured distance can be calculated according to the time interval. The formula will be:

$$Distance = \frac{(high\ level\ time * speed\ of\ light)}{2}$$

The third sensor is the Arducam OV2640, a 2MP camera module that can detect walls by capturing an image and processing it to identify the wall. We will process the data with an image-processing algorithm. We will use edge detection to allow the car to recognize obstacles because edges are areas in the image with a rapid change in intensity. We will use the Canny edge detection algorithm in Python. The Canny edge detection operator is a multi-level edge detection algorithm developed by John F. Canny in 1986. The first step in the algorithm is to smooth the image with a Gaussian filter. It means that the Gaussian filter (kernel) discretizes the Gaussian function and substitutes the corresponding horizontal and vertical coordinate indexes in the filter into the Gaussian function to obtain the corresponding value. The (2k+1)*(2k+1) filter is calculated as follows:

$$H[i,j] = \frac{1}{2\pi\sigma^2} e^{\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}}$$

The most common Gaussian filter is size=5, and its approximate value is:

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

The next step is to calculate the gradient magnitude and direction. Common methods use Sobel filters in computing gradients and directions. Compute the difference of x and y using the following convolution array:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

The gradient magnitude and direction are calculated using the following formulas:

$$G = \sqrt{(G_x^2 + G_y^2)}$$

$$\theta = arctan\frac{G_y}{G_x}$$

The gradient angle θ ranges from radians -π to π, and then approximates it to four directions, representing horizontal, vertical and two diagonal directions (0°, 45°, 90°, 135°). It can be separated by $\pm\frac{i\pi}{8}$ (i=1,3,5,7), and the gradient angle falling on each region is given a specific value, representing one of the four directions. The last step is to perform non-maximum suppression on the gradient magnitude. Non-maximum value suppression is an important step in edge detection. In the popular sense, it refers to finding the local maximum value of pixels. Along the gradient direction, compare the gradient value before and after it, and remove it if it is not a local maximum. We will use python coding to achieve this by using the OpenCV library, which contains the functions for Gaussian smoothing and canny operators (such as cv2.GaussianBlur and cv2.Canny function).

Compared with other projects which use similar cameras (such as the Logitech C920s webcam and the Nvidia Jetson 2), the largest advantage of Arducam is the lower cost. Also, Arducam is better than other cameras for use in autonomous driving projects because it is specifically designed for use with microcomputers. This makes it more compatible and easier to integrate with the systems used in autonomous driving projects. Additionally, the Arducam camera is designed to be durable and reliable, which is essential for our auto-driving vehicle, which may operate in harsh and potentially hazardous environments. The Arducam camera offers high-resolution imaging, which can be processed using image-processing algorithms to extract important information about the environment. The autonomous driving system can then use this information to make decisions and navigate the environment. It can capture images at high frame rates, which is important for real-time image processing and decision-making in our autonomous driving system.

2 Proposal

2.1 System Overview and Diagram

Below is a system diagram of the physical system used in the robot. The components will be explained in further detail.

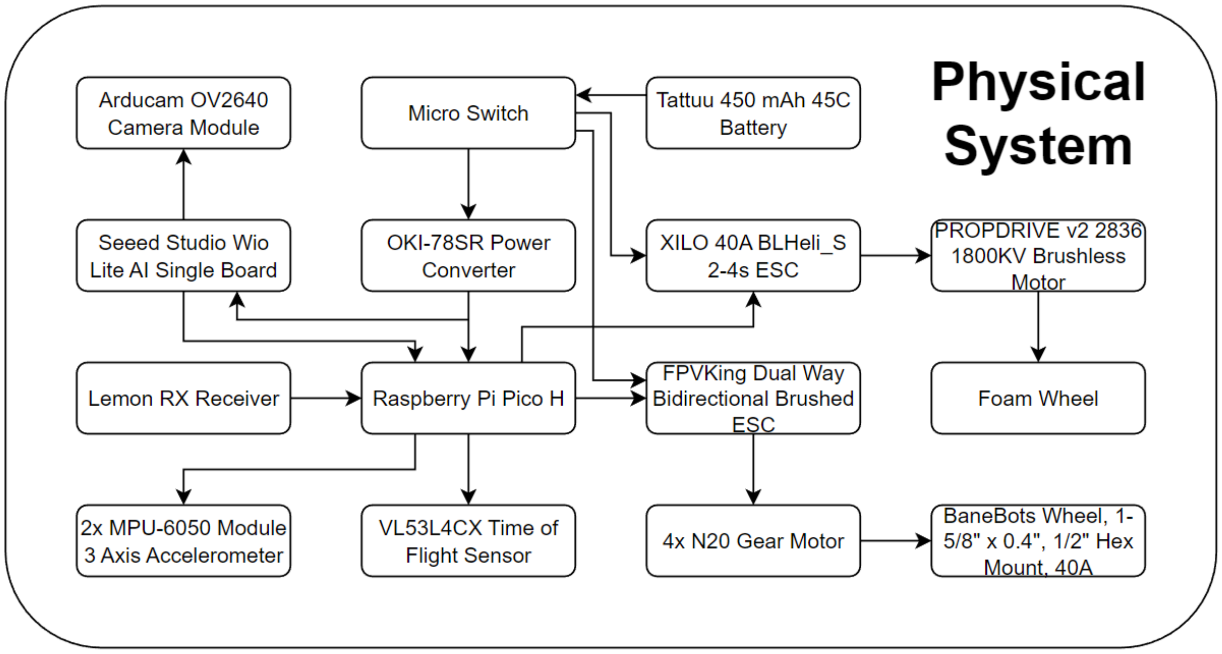
Physical System
Arducam OV2640 Camera Module
Micro Switch
Tattuu 450 mAh 45C Battery
Seeed Studio Wio Lite AI Single Board
OKI-78SR Power Converter
XILO 40A BLHeli_S 2-4s ESC
PROPDRIVE v2 2836 1800KV Brushless Motor
Lemon RX Receiver
Raspberry Pi Pico H
FPVKing Dual Way Bidirectional Brushed ESC
Foam Wheel
2x MPU-6050 Module 3 Axis Accelerometer
VL53L4CX Time of Flight Sensor
4x N20 Gear Motor
BaneBots Wheel, 1-5/8" x 0.4", 1/2" Hex Mount, 40A

Figure 1. Proposed Physical System

The primary vision computational bulk will be done using the Seeed Studio Wio Lite AI Single Board, a single-board computer that we will use for image processing. It will pass on its data to the Raspberry Pi Pico H, which will be connected to sensors as well as controlling the robot. We will be able to turn on the Pi through the Lemon RX receiver, which will be a manual override for safety. The Pi will also be powered using a step down 12V to 5V converter from the battery, with a micro switch for safety regulations.

The sensors that will be used will be a MPU 6050 Module 3 Axis Accelerometer, which can allow us to determine our position alongside our rotation in the X, Y and Z axes. We will also use multiple sensor HC-SR04 Ultrasonic sensors for determining our distance away from the walls, which will allow us to pinpoint our exact location within the area. Lastly, we will be using an Arducam HM01B0, which will allow us to capture live color video in 1600x1200 for computer vision.

We will also be using two electronic speed controllers, or ESCs: one for our drive motors, four N20 gearmotors connected to wheels. The second ESC will be used for the spinner motor, a PROPDRIVE v2 2836 outrunner motor attached to a foam wheel. These will be powered by a 450 mAh Tattu R-Line LiPo battery.

All of these electronics will be housed in a 3D printed chassis frame that has been previously designed on Thingiverse.

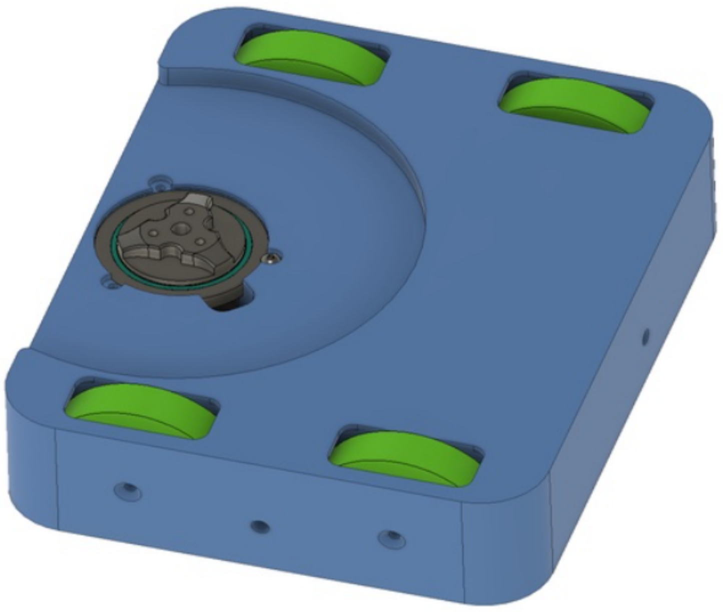

Figure 2: 3D mockup of chassis, found on Thingiverse, by Chris Miksovsky.

We will also cover the computational logic used by the robot while it is on. Below is a diagram of the system that the robot will be using.

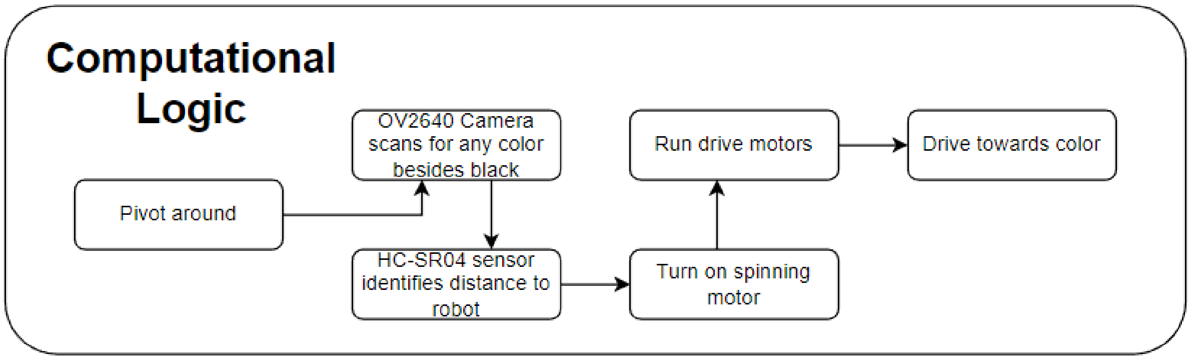
Computational Logic
Pivot around
OV2640 Camera scans for any color besides black
HC-SR04 sensor identifies distance to robot
Turn on spinning motor
Run drive motors
Drive towards color

Figure 3. Proposed Computational Logic System

First the robot will pivot around, while the camera will scan for any color that stands out against the black background. Once the camera identifies any specific color that exceeds the tolerance, the ultrasonic sensor will identify the distance the object is away from the robot. Then, it will turn on the spinning motor, and run the drive motors towards the object.

2.2 Parts List

The required parts for this project would easily exceed $200 of budget, but the iRobotics program "Design. Print. Destroy" has offered us a variety of free parts that will allow us to develop our bot under budget. Additionally, we will be required to purchase some additional parts that are not offered by any of the vendors listed. This is also still incomplete, and will be finalized for the final parts order.

Part Name	Purchase Link	Price	Purpose	Supplied
Raspberry Pi Pico H	https://www.digikey.com/en/products/detail/raspberry-pi/SC0917/	$5.00	Microcomputer for sensor and motor control	ECE budget
Seeed Studio Wio Lite AI Single Board	https://www.digikey.com/en/products/detail/seeed-technology-co-ltd/102110607/15277445	$39.24	Microcomputer for OpenCV usage	ECE budget
OKI-78SR Power Converter	https://www.mouser.com/ProductDetail/Murata-Power-Solutions/OKI-78SR-5-15-W36-C/	$7.50	Stepdown converter for powering both boards	ECE budget
MPU-6050 Module 3 Axis Accelerometer	https://www.digikey.com/en/products/detail/tdk-invensense/MPU-6050/4038009	$18.24	Gyroscopic Sensor	ECE budget
VL53L4CX Time of Flight Distance Sensor	https://www.adafruit.com/product/5425	$14.95	Ultrasonic sensor for distance calculation	ECE budget

XILO 40A BLHeli_S 2-4s ESC	https://www.getfpv.com/xilo-40a-blheli-s-2-4s-esc.html	$0.00	ESC for spinner motor	DPD robot kit
FPVKing Dual Way Bidirectional Brushed ESC	https://www.amazon.com/FPVKing-Bidirectional-Brushed-2S-3S-Control/dp/B081LBL42J/	$0.00	ESC for drive motors	DPD robot kit
Power Switch	https://www.fingertechrobotics.com/proddetail.php?prod=ft-mini-switch	$0.00	Safety Switch	DPD robot kit
Tattu 450 mAh 45C Battery	https://www.getfpv.com/tattu-450mah-11-1v-45c-3s1p-lipo-battery-pack-jst-syp.html	$0.00	Battery	DPD robot kit
Lemon-RX Receiver	https://lemon-rx.com/index.php?route=product/product&product_id=258	$0.00	Receiver	DPD robot kit
N20 Gear motor	https://www.amazon.com/Greartisan-1000RPM-Torque-Reduction-Gearbox/dp/B07FVMVGM3	$11.99	Drive motor	Self-purchase
PROPDRIVE v2 2836 1800KV Brushless Outrunner Motor	https://hobbyking.com/en_us/propdrive-v2-2836-1800kv-brushless-outrunner-motor.html	$21.99	Spinner motor	Self-purchase
Arducam OV2640 Camera Module	https://www.uctronics.com/arducam-ov2640-camera-module-2mp-mini-ccm-compact-camera-modules.html	$6.99	Color camera for object detection	Self-purchase
4x BaneBots Wheel, 1-5/8" x 0.4", 1/2" Hex Mount, 40A	https://banebots.com/banebots-wheel-1-5-8-x-0-4-1-2-hex-mount-40a-black-orange/	$10.00	Wheels	Self-purchase
4x BaneBots 3mm Shaft to Hex Hub	https://banebots.com/hub-hex-series-40-set-screw-3mm-shaft-1-wide/	$16.00	Wheels	Self-purchase

Total ECE 110 Honors budget	Total Self-Purchase budget
$84.93	$66.97

We plan to use the ECE honors remaining budget on any parts that we have not been able to anticipate, like screws that we may need.

2.3 Timeline

Estimated timeline by week:

- Week of January 31: submit initial proposal, do preliminary research on computer vision

- Week of February 7: complete final proposal, begin 3D CAD for printing, order parts
- Week of February 14: Start computer vision work, finalize CAD, print chassis
- Week of February 21: Continue computer vision, start testing sensors on raspberry pi
- Week of February 28: Continue computer vision, identify needed parts and screws
- Week of March 7: Midsemester checkpoints, submit final parts list
- Week of March 14: Work on developing computer logic, test motors
- Week of March 21: Assemble initial prototype, check for movement
- Week of March 28: Catch-up week
- Week of April 4: Solve any structural issues, reprint chassis if needed, fine tune vision
- Week of April 11: Work on field-centric movement, use gyro and ultrasonic sensors
- Week of April 18: Finalize field-centric movement, final assembly of robot
- Week of April 25: Fine-tune issues, begin final report
- Week of May 2: Fix any remaining bugs, submit final report and video

2.4 Possible Challenges

This project will likely require a lot of software development, and we will need to learn more about computer vision, and working with the Raspberry Pi Pico. We will also need to be wary about ensuring that all of the electronics will work together without fear of overvoltage, issues which have caused past Arduino and microcontroller projects to fail in the past.

An issue that I (Jason) will face will be that of scheduling time to work on the project. I am very busy this semester, so I will have to be flexible in using time properly. Additionally, we will be heavily relying on past 3D printed robots, and identifying a reliable source for our base will be important in developing a robot that will work consistently.

The challenge I (Stephen) will meet is the software part. I need to learn how to write coding on the Raspberry Pi Pico to connect with sensors to ensure our robot moves automatically. Moreover, my schedule is also a bit dense this semester because I will have to do research about parallel programming and architecture, so managing my time and studying effectively is necessary.

References

[1]

Thingiverse.com, “Blue Screen of Death - Antweight combat robot (1lb) by chrismik.” https://www.thingiverse.com/thing:4075681 (accessed Jan. 31, 2023).

[2]

Blue Screen of Death - 1lb Antweight Combat Robot, Part 1 - Design. Accessed: Jan. 31, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=FOHKxxFWmgA

[3]

AGTx, “Driving an ESC/Brushless-Motor Using Raspberry Pi,” *Instructables*. https://www.instructables.com/Driving-an-ESCBrushless-Motor-Using-Raspberry-Pi/ (accessed Jan. 31, 2023).

[4]

M. Lab, “MPU6050 with Raspberry Pi Pico (Accelerometer, Gyroscope, and Temperature),” *Microcontrollers Lab*, Feb. 27, 2022. https://microcontrollerslab.com/mpu6050-raspberry-pi-pico-micropython-tutorial/ (accessed Feb. 02, 2023).

[5]

“Arducam Mini 2MP Plus – OV2640 SPI Camera Module for Arduino UNO Mega2560 Board & Raspberry Pi Pico.” https://www.arducam.com/product/arducam-2mp-spi-camera-b0067-arduino/ (accessed Feb. 02, 2023).

[6]

L. P. last updated, “How to Use an Ultrasonic Sensor with Raspberry Pi Pico,” *Tom’s Hardware*, Jan. 29, 2021. https://www.tomshardware.com/how-to/raspberry-pi-pico-ultrasonic-sensor (accessed Feb. 02, 2023).

[7]

“PenguinTutor - RC Remote Control for a Raspberry Pi Pico.” http://www.penguintutor.com/news/electronics/rc-pico (accessed Feb. 02, 2023).

[8]

“Autonomous Car - ECE 110/120 Honors Lab Section - Illinois Wiki.” https://wiki.illinois.edu/wiki/display/ECE110HLSF15/Autonomous+Car (accessed Feb. 03, 2023).

[9]

S. Hymel, “Raspberry Pi Pico and RP2040 - MicroPython Part 2: I2C Sensor and Module.” https://www.digikey.com/en/maker/projects/raspberry-pi-pico-and-rp2040-micropython-part-2-i2c-sensor-and-module/b43e7958153f41fc9e7403df4d626ba5

[10]

“OpenCV: Canny Edge Detection.” https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (accessed Feb. 04, 2023).

[11]

A. Rosebrock, “OpenCV Edge Detection (cv2.Canny),” *PyImageSearch*, May 12, 2021. https://pyimagesearch.com/2021/05/12/opencv-edge-detection-cv2-canny/ (accessed Feb. 04, 2023).

[12]

A. Bondarev and A. Kalmuk, “Benchmarking OpenCV on STM32 MCUs,” Jul. 02, 2021. https://www.embedded.com/benchmarking-opencv-on-stm32-mcus/ (accessed Feb 04, 2023).