Autonomous trash-collecting robot

Nguyen T. K. Trinh, Dang M. Vu

VNU-HCM High School for the Gifted

Instructed by: **Dr. Phùng Mạnh Dương** (Fulbright University Vietnam), **Dr. Đoàn Nhật Minh** (Fulbright University Vietnam), **Dr. Lê Quân** (Fulbright University Vietnam)

This project is part of the 2023 PTNK Innovation Initiative (PII) Research Internship Program at Makerspace – Fulbright University Vietnam under the topic "4d. Smart Robot" from June 15, 2023, to September 5, 2023.

Abstract

This research focuses on implementing a trash-collecting robot to automatically collect plastic bottles. This research aims to assemble an autonomous bottle-collecting robot with obstacle avoidance algorithms and computer vision. Its results can be applied to future research to develop a robot system that operates independently. The theoretical basis and research results will be presented in the next sections.

Introduction

Nowadays, pollution caused by plastic trash surges, resulting in severe consequences for the environment. However, automatic trash-collecting robots on the market today are somewhat difficult to access and quite complicated. According to *Advantages and Disadvantages of Automation* (2023, July 4), Akshay Badkar mentioned that automated systems have been previously deployed but have not been optimized in terms of cost and hardware. The goal of this research is to improve and create a more optimized robot system after designing, simulating obstacle avoidance algorithms, and retraining the object detection model.

Research questions

For this robot project, we proposed 3 following research questions:

1. How to automatically collect plastic bottles?
2. Can the robot collect all plastic bottles in a specified space?
3. Can the robot recognize water bottles on its path fast enough when collecting?

Methodology

Part 1: Mechanical Design

1.1 Measurements

Firstly, we took measurements of the length and width of the components needed in the robot system. The objects that have been measured include:

- Water bottles
- A rechargeable (portable) charger
- Two battery holders, each containing two 18650 batteries
- A Jetson Nano computer
- A1M8 Lidar
- DC motors
- An Arduino Uno board
- A USB Camera
- A rotating front blade to collect water bottles

Detailed measurements of the above components can be found in this repository.

1.2 Design

After obtaining data about the components that need to be placed on the robot, we began to design a 3D model for the robot's frame. The 3D model file can be found in this repository.

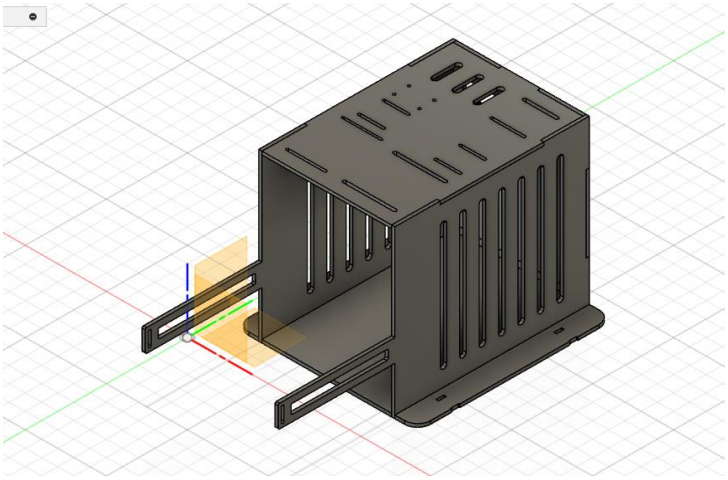

Figure 1.2. The robot's frame

Part 2: Electronics

In this part, we will introduce every electronic component we used in our robot and specify why we chose each.

2.1 Overview

Our robot includes two processing units: a Jetson Nano computer and an Arduino Uno board. We use 6 direct current (DC) 3V–9V motors connected to the Arduino Uno via two L298N drivers, of which 4 are for controlling the robot's wheels and 2 are for controlling the front blade. A lidar for obstacle avoidance and a camera for bottle detection are also added.

	Component	Function
Controllers	Jetson Nano	Process sensor's information and send commands to Arduino Uno via serial communication
	Arduino Uno	Receive serial commands and control motors' direction
Drivers	L298N motor driver	Control motors' speed via pulse-width modulation
Actuators	DC motors	Run the robot's wheels and rotating blade
Sensors	SLAMTEC RPLidar A1M8	Locate obstacles and record their data
	Logitech C270 Webcam	Capture real-time video

2.2 Controllers

In this robot, the Jetson Nano receives sensor information, processes it, and sends commands to the Arduino Uno via serial communication. Subsequently, Arduino Uno has to process these commands to actuate the motors.

2.2.1 Jetson Nano

Initially, for the robot's computer, we had to choose between a Raspberry Pi 3 or an NVIDIA Jetson Nano. After determining the goal of the robot and choosing a training model, we decided to choose the Jetson Nano for the controller. The Raspberry Pi 3 has a *4x ARM Cortex A53 CPU processor at 1.4 GHz* and a *Broadcom VideoCore IV GPU at 250 MHz.* In comparison, the Jetson Nano has a *Quad-Core ARM Cortex-A57 64-bit CPU at 1.4 GHz* and a *128-core NVIDIA Maxwell architecture-based GPU*. The benchmarks created by Arnab Kumar Das demonstrate this difference:

Model	Application	Framework	NVIDIA Jetson Nano	Raspberry Pi 3
ResNet-50 (224×224)	Classification	TensorFlow	36 FPS	1.4 FPS
MobileNet-v2 (300×300)	Classification	TensorFlow	64 FPS	2.5 FPS
SSD ResNet-18 (960×544)	Object Detection	TensorFlow	5 FPS	DNR
SSD ResNet-18 (480×272)	Object Detection	TensorFlow	16 FPS	DNR
SSD ResNet-18 (300×300)	Object Detection	TensorFlow	18 FPS	DNR
SSD Mobilenet-V2 (960×544)	Object Detection	TensorFlow	8 FPS	DNR
SSD Mobilenet-V2 (480×272)	Object Detection	TensorFlow	27 FPS	DNR
SSD Mobilenet-V2 (300×300)	Object Detection	TensorFlow	39 FPS	1 FPS
Inception V4 (299×299)	Classification	PyTorch	11 FPS	DNR
Tiny YOLO V3 (416×416)	Object Detection	Darknet	25 FPS	0.5 FPS
OpenPose (256×256)	Pose Estimation	Caffe	14 FPS	DNR
VGG-19 (224×224)	Classification	MXNet	10 FPS	0.5 FPS
Super Resolution (481×321)	Image Processing	PyTorch	15 FPS	DNR
Unet (1x512x512)	Segmentation	Caffe	18 FPS	DNR

Source: Nvidia Jetson Nano Review and Benchmark (arnabkumardas.com)

With better processing speed for tasks related to images, object detection, and segmentation, Jetson Nano can run code programs, especially bottle detection, better.

2.2.2 Arduino Uno

We chose the Arduino Uno for controlling the robot motors' direction and to serially communicate with the Jetson Nano. The Arduino Uno is capable of handling high power requirements, is widely compatible with various drivers and motor control circuits, and has rich, easy-to-use support library resources within the Arduino IDE.

2.3 Drivers

The driver is used to control the motors' speed through pulse-width modulation (PWM). The common L298N drivers can handle high voltages, feature circuit protection, and can be used with an Arduino Uno.

2.4 Actuators

The 3V–9V DC motors used in this robot are widely used in automatic robots, are easy to use, and are compatible with the L298N drivers.

2.5 Sensors

2.5.1 Lidar

To locate and avoid obstacles, we used a lidar. Given the low complexity of the task and the available resources, we chose the SLAMTEC RPLidar A1M8 for its stable accuracy, simple construction, and affordability.

2.5.2 Camera

For detecting plastic bottles, we chose a regular Logitech C270 webcam, which can capture up to 30 frames per second (fps). It sends information to the Jetson Nano via USB.

2.6 Batteries

2.6.1 Circuit power supply batteries

We use four rechargeable 18650 AA batteries to supply the entire circuit.

2.6.2 Jetson Nano computer power supply

We used a portable charger to power the Jetson Nano via a micro USB port.

Part 3: Software and Algorithms

For this project, we set out three main tasks for the robot: recognizing water bottles, avoiding obstacles, and operating the robot. Our codes can be found in this GitHub repository.

3.1 Booting the Jetson Nano

To boot the Jetson Nano for usage on the robot, we did the following installations:

- Ubuntu Bionic Beaver operating system on the Jetson Nano
- ARM64 architecture Visual Studio Code IDE
- Arduino IDE 1
- ROS Melodic, RPlidar package, Hector Slam, and Turtlebot 3 Simulation
- Python 2.7 (used for ROS Melodic), Python 3.8 (used for plastic bottle recognition), OpenCV, ImageAI, and other computer vision supporting packages: ImageAI/requirements.txt at master · OlafenwaMoses/ImageAI (github.com) ImageAI/requirements_gpu.txt at master · OlafenwaMoses/ImageAI (github.com)

3.2 Obstacle Avoidance Algorithms

Initially, we had two directions to use the Lidar technology to avoid obstacles: (1) scan the map of the area first and use an algorithm to let the robot go through that map, and (2) use Lidar to scan data in real-time while the robot moves while at the same time performing the obstacle avoidance algorithm.

3.2.1 Mapping the area

To scan and save the area's map as a *.tiff* file (Tag File Format), we installed ROS Melodic, including the RVIZ and Hector SLAM packages. These packages are used to visualize input data from the lidar and save it.

To control the robot remotely, we installed the RealVNC software to access the Jetson Nano and transmit and receive data remotely. From there, we controlled the robot through the functions *forward(), backward(), turnLeft(), turnRight(), rotateLeft(),* and *rotateRight().* After obtaining the area's map, we can use it. We used Gazebo built into ROS to simulate obstacle avoidance algorithms in cost map layers.

This method is optimal in many aspects, as we can overlap multiple cost map layers to avoid obstacles in a pre-defined area.

3.2.2 Getting real-time data

To get real-time data from lidar, we use the *iter_scans()* function in the rplidar package, thereby obtaining the distance to the nearest obstacle corresponding to angles from 0 degrees to 360 degrees. Then directly execute the algorithm while scanning the data.

This method can be deployed quickly and can be run in the same program as the bottle detection code via multi-threading.

3.2.3 Algorithm

Eventually, our team decided to use the live data collection option because this option is more time-optimized and gives results with moderate accuracy. The angle error of the lidar is within 1 degree, while the distance error is 2 cm. The areas where data is collected and the robot's actions with obstacles located in the corresponding areas are shown in the diagram:

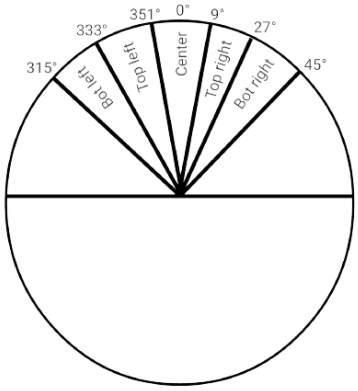

Figure 3.2a. Angle range divisions

After dividing the obstacle detection areas, the robot will execute the obstacle avoidance algorithm following this flow chart:

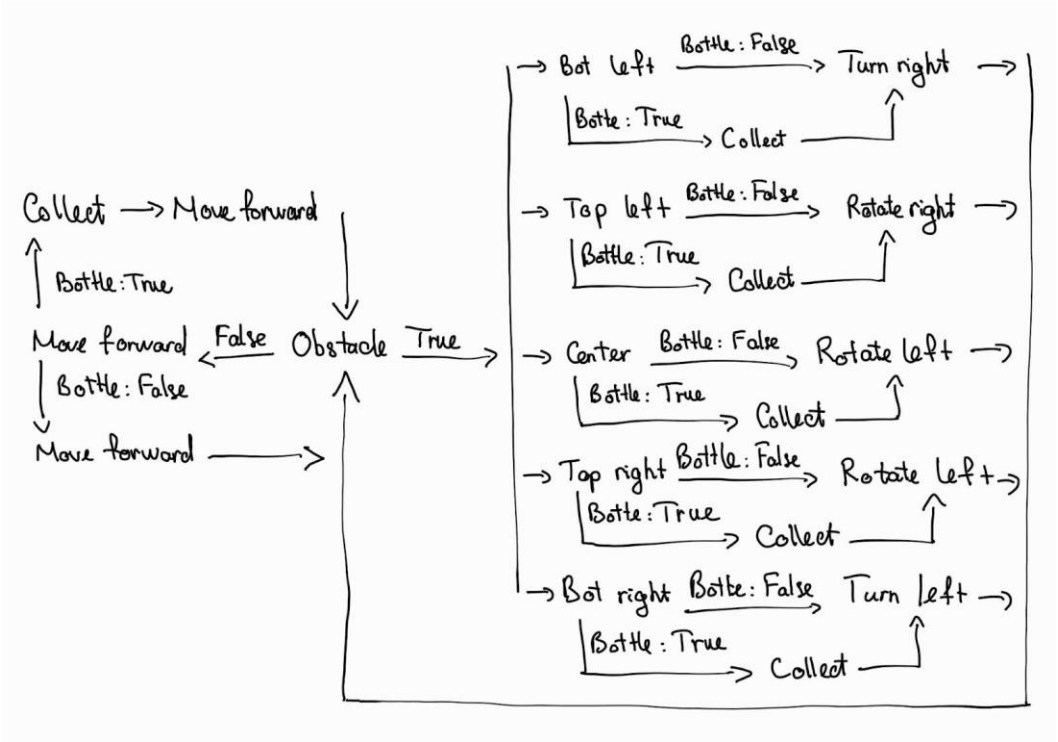

Figure 3.2b. Obstacle avoidance algorithm

3.3 Bottle Detection Algorithms

One of our robot's main features is to detect bottles from the camera's input video. To achieve this, we used the OpenCV library to capture the video in real-time and the ImageAI library and YOLO object detection model to recognize bottles in the video.

3.3.1 Capturing Real-time Video from the Camera

To capture video from the camera in real-time, we used the OpenCV library in Python. First, the camera is connected via *VideoCapture()*. Frames are captured via the looped *read()* function to process each frame. Then, we use the *imshow()* function to display the frame in the computer's window, making it convenient to monitor camera activity.

3.3.2 Object Detection Models

For the water bottle detection task, we used the ImageAI library and an object detection model compatible with it - YOLO. We initially tested the YOLOv3 version on a personal computer. This model achieves a recognition rate of 4 frames per second (fps) with a probability (confidence) ranging from 93% to 98%.

However, after trying to run YOLOv3 on the Jetson Nano, the detection speed dropped significantly to 0.6 fps, creating a noticeable delay of about 9-15 seconds. With its low detection speed and significant latency, this model cannot be used on our robot.

To enhance the performance, we tested TinyYOLOv3, a lighter version of YOLOv3, on a personal computer. The detection rate and file size of each model are shown below:

Model	Detection rate (fps)	File size (MB)
YOLOv3	49	237
TinyYOLOv3	277	34

This model has a recognition speed of up to 30 fps on a personal computer. However, the detection probability drops to 60%-72%.

To improve accuracy, we trained the TinyYOLOv3 model with a data set of water bottle images by Vnu (2022, December) from Roboflow. This trained model has a similar detection rate of 25-30 fps, but the probability has improved up to 75%-81%. This model balances both the speed and accuracy of the water bottle recognition task.

Finally, we ran the trained TinyYOLOv3 model on the Jetson Nano. This model achieves a detection speed of 12-18 fps, slightly lower than when running on a personal computer, while the recognition probability remains stable at 75%-81%. This model is best suited for the bottle detection task on the Jetson Nano.

Model	Detecting rate (fps)	Probability (%)
YOLOv3	4	93 - 98
YOLOv3 - trained	0,6	90 - 97
TinyYOLOv3	30	60 - 72
TinyYOLOv3 – trained (final)	12 - 18	75 - 81

3.3.3 Calculating the Distance from the Robot to the Bottle

After successfully detecting a bottle, the robot must determine the distance to the water bottle to calculate the approximate time for the robot to run to the water bottle. Because the camera is fixed to the robot, we can use the image from the camera to calculate the distance to the water bottle.

Initially, we applied the image projection method. Through calibrating the camera, we found the parameters of distortion of the image through the camera to apply to the distance calculation formula. However, the results gave an error bound of up to 60 centimeters (cm). Therefore, we decided to establish our method of calculating the approximate distance based on the following steps:

- Create a coordinate plane on the camera frame: Take the horizontal line at the bottom edge of the frame as the *x*-axis; the line dividing the frame vertically in half as the *y*-axis; and the intersection of the two lines is the origin O, which is considered the center of the robot.
- Divide the frame into 10 equal parts along the *y*-axis, and find the real-life length of the parts by placing a ruler right at the dividing line. After doing this, we put the data into the function equation finder tool to find d_y, the real-life distance from the object to the real point of O if the object is placed on the *y*-axis,

$$d_y = \frac{(309.059 \cdot e^{1.04215y} + 130.13)}{10}\ cm$$

  where *y* is the distance in pixels from the object image to the *y*-axis.
- We can calculate the angle, α, between the object's image and the origin O by using the equation

$$\alpha = tan^{-1}\left(\frac{y}{x}\right)$$

  where *x* is the distance in pixels from the object image to the *x*-axis.
- From there, we find the approximate distance, *d*, from the object to the robot's center

$$d = \frac{d_y}{\cos(\alpha)}\ cm$$

This method gives optimistic results with an error bound of less than 10cm. The coordination axes, variables, and calculations of the above formula can be visualized with the following diagrams:

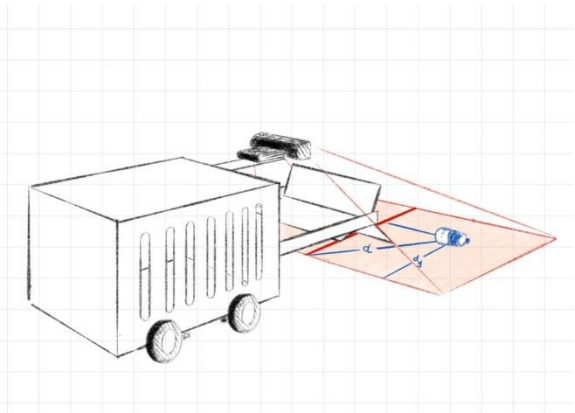

Figure 3.3a. Camera's capture area

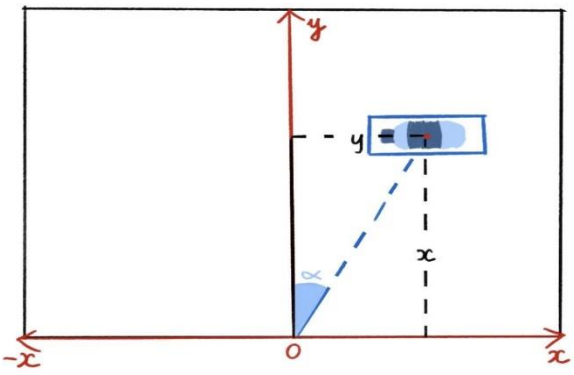

Figure 3.3b. A bottle's picture on the camera's frame

3.4 Multi-threading and Controlling Motors

3.4.1 Multi-threading

To run two programs, bottle detection and obstacle avoidance, in parallel, we have divided the programs into threads and launched multi-threading. Because our tasks can be performed collectively on one CPU core, to reduce processing time, we chose multi-threading over multi-processing.

3.4.2 Serial Communication with Arduino Uno

Jetson Nano and Arduino Uno communicate via the 9600 baud rate serial port. Below is a list of commands we used for serial communication between the Jetson Nano and Arduino Uno:

Command	Sent from	Action
forward, backward	Jetson Nano	Moving forward, backward
collect	Jetson Nano	Moving forward and activating the blade to collect bottle
turnLeft, turnRight	Jetson Nano	Turn left, right
rotateLeft, rotateRight	Jetson Nano	Rotate left, right
stop	Jetson Nano	Stop current action
done executing	Arduino Uno	(For functions in seconds) alerting that action is finished

Results

1. **Robot's efficiency**

Our robot is capable of simultaneously detecting, collecting bottles, and avoiding obstacles while navigating. The programs for capturing real-time video and recognizing bottles; handling lidar's live data and avoiding obstacles; and communicating serially between the Jetson Nano and the Arduino Uno had been established. To optimize the bottle detection program, we have successfully trained the TinyYOLOv3 model to increase detection speed and efficiency. A custom formula for approximating the distance to a target bottle is also formed.

2. **Cost-effectiveness**

At the end of the internship period, our group had successfully implemented an autonomous robot system. The robot components' prices are shown:

- Jetson Nano: $185 to $205 *(4,500,000 to 5,000,000 VND)*
- RPLidar A1M8: $103 to $111 *(2,500,000 to 2,700,000 VND)*
- Arduino Uno, DC motors, L298N drivers, wires: $29 to $41 *(700,000 to 1,000,000 VND)*
- Logitech C270 Webcam: $12 to $21 *(300,000 to 500,000 VND)*
- Frame material (mica): $12 to $21 *(300,000 to 500,000 VND)*

By contrast, the price of automatic cleaning robots on the market today is around $22000 *(530,000,000 VND)* (according to *Driverless Intelligent Cleaning Robot Commercial Industrial Floor Washer Machine Mop Vacuum Sweeping Cleaning Robot* on alibaba.com). The total cost of our robot system ranges from only $341 to $400 *(8,300,000 to 9,700,000 VND)*. Once cost-optimized, this system becomes more accessible than other currently available trash collection systems.

Discussion

1. Experiment design

The goal of this robot is to run in closed spaces such as classrooms and offices to solve the problem of littering, especially water bottles and plastic waste. Hence, this robot can collect, classify, and process plastic waste automatically.

2. Note

It is necessary to design a 3D file with the smallest division of 4 mm, which is suitable for CNC routers. When supplying power to the circuit, it is important to determine the correct source voltage and current intensity accordingly to avoid causing damage, fire, explosion, or a short circuit.

In addition, to improve the processing rate of Jetson Nano, we should not use a TF memory card for running the Ubuntu operating system for Jetson Nano. Instead, initializing programs from eMMC memory on the Jetson Nano board can enhance its processing speed.

3. Issues

After finishing our robot, we propose several drawbacks to our robot system that can be resolved in future similar projects:

- The SLAM problem can be solved over a longer research period.
- This robot cannot scan obstacles that are not at the same height. To tackle this problem, we can use the obstacle avoidance method in the pre-scanned and limited area map. Another idea is to use an additional ultrasonic sensor that communicates with the Arduino to identify obstacles that are not at the same height as the lidar.
- The robot's frame design also has some aesthetic and safety issues as the internal components are not protected. To resolve this, we can redesign the robot or create a cover on top to shield the components.

Conclusion

1. Demand for Garbage Cleaning Robots

As the important issue related to environmental pollution due to plastic waste becomes more severe, the need for trash cleaning robot systems rises. The goal of the research is to create robots that can solve this problem automatically.

2. Optimizing Garbage Cleaning Robots

The currently available automatic trash cleaning robots on the market are still difficult to access and deploy. Hence, this research focused on improving and optimizing robots to reduce costs and increase efficiency.

3. Object Detection Model’s Performance

This research focused on training and improving the object detection model on the robot. After using the Tiny YOLOv3 model, we improved the speed and accuracy of bottle detection.

References

Akshay Badkar. (2023, July 4). Advantages and Disadvantages of Automation. Retrieved from https://www.simplilearn.com/advantages-and-disadvantages-of-automation-article

Arnab Kumar Das (2019, May 26). Nvidia Jetson Nano Review and Benchmark. Retrieved from https://www.arnabkumardas.com/platforms/nvidia/nvidia-jetson-nano-review-and-benchmark/

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools.*

OpenCV. (2014). The OpenCV Reference Manual (2.4.13.7). Retrieved April, 2014, from http://docs.opencv.org/

Vnu. (2022, December). *Plastic_Bottle Dataset* [Open Source Dataset]. *Roboflow Universe*. Roboflow. Retrieved from https://universe.roboflow.com/vnu-jozz8/plastic_bottle-lxw9d

Moses. (2018). ImageAI, an open-source Python library built to empower developers to build applications and systems with self-contained Computer Vision capabilities [Software]. Retrieved from https://github.com/OlafenwaMoses/ImageAI

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779–788). Retrieved from https://doi.org/10.1109/CVPR.2016.91

Jiang, Zicong & Zhao, Liquan & Li, Shuaiyang & Jia, Yanfei. (2020). Real-time object detection method based on improved YOLOv4-tiny.

Acknowledgment

We would like to express our sincere gratitude to all those who have contributed to the completion of this project.

Firstly, we truly value the guidance and aid given to us during our project by our instructors, Mr. Phung Manh Duong, Mr. Doan Nhat Minh, and Mr. Le Quan. Their insightful feedback and suggestions helped to shape the project's scope. Secondly, we would also like to express gratitude to Mr. Nguyen Thanh Long and Mr. Nguyen Quoc An, our mentors, for their unwavering assistance whenever our project faced technical issues. Additionally, we would like to thank Makerspace - Fulbright University Vietnam for giving us the supplies, environment, and resources that were crucial to the completion of our project. Lastly, we wish to acknowledge the PII coordination team for their continuous supervision and assistance during our participation in the program, as well as the PII organizing team for developing such a valuable learning and experience program.

Without the contributions and assistance of all those mentioned above, this project would not have been possible. Thank you all for your valuable contributions and support.

Exclusive Appendix for the PII program

This research project is carried out within the PTNK Innovation Initiative (PII) Summer Research Internship Program under the topic "4d. Smart Robot" at the research facility "Makerspace - Fulbright University Vietnam" from June 15, 2023, to September 5, 2023.

We would like to express our gratitude to our instructors, Mr. Phung Manh Duong, Mr. Doan Nhat Minh, and Mr. Le Quan, as well as our mentors, Mr. Nguyen Thanh Long and Mr. Nguyen Quoc An, for their dedicated guidance and support throughout the project. We would also like to thank Makerspace - Fulbright University Vietnam for providing the necessary equipment, workspace, and resources to carry out this project. Additionally, we would like to thank the PII organizing team for creating this program as a valuable learning and experiential opportunity and the PII coordination team for closely monitoring and supporting our work throughout the program.

This research project has achieved significant milestones, including:

Time period	Milestone
21/06 - 30/06/2023	Preparing and designing the robot
03/07 - 08/07/2023	Getting to know Arduino and the Ubuntu operating system on Jetson Nano
10/07 - 27/07/2023	Working on bottle detection and obstacle avoidance programs
31/07 - 14/08/2023	Optimizing the robot

With the aim of sharing knowledge, aligned with the spirit of continuous learning and accumulation in science, we would like to share the tools, knowledge, and useful keywords that have been beneficial to us during the course of this project, including:

- Fusion 360, a 3D design software
- The ImageAI library
- Depth Camera
- Lidar

At the end of the program, we have gained valuable experiences for ourselves, and we are ready to continue our path of continuous learning and research. We hope that the results we share will make a meaningful contribution to the community and will be continuously built upon.