June 7, 2024

QuackTrack Final Report

Written By: Cole Abbott, Evangeline Michael John Bose, and Noah Millan

Electronic System Design II: Project, Team 3
Instructor: Professor Ilya Mikkelson

Contents

Sections

Figures

Tables

Summary

Over the course of this quarter, our team designed and constructed a robot shaped like a duck that can follow someone around. To begin this process, we researched parts that would allow for a car system that included motors, a camera, and ESP32 microcontrollers. We were quickly able to assemble a car that was controlled by a website to go forward, backwards, and spin to the left or to the right. We also separately set up a camera that streamed the images it captured through an ESP32 to a website and displayed them there. The next step was to implement a strategy for detecting people within the camera frame. We eventually landed on the YOLOv3 training dataset and used that to be able to detect when people were in frame on the website. Using this information, we determined where in frame the person was and used Websockets to transfer that information from the website back to the ESP32 that controlled the motors.

The next step was to control the motors using the location provided by image processing concerning the location of the person in frame. We tested out some thresholds and landed on something that was reliable at determining which way to move in order to follow a person around based on the data received through the Websockets channel. Once we had a functional device, we wanted to consolidate the functionality of the device to a single ESP32, as previously the microcontroller that handled the camera and the one that handled the motors were different. We ran into some issues initially, but eventually we were able to make the car system smaller, while still retaining the functionality from before. Since we had achieved functionality, we began to prepare for the final design by ordering more parts and printing the final enclosure in the shape of a duck.

However, we were experiencing unreliability in the speed of our video streams, reaching peaks of 8.5 fps but often dropping as low as 0.1 fps with no indication as to the cause. After searching for a time, it was determined that the cause of the slowdown was interference coming from the clock signal of the camera, and slowing that down allowed for the device to have reliable functionality at about 6.5 fps. We also included on the website a manual mode which turned off the automatic tracking in favor of directional buttons on the website that controlled the device. Once we eliminated all of the bugs and assembled everything, we were left with a finished duck that could use its camera to follow someone around or be controlled directly using the website.

Introduction

The goal of our design project was to create a device that could record images and perform image processing on them in order to detect where a person is in reference to the device and be able to follow them around as they moved. We later specified this goal to have our

product at least be functional when ignoring major obstacles and when no more than one person was in the camera frame. We also determined that the device should be duck-shaped, to make it a more marketable and fun device for children to be able to play with. To accomplish this, we first developed a car, image processing strategy, and video stream separately, then combined these elements in order to create the completed design.

While the device itself is rather unique in its functionality, the individual parts can be traced back to other designs. Firstly, the car with two wheels was inspired by a similar device that Cole had constructed previously, even using some of the same parts. However, this is not a cause for concern as the car itself is rather simple, just a chassis with two wheels and motor for the wheels. The camera stream uses a built-in protocol that came with the ESP32, however that functionality is part of the rationale behind purchasing the ESP32-CAM module specifically, so there are no qualms in using these protocols. Finally, the image processing that we used for this device has certainly been used for applications outside of our own, and we did not create it. However, it is an open source program, so we would not be in violation were we to actually sell this device. Overall, it is the assembly of these three elements and the additional new code that we added for the functionality of our device that makes it unique compared to any of the individual components.

Figure 1: Image of QuackTrack

Design Constraints and Requirements

When designing this device, there were a few constraints that we had to work under. First, the design had to be battery-powered, as the device was intended to be mobile. We settled on using a Lithium Ion battery early on, and the rest of the components were chosen using the battery we chose as a reference point. The main power draw, other than the ESP32, were the

motor drivers that would control and rotate the wheels of the device. The motor drivers we chose were low current draw, and we had to purchase an additional step up voltage regulator in order to provide the adequate voltage for the motor drivers. This voltage regulator also provided the different voltage levels that we needed to power all of the components of the device. The next constraint that we considered was the device size. We knew that we wanted the device to be small, but this changed other elements of the design. We needed to ensure that the microcontroller and the necessary components were small enough that the enclosure could fit them all while remaining relatively small. We also needed to take extra care with the camera, as it would not be operating from the same frame of reference as most image processing datasets had been trained. Furthermore, the image processing had to be relatively fast and reliable enough to almost always detect a person when they are in frame and give their location accurately. Since the camera would be in a somewhat strange position, there was also a benefit to being able to detect parts of a person, such as just a leg. The speed of the data stream was the final constraint that affected the development of our device. Since the device was expected to track users in real-time, it was necessary to have a video stream that could keep up with someone moving at a reasonable pace. This constraint imposed limitations on factors such as the quality of the video stream. Finally, while our device was under a cost constraint as well, we only ended up spending about half of the total budget allotted for this project.

Engineering Standards

In the development of the QuackTrack, adherence to various engineering standards ensures that the device is safe, reliable, and performs optimally. These standards encompass physical design, programming languages, digital design, communication protocols, peripherals and digital interfaces, artificial intelligence, security, testing and verification, and safety. Below, we delve into the relevant standards and their application in our project, emphasizing technical details.

Programming Language Standards
ISO/IEC 9899:2018 - Programming Languages - C

Application: The firmware for the ESP32 microcontroller is developed using C, adhering to the ISO/IEC 9899:2018 standard.
Relevance: This standard defines the syntax and semantics of the C programming language, ensuring our code is portable, reliable, and maintainable. Adhering to this standard facilitates compatibility with various compilers and platforms, which is crucial for embedded systems like the ESP32.
Reference: International Organization for Standardization (ISO)

Digital Design Standards

IEEE 1076-2008 (VHDL) - Standard VHDL Language Reference Manual

Application: Although VHDL is not directly used in our project, familiarity with this standard is essential for digital design and interfacing.
Relevance: VHDL (VHSIC Hardware Description Language) is used for describing the behavior and structure of electronic systems. Understanding this standard ensures robust design and debugging practices, particularly for complex digital systems and hardware description.
Reference: Institute of Electrical and Electronics Engineers (IEEE)

Communication Protocols
IEEE 802.11 - Wireless LAN (Wi-Fi) Standards

Application: The ESP32 module employs Wi-Fi capabilities in compliance with IEEE 802.11 standards for communication with the server.
Relevance: IEEE 802.11 standards define the protocols for implementing wireless local area network (WLAN) communications. Compliance ensures reliable, secure, and efficient wireless data transmission, which is critical for real-time video streaming and control commands in our system.
Reference: Institute of Electrical and Electronics Engineers (IEEE)

Peripherals and Digital Interfaces
I2C-bus specification and user manual (UM10204)

Application: The I2C protocol is utilized for potential expansion, allowing additional sensors or peripherals to interface with the ESP32. This functionality enhances the toy's capabilities and provides users with greater control and customization options.
Relevance: The I2C (Inter-Integrated Circuit) protocol facilitates efficient, low-speed communication between microcontrollers and peripherals. By adhering to this industry-standard, our product ensures reliable communication and seamless integration of additional components. This compliance also streamlines the development process and eases maintenance efforts.
Reference: NXP Semiconductors

Artificial Intelligence
ISO/IEC JTC 1/SC 42 - Artificial Intelligence

Application: The YOLO V3 algorithm is employed for real-time person detection, and OpenCV is utilized for image processing. These advanced technologies enable the QuackTrack to accurately identify individuals, providing users with an enhanced interactive experience.
Relevance: This standard provides guidelines for AI concepts, terminology, and frameworks. By adhering to these standards, our AI algorithms ensure responsible and ethical development of the

QuackTrack. This compliance promotes transparency, accountability, and trust in AI applications, fostering user acceptance and regulatory compliance.
Reference: International Organization for Standardization (ISO)

Broader Considerations

Societal Impact
If the QuackTrack design were to become widely used, it could have several positive impacts on society:

Enhanced Child Engagement: The QuackTrack, being an interactive and engaging toy, could significantly enhance playtime for children. Its ability to follow and interact with children encourages physical activity, creativity, and social interaction.

Parental Monitoring: By integrating a camera and streaming capability, the QuackTrack provides parents with a way to monitor their children remotely, offering peace of mind and an added layer of security.

Educational : The QuackTrack could inspire interest in STEM (Science, Technology, Engineering, and Mathematics) fields from an early age.

Market Innovation: The QuackTrack could push the boundaries of what is expected in a toy market, fostering innovation and competition, leading to more advanced and diverse products in the market.

Technical Community Impact
Advancement in Robotics: The QuackTrack could contribute to advancements in small-scale robotics, particularly in the areas of image processing and real-time object detection using lightweight hardware.

Open Source Contributions: If parts of the project, such as the software for image processing and control algorithms, were to be made open-source, it could aid other developers and researchers in their projects, fostering a collaborative technical community.

The main potential risk with this device concerns privacy. Since the deck is recording and streaming live video to a website, it is possible that the stream is intercepted by an outside party. To mitigate this risk, adding some form of confirmation to restrict stream access to only those in direct possession of the device could be used. While we were testing, we restricted access to the video feed by only streaming video to devices that were owned by approved users.

Design Description

a) System Overview

The QuackTrack project is a cutting-edge toy designed to resemble a duckling that follows a person around. It aims to provide both entertainment and educational value to young children through the integration of advanced technologies such as image processing, robotics, and web-based control interfaces. The overall system architecture consists of three primary components: the robotic device, the backend server, and the web-based frontend.

Robotic device
It is equipped with the following key components:

ESP32-CAM Module: This module integrates an ESP32 microcontroller with an onboard OV2640 camera, enabling image capture.
Micro Metal Gear Motors: Two motors are employed to enable the robot to move, allowing it to track and follow a person.
Motor Driver: A dual motor driver controls the power and direction of the motors based on commands received from the server.
Power Supply: A rechargeable lithium-ion battery provides portable power to the robot, ensuring extended operation time.
Control Loop: The robot operates on a control loop that processes image data to determine the direction and speed necessary to follow a person.

Backend Server
The backend server handles the heavy computational tasks, including image processing and communication management. It is hosted on an Amazon Web Services (AWS) EC2 instance and utilizes the following technologies:

Tornado Web Server: This server handles incoming and outgoing WebSocket connections, facilitating real-time communication between the robot and the frontend.
OpenCV Library: The OpenCV library is employed for image processing tasks, specifically for analyzing the images sent by the robot to detect the presence and position of a person.
YOLO V3 Algorithm: The YOLO (You Only Look Once) V3 algorithm is employed for real-time object detection, identifying and locating the person within the captured images.

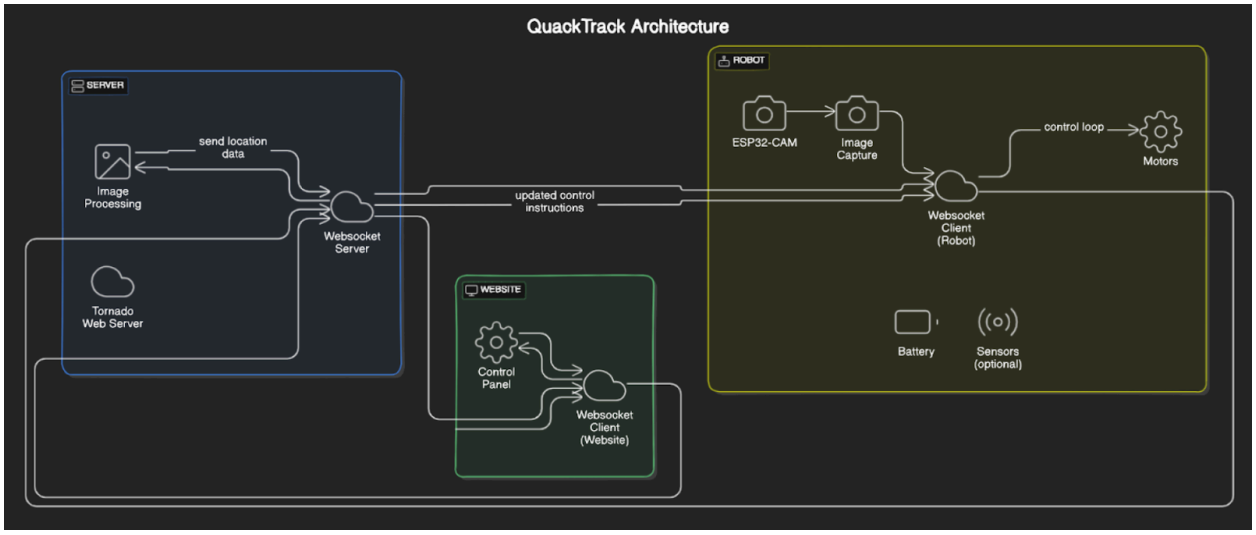

Figure 2: Block Diagram

b) Algorithm and Code

The process begins with the camera capturing an image and sending it to the web server using Websockets. Once the website has the image data, it uses the model trained on the YOLOv3 dataset to determine the presence and location of any people in the camera frame and draws a number of bounding boxes around them. Using those bounding boxes, the server determines the coordinates to send back to the ESP32 as the position of the person in frame [A]. Once the ESP32 has the location information, it uses a P controller to set the angular velocity that it should turn at to keep the person in frame, and another P controller to set the speed it should drive forward or backward [B]. This is then fed into another PI controller that uses an IMU to measure the angular velocity of the robot, and can control the motor PWM signal appropriately [C]. If no person is detected in the frame, the robot slowly decreases the velocity setpoints and waits for the next frame, allowing the robot to interpolate the motion of a person if one frame does not detect a person correctly.

c) 3D Printing

There were a few 3D printed components that we used while designing this product. For the first prototype, we utilized a 3D printed rectangular chassis as well as 3D printed wheels. The design of this initial prototype was very simple, but it was especially designed to fit the breadboard we were using and allow the motors and wheels to be easily attachable. The second prototype used the same set of wheels with a smaller rectangular chassis to fit a smaller breadboard. The smaller chassis also had a spot where we could slide in the ESP32-CAM system in order to attach the camera to the car. Again, this design was relatively simple. The enclosure for the final design was more involved. First, we found a 3D model of a duck online, then added a thickness to the exterior so that it could be printed. Furthermore, we added a mechanism so that

the top half of the duck could slide on and off of the bottom half. Finally, exterior holes for the camera and wheels were added as well. We used a multicolor print with orange, yellow, and black filament in order to print the enclosure that we used in the final design, allowing for the duck to have a well defined beak and eyes.

Final Product

a) Initial Goal vs Final Product

The minimum specifications that we established for ourselves at the outset of the quarter were as follows: a duck-shaped robot that can detect a user with a camera and follow them around, if they are the only person in frame and the room is empty. We were able to accomplish these goals that we set, but not much outside of that. We had a few ideas for additional goals if we achieved our minimum specifications in time, but we did not have enough time to implement them. This includes an external charging port, obstacle detection, a battery sensor, distinguishing between the user and other people in frame, and additional commands for certain movements. While we were not able to include any of our more complex goals, the device that we constructed does soundly fulfill the main specifications, and given more time many of the additional specifications could reasonably be included in the design as well.

b) Performance and Limitations

The video stream from the camera to the website display, including the amount of time necessary for the image processing, is reliably around 6.5 frames per second. This allows the car to move and adjust in time to not lose the person that it is trying to track. However, the movement speed on the car is not high enough to keep up with someone if they are sprinting, so it may struggle to track a child that is running around. Furthermore, the main limiting factor on the speed of the stream is the speed at which the website can complete the image processing task, meaning that the device that is running the server has a tangible effect on how well the device is able to perform.

Challenges Encountered

The first hurdle involved the image processing protocol for our device. We first tried OpenCV using the built in person detection model, but it required the full body to be in frame in order to detect the presence of a person, and even then was somewhat unreliable. This would not have worked for our design, as we were willing to sacrifice some speed in order to find a protocol that could accurately determine whether a person was in frame or not and return their location. The

next attempt involved a dataset called OpenPose, which was able to detect not only the presence of a person, but also where certain body parts were in frame. This was a very robust detection system, noticing even small amounts of a person in frame. However, translating this code into a form that would be usable for our project proved to be a challenge, and furthermore, the processing took far too long to analyze the image and produce an output. Eventually, we were able to find the YOLOv3 dataset, which trained a model that was able to detect people with relatively high reliability at speed fast enough for our device to perform the tasks that we had set out to complete.

The next major setback involved the transition from two microcontrollers to one. In our initial prototype the camera and the motors were controlled by two separate ESP32 microcontrollers. However, there were enough pins that it seemed possible to control the entire device using a single ESP32 microcontroller. When we tried to consolidate, however, we ran into some issues. The device would not reliably function, with elements either not connecting or failing to work at all. Eventually it was determined that a pin that was not listed as one to typically look out for, GPIO 16, had been causing the issue, as reconnecting the devices avoiding that particular pin allowed the device to be functional reliably.

The final challenge was an issue that had persisted over the entire quarter, but came to a head while completing the final prototype. The camera stream sometimes had a significant amount of latency, and occasionally the speed of the stream would become drastically smaller. The stream's fastest speed was around 8 fps, but seemingly randomly it would drop to as low as 0.1 fps or lower. We found that physically touching the device seemed to improve the speed of the stream, so initially we tried clamping the ESP32 to mimic the effect of squeezing it physically. When this did not improve the stream speed, we reasoned that perhaps while touching the device we were serving as an antenna that allowed for data to flow more easily, so we purchased a stronger antenna to use in hopes of mimicking the effect once again. However, this did not improve the reliability of the speed of the image stream either. Eventually, we determined that the clock signal for the camera was interfering with the wifi signal because of a hardware issue in the ESP32-CAM board. By reducing the clock signal, we were able to achieve a data stream of around 6.5 fps reliably, which was fast enough for our device to be functional.

Planning and Organization

d) Gantt Chart

	Start Date	End Date	Timeline	Status
ESD2	Mar 26, 2024	Jun 7, 2024		
Milestone 1	Mar 26, 2024	Apr 5, 2024		Complete
Define project goals	Mar 26, 2024	Apr 1, 2024		Complete
Research components	Apr 2, 2024	Apr 4, 2024		Complete
Prepare order	Apr 4, 2024	Apr 5, 2024		Complete
Milestone 2	Apr 6, 2024	Apr 21, 2024		Complete
Test each component with minimal code	Apr 12, 2024	Apr 16, 2024		Complete
Put together breakout prototype	Apr 14, 2024	Apr 17, 2024		Complete
Camera functionality	Apr 12, 2024	Apr 21, 2024		Complete
Make prototype car	Apr 12, 2024	Apr 21, 2024		Complete
Interface all components and write minimal code to show functionality	Apr 18, 2024	Apr 21, 2024		Complete
Milestone 3	Apr 22, 2024	May 5, 2024		Complete
Implement image processing to track users	Apr 22, 2024	Apr 26, 2024		Complete
Translate the image processing result into commands for the vehicle	Apr 26, 2024	Apr 30, 2024		Complete
Combine all hardware, firmware, and software for full funtionality	Apr 29, 2024	May 5, 2024		Complete
Fix any bugs/issues with the assembled prototype	May 1, 2024	May 5, 2024		Complete
Milestone 4	May 6, 2024	May 21, 2024		Complete
Fix inefficiencies in prototype	May 6, 2024	May 10, 2024		Complete
Research potential additional functionalities	May 11, 2024	May 13, 2024		Complete
Design PCB	May 11, 2024	May 17, 2024		Complete
Research and order new parts	May 11, 2024	May 16, 2024		Complete
Complete any remaining work for new version	May 15, 2024	May 19, 2024		Complete
Milestone 5	May 20, 2024	Jun 2, 2024		Complete
Update Website	May 20, 2024	May 31, 2024		Complete
Improve code for device	May 22, 2024	May 30, 2024		Complete
Design 3D printed enclosure	May 23, 2024	May 30, 2024		Complete
Assemble final prototype	May 28, 2024	Jun 2, 2024		Complete
Milestone 6	May 27, 2024	Jun 7, 2024		Complete
Write report (shared sections)	May 28, 2024	Jun 7, 2024		Complete
Prepare online project	May 28, 2024	Jun 5, 2024		Complete
Write report (individual sections)	Jun 2, 2024	Jun 7, 2024		Complete
		Burndown		

Figure 3: Gantt Chart

Throughout the term our group utilized the chart above to track our progress, mostly by ensuring that we allocated enough time for all of our tasks. The chart was not usually used as a strict guideline that we followed from week to week, but more of a sanity check ensuring that we would not become too overwhelmed for any particular period of time. It was also a useful metric for being able to have a visualization of the ground that we had made and the ground we had left to cover over the course of the quarter. The perspective granted by being able to observe these factors allowed the process to feel less daunting on a week to week basis.

e) Bill of Materials

Item Desc.	Mfg. Part #	Unit Price	1000 Unit Price	Quantity	URL	In Final Design	Total Unit Price	Total Bulk Price	Total Order Price
Motors	N20 Motor 1000	$10.990		1	https://www.ama	No	$0.00	$0.00	$68.22
Motor Driver	DRV8835	$4.950	$3.850	1	https://www.polo	No			
3.3V regulator	S7V8F3	$7.950	$6.190	1	https://www.polo	No			
Li ion Battery	DTP603443	$5.500		1	https://www.spar	No			
Li ion charger	MCP73831T	$9.950		1	https://www.spar	No			
JST Connector		$1.050	$0.950	1	https://www.spar	No			
6v regulator	U3V40F6	$7.850	$6.190	1	https://www.polo	No			
ESP32-CAM		$19.990		0	https://www.ama	No	Free from Ilya ??		
Ultrasonic sensor		$9.990	$7.992	1	https://www.ama	No			
Voltage sensor and current senso		$9.990	$1.998	1	https://www.ama	No			

Table 1: Order 1

Item Desc.	Mfg. Part #	Unit Price	1000 Unit Price	Quantity	URL	In Final Design	Total Unit Price	Total Bulk Price	Total Order Price
Motors	N20 Motor 1000	$10.990		1	https://www.ama	No	$0.00	$0.00	$83.17
Motor Driver	DRV8835	$4.950	$3.850	1	https://www.polo	No			
3.3V regulator	S7V8F3	$7.950	$6.190	1	https://www.polo	No			
Li ion Battery	DTP603443	$5.500		1	https://www.spar	No			
Li ion charger	MCP73831T	$9.950		1	https://www.spar	No			
6v regulator	U3V40F6	$7.850	$6.190	1	https://www.polo	No			
ESP32-CAM		$19.990		1	https://www.ama	No			
OV2640 w/ long cable		$10.990		1	https://www.ama	No	**Color: 80°-2 million**		
Wifi Antenna		$2.500	$2.500	2	https://www.adaf	No			

Table 2: Order 2

Item Desc.	Mfg. Part #	Unit Price	1000 Unit Price	Quantity	URL	In Final Design	Total Unit Price	Total Bulk Price
Motors	N20 Motor 1000	$10.990		2	https://www.ama	Yes	$66.18	$61.66
Motor Driver	DRV8835	$4.950	$3.850	2	https://www.polo	Yes		
3.3V regulator	S7V8F3	$7.950	$6.190	2	https://www.polo	Yes		
Li ion Battery	DTP603443	$5.500		2	https://www.spar	Yes		
Li ion charger	MCP73831T	$9.950		2	https://www.spar	Yes		
JST Connector		$1.050	$0.950	2	https://www.spar	No		
6v regulator	U3V40F6	$7.850	$6.190	2	https://www.polo	Yes		
ESP32-CAM		$19.990		1	https://www.ama	Yes	this is a 2 pack, only used 1	
Ultrasonic sensor		$9.990	$7.992	2	https://www.ama	No		
Voltage sensor and current senso		$9.990	$1.998	2	https://www.ama	No		
OV2640 w/ long cable		$10.990		1	https://www.ama	No	**Color: 80°-2 million**	
Wifi Antenna		$2.500	$2.500	2	https://www.adaf	Yes		
MPU6050	MPU6050	6.49	6.49	1	https://www.ama	Yes		

Table 3: Bill of Materials and Unit Price

The Bill of Materials documents the parts that we purchased over the course of the design process. The purpose of most of the parts is simple, motors and motor drivers to operate the wheels, a battery to power the device, and a voltage regulator to provide the necessary voltage levels. The voltage regulator that we purchased included a step up circuit in order to provide a higher voltage for the motors. The ultrasonic sensor was purchased with the intent of an obstacle detection protocol for the device, but that was never implemented. Similarly, we purchased a voltage and current sensor that was intended to be used as a measure of the remaining battery level, but that was not implemented either. The Wifi antenna was purchased as a possible solution to a slow streaming speed and the camera with a long cable was purchased to give more flexibility in the positioning of the ESP32 in the enclosure. Many of the same pieces were

ordered again to avoid needing to disassemble the prototype that we had working when we placed the second order.

f) Communication Among Team Members

Our team met about once every week in order to discuss the required tasks for upcoming milestones. We clarified often that if anyone was struggling to complete any portion of the work that was expected of them that they should share that information with the rest of the group so as to not have the group as a whole fall too far behind. This strategy was implemented a few times during particularly difficult steps in the process, such as during the search for an image processing strategy that satisfied the needs for our specific project. We also typically held meetings after we were already in the same location for some other reason, such as the biweekly meetings with Professor Mikkelson. It was important that everyone in our team knew where we stood in the design process and what had changed in the time since we had previously met, so our team made sure that no one was left in the dark about our project.

g) Splitting Tasks Among Team Members

Work was primarily divided by clarifying all of the tasks to finish within the week and then asking each member if there was a particular task that they felt the most comfortable with. As the quarter went on, the roles in the team were more established, and tasks were typically given out in accordance to what each group member had demonstrated to be the work that they gravitated towards. Cole predominantly worked on the code and the mechanics of the device, as he had constructed a similar vehicle in a previous class. Noah predominantly did the written requirements, completing all but the first progress report along with heading the presentations and interim reports. Evangeline assisted whichever side of the project had a larger workload from week to week, providing extra support so no singular member got too stressed. In weeks in which developmental design or written portions of the project were few, Cole and Noah would pivot away from their predominant work in order to more efficiently complete the remaining tasks for the week. For instance, Noah constructed a 3D model for one of the prototypes and helped in the search for the image processing infrastructure, and Cole was given portions of the reports and presentations. Overall, the team was set up so that there was no confusion over who was doing which task, and group meetings always ended with a clear establishment of each group members' expectations and responsibilities for the week.

Market Research

Our product was intended to be used as a consumer good, so we researched the likely consumer base that we would be targeting with our design. In looking for similar designs in the market space, we found no devices with the exact functionality that we were proposing, and very

few that acted similarly. The most similar device was a Baby Yoda toy, which you could walk away from and then press a button on a controller to have it come find you [1]. Since our device would not require a controller, it presented a new idea into the space. In terms of the market size we were going after our expected consumer base included parents of children around the age of 5. According to census data, in 2022 there were approximately 3,500 children under the age of 5 in the Evanston area [2]. As part of our research, we conducted interviews with both members of the community as well as people experienced with design.

a) Interviews With Non-experts

The first interviews that were conducted concerning our design were with people unfamiliar with the engineering fields or with the design process. In order to ensure relevance to our project, we attempted to focus on individuals who had young children or were experienced in interacting with young children. We asked a number of questions, mostly focused on features that would be relevant if this product were brought to market, as the potential consumers being interviewed would be able to speak more accurately on their own purchasing habits than on the functionality of the device. Many of the features that these interviewees expected to be included we had already planned for, such as durability, manual control, and small, lightweight size. They gave us an age range similar to what we were expecting, about 4-9 years old. However, they did bring up a few elements that could be included that we hadn't considered, such as the privacy concerns about the video stream and the possibility to add some sort of educational functionality to the device.

b) Interviews With Experts

In the interviews conducted with the technical experts, we also asked about what features would be expected in the device that we described, as they had experience and would know what the baseline for a project such as ours should be. They provided some interesting insights, such as a protocol for when the device did not detect someone in frame or social connectivity. When asked about our specific design, they confirmed that the choices we had made and the parts we were using were sufficient for the goals of this quarter. They also emphasized the need for security if this product were to go to market. While we weren't able to implement all of their suggestions, we were able to confirm that we were on the right path and finish the construction with confidence.

Conclusion

Our team gained invaluable expertise in system integration, including the use of motors, cameras, and microcontrollers. We honed our skills in implementing the YOLOv3 dataset for real-time object detection and mastered the use of Websockets for seamless communication between the website and ESP32 microcontrollers. Troubleshooting video stream issues reinforced the importance of identifying root causes and refining our system architecture for enhanced reliability.

Given another opportunity, we would initiate component integration earlier to identify compatibility issues at an earlier stage and optimize hardware selection to avoid performance bottlenecks. More meticulous planning and documentation would streamline development processes and improve team coordination. Proactively identifying and addressing potential technical risks and iteratively incorporating user feedback would enhance the design process.

With additional time and resources, we would focus on refining image processing algorithms for improved accuracy and speed, incorporate advanced features such as obstacle detection and voice commands, and optimize power management for prolonged operation. Thorough market research and extensive user testing would guide the product roadmap, helping us identify potential market segments and customize our offering accordingly. Rigorous testing would ensure the product's durability and reliability in diverse environments. These steps would transform our prototype into a polished, market-ready product with enhanced functionality and user experience.

References

[1] https://mymodernmet.com/baby-yoda-real-moves-toy/
[2]https://www.census.gov/quickfacts/fact/table/evanstoncityillinois/AGE135222

Appendices

[A]: Image processing code to detect and locate humans

```python
# detects a person in the image and draws a rectangle around them
def detect_person(image):
    # Get image shape
    height, width, channels = image.shape

    # Create blob and do forward pass
    blob = cv2.dnn.blobFromImage(
        image, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    # average all the boxes and draw a dot in the center
    x_avg = 0
    y_avg = 0
    top_avg = 0
    count = 0

    # Information for each object detected
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5 and class_id == 0:  # Class ID 0 is human
                # Object detected
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                top_avg += center_y
                x_avg += center_x
                y_avg += center_y
                count += 1

                cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 255), 2)

    # Draw a dot in the center
    if count > 0:
        x_avg = int(x_avg / count)
        y_avg = int(y_avg / count)
        top_avg = int(top_avg / count)
        cv2.circle(image, (x_avg, y_avg), 10, (0, 0, 255), -1)


    return image, x_avg, top_avg, count
```

[B] Person Detection Control loop

```
void camera_PID_task(void *parameter)
{
    float I = 0;
    float prevError = 0;
    int count = 0;

    int setPoint = 120;

    while (1)
    {
        // wait for new setpoint
        while (!_global_camera_flag || !_global_mode)
        {
            delay(10);
        }

        _global_camera_flag = 0;

        int error = setPoint - _global_camera_x;

        I += error;
        if (I > camera_I_max)
        {
            I = camera_I_max;
        }
        else if (I < -camera_I_max)
        {
            I = -camera_I_max;
        }

        float result = camera_Kp * error + camera_Ki * I + camera_Kd * (error - prevError);
        prevError = error;

        // set speed based on threshold of y axis
        int speed = 0;
        if (_global_camera_y > Y_THRESHOLD + Y_TOLERANCE)
        {
            speed = -90;
        }
        else if (_global_camera_y < Y_THRESHOLD - Y_TOLERANCE)
        {
            speed = 90;
        }

        // set the imu setpoint
        set_imu_setpoint(result, speed);
        // set_imu_setpoint(result, 0);

        Serial.printf(">Camera:%d\n>Error:%d\n>Result:%f\n", _global_camera_x, error, result);
        Serial.printf(">_global_camera_y:%d\n", _global_camera_y);
    }
}
```

[C] Angular Velocity Control Loop

```
while (1)
{

    // read the gyro data
    Wire.beginTransmission(MPU);
    Wire.write(0x43); // starting with register 0x43 (GYRO_XOUT_H)
    Wire.endTransmission(false);
    Wire.requestFrom(MPU, 6);
    int16_t x = Wire.read() << 8 | Wire.read();
    int16_t y = Wire.read() << 8 | Wire.read();
    int16_t z = Wire.read() << 8 | Wire.read();

    // convert gyro data to rad/s
    float gyroZ = z * 250.0 / 32768.0 * 3.14159 / 180.0;

    // use PID controller to keep angular velocity around z axis at a set point

    // Set the speed of the motors based on the angular velocity around the z axis
    float angle_setpoint = _global_imu_angle_setpoint;

    // float error = setPoint - g.gyro.z;
    float error = angle_setpoint - gyroZ;

    // Calculate the PID terms

    I += error;
    if (I > imu_I_max)
    {
        I = imu_I_max;
    }
    else if (I < -imu_I_max)
    {
        I = -imu_I_max;
    }

    float turnResult = imu_Kp * error + imu_Ki * I + imu_Kd * (error - prevError);
    prevError = error;

    // Calculate the speed of the motors
    int result_R = turnResult + _global_imu_speed_setpoint;
    int result_L = -turnResult + _global_imu_speed_setpoint;

    // Set the speed of the motors

    motorL.setSpeed(result_L);
    motorR.setSpeed(result_R);

    // Serial.printf(">Setpoint: %f\n", angle_setpoint);
    // Serial.printf(">Gyro: %f\n", gyroZ);
    // Serial.printf(">Error: %f\n", error);
    // // Serial.printf(">I: %f\n", I);
    // Serial.printf(">Result_R: %d\n", result_R);
    // Serial.printf(">Result_L: %d\n", result_L);

    delay(5);
}
```

Class Feedback

Yes, I certainly learned a great deal in this course. The hands-on experience with designing and constructing a functional robot using the ESP32 microcontroller was invaluable. The practical application of concepts such as image processing, motor control, and microcontroller communication greatly enhanced my understanding.I thoroughly enjoyed working with the ESP32 due to its robust capabilities and the variety of functionalities it offers. It was an excellent choice for this project. This project was both challenging and rewarding, requiring a combination of hardware and software skills and providing a comprehensive learning experience. The class structure effectively supported the development of a complex project, and the hands-on approach was particularly effective. Overall, I am very satisfied with the knowledge and skills I gained through this course. One important consideration is the significant amount of time and effort invested by the team to troubleshoot and resolve issues. Additionally, the collaborative nature of the project helped develop not only technical skills but also teamwork and project management abilities. These aspects were crucial in achieving our project goals. Another important aspect is the versatility of the ESP32 microcontroller. Its ability to interface with various sensors and actuators, as well as its built-in Wi-Fi and Bluetooth capabilities, made it an ideal choice for this project. This flexibility allowed us to explore and implement a variety of features, further enhancing our learning experience.