University of Southern Indiana
Pott College of Science, Engineering, and Education
Engineering Department

8600 University Boulevard
Evansville, Indiana 47712

CLASS PET ROBOT

Nathan Dawkins and Hagan Hollinger
ENGR 491 – Senior Design
Spring 2024

Approved by: ______________________________
Faculty Advisor: Ron Diersing, Ph.D. Date

Approved by: ______________________________
Department Chair: Paul Kuban, Ph.D. Date

ACKNOWLEDGEMENTS

In this section, we would like to acknowledge:

- Dr. Ron Diersing, Associate Professor of Electrical Engineering, for his guidance as faculty advisor.
- Dr. Ryan Integlia, Associate Professor of Electrical Engineering, for his support during ECE 471.
- Paige Ausenbaugh, Elementary and Special Education student, for her guidance and assistance in developing the educational resources for the project.
- Kyle Martin, Senior M.F.E.T. Student, for his assistance with 3D modeling.
- Dr. Jeff Thomas, Professor of Education, for faculty guidance for education resources.
- Allison Grabert, SwISTEM Director.
- Jamie Curry, Administrative Assistant.
- Dr. Paul Kuban, Engineering Department Chair.

ABSTRACT

The purpose of the Class Pet Robot is to expose more students to Science, Technology, Engineering, and Mathematics (S.T.E.M.) in middle school. This exposure is needed because not enough students are pursuing careers in S.T.E.M. when graduating high school. The primary benefits of the Class Pet Robot are its pet-like design, affordable price, and beginner-friendly programming. The pet-like design implements behaviors like tail-wagging and barking to engage more students. The affordable price by using low-cost and recyclable parts helps the project fit into school budgets. The beginner-friendly programming interface was designed to mimic Scratch, a language known for its ease-of-use. Additionally, the robot fits into Indiana Education Standards, which makes it easier for teachers to introduce it as part of their curriculum. When complete, the Class Pet Robot will make S.T.E.M. education more affordable, engaging, and accessible for Indiana school systems.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

CLASS PET ROBOT

1 INTRODUCTION

1.1 BACKGROUND

In the Evansville area, there is an array of valuable resources for Science, Technology, Engineering, and Mathematics (S.T.E.M.) education, such as the Southern Indiana Career and Technical Center or the ABET-accredited B.S. in Electrical Engineering program at the University of Southern Indiana (USI) [1]. These education resources are a fantastic funnel into S.T.E.M. careers later in life, which is the fastest growing career field in the United States [2]. However, these resources are underutilized because not enough people are interested in them [3]. This issue can be traced back to grade school. Many schools have little room in their budgets and cannot afford robotics kits. Even when they can afford them, the robotics kits may have issues. Sometimes, students spend so much time trying to build the kit that they barely get to learn any programming. Other times, the kit of loose parts may simply not grab the student's attention [1]. Therefore, a robotics opportunity for school with a greater focus on affordability, ease of use, and engagement is needed.

1.2 CURRENT SOLUTIONS

1.2.1 Lego Education

Lego Education is a robotics kit which lets students use Lego bricks and digital tools to solve problems creatively and to think critically to excel while working with others. The positive skills students develop are collaboration, hands on learning, and creative thinking. The main limiting factor of Lego Education kits is the price of one kit is $399.99. Additionally, there is a lack of resources for guiding teachers on how to teach students how to use the kits [4]. The class pet robot will provide similar fundamental skills to the students as Lego Education does, also have a reasonable price to be accessible for all schools and be plenty durable for many years to come.

Figure 1 - LEGO® Education SPIKE™ Prime Set [5]

1.2.2 Duckietown

Duckietown is a project developed at MIT to easily teach AI robotics to college students. The project is used in classes at many different undergraduate universities. The kit provides the robots as well as software, lesson plans, and anything else the professor may need to teach a class about AI robotics [6]. Duckietown is a great introduction to robotics for middle school students. This is because Duckietown's teaches concepts like Robotic systems, Detection, Principles of Vision, Modeling and more. With that being said, the class pet robot project will take inspiration from Duckietown and deliver the same set of ready-to-teach tools.

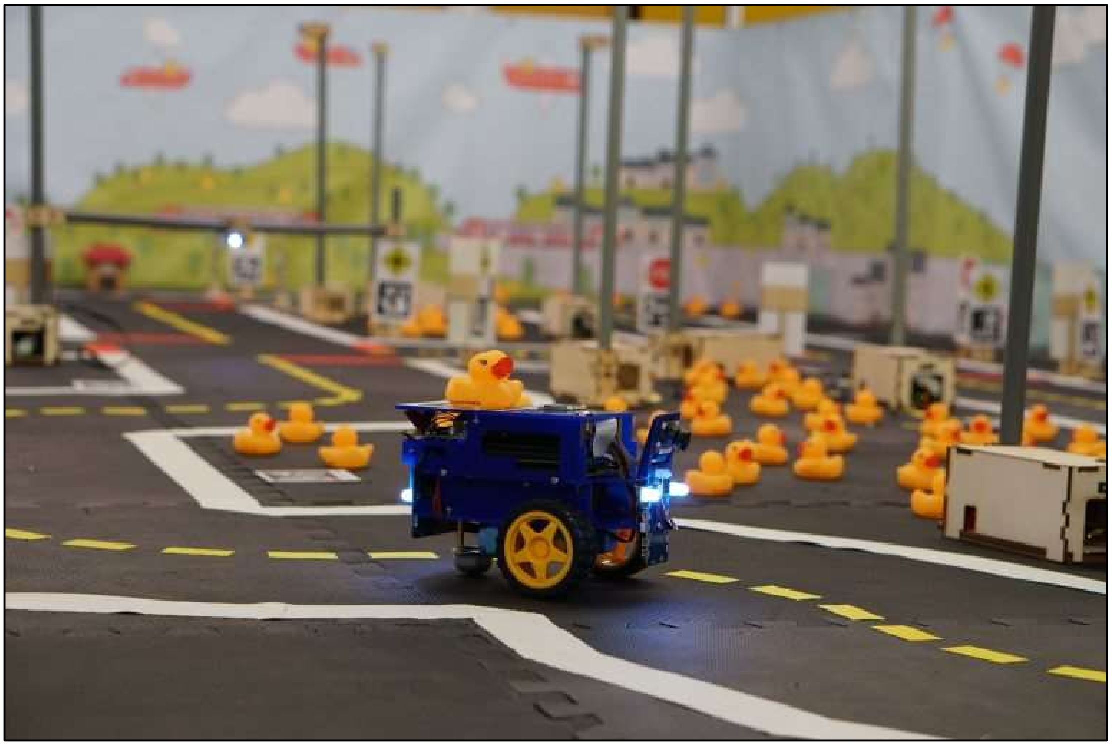


Figure 2 – Duckiebot Driving in Duckietown [7]

1.2.3 Code.org

Code.org is another highly successful and popular tool for teaching programming. The website offers a wide variety of lessons for various languages and ages [8]. However, a common issue for uninterested students is the fact that everything happens on one screen. The class pet robot addresses this problem by providing physical way for students to see their code affect a real-life robot.

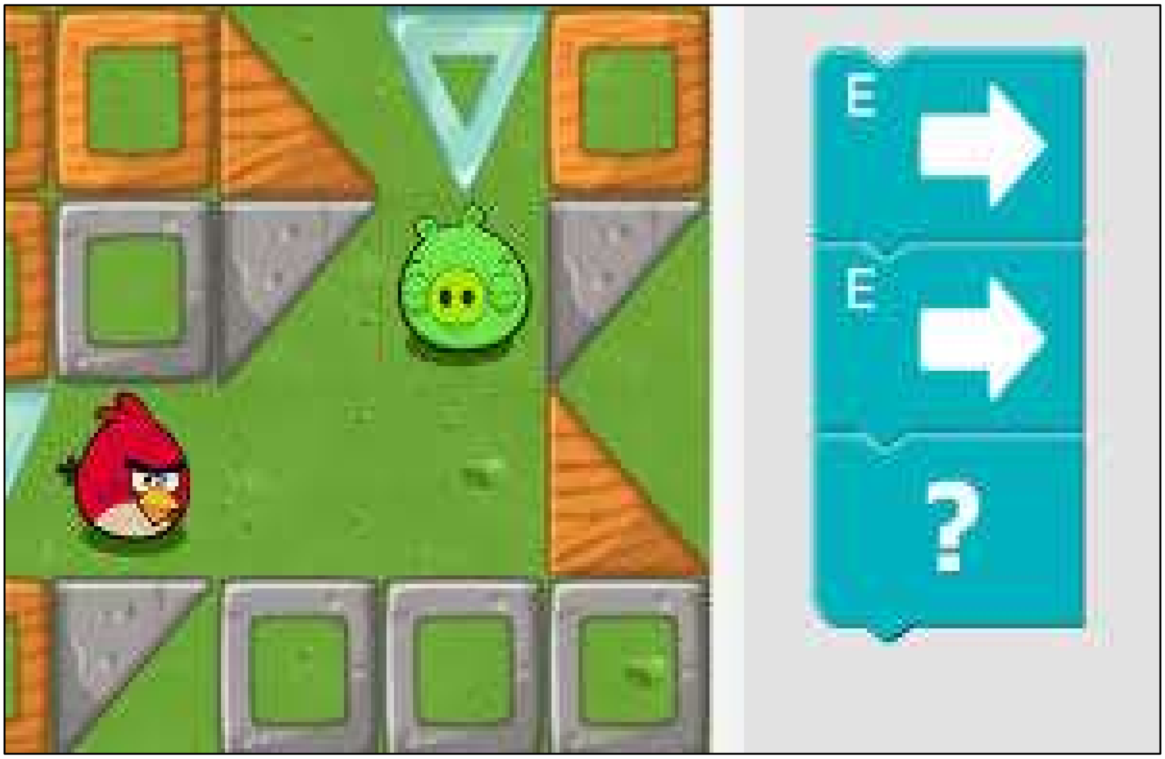
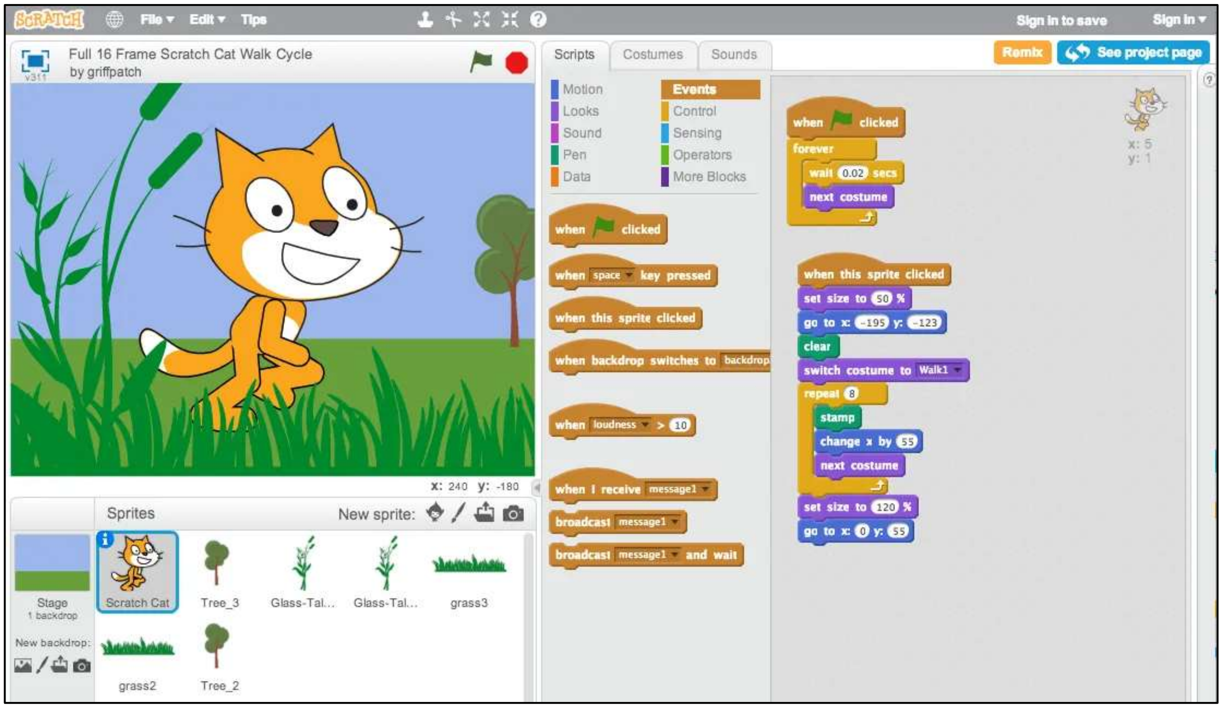


Figure 3 – Code.org Debugging Course [9]

1.2.4 Scratch

Scratch programming, developed by MIT, is a beginner-friendly platform for creating interactive stories, games, and animations using block-based programming. It simplifies coding concepts by focusing on logic and creativity without syntax. Students learn fundamental coding concepts such as loops and variables in an engaging way. Scratch also encourages collaboration within its online community. The class pet robot takes heavy inspiration from Scratch as one of the best beginner-friendly coding programs.



```
when clicked
forever
    wait 0.02 secs
    next costume

when this sprite clicked
set size to 50 %
go to x: -195 y: -123
clear
switch costume to Walk1
repeat 8
    stamp
    change x by 55
    next costume
set size to 120 %
go to x: 0 y: 55
```

Figure 4 – Scratch Programming Demo [10]

2 OBJECTIVE STATEMENT

The objective of the class pet robot project is to create a programmable, affordable, and engaging robot for classroom use to improve ECE exposure in grade school.

2.1 Requirements and Constraints

2.1.1 Education Requirements

In the United States, all states have education standards that describe what topics should be taught in classrooms. Teachers can venture outside the standards if they wish, but they must cover the state's standards first. By the time teachers finish all of the content required by the state, the school year is or almost is over [11]. So, for the class pet robot to easily be introduced to a classroom, it must align with one or more state standard. The Indiana standard that aligns most with the class pet robot is Computer Science 6-8.PA.1: "Design and iteratively develop programs that combine the following: sequencing, looping (including nested loops), conditionals (including compound conditionals), expressions, variables, functions, and parameters." [12] If the project can be matched to more standards, it will help its likelihood of being introduced to the classroom, but the project will focus on this standard first.

In addition to state standards, there are other education requirements for the project:

1. The project must be accompanied with training, curriculum, and/or manuals to assist teachers who are unfamiliar with the project and/or associated topics.
2. The project must be affordable so that schools can easily purchase the robots.
3. The project should be accessible to students with disabilities.
4. The project's user interface shall be simple and intuitive.

2.1.2 Technical Constraints

The following technical constraints apply to the class pet robot project:

1. All Python3 code must conform to Python Enhancement Proposals (PEP) formatting standards to make the software easy to edit and read for future contributors.
2. The robot must implement an external microcontroller to allow room for adding additional components.
3. The robot programming process must use Universal Serial Bus (USB) compliant cables.

2.2 Specific Aims

The specific aims of the class pet robot project are:

1. Create a block style programming interface capable of writing code with all features listed in the Computer Science 6-8.PA.1 Indiana State Standard.
2. Make the robot mimic pet behaviors by incorporating movement, sounds, tail-wagging, and chin scratching.
3. Implement at least 1 additional sensor and 1 additional actuator to create a framework for future robot expansions and modifications.

3 PROJECT DESIGN

3.1 SYSTEM OVERVIEW

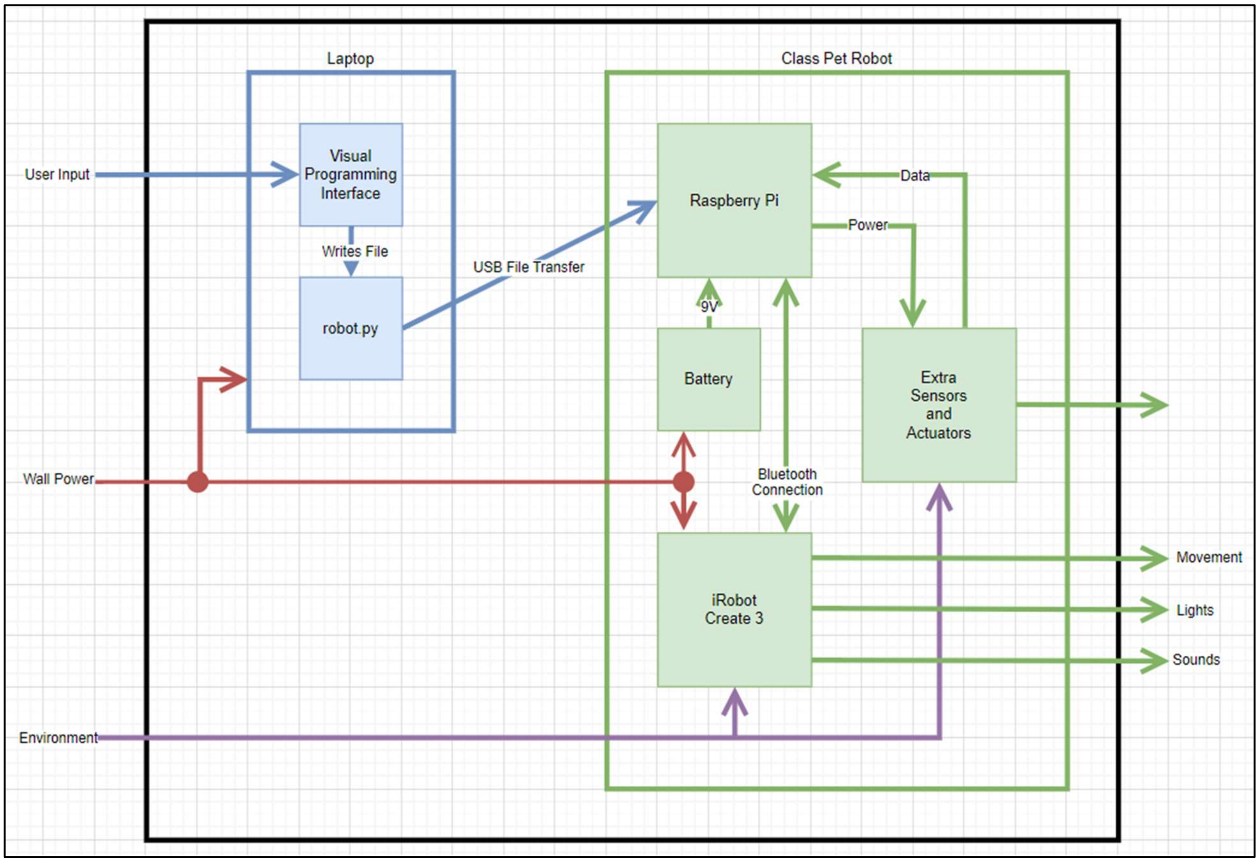


Figure 5 – System Block Diagram

Figure 5 shows the functionality of the class pet robot as a block diagram. The project has two main subsystems: the programming tool (in blue) and the actual robot (in green). The main process begins with a user opening the visual programming interface and making a block program. Once their program is finished, the user hits the program button. This starts a process that compiles their program into an actual Python program and transfers the file to the microcontroller (a Raspberry Pi) of the robot over a USB cable. Then, the program it received from the programmer is run to control the main robot and the additional modules.

The design process began with the robot system. First, a core robot was selected. Next, a microcontroller able to interact with the core was selected. Finally, several modules compatible with the microcontroller were selected to implement additional functionality. Once the robot was designed, the programming interface was next. Designs for both systems are discussed in greater detail in the following sections.

3.2 ROBOT SYSTEM

The pet robot is the complex centerpiece of the project. The Create 3 (Figure 6) serves as the core of the robot. The Create 3 is a Roomba without a vacuum inside, with more design capabilities than the average Roomba. This robot was selected for its availability and the potential to use donated and recycled unused Roombas that have Bluetooth capabilities. The robot has bumper sensors, IR sensors, and encoded motorized wheels-built in. iRobot has released an article on All Roomba Features and Compatibility within has a list on all Roomba models that have Bluetooth capabilities [17] to help know which models can be recycled/donated for the class pet robot project. A Raspberry Pi 3B+ (Figure 7) functions as the brain of a robot. The Pi connects to the Roomba via Bluetooth connection, so that's why a Roomba with Bluetooth capabilities is necessary. To add more functionality options for students, various sensors and actuators will be added. Figure 5 shows how it interacts with the rest of the robot. The Raspberry Pi 3B+ was selected with the teams experience in mind and the flexibility the Pi offers.

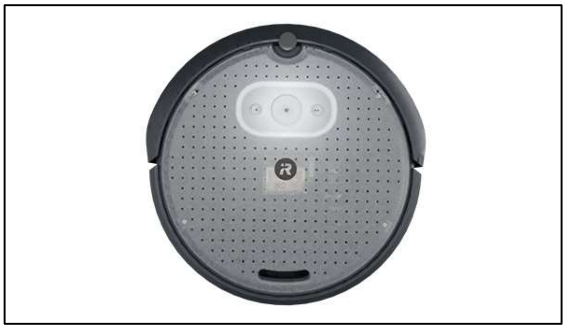


Figure 6 – Create 3 [13]

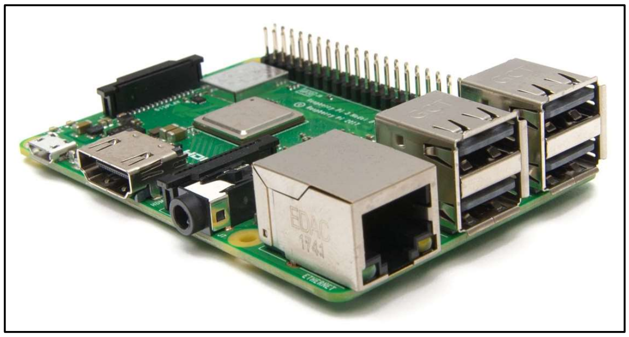


Figure 7 – Raspberry Pi 3B+ [14]

3.2.1 Robotic Base – iRobot Create 3

A robotic base (body) was needed for the class pet robot, and two options were considered, the iRobot Create 2 and Create 3. After careful evaluation, the Create 3 was selected for a couple of reasons. Firstly, it offers upgraded hardware compared to the Create 2, providing enhanced performance and capabilities. Additionally, the Create 3 offers greater customizability, allowing for more flexibility for the project's needs. Furthermore, the availability of free online example code, and libraries from the developers for the Create 3 provided valuable for the beginning development and testing phases of the project.

3.2.2 MCU – Raspberry Pi 3 B+

The brain of the project, the microcontroller unit (MCU) is needed to integrate additional sensors and actuators into the class pet robot, enhancing its pet-like characteristics and engagement for students. After deciding between various options available online, the Raspberry Pi 3 B+ was chosen. The team's familiarity with the Raspberry Pi platform, along with the extensive active community surrounding it, ensures support for the project's development and mostly troubleshooting. Additionally, the Raspberry Pi 3 B+ was preferred over newer models due to its lower price and older models for capabilities, despite being overkill for current project requirements.

3.2.3 Additional Components

With the additional sensors and actuators connected to the Raspberry Pi and being run in Python, the team aims to enhance the class pet robot's abilities by having it wag its tail, display messages, and move freely around the classroom environment. When considering options for the servo motor, the team chose a 3.7V ~ 6V micro servo motor due to its compact size, precise control over rotation speed, and its compatibility with the Raspberry Pi GPIO pins. Also, the servo motor is within range of the 5V supply voltage from the Pi. This was chosen over standard-sized servo motors for its suitability for small-scale applications, despite the potential limitation in torque. Also, the micro servo motor was selected for its low cost and simple function to move back and forth, making it ideal for implementing the tail-wagging behavior. For the appearance of the tail, a simple design using a straw and a cotton ball was chosen over Styrofoam, but later in the future, a detailed Styrofoam tail would be preferred. The use of simple materials like straws and cotton balls not only reduces costs but also gives more of a pet-like feel to the class pet robot overall.

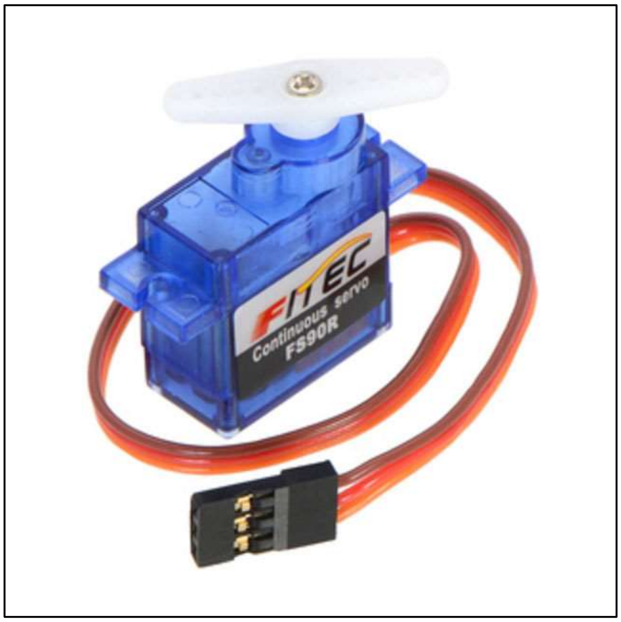


Figure 8 - Micro Servo Motor [15]

Now for a text display, the team picked a 16x2 character, 5V, LCD for being small, satisfying the Pi's supply voltage, and being compatible with the Raspberry Pi GPIO pins. While OLED and TFT color displays have a higher resolution and advanced graphics capabilities, the 16x2 character LCD was chosen for its plethora of online tutorials and for displaying basic text to the students and teachers. Also, the LCD was chosen for its cheap price and simple design. The LCD ensures that students have another way to interact with the robot and see programmed text displayed, overall boosting their engagement with the robot.

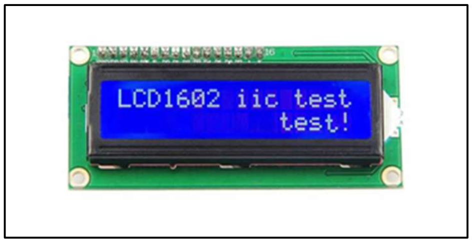


Figure 9 - Liquid Crystal Display (LCD) [16]

Lastly, an analog joystick sensor was chosen for the movement input. But before that, an ADC chip was needed to wire with the Raspberry Pi, due to the 3B+ not having ADC pins for the Joystick's analog input. A Freenove, 8pin, ADC chip was used for converting the Joystick to digital output for the Pi to read and has extra pins available for future additions. The joystick sensor offers a simple and touch-sensitive intuitive control for movement capabilities and has a max operating range of 5V which the Pi does not exceed making the Joystick compatible. Additionally, the joystick sensor was selected for its familiarity with students who have played video games, providing a comfortable and intuitive interface for controlling the robot's movement. Moreover, even for students who may not be familiar with joystick sensors, the simple design and operation make it easy to understand and use, ensuring accessibility in the learning experience.

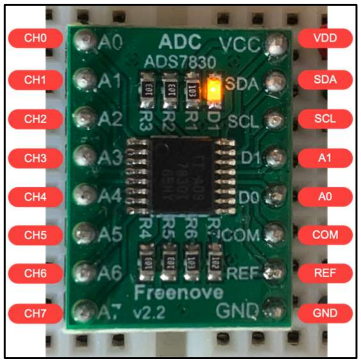


Figure 10 - Freenove ADC Chip [17]

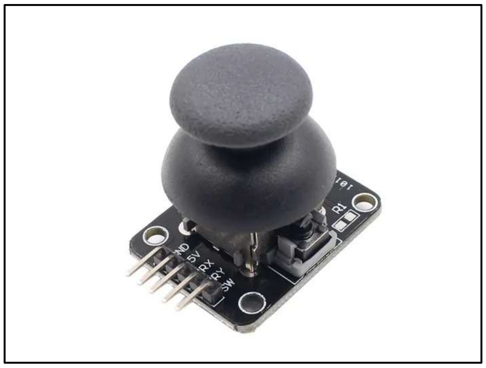


Figure 11 - Analog Joystick Sensor [18]

Overall, these mindful considerations ensure that the class pet robot's additional components are chosen for their performance, compatibility, simple to integrate, and cost. By prioritizing cost-effective components and familiar interfaces, the class pet robot showcases a well-rounded engagement and learning for students of all backgrounds and skill levels.

3.3 Programming System

The programming system is entirely software. As far as the user is concerned, it is a single application they run to program their robot. However, the system is much more complex than that. The entire programming system is written in Python for a multitude of reasons. First, the team member working on the system, Hagan, is very familiar with the language. Second, the Software Development Kit (SDK) used to interact with the Create 3 is in Python. Finally, Python's whitespace-based syntax is similar to the block programming's syntax. For example, an indent in the block program is translated as a tab indent in Python. In a language like C, there would need to be complicated bracket handling instead.

The Python code for the project is also written in an Object-Oriented Programming (OOP) style. Python offers four main programming styles: imperative, procedural, functional, and OOP. OOP is best used when a program contains a lot of components that use very similar code [19]. In the project's program, the easiest example of this is the code blocks. They all use a lot of lines of code, but each block is slightly different. So, the OOP approach allows a much easier way of creating these blocks.

3.3.1 User Interface

The first part of the programming interface is the user interface. This is what the user sees and interacts with. It needs to have the ability to display code blocks, a place for a user to select code blocks, a place for the user to organize code blocks into a program, and buttons to compile and run the block programs. All of these functions need to be in a grid layout for easy organization. Based on these requirements, Python's Tkinter (TK) library was selected to create the user interface. TK offers a lot of useful functions and classes for creating windows, grids, buttons, and more. Figure 12, on the next page, shows the gridded layout described. Table 1, also on the next page, shows a breakdown of the final class design.

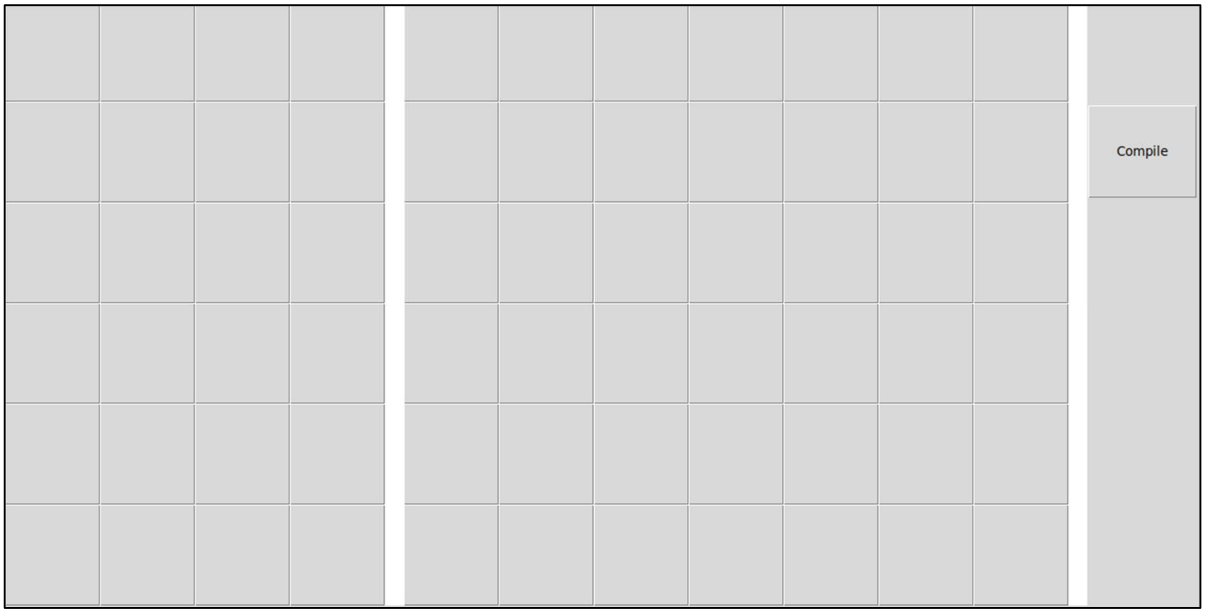


Figure 12 – The Empty User Interface

Table 1 – User Interface Class

class ProgramWindow		
Class for the program window (user interface) that handles the organization and display of the entire window		
attribute	*window_size*	Size of the window, in blocks
attribute	*tool_width*	Width of the toolbox, in blocks
attribute	*window*	TK window instance
attribute	*block_size*	Size of each block, in pixels
attribute	*block_map*	Matrix of blocks in the program space
method	*initialize*	When window is created, draws the toolbox, program space, and everything else in the user interface
method	*run*	Creates and displays the UI
method	*compile_cmd*	Links compile button to compile function
method	*place_in_prog*	Places a program block

3.3.2 Code Blocks

The next part of the programming interface is the code blocks. Table 2 shows all the blocks that will be implemented in the programming interface. The first set of blocks implement all the features of the robot described in the hardware design section. The second set of blocks implement all the features required by Indiana Computer Science 6-8 PA.1. The first block created was the Forward Block, which is a movement block that makes the robot move forward continuously. The process for making the Forward Block was as follows:

Table 2 – All Code Blocks.

Source	Block
Robot	Movement
Robot	Sound Output
Robot	Touch Input
Robot	Tail Wag Output
Robot	Joystick Input
Robot	Screen Output
IN C.S. 6-8.PA.1	Sequencing
IN C.S. 6-8.PA.1	Looping
IN C.S. 6-8.PA.1	Nested Looping
IN C.S. 6-8.PA.1	Conditionals
IN C.S. 6-8.PA.1	Compound Conditionals
IN C.S. 6-8.PA.1	Expressions
IN C.S. 6-8.PA.1	Variables
IN C.S. 6-8.PA.1	Functions
IN C.S. 6-8.PA.1	Parameters

1. An icon/image representing the forward block was designed.
2. Python code was written and tested for making the robot move forward.
3. A Forward Tool Block (See Table 4) was created:
    a. The forward image was packed into the widget and stored in the widget attribute.
    b. The "create_code" method was rewritten to output code that makes the robot move forward when called.
    c. As a tool block, the Forward Tool Block inherits all attributes and functions from the grandparent block class (Table 3) and parent tool block class (Table 4) that were not redefined or overwritten.

Table 3 – The Generic Block Class.

class Block		
Class for a generic block, inherited by all other block classes.		
method	*fetch_loc*	Fetches block's location based on mouse event

Table 4 – The Tool Block Class

class ToolBlock		
A block in the toolbox that can make program blocks. Inherits from the generic Block class.		
attribute	*widget*	A TK widget displayed to the user
method	*create_code*	A placeholder method overwritten when creating a code block.
method	*clone_to_prog*	When called, creates a program block using this tool block's widget and clone_to_prog.
method	*initialize*	Takes the inputted widget, displays it in the toolbox, and ties a left mouse click on the widget to the clone_to_prog method.

Now, the Forward block exists in the toolbox. When it is clicked, it will create a Forward Program Block (Table 5) in the program space using the forward block's widget and "create_code" method. This block can be dragged and dropped around the program space to create a program. The user makes programs based on what they see (the widgets) while the compiler makes a program by calling the "create_code" method from each program block in order. The creation of all other blocks followed the same process as the Forward Block.

Table 5 – The Program Block Class

class ProgBlock		
A block that can be dragged and dropped through the program space and read by the compiler to create Python code. Inherits from the generic Block class.		
method	*create_code*	When called by the compiler, generates this block's Python Code equivalent.
attribute	*widget*	A TK widget displayed to the user
method	*start_drag*	Takes the block out of the block map
method	*during_drag*	Moves the block with the mouse
method	*stop_drag*	Places the block in the block map
method	*initialize*	Ties the a mouse click and drag on the widget to the three methods above.

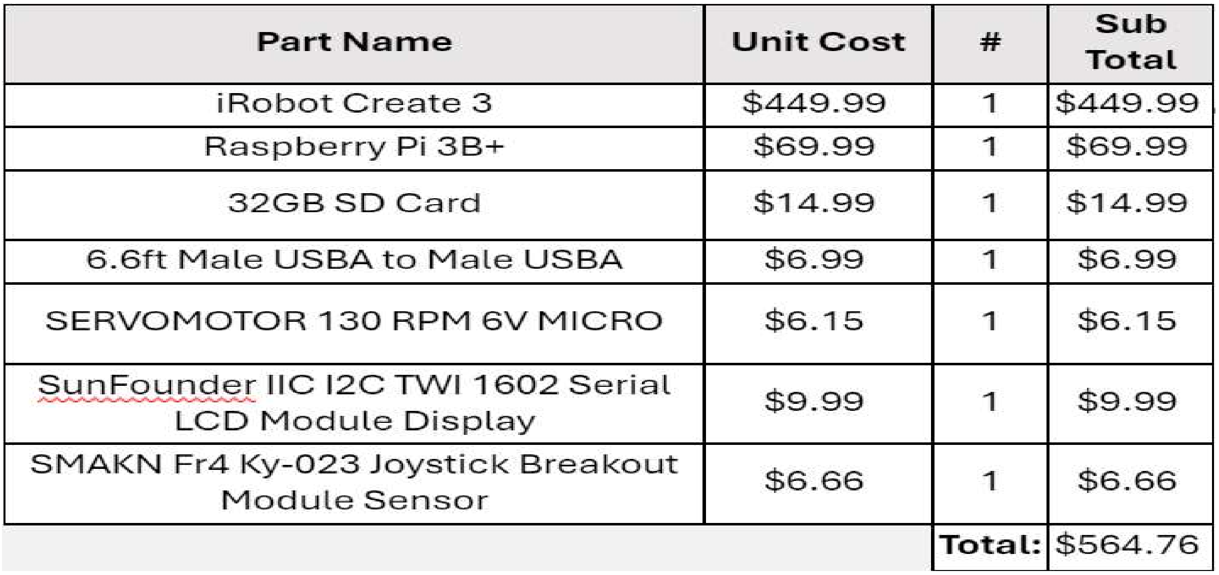
3.3.3 Compiler

Finally, there is the compiler for the programming interface. When the user is done with their program and hits the "compile" button, the compiler turns their block program into a Python program readable by the Raspberry Pi. To do this, it first creates a new, empty Python file. Then, it writes all of the always required code to this file, such as importing libraries and connecting to the robot. After that, it reads the grid of the user's code blocks by row. For each row, it calls each block's "create_code" method one by one to generate Python code equivalent to the block program row. After all the rows have been translated, the finished Python file is transferred to the Raspberry Pi over a USB cable.

4 PROJECT MANAGEMENT

4.1 BILL OF MATERIALS

Table 6 – Bill of Materials



Part Name	Unit Cost	#	Sub Total
iRobot Create 3	$449.99	1	$449.99
Raspberry Pi 3B+	$69.99	1	$69.99
32GB SD Card	$14.99	1	$14.99
6.6ft Male USBA to Male USBA	$6.99	1	$6.99
SERVOMOTOR 130 RPM 6V MICRO	$6.15	1	$6.15
SunFounder IIC I2C TWI 1602 Serial LCD Module Display	$9.99	1	$9.99
SMAKN Fr4 Ky-023 Joystick Breakout Module Sensor	$6.66	1	$6.66
		Total:	$564.76

The bill of materials is a set part list of all the parts needed for the project. The iRobot Create 3 as stated before can be replaced with donated and recycled Roomba counterparts. Also, donated Roombas would depend on supporters of the local school system. If not able to get any donated Roombas, later Roomba models could be bought to produce the same results as the expensive $449.99 iRobot Create 3. With that said, Roombas are the most wasted and thrown-away robots on the market and nowadays middle schools have laptops for the classroom. Within a few years of upcycling Roombas and the accessibility of laptops in middle schools, the overall price will be dramatically lower. From the $564.76 price range to a manageable $114.77 worth of parts is a lot more affordable for schools to order. Especially if a classroom is split into multiple groups, the low price helps schools justify the purchases when compared to other current solutions.

4.2 Teamwork Breakdown

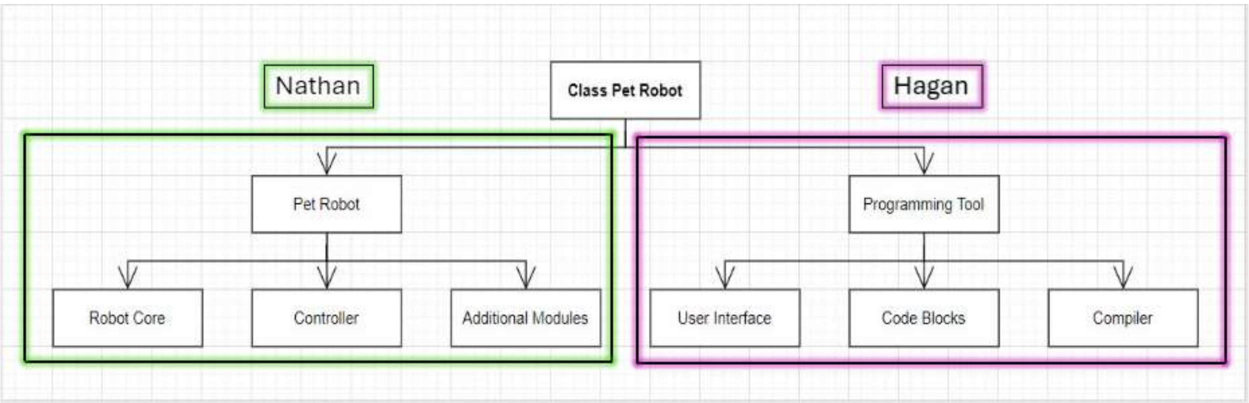
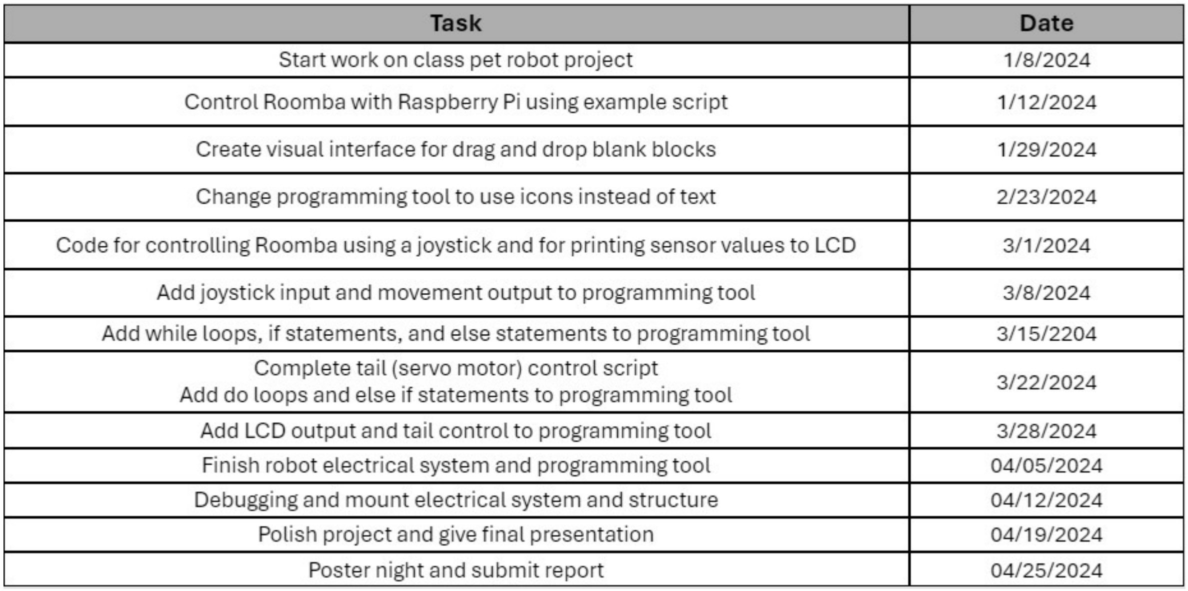


Figure 13 – Teamwork Breakdown Diagram

In the teamwork breakdown section Nathan and Hagan assigned tasks within the project to keep up with scheduled deadlines. Shown below in Table 2 are the project's tasks that the team set on dates. Tasks have been assigned with the person's strengths and interests in mind in order to help the project's productivity and overall success.

4.3 Project Timeline

Table 7 – Project Timeline Breakdown



Task	Date
Start work on class pet robot project	1/8/2024
Control Roomba with Raspberry Pi using example script	1/12/2024
Create visual interface for drag and drop blank blocks	1/29/2024
Change programming tool to use icons instead of text	2/23/2024
Code for controlling Roomba using a joystick and for printing sensor values to LCD	3/1/2024
Add joystick input and movement output to programming tool	3/8/2024
Add while loops, if statements, and else statements to programming tool	3/15/2204
Complete tail (servo motor) control script Add do loops and else if statements to programming tool	3/22/2024
Add LCD output and tail control to programming tool	3/28/2024
Finish robot electrical system and programming tool	04/05/2024
Debugging and mount electrical system and structure	04/12/2024
Polish project and give final presentation	04/19/2024
Poster night and submit report	04/25/2024

5 IMPLEMENTATION

5.1 *Test Plan*

Here are the tests to verify the functionality of a project.

1. For each output code block, such as move forward or make a noise, write a program with just the output block, and run it. Verify that the output block functions as expected.
2. For each input code block, use an if statement with any output block to verify that each input is working as expected. An if-else may also be used.
3. For the rest of the blocks, write programs implementing them and run them to verify each block. Be sure to only introduce one new block at a time.

Once all blocks are verified to function as expected, write a few complex programs. If no faults are still found, the project is working.

5.2 *Current State*

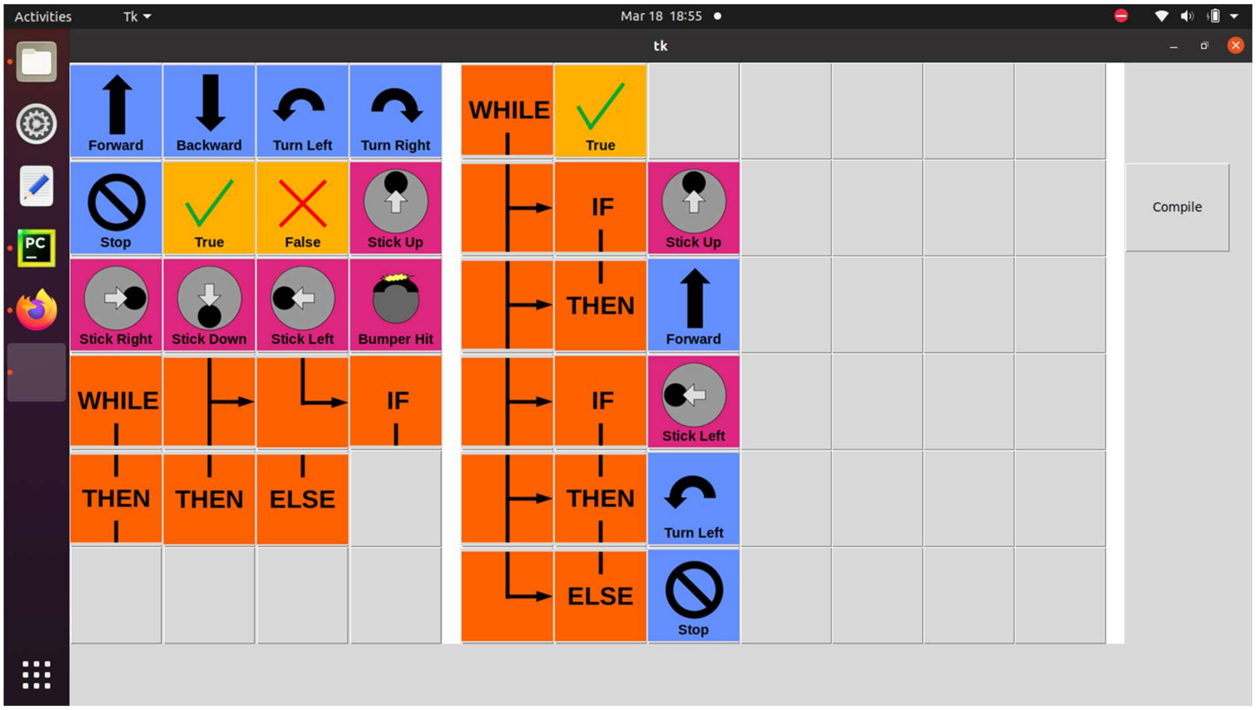


Figure 14 – An Implemented Program.

Figure 10 above shows the current state of the project's programming interface. On the left side is the "toolbox", where every unique code block is stored. Clicking a box in the toolbox copies it to the right side, known as the "programming space." In the programming space, blocks can be dragged and dropped anywhere in the grid. Lines of code are read row by row from left to right. When a program is finished, the compile button on the far right generates the Python file. The exact program shown above is verified to function correctly on the robot.

6 CONCLUSION

In summary, our project addresses the challenge of waning interest in Electrical Engineering among junior high students in Evansville with a targeted and innovative solution. Through the creation of an educational robot class pet featuring a user-friendly interface, we aim to foster an engaging and interactive learning environment. This hands-on educational tool, supported by teachers ready to assist when needed, seeks to ignite a passion for STEM fields in the budding minds of our community. Our impact extends beyond the classroom, fostering innovation within our community and contributing to the goal of building a diverse and vibrant STEM workforce for the future.

6.1 NATHAN'S REFLECTION

First off, when starting this project, I had no experience in embedded systems but always wanted to learn about them. Now I can say that by taking embedded systems and working on this project at the same time, I am extremely satisfied with how and where we were able to get the project this semester. The project has plenty of room for improvement and ultimately be able to be put into a classroom for students to enjoy and learn from. Areas where I could have done better were asking for help sooner than stressing to get results at the last moment. I have always struggled with this but after the project, I can take a step back and realize how so many tiny issues could have been solved sooner with a simple request for help. With this maybe I could have gotten the robot a shell to cover up all the open wire on top, but I guess that will be a future task now. On another note, not knowing any Python going into this project was scary at first, but with the help of my fantastic project partner Hagan helped me at first and then I started to pick up a lot with each new component I added to the robot. Honestly, when I got even one of the components working with the robot I was having the best time of my college career. Just knowing that I was accomplishing something with a greater purpose in the future. Whoever takes this project next I would recommend finding a better way to mount the components on the robot and overall making the robot look a lot more pet-like (maybe adding a shell that students could customize in the classroom and give it a sense of identity and makes the students more engaged). Was a blast overall!

6.2 HAGAN'S REFLECTION

First, I am very satisfied with how this project turned out. I think it has lots of potential for expansion, probably enough for four more senior design teams to work on this. We did not get as many of the blocks implemented as we wanted to, but we proved the concept very well. What was my downfall in the end was rushing the code too much. Up through creating the block classes I was fine, but when I got to each of the code blocks, I started rushing through them so I could make as many as I could. I should have planned how they would interact with each other and the compiler more carefully, because once we hit the 19 code blocks shown in Figure 11, it became very difficult to create new blocks. This is because the way I created the compiler was sloppy. Instead of finding a careful, generic solution, I began trying to hard-code every possible sequence of blocks into the compiler. It worked for a while, but I eventually hit the wall. I attempted to go back and correct my mistakes, but I never got enough done to add more blocks into the program. The next group's first task should be to fix my mistake: scrap the compiler and rework how code blocks are translated.

REFERENCES

[1] A. Grabert, Interviewee, *Consultation with USI STEM Outreach.* [Interview]. 19 10 2023.

[2] U.S. BUREAU OF LABOR STATISTICS, "Employment in STEM occupations," 17 April 2024. [Online]. Available: https://www.bls.gov/emp/tables/stem-employment.htm. [Accessed 2024 17 April].

[3] B. Paykamian, "Study: Gen Z Interested in STEM but Lacks Exposure," Government Technology, 2023. [Online]. Available: https://univsouthin.idm.oclc.org/login?qurl=https%3A%2F%2Fwww.proquest.com%2Fmagazines%2Fstudy-gen-z-interested-stem-lacks-exposure%2Fdocview%2F2898167016%2Fse-2%3Faccountid%3D14752. [Accessed 3 April 2024].

[4] S. Orlando, E. Gaudisos and F. De La Paz, "Supporting Teachers to Monitor Student’s Learning Progress in an Educational Environment With Robotics Activities," *IEEE Access,* vol. 8, pp. 48620-48631, 2020.

[5] LEGO® Education, "LEGO® Education SPIKE™ Prime Set," [Online]. Available: https://assets.education.lego.com/v3/assets/blt293eea581807678a/blt96801dfdfb5b7521/627118b9941a2939d3d00593/45678_prod_packaging_SPIKE_PRIME_Set_01.png?locale=en-us&auto=webp&format=jpeg&width=1600&quality=90&fit=bounds. [Accessed 7 April 2024].

[6] L. e. a. Paull, "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," *2017 IEEE International Conference on Robotics and Automation,* pp. 1497-1504, 2017.

[7] DUCKIETOWN, "Duckiebot," [Online]. Available: https://duckietown.com/wp-content/uploads/2020/10/DSC04328-min-768x512.jpg. [Accessed 7 April 2024].

[8] K.-H. Yang and H.-Y. Lin, "Exploring the Effectiveness of Learning Scratch Programming with Code.org," *8th International Congress on Advanced Applied Informatics,* pp. 1057-1058, 2019.

[9] Code.org, "Lesson 5: Maze: Debugging," [Online]. Available: https://www.google.com/url?sa=i&url=https%3A%2F%2Fstudio.code.org%2Fs%2Fcourse1%2Flessons%2F5%2Flevels%2F12&psig=AOvVaw2mQKOvco5GQ4NAUg3nNWOg&ust=1712634069691000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCLi4_qPZsYUDFQAAAAAdAAAAABAS. [Accessed 7 April 2024].

[10] Scratch, "Scratch Programming Demo," [Online]. Available: https://miro.medium.com/v2/format:webp/0*oDbEFNhoM75KOnvF.. [Accessed 7 April 2024].

[11] Dr.Thomas, Interviewee, *Consultation with USI Education Department.* [Interview]. 17 10 2023.

[12] Indiana Department of Education, "Science & Computer Science," 2023. [Online]. Available: https://www.in.gov/doe/students/indiana-academic-standards/science-and-computer-science/. [Accessed 10 October 2023].

[13] iRobot, "Create 3," [Online]. Available: https://edu.irobot.com/what-we-offer/create3. [Accessed 5 December 2023].

[14] Raspberry Pi, "Raspberry Pi 3B+," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. [Accessed 5 December 2023].

[15] DigiKey, "Servo Motor Micro," Pololu Corporation 2820, [Online]. Available: https://www.digikey.com/en/products/detail/pololu-corporation/2820/10450037?utm_adgroup=&utm_source=google&utm_medium=cpc&utm_campaign=PMax_DK%2BProduct_Product%20Categories%20-

%20Top%2015&utm_term=&utm_content=&utm_id=go_cmp-19646629144_adg-_ad-__dev-c_e. [Accessed 12 March 2024].

[16] Parallax, "16×2 I2C LCD Display Module with Blue Backlight," [Online]. Available: https://www.parallax.com/product/16x2-i2c-lcd-display-module-with-blue-backlight/. [Accessed 10 March 2024].

[17] js4iot.com, "Analog To Digital Conversion (I2C) with RPIO," [Online]. Available: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.js4iot.com%2F2021%2F06%2F06%2FADC_RPIO.html&psig=AOvVaw3CxkvhtaFZ4nhPiq0AMkHD&ust=1714242817856000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCNiJ6MjC4IUDFQAAAAAdAAAAABAE. [Accessed 10 March 2024].

[18] M. Chow, "Using a joystick sensor on an Arduino," [Online]. Available: https://medium.com/@melaniechow/using-a-joystick-sensor-on-an-arduino-3498d7399464. [Accessed 10 March 2024].

[19] J. Grover, "opensource.com," Red Hat, 18 October 2019. [Online]. Available: https://opensource.com/article/19/10/python-programming-paradigms#:~:text=Python%20supports%20four%20main%20programming,all%20four%20available%20and%20working.. [Accessed 2 April 2024].

APPENDIX A – STANDARDS

	Standard Name	Organization	Purpose in Project
1	PEP8	Python Software Foundation	Code readability and mutability for future developers.
2	Grades 6-8 Computer Science	Indiana Department of Education	Ensuring project can be properly implemented in classroom curriculum.
3	USB 4.0	USB Implementers Form	USB file transfer

APPENDIX B – DATASHEETS

iCreate 3	Raspberry Pi 3B+	Sensor Kit