Federal University of Technology - Paraná – UTFPR
Academic Department of Electronics – DAELN
Academic Department of Informatics – DAINF
Computer Engineering
Integration Workshop 3 (ELEX22) – S71 – 2024/1

Technical Report

Ash: the table tennis ball fetcher

Alfons Andrade – alfons@alunos.utfpr.edu.br
João Vitor Caversan dos Passos – joaopassos@alunos.utfpr.edu.br
Guilherme Gomes Barboza – guilhermebarboza@alunos.utfpr.edu.br
Gustavo Valente Gulmine – ggulmine@alunos.utfpr.edu.br
Matheus Diniz – matheusdiniz@alunos.utfpr.edu.br

June 2024

Abstract

The ASH table tennis ball fetcher project is a robot developed to catch table tennis balls in a controlled environment (training court) autonomously. This paper presents the development, including the mechanical design, hardware implementation, and software development. The mechanical structure includes a chassis, a motion system, and integration with a vacuum tube for ball collection. The electronic design addresses the power supply, motor control, and peripheral connections. It features a custom Raspberry Pi shield and ASH's docking base circuit that is used as a charging station and a place to return the fetched balls. The software component includes embedded firmware and an Android mobile application for user interaction. The firmware manages tasks such as Bluetooth communication, odometry for tracking the robot's position, ball detection using computer vision, and static object detection. The mobile application enables command transmission and status monitoring via an interface, ensuring smooth operation and real-time updates. The ASH project provides an autonomous ball collection solution for table tennis ITTF (International Table Tennis Federation) courts [1].

1 Introduction

As with any other sport, table tennis requires a lot of practice to master. Since precise movements are so important during matches, training sessions usually involve hundreds of movement repetitions, so it becomes possible to fine-tune them. That implies hundreds of table tennis balls being used and scattered around the training area after each exercise. This is usually not a problem when training in big groups since the task of picking up the balls can be separated among all of the individuals, but for small groups or even someone training alone with the help of an automated ball launcher, picking up hundreds of balls

can get tiring and time-consuming. In this context, automating the ball collection process could improve the dynamism of training sessions and allow the athletes to enjoy their rest time without worries.

1.1 Proposed solution overview

The proposed solution to the problem, presented in Figure 1, is ASH, an autonomous robot capable of identifying table tennis balls within the play area and collecting them. The complete solution will also include a docking base and a mobile application.

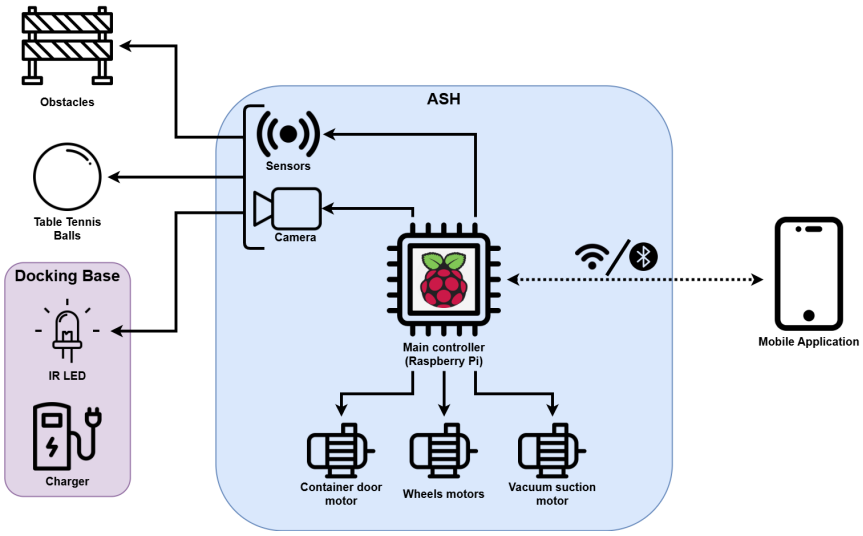


Figure 1: Proposed solution system block diagram

To autonomously navigate the play area without concerns, ASH needs to be capable of detecting table tennis balls and obstacles and the location of its docking base, which doubles as a charging station, using the IR LED on top of it. To do so, ASH is equipped with cameras and multiple sensors that will allow it to collect a lot of data regarding the environment. Using computer vision algorithms, ASH identifies where and how far the balls in front of it are located so it can fetch them using its exclusive vacuum suction system, storing the collected balls in a container. Since a lot of processing power is required to perform all of these measurements while also controlling the robot's movements, ASH is equipped with a Raspberry Pi 5 as its main processor.

Controlling the robot's operation with the specially designed mobile application will be very simple for the user. Connecting to the robot wirelessly through Bluetooth or Wi-Fi protocols, the mobile application serves as a control center for the robot, offering users the ability to start, pause, or stop an operation, schedule a start, and even track, in real-time, how many balls ASH has collected as well as what it is currently identifying in the environment. This user-friendly

interface puts the power in your hands, making the operation of the ASH robot a seamless and enjoyable experience.

2 Project Specification

The requirements for this project comprise an extensive list of features necessary for the robot to operate correctly, as well as the rules that define its operation. All functional, non-functional, and anti-requirements defined were further divided among each part of the project (mechanics, hardware, firmware, and mobile application), with the latter also including environmental restrictions that ensure correct operation. Due to the complexity of the project, the complete requirements list ended up with a total of more than 150 requirements, which are available in full extent online at ASH Notion blog, with some of the most important functional requirements being presented at Table 1.

Req.	Description
MFR10	The robot must have guide flaps to lead table tennis balls to the suction tube entrance.
MFR11	The robot must have a storage container able to store at least 10 balls and no more than 15.
MFR13	The robot's vacuum ball collection system, including the storage container and suction tube, must be as best sealed as possible to prevent pressure loss and ensure optimal suction efficiency.
MFR15	The robot must have the camera looking forward and at an angle of 85 to 95 degrees relative to the ground.
MFR16	The robot must have 2 wheels connected to the motors and one swivel wheel, to better distribute the weight and still have the maneuverability needed.
MFR22	The return base must have magnets on its wall so that the robot can know if it has parked.
MFR23	The return base shall have a 25cm tall tower with a led controlled bulb on top to act as a landmark.
HWFR1	The robot must be powered using two 12V rechargeable battery packs to split the current consumption and be mobile.
HWFR4	The robot's power supply system must be capable of supplying 5V to the main controller and the peripherals.
HWFR5	The robot's power supply system must be capable of supplying 6V to the wheels motors and the container door servo motor.
HWFR14	The robot must be able to measure the distances in all four directions (front, back, left and right sides) to help with obstacle detection.
HWFR15	The robot must detect if a ball has correctly entered and exited the tube.
HWFR18	The robot charger connection must prevent damage when connected to unsupported chargers.
HWFR19	The return base must have a signaler recognizable by the robot using only the camera
SFR1	The robot must autonomously navigate in environments in agreement with the imposed environmental restrictions.
SFR1.1	The robot must recognize objects within its operational area.
SFR1.7	The robot must autonomously navigate to the return base to unload the stored balls or wait to be recharged.
SFR5	The robot must communicate data to the user interface, including battery levels, sensor readings, ball position, and collection count.
UIFR1	The user interface must allow the user to start and end the current robot cycle immediately.
UIFR7	The user interface must provide a Field Of View estimation map with identified balls.

Table 1: Main project requirements. Source: Authors (2024). | **FR-**Functional Requirement | **MFR-**Mechanical Functional Requirement | **HWFR-**Hardware Functional Requirement | **SFR-**Software Functional Requirement | **UI-**User Interface Functional Requirement |

The **environmental restrictions** for the ASH table tennis ball fetcher project ensure correct operation within typical training environments. According to the International Table Tennis Federation (ITTF) Statutes, the play area must be enclosed by surrounds about 75 cm high, with a dark-colored background and no bright light sources. The floor must be non-reflective, non-slippery, and level. Training environments typically have compact court sizes, approximately 9m long by 4m wide, compared to the official 14m by 7m dimensions. The project uses only white table tennis balls with a diameter of 40mm and a weight of 2.7g,

as specified by the ITTF. These restrictions guide the design and operation of the robot to ensure it functions effectively in realistic training settings.

3 Development

The project development was performed on three fronts, comprising the design and implementation of the four main pillars of the proposed solution, as seen in section 2. The first was the mechanical structure, presented in subsection 3.1. The second was the hardware design and assembly, detailed in subsection 3.2. Finally, the last front was software development, which included firmware and mobile application development, as further discussed in subsection 3.3.

3.1 Mechanical Structure

The mechanical structure of the ASH project encompasses three primary components: the structural framework, motion system, and integration with the vacuum tube system. This section provides an in-depth analysis of the proposed mechanical framework and its alignment with the project's functional requisites.

The primary structure of the robot is engineered as a resilient chassis capable of housing all essential components. Balancing lightweight construction with durability, the chassis aims to deliver strength and mobility. The structural considerations of the robot embrace aspects such as:

- **Strength and Durability:** The chassis has been engineered to withstand self-inflicted impacts and collisions, safeguarding internal components from damage in compliance with [MFR1].
- **Weight Distribution:** Central placement of the batteries between the wheels ([MFR2]) and positioning as close to the ground as possible ([MFR3]) work to establish a low center of gravity, bolstering stability.
- **Mobility:** Designed to attach and fix two DC motors for the 68 mm wheels and the swivel wheel below the second container for robot movement.
- **Vacuum Motor Integration:** Design to seamlessly integrate with the vacuum motor system to ensure efficient suction performance.

The robot's vision system is equipped with a forward-facing camera positioned at an optimal angle between 85 and 95 degrees relative to the ground to provide a clear field of view for navigation and ball detection.

The robot's three-wheel configuration, featuring two drive wheels and one swivel wheel, is designed to optimize weight distribution and maneuverability.

All internal components, including the battery, vacuum motor, and other electronics, could be securely housed within the robot framework, ensuring protection from external elements and impacts.

The Figures 2,3,4 and 5 presents a series of visual representations illustrating the conceptual design made on design software (left) and the actual 3D-printed results of ASH's mechanical structure (right).

(a) 3D Design (b) 3D Printed

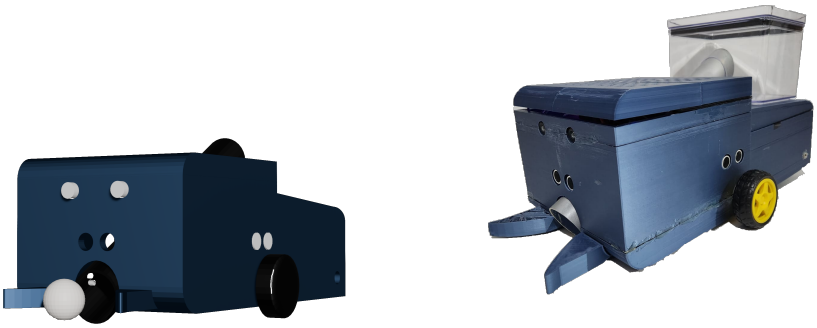
Figure 2: Left view. Source: Authors (2024)

(a) 3D Design (b) 3D Printed

Figure 3: Front view. Source: Authors (2024)

(a) 3D Design (b) 3D Printed

Figure 4: Upper view. Source: Authors (2024)



(a) 3D Design (b) 3D Printed

Figure 5: Isometric view. Source: Authors (2024)

The return base serves as a docking station for the robot, incorporating key features. It includes a ball storage basket, a charging circuit, parking assistance utilizing magnets to interact with the robot's Hall sensor and an antenna with an infrared LED that is recognized by ASH's vision system. Additionally, the return base is designed to handle IMU (Inertial Measurement Unit) errors by zeroing the IMU once the robot parks, ensuring accurate navigation for subsequent tasks.

The return base was crafted using a plastic case and a custom-cut PVC antenna to accommodate the infrared feature, ensuring precise tailoring to the robot's requirements. See Figure 6.

(a) Front view ASH's Base

(b) Left view ASH's Base

(c) Upper view ASH's Base

(d) Isometric view ASH's Base

Figure 6: ASH's Base views. Source: Authors (2024)

3.2 Hardware

In order to provide all of the functionalities defined in the requirements, the designed hardware solution for the ASH project had many hurdles to overcome. Considering the high power consumption of the robot's internal components (mainly RaspBerry Pi and the motors), it became necessary to have a dedicated and intricate power supply system inside it, capable of supplying power on multiple voltage levels and protecting the internal components from damage. On top of that, supplying and controlling four different motors at two different voltage levels required ASH to have a dedicated motor driver module, which is responsible only for these functionalities. Furthermore, all of the peripherals required to ensure full functionality for the robot made it necessary to create a dedicated Raspberry Pi shield to organize all of the main controller connections better. Finally, the hardware solution for ASH also included a docking base, housing an infrared LED for recognition and a fixed 12V, up to 20A, power supply

that would double as the default charger for the robot's batteries.

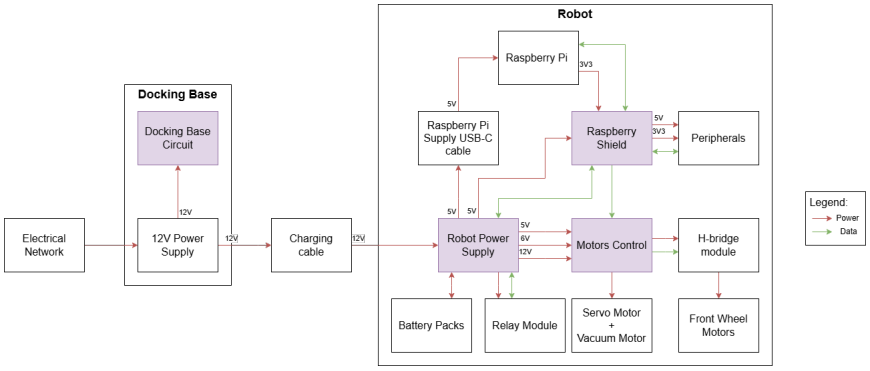


Figure 7: System electric/electronic block diagram. Source: Authors (2024).

The final system-level electric/electronic design can be seen at Figure 7, highlighting the relationship between the major system components. The purple blocks represent the four designed PCBs that solve the aforementioned hurdles, as further explained in subsections 3.2.1 to 3.2.4. The hardware design description concludes with a brief presentation of ASH's peripherals in subsection 3.2.5.

3.2.1 Docking base

The Docking base circuit, whose schematic is presented in Figure 8, is relatively simple. It consists of a 3W infrared power LED that is turned on/off according to the current state of the circuit. If a basket is connected to the base, the connection on the microswitch will be closed and the LED stays on to signalize it. If not, an NE555 IC configured as an astable oscillator will make the LED blink with a period of one second at approximately 50% duty cycle.

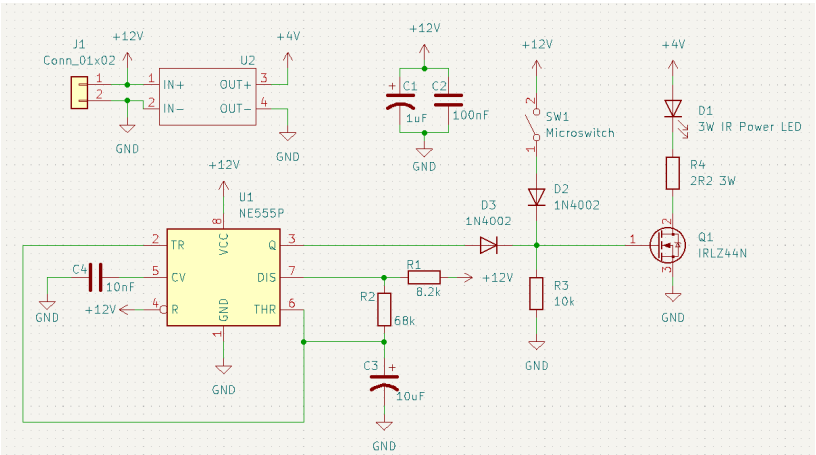


Figure 8: Docking base schematic. Source: Authors (2024).

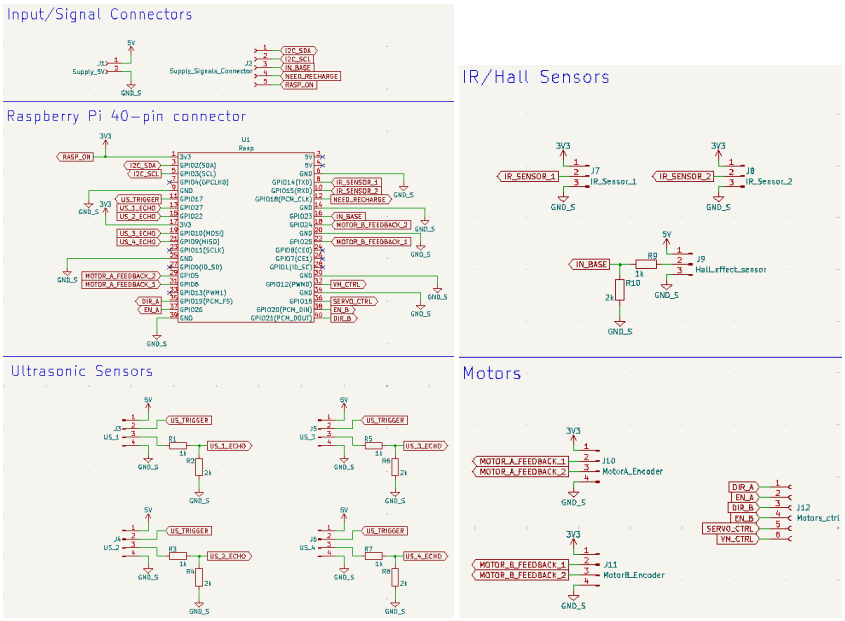


Figure 9: Raspberry Pi connector shield schematic. Source: Authors (2024).

3.2.2 RaspberryPi Connector Shield

The second designed hardware component was the Raspberry Pi Connector Shield, presented in the schematic at figure 9. This module is responsible for grouping up signals to facilitate connections to other modules and providing the connectors to each of the peripherals. In order to alleviate the Raspberry Pi's power consumption, all peripherals that operate at 5V are connected to a power input connector so that they can be supplied directly by ASH's power supply system.

3.2.3 Motor drivers module

The motor drivers module shown in Figure 10 packs the driver circuits for the four motors designed to be inside the robot. It operates at three different voltages: 3.3V for the Raspberry Pi control signals, 6V for the front wheel and servo motors, and 12V for the vacuum suction motor.

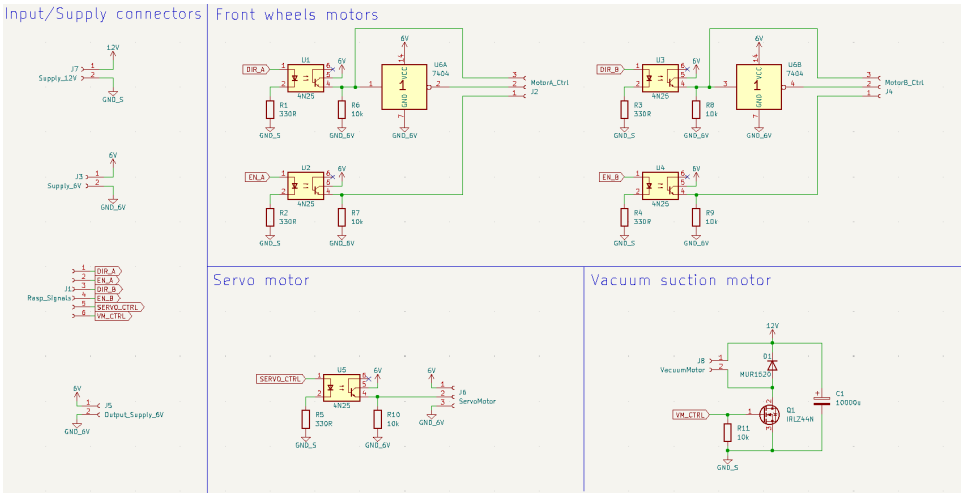


Figure 10: ASH motor drivers module schematic. Source: Authors (2024).

Each one of the motors is controllable using this module in different ways:

- The servo motor control is the simplest of the three, with the signal passing through an optocoupler and going directly to the motor itself.
- The vacuum motor is controlled directly by a PWM signal coming from the Raspberry Pi, switching the IRLZ44N MOSFET transistor to control the state of the motor. Since the current peak from the motor can be really high, a big capacitor was added in parallel to the battery.
- The front wheel motors can be controlled in two ways, using an external L298 H-bridge module. If the control is performed using a PWM on the enable pin and the direction is set, a NOT gate is provided, so it's possible to control the direction using only one input signal. On the other hand, if the control is performed only using the direction pins, a jumper must be added to the enable pins on the H-bridge, and the NOT gate output can be ignored.

3.2.4 Power Supply

Lastly, the Power Supply module, presented at Figure 11, is the most complex circuit developed for the ASH project. It can detect whether a charger has been connected to the robot and if the polarity on this charger is correct. That, paired with a fuse and a varistor tied to the charger connector, allows this design to protect the internal circuitry from overcurrent, overvoltage, or reverse polarity.

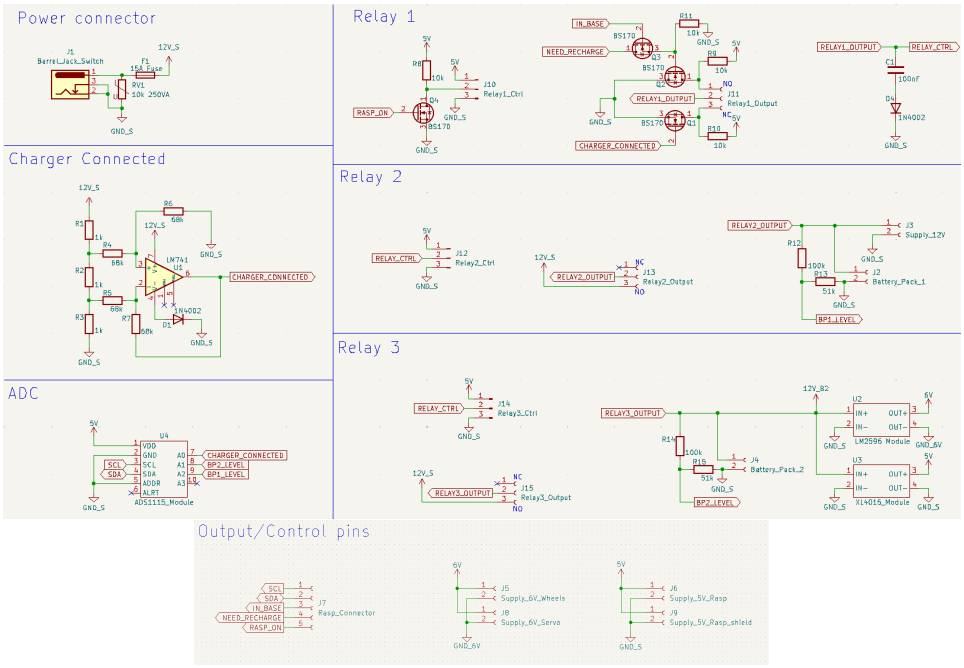


Figure 11: ASH Power supply system schematic. Source: Authors (2024).

Since the default charger provides a fixed 12V output, it was necessary to design the reverse polarity protection to minimize the voltage drop; otherwise, the batteries could not be fully charged. That was achieved using a relay setup to decouple the batteries from the charging connector unless charging is required. The relays control is performed using MOSFETs to switch the relays states depending on four input logic signals (*RASP_ON*, *NEED_RECHARGE*, *IN_BASE* and *CHARGER_CONNECTED*), connecting the batteries to the charger only if:

1. In case the main processor is on (*RASP_ON* = HIGH),the connection will happen only when the robot is docked in base (*IN_BASE* = HIGH) and the main processor signals it needs to recharge (*NEED_RECHARGE* = HIGH).

2. In case the main processor is off (*RASP_ON* = LOW), the connection will happen only when a charger is connected with the correct polarity.

The power supply system also packs in an ADS1115 analog-to-digital converter to measure battery levels and the presence of a charger, an LM2596 step-down voltage regulator module, responsible for supplying 6V to the motors, and an XL4015 step-down voltage regulator, that supplies 5V to the Raspberry Pi and the peripherals, all of which are connected/soldered directly in the board.

3.2.5 Peripherals and utilities

The complete hardware design for ASH includes multiple off-the-shelf peripherals to fulfill many of its requirements as plug-and-play as possible. A total of

13 peripherals are connected to the main controller to perform these tasks, with them being:

- **[2x] Infrared Obstacle Sensor module:** Coupled at the start and end of the collection tube to identify if a ball has correctly entered and exited it.
- **[4x] Ultrasonic Sensor module HC-SR04:** Each sensor is fixed at one of the four sides of the robot, allowing it to detect the presence of obstacles close to it.
- **[2x] Motor Encoders:** Each wheel motor is directly coupled to an encoder module, allowing the robot to measure the rotation on each wheel to help with precise movement and minimize errors.
- **[2x] IMX219-83 Cameras:** Two independent cameras integrate a single module and are connected directly to the Raspberry Pi's CSI interfaces. They are the robot's " eyes, " being used for all computer vision functionalities.
- **[1x] ICM20948:** Also part of the IMX219-83, this I²C module packs an accelerometer, gyroscope, and magnetometer and is used to improve angle and movement detection.
- **[1x] Hall-effect sensor:** A rather simple magnetic field detector is used to identify if the robot is close/coupled to the base.
- **[1x] ADC-1115 Analog-to-digital converter:** An I²C module is used to measure both batteries' voltage levels and verify if a charger is correctly connected.

Considering all that was presented in this section, the final internal hardware assembly of the ASH project is represented in the Figure 12.

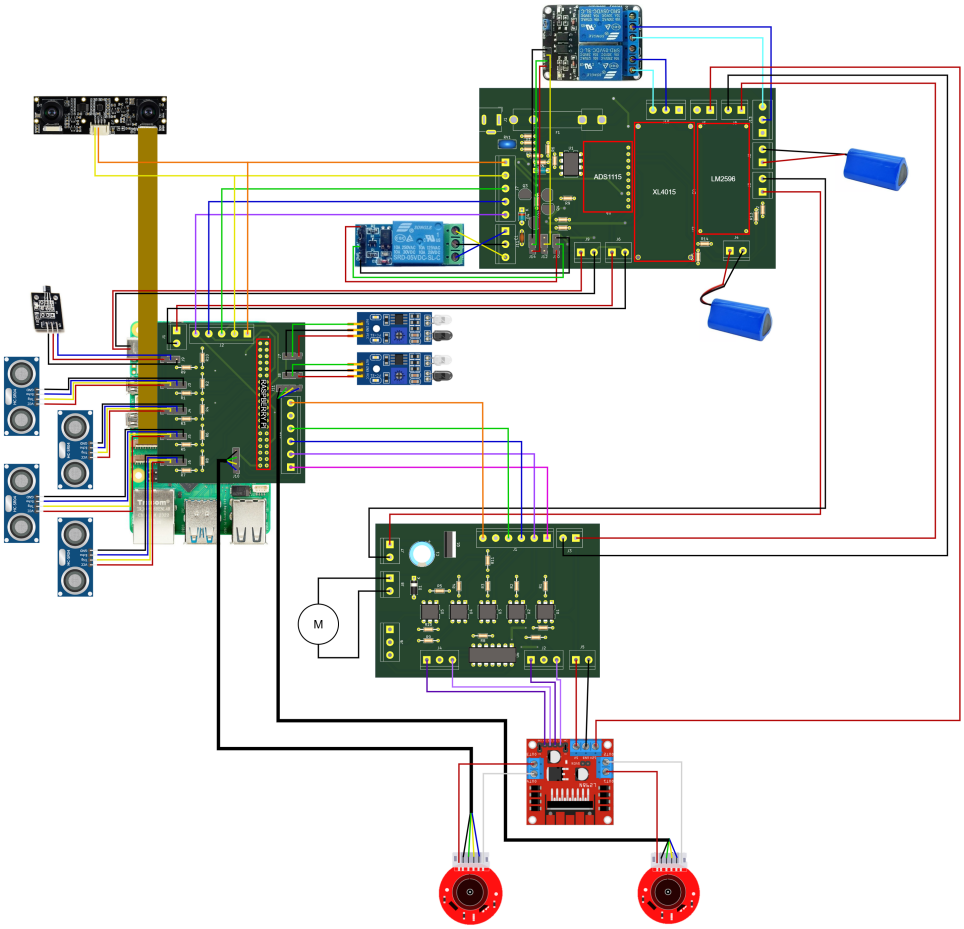


Figure 12: Robot internal hardware assembly diagram. Source: Authors (2024), using images from [2], [3], [4], [5], [6], [7], [8] and [9].

3.3 Software

The software was divided into two main parts: the embedded software for controlling the robot and the mobile application for the user interface for command transfer and process monitoring. This section will present each of these parts.

3.3.1 Firmware/Embedded Software

For better visualization of the software architecture, figure 13 shows the designed class diagram for the firmware.

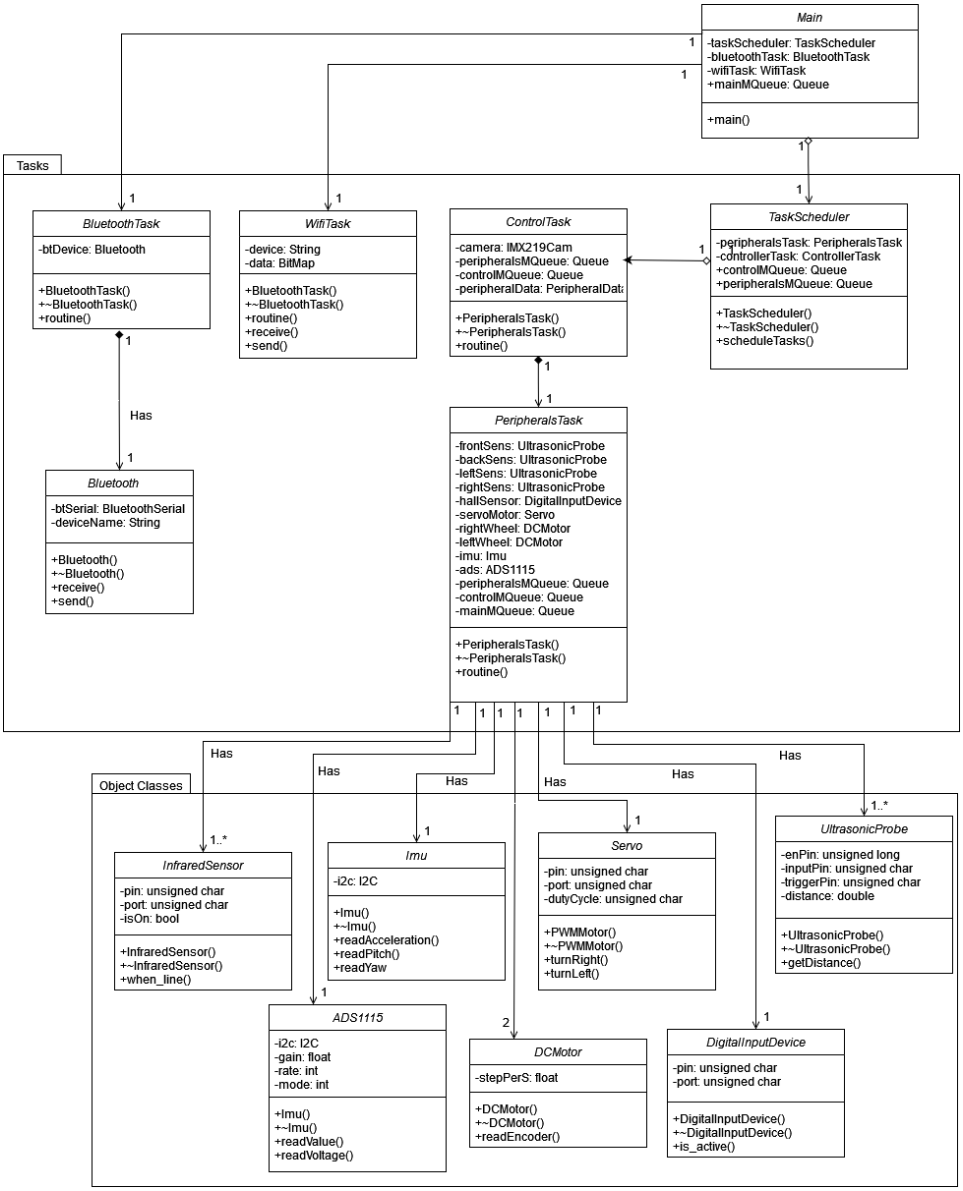


Figure 13: Embedded software class diagram. Source: Authors (2024)

As seen in Figure 13, the embedded software was divided into three sets: the main loop, the core tasks, and the auxiliary object classes. These will be described in the following sections.

Main loop:

It is the module where all message queues for task communication are instantiated and where the Bluetooth task and task scheduler are declared. It checks if a start command or a schedule command was received and awakens the task scheduler if it is. It also signalizes the Bluetooth task to send update

messages to the app whenever it receives a message from the Control task.

Auxiliary object classes:

These object classes describe each peripheral instantiated in the peripherals task. Every Object has distinct properties that must be correctly configured to work as described in section 3.2.5.

Some classes could be used directly from the [10]GpioZero library, while others had to be implemented using general-purpose I/O classes, such as the hall effect sensor and the vacuum motor ones.

Core tasks:

The core tasks regulate the functioning of the robot. Their communication and synchronized behavior keep the algorithm stable, allowing the main task of caching balls to run in parallel to multiple other tasks.

The **Bluetooth task** initializes the Bluetooth module in the Raspberry and listens for connections. Whenever a connection is created, all received messages, which arrive in a serialized *json* format, are then parsed and analyzed. If a start or schedule command is received, it sets flags to let the scheduler know when to start the Control task. If other robot commands are received, messages are sent to the control task via its message queue.

The **task Scheduler** is responsible for creating, scheduling, and/or awakening the control task. Whenever a start or schedule command is received, the main loop calls the task scheduler, which uses the standard Python library *schedule* to schedule the call of a function on the time given by the user.

The **Peripherals task** is responsible for keeping all the peripherals information updated for the control task to use. It declares all the used peripherals via the [10]GpioZero peripherals library and the [11]Adafruit ADC and IMU communication libraries. The infrared sensors are configured to generate interruptions for the ball collection counter. All other sensors are read via polling.

Finally, the **Control task** is what can be considered as the cognitive part of the robot. All its movement logic, data retrieving, image processing, and state management are done inside this class. The main algorithm for fetching balls is a state machine that takes all the peripheral data to assert a current state and a future state for the robot. The diagram in figure 14 represents all of the states the robot can be in and all the conditions for transitions to happen.

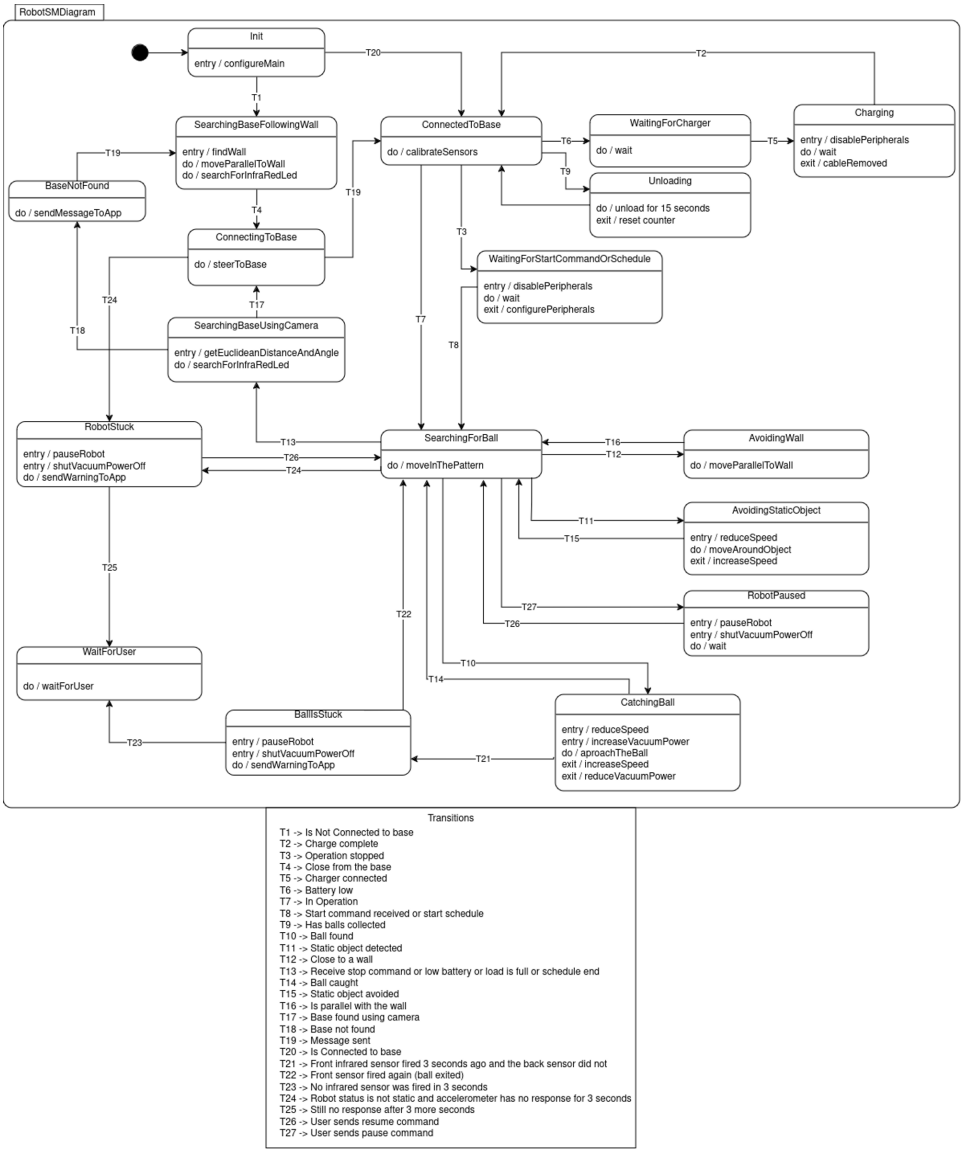


Figure 14: Ash state transitions for the controller task state machine.

To achieve such features, sub-tasks had to be created to decouple core functionalities so that they wouldn't run into any conflicts. The sub-tasks created were:

- **Bluetooth Handler:** A thread responsible for checking the message queue for command messages arriving from the Bluetooth task and setting the corresponding variables. It also sends periodic messages to the main loop to update the mobile application UI with the information read by the sensors;

- **Odometry:** A constantly updating odometry measurement that does its calculus based on the encoder readings from the wheel's motor encoders. Used to estimate the rotation and distance of the robot from the return base;
- **Ball detection:** A thread that uses the camera inputs and [12]OpenCV library to detect the table tennis balls, make distance measurements based on them and go after them to fetch the balls as desired;
- **Static object detection:** A thread that also uses the camera inputs to detect static objects and avoid them or use them to complement the algorithm further.

The Odometry is based on a constant reading of the wheel encoders that, for each iteration, takes the difference in pulses of each wheel compared to the last iteration and transforms it into traveled distance, allowing the calculation of angle and position relative to the base.

The ball detection feature took several steps to be accomplished. After installing the [12]OpenCv library, the first step was to create a curve of distance (cm) X ball radius (pixels) so that the ball distance estimation could work. To do that, pictures were taken with a table tennis ball fixed in known distances, as seen in Figure 15.

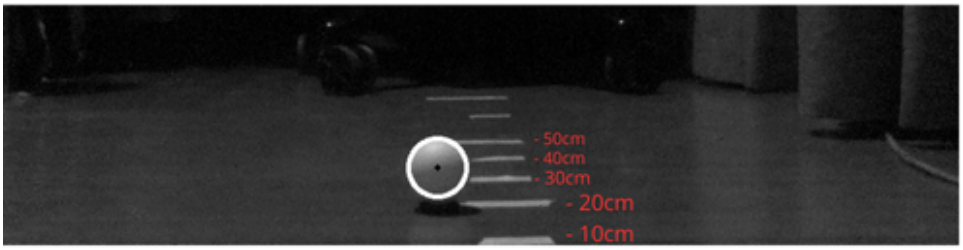


Figure 15: Ball in fixed positions to create the distance curve.

With the distance outputs obtained from the process above, the final distance fit curve used in the algorithm was achieved, as shown in Figure 16.

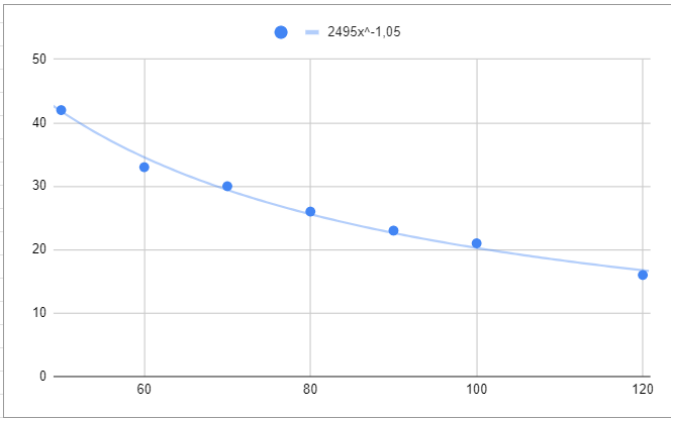


Figure 16: Final distance (cm) X ball radius (pixels) fit curve.

After calculating the distance curve, the next step was to calculate the angle of the ball relative to the center of the robot. This determines the angle the robot must turn to fetch a ball. The distance between the ball's center and the image's center must be calculated to do that.

To do so, the distance d_{est} from the camera center to the ball is calculated using the fit curve from image 16. After that, considering that the ball has a radius of 2 cm, which translates to a certain amount of pixels, it is possible to obtain the estimated horizontal distance to the image center (d_h) by comparing the pixel count. With all this information and the distance of a camera from the center of the robot, the estimation of the Θ_{cf} angle between the ball and the center of the robot can be done, as shown in Figure 17 and equation 1.

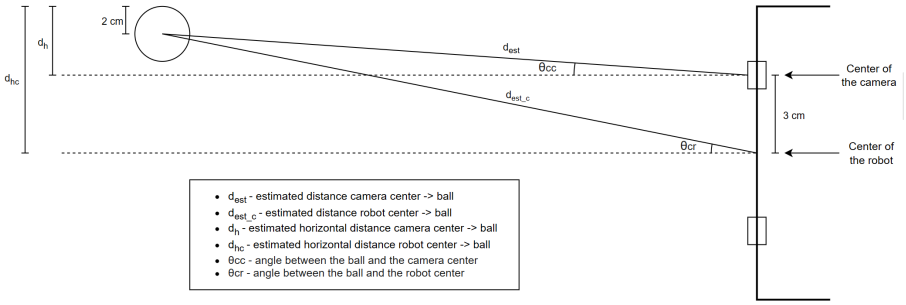


Figure 17: Ball angle relative to the robot center.

$$d_h = \frac{2}{pixels_{ballRadius}} \cdot pixels_{ballCenter}$$

$$\Theta_{cc} = sen^{-1}(\frac{d_h - 2}{d_est}) \quad (1)$$

Finally, by being able to detect balls and knowing their distance and angle relative to the robot, the only remaining part is rotating the robot and moving it forward to pull the balls with the vacuum, following a predefined algorithm.

3.3.2 Mobile Application

The ASH mobile application connects with the ASH - the table tennis ball fetcher robot. This app enables users to control and monitor the robot's actions. The following details describe the class diagram, the data exchanged between the mobile application and the robot via Bluetooth, the functions of each screen, and their related activities.

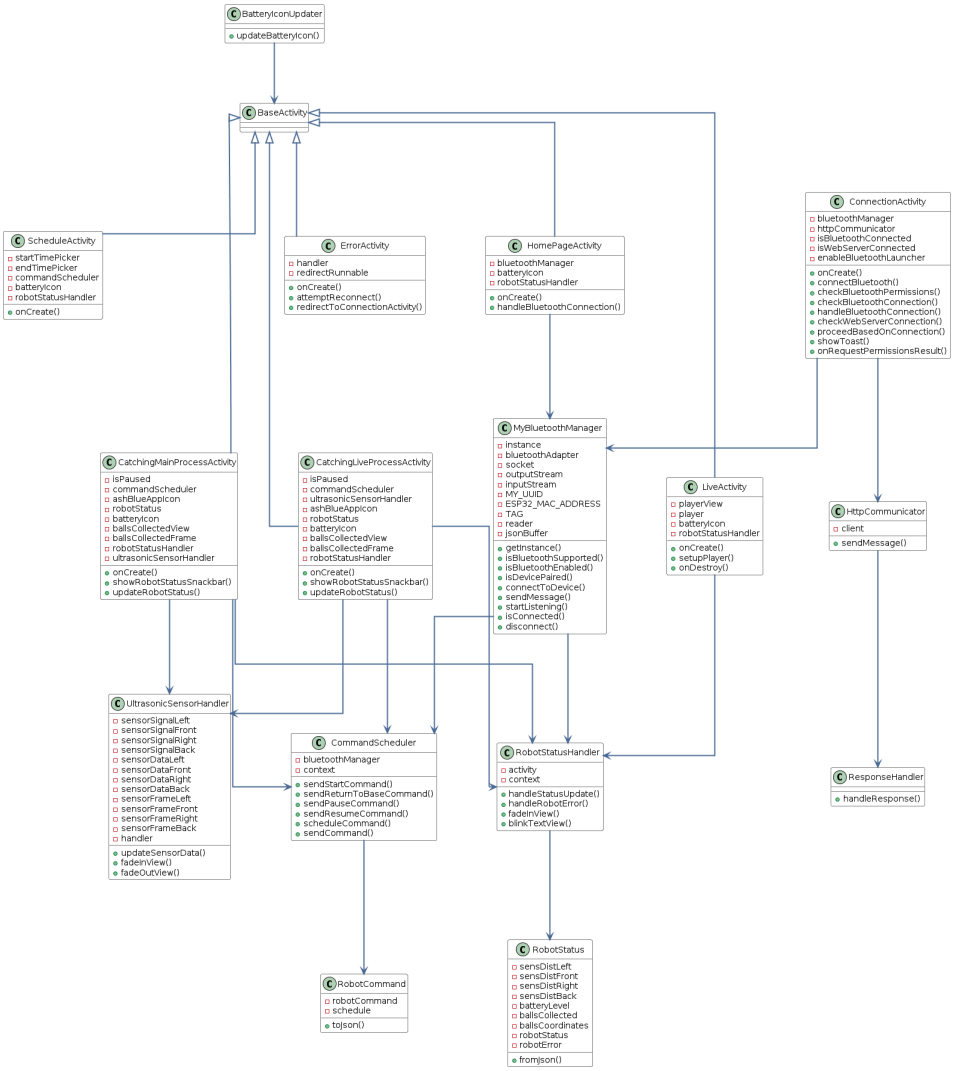


Figure 18: App Class Diagram

Bluetooth Data Exchange The ASH mobile application communicates with the robot via Bluetooth, sending commands and receiving status updates. Below are the details of the data exchanged:

- **Data Sent to the Robot:**
  - **Commands:** The application sends various commands to the robot, such as:
    - `start`: Initiates the ball fetching process.
    - `return_to_base`: The robot returns to its base.
    - `pause`: Pauses the current operation.
    - `resume`: Resumes a paused operation.
    - `schedule`: Sends a schedule for the fetching process with specified start and end times.
- **Data Received from the Robot:**
  - **Status Updates:** The application receives JSON formatted status updates from the robot, which include:
    - `sens_dist_left`: Distance measurement from the left sensor.
    - `sens_dist_front`: Distance measurement from the front sensor.
    - `sens_dist_right`: Distance measurement from the right sensor.
    - `sens_dist_back`: Distance measurement from the back sensor.
    - `battery_level`: Current battery level of the robot.
    - `balls_collected`: Number of balls collected.
    - `balls_coordinates`: List of coordinates of the balls detected by the robot.
    - `robot_status`: Current status of the robot (e.g., collecting balls, returning to base).
    - `robot_error`: Error message if the robot encounters an issue.

The ‘RobotStatusHandler‘ class manages status updates and adjusts the user interface accordingly. The ‘CommandScheduler‘ class transmutes commands to the robot, ensuring seamless communication between the mobile application and the robot. The ‘MyBluetoothManager‘ class oversees the Bluetooth connection and facilitates data exchange.

Connection Activity The connection screen is displayed when the application is launched. It features the ASH robot logo and a Bluetooth icon indicating the connection status.(See Figure 19a)

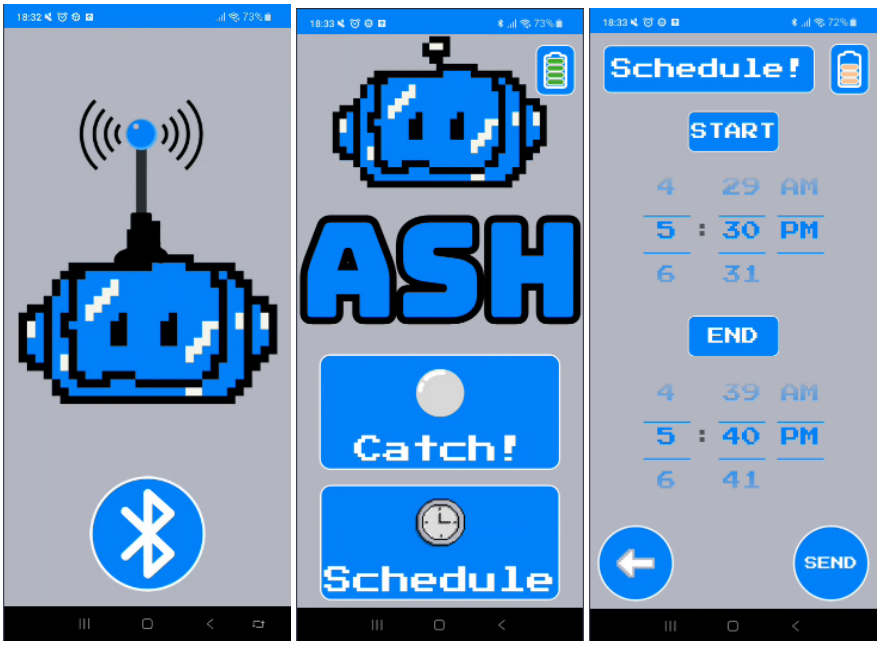
Home Page The home page provides options for the user to start or schedule the ball fetching process. Displays the ASH robot logo, battery icon, and two buttons: "Catch!" to start fetching balls and "Schedule" to schedule the fetching process. (See Figure 19b)

Schedule Activity The schedule activity allows users to set a start and end time for the ball fetching process. Provides time pickers for selecting start and end times and buttons to send the schedule to the robot. (See Figure 19c)

Catching Process Screen This screen is displayed when the ball fetching process is active. It shows the robot's status and sensor data. Displays sensor data from all four directions (left, front, back, right), the current status of the robot, and buttons to pause, resume, or return the robot to the base. It also has a button to view the Field of View (FOV). (See Figure 20a)

Field of View (FOV) Screen The FOV screen visually represents the robot's surroundings and the positions of detected balls. Displays a map showing the positions of balls detected by the robot's sensors. (See Figure 20c)

Error Screen When the robot encounters an issue, an error screen is displayed, showing different messages based on the specific error, such as 'base not found,' 'robot stuck,' or 'ball stuck in the tube.' The screen also includes a reconnect button, providing a comeback solution if the problem is resolved. (See Figure 21)

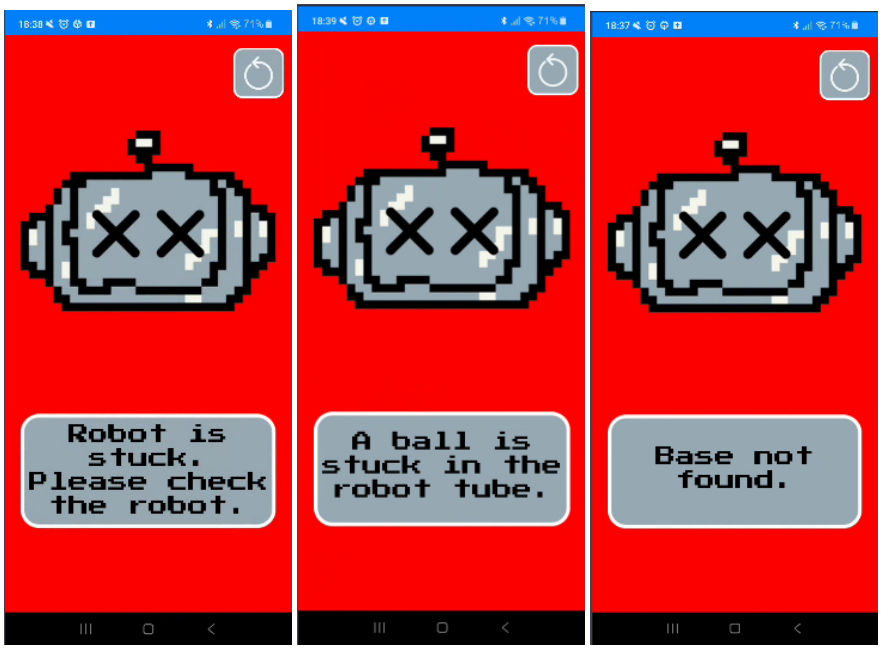


(a) Connection Activity (b) Home Screen Activity (c) Schedule Activity

Figure 19: Activities

(a) Catching Main Activity (b) Catching Live Activity (c) FOV Activity

Figure 20: User main Activities



(a) Error screen when the robot got stuck (b) Error screen when the ball got stuck (c) Error screen when the base couldn't be found

Figure 21: Error Activities

4 Results

4.1 Budget

The project's original budget was R$1200. Additional expenses were incurred to manage risks stemming from the project's complexity. Despite efforts to optimize resources and control costs, the need for spare parts and other foreseen adjustments increased the total expenses. As a result, the final budget amounted to R$1394.49.

Item	**Project Relation**	**Cost (R$)**
Raspberry Pi 5	Original Estimation	311.00
Structure	Original Estimation	200.00
Keyed source	Original Estimation	39.00
Batteries	Original Estimation	58.00
Spare Battery	Risk Mitigation	29.00
Voltage and current regulator	Original Estimation	15.99
Motors	Original Estimation	150.00
Servo Mg995	Original Estimation	23.10

IMU	Original Estimation	90.00
Infrared sensors	Original Estimation	10.00
Ultrasonic Sensors	Original Estimation	34.00
IMX219-83 Stereo Camera	Original Estimation	90.00
Electronic Components	Original Estimation	36.00
Spare Electronic Components	Risk Mitigation	36.00
Vacuum Cleaner	Original Estimation	96.00
Spare Vacuum Cleaner	Risk Mitigation	110.00
Swivel Wheel	Original Estimation	7.50
Wheel	Original Estimation	12.10
Infrared Led 3W	Original Estimation	3.00
Spare Infrared Led 3W	Risk Mitigation	13.00
Analog-digital conversion module	Original Estimation	9.00
Total		**1394.49**

Table 2: Budget

4.2 Schedule

Each deliverable was assigned specific hours based on the complexity and requirements of the tasks involved. The Mechanical section (row 1) includes the design and assembly of the robot's structure. The Hardware section (row 2) covers the design and implementation of the electronic components. The Software section (row 3) encompasses both the embedded firmware and the mobile application development. Integration involves the assembly and testing of all components to ensure seamless operation. Presentation preview accounts for coordinating and recording project progress and outcomes. Defense counts on the last efforts for the defense of the project.

The following data represent the hours allocated and spent on each part of the project.

Name	**Estimated Hours**	**Worked Hours**
Mechanical Section	92	72
Hardware Section	102	87
Software Section	98	80
Integration 1	107	85
Integration 2	92	81
Final Integration	233	263
Presentation Preview	105	144
Defense	80	78
Total	**909**	**890**

Table 3: Schedule

Figure 22 shows the bar graph representing the estimated and worked hours for each deliverable.

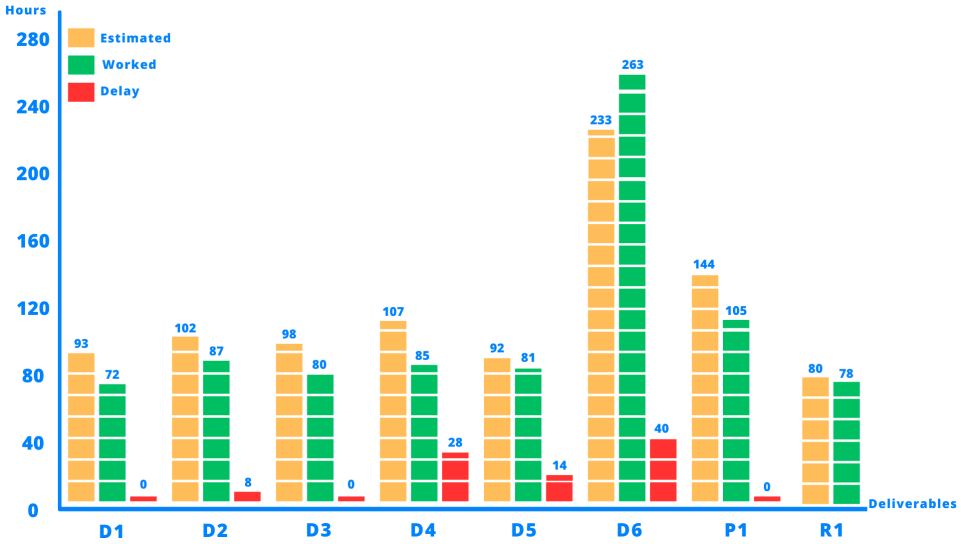


Figure 22: Delivered Hours Bar grap. Source: Authors (2024)

Figure 23 illustrates the functional requirements achieved, categorized into completed non-optional, uncompleted non-optional, completed optional, and uncompleted optional requirements. The uncompleted optional requirements include ASH being capable of unloading the balls on the base autonomously and transmitting the video from the camera through Wi-Fi for the mobile application. The completed optional requirement is that ASH should be able to dock in the base autonomously.

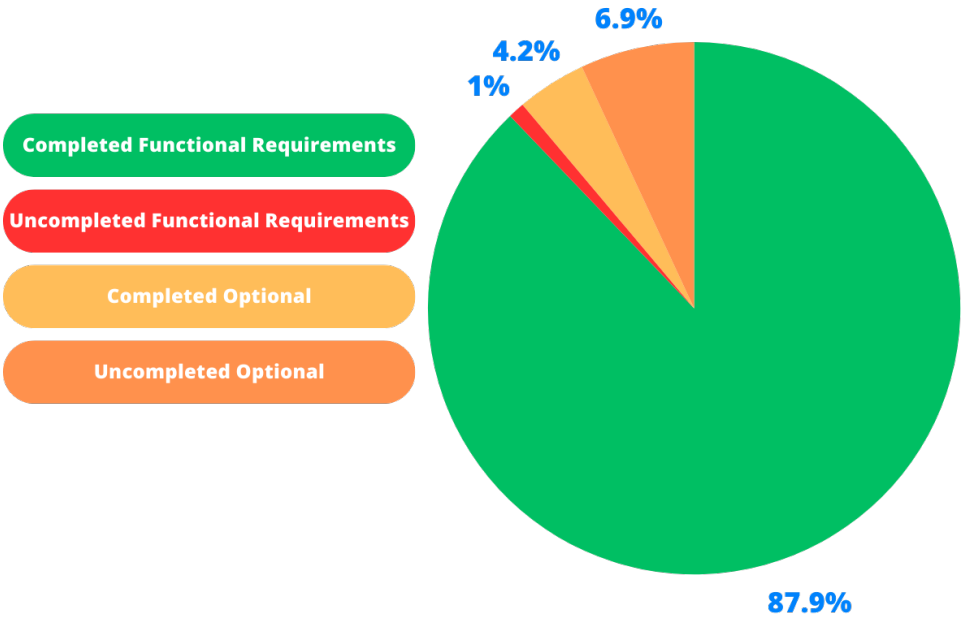


Figure 23: Functional Requirements Achieved. Source: Authors (2024)

5 Conclusions

This work proposed to develop ASH, an autonomous table tennis ball fetcher system to assist in training sessions. In order to attain the requirements presented in section 2, the project development, detailed in section 3, was a long and complex process, approaching it in four different pillars: a) Mechanical, creating a lightweight and durable structure to house the robot's components; b) Electronic, with the development of a power supply system, motor drivers and other pieces of hardware required for the robot operation; c) Firmware, integrating all of the robot's resources into a properly functional system, and; d) Mobile application, giving ASH a simple and complete control center for user operation.

The project schedule analysis shows 909 estimated hours versus 890 worked hours, indicating an underestimation in planning. Despite this, significant milestones were achieved. The bar graph shows the distribution of estimated and worked hours across deliverables. From all the functional requirements chart shows that 98.6% of obligatory requirements were completed, with 1.4% remaining uncompleted. For optional requirements, 35% were completed and 65% were not. This analysis highlights efficient resource management and effective task execution, resulting in a functional table tennis ball fetching robot. The completion rate of critical objectives confirms the project's success in meeting its primary goals and ensuring overall functionality.

Even with the positive outlook of the project, there are still some points that

could see some improvements. Since some optional features were not implemented, they are very likely to be the first step towards an even more complete and robust solution. The uncompleted optional requirements include ASH being capable of unloading the balls on the base autonomously and transmitting the video from the camera through Wi-Fi for the mobile application. Additionally, some areas could receive further investments in order to increase ASH's overall construction quality and maintainability, for example, with better cable routing (hardware).

References

[1] The international table tennis federation statutes. https://documents.ittf.sport/sites/default/files/public/2024-02/2024_ITTF_Statutes_clean_version.pdf, 2024. effective 1st January 2024.

[2] H1B1T. How to use the l298n motor driver module. https://www.hibit.dev/posts/89/how-to-use-the-l298n-motor-driver-module. Access in: 05/06/2024.

[3] RobotBanao. 5v dual-channel relay module shield with arduino. https://www.robotbanao.com/products/5v-10a-2-channel-2-ch-relay-module-shield-for-arduino. Access in: 05/06/2024.

[4] Single-channel relay module shield for arduino image. https://www-konga-com-res.cloudinary.com/w_400,f_auto,fl_lossy,dpr_3.0,q_auto/media/catalog/product/I/F/58064_1671628931.jpg. Access in: 05/06/2024.

[5] JGP. Bateria li-ion 18650 12v 2200mah placa bms + carregador. https://www.jgpassistencia.com.br/MLB-1781820987-bateria-li-ion-18650-12v-2200mah-placa-bms-carregador-_JM. Access in: 05/06/2024.

[6] Blog da Robótica. Como utilizar o sensor de obstáculo reflexivo infravermelho ir com arduino. https://www.blogdarobotica.com/2023/04/18/como-utilizar-o-sensor-de-obstaculo-reflexivo-infravermelho-ir-com-arduino/. Access in: 05/06/2024.

[7] UsinaInfo. Sensor ultrassônico de distância hc-sr04. https://www.usinainfo.com.br/sensor-ultrassonico/sensor-ultrassonico-de-distancia-hc-sr04-2295.html. Access in: 05/06/2024.

[8] Control Automático Educación. Motor dc con encoder – velocidad – posición. https://controlautomaticoeducacion.com/arduino/motor-dc-encoder/. Access in: 05/06/2024.

[9] Waveshare. Binocular camera module, dual imx219, 8 megapixels, applicable for jetson nano and raspberry pi, stereo vision, depth vision. https://www.waveshare.com/imx219-83-Stereo-camera.htm. Access in: 05/06/2024.

[10] Gpiozero. https://gpiozero.readthedocs.io/en/latest/. Raspberry pi GPIO library.

[11] Adafruit. https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/ads1015-slash-ads1115. Adafruit ADC and IMU libraries.

[12] Opencv. https://opencv.org/. Image processing python library.