

Table of Contents

1.0 Introduction	3
1.1 Background and Motivation	3
1.2 Project Scope and Objectives.....	3
2.0 Hardware Configuration	4
2.1 Component Overview	4
2.2 Circuit Diagram and Connections	5
2.2.1 Whisker Sensors	5
2.2.2 Light Source and Phototransistor	6
2.2.3 Feedback Components	7
2.3 Physical Construction and Setup	7
3.0 Algorithm Development and Implementation	8
3.1 System Architecture	8
3.2 Variable Initialisation.....	9
3.3 Navigation and Obstacle Avoidance	10
3.3.1 Movement Functions	10
3.3.2 Basic Navigation Logic.....	11
3.4 Corner Detection and Escape.....	12
3.5 Metal Detection Algorithm and Implementation.....	12
3.6 Feedback Mechanism Design and Implementation.....	13
3.7 Overall Implementation.....	14
4.0 Testing and Results	15
4.1 Testing Environment and Scenarios	15
4.2 Experimental Observations.....	15
4.2.1 Obstacle Avoidance Performance	15
4.2.2 Metal Detection Capabilities	16
4.2.3 Feedback System Effectiveness	16
5.0 Discussion and Future Work	16
5.1 Challenges and Problem-Solving Approaches.....	16
5.2 Experimental Limitations.....	16
5.3 Potential Technological Enhancements	17
6.0 Conclusion	17
7.0 Reference List.....	18
8.0 Appendices	19
8.1 Appendix 1 – Copy of Source Code (For Verification).....	19
8.2 Appendix 2 - Image Evidence	20
8.3 Appendix 3 - Video Evidence.....	21

Table of Figures

Figure 1 Breadboard Layout Diagram	5
Figure 2 Whisker Circuit Schematic (Left)	5
Figure 3 Whisker Circuit Schematic (Right)	5
Figure 4 Light Source LED Circuit Schematic	6
Figure 5 Phototransistor Circuit Schematic	6
Figure 6 Cyberbot Built-In LEDs and Piezospeaker Port (Parallax Inc., 2024).....	7
Figure 7 Top View of the Robot.....	7
Figure 8 Front View of the Robot	7
Figure 9 Simplified Flowchart For the Algorithm	8
Figure 10 Partial Code - Initialisation.....	9
Figure 11 Partial Code - Movement Functions	10
Figure 12 Partial Code - Basic Navigation Logic	11
Figure 13 Partial Code - Corner Detection and Escape.....	12
Figure 14 Partial Code - Metal Detection.....	13
Figure 15 Partial Code - Feedback Mechanism.....	13
Figure 16 Complete Program Code.....	14
Figure 17 Test Setup 1.....	15
Figure 18 Test Setup 2.....	15

List of Tables

Table 1 Component Overview.....	4
---------------------------------	---

1.0 Introduction

1.1 Background and Motivation

The applications of metal detection technology are significant, which across numerous industries. For example, security screening, construction, archaeological research, and even military operations. However, traditional metal detection can be time-consuming, labour intensive, and potentially dangerous in certain environments such as confined spaces with potential chemical leaks, due to that fact that it typically involves manual operation. Hence, by combining metal detection capabilities with autonomous navigation, the opportunities for efficiency, security, and thoroughness enhancement are presented in these applications. To illustrate that, there are several key advantages offered through the integration of robotics with metal detection. For example, through detection process automation, human exposure to hazardous environments can be minimised, together with the coverage consistency and detection rates improvement. To elaborate that, the injury or fatality risk can be drastically reduced by removing human operator from area such as minefields or industrial sites with chemicals leaks, which are extremely dangerous (Caleb, 2024). Furthermore, by equipping advanced sensors and algorithms, robotics systems can achieve real-time data analysis, noise filtering, thus enhancing the detection of subtle metallic signatures (Ali & Hakim Adil Kadhim, 2018). In short, this algorithmic processing has led to improved detection rates, particularly in environments with high levels of interference.

1.2 Project Scope and Objectives

This project aims to develop a robot that capable of autonomously perform the tasks including navigating an environment and avoiding obstacles, while providing appropriate feedback after detecting metallic object beneath it. It is selected the Cyberbot platform as the base for this project due to its flexibility, programmability, and integrated sensor capabilities. In brief, the project primary objectives included the configuration of the robot hardware components to achieve obstacle avoidance and metal detection. Next, it included the development of robust algorithms that can achieve autonomous navigation and obstacle avoidance, while also providing special handling on challenging scenarios such as corners. Following that, it is required for the implementation of effective metal detection through phototransistor-based sensing systems. Subsequently, the objectives also comprised designing and programming appropriate feedback mechanisms that incorporate both visual and auditory elements for metal detection. Moving on, it is required to conduct test on the robot in a simulated environments with obstacles and metallic object. Lastly, the performance of the robot is then analysed and identify for potential improvements.

The documentation will be structured in the way of starting with the introduction to the project background and objectives, followed by presenting the hardware configuration details of the robot, then documented with the justification on algorithm development and implementation, subsequently showcasing the experiments and results, then, moving on with comprehensive discussion on various aspects, including algorithm effectiveness, challenges encountered, and potential improvements, and finally with a conclusion to summarise the entire project.

2.0 Hardware Configuration

2.1 Component Overview

The robot is constructed using the Cyberbot platform with the following components as shown in *Table 1*.

Table 1 Component Overview

	Component	Description	Connection (Pin)
1.	Cyberbot Base	Provides the microcontroller and chassis for the robot	-
2.	Servo Motors	Two servo motors for differential drive management control	Left – P18 Right – P19
3.	Whisker Sensors	Two whisker switches for obstacle detection through physical contact	Left – P7 Right – P9
4.	Phototransistor	Used for metal detection by measuring reflectivity of surfaces	P0
5.	LED Light Source	Illuminates the surface for the phototransistor to enhance detection reliability	P3
6.	Piezospaker (Buzzer)	For Auditory Feedback when metal is detected	P22
7.	LED (Cyberbot Built-In)	For Visual Feedback when metal is detected	P20

The hardware configuration prioritises on ensuring reliable obstacle detection through robust whisker sensor implementation. Moreover, it also focuses on optimal positioning phototransistor and light source to ensure sensitive metal detection. Furthermore, for detection events, it also provides clear and noticeable feedback mechanisms.

2.2 Circuit Diagram and Connections

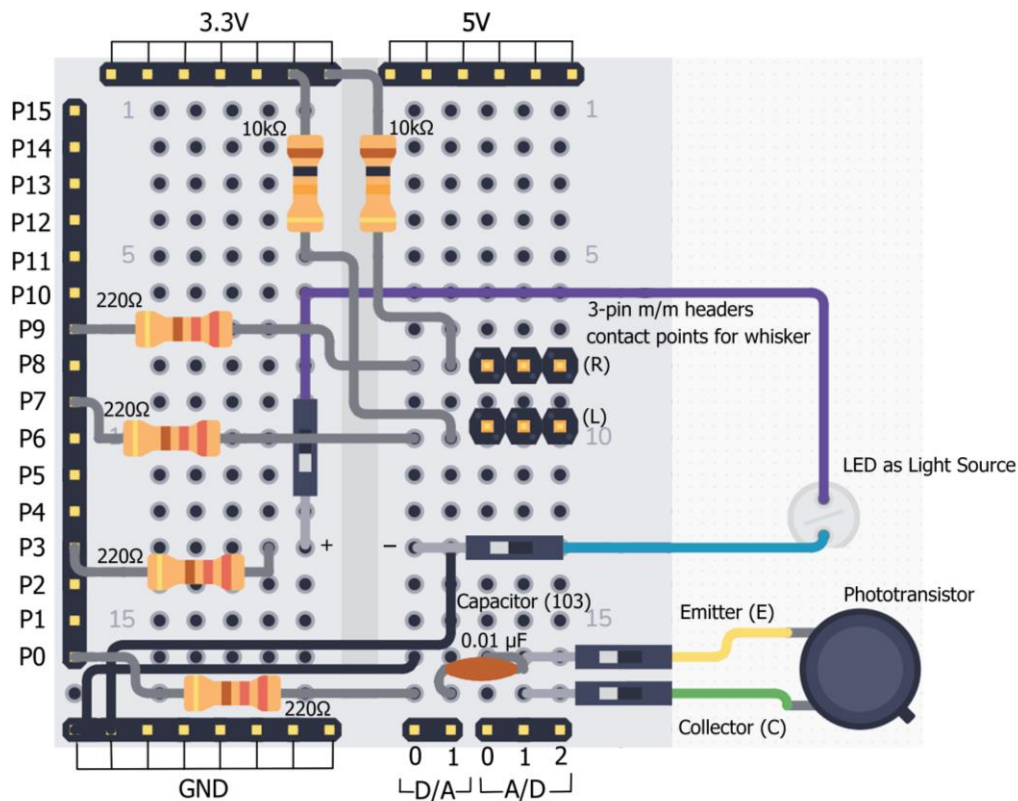


Figure 1 Breadboard Layout Diagram

The overall hardware configuration is illustrated in the diagram as shown in *Figure 1*. Following that, the key electrical connections are detailed below:

2.2.1 Whisker Sensors

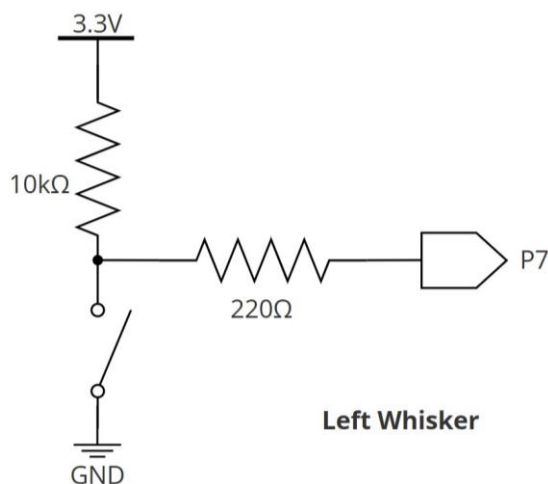


Figure 2 Whisker Circuit Schematic (Left)

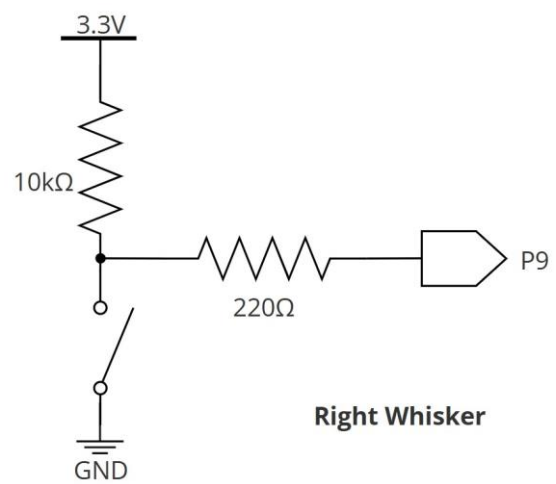


Figure 3 Whisker Circuit Schematic (Right)

As shown in *Figure 1*, there are two whisker switches that will be contacted with the 3-pin male-to-male headers that labelled as left (L) and right (R) when the whiskers are being pushed. The circuit schematics of both left and right whisker are shown in *Figure 2* and *Figure 3*. Left whisker is connected to pin P7 while right whisker is connected to pin P9. Each of them is connected with a current-limiting resistor (220Ω), and a pull-up resistor ($10k\Omega$) to the 3.3V

power supply. When a whisker is being pressed, it will then connect to ground (GND), thus providing a digital low signal (0), to the microcontroller. Conversely, when it is not pressed, the pull-up resistor ensures a digital high signal (1) is read. In short, the whisker circuit creates a voltage divider that prevents excessive current flow when the whisker is pressed, ensures clean digital signals with proper pull-up resistors as well as provides reliable state detection with minimal false readings.

2.2.2 Light Source and Phototransistor

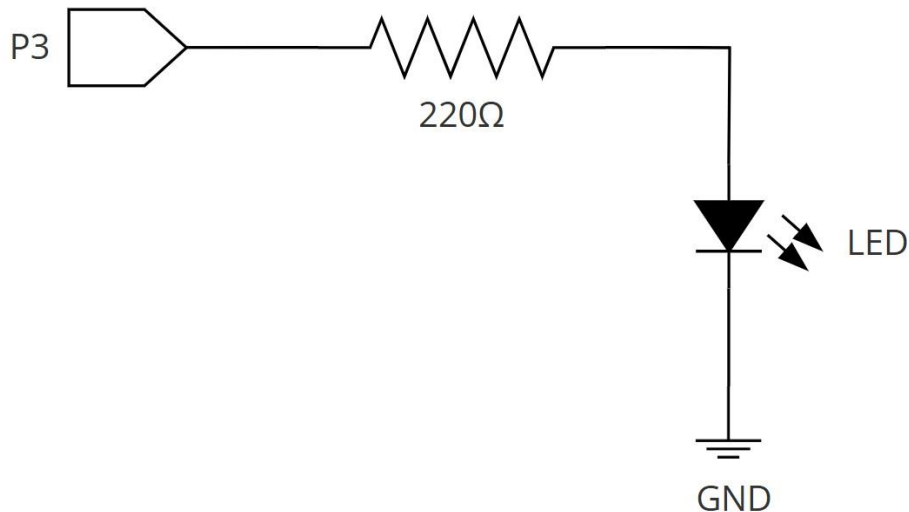


Figure 4 Light Source LED Circuit Schematic

As shown in *Figure 4*, the LED as the light source for phototransistor is connected to pin P3 through a current-limiting resistor ($220\ \Omega$) to the ground (GND).

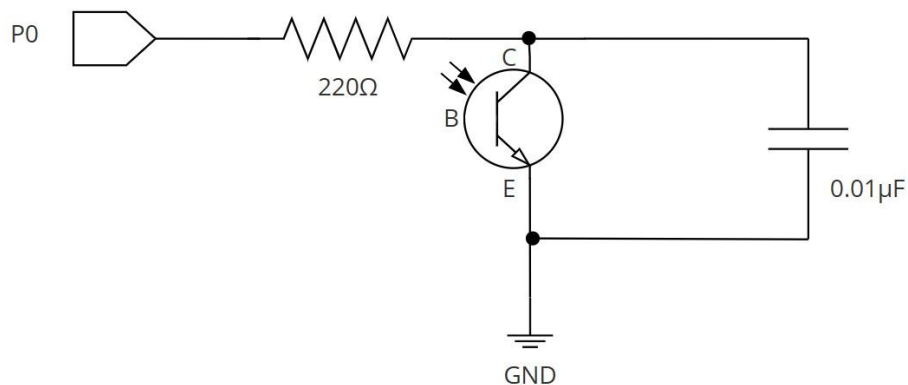


Figure 5 Phototransistor Circuit Schematic

On the other hand, as illustrated in *Figure 5*, the phototransistor is connected to pin P0 with a $220\ \Omega$ resistor and a $0.01\ \mu\text{F}$ capacitor to ground. For elaboration, this circuit is operating on the RC time measurement principle where the capacitor is initially charged, the phototransistor then discharges the capacitor at a rate proportional to the amount of light detected, and the time taken for the capacitor to discharge below a threshold value is being measured by the microcontroller (Parallax Inc., 2024). Therefore, a shorter discharge time will be indicating greater light is being detected, thus implying that potentially metal due to higher reflectivity as in this project context.

2.2.3 Feedback Components

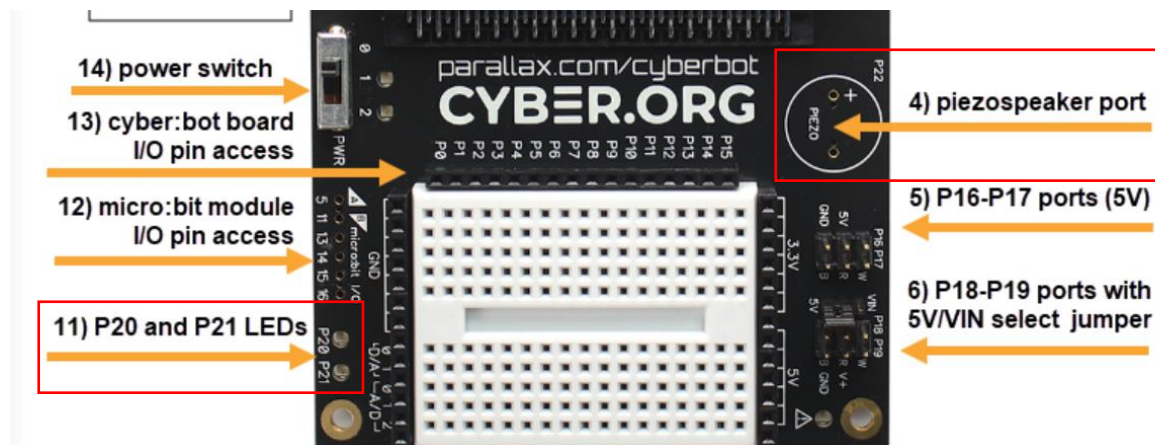


Figure 6 Cyberbot Built-In LEDs and Piezospeaker Port (Parallax Inc., 2024)

For the feedback components when metal is detected, it will be having both auditory and visual feedback that utilises the Cyberbot built-in LEDs and piezospeaker port as shown in **Error! Reference source not found.** that marked with red rectangle. In terms of the visual feedback, it is inserted a piezospeaker on the pin P22 port, that will act as a buzzer which capable of generating tones at specific frequencies. On the other hand, for the visual feedback, it will be utilising the tiny built-in LED, specifically the pin P20 LED.

2.3 Physical Construction and Setup

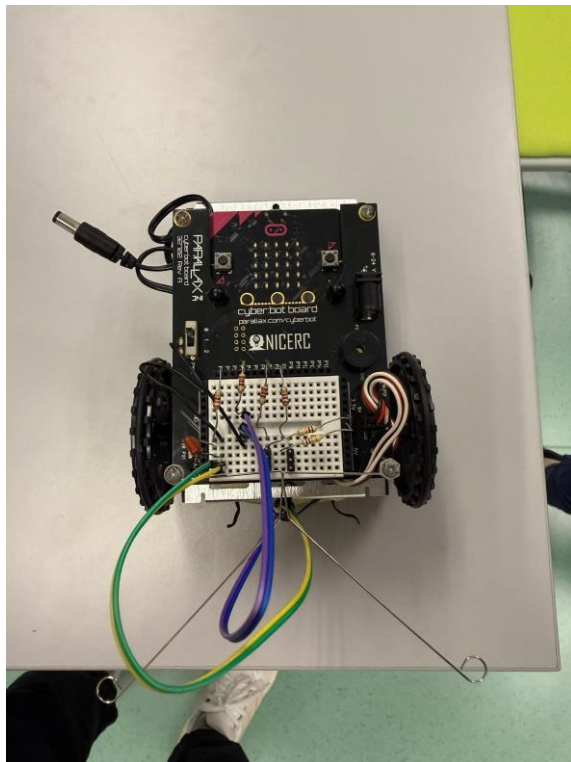


Figure 7 Top View of the Robot

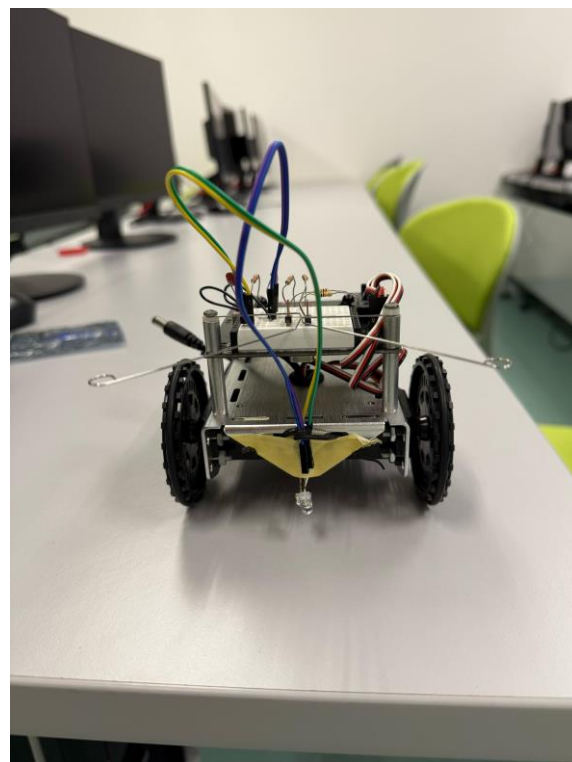


Figure 8 Front View of the Robot

The robot physical construction is designed to optimise capabilities on both navigating and metal detecting. For example, in order to achieve maximum coverage for obstacle detection, both left and right whisker are adjusted to extend from the front at approximately

45° angles as shown in *Figure 7*. Moreover, to enable the robot to detect reflective surfaces, which is the metal beneath it, the phototransistor is centrally located and positioned in front of the robot which is close to the underside of the robot as shown in *Figure 8*. In corresponding to that, the LED light source is angled to optimise illumination of the surface beneath by positioning it adjacent to the phototransistor. In essence, the components positioning is carefully considered to ensure the sensor coverage as well as minimise interference by allowing clean cable routing.

3.0 Algorithm Development and Implementation

3.1 System Architecture

The robot operates on a reactive control architecture with four key components, which including the sensor input processing, decision logic with state management, movement control, and feedback control. To illustrate that, the system continuously cycling through sensing, decision-making, and action execution without the needs on complex planning or mapping capabilities. The cycle will finally end when a metal is detected, and feedback is given. Moreover, a modular approach is utilised for the implementation, which focuses on separate functions for movement patterns and a main loop is used for processing the sensor input and the corresponds decision-making and action, which will be discussed in the below sections.

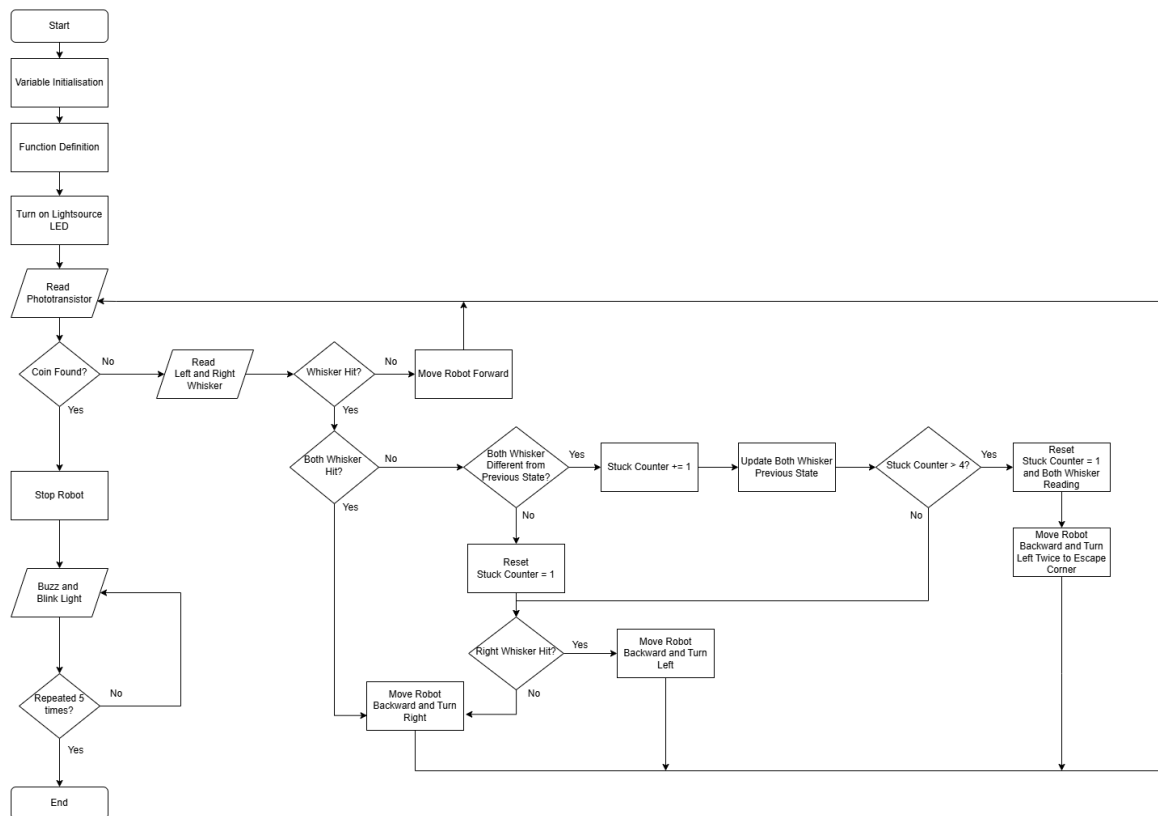


Figure 9 Simplified Flowchart For the Algorithm

The simplified flowchart is illustrated in the figure above to brief the basic logic of the algorithm.

3.2 Variable Initialisation

```
# Import the cyberbot library for controlling the robot
from cyberbot import *

# For coin detection
coin_found = False # Flag to indicate if a coin has been found

# For avoiding the robot stuck in corner
counter = 0 # Counter to track if the robot is stuck in a corner
w_l = 0 # State of the left whisker (0 = pressed, 1 = not pressed)
w_r = 0 # State of the right whisker
o_w_l = 0 # Previous state of the left whisker
o_w_r = 1 # Previous state of the right whisker
"""o_w_r is initialised to 1 to ensure the corner detection logic works correctly on the first iteration."""

# Light source for the phototransistor
bot(3).write_digital(1) # Turn on the LED (P3)
```

Figure 10 Partial Code - Initialisation

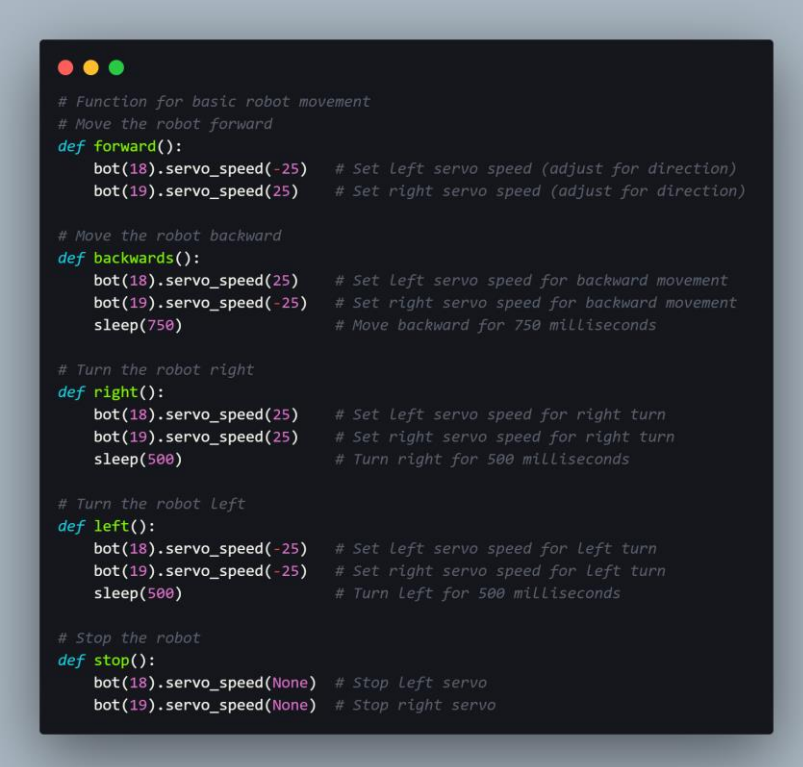
As shown in the code snippet above, there are several variable initialisation for the state management. For illustration, the robot's state is tracked through variables including `coin_found` to indicate metal detection status, while the other for stuck tracking. To elaborate that, `counter` tracks potential corner situations, `w_l` and `w_r` represent current whisker states, while `o_w_l` and `o_w_r` store previous whisker states for change detection. Furthermore, as written in the code comment, the initialisation of `o_w_r` to 1 ensures the corner detection logic functions correctly on the first iteration by providing a valid previous state for comparison. In addition, the light source LED for phototransistor is also initialised in this state to ensure it turns on all the time.

3.3 Navigation and Obstacle Avoidance

The design of navigation algorithm is focused on allowing the robot to move autonomously and effectively avoiding obstacles. Whisker sensors as mentioned in above section is the core of this algorithm on obstacles detection and appropriate evasive manoeuvre triggering.

3.3.1 Movement Functions

The implementation of robot's navigation capabilities is achieved through modular movement functions that encapsulate basic locomotion capabilities.



```
# Function for basic robot movement
# Move the robot forward
def forward():
    bot(18).servo_speed(-25) # Set Left servo speed (adjust for direction)
    bot(19).servo_speed(25) # Set right servo speed (adjust for direction)

# Move the robot backward
def backwards():
    bot(18).servo_speed(25) # Set Left servo speed for backward movement
    bot(19).servo_speed(-25) # Set right servo speed for backward movement
    sleep(750) # Move backward for 750 milliseconds

# Turn the robot right
def right():
    bot(18).servo_speed(25) # Set Left servo speed for right turn
    bot(19).servo_speed(25) # Set right servo speed for right turn
    sleep(500) # Turn right for 500 milliseconds

# Turn the robot Left
def left():
    bot(18).servo_speed(-25) # Set Left servo speed for left turn
    bot(19).servo_speed(-25) # Set right servo speed for Left turn
    sleep(500) # Turn Left for 500 milliseconds

# Stop the robot
def stop():
    bot(18).servo_speed(None) # Stop Left servo
    bot(19).servo_speed(None) # Stop right servo
```

Figure 11 Partial Code - Movement Functions

As illustrated in the figure above, these movement parameters such as the speed and duration are carefully tuned to ensure the robot avoid the obstacles efficiently. For example, 750 milliseconds (ms) are set for backward movement duration to ensure sufficient distance when backing away from obstacles. Moreover, it is also set as 500ms duration for both left and right turns to ensure appropriate turning angles and thus maintaining smooth transitions between manoeuvres. Besides, the forward movement operates continuously at a -25/25 speed (left/right motors), while turning movements use equal speed in both motors, which is 25/25 for right turns, -25/-25 for left turns.

3.3.2 Basic Navigation Logic

The obstacle avoidance algorithm is implemented in the main control loop, utilising whisker sensor input to trigger appropriate evasive manoeuvres following the rules as below:

- If both whiskers are pressed, move backward and turn right
- If only the right whisker is pressed, move backward and turn left
- If only the left whisker is pressed, move backward and turn right
- If no whiskers are pressed, move forward



```
# Read whisker switches for robot navigation
w_l = bot(7).read_digital() # Left whisker (P7)
w_r = bot(9).read_digital() # Right whisker (P9)

# Whisker-based obstacle avoidance
# Both whiskers hit
if w_l == 0 and w_r == 0:
    backwards() # Move the robot backwards using the defined function
    right() # Make the robot turns right using the defined function
# Right whisker hit
elif w_r == 0:
    backwards() # Move the robot backwards using the defined function
    left() # Make the robot turns left using the defined function
# Left whisker hit
elif w_l == 0:
    backwards() # Move the robot backwards using the defined function
    right() # Make the robot turns right using the defined function
# No obstacle
else:
    forward() # Move the robot forward
```

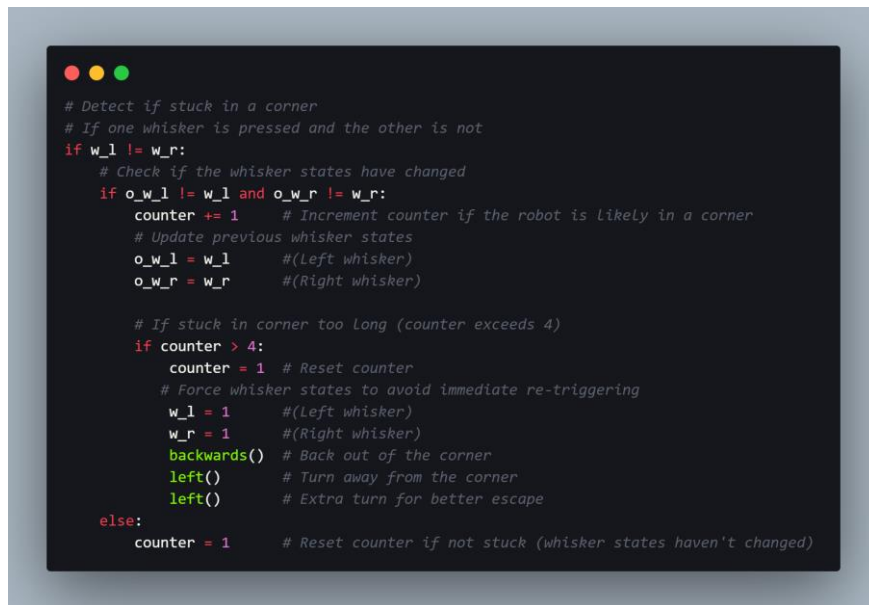
Figure 12 Partial Code - Basic Navigation Logic

As shown in Figure 11 *Figure 12*, the left and right whisker sensor input is read from pin P7 and pin P9, when the whisker is pressed, the input will read as 0, else it will read as 1 if it is not pressed. Even though it is a simple logic, however it effectively allows the robot to navigate around obstacles by retreating from direct collisions, turning away from the side where an obstacle is detected, and proceeding forward when the path is clear.

3.4 Corner Detection and Escape

There is a significant enhancement to the basic navigation logic, which is adding the corner detection mechanism to prevent the robot from trapping in the corners. For illustration, it might repeatedly hit one wall, then back up and turn, and hit another wall again, which resulting in endless cycle.

In brief, the corner detection algorithm works by tracking changes in whisker states between iterations. To elaborate that, it will be incrementing a counter when both whiskers change state while obstacle detection is still showing, and it will finally triggering an extended escape manoeuvre when the counter exceeds a threshold of 4.



```
# Detect if stuck in a corner
# If one whisker is pressed and the other is not
if w_l != w_r:
    # Check if the whisker states have changed
    if o_w_l != w_l and o_w_r != w_r:
        counter += 1 # Increment counter if the robot is likely in a corner
        # Update previous whisker states
        o_w_l = w_l # (Left whisker)
        o_w_r = w_r # (Right whisker)

    # If stuck in corner too long (counter exceeds 4)
    if counter > 4:
        counter = 1 # Reset counter
        # Force whisker states to avoid immediate re-triggering
        w_l = 1 # (Left whisker)
        w_r = 1 # (Right whisker)
        backwards() # Back out of the corner
        left() # Turn away from the corner
        left() # Extra turn for better escape
    else:
        counter = 1 # Reset counter if not stuck (whisker states haven't changed)
```

Figure 13 Partial Code - Corner Detection and Escape

As illustrated in the implementation above, the extended escape manoeuvre consists of moving backward to create adequate distance from the corner, then it will be executing two consecutive left turns for larger turn angle to better escape the corner. Moreover, in order to prevent immediate re-triggering, it also involves resetting the corner detection counter and forcing whisker states to non-pressed. Hence, the robot's ability to navigate complex environments is significantly enhanced by this approach, as it is capable of recognising and escaping from potential trap situations.

3.5 Metal Detection Algorithm and Implementation

The phototransistor's ability is leveraged in this metal detection algorithm to detect differences in surface reflectivity. Metallic objects, which is simulated by a reflective coin in the testing environment of this project will reflect more light than non-metallic surfaces, thus allowing for detection through light-based sensing.

The detection process uses RC time measurement as mentioned in section [2.2.2 Light Source and Phototransistor](#), where the phototransistor measures the amount of reflected light. Hence, a threshold value of 200 is determined through extensive experimenting and calibration to distinguish between reflective and non-reflective surfaces while minimising false positives and negatives.

```
# Phototransistor setup
bot(0).write_digital(1) # Turn on the phototransistor (P0)
qt = bot(0).rc_time(1) # Read the phototransistor value

if qt < 200: # Check if the phototransistor detects a coin (Low reading indicates a shiny surface which means coin is detected)
    coin_found = True # Set the coin detection flag to true
```

Figure 14 Partial Code - Metal Detection

As illustrated in above diagram, the code sets the phototransistor pin, P0 to output mode and writes a high value to charge the capacitor for 1 millisecond to make it fully charged. It then measures the time for the capacitor to discharge below the digital input threshold. Following that, it compares the measured time against the calibrated threshold of 200, when the measured value indicates a reflective surface, it will then set the coin_found flag to true.

3.6 Feedback Mechanism Design and Implementation

The robot will provide both visual and auditory feedback to alert users when metal is detected. This approach ensures that detection events are clearly communicated regardless of the environment.

```
# If the robot has detected the coin
if coin_found:
    stop() # Immediately stop
    for _ in range(5): # Blink LED and buzzer 5 times
        bot(22).tone(5000, 500) # Buzz as coin found (5000 Hz for 500 ms) (P22)
        # Turn on and off LED for blinking to indicate coin found
        bot(20).write_digital(1) # Turn on the LED (P20)
        sleep(100) # LED on for 100 milliseconds
        bot(20).write_digital(0) # Turn off the LED (P20)
        sleep(500) # Pause for 500 milliseconds between alert cycles
```

Figure 15 Partial Code - Feedback Mechanism

As shown in *Figure 15*, the feedback sequence work in the way of immediately stopping the robot's movement to mark the detection location. It then followed by five cycles of combined visual and auditory feedback. To illustrate that, each cycle includes buzzer activation at 5000Hz for 500ms and LED illumination for 100ms during the buzzer activation, with a 500ms pause between cycles, which resulted in 5 blinking light and buzzing sound as alerts to user.

In addition, several user experience considerations are incorporated in the feedback design. To elaborate it, the selection of 5000Hz frequency for the buzzer is due to the fact that it is within the optimal range for human hearing. Moreover, the multiple feedback cycles is to ensure the event is noticed. Last but not least, as briefly mentioned above, the synchronisation of both visual and auditory feedback reinforces the detection event, and the immediate stopping of the robot helps users identify the exact location of detection thus spotting the metal.

3.7 Overall Implementation

By integrating the sensor reading, decision logic on metal detection and obstacle avoidance with state management, movement control, and feedback control components mentioned above together in a continuous loop, the complete implementation is formed:

```

1 # Import the cyberbot library for controlling the robot
2 from cyberbot import *
3
4 # For coin detection
5 coin_found = False # Flag to indicate if a coin has been found
6
7 # For avoiding the robot stuck in corner
8 counter = 0 # Counter to track if the robot is stuck in a corner
9 w_l = 0 # State of the left whisker (0 = pressed, 1 = not pressed)
10 w_r = 0 # State of the right whisker
11 o_w_l = 0 # Previous state of the left whisker
12 o_w_r = 1 # Previous state of the right whisker
13 """o_w_r is initialised to 1 to ensure the corner detection logic works correctly on the first iteration."""
14
15 # Function for basic robot movement
16 # Move the robot forward
17 def forward():
18     bot(18).servo_speed(-25) # Set left servo speed (adjust for direction)
19     bot(19).servo_speed(25) # Set right servo speed (adjust for direction)
20
21 # Move the robot backward
22 def backwards():
23     bot(18).servo_speed(25) # Set left servo speed for backward movement
24     bot(19).servo_speed(-25) # Set right servo speed for backward movement
25     sleep(750) # Move backward for 750 milliseconds
26
27 # Turn the robot right
28 def right():
29     bot(18).servo_speed(25) # Set left servo speed for right turn
30     bot(19).servo_speed(25) # Set right servo speed for right turn
31     sleep(500) # Turn right for 500 milliseconds
32
33 # Turn the robot left
34 def left():
35     bot(18).servo_speed(-25) # Set left servo speed for left turn
36     bot(19).servo_speed(-25) # Set right servo speed for left turn
37     sleep(500) # Turn left for 500 milliseconds
38
39 # Stop the robot
40 def stop():
41     bot(18).servo_speed(None) # Stop left servo
42     bot(19).servo_speed(None) # Stop right servo
43
44 # Light source for the phototransistor
45 bot(1).write_digital(1) # Turn on the LED (P3)
46
47 while True:
48     # Phototransistor setup
49     bot(0).write_digital(1) # Turn on the phototransistor (P0)
50     qt = bot(0).rc_time(1) # Read the phototransistor value
51
52     if qt < 200: # Check if the phototransistor detects a coin (Low reading indicates a shiny surface which means coin is detected)
53         coin_found = True # Set the coin detection flag to true
54
55     # If the robot has detected the coin
56     if coin_found:
57         stop() # Immediately stop
58         for i in range(5):
59             bot(22).tone(5000, 500) # Blink LED and buzzer 5 times
60             # Buzz as coin found (5000 Hz for 500 ms) (P22)
61             # Turn on and off LED for blinking to indicate coin found
62             bot(20).write_digital(1) # Turn on the LED (P20)
63             sleep(100) # LED on for 100 milliseconds
64             bot(20).write_digital(0) # Turn off the LED (P20)
65             sleep(500) # Pause for 500 milliseconds between alert cycles
66
67     # Read whisker switches for robot navigation
68     w_l = bot(7).read_digital() # Left whisker (P7)
69     w_r = bot(9).read_digital() # Right whisker (P9)
70
71     # Detect if stuck in a corner
72     # If one whisker is pressed and the other is not
73     if w_l != w_r:
74         # Check if the whisker states have changed
75         if o_w_l != w_l and o_w_r != w_r:
76             counter += 1 # Increment counter if the robot is likely in a corner
77             # Update previous whisker states
78             o_w_l = w_l # (Left whisker)
79             o_w_r = w_r # (Right whisker)
80
81         # If stuck in corner too long (counter exceeds 4)
82         if counter > 4:
83             counter = 1 # Reset counter
84             # Force whisker states to avoid immediate re-triggering
85             w_l = 1 # (Left whisker)
86             w_r = 1 # (Right whisker)
87             backwards() # Back out of the corner
88             left() # Turn away from the corner
89             left() # Extra turn for better escape
90         else:
91             counter = 1 # Reset counter if not stuck (whisker states haven't changed)
92
93     # Whisker-based obstacle avoidance
94     # Both whiskers hit
95     if w_l == 0 and w_r == 0:
96         backwards() # Move the robot backwards using the defined function
97         right() # Make the robot turns right using the defined function
98     # Right whisker hit
99     elif w_r == 0:
100         backwards() # Move the robot backwards using the defined function
101         left() # Make the robot turns left using the defined function
102     # Left whisker hit
103     elif w_l == 0:
104         backwards() # Move the robot backwards using the defined function
105         right() # Make the robot turns right using the defined function
106     # No obstacle
107     else:
108         forward() # Move the robot forward

```

Figure 16 Complete Program Code

In summary, the reactive control model is implemented through this structure, which will cycle continuously through reading phototransistor and whiskers (sensing), detecting metal and navigating (decision-making), and controlling the movement (action), and finally end the cycle and provide feedback when a metal is detected, thus enabling the robot to operate autonomously in complex environments.

4.0 Testing and Results

4.1 Testing Environment and Scenarios

In order to provide a controlled yet challenging space for the robot evaluation, the testing environment was constructed with consideration. For elaboration, to enhance the contrast for metal detection, a dark grey macaron paper is placed as the testing environment, which featuring a flat surface with dark, low-reflective background. Moreover, the environment is incorporated with obstacles like box, bottle, and walls to challenge the robot's navigation algorithms of the robot.

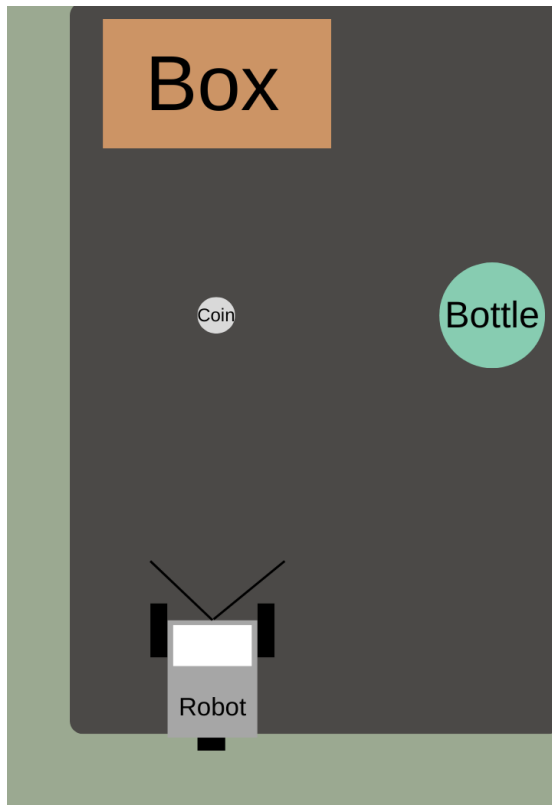


Figure 117 Test Setup 1

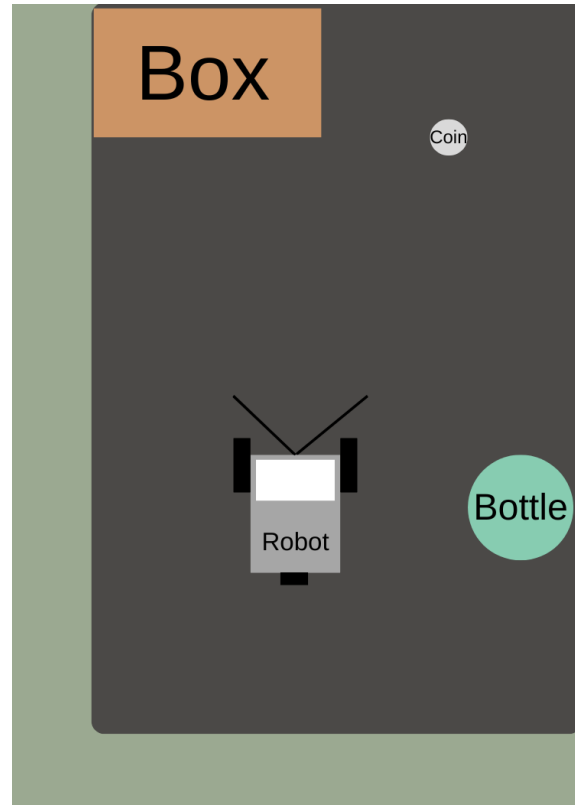


Figure 118 Test Setup 2

The robot is mainly tested with two test scenarios that meticulously crafted to evaluate its core capabilities. The first scenario setup is focused on testing the metal detection function along with the feedback mechanism by precisely placing a reflective coin at a predetermined location as shown in *Figure 117*. While on the other hand, as illustrated in *Figure 118*, the second scenario is concentrated on validating the obstacle avoidance capability of the robot by strategically positioning a stationary box within the robot's anticipated path. The second test scenario have also placed a reflective coin in the corner to assess whether the robot will detect the metal and provide feedback after avoiding the obstacle. In essence, a direct evaluation of the robot's sensory and navigational performance is enabled through this meticulously designed environment under controlled conditions.

4.2 Experimental Observations

The video evidence of the experiments conducted are provided in section **8.3 Appendix 3 - Video Evidence**.

4.2.1 Obstacle Avoidance Performance

The obstacle avoidance test demonstrated in Test Setup 2 have presented the remarkable capability of the robot to detect and navigate around stationary obstacle, which is

the box in this case. To illustrate that, the whisker-based navigation system has emerged to be particularly effective, where upon encountering the strategically placed box, an instantaneous avoidance response is triggered to prompt the robot to alter its trajectory. Hence, this preliminary assessment suggested the potential efficacy of the reactive navigation approach, indicating a robust foundational mechanism for autonomous navigation (Lindqvist, 2023).

4.2.2 Metal Detection Capabilities

A compelling evidence of the robot's sophisticated sensory capabilities is presented by the metal detection experiment demonstrated in both Test Setup 1 and Test Setup 2. To elaborate that, the robot successfully initiated its detection protocol when it navigates to the point that reflective coin is beneath the phototransistor. It immediately stops at the point of detection and simultaneously activating visual and auditory feedback channels to signal the detection event with remarkable precision.

4.2.3 Feedback System Effectiveness

The multimodal feedback system that combines LED visual indicators with a 5000Hz auditory signal have effectively alert users on detection event. For elaboration, a comprehensive user awareness is ensured through this integrated approach by engaging multiple sensory channels, thus the probability of detection notification across diverse environmental conditions and user contexts can be maximised (Ogunsina et al., 2024).

5.0 Discussion and Future Work

5.1 Challenges and Problem-Solving Approaches

Throughout the development of this robotic prototype, a significant technical challenge has been encountered. To illustrate that, an infrared (IR) has initially been selected and deployed as the sensor for the robot's navigation system. However, during preliminary testing phase, it is observed that the robot keeps moving backward even though there is no obstacle in front of it. After multiple troubleshooting, it is figured out that one of the infrared sensor is impaired, resulting in false sensor reading and hence compromising the overall navigation reliability.

In response to this challenge, it is pivoted to utilise whisker sensor for the navigation system. While this alternative approach represents a significant departure from the original design, including the hardware configuration as well as the algorithm, however it provided a navigation mechanism that is more mechanical and potentially more reliable. Nevertheless, the transition from IR to whisker sensors was not without consequences. A considerable time investment resulted by the unexpected sensor replacement, thus forcing an experimental timeline compression. Therefore, it has led to limitations in experimental scope and depth as the initial comprehensive testing plan was necessarily condensed. To illustrate that, the expected comprehensive evaluation of the robot's capabilities was reduced to a core functionality validation and demonstration. Nonetheless, a critical point of reflection is gained through this limitation, where the importance of reliability of sensor as well as the potential risks associated with the technological pivots during the process of research and development are highlighted.

Despite these challenges, the whisker sensor-based system is proved to be more reliable in the obstacle avoidance scenario. This is due to its direct, mechanical nature provided a more robust alternative to the previously problematic infrared approach.

5.2 Experimental Limitations

As mentioned in section **5.1 Challenges and Problem-Solving Approaches**, due to the time constraints, the final experimental design adopted have restricted the comprehensive performance assessments of the robotic system. For elaboration, the critical limitations

included the narrow range of testing scenarios, the absence of extensive performance metrics such as the confusion matrix, as well as the inability for complex navigation scenarios demonstration, such as the advanced corner escape mechanisms. Hence, future work should prioritise on a more comprehensive testing protocols development that address the current project's methodological limitations. For illustration, it is suggested to expand the test scenarios to involve navigational challenges that are more challenged, along with the implementation of rigorous quantitative performance tracking mechanisms to evaluate the robotics system. In essence, advanced scenarios such as multiple obstacle navigation, corner escape capabilities, and performance evaluation under varied environmental conditions should be explored by the future experiments.

5.3 Potential Technological Enhancements

Building upon the initial prototype, several promising technological improvements emerge as potential research trajectories. For example, sensor technology represents a critical area for potential advancement. To elaborate that, it possesses the opportunities for implementing advanced distance sensors. For example, Light Detection and Ranging (LiDAR) that capable of mapping the surroundings environment of the robot and detecting obstacles through creating precise 3D spatial data using laser pulses emission (Shashank Pasupuleti, 2024). Moreover, the development of multi-directional detection capabilities is also a potential enhancement. To illustrate that, instead of just having front detection capabilities, it can add in multi-view for detecting object at left, right, and back. including In addition, another potential enhancement included the exploration of adaptive sensing technologies. For elaboration, as discussed in the study by Schwager et al., (2017), which they proposed the algorithm that could provide optimal sensor coverage of an environment through enabling the robots adjust their positions dynamically. Simultaneously, algorithmic improvements could focus on advanced navigation algorithms such as A-star (A*) algorithm for efficient path planning to avoid obstacles, enhanced signal processing methodologies like sensor fusion which combine data from multiple sensors, and machine learning-based adaptive navigation techniques such as enhancing the perception and decision-making using neural network (Bharatbhai Pravinbhai Navadiya, 2024; Huang et al., 2022).

6.0 Conclusion

In conclusion, using the Cyberbot platform, the project successfully developed a metal-detecting robot with capabilities of obstacle avoiding and providing feedback. To elaborate that, as discussed in the section 4.2 Experimental Observations, the robot effectively navigates the environment using whisker sensors to detect obstacle and a phototransistor to detect metal, and a piezospeaker and LED to provide auditory and visual feedback when metal, which is coin in this project is detected. Therefore, with the robot avoiding obstacles and accurately detecting coin in the experiments, it is believed that good reliability and effectiveness is demonstrated by the implemented algorithms. While there are limitations to the current experiment, particularly in terms of preliminary demonstration of robotic navigation and metal detection capabilities without quantitative metrics for performance measure, the project still provides a solid foundation for further developments. In corresponding to that, future work could focus on conducting experiment methodology that are more comprehensive, enhancing the sensing capabilities, refining the navigation algorithms, and expanding the metal detection features. With these improvements, similar robots could find applications in various fields, from security screening to archaeological research.

Overall, the potential of combining autonomous navigation with specialised sensing capabilities to create robots that are able to perform useful tasks in real-world environments is demonstrated through this project.

7.0 Reference List

- Ali, N. S., & Hakim Adil Kadhim. (2018). Management and Achieving System for Metal Detection Robot Using Wireless-Based Technology and Online... *International Journal of Power Electronics and Drive Systems (IJPEDS)*, 10(1), 2088–8694. <https://doi.org/10.11591/ijpeds.v10n1.pp219-229>
- Bharatbhai Pravinbhai Navadiya. (2024, December 21). *MACHINE LEARNING-BASED CONTROL SYSTEMS FOR AUTONOMOUS ROBOTICS IN DYNAMIC ENVIRONMENT*. ResearchGate. https://www.researchgate.net/publication/388195906_MACHINE_LEARNING-BASED_CONTROL_SYSTEMS_FOR_AUTONOMOUS_ROBOTICS_IN_DYNAMIC_ENVIRONMENT
- Caleb, A. (2024). *Advancements in Robotics for Hazardous Environment Inspection and Maintenance*. ResearchGate. https://www.researchgate.net/publication/385885310_Advancements_in_Robotics_for_Hazardous_Environment_Inspection_and_Maintenance
- Huang, Y., Huang, S., Wang, H., & Meng, R. (2022). 3D Path Planning and Obstacle Avoidance Algorithms for Obstacle-Overcoming Robots. *2nd IEEE International Conference on Electronic Communications, Internet of Things and Big Data Conference 2022 (IEEE ICEIB 2022)*. <https://doi.org/10.48550/arxiv.2209.00871>
- Lindqvist, B. (2023). *Reactive Navigation Methods for Autonomous Robots: Safety, Coordination, and Field Deployment*. Luleå University of Technology. <https://www.diva-portal.org/smash/get/diva2:1798904/FULLTEXT01.pdf>
- Ogunsina, M., Efunniyi, C. P., Osundare, O. S., Olaoluwa, S., & Anthony, L. (2024). Robust multimodal perception in autonomous systems: A comprehensive review and enhancement strategies. *Engineering Science & Technology Journal*, 5(9), 2694–2708. <https://doi.org/10.51594/estj.v5i9.1545>
- Parallax Inc. (2024, January 6). *Build your cyber:bot (Rev C Board)*. Parallax Learn. <https://learn.parallax.com/courses/build-your-cyberbot-rev-c-board/lessons/meet-the-cyberbot-board-2/?>
- Parallax Inc. (2024, November 19). *Visible Light Navigation for the cyber:bot*. Parallax Learn. <https://learn.parallax.com/courses/visible-light-navigation-for-the-cyberbot/lessons/build-the-light-sensor-eyes/>
- Schwager, M., Vitus, M. P., Powers, S., Rus, D., & Tomlin, C. J. (2017). Robust Adaptive Coverage Control for Robotic Sensor Networks. *IEEE Transactions on Control of Network Systems*, 4(3), 462–476. <https://doi.org/10.1109/tcns.2015.2512326>
- Shashank Pasupuleti. (2024). Advanced Sensor Technologies in Autonomous Robots: Improving Real-time Decision Making and Environmental... *ResearchGate, Volume 10*(Issue 6), 1–10. <https://doi.org/10.5281/zenodo.14759551>

8.0 Appendices

8.1 Appendix 1 – Copy of Source Code (For Verification)

```
1 # Import the cyberbot library for controlling the robot
2 from cyberbot import *
3
4 # For coin detection
5 coin_found = False # Flag to indicate if a coin has been found
6
7 # For avoiding the robot stuck in corner
8 counter = 0 # Counter to track if the robot is stuck in a corner
9 w_l = 0 # State of the left whisker (0 = pressed, 1 = not pressed)
10 w_r = 0 # State of the right whisker
11 o_w_l = 0 # Previous state of the left whisker
12 o_w_r = 1 # Previous state of the right whisker
13 """o_w_r is initialised to 1 to ensure the corner detection logic works correctly on the first iteration."""
14
15 # Function for basic robot movement
16 # Move the robot forward
17 def forward():
18     bot(18).servo_speed(-25) # Set left servo speed (adjust for direction)
19     bot(19).servo_speed(25) # Set right servo speed (adjust for direction)
20
21 # Move the robot backward
22 def backwards():
23     bot(18).servo_speed(25) # Set left servo speed for backward movement
24     bot(19).servo_speed(-25) # Set right servo speed for backward movement
25     sleep(750) # Move backward for 750 milliseconds
26
27 # Turn the robot right
28 def right():
29     bot(18).servo_speed(25) # Set left servo speed for right turn
30     bot(19).servo_speed(25) # Set right servo speed for right turn
31     sleep(500) # Turn right for 500 milliseconds
32
33 # Turn the robot left
34 def left():
35     bot(18).servo_speed(-25) # Set left servo speed for left turn
36     bot(19).servo_speed(-25) # Set right servo speed for left turn
37     sleep(500) # Turn left for 500 milliseconds
38
39 # Stop the robot
40 def stop():
41     bot(18).servo_speed(None) # Stop left servo
42     bot(19).servo_speed(None) # Stop right servo
43
44 # Light source for the phototransistor
45 bot(3).write_digital(1) # Turn on the LED (P3)
46
47 while True:
48     # Phototransistor setup
49     bot(0).write_digital(1) # Turn on the phototransistor (P0)
50     qt = bot(0).rc_time(1) # Read the phototransistor value
51
52     if qt < 200: # Check if the phototransistor detects a coin (Low reading indicates a shiny surface which means coin is detected)
53         coin_found = True # Set the coin detection flag to true
54
55     # If the robot has detected the coin
56     if coin_found:
57         stop() # Immediately stop
58         for _ in range(5): # Blink LED and buzzer 5 times
59             bot(22).tone(5000, 500) # Buzz as coin found (5000 Hz for 500 ms) (P22)
60             # Turn on and off LED for blinking to indicate coin found
61             bot(20).write_digital(1) # Turn on the LED (P20)
62             sleep(100) # LED on for 100 milliseconds
63             bot(20).write_digital(0) # Turn off the LED (P20)
64             sleep(500) # Pause for 500 milliseconds between alert cycles
65
66     # Read whisker switches for robot navigation
67     w_l = bot(7).read_digital() # Left whisker (P7)
68     w_r = bot(9).read_digital() # Right whisker (P9)
69
70     # Detect if stuck in a corner
71     # If one whisker is pressed and the other is not
72     if w_l != w_r:
73         # Check if the whisker states have changed
74         if o_w_l != w_l and o_w_r != w_r:
75             counter += 1 # Increment counter if the robot is likely in a corner
76             # Update previous whisker states
77             o_w_l = w_l # (Left whisker)
78             o_w_r = w_r # (Right whisker)
79
80             # If stuck in corner too long (counter exceeds 4)
81             if counter > 4:
82                 counter = 1 # Reset counter
83                 # Force whisker states to avoid immediate re-triggering
84                 w_l = 1 # (Left whisker)
85                 w_r = 1 # (Right whisker)
86                 backwards() # Back out of the corner
87                 left() # Turn away from the corner
88                 left() # Extra turn for better escape
89             else:
90                 counter = 1 # Reset counter if not stuck (whisker states haven't changed)
91
92     # Whisker-based obstacle avoidance
93     # Both whiskers hit
94     if w_l == 0 and w_r == 0:
95         backwards() # Move the robot backwards using the defined function
96         right() # Make the robot turns right using the defined function
97     # Right whisker hit
98     elif w_r == 0:
99         backwards() # Move the robot backwards using the defined function
100         left() # Make the robot turns left using the defined function
101     # Left whisker hit
102     elif w_l == 0:
103         backwards() # Move the robot backwards using the defined function
104         right() # Make the robot turns right using the defined function
105     # No obstacle
106     else:
107         forward() # Move the robot forward
```

8.2 Appendix 2 - Image Evidence



8.3 Appendix 3 - Video Evidence

Test Setup 1:

https://uowmalaysia-my.sharepoint.com/:v:/g/personal/0207368_student_uow_edu_my/ET-W5XaeUr1IpON7_X34KVIBXhavWrC7BjzFJR5Ob7kdRQ

Test Setup 2:

https://uowmalaysia-my.sharepoint.com/:v:/g/personal/0207368_student_uow_edu_my/EbX2sAL8EURCuJUHaz1QjfwBQz2AICbPj1zQ5DbUiSMw3A