



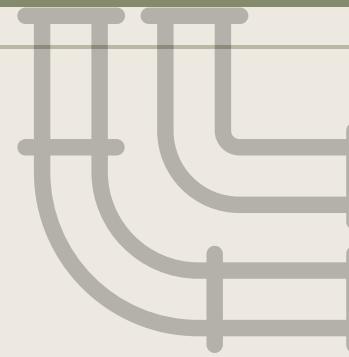
In-Pipe Inspection Robot (IPIR)

Ghadi Alshehri 2112956 Rawnaa Aljohani 2110160

Raghad Alghamdi 2111376 Shahad Alharbi 2111904

Raheed Karah 2110296

Supervised by : Dr.Sabra Elferchichi



Poster



Problem Statement
Companies like SGS in Saudi Arabia offer smart PIGs for pipeline [1]. Smart PIGs have upgraded features compared to regular PIGs, such as electronic sensors and ultrasonic sensors to Detect problems down the pipeline such as loss of metal, welding, cracking bending, and temperature anomaly [2]. Smart PIGs main issue is that it can only be performed by trained and qualified operators; it requires steps that carry safety and integrity risks for pigging, which is the operational procedure to launch a PIG [1]. Another disadvantage is that smart PIGs are very costly and difficult to afford, with no use of AI tools or cameras to capture defects in pipes. The project's goal is to address the previously discussed limitations associated with PIG.

Aims
This project aims to introduce an advanced, cost-effective in-pipe inspection robot that detects and localizes pipeline defects. The robot leverages AI, IOT, and real-time communication and notification to enable smart navigation and autonomous control, reducing manual intervention. This supports the transition to smart cities and helps prevent environmental damage.

Objectives

- Use computer vision to classify defects.
- Semi Autonomous motion control of the robot.
- Real-time localization of defects in the pipeline.
- Collect images of identified defects and store them in a database.

Abstract
This project presents the development of an Autonomous In-Pipe Inspection Robot (IPIR) designed to address the challenges of current pipeline inspection practices, which often rely on smart Pipeline Inspection Gauges (PIGs) offered by companies like SGS in Saudi Arabia. While these advanced technologies utilize electronic and ultrasonic sensors to detect issues such as metal loss and welding defects, their deployment presents significant challenges, including high costs, operational complexity, and inherent safety risks during the pigging process. Additionally, the lack of AI tools and cameras for defect visualization hinders efficient data analysis. The IPIR integrates modern advancements in artificial intelligence and autonomous navigation. It features a Raspberry Pi 4 as the central processing unit running the YOLOv9t-based defect detection model and an OV5647 camera for visual inspections. The robot's movement is controlled by an Arduino microcontroller with an H-bridge motor driver, while a single optical encoder provides accurate navigation feedback. Extensive testing has demonstrated that the IPIR effectively detects and classifies common pipeline defects, including cracks, holes, and obstacles, in real time, achieving a model accuracy of 0.77. A web-based dashboard enhances usability by visualizing defect types, locations, and captured images, allowing the maintenance team to make informed decisions. The robot can navigate effectively through T and L junction turns inside the pipe.

In conclusion, this research provides a scalable and cost-effective solution for pipeline inspections, addressing critical challenges in underground maintenance. Future work will focus on enhancing wireless communication for reliable data transmission, developing advanced control algorithms, and upgrading the robot's processing capabilities. This study contributes to improving pipeline integrity management and offers a safer alternative to traditional PIG technologies.

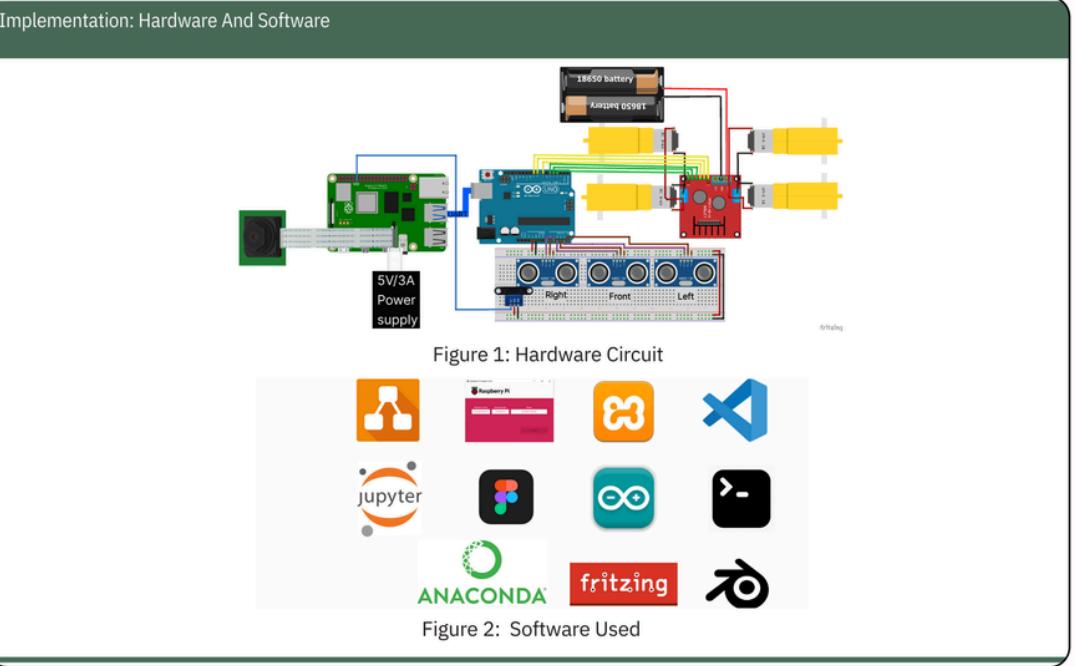
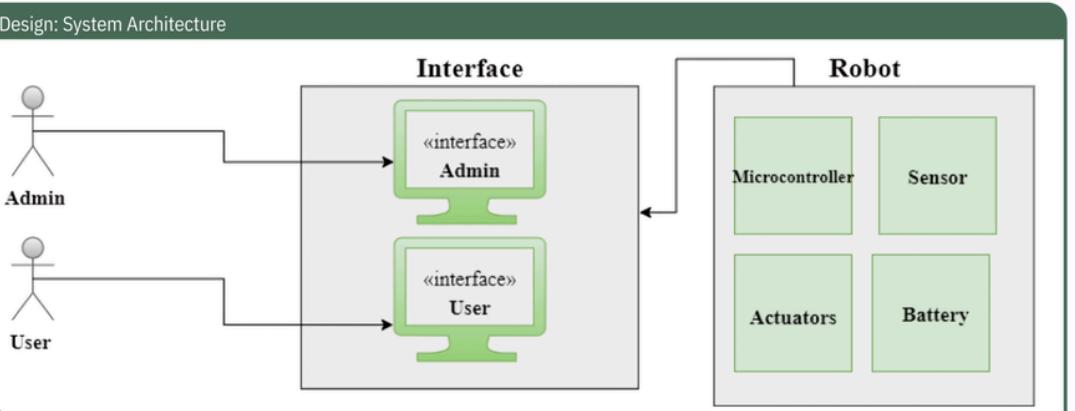
Conclusion
This research demonstrates significant progress in developing an Autonomous In-Pipe Inspection Robot (IPIR). Key achievements include accurate defect classification through computer vision, robust semi-autonomous motion control, and real-time defect localization within the pipeline.

Autonomous Robot for In-Pipe Defect Detection

Shahad Alharbi; Raghad Alghamdi; Raheed Karah; Rawnaa Aljohani; Ghadi Alshehri

Supervised by Dr. Sabra Elferchichi (DECEMBER 2024)

College of Computer Science and Engineering, University of Jeddah
Bachelor of Science in Computer and Network Engineering



FutureWork

- Investigate and implement advanced wireless communication technologies, such as long-range networks for robust and reliable data transmission in challenging pipe environments.
- Develop and implement sophisticated control algorithms, such as advanced PID controllers or model predictive control, to enhance the robot's navigation within complex pipe geometries.
- Development of a robust and standardized system for classifying pipe defects based on their severity. This system will provide a consistent framework for assessing the criticality of defects, enabling more informed decisions regarding maintenance and repair priorities.
- enhance the robot's capabilities by integrating a more powerful processor and a high-capacity power bank. This combination will enable the simultaneous execution of wall-following and defect-detection algorithms.



Figure 3: Robot Active view Figure 4: Robot Dark view

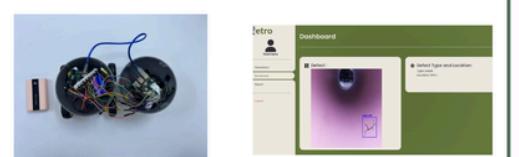


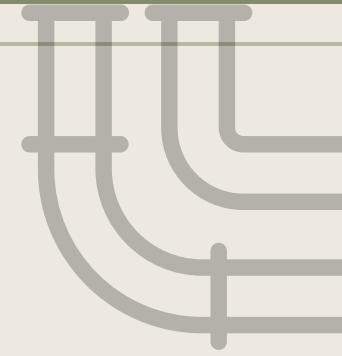
Figure 5: Interior view Figure 6 : Dashboard defect



Figure 6: Fully Robot

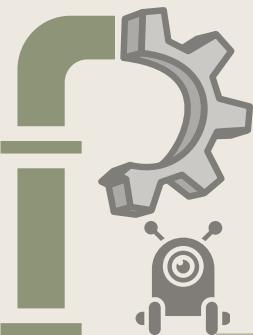
References

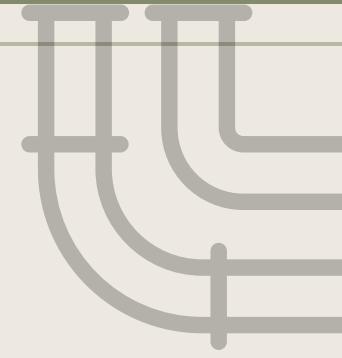
- [1] SmartpiggingSolutions.
- [2] Content Seo Copilot. Intelligent Pigging procedure - How does it work? - Pigtek, 1 2024.
- [3] Shin Takahashi Ayman Atia and Jiro Tanaka. Smart gesture sticker: Smart hand gestures profiles for daily objects interaction. Computer and Information Science, ACIS International Conference on, 0:482–487, 2010.



Proposed Solution

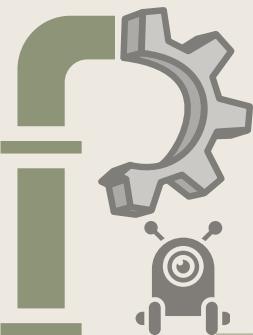
- ❑ Develop an **Semi-autonomous In-Pipe Inspection Robot (IPIR)** designed to inspect the internal surfaces of oil and gas underground pipelines.
- ❑ Key features include:
 - ❑ Deep learning defect detection using a camera as a non-traditional NDT (Non-Destructive Testing) technique for inspection process
 - ❑ Localization capabilities to accurately determine the defect's position within the pipeline.
 - ❑ Real-time data transmission to a dashboard for immediate analysis and action.

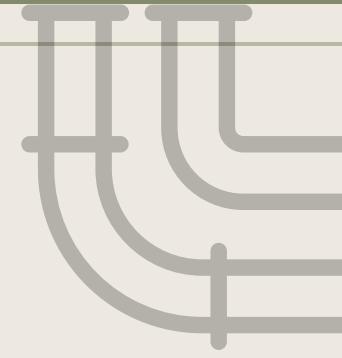




Objectives

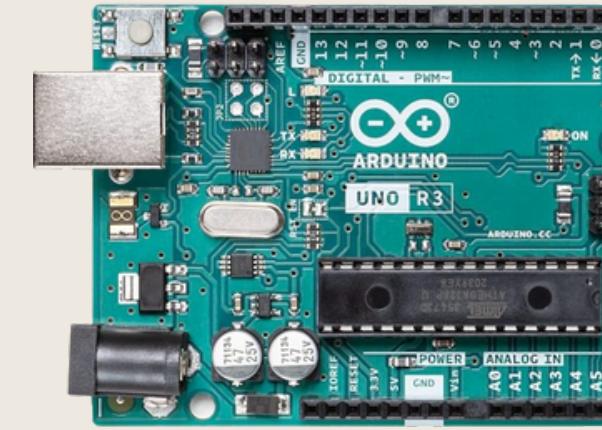
- ❑ Use computer vision to classify defects.
- ❑ Semi Autonomous motion control of the robot.
- ❑ Real-time localization of defects in the pipeline.
- ❑ Collect images of identified defects and store them in a database.



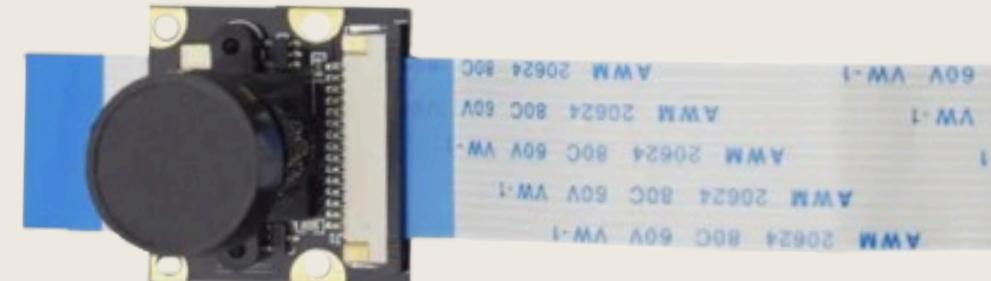


Hardware Used

❖ Processing and Control



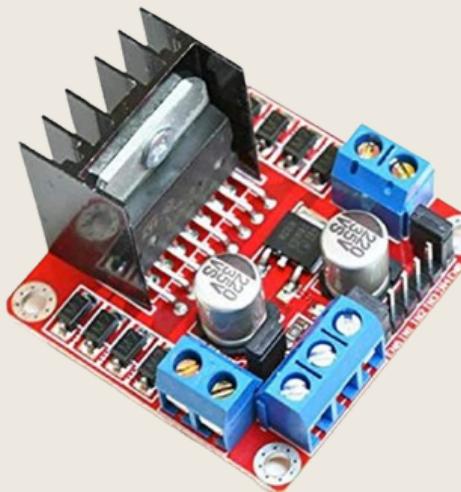
❖ Inspection Camera

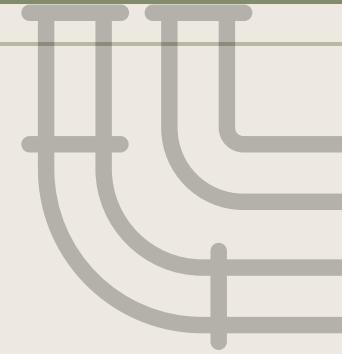


❖ Navigation and Localization



❖ Movement and Actuation





Software Used

Coding



Database & Web Page



AI



3D Modeling



Design Diagram



Hardware & AI



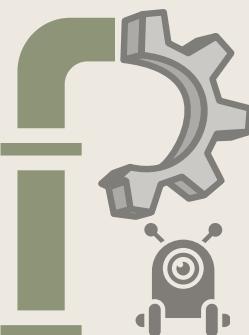
Design Interface

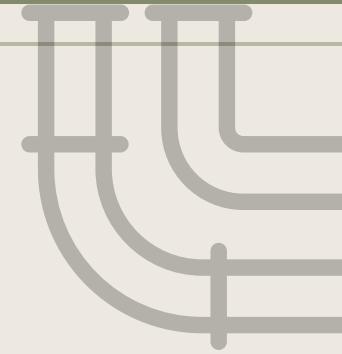


Dataset Augmentation

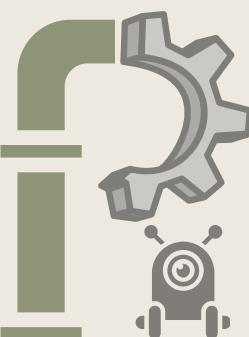
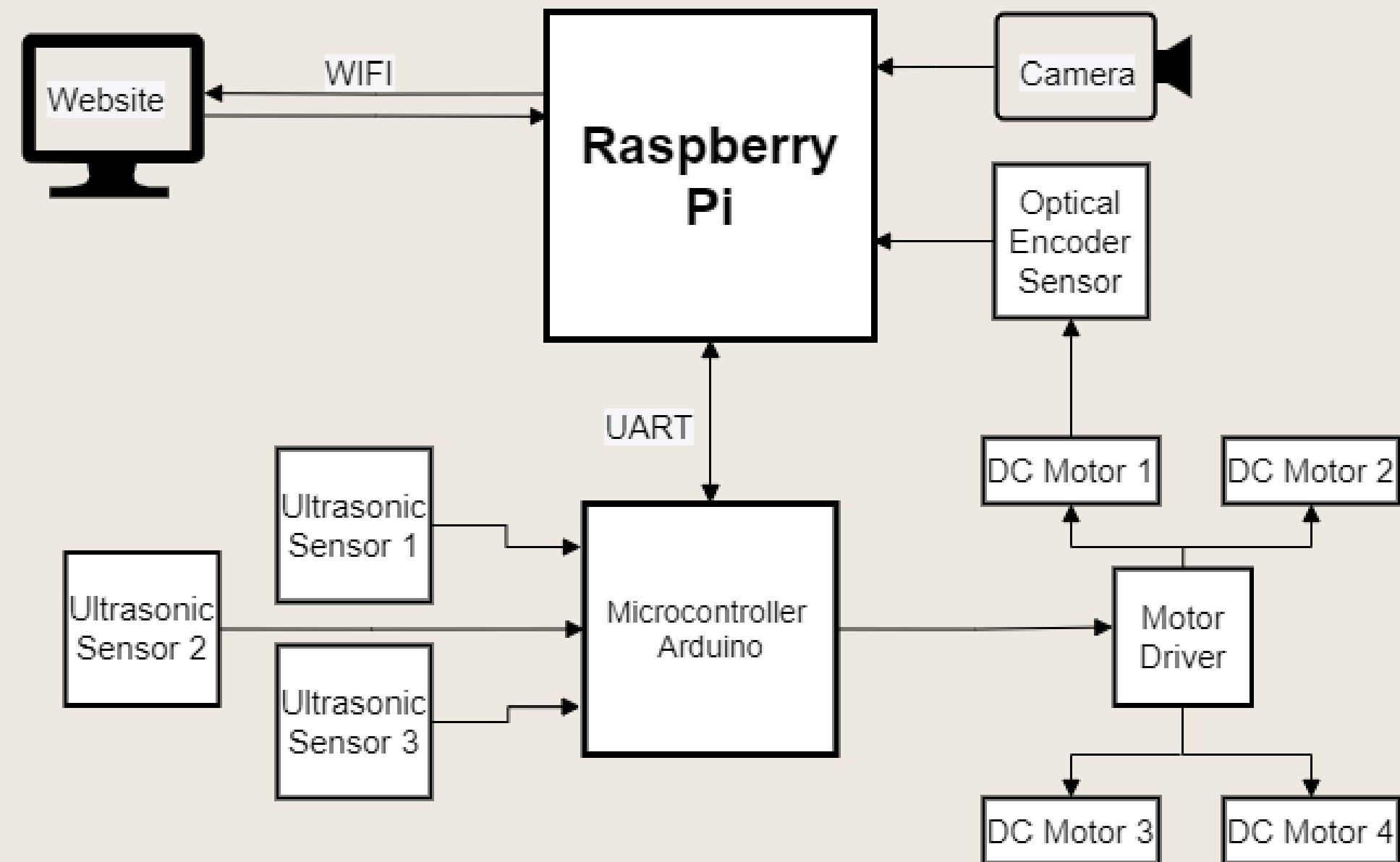


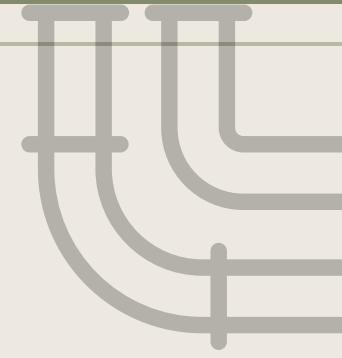
Prototyping



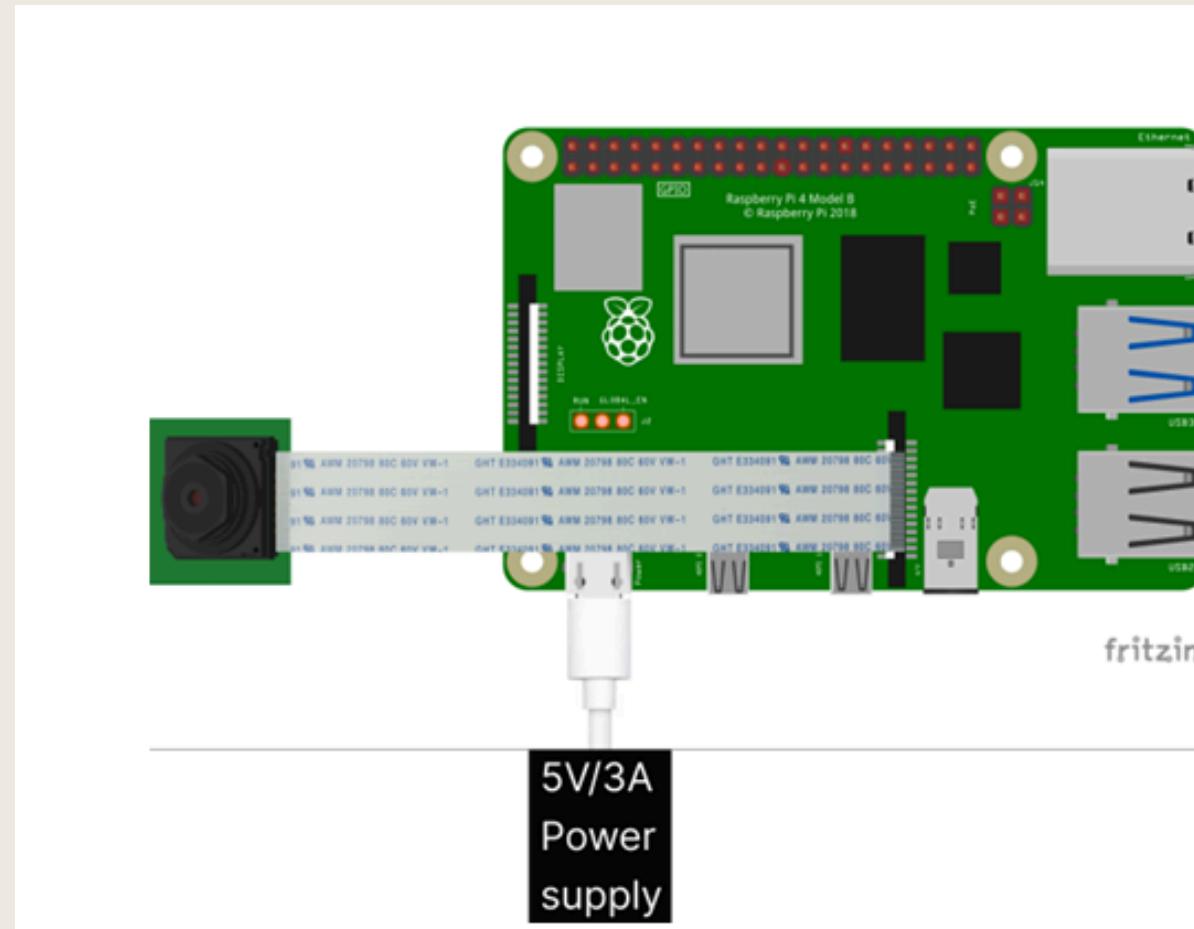


Block Diagram

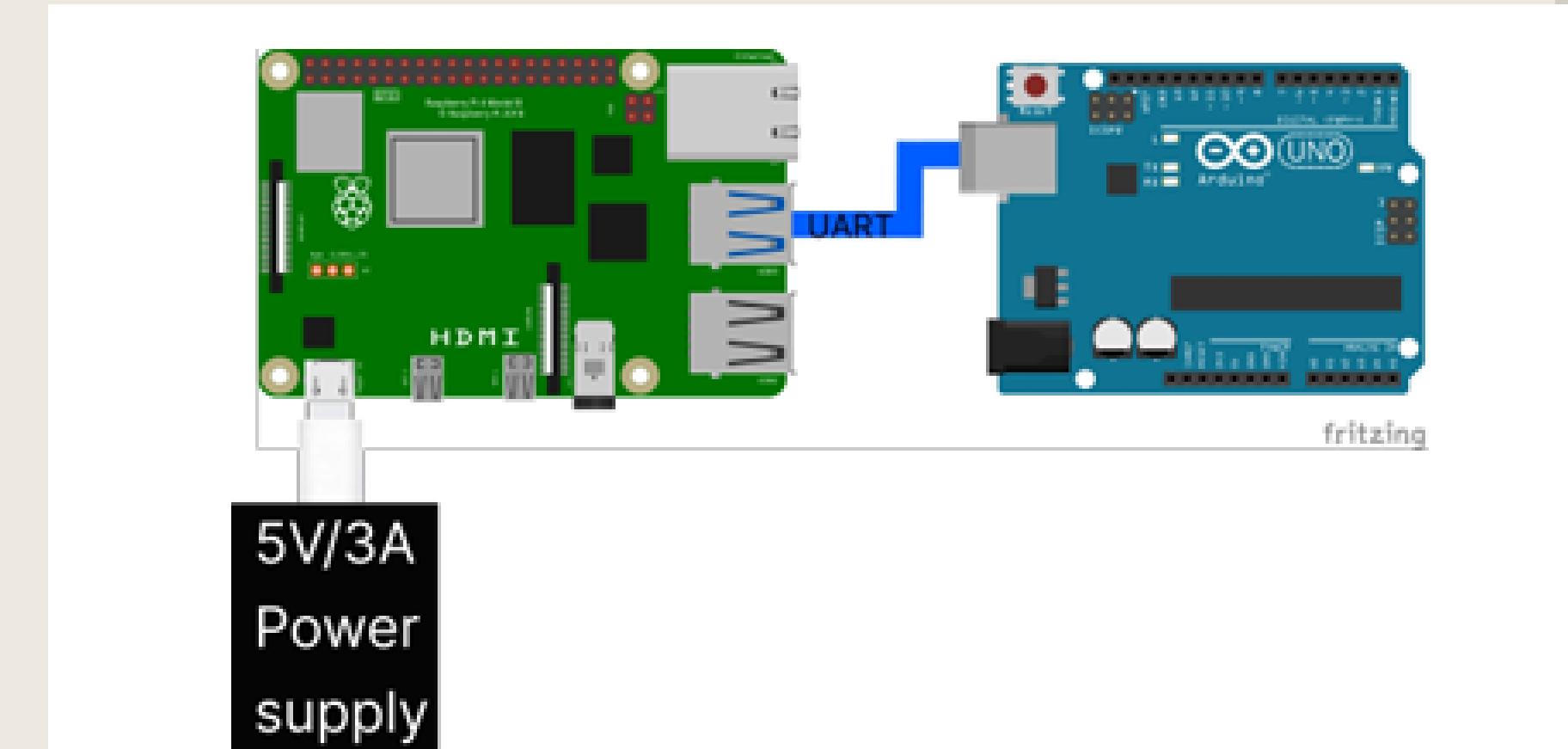




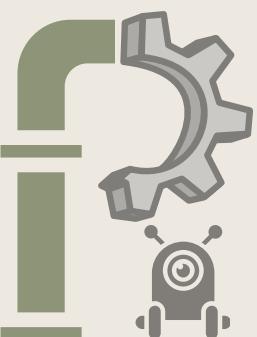
Circuit



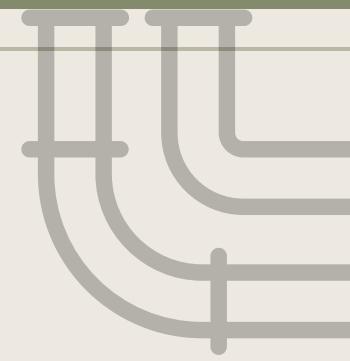
camera inspection system



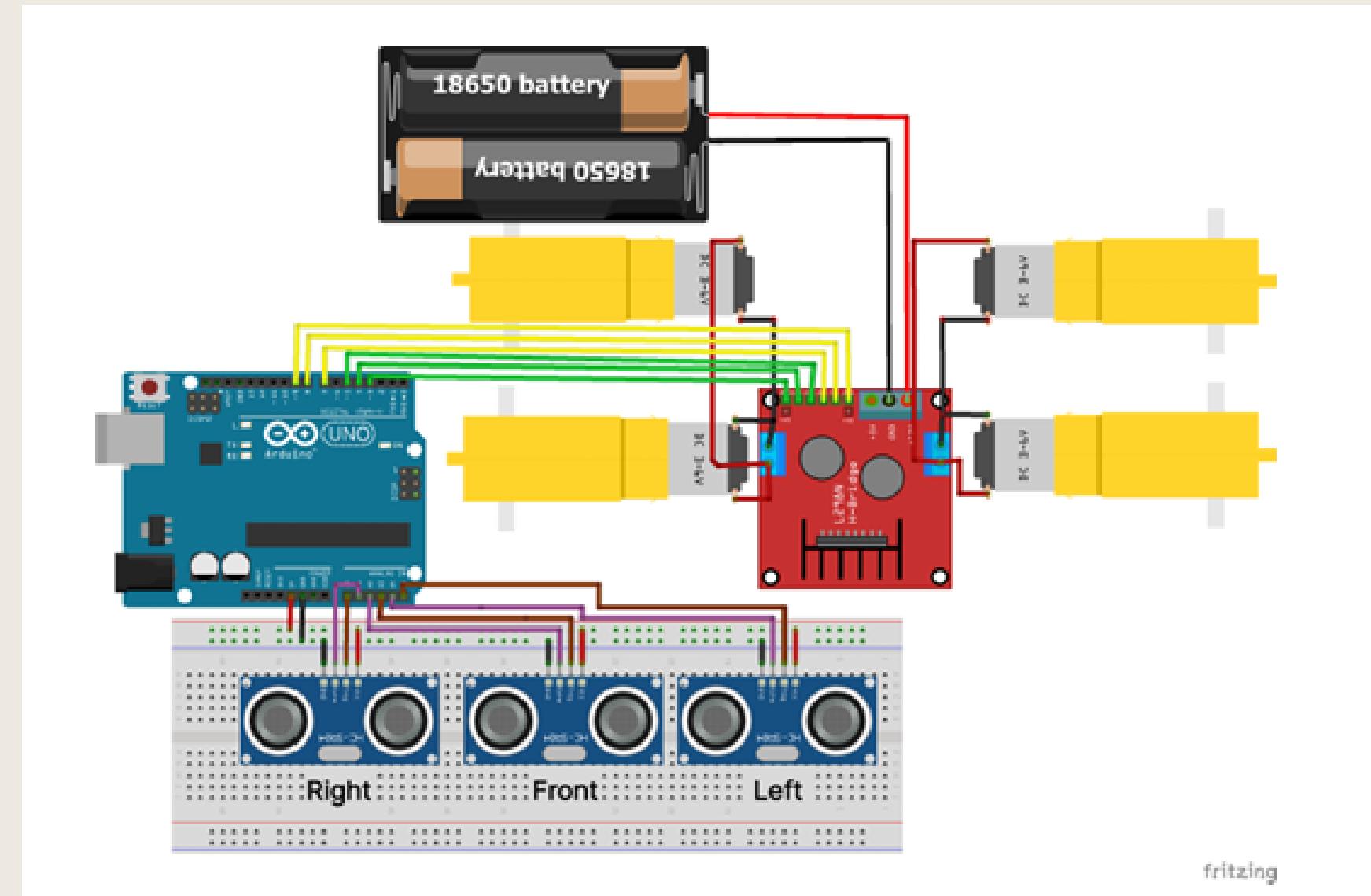
serial communication between the RPI and Arduino



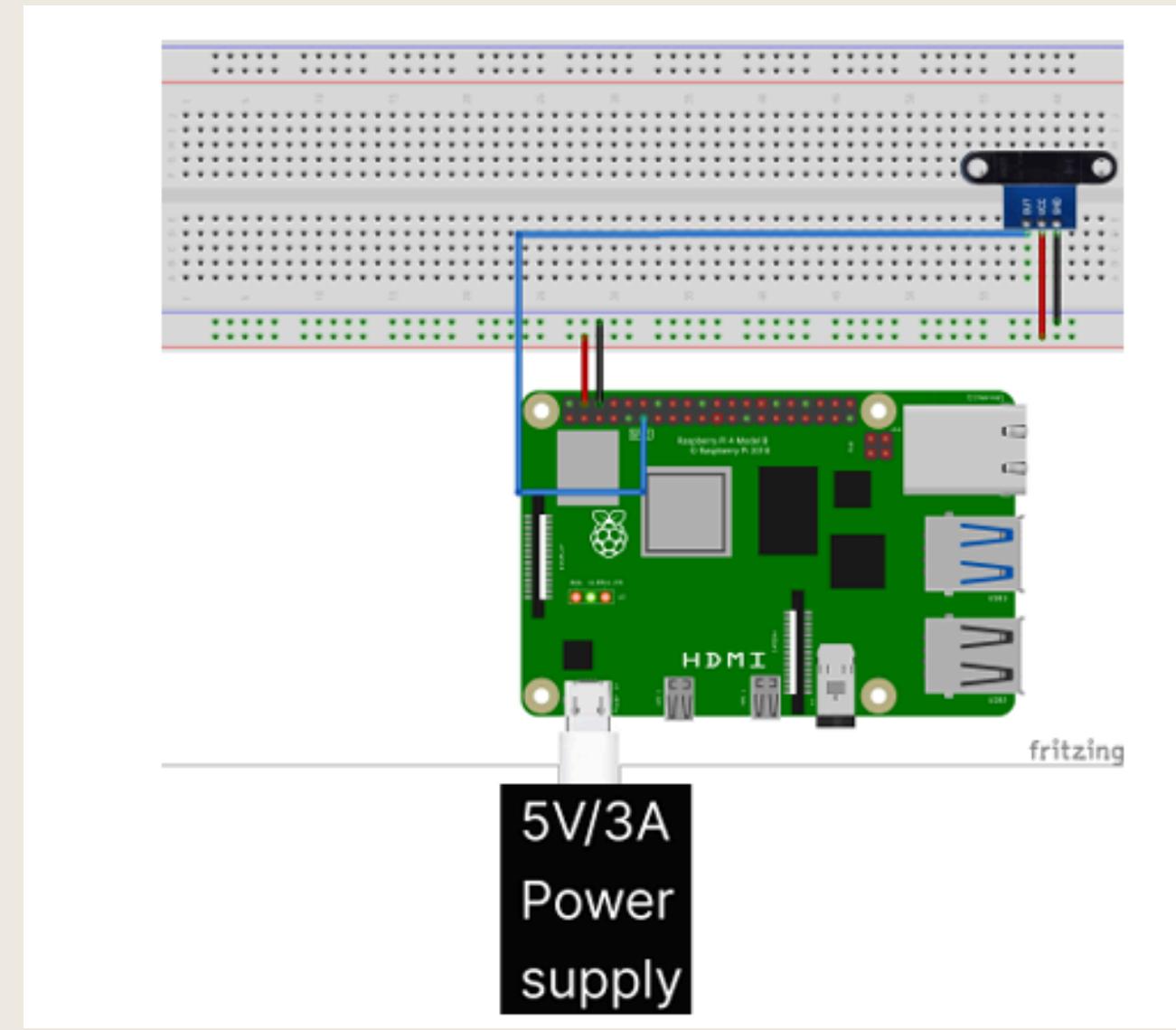
CIRCUIT



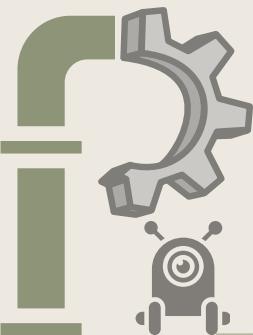
Circuit



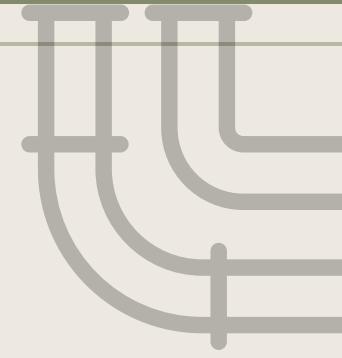
Circuit diagram for robot navigation system



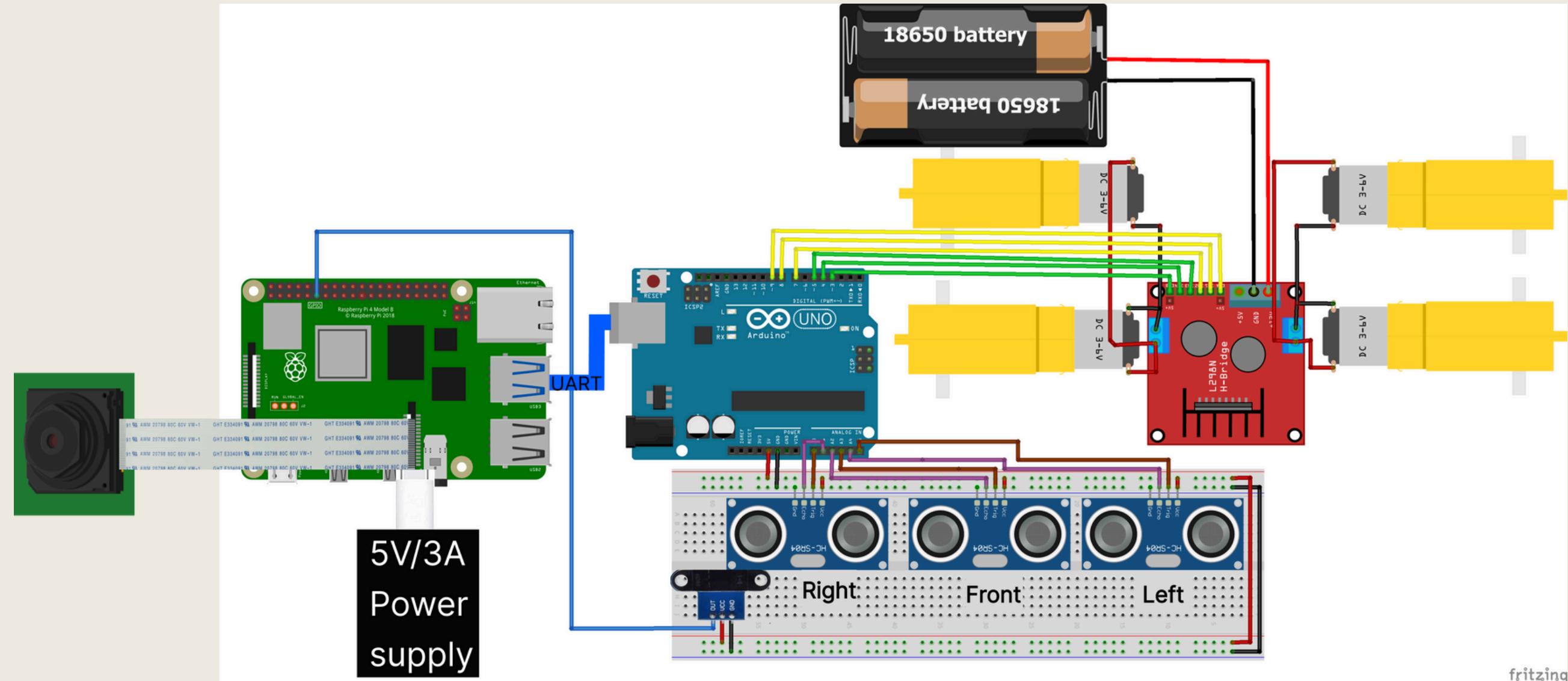
Circuit diagram for robot localization system



CIRCUIT



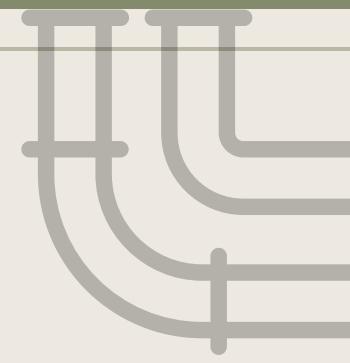
Circuit



Full robot navigation system



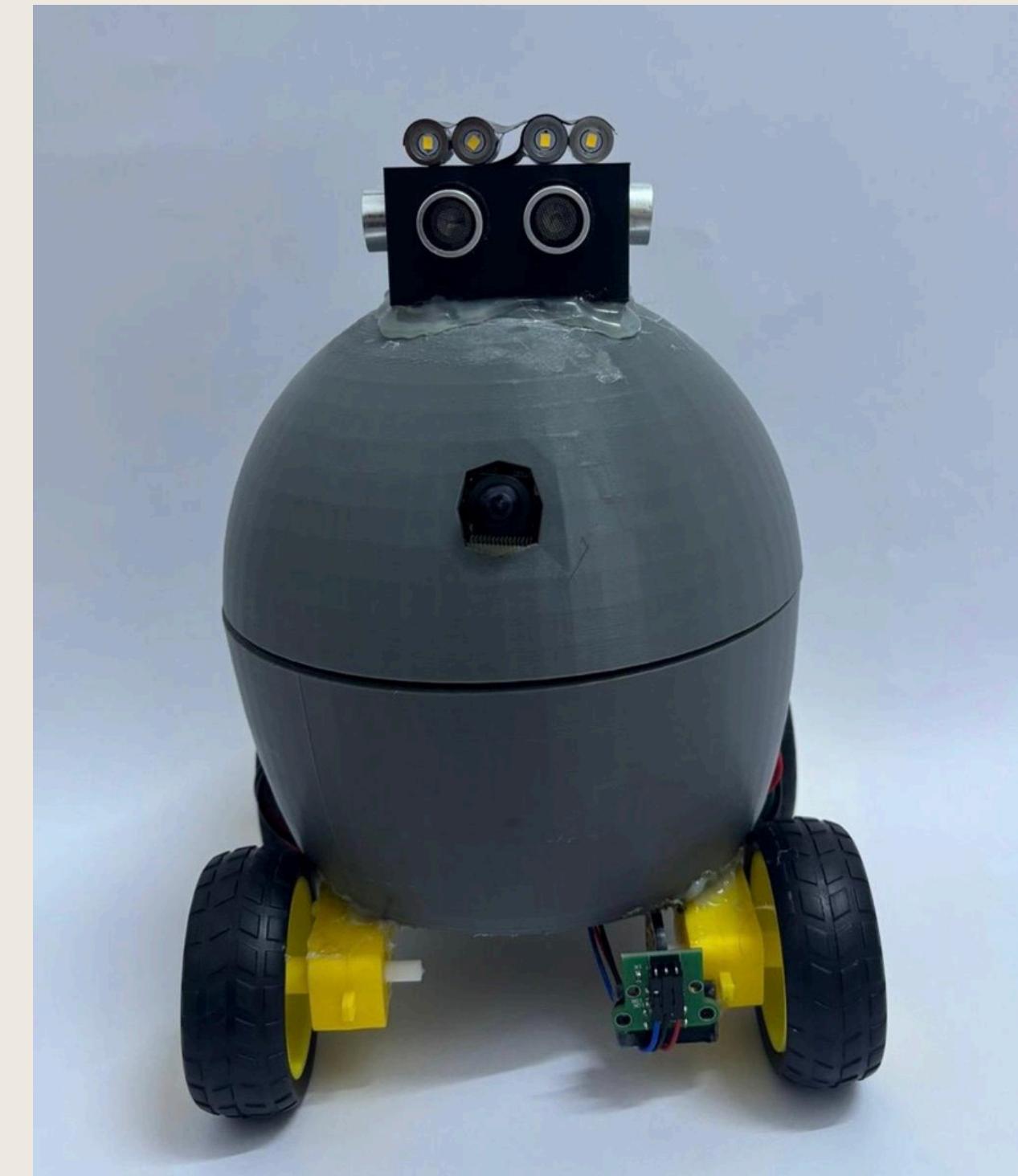
CIRCUIT

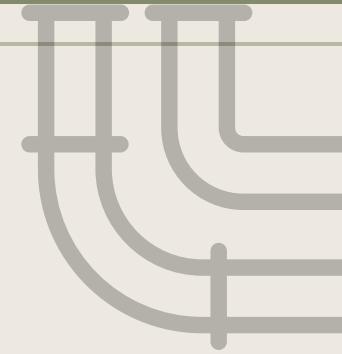


Prototype



PROTOTYPE





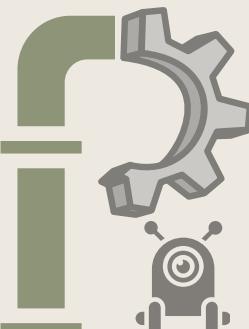
Implementation Details

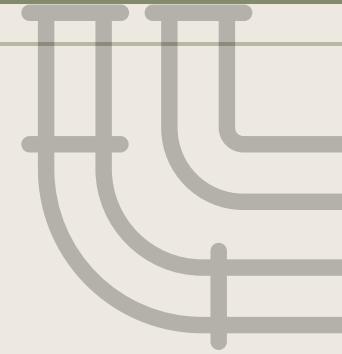
- **Dataset Acquisition**

- We used multiple sources to gather a better defect sample.
- we selected the types of defects commonly found in oil and gas pipelines

NO.	pipeline type	Link	NO. of images in dataset
1	storm drain	com/new-workspace	1k
2	not mention	ik/browse?queryTex	
3	not mention	flow.com/tongji/pip	804
4	not mention	/quickwork-qxpmp/	278
5	not mention	atasets/manaidu/pi	19,908
6	not mention	atasets/simplexitypip	22,120
7	drain pipe	ow.com/drainimage/	338
8	not mention	low.com/project-c2s	2,022
9	sewer	joflow.com/sewage/	2400

Various sources for collecting datasets





Implementation Details

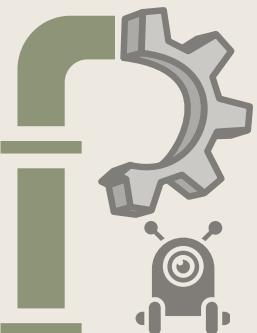
- **Dataset Acquisition**
- To collect more datasets and improve accuracy, we created our own datasets using the pipe.

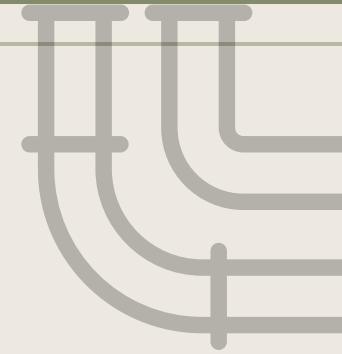
Type of defect	Number of images
crack	1.083
holes	166
obstacle	605
total	1.612

Full dataset number



Create our dataset

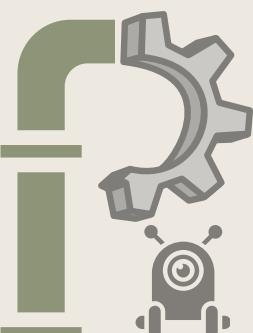


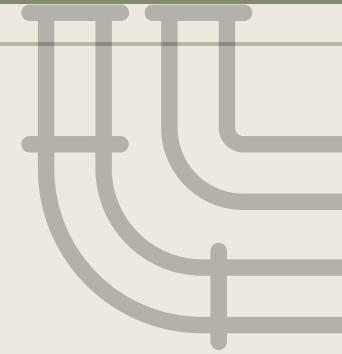


Implementation Details

- **Model Training**
- The dataset was trained using YOLOv9t with the following parameters:

Parameters	Input
epochs	100
Images size	640
freeze	9
Multiscale	True





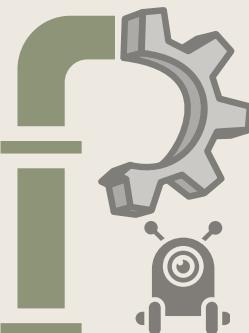
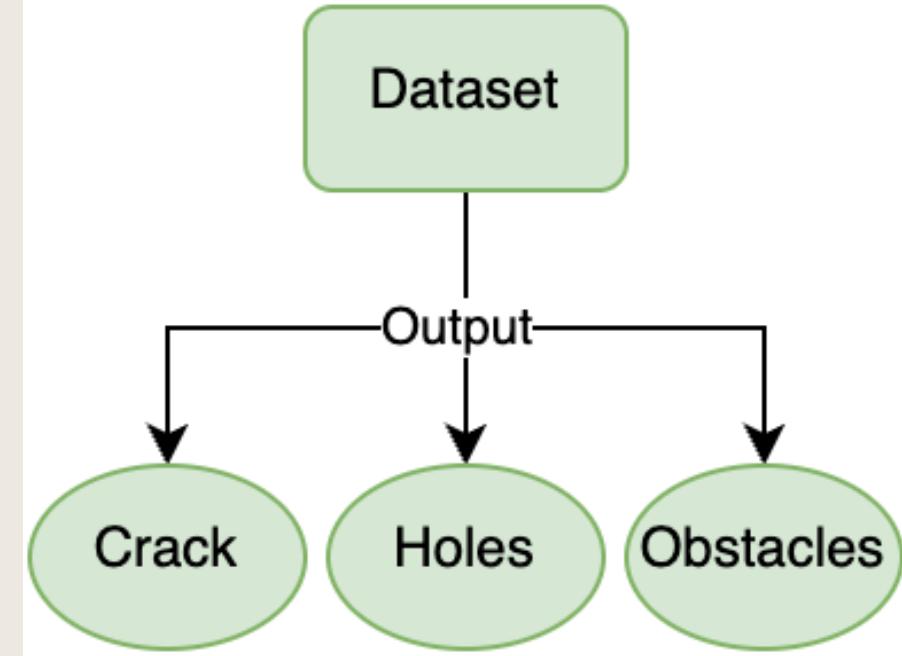
Implementation Details

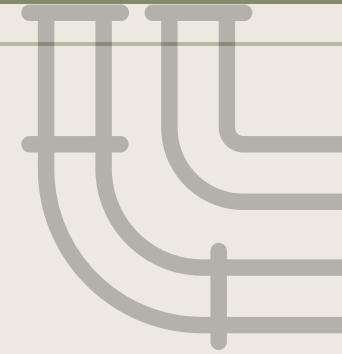
- Classification and detection results
 - The camera immediately detects, captures, and classifies the defect as a hole within 1.7 seconds.

A screenshot of a computer interface showing the implementation details. On the left, a terminal window displays Python code for a script named `geany_run_script_OPRMY2.sh`. The code uses `picamera2`, `pandas`, and `ultralytics YOLO` libraries. The output shows the script running and detecting defects in real-time. On the right, a camera feed window titled "Camera" shows a purple-tinted image of a surface with a red bounding box and the word "hole" indicating a detected defect.

```
from picamera2 import Picamera2
import pandas as pd
from ultralytics import YOLO

# Edit Tabs Help
# speed: 7.1ms preprocess, 1627.9ms inference, 2.3ms postprocess per image at shape (1, 3, 480, 640)
# : 480x640 1 obstacle, 1636.4ms
# speed: 7.1ms preprocess, 1636.4ms inference, 2.3ms postprocess per image at shape (1, 3, 480, 640)
# 0: 480x640 2 cracks, 1628.6ms
# Speed: 6.9ms preprocess, 1628.6ms inference, 3.7ms postprocess per image at shape (1, 3, 480, 640)
# 10: 480x640 1 crack, 1 hole, 1619.8ms
# Speed: 7.0ms preprocess, 1619.8ms inference, 2.5ms postprocess per image at shape (1, 3, 480, 640)
# 20: 480x640 1 hole, 1612.9ms
# Speed: 7.0ms preprocess, 1612.9ms inference, 2.4ms postprocess per image at shape (1, 3, 480, 640)
# 30: 480x640 1 hole, 1622.1ms
# Speed: 10.8ms preprocess, 1622.1ms inference, 2.7ms postprocess per image at shape (1, 3, 480, 640)
# 35: cv2.imshow('Camera',im)
# 36: if cv2.waitKey(1)==ord('q'):
# 37:     break
# 38: cv2.destroyAllWindows()
# 43
```



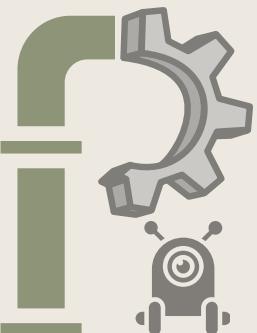


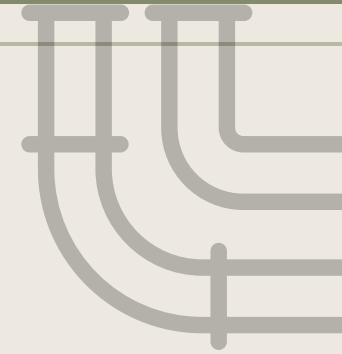
Implementation Details

- Database tables
- Users Table
- This table stores information about the users who interact with the robot system,

The screenshot shows the phpMyAdmin interface for the 'users' table in the 'petro' database. The table has six columns: id, role, username, password, email, and date. The 'date' column uses the current_timestamp() function for its default value.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	varchar(5)	utf8mb4_general_ci		No	None			Change Drop More
2	role	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
3	username	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
4	password	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
5	email	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
6	date	timestamp			No	current_timestamp()	ON UPDATE CURRENT_TIMESTAMP()		Change Drop More





Implementation Details

- Database tables
- Parameters Table
 - The Parameters table is designed to store various operational settings for the robot, which are provided by the admin.

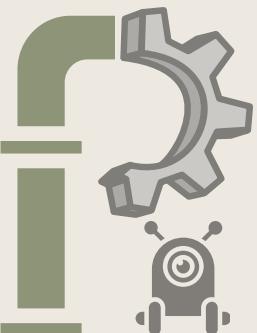
The screenshot shows the phpMyAdmin interface with the following details:

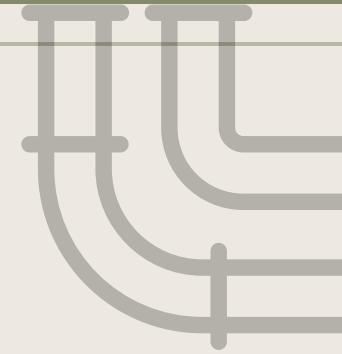
- Server:** 127.0.0.1
- Database:** petro
- Table:** param

The 'param' table structure is displayed in 'Table structure' view:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	missionNO	int(3)	utf8mb4_general_ci		No	None		AUTO_INCREMENT	Change Drop More
2	id	varchar(5)	utf8mb4_general_ci		No	None			Change Drop More
3	speed	varchar(3)	utf8mb4_general_ci		No	None			Change Drop More
4	diameter	varchar(4)	utf8mb4_general_ci		No	None			Change Drop More
5	distance	varchar(4)	utf8mb4_general_ci		No	None			Change Drop More

Below the table structure, there are buttons for 'Check all', 'With selected:', and actions like 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Remove from central columns'.





Implementation Details

- Database tables
- Defects Table
- The Defects Table is designed to record any defects detected by the robot during its operation,

phpMyAdmin

Server: 127.0.0.1 » Database: petro » Table: defect

Browse Structure SQL Search Insert Export Import Privileges Operations

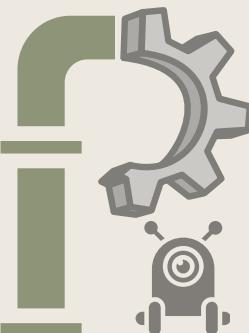
Table structure Relation view

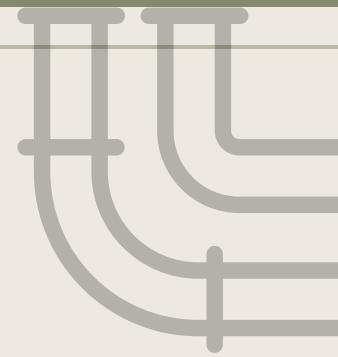
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(3)			No	None		AUTO_INCREMENT	Change Drop More
2	missionNO	int(3)			No	None			Change Drop More
3	defLocation	varchar(15)	utf8mb4_general_ci		No	None			Change Drop More
4	defect	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
5	img	blob			No	None			Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Spatial

Remove from central columns

The screenshot shows the phpMyAdmin interface for the 'defect' table in the 'petro' database. The table has five columns: 'id' (int(3), primary key, auto-increment), 'missionNO' (int(3)), 'defLocation' (varchar(15)), 'defect' (varchar(20)), and 'img' (blob). The 'defLocation' column uses the utf8mb4_general_ci collation.





Implementation Details

Pseudocode for Robot Motion

```
START
LOOP:
    IF SerialInputAvailable THEN
        ReadInputString a

        IF a == "0" THEN
            frontSensor = ReadFrontSensor()

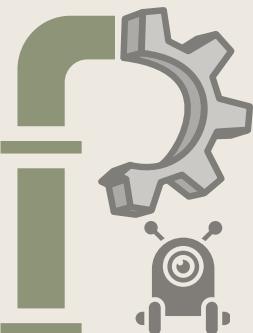
            IF frontSensor <= 15 THEN // Obstacle detected
                leftSensor = ReadLeftSensor()
                rightSensor = ReadRightSensor()
                WAIT 1 second

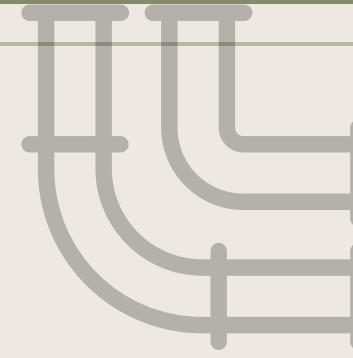
                IF rightSensor > leftSensor THEN
                    SoftRight()
                    WAIT 580 ms
                ELSE IF leftSensor > rightSensor THEN
                    SoftLeft()
                    WAIT 850 ms
                END IF

                ELSE
                    // No obstacle in front
                    Forward()
                    WAIT 300 ms
                    Stop()

                    frontSensor = ReadFrontSensor()
                    IF frontSensor <= 15 THEN
                        RETURN
                    END IF
                END IF

                ELSE
                    Stop()
                    WAIT 2000 ms
                ELSE IF a == "1" THEN
                    Stop()
                    WAIT 5000 ms
                ELSE
                    Stop()
                END IF
            END IF
        END IF
    END IF
```

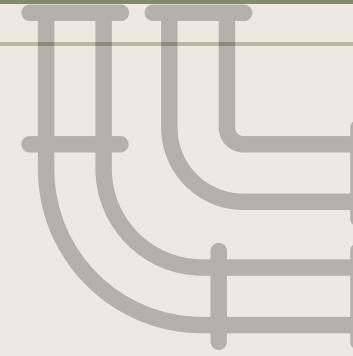




Scenarios Details

- ❑ Unit testing scenarios
- ❑ Integration testing scenarios
- ❑ Validation and system testing scenarios



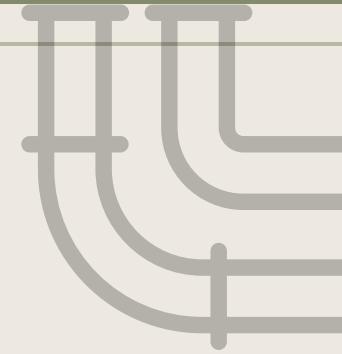


Testing Details

❖ Unit testing details :

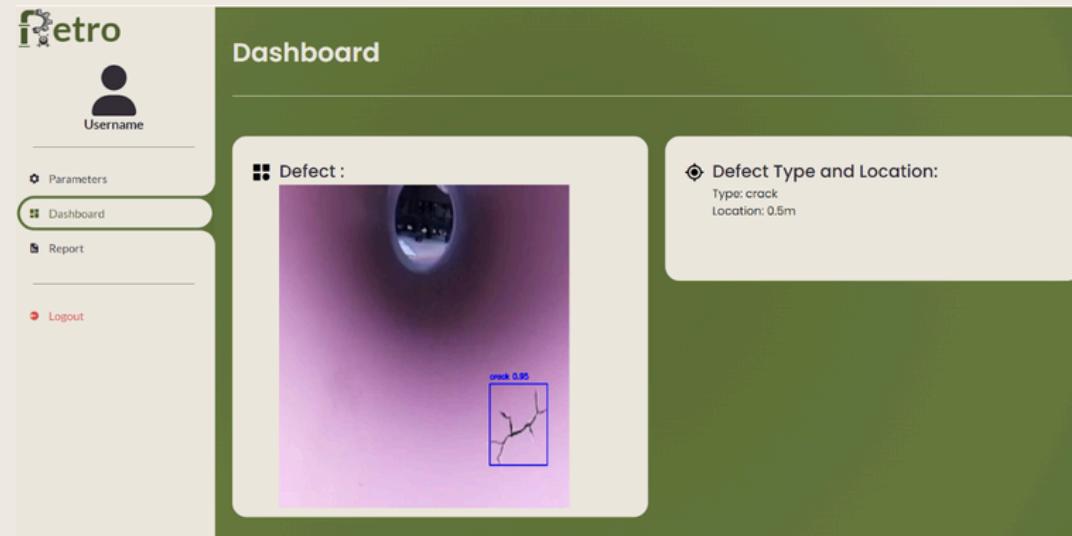
- ❖ Web pages testing
- ❖ Camera testing
- ❖ Ultrasonic testing
- ❖ Encoder testing





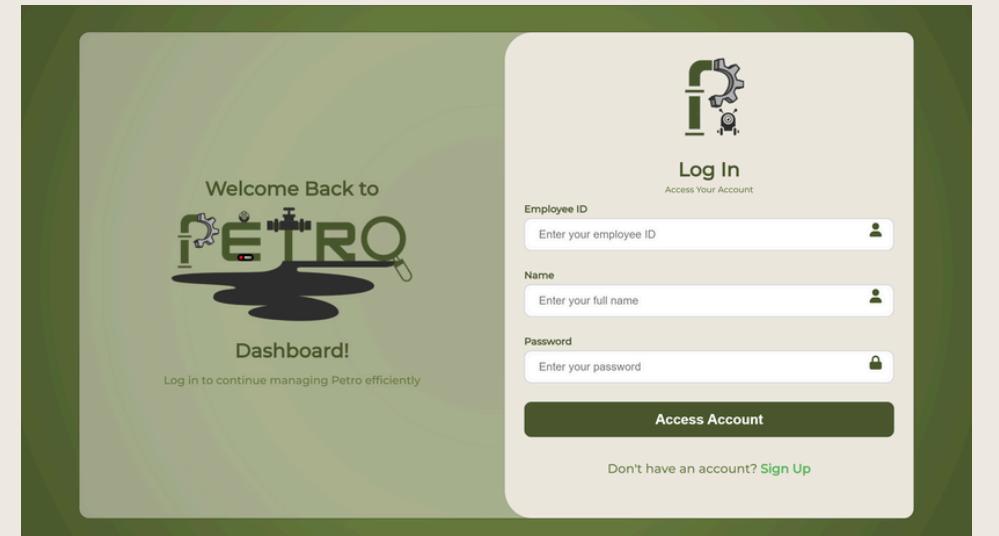
Testing Details

Web page testing:



The screenshot shows the Petro dashboard. On the left, a sidebar menu has 'Dashboard' selected. The main area displays a large image of a pipeline defect with a bounding box and the text 'crack 0.5m'. To its right, a box shows 'Defect Type and Location: Type: crack Location: 0.5m'.

Dashboard Page



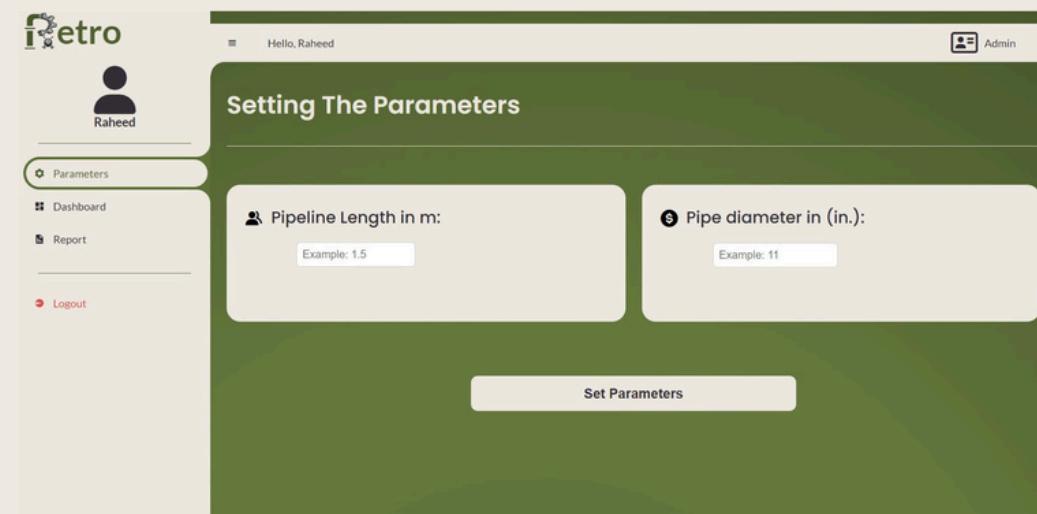
The screenshot shows the Petro login page. It features a logo with the word 'PETRO' and a gear. The text 'Welcome Back to PETRO Dashboard!' is displayed. A form for 'Employee ID', 'Name', and 'Password' is present, along with a 'Log In' button and a 'Sign Up' link.

Login Page



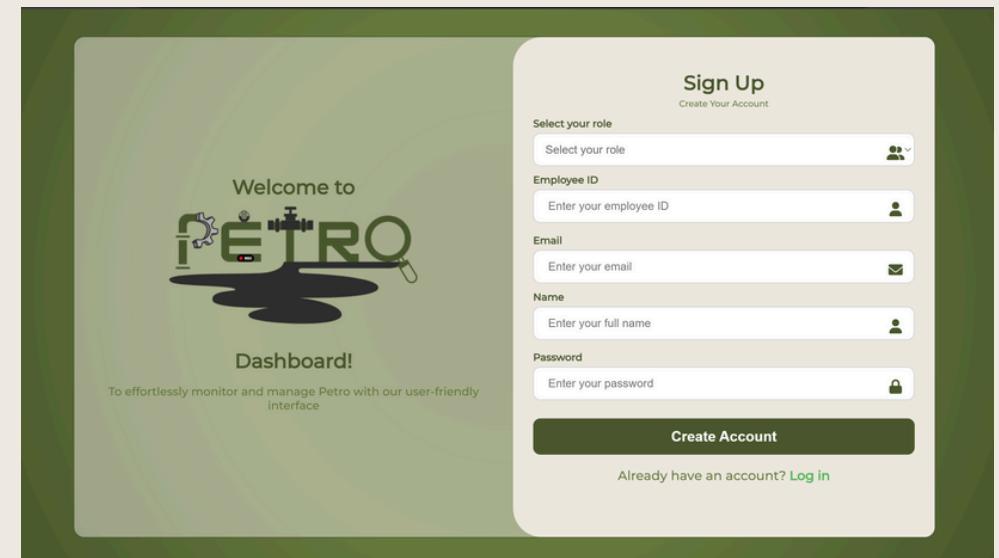
The screenshot shows the Petro report page. It includes a sidebar with 'Report' selected. The main content area shows a button labeled 'All Defects' and a table titled 'Defect Details' with one row: 'Defects for Mission No: 9' with columns 'Image', 'Defect Type', and 'Location' showing a crack at 0.5m.

Report Page



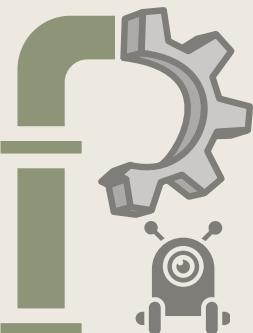
The screenshot shows the 'Setting The Parameters' page. The top bar shows 'Hello, Raheed' and 'Admin'. The main area has two input fields: 'Pipeline Length in m:' with 'Example: 1.5' and 'Pipe diameter in (in.):' with 'Example: 11'. A 'Set Parameters' button is at the bottom.

Parameter Page



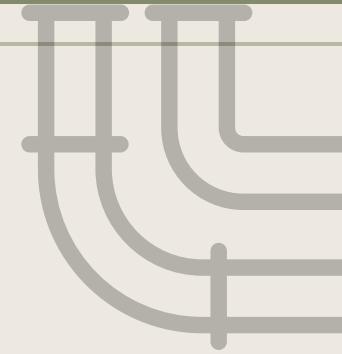
The screenshot shows the Petro sign-up page. It features a logo with 'PETRO' and a gear. The text 'Welcome to PETRO Dashboard!' is displayed. A form for 'Select your role', 'Employee ID', 'Email', 'Name', and 'Password' is present, along with a 'Create Account' button and a 'Log in' link.

Sign up Page



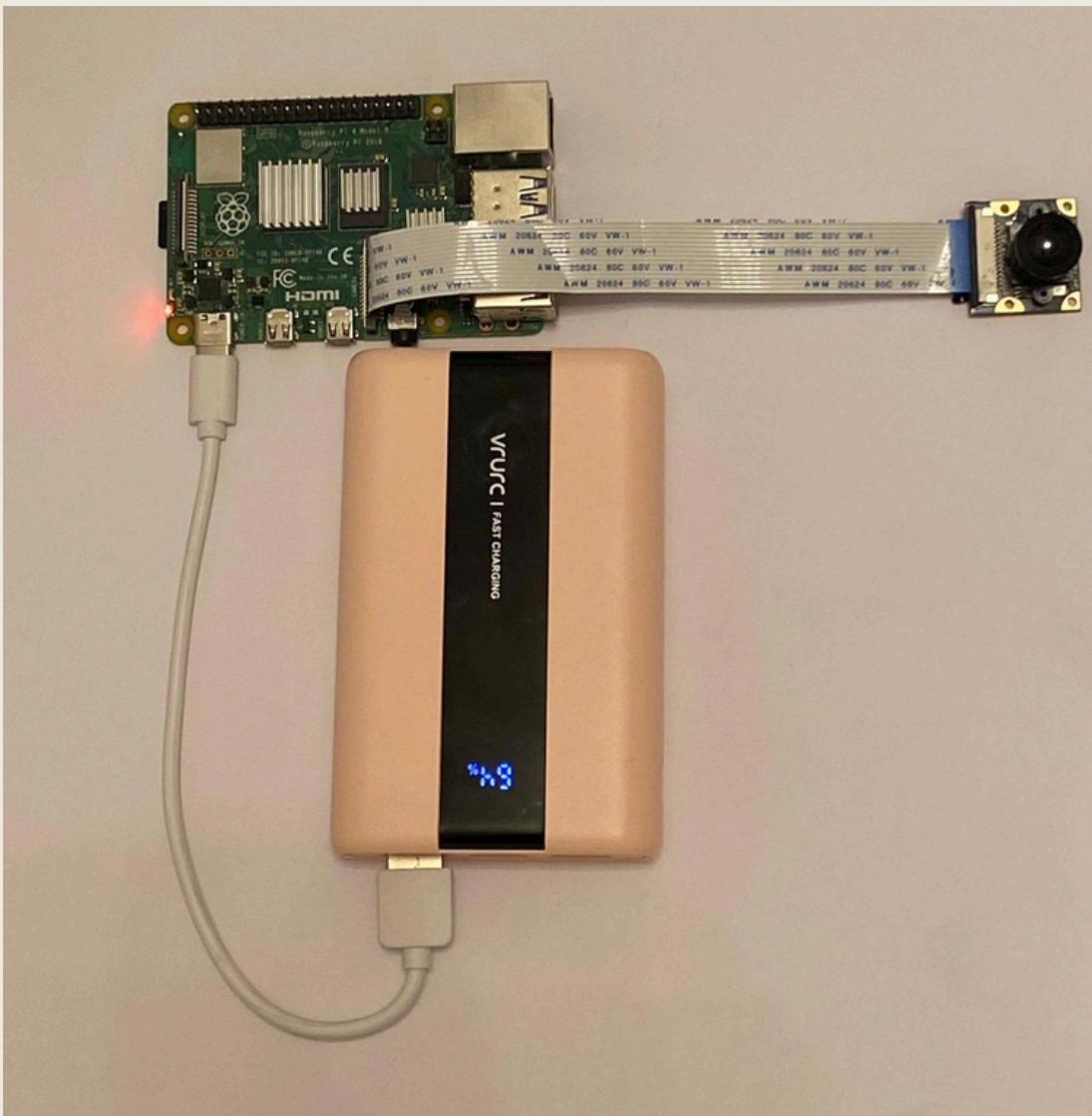
TESTING

PAGE 22

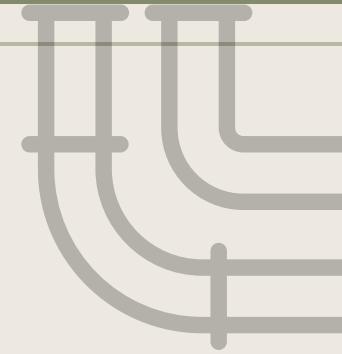


Testing Details

Camera testing details:

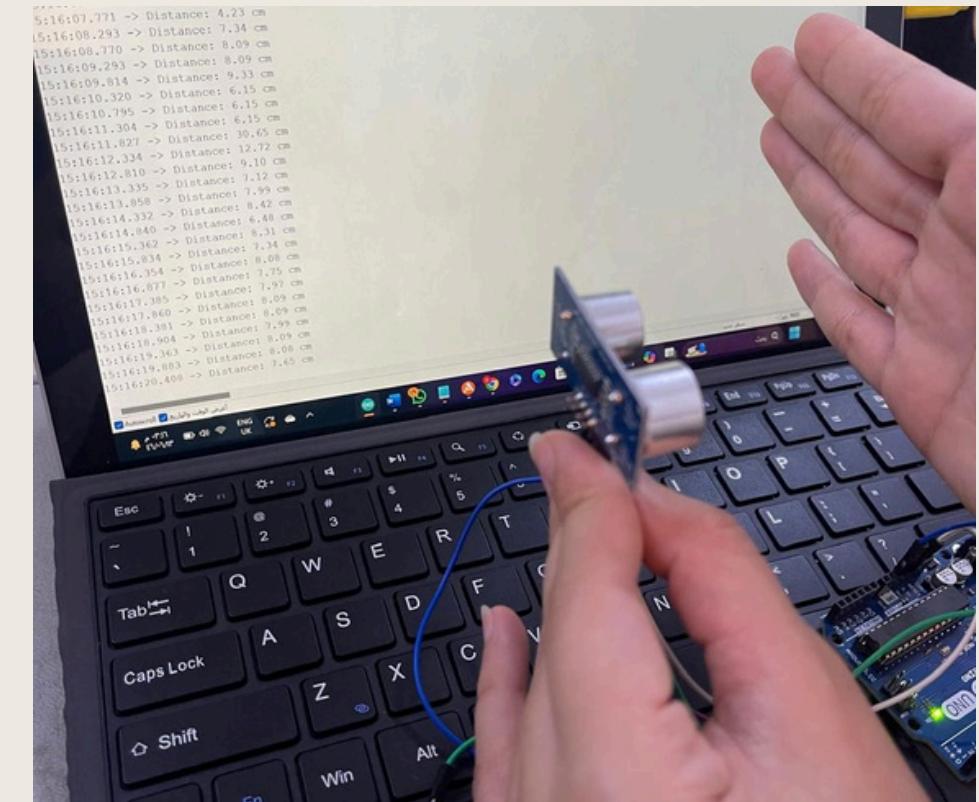
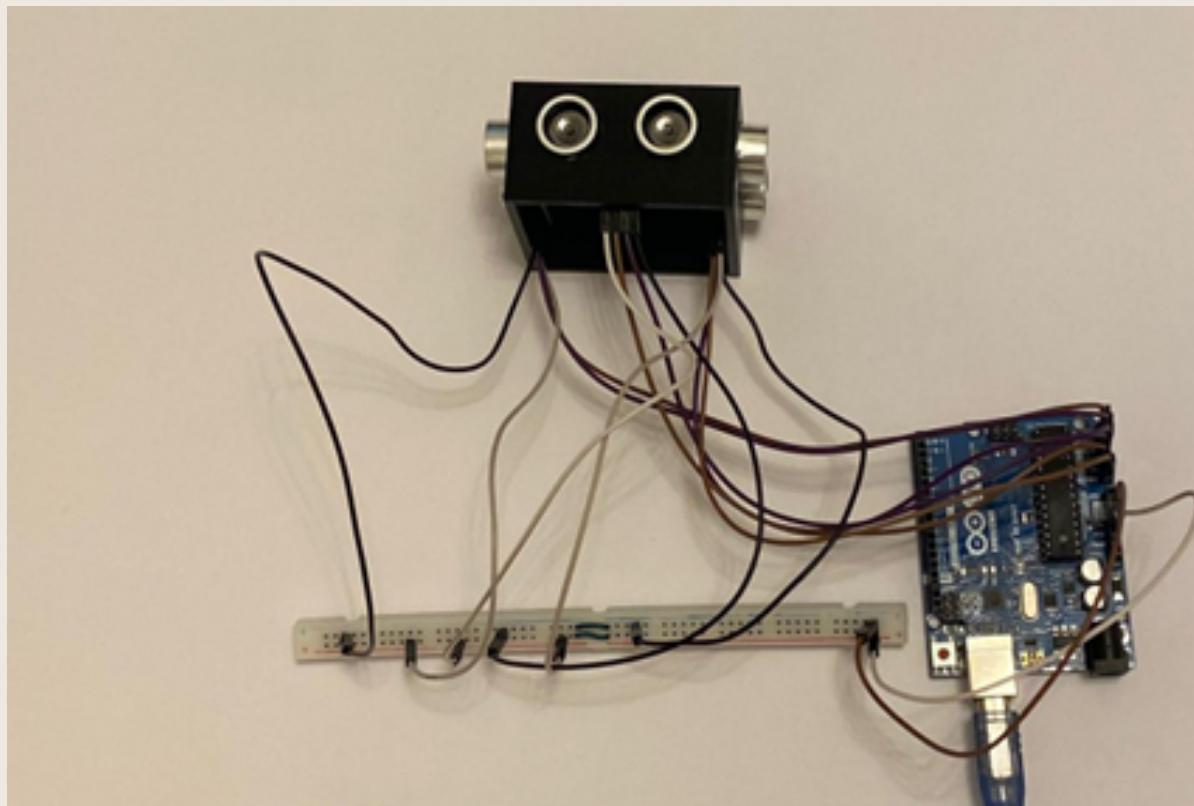


TESTING

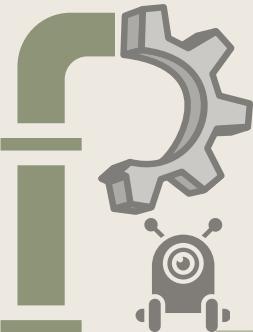


Testing Details

Ultrasonic testing details:



```
Output Serial Monitor X  
Message (Enter to send message to 'Arduino Uno' on 'COM3')  
  
right:10  
left:11  
right:10  
left:11  
right:11  
left:11  
right:10  
left:11  
right:10  
left:11  
right:11
```



TESTING

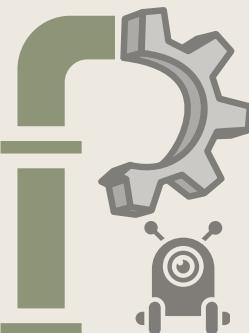


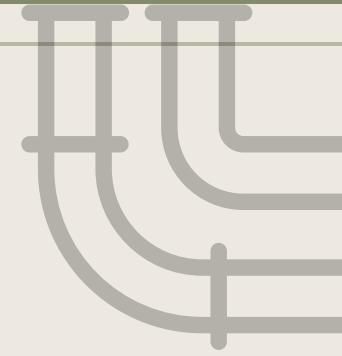
Testing Details

Encoder testing details:



```
Output  Serial Monitor x
Message (Enter to send message)
Distance: 35.74
Distance: 40.84
Distance: 45.95
Distance: 51.05
Distance: 56.15
Distance: 61.26
```



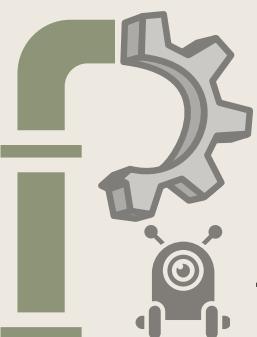


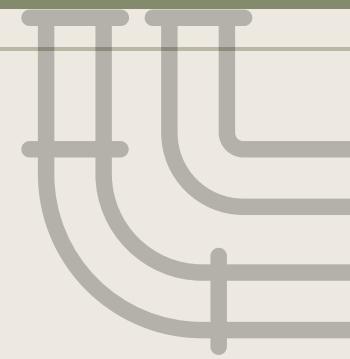
Testing Details



❖ Integration testing details:

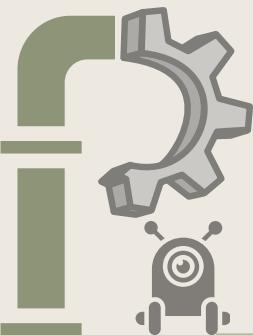
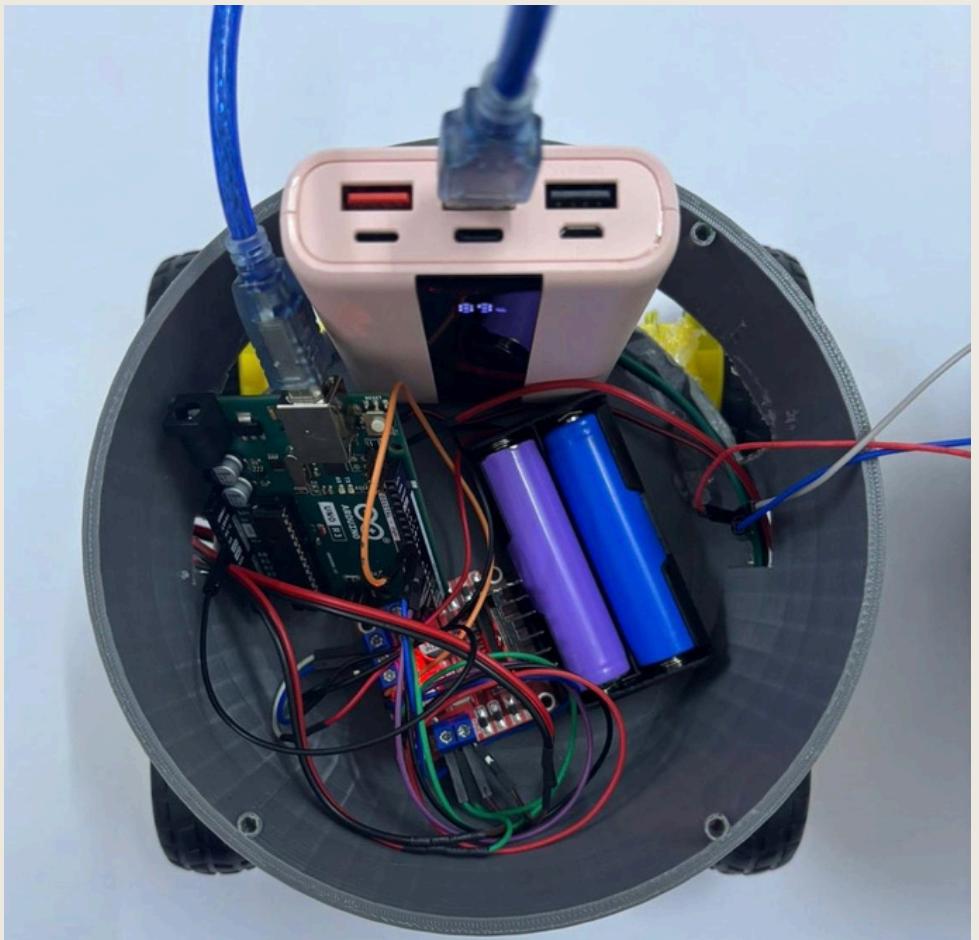
- ❖ Robot motion integration testing
- ❖ Defect detection and classification integration testing
- ❖ Localization integration testing
- ❖ Website communication integration testing





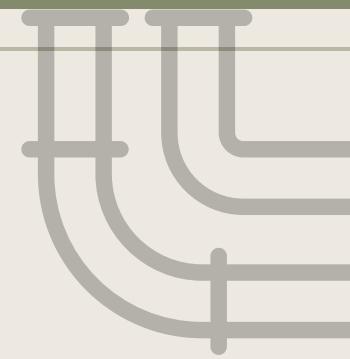
Testing Details

⌚ Robot motion integration testing



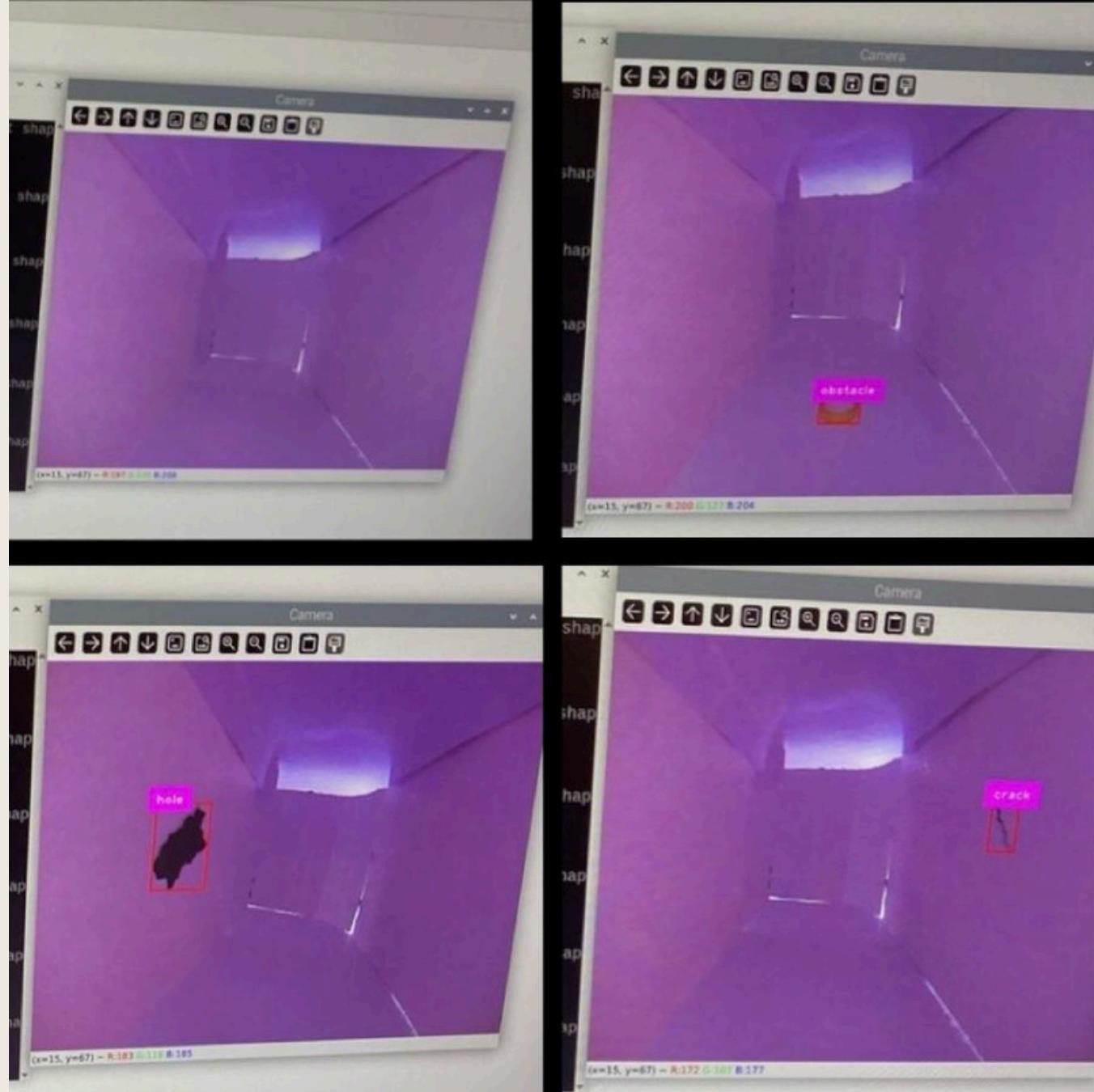
TESTING

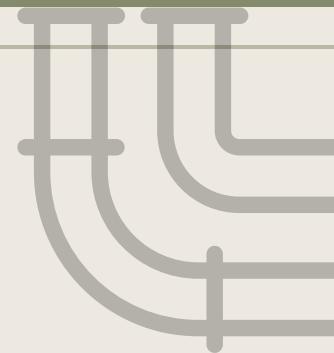
PAGE 27



Testing Details

Defect detection and classification integration testing





Testing Details

LOCALIZATION INTEGRATION TESTING

phpMyAdmin

Server: 127.0.0.1 » Database: petro » Table: defect

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Print](#)

Recent Favorites

New information_schema mysql performance_schema petro New defect param users phpmyadmin test

Showing rows 0 - 1 (2 total, Query took 0.0002 seconds.)

SELECT * FROM `defect`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table Sort by key: N

Extra options

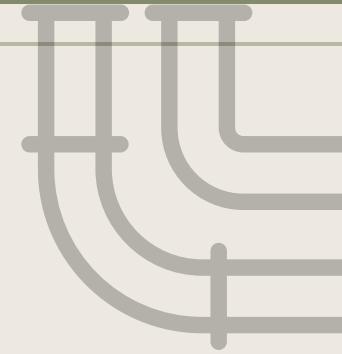
	T					
	id	missionNO	defLocation	defect	img	
<input type="checkbox"/>	Edit	Copy	Delete	10	9 0.3m	hole [BLOB - 9 B]
<input type="checkbox"/>	Edit	Copy	Delete	11	9 0.5m	crack [BLOB - 9 B]

[Check all](#) With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

Show all Number of rows: 25 Filter rows: Search this table Sort by key: N

Query results operations





Testing Details

Website communication integration testing

phpMyAdmin

Server: 127.0.0.1 » Database: petro » Table: defect

Browse Structure SQL Search Insert Export Import Privileges Operations

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(3)			No	None	AUTO_INCREMENT		Change Drop More
2	missionNO	int(3)			No	None			Change Drop More
3	defLocation	varchar(15)	utf8mb4_general_ci		No	None			Change Drop More
4	defect	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
5	img	blob			No	None			Change Drop More

New information_schema mysql performance_schema petro New defect param users phpmyadmin test

Check all With selected: Browse Change Drop Primary Unique Index Spatial Remove from central columns

Petro

Username

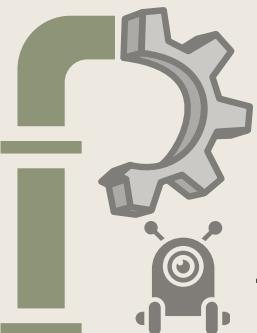
Parameters Dashboard Report Logout

Dashboard

Defect :

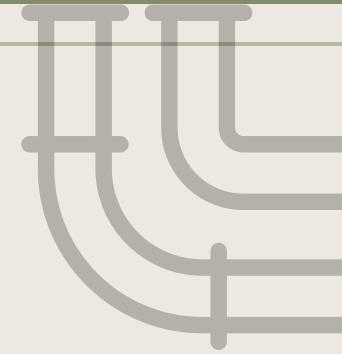
Defect Type and Location:

Type: crack
Location: 0.5m



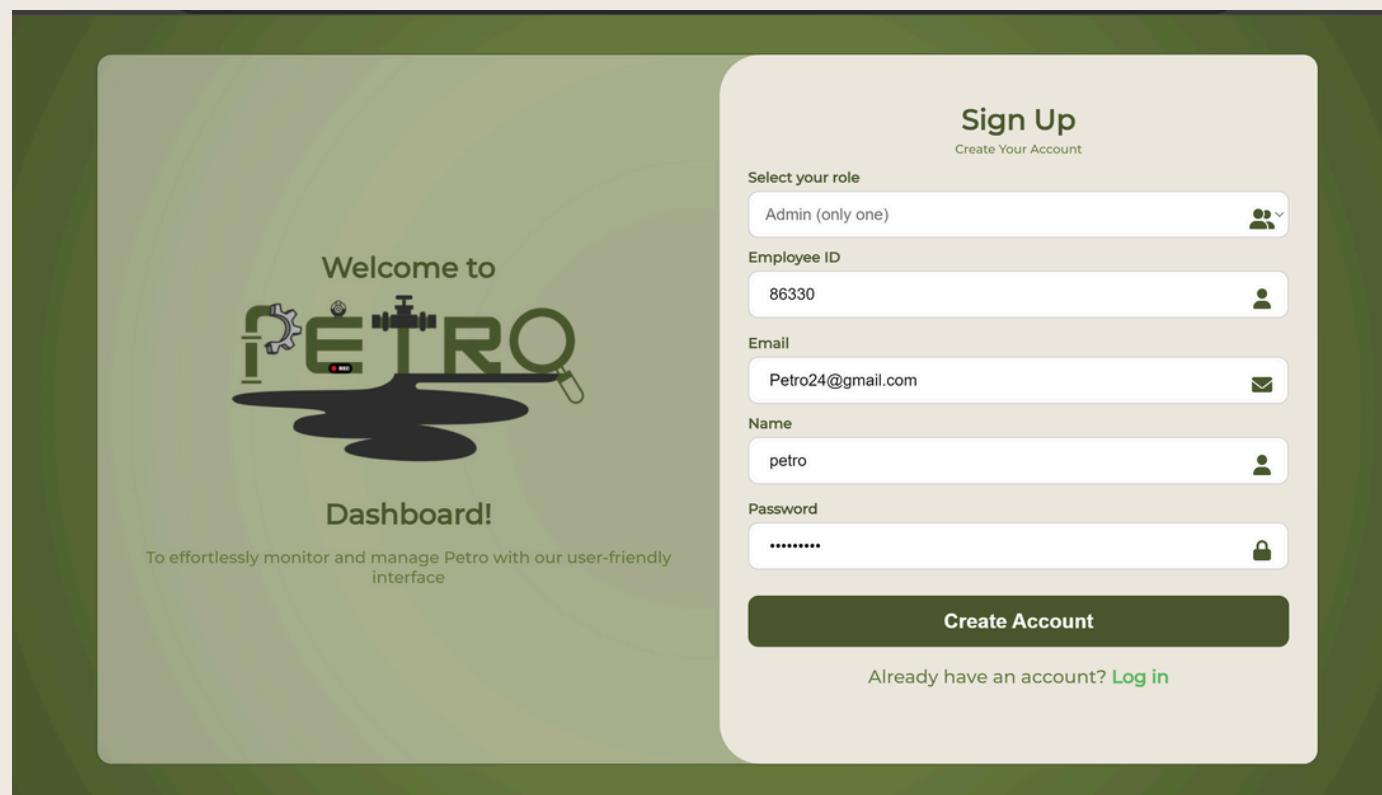
TESTING

PAGE 30



Testing Details

- ❑ Validation and system testing details :
- ❑ Testing the whole system



Welcome to PETRO Dashboard! To effortlessly monitor and manage Petro with our user-friendly interface.

Sign Up
Create Your Account

Select your role: Admin (only one)

Employee ID: 86330

Email: Petro24@gmail.com

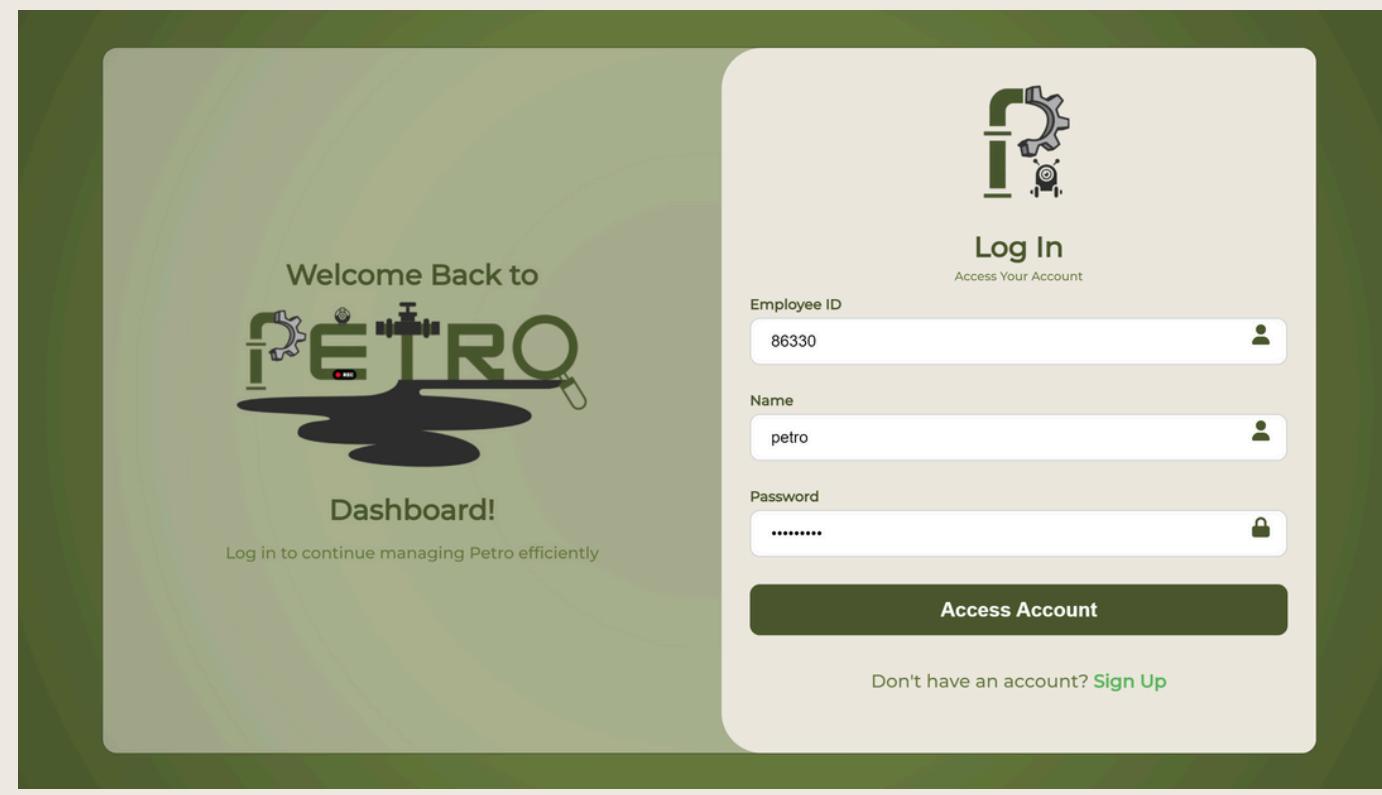
Name: petro

Password: *****

Create Account

Already have an account? [Log in](#)

1 - SIGN UP PAGE REGISTRATION



Welcome Back to PETRO Dashboard! Log in to continue managing Petro efficiently.

Log In
Access Your Account

Employee ID: 86330

Name: petro

Password: *****

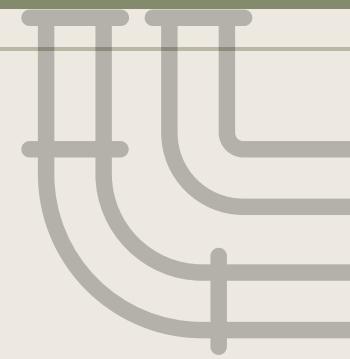
Access Account

Don't have an account? [Sign Up](#)

2 - LOGIN PAGE REGISTRATION



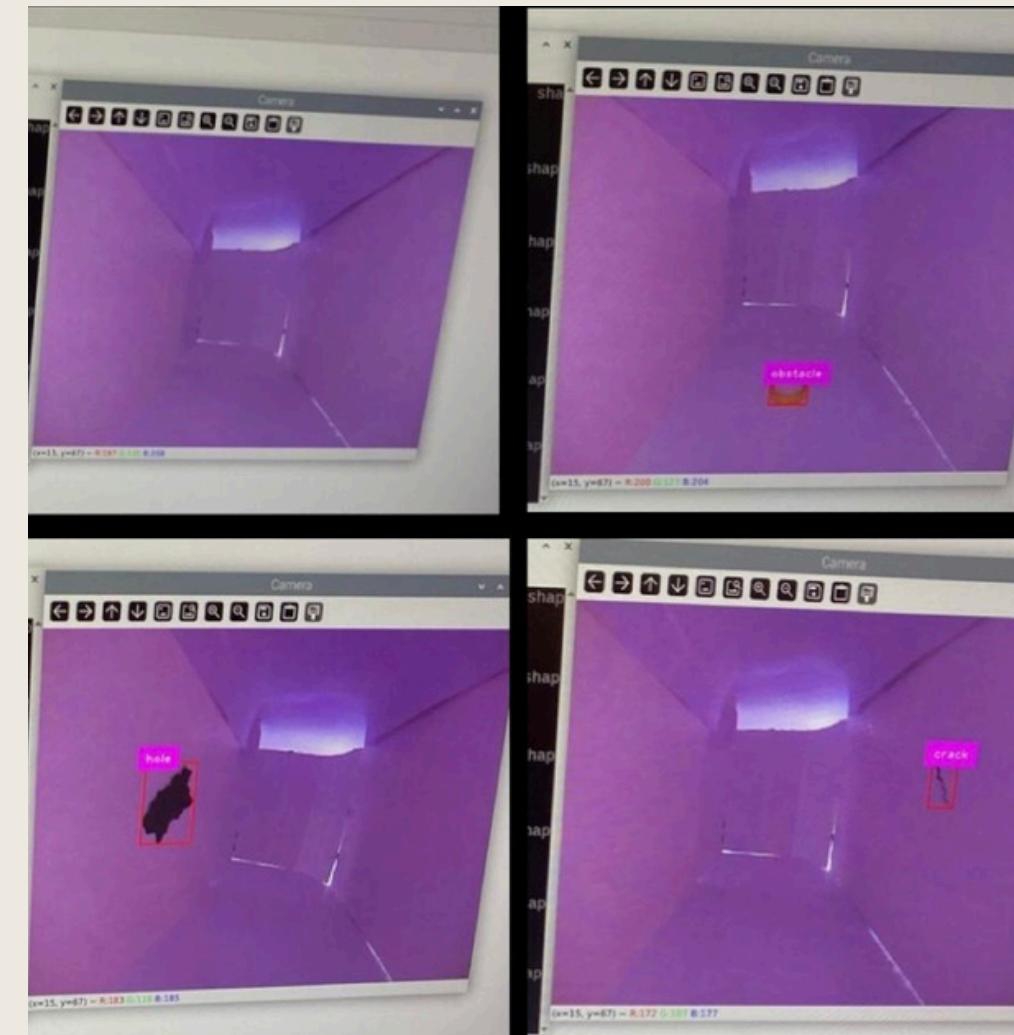
TESTING



Testing Details

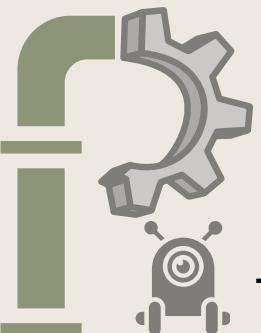
- ❑ Validation and system testing details :
- ❑ Testing the whole system

The screenshot shows the 'Setting The Parameters' page of the Retro software. On the left, there's a sidebar with a user profile for 'Raheed' and navigation links for 'Parameters', 'Dashboard', 'Report', and 'Logout'. The main area has two input fields: 'Pipeline Length in m:' with the value '2' and 'Pipe diameter in (in.):' with the value '12'. Below these fields is a 'Set Parameters' button.



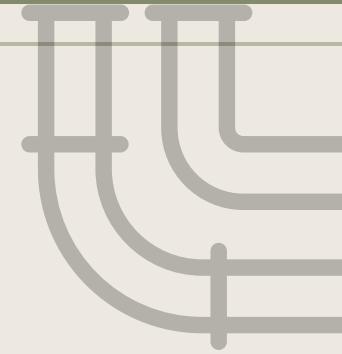
3 - PARAMETER PAGE INFORMATION

4 - ALL POSSIBLE DEFECTS INSIDE THE PIPE



TESTING

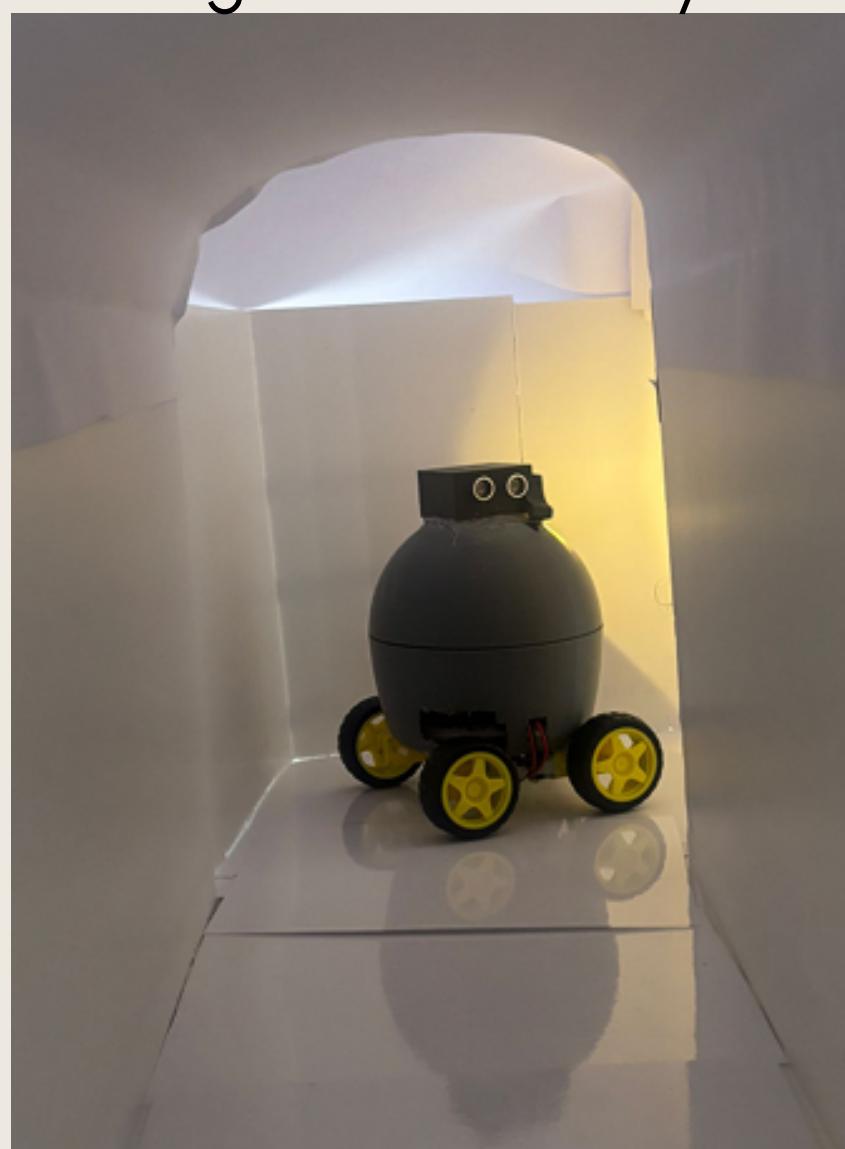
PAGE 32



Testing Details

❑ Validation and system testing details :

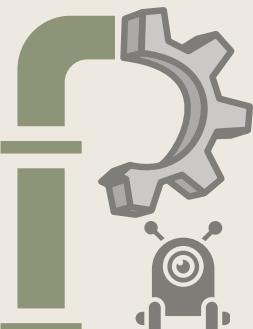
❑ Testing the whole system



The screenshot shows a software interface with a dark green header bar. On the left, there is a sidebar with a user profile for 'Raheed' and links for 'Parameters', 'Dashboard', 'Report' (which is highlighted in green), and 'Logout'. The main area is titled 'Report' and contains two sections: 'Hit the button to present all defect for last mission:' with a 'All Defects' button, and 'Defect Details: Defects for Mission No: 9' with a table.

Image	Defect Type	Location
	hole	0.0m
	crack	0.0m

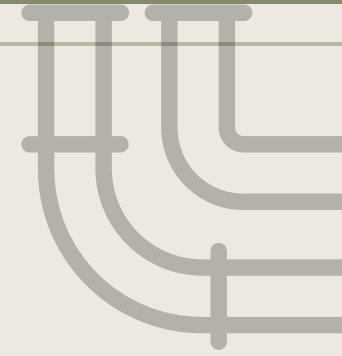
6 - REPORT PAGE INFORMATION



5 - THE ROBOT MOVEMENT

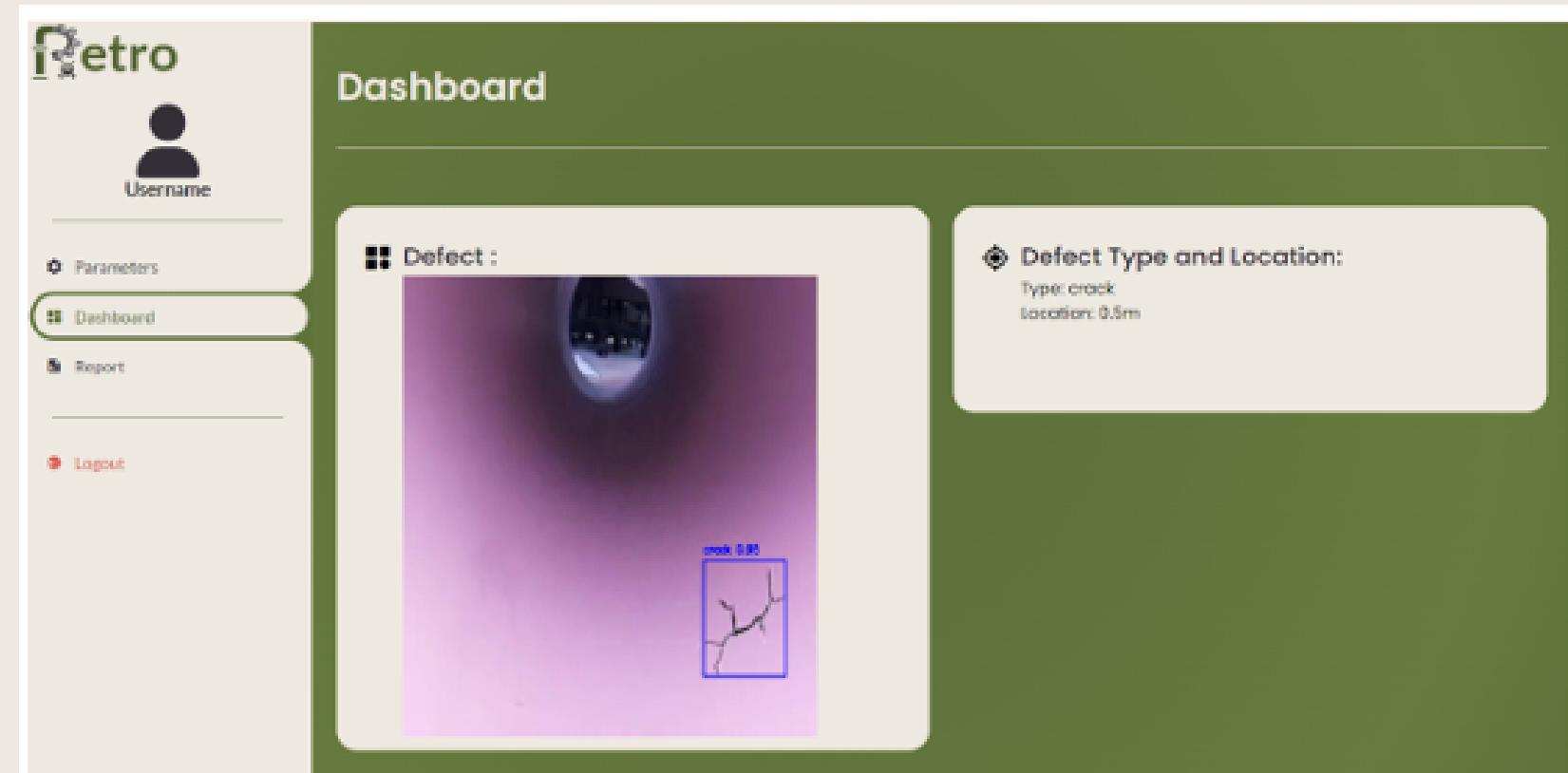
TESTING

PAGE 33

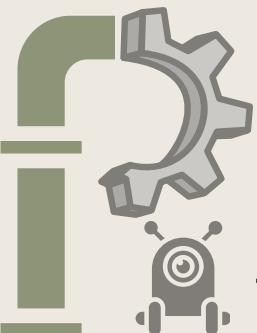


Testing Details

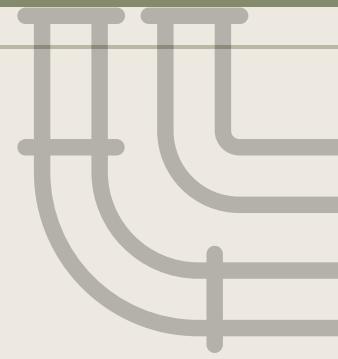
- ❑ Validation and system testing details :
- ❑ Testing the whole system



7 - DASHBOARD PAGE INFORMATION



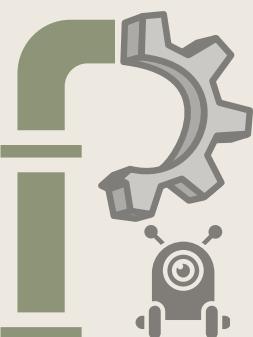
TESTING

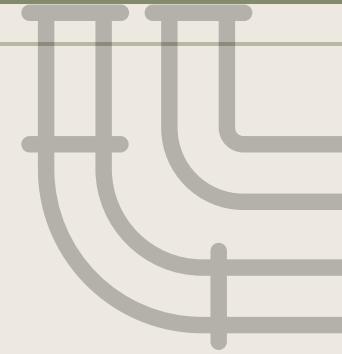


Conclusion

Key Findings:

- YOLOv9t excels: Achieved 77% accuracy, balancing precision and robustness. Approved for future development.
- Robot Semi-Automation: Minimal human intervention, boosting efficiency and reducing oversight.
- Defect Location Identification: Improves accuracy, saving time and resources.
- Real-world Dataset: Enhanced model performance through realistic training data.





Conclusion

Challenges and Limitations

Environmental Factors:

- The project requires pipes exceeding 10 inches in diameter.
- Suitable 16-inch pipes are only available in bulk.

Data Collection:

- Lack of a dedicated dataset for oil and gas pipe defects
- Limited access to real defect images from companies like Aramco and SAPIC
- A general pipe defect dataset was used but required significant cleaning and augmentation.

Operational Limitations:

- The semi-automated mode requires initial user interaction to position the robot at the center.
- Proper positioning is essential for ensuring correct orientation during movement.

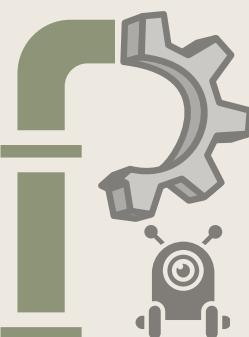




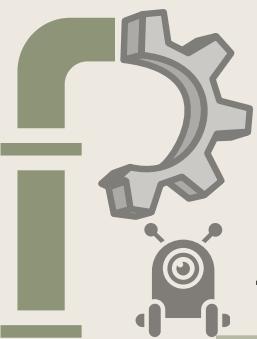
Conclusion

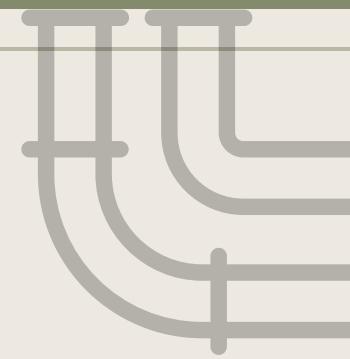
Future Work for Enhancing the semi-autonomous In-Pipe Inspection Robot (IPIR)

- ❑ **Expanded Defect Categories:** Add corrosion, erosion, and pipe displacement to the defect classification system.
- ❑ **Enhanced Hardware:** Upgrade to a powerful processor and high-capacity power bank to support simultaneous execution of wall-following and defect-detection algorithms.
- ❑ **Advance Wireless Communication:** Implement long-range networks for reliable data transmission in pipelines.
- ❑ **Improve Control Algorithms:** Use advanced PID or model predictive control for better navigation in complex pipe geometries.
- ❑ **Defect Classification System:** Standardize defect severity assessment for effective maintenance prioritization.



**THANK YOU
ANY QUESTIONS?**

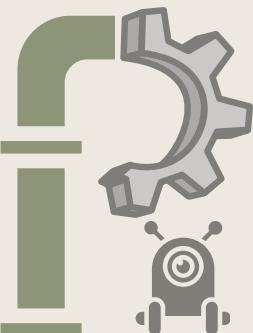




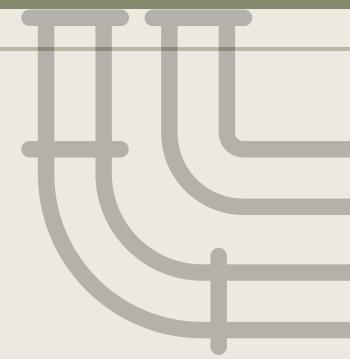
why YOLOv9t

- Comparison Result

Models name	MAP50
Yolov5n	0.699
Yolov8s	0.643
Yolov9t	0.69
Yolov10n	0.524
Yolov11n	0.656



YOLOv9t



Why YOLOv9t

- Result Yolov 5n

```
100 epochs completed in 1.983 hours.  
Optimizer stripped from runs\detect\train2\weights\last.pt, 5.3MB  
Optimizer stripped from runs\detect\train2\weights\best.pt, 5.3MB  
  
Validating runs\detect\train2\weights\best.pt...  
Ultralytics 8.3.28 Python-3.12.7 torch-2.5.1 CUDA:0 (NVIDIA GeForce GTX 1060, 6144MiB)  
YOLOv5n summary (fused): 193 layers, 2,503,529 parameters, 0 gradients, 7.1 GFLOPs  


| Class    | Images | Instances | Box(P) | R     | mAP50 | mAP50-95): 100% | 4/4 [00:0 |
|----------|--------|-----------|--------|-------|-------|-----------------|-----------|
| all      | 114    | 186       | 0.836  | 0.686 | 0.734 | 0.531           |           |
| crack    | 44     | 97        | 0.755  | 0.309 | 0.394 | 0.246           |           |
| hole     | 14     | 14        | 0.854  | 0.929 | 0.945 | 0.743           |           |
| obstacle | 38     | 75        | 0.898  | 0.821 | 0.863 | 0.602           |           |

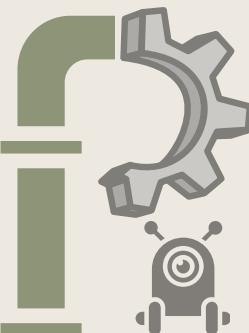
  
Speed: 1.1ms preprocess, 7.2ms inference, 0.0ms loss, 3.0ms postprocess per image  
Results saved to runs\detect\train2
```

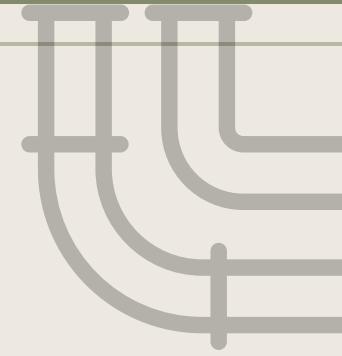
- Result Yolov 11

```
100 epochs completed in 1.465 hours.  
Optimizer stripped from runs\detect\train\weights\last.pt, 5.5MB  
Optimizer stripped from runs\detect\train\weights\best.pt, 5.5MB  
  
Validating runs\detect\train\weights\best.pt...  
Ultralytics 8.3.28 Python-3.12.7 torch-2.5.1 CUDA:0 (NVIDIA GeForce GTX 1060, 6144MiB)  
YOLO11n summary (fused): 238 layers, 2,582,737 parameters, 0 gradients, 6.3 GFLOPs  


| Class    | Images | Instances | Box(P) | R     | mAP50 | mAP50-95): 100% | 4/4 [00:02<0 |
|----------|--------|-----------|--------|-------|-------|-----------------|--------------|
| all      | 114    | 186       | 0.816  | 0.707 | 0.728 | 0.526           |              |
| crack    | 44     | 97        | 0.628  | 0.366 | 0.375 | 0.216           |              |
| hole     | 14     | 14        | 0.974  | 0.929 | 0.934 | 0.739           |              |
| obstacle | 38     | 75        | 0.845  | 0.827 | 0.874 | 0.624           |              |

  
Speed: 0.9ms preprocess, 9.7ms inference, 0.0ms loss, 2.4ms postprocess per image  
Results saved to runs\detect\train
```





Why YOLOv9t

- yolov9t Architecture

```
1 # YOLOv9
2
3 # parameters
4 nc: 80 # number of classes
5 depth_multiple: 1.0 # model depth multiple
6 width_multiple: 1.0 # layer channel multiple
7 #activation: nn.LeakyReLU(0.1)
8 #activation: nn.ReLU()
9
10 # anchors
11 anchors: 3
12
13 # gelan backbone
14 backbone:
15 [
16     # conv down
17     [-1, 1, Conv, [16, 3, 2]], # 0-P1/2
18     # conv down
19     [-1, 1, Conv, [32, 3, 2]], # 1-P2/4
20     # elan-1 block
21     [-1, 1, ELAN1, [32, 32, 16]], # 2
22     # avg-conv down
23     [-1, 1, AConv, [64]], # 3-P3/8
24     # elan-2 block
25     [-1, 1, RepNCSPELAN4, [64, 64, 32, 3]], # 4
26     # avg-conv down
27     [-1, 1, AConv, [96]], # 5-P4/16
28     # elan-2 block
29     [-1, 1, RepNCSPELAN4, [96, 96, 48, 3]], # 6
30     # avg-conv down
31     [-1, 1, AConv, [128]], # 7-P5/32
32     # elan-2 block
33     [-1, 1, RepNCSPELAN4, [128, 128, 64, 3]], # 8
```



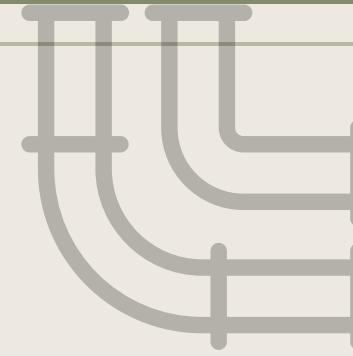
YOLOv9t

- Parameter Yolov 9t

```
from ultralytics import YOLO

# Load a model
model = YOLO("yolov9t.pt")

# Train the model
train_results = model.train(
    data="C:/Users/pca/Desktop/yolo9/yolov9Dataset/data.yaml", # path to dataset YAML
    epochs=100, # number of training epochs
    imgsz=640, # training image size
    freeze = 9,
    device=0, # device to run on
)
```



Why YOLOv9t

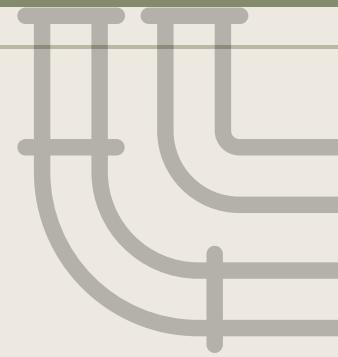
- yolov9t multi-scale parameter tuning

```
: from ultralytics import YOLO

# Load a model
model = YOLO("C:/Users/pca/Desktop/yolo9/runs/detect/train5/weights/best.pt")

# Train the model
train_results = model.train(
    data="C:/Users/pca/Desktop/yolo9/yolov9Dataset/data.yaml", # path to dataset YAML
    epochs=100, # number of training epochs
    imgsz=640, # training image size
    multi_scale=True, #Multi-Resolution
)
```





Why YOLOv9t

- yolov9t Result

```
100 epochs completed in 4.082 hours.  
Optimizer stripped from runs\detect\train5\weights\last.pt, 4.6MB  
Optimizer stripped from runs\detect\train5\weights\best.pt, 4.6MB  
  
Validating runs\detect\train5\weights\best.pt...  
Ultralytics 8.3.28 Python-3.12.7 torch-2.5.1 CUDA:0 (NVIDIA GeForce GTX 1060, 6144MiB)  
YOLOv9t summary (fused): 486 layers, 1,971,369 parameters, 0 gradients, 7.6 GFLOPs  


| Class    | Images | Instances | Box(P) | R     | mAP50 | mAP50-95): 100% | 4/4 [00:04<0 |
|----------|--------|-----------|--------|-------|-------|-----------------|--------------|
| all      | 123    | 197       | 0.813  | 0.711 | 0.736 | 0.553           |              |
| crack    | 46     | 99        | 0.67   | 0.323 | 0.375 | 0.25            |              |
| hole     | 14     | 14        | 0.889  | 0.929 | 0.924 | 0.745           |              |
| obstacle | 47     | 84        | 0.879  | 0.881 | 0.909 | 0.666           |              |

  
Speed: 1.4ms preprocess, 22.6ms inference, 0.0ms loss, 2.1ms postprocess per image  
Results saved to runs\detect\train5
```

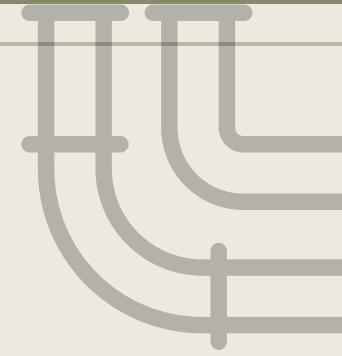
- Final Reslut Yolov 9t

```
100 epochs completed in 8.837 hours.  
Optimizer stripped from runs\detect\train6\weights\last.pt, 4.6MB  
Optimizer stripped from runs\detect\train6\weights\best.pt, 4.6MB  
  
Validating runs\detect\train6\weights\best.pt...  
Ultralytics 8.3.28 Python-3.12.7 torch-2.5.1 CUDA:0 (NVIDIA GeForce GTX 1060, 6144MiB)  
YOLOv9t summary (fused): 486 layers, 1,971,369 parameters, 0 gradients, 7.6 GFLOPs  


| Class    | Images | Instances | Box(P) | R     | mAP50 | mAP50-95): 100% | 4/4 [00:04<0 |
|----------|--------|-----------|--------|-------|-------|-----------------|--------------|
| all      | 123    | 197       | 0.86   | 0.727 | 0.768 | 0.573           |              |
| crack    | 46     | 99        | 0.742  | 0.384 | 0.403 | 0.247           |              |
| hole     | 14     | 14        | 0.939  | 0.929 | 0.972 | 0.78            |              |
| obstacle | 47     | 84        | 0.897  | 0.869 | 0.93  | 0.692           |              |

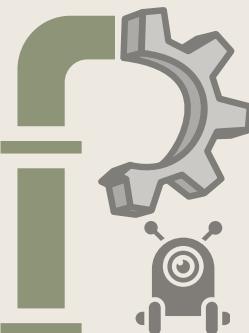
  
Speed: 0.8ms preprocess, 13.1ms inference, 0.0ms loss, 6.3ms postprocess per image  
Results saved to runs\detect\train6
```





Why RPI4

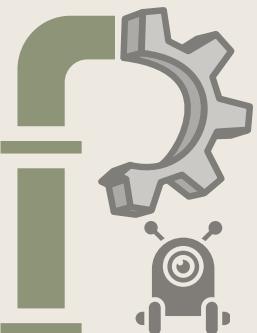
Feature	Raspberry Pi 2	Raspberry Pi 3	Raspberry Pi 4	Raspberry Pi 5
Processors (Quad-core)	ARM Cortex-A7	ARM Cortex-A53	ARM Cortex-A72	Arm Cortex-A76
Clock Speed	900 MHz	1.2 GHz	1.5 GHz	2.4 GHz
Wi-Fi	No	Yes	Yes	Yes
Bluetooth	No	Yes	Yes	Yes
USB Ports	4 × USB 2.0	4 × USB 2.0	2 × USB 2.0 2 × USB 3.0	2 × USB 2.0 2 × USB 3.0
Ethernet Port	Yes	Yes	Yes	Yes
Power Supply	5V/2A	5V/2.5~3A	5V/3A	5V/5A





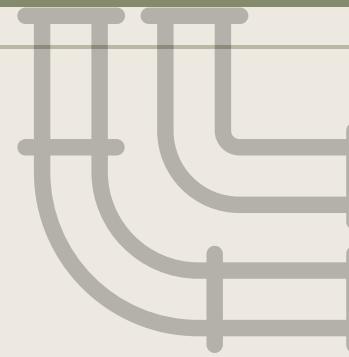
Why Ultrasonic

Feature	HC-SR04	JSN-SR04T	A02YYUW
Type	Non-waterproof	Waterproof	Waterproof
Operating Voltage	5V	5V	3.3V-5V
Measuring Range	2 cm to 400 cm	20 cm to 600 cm	3cm m to 450 cm
Accuracy	±0.3 cm	±1 cm	±1 cm
Weight	Lightweight	Heavier	Lightweight
Cost	Lower cost	Higher cost	Higher cost
Availability	Available	Available	Unavailable



ULTRASONIC

PAGE 45



Implementation Details

Robot Navigation and Localization Code:

The screenshot shows the Thonny IDE interface with the file `encoder.py` open. The code implements a simple encoder-based distance measurement system using a Raspberry Pi's GPIO pins and a serial connection to a ACM0 port.

```
import serial
import time
import RPi.GPIO as GPIO
import time
from time import sleep

ser=serial.Serial('/dev/ttyACM0',9600,timeout=1)
ser.reset_input_buffer()
stateCountTotal=0
GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.IN,pull_up_down=GPIO.PUD_UP)
stateLast = GPIO.input(17)
rotationCount = 0
stateCount=0
stateCountTotal=0
|
circ=204.2
statePerRotation=40
distancePerStep = circ /statePerRotation

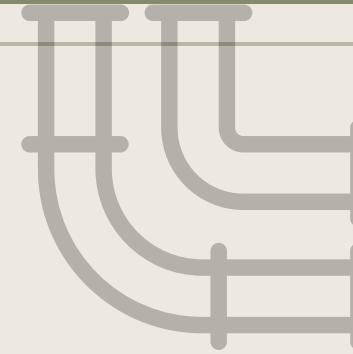
while 1:
    print("send")
    ser.write("0\n".encode('utf-8'))
    stateCurrent= GPIO.input(17)
    if stateCurrent!=stateLast:
        stateLast=stateCurrent
        stateCount+=1
        stateCountTotal +=1

    if stateCount==statePerRotation:
        rotationCount+=1
        stateCount=0

    distance = distancePerStep * stateCountTotal
    print("Distance",distance)
    if distance >= 1000:

        ser.write("1\n".encode('utf-8'))
```

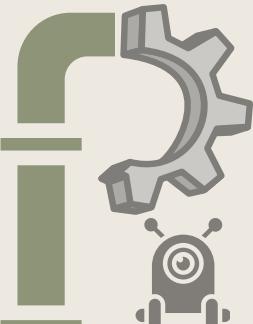


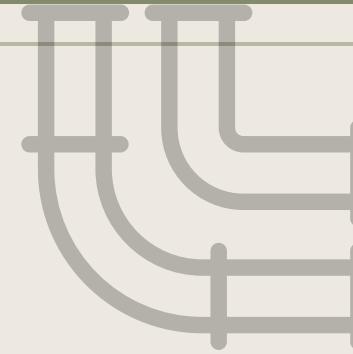


Implementation Details

Functions to control motors speed and direction

```
81 void left() {  
82     analogWrite(enA, 255);  
83     analogWrite(enB, 255);  
84     digitalWrite(in1, LOW);  
85     digitalWrite(in2, HIGH);  
86     digitalWrite(in3, HIGH);  
87     digitalWrite(in4, LOW);  
88 }  
89 void right() {  
90     analogWrite(enA, 255);  
91     analogWrite(enB, 255);  
92     digitalWrite(in1, HIGH);  
93     digitalWrite(in2, LOW);  
94     digitalWrite(in3, LOW);  
95     digitalWrite(in4, HIGH);  
96 }  
97 void stop() {  
98     analogWrite(enA, 0);  
99     analogWrite(enB, 0);  
100    digitalWrite(in1, LOW);  
101    digitalWrite(in2, LOW);  
102    digitalWrite(in3, LOW);  
103    digitalWrite(in4, LOW);  
104 }  
105 void forward(){  
106     analogWrite(enA, 150);  
107     analogWrite(enB, 150);  
108     digitalWrite(in1, HIGH);  
109     digitalWrite(in2, LOW);  
110     digitalWrite(in3, HIGH);  
111     digitalWrite(in4, LOW);  
112 }  
113 void backward(){  
114     analogWrite(enA, 200);  
115     analogWrite(enB, 200);  
116     digitalWrite(in1, LOW);
```





Implementation Details

navigation with localization code:

```
void loop() {
    if (Serial.available()>0){
        String a = Serial.readStringUntil('\n');
        if( a == "0"){
        }
        int frontSensor = sensorFront();
        if(frontSensor <= threshold){
            int leftSensor = sensorLeft();
            int rightSensor = sensorRight();
            delay(1000);
            if(rightSensor>leftSensor){
                right();
                delay(690);
            }
            else if(leftSensor>rightSensor){
                left();
                delay(780);
            }
            else{
                Stop();
            }
        }
        else{
            frontSensor = sensorFront();
            forward();
            delay(300);
            Stop();
            if(frontSensor <= threshold){
                return;
            }
            else{
                Stop();
                delay(3000);
            }
        }
    } else if ( a == "1"){
        Stop();
        delay(5000);
    }
}
else {
    Stop();
}
```