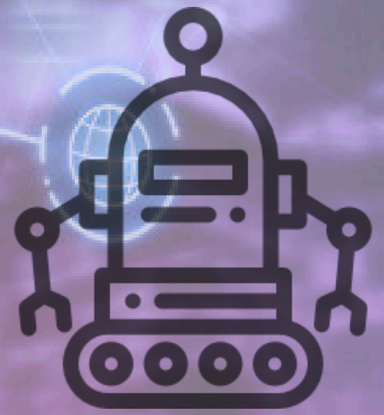
Optimizing Warehouse Outdoor Security with Autonomous Patrol Robots

24-25J-053

Crisis Cop

SLIIT UNI

THE KNOWLEDGE UNIVERSITY



Group Details

PROJECT SUPERVISOR

Ms. Dinithi Pandithage

Lecturer

Faculty of Computing | Computer Systems Engineering

PROJECT CO- SUPERVISOR

Mr. Uditha Dharmakeerthi

Academic Fellow

Faculty of Computing | Computer Systems Engineering

Ashinshana G.I

IT21266232

Madhusankha W.V.S.

IT21252372

Gunawardhana K.A.S.H

IT21242472

D.L.B.S Liyanaarachchi

IT21210860|

SLIIT UNI

THE KNOWLEDGE UNIVERSITY

Introduction

- Patroling larger outdoor environments are labour intensive
- Problems with instantly discovering unusual behaviours
- Need to respond to incidents instantly for a reliable patrolling system.
- Recording the specific details about the incidents is crucial.
- Summarize all the incidents to get an overall idea is an important factor.

Research Question

- How does the integration of human patrolling in larger indoor and outdoor environments influence the reliability of security, and what improvements can be achieved through the implementation of IoT based robotic patroling system with multiple robots that can communicate with each other

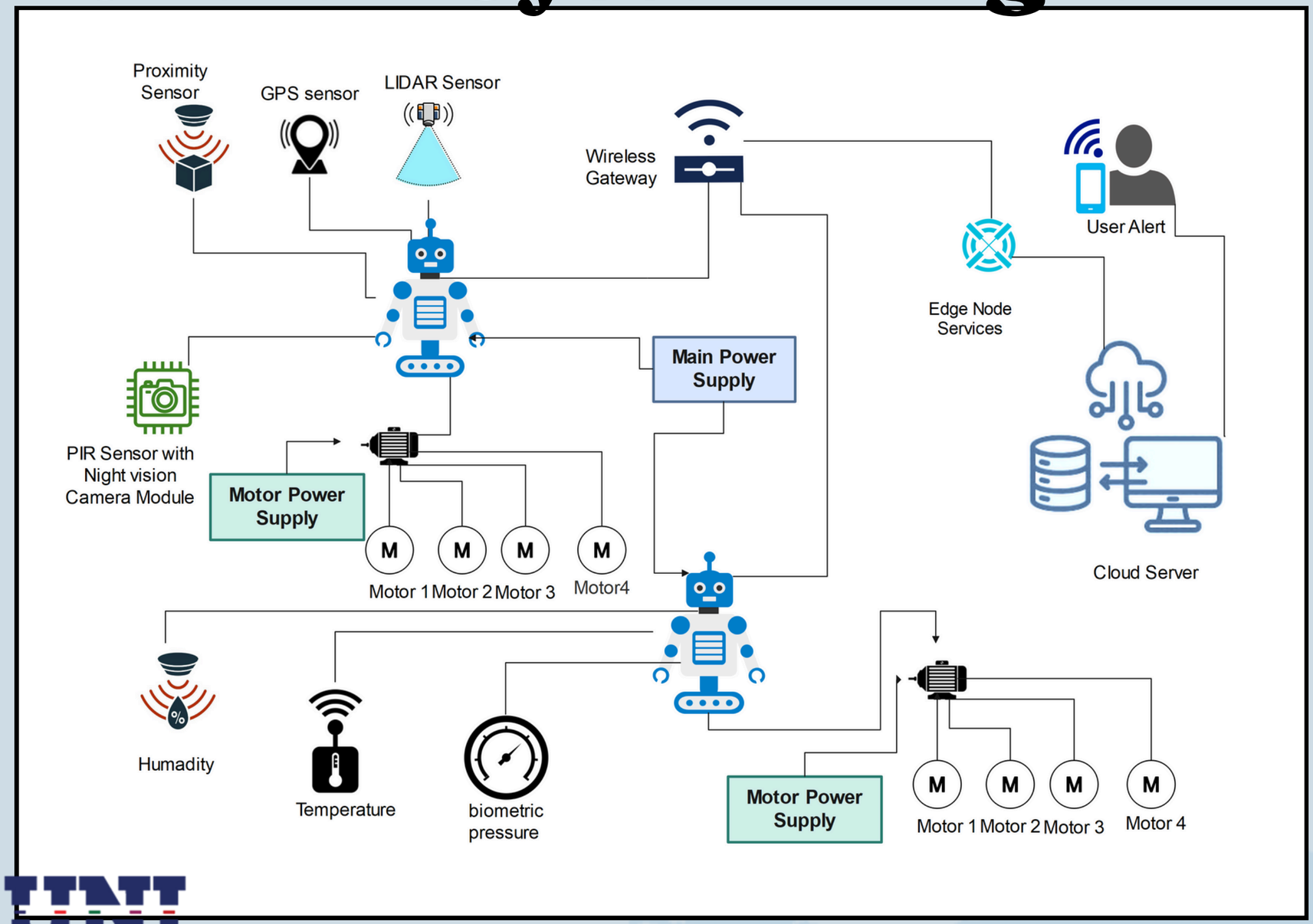
Main objective

- Our main goal is to deploying outdoor patrol robots can greatly improve security and surveillance by addressing challenges in mobility, detection, navigation, communication, and weather conditions. These robot enable continuous and efficient monitoring,recording the incidents, escalating and reducing the reliance on human patrols and offering a scalable solution for large areas.

Sub objectives

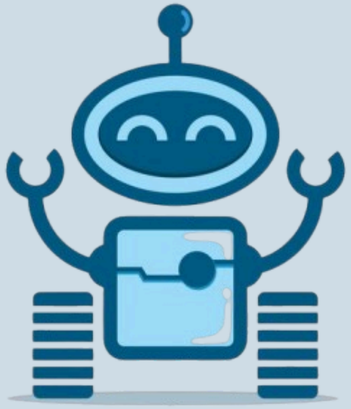
- Create a low-power algorithm to efficiently upload sensor and video data, using edge computing to reduce cloud data size
- Create an algorithm for robots to follow patrol routes and adapt to changes in the warehouse using GPS, waypoints, and sensors for navigation and obstacle detection.
- Equip slave robots with sensors and use a fusion algorithm to predict environmental changes and ensure optimal performance.
- Create an algorithm for slave robots to coordinate and communicate during patrols.

Overall System Diagram



IT21266232 |Ashinshana G.I

BSc (Hons) in Information Technology
Computer Systems & Network Engineering



Background

- Data generated from the devices must be processed and responded in short amount of time
- Need Energy efficient method to transfer these data to cloud
- Edge computing allowed to offload computational task from the cloud
- Existing technologies like Lora have high latency when there is multiple sensor nodes
- Need data compression when sending the sensor data to the cloud

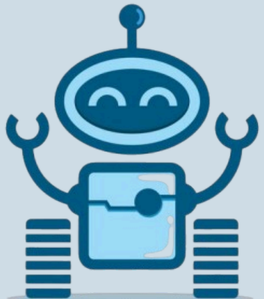
Research Gap

- Existing systems have high latency when computing data from multiple sensors
- The data quality is lost due to compression when uploading to the cloud and they are not adaptive acording to data type.
- lot of energy is consumed in the process of computing and uploading data to the cloud.

Research Question

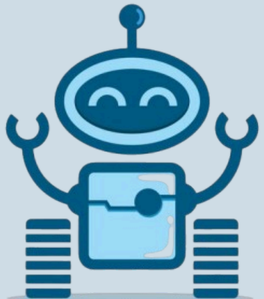
- How can a real-time adaptive data compression algorithm be developed to minimize quality loss while reducing latency in the transmission of video and sensor data from edge devices to the cloud?

- What impact do different edge computing architectures have on energy consumption and data processing efficiency when handling data from multiple sensors



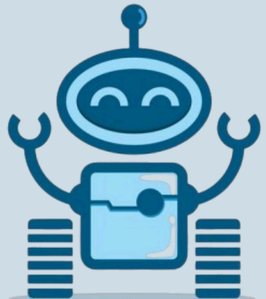
Main Objective

- Develop an energy-efficient algorithm that minimizes power consumption while ensuring real-time monitoring and secure data transmission through edge computing.



Sub Objective

- Energy-Efficient Algorithm Development
- Real-Time Monitoring System
- Secure and Efficient Cloud Transmission
- Automated Alert & Notification System
- Monitoring Dashboard Development



Comparison

	Proposed research	Research Paper [1]	Research Paper [2]	Research Paper [3]
Low latency Edge Computing	✓	✓	✗	✗
Lossless Compression of Data	✓	✗	✓	✗
Energy Efficent Computatuion and Aggregation of data	✓	✗	✓	✓

Completion of the project

- ✓ Implementing function to compress the sensor data (using LZ4).
- ✓ Imprlementing function to encrypt the data (using ChaCha20Poly1305)
- ✓ Implementing Encryption & video compression based on the battery precentage of the robot
- ✓ Video frames and sensor data are processed locally (edge device) before sending to the cloud
- ✓ Intergrating with the Web Application

Implementation

Implementing function to encrypt the data (using ChaCha20Poly1305)

```
# Encryption key (32 bytes for ChaCha20)
ENCRYPTION_KEY = os.urandom(32)
NONCE_SIZE = 12  # ChaCha20 nonce size
```

```
# Function to encrypt data using ChaCha20-Poly1305
def encrypt_data(data: bytes) -> bytes:
    nonce = os.urandom(NONCE_SIZE)
    cipher = ChaCha20Poly1305(ENCRYPTION_KEY)
    encrypted_data = cipher.encrypt(nonce, data, None)  # None for associated data
    return nonce + encrypted_data


# Function to decrypt data using ChaCha20-Poly1305
def decrypt_data(encrypted_data: bytes) -> bytes:
    nonce = encrypted_data[:NONCE_SIZE]
    ciphertext = encrypted_data[NONCE_SIZE:]
    cipher = ChaCha20Poly1305(ENCRYPTION_KEY)
    return cipher.decrypt(nonce, ciphertext, None)  # None for associated data
```

Implementing function to compress the data

```
# Function to compress data using LZ4
def compress_data(data: bytes) -> bytes:
    return lz4.frame.compress(data)


# Function to decompress data using LZ4
def decompress_data(compressed_data: bytes) -> bytes:
    return lz4.frame.decompress(compressed_data)
```

Implementation

Implementing Encryption & video compression based on the battery precentage of the robot

```python
# Video compression levels based on battery status
VIDEO_COMPRESSION_LEVELS = {
    "low": 60,  # High compression (low quality)
    "medium": 40,  # Medium compression
    "full": 20,  # Low compression (high quality)
}
```

```python
# Function to process video frames based on battery level
def process_video_frame(frame, battery_level: str):
    compression_level = VIDEO_COMPRESSION_LEVELS.get(battery_level, 40)  # Default to medium
    encode_param = [int(cv2.IMWRITE_JPEG_QUALITY), compression_level]
    _, buffer = cv2.imencode(".jpg", frame, encode_param)
    return buffer.tobytes()
```

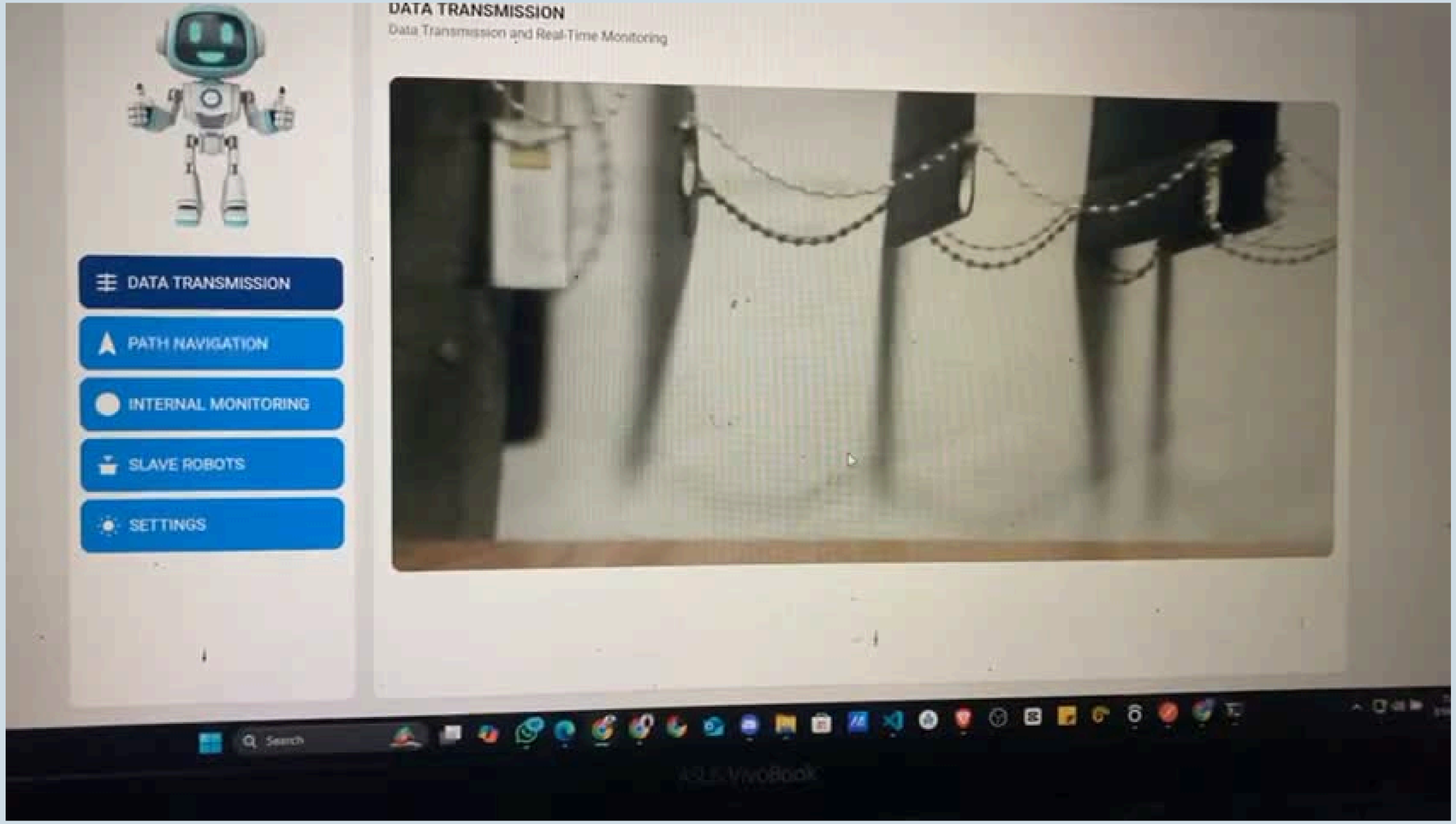
Implementation

Video frames and sensor data are processed locally (edge device) before sending to the cloud

```python
# API endpoint to upload video frames and sensor data
@app.post("/upload")
async def upload_data(battery_level: str, file: UploadFile = File(...)):
    try:
        # Read video frame
        frame_data = await file.read()
        frame = cv2.imdecode(np.frombuffer(frame_data, np.uint8), cv2.IMREAD_COLOR)
        if frame is None:
            raise HTTPException(status_code=400, detail="Invalid image data")

        # Process video frame based on battery level
        processed_frame = process_video_frame(frame, battery_level)

        # Compress and encrypt the frame
        compressed_frame = compress_data(processed_frame)
        encrypted_frame = encrypt_data(compressed_frame)

        # Save or send the encrypted frame to the cloud

        return {"message": "Data uploaded successfully", "encrypted_size": len(encrypted_frame)}

    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

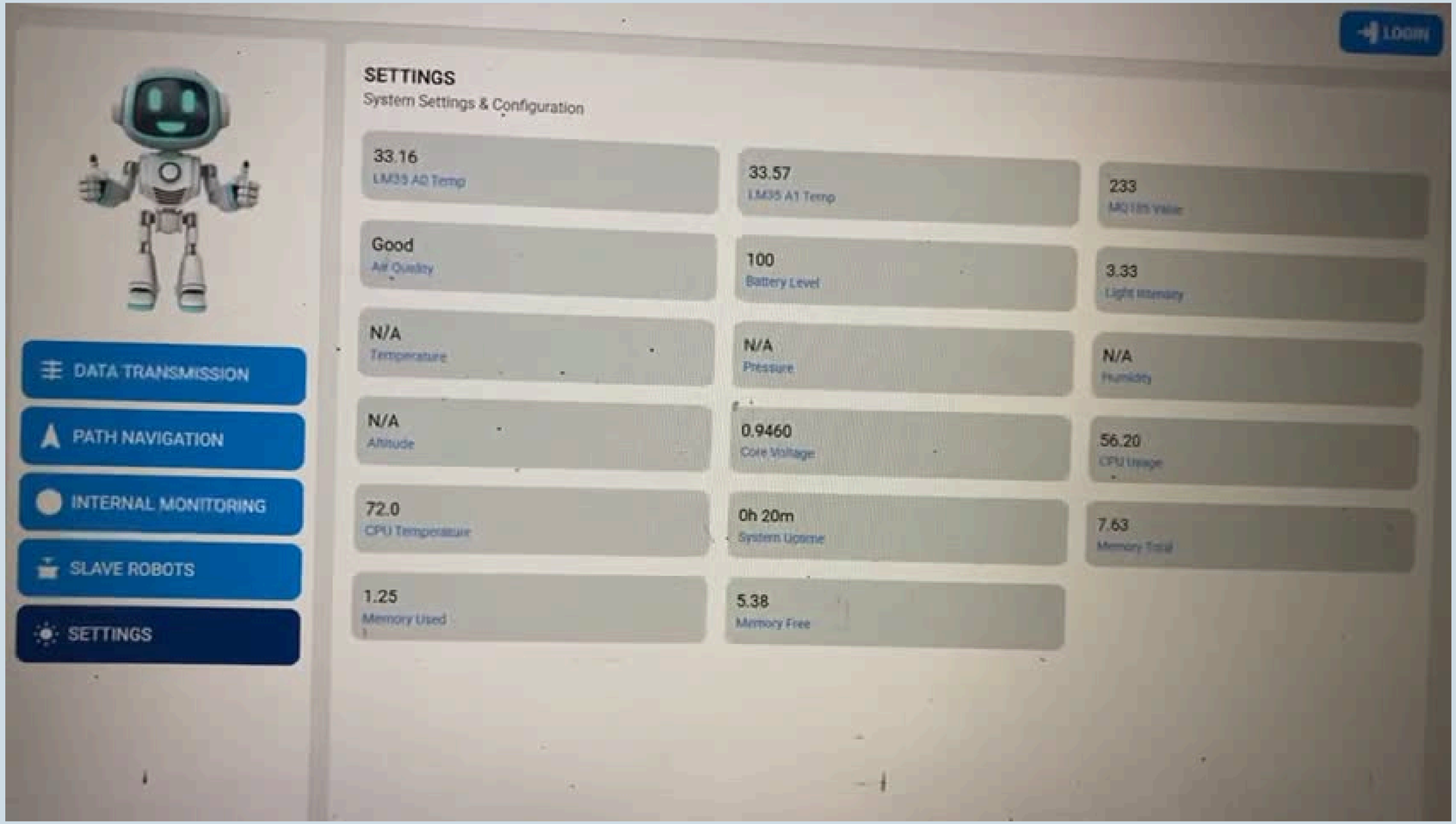
Demonstration

Intergrating with the Web Application

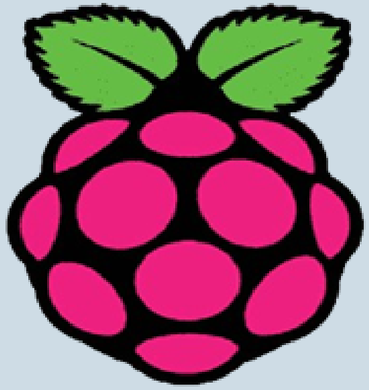


Demonstration

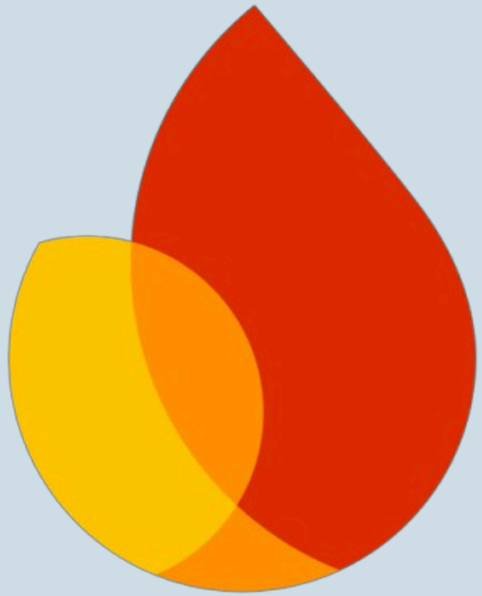
Intergrating with the Web Application



Technologies



RaspberryPi



Reference

[1] T. N. Gia, L. Qingqing, J. P. Queralta, H. Tenhunen, Z. Zou, and T. Westerlund, "Lossless compression techniques in edge computing for mission-critical applications in the IoT," in Proc. 2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU), Kathmandu, Nepal, 2019, pp. 1–2, doi: 10.23919/ICMU48249.2019.9006647.

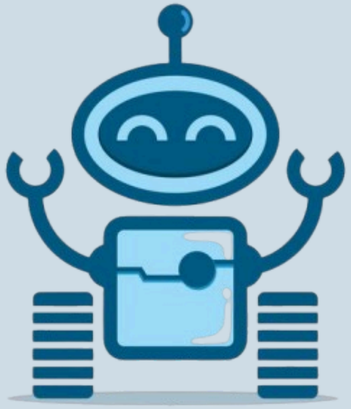
[2] H. Xue, B. Huang, M. Qin, H. Zhou, and H. Yang, "Edge Computing for Internet of Things: A Survey," in Proc. 2020 International Conferences on Internet of Things (iThings), IEEE Green Computing and Communications (GreenCom), IEEE Cyber, Physical and Social Computing (CPSCom), IEEE Smart Data (SmartData), and IEEE Congress on Cybermatics (Cybermatics), Rhodes, Greece, 2020, pp. 755–760, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00130.

[3] M. Talebkhah, A. Sali, M. Marjani, M. Gordan, S. J. Hashim, and F. Z. Rokhani, "Edge computing: Architecture, Applications and Future Perspectives," in Proc. 2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), Kota Kinabalu, Malaysia, 2020, pp. 1–6, doi: 10.1109/IICAIET49801.2020.9257824.

IT21252372 |Madhusankha W.V.S.

BSc (Hons) in Information Technology
Computer Systems & Network Engineering

SLIIT UNI
THE KNOWLEDGE UNIVERSITY



Background

- Automation is crucial in large industrial settings to ensure reliable and continuous monitoring of human security patrols, as they may become tired and inconsistent.

- Continuous observation, lower operating costs, and the removal of inefficiencies caused by humans are all possible with robotic patrol systems. The robots traverse hazardous and challenging terrain.

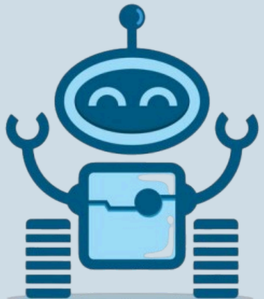
- In order for the robot to effectively adapt to changing situations, data from sensors such as Lidar, GPS, and proximity sensors are combined to enable precise navigation and obstacle avoidance.

Research Question

- How could possibly a multi-sensor fusion technique improve the patrol robot's capacity to follow predetermined paths while dynamically adjusting to changing environmental conditions?

- What algorithms can be built to allow the robot to adjust its patrol route automatically in response to detected barriers or dynamic situations, such as intense light or motion?

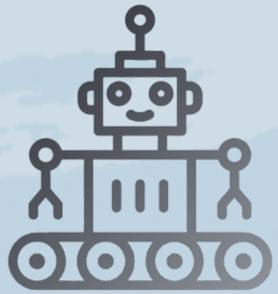
- How can the robot use Lidar and GPS data to accurately map new routes for future investigations while avoiding obstacles?



Research Gap

- ·There has been a lack of extensive research into properly integrating numerous sensors (Lidar, GPS) for adaptive security patrol robots, limiting developments in navigation and decision-making.

- Current algorithms primarily address predefined routes or basic obstacle avoidance.



Comparison

	Proposed research	Research Paper 1	Research Paper 2	Research Paper 3
Multi-Sensor Fusion (combination of Lidar, cameras, GPS, and proximity sensors)	✓	✓	✗	✓
Dynamic Route Adaptation	✓	✗	✓	✓
Autonomous Mapping and Path Planning	✓	✗	✗	✗
Real-time Environmental Adaptability	✓	✗	✓	✗

Specific Obective

- Design a robotic system with Lidar, and GPS that patrols the warehouses autonomously. The route should automatically be updated according to environmental conditions.

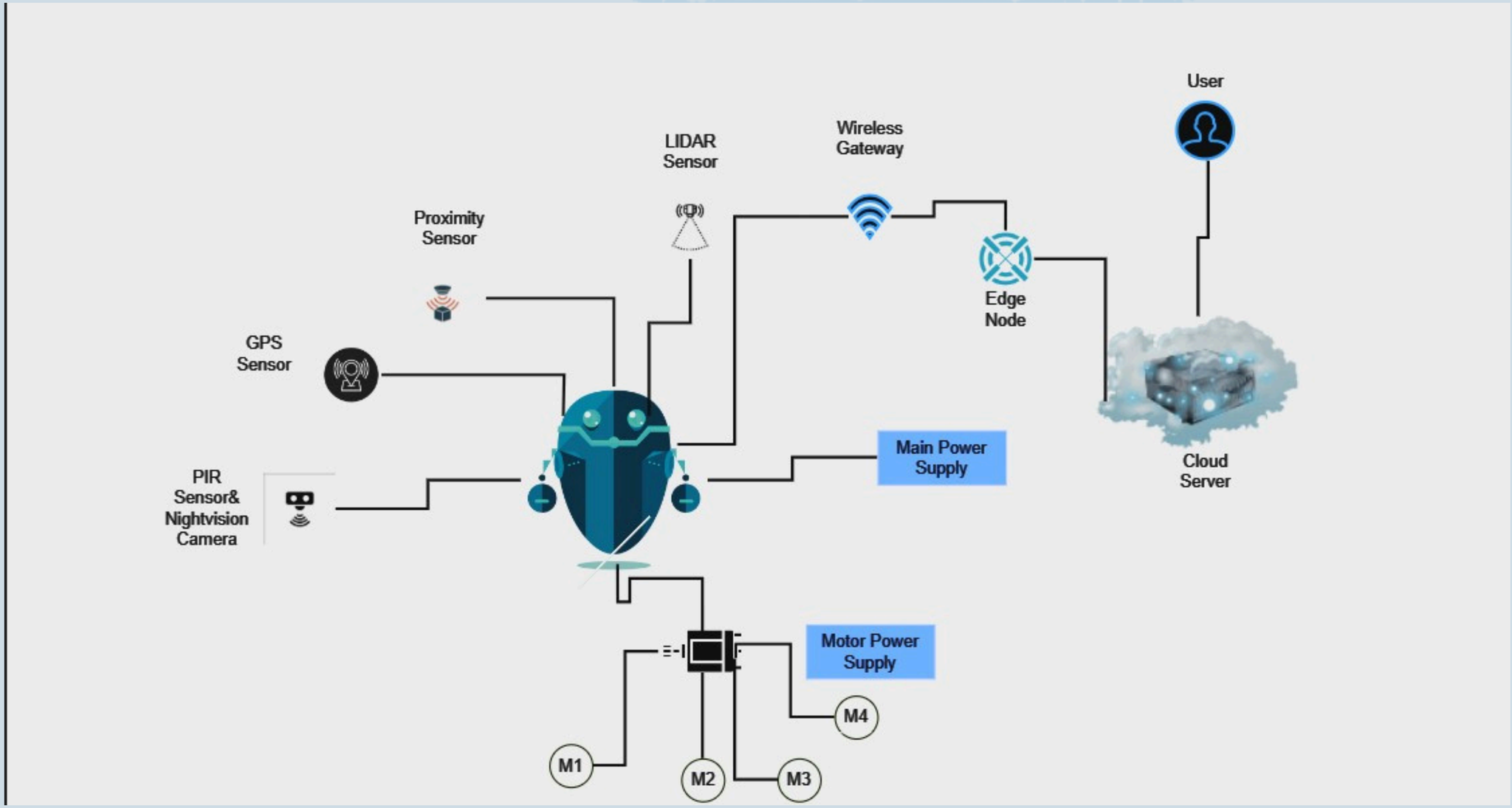
Sub Objective

- Create algorithms that allow the robot to independently follow established patrol routes while adapting to changing warehouse conditions.

- Create algorithms that allow the robot to adjust its patrol route in response to sensor and camera data, addressing conditions such as intense light or motion.

- Integrate Lidar to ensure the robot can identify obstacles, map new paths, and navigate effectively and safely.

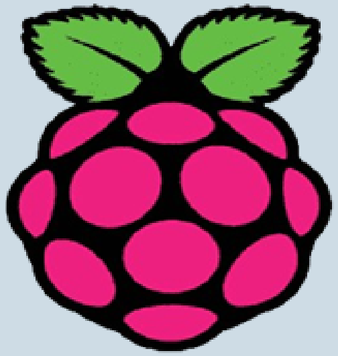
System Diagram



Technologies

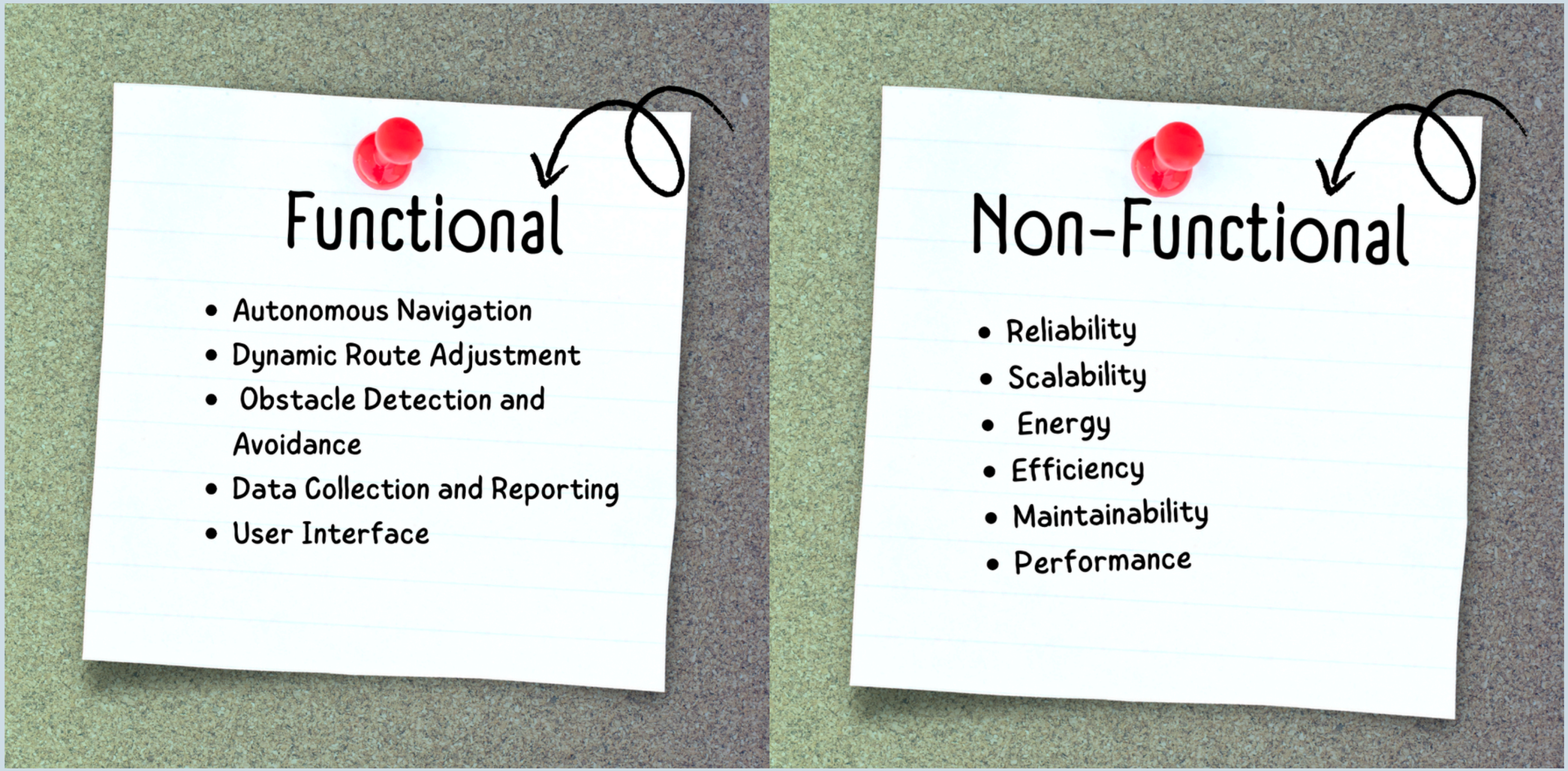
Firebase

MQTT



RaspberryPi

Fuctonal & Non-Functional Requirement

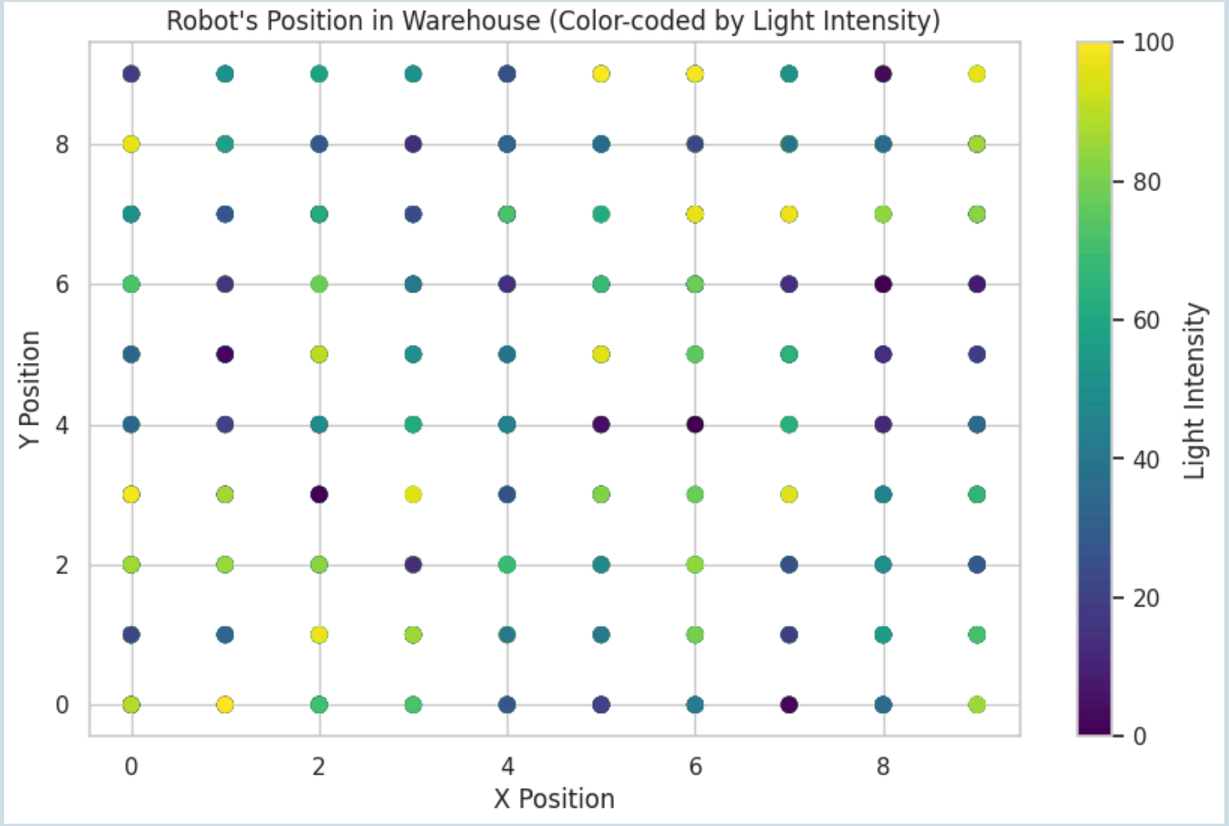


Completion of the Project

- Implement a machine learning model (Reinforcement learning) ,Used Q-Learning: A Popular RL Algorithm.
- Created a data Visualization part to get a deep idead about the Machine learning.
- Use LIDAR SLAM for Autonomous Navigation (Initial Map Creation).
- Store Path Information with Waypoints for Different Goals.
- Patrol Route Navigation with ROS and Obstacle Avoidance.
- Create a Map Using LIDAR by Rotating the Robot.

Data visualization Graph

```
plt.figure(figsize=(10, 6))
plt.scatter(df['x_position'], df['y_position'], c=df['light_intensity'], cmap='viridis', s=50)
plt.colorbar(label='Light Intensity')
plt.title("Robot's Position in Warehouse (Color-coded by Light Intensity)")
plt.xlabel('X Position')
plt.ylabel('Y Position')
plt.show()
```



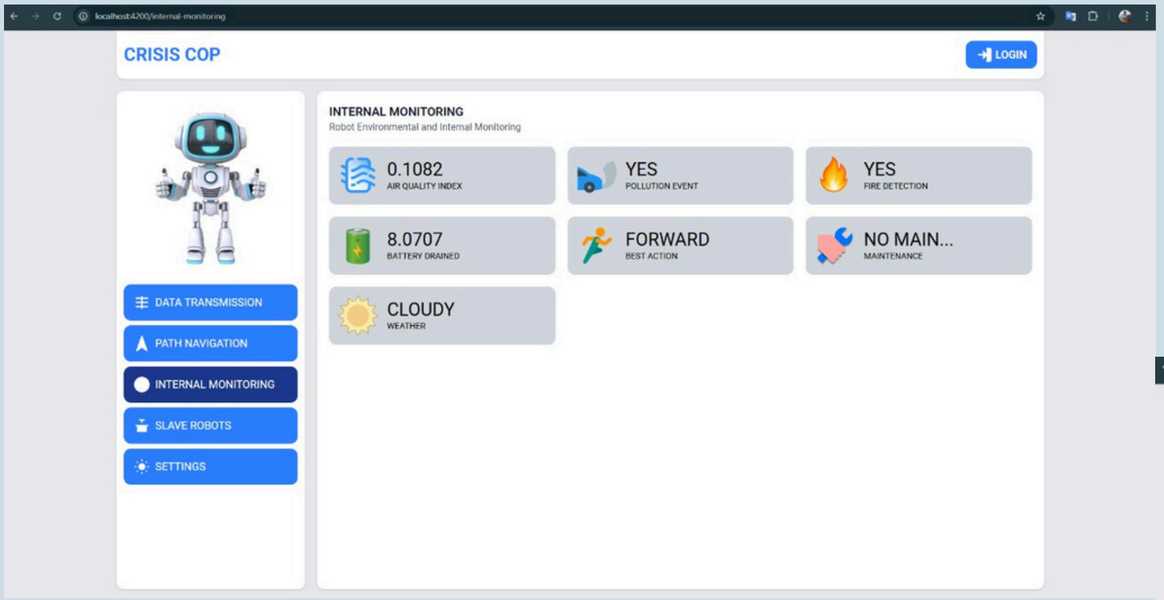
Q learning Algorithum

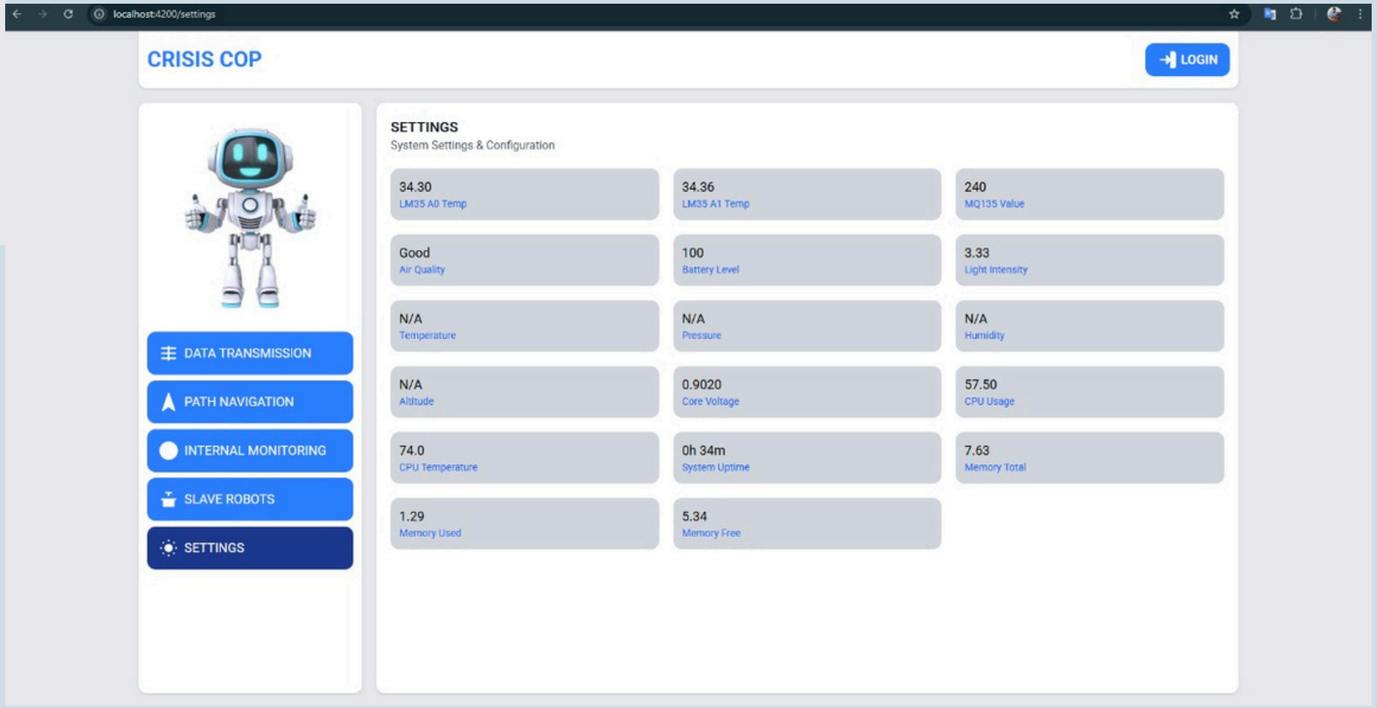
- Q-learning is a specific RL algorithm that helps an agent learn the best action to take in different situations. It is a model-free method, meaning it does not require prior knowledge of the environment. It learns purely from experience.

Model Development (Q learning Alogrithum)

```python
for episode in range(episodes):
    # Random initial state selection from dataset
    row = df.sample(1).iloc[0]
    state = (row['x_position'], row['y_position'], row['light_intensity'],
             row['motion_detected'], row['obstacle_distance'], row['obstacle_present'])

    total_rewards = 0

    done = False
    for _ in range(steps_per_episode):
        # Choose action using epsilon-greedy strategy
        if np.random.rand() < epsilon:
            action = random.choice(action_space)  # Explore
        else:
            action = action_space[np.argmax(q_table[state])]  # Exploit best action

        # Get reward based on action taken
        reward = get_reward(row, action)
        total_rewards += reward

        # Get next state by sampling another row (simulate environment change)
        next_row = df.sample(1).iloc[0]
        next_state = (next_row['x_position'], next_row['y_position'], next_row['light_intensity'],
                      next_row['motion_detected'], next_row['obstacle_distance'], next_row['obstacle_present'])

        # Update Q-table using Q-learning formula
        q_table[state][action_space.index(action)] = q_table[state][action_space.index(action)] + learning_rate * (
            reward + discount_factor * np.max(q_table[next_state]) - q_table[state][action_space.index(action)]
        )
```

Web Application





Reference

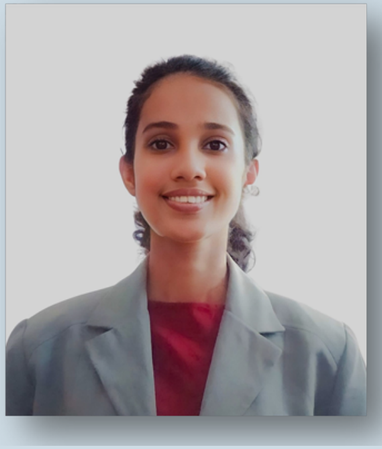
[1]S. Joy, Richard Lincoln Paulraj, Punith M, Shalini M, Srushti Goudar, and Ruthesh S, “A Raspberry Pi based Smart Security Patrol Robot,” Feb. 2023, doi: https://doi.org/10.1109/iccmc56507.2023.10083908.

[2]“An Autonomous Surveillance Robot with IoT based Rescue System Enhancement ,” ResearchGate, Mar. 03, 2022.https://www.researchgate.net/publication/359134908_An_Autonomous_Surveillance_Robot_with_IoT_based_Rescue_System_Enhancement

[3]J.-H. Jean and J.-L. Wang, “Development of an indoor patrol robot based on ultrasonic and vision data fusion,” Aug. 2013, doi: https://doi.org/10.1109/icma.2013.6618090.

[4]L. Huang, M. Zhou, K. Hao, and E. Hou, “A survey of multi-robot regular and adversarial patrolling,” IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 4, pp. 894–903, Jul. 2019, doi:https://doi.org/10.1109/jas.2019.1911537.

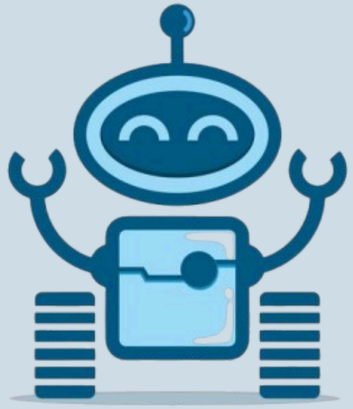
[5]Chun, W. H., & Papanikolopoulos, N. (2016). Robot surveillance and security. In Springer handbooks (pp. 1605–1626). https://doi.org/10.1007/978-3-319-32552-1_61



IT21242472 |Gunawardhana K.A.S.H.

BSc (Hons) in Information Technology
Computer Systems & Network Engineering

SLIIT UNI
THE KNOWLEDGE UNIVERSITY



Background

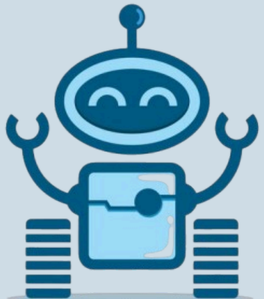
- Although autonomous robots are becoming more and more popular in the security, surveillance, and maintenance sectors, their outdoor monitoring and patrol duties demand dependable performance in a variety of weather situations.

- Particularly in harsh environments like snow, rain, fog, and fluctuating temperatures, outdoor robots struggle to adjust to quickly changing weather conditions, which lowers their performance and dependability.

- With the use of several sensors and a multi-sensor fusion algorithm, slave robots can anticipate changes in their surroundings and modify their physical systems in real time, improving performance in a variety of weather scenarios.

Research Question

- How can a sensor-driven monitoring system optimize the environmental adaptability, internal health, and power efficiency of autonomous patrol robots while ensuring real-time forecasting and early warning capabilities in dynamic environments?



Research Gap

- Limited research on using multi-sensor fusion specifically for outdoor patrol robots in dynamic environments.

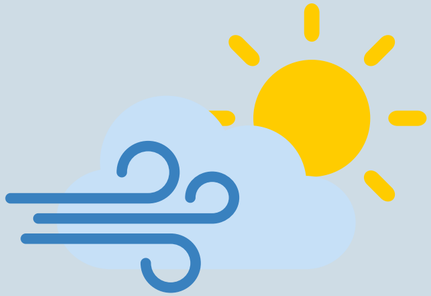
- Most existing studies focus on individual sensor types or indoor environments.

Comparison

	PROPOSED SYSTEM	RESEARCH PAPER 1	RESEARCH PAPER 2	RESEARCH PAPER 3	RESEARCH PAPER 4
MULTIPLE ONBOARD SENSORS (TEMPERATURE, HUMIDITY, ETC.)	✓	✓	✓	✓	✓
MULTI-SENSOR FUSION ALGORITHM	✓	✓	✗	✓	✗
INTELLIGENT BEHAVIOR ADAPTATION TO WEATHER CONDITIONS	✓	✗	✓	✓	✗
ADAPTIVE MECHANISMS FOR SNOW/RAIN (INCREASED TRACTION)	✓	✗	✓	✓	✗
FOG RESPONSE (NIGHT VISION ACTIVATION)	✓	✗	✗	✗	✗

Specific Obective

- A sensor-based monitoring system for slave robots, enabling adaptive weather Prediction, AQI prediction, fault detection, and traction control. With real-time IoT data transmission and an web dashboard, it ensures optimal performance, reliability, and autonomous decision-making.



Sub Objective

1.Weather Forecasting Model Development

2. Air Quality Prediction System

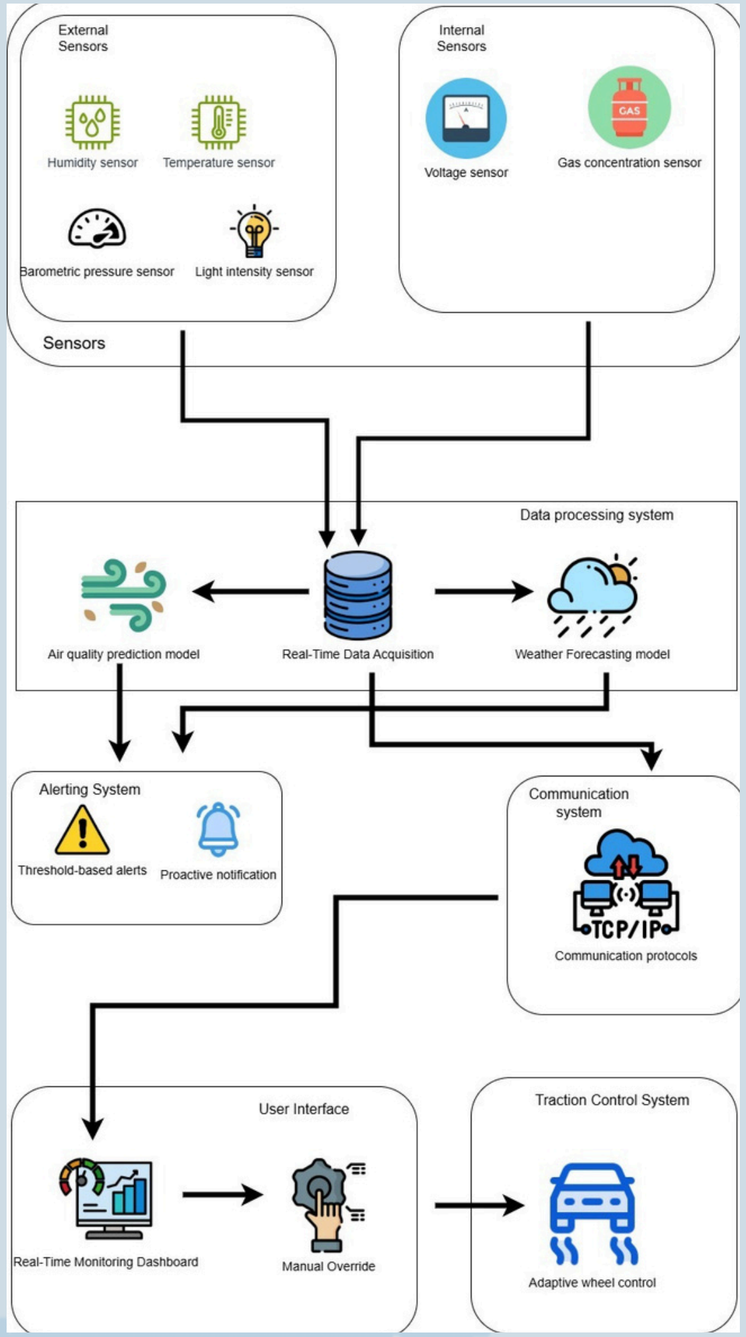
3. Internal Health Monitoring & Fault Prediction

4. Adaptive Traction Control Mechanism

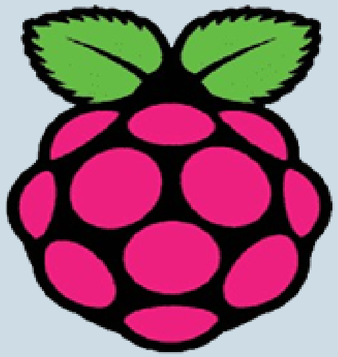
5. IoT-Based Data Collection & Transmission

6. Web-Based Dashboard for Real-Time Monitoring & Alerts

System Diagram



Technologies



RaspberryPi

Fuctonal & Non-Functional Requirement

Functional Requirements

- Monitor & Predict
- Health Tracking
- Traction Control
- IoT Communication
- Dashboard & Alerts

Non-Functional Requirements

- 1.Performance
- 2.Security
- 3.Usability
- 4.Data Storage

Data Sets For Weather ML Model

Total Samples: 2000

Class Distribution:

Cloudy: 800 samples

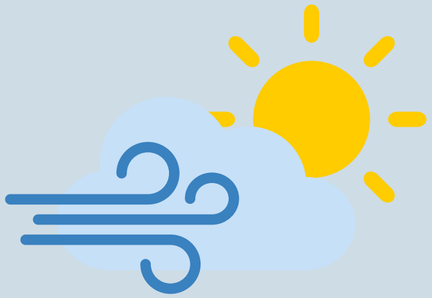
Rainy: 700 samples

Clear: 500 samples

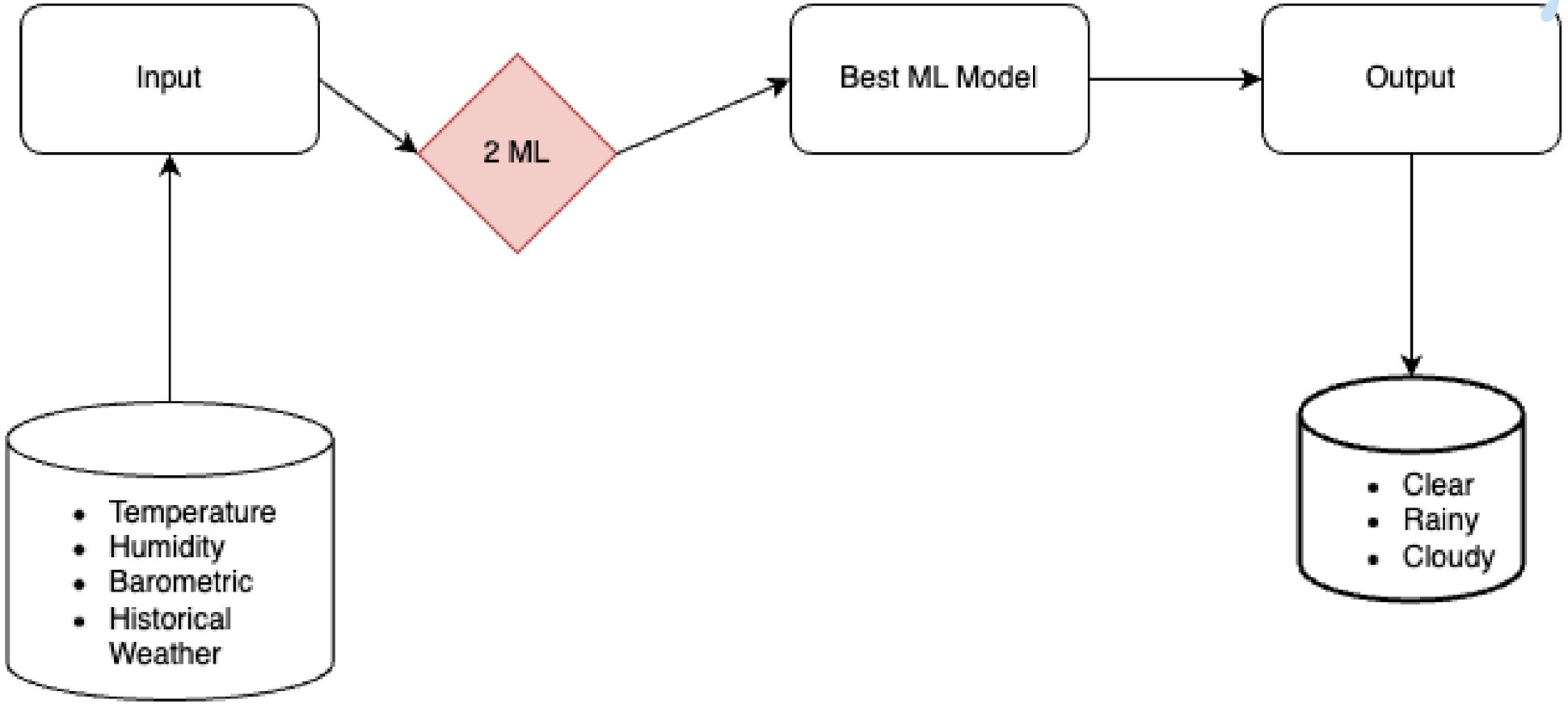
Temperature (°C)	Humidity (%)	Pressure (hPa)	Weather Class
33.74001847	10.306834	1018.330283	Clear
37.30694221	40.46225411	956.3113435	Clear
39.04397272	29.25619047	986.09049	Clear
32.02595452	49.78707916	989.3179659	Clear
31.87565234	19.37117568	950.1184946	Clear
35.45285081	28.28941141	969.9233203	Clear
37.4608354	28.92401645	1034.281172	Clear
37.48738646	18.78063141	1001.073456	Clear
33.11494221	12.30949044	988.7201168	Clear
38.87932403	30.81710681	1002.640621	Clear
33.72278833	16.37878554	961.1050453	Clear
34.5700718	23.9330872	1001.203577	Clear
30.64013375	30.3735549	1040.00369	Clear
38.66899248	47.34541907	1027.809783	Clear
34.06797505	39.1639405	1027.705441	Clear
38.77235028	34.73361461	997.0496192	Clear
34.94065258	24.74833681	1002.41589	Clear
39.49123656	10.21792453	957.6264994	Clear

	17.38277465	59.75274315	1046.472788	Cloudy
	17.16069506	65.17459562	1035.1756	Cloudy
	16.69972041	43.97615239	1002.291168	Cloudy
	22.85516418	53.88696376	1030.178067	Cloudy
	21.04796289	35.49785769	1000.312263	Cloudy
	24.53530507	31.70728456	1015.475798	Cloudy
	17.00167372	62.90960833	1046.010904	Cloudy
	29.36536257	66.98975201	1003.49078	Cloudy
	15.34120614	60.59598221	1046.196309	Cloudy
	10.30904946	68.0560594	1013.058718	Cloudy
	20.66862039	43.16470582	1027.843826	Cloudy
	13.40597134	67.25335579	1008.561323	Cloudy
	24.37980586	49.76066613	1005.184261	Cloudy
	29.6371174	55.884126	1011.258763	Cloudy
	17.84077812	37.33463329	1014.12262	Cloudy

530	20.17781111	96.144744	954.0180542	Rainy
531	16.56247056	80.31286996	986.5411835	Rainy
532	27.49518508	82.54322564	967.699516	Rainy
533	18.97501316	79.79644243	988.0728516	Rainy
534	16.19918339	78.09532424	977.0070499	Rainy
535	21.07196875	79.06726031	977.7158132	Rainy
536	25.50398361	71.67498672	969.7255927	Rainy
537	18.97258713	70.87114519	988.9309136	Rainy
538	22.54700929	98.77065961	968.7059482	Rainy
539	28.85270759	70.21632816	966.8174659	Rainy
540	28.70441246	87.44768727	994.4438532	Rainy
541	17.5234402	87.35376489	950.3498427	Rainy
542	18.2784015	89.54304159	966.674569	Rainy
543	22.31640117	99.69856337	970.4552427	Rainy
544	16.70799641	97.36196946	965.0490853	Rainy
545	26.87340561	90.1167895	963.9013974	Rainy
546	19.21962546	84.40037545	983.2745539	Rainy
547	15.66857081	91.258449	956.7683238	Rainy



Weather Forecasting



Demonstration

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

[1] ✓ 4.0s

Implements two machine learning models

- Random Forest
- Support Vector Machine

Why I Choose this ML Models

1. Random Forest Classifier

- An ensemble learning method using multiple decision trees
- Provides high accuracy and handles non-linear relationships well.

2. Support Vector Machine (SVM) Classifier

- A kernel-based classifier that finds the optimal decision boundary between classes.
- Uses an RBF kernel to capture complex relationships in data.

Data transformation and accuracy of the weather model

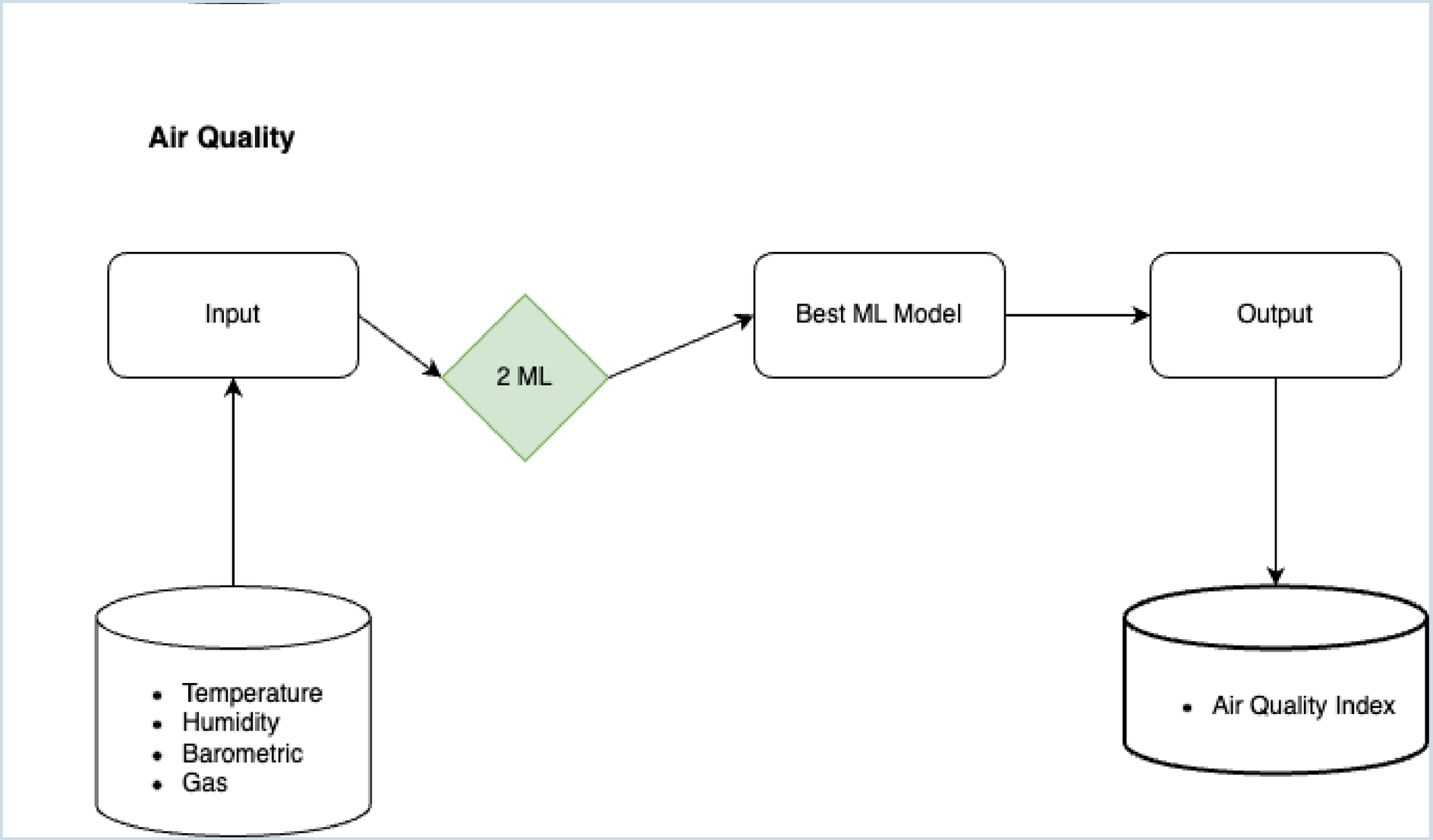
```
rf_accuracy
[26] ✓ 0.0s
... 1.0

rf_report
[27] ✓ 0.0s
... {'0': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 104.0},
 '1': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 129.0},
 '2': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 167.0},
 'accuracy': 1.0,
 'macro avg': {'precision': 1.0,
  'recall': 1.0,
  'f1-score': 1.0,
  'support': 400.0},
 'weighted avg': {'precision': 1.0,
  'recall': 1.0,
  'f1-score': 1.0,
  'support': 400.0}}
```

```
# Testing

sample_input = [[25, 65, 1010]]  # Example: 25°C, 65% humidity, 1010 hPa pressure
sample_scaled = scaler.transform(sample_input)

rf_prediction = rf_model.predict(sample_scaled)

print("Random Forest Prediction:", rf_prediction)
[22] ✓ 0.1s
... Random Forest Prediction: [2]
```

Data Sets For Air Quality ML Model

Total : 10000

Gas Concentration (ppm)	Temperature (°C)	Humidity (%)	Barometric Pressure (hPa)	AQI Value	Pollution Event	Fire Detection
187.270059	21.209225	78.399865	1013.814457	149	No	Yes
475.357153	19.987363	34.760960	995.929245	47	No	Yes
365.996971	15.284617	47.731176	1046.449852	63	No	No
299.329242	28.218000	73.062451	971.897845	90	Yes	Yes
78.009320	24.298725	58.567148	1008.785642	101	Yes	Yes


Air Quality
Input
2 ML
Best ML Model
Output
Temperature
Humidity
Barometric
Gas
Air Quality Index

Implement Air Quality ML Model

Hybrid Model Approach for Regression and Classification

```
[1]  ✓ 2.0s                                                                    Python

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor, XGBClassifier
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, accuracy_score, classification_report
from imblearn.over_sampling import SMOTE
```

```
[5]  ✓ 0.0s

X = df.drop(columns=["AQI Value", "Pollution Event", "Fire Detection"])
y_reg = df["AQI Value"]  # Regression target
y_cls_pollution = df["Pollution Event"]  # Classification target 1
y_cls_fire = df["Fire Detection"]  # Classification target 2
```

Why use Hybrid Model

A hybrid model approach was chosen to leverage the strengths of both regression and classification in handling different aspects of AQI prediction and pollution event detection.

XGBoost was selected due to its efficiency, robustness, and superior handling of structured data.

Using both regression and classification allows us to:

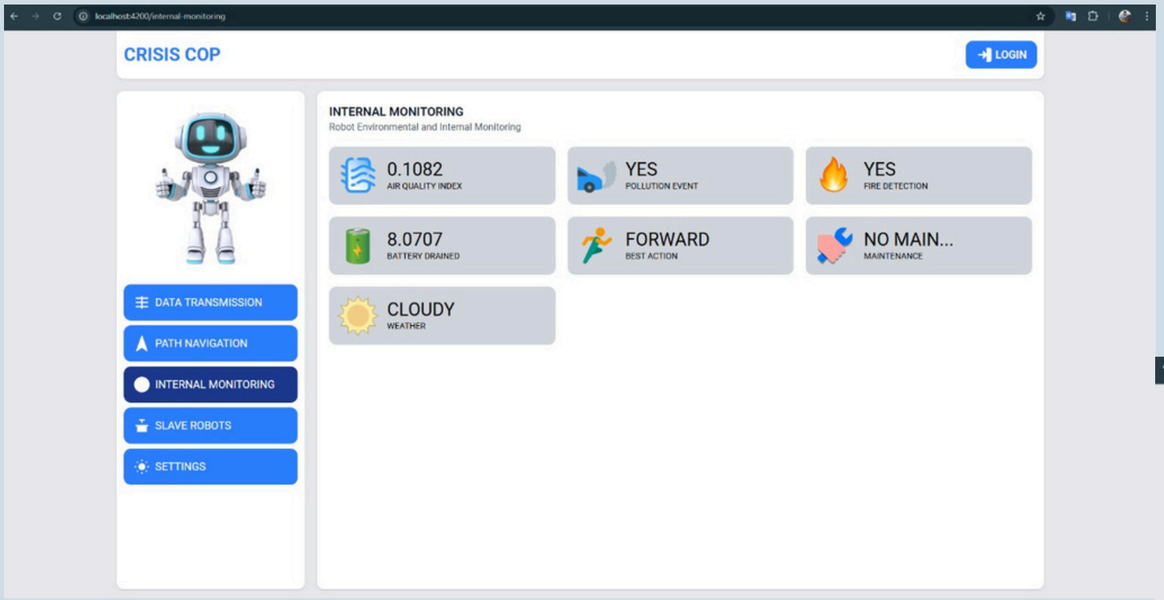
- Predict AQI values accurately while simultaneously detecting pollution events.
- Improve overall system performance by capturing both continuous and categorical patterns.
- Enhance interpretability and decision-making for environmental monitoring.

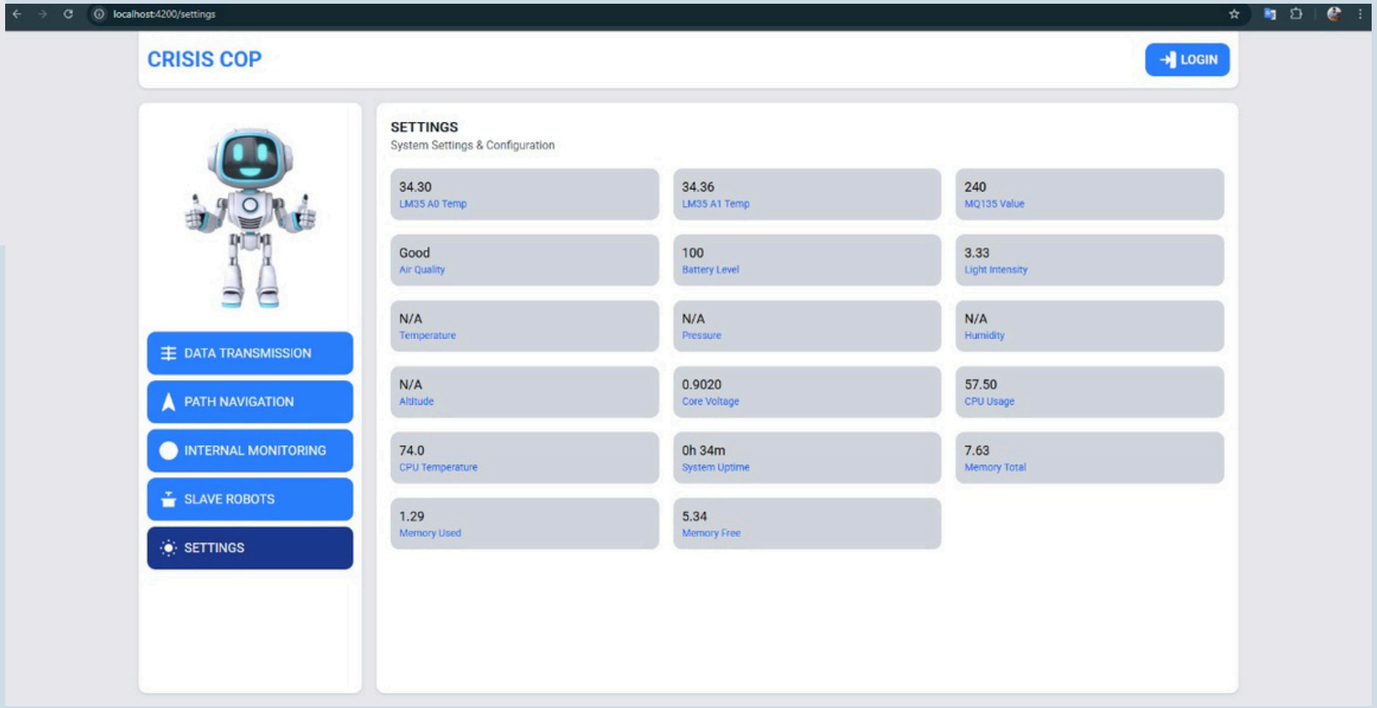
Data transformation and accuracy of the Air Quality model

```
Classification for Pollution Event:
Neural Network Accuracy: 0.503
              precision    recall  f1-score   support

           0       0.50      0.54      0.52      1001
           1       0.50      0.46      0.48       999

    accuracy                           0.50      2000
   macro avg       0.50      0.50      0.50      2000
weighted avg       0.50      0.50      0.50      2000

------------------------------------------------
Classification for Fire Detection:
Neural Network Accuracy: 0.5145
              precision    recall  f1-score   support

           0       0.52      0.37      0.43       997
           1       0.51      0.66      0.58      1003

    accuracy                           0.51      2000
   macro avg       0.52      0.51      0.50      2000
weighted avg       0.52      0.51      0.50      2000

------------------------------------------------
```

```
Neural Network Regression Evaluation:
MAE: 0.8990670939706 74
RMSE: 1.047464486209311
```

Web Application





Reference

[1]S. Joy, Richard Lincoln Paulraj, Punith M, Shalini M, Srushti Goudar, and Ruthesh S, “A Raspberry Pi based Smart Security Patrol Robot,” Feb. 2023, doi: https://doi.org/10.1109/iccmc56507.2023.10083908.

[2]“An Autonomous Surveillance Robot with IoT based Rescue System Enhancement ,” ResearchGate, Mar. 03, 2022.https://www.researchgate.net/publication/359134908_An_Autonomous_Surveillance_Robot_with_IoT_based_Rescue_System_Enhancement

[3]J.-H. Jean and J.-L. Wang, “Development of an indoor patrol robot based on ultrasonic and vision data fusion,” Aug. 2013, doi: https://doi.org/10.1109/icma.2013.6618090.

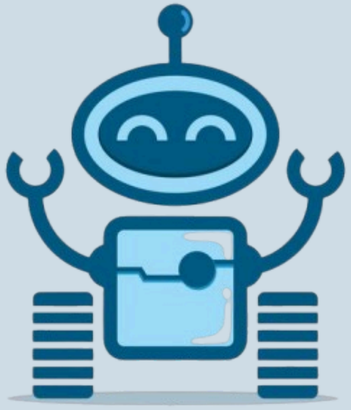
[4]L. Huang, M. Zhou, K. Hao, and E. Hou, “A survey of multi-robot regular and adversarial patrolling,” IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 4, pp. 894–903, Jul. 2019, doi:https://doi.org/10.1109/jas.2019.1911537.

[5]Chun, W. H., & Papanikolopoulos, N. (2016). Robot surveillance and security. In Springer handbooks (pp. 1605–1626). https://doi.org/10.1007/978-3-319-32552-1_61

IT21210860|D.L.B.S Liyanaarachchi

BSc (Hons) in Information Technology
Computer Systems & Network Engineering

SLIIT UNI
THE KNOWLEDGE UNIVERSITY



Background

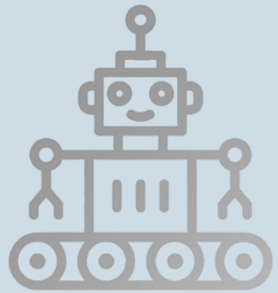
- Autonomous robot industry is getting more and more popular. And those robots are used in various subjects like mining , transportation , defense and security etc.
- Whatever the industry these robots need to work 24/7 and reliability is a must. A little bit of downtime will cause lot of trouble.
- When multiple robots work together there should be a way to communicate with each other in order to do the work efficiently.

Research Question

- What is the most effective algorithm and communication protocol for the slave robots to communicate with each other real time while reducing latency and saving the battery life ?

- How decision making should be implement in both of the robots to determine which robot should be send to the event location based on the robots battery life , distance to the incident location while reducing the response time ?

- How to design the autonomous navigation and path planning to determine the best and the nearest path ?



Research Gap

- Most of the existing solutions focused on predefined paths for the patrolling robots.
- Most of the existing systems need human interaction in one way or another.
- Limited researches using two patrol robots communicating with each other to do a task

Comparison

	Proposed System	Research paper 01	Research paper 02	Research paper 03
Intelligent path planning based on various factors like available battery life etc.	✓	✗	✗	✗
Use of inter-robot communication	✓	✗	✗	✗
Send alerts to user anywhere anytime	✓	✗	✗	✗
User interaction is not necessary	✓	✗	✗	✓
Fire , motion detection and alerts	✓	✓	✓	✗

Specific Obective

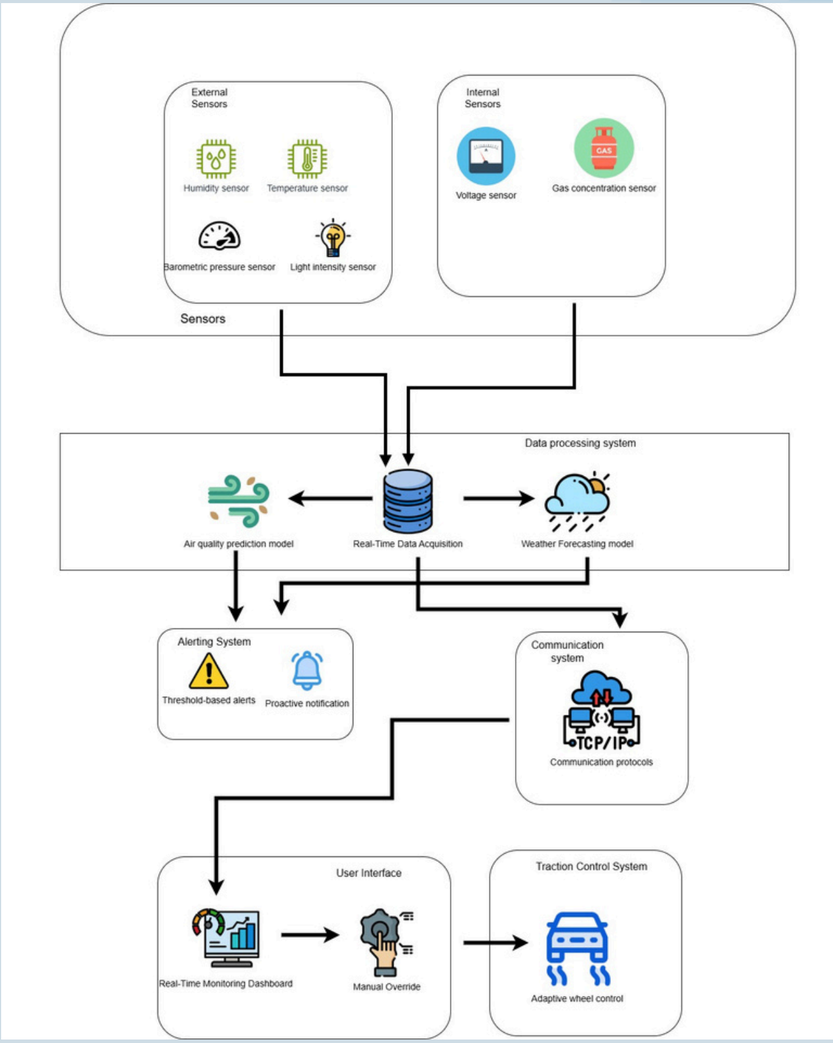
- Develop an algorithm for the robots that enables real time communication and coordination while reducing latency and response time. This algorithm is also capable of detect incidents autonomously and change patrol routes for additional coverage.

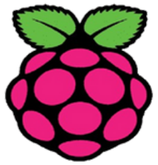
Sub Objective

- Equip robot with the best communication protocol which reduces latency , response time and saves battery life.

- Develop a suitable algorithm which enables autonomous navigation and intelligent path planning and select the best path available according to the terrain , distance and battery life of the robots.

System Diagram



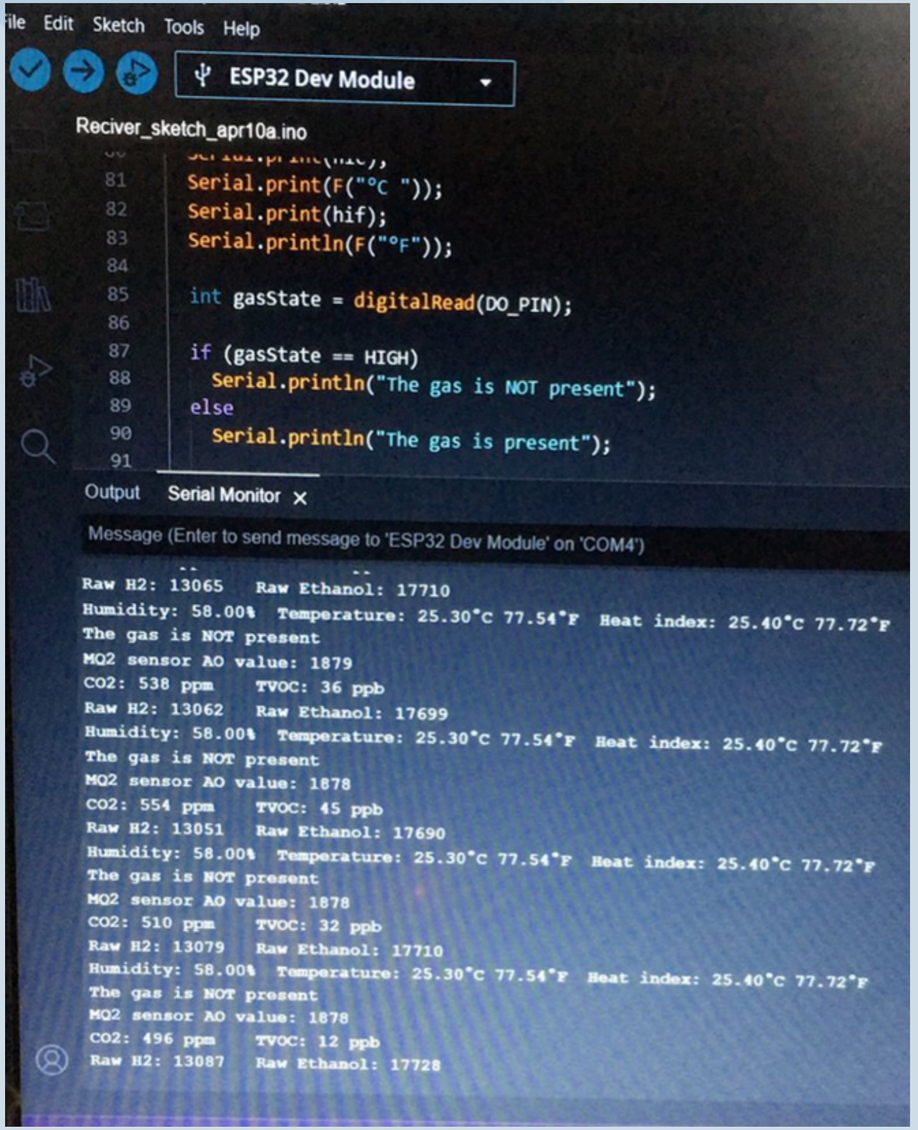


RaspberryPi

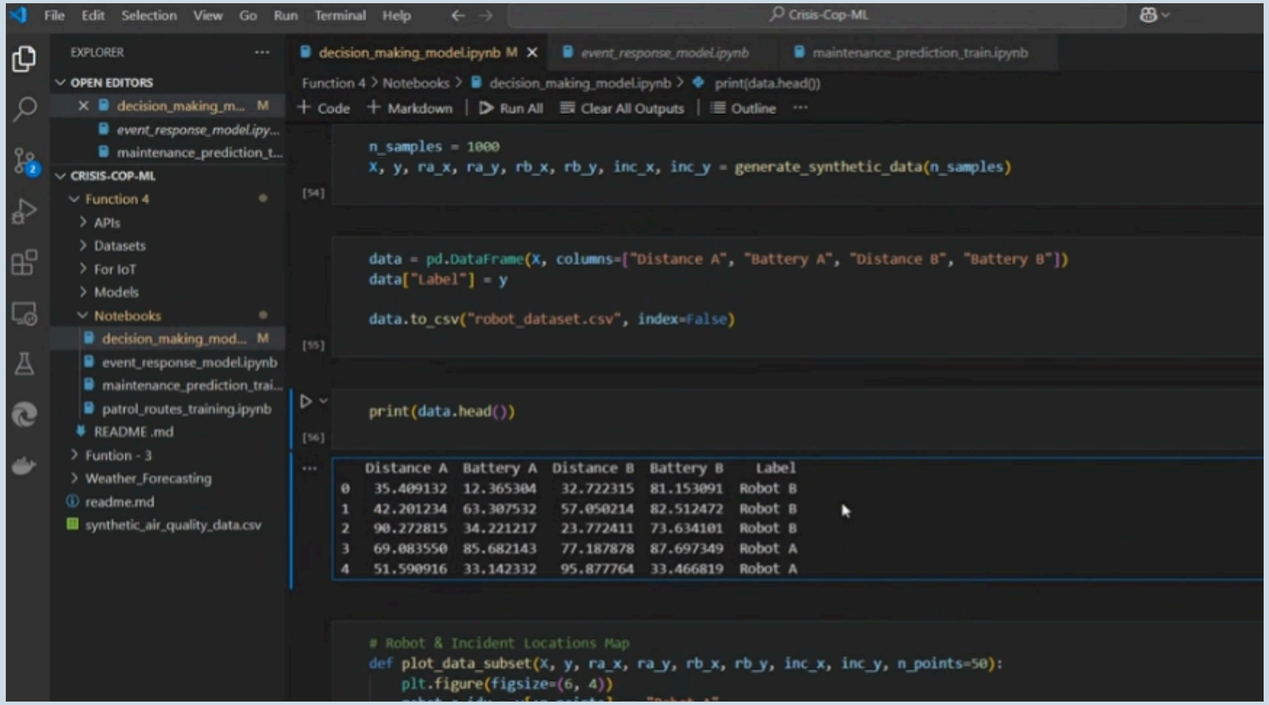
Completion up to PP1

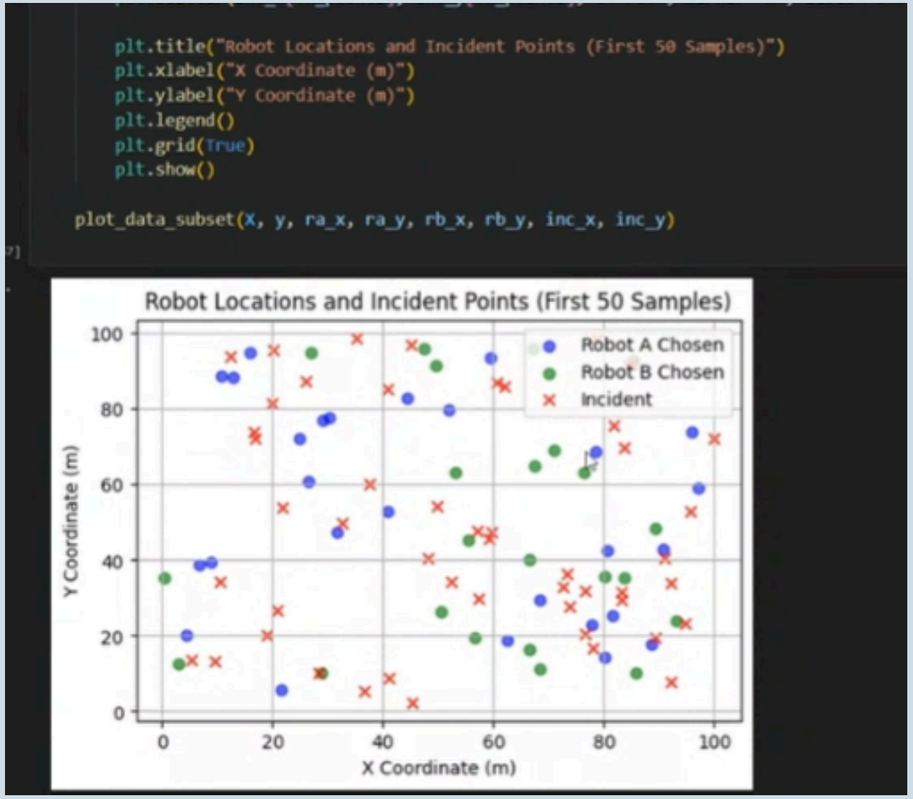
- ✓ Smoke sensor configuration and implementation.
- ✓ Pressure and temperature sensor configuration and implementation.
- ✓ Ultrasonic sensor configuration and implementation.

Initial sensor test



Decision making model





Event response model

```
data = pd.DataFrame(X, columns=[f"Distance {i}" for i in range(5)] + [f"Battery {i}" for i in range(5)])
data["Label"] = y
data.to_csv("robot_event_response_dataset.csv", index=False)
print(data.head())
```

```
   Distance 0  Distance 1  Distance 2  Distance 3  Distance 4  Battery 0  \
0   14.791629   12.138308   67.245833   48.659264   50.127652  17.981504
1   74.315726   33.150102   21.602382   84.843502   39.723920  62.566358
2   67.850761   14.480191   72.080485   57.670929   96.281375  55.807725
3   10.417894   45.383302   12.022846   69.229991   19.252317  16.193335
4   27.174979   40.630194   60.103296   14.943526  100.074066  37.830688

   Battery 1  Battery 2  Battery 3  Battery 4    Label
0  57.967100  25.085113  90.485004  18.080102  Robot A
1  72.338975  86.759067  21.904727  20.562812  Robot B
2  24.300726  37.526264  32.929476  39.745661  Robot D
3  36.340110  29.803872  66.247504  20.248289  Robot B
4  89.232719  42.623956  30.810660  32.525174  Robot A
```

```
    print(f"Model Accuracy: {accuracy:.2f}")
    print("\nClassification Report:")
    print(classification_report(y_test_str, y_pred, target_names=["Robot A", "Robot B", "R
```

```
Model Accuracy: 0.93

Classification Report:
              precision    recall  f1-score   support

     Robot A       0.96      0.92      0.94       228
     Robot B       0.93      0.91      0.92       188
     Robot C       0.92      0.93      0.93       183
     Robot D       0.91      0.98      0.94       209
     Robot E       0.93      0.93      0.93       192

    accuracy                           0.93      1000
   macro avg       0.93      0.93      0.93      1000
weighted avg       0.93      0.93      0.93      1000
```

Model accuracy

Data sets

Robot_X	Robot_Y	Battery_Level	Distance_Traveled	Time_Elapsed	Obstacle	Incident_X	Incident_Y	Direction	Obstacle_	Action
3	4	100	0	0	0	6	4	0	1	Left
3	4	98	0	1	0	6	4	3	1	Recalculate
3	4	96	0	2	0	6	4	3	1	Forward
3	3	94	1	3	1	6	4	3	0	Right
3	3	92	1	4	1	6	4	0	0	Left
3	3	90	1	5	1	6	4	3	0	Forward
3	2	88	2	6	0	6	4	3	1	Recalculate
3	2	86	2	7	0	6	4	3	1	Forward
3	1	84	3	8	0	6	4	3	1	Left
3	1	82	3	9	0	6	4	2	1	Right
3	1	80	3	10	0	6	4	3	1	Forward
3	0	78	4	11	0	6	4	3	1	Recalculate
3	0	76	4	12	0	6	4	3	1	Forward
3	0	74	4	13	0	6	4	3	1	Forward
3	0	72	4	14	0	6	4	3	1	Right
3	0	70	4	15	0	6	4	0	1	Forward
2	0	68	5	16	1	6	4	0	0	Recalculate
2	0	66	5	17	1	6	4	0	0	Forward
1	0	64	6	18	0	6	4	0	1	Forward

Patrol datasets

Robot_ID	Event_ID	Timestamp	Motion_Detected	Battery_Level	Robot_He	Obstacle_	Temperat	Humidity	Robot_Response
4	1	##########	0	96.53746072	Normal	0	30.15479	50.36537	Reroute
3	2	##########	0	89.04447813	Normal	0	27.46117	66.34216	Reroute
2	3	##########	0	96.92072429	Normal	0	22.64916	69.70257	Reroute
5	4	##########	0	96.24040707	Normal	0	24.94375	51.58577	Reroute
2	5	##########	0	95.96898312	Normal	0	26.53951	38.3035	Reroute
2	6	##########	0	79.46301205	Normal	0	26.20189	89.83888	Reroute
2	7	##########	0	94.40648757	Normal	0	24.67167	76.40197	Reroute
2	8	##########	1	98.28874932	Normal	0	24.51371	-14.8122	Investigate
4	9	##########	1	93.38156672	Normal	1	22.37376	37.00299	Reroute
1	10	##########	0	84.58264791	Normal	0	24.97751	51.31579	Reroute
3	11	##########	0	98.31101693	Normal	0	23.34807	73.90841	Reroute
1	12	##########	0	97.117276	Normal	0	19.10515	20.2354	Reroute
1	13	##########	0	94.06835316	Normal	0	28.67476	34.36333	Reroute
5	14	##########	0	69.46005617	Normal	0	26.1939	55.23175	Reroute
1	15	##########	0	95.61688181	Normal	1	21.80717	63.21873	Reroute
1	16	##########	0	93.85463439	Normal	1	23.97599	46.33344	Reroute
4	17	##########	0	98.98232261	Normal	0	25.37952	66.08637	Reroute

Event response model data set

Future implementations

- Implementation of the second robot
- GPS sensor integration
- Fine tune of models
- Provide constant 5V for mq135 sensor

Gantt Chart

Milestone	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun
Research about topic and tasks	■											
Group discussion	■											
Technology analysis	■	■										
Resource evaluation		■										
Topic assessment evaluation		■										
Hardware design and specifications		■	■									
Project implementation phase 1		■	■	■	■							
Integration testing				■								
Progress presentation1					■							
Project implementation phase 2					■	■	■	■	■			
Progress presentation2									■			
Testing and evaluation										■	■	■
Final report and conclusion												■

Completion up to PP2

- ✓ Main robot hardware implementation
- ✓ Decision making model implementation.
- ✓ Ultrasonic sensor configuration and implementation.

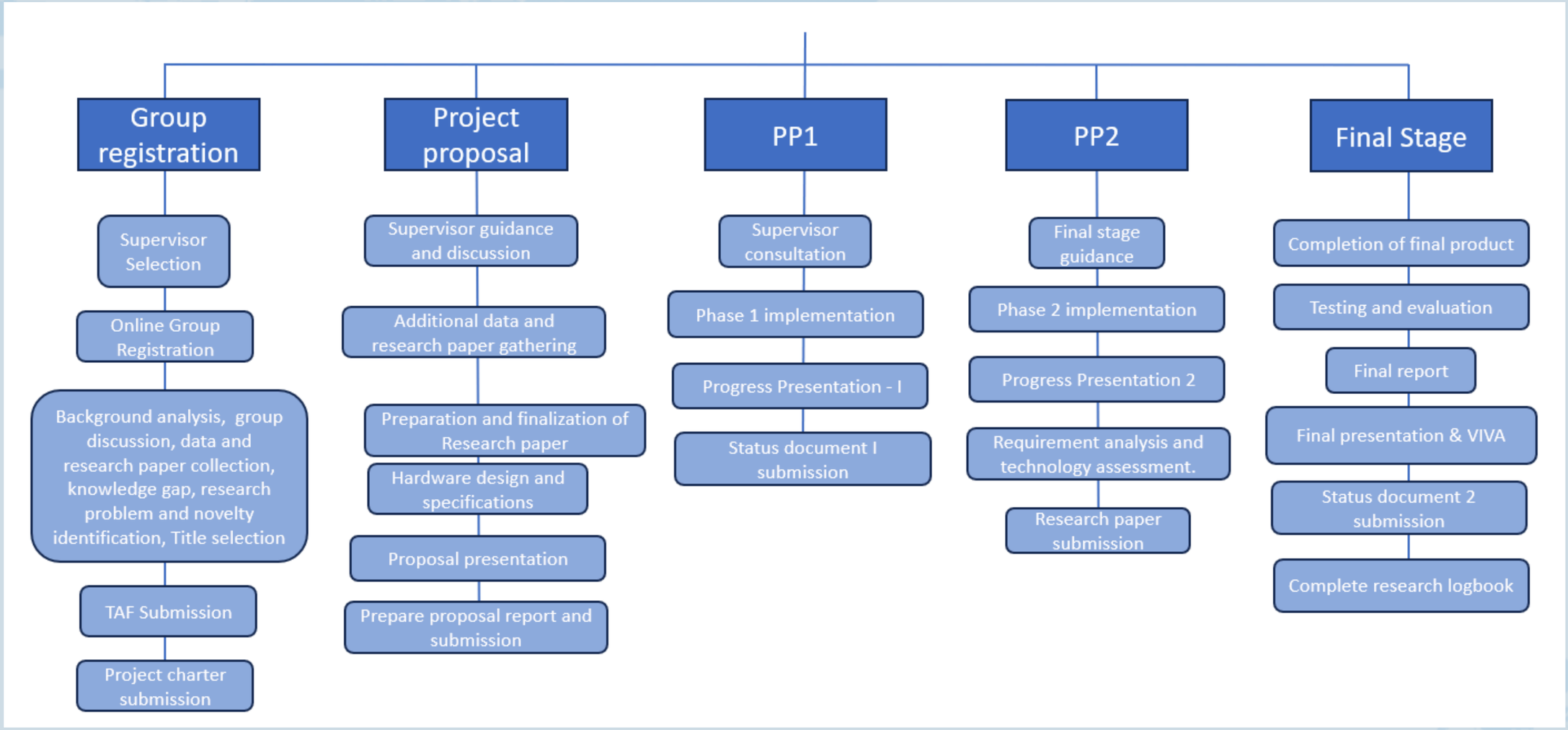
Reference

[1] Prof. M. K. Tiwari, 2Pooja Sahebrao Suryawanshi, 3Kulasum bi Kalim Sheikh, 4Shankar Bhagwan Teli, 5Jayshari Raju Bairagi | Patrolling Robot | DOI : https://doi.org/10.62226/ijarst20241385

[2]Suniksha B S1 ,Shubhanchal Priya2 , Varshini M3 , Deepthi Raj4 “Warfield Spy Robot with Night Vision Wireless Camera”, International Journals of Latest Technology in Engineering, Managements& Applied Sciences (IJLTEMAS) Volume IX, Issues VII, July 2020

[3]Department of Ece, K Stella. (2023). NIGHT PATROLLING ROBOT. European Chemical Bulletin. 12. 3164-3169. 10.31838/ecb/2023.12.s1 B.318.

Work breakdown structure



Commercialization

Target Group

- Logistics and Supply Chain Companies
- E-commerce and Retail Companies
- Manufacturing Industries
- Third-Party Logistics Providers (3PLs)
- Food and Beverage Companies

CRISIS COP

SLIIT UNI
THE KNOWLEDGE UNIVERSITY

Thank You!