Project Report: IoT Robot for Parcel Delivery

Summary

Background:

The Internet of Things (IoT) has revolutionized the way we interact with our environment by enabling devices to collect and exchange data. This project aims to develop an IoT-based robot for parcel delivery within a controlled environment, such as an office or warehouse. The robot is designed to navigate autonomously, detect obstacles, scan QR codes for parcel identification, and store relevant data in a central database.

Proposed System:

The proposed system comprises an Arduino-based robot equipped with ultrasonic sensors for obstacle detection and startup functionality, a QR code scanner for parcel identification, and DC motors for movement. The data collected by the robot is sent to a Raspberry Pi, which acts as an edge device, storing the data in a MariaDB database. A Flask-based web dashboard is used to monitor and analyse the collected data, providing real-time insights and statistics.

Conceptual Design

Block Diagrams:

The hardware and software components of the IoT architecture are illustrated in the block diagrams below.

Hardware Block Diagram:

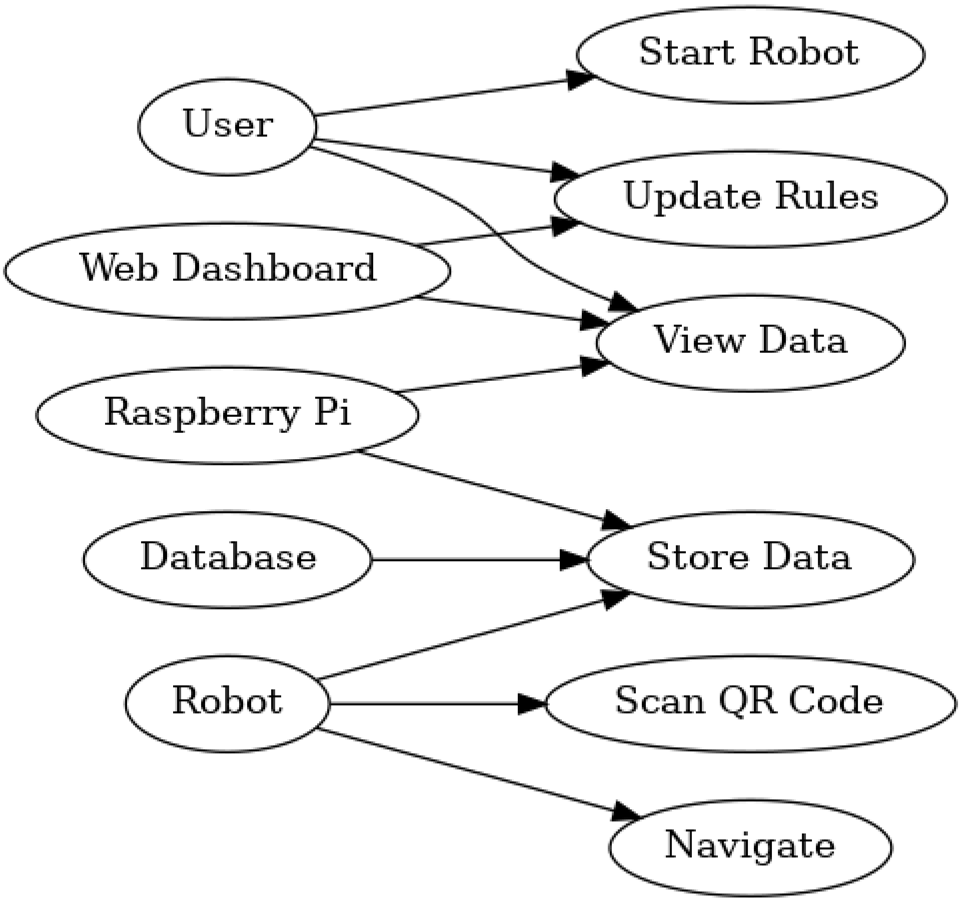
- Ultrasonic Sensors: startup robot.
- PIR Sensors: Measure distance for obstacle detection.
- DC Motors and Servo: Control robot movement.
- Arduino UNO: Central controller for sensors and actuators.
- Raspberry Pi: Edge device for data storage and processing.
- MariaDB: Database for storing sensor data.
- Flask Application: Web interface for data visualization.

Software Block Diagram:

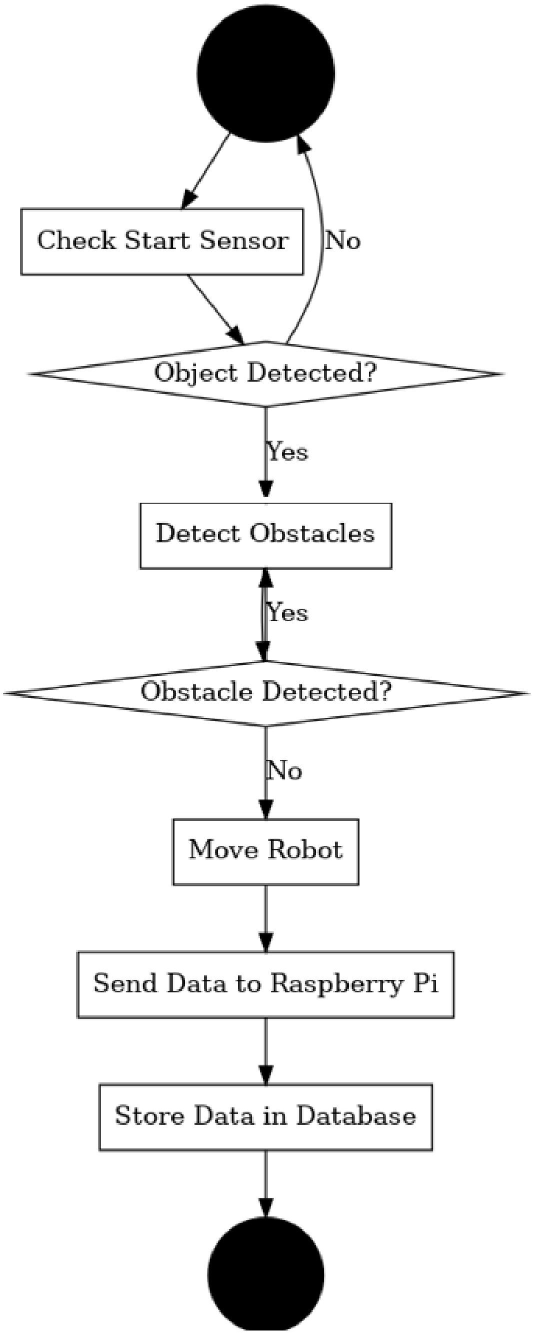
- Arduino Sketch: Code for controlling sensors and actuators.
- Python Scripts: Handle data transfer from Arduino to Raspberry Pi.
- Flask Application: Web server for dashboard.

UML Diagrams:

Use Case Diagram:



Activity Diagram:



#Implementation

Sensors:

1. Ultrasonic Sensor:

Purpose: initiate robot startup.

Integration: Connected to Arduino UNO. triggers actions based on predefined thresholds.

2. QR Code Scanner:

Purpose: Measure distance to detect obstacles

Integration: Reads distance values and triggers actions based on predefined thresholds.

Actuators:

1. DC Motors:

   Purpose: Move the robot forward, backward, and turn.Integration: Controlled by Arduino UNO via motor driver.

   Commands for movement are based on sensor inputs.

2. Servo Motor:

   Purpose: Control the direction of the front wheel for turning.

   Integration: Connected to Arduino UNO. Adjusts the wheel angle based on obstacle detection.

Software/Libraries:

Arduino Libraries:

- \`Servo.h\`: Control the servo motor.
- \`NewPing.h\`: Interface with ultrasonic sensors.

Python Libraries:

- \`serial\`: Handle serial communication between Arduino and Raspberry Pi.
- \`mysql.connector\`: Interface with MariaDB for data storage.
- \`flask\`: Create the web dashboard.

Resources:

- Arduino Documentation: https://www.arduino.cc/en/Reference/HomePage

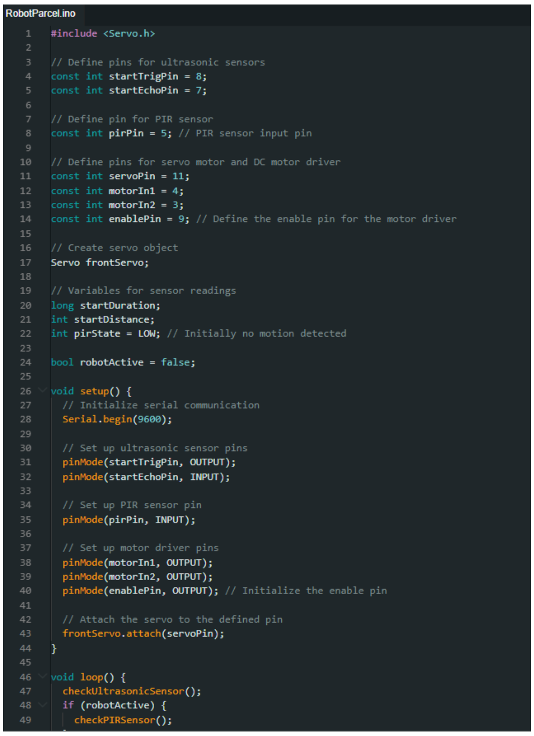
- Flask Documentation: https://flask.palletsprojects.com/en/2.0.x/

- MariaDB Documentation: https://mariadb.com/kb/en/documentation/

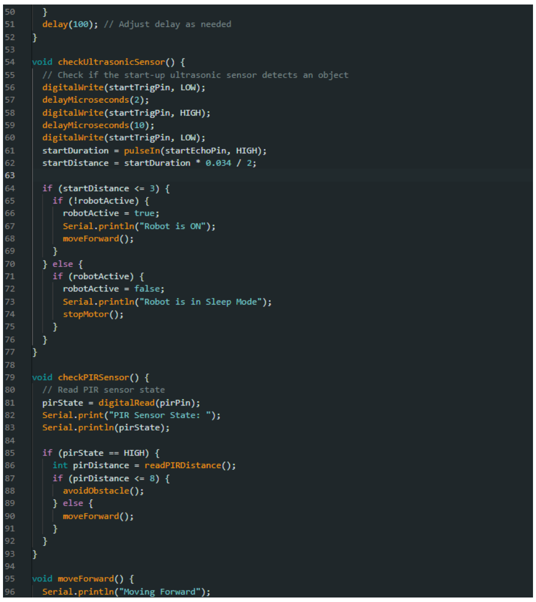
- Various online tutorials and guides for Arduino, Flask, and Raspberry Pi integration.

#Appendix

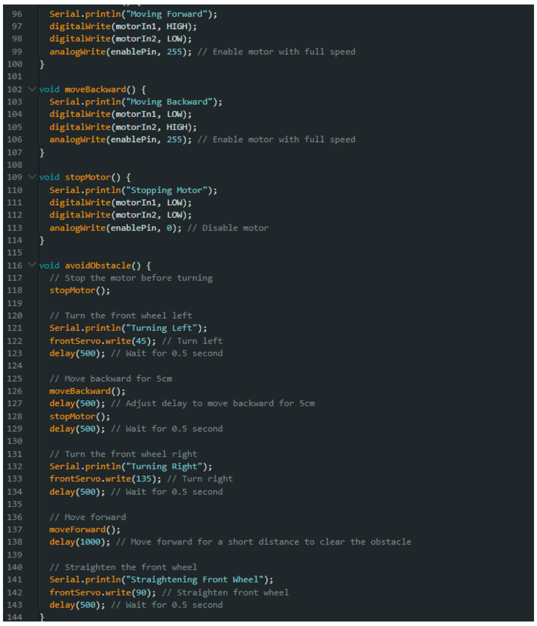
Arduino Sketch:



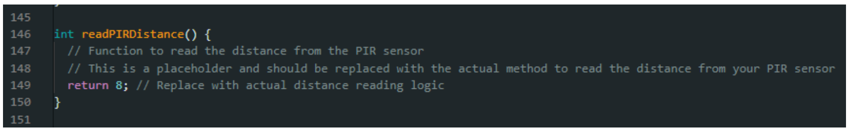
```
#include <Servo.h>

// Define pins for ultrasonic sensors
const int startTrigPin = 8;
const int startEchoPin = 7;

// Define pin for PIR sensor
const int pirPin = 5; // PIR sensor input pin

// Define pins for servo motor and DC motor driver
const int servoPin = 11;
const int motorIn1 = 4;
const int motorIn2 = 3;
const int enablePin = 9; // Define the enable pin for the motor driver

// Create servo object
Servo frontServo;

// Variables for sensor readings
long startDuration;
int startDistance;
int pirState = LOW; // Initially no motion detected

bool robotActive = false;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Set up ultrasonic sensor pins
  pinMode(startTrigPin, OUTPUT);
  pinMode(startEchoPin, INPUT);

  // Set up PIR sensor pin
  pinMode(pirPin, INPUT);

  // Set up motor driver pins
  pinMode(motorIn1, OUTPUT);
  pinMode(motorIn2, OUTPUT);
  pinMode(enablePin, OUTPUT); // Initialize the enable pin

  // Attach the servo to the defined pin
  frontServo.attach(servoPin);
}

void loop() {
  checkUltrasonicSensor();
  if (robotActive) {
    checkPIRSensor();
```



```
  }
  delay(100); // Adjust delay as needed
}

void checkUltrasonicSensor() {
  // Check if the start-up ultrasonic sensor detects an object
  digitalWrite(startTrigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(startTrigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(startTrigPin, LOW);
  startDuration = pulseIn(startEchoPin, HIGH);
  startDistance = startDuration * 0.034 / 2;

  if (startDistance <= 3) {
    if (!robotActive) {
      robotActive = true;
      Serial.println("Robot is ON");
      moveForward();
    }
  } else {
    if (robotActive) {
      robotActive = false;
      Serial.println("Robot is in Sleep Mode");
      stopMotor();
    }
  }
}

void checkPIRSensor() {
  // Read PIR sensor state
  pirState = digitalRead(pirPin);
  Serial.print("PIR Sensor State: ");
  Serial.println(pirState);

  if (pirState == HIGH) {
    int pirDistance = readPIRDistance();
    if (pirDistance <= 8) {
      avoidObstacle();
    } else {
      moveForward();
    }
  }
}

void moveForward() {
  Serial.println("Moving Forward");
```

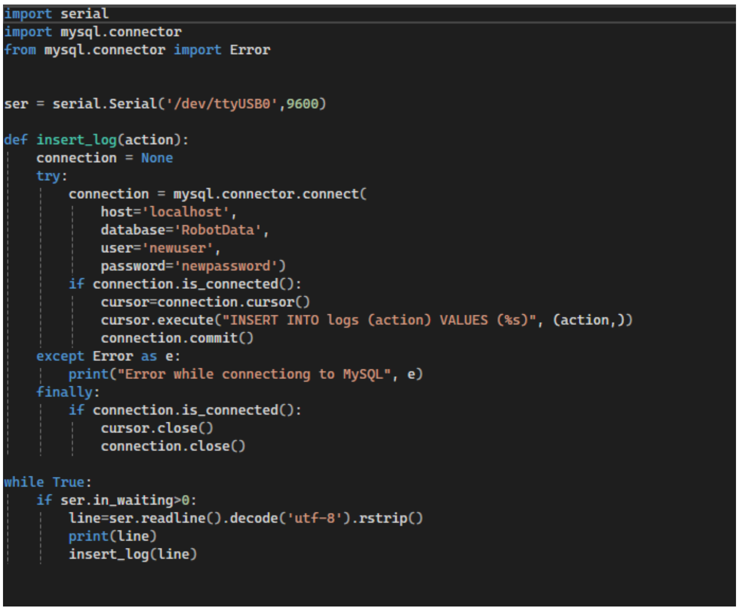


```cpp
  Serial.println("Moving Forward");
  digitalWrite(motorIn1, HIGH);
  digitalWrite(motorIn2, LOW);
  analogWrite(enablePin, 255); // Enable motor with full speed
}

void moveBackward() {
  Serial.println("Moving Backward");
  digitalWrite(motorIn1, LOW);
  digitalWrite(motorIn2, HIGH);
  analogWrite(enablePin, 255); // Enable motor with full speed
}

void stopMotor() {
  Serial.println("Stopping Motor");
  digitalWrite(motorIn1, LOW);
  digitalWrite(motorIn2, LOW);
  analogWrite(enablePin, 0); // Disable motor
}

void avoidObstacle() {
  // Stop the motor before turning
  stopMotor();

  // Turn the front wheel left
  Serial.println("Turning Left");
  frontServo.write(45); // Turn left
  delay(500); // Wait for 0.5 second

  // Move backward for 5cm
  moveBackward();
  delay(500); // Adjust delay to move backward for 5cm
  stopMotor();
  delay(500); // Wait for 0.5 second

  // Turn the front wheel right
  Serial.println("Turning Right");
  frontServo.write(135); // Turn right
  delay(500); // Wait for 0.5 second

  // Move forward
  moveForward();
  delay(1000); // Move forward for a short distance to clear the obstacle

  // Straighten the front wheel
  Serial.println("Straightening Front Wheel");
  frontServo.write(90); // Straighten front wheel
  delay(500); // Wait for 0.5 second
}
```



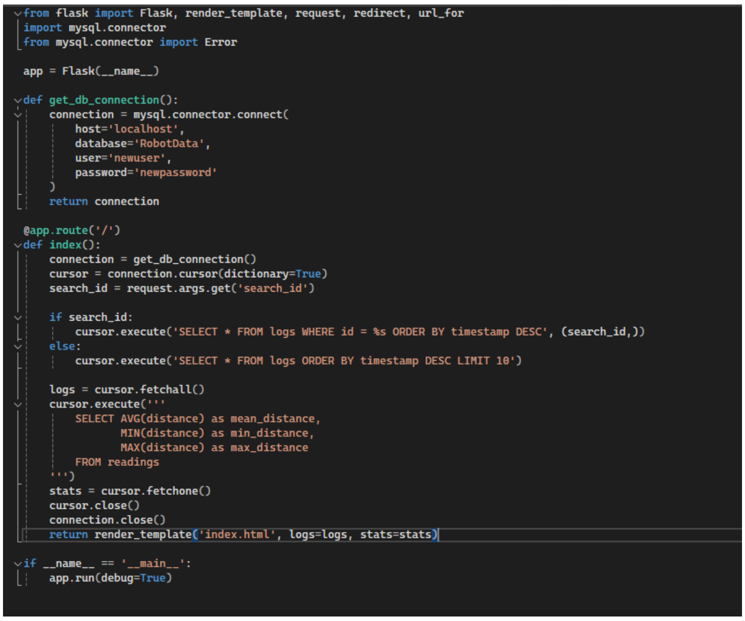
```cpp

int readPIRDistance() {
  // Function to read the distance from the PIR sensor
  // This is a placeholder and should be replaced with the actual method to read the distance from your PIR sensor
  return 8; // Replace with actual distance reading logic
}

```

Python Script (store_robot_data.py):



```python
import serial
import mysql.connector
from mysql.connector import Error


ser = serial.Serial('/dev/ttyUSB0',9600)

def insert_log(action):
    connection = None
    try:
        connection = mysql.connector.connect(
            host='localhost',
            database='RobotData',
            user='newuser',
            password='newpassword')
        if connection.is_connected():
            cursor=connection.cursor()
            cursor.execute("INSERT INTO logs (action) VALUES (%s)", (action,))
            connection.commit()
    except Error as e:
        print("Error while connectiong to MySQL", e)
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()

while True:
    if ser.in_waiting>0:
        line=ser.readline().decode('utf-8').rstrip()
        print(line)
        insert_log(line)
```

Flask Application (app.py):



```python
from flask import Flask, render_template, request, redirect, url_for
import mysql.connector
from mysql.connector import Error

app = Flask(__name__)

def get_db_connection():
    connection = mysql.connector.connect(
        host='localhost',
        database='RobotData',
        user='newuser',
        password='newpassword'
    )
    return connection

@app.route('/')
def index():
    connection = get_db_connection()
    cursor = connection.cursor(dictionary=True)
    search_id = request.args.get('search_id')

    if search_id:
        cursor.execute('SELECT * FROM logs WHERE id = %s ORDER BY timestamp DESC', (search_id,))
    else:
        cursor.execute('SELECT * FROM logs ORDER BY timestamp DESC LIMIT 10')

    logs = cursor.fetchall()
    cursor.execute('''
        SELECT AVG(distance) as mean_distance,
               MIN(distance) as min_distance,
               MAX(distance) as max_distance
        FROM readings
    ''')
    stats = cursor.fetchone()
    cursor.close()
    connection.close()
    return render_template('index.html', logs=logs, stats=stats)

if __name__ == '__main__':
    app.run(debug=True)
```

Resources

Arduino Documentation: [Arduino Reference](https://www.arduino.cc/en/Reference/HomePage)

Flask Documentation: [Flask](https://flask.palletsprojects.com/en/2.0.x/)

MariaDB Documentation: [MariaDB Knowledge Base](https://mariadb.com/kb/en/documentation/)

Online Tutorials: Various guides and tutorials on integrating Arduino with sensors, Python serial communication, and building Flask web applications.