IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429

Volume 5, Issue 7, June 2025

Impact Factor: 7.67

Automatic Pothole Repair System

Mrs. Mamatha Poreddy, Gajula Mahesh Kumar, Niharika Ceekonda
Ganji Ganesh, Ella Sai Manikanta
ACE Engineering College, Hyderabad, India

Abstract: *The "Automatic Pothole Repair System" presents a smart and efficient solution to road damage maintenance using IoT and embedded systems. This project automates the identification and repair of potholes using ESP32 microcontroller, GPS and ultrasonic sensors, and robotic arms for repair, significantly reducing manual intervention. The robot identifies pothole locations, calculates dimensions, sends data to a cloud server, and performs on-spot repair with concrete mix. The system integrates a web interface for live monitoring and uses Python Anywhere for cloud hosting. This approach enables real-time analytics, low-cost repair, improved road safety, and seamless automation in road infrastructure management.*

Keywords: *Python*

I. INTRODUCTION

Potholes are a recurring problem in road networks leading to accidents, traffic jams, and increased maintenance costs. Traditional repair techniques are slow and labor-intensive. This paper presents an automated system that combines robotics and IoT to detect, measure, and repair potholes. With components like ESP32, GPS, and ultrasonic sensors, the robot identifies potholes, calculates their volume, and fills them with appropriate material. It also updates data on a cloud dashboard accessible to administrators.

II. OBJECTIVES

- Detect potholes autonomously using ultrasonic sensors.
- Estimate pothole volume for precise repair.
- Fill potholes using cement mixtures via servo-controlled dispensers.
- Send repair data to a cloud server.
- Enable real-time monitoring through a web dashboard.
- Ensure road safety by deploying temporary traffic cones.

III. PROBLEM STATEMENT

Manual pothole repairs are inefficient, inconsistent, and risky. They lack data analytics, delay maintenance, and pose safety hazards to workers and drivers. Existing automated systems focus mostly on detection, not repair. This project addresses these issues by offering a complete end-to-end automated detection and repair system with data tracking and cloud connectivity.

IV. PROPOSED SYSTEM

The robot subsystem includes ESP32, ultrasonic and GPS sensors, servo motors, cement dispensers, and a wireless communication unit. Detected pothole coordinates and volume are transmitted to the cloud (PythonAnywhere). Once acknowledged, the robot initiates repair. The user can track status via a web browser that retrieves data from the server.

V. HARDWARE AND SOFTWARE REQUIREMENTS

Hardware:

- ESP32 microcontroller

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

- Ultrasonic sensors (HC-SR04)
- GPS module (NEO-6M)
- DC motors, Servo motors
- Motor driver (L298N)
- Cement mixer motor
- Battery pack

Software:

- Arduino IDE
- Python (Flask for cloud server)
- HTML/CSS for dashboard
- PythonAnywhere (cloud hosting)

VI. TECHNOLOGY DESCRIPTION

ESP32 is a low-cost microcontroller with built-in Wi-Fi and Bluetooth, ideal for IoT and automation tasks. It handles sensor data, logic control, and server communication. The web interface uses Flask hosted on PythonAnywhere to manage and visualize pothole data.

VII. PACKAGES USED

- Flask (Python web framework)
- Chart.js (for visualization in web dashboard)
- Arduino libraries (for sensors and servos)

VIII. ALGORITHM

Start the ESP32 system.
Measure road depth using ultrasonic sensors.
If pothole detected:
Stop the robot.
Record GPS location.
Calculate volume.
Mix and dispense repair material.
Send data to server.
Receive acknowledgment.
Resume operation.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <HTTPClient.h>
#include <TinyGPS++.h>
#include <ESP32Servo.h>
#include <math.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <freertos/queue.h>

const char* ssid = "Unknown";
const char* password = "@@##$$";
```


DOI: 10.48175/IJARSCT-28008

```
const char* serverName = "ganeshganji.pythonanywhere.com";
const String dataEndpoint = "/receive_data";
const String statusEndpoint = "/update_status";

const int trigPinFront = 23;
const int echoPinFront = 22;
const int trigPinBottom = 21;
const int echoPinBottom = 19;

const int motor1_IN1 = 25;
const int motor1_IN2 = 26;
const int motor2_IN3 = 27;
const int motor2_IN4 = 14;

Servo cementServo;
Servo waterServo;
Servo mixerServo;
Servo releaseServo;

const int cementServoPin = 18;
const int waterServoPin = 5;
const int mixerServoPin = 17;
const int releaseServoPin = 16;

const int SERVO_CLOSED_ANGLE = 20;
const int SERVO_OPEN_ANGLE = 60;
const int MIXER_START_ANGLE = 0;
const int MIXER_END_ANGLE = 180;

const int waterLevelPin = 4;

HardwareSerial gpsSerial(1);
TinyGPSPlus gps;
const int GPS_RX_PIN = 32;
const int GPS_TX_PIN = 33;

const float POTHOLE_THRESHOLD_DISTANCE_CM = 5.0;
const float OBSTACLE_DETECTION_DISTANCE_CM = 20.0;
const float VEHICLE_SPEED_CM_PER_SEC = 30.0;
const unsigned long INITIAL_BOOT_DELAY_MS = 5000;
const unsigned long SERVO_ACTION_DURATION_PER_CM3_MS = 100
```

Link for Code

https://github.com/ganesh949152/AutomaticPotholeRepairSystem

Web Server Code

```
from flask import Flask, render_template, request, jsonify
import json
from datetime import datetime
import os
```

```
import logging
import traceback

app = Flask(_name_)
DATA_FILE = "/home/ganeshganji/APRS/repair_data.json"
logging.basicConfig(level=logging.ERROR)
logger = logging.getLogger(_name_)

def load_data():
    """Load data from the JSON file."""
    try:
        logger.debug(f"Attempting to open {DATA_FILE} for reading...")
        with open(DATA_FILE, 'r') as f:
            logger.debug(f"Successfully opened {DATA_FILE} for reading.")
            try:
                data = json.load(f)
                logger.debug(f"Successfully loaded JSON data: {data}")
                return data
            except json.JSONDecodeError as e:
                logger.error(f"JSONDecodeError in load_data: {e}")
                logger.error(f"File contents at time of error: {f.read()}")
                return []
    except FileNotFoundError:
        logger.warning(f"FileNotFoundError: {DATA_FILE} not found. Returning empty list.")
        return []
    except Exception as e:
        logger.error(f"Exception in load_data: {e}")
        logger.error(traceback.format_exc())
        return []

def save_data(data):
    """Save data to the JSON file."""
    try:
        logger.debug(f"Attempting to open {DATA_FILE} for writing...")
        with open(DATA_FILE, 'w') as f:
            json.dump(data, f, indent=4)
            logger.debug(f"Successfully saved data to {DATA_FILE}: {data}")
    except Exception as e:
        logger.error(f"Exception in save_data: {e}")
        logger.error(traceback.format_exc())

def append_data(new_data):
    """Append new data to the JSON file."""
    logger.debug(f"Appending data: {new_data}")
    data = load_data()
    data.append(new_data)
    save_data(data)
```

```
@app.route('/')
def dashboard():
   logger.debug("Entering dashboard route")
   repair_records = load_data()
   logger.debug(f"Loaded repair records: {repair_records}")
   volumes = [record.get('pothole_volume_cm3', 0) for record in repair_records]
   timestamps = [record.get('timestamp', '') for record in repair_records]
   cement_used = sum(record.get('cement_units_used', 0) for record in repair_records)
   water_used = sum(record.get('water_units_used', 0) for record in repair_records)
   pothole_counts = sum(record.get('cumulative_potholes_fixed',0) for record in repair_records)
   total_distance = sum(record.get('cumulative_distance_m', 0.0) for record in repair_records) * 100 #convert to
cm
   logger.debug(
      f"Volumes: {volumes}, Timestamps: {timestamps}, Counts: {pothole_counts}, Distance: {total_distance},
Cement: {cement_used}, Water: {water_used}")

   return render_template('dashboard.html',
                  records=repair_records,
                  volumes=volumes,
                  timestamps=timestamps,
                  pothole_counts=pothole_counts,
                  total_distance=total_distance,
                  cement_used=cement_used,
                  water_used=water_used)

@app.route('/receive_data', methods=['POST'])
def receive_data():
   logger.debug("Entering receive_data route")
   try:
      data = request.get_json()
      logger.debug(f"Received data: {data}")
      if data:
         data['timestamp'] = datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S')
         append_data(data)
         return 'Data received and saved successfully', 200
      else:
         return 'No data received', 400
   except Exception as e:
      logger.error(f"Exception in receive_data: {e}")
      logger.error(traceback.format_exc())
      return 'Error processing data', 400

@app.route('/update_status', methods=['POST'])
def update_status():
   logger.debug("Entering update_status route")
   return 'Status update endpoint (currently no action)', 200
```

```
if _name_ == '_main_':
    if not os.path.exists(DATA_FILE):
        logger.info(f"Data file {DATA_FILE} does not exist, creating it.")
        with open(DATA_FILE, 'w') as f:
            json.dump([], f)
        logger.info(f"Data file {DATA_FILE} successfully created.")
    else:
        logger.info(f"Data file {DATA_FILE} already exists.")
    app.run(debug=False,port = 5023)
```

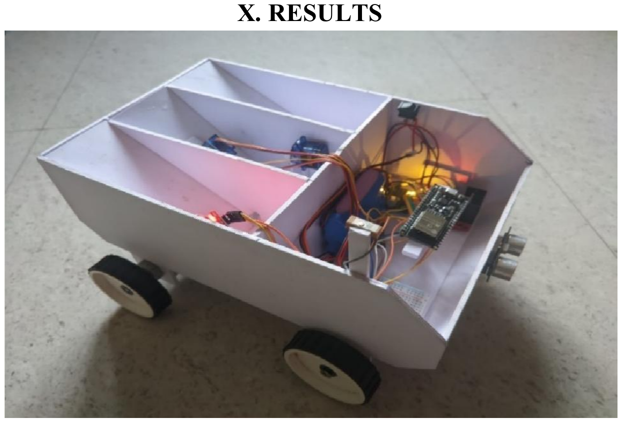
repair_data.json example structure


```
[
    {
        "latitude": 0.0,
        "longitude": 0.0,
        "pothole_length_cm": 0.36,
        "pothole_volume_cm3": 0.01,
        "cement_units_used": 1,
        "water_units_used": 1,
        "cumulative_distance_m": 0.56,
        "cumulative_potholes_fixed": 1,
        "timestamp": "2025-05-17 09:13:30"
    },]
```


IX. TESTING

Both black-box and white-box testing were conducted. Pothole detection accuracy, volume estimation, and servo operations were validated. Cloud communication and data display on the dashboard were also tested.

X. RESULTS



IJARSCT

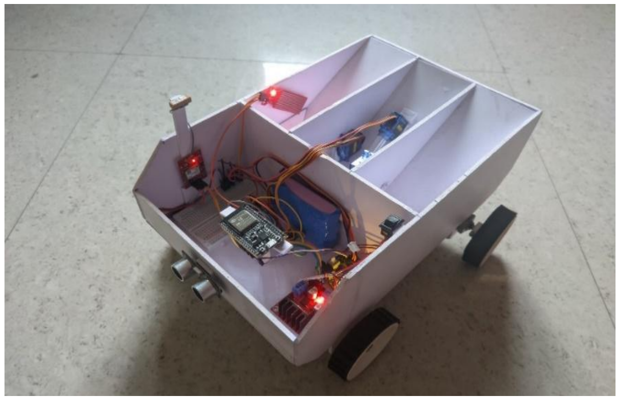
International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

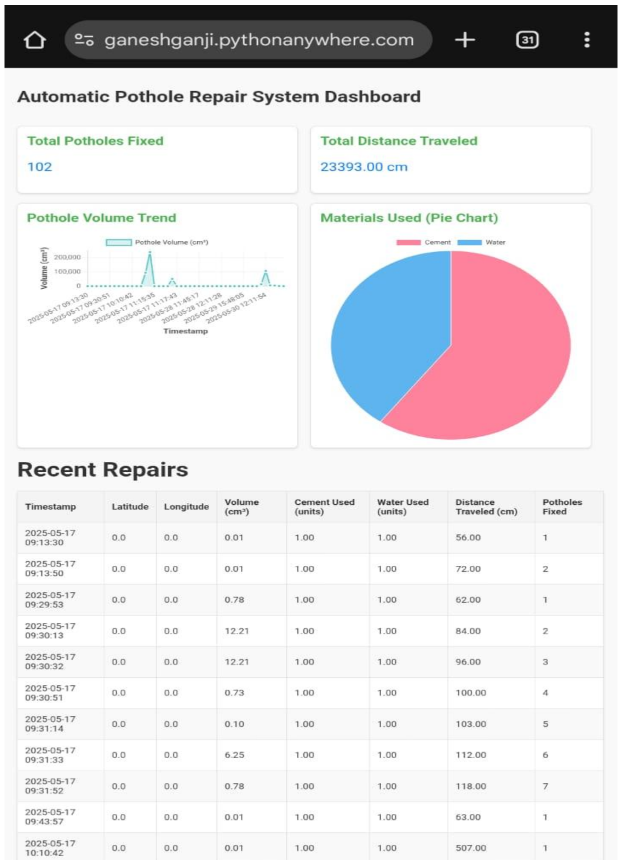
ISSN: 2581-9429

Volume 5, Issue 7, June 2025

Impact Factor: 7.67



WORKING MODEL



Timestamp	Latitude	Longitude	Volume (cm³)	Cement Used (units)	Water Used (units)	Distance Traveled (cm)	Potholes Fixed
2025-05-17 09:13:30	0.0	0.0	0.01	1.00	1.00	56.00	1
2025-05-17 09:13:50	0.0	0.0	0.01	1.00	1.00	72.00	2
2025-05-17 09:29:53	0.0	0.0	0.78	1.00	1.00	62.00	1
2025-05-17 09:30:13	0.0	0.0	12.21	1.00	1.00	84.00	2
2025-05-17 09:30:32	0.0	0.0	12.21	1.00	1.00	96.00	3
2025-05-17 09:30:51	0.0	0.0	0.73	1.00	1.00	100.00	4
2025-05-17 09:31:14	0.0	0.0	0.10	1.00	1.00	103.00	5
2025-05-17 09:31:33	0.0	0.0	6.25	1.00	1.00	112.00	6
2025-05-17 09:31:52	0.0	0.0	0.78	1.00	1.00	118.00	7
2025-05-17 09:43:57	0.0	0.0	0.01	1.00	1.00	63.00	1
2025-05-17 10:10:42	0.0	0.0	0.01	1.00	1.00	507.00	1

WEB INTERFACE

XI. CONCLUSION

The Automatic Pothole Repair System offers a smart and scalable way to tackle road maintenance using IoT, robotics, and cloud computing. It reduces labor, increases accuracy, and improves safety and response time. With real-time data analytics and autonomous repair, it represents a major step toward smarter road infrastructure.

DOI: 10.48175/IJARSCT-28008

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

REFERENCES

[1]. Zhang, Y., & Patel, R. (2023). "Eco-Friendly Materials in Urban Road Repairs: A Review." *Journal of Sustainability in Construction*.
[2]. Green, T. (2023). "The Future of Smart Cities: Integrating IoT with Infrastructure." *UrbanTech Review*.
[3]. NYC Department of Transportation (2022). "LIDAR in Urban Maintenance." Retrieved from NYC DOT.
[4]. Patel, R., & Singh, T. (2022). "IoT-Based Road Surface Monitoring Using GPS and Ultrasonic Sensors."
[5]. https://github.com/ganesh949152/AutomaticPotholeRepairSystem