Advanced Robotic Surveillance for Urban Air Quality Safety

Guillaume Hansen, James R. Benson, Muhammad Rafay Danish, Khoa Truong, Xinrui Yu and Jafar Saniie
Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

***Abstract*— This paper presents a multifunctional quadruped robot, specifically engineered for comprehensive air quality monitoring and emergency assistance. Designed to navigate through urban environments, the robot autonomously collects and transmits real-time air quality data to a centralized database. This interdisciplinary paper integrates robotics, environmental science, and emergency response protocols to propose novel solutions for smart city infrastructure, public health, and disaster management. Specifically, this research highlights the critical role that autonomous systems play in monitoring environmental conditions and advancing safety protocols. This research leverages a comprehensive array of sensors including CO2 (MH-Z19), particle (SDS 011), GPS (ADA 746), DHT 22, MQ 9, and ADS 1115 converter, each meticulously implemented to ensure precise environmental data collection and analysis. Furthermore, the integration of cutting-edge technologies such as Robot Operating System (ROS), .NET MAUI, Python, and MariaDB establishes a robust framework for seamless operation, data processing, and secure storage within the quadruped robot system.**

Keywords*— *quadruped robot, smart system, ROS, air quality sensor, remote control API, database

I. Introduction

This paper revolves around the deployment of a quadruped robot designed to assess and monitor air quality in various environments, subsequently transmitting this data to a central database. Primarily envisioned for urban areas, the robot's application extends beyond air quality monitoring in large cities. It possesses the versatility to assist in several critical scenarios: Assisting firefighters in locating gas leaks, thereby minimizing risk to human life; monitoring radioactivity levels and detecting the presence of chemical gases; contributing to anti-terrorism efforts.

The robot is capable of operating in diverse settings, including stadiums, urban streets, factories, industrial facilities, and cruises. A key application involves tracking and monitoring air quality in specific urban areas. To facilitate this, an operator, remotely located, will control the robot's movements, focusing on areas requiring air quality assessment. Both the robot and the operator will connect to the local city's Wi-Fi network. In Wi-Fi blind spots, it may be necessary for the city to install additional routers to ensure continuous monitoring.

This paper focuses on the deployment of a quadruped robot specifically designed for assessing and monitoring air quality in diverse environments, with the capability to transmit data to a central database. While inspired by existing projects like Boston Dynamics Spot and ANYmal by ANYbotics, our robot distinguishes itself through its specialized features and applications. Similar to Boston Dynamics Spot, our robot shares a highly mobile quadruped platform and adaptable sensors, but it diverges in its focus on air quality-specific design elements such as enhanced sensor placement and specialized locomotion for navigating tight urban spaces efficiently. Moreover, unlike Spot, our robot emphasizes autonomous pathfinding tailored to identify and assess air quality anomalies, complemented by specialized data processing and visualization tools for detailed analysis. Conversely, compared to ANYmal by ANYbotics, our robot is uniquely suited to the complexities of urban environments, with advanced capabilities for gas and particulate analysis tailored specifically to air quality monitoring. Additionally, our solution offers simplified deployment strategies for quick setup in emergency response scenarios, addressing a need that may not be prioritized by ANYmal due to its industrial inspection focus. Through these distinctions, our robot presents a comprehensive solution for urban air quality monitoring, offering versatility and effectiveness across various scenarios and environments.

Deploying a quadruped robot for air quality monitoring and emergency assistance offers numerous advantages. Firstly, its four-legged design ensures enhanced stability and maneuverability, facilitating navigation through diverse urban environments, including confined spaces and uneven terrain. This agility enables access to hard-to-reach areas crucial for comprehensive air quality assessment. Secondly, the robot's compact size and minimal environmental footprint minimize disruption in populated areas while maintaining effectiveness. Additionally, its versatility allows for deployment in various critical scenarios, from detecting gas leaks during firefighting to monitoring hazardous substances. With adaptability to different settings, including stadiums, factories, and urban streets, the quadruped robot proves invaluable for addressing urban air quality challenges and emergency response needs effectively.

II. System Design

A. Control logic

The robot is based on a quadruped chassis, with a Raspberry Pi running ROS (Robot Operating System) working as the controller of the robot. Other sensors include air quality sensors, a lidar, and a camera. The lidar is used to scan and detect objects around the robot, and the camera is used to provide useful visual information for the operator of the robot. This configuration is shown in Fig. 1.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

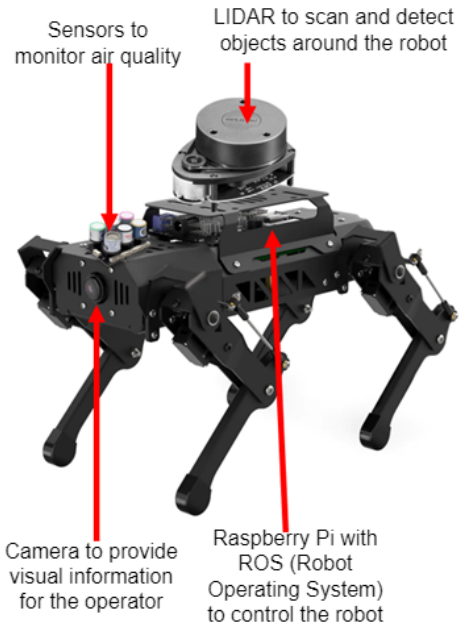


Fig. 1. Hardware Architecture of Our System

These are the steps we will need to do to implement the control logic of the robot:

Initially, we will implement the Robot Operating System (ROS) on the Raspberry Pi (RPi) and begin working with ROS. Subsequently, we will focus on controlling the robot's movements using ROS. Following that, we will take control of the robot's camera, again utilizing ROS. Next, we will develop a method for remote control of the robot. Afterward, our attention will turn to controlling the LIDAR system via ROS. Then, we will proceed with the implementation of air quality sensors. Finally, we will record the power consumption and conduct a battery test.

1) Implementation of ROS: The Robot Operating System (ROS) is an open-source, meta-operating system designed for robotics projects. It provides the services expected from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [1]. The RQT graph with all nodes and topics of our system is shown in Fig. 2.

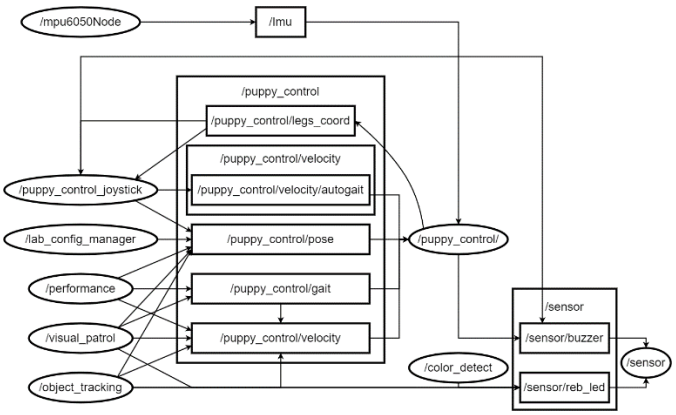


Fig. 2. RQT graph of our system

2) Control of the Movements of the Robot: Gait [2] is the pattern of movement of the limbs of animals. Generally speaking, it is used to describe how the animal walks. The common quadruped gaits include Trot, Walk, and Amble. Trot [3], [4], [5] is a gait at medium to low speed where the feet at diagonal opposite ends of the body strike the ground together. This most frequently used quadruped gait has a wide motion range and combines stability and speed. While PuppyPi is moving, there are two types of parameters, including the motion parameter and the gait parameter. The gait parameters include the time when 4 legs all touch the ground, the time when all legs lift, and the interval to switch between the front leg and hind leg. Motion parameters include moving speed and rotation speed. We implemented using a Python script 4 types of movements: go forward, go backward, turn left, and turn right. In the same program, we also designed 2 statics parameters: the height and the pitch of the robot. The Pitch and the roll of our system are shown in Fig. 3. Thanks to those parameters, we were able to control the angle of the camera using, for instance, a look-up posture [6][7][8][9].

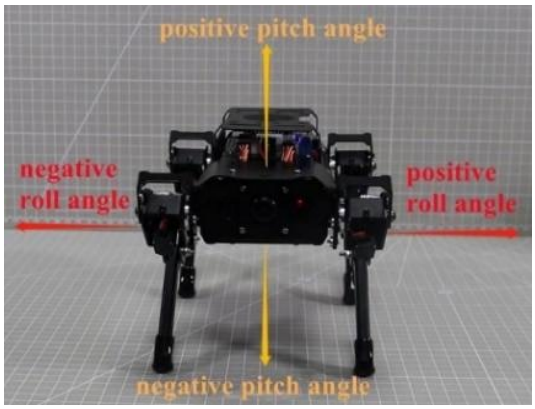


Fig. 3. Pitch and Roll angle illustration

3) Access to camera feedback: An important library that we are going to use in this paper is OpenCV. OpenCV (Open-Source Capture Vision) is a computer vision library for free handling various tasks about image and video, for example, displaying the image collected by the camera and making the robot recognize the real object. The idea of our program is to open a new window on the Puppy Pi Raspberry and receive the camera feedback.

4) Remote control of the robot: We implemented a TCP client/server protocol to control remotely the robot. This involved implementing Python sockets to establish seamless communication between a control computer and the robot Raspberry Pi. In our case, the client program will run on the control computer and the server program will run on the robot Raspberry Pi. The communication protocol involved the transmission of individual keys, each representing specific actions. For instance, pressing the W key signaled the robot to move forward [10], [11].

5) Lidar: The idea behind the use of our Lidar, is to have direct feedback of the environment where the robot evolves. We are going to create an application that will display the lidar point on Rviz [12]. An example of an outcome using Rviz is shown in Fig. 4.

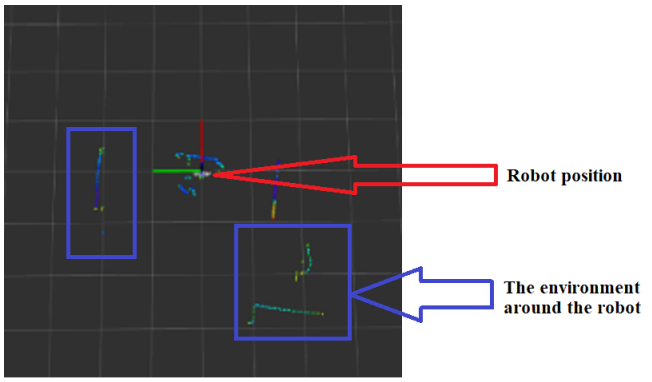


Fig. 4. Example of LIDAR feedback using Rviz

6) Sensor implementation: Of course, one of the key points of our paper is the implementation of the sensors. We are going to implement the major part of the sensors on top of the robot, only the particle one will be under the robot. Some pictures of the robot after this implementation are shown in Fig. 5.

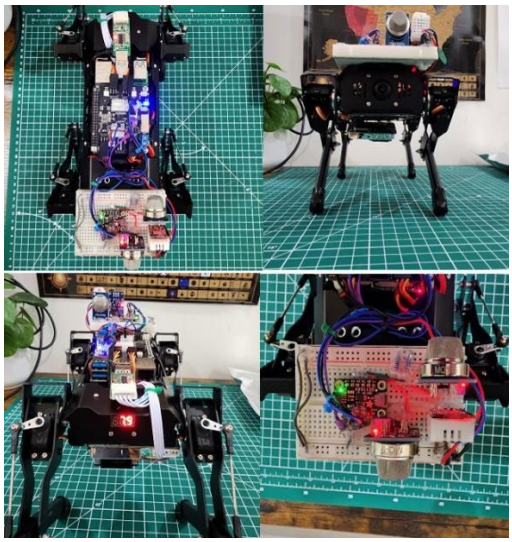


Fig. 5. Robot after the sensor implementation

7) Power consumption and battery test: We are going to test the real charge time of the robot. the robot is fully charged in 1 hour and 20 minutes. The battery life of our system is around 1 hour and 15 minutes. Finally, the average power consumption is P = U*I = 7.6 * 1.6 = 12.16 W.

B. Sensor

This section outlines the implementation of various sensors for our paper, each serving a specific purpose. The CO2 sensor (MH-Z19) measures atmospheric CO2 levels, categorizing them into different concentration ranges for air quality assessment. The particle sensor (SDS 011) detects PM 10 and PM 2.5 particles using a laser and a USB-to-serial adapter. A GPS sensor (ADA 746) from Adafruit is employed for tracking the robot, capable of connecting with up to 22 satellites. The DHT 22 sensor monitors temperature and humidity, easily connecting to a Raspberry Pi. For methane detection, the MQ 9 sensor is used, requiring an analog-to-digital converter due to its dual signal output. Lastly, the ADS 1115 converter interfaces with the Raspberry Pi to read the analog signal from the MQ 9 sensor, supporting up to four sensors simultaneously with a 16-bit resolution. The implementation outline of our sensors is shown in Fig. *6.*

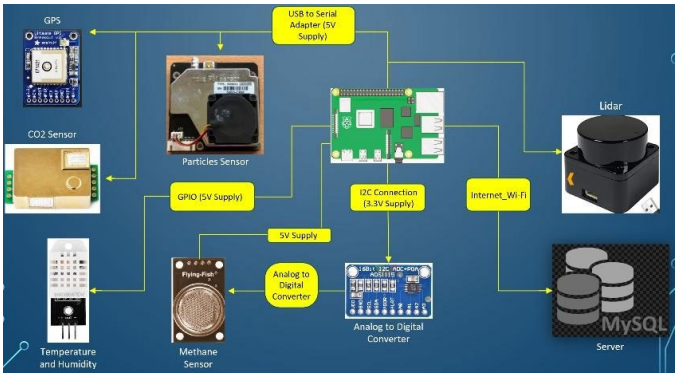


Fig. 6. Implementation outline of our sensors

C. Application

1) Technology stack: The foundation of our application's functionality begins with the Robot Operating System (ROS), which is integral to the robot's movement and operation. ROS provides a flexible framework for writing robot software, and it is a collection of tools and libraries designed to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Building on this foundation, we've chosen to develop our application using .NET MAUI, with C# as the programming language of choice. The .NET MAUI framework allows for the development of cross-platform applications with ease, making it an ideal choice for an application that needs to function seamlessly across various devices, including Android and MacOS. Python will play a pivotal role as well, particularly for its rich data libraries and its ability to facilitate a smooth connection with ROS. This will be crucial for processing and interpreting the data collected by the robot. For our database needs, we selected an SQL database, augmented with MariaDB. MariaDB offers an efficient and reliable storage solution for critical data such as sensor readings and GPS location information. This robust combination ensures that our application not only collects but also securely manages the wealth of information provided by the robot.

2) Application pages: The application features a Home Page that is designed with simplicity and ease of use in mind. It includes essential functionalities like a Login Page and an option to Quit the Application. The Login Page is a gateway to secure access, linking users to the robot via SQL database authentication, emphasizing the paper's focus on security and data integrity. The Dashboard Page serves as the control hub, leading to various functionalities like the Robot Controller Page, Raw Data Page, and Cleaned Data Page. Each page caters to specific aspects of the robot's operation and data analysis. The Robot Controller Interface seamlessly connects the robot with the application and ROS, integrating a Python script for accessing the camera footage. The robot controller page is shown in Fig. 7.

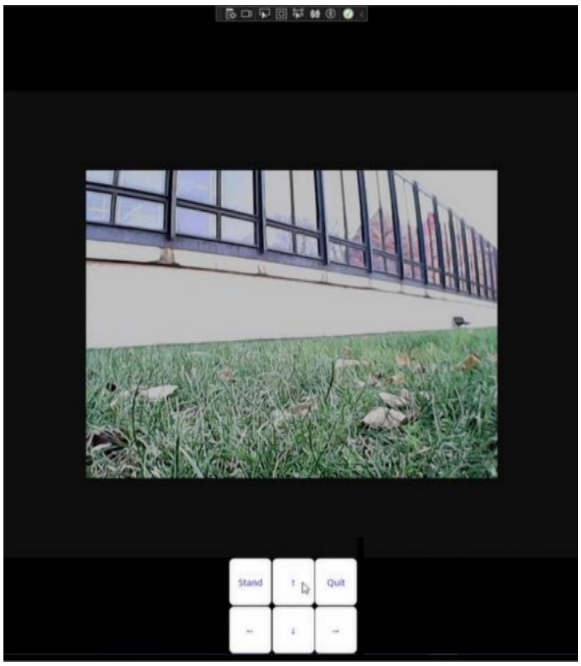
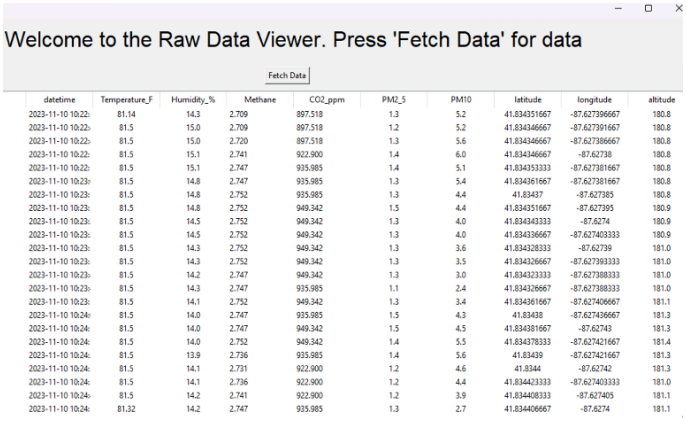


Fig. 7. Robot controller page

3) Data and Values: In terms of data handling, the Raw Data Page is particularly noteworthy. It presents sensor data in an Excel-like table format, allowing users to view comprehensive information in absolute values. This format offers the advantage of seeing all relevant data together, aiding in straightforward analysis and decision-making. The Raw data page is shown in Fig. 8.



datetime	Temperature_F	Humidity_%	Methane	CO2_ppm	PM2_5	PM10	latitude	longitude	altitude
2023-11-10 10:22:	81.14	14.3	2.709	897.518	1.3	5.2	41.834351667	-87.627396667	180.8
2023-11-10 10:22:	81.5	15.0	2.709	897.518	1.2	5.2	41.834346667	-87.627391667	180.8
2023-11-10 10:22:	81.5	15.0	2.720	897.518	1.3	5.6	41.834346667	-87.627386667	180.8
2023-11-10 10:22:	81.5	15.1	2.741	922.900	1.4	6.0	41.834346667	-87.62738	180.8
2023-11-10 10:22:	81.5	15.1	2.747	935.985	1.4	5.1	41.834353333	-87.627381667	180.8
2023-11-10 10:23:	81.5	14.8	2.747	935.985	1.3	5.4	41.834361667	-87.627381667	180.8
2023-11-10 10:23:	81.5	14.8	2.752	935.985	1.3	4.4	41.83437	-87.627385	180.8
2023-11-10 10:23:	81.5	14.8	2.752	949.342	1.5	4.4	41.834351667	-87.627395	180.9
2023-11-10 10:23:	81.5	14.5	2.752	949.342	1.3	4.0	41.834343333	-87.6274	180.9
2023-11-10 10:23:	81.5	14.5	2.752	949.342	1.3	4.0	41.834336667	-87.627403333	180.9
2023-11-10 10:23:	81.5	14.3	2.752	949.342	1.3	3.6	41.834328333	-87.62739	181.0
2023-11-10 10:23:	81.5	14.3	2.752	949.342	1.3	3.5	41.834326667	-87.627393333	181.0
2023-11-10 10:23:	81.5	14.2	2.747	949.342	1.3	3.0	41.834323333	-87.627388333	181.0
2023-11-10 10:23:	81.5	14.3	2.747	935.985	1.1	2.4	41.834326667	-87.627388333	181.0
2023-11-10 10:23:	81.5	14.1	2.752	949.342	1.3	3.4	41.834361667	-87.627406667	181.1
2023-11-10 10:24:	81.5	14.0	2.747	935.985	1.5	4.3	41.83438	-87.627436667	181.3
2023-11-10 10:24:	81.5	14.0	2.747	949.342	1.5	4.5	41.834381667	-87.62743	181.3
2023-11-10 10:24:	81.5	14.0	2.752	949.342	1.4	5.5	41.834378333	-87.627421667	181.4
2023-11-10 10:24:	81.5	13.9	2.736	935.985	1.4	5.6	41.83439	-87.627421667	181.3
2023-11-10 10:24:	81.5	14.1	2.731	922.900	1.2	4.6	41.8344	-87.62742	181.3
2023-11-10 10:24:	81.5	14.1	2.736	922.900	1.2	4.4	41.834423333	-87.627403333	181.0
2023-11-10 10:24:	81.5	14.2	2.741	922.900	1.2	3.9	41.834408333	-87.627405	181.1
2023-11-10 10:24:	81.32	14.2	2.747	935.985	1.3	2.7	41.834406667	-87.6274	181.1

Fig. 8. Raw data page

4) Histogram: The application also features a histogram on a normalized scale of 0-100, complete with comfortable ranges, on the Cleaned Data Page. This visualization tool is significant for its ability to present data in a user-friendly manner, highlighting crucial environmental parameters within acceptable limits. The advantage here is the clear depiction of data, making it easier to understand and interpret. The Histogram page is shown in Fig. 9.

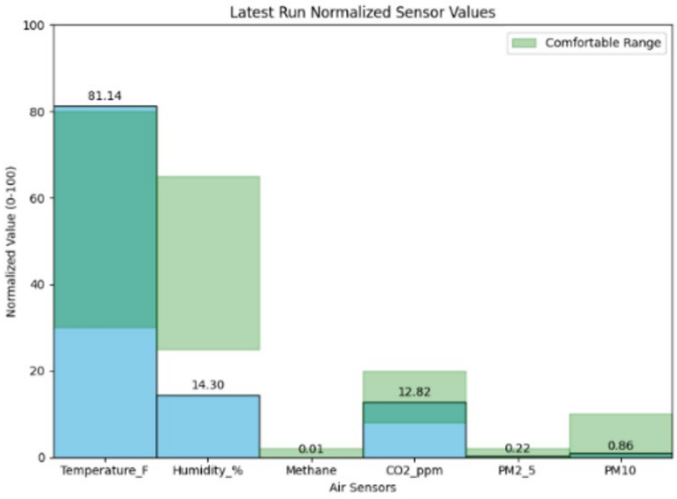


Fig. 9. Histogram with normalized sensor values

5) Line plot: Additionally, the Cleaned Data Page includes a line plot that is invaluable for observing trends, spikes, and correlations in environmental data. This aspect of data presentation is crucial for identifying patterns over time and offering insights into environmental changes and potential anomalies. The advantage of this feature lies in its ability to provide a detailed and dynamic analysis of the data, facilitating informed decisions and actions. The Line plot is shown in Fig. 10.

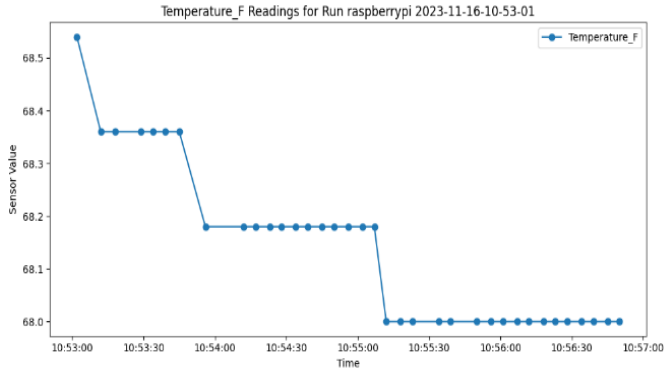


Fig. 10. Line plot of temperature readings in Fahrenheit

D. Server

1) Video processing: Video transmission: We implement a Python video transmission program. This code interfaces with the robot's USB camera, transmitting the video stream over sockets to a designated receiving client. We further optimized the code to facilitate dynamic adjustments to resolution and color depth parameters, enabling scalability and adaptability. The resolution can be scaled down from the input image, a strategy aimed at conserving internet bandwidth. Similarly, color depth reduction minimizes information transmitted over the internet connection. In the current version, we achieved a substantial reduction in required bandwidth, down to 4.3 megabytes per second. Future enhancements involve exploring alternative codecs, such as AV1, renowned for significantly reducing the bandwidth needed for video stream transmission.

Video transcoder: The application's essential feature is its ability to display a live video feed from a quadruped robot's camera, achieved by streaming through an HTTP IP address from the Raspberry Pi server. We developed this functionality. The video transcoder Python program plays a crucial role as an intermediary, receiving the raw video stream from the robot and converting it into a format compatible with HTTP connections. This enables efficient hosting on a web server, making the live video stream accessible through an HTTP address and seamlessly integrating it into the application's interface. The code addresses the practical need for a streamlined, real-time video streaming solution, optimizing the transcoding process to achieve a maximum transfer speed of approximately 4.3 megabytes per second. This underscores the method's effectiveness in delivering a responsive and efficient live video streaming experience within the application.

2) Database: In our research, configuring a Raspberry Pi with Raspbian OS and integrating MariaDB as the SQL server played a pivotal role in establishing a robust infrastructure. Raspbian OS installation ensured hardware compatibility, and MariaDB brought scalability to our database system. This configuration served as a standalone database and facilitated connectivity between our application and the sensor collector program, creating an interconnected ecosystem that enhanced real-time data retrieval. The optimized integration underscored the Raspberry Pi's versatility, transforming it into a multifunctional platform supporting diverse functionalities. Opting for an SQL database for sensor data offers advantages in systematic organization, efficient querying, and support for complex analyses. The structured, relational, and secure nature of SQL databases makes them an optimal choice for managing and deriving insights from sensor data across various applications. An overview of our MySQL management portal is shown in Fig. 11.

Fig. 11. MySQL Management portal

III. Experimental Results

Our endeavor has resulted in the development of a sophisticated product capable of capturing and storing sensor data seamlessly into the MySQL database. This MySQL database was hosted on a Raspberry PI 4 giving us a highly efficient platform that we could utilize as a server. Furthermore, this innovative creation allows users to control the quadruped robot remotely through an application, providing a live stream feature that ensures accessibility from any location on the Illinois Institute of Technology (IIT) campus. We managed to conduct partial functionality tests in outdoor terrains by employing a smaller-sized version of the robot. We subjected the robot to various environments. The factors we tested included temperature variations, such as warm and cold conditions, as well as wind conditions, distinguishing between windy and calm days. Additionally, we explored different types of terrain, such as mud and soft ground.

The result is a fully functional quadruped robot, capable of controlled movement and real-time sensor data collection. The information gathered by the robot as it traverses its environment provides valuable insights into air quality.

Acknowledging its size and agility, we recommend the robot for indoor applications, given its compact design and the inherent limitations associated with navigating outdoor terrains, especially dirt. This strategic decision is based on practical considerations, as the robot's smaller size might not optimize speed in uneven outdoor environments. In contrast, a full-size quadruped robot would navigate at a swifter pace, enabling it to cover larger areas efficiently—a critical factor for comprehensive data collection. An example final result of our paper is shown in Fig. 12. In this figure it can be seen that our robot is walking a grass field in calm and moderate weather, to test our robot in optimal condition. Being outside provided the GPS module with very good conditions for connecting to the positioning satellite, allowing for the collected sensor data to have accurate location data for when it was sampled. Providing further analysis of the data based on its location and when it was sampled. Another condition that the robot was tested under was the use of a 2.4GHz WiFi that covered the Illinois Institute of Technology. Having this as the connection method enabled further range between access points that are spread out and about on campus than if we connected on the 5GHz method. Running it on the 5GHz would not have provided any significant improvement since 2.4GHz has a better coverage area and penetrating objects. Given all of this, our system was able to be controlled remotely via our application, provide insight into the collected data to analyze the air quality, and store all of our data.

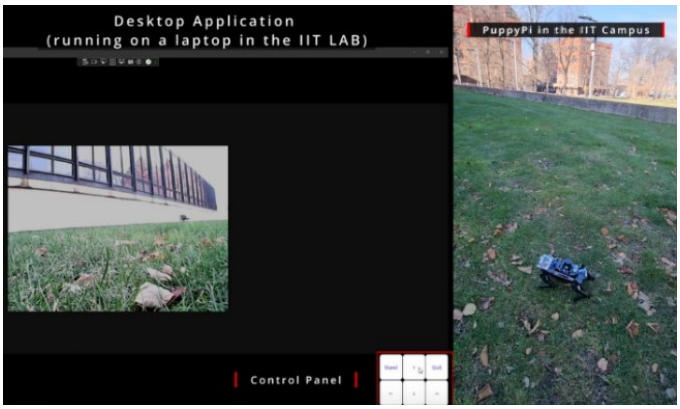


Fig. 12. Robot with live stream being controlled by our application

IV. Conclusion

Our project has multiple parts that are required to make it all come together as a functional system. Each part was delegated to members of the team enabling the person to work on the task independently from one another while tagging up with each other during our team meetings. This would allow members to bring up any issues that they are running into and ask for help from the team on the task.

Core components of our system come from the robot, sensors, application, and backend server. Each member had a responsibility to that component along with assisting any member when they asked for help. This greatly helped with taking on problems and keeping a fast pace. More on the core components, we chose the particular robot due to the cost and being able to design a model that could be scaled up. The sensor we chose allowed us to dig into different metrics that would help signify any issues that would appear in the air. C# was chosen for being able to develop applications for multiple platforms which is why we designed our application on it. The backend server was based on Rasbian OS running MySQL and Python for processing due to both being open source and supported on our platform.

The realization of our idea faced challenges stemming from various factors, including the dimensions of the robot, the quality of sensors utilized, and the incorporation of a Dynamic DNS client on the robot. The size of the robot significantly influenced the outcomes of our experiments, particularly when operating outdoors. The limitations in speed and mobility, attributed to its size, hindered the collection of comprehensive results.

Upon completion of our testing phase, it became evident that the size constraint led to slow movement, and attempts to maximize speed introduced stability issues. The robot exhibited noticeable shakiness while traversing non-flat surfaces, with a speed variation from 5 cm per second to 10 cm per second. Despite these challenges, the robot demonstrated excellent functionality on smooth terrain and indoors. Surprisingly, an unexpected issue arose during the setup of the Dynamic DNS client.

In addressing the challenges faced by the quadruped robot in negotiating diverse terrains and implementing the Dynamic DNS client, an increase in budget emerges as a potential solution. A larger budget would facilitate the enlargement of the quadruped robot, enabling it to effectively operate in rugged outdoor terrains. Moreover, the enhanced budget could extend the temperature range of the sensors, ensuring functionality in colder weather conditions. Allocating more time to the setup of the Dynamic DNS client is essential to rectify issues related to IP changes during reboots and shutdowns.

In summary, overcoming the limitations posed by size, stabilizing mobility and speed, and refining the Dynamic DNS client setup are pivotal aspects that could be addressed, ultimately enhancing the overall performance of the quadruped robot in various environmental conditions. The collaborative efforts have resulted in the successful design and creation of a quadruped robot, seamlessly controllable through a desktop application. This application not only provides a means of controlling the robot but also offers a comprehensive visualization of both raw sensor data and information detailing sensor boundaries, crucial for assessing air quality. In essence, our paper has not only overcome challenges but has also delivered a versatile and functional quadruped robot poised to contribute significantly to indoor air quality monitoring within the confines of the IIT campus.

References

[1] S. Chang, H. Du, Y. Cong, F. Xie and J. Zhang, "Gait Planning of Quadruped Robot Based on ROS," 2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS), Guangzhou, China, 2020, pp. 761-766.

[2] H Liu, X Zhang, C Fu et al., "Compliant gait generation for a quadruped bionic robot walking on rough terrains[J]", Robot, vol. 36, no. 5, pp. 584-591, 2014.

[3] Z Si, J Gao, X Duan, et al., Trot pattern generation for quadruped robot based on the ZMP stability margin, 2013.

[4] M H. Raibert, "Trotting pacing and bounding by a quadruped robot", Biomechanics, vol. 23, no. supp-S1, pp. 79,83-81,98, 1990.

[5] B. Ma, Z. Liu, C. Peng and X. Li, "Trotting gait control of quadruped robot based on Trajectory Planning," 2021 4th World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM), Shanghai, China, 2021, pp. 105-108.

[6] V Matos and C P. Santos, "Omnidirectional locomotion in a quadruped robot: A CPG-based approach", 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.

[7] W Du, M Fnadi and F B. Amar, "Rolling based locomotion on rough terrain for a wheeled quadruped using centroidal dynamics", Mechanism and Machine Theory, vol. 153.

[8] R Kurazume, Y Kan and S. Hirose, "Feedforward and Feedback Dynamic Trot Gait Control for Quadruped Walking Vehicle", Autonomous Robots, vol. 12, no. 2, pp. 157-172, 2002.

[9] A. A and V. R. Jisha, "Reinforcement Learning based Control of a Quadruped Robot," 2022 IEEE 19th India Council International Conference (INDICON), Kochi, India, 2022, pp. 1-6.

[10] H Xie, M Ahmadi, J Shang, et al., "An intuitive approach for quadruped robot trotting based on virtual model control", Proceedings of the Institution of Mechanical Engineers Part I Journal of Systems & Control Engineering, vol. 229, no. I4, pp. 342-355, 2015.

[11] J E Pratt, G A Pratt and E. Pratt, Virtual Model Control of a Biped Walking Robot, 2000.

[12] B. Siregar, H. Bagustiawan and O. S. Sitompul, "Environmental Mapping Automation by Quadruped Robot Using Lidar Technology," 2023 8th International Conference on Electrical, Electronics and Information Engineering (ICEEIE), Malang City, Indonesia, 2023, pp. 1-6.