Volume XXIII 2020
ISSUE no.1
MBNA Publishing House Constanta 2020

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

Smart mobile system for environmental recognition and monitoring

To cite this article: Alexandru Constantin and Eugen Petac, Scientific Bulletin of Naval Academy, Vol. XXIII 2020, pg.104-115.

Available online at www.anmb.ro

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-20-I1-014
SBNA© 2020. This work is licensed under the CC BY-NC-SA 4.0 License

Smart Mobile System for Environmental Recognition and Monitoring

A. Constantin[1], **E. Petac**[2]

[1,2]"Ovidius" University of Constanța, 124, Mamaia Blvd., Constanta, Romania
[1]constantin.alex96@icloud.com, [2]epetac@univ-ovidius.ro

Abstract. This paper proposes a mobile system for temperature, humidity, dewpoint, thermal comfort index and gas monitoring, as well as environmental recognition, performed remotely. The application is called ReconBot, and from an architecture standpoint, the application is based on the client-server paradigm. Our solution works in a secure and reliable way in terms of storage, transmission and authentication. The results offered by our solution can be used in environmental research scenarios.
Key words: Environmental Monitoring, Environmental Recognition, Mobile System.

1. Introduction

The human body is affected by sudden temperature and humidity changes, as well as elevated gas levels. If thermal comfort index values are unusually low or high, cellular-level changes occur, and human health is put at risk [1]. To evaluate the thermal comfort index, we need to assess the dewpoint temperature [2]. The dewpoint corresponds to the temperature at which the surrounding air is saturated with water vapors. Extreme humidity and temperature conditions affect us, humans, but affect animals [2] and plants [3] as well [4].

As for gas levels, it is important to mention that elevated carbon monoxide (CO) levels represent a real threat to humans. Carbon monoxide is an odorless, colorless gas which results from fuel burning, and symptoms of CO poisoning are headache, dizziness, weakness, vomiting and chest pain [5]. To calculate gas values, we use a measurement unit called PPM (parts per million), which is equivalent to units of mg/m3 [6]. Values in excess of 200 PPM represent an immediate danger, and values in excess of 6400 PPM can lead to death within minutes [7]. Talking about LPG levels, it is important to note that values in excess of 2000 PPM represent an immediate danger, and values in excess of 100.000 PPM can lead to asphyxia and death within minutes [8].

In this paper we propose a robust system called ReconBot, which is based on the Raspberry Pi 3 Model B [9] single board computer (SBC). From a hardware standpoint, the SBC is mounted on a car-like platform, the wheels being driven by an L298N module [10] which is connected to the SBC. The temperature and humidity readings are obtained using the DHT11 sensor [11], and the gas related readings are obtained using a MQ2 gas sensor [12], which is part of a setup that also includes the MCP3008 analog-to-digital converter [13], and a logic level converter [14]. The use of such converter is required due to the fact that the MQ-2 sensor operates at 5 volts, which is too much for the GPIO pins of the Raspberry Pi board which were design to operate at 3,3 volts. The MCP3008 module is required to obtain accurate PPM readings.

Talking about the software side of the system (which is presented in more detail in Section 3), and the client-server paradigm, the SBC hosts the main server, called ReconBotMaster, a database server used for accessing sensor data that was obtained in the past, and a server which is used to access the camera feed, which is used for environmental recognition. The main server uses the Transmission Control Protocol (TCP), one of the first and most important protocols of the Internet protocol suite. The main advantages of TCP are the reliability of the data transfers, and the fact that it works in a device-agnostic manner [15].The client, called ReconBotClient, is an application designed for the Windows operating system, which allows users to connect to the device and gather data. The application was developed using the C# and Python programming languages, and the .NET Framework. The solution offers security, reliability, and ease of use, thanks to the layout of the Windows client, which is able to generate statistics and graphs regarding the monitoring status of a certain day and hour. Some results are presented in Section 4.

2. Theoretical Background

The client-server model is a distributed application architecture which assigns tasks to the entities that provide services or resources, called servers, and to the entities that request such resources, called clients [16]. Usually, clients and servers are located in separate networks.

C# [17] is an object-oriented programming language created by Microsoft, derived from the C and C++ programming languages. It also includes influences from other programming languages, most notably Java, being a robust solution for developing robust industrial-grade web-based and desktop applications.

.NET Framework [18] is a software platform which provides resources that allow programmers to write cross-platform applications. The applications which are powered by .NET Framework work on a variety of machines, if certain requirements are met. Also, the development process is easier, because the platform manages the memory allocated to its applications automatically, not to mention the useful debugging and error handling features it provides. Another important feature is the variety of programming languages which can be used with .NET Framework. It is important to mention that the platform is based on two main components, namely Common Language Runtime (CLR), which handles application execution, memory management and exceptions, and Base Class Library (BCL) which provides a lot of useful resources, such as code necessary to implement user interfaces, database connections, network communications and so on. To develop our own application, we use both BCL code and our own code [4].

Python [19] was created as a high-level multipurpose programming language by Guido Van Rossum, with the goal of developing an object-oriented language which makes use of a clean syntax. Python is the language of choice for the Raspberry Pi board, because of 3 main reasons: it is an open-source programming language, which led to its adoption at a large scale; the software created with Python utilizes system resources in a very optimal way, a key feature when we talk about systems that work with limited resources; it has a very large number of libraries, developed both by the Python Software Foundation and the developer community. Also, Python is preferred in automatization, artificial intelligence and data analysis scenarios.

To calculate the dewpoint temperature, identified by TD and expressed in Celsius degrees, the following formula is used [20]:

$$TD = 243.04 * (LN(RH/100) + ((17.625*T)/(243.04+T)))/(17.625-LN(RH/100) - ((17.625*T)/ (243.04+T)))$$

where:

- T - environment temperature in Celsius degrees;
- RH – relative humidity in percentages;
- LN – natural logarithm.

To calculate the thermal comfort index value, identified by TCI, we use the following formula [21]:

$$TCI = (T*1.8+32)-(0.55-0.0055*RH)*((T*1.8+32)-58)$$

where:

- T - environment temperature in Celsius degrees;
- RH – relative humidity in percentages;
- $TCI \leq 65$ – comfort;
- $66 \leq TCI \leq 79$ – alert;
- $TCI \geq 80$ – discomfort.

For environmental recognition we use a camera and a background subtraction algorithm in order to detect motion in the surrounding environment. The implementation of the algorithm makes extensive use of OpenCV, a computer vision and machine learning library [22], and consists of the following steps:

- The first step is to calculate the weighted average of the previous frames, this average being also called background model;
- The second step is to subtract the current frame from the average we previously calculated;
- After that, the third step is to use the thresholding technique to highlight the regions of the subtraction we calculated earlier with differences in pixel values; white – foreground, black – background; so far, we applied the basic steps of the algorithm
- For the fourth step we use basic techniques such as erosions and dilations to remove noise and blobs from the image
- Finally, for the fifth step we use contour detection to extract the regions containing motion [23].

It is important to mention that other motion detection algorithms do exist, like the frame difference algorithm, which represents the simplest form of background subtraction.
This technique falls short when it comes to noise and variations in illumination, and because of the way it works, by only using a single previous frame instead of using an average of the previous frames (like the background subtraction method), it has problems with uniformly colored moving objects and with non-background objects if they stop moving [24], thus, this method proves to be unreliable in our use case, since the scenarios in which the device will work are varied and unpredictable (i.e. an open field, or indoors).

3. Proposed Method

3.1. The Hardware Solution

3.1.1. Raspberry Pi 3 Model B. The Raspberry Pi 3 Model B board was developed by the Raspberry Pi Foundation, its main advantages being the low acquisition cost and the open-source nature of the whole project. This board has enough computational power to be used in many Internet of Things [25] projects, such as IP camera projects, and sensor-based projects. The board was designed to use Linux-based operating systems, such as Raspbian, which was developed by the same Raspberry Pi Foundation.

The board offers a 40-pin General Purpose Input/Output (GPIO) header which is useful for connecting external modules. The configuration of the pin header is depicted in Figure 1 [26].

The 3V3 and 5V pins are used to provide power to the external modules, while the "GPIO" pins are used for data transmission.

3.1.2. The DHT11 humidity and temperature sensor. The DHT11 digital sensor is a popular solution used to obtain temperature and humidity readings. It uses a capacitive module and a thermistor to analyse the surrounding air. By using 3 female-female cables we can connect the sensor to the board. The "VCC" pin on the sensor corresponds with a 3V3 power pin on the board. The "DATA" pin on the sensor corresponds with any "GPIO" pin on the board, while the "GND" pin on the sensor corresponds with a ground pin on the board.

3.1.3. The MQ-2 gas sensor. This sensor offers an efficient way of detecting LPG, carbon monoxide, and smoke. MQ-2 is a popular and cost-efficient sensor, another main advantage of its use being the multitude of readily available libraries which allow for an easier development process. In order to acquire reliable data from the MQ-2 sensor, we need an analog-to-digital (ADC) converter, like MCP3008.

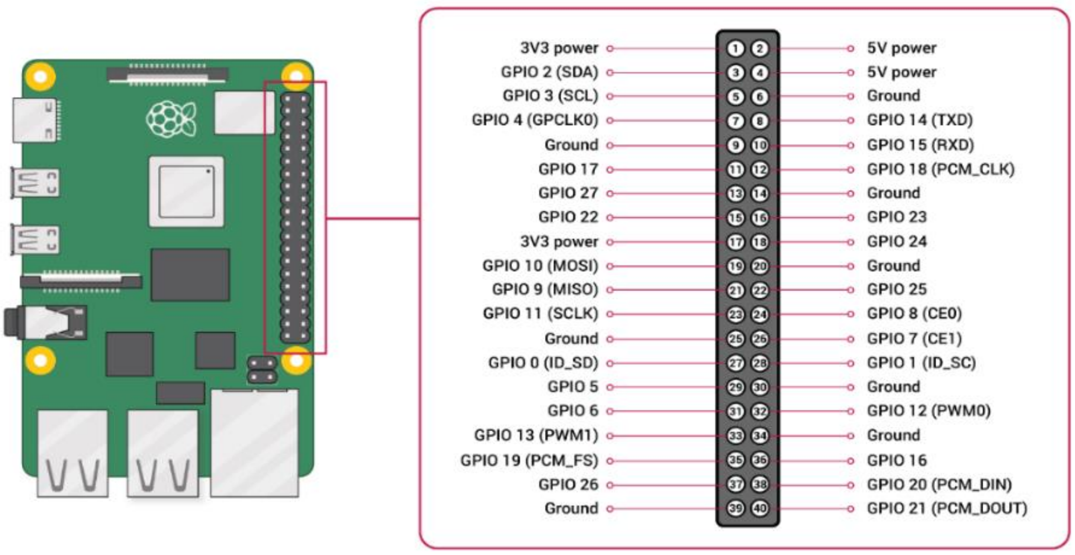


Figure 1. The GPIO header layout. Source: Raspberry Pi Foundation.

3.1.4. The MCP3008 analog-to-digital converter. In our case, the role of the MCP3008 module is to transform the analog signal from the MQ-2 sensor into a digital signal, then to feed it to the board. By using a library [27], we are able to process the signal from the MCP3008 module and provide accurate gas data.

3.1.5. Logic-level converter. In our case, this module is used to convert the 5V signal from the MCP3008 converter (originating from the MQ-2 sensor) which is too high for the GPIO pins of the board, into a 3,3V signal, which the pins were designed to use. Feeding a 5V signal to the GPIO pins can physically damage the board.

3.1.6. The L298N driver module. This module is responsible for driving the four wheels attached to the car platform, each of these wheels being powered by a DC motor.

3.1.7. The Pi NoIR camera. To obtain a video feed of the surroundings, which we need in order to drive the car and apply the motion detection algorithm, we used the Pi NoIR camera [28]. It works very well during night-time or when the luminosity is scarce. To connect it to the board, we used the CSI port.

3.2. The Software Solution

3.2.1. The main server – ReconBotMaster. The main server has five components:

- The "*data.py*" Python script which acquires data from the DHT11 sensor, and reads the "*gas_val.txt*" file which contains gas data

- The "*gas.py*" script which obtains data from the MQ-2 sensor and writes it in the "*gas_val.txt*" file
- The server executable, "*ReconBotMaster.exe*", which accepts inbound TCP connections from clients and replies to their messages; it was written using C#, and works under Raspbian via the Mono platform
- The system service which manages the executable of the main server
- The system service for "*gas.py*", its main purpose being to launch the script at system startup, to allow the MQ-2 sensor to calibrate

3.2.2. data.py – the script which reads and sends data from the sensors subsubsection. To create this script, we used the DHT11 Python [29] library, which allows us to communicate with the DHT11 sensor in order to obtain temperature, humidity, thermal comfort index and dewpoint values, while the script also reads data from the "gas_val.txt" file.

3.2.3. gas.py – the script which obtains data from the MQ-2 sensor subsubsection. This script is meant to run at system startup, while the device is placed in a clean air environment, for a proper calibration of the MQ-2 sensor. After that, by using the Raspberry Pi Gas Sensor MQ [30] library, we gather the gas data from the sensor, through the MCP3008 module, and we write it down in the "gas_val.txt" file, so the "data.py" script can read that data.

3.2.4. The executable of the main server subsubsection. This executable is a console application, which makes use of the C# programming language, and the .NET Framework. In order to run this application on a Linux-based operating system, we used Mono.

Mono is a software platform which helps in developing cross-platform applications, using the C# programming language and the .NET Framework [31]. Considering the fact that the executable was written on Windows, Mono will allow us to run it on Raspbian.

3.2.4.1. The TCPServer class. The "Start" method allows to server to listen for new client connections. The parameters required by this method are the TCP port on which the application will listen for new connections, and the encryption/decryption password necessary to process messages. This password is only known by the device administrator and the clients that wish to connect. The client-server communication is encrypted using the Advanced Encryption Standard (AES) algorithm, in Galois/Counter Mode (GCM) [32], passwords being derived into secure keys by using the PBKDF2 (Password-Based Key Derivation Function 2) password hashing algorithm [33].

The application generates logs for every important event, through the "*WriteToLog*" method.

When a connection with a client is successfully established, a new thread will be created, in which the requests will be processed.

The "*ProcessClientRequests*" method is used to process commands received from clients.

When the "*!connection*" command is issued, the server will reply with a confirmation message that the connection was established successfully.

When the "*!data*" command is issued, the server will run the "*data.py*" script, and will send its output to the client.

The "*!Info*" command allows us to gather information about the server's uptime, its version, and the number of connected clients.

The "*!exit*" command indicates that a client wishes to disconnect.

The commands "*!right*", "*!left*", "*!forward*" and "*!reverse*" are used to drive the platform, the server running the corresponding Python scripts that work with the L298N module.

The "*!shutdown*" command prompts the server to shut down the board.

If the server is not able to decrypt the data received from the client successfully, which means that a wrong password was used, the connection is terminated.

The "*Stop*" method will stop the server.

3.2.4.2. The Program class. To launch the application, we need the Program class. It will read the contents of the "*server.cfg*" configuration file, which is located in the same directory as the executable, obtaining the TCP port used for connections, and the security password, which must be between 15 and 30 characters long.

3.2.5. dbwriter.py – the script which populates the database with sensor data. Like the "*gas.py*" script, "*dbwriter.py*" will run at system startup, in order to populate the local database with data from the sensors. Thus, a large dataset will be stored, which can be used for generating statistics.

We use the MariaDB database management system [34], which is a fork of the popular MySQL DBMS, thus we used the MySQL connector in order to establish a connection with the database server.

3.2.6. webstreaming.py – the script which delivers the camera feed subsubsection. With this script we will create a webpage, using the Flask framework [35]. Flask is a web framework written in Python, which provides a simple but scalable way of providing web resources. The web resource that we will provide requires an authentication header, and by sending a POST request containing the "X-Password" parameter, we will gain access to the camera feed. We will apply the motion detection algorithm, and the region where motion is detected will be marked by a coloured square. When the device is moving, the algorithm will not be applied, to avoid false positives, thus the device needs to remain stationary for a short period of time for the script to apply the motion detection algorithm. The web resource provided by this script will be accessed from the Windows client.

3.2.7. The system services. In order to run the main server, the script which populates the database and the script for the gas sensor as system services, we used "*systemd*" [36], which is a manager that comes preinstalled on Raspbian, providing parallelization capabilities, as well as the possibility of configuring and monitoring system services. With "*systemd*" we can start, stop and restart these services, and most importantly, we can run them at system startup.

We need to create a "*.service*" file for each service we wish to install, which contains instructions necessary for the systemd utility, like the path of the application. After we create those files, we need to move them into the "*/etc/systemd/systemd*" folder, and we also need to restart systemd for the changes to occur.

3.2.8. The Windows client – ReconBotClient. It was written using the same technologies involved in the development of the main server, namely C# and .NET Framework. The client is comprised of four main classes:

- TCPClient – which contains the logic necessary to establish a connection to the main server, and to communicate with it
- DBConnection – which contains the logic necessary to retrieve data from the database server, and to display it
- UI and StoredDataUI – which contain the logic necessary for the main interface, and for the interface which displays data stored on the database server by using graphs and tables

To display the camera feed, we use GeckoFX, a library for C# applications that allows us to use the Gecko browser engine [37], which was developed by Mozilla, being the cornerstone of the popular Firefox web browser. The engine was built with performance in mind, and it works with most open web standards [38].

4. Using the solution and experimental results

The main use cases for the application are depicted in Figure 2:

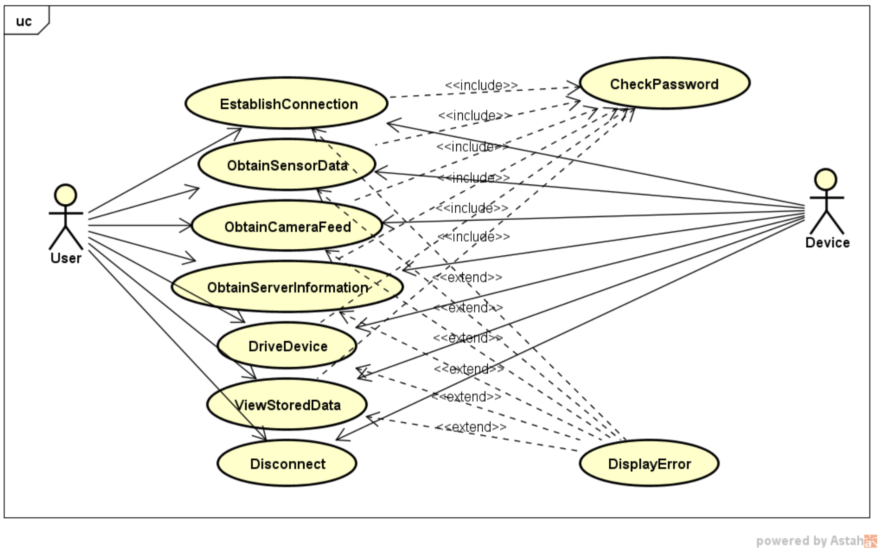
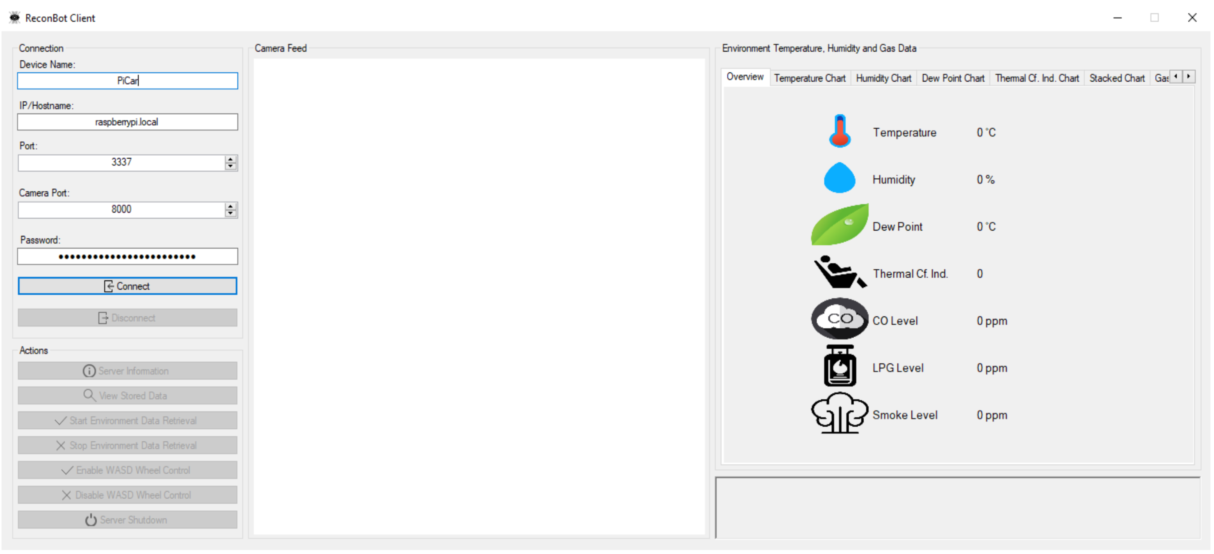


Figure 2. Application use cases. Source: Authors' processing using Astah.

In order to establish remote client connections to the device (connections from different networks, the device not being part of those networks), if the device is connected to the internet via a router, we need to enable port forwarding for the respective router on the ports which will be used to accept connections and communication from clients [4]. For the MariaDB server, the port is usually 3306. The port necessary for the web application which will serve the camera feed is 8000, and the port for the main server (ReconBotMaster) is 3337.

In order to connect directly to the Raspberry Pi board, we can use the SSH protocol [39], and an application such as Hyper [40]. After successfully logging in, in the "*master*" directory of the application, we can find the "*logs*" folder. This folder contains logs generated by the main server, which can be useful to identify the clients which connected successfully, and what their requests were. We can also manage the system services we talked about in Section 3 using "*system*".

The main user interface (UI) of ReconBotClient is depicted in Figure 3. After filling in the necessary fields, by pressing "Connect" a connection attempt is carried out.


ReconBot Client

Connection

Device Name: PiCar

IP/Hostname: raspberrypi.local

Port: 3337

Camera Port: 8000

Password: ••••••••••••••••••••••

Connect

Disconnect

Actions

Server Information

View Stored Data

Start Environment Data Retrieval

Stop Environment Data Retrieval

Enable WASD Wheel Control

Disable WASD Wheel Control

Server Shutdown

Camera Feed

Environment Temperature, Humidity and Gas Data

Overview | Temperature Chart | Humidity Chart | Dew Point Chart | Thermal Cf. Ind. Chart | Stacked Chart | Gas

Temperature	0 °C
Humidity	0 %
Dew Point	0 °C
Thermal Cf. Ind.	0
CO Level	0 ppm
LPG Level	0 ppm
Smoke Level	0 ppm

Figure 3. Main UI for ReconBotClient in a disconnected state. Source: Authors' contribution.

After a successful connection, by pressing "*Server Information*" we can gather main server information such as uptime, version, and the number of connected clients. To start or stop retrieving data from the sensors, the "Start Environment Data Retrieval" and "*Stop Environment Data Retrieval*" buttons are used. Also, the camera transmission will be displayed in the "*Camera Feed*" section of the main UI, the regions where motion is detected being marked with red. The timestamp is also displayed.

The "Gas Chart" tab displays a graph with data originating from the MQ-2 sensor, an example being depicted in Figure 4. The chart can be saved in .png format, by pressing "Save Chart", the picture in this case being saved in the "charts/gas" directory of the application.

The "Humidity Chart", "Dew Point Chart" tabs, as well as the other tabs work in an identical way.

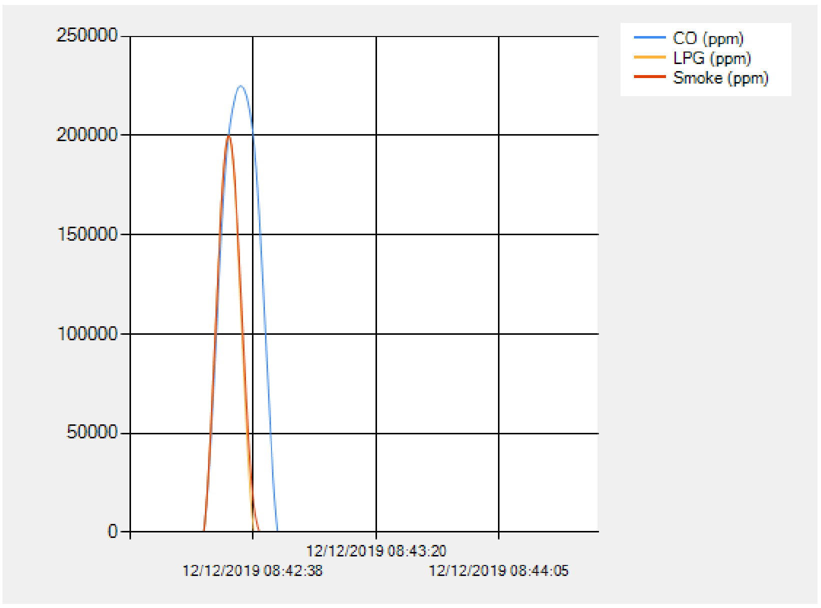


Figure 4. Values from the MQ-2 sensor. Source: Authors' contribution.

In order to drive the platform, we need to enable the functionality first, by pressing the "*Enable WASD Wheel Control*" button. This way, by pressing the W, A, S and D keyboard keys we can move the device in the following directions: forward (W), left (A), reverse (S) and right (D).

Another focal point of the client is the "*View Stored Data*" button. By pressing it, a new window will be opened, depicted in Figure 5.

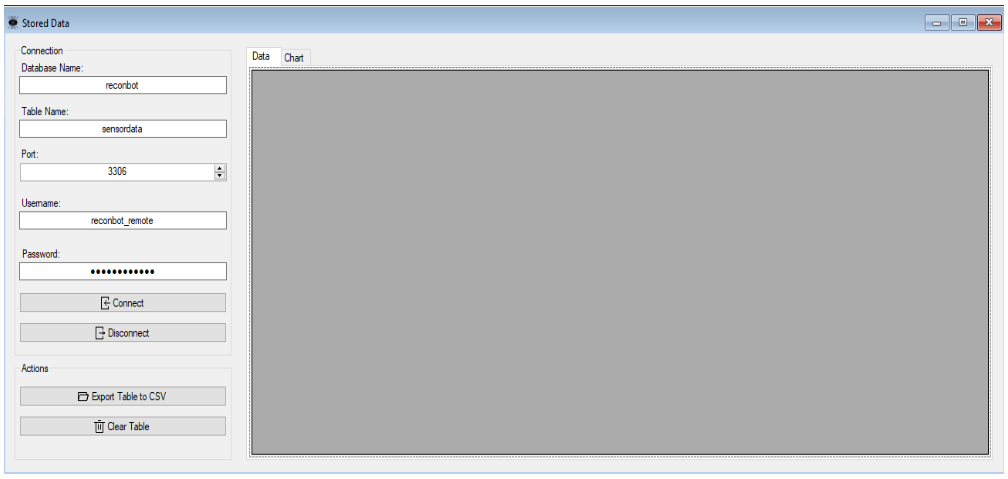
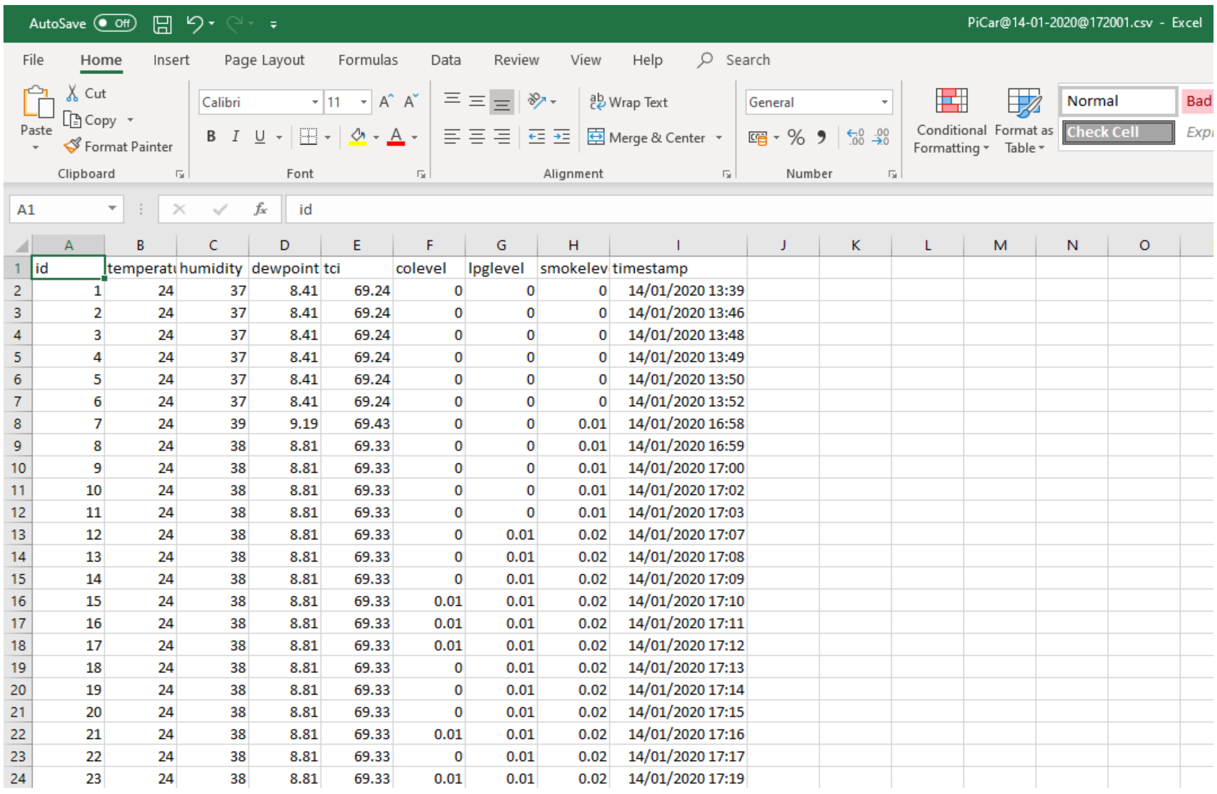


Figure 5. Stored Data UI in a disconnected state. Source: Authors' contribution.

After filling in the necessary information, a connection to the database server located on the device is established. We will now have access to the sensor data that is being collected and stored constantly, a behavior that we talked about in Section 3, point 3.2.5.

After establishing a successful connection, the "*Data*" tab will display a grid containing the sensor data that we talked about previously. The grid data can be saved as a .csv file [41], in the csv directory of the application. Figure 6 depicts a .csv file that the program was able to generate.



	A	B	C	D	E	F	G	H	I
1	id	temperatu	humidity	dewpoint	tci	colevel	lpglevel	smokelev	timestamp
2	1	24	37	8.41	69.24	0	0	0	14/01/2020 13:39
3	2	24	37	8.41	69.24	0	0	0	14/01/2020 13:46
4	3	24	37	8.41	69.24	0	0	0	14/01/2020 13:48
5	4	24	37	8.41	69.24	0	0	0	14/01/2020 13:49
6	5	24	37	8.41	69.24	0	0	0	14/01/2020 13:50
7	6	24	37	8.41	69.24	0	0	0	14/01/2020 13:52
8	7	24	39	9.19	69.43	0	0	0.01	14/01/2020 16:58
9	8	24	38	8.81	69.33	0	0	0.01	14/01/2020 16:59
10	9	24	38	8.81	69.33	0	0	0.01	14/01/2020 17:00
11	10	24	38	8.81	69.33	0	0	0.01	14/01/2020 17:02
12	11	24	38	8.81	69.33	0	0	0.01	14/01/2020 17:03
13	12	24	38	8.81	69.33	0	0.01	0.02	14/01/2020 17:07
14	13	24	38	8.81	69.33	0	0.01	0.02	14/01/2020 17:08
15	14	24	38	8.81	69.33	0	0.01	0.02	14/01/2020 17:09
16	15	24	38	8.81	69.33	0.01	0.01	0.02	14/01/2020 17:10
17	16	24	38	8.81	69.33	0.01	0.01	0.02	14/01/2020 17:11
18	17	24	38	8.81	69.33	0.01	0.01	0.02	14/01/2020 17:12
19	18	24	38	8.81	69.33	0	0.01	0.02	14/01/2020 17:13
20	19	24	38	8.81	69.33	0	0.01	0.02	14/01/2020 17:14
21	20	24	38	8.81	69.33	0	0.01	0.02	14/01/2020 17:15
22	21	24	38	8.81	69.33	0.01	0.01	0.02	14/01/2020 17:16
23	22	24	38	8.81	69.33	0	0.01	0.02	14/01/2020 17:17
24	23	24	38	8.81	69.33	0.01	0.01	0.02	14/01/2020 17:19

Figure 6. CSV file example. Source: Authors' contribution.

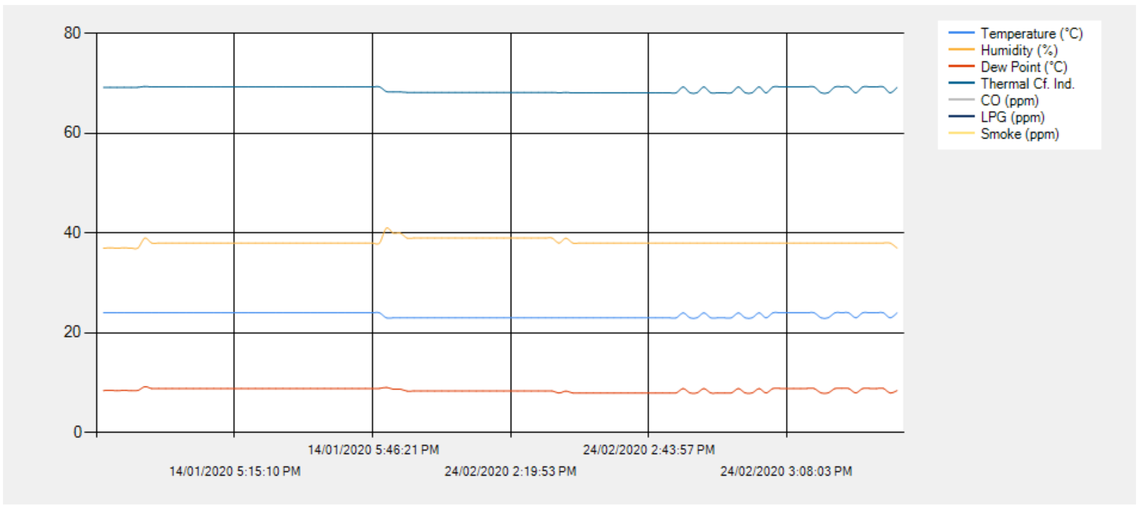


Figure 7. Stored Data graph, data collected in a clean room. Source: Author's contribution

The "*Chart*" tab displays a graph that contains the grid data. The graph can be saved, Figure 7 depicting a graph that was saved as a .png file.

Coming back to the main UI, the "*Server Shutdown*" button will prompt the main server to run a script which will in turn shut down the board.

5. Conclusion

Our solution makes use of hardware that is readily available, at a reasonable cost, and that is well supported by entities such as the Raspberry Pi Foundation and the enthusiast community[4].

A great benefit of the mobility aspect of the solution is that it can protect its users from dangerous situations, with the device being driven remotely to key areas of interest. Thus, it can be used for example in military scenarios to scout dangerous areas, with chemical activity, that pose a threat to human health. Another key element is the motion detection capability of the solution, which will prove crucial in surveillance and recognition scenarios, with users obtaining an accurate representation of the surrounding environment.

On the software side of things, the application is written using well documented languages and technologies, respecting programming principles.

As with all things, there is room for improvement. On a hardware level, one improvement could be waterproofing the device. This would allow it to work in a more hostile environment. On the software side, maybe a client for the Android operating system [42] could prove useful, to cover a larger array of devices which could run the client.

References

[1] Pfluger, R., Feist, W., Tietjen, A. and Neher, A., 2013. Physiological impairments of individuals at low indoor air humidity. [online] Gefahrstoffe Reinhaltung der Luft. Available at: https://passipedia.org/_media/picopen/ low_humidity.pdf [Accessed 22 Apr. 2019].

[2] Silva, D.C. and Passini, R., 2017. Physiological responses of dairy cows as a function of environment in holding pen. [online] Engenharia Agrícola, 37(2), pp.206-214. Available at: http://www.scielo.br/scielo.php?script=sci_arttext&pid= S0100-69162017000200206 [Accessed 22 Apr. 2019].

[3] Xu, Y., Yan, B. and Tang, J., 2015. The effect of climate change on variations in dew amount in a paddy ecosystem of the Sanjiang Plain, China. [online] Advances in Meteorology. Available at: https://www.hindawi.com/journals/amete/2015/793107/ [Accessed 22 Apr. 2019].

[4] Petac, E. and Constantin, A., 2019. IoT Embedded System for Environmental Monitoring. [online] Available at: http://stec.univ-ovidius.ro/html/anale/RO/wp-content/uploads/2019/08/21-1.pdf [Accessed Nov. 2019].

[5] Centers for Disease Control and Prevention, 2018. Carbon Monoxide Poisoning. [online] Available at: https://www.cdc.gov/co/faqs.htm [Accessed Nov. 2019].

[6] Lenntech B.V., 2008. Parts per Million by Volume (or mole) in Air. [online] Available at: https://www.lenntech.com/calculators/ppm/converter-parts-per-million.htm [Accessed Nov. 2019].

[7] DetectCarbonMonoxide.com, 2019. CO Health Risks. [online] Available at: https://www.detectcarbonmonoxide.com/co-health-risks/ [Accessed Nov. 2019].

[8] Centers for Disease Control and Prevention, 2014. Table of IDLH Values. [online] Available at: https://www.cdc.gov/niosh/idlh/68476857.html [Accessed Nov. 2019].

[9] Raspberry Pi Foundation, 2019. Raspberry Pi 3 Model B. [online] Available at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ [Accessed Nov. 2019].

[10] DUAL FULL-BRIDGE DRIVER L298N. [online] Available at: https://www.smart-prototyping.com/image/data/9_Modules/101861%20LN298N%20dual%20H-bridge%20driver%20motor/L298DATASHEET.pdf [Accessed Nov. 2019].

[11] Learn.adafruit.com, 2019. DHT temperature & humidity sensors. [online] Available at: https://learn.adafruit.com/dht/overview [Accessed 24 Apr. 2019].

[12] COMPONENTS101, 2018. MQ2 Gas Sensor. [online] Available at:

https://components101.com/mq2-gas-sensor [Accessed Nov. 2019].
[13] DiCola, T., 2016. MCP3008. [online] Available at: https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008 [Accessed Nov. 2019].
[14] Brainy Bits, 2018. Use 3.3V modules safely using a Logic Level Converter. [online] Available at: https://www.brainy-bits.com/what-is-a-logic-converter-arduino/ [Accessed Nov. 2019].
[15] Petac E., 2015. Networks and Distributed Systems, Chapter 4, pp.103-206, Into the Book: Distributed Multimodal Virtual Environments, vol.IV, Publisher: Pro Universitaria Bucharest, Romania.
[16] Oluwatosin, H.S., 2014. Client-Server Model. [online] Available at: https://www.researchgate.net/profile/Shakirat_Sulyman/publication/271295146_Client-Server_Model/links/5864e11308ae8fce490c1b01/Client-Server-Model.pdf [Accessed Nov.2019].
[17] Nagel, C., 2018. Professional C# 7 and .NET Core 2.0. 7th ed. New York: John Wiley & Sons Inc.
[18] GoalKicker.com, 2018. .NET Framework Notes for Professionals. [online] Available at: https://books.goalkicker.com/DotNETFrameworkBook/ [Accessed 20 Apr. 2019].
[19] Jaworski, M. and Ziade, T., 2016. Expert Python Programming. Birmingham, UK: Packt Publishing Ltd.
[20] McNoldy, B. D., 2015. Calculate Temperature, Dewpoint, or Relative Humidity. [online] University of Miami. Available at: http://bmcnoldy.rsmas.miami.edu/Humidity.html [Accessed 24 Apr. 2019].
[21] Teodoreanu, E., 2016. THERMAL COMOFORT INDEX. [online] Available at: https://www.researchgate.net/publication/311356785_Thermal_Comfort_Index [Accessed Nov. 2019].
[22] GeeksforGeeks, 2019. OpenCV – Overview. [online] Available at: https://www.geeksforgeeks. org/opencv-overview/ [Accessed Nov. 2019].
[23] Rosebrock, A., 2019. OpenCV – Stream video to web browser/HTML page. [online] Available at: https://www.pyimagesearch.com/2019/09/02/opencv-stream-video-to-web-browser-html-page/ [Accessed Nov. 2019].
[24] Gupta P., Gupta M. and Singh, Y., 2014. Moving Object Detection Using Frame Difference, Background Subtraction And SOBS For Video Surveillance Application. [online] Available at: http://tmu.ac.in/college-of-computing-sciences-and-it/wp-content/uploads/sites/17/2016/10/T208.pdf [Accessed Nov. 2019].
[25] Gartner, Inc., 2019. Internet of Things. [online] Available at: https://www.gartner.com/it-glossary/internet-of-things/ [Accessed 24 Apr. 2019].
[26] Raspberry Pi Foundation, 2019. GPIO. [online] Available at: https://www.raspberrypi.org/documentation/usage/gpio/ [Accessed 24 Apr. 2019].
[27] Tutorials-RaspberryPi, 2016. Configure and read out the Raspberry Pi gas sensor (MQ-X). [online] Available at: https://tutorials-raspberrypi.com/configure-and-read-out-the-raspberry-pi-gas-sensor-mq-x/ [Accessed Nov. 2019].
[28] Raspberry Pi Foundation, 2019. Camera PI Noir (Infrared) Raspberry PI version2 2. [online] Available at: https://www.raspberrypi.org/documentation/hardware/camera/ [Accessed Nov. 2019].
[29] Github, Inc., 2019. DHT11 Python library. [online] Available at: https://github.com/ szazo/DHT11_Python [Accessed 24 Apr. 2019].
[30] Github, Inc., 2019. Raspberry Pi Gas Sensor MQ Python Example. [online] Available at: https://github.com/tutRPi/Raspberry-Pi-Gas-Sensor-MQ [Accessed Nov. 2019]
[31] Mono-project.com, 2019. Mono. [online] Available at: https://www.mono-project.com/ [Accessed 24 Apr. 2019].
[32] Gueron, S., Langley, A. and Lindell, Y., 2017. AES-GCM-SIV: Specification and Analysis, Report 2017/168. [online] Cryptology ePrint Archive. Available at: https://eprint.iacr.org/ 2017/168.pdf [Accessed 22 Apr. 2019].
[33] Iuorio, A. F. and Visconti, A., 2018. Understanding Optimizations and Measuring Performances of PBKDF2. [online] International Conference on Wireless Intelligent and Distributed Environment

for Communication, Springer, pp. 101-114. Available at: https://eprint.iacr.org/2019/161.pdf [Accessed 22 Apr. 2019].

[34] Github, Inc., 2019. MariaDB: drop-in replacement for MySQL. [online] Available at: https://github.com/MariaDB/server [Accessed Nov. 2019].

[35] Ronacher, A., 2019. Flask. [online] Available at: https://palletsprojects.com/p/flask/ [Accessed Nov. 2019].

[36] Debian Project, 2019. systemd. [online] Available at: https://wiki.debian.org/systemd [Accessed 24 Apr. 2019].

[37] Microsoft Corp., 2018. Geckofx45. [online] Available at: https://www.nuget.org/packages/Geckofx45/ [Accessed Nov. 2019].

[38] Mozilla Foundation, 2019. Gecko. [online] Available at: https://developer.mozilla.org/en-US/docs/Mozilla/Gecko [Accessed Nov. 2019].

[39] Ylonen, T. and Lonvick, C., 2006. The Secure Shell (SSH) Protocol Architecture. [online] Available at: https://www.hjp.at/doc/rfc/rfc4251.html [Accessed Nov. 2019].

[40] ZEIT, Inc., 2019. Hyper. [online] Available at: https://hyper.is/ [Accessed Nov. 2019].

[41] Shafranovich, Y., 2005. Common Format and MIME Type for Comma-Separated Values (CSV) Files. [online] Available at: https://www.hjp.at/doc/rfc/rfc4180.html [Accessed Nov. 2019].

[42] Google LLC, 2019. Build anything on Android. [online] Available at: https://developer.android.com/ [Accessed Nov. 2019].