IoT Embedded System for Environmental Monitoring

Eugen Petac
Alexandru Constantin
"Ovidius" University of Constanța, Faculty of Mathematics and Computer Science
epetac@univ-ovidius.ro
constantin.alex96@icloud.com

Abstract

Environmental monitoring aims to provide an objective image as close as possible to reality. This paper proposes an IoT embedded system for temperature, humidity and dewpoint remote monitoring. The main contribution of this paper is in providing a robust and reliable multithreaded client-server application called THbot. Our solution offers security and reliability for data storage and transmission, authentication, and architecture support for cloud integration. The management of the IoT embedded system is done safely and securely. The results of our THbot solution can be easily integrated into environmental control applications.

Key words: Environmental Monitoring, IoT, Embedded System, Dewpoint
J.E.L. classification: L8, M1, Q5

1. Introduction

Weather affects the human body (Lee et al., 2018), which clearly notices the impact of sudden temperature and humidity changes. For values lower or higher than normal of the temperature-humidity index, which is also called thermal comfort index, cellular-level changes occur, and the overall health status of the human body is altered (Mora et al., 2018). Besides those extreme temperature values, the dewpoint is used to evaluate the thermal comfort index (Pfluger, Feist and Neher, 2013). The dewpoint corresponds to the temperature at which the water vapor concentration in the air is saturated. Weather affects us, humans, but affects animals (Silva and Passini, 2017) and plants (Xu, Yan and Tang, 2015) as well. For a correct diagnosis, specialized literature recommends gathering data regarding the state of the body or the state of the plants related to the weather state. Thus, indoor and outdoor environmental monitoring over a time period becomes a major concern for the quality of life (Jayaratne et al., 2018).

In this paper we propose a robust system called THbot, with THbot_Server and THbot_Client parts, as an IoT embedded system for temperature, humidity and dewpoint remote monitoring. As an Internet of Things (IoT) embedded system (Vermesan et al., 2018), the hardware part of THbot_Server includes the Single Board Computer (SBC) Raspberry Pi Zero WH (Adafruit.com, 2019), the DHT11 humidity and temperature sensor (Learn.Adafruit.com, 2019), and a 0.96-inch OLED screen based on the SSD1306 chip. The screen displays the temperature and humidity values and it can be used as an alternative for the client application when we are situated in the proximity of the SBC. Our THbot software application, presented in Section 3, is a distributed client-server application developed using the C# and Python programming languages and the .NET Framework. This implements a remote monitoring system for temperature, humidity and dewpoint values. The application offers security and reliability for data storage and transmission, authentication, and architecture support for cloud integration (Jassas et al., 2017). The application's main achievements are the stability and reliability of the transmitted data regardless of the geographical position of the servers and clients, and also the ease of use. The THbot system 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

According to Gartner "The Internet of Things (IoT) is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment" (2019). Each object that is part of the IoT system is equiped accordingly, and can send and receive data. The object must be capable of capturing data, usually through sensors, and must be capable of sending the captured data through the internet. IoT has become a network of billions of smart devices which connects people, systems and other applications together, in order to collect, process and share data.

Created by Microsoft, C# (Nagel, 2018) is a object-oriented programming language, suitable for developing web-based and desktop applications. Derived from the C and C++ programming languages, including influences from other languages, most notably Java, C# allows for developing robust industrial-grade applications, which work under a variety of operating systems, being adequate for systems which target real-world applications as well.

The most important features that recommend C# for such applications are: features that enable direct implementation of software components, such as properties, methods and events; the possibility of working in an environment with multiple languages; automated management for used system memory; the possibility of using features and APIs which belong to the operating system.

.NET Framework (GoalKicker.com, 2018) is an environment that allows for developing and running applications and web services in a cross-platform manner. The automated memory management, the interoperability of the languages and the security and portability of the applications are the main features of the architecture of this environment. .NET Framework has two main components, Common Language Runtime (CLR) and Base Class Library (BCL). CLR is the execution environment of the applications, which also handles memory management and exceptions. BCL covers a large area of programming work, including user interfaces, connections to databases, web application development, network communcations and others. The code of the library is precompiled, being encapsulated in methods, which programmers can call in their own programs. In turn, methods belong to classes, and classes are organized in namespaces. To create applications, programmers combine their own code with BCL code.

Created by programmer Guido van Rossum, Python (Jaworski and Ziade, 2016) is a multipurpose, portable, interpreted, high-level programming language, which makes use of the object-oriented programming paradigm, and allows for imperative, functional or procedural programming. Also, it is a dynamically typed language, which combines coding power and a clean syntax. The Python API provides many modules for a large number of functionalities, from basic functionalities such as string and file handling, to complex functionalities such as processes and thread handling, sockets, serializations, and many more. Python is used in embedded systems, automatization, web applications, artificial intelligence, data analysis, etc.

Healthcare, hydro-climatic and agro-climatic researches, tourism (Scott and Lemieux, 2010), civil buildings (Kharseh et al., 2017) and the automotive industry (Pillmann et al., 2017), are some fields of activity that have major interest in monitoring temperature, humidity and dewpoint values. The dewpoint corresponds to the temperature at which the water vapor concentration in the air is saturated. The dewpoint's value depends mostly on the relative humidity and temperature of the environment. To obtain the dewpoint temperature, denoted with TD and expressed in Celsius degrees, we used the following relationship (McNoldy, 2015):

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

where:
T - stands for the environment temperature in Celsius degrees;
RH – stands for relative humidity in percentages;
LN – stands for natural logarithm.

3. Research methodology

Our THbot research system is a distributed client-server application developed using the C# and Python programming languages and the .NET framework.

Named THbot_Server, the server side of the application allows for local and remote connections. Opting for the multithreaded programming paradigm, the overall speed of the application is improved, and system resources are optimally used. The use of multiple threads allows for the parallelization of the application tasks. The Raspberry Pi Zero WH board uses the Raspbian operating system. Raspbian is an open-source Debian-based Linux operating system, its main advantages being its low system resources usage and the official support from the Raspberry Pi Foundation. The *systemd* system manager (Debian.com, 2019), which comes preinstalled on Raspbian, provides parallelization capabilities as well as the possibility of configuring and monitoring system services.

THbot_Client is the client side of the application. Developed as a client for the Windows operating system, THbot_Client allows connections to multiple servers. It is capable of retrieving data from THbot_Server, to process them, to create graphs and send notifications to the users. For a reliable communication between THbot_Client and THbot_Server, the application is based on Transmission Control Protocol (TCP). To secure the communication between THbot_Server and THbot_Client the AES-GCM algorithm is used, which stands for Advanced Encryption Standard (AES) in GCM mode (Galois/Counter Mode). THbot_Client can run in a private IoT cloud, local or remote connection.

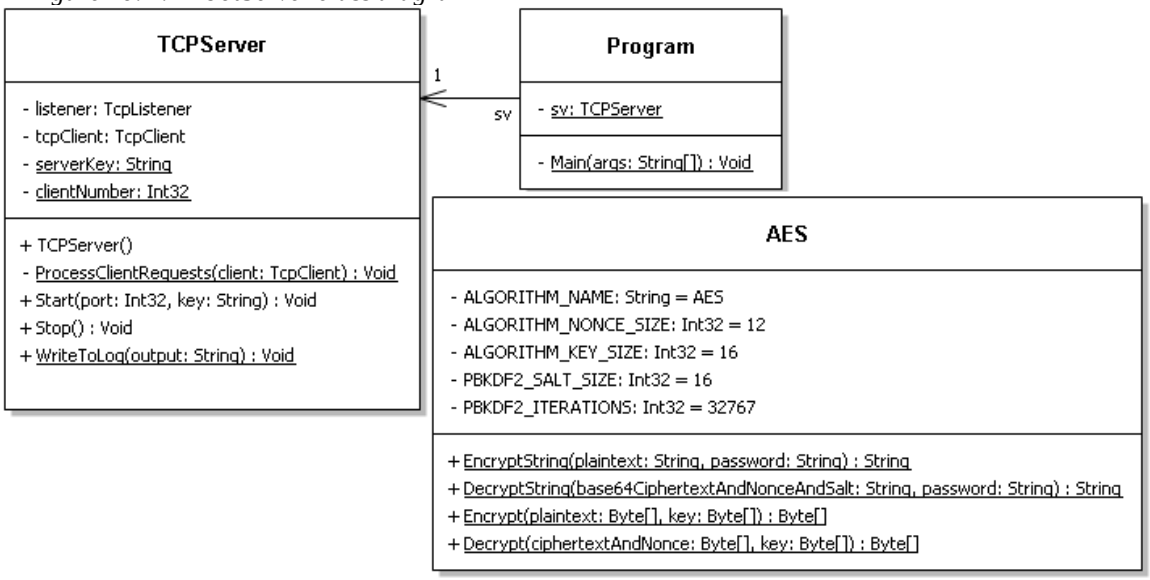
3. Results

3.1. THbot Server

After the installation and configuration of the Raspbian operating system, the SBC is ready to host the server application. It has three main components:

- The THbot.py Python script that reads and displays sensor data.
- The THbotServer.exe server application, written in C#, that allows for new client connections and processes their requests. Figure 1 represents the class diagram of the server application.
- The server system service, which allows for starting, stopping and restarting the server.

Figure no. 1. THbotServer class diagram



Source: Authors' processing using *NClass*

3.1.1. THbot.py

The script THbot.py uses the DHT11 Python library which enables communication with the sensor in order to retrieve temperature and humidity values. The commands *"sudo apt-get update"* and *"sudo apt-get install rpi.gpio"* once run, install the RPi. GPIO package which is mandatory for the DHT11 library to work.

After the installation of all required packages and libraries, and after the initialization of the GPIO pins, we are able to read temperature and humidity values from the GPIO24 pin. Data retrieval is performed once every one second, considering the technical specifications of the DHT11 sensor (Mouser.com, 2019).

3.1.2. THbotServer.exe

The THbotServer application is written on a machine that runs the Windows operating system. It is a console application, which utilizes the .NET framework and the C# programming language. THbotServer generates logs for every important event. After a client connection, a new thread is created, on which the server will process messages coming from the client and send response messages accordingly. Further, a description of the application will be presented.

3.1.2.1 The TCPServer class

The *Start* method allows the server to accept new connections from clients. The passed parameters are the port number on which the server will listen for new connections and the encryption/decryption password of the sent and received messages, which also works as an authentication password, being only known by the owner of the server and the clients that wish to connect. The *ProcessClientRequests* method analyses messages from each client individually.

The following messages are the ones the server is looking after:

"!connection" – the server responds with a confirmation message that the connection was established successfully;

"!temp" – the application will open a new process in which it will obtain the result displayed by the THbot.py script, and the result will be send to the client. This message marks the fact that the client wanted to obtain the sensor values at a given time;

"!info" – the server sends a message regarding its version, uptime and the number of connected clients;

"!exit" – this message tells the server that a client wishes to disconnect.

When a message received from a client can't be decrypted successfully, which means an unknown encryption/decryption password and implicitly a wrong connection password was used, the authentication is unsuccessful, and the server closes the connection with the client. The *Stop* method allows the server to be stopped. The *WriteToLog* method is used to generate logs for important server events.

3.1.2.2 The AES class

The AES class makes use of the Bouncy Castle API, which provides access to encryption algorithms and other encryption-related resources. The sent and received messages are crypted and decrypted using the *EncryptString* and *DecryptString* methods. The Advanced Encryption Standard (AES) algorithm is used, in Galois/Counter Mode (GCM) mode (Gueron, Langley and Lindell, 2017). Each plain text message is divided into 128-bit blocks, called generic data. If the last block is incomplete, it will be padded. If the message is divided in uniform 128-bit blocks, a dummy 128-bit block will be appended. For client-server communication, AES-GCM provides three security services, confidentiality, integrity, and authentication. The passwords used by the clients and the server are derived into secure keys using PBKDF2 (Password-Based Key Derivation Function 2), a password hashing algorithm (Iuorio and Visconti, 2018).

3.1.2.3 The Program class

In order to launch the THbotServer application the Program class is used. From the *"server.cfg"* file which is located in the same directory as the application's executable file, the port number and the encryption/decryption password are obtained, then the server is started. The password must have between 15 and 30 characters in length.

In order to run the THbotServer.exe executable on *Raspbian*, the Mono software platform (Mono-project.com, 2019) was used, which helps in developing cross-platform applications using the C# programming language and .NET Framework. Mono can be installed on Raspbian using the *"sudo apt-get install mono-complete"* command, and allows applications created on Windows to run on operating systems that belong to the Linux family without needing to develop a special version dedicated to Linux.

3.1.3 Running the sever as a system service

In order to run THbot Server as a system service, *systemd* was used. This is a system and service manager for Linux-based operating systems. With *systemd* we can start, stop and restart THbotServer, and most importantly, we can run THbotServer automatically after the system starts.

The *THbotServer.service* file contains the necessary instructions. The commands that copy the *THbotServer.service* file in the *systemd* directory and update the list of services managed by systemd are *"sudo cp THbotServer.service /etc/systemd/system"* and *"sudo systemctl daemon-reload"* respectively and can be found in the *install_app_services.sh* bash script, which installs the system services for both the server and the screen.

To make server interaction easier, the following bash scrips were created, which have the role to manage THbot Server:

enable_startup.sh – the THbot Server service will automatically start after the system starts after running this script;

restart_server.sh – restart the server service;

server_status.sh – check server status and displayed messages;

start_server.sh – start the server service;

stop_server.sh – stop the server service.

Every script has its permisssions set up by the following command: *"chmod u+x <script_name>.sh"*.

3.1.4 Displaying values on the screen

In order to display temperature, humidity and dewpoint values on the screen and read them when located in the SBC's proximity, an OLED screen was used, which boasts a 0.96-inch diagonal and a resolution of 128x64 pixels. This screen is based on the SSD1306 chip and provides an I2C connection, low energy consumption (0.08W max) and a very good color contrast. The advantage which this solution provides is that there is no need for a client application when located in the SBC's proximity. The Python script that displays values on the screen is called screen.py and makes use of the Adafruit-SSD1306 library to display text on screen, via the I2C connection. To obtain sensor data the DHT11 library is used. The dewpoint temperature is calculated using the formula that was presented earlier. To run screen.py as a system service, a similar approach to THbot Server was used, by creating the *THbotScreen.service* file, registering the *THbotScreen* service, and by creating the *screen_enable_startup.sh*, *start_screen.sh*, *stop_screen.sh*, *restart_screen.sh* and *screen_status.sh* bash scripts, and finally setting the permissions of these scripts.

3.2 Thbot_Client

Written using the C# programming language and the .NET framework, Thbot_Client allows for connections to Thbot_Server, which provides temperature, humidity and dewpoint values. Thbot_Client has as its main classes the TCPClient class and the AES class which is identical to the AES class found in the server's code, Section 3.1.2.2.

3.2.1 The TCPClient class

Values passed to the constructor of the class are the IP address and port number of the server, as well as password for the messages transmitted between the client and server. In order to retrieve the data provided by the server, the *GetData* method is used after the connection to the server is established successfully by using the *ConnectionConfirmation* and *GetServerVersion* methods. The *GetServerVersion* method allows the client to retrieve server information such as software

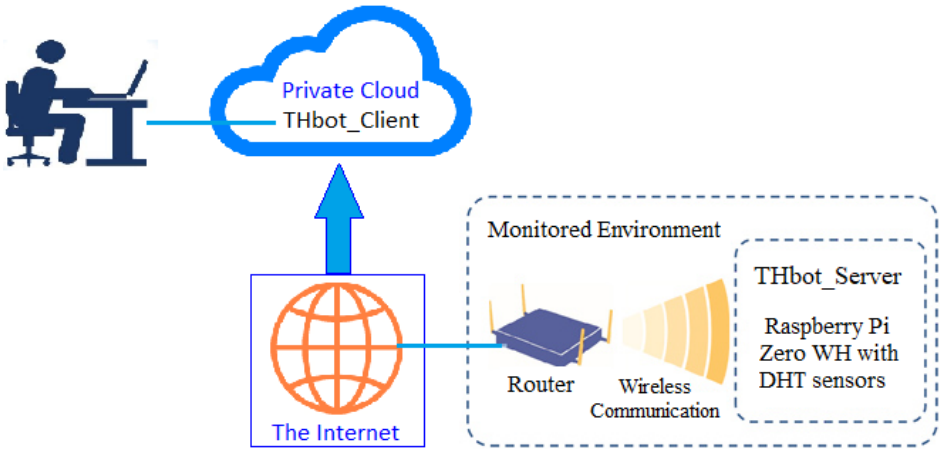
version, server uptime and the number of connected clients. The methods which belong to the TCPClient class are used in the UI class, which displays the graphical user interface.

4. Experimental Results

4.1 Running THbot_Server

For the SBC (Raspberry Pi Zero WH) which hosts THbot_Server and is connected to the internet through a router (Figure 2), we need to enable port forwarding on the port which will be used to accept connections and messages from clients. THbot_Server is configured to run as a system service.

Figure no. 2. THbot Data communication



Source: Authors' contribution

There is the possibility of starting, stopping and restarting the server manually, using the scripts mentioned in Section 2. For this, a SSH connection to the SBC is needed, which can be achieved with the Hyper or PuTTY applications. The *server_status.sh* script allows us to check the status of the server (Figure 3).

Figure no. 3. Status of the THbotSever

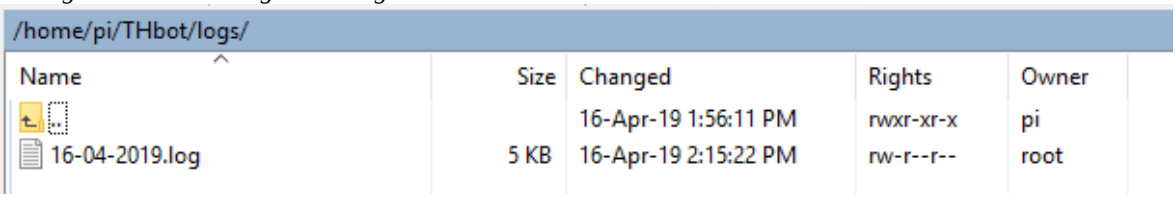
```
pi@raspberrypi:~ $ cd THbot
pi@raspberrypi:~/THbot $ ./server_status.sh
● THbotServer.service - THbotServer
   Loaded: loaded (/etc/systemd/system/THbotServer.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2019-04-23 14:36:43 EEST; 22h ago
 Main PID: 14475 (Main)
   CGroup: /system.slice/THbotServer.service
           └─14475 /usr/bin/mono /home/pi/THbot/THbotServer.exe

Apr 23 14:37:26 raspberrypi mono[14475]: [23/04/2019 14:37:26] From client (192.168.0.103) > Temp & humidity data request
Apr 23 14:37:29 raspberrypi mono[14475]: [23/04/2019 14:37:29] THbot: Temperature: 27 C @ Humidity: 32 % @ Dew Point: 8.9 C
Apr 23 14:37:36 raspberrypi mono[14475]: [23/04/2019 14:37:36] Closing client (192.168.0.103) connection!
Apr 23 17:34:00 raspberrypi mono[14475]: [23/04/2019 17:34:00] Accepted new client connection: 192.168.0.103
Apr 23 17:34:00 raspberrypi mono[14475]: [23/04/2019 17:34:00] Waiting for client connections...
Apr 23 17:34:00 raspberrypi mono[14475]: [23/04/2019 17:34:00] From client (192.168.0.103) > Connection confirmation
Apr 23 17:34:00 raspberrypi mono[14475]: [23/04/2019 17:34:00] From client (192.168.0.103) > Server information
Apr 23 17:34:02 raspberrypi mono[14475]: [23/04/2019 17:34:02] From client (192.168.0.103) > Temp & humidity data request
Apr 23 17:34:05 raspberrypi mono[14475]: [23/04/2019 17:34:05] THbot: Temperature: 27 C @ Humidity: 31 % @ Dew Point: 8.43 C
Apr 23 17:34:11 raspberrypi mono[14475]: [23/04/2019 17:34:11] Closing client (192.168.0.103) connection!
pi@raspberrypi:~/THbot $
```

Source: Authors' processing with *server_status.sh*

With the WinSCP application we can check server logs, which are located in the "logs" directory (Figure 4).

Figure no. 4. Viewing server logs with WinSCP

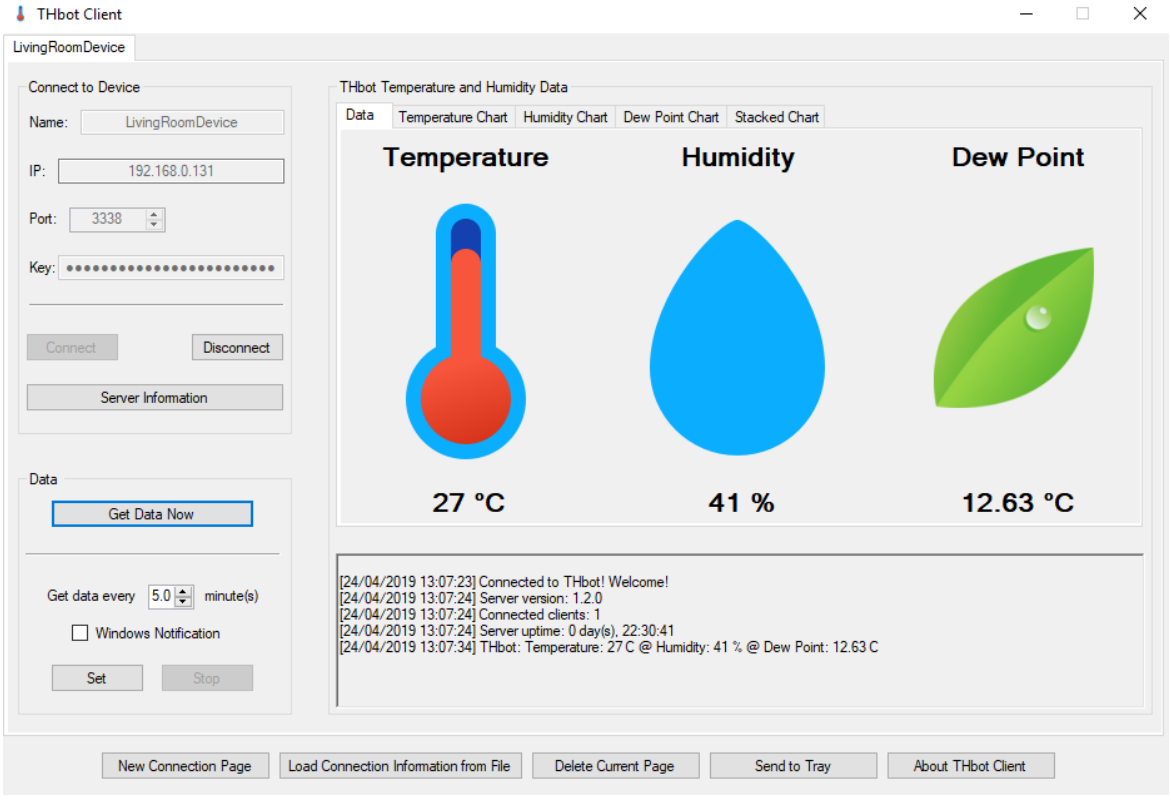


Source: Authors' processing with *WinSCP*

4.2 Running THbot_Client

The graphical User Interface (UI) of THbot_Client is depicted in Figure 5. After inserting the appropriate data in the corresponding fields, by pressing the *"Connect"* button a connection to the desired server is established. The *"Server Information"* button helps in obtaining server information such as software version, server uptime and the number of connected clients. To retrieve data regarding temperature, humidity and dew point temperature values, the *"Get Data Now"* button is used.

Figure no. 5. Main UI for the THbot Client



Source: Authors' contribution

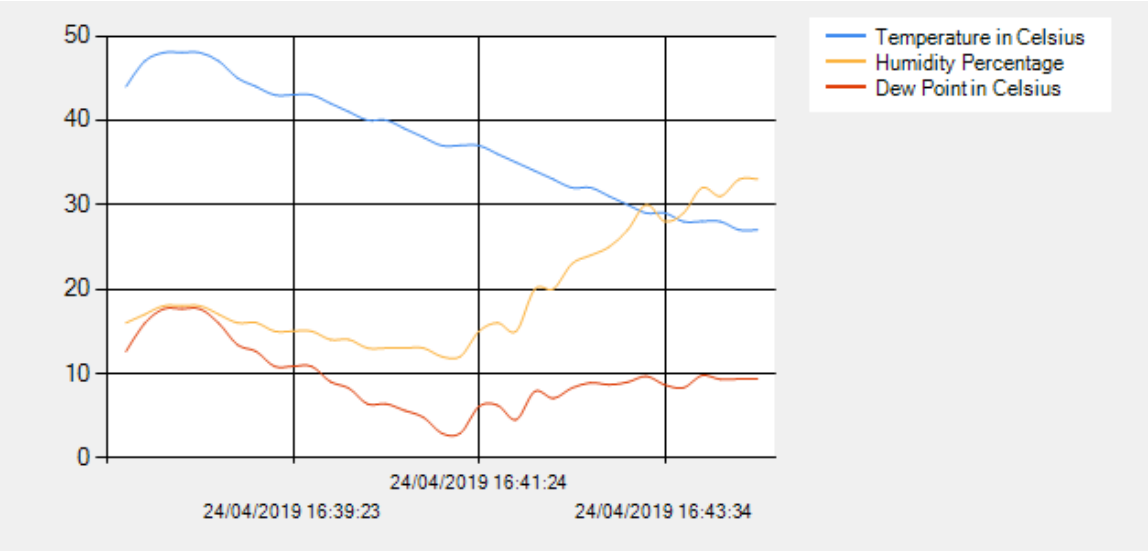
After selecting a time interval measured in minutes, pressing the*"Set"* button will yield value readings at the selected time interval, and notifications can be sent. The *"Temperature Chart"* tab displays a graph which can have different styles, column, spline or bar. The *"Save Chart"* button saves the chart as a .png file image which depicts the state of the graph when the button was pressed. An example graph is shown in Figure 6. The graph shows the relationship between the dewpoint temperature and the temperature and humidity percentage, over a period of time, according to the relationship mentioned in the 2.1.1 Section.

The *"Humidity Chart"*, *"Dew Point Chart"* and *"Stacked Chart"* tabs work in a similar way. Placed at the bottom of the UI, the *"New Connection Page"* button will open a new connection page, which allows for a new connection to a server.

The *"Load Connection Information from File"* button allows for adding new connection pages. These pages have the *"Name"*, *"IP"*, *"Port"* and *"Key"* fields automatically filled in with the information read from a text file chosen by the user.

The text file contents should comply with the following format: *<SBCDeviceName WithoutWhitespaces>[space]<ServerIP>[space]<PortNumber>[space]<Password>*. The *"Send to Tray"* button will send the application to background, and the application icon will be shown in tray.

Figure no. 6. Temperature, Humidity Percentage and Dewpoint Temperature curves over a period of time



Source: Authors' contribution

5. Conclusions and Future Work

Environmental monitoring is a decisive factor in maintaining a good quality of life and preserving materials. The THbot application uses hardware which is readily available at a reasonable cost, and that is well supported by entities like the Raspberry Pi Foundation and the enthusiast community. The main contribution of this paper is in providing a robust and reliable multithreaded client-server application. Our THbot software application is easily modifiable and extendable, and it's written with programming principles and readability in mind. The application achieves its goals, but the field of IT and programming constantly proposes improvements. One way of improving the application would be the addition of a gas and pressure sensor, to further improve the quality of the retrieved data. Another way of improving the application would be the addition of a CSI camera module, which would be helpful in monitoring movements, to further extend the functionality and feature set of the application.

6. References

- Adafruit.com, 2019. *Raspberry Pi Zero WH*, [online] Available at: https://www.adafruit.com/product/3708 [Accessed 24 Apr. 2019].
- Debian.com, 2019. *systemd*, [online] Available at: https://wiki.debian.org/systemd [Accessed 24 Apr. 2019].
- Gartner, Inc., 2019. *Internet of Things*, [online] Available at: https://www.gartner.com/it-glossary/internet-of-things/ [Accessed 24 Apr. 2019].
- GoalKicker.com, 2018.. *NET Framework Notes for Professionals*. [online] Available at: https://books.goalkicker.com/DotNETFrameworkBook/ [Accessed 20 Apr. 2019].
- 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].

- 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].
- Jassas, M., Mathew, J., Azim, A., and Mahmoud, Q. H., 2017. A framework for extending resources of embedded systems using the Cloud. [online] *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1-5. Available at: https://www.researchgate. net/publication/317723164_A_framework_for_extending_resources_of_embedded_systems_using_th e_Cloud [Accessed 22 Apr. 2019].
- Jaworski, M. and Ziade, T., 2016. *Expert Python Programming*. Birmingham, UK: Packt Publishing Ltd.
- Jayaratne, R., Liu, X., Thai, P., Dunbabin, M. and Morawska, L., 2018. The influence of humidity on the performance of a low-cost air particle mass sensor and the effect of atmospheric fog. [online] *Atmospheric Measurement Techniques*, 11(8), pp.4883-4890. Available at: https://www.atmos-meas-tech.net/11/4883/2018/amt-11-4883-2018.pdf [Accessed 22 Apr. 2019].
- Kharseh, M., Ostermeyer, Y., Nägeli, C., Kurkowska, I. and Wallbaum, H., 2017. Humid Wall: Review on Causes and Solutions. [online] *Conference Proceedings of World Sustainable Built Environment Conference 2017 Hong Kong*. Available at: http://wsbe17hongkong.hk/download/ WSBE17%20Hong%20Kong%20-%20Conference%20 Proceedings.pdf [Accessed 22 Apr. 2019].
- Learn.adafruit.com, 2019. *DHT temperature & humidity sensors*. [online] Available at: https://learn.adafruit.com/ dht/overview [Accessed 24 Apr. 2019].
- Lee, M., Ohde, S., Urayama, K., Takahashi, O. and Fukui, T., 2018. Weather and health symptoms. [online] *International journal of environmental research and public health*, *15*(8), p.1670. Available at: https://www.mdpi.com/ 1660-4601/15/8/1670 [Accessed 24 Apr. 2019].
- 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].
- Mono-project.com., 2019. *Mono*, [online] Available at: https://www.mono-project.com/ [Accessed 24 Apr. 2019].
- Mora, R. and Meteyer, M., 2018. Using Thermal Comfort Models in Health Care Settings: A Review. [online] *ASHRAE Transactions*, *124*. Available at: https://commons.bcit.ca/besys/files/ 2018/08/Thermal-comfort-health-care.pdf [Accessed 22 Apr. 2019].
- Mouser.com, 2019. *DHT11 Humidity & Temperature Sensor*, [online] Available at: https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf [Accessed 24 Apr. 2019].
- Nagel, C., 2018. *Professional C# 7 and .NET Core 2.0*. 7th ed. New York: John Wiley & Sons Inc.
- 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].
- Pillmann, J., Wietfeld, C., Zarcula, A., Raugust, T. and Alonso, D.C., 2017. Novel common vehicle information model (cvim) for future automotive vehicle big data marketplaces. [online] *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1910-1915. Available at: https://arxiv.org/ pdf/1802.09353 .pdf [Accessed 22 Apr. 2019].
- Scott, D. and Lemieux, C., 2010. Weather and climate information for tourism. [online] *Procedia Environmental Sciences*, 1, pp.146-183. Available at: https://www.researchgate.net/ publication/ 236018338_Weather_and_Climate_ Information_for_Tourism/ [Accessed 22 Apr. 2019].
- 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].
- Vermesan, O., Eisenhauer, M., Serrano, M., Guillemin, P., Sundmaeker, H., Tragos, E.Z., Valino, J., Copigneaux, B., Presser, M., Aagaard, A. and Bahr, R., 2018. The Next Generation Internet of Things–Hyperconnectivity and Embedded Intelligence at the Edge. In: O. Vermesan and J. Bacquet, ed., *Next Generation Internet of Things. Distributed Intelligence at the Edge and Human Machine-to-Machine Cooperation*, 1st ed. Delft: River Publishers, pp. 19-102
- 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].