World Journal of Advanced Research and Reviews

eISSN: 2581-9615 CODEN (USA): WJARAI

Cross Ref DOI: 10.30574/wjarr

Journal homepage: https://wjarr.com/

(Research Article)

# Design and implementation of a hybrid IoT system of data collection and visualization

Ahmed-Mouhamadou WADE *, Ibrahima GUEYE and Samba SIDIBE

*LTISI laboratory GIT, Polytechnic School of Thiès (EPT), Thiès, Sénégal.*

World Journal of Advanced Research and Reviews, 2022, 13(01), 754–762

Publication history: Received on 27 December 2021; revised on 29 January 2022; accepted on 30 January 2022

Article DOI: https://doi.org/10.30574/wjarr.2022.13.1.0109

## Abstract

To carry out a scientific study on a given problem (medical, environmental, etc.), it is often necessary to have recourse to a reliable dataset. In Africa, access to data is generally a problem for the scientific community. Most of the existing datasets are detained by some tiers organization who often do not give access to it. To overcome these problems related to access to data, scientists generally use traditional methods (surveys, on-site readings, etc.) for their study. However, these methods are not only expensive, but also require a regular visit to the site for updates.

In this article, we propose an autonomous solution for collecting and analyzing data based on the Internet of Things (IoT). The solution can be used to collect data of different kinds. Unlike traditional data collection methods, the proposed solution is much cheaper and allows real time data collection. To demonstrate its proper functioning, a practical case on the environmental data collection will be studied in this paper.

**Keywords:** Internet of Thing; Data collection; MQTT; Architecture

## 1. Introduction

Nowadays, data analysis tools make it possible to predict events in the near future. This can be crucial in strategic decision-making. Having reliable data and analyzing it well has become strategic. For example, having data on the movement of the population in real time of a country or a city could make it possible to predict the evolution of a pandemic like Covid 19.

In Africa in general and in Senegal in particular, data is generally held by companies which often do not give access to it to the scientific community. In Senegal, for example, information on population mobility can be drawn from data held by telephone operators. But because of their confidentiality policy and protection of customer privacy, their access to the scientific community is almost impossible. It is therefore more than urgent to have several anonymous databases accessible to the entire scientific community for carrying out strategic studies.

We can rely on the Internet of Things (IoT) to provide a solution to this data collection problem. The IoT enables things to hear, listen, collect, speak and act autonomously. Internet of things common definition is a network of physical objects. The internet is not only a network of computers, but it has evolved into a network of device of all type and sizes , vehicles, smart phones, home appliances, toys, cameras, medical instruments and industrial systems, animals, people, buildings, all connected, all communicating and sharing information based on stipulated protocols in order to achieve smart reorganizations, positioning, tracing, safe and control and even personal real time online monitoring , online upgrade, process control and administration [1].

* Corresponding author: Ahmed-Mouhamadou WADE
LTISI laboratory GIT, Polytechnic School of Thiès (EPT), Thiès, Sénégal.

In this article, we propose a generic IoT-based solution for collecting and analyzing different types of data. A practical case on environmental data collection will be studied.

The article is organized as follows: section 2 gives a state of the art on existing IoT-based data collection solutions, in section 3, we present the implementation of the environmental data collection solution that we propose. The results obtained will be presented in section 4. Conclusion and perspectives are given in section 5.

## 2. Related Work

In order to create a network architecture for data acquisition, the first thing to do is to understand the data analysis needs. If some of the data is going to be processed and analyzed by the devices that collect it (pure edge computing) [2], and the rest will be sent to the cloud for analysis, then it's a relatively simple architecture. But if fog [15] is to be applied, where there may be intermediate nodes between the devices and the cloud, the architecture may be a bit more complicated. It is also necessary to create a cloud computing architecture capable of managing the aggregation and processing of the large amount of data that the IoT network will generate [3]. Configuring the correct databases, server types, load balances, and other cloud components will be important to ensure an efficient data aggregation and analysis process.

By moving data computing and services from the cloud to the edge of the network, edge computing has become a promising solution to address the limitations of cloud computing by supporting delay-sensitive and context-aware services in the Internet age of Internet of Things.  However, to truly achieve edge computing in IoT applications, many challenges still need to be solved, such as efficient distribution and management of storage and processing, collaboration with cloud computing for more scalable services. As well as how to secure and keep the privacy of the entire system. In this section we will present some research trends, recent advances, and potential avenues of research in the emerging field of edge computing.

Despite existing connectivity solutions, more comprehensive work is needed in designing new architectures that provide communication and computational resources to successfully support massive IoT deployments. The new architectures are expected to evolve operationally and economically with the expansion of IoT and provide intelligent functionality for autonomous reasoning between connected objects. Several works propose architectures to improve connectivity for specific IoT applications. Castellani et al. [5] proposed to connect sensors and actuators to the Internet for smart office applications. Similarly, Schleicher et al. [4] developed an architecture for smart city applications and identified key aspects for its implementation. Wang et al. [6] presented an energy-saving architecture for industrial IoT and proposed a shutdown and recovery scheduling protocol to extend the lifetime of the whole system. Xu et al. [7] analyzed the integration of IoT into current network systems, including cloud, smart phone, and industrial networks. With the large-scale expansion of IoT, cloud-based architectures aim to provide comprehensive coverage of processing, compute, and storage needs in data centers [8]. However, the centralization of cloud computing and the increasing traffic demands of the IoT can lead to significant bottlenecks that degrade network performance. Edge computing architectures could potentially overcome the drawbacks of this approach, bringing the service offering closer to the edge of the network [9]. Three edge computing architectures have been proposed so far: mobile edge computing [10], fog computing [14] and cloudlets [11]. Mobile edge computing deploys cloud servers in base stations (BS), bringing computing capabilities closer to end users. The business and technical benefits of mobile edge computing and its integration with IoT are discussed in [12]. Fog computing routers, initially offered by Cisco for IoT traffic, bring computing tasks even closer to users. Sun et al. [13] proposed a hierarchical fog-like architecture to provide flexible IoT services. They showed that their solution significantly reduces traffic load in the core network and latency between IoT devices and computing resources compared to traditional IoT architectures. The cloudlets approach is an extension of the cloud integrated with Wi-Fi and cellular networks. It's driven by the fact that it's easy to deploy in cafes or offices for near real-time provisioning.

In our work, we rely on some edge of fog principles used in many of these related works. Although, we simplify the communication mechanism between sensors and the gateway. We propose a more general architecture that covers the different parts of the whole process: data acquisition, transmission, storage and processing. Even if the sensors capture the data continuously, we implemented a batch processing approach that only transmit data at fixed periods (pretty frequent). This leads to reducing the traffic load.

## 3. Implementation

The generic data collection solution that we propose in this article is based on a hybrid architecture that is composed of two parts connected by a gateway. On the one side, we have a sensor network based on WiFi technology and using the MQTT protocol for communication between sensors (nodes) and gateway. On the other side, we have a cloud part composed of three components: storage, processing and visualization: see Figure 1.

To better understand how it works, we will now focus on the example that we have implemented and which concerns the collection of environmental data (ambient temperature, contact temperature and relative humidity). However, our solution can be used to collect data of different kinds.

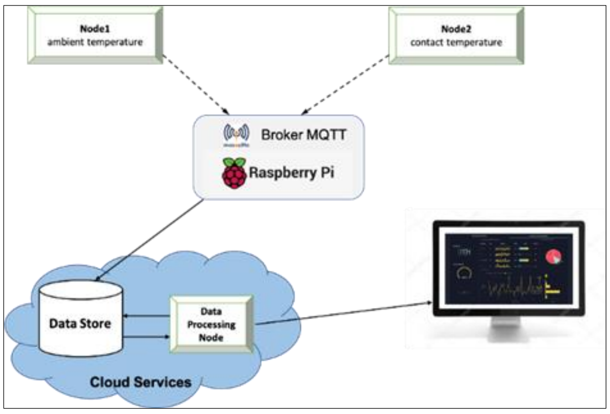
### 3.1. Operation

The data collection operates as follows: every 15 minutes, node 1 (respectively node 2), captures the ambient temperature and relative humidity values, (respectively the contact temperature of a surface) and publishes them on a broker using the MQTT protocol. The broker is installed on the gateway. Once the data has been published, a client (subscriber in this case) installed on the gateway retrieves the published data and sends it to the database node on AWS RDS [16]. Between the reception and the sending of the data, an intermediate treatment is carried out on the gateway. This consists of calculating the averages per hour and per day of the data (values) collected. The data is either stored in the database and temporarily on the gateway in a CSV format. For the processing and analysis part, we use another node that reads the data from the database via a pipeline and transforms it in order to make it available in Json format for viewing and exporting operations. The visualization is done with the Plotly Dash Framework on a separate node.

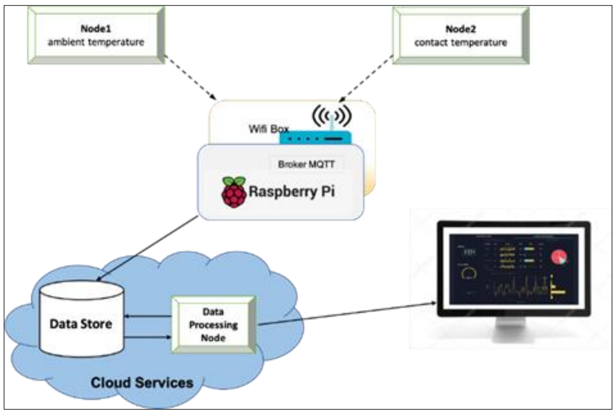
### 3.2. Hardware architecture

We propose two types of architecture for our data collection solution (Figure 1). The architectures are composed of three parts: the hardware, the software and the communication protocol.

Depending on how the communication is managed between the nodes and the gateway, we have architecture A and architecture B. The main difference between these two architectures lies in the way the communication between the nodes and the gateway is managed.



Architecture A



Architecture B

**Figure 1** Architectures

In both architectures we have two nodes, one that gives the ambient temperature and the relative humidity of an environment and the other that gives the contact temperature of a surface; a gateway which, in addition to transferring the data collected to a database in the Cloud, plays the role of backup (stores the data collected locally). We also use the computing power of the gateway to make intermediate calculations (average per hour and per day) before storing it in the database [15]. In both architectures, the MQTT protocol is used for communication between nodes and gateway. A broker (Mosquitto) and a subscriber (mqtt in de node-RED) are installed on the gateway (Raspberry Pi) and the nodes act as publishers.

The difference between the two architectures (A and B) lies in the WiFi network which makes communication between the nodes and the gateway possible. In architecture A, the Raspberry Pi emits its WiFi and acts as an access point. The nodes (ambient temperature and humidity and surface temperature sensors) connect to the Raspberry Pi WiFi and use the MQTT protocol to send the measured quantities to the Broker server. In architecture B, we use a 3G/4G router as a WiFi access point and the nodes and the Raspberry Pi connect to the network transmitted by the access point to exchange data. Depending on the architecture used, we have advantages and limitations.

### 3.3. Hardware components

The hardware is composed of two nodes, a gateway and a WiFi box.

- Node 1 consists of an arduino MKR WiFi 1010 board, a DHT22 sensor and a 1050mAh LiPo battery (see Figure 2).The Arduino board is equipped with a SAMD21 Cortex®-M0+ 32bit low power ARM microcontroller and the Nina W102 uBlox module WiFi board. The latter is used to send the collected data to the gateway. The board has an integrated LiPo battery connector, which allows it to be made autonomous by plugging in a LiPo battery. A DHT22 sensor is connected to the Arduino MKR WiFi 1010 board to retrieve temperature and humidity. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements are connected with an 8-bit single-chip computer. The sensor can capture temperatures ranging from -40 to 80 degrees Celsius with an accuracy of ±0.5°C and 0 to 100% relative humidity with an accuracy of ± 2% RH [17].

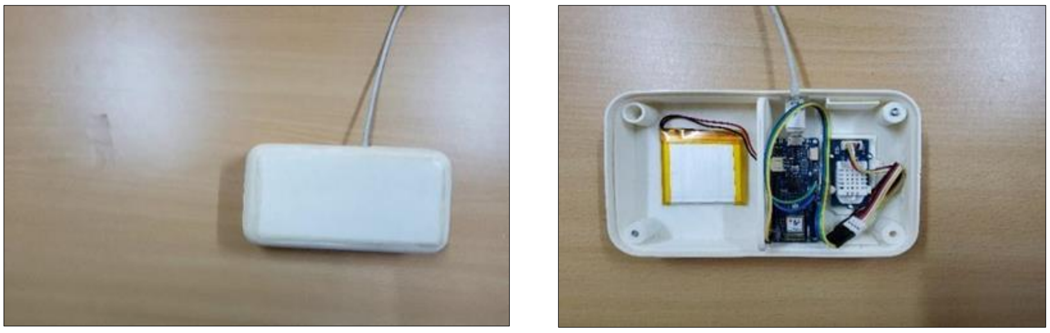


**Figure 2** Node 1

- Node 2 (Figure 3) is also composed of an Arduino MKR WiFi board with the same characteristics as the node 1, a 1050mAh LiPo battery and a type k Thermocouple. The Thermocouple is used to capture the contact temperatures of a surface. The version we use (the TF-500) can measure temperatures ranging from -40 to 200 degrees Celsius with an accuracy of ± 1.5°C [18].

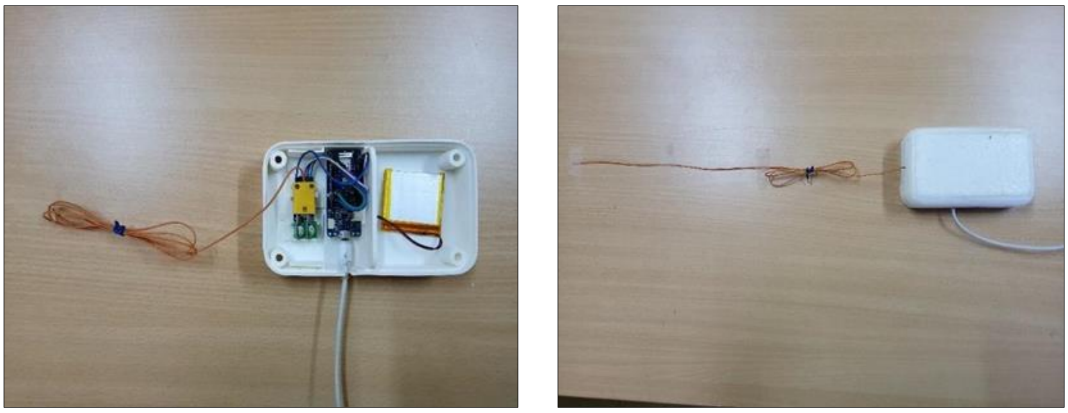


**Figure 3** Node 2

- A Raspberry Pi 4 model B is used in our solution as a gateway. In addition to playing the role of the internet gateway, the Raspberry Pi serves as a backup to the database. Intermediate calculations are also performed on the latter to optimize calculation times. The Raspberry Pi is equipped with an A-Series Quad-Core A8-3500M processor, 4GB of SDRAM, input/output devices, a WiFi card and an RJ45 connector [19]. Its two Ethernet connections allow it to be connected to the sensors on one network (via its WiFi card) and the database on another network (Wired).

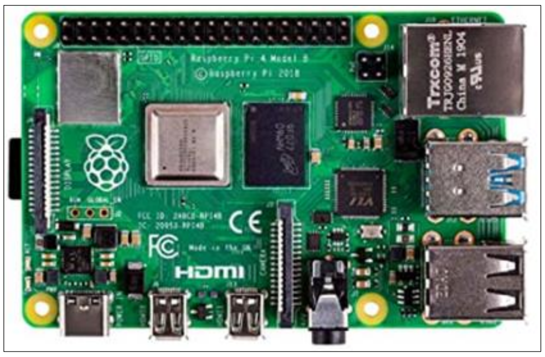


**Figure 4** The gateway

### 3.4. Software components

In order to operate the hardware, a software component is required. In this section, we will detail the software components used for data collection and processing.

- For data collection, each node executes the following algorithm (DataCollection( )) in a loop.

The node wakes up (by executing the wakeUp( ) procedure), then it reads the data captured by the sensors connected to it (by executing the readSensor( ) procedure), then it sends the data read (by executing the sendData( ) procedure) to the broker via the MQTT protocol then it goes to sleep until the next reading (by executing the goToSleep( ) procedure). This algorithm will save battery consumption. Since the connection between the nodes and the gateway is provided by WiFi, it may happen that the connection is lost during data transmission. To compensate for a possible loss of data caused by the WiFi connection, we add to the algorithm the debuglog( ) procedure which will allow the nodes to keep the data not transmitted in the flash memory of the microcontroller and to transmit them at the next connection. This temporary storage minimizes data loss related to connection loss issues. Once deployed, the nodes must be autonomous and no human intervention is planned except to change a defective component. It often happens that an electronic component encounters operating problems that require a hard reset to return to normal. To avoid human intervention to resolve these kinds of problems, we implement the MyWatchDoggy.clear( ) procedure to allow a logical reset in the event of a problem of this kind. The procedure runs (hardreset) automatically if the execution of the algorithm exceeds the time needed to wake up, read the data, send it to the broker and go back to sleep.

| **Algorithm 1 : DataCollection( )** | | |
|---|---|---|
| Step 1 | : | Loop |
| Step 2 | : | wakeUp( ); |
| Step 3 | : | readSensor( ); |
| Step 4 | : | sendData( ); |
| Step 5 | : | debuglog( ); |
| Step 6 | : | goToSleep( ); |
| Step 7 | : | MyWatchDoggy.clear( ); |

- The Node-RED programming environment which is based on Node.js is used on the gateway for receiving, processing and sending data.We use among others the nodes mqtt in, file, exec, mysql, delay, inject and function of Node-RED for this purpose (figure 6). The mqtt in node is used to subscribe to different topics and read published data. After each reading, an injection is performed, thanks to the inject node, to add the date and time (exec node) of the data acquisition.The data read is then stored locally in a CSV file (file node) before being sent to a cloud database (mysql node). Function nodes are used throughout the program to do type conversion, string concatenation and more.

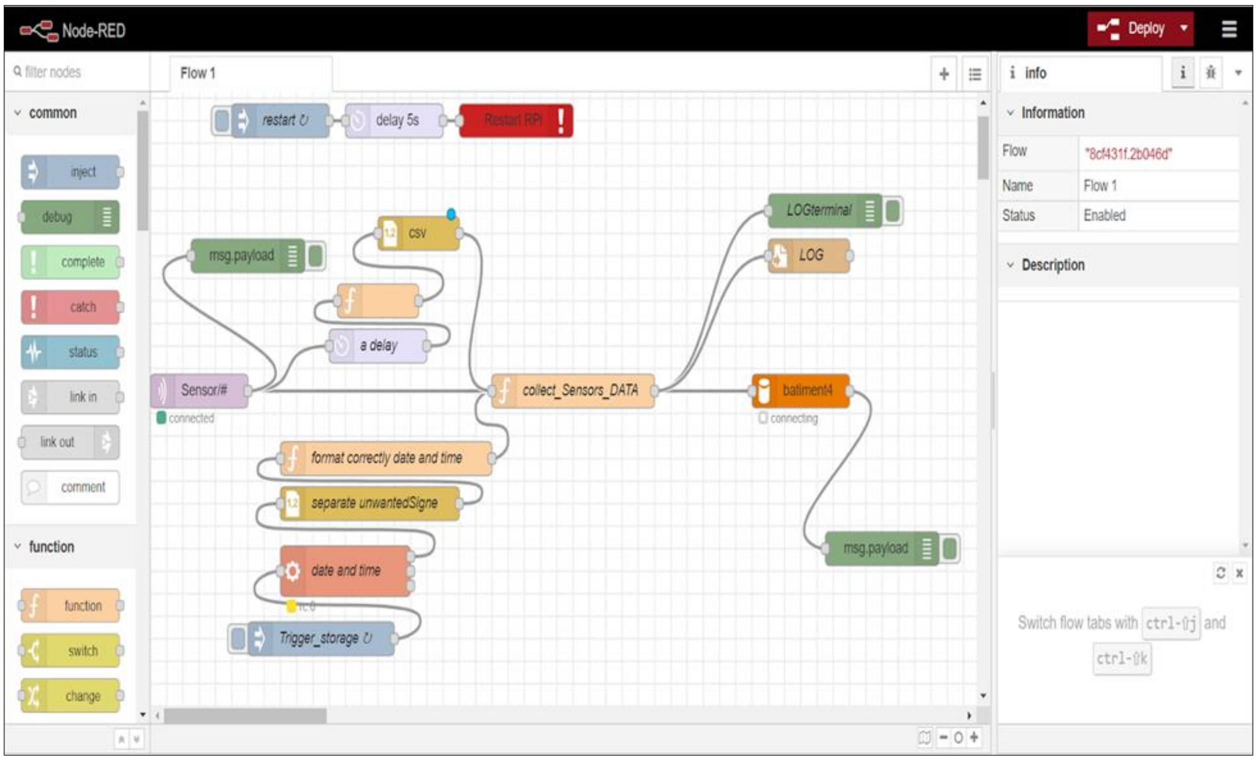


**Figure 5** Node-RED

## 4. Results and discussion

In the following figures we show some of the results obtained with our data collection system. On the x-axis (all curves) we have the timeline. Indeed, each point of each curve is an average value over an hour. For example, the point 10 of the x-axis corresponds to the average value between 9h and 10h. We retrieve the data measured by the sensors every fifteen minutes and then calculate the rolling averages per hour and per day. On figures 7, 8, 9 and 10 we have displayed the results of one day (24 points going from 00h to 23h).

In Figure 7, we track the outdoor ambient relative humidity while in Figures 8 and 9 we have the outdoor ambient temperatures and the outdoor surface temperatures respectively.

Figure 10, on the other hand, shows a view combining several of these curves. It allows for easier comparative analysis; for example, follow the evolution of the outside ambient temperature compared to the surface temperature.

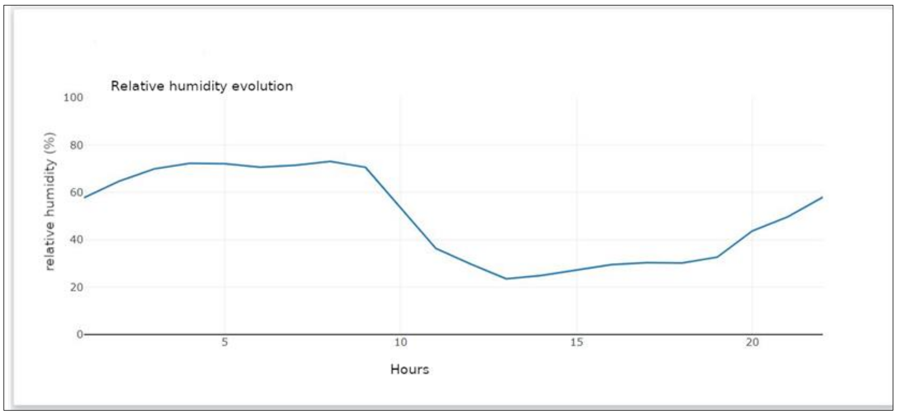


**Figure 6** Relative Humidity Monitoring

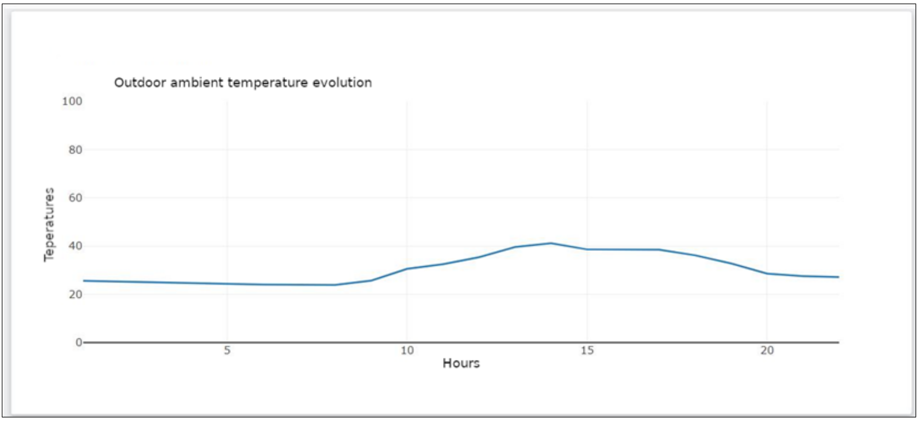


**Figure 7** Ambient Temperature Monitoring

More generally, these different figures show that the system we have implemented works quite well and allows not only to collect data of environmental origin (in this case) but also to store it, analyze it and visualize some results.

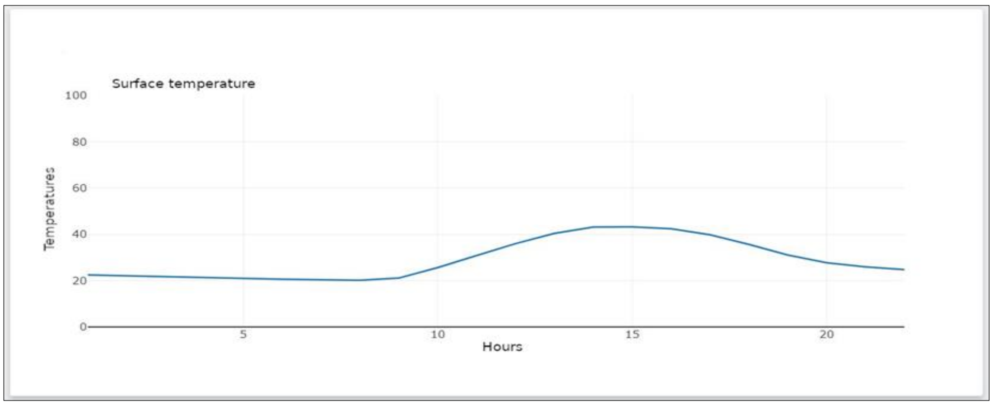


**Figure 8** Outdoor Surface Temperature Monitoring

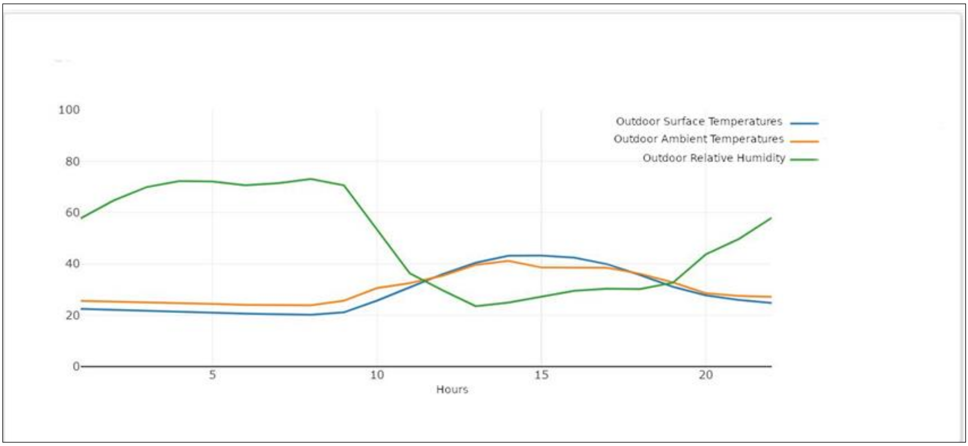


**Figure 9** Comparing Some Observed Values

## 5. Conclusion

In this paper, we presented a general hybrid architecture for data collection and analysis. We showed how to implement an IoT based solution for data capture and transmission to a cloud platform for storage and visualization. We also explored the possible usage of edge or fog mechanism to process some operation as earlier as possible. The general solution we proposed here is quite cheap and simple to implement. We believe it as a concrete contribution that will help advance the data collection means for research in Africa and even elsewhere.

In our current work, we are comparing different approaches for the architecture in order to choose the one that is more efficient for connectivity and more resilient against faults.

## Compliance with ethical standards

### *Acknowledgments*

The authors would like to thank Cool Roof France and Emasol Sénégal for funding the material used for the experiments.

### *Disclosure of conflict of interest*

The authors declare that there is no conflict of interests regarding the publication of this paper

## References

[1] Vermesan O, Friess P. (Eds.). Internet of things: converging technologies for smart environments and integrated ecosystems. River publishers. 2013.

[2] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li and Lanyu Xu. Edge Computing: Vision and Challenges, Wayne State University {weisong, jiecao, quan.zhang, huizi, xu.lanyu}@wayne.edu.

[3] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems, Volume 25, Issue 6, 2009, Pages 599-616, ISSN 0167-739X JM Schleicher, M Vögler, S Dustdar, C Inzinger. Application Architecture for the Internet of Cities: Blueprints for Future Smart City Applications," IEEE Internet Computing. 2016; 20(6): 68 – 75.

[4] Y Xu, A Helal. Scalable Cloud–Sensor Architecture for the Internet of Things, IEEE Internet of Things Journal. 2016; 3(3): 285 – 298.

[5] AP Castellani, N Bui, P Casari, M Rossi, Z Shelby, M Zorzi. Architecture and protocols for the Internet of Things: A case study, in Proc. 8th IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM). 2010; 678 – 683.

[6] K Wang, Y Wang, Y Sun, S Guo, J Wu. Green Industrial Internet of Things Architecture: An Energy-Efficient Perspective, IEEE Communications Magazine. 2016; 54(12): 48 – 54.

[7] K Xu, Y Qu, K Yang. A tutorial on the internet of things: From a heterogeneous network integration perspective, IEEE Network. 2016; 30(2): 102 – 108.

[8] Y Xu, A Helal. Scalable Cloud–Sensor Architecture for the Internet of Things, IEEE Internet of Things Journal. 3(3): 285 – 298.

[9] S Wang, X Zhang, Y Zhang, L Wang, J Yang, W Wang. A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications, IEEE Access. 2017; 5: 6757 – 6779.

[10] Michael Till Beck, Martin Werner, Sebastian Feld, Thomas Schimper. Mobile Edge Computing: A Taxonomy. In AFIN 2014 : The Sixth International Conference on Advances in Future Internet, ISBN: 978-1-61208-377-3, 2014. Dijiang Huang, Huijun Wu. Mobile Cloud Computing Taxonomy, In Mobile Cloud Computing, Morgan Kaufmann. 2018; 5-29.

[11] YC Hu, M Patel, D Sabella, N Sprecher, V Young. Mobile edge computing—A key technology towards 5G. ETSI White Paper. 2015; 11: 1 – 16.

[12] X Sun, N Ansari. EdgeIoT: Mobile Edge Computing for the Internet of Things, IEEE Communications Magazine. 2016; 54(12): 22 – 29.

[13] Jagdeep Singh, Parminder Singh, Sukhpal Singh Gill. Fog computing: A taxonomy, systematic review, current trends and research challenges". In Journal of Parallel and Distributed Computing. 2021; 157: 56-85.

[14] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing (MCC '12). Association for Computing Machinery, New York, NY, USA. 2012; 13–16.

[15] Amazon Web Services, Inc. Amazon Relational Database Service (RDS). 2022.

[16] Alldatasheet.com. DHT22 sensor Datasheet. 2022.

[17] Arduino.cc. MKR WiFi 1010, The easiest entry point to basic IoT and pico-network application design. 2022.

[18] Raspberry.org. Raspberry Pi 4 Computer Model B. Tech Specs. 2022.