People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Dr. Tahar Moulay University of Saida
Faculty of Technology
Department of Electronics
INSTRUMENTATION

DESIGN AND IMPLEMENTATION OF AN IOT CLOUD TECHNOLOGY INTO WIRELESS SENSOR NETWORK APPLICATION

Master Thesis

Submitted by: Ms. Sara RACHEDI

Board of Examiners

Supervisor: Dr.Lotfi MOSTEFAI	**Dr Tahar Moulay University, Saida**
Co-Supervisor: Mr.Merzoug BOUHAMDI	**Dr Tahar Moulay University, Saida**
Member: Dr.Ameur DAHANI	**Dr Tahar Moulay University, Saida**
Member: Mr.Redouane BERBER	**Dr Tahar Moulay University, Saida**

September 2020

Dedication

This Work Is Dedicated

To my beloved mother, to my father whose support, encouragement and constant love has sustained me throughout life.

To my brothers and sisters;

To all my relatives and friends,

To the best things I gained from the field,

And finally to you, dearest reader.

Acknowledgments

This thesis marks an end and a beginning of an arduous but insightful journey in scientific research. It would have never been completed without constant support and encouragement from my supervisor, Dr. Lotfi MOSTFAI & Mr. Merzoug BOUHAMDI. Their wisdom, patience, stimulating suggestions and encouragement gave me the energy to complete what at times seemed to be an unattainable goal. I would like to thank them for their invaluable advice and guidance throughout the course of my studies.

My sincere thanks and gratitude go to the members of the board of examiners for accepting to read and evaluate this work.

I must acknowledge a special debt of gratitude to my friend, the future Dr, Apolytos who never doubted my ability to succeed and who supported me to overcome a lot of difficulties while investigating.

I want to express my sincere appreciation to my GDG community for helping me conducting some exclusive researches and for having access to all the resources. I am indebted to them for providing me with all what it takes to experiment and improve, to confirm that GDGers are able to develop their competencies provided that they are motivated and well trained.

My deepest thanks go to all my friends who have constantly given me support and strength to continue this research and are extended to my Asterism family, my colleagues and classmates.

I am also grateful to kamar who gave me a hand in computing.

Finally, I would like to express my great pride to my beloved parents, my brothers and sisters for their endless love, sacrifice and support in order to finish this study.

SUMMARY

List of Figures

CHAPTER ONE

CHAPTER TWO : THE HARDWARE SETTING

CHAPTER THREE : THE SOFTWARE DEVELOPMENT

List of Tables

CHAPTER TWO : THE HARDWARE SETTING

نبذة مختصرة

لقد لاحظنا مؤخرًا العديد من الثورات التكنولوجية، فضلاً عن الانتقال من العالم التناظري إلى نظيره الرقمي ومن الحلول السلكية المركزية إلى الأنظمة اللاسلكية الموزعة والمنتشرة. على وجه الخصوص ، أدى ظهور أجهزة الإرسال والاستقبال ذات الطاقة المنخفضة والميسورة التكلفة ، إلى جانب تطوير مكدسات قياسية صغيرة الحجم ومفتوحة ، إلى إنشاء شبكات استشعار لاسلكية (WSN) ، تم اعتمادها في الغالب لتطبيقات المراقبة المنزلية والمكتبية والصناعية. الهدف الطموح حاليًا هو أخذ عينات وجمع وتحليل كل قطعة من البيانات من حولنا لتحسين كفاءة الإنتاج وضمان الاستهلاك الأمثل للموارد. إن "إنترنت الأشياء (IoT) "، أي توصيل الأشياء اليومية مثل المستشعرات والمشغلات بالإنترنت حيث يتم ربط الأجهزة معًا بذكاء مما يتيح أشكالًا جديدة من الاتصال بين الأشياء والبشر ، وبين الأشياء نفسها ، هو الحل الأساسي لهذا طلب. قد يتم التعامل مع الكم الهائل من البيانات التي يتم إنشاؤها نتيجة لذلك بشكل مربح باستخدام خدمات "السحابة" ، أي أطر عمل / برمجيات مرنة وقوية قادرة على تقديم الحوسبة كخدمة. من خلال هذا العمل ، نهدف إلى اقتراح بنية Iot مرنة وقابلة للتوسيع لدمج الخدمات السحابية في شبكات الاستشعار اللاسلكية.

Abstract

Recently, we have witnessed several technological revolutions, as well as the transition from the analog world into its digital counterpart and from centralized wired solutions to distributed and pervasive wireless systems. Especially, the appearance of affordable and low-power transceivers, beside the development of compact-size and open standard stacks, have created potential Wireless Sensor Networks (WSNs), mostly adopted for home, office and industrial monitoring applications. Currently, the ambitious aim is to sample, collect and analyze every piece of data around us to improve production efficiency and ensure optimal resource consumption. The "Internet of Things" (IoT), i.e. connecting everyday objects like sensors and actuators to the Internet where the devices are intelligently linked together enabling new forms of communication between things and human beings, and between things themselves, is the key answer to this request. The huge amount of data being consequently generated may be profitably handled using "cloud" services, i.e. flexible and powerful hardware/software frameworks capable to deliver computing as a service. Within this work we aim at suggesting an extensible and flexible Iot architecture for integrating Cloud services in Wireless Sensor Networks.

GENERAL INTRODUCTION

1. Background of the Study

The “Internet of Things” (IoT), is the ubiquity of the Internet by integrating all objects for interaction via embedded systems, leading to a highly distributed network of devices communicating with people as well as other devices. Building IoTs has advanced considerably within the last few years since it has added a new dimension to the world of information and communication technologies. Now anyone, from almost anytime and anyplace can have connectivity for anything and it is expected that these connections will extend and make a wholly advanced dynamic network of IoTs. The development of the Iot will revolutionize a varety of sectors, from wireless sensors to nanotechnology.

In fact, one of the important elements in the Iot paradigm is wireless sensor networks (WSNs). WSNs consist of sensing nodes with embedded CPUs, sensors which are used to monitor environmental conditions such as temperature, pressure, humidity.. Briefly, the purpose of the WSN is to provide sensing services to the users. Since, the number of users of the Internet is growing therefore; it is wise to provide WSN services to this ever increasing community.

Cloud computing is a flexible, powerful and cost-effective framework in providing real-time data to users at anytime, anywhere with immense coverage and quality. The Cloud inclouds of hardware, networks, services, storage, and interfaces that enable the delivery of computing as a service. Additionally, it’s possible to add the data obtained from the wireless sensor nodes to the Web services. By connecting, evaluating and linking these sensor networks, information conclusions can be made in real-time, trends can be predicted and hazardous situations can be avoided.

2. Statement of the problem

These days, Iot based projects require an important amount of informations and data that need to be stored and analyzed frequently to guarantee acceptable performances. This kind of tasks is hard be done manually and must be automated to meet current standards which involves knowledge of deploying, designing, and maintaining the infrastructure of data streaming.

In addition to this, it takes a lot of time dealing with networking issues, including making sure everything still works when it goes offline, wondering if it has the right security and privacy while providing a scalable service no matter how many millions of users it might have, and without affecting the flexibility of the whole system.

3. Design Issues and Challenges

Most of the issues and challenges in the design of Sensor Clouds design issues and challenges arise due to the inherent limitations of sensor devices such as limited processor performance, small storage capacity, limited battery power, and unreliable low-bandwidth wireless communication and some issue arise due issue de-facto standard of cloud such as reliability, back up, privacy, security ownership etc.

3.1 Sensor Issues

1) Power Management:

Power management is a major concern as sensor nodes do not have fixed power sources and relies on limited battery power. Sensor applications executing on these devices have to make tradeoffs between sensor operation and conserving battery life. The sensor nodes should provide adaptive power management facilities that can be accessed by the applications. From the SensorCloud perspective, the availability of sensor nodes is not only dependent on their load, but also on their power consumption. Thus, the Sensor Cloud's resource management component has to account for power consumption.

2) Scalability:

Scalability is the ability to add sensor resources to a Sensor-Cloud to increase the capacity of sensor data collection, without substantial changes to its software architecture. The Sensor-Cloud architecture should allow multiple wireless sensor networks, possibly owned by different virtual organizations, to be easily integrated with compute and data cloud resources. This would enable an application to access sensor resources across increasing number of heterogeneous wireless sensor networks.

3) Network Connectivity and Protocols:

The network connections are usually fast and reasonably reliable in cloud. On the other hand, the sensor nodes in Sensor Clouds are connected via wireless ad hoc networks which are low-bandwidth, high-latency, and unreliable. The network connectivity of sensor nodes is dynamic in nature, and it might be irregular and vulnerable to faults due to noise and signal degradation caused by environmental factors. The Sensor-Cloud has to gracefully handle unexpected network disconnections or prolonged periods of disconnection. Thus, efficient techniques to interface sensor network protocols with cloud networking protocols are necessary.

4) Scheduling:

In wireless sensor networks, scheduling of sensor nodes is often performed to facilitate power management and sensor resource management. Researchers have developed algorithms to schedule the radio communication of active sensor nodes, and to turn off the radio links of idle nodes to conserve power. Similarly, for applications like target tracking, sensor management algorithms selectively turn off sensor nodes that are located far away from the target, while ` to improve the availability of sensor nodes are necessary. Sensor Clouds should support job and service migration, so that a job can be migrated from a sensor node that is running out of power or has failing hardware to another node.

3.2 Cloud Issues

1) Reliability:

Stability of the data storage system is of important consideration in clouds. Generally, people worry about whether a cloud service provider is financially stable and whether their data storage system is trustworthy. Most cloud providers attempt to mollify this concern by using redundant storage techniques, but it is still possible that a service could crash or go out of business, leaving users with limited or no access to their data.

2) Data Backup:

Cloud providers employ redundant servers and routine data backup processes, but some customers worry about being able to control their own back-ups. Many providers are now offering data dumps onto media or allowing users to back up their data through regular downloads.

3) Privacy:

The Cloud model has been criticized by privacy advocates for the greater ease in which the companies hosting the Cloud services control and monitor communication and data stored between the user and the host company lawfully or unlawfully. There have been efforts to "harmonize" the legal environment by deploying local infrastructure and allowing customers to select "availability zones."

4) Security:

Cloud service providers employ data storage and transmission encryption, user authentication, and authorization. Many clients worry about the vulnerability of remote data to criminals and hackers. Cloud providers are enormously sensitive to this issue and apply substantial resources to mitigate this problem.

5) Ownership:

Once data has been relegated to the cloud, some worry about losing their rights or being unable to protect the rights of their customers. Many cloud providers address this issue with well-skilled user-sided agreements. According to the agreement, users would be wise to seek advice from their favorite legal representative.

6) Availability and Performance:

Business organizations are worried about acceptable levels of availability and performance of applications hosted in the cloud.

7) Legal:

There are certain points of concern for a cloud provider and a client receiving the service like location of the cloud provider, location of infrastructure, physical location of the data and outsourcing of the cloud provider's services etc.

4. Aims of the Study and Contribution

In this work, we present the design, development and integration of an extensible architecture for WSN with the Cloud services, where the collected data from the sensors are processed, stored and viewed in real time.

We have used Cloud Firestore from Firebase and Google Cloud Platform, it's a flexible, scalable database for mobile, web, and server. And this solution can be integrated into other application domains like smart homes, e-health, care services, or even vehicular area networks.

For proof of concept, we have created a web and mobile app, which enables data visualization from any device and anywhere.

5. Structure of the Work

The remaining of this work is organized as follows:

Chapter 1, discusses briefly the related work and describes the proposed architecture while Chapter 2 outlines the hardware design. Chapter 3 discusses the software implementation. Chapter 4 presents the implementation results and finally, some conclusions are presented.

6. Conclusion

Sensor networks is an emerging area and there are many research issues pertaining to sensor networks such as energy management, coverage, localization, medium access control, routing and transport, security etc. Research in cloud computing is also in fantasy stage. It also has a number of research challenges such as efficient resource allocation, high resource utilization and security etc. Apart from the afore-mentioned research issues in sensor networks and cloud computing, sensor-cloud computing gives rise to additional research challenges, especially when it is used in missioncritical situations. These research challenges are: web services and service discovery which work across both sensor networks and the cloud, interconnection and networking, coordinated quality of services etc.

The goal is to integrate it with WSN, as the core of data management system, which represent a challenge itself when it comes to choosing the right provider, the right service and to configure it properly.

In order to respond correctly to such severe constraints, "Firebase services" from Google cloud platform are proposed.

CHAPTER ONE : RELATED WORK

1. Introduction

The Internet of Things (Iots) offers potentialities which make it possible for the development of so many applications. Some of the mentioned application domains are transportation, healthcare, smart environment, personal and social domains. Each of the domains embody its own unique characteristics in terms of real-time processing, volume of data collection and storage, identification and authentication, and security considerations. For example, real-time processing is of utmost importance within the transportation industry, while identification and authentication are important aspects in healthcare.

Cloud services, with its virtually unlimited resources of computing power, storage space, and networking capabilities, is well appropriate for scaling in the IoT world.

As of late, an extensive measure of research in the field of the probability of integrating cloud computing with WSNs has been explored. This paradigm has been proposed as a feasible mechanism to accomplish the best use of a wireless sensor infraconfiguration and allows data sharing among multiple applications.

2. Related work

Wireless sensor platforms have been deployed in a wide range number of applications, from health care and medical such as Alarm-Net [1], or CodeBlue [2] to environmental monitoring [3-5]. The architecture of these systems has been designed in a very ad hoc fashion and is not flexible to adapt to other applications or scenarios while the core problem is the same, remote monitoring using sensor networks. Throughout the last few years, many researchers have investigated on ways to connect wireless sensor networks to the Cloud [6]. Authors in [7-11] have presented Internet protocols for connecting wireless sensor networks to the Internet but no real implementations have been shown. Much of the previous work has been on theoretical aspects of system architecture rather than actual deployment and testing of wireless sensor networks with the Clouds. Use of Web services to connect sensor networks with external networks have also been suggested by researchers in [12, 13].

Wireless sensor networks (WSN) [13] have a great role, enabling the mitigation of many of the problems in traditional monitoring systems. They provide a continuous and distributed operation, which is very important, achieving the necessary increase in the spatial density of the measurements. Furthermore, in this way, contaminant measurement systems with low-energy consumption, complexity, and cost are achieved.

From the other side, Cloud can reach out to the "real world" through the IoT [14, 15]. Although it is a new infrastructure, this type of application has already emerged in different areas such as health [16, 17], smart cities [18, 19], or environmental monitoring [20–22].

In addition, following the merger of Cloud computing and IoT, some platforms that offer services to store and/or process information from the IoT in the cloud have emerged [23]. Among them are open source projects, such as OpenIoT [24], or commercial clouds offered by service providers such as Xively [25], ThingSpeak [26], CloudPlugs [27], DeviceCloud [28], Thinkingthings [29], SensorCloud [30], AWS IoT [31], or Google Cloud IoT [32].

3. Service Creation and Innovation

Capabilities Sensors are very limited and specific to their applications or services when they are linked to a typical sensor network. Therefore, the numbers of organizations that can provide the sensor services are very limited. However, when the services of sensors move onto the cloud, it is possible to include them to realize a variety of applications [33,37].

A number of services can be provided to the users for different applications such as health applications, environmental monitoring, industrial tasks (e.g., refining), surveillance, senior residents monitoring, or even the applications that monitor the vibration in buildings during an earthquake.

In the Sensor-Cloud infrastructure, the sensors and service templates are constructed as catalog menu service on the cloud, and the requesters can create new sensor services with the existing sensors in these service instances. For example, service requester can create a sensor service to analyze the impact of earthquake to each floor or room of the rehabilitation center or hospital, and at the same time it can also create sensor services to support older residents with the same set of sensors (virtualized sensors).This service will then help the caregiver to shift the older adults one by one into a safe place. Using the identical sensor services for healthcare, another service requester can create dissimilar sensor service to track the patient's medicine intake and then to analyze the effectiveness of pills through the use of some selected healthcare sensors. Thus, the service requesters can be provided with new services using the same set of sensors on cloud service platforms. This will reduce the cost for resource usage and could have numerous elastic merits to it. In this section, several existing Sensor-Cloud applications are described.

3.1. Existing Sensor-Cloud Applications.

There exist a number of services based on Sensor-Cloud infrastructure to store and process the sensor-based information. Few of them are described briefly as below.

3.1.1. Nimbits.

Nimbit [38] is a free and social service that is used to record and share sensor data on cloud. It is a cloud-based data processing service and is an open-source platform for the IoT (Internet of Things). We can feed the versatile numeric, text-based, JSON, GPS, or XML values by creating a data point in the cloud. The data points can be connected to Scalable Vector Graphic (SVG) process control, spreadsheets, diagrams, websites, and more. Data points can also be configured to generate alerts data-relay to social networks and to perform calculations. Nimbits also provide an alert management mechanism, data compression mechanism, and data calculation on received sensor data by employing some simple mathematical formulas.

3.1.2. Pachube Platform.

Pachube [39] is one of the first online database service providers, which allows us to connect sensor data to the web. It is a real-time cloud-based platform for IoT with a scalable infrastructure that enables us to configure IoT products and services, store, share, and discover realtime energy, environment, and healthcare sensor data from devices and buildings around us. Pachube has a very interactive website for managing the sensor data and an open easily-accessible API. Pachube system provides free usage and has several numbers of interfaces for producing a sensor or mobile-based applications for managing the sensor data within a cloud framework anytime.

3.1.3. IDigi.

iDigi [40] is a machine-to-machine (M2M) platform as a service PaaS that minimizes the barriers to build scalable, secure, and cost-effective solutions, which can bind the enterprise applications and device assets together. iDigi eases the connectivity of remote assets devices and provides all the tools to manage, store, connect, and move the information across the enterprise irrespective of its reache. To simplify the remote device connectivity and integration, it uses connector software called iDigi Dia. Regardless of the network location, iDigi platform manages communication between remote device assets and enterprise applications.

3.1.4. ThingSpeak.

ThingSpeak [26] is another open source IoT application and has an open API to store and retrieve data from device assets or things via LAN or using HTTP over the Internet. With this platform, location tracking applications, sensor logging applications, and social network of device assets with proper update of its status can be created. ThingSpeak allows numeric data processing like averaging, timescaling, rounding, median, summing, and so forth to store and retrieve the numeric and alphanumeric data.ThingSpeak application features a JavaScript-based charts, read/write API key management, and a time-zone management. Although the above services are able to visualize the sensor data and sensor-driven information, they are lacking secure access to data and interface availability for linking the external or mobile applications for further processing. It means that most of these aforementioned projects do not address the issues of data management and interoperability issues caused by heterogeneous data resources found in the present modern environmental tracking or electronic healthcare systems. But introducing these aforementioned works with Cloud computing infrastructure may overcome the issues related to heterogeneous data access and data management functionality [41].

3.2. Emerging Sensor-Cloud Applications.

There are many other applications that are emerging based on the SensorCloud infrastructure, which can be summarized as follows.

3.2.1. Ubiquitous Healthcare Monitoring.

Sensor-Clouds can be used for health monitoring by using a number of easily available and most often wearable sensors like accelerometer sensors, proximity, ambient light and temperature sensors, and so forth to collect patient's health-related data for tracking sleep activity pattern, blood sugar, body temperature, and other respiratory conditions [42]. These wearable sensor devices must have support of BWI (Bluetooth's wireless interface), UWB (Ultra wideband), and so forth interface for streaming of data and are connected wirelessly to any smartphone through this interface. These smart phone devices pretend to function like a gateway between the remote server and sensor through the Internet, maybe GPRS/Wi-Fi, or other sort of gateways.

To transform this system into services-based structure, web-services-based interfaces are used by smart phone device to connect to the server [43]. The system prototype should have made to be robust, mobile, and scalable. Robust in the sense means that it should recover itself from circumstances, which may lack connectivity issues due to power (i.e., battery), failure, or gateway cutoff to patient's wearable devices [44]. Mobile in the sense means that it should be capable of tracking signals into heterogeneous environments; that is, it must catch the signals irrespective of whether the patient went outside or still resided into the hospital/building. It should be scalable so that it could be deployed easily for several users concurrently without affecting the performance metrics.

Finally, such prototype system should be retargetable and extensible in nature. Retargetable refers to the fact that it can handle various displays with distinct form factors and screen resolution. It means that the same health applications can be displayed to any smartphone display like PDA (personal digital assistant) or to a bigger console device in a hospital where doctors, helpers, or nurses may track the acquired data or processed information from distance. The extensibility aspect requires that if any newer sensing devices are introduced into the system for acquiring the patient's healthbased information, the system should function efficiently and conveniently without affecting backend server of the services [45].

In this platform, context awareness can be achieved that can direct us to derive a better level of emergency services to the patient. The information regarding recent operational laboratories, missing doses of pills, number of handicaps, and other situations would be helpful in health monitoring. The system should not adhere to any changes made into the operating system or intermediate components of sensing devices and is designed in such a way that it would cause minimal disturbance to services provided to existing end users of the system.

In this scenario, several numbers of sensors pick up the patient data, and these accumulated data are uploaded to a server on cloud. If any noise data is found, they are filtered using some filtering mechanism on a server. The doctors/health employees, nurses, and others can then access the patients' data on cloud through a web service portal after being authenticated/permitted by the patient.

3.2.2. Environmental Monitoring for Emergency/Disaster Detection.

In environmental applications, it is possible to detect the earthquake and volcano explosion before its eruption by continuously monitoring them through the use of several numbers of different sensors like strain, temperature, light, image, sound, acceleration, barometer sensors, and so forth through the use of wireless sensor networks [46]. Through the Sensor-Cloud infrastructure, the sensor instances engaged in environmental monitoring can be used in parallel with several other sensor instances, for example, by the healthcare department to avoid any future casualty, or with crop harvesting application services to avoid the damage caused by bad weather condition.

3.2.3. Telematics.

Sensor-Clouds can be used for telematics, meant to deploy the long distance transmission of our computerized or information to a system in continuum. It enables the smooth communication between system and devices without any intervention.

3.2.4. Google Health.

It is a centralization service of Google that provides personal health information [47] and serves as cloud health data storages. Google users are allowed to monitor their health records by logging into their accounts at collaborated cloud health service providers into the Google health system. However, in a recent declaration Google has announced the discontinuation of this health service.

3.2.5. Microsoft HealthVault.

This cloud platform is developed by Microsoft to store and maintain health and fitnessrelated information [48]. HealthVault helps users to store, gather, and share their health relevant information and its data can be acquired from several pharmacies, cloud providers, health employees, health labs, equipment, and from the users itself.

3.2.6. Agriculture and Irrigation Control (Field Server Sensors).

Sensor-Cloud can be used in the field of agriculture to monitor the crop fields in order to upkeep it. For this, a field server is developed that comprises of a camera sensors, air sensor, temperature sensor, CO2 concentration sensor, soil moisture and temperature sensors, and so forth. These sensors continuously upload the field data via Wi-Fi access point to the field owner to track the health of their crops [49]. This can also be used for harvesting.

3.2.7. Earth Observation.

A sensor grid is developed for data gathering from several GPS stations, to process, analyze, manage, and visualize the GPS data [50].This GPS data would then be uploaded onto the cloud for efficient monitoring, early warning, and decision-making capability for critical situations like the volcanic eruptions, earthquakes, tsunamis, cyclones, and so forth to the users all around the world.

3.2.8. Transportation and Vehicular Traffic Applications.

Sensor-Cloud can be used to provide an efficient, stable, equilibrium, and sustainable tracking system. Earlier existing technologies like GPS navigation can only track the status and current location of vehicle. On the other hand, when vehicle monitoring is implemented using cloud computing, it is possible to incorporate centralized web service, GPS and GSM enabled devices, and embedded device with sensors [51], which will provide the following benefits:

(i) to identify the current name of the location, (ii) to predict the time of arrival,

(iii) to find status of driver via alcohol breath sensor , (iv) to find the total distance covered,

(v) to track the level of fuel.

All the data fetched are stored onto some centralized server that will be resided into the cloud. The vehicle owner can access this data on cloud via web portal and can retrieve all data on cloud in real time to visualize the vehicle information.

3.2.9. Tunnel Monitoring.

WSN can be used to implement the distributed sensing of light levels inside the tunnel and underbridges to provide necessary input information for adapting light functionality. This tunnel information can be put onto the cloud and is used to monitor the light intensity in real time to avoid the automobile users (drivers) casualty and to save the energy spent unnecessarily for lightening throughout the day.

3.2.10. Wildlife Monitoring.

Sensor-Cloud can also be used for tracking the wildlife sanctuaries, forests, and so forth to regularly monitor the endangered species in real time.

However, in this work, we have chosen to use our own cloud, since it gives us the advantage of customizing it according to our application requirements, so in order to address the above mentioned issues of deployment and testing of wireless sensor networks with the Clouds, we designed and implemented a flexible architecture for integrating WSN to Cloud using Firebase from Google Cloud Platform, which can be directly integrated into other applications.
The architecture presented in this work can be customized in different ways in order to accommodate different application scenarios with minimum redesign.

4. Description of proposed architecture

The architecture of the proposed system is divided into three parts: Sensor part, the Coordinator Layer and the Cloud Layer.

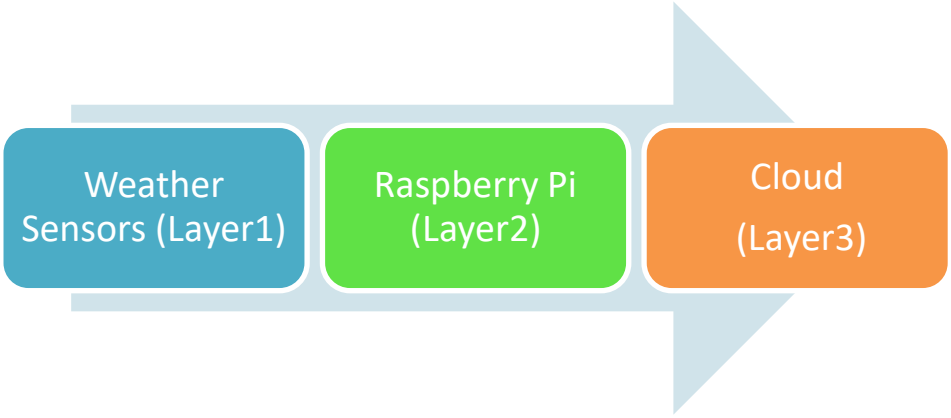

Figure.1 : Diagram of the architecture

1) The Sensor Layer consists of a wireless sensor network that can be roughly defined as the ensemble of spatially distributed, autonomous sensor that cooperate to monitor physical or environmental quantities of interest (temperature, humidity and rain drop) transducing the physical quantity of interest into an electric signal., such a large number of such wireless sensor nodes possibly interacting one with each other constitute a sensor network.

Formally speaking, different sensors are adopted, such as temperature, humidity and rain drop sensor**.** capable to simultaneously monitor environmental conditions at different locations; locally extracted information can be consequently forwarder to a peculiar sink node for further processing.

2)The Coordination Layer is responsible for the management of the data received from the sensor network. It temporarily stores the gathered data and sends it to the Supervision layer at predefined intervals. This base station which comprises of Raspberry pi (connected to the Internet and powered using an AC adaptor) has some computational resources and gathers data from wireless sensors using the communication protocols and sends this data to Cloud based sensor data platforms.

3) Finally, the Supervision Layer, the cloud accommodates the base station with Firebase to connect and publish the sensor data on the Internet. This layer stores the sensor data in a database and also offers a Web interface for the end users to manage the sensor data and generate statistics.

According to its implementation, we have used Firebase from GCP Services which provides many features to publish and access the sensor data and offers a graphical interface for real-time monitoring of systems to retrieve the sensor values using any device type and timestamp. Alerts can also be automatically generated to notify the user each time if the desired event has been sensed by the domain rules programmed in the base station.

5. Conclusion:

Once highlighted some applications on this subject, features and advantages of both WSNs and cloud computing, it is quite evident that these two paradigms can be mixed together to allow for easily sharing and analyzing real-time sensor data on-the-fly [6].

This mixture also allowed "us" too, for providing sensor data or sensor event as a service over the Internet, so that sensor data can be easily analyzed not only locally, but also from everywhere around the world.

We merged these two technologies to usefully exploite a large number of different applications, In the following sections, an application of cloud sensing for monitoring is developing on :

Weather Forecasting; data collected by environmental sensors represent an example of the so called “bigdata” issue, that cannot be easily maintained using the traditional database approaches [14], but could be profitably solved by low-cost large computational power of the cloud.

CHAPTER TWO : THE HARDWARE SETTINGS

1. Introduction:

This chapter outlines the design and the process of realizing the system and the Hardware implementation of what was stated in the previous chapters.

2. System Architecture :

As a proof of concept, here is a project designed and created to see changes in weather that gets data from weather sensors using Raspberry PI. And loads it into the cloud, using Firebase.

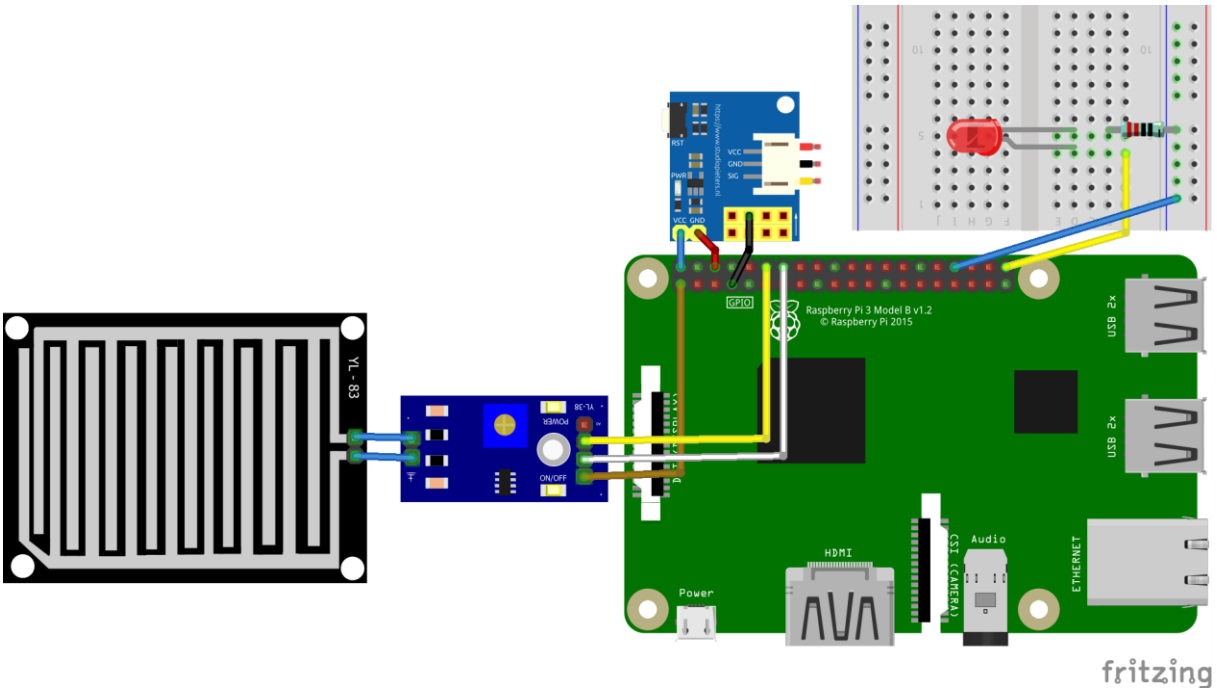

Figure.2: The Experiment « RPI Weather Station ».

2.2 Multiple Low-Cost Sensors :

In order to collect data and sense the physical quantity, like capturing the rapid climate changes, there is no doubt that an integrated sensor system must contain comprehensive sensors so that we can know the exact state in a localized area. The sensors need to be able to measure temperature, humidity, and the rain drop.

Sensor type	Sensed physical quantity	Cost (DZ)
DHT22	Temperature & Humidity	600
Rain Sensor	Rain Drop	700

Table.1 : Weather sensors.

2.2.1 The DHT-22 Sensor

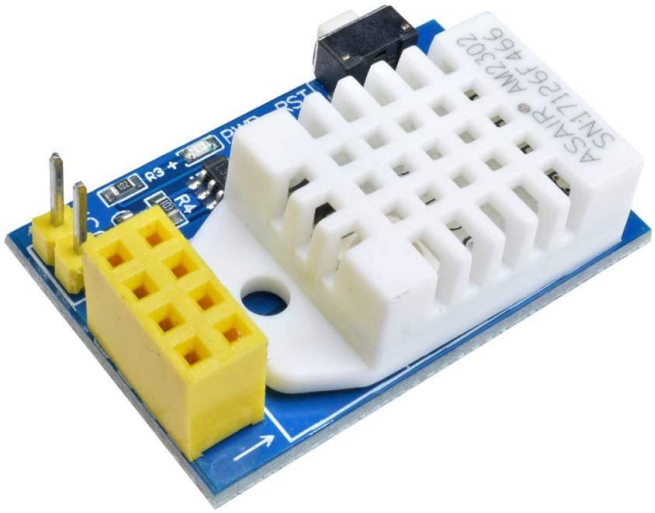

Figure.3: The DHT22 Sensor.

The DHT-22 is a digital-output relative humidity and temperature sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin.

The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).

Technical details:

- Power: 3-5V
- Max Current: 2.5mA
- Humidity: 0-100%, 2-5% accuracy
- Temperature: -40 to 80°C, ±0.5°C accuracy

Pin Configuration of DHT22 Sensor:

S.No	NAME	FUNCTION
1	VCC (the first pin on the left)	Connects supply voltage- 5V
2	GND(the second pin)	Connected to ground(P6).
3	D0 (The Digital data pin)	D0 outputGPIO 4 (P7)

Table.2 : Configuration of DHT22.

2.2.2 Rain drop Sensor Module

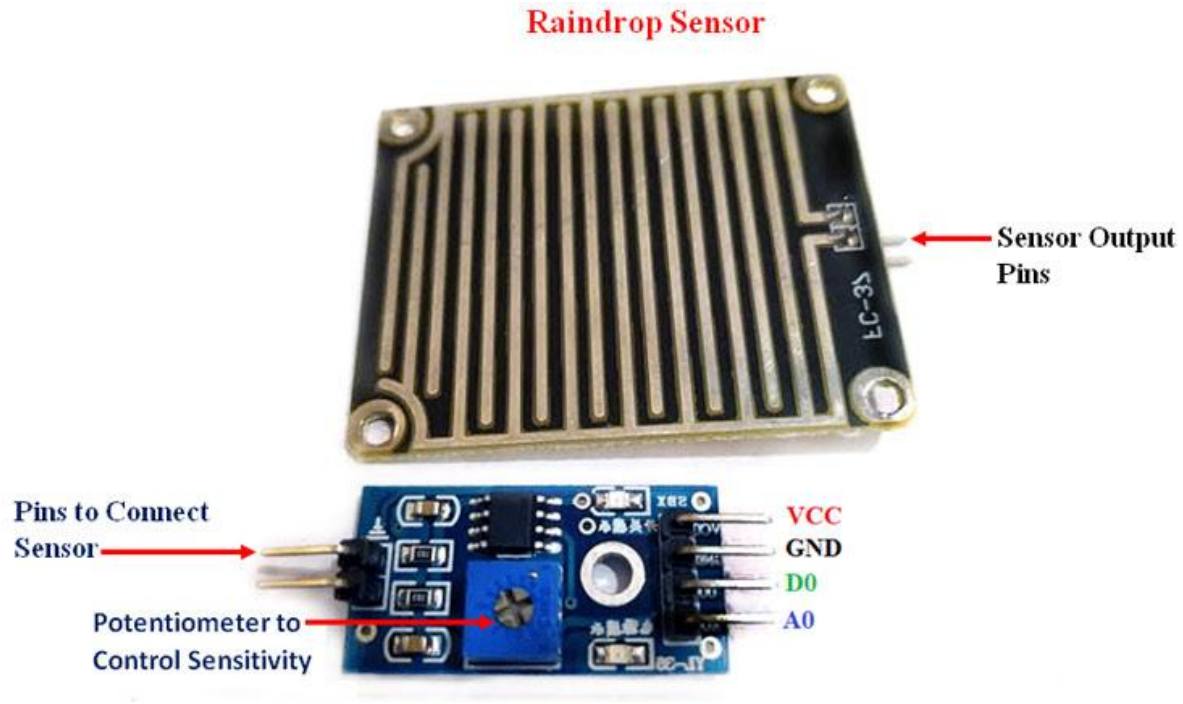

Figure.4: The Rain Drop Sensor.

Raindrop Sensor is a tool used for sensing rain. It consists of two modules, a rain board that detects the rain and a control module, which compares the analog value, and converts it to a digital value. The raindrop sensors can be used in the automobile sector to control the windshield wipers automatically, in the agriculture sector to sense rain and it is also used in automation systems.

Raindrop Sensor Features:

- Working voltage 5V
- Output format: Digital switching output (0 and 1), and analog voltage output AO
- Potentiometer adjust the sensitivity
- Comparator output signal clean waveform is good, driving ability, over 15mA
- Anti-oxidation, anti-conductivity, with long use time
- Easy installation
- Small board PCB size: 3.2cm x 1.4cm

Pin Configuration of Rain Sensor:

S.No	NAME	FUNCTION
1	VCC	Connects supply voltage3.3V
2	GND	Connected to ground (P 14)
3	D0	D0 pin GPIO 18 (P12)
4	A0 (No need)	A0 pin to get analog output

Table.3: Configuration of the Rain sensor.

How to use Raindrop sensor:

Interfacing the raindrop sensor is simple. The rain board module is connected with the control module of the raindrop sensor as shown in the below diagram.

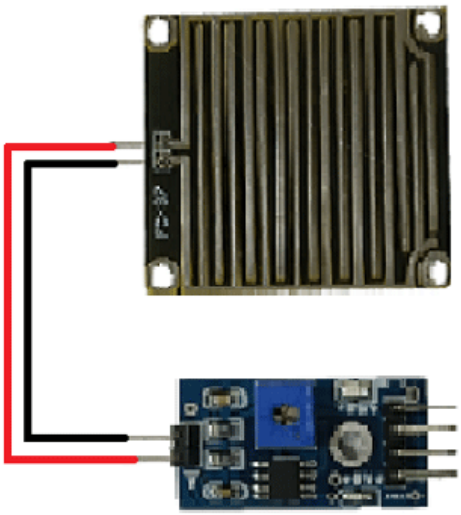

Figure.5: The Rain Drop module.

The control module of the raindrop sensor has 4 outputs. VCC is connected to a 5V supply. The GND pin of the module is connected to the ground. The D0 pin is connected to the digital pin of the Raspberry Pi for digital output (or the analog pin can be used). The sensor module consists of a potentiometer, LN393 comparator, LEDs, capacitors and resistors. The pinout image above shows the components of the control module. The rainboard module consists of copper tracks, which act as a variable resistor. Its resistance varies with respect to the wetness on the rainboard.The below fig shows the rain board module.

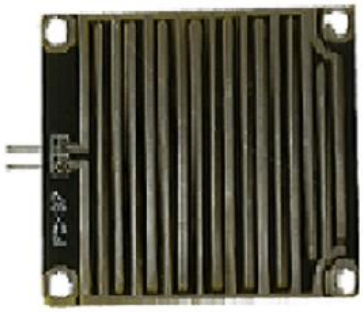

Figure.6: The Rain board Sensor.

The circuit diagram of a raindrop sensor module is given below.

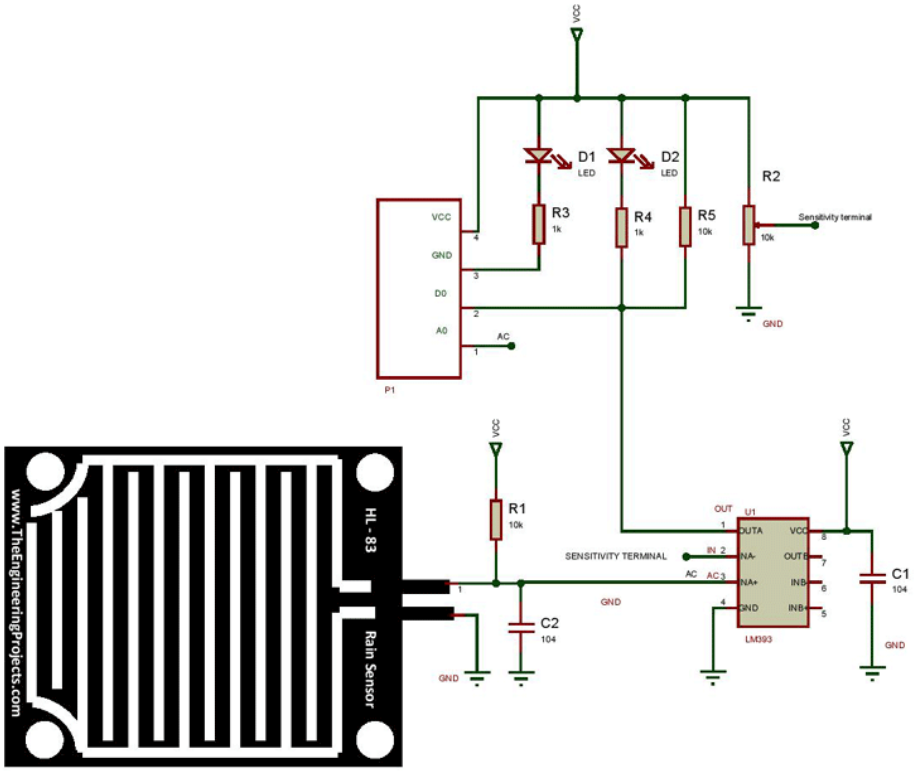

Figure.7: The circuit diagram of a raindrop sensor.

As shown in the above figure, the R1 resistor and the rain board module will act as a voltage divider. Capacitors C1 and C2 are used as a biasing element. The input for the Non-inverting terminal is taken from the connection point of the R1, and rain board module. Another point is taken from this connection and connected to the A0 terminal of the control module.

The input to the inverting terminal of the LM393 is taken from the potentiometer (R2). The R2 resistor acts as a voltage divider, and by varying R2 we can vary the input voltage to the inverting terminal, which in turn affects the sensitivity of the control module. The connections are shown in the above fig. The resistors R3 and R4 will act as current limiting resistors, while resistor R5 will act as a pull-up resistor to keep the bus in a high state when not in use.

Working of Rain Sensor:

Case1: When the input of the inverting terminal is higher than the input of the non-inverting terminal.

Case2: If the input of the inverting terminal is lower than the input of the non-inverting terminal.

The input to the inverting terminal is set to a certain value by varying the potentiometer and the sensitivity is set. When the rain board module's surface is exposed to rainwater, the surface of the rainboard module will be wet, and it offers minimum resistance to the supply voltage. Due to this, the minimum voltage will be appearing at the non-inverting terminal of LM393 Op-Amp. The comparator compares both inverting and non-inverting terminal voltages. If the condition falls under case(1), the output of the Op-Amp will be digital LOW. If the condition falls under case(2), the output of the Op-Amp will be digital HIGH.

The below diagram shows the equivalent circuit of both the conditions.

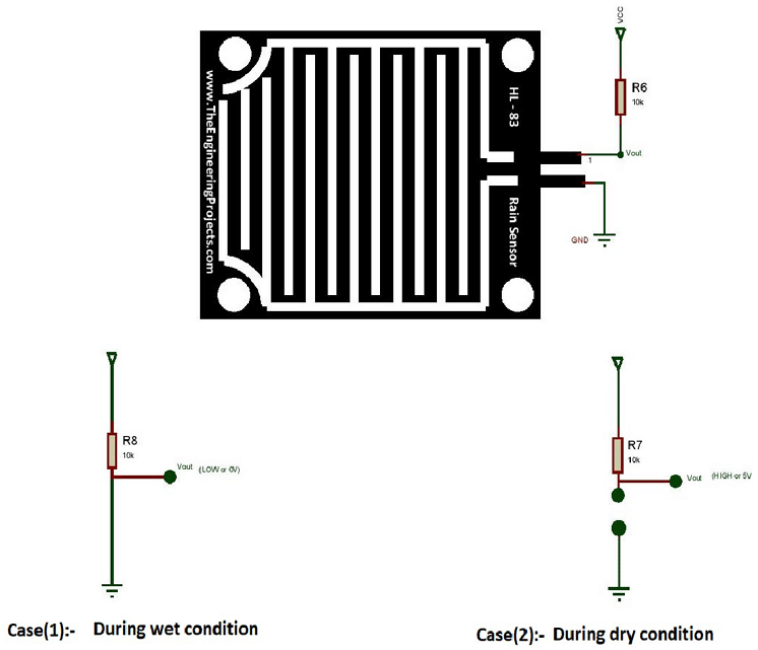

Figure.8: The circuit diagram of a raindrop sensor.

When the A0 pin is connected to the microcontroller, an additional analog to digital converter (ADC) circuit is used. In the case of Raspberry pi, it can be directly used for calculation purposes.

2.3 Raspberry Pi

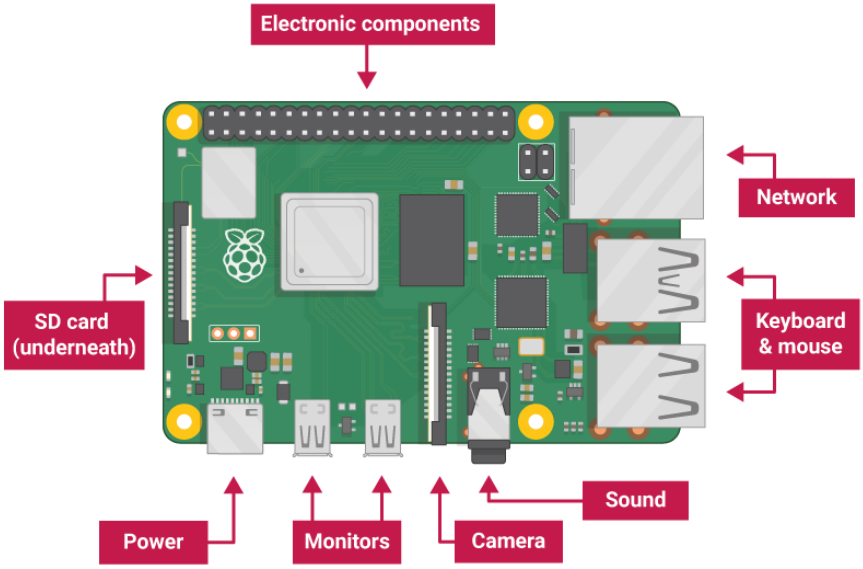

Figure.9: The Raspberry PI board.

Raspberry Pi board is one of the prevailing single-board computers specially designed for open-source development. We selected Raspberry 3 to construct sensor node. Raspberry 3 typically consumes power less than 2 W when conducting scientific computing. With a Broadcom BCM2836 system-on-chip (SoC), which consists of a 900 MHz quad-core ARM Cortex-A7 CPU and 1 GB RAM, Raspberry Pi is capable of some complex operations. What is more, Raspberry Pi has 4 USB ports, 40 GPIO pins, full HDMI port, ethernet port, display interface (DSI), micro SD card slot, and 3D graphics core, which provide more possibilities for its application.

As Figure -*System Architecture*- shows, all sensors are connected to Raspberry Pi through different GPios to start sensing and getting data. Raspberry Pi receives the data and then sends results to Firebase through Wi-Fi (wireless fidelity) communication module, for an eventuel treatment, Cloud Firestore will receive data package in real-time.

Note : (SSH into Raspberry Pi)

To run commands on the Raspberry Pi without needing to plug in a display, keyboard, mouse and having to move to the location of the Raspberry Pi each time, logging into the Pi via SSH (Secure Shell) from any other computer, ofc after running Raspbian on the Pi and successfully connect to a network via Ethernet or WiFi.

3. The Actuator

For event notification

An event notification system is also implemented with a LED, based on sensor measurements and predefined If conditions. This makes it possible to supervise and monitor the data recieved. When the server receives the weather data for each sensor through the Raspberry Pi, the LED will turn on flash alerts.

For example, it can send notification alert to user via push emails.

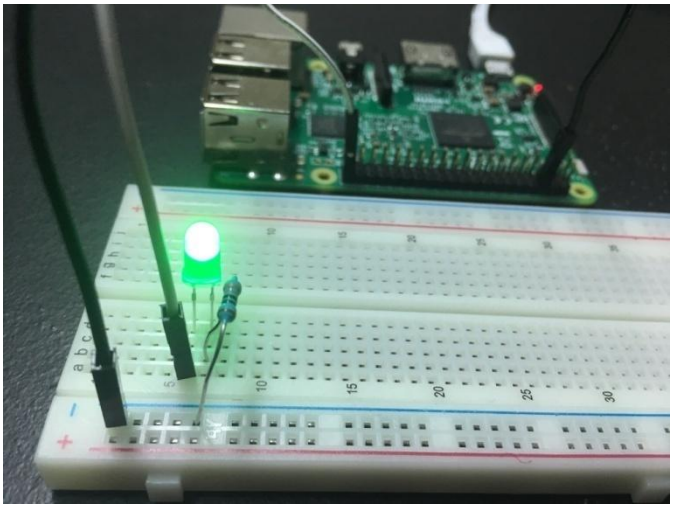

Figure.10: Connecting the LED with the circuit.

The LED has 2 legs. The longer leg, 'anode', is always connected to positive supply. The shorter leg,'cathode', is always connected to ground.

Components: To connect the circuit:

1. Raspberry Pi
2. LED
3. Resistor - 330 ohm to limit the amount of current in the circuit. Without the resistor the current flowing through the LED will be much larger and lead to a short damaging the circuit.
4. Breadboard
5. 2 Male-Female Jumper Wires

Connecting The Circuit :

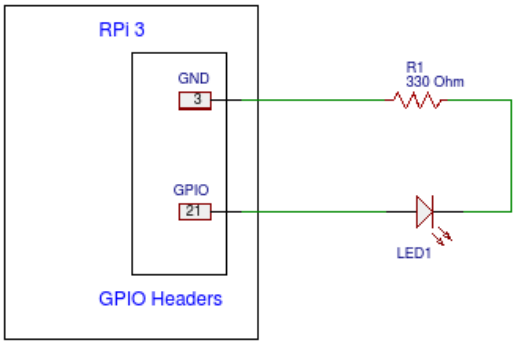

Figure.11: The LED circuit.

- A jumper wire to connect the ground (Pin 34) of GPIO to rail marked in blue on the breadboard.
- The resistor is connected from the same row on the breadboard to a column on the breadboard.
- Connect the LED with the cathode in the same row as the resistor. Insert the anode in the adjacent row.
- Another jumper cable to connect the GPIO Pin 21 (3.3 V) in the same row as the anode of LED.

Experiment picture :

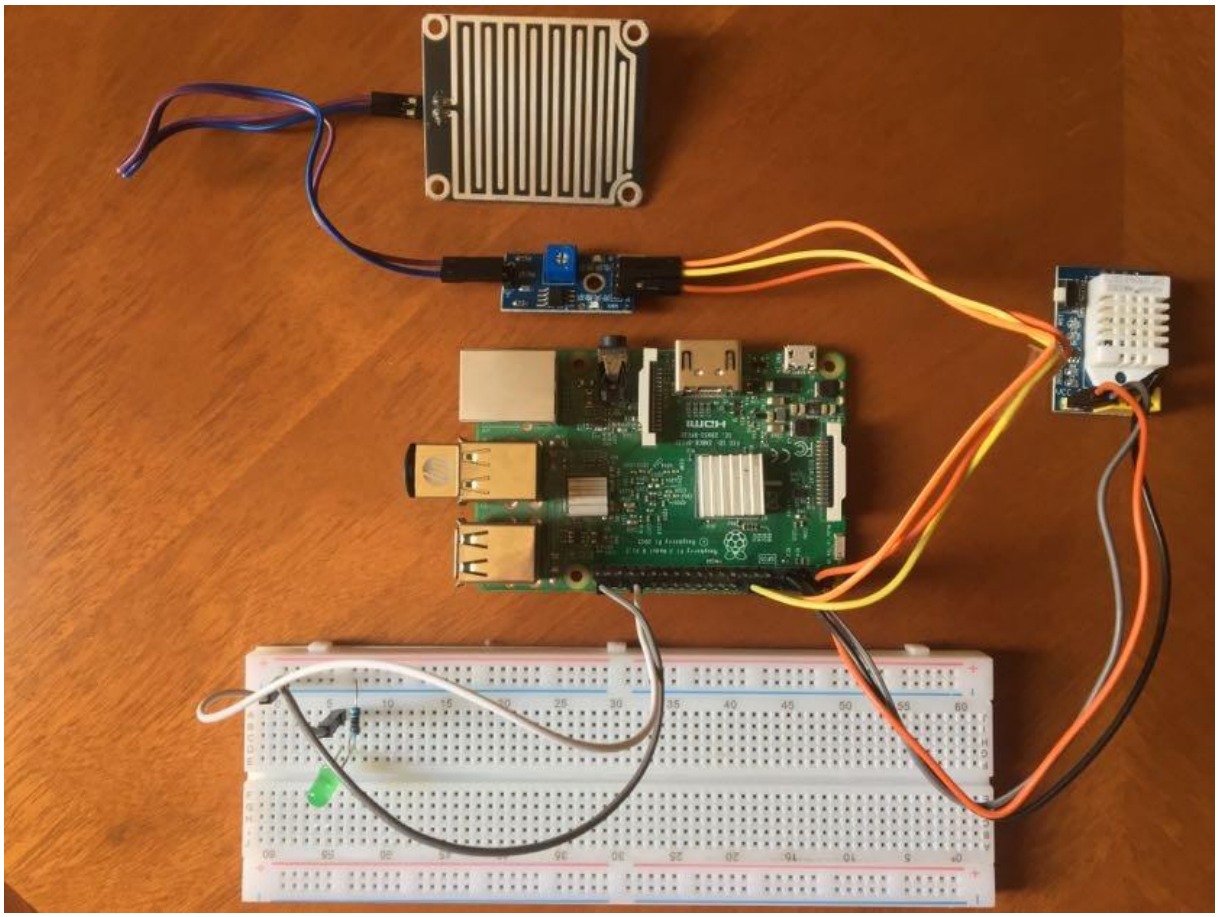

Figure.12: The Circuit.

4. Conclusion

This chapter provides details of the main parts of our system, which are low-cost sensors, local computing platform based on Raspberry Pi, and some informations are introduced to explore the internet of things and to see how the informations are collected from different devices and sensors, the tracked data (the weather changes) and even examples to react with the system.

CHAPTER THREE : THE SOFTWARE DEVELOPMENT

1. Introduction

Different software products were developed for this experiment in order to establish the sensor interface, configure the Pi and manage the sensed data for receiving, storing and publishing it on the Cloud. Each development phase is described as follows.

2. Information Processing Software

The obtained sensors data are recieved by the Raspberry PI. The cloud firestore from Firebase can get the sensor data stored on Raspberry Pi through its interaction with Raspberry Pi's data management system.

The sensors data exchanges between Raspberry Pi and the server « Firebase » which will be discussed in later part. The sensor data are interpreted by Node Js that is installed on Raspberry Pi. The data processing algorithm is the core of the whole software.

The programs is the test results of several algorithms and select the best fitting model according to the accuracy and efficiency based on the system chosen.

Raspberry Pi sends the data of the weather through Wi-Fi, and the data will be updated by Firebase periodically. The sensors data in certain period will be transmitted from Raspberry Pi to Firebase through GPIO protocols.

Communication and Sensor Layers in Raspberry Pi

To successfully communicate with the sensor nodes from the Raspberry PI, communication and sensor layers have been implemented on the RPI. The libraries in the communication layer are used to establish a reliable connection between the sensor nodes and to communicate with the server. The Node Js libraries are used to receive data on RPI.

The Raspberry Pi is connected to Firebase. Since Firebase already supports Node Js, we have focused on implementing software to connect it.

When the Raspberry Pi is turned on, it first connects to Node Js using a static IP address. Once the connection is successful, the Raspberry Pi requests for the data from the Sensors. Upon successful reception of data, These data are then updated on the Firebase platform using Cloud Firestore which is described in details later.

The layers of our software are shown in this Figure. From top to down :

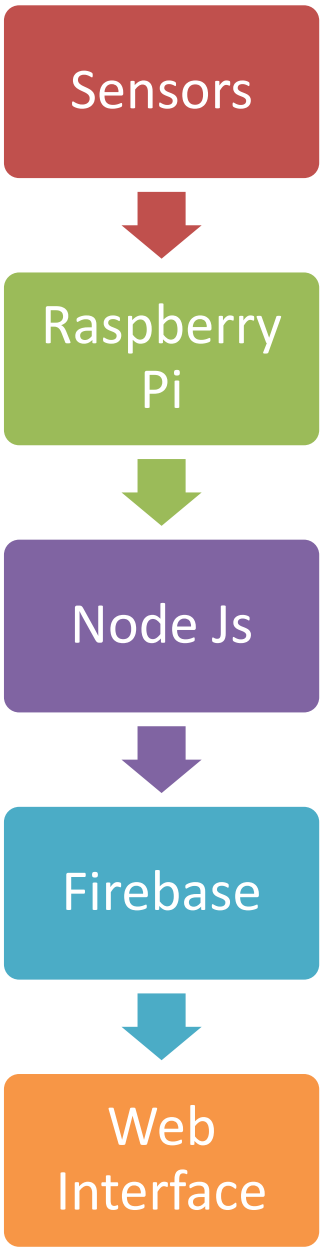

Figure.13: Software Architecture.

Connecting Sensor Network to Cloud Service

As mentioned previously, the access to Cloud services has to be easy, direct, open and interoperable. That is, the provided communication means and programming interfaces (APIs) shall be easy to implement on every platform and developing environment. One of the most open and interoperable ways to provide access to remote services or to enable applications to communicate with each other is to utilize Firebase from Google Cloud services.

Firebase

Firebase is a flexible, scalable database for mobile, web, and server development from Google Cloud Platform.

Cloud Firestore let us store the data in the cloud, so we can sync it across all our devices or share them among multiple users. And it comes with all the conveniences we'd expect from a Firebase product, like Libraries, full support for offline mode so the project will continue to work just fine whether it's connected or not even on the web, a comprehensive set of security rules to help you manage access, and an easy-to-use data browsing tool. It also lets us structure the data in ways that make sense to us. Cloud Firestore works in near real time, automatically fetching changes from our database as they happen, or we can request and fetch data manually. It's completely up to us.

Getting started with Firebase

Prerequisites

- Installing editor or IDE.
- Signing into Firebase using Google account.

Step 1: Creating a Firebase project

1. In the Firebase console, by clicking on **Add project**, then entering a **Project name**.

   Firebase automatically assigns a unique ID to the Firebase project.

2. **Continue**.
3. **Create project**.

Firebase automatically provisions resources for the Firebase project. When the process completes, it will be taken to the overview page for the Firebase project in the Firebase console.

Relationship between Firebase projects and Google Cloud Platform (GCP)

When creating a new Firebase project in the Firebase console, it's actually creating a Google Cloud Platform (GCP) project behind the scenes. A GCP project as a virtual container for data, code, configuration, and services**. A Firebase project is a GCP project that has additional Firebase-specific configurations and services.**Since a Firebase project *is* a GCP project:

- Projects that appear in the Firebase console also appear in the GCP console and Google APIs console.
- Billing and permissions for projects are shared across Firebase and GCP.
- Unique identifiers for a project (like project ID) are shared across Firebase and GCP.
- It can use products and APIs from both Firebase and GCP in your project.
- Deleting a project deletes it across Firebase and GCP.

Setting up a Firebase project :

The project name : When creating a project, it provide a project name. This identifier is the internal-only name for your project in the Firebase console, the GCP console, and the Firebase CLI. The project name is not exposed in any publicly visible Firebase or GCP product, service, or resource; it simply serves to help more easily distinguish the various projects.

Note: The project ID is the truly unique identifier for the project across all of Firebase and GCP.

The project ID : The Firebase project (and its associated GCP project) has a **project ID** which is the globally unique identifier for your project across all of Firebase and GCP. When creating a Firebase project, Firebase automatically assigns a unique ID to your project, but you can edit it during setup.

Firebase resources and the project ID : The project ID displays in publicly visible Firebase resources, for example:

- Default Hosting subdomain — projectID.web.app and projectID.firebaseapp.com
- Default Realtime Database URL — projectID.firebaseio.com
- Default Cloud Storage bucket name — projectID.appspot.com

For all of the aforementioned resources, it can create non-default instances. The publicly visible names of non-defaults are fully-customizable. And can connect custom domains to the Firebase-hosted site, shard the Realtime Database, and create multiple Cloud Storage buckets.

After Firebase provisions resources for the Firebase project, the project ID cannot be changed. To use a specific identifier for the Firebase resources, it must edit the project ID during the initial creation of the project.

Managing a Firebase project

Firebase console

The Firebase console offers the richest environment for managing Firebase project-level settings.

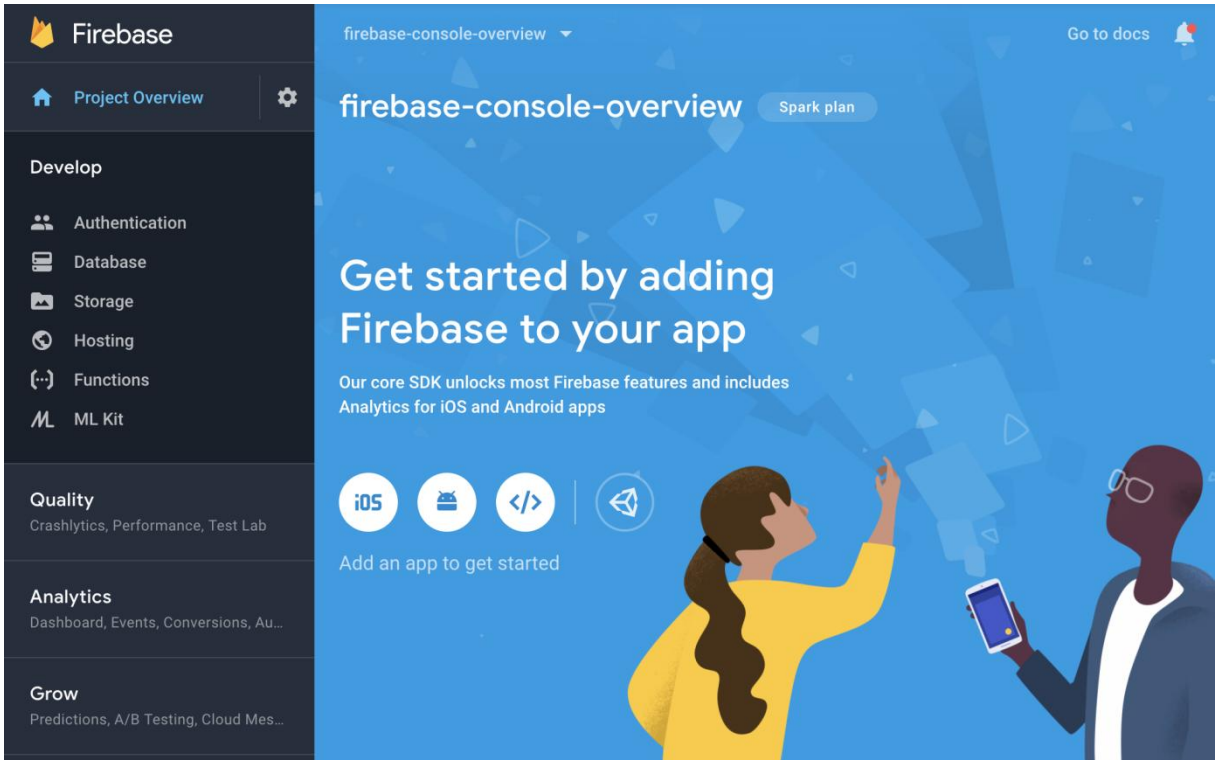

Figure.14: Firebase console

The left-side panel of the console lists the Firebase products, organized by top-level categories. At the top of the left-side panel, access the project settings by clicking settings (settings include integrations, access permissions, and billing).

The middle of the console displays buttons that launch setup workflows to add various types of apps. After you start using Firebase, the main area of the console changes into a dashboard that displays stats on the products you use.

Tools to manage the project Html, css, javascript & Node Js

HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript are the languages that run the web. They're very closely related, but they're also designed for very specific tasks. Node allows to use the same programming language Js on both the front and back-end sides.

- HTML is for adding meaning to raw content by marking it up.
- CSS is for formatting that marked up content.
- JavaScript is for making that content and formatting interactive.

The HTML is the abstract text and images behind the web page, CSS is the page that actually gets displayed, and JavaScript is the behaviors that can manipulate both HTML and CSS.

Node is growing so popular: it allows to use the same programming language on both the front and back-end sides.

And of course **Firebase** for HOSTING and REALTIME database.

Note : Mastering HTML, CSS, and JavaScript basics is a *prerequisite.*

For example, some particular run of text as a title with this HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>WELCOME TO MY WEATHER APP</title>
</head>
```

Then, the size and color can be set with some CSS:

```
.navbar__title {
        color: #ffffff;
        font-size: 24px;
        font-weight: bold;
      }
```

Adding Firebase to JavaScript app

1. In the center of the Firebase console's project overview page, the **Web** icon (plat_web) launch the setup workflow.
2. Enter the app nickname, it is an internal, convenience identifier and is only visible in the Firebase console.
3. *(Optional)* Set up Firebase Hosting for the web app.

- Firebase Hosting can be set at any time in the Project settings.
- To set up Hosting up now, select a site from the dropdown list to link to Firebase Web App.
- This list displays the project's default Hosting site and any other sites that have been set up in the project.
- Any site that is already linked to a Firebase Web App is unavailable for additional linking. Each Hosting site can only be linked to a single Firebase Web App.

Adding Firebase SDKs and initializing Firebase

Adding Firebase SDKs to the app depends on whether what is chosen to use Firebase Hosting for the app, what tooling are using with the app (like a bundler), or if are configuring a Node.js app.

Caution: The following instructions are for using the Firebase JavaScript SDK as a client for end-user access (for example, in a Node.js desktop or IoT application).

1. Install the Firebase JavaScript SDK:

a. Creat a `package.json` file, by running the following command from the root of JavaScript project:

```
npm init
```

b. Install the `firebase` npm package and save it to your `package.json` file by running:

```
npm install --save firebase
```

2. Use one of the following options to use the Firebase module in your app:

- **`Require` modules from any JavaScript file**

To include only specific Firebase products (like Authentication and Cloud Firestore):

```
// Firebase App (the core Firebase SDK) is always required and
// must be listed before other Firebase SDKs
var firebase = require("firebase/app");

// Add the Firebase products that you want to use
require("firebase/auth");
require("firebase/firestore");
```

Include the entire Firebase JavaScript SDK, rather than individual SDKs *(not recommended for production apps)*

- **ES2015 to `import` modules**

  To include only specific Firebase products (like Authentication and Cloud Firestore):

```
// Firebase App (the core Firebase SDK) is always required and
// must be listed before other Firebase SDKs
import * as firebase from "firebase/app";

// Add the Firebase services that you want to use
import "firebase/auth";
import "firebase/firestore";
```

  Include the entire Firebase JavaScript SDK, rather than individual SDKs *(not recommended for* production apps)

  Initialize Firebase in the app:

```
// TODO: Replace the following with your app's Firebase project configuration
// For Firebase JavaScript SDK v7.20.0 and later, `measurementId` is an optional field
var firebaseConfig = {
  // ...
};

// Initialize Firebase
firebase.initializeApp(firebaseConfig);
```

  Learning about the Firebase config object

  To initialize Firebase in your app, it needs to provide the app's Firebase project configuration.

- Manually editing the config object is not recommend, especially the following required "Firebase options": `apiKey`, `projectId`, and `appID`. If you initialize your app with invalid or missing values for these required "Firebase options", users of your app may experience serious issues.

  Here's the format of a config object with all services enabled (these values are automatically populated):

```
// For Firebase JavaScript SDK v7.20.0 and later, `measurementId` is an optional field
var firebaseConfig = {
  apiKey: "API_KEY",
  authDomain: "PROJECT_ID.firebaseapp.com",
  databaseURL: "https://PROJECT_ID.firebaseio.com",
  projectId: "PROJECT_ID",
  storageBucket: "PROJECT_ID.appspot.com",
  messagingSenderId: "SENDER_ID",
  appId: "APP_ID",
  measurementId: "G-MEASUREMENT_ID",
};
```

Here's a config object with values:

```
// For Firebase JavaScript SDK v7.20.0 and later, `measurementId` is an optional field
var firebaseConfig = {
  apiKey: "AIzaSyDOCAbC123dEf456GhI789jKl01-MnO",
  authDomain: "myapp-project-123.firebaseapp.com",
  databaseURL: "https://myapp-project-123.firebaseio.com",
  projectId: "mydht22",
  storageBucket: "myapp-project-123.appspot.com",
  messagingSenderId: "65211879809",
  appId: "1:65211879909:web:3ae38ef1cdcb2e01fe5f0c",
  measurementId: "G-8GSGZQ44ST"
};
```

(Optional) **Installing CLI and deploying to Firebase Hosting**

To deploy to Firebase, you'll use the Firebase CLI, a command-line tool.

1. Run the following command from the root of the local app directory:

```
firebase init
```

What does this initialization command do?

2. Deploy your content and hosting configuration to Firebase Hosting. By default, every Firebase project has free subdomains on the web.app and firebaseapp.com domains (***project-id***.web.app and ***project-id***.firebaseapp.com).

a. Deploy the site. Run the following command from the app's root directory:

```
firebase deploy
```

b. View the site at either of default sites:

- ***project-id***.web.app
- ***project-id***.firebaseapp.com

1. Add a defer flag to each script tag for the Firebase SDKs, then defer the initialization of Firebase using a second script, for example:

```
<script defer src="https://www.gstatic.com/firebasejs/7.21.1/firebase-app.js"></script>

<script defer src="https://www.gstatic.com/firebasejs/7.21.1/firebase-auth.js"></script>
<script defer src="https://www.gstatic.com/firebasejs/7.21.1/firebase-firestore.js"></script>

// ...

<script defer src="./init-firebase.js"></script>
```

2. Create an init-firebase.js file, then include the following in the file:

```
// TODO: Replace the following with your app's Firebase project configuration
var firebaseConfig = {
  // ...
};

// Initialize Firebase
firebase.initializeApp(firebaseConfig);
```

Run a local web server for development

Some parts of the Firebase JavaScript SDK require that you serve your web app from a server rather than from the local filesystem. use the Firebase CLI to run a local server.

To serve your web app, use the Firebase CLI, a command-line tool.

1. Initialize Firebase project. Run the following command from the root of local app directory:

```
firebase init
```

What does this initialization command do?

2. Start the local server for development. Run the following command from the root of local app directory:

```
firebase serve
```

HTML, CSS, and JavaScript are totally different languages, but they all refer to one another in some way. The app rely on all three, but the appearance of *every* website is determined by HTML and CSS.

3. Implementation Results and Discussion

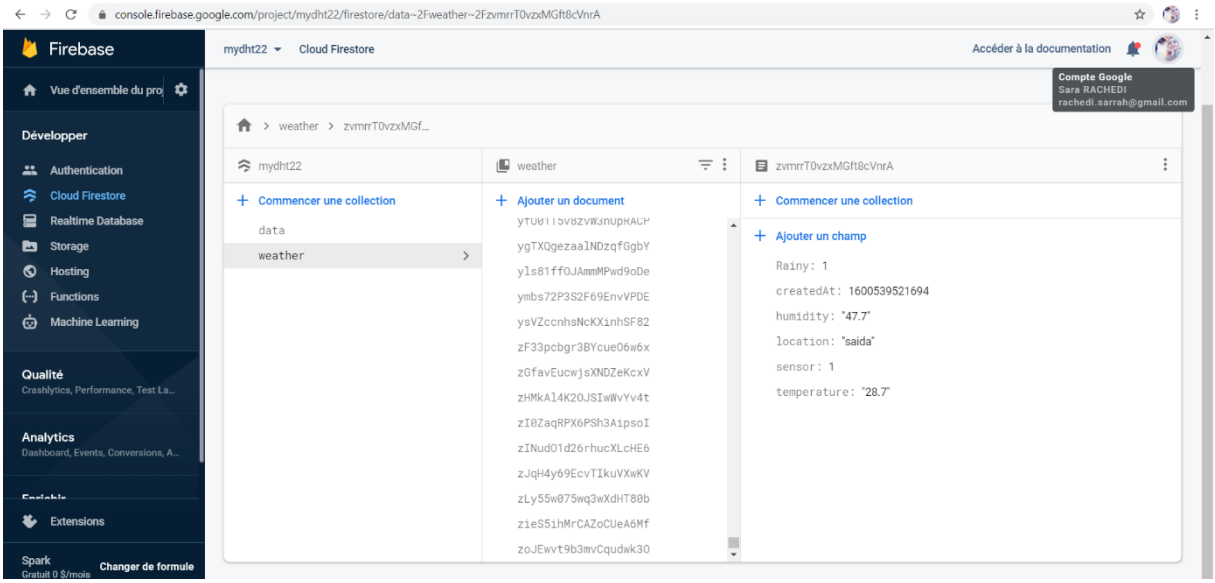

Figure.15: The Firebase project

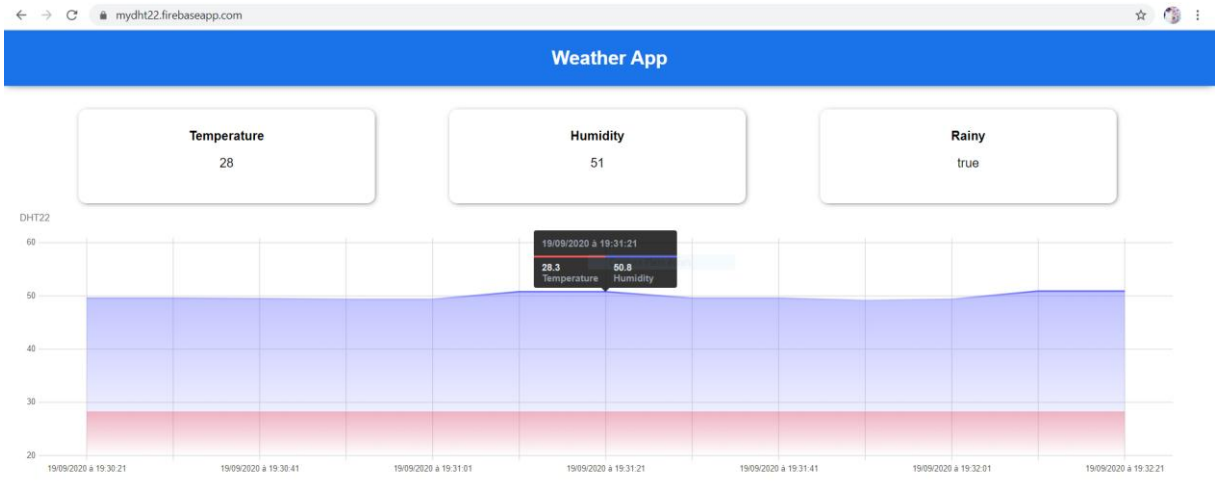

Figure.16: The application interface.

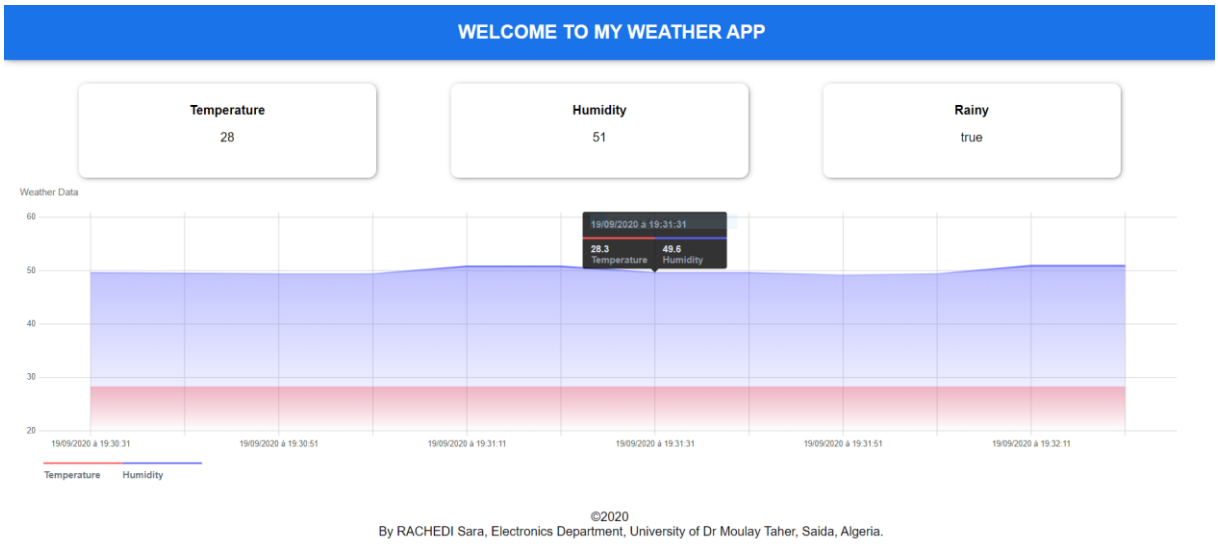

Figure.16: The application interface.

In order to evaluate and demonstrate the proposed model, we implement it by using the technical approach which is described in the above chapters. A WSN was created to collect temperature, humidity and rain drop readings and to connect the Sensor Network to the Cloud Services. Preliminary experiments were performed to evaluate the system in terms of sensor data accessibility and alert notification. Furthermore, a Web and mobile Application was created on Firebase from Google Cloud Platform to present the collected data in an easy and meaningful way.

Limitations of the Study

This study raised a number of issues and questions that may provide a basis for future research. This is partly due to some limitations identified in this study and partly because of issues and concerns that rose in the analysis and could not be pursued as part of this inquiry. Thus, a brief reference to the limitations will be presented.

This is a short term study conducted over a space of one semester. As it was not a longitudinal study and did not allow the research to deal with more sensors, any conclusions established do not provide a full picture of the effects of the Competency-Based Approach.

4. Conclusion

The system has been working perfectly so far. All sensor data and results conducted in Raspberry Pi present enormous significance on Firebase in an easy and meaningful way, especially in exploring local meteorological differences, Temperature, humidity and warning of heavy rain.

CONCLUSIONS

The IoT, i.e. the capability to interconnect every possible device, opens new scenarios in WSNs. Cloud computing services and the availability of powerful and inexpensive smart devices allow to optimize information management, sharing measurement results and improving quality of services.

In addition, smartdevice sensing capabilities are improving, and the development of these wireless sensor networks requires technologies from three very different research areas, i.e. technologies related to the development of the sensor, of the communication device, and of computing device (not limited to the hardware, but also including software and algorithms). Combined and separate advancements in each of these areas have driven research in this field.

This work proposed a flexible architecture for integration of Wireless Sensor Networks to the Cloud for sensor data collection and sharing using Firebase from Google Cloud services.

Accurate and in-time weather data is a core challenge in environmental sensor networks. we developed a Raspberry Pi based intelligent wireless sensor node that can self-collect environmental data and publish forecast in the cloud using Wi-Fi.

The designed system can be directly integrated into other applications, to avoid loss of data and we embedded intelligence at different architectural layers to accommodate for the diverse requirements of possible application scenarios with minimum redesign and recoding. The evaluation results illustrate that the sensor data can be accessed by the users anywhere and on any mobile device with internet access.

ملخـــــــص

إنترنت الأشياء ، أي القدرة على ربط كل جهاز ممكن ، يفتح سيناريوهات جديدة في شبكات WSN. تتيح خدمات الـcloud الحوسبة السحابية، وتوافر الأجهزة الذكية القوية وغير المكلفة تحسين إدارة المعلومات ومشاركة نتائج القياس وتحسين جودة الخدمات.

بالإضافة إلى ذلك ، تتحسن قدرات استشعار الأجهزة الذكية ، ويتطلب تطوير شبكات الاستشعار اللاسلكية هذه تقنيات من ثلاثة مجالات بحث مختلفة، مثل التقنيات المتعلقة بتطوير المستشعر وجهاز الاتصال وجهاز الحوسبة (لا يقتصر على الأجهزة ، ولكن أيضًا بما في ذلك البرامج والخوارزميات). دفعت التطورات المشتركة والمنفصلة في كل من هذه المجالات البحث في هذا المجال.

اقترح هذا العمل بنية مرنة لدمج شبكات الاستشعار في السحابة لجمع بيانات المستشعر ومشاركتها باستخدام Firebase من خدمات Google Cloud.

تمثل بيانات الطقس الدقيقة وفي الوقت المناسب تحديًا أساسيًا في شبكات الاستشعار البيئية. قمنا بتطوير عقدة حساس ذكية قائمة على Raspberry Pi يمكنها جمع البيانات البيئية ونشرها في السحابة.

يمكن دمج النظام المصمم مباشرة في العديد من التطبيقات الأخرى، لتجنب فقدان البيانات، اذ قمنا بتصميم النموذج في طبقات مختلفة لاستيعاب المتطلبات المتنوعة لسيناريوهات التطبيق الممكنة مع الحد الأدنى من إعادة التصميم وإعادة الترميز. توضح نتائج التقييم أنه يمكن الوصول إلى بيانات المستشعر من قبل المستخدمين في أي مكان في أي وقت وعلى أي جهاز محمول متصل بالإنترنت.

References

[1] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, et al., "ALARM-NET: Wireless sensor networks for assisted-living and residential monitoring," University of Virginia Computer Science Department Technical Report, 2006.

[2] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," in International workshop on wearable and implantable body sensor networks, 2004.

[3] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, et al., "Deploying a wireless sensor network on an active volcano," Internet Computing, IEEE, vol. 10, pp. 18-25, 2006.

[4] J. Tooker, X. Dong, M. C. Vuran, and S. Irmak, "Connecting soil to the cloud: A wireless underground sensor network testbed," in Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on, 2012, pp. 79-81.

[5] F. Kausar, E. Al Eisa, and I. Bakhsh, "Intelligent Home Monitoring Using RSSI in Wireless Sensor Networks," International Journal of Computer Networks & Communications (IJCNC), vol. 4, pp. 33-46, 2012.

[6] H. ElAarag, D. Bauschlicher, and S. Bauschlicher, "System Architecture of HatterHealthConnect: An Integration of Body Sensor Networks and Social Networks to Improve Health Awareness," International Journal of Computer Networks & Communications, vol. 5, p. 22, 2013.

[7] P. A. C. d. S. Neves and J. J. P. C. Rodrigues, "Internet Protocol over Wireless Sensor Networks, from Myth to Reality," JOURNAL OF COMMUNICATIONS, vol. 5, pp. 189-195, 2010.

[8] M. R. Kosanović and M. K. Stojčev, "CONNECTING WIRELESS SENSOR NETWORKS TO INTERNET," FACTA UNIVERSITATIS, Mechanical Engineering, vol. 9, pp. 169-182, 2011.

[9] A. E. Kouche, "Towards a wireless sensor network platform for the Internet of Things: Sprouts WSN platform," in Communications (ICC), 2012 IEEE International Conference on,, 2012, pp. 632-636.

[10] B. Li and J. Yu, "Research and Application on the Smart Home Based on Component Technologies and Internet of Things," Procedia Engineering, vol. 15, pp. 2087-2092, // 2011.

[11] N. Mitton, S. Papavassiliou, A. Puliafito, and K. S. Trivedi, "Combining Cloud and sensors in a smart city environment," EURASIP Journal on Wireless Communications and Networking, vol. 2012, p. 247, 2012.

[12] D. Guinard and V. Trifa, "Towards the web of things: Web mashups for embedded devices," in Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain, 2009.

[13] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: design and implementation of interoperable and evolvable sensor networks," in Proceedings of the 6th ACM conference on Embedded network sensor systems, 2008, pp. 253-2

[14] Liu, Y.; Hu, L.; Yang, D.; Liu, H. Air-Sense: Indoor environment monitoring evaluation system based on ZigBee network. IOP Conf. Ser. Earth Environ. Sci. 2017, 81, 12208.

[15] Yang, J.; Zhou, J.; Lv, Z.; Wei, W.; Song, H. A Real-Time Monitoring System of Industry Carbon Monoxide Based on Wireless Sensor Networks. Sensors 2015, 15, 29535–29546. [CrossRef]

[16] Botta, A.; de Donato, W.; Persico, V.; Pescapé, A. Integration of Cloud computing and Internet of Things: A survey. Futur. Gener. Comput. Syst. 2016, 56, 684–700. [CrossRef]

[17] Díaz, M.; Martín, C.; Rubio, B. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. J. Netw. Comput. Appl. 2016, 67, 99–117. [CrossRef]

[18] Kumar, P.M.; Lokesh, S.; Varatharajan, R.; Gokulnath, C.; Parthasarathy, P. Cloud and IoT based disease prediction and diagnosis system for healthcare using Fuzzy neural classifier. Futur. Gener. Comput. Syst. 2018. [CrossRef]

[19] Gachet, D.; De Buenaga, M.; Aparicio, F.; Padron, V. Integrating internet of things and cloud computing for health services provisioning: The virtual cloud carer project. In Proceedings of the 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Palermo, Italy, 4–6 July 2012; pp. 918–921.

[20] Ray, P.P. A survey of IoT cloud platforms. Futur. Comput. Informat. J. 2016, 1, 35–46.[CrossRef]

[21] OpenIoT Web Page. Available online: http://www.openiot.eu/

[22] xively Web Page. Available online: https://xively.com/

[23] ThingSpeak Web Page. Available online: https://thingspeak.com/

[24] CloudPlugs Web Page. Available online: https://cloudplugs.com/

[25] Device Cloud Web Page. Available online: https://devicecloud.digi.com

[26] Thinking Things Web Page. Available online https://iot.telefonica.com/thinking-things

[27] SensorCloud Web Page. Available online:http://www.sensorcloud.com/

[28] Amazon Web Services Web Page. Available online: https://aws.amazon.com/

[29] Google Cloud Platform. Available online:https://cloud.google.com

[30] Aleixandre, M.; Gerboles, M. Review of small commercial sensors for indicative monitoring of ambient gas.Chem. Eng. Trans. 2012, 30, 169–174. [CrossRef]

[31] Lewis, A.C.; Lee, J.; Edwards, P.M.; Shaw, M.D.; Evans, M.J.; Moller, S.J.; Smith, K.; Ellis, M.; Gillott, S.;White, A.; et al. Evaluating the performance of low cost chemical sensors for air pollution research. FaradayDiscuss. **2016**, 189, 85–103. [CrossRef] [PubMed]

[32] Piedrahita, R.; Xiang, Y.; Masson, N.; Ortega, J.; Collier, A.; Jiang, Y.; Li, K.; Dick, R.P.; Lv, Q.; Hannigan, M.;et al. The next generation of low-cost personal air quality sensors for quantitative exposure monitoring.Atmos. Meas. Tech. 2014, 7, 3325–3336. [CrossRef]

[33] [8] M. Yuriyama, T. Kushida, and M. Itakura, "A new model of accelerating service innovation with sensor-cloud infrastructure," in Proceedings of the annual SRII Global Conference (SRII'11), pp. 308–314, 2011.

[34] [9] J. Yick, B. Mukherjee, and D. Ghosal, Wireless Sensor NetworkSurvey, Elsevier, 2008.

[35] [10] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches, Wireless Communications and Mobile Computing-Wiley Online Library, 2011.

[36] [11] W. Kim, "Cloud computing: today and tomorrow," Journal of Object Technology, vol. 8, pp. 65–72, 2009.

[37] [12] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure physical sensor management with virtualized sensors on cloud computing," in Proceedings of the IEEE 13th International Conference on Network-Based Information Systems (NBiS '10), pp. 1–8, September 2010.

[38] Nimbits Data Logging Cloud Sever, http://www.nimbits.com.

[39] Pachube Feed Cloud Service, http://www.pachube.com.

[40] iDigi—Device Cloud, http://www.idigi.com.

[41] C. Doukas and I. Maglogiannis, "Managing wearable sensor data through cloud computing," in Proceedings of the IEEE 3rd International Conference on Cloud Computing, 2011.

[42] G. Demiris, B. K. Hensel, M. Skubic, and M. Rantz, "Senior residents' perceived need of and preferences for "smart home" sensor technologies," International Journal of Technology Assessment in Health Care, vol. 24, no. 1, pp. 120–124, 2008.

[43] K. Lee, D. Murray, D. Hughes, and W. Joosen, "Extending sensor networks into the Cloud using Amazon web services," in Proceedings of the 1st IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA '10), pp. 1–7, November 2010.

[44] B. Jit, J. Maniyeri, S. Louis, K. Gopalakrishnan, and P. Yap, "Design and trial deployment of a practical sleep activity pattern monitoring system," in Proceedings of the International Conference on Smart Homes and Health Telematics (ICOST '09), Tours, France, June 2009.

[45] B. Jit, J. Maniyeri, S. Louis, and L. K. P. Yap, "Fast matching of sensor data with manual observations," in Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society: Engineering the Future of Biomedicine (EMBC '09), pp. 1675–1678, September 2009.

[46] N. Kurata, M. Suzuki, S. Saruwatari, and H. Morikawa, “Actual application of ubiquitous structural monitoring system using wireless sensor networks,” in Proceedings of the 14th World Conference on Earthquake Engineering (WCEE ’08), 2008.

[47] Google Health, http://www.google.com/health.

[48] Korea u-Life care system.

[49] http://www.apan.net/meetings/HongKong2011/Session/Agriculture.php/.

[50] H. H. Tran and K. J. Wong, “Mesh networking for seismic monitoring—the sumatran cGPS array case study,” in Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC ’09), April 2009.

[51] Tunnel Monitoring System: http://www.advantech.com/intelligent-automation/Industry%20Focus/%7BC274D52C-95- D2-499E-9E16-6C1F41D1CD6/.