SEBHA UNIVERSITY JOURNAL OF PURE & APPLIED SCIENCES VOL.XX NO. X 20XX
DOI: 10.51984/JOPAS.V24I1.3798

مجلة جامعة سبها للعلوم البحتة والتطبيقية
Sebha University Journal of Pure & Applied Sciences

Journal homepage: www.sebhau.edu.ly/journal/jopas

Advancing Environmental Monitoring Systems Using IoT and Sensors

Atiya Alsnousi[a*], Asma Agaal[a], Ateya Mohmed[b], Ali Mohmmed[a], Abbas Abubaker[a]

[a]Department of Computer Technologies, Faculty of Technical Sciences, Sebha, Libya
[b]Department of Electrical and Ecteronic, Faculty of Technical Sciences, Sebha, Libya

Keywords:
Internet of Things (IoT).
Wireless Sensor Networks (WSN).
Sensor Nodes.
Soil Moisture Sensor.
Temperature Sensor.
Light Dependent Resistor (LDR) Sensor.

A B S T R A C T

This paper presents the design and implementation of an environmental monitoring system for Sebha International Airport, Libya, utilizing the Internet of Things (IoT) and a Wireless Sensor Network (WSN). The system integrates sensor nodes to monitor temperature, light intensity, and soil moisture. Each node incorporates a Negative Temperature Coefficient (NTC) thermistor (TTC05203), a Cadmium Sulfide (CdS) photoresistor (LDR), and, in one node, a soil moisture sensor based on resistivity. Data acquisition is managed by PIC18F4620 microcontrollers, using Time-Division Multiple Access (TDMA) on a 2.4 GHz Zigbee network for reliable wireless communication. A Raspberry Pi Zero W acts as a gateway, collecting data from the WSN and uploading it to the Thing Speak IoT platform for visualization and analysis. The system successfully monitors environmental conditions in real time, with Cyclic Redundancy Check (CRC) ensuring data integrity. This solution enhances passenger comfort by automating climate control based on real-time temperature and light readings, while also supporting sustainable management of airport green spaces through soil moisture monitoring. Further research will explore incorporating additional environmental parameters, such as air quality and noise levels, and developing predictive analytics capabilities.

تطوير أنظمة مراقبة البيئة باستخدام إنترنت الأشياء وأجهزة الاستشعار

عطية السنوسي[a*]، أسمة أعجال[a]، عطية محمد[b]، علي محمد[a]، وعباس أبوبكر[a]

[a]قسم تقنيات الحاسوب، كلية العلوم التقنية- سبها, ليبيا.
[b]قسم الهندسة الكهربائية والالكترونية كلية العلوم التقنية- سبها, ليبيا.

الكلمات المفتاحية:
إنترنت الأشياء.
شبكات المستشعرات اللاسلكية.
عقد المستشعرات.
مستشعر رطوبة التربة.
مستشعر درجة الحرارة.
مستشعر المقاومة المعتمدة على الضوء.

الملخص

تقدم هذه الورقة تصميم وتنفيذ نظام مراقبة بيئي لمطار سبها الدولي، ليبيا، باستخدام إنترنت الأشياء (IoT) وشبكة استشعار لاسلكية (WSN). يدمج النظام عقد استشعار لمراقبة درجة الحرارة وشدة الضوء ورطوبة التربة. تتضمن كل عقدة ترموستورًا بمعامل درجة حرارة سلبي (NTC) (TTC05203)، ومقاوم ضوئي من كبريتيد الكادميوم (CdS)، وفي عقدة واحدة، مستشعر رطوبة التربة بناءً على المقاومة. تتم إدارة اكتساب البيانات بواسطة متحكمات PIC18F4620، باستخدام الوصول المتعدد بتقسيم الزمن (TDMA) على شبكة Zigbee 2.4 جيجا هرتز للاتصال اللاسلكي الموثوق. يعمل Raspberry Pi Zero W كبوابة، حيث يجمع البيانات من WSN ويحملها إلى منصة ThingSpeak IoT للتصور والتحليل. يراقب النظام بنجاح الظروف البيئية في الوقت الفعلي، مع ضمان سلامة البيانات من خلال فحص التكرار الدوري (CRC). يعزز هذا الحل راحة الركاب من خلال أتمتة التحكم في المناخ استنادًا إلى قراءات درجة الحرارة والضوء في الوقت الفعلي، مع دعم الإدارة المستدامة للمساحات الخضراء في المطار من خلال مراقبة رطوبة التربة. ستستكشف الأبحاث الإضافية دمج معلمات بيئية إضافية، مثل جودة الهواء ومستويات الضوضاء، وتطوير قدرات التحليلات التنبؤية.

*Corresponding author.
E-mail addresses: auhida79@gmail.com, (A. Agaal) asma.agaal@sebhua.edu.ly, (A. Mohmed) attiafakhiri62@yahoo.com, (A. Mohmmed) alibusaada666@ctss.edu.ly, (A Abubaker) Yabbas5052@yahoo.com.

Article History : Received 02 Februray 25 - Received in revised form 30 April 25 - Accepted 26 May 25

1.Introduction

Airports are bustling ecosystems with complex infrastructures and diverse environmental considerations. As global awareness of environmental issues rises, airports face mounting pressure to innovate and optimize their operations for sustainability and efficiency, airports face mounting pressure to innovate and optimize their operations for sustainability and efficiency [1]. Consequently, there exists a pressing need for solutions that can not only monitor and regulate environmental variables in airports, but also adapt in real-time to the evolving needs [2]. Additional environmental monitoring methods have evolved significantly with the advent of advanced technologies the relentless advancement of wireless technology and embedded electronics has spurred the utilization of compact devices across diverse sectors. These devices are instrumental in transforming conventional environments [3]. Characterized by their capacity for sensing, computation, and wireless communication, these devices facilitate the exchange of data obtained through integrated sensors [4]. Environmental monitoring systems present an innovative solution for in the pursuit of environmental sustainability by harnessing the synergy between Internet of Things (IoT) technology, wireless sensors, this integration enables informed decisions for conservation and savings while controlling environmental quality using various parameters [5]. The Internet of Things (IoT) and Wireless Sensor Networks (WSNs) holds immense potential for enhancing the functionality and interconnectedness of environmental Monitoring systems within airports, by deploying IoT-enabled sensors and devices, airports can create a networked ecosystem capable of capturing real-time environmental data from diverse sources, and optimize their operations for sustainability and efficiency [1,6].
particularly in the domain of monitor and manage the factors impacting the environment more precisely in real time. IoT devices and applications serve to process data acquired from (WSNs) deployments, enabling the transmission of this information to remote locations, thereby facilitating comprehensive monitoring, advanced analysis, and responsive control functionalities [7].
The Internet of Things (IoT) can be defined as a network encompassing physical devices, vehicles, appliances, and other tangible objects equipped with sensors, software, and network connectivity, enabling collaborative action towards shared objectives [8]. A Wireless Sensor Network (WSN) comprises a collection of sensor and routing nodes, which operate on AI based monitoring and controlling methods [9]. Wireless sensor networks (WSNs), which use modern sensors and, (IoT) to monitor and control the environment, are becoming increasingly popular [10].
The proposed system is intended to improve both comfort levels and operational efficiency within the Sebha International Airport. The elevated temperatures frequently observed at airport can lead to considerable discomfort for the large volume of travelers passing through the facility. To mitigate this, temperature sensors will be strategically deployed to automatically activate the air conditioning system when temperatures exceed 23°C, a threshold approximating human body temperature. This proactive strategy ensures a comfortable environment for passengers, especially during peak periods of heat. Conversely, during winter months, should temperatures fall below 18°C, the system will initiate heating to maintain a warm and welcoming ambiance, by regulate heating, ventilation, and air conditioning systems to maintain comfort while minimizing energy use.
Airport waiting areas serve as spaces for passenger relaxation, necessitating the maintenance of a comfortable and inviting atmosphere. Utilizing light-dependent sensors (LDRs) , the system continuously measures ambient light levels and dynamically regulates the brightness of indoor lights through microcontroller-based circuitry, when natural light is sufficient, the system dims or turns off the indoor lights; conversely, when natural light is low, it increases the artificial light intensity to meet the required level. This adaptive mechanism not only reduces energy consumption but also enhances user comfort by providing consistent lighting conditions. Furthermore, the airport's garden areas require diligent management to promote healthy growth. A soil moisture sensor will be integrated to monitor soil moisture content. When moisture levels fall below 15%, the system will trigger irrigation to sustain the health of the garden.
This system is designed not only to enhance passenger comfort through effective climate management but also to contribute to the long-term sustainability of the airport's green infrastructure. It provides control over energy usage, offering actionable insights and recommendations for improving efficiency and ensuring reliable energy management. The implementation of these technologies underscores a commitment to enhancing the overall airport experience while adapting to demanding environmental conditions.
The remainder of this paper is structured as follows: Part I provides an overview of the research objectives. Part II presents a review of relevant literature and related works, including a discussion of the study's progress. Part III details the design and implementation of the sensor systems. Part IV presents the results, discussion, and conclusions derived from the study. Part V concludes the paper.

2. Research Objective

The aim of this work is to establish environmental control inside and outside airport terminals using Internet of Things (IoT) and sensor network devices. The proposed system (see Figure 1) facilitates the monitoring of various environmental variables by incorporating sensors to measure temperature and light intensity inside airport waiting areas, as well as soil moisture levels in the airport's garden.
The chosen sensors ensure high calibration accuracy for measuring the required information. Additionally, the selected IoT devices are of high technical quality, enabling real-time data retrieval from any of them.

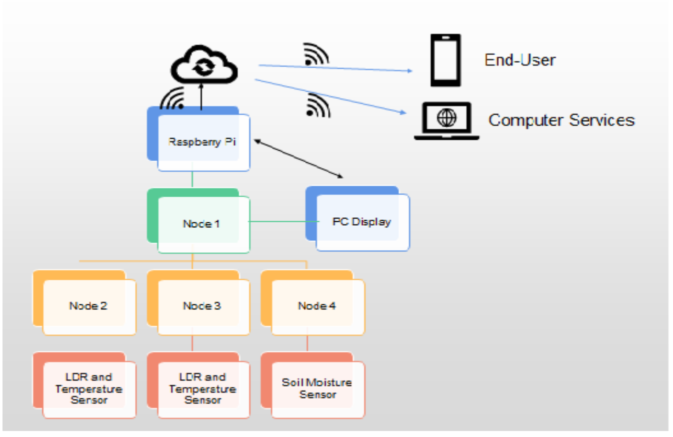


Fig 1. A block diagram illustrating the proposed system.

This environmental monitoring system enhances operational efficiency at airports while contributing to sustainability efforts, establishing it as a pivotal project in the realm of environmental management.

3.Literature Review and Related Work

Environmental monitoring systems have become integral to the design and management of airports, especially as the aviation industry seeks to enhance operational efficiency, safety, and sustainability. As smart airports evolve, IoT technologies play a key role in delivering real-time data and automation to ensure effective environmental monitoring for real-time monitoring. From air quality to energy management, the seamless integration of environmental sensors, communications technologies, and IoT systems provides valuable insights to airport operators, helping them optimize infrastructure and operations [1]. Enhancing environmental standards is largely influenced by IoT. It greatly advances sustainable living with its innovative and cutting-edge techniques for temperature, atmospheric pressure, humidity, soil moisture, light intensity, monitoring air quality and treating water, With the aid of various sensors [11]. For instance, other work proposes a system that collects some environmental information such as temperature, humidity, carbon monoxide and carbon dioxide from sensor nodes in airports by using

(IoT) [12]. Measurements for environmental monitoring in aeronautical meteorological applications are crucial for providing information such as dry temperature, wet temperature, atmospheric pressure, and dew point, due to their significant impact on air traffic management.[13]. With the integration of IoT technology, appliances, such as the air conditioning system, machines, and lighting systems, can be remotely controlled and monitored through a single interface, can monitor and control energy consumption, reducing waste and saving on energy bills. [9].

The IoT vision aims to provide direct merger of the physical world with Internet-connected computer-based systems to improve efficiency and cost as well as minimize human involvement [14]. Thus, the IoT ecosystem includes network users, computing systems and interconnected physical/virtual devices with sensing as well as actuating capabilities [15].

The backbone of the system is a wireless sensor networks (WSNs) that is establishing the actual interface between IoT devices and data captured through various types of sensors. The WSNs provide the connectivity of the data, captured by employing sensors and IoT devices, used to record, monitor and control various environmental conditions, such as temperature, light intensity, soil moisture, etc.

Within the realm of wireless communication for sensor networks, various standards exist, including Bluetooth, Wi-Fi, and Ultra-Wideband (UWB) [16]. However, for environmental monitoring applications requiring long-term, low-power operation, Zigbee has emerged as a particularly suitable choice. Its low power consumption allows sensor nodes to operate for extended periods, potentially years, before battery replacement, a crucial factor for large-scale deployments [17]. Furthermore, Zigbee simplifies communication protocols and reduces data rates, making it cost-effective for applications ranging from in-home monitoring to smart energy management [5]. Furthermore, it is widely used for applications such as home automation, smart energy management, and industrial control. Zigbee operates in the IEEE 802.15.4 standard and is specifically tailored to handle small amounts of data over short distances with minimal energy consumption [18]. Previous studies have successfully utilized Zigbee in environmental monitoring, demonstrating its effectiveness in agricultural settings for soil moisture and temperature monitoring, and in building automation for climate control [19]. Our study builds upon this established foundation by leveraging Zigbee for environmental monitoring at our system.

Gateway in the Internet of Things arena acts as an intermediary between numerous sensing networks and cloud platforms or data centers connected through the Internet. Raspberry Pi has become a top choice for IoT (Internet of Things) projects due to its compact size, low power consumption, affordability, and versatility [20]. It is widely used in embedded systems, smart devices, and DIY electronics thanks to its robust performance and accessible ecosystem, showcasing its capability in handling data from various sensor types and facilitating seamless data transfer to cloud platforms [21]. In our research, we utilize the Raspberry Pi to aggregate data from our sensor network and transmit it to the cloud for analysis and visualization.

To ensure reliable data transmission in wireless sensor networks, especially in potentially noisy environments, techniques like Time Division Multiple Access (TDMA) and Cyclic Redundancy Check (CRC) are essential. TDMA enhances parallelism by allowing multiple nodes to transmit data without interference by allocating specific time slots [22]. CRC coding, on the other hand, provides a robust mechanism for verifying data integrity by detecting transmission errors. Prior research on WSN communication protocols has extensively explored the benefits of TDMA and CRC in improving data reliability and minimizing data loss in various application scenarios [23]. Our system incorporates both TDMA and CRC to ensure the accuracy and completeness of the environmental data collected.

For data visualization and analysis in IoT applications, cloud platforms like Thing Speak offer readily available and user-friendly solutions. These platforms provide tools for data storage, real-time visualization, and basic data analysis, simplifying the process of extracting meaningful insights from sensor data. Thing Speak has been successfully used in various environmental monitoring projects, enabling researchers and practitioners to easily access, visualize, and interpret sensor data for diverse purposes, from environmental research to smart agriculture[24]. In this study ThingSpeak serves as the platform for visualizing and analyzing the data collected from our system, facilitating real-time insights and informed decision-making.

4.SENSOR DESIGN AND IMPLEMENTATION

a. Sensor and interface

In this study, we utilized three different types of sensors to monitor environmental conditions effectively [7][25].

i. Temperature Sensor

The temperature sensor, which is NTC (Negative Temperature Coefficient) type, is the product of the TTC05203(20kΩ). NTC is a circuit element whose electrical resistance decreases as the temperature of the environment or the contact surface increases. The operating range of our sensor is -30 to +125 degrees Celsius and the sensor is an analog sensor. For this sensor recommended applications are place appliances, automotive electronics, computers, switch mode power supplies and adapters.

In our study, we used that sensor and we choose maximum temperature is 45 ℃ and minimum temperature is 35 ℃. When we start the design of temperature sensor board, we used LM358 which is single supply dual operational amplifier. Since the amplitude of the antilog signal detected by the temperature sensor is low, we have to amplify it using an amplifier, so we used an op-amp.

Since the op-amp we use is in a dual structure, there are two input and two output pins. While designing the sensor circuits, we used an input and output of the op-amp (Input B and Output=PIN7) for the temperature sensor. We also used Input A and Output=PIN1 for LDR. We used the formulas below because the amplifier we employed for calculating the resistors in the sensor circuit design was of the non-inverting type. (Refer to Figure 2 and Figure 3 for illustration.)

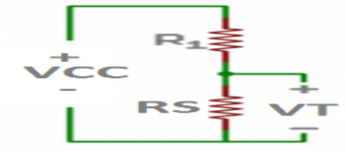


Fig 2 Voltage Divider Circuit

The formula of Voltage Divider is:

$$V_T = V_{in} \times (R_S/(R_1 + R_S))$$

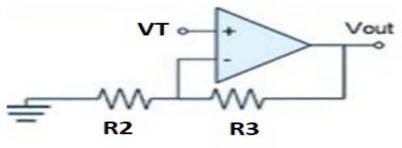


Fig 3: Non-Inverting Op-Amp

The formula of non-inverting op-amp is:

$$V_o = V_T \times \left(1 + (\frac{R_2}{R_1})\right)$$

$$V_o = V_{in} \times (R_S/(R_1 + R_S)) \times \left(1 + (\frac{R_2}{R_1})\right)$$

The Op-Amp supply voltage range is 3.0V - 32V. And our Vcc is 3.3 V.

Resistor Calculation for Temperature Sensor Board

$We\ assume\ that\ Gain = 2\ and\ Vout\ must\ be\ less\ than\ 3.3\ V;$

$$v_T = v_{CC} \times \frac{R_T}{R_1 + R_T}$$

$$\frac{3.3 \times R_T}{R_1 + R_T} < \frac{3.3}{2}$$

Case 1(Temperature=35°C which is $R_T = 15k\Omega$)

$$\frac{`15k\Omega}{R_1 + 15k\Omega} < \frac{1}{2}$$

$$R_1 > 15k\Omega$$

Case 2 (Temperature=45°C which is $R_T = 8k\Omega$)

$$\frac{8k\Omega}{8k_{\Omega}+R_1}<\frac{1}{2}$$

$$R_1 > 8k\Omega$$

Thus,

$$R_1 > 15k\Omega$$

And if Vout is minimum value;
PIC18F4620 ADC bit size is 10.

$$v_{m_in}=\frac{3.3}{2^{10}}=0.003v$$

$$v_T=\frac{v_{CC}\times R_T}{R_1+R_T}>v_{min}$$

$$\frac{R_T}{R_1+R_T}>\frac{0.003}{3.3}$$

$$1100R_T > R_1 + R_T$$

$R_1 < 1099R_T$Case 1(Temperature=35°C which is $R_T = 15k\Omega$)

$$R_1 < 1099x15k\Omega$$

Case 2 (Temperature=45°C which is $R_T = 8k\Omega$)

$$R_1 < 1099x8k\Omega$$

Thus

$$R_1 < 1099x8k\Omega$$

So the range of R1 is;

$$15 < R_1 < 1099x8k\Omega$$

We choose R1=82 kΩ

$$Gain=\frac{v_0}{v_{in}}=2$$

$$v_0=v_{in}\times\left(\frac{1+R_2}{R_3}\right)$$

$$R_2 = R_3$$

We choose R2=R3=10 kΩ

ii. **Light Dependent Resistor (LDR) Sensor**

LDR is a light sensitive circuit element. It has a working principle that is inversely proportional to the intensity of the light falling on it. According to this working principle, as the intensity of the light falling on it increases, the value of its resistance decreases. As the light intensity decreases, its resistance value increases.

Our LDR has the CdS photoconductive cell which is on a TO-18 ceramic and the photocell surface is plastic encapsulated for moisture resistance. Its opto-electrical for light resistance parameter typical range is 10 kΩ for 2ftc- 700Ω for 100 ftc. It is used for industrial applications.

For the LDR circuit board we used the same Op-amp we used for the temperature sensor. Finding which resistor is the suitable for our circuit board we had a calculation. That calculation is:

$We\ assume\ that\ Gain = 2\ and\ Vout\ must\ be\ less\ than\ 3.3\ V;$

$$v_T=v_{CC}\times\frac{R_L}{R_1+R_L}$$

$$\frac{3.3\times R_L}{R_1+R_L}<\frac{3.3}{2}$$

Case 1(Light is max $R_L = 0.7k\Omega$)

$$\frac{0.7k\Omega}{R_1+k\Omega}<\frac{1}{2}$$

$$R_1 > 0.7k\Omega$$

Case 2 (Light is min $R_L = 10k\Omega$)

$$\frac{10k\Omega}{10k_{\Omega}+R_1}<\frac{1}{2}$$

$$R_1 > 10k\Omega$$

Thus,

$$R_1 > 10k\Omega$$

And if Vout is minimum value; PIC18F4620 ADC bit size is 10.

$$v_{m_in}=\frac{3.3}{2^{10}}=0.003v$$

$$v_L=\frac{v_{CC}\times R_L}{R_1+R_L}>v_{min}$$

$$\frac{R_L}{R_1+R_L}>\frac{0.003}{3.3}$$

$$1100R_L > R_1 + R_L$$

$$R_1 < 1099R_L$$

Case 1(Light is max $R_L = 0.7k\Omega$)

$$R_1 < 1099x0.7k\Omega$$

Case 2 (Light is min $R_L = 10k\Omega$)

$$R_1 < 1099x10k\Omega$$

Thus

$$R_1 < 1099x10k\Omega$$

So, the range of R1 is;

$$10 < R_1 < 1099x10k\Omega$$

We choose R1=18 kΩ

$$Gain=\frac{v_0}{v_{in}}=2$$

$$v_0=v_{in}\times\left(\frac{1+R_2}{R_3}\right)$$

$$R_2 = R_3$$

We choose R2=R3=10 kΩ

iii.**Soil Moisture Sensor**

The soil moisture sensor is designed to detect moisture levels in the soil and determine the presence of water around the sensor. This sensor operates based on the measurement of soil resistivity [26].

As illustrated in the schematic diagram (Figure 6), the sensor consists of two pads, two resistors, and one NPN-type transistor. It can detect whether the soil is dry, wet, or humid, thanks to the function of the pads. The transistor acts as a switch in this setup.

- The base pin of the transistor is connected to PAD2, while the collector pin is connected to Vcc.
- PAD1 is also connected to Vcc through a 100 Ω resistor.

The output specifications for this sensor are as follows:

- **Dry Soil:** 0-300 output value
- **Humid Soil:** 300-700 output value
- **Water:** 700-950 output value

This functionality makes the soil moisture sensor an effective tool for moisture detection and monitoring in agricultural and environmental applications [27].

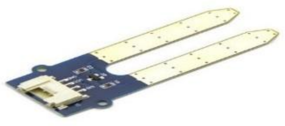


Fig 4. Soil Moisture Sensor

b. Calibration and Implementation

Calibration is a critical process involving the verification and adjustment of measuring instruments to ensure their accuracy. It consists of a series of operations that establish the relationship between the values indicated by the measuring system and the known corresponding values. In our study, we performed calibration on the Light Dependent Resistor (LDR) and temperature sensors.

In our study we used calibration for LDR and temperature sensors. For LDR sensor, first we measured light of room by using LUX measurement device, and at the same time we measured the LDR sensor board output voltage. After we covered the LDR sensor and LUX measurement by same box, and we measured LDR sensor board output voltage. We noted what we get. In figure 5 illustrates the graph of our results when we measured LDR sensor calibration. Figure 6 shows a graph of the results when measuring the calibration of the temperature sensor. Figure 7 is a graphical representation of the results of measuring the calibration of the soil moisture sensor.

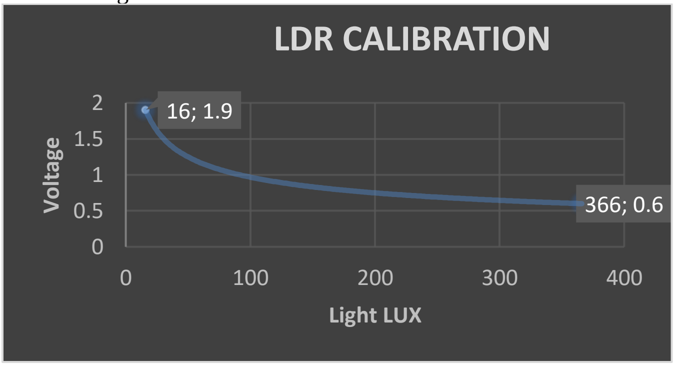


Fig 5. Calibration result of LDR sensor board (Node 1)

The result is exponential as in the graph.
So, our calculation formula is:
Light (Lux)= $a \times e^{(-v)} + b$ which is V refers voltage.
When we put the values in the formula, we got a=950.66 and b= -100.
In our code:

```
>> Light=950.66*pow(2.72,-(float)ADCresult1*3.1/1023)-100;
```

"pow" function is for using exponential numbers. " e "equals approximately 2.72.
When we measured the Vcc in real, we got 3.1V. PIC18F4620 has 10 bits (0-1023) ADC. Each channel has 3.1/ (2^10-1)) = 0.003 V.
The number is fractional that is why we choose "float" function.

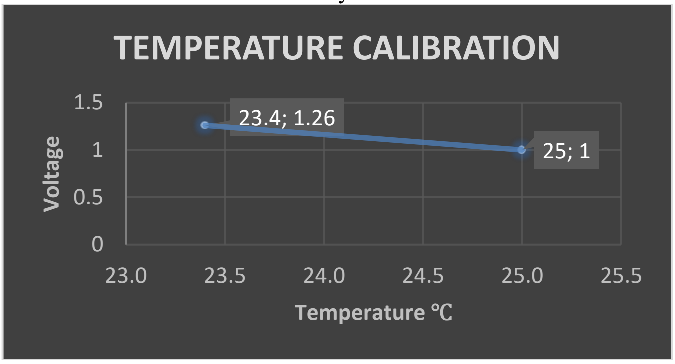


Fig 6. Calibration of temperature sensor result

The result is linear as in the graph. So, our calculation formula is: Temperature (°C) =$a \times V + b$ which is V refers voltage. When we put the values in the formula, we got a=-6 and b= 31.
In our code: >> Temperature=31-6*(float)ADCresult2*3.1/1023;
When we measured the Vcc in real, we got 3.1 V. PIC18F4620 has 10 bits (0-1023) ADC. Each channel has 3.1/ (2^10-1)) = 0.003 V. The number is fractional that is why we choose "float" function.

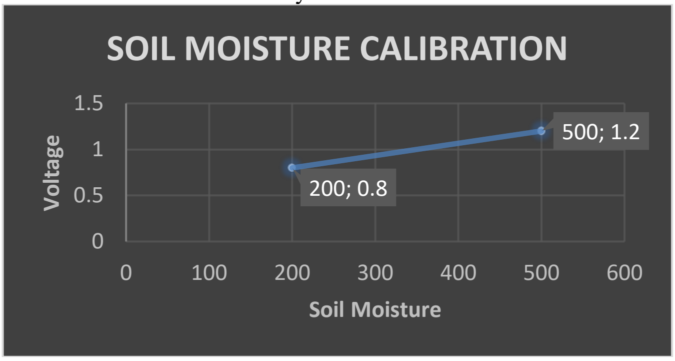


Fig 7. Calibration of soil moisture sensor

The result is linear as in the graph. So, our calculation formula is:
Soil Moisture =$a \times V + b$ which is V refers voltage.
As we know our soil moisture has a range for different position. And when we measured the humid soil, we got 1.2 V output voltage and looking the sensor datasheet, we choose that soil moisture is 500. After measuring the dry soil, the output voltage was 0.8 V and chosen data was 200. When we put the values of the formula, we got a=678 and b=-542.
In our code: >> Soil Moisture=678*SoilMoisture-542;
When we measured the Vcc in real, we got 3.1 V. PIC18F4620 has 10 bits (0-1023) ADC. Each channel has 3.1/ (2^10-1)) = 0.003 V. The number is fractional that is why we choose "float" function.

c. TDMA Communication

All stations in the network reach the satellite using the same frequency channel in different time slot. For instance, in this project, we have 4 nodes and for LDR 3 nodes are using Channel 3; for temperature 3 nodes are using Channel 4. We can see in the code:

```
>>OpenADC (ADC_FOSC_32 & ADC_RIGHT_JUST & ADC_8_TAD,
        ADC_CH3 & ADC_INT_OFF & ADC_REF_VDD_VSS,0);
   Delay10TCYx (5);
```

We connected LDR sensor board output to PIC18F4620 ADC Channel 3.

```
>> OpenADC (ADC_FOSC_32 & ADC_RIGHT_JUST & ADC_8_TAD,
        ADC_CH4 & ADC_INT_OFF & ADC_REF_VDD_VSS,0);
   Delay10TCYx (5);
```

We connected temperature sensor board output to PIC18F4620 ADC Channel 4. For our study, connection between outputs of sensor boards and MCU ADC channel is in the below picture.

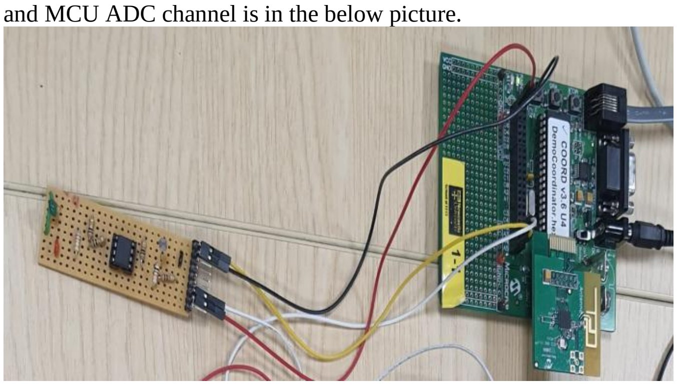


Fig 8. Connection between sensor board and Node 1

We linked the yellow wire which is temperature sensor output to Channel 4 and white wire which is LDR sensor output to Channel 3. Since we use the same sensor boards and same Channels for 3 nodes in our study, we used TDMA technique to prevent possible collisions. TDMA Working Principle is the control station (Master) divides the TDMA Frame into Time Slots according to the stations and periodically broadcasts it to the network. In this way, all remote stations learn their tTime Slots and can transmit on the common frequency in their Time Slot range. The sent packet is listened by all other stations in the network. However; only the receiving station, which is addressed as a destination in the sending packet, transmits this data to the user ports, while the others block and delete it. All stations except the transmitting station cannot use this frequency in this time slot. Figure 9 is shown that principle of working TDMA [23].

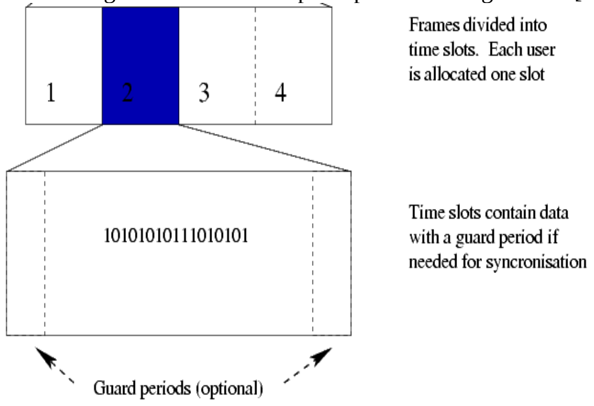

1
2
3
4
Frames divided into time slots. Each user is allocated one slot
10101010111010101
Time slots contain data with a guard period if needed for syncronisation
Guard periods (optional)

Fig 9: TDMA frame structure showing a data stream divided into frames and those frames divided into time slots[1*]

We used separate code for each node. Node 1, which is the master node, became both a transmitter and a receiver. Node 2, 3 and 4 are slave nodes, so they only act as receivers. As it can be understood from our code, we put 25 TCY delays in order for the data to reach the channel sequentially for node 2, 3 and 4.

```
For Node 2 Receiver Part:       >>Delay10KTCYx(25);
Fore Node 3 Receiver Part:     >>Delay10KTCYx(50);
Fore Node 4 Receiver Part:   >>Delay10KTCYx(75);
```

When the sensors data come from the ADC Channels, Node 1 decides in which order the data from all nodes will be transmitted to the channel, and when the 25 second time slot we have determined is completed, node1 will transmit. Thus, possible collisions are prevented and we can see the data in Putty in order.

d. Interrupt

We used the PIC18F4620 microprocessor in our study and examined

[1*] Wikipedia contributors. (2021b, November 10). Time-division multiple access. Wikipedia. https://en.wikipedia.org/wiki/Time-division_multiple_access

its datasheet. There are ten registers which are used to control interrupt operation.[2†]
These registers are:
• RCON
• INTCON
• INTCON2
• INTCON3
• PIR1, PIR2
• PIE1, PIE2
• IPR1, IPR2
And we used INTCON (Interrupt Control Register) register in our code. The INTCON registers are readable and writable registers, which contain various enable, priority and flag bits. We have to assign register's logic gates. If we want to INTCON register is active we have to make a decision looking Interrupt logic diagram. (Figure 10)

FIGURE 10-1: PIC18 INTERRUPT LOGIC

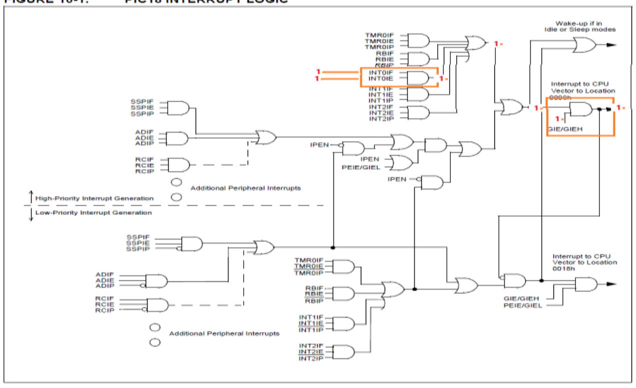

Wake-up if in Idle or Sleep modes
Interrupt to CPU Vector to Location 0008h
GIE/GIEH
IPEN
PEIE/GIEL
High-Priority Interrupt Generation
Low-Priority Interrupt Generation
Additional Peripheral Interrupts
Interrupt to CPU Vector to Location 0018h
GIE/GIEH
PEIE/GIEL

Fig 10. Interrupt Logic Diagram that determines whether the register bits are 0 or 1

When we look at the Logic Diagram, for interrupt activation we have to choose INT0 External Interrupt Enable bit (INT0IE) and INT0 External Interrupt Flag bit (INT0IF). Because these bits are connecting with the AND gate. Their output should be 1 and that's why they have to be 1.
The same scenario applies to the GIEH bit.
In our code: >>INTCONbits.INT0IF = 0;
>>INTCONbits.INT0IE=1
>>INTCONbits.GIEH = 1;

e. TIMER

Timer modules are used to measure time or create accurate time delay. Also, the microcontroller can generate/measure the required time delays by running loops on its own, but the timer module frees the microprocessor from this redundant and repetitive task and allows it to allocate maximum processing time for other tasks.

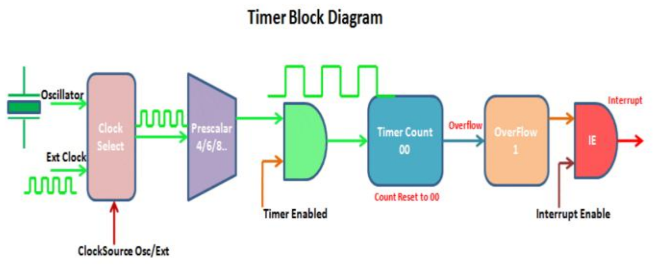


Fig 11. Timer Block Diagram[3‡]

The timer functions as a basic binary counter that can be configured to count clock pulses, either from internal or external sources. When it reaches its maximum value, it resets the Overflow flag to zero and triggers an interrupt if it is enabled.
In our study, we utilized TIMER0, which includes the following features:

- Software-selectable operation as either a timer or counter in both 8-bit and 16-bit modes.
- Readable and writable registers for easy access.
- Dedicated 8-bit prescaler, programmable via software.
- Selectable clock source (internal or external).
- Edge select for external clock[4§]

In our code:
OpenTimer0(TIMER_INT_ON & T0_16BIT & T0_SOURCE_INT & T0_PS_1_16);
//setup timer 0 with prescaler x16
WriteTimer0(3036);
unsigned char FirstTFlag = 1;
Calculation of Count Time:
Zigbee board uses 4 MHz quartz oscillator fosc=4 MHz; the clock period/cycle is then

$$Tosc = \frac{1}{fosc} = \frac{1}{4x10\text{^}6} = 0.25\mu s$$

One instruction cycle is the minimum time to perform any operation:

$$TCY = 4\,Tosc = 1\ \mu s$$

PIC 18F4620 has two different TIMER0 Mode. One of them is 8-bit and the other is 16-bit. If we want to use just Timer mode, we should use 8-bit mode but if we want to use both timer and counter mode, we should use 16-bit mode. In this study we used 16-bit mode. And we choose prescaler 1:16 using Timer0 Prescaler Select bits, T0PS2:T0PS0=011.

$$CountTime = (2^{16} - CountValue)PrescalerValue\ x\ Tosc$$
$$CountTime = (65536 - 3036)x16x10^{-6}$$

$CountTime$ =1 second

f. RSSI

The Received Signal Strength Indicator (RSSI) quantifies the strength of radio signals used by wireless network devices to evaluate the status of a transmission channel (whether it is idle or busy). Many wireless standards, including the IEEE 802.11 and IEEE 802.15.4 families, provide RSSI estimates and utilize this metric.
During wireless communication, various types of noise—such as electromagnetic waves, detectors, and cordless phones—can interfere with the signal. In this context, the effective operation of RSSI is crucial. The presence of interference can lead to inaccurate RSSI readings, which can have serious implications for overall network performance.
We used in our study IEE 802.15.4 (Zigbee) Protocol.
Features of IEEE 802.15.4 (Zigbee):

- 2.4GHz spread spectrum radio standard
- 250kbps peak data rate
- ERF 0dBm (1mW)
- Max range ~100m, (40m indoors)
- 16 frequency channels
- RSSI outputs

For using that protocol, we have MRF24J40 transceiver in the antenna module.
When we initialize the code from the MRF24 header file to the main.c file, we saw the RSSI output data on the Putty screen.

g.Raspberry PI

The Raspberry Pi Zero is a single-board minicomputer primarily used for designing IoT study based on embedded systems. Its low cost, small size, and open-source design make it suitable for applications such as weather monitoring, motion sensor cameras, and small computers for power adapters [20].
In this study, we used the Raspberry Pi Zero module, which features a mini connector with a 40-pin GPIO, making it compact, robust, and flexible. It is a credit card-sized computer with multiple I/O pins for various common computer operations like programming, surfing the internet, and writing documents. The module contains a 1GHz single-core CPU that performs basic arithmetic, logical, and I/O operations by executing instructions from computer programs. The clock frequency of this computer is 400MHz, necessary for synchronizing internal functions. There are no Wi-Fi or Ethernet connections

[2†] PIC18F Datasheet
[3‡] D. (2021, July 22). *PIC18F4520 Timer/ Zamanlayıcı Modülleri TIMER0 TIMER1*. https://devreyakan.com/pic18f4520-timer-zamanlayici-modulleri/
[4,] PIC18F Datasheet

available on the board. The module includes 512MB of RAM for storing running programs, which is volatile memory, meaning data is lost when powered down.

The board is equipped with a micro-USB OTG port for reading data without a computer and is powered via a micro-USB power supply. A mini-HDMI connector is integrated for sending audio and video signals, and a CSI camera connector is provided for connecting an external camera This module supports various communication protocols like SPI, I2C, and UART for interfacing with external devices. It uses a micro-SD card for initial storage and can be expanded with USB-connected peripherals. With an SD card, extended storage is easily accessible. The Raspberry Pi Zero has a single USB 2.0 port for connecting external peripherals such as mice, keyboards, and webcams. Additionally, it can be connected to a computer via USB, allowing users to operate it like a microcomputer with features similar to a regular desktop computer.

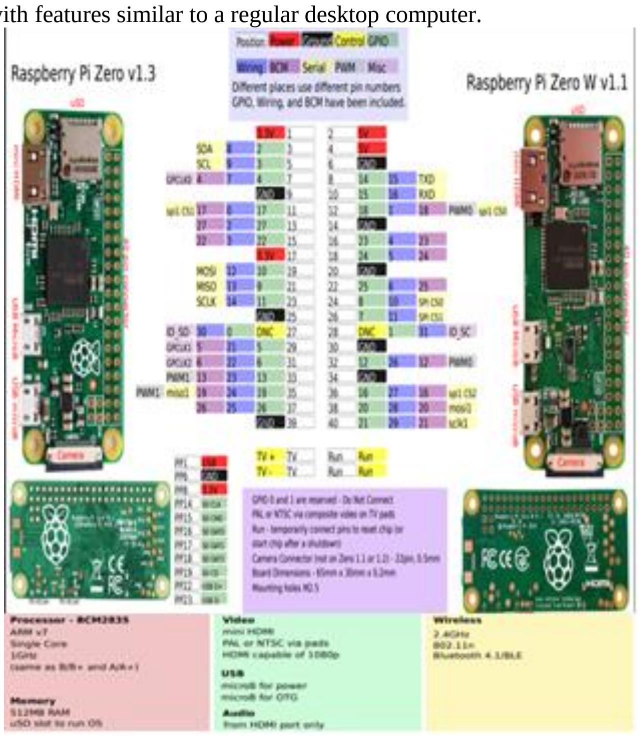
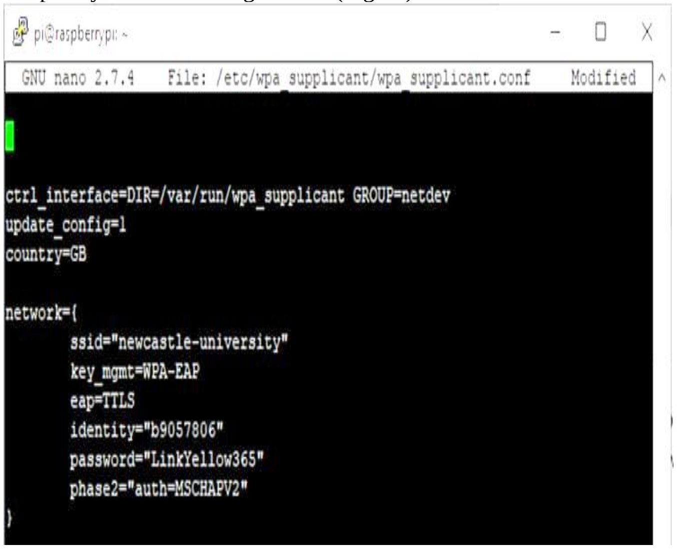


Fig 12. Raspberry Pi GPIO Interfaces

In our study, we used Putty to connect to your Raspberry PI. Therefore, you need to make sure that the internet used by the Raspberry PI is the same as the internet on our computer. In this case, we used the College network. First, we need to create a script with the Raspberry PI WIFI configuration. (Fig 13)

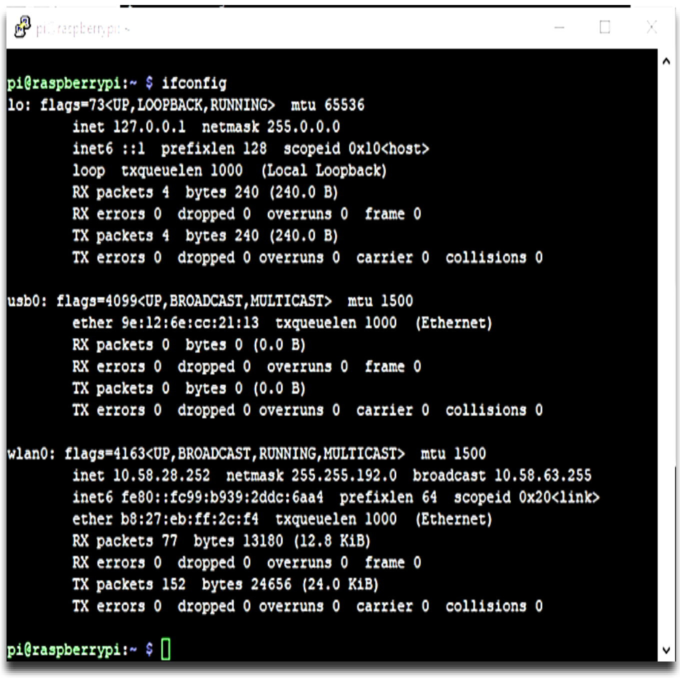


```
pi@raspberrypi: ~

GNU nano 2.7.4    File: /etc/wpa_supplicant/wpa_supplicant.conf    Modified


ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
        ssid="newcastle-university"
        key_mgmt=WPA-EAP
        eap=TTLS
        identity="b9057806"
        password="LinkYellow365"
        phase2="auth=MSCHAPV2"
}
```

Fig 13. Wi-Fi Connection Using Raspberry Pi

When this script finishes, just enter the command `sudo reboot`. Then wait for it to automatically connect to the College network.

After that we got an IP addresss and connect with the computer remotely using SSH (Secure Shell) Protocol. (Figure 14)



```
pi@raspberrypi: ~

pi@raspberrypi:~ $ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

usb0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 9e:12:6e:cc:21:13  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.58.28.252  netmask 255.255.192.0  broadcast 10.58.63.255
        inet6 fe80::fc99:b939:2ddc:6aa4  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:ff:2c:f4  txqueuelen 1000  (Ethernet)
        RX packets 77  bytes 13180 (12.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 152  bytes 24656 (24.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $ 
```

Fig 14. IP address Configuration

h. UART

In this study, we used UART (Universal Asynchronous Receiver Transmitter) communication, which is one of the serial communication types, to facilitate data transfer from the PIC (Peripheral Interface Controller) to the Raspberry Pi.

UART is a communication protocol that enables communication between computers and microcontrollers or between microcontrollers and peripherals. Since UART operates asynchronously, it does not require a clock signal for synchronization. USART (Universal Synchronous Asynchronous Receiver Transmitter) can work both synchronously and asynchronously. It is a more advanced protocol than UART. The communications work in the same logical way, but USART can also handle synchronous communications. When you look at the datasheet of a newly released microprocessor, we usually see these units as USART units because USART is designed as a unit that also includes UART. We see the USART communication unit in the datasheet of the microprocessor we use.

We have configured it in C code to use UART communication.

Open USART (USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &USART_EIGHT_BIT & USART_CONT_RX & USART_BRGH_HIGH,25); //setup USART @ 9600 Baud

We are using 16-bit Asynchronous Mode and BRGH=0, since our formula is;

$$Baud\ Rate = \frac{fosc}{16x(n+1)}$$

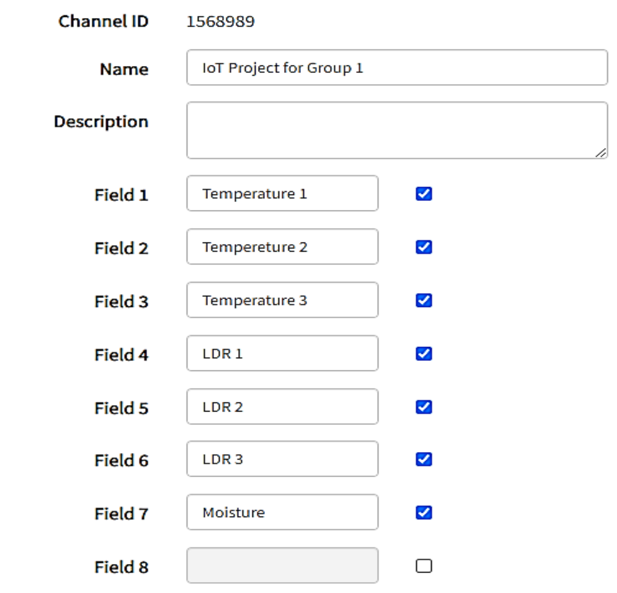
When we look at our code, we choose n=25.And PIC oscillator frequency is 4 MHz so Baud Rate=9600.

When we finish the configuration part of USART,

IOT connection

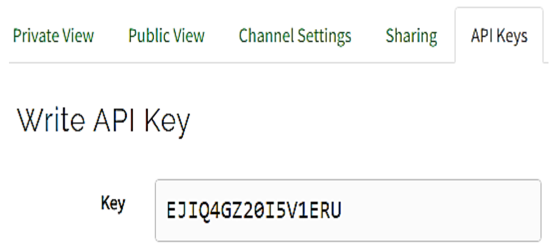
In this study, we use Thing Speak for IoT connection. Thing Speak is an open-source program composed in Ruby which permits clients to communicate with web empowered devices. And firstly, we have to create an account to enter Thing Speak

platform. After we create 7 channels for seeing our 7 output results which are coming from sensor boards. And we copied our channels' API keys and put into our Phyton code which is below:



Channel ID	1568989
Name	IoT Project for Group 1
Description	
Field 1	Temperature 1 ✓
Field 2	Tempereture 2 ✓
Field 3	Temperature 3 ✓
Field 4	LDR 1 ✓
Field 5	LDR 2 ✓
Field 6	LDR 3 ✓
Field 7	Moisture ✓
Field 8	☐

Fig 15. Channels Definition on ThingSpeak



Write API Key

Key EJIQ4GZ20I5V1ERU

Fig16. API Key

When we run the code, our ThingSpeak channels stored data. We can see data changes every 15 seconds. Because we used free version of ThingSpeak.

5.Result Discussion and Conclusion

a. Hardware Design

In this study, we used Proteus as the hardware design program. First of all, we made the PIN connections in accordance with the datasheets of the circuit elements. Proteus consists of two parts, ISIS/ARES, and ISIS is used for schematic circuit design and ARES for PCB circuit design. Temperature and LDR sensor circuit are shown below.

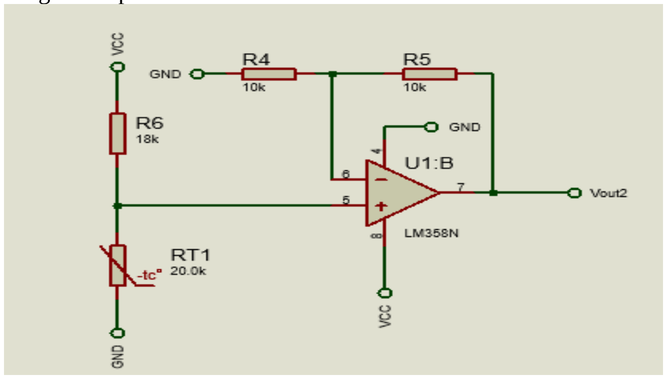


Fig 17. Schematic Diagram of Temperature Sensor

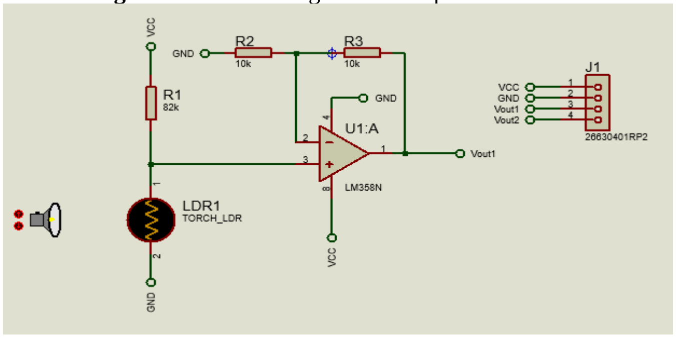
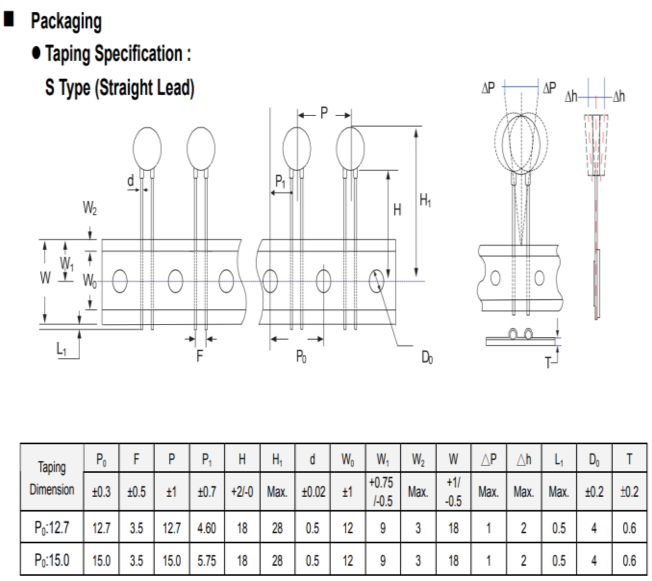


Fig 18. Schematic Diagram of LDR Sensor

Circuit elements are generally defined in Proteus, and you need to determine the schematic packages and PCB packages of some circuit elements while drawing the circuit. You can learn this by looking at the datasheet. For example, in the datasheet of the temperature sensor, the pin leg lengths, length and thickness that should be used for the sensor are given. While we were making the schematic drawing of the soil moisture circuit, we created it ourselves in Proteus due to the lack of PAD legs. We added it to the library.



Taping Dimension	P_0	F	P	P_1	H	H_1	d	W_0	W_1	W_2	W	△P	△h	L_1	D_0	T
	±0.3	±0.5	±1	±0.7	+2/-0	Max.	±0.02	±1	+0.75 /-0.5	Max.	+1/ -0.5	Max.	Max.	Max.	±0.2	±0.2
P_0:12.7	12.7	3.5	12.7	4.60	18	28	0.5	12	9	3	18	1	2	0.5	4	0.6
P_0:15.0	15.0	3.5	15.0	5.75	18	28	0.5	12	9	3	18	1	2	0.5	4	0.6

Fig 19. Package size of temperature sensor

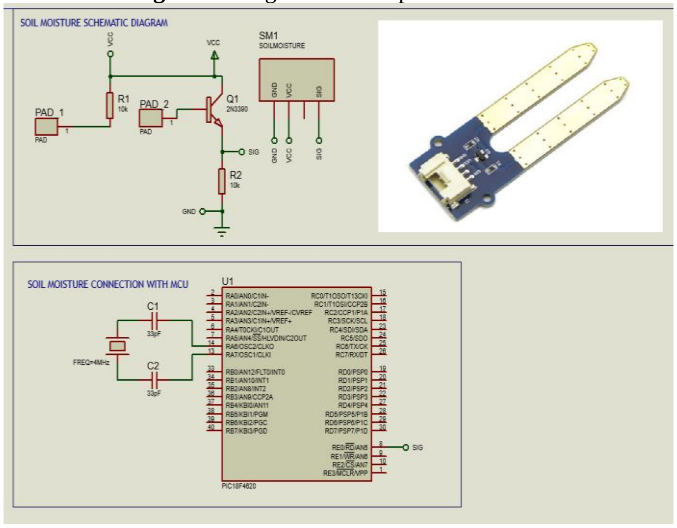


Fig 20. Schematic Diagram of Soil Moisture

After determining the packages for the PCB drawing and making the pin connections, you can choose the layers of the board from the Layer option. It will be very useful to use in multi-layered projects. But in our study, it was enough to use both sides of the board. We set the top layer (red) as Vcc and the bottom layer (blue) as ground. While completing the pin connections, we chose the wire thickness as 40th.

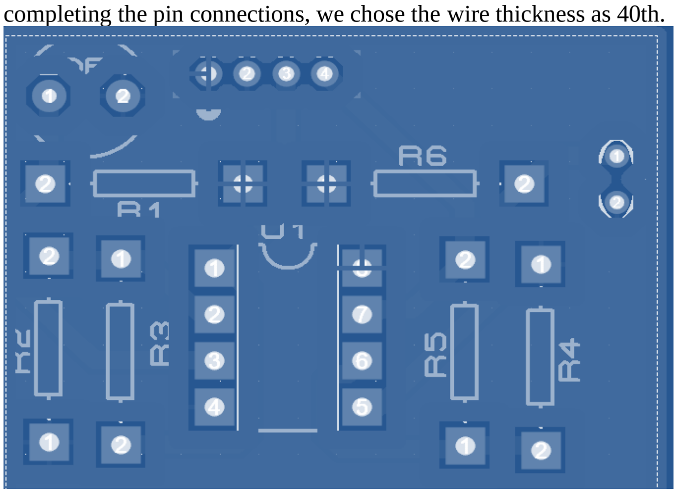


Fig 21. PCB Design when top layer(red) is Vcc

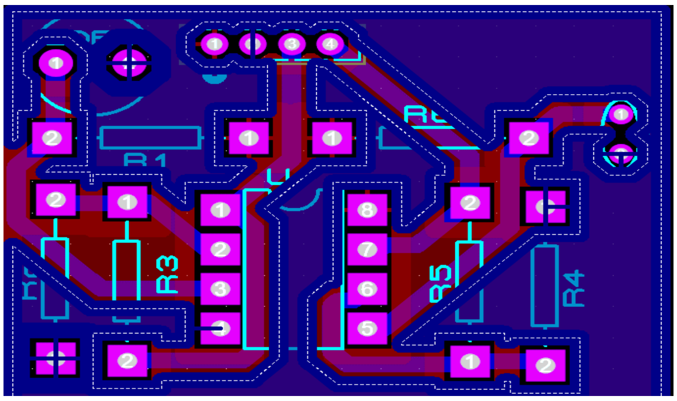


Fig 22. PCB Design when bottom layer (blue) is Ground

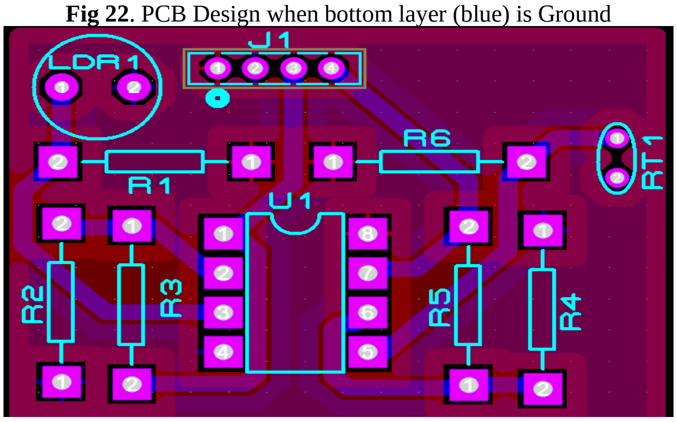


Fig 23. PCB Design of LDR and Temperature Sensors Boards

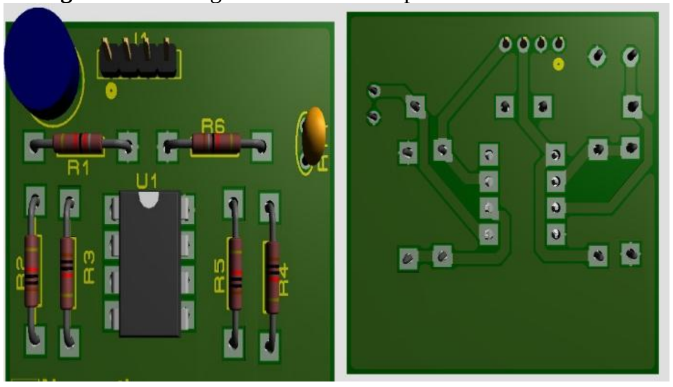


Figure 24. 3D view of circuit

b. Components Used for Hardware

Four PICDEM Z DEMO KIT which contain (Microchip PICDEM Z, PIC18F4620, MRF24J40 transceiver, RS-232 serial port, and Digital/analog I/O pins on J6)

Microchip PICDEM Z was connected by PIC18F4620 as this PIC18F4620 type was selected to be our microcontroller.

The MRF24J40 transceiver was connected to the Microchip PICDEM Z as well. Basically, it is a wireless transceiver which features an integrated PCB antenna, matching circuitry, and supports ZigBee protocol and MiWi™. It is an ideal solution for wireless sensor networks, home automation, and is compatible with hundreds of PIC microcontrollers via a 4-wire SPI interface. In this study the MRF24J40 is responsible for transmitting and receiving data from to the four nodes.

Zigbee protocol was used. Zigbee is a low-cost, low-power wireless protocol that was developed as a global standard to meet the requirements of IoT network [18]. The Zigbee protocol allows devices (In this study they are the 4 Nodes) to communicate in a variety of network topologies, and its battery life can last several years.

Serial communication is the process of sending data serially over a computer network, which means data is sent bit by bit. RS-232 serial port is a serial communication protocol used to allow serial data exchange between a computer and its peripheral devices.

The MPLAB ICD 3 In-Circuit Debugger is a powerful and complex debugger system used to develop hardware and software for PIC microcontrollers and digital signal controllers. The MPLAB ICD 3 In-Circuit Debugger is supporting the fastest processors at their maximum speeds, allowing embedded engineers or users to debug applications on their own hardware in real-time.

c. Final Hardware Connection

Four PCBs were used; each node is connected to a sensors board. Node 1,2 and 3 are connected to the built circuit of the light and Temperature sensors. Channel 3 (pin 6 RA3) for each of the mentioned nodes was connected to the light sensor outputs, channel 4 (pin 2 RA5) for each of the mentioned nodes was connected to the output of the temperature sensor.

Node 4 is connected to the moisture sensor, and channel 5 of the node 4 (pin 10 RE0) is connected to the signal output of the moisture sensor. The moisture sensor was plugged to dry and watered soil to measure the moisture of the soil. The figure below displays the whole connections of the sensors to the nodes.

Node 1 is set in the code to be the master node, it is transmitting and receiving data. Node 2,3 and 4 are set in the code to be the slave nodes, which are only transmitting data to the master node (1).

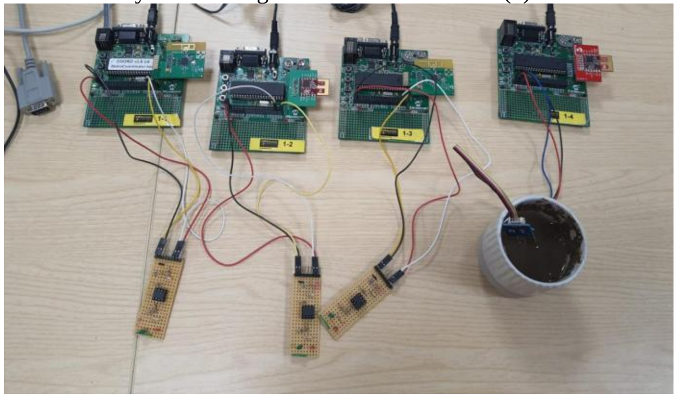
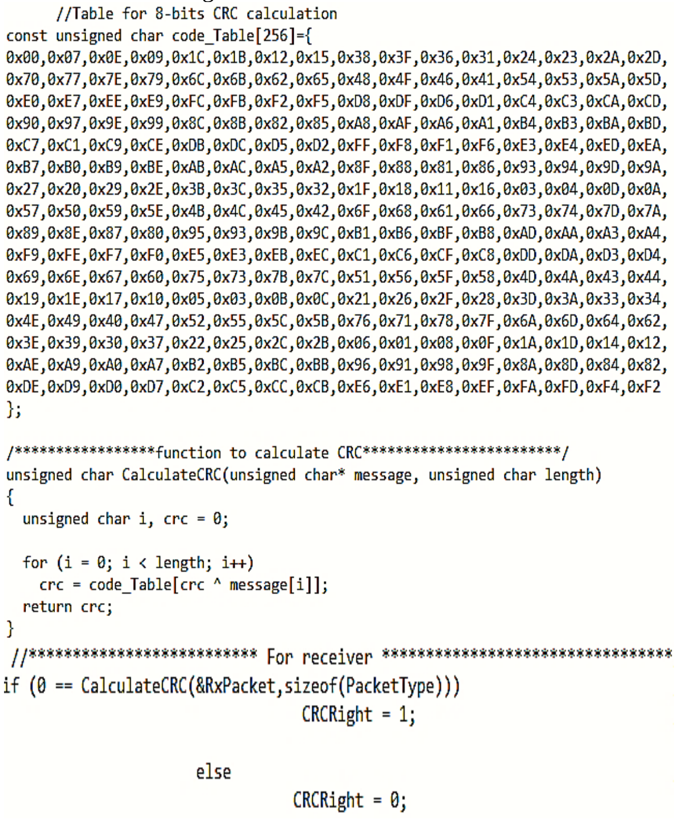


Fig 25. Final Hardware Connection

d. CRC

Cyclic Redundancy check is a form of checksum is used to detect error in digital data. Data blocks entering these systems are assigned a short check value based on the remainder of a polynomial division of their contents. When retrieving data, checking is repeated if the check values do not match, and CRCs can be used to correct errors [24].

A table for 8-bits CRC calculation was used in our code for detecting errors. The reason of using the CRC table calculation is to check any value transmit and receiving a correct reading. We used table for 8-bits as our bits are eight.



```
      //Table for 8-bits CRC calculation
const unsigned char code_Table[256]={
0x00,0x07,0x0E,0x09,0x1C,0x1B,0x12,0x15,0x38,0x3F,0x36,0x31,0x24,0x23,0x2A,0x2D,
0x70,0x77,0x7E,0x79,0x6C,0x6B,0x62,0x65,0x48,0x4F,0x46,0x41,0x54,0x53,0x5A,0x5D,
0xE0,0xE7,0xEE,0xE9,0xFC,0xFB,0xF2,0xF5,0xD8,0xDF,0xD6,0xD1,0xC4,0xC3,0xCA,0xCD,
0x90,0x97,0x9E,0x99,0x8C,0x8B,0x82,0x85,0xA8,0xAF,0xA6,0xA1,0xB4,0xB3,0xBA,0xBD,
0xC7,0xC1,0xC9,0xCE,0xDB,0xDC,0xD5,0xD2,0xFF,0xF8,0xF1,0xF6,0xE3,0xE4,0xED,0xEA,
0xB7,0xB0,0xB9,0xBE,0xAB,0xAC,0xA5,0xA2,0x8F,0x88,0x81,0x86,0x93,0x94,0x9D,0x9A,
0x27,0x20,0x29,0x2E,0x3B,0x3C,0x35,0x32,0x1F,0x18,0x11,0x16,0x03,0x04,0x0D,0x0A,
0x57,0x50,0x59,0x5E,0x4B,0x4C,0x45,0x42,0x6F,0x68,0x61,0x66,0x73,0x74,0x7D,0x7A,
0x89,0x8E,0x87,0x80,0x95,0x93,0x9B,0x9C,0xB1,0xB6,0xBF,0xB8,0xAD,0xAA,0xA3,0xA4,
0xF9,0xFE,0xF7,0xF0,0xE5,0xE3,0xEB,0xEC,0xC1,0xC6,0xCF,0xC8,0xDD,0xDA,0xD3,0xD4,
0x69,0x6E,0x67,0x60,0x75,0x73,0x7B,0x7C,0x51,0x56,0x5F,0x58,0x4D,0x4A,0x43,0x44,
0x19,0x1E,0x17,0x10,0x05,0x03,0x0B,0x0C,0x21,0x26,0x2F,0x28,0x3D,0x3A,0x33,0x34,
0x4E,0x49,0x40,0x47,0x52,0x55,0x5C,0x5B,0x76,0x71,0x78,0x7F,0x6A,0x6D,0x64,0x62,
0x3E,0x39,0x30,0x37,0x22,0x25,0x2C,0x2B,0x06,0x01,0x08,0x0F,0x1A,0x1D,0x14,0x12,
0xAE,0xA9,0xA0,0xA7,0xB2,0xB5,0xBC,0xBB,0x96,0x91,0x98,0x9F,0x8A,0x8D,0x84,0x82,
0xDE,0xD9,0xD0,0xD7,0xC2,0xC5,0xCC,0xCB,0xE6,0xE1,0xE8,0xEF,0xFA,0xFD,0xF4,0xF2
};

/*****************function to calculate CRC************************/
unsigned char CalculateCRC(unsigned char* message, unsigned char length)
{
  unsigned char i, crc = 0;

  for (i = 0; i < length; i++)
    crc = code_Table[crc ^ message[i]];
  return crc;
}
 //************************ For receiver ********************************
if (0 == CalculateCRC(&RxPacket,sizeof(PacketType)))
                                        CRCRight = 1;


                         else
                                        CRCRight = 0;
```

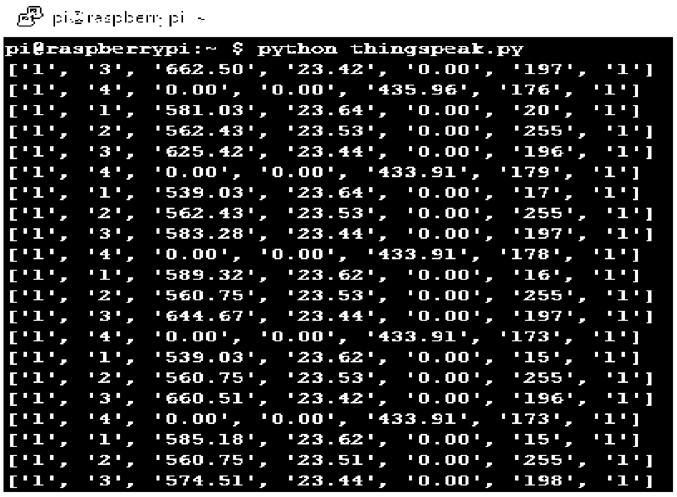
Figure 26: CRC Control Code in MPLAB

If calculate CRC to 0, that means the received data is right, then setting the CRCRight flag as 1, which means CRC is correct, as it is displaying in the figure below. In conclusion whenever the value of CRC is 1 in Putty console, that means we are receiving a right value.

e. TDMA

With the TDMA function, the data from the four nodes are displayed one after another. In Figure below, the first digit in each sequence indicates that this data is from Group1. The second number shows the board from which this data came from. If all arrays are functioning properly, the output will be the sequence shown. Otherwise, you will lose the sequence of failed arrays. For example, if Node2 is defective, only output data 1, 3, 4, 1, 3, 4, etc. will be displayed.

The 3 arrays reflected after the group number and node number reflect the data from the LDR, temperature and soil moisture sensors. Since the first 3 nodes are only connected to the LDR and temperature sensor, the output data of the LDR and temperature sensor are seen in Node 1,2 and 3. In Node 4 only soil moisture sensor is connected. And since it depends on the soil to which water has been added, its data seems to be high.

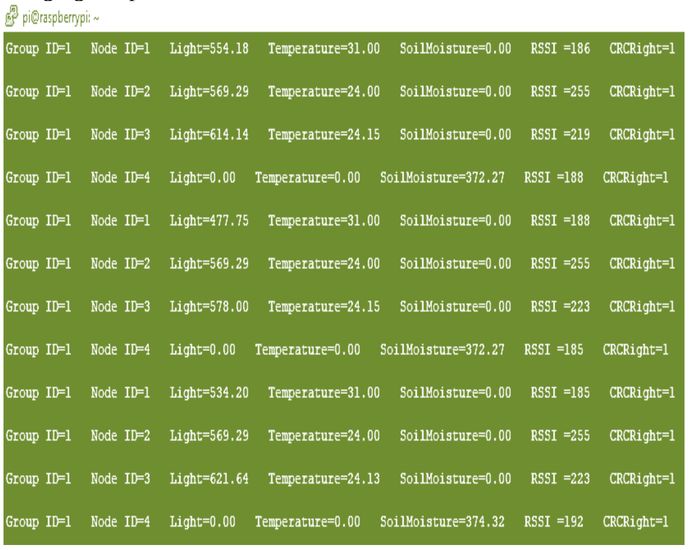


```
pi@raspberrypi:~ $ python thingspeak.py
['1', '3', '662.50', '23.42', '0.00', '197', '1']
['1', '4', '0.00', '0.00', '435.96', '176', '1']
['1', '1', '581.03', '23.64', '0.00', '20', '1']
['1', '2', '562.43', '23.53', '0.00', '255', '1']
['1', '3', '625.42', '23.44', '0.00', '196', '1']
['1', '4', '0.00', '0.00', '433.91', '179', '1']
['1', '1', '539.03', '23.64', '0.00', '17', '1']
['1', '2', '562.43', '23.53', '0.00', '255', '1']
['1', '3', '583.28', '23.44', '0.00', '197', '1']
['1', '4', '0.00', '0.00', '433.91', '178', '1']
['1', '1', '589.32', '23.62', '0.00', '16', '1']
['1', '2', '560.75', '23.53', '0.00', '255', '1']
['1', '3', '644.67', '23.44', '0.00', '197', '1']
['1', '4', '0.00', '0.00', '433.91', '173', '1']
['1', '1', '539.03', '23.62', '0.00', '15', '1']
['1', '2', '560.75', '23.53', '0.00', '255', '1']
['1', '3', '660.51', '23.42', '0.00', '196', '1']
['1', '4', '0.00', '0.00', '433.91', '173', '1']
['1', '1', '585.18', '23.62', '0.00', '15', '1']
['1', '2', '560.75', '23.51', '0.00', '255', '1']
['1', '3', '574.51', '23.44', '0.00', '198', '1']
```

Fig 27. Output of Sensors Communication on Raspberry Pi

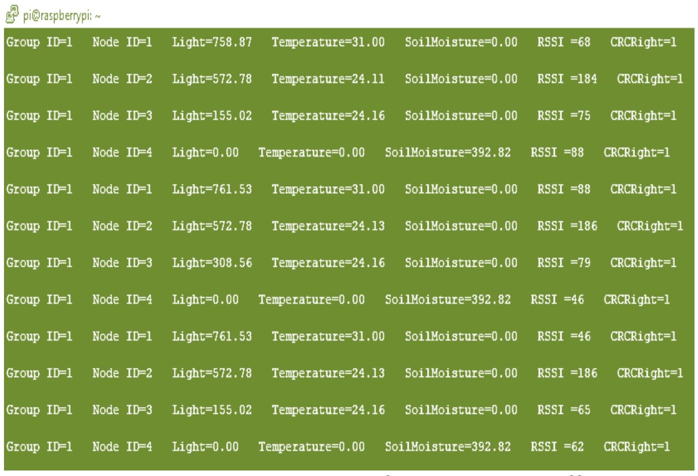
f. The Application of RSSI

The data in the 6th row in the picture below reflecting the data outputs is RSSI data, and as we mentioned in Chapter 3, it is affected by electromagnetic waves, when slave nodes move away from the master node or by various environmental factors and the RSSI output becomes different. To prove this, we examined the RSSI change by bringing our phones closer to the master node.



```
Group ID=1   Node ID=1   Light=554.18   Temperature=31.00   SoilMoisture=0.00   RSSI =186   CRCRight=1

Group ID=1   Node ID=2   Light=569.29   Temperature=24.00   SoilMoisture=0.00   RSSI =255   CRCRight=1

Group ID=1   Node ID=3   Light=614.14   Temperature=24.15   SoilMoisture=0.00   RSSI =219   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=372.27   RSSI =188   CRCRight=1

Group ID=1   Node ID=1   Light=477.75   Temperature=31.00   SoilMoisture=0.00   RSSI =188   CRCRight=1

Group ID=1   Node ID=2   Light=569.29   Temperature=24.00   SoilMoisture=0.00   RSSI =255   CRCRight=1

Group ID=1   Node ID=3   Light=578.00   Temperature=24.15   SoilMoisture=0.00   RSSI =223   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=372.27   RSSI =185   CRCRight=1

Group ID=1   Node ID=1   Light=534.20   Temperature=31.00   SoilMoisture=0.00   RSSI =185   CRCRight=1

Group ID=1   Node ID=2   Light=569.29   Temperature=24.00   SoilMoisture=0.00   RSSI =255   CRCRight=1

Group ID=1   Node ID=3   Light=621.64   Temperature=24.13   SoilMoisture=0.00   RSSI =223   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=374.32   RSSI =192   CRCRight=1
```

Fig 28. RSSI Output without any effect



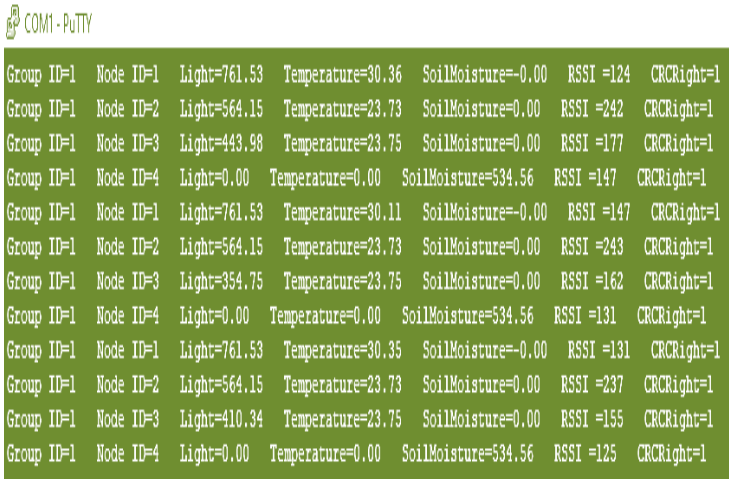
```
Group ID=1   Node ID=1   Light=758.87   Temperature=31.00   SoilMoisture=0.00   RSSI =68   CRCRight=1

Group ID=1   Node ID=2   Light=572.78   Temperature=24.11   SoilMoisture=0.00   RSSI =184   CRCRight=1

Group ID=1   Node ID=3   Light=155.02   Temperature=24.16   SoilMoisture=0.00   RSSI =75   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=392.82   RSSI =88   CRCRight=1

Group ID=1   Node ID=1   Light=761.53   Temperature=31.00   SoilMoisture=0.00   RSSI =88   CRCRight=1

Group ID=1   Node ID=2   Light=572.78   Temperature=24.13   SoilMoisture=0.00   RSSI =186   CRCRight=1

Group ID=1   Node ID=3   Light=308.56   Temperature=24.16   SoilMoisture=0.00   RSSI =79   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=392.82   RSSI =46   CRCRight=1

Group ID=1   Node ID=1   Light=761.53   Temperature=31.00   SoilMoisture=0.00   RSSI =46   CRCRight=1

Group ID=1   Node ID=2   Light=572.78   Temperature=24.13   SoilMoisture=0.00   RSSI =186   CRCRight=1

Group ID=1   Node ID=3   Light=155.02   Temperature=24.16   SoilMoisture=0.00   RSSI =65   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=392.82   RSSI =62   CRCRight=1
```

Fig 29. RSSI output as a result of electromagnetic effect

Our RSSI output, which we do not bring our phone closer to, is in the range of 185-255, while when we bring the phone closer, it is in the range of 46-68.

g. Raspberry Pi

Once the three sensor boards and the grove moisture sensor were connected to the 4 PCBs. And RS-232 serial cable was connected from the I/O port-RS-232 of the PCBs to the computer, to allow sending serial data from the PIC to the PC and see our readings using Putty. The Putty console shown below presents the serial data of the sensor's readings.



```
Group ID=1   Node ID=1   Light=761.53   Temperature=30.36   SoilMoisture=-0.00   RSSI =124   CRCRight=1
Group ID=1   Node ID=2   Light=564.15   Temperature=23.73   SoilMoisture=0.00   RSSI =242   CRCRight=1
Group ID=1   Node ID=3   Light=443.98   Temperature=23.75   SoilMoisture=0.00   RSSI =177   CRCRight=1
Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=534.56   RSSI =147   CRCRight=1
Group ID=1   Node ID=1   Light=761.53   Temperature=30.11   SoilMoisture=-0.00   RSSI =147   CRCRight=1
Group ID=1   Node ID=2   Light=564.15   Temperature=23.73   SoilMoisture=0.00   RSSI =243   CRCRight=1
Group ID=1   Node ID=3   Light=354.75   Temperature=23.75   SoilMoisture=0.00   RSSI =162   CRCRight=1
Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=534.56   RSSI =131   CRCRight=1
Group ID=1   Node ID=1   Light=761.53   Temperature=30.35   SoilMoisture=-0.00   RSSI =131   CRCRight=1
Group ID=1   Node ID=2   Light=564.15   Temperature=23.73   SoilMoisture=0.00   RSSI =237   CRCRight=1
Group ID=1   Node ID=3   Light=410.34   Temperature=23.75   SoilMoisture=0.00   RSSI =155   CRCRight=1
Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=534.56   RSSI =125   CRCRight=1
```

Fig 30. Reading Serial RS-232 data in Putty

The Putty console above displays and describes the final our 4 PCBs output, the group ID, which is group 1, different node IDs, as we used 4 nodes, also demonstrates the readings of light, temperature, and moisture sensors. It is clear to see that node 1, 2 and 3 shows only the light and the temperature sensors readings, as these sensors were connected to the mentioned node only. And node 4 only displays the moisture sensor as it is connected to node 4.

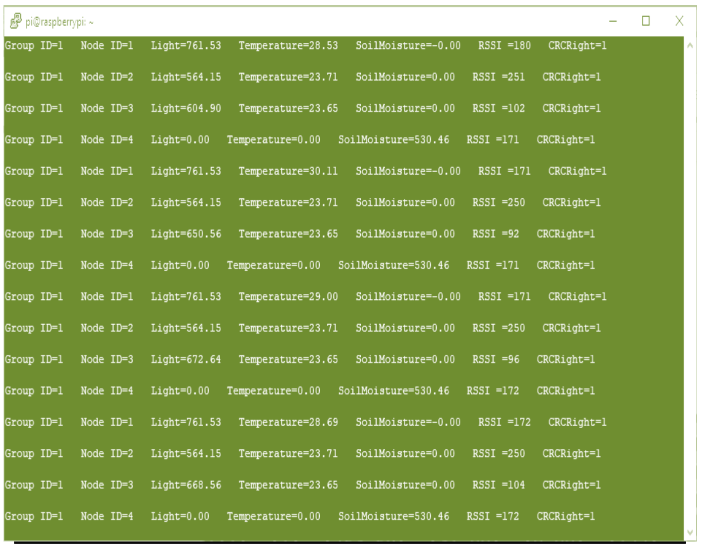
The TDMA working correctly as no collision occur on the output readings of the sensors, the nodes are sequentially displayed with their data.

The value of received signal strength indicator (RSSI) is less than 256 in all nodes, as we defined its data type to be unsigned integer, and the value of CRCRight =1, as it set its flag to be 1 when CRC is right, which means we are receiving a correct value for our readings.

The whole readings are in digital data, analogue to digital converter channels were opened for light sensor data using channel 3, temperature sensor data using channel 4, and moisture sensor data using channel 5.

Finally, Connecting RS-232 cable between the master node and the raspberry pi, to run Python read_serial.py code. The figure below displays the serial data of the sensor's readings using raspberry pi.

The raspberry pi outputting the same Reading Serial RS-232 data in Putty.exe, it describes the final our 4 PCBs output, as it is illustrated in the figure above.



```
Group ID=1   Node ID=1   Light=761.53   Temperature=28.53   SoilMoisture=-0.00   RSSI =180   CRCRight=1

Group ID=1   Node ID=2   Light=564.15   Temperature=23.71   SoilMoisture=0.00   RSSI =251   CRCRight=1

Group ID=1   Node ID=3   Light=604.90   Temperature=23.65   SoilMoisture=0.00   RSSI =102   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=530.46   RSSI =171   CRCRight=1

Group ID=1   Node ID=1   Light=761.53   Temperature=30.11   SoilMoisture=-0.00   RSSI =171   CRCRight=1

Group ID=1   Node ID=2   Light=564.15   Temperature=23.71   SoilMoisture=0.00   RSSI =250   CRCRight=1

Group ID=1   Node ID=3   Light=650.56   Temperature=23.65   SoilMoisture=0.00   RSSI =92   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=530.46   RSSI =171   CRCRight=1

Group ID=1   Node ID=1   Light=761.53   Temperature=29.00   SoilMoisture=-0.00   RSSI =171   CRCRight=1

Group ID=1   Node ID=2   Light=564.15   Temperature=23.71   SoilMoisture=0.00   RSSI =250   CRCRight=1

Group ID=1   Node ID=3   Light=672.64   Temperature=23.65   SoilMoisture=0.00   RSSI =96   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=530.46   RSSI =172   CRCRight=1

Group ID=1   Node ID=1   Light=761.53   Temperature=28.69   SoilMoisture=-0.00   RSSI =172   CRCRight=1

Group ID=1   Node ID=2   Light=564.15   Temperature=23.71   SoilMoisture=0.00   RSSI =250   CRCRight=1

Group ID=1   Node ID=3   Light=668.56   Temperature=23.65   SoilMoisture=0.00   RSSI =104   CRCRight=1

Group ID=1   Node ID=4   Light=0.00   Temperature=0.00   SoilMoisture=530.46   RSSI =172   CRCRight=1
```

Fig 31. Reading Serial RS-232 data in Raspberry Pi

h. Thing Speak Outputs

Running the Thing Speak code on raspberry pi using the following command (python thingspeak.py) Cloud results through Thing Speak Using Raspberry Pi.

The diagrams below display the light sensors readings.

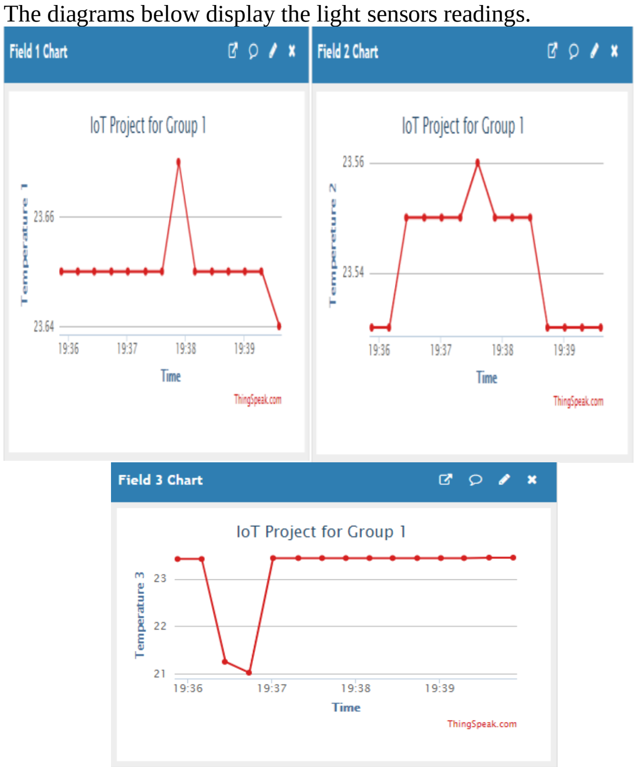
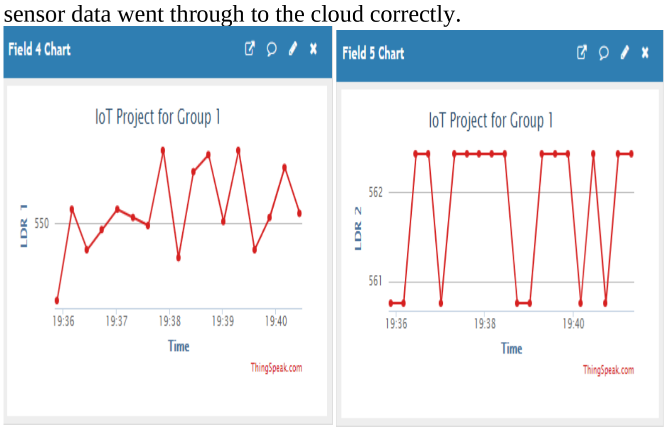
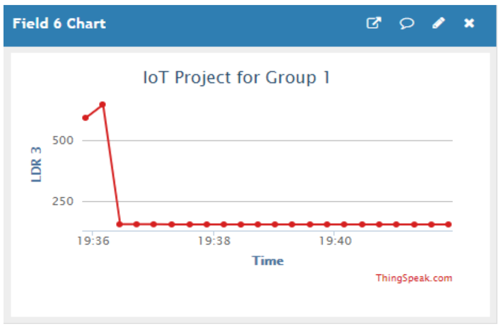


Fig 32: LDR Outputs on ThingSpeak

The figures above are illustrated the 3 temperature sensors readings. It is evident that the third temperature sensor on field 3 decreased to 21°C as we were holding the sensor by our hand to ensure that the sensor data went through to the cloud correctly.





1. **Fig 33**: Temperature Sensor Outputs on ThingSpeak

The figures above are demonstrated the 3 light sensors readings. As can be seen each light sesonr has a diferent sensor as we were covering the light sesor it self to make these changes and have a diffent reading to make sure that we are sending a correct data to the cloud.

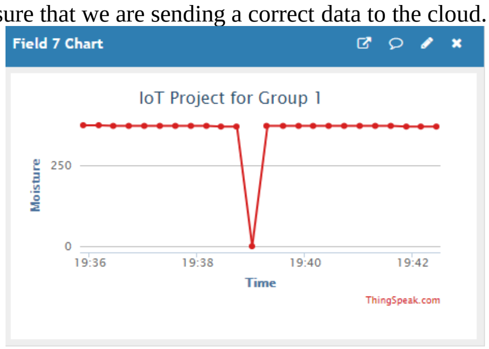


Fig 34: Soil Moisture Sensor Output on ThingSpeak

The figures above are evidenced the moisture sensor readings on the cloud. It's obvious that the reading of the moisture was stable for the 3 seconds as it was plugged to a humid soil and the output was around 400, and we suddenly and quickly pulled the sensor out and putting it back in the soil as it is shown on the diagram at 19.39, just to see the different readings of the moisture sensor.

6.Conclusion and Further Work

In conclusion, this study successfully designed and implemented a real-time environmental monitoring system. leveraging the capabilities of IoT and WSN technologies. The system effectively integrates low-power, cost-effective wireless sensors to monitor critical environmental parameters, namely temperature, light intensity, and soil moisture, addressing the specific climatic challenges of the airport environment. The deployment of a four-node sensor network, utilizing Zigbee communication and a Raspberry Pi gateway, demonstrated the feasibility of real-time data acquisition, reliable wireless transmission secured by TDMA and CRC, and cloud-based visualization via the ThingSpeak platform. The system's ability to capture and transmit sensor data, as evidenced by the ThingSpeak integration and data analysis, confirms its potential to enhance passenger comfort through automated climate control and promote sustainable airport operations through optimized irrigation management. Looking ahead, future work should focus on expanding the sensor network to incorporate air quality and noise level monitoring, providing a more comprehensive assessment of the airport environment. Furthermore, investigating and implementing predictive analytics algorithms would enable proactive environmental management, allowing for preemptive adjustments to climate control and irrigation systems based on anticipated environmental changes. Longitudinal studies evaluating the system's performance over extended periods and across varying seasonal conditions are also recommended to quantify its long-term impact and refine its operational parameters for optimal effectiveness and sustainability within diverse airport workspaces and green spaces.

7.References

[1]Fiona, G., et al (2020)., Airports and environmental sustainability: a comprehensive review, Environmental Research Letters, DOI:10.1088/1748-9326/abb42a

[2]Chaitanya, R. , et al. , (2024) ,Environmental Monitoring using IoT using Ardunio UNO and Data Analysis.,International

Conference on Automation, Computing and Renewable Systems (ICACRS) DOI: 10.1109/ICACRS62842

[3]Jie, Liu., et al. , (2024), Internet of things challenges and future scope for enhanced living environments. Advances in Computers, Volume 133, Pages 201-246, https://doi.org/10.1016/bs.adcom.

[4]Aceto, G., et al., (2019), A survey on information and communication technologies for industry 4.0: State-of-the-art, taxonomies, perspectives, and challenges.. 21(4): p. 3467-3501.

[5]Karuna , G, et al. ., (2024) ,Smart energy management: real-time prediction and optimization for IoT-enabled smart homes, Article: 2390674 | Received 12 Jun, Accepted 06 , Published online:

[6]Ahmad, W., Rasoo A, Javed; Baker, A.R.T, (2022), Jalil, Z. Cyber. "Security in IoT-Based Cloud Computing: A Comprehensive.,Electronics , 11(1),16; https://doi.org/10.3390/electronics11010016

[7]Majid, M., et al., (2022), Applications of wireless sensor networks and internet of things frameworks in the industry revolution 4.0: A systematic literature review.. 22(6): p. 2087.

[8]Landaluce, H., et al., (2020), A review of IoT sensing applications and challenges using RFID and wireless sensor networks.,. 20(9): p. 2495.

[9]Sandro, N., et al., (2020), Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable futureJ Clean Prod ., 19;274:122877.doi: 10.1016/j.jclepro.2020.122877

[10] Cho, Y. M, Kim and S, Woo ., (2018) ,Energy efficient IoT based on wireless sensor networks, 20th International Conference on Advanced Communication Technology (ICACT).. IEEE.

[11] Lakshmi, N., et al (2024)., Advances in real time smart monitoring of environmental parameters using IoT and sensors, Volume 10, Issue 7,, e28195 https://doi.org/10.1016/j.heliyon.2024.e28195

[12] Sudarsono, A.; Huda, S.; Fahmi, N.; Al-Rasyid . (2016), M.U.H.; Kristalina, P. Secure Data Exchange in Environmental Health Monitoring System through Wireless Sensor Network. Int. J. Eng. Technol. Innov. , 6, 103–122.

[13] A. Atiya, M. Khaled, and Abu. Mahdi," Implementation of the modern control engineering mechanism through the USB technology for the meteorological system at Sabha International Airport", October 18 No. 4 (2022): The second International Conference on Science and Technology,

[14] Ullo, S.L. and G.R.J.S. Sinha, Advances in smart environment monitoring systems using IoT and sensors. 2020. 20(11): p. 3113.

[15] Sheral, Z., et al (2021) ., Cryptographic technologies and protocol standards for internet of things, Internet Things, Volume 14, 100075

[16] Hosenkhan. M., Pattanayak .B ., (2020), Security issues in Internet of things (IoT): a comprehensivereview. New Paradigm in Decision Science and Management,: 359 369.https://doi.org/10.1007/978-981-13-9330-3_36

[17] Biswal., A., et al ., (2021) , Adaptive fault-tolerant system and optimalpower allocation for smart vehicles in smart cities using controller area network. Security andCommunication Networks. https://doi.org/10.1155/2021/2147958

[18] Kocakulak, M. and I. Butun., (2017) , An overview of Wireless Sensor Networks towards internet of things, IEEE 7th annual computing and communication workshop and conference

[19] Sadikin, F., T , (2020) ,Van Deursen, and S.J.I.o.T. Kumar, A ZigBee intrusion detection system for IoT using secure and efficient data collection.. 12: p. 100306.

[20] Ghael, D, Solanki. L, Sahu. G. , (2021) , A Review Paper on Raspberry Pi and its Applications,International Journal of Advances in Engineering and Management (IJAEM) Volume 2, Issue 12, pp: 225-227,

[21] Anwaar, W. and M.A.J.E. ,.(2015) Shah, Energy efficient computing: A comparison of raspberry pi with modern devices.. 4(02).

[22] Das, S., et al. ,..(2014) Performance analysis of TDMA based data transmission in WSN. in 14th International Conference on Intelligent Systems Design and Applications.. IEEE.

[23] Phan,G. , (2024) , TDMA-Based MAC Protocol for Wireless Sensor Networks: Design and Evaluation on the WLAN Physical Layer "Master's Programme in Computer, Communication and Information Science, Date 31.07. Number of pages 55.

[24] Pratap, A., Tomar . P, (2021) , Chapter 4 - AI and IoT Capabilities: Standards, Procedures, Applications, and Protocols, Available online , Version of Record 15 January.

[25] Ou, C.-H., et al,. (2020) Design and implementation of anomaly condition detection in agricultural IoT platform system. in 2020 International Conference on Information Networking (ICOIN).. IEEE.

[26] Samimian, N., M. Mousazadeh, and A. Khoie. A time-based all-digital analog to digital converter for IOT applications. in 2019 27th Iranian Conference on Electrical Engineering (ICEE). 2019. IEEE.

[27] Anchit, G., et al (2016)., APPLICATION OF SOIL MOISTURE SENSORS IN AGRICULTURE: A REVIEW., Conference: International Conference on Hydraulics, Water Resources and Coastal Engineering.