ILKOM Jurnal Ilmiah Vol. 17, No. 2, August 2025, pp. 120-130
Accredited 2nd by KEMDIKTISAINTEK No. 10/C/C3/DT.05.00/2025; E-ISSN 2548-7779 | P-ISSN 2087-1716

Low Power Consumption IoT Weather Monitoring System for Coastal Areas

Muhammad Hibrian Wiwi [a,1,*]; Muhammad Awaluddin[a,2]; Risky Saharis[a,3]
[a] *Universitas Muhammadiyah Buton, Jl. Betoambari No. 36, Baubau city, Indonesia*
[1] *muhammadhibrian@gmail.com;* [2] *awalmuh.ismail@gmail.com;* [3]*risky25saharis@gmail.com*
* *Corresponding author*

Article history: Received September 28, 2024; Revised December 03, 2025; Accepted April 30, 2025; Available online August 14, 2025

Abstract

This research aims to design and develop a real-time weather monitoring system in coastal areas using weather sensors integrated with Internet of Things (IoT) technology. This system wants to provide accurate weather information that fishermen can access directly through web-based platforms or mobile devices. Fishermen can make the right and safe decisions before going to sea, improving fishing activities' safety and efficiency. How to overcome the limitations of traditional weather monitoring methods that are manual, non-real-time, and discontinuous, and how to design systems that can transmit weather data automatically and in real-time in coastal areas that are often difficult to reach by power grids and stable internet. Another essential issue formulated is how to optimize the power consumption of sensor and communication systems so that they can operate efficiently using battery power resources for an adequate period. This study confirms the importance of energy efficiency in battery-based monitoring systems, especially in sensor nodes installed in remote locations without a fixed power source. The test results show the difference between the estimated operating time based on theoretical calculations and the results of simulations or real tests in the field. In idle mode, theoretical calculations estimate the system can last up to about 12.8 hours, but actual simulations show an endurance of about 7.3 hours. Meanwhile, in active mode, the estimated calculation is around 5.5 hours, while the simulation shows a slightly longer endurance, which is about 7.2 hours.

Keywords: Internet of Things; Weather Monitoring; Coastal Area; Fishermen Activities

Introduction

Weather information is essential in everyday life, for example in transportation, agriculture, fisheries, marine affairs, and so on. [1], [2]. Weather is the condition of the air in a place that covers a certain area [3], [4]. In the traditional method, weather measurements are made using data sources that are not real time, not continuous and are laborious [5], [6]. The process of retrieving data on weather conditions in real time can use internet networks such as the Internet of Things Technology [7][8] [9] and there is also real time weather monitoring also without using an internet network that directly sends the data to end users, namely by using wireless sensor network (WSN) technology [10], [11] The utilization of internet of things (IoT) and WSN technology is not only used in smart home electronic devices [12], smart city [13] health technology and can also be applied to marine and maritime technology [14].

Weather or climate conditions in uncertain sea waters can disrupt the activity of the fishing process, even in the process of sailing sometimes it can threaten the health and safety of the fishermen [15], [16]. The process of seeing weather conditions is still conventional, which must be seen directly to the coastal area between seeing the conditions of wave height, rainfall and wind speed, to determine sailing in the ocean [16], [17]. Air quality and weather monitoring have also been researched in urban areas, using wireless sensor network technology to collect data in the form of temperature, humidity and carbon dioxide, the data is processed to be sent to the server. This research is in line with the plantation automation system and weather monitoring [18]. Automatic weather monitoring that sends climate data dynamically, continuously and in realtime[19], [20] using internet of things technology and embedded systems that transmit humidity, temperature and climate change data and then display it in the form of graphs.

Some previous studies used several sensors and transmitted data using a wireless network and then connected to the internet. Ronak Salvi et al., developed a weather prediction system and evacuation procedures based on satellite communication that are more efficient, simple, and widely accessible, especially to help people in rural and remote areas such as farmers and fishermen [21]. Stoyan Stoyanov et al. This research developed a weather monitoring system that uses data collected by an IoT (Internet of Things)-based automatic weather station that is made by yourself, but

http://dx.doi.org/10.33096/ilkom.v17i2.2456.120-130

Dependence on Wi-Fi connection between the sensor module and the central module. The connection quality must be good and the distance must not be more than 25 meters [7]. Zhang Huaqing et al. put forward the management of road safety in bad weather conditions, reducing the number of traffic accidents due to bad weather, and ensuring the safety of people's property [22] Arleiny Arleiny et al. conducted a study on optimizing battery consumption for Android-based Automatic Identification System (AIS) applications used by Indonesian fishermen by applying techniques such as data transmission. Arleiny Arleiny et al. conducted a study on optimizing battery consumption for Android-based Automatic Identification System (AIS) applications used by Indonesian fishermen by applying techniques such as data transmission [23], [24]. Rohi Bindal et al. designed an efficient Internet of Things (IoT)[25] based Weather Monitoring Station using NodeMCU ESP8266 to measure temperature, pressure, humidity, and rainfall in the surrounding environment [26]. Several previous studies depended on a stable internet connection to transmit data and did not explain the use of battery power or resources used by the sensors and microcontrollers. This study uses two Raspberry Pi controllers as single-board computers to control sensor components and communication modules, one as a server and the other as a client, in order to efficiently use power consumption and utilize xbee communication from the client to the server to optimize data delivery to the internet. The sensors and actuators installed are wind sensors, water level sensors, rainfall sensors, and light intensity sensors, while the communication module uses XBEE.

Internet of things utilization[27] and implementation of embed systems in everyday life[28] and simulation of using wireless sensor network [29] previously researched. The research conducted after this is expected to help fishermen in particular and people who live around the coast in general, to take advantage of technology in paying attention to the weather when they are in the beach area or people who need real-time information when they want to catch fish and go to sea.

Method

The research methods used are quantitative and experimental methods. The reason for using this method is because this research is building and designing a prototype on an object. In addition, the research was carried out in two places, the first in the Computer Systems Engineering Laboratory as an initial testing ground and in the second place in the field or in the coastal area to conduct direct tool testing.

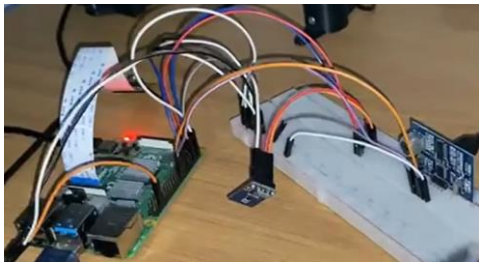

Figure 1. Tool Testing in the Laboratory

The general description of this system is that it can provide data directly or can monitor climate or weather conditions such as sending data in the form of rainfall sensors, wind speed sensors, water level sensors and light intensity sensors and mini computers (Raspberry Pi) from one node to the end node, the other node does not have a power source but uses a battery so it is easy to place anywhere, then the data will be displayed in real time on the website. The website will interact with the database to request and send data in the field. Then it will be displayed to the user, in this case the fishermen can access it using a laptop or mobile phone device connected to the internet network. The stages or research methodology performed are identification of problems and observing the research location. The research location will be carried out in the coastal area of the beach then supported by a review of relevant literature and supporting theories that are related and sustainable to the research problem being carried out.

Figure 2. Equipment Test on The Coast

Before designing and designing a system, a needs analysis is carried out with the aim of being able to create a system that is needed. Examples such as Analysis of the external needs of the system and Analysis of the internal needs of the system. Furthermore, by analysing existing needs and designing or designing the system is needed, for example, such as how to connect various microcontroller devices as controllers and wireless sensor networks (JSN) as a wireless-based communication medium. The process of searching, processing, and sending information automatically and continuously can run well whose data is from the field so that it can be read by users. The system built is a system on hardware and software. The hardware system as shown in **Figure 1**. The software used uses the python programming language. The prototype made will be tested indoors and outdoors or on the coast and other devices placed in a house or building connected to a modem connected to the internet. Monitoring will be monitored using a laptop or mobile phone to see the state or condition of the weather in real time whose devices are connected to the internet as well.

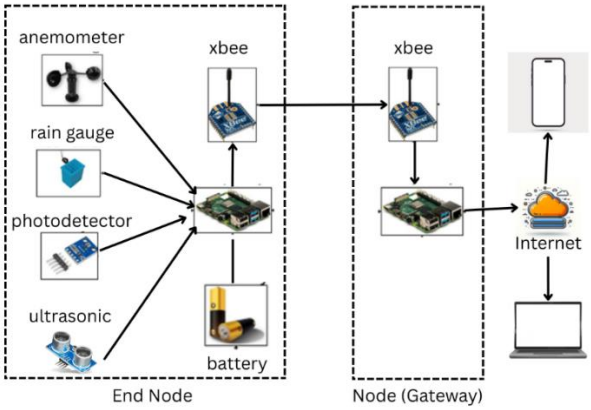

Figure 3. Real-Time Weather Monitoring System Architecture

The figure above describes the sensors and actuators placed around the coastal area. Wind sensor to measure and send wind speed data Water level sensor to measure sea level Rainfall sensor to measure rainfall intensity or parameters [30], [31] Light intensity sensor functions to convert light intensity into electrical signals . Then what acts as a controller or processor is a raspberry pi [32] and which acts as a data sender and receiver using xbee[33]. The system consists of an end node equipped with an anemometer, rain gauge, photodetector, and ultrasonic sensor, all powered by a battery. Data collected is transmitted wirelessly via an XBee module to a gateway node, which also utilizes an XBee module for communication. The gateway is connected to the Internet, facilitating data transfer to the cloud. Users can then access this data remotely through devices like smartphones or computers.

Table 1. Changes in Weather Conditions

Current Time	Number of Tips	Rainfall per Minute	Rainfall per Hour	Rainfall Today	Weather	Rotation per Second	Speed (m/s)	Speed (km/h)	Weather Light
18:20:08	0	0	0	0	Overcast	0	0	0	0
18:20:18	0	0	0	0	Overcast	0.2	1.681956	6.0550416	Cloudy (48.33 lux)
18:20:48	17	6.3	6.3	11.9	Light rain	0.3	1.819641	6.550708	Cloudy (1.67 lux)
18:20:58	24	5.6	11.9	16.8	Light rain	0.3	1.819641	6.550708	Cloudy (9.17 lux)
18:21:08	30	5.6	17.5	21	Moderate rain	0.3	1.819641	6.550708	Night/Rain (14.27 lux)
18:21:18	37	4.2	21.7	25.9	Moderate rain	0.3	1.819641	6.550708	Night/Rain (14.27 lux)
18:27:38	49	0	34.3	34.3	Moderate rain	0.9	2.638149	9.4973364	Cloudy (43.33 lux)
18:27:48	49	0	34.3	34.3	Moderate rain	0.9	2.638149	9.4973364	Cloudy (45.83 lux)
18:28:28	49	0	34.3	34.3	Moderate rain	0.9	2.638149	9.4973364	Cloudy (47.50 lux)
18:28:38	49	0	34.3	34.3	Moderate rain	0.9	2.638149	9.4973364	Cloudy (47.50 lux)
18:28:48	49	0	34.3	34.3	Moderate rain	1	2.7733	9.98388	Cloudy (47.50 lux)
18:29:58	49	0	34.3	34.3	Moderate rain	0.9	2.638149	9.4973364	Cloudy (1.67 lux)

Current Time	Number of Tips	Rainfall per Minute	Rainfall per Hour	Rainfall Today	Weather	Rotation per Second	Speed (m/s)	Speed (km/h)	Weather Light
18:30:08	49	0	34.3	34.3	Moderate rain	0.5	2.093925	7.53813	Cloudy (49.17 lux)
18:30:18	49	0	34.3	34.3	Moderate rain	0.8	2.502636	9.00949	Cloudy(11.67 lux)

Table 1 The captured data shows detailed changes in weather conditions at each time interval. The observation started with overcast conditions and light rain, as seen at 18:20:38 when rainfall reached 5.6 mm, characterised by 8 measured raindrops (tip count). During this period, the wind speed was measured to be around 6.55 km/h with a very low light intensity of only 0.83 lux, indicating a dark night with light rain.

Over time, the intensity of the rain increased. At 18:26:28, the cumulative rainfall for the day reached 34.3 mm, reflecting continuing moderate rainy conditions. Despite the ongoing rain, the light intensity increased to 47.50 lux, indicating that despite the overcast, there was a slight increase in brightness in the environment. Wind speed also experienced mild variations, with sensor rotations recording movements between 0.9 to 1 rotation per second, indicating wind speed fluctuations from 9.49 km/h to 9.98 km/h. The weather conditions recorded in the data ranged from 'Light Rain' to 'Moderate Rain', with the light intensity decreasing as the rain became heavier, reaching only 1.67 lux at times. Overall, the data reflects dynamic weather changes with persistent rain and small variations in wind speed and light intensity.

Results and Discussion

The results of this study show that the developed Raspberry Pi-based weather monitoring system is capable of integrating various sensors (wind speed, water level, rainfall, and light intensity), and XBEE communication modules to monitor and send real-time weather data from coastal areas to a web-based platform. The data can be accessed by coastal communities, especially fishermen, through laptops or smartphones, providing them with up-to-date weather information that supports safer and more efficient fishing activities. Testing of the prototype was conducted in both laboratory conditions and real coastal environments. The system successfully collects and transmits weather data continuously, utilising wireless sensor networks and Internet of Things (IoT) technology. As a result, the system is able to provide real-time and automatic weather updates, thereby increasing the effectiveness of weather monitoring in the maritime sector and reducing dependence on traditional weather monitoring methods that are manual and not continuous.

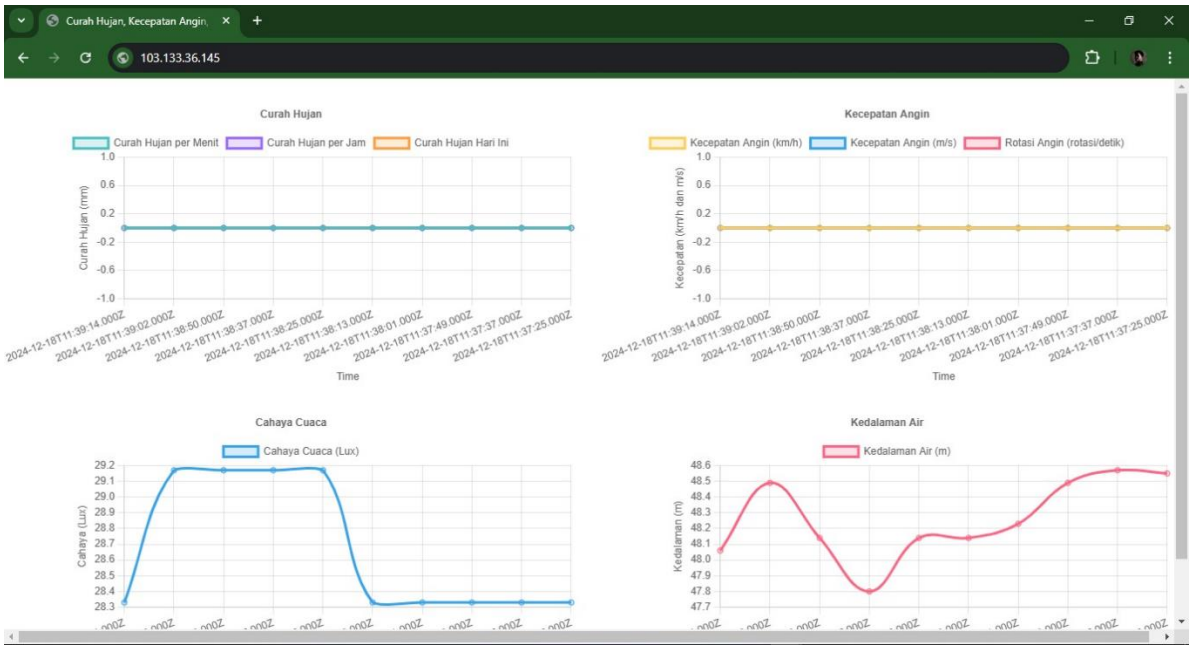

Figure 4. Real-time Weather Monitoring of Rainfall, wind speed, light intensity, and water surface elevation

Figure 4 The results showed that a weather monitoring system based on Raspberry Pi and Python programming language was successfully developed to collect real-time weather data using various sensors. Raspberry Pi acts as the main controller connected to several weather sensors, such as wind speed sensor (anemometer), rainfall sensor (rain gauge), water level sensor, and light intensity sensor. Data from these sensors is taken automatically through the GPIO

pin on the Raspberry Pi, with data processing done using Python scripts. The Raspberry Pi is able to read the signals from the sensors and convert them into relevant data. For example, the anemometer measures wind speed based on the number of rotations produced by the wind blowing, which the Raspberry Pi calculates and converts into wind speed in meters per second. Similarly, the rainfall sensor measures the amount of water collected and sends a signal that is processed into total rainfall in millimeters.

Data transmission is done wirelessly via an XBEE communication module connected to a Raspberry Pi. The collected and processed data is then sent to a web-based platform using the HTTP protocol, allowing users to monitor the weather in real-time via a laptop or smartphone. The system was tested both in a laboratory environment and in a coastal area, with results showing that the system can monitor the weather continuously and provide accurate data in real time. The test results also showed that the use of Raspberry Pi and Python provided flexibility in programming as well as full automation capabilities. This allows the system to operate the process of capturing, processing and transmitting weather data without manual intervention, improving efficiency and accuracy in monitoring weather conditions.

Wind speed sensor data transmission algorithm

1. INITIALIZATION VARIABLES
- *GPIO_PULSE = 18 # GPIO pin for receiving anemometer signals*
- *UPDATE_INTERVAL = 10 seconds # Data update interval*
- *rpmcount = 0 # Number of anemometer rotations*
- *last_micros = current time in microseconds*
- *timeold = current time*
- *rotations_per_second = 0.0*
- *speed_meters_per_second = 0.0*
- *speed_kilometers_per_hour = 0.0*

2. FUNCTION rpm_anemometer(channel):
- *Get the current time in microseconds*
- *If the difference between the current time and the previous time is greater than or equal to 5 milliseconds:*
- *Add 1 to rpmcount*
- *Update last_micros with the current time*

3. FUNCTION get_system_time():
- *Get the hours, minutes, and seconds from the current system time*
- *Return the hours, minutes, and seconds*

4. FUNCTION get_wind_data():
- *Return weather data in the form of:*
- *rotations_per_second*
- *speed_meters_per_second*
- *speed_kilometers_per_hour*

5. FUNCTION print_serial(hours, minutes, seconds):
- *Convert hours, minutes, and seconds to two digits*
- *(Comment: Print the hour and wind speed in m/s and km/h)*

6. FUNCTION run_anemometer():
- *GPIO SETUP:*
- *Use BCM numbering mode on the Raspberry Pi*
- *Configure the GPIO_PULSE pin as an input*
- *Set up event detection to detect signals from the anemometer*

- FOREVER LOOP:
- *Get the current time*
- *Calculate the time difference between now and the last update*
- *If the time difference is greater than or equal to UPDATE_INTERVAL (10 seconds):*
- *Calculate rotations per second = rpmcount / UPDATE_INTERVAL*
- *Calculate the wind speed in meters per second using the formula:*
- *speed_meters_per_second = -0.0181 * (rotations_per_second ^ 2) + 1.3859 * rotations_per_second + 1.4055*
- *If the speed is below 1.5 m/s, set the speed to 0*
- *Calculate the wind speed in kilometers per hour by multiplying the speed in meters per second by 3.6*
- *Get the system time (hours, minutes, seconds)*
- *Print the data using the print_serial() function*
- *Reset rpmcount*
- *Update the last update time*
- *Sleep for 1 second to prevent high CPU usage*
- *If the user terminates the program (KeyboardInterrupt):*
- *Print the message "Program stopped by user"*
- *Finally, clear the GPIO configuration.*

7. RUN the run_anemometer() function

This programme uses a Raspberry Pi to measure wind speed with an anemometer connected via GPIO pins. In initialisation, variables such as the number of revolutions (rpmcount), the last time (last_micros), and the wind speed in metres per second and kilometres per hour are set up. The anemometer gives a signal every time it rotates, and the *rpm_anemometer* function detects the signal and counts the number of revolutions. After a 10-second interval, the wind speed is calculated based on the rotation frequency of the anemometer. The wind speed in metres per second is calculated by a formula that considers the number of revolutions per second, and then converted to kilometres per hour. If the wind speed is below 1.5 metres per second, it is considered that there is no wind and the speed is set to zero. The programme continuously runs in a loop that checks every second to ensure data updates are made at the specified interval. Time and wind speed data can be displayed, and the programme can be safely stopped by the user by clearing the GPIO configuration on exit.

Rainfall sensor data transmission algorithm

1. Initialize Constants and Variables:
- Set PIN_INTERRUPT to GPIO pin 25.
- Set the MILLIMETER_PER_TIP conversion to 0.70 mm per rain tip.
- Set UPDATE_INTERVAL to 10 seconds.
- Initialize variables such as the number of rain tips, rainfall per minute, per hour, and per day.
2. Rainfall_Calculation() function:
- When there is an interrupt from the rain sensor, set the flag variable to True to indicate that rain has been detected.
3. get_system_time() function:
- Get the system time (hour, minute, second) for use in calculations and output.
4. get_rainfall_data() function:
- Return rainfall data including the number of tips, rainfall per minute, hour, day, and current weather conditions.
5. The print_serial() function:
- Formats the clock, weather, tip count, and rainfall data into a formatted string (e.g., mm rainfall) for display.

6. The main function of run_rain_gauge():
- Initialize GPIO:
- Set GPIO mode and setup for interrupt on PIN_INTERRUPT.
- Register a callback to call the calculate_rain_gauge function whenever an interrupt is detected.

- Main loop (while True):
- Get the current time.
- Calculate the elapsed time since the last update.

- If rain is detected (flag == True):
- Add MILLIMETER_PER_TIP to the total rainfall.
- Add 1 to the tip count.
- Set the flag back to False.

- Update the weather conditions based on today's rainfall:
- If rain is less than 0.5 mm, the weather is "Overcast".

- If it's more than 0.5 mm to 20 mm, the weather is "Light Rain," and so on up to "Extreme Rain."

- If the update interval is reached (>= 10 seconds):
- Calculate the rainfall per minute, hour, and day.
- If the hour or minute is zero, reset the rainfall per hour or day.
- If the time is 00:00, reset the rainfall for the new day.
- Display the updated rainfall data.

- Sleep for 1 second to prevent high CPU usage.

7. Close the program with KeyboardInterrupt:
- When the user terminates the program, the GPIO is cleared.

The pseudocode that has been created explains the structure and workflow of the rainGauge.py programme in a simple and understandable way. First, the program initialises several constants, such as `PIN_INTERRUPT`, which is set to GPIO pin 25 to detect the signal from the rain sensor, and `MILLIMETER_PER_TIP`, which indicates the amount of rainfall measured each time the sensor detects a 'tip.' Variables are also initialised to store rainfall-related data, such as `number_tip`, `rainfall`, and `rainfall_per_minute`. Next, there is the calculate_rainfall() function, which is called whenever the sensor detects an interrupt signal, signalling that rainfall has been detected. The get_system_time() function is responsible for getting the current system time, while the get_rainfall_data() function returns rainfall information in the form of a dictionary. The print_serial() function then formats and displays the rainfall data and weather conditions for the terminal.

The main function, run_rain_gauge(), sets the GPIO mode and prepares the pins to detect an interrupt from the rain sensor. Inside the main loop, the programme measures the elapsed time and detects whether rain is detected. If yes, the rainfall value and the number of tips will be incremented, and the flag will be reset. The programme also updates the weather condition based on the total rainfall recorded. When the elapsed time reaches UPDATE_INTERVAL, the programme calculates and resets the rainfall values per minute, per hour, and per day, and displays the data if any changes occur. To avoid excessive CPU usage, the programme pauses for one second. Finally, the programme handles situations where the user manually stops the programme by clearing the GPIO settings, ensuring all pins return to a safe state. As such, this pseudocode provides a clear overview of the programme's functionality, allowing developers to understand and modify the programme's logic flow as needed.

This study also analyzes the power consumption and estimated operating time of battery-based sensor systems, especially for weather or environmental monitoring devices. The data in this file is arranged in several main sections which include sensor technical specifications, power consumption calculations, and battery durability simulations under various operating conditions.

list of the names of the tools or sensors used, such as Xbee Zigbee, Bh1750, Ultrasonic, Anemometer, Rain Gauge, and Raspberry Pi 4 Model B. For each device, the operating voltage, operating current (both idle and active) are listed, and a description of its usage status. This data is the basis for calculating power consumption on the system as a whole.

Table 2. Power Consumption in Idle Conditions

Tool Name	Operating Voltage	Operational Flow (A)	Description
Raspberry Pi 4 model B	5.1 V	0.54	Idle
	5.1 V	1.01	Active
Xbee zigbee	3,3 V	0.04	Idle
	3,3 V	0.25	Active (transmit)
Bh1750	5 V	0.12	Active
Ultrasonic	5 V	0.015	Active
Anenometer	5 V	0.004	Active
Rain Gauge	5 V	0.002	Active
Total Power Consumption		0.58	Idle

Power consumption data for various electronic devices (**Table 2**), particularly sensor devices and communication modules commonly used in Raspberry Pi-based monitoring systems. The Operating Current column (A) shows the amount of electrical current consumed by each appliance in units of ampere. This current value varies depending on the operational status of the appliance, for example the Raspberry Pi 4 model B consumes a current of 0.54 A at idle and 1.01 A when active. The Xbee Zigbee also has two statuses, namely idle (0.04 A) and active transmit (0.25 A).This table 2 consists of several main columns, namely Tool Name, Operating Voltage, Operating Current (A), and Description. At the bottom of the table is the "Total Power Consumption" row with a current value of 0.58 A at idle. This value is the result of the sum of the operating currents of all the devices while idle, which can be used to estimate the total power requirements of the system in standby mode.

Table 3. Power Consumption in Active Conditions

Tool Name	Operating Voltage	Operational Flow (A)	Description
Raspberry Pi 4 model B	5.1 V	0.54	Idle
	5.1 V	1.01	Active

Tool Name	Operating Voltage	Operational Flow (A)	Description
Xbee zigbee	3,3 V	0.04	Idle
	3,3 V	0.25	Active (transmit)
Bh1750	5 V	0.12	Active
Ultrasonic	5 V	0.015	Active
Anenometer	5 V	0.004	Active
Rain Gauge	5 V	0.002	Active
Total Power Consumption		1.401	Active

The description column provides information about the operational status of each appliance, whether it is idle or active. Most of the devices in this table are in an active state, which is marked by a yellow background color in the description column. In the operational voltage column, each tool has a different value according to the needs of the device, such as the Raspberry Pi which uses 5.1 V, the Xbee Zigbee with 3.3 V, and other sensors generally use 5 V. The operating current column shows the amount of electrical current consumed by each tool during operation, which varies from 0.002 A on the Rain Gauge to 1.01 A on the Raspberry Pi 4 model B in active condition. This status is important to know the maximum power consumption of the system, as current consumption is usually higher when the appliance is actively working than when it is idle.This table 3. also shows the total current consumption of all devices when they are in active condition, which is 1,401 A. This value is the result of summing the operating current of all devices that are currently active, and is very important for the calculation of power requirements and the estimation of battery life in the monitoring system.

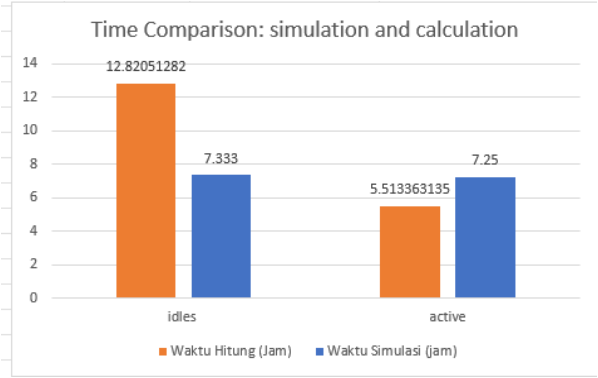

Figure 5. Comparison Simulation and Calculation

The bar chart shows a comparison between the calculation time and the simulation time in two system operating conditions, namely idle and active. The horizontal axis describes the operating conditions, while the vertical axis shows the duration of time in hours units. In idle conditions, the calculation time (marked by the orange bars) was longer, which was about 12.82 hours, compared to the simulation time (blue bars) which was only about 7.33 hours. This suggests that the theoretical calculations estimate that the system can last longer in standby mode compared to more realistic simulation or test-based simulation results. On the other hand, in the active condition, the calculation time was actually shorter, around 5.51 hours, compared to the simulation time which reached 7.25 hours. This means that in active mode, the simulation shows the system could last slightly longer than the calculation estimates.

This diagram shows the difference between the estimated operating time based on mathematical calculations and the simulation results that may consider other factors such as battery efficiency, fluctuations in power consumption, or actual conditions of use. These differences are important to understand so that power and battery capacity planning can be done more accurately and realistically.

Conclusion

This research is the creation of a weather monitoring system using Raspberry Pi technology and IoT-based sensors by integrating various components to collect and analyze weather data in real-time. The results of this research also show that the developed system can provide useful weather information for the community, especially fishermen, by accessing weather data directly through a web-based platform. Research can also consider the application of renewable energy technologies, such as solar panels, to support the operation of the system, especially in remote areas that may not have access to electricity. Furthermore, the development of interactive mobile applications can enable users, such as fishermen, to obtain weather data directly on their mobile devices. This Research can be conducted to

apply this system in different geographical locations and environments, including mountainous and urban areas, to understand the variations and challenges that may arise. In terms of power consumption, the study also highlights the importance of energy efficiency in battery-based monitoring systems, especially for sensor nodes that are placed in remote locations and are not connected to a permanent power source. The test results show a difference between the estimated operating time based on theoretical calculations and the results of real simulations/tests. In idle mode, calculations show that the system can last up to about 12.8 hours, but the actual simulation results in the field are only about 7.3 hours. In contrast, in active mode, the estimated operating time is around 5.5 hours, while the simulation results show that the system can last up to 7.2 hours. This difference signifies that factors such as battery efficiency, fluctuations in power consumption, and real environmental conditions greatly affect the durability of the system.

Acknowledgement

The authors would like to thank the DRTPM grant (Directorate General of Higher Education, Ministry of Education and Culture of the Republic of Indonesia) for the financial and moral support that has been channelled to us. This grant has made a significant contribution in supporting the implementation of this research. The assistance provided not only made this research possible, but also opened up opportunities for further development in the field of Computer Engineering.

References

[1] P. Megantoro, B. A. Pramudita, P. Vigneshwaran, A. Yurianta, and H. A. Winarno, "Real-time monitoring system for weather and air pollutant measurement with html-based ui application," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, pp. 1669–1677, Jun. 2021, doi: 10.11591/eei.v10i3.3030.

[2] T. L. Narayana *et al.*, "Advances in real time smart monitoring of environmental parameters using IoT and sensors," *Heliyon*, vol. 10, no. 7, p. e28195, 2024, doi: 10.1016/j.heliyon.2024.e28195.

[3] P. Megantoro, S. A. Aldhama, G. S. Prihandana, and P. Vigneshwaran, "IoT-based weather station with air quality measurement using ESP32 for environmental aerial condition study," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 19, no. 4, pp. 1316–1325, 2021, doi: 10.12928/TELKOMNIKA.v19i4.18990.

[4] A. Vaidya and V. Bansode, "Iot Based Weather Monitoring System for Agriculture," *International Journal of Engineering Applied Sciences and Technology*, vol. 7, no. 5, pp. 119–122, 2022, doi: 10.33564/ijeast.2022.v07i05.021.

[5] Y. NarasimhaRao, P. Surya Chandra, V. Revathi, and N. Suresh Kumar, "Providing enhanced security in IoT based smart weather system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 1, pp. 9–15, 2019, doi: 10.11591/ijeecs.v18.i1.pp9-15.

[6] N. Ahmed and N. Shakoor, "Advancing agriculture through IoT, Big Data, and AI: A review of smart technologies enabling sustainability," *Smart Agricultural Technology*, vol. 10, no. February, p. 100848, 2025, doi: 10.1016/j.atech.2025.100848.

[7] S. Stoyanov, Z. Kuzmanov, and T. Stoyanova, "Weather Monitoring System Using IoT-based DIY Automatic Weather Station," *2024 9th International Conference on Energy Efficiency and Agricultural Engineering, EE and AE 2024 - Proceedings*, no. June, pp. 1–6, 2024, doi: 10.1109/EEAE60309.2024.10600523.

[8] A. Ramya, R. Rohini, and S. Ravi, "Iot based smart monitoring system for fish farming," *Int J Eng Adv Technol*, vol. 8, no. 6 Special Issue, 2019, doi: 10.35940/ijeat.F1089.0886S19.

[9] J. Junaedi and H. Ki, "Smart Aquarium with IoT based as Monitoring in Fish Farming," *bit-Tech*, vol. 4, no. 3, pp. 116–122, 2022, doi: 10.32877/bt.v4i3.441.

[10] E. Murdyantoro, R. Setiawan, I. Rosyadi, A. W. W. Nugraha, H. Susilawati, and Y. Ramadhani, "Prototype weather station uses LoRa wireless connectivity infrastructure," in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Nov. 2019. doi: 10.1088/1742-6596/1367/1/012089.

[11] V. Choudhary, P. Guha, G. Pau, and S. Mishra, "An overview of smart agriculture using internet of things (IoT) and web services," *Environmental and Sustainability Indicators*, vol. 26, no. April 2024, 2025, doi: 10.1016/j.indic.2025.100607.

[12] P. Franco, J. M. Martinez, Y. C. Kim, and M. A. Ahmed, "IoT Based Approach for Load Monitoring and Activity Recognition in Smart Homes," *IEEE Access*, vol. 9, pp. 45325–45339, 2021, doi: 10.1109/ACCESS.2021.3067029.

[13] R. O. Andrade, S. G. Yoo, L. Tello-Oquendo, and I. Ortiz-Garces, "A Comprehensive Study of the IoT Cybersecurity in Smart Cities," *IEEE Access*, vol. 8, pp. 228922–228941, 2020, doi: 10.1109/ACCESS.2020.3046442.

[14] G. Thamarai Selvi, B. Vishwa, R. Vishwanath, and N. T. Ramprasath, "Comprehensive Security Systems for Fishermen," *4th International Conference on Power, Energy, Control and Transmission Systems: Harnessing Power and Energy for an Affordable Electrification of India, ICPECTS 2024*, pp. 1–5, 2024, doi: 10.1109/ICPECTS62210.2024.10780385.

[15] M. H. Vora, G. Bekaroo, A. Santokhee, S. Juddoo, and D. Roopowa, "JarPi: A low-cost raspberry pi based personal assistant for small-scale fishermen," *IEEE 4th International Conference on Soft Computing and Machine Intelligence, ISCMI 2017*, vol. 2018-Janua, pp. 159–163, 2017, doi: 10.1109/ISCMI.2017.8279618.

[16] A. S. Ajith, O. S. Dhotre, K. P. Hegde, D. R. Naveen, and V. M. Snehalatha, "Border and Weather Alert Security System for Fishermen," *Proceedings - 2023 7th International Conference on Design Innovation for 3 Cs Compute Communicate Control, ICDI3C 2023*, pp. 263–267, 2023, doi: 10.1109/ICDI3C61568.2023.00060.

[17] O. Elijah *et al.*, "Effect of Weather Condition on LoRa IoT Communication Technology in a Tropical Region: Malaysia," *IEEE Access*, vol. 9, pp. 72835–72843, 2021, doi: 10.1109/ACCESS.2021.3080317.

[18] E. Rohadi, R. Admiral Abdurrahman, R. Andrie Asmara, I. Siradjuddin, F. Ronilaya, and A. Setiawan, "SISTEM AUTOMASI PERKEBUNAN DAN PEMANTAUAN CUACA MENGGUNAKAN AWS BERBASIS RASPBERRY PI," vol. 5, no. 6, pp. 711–716, 2018, doi: 10.25126/jtiik.201851121.

[19] J. Mabrouki, M. Azrour, D. Dhiba, Y. Farhaoui, and S. El Hajjaji, "IoT-based data logger for weather monitoring using arduino-based wireless sensor networks with remote graphical application and alerts," *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 25–32, Mar. 2021, doi: 10.26599/BDMA.2020.9020018.

[20] M. Yassir *et al.*, "Performance Analysis of LoRaWAN Communication Utilizing the RFM96 Module," *ILKOM Jurnal Ilmiah*, vol. 16, no. 3, pp. 255–270, 2024, doi: 10.33096/ilkom.v16i3.2326.255-270.

[21] R. Salvi, K. Shinde, J. Jadhav, S. Ubale, and J. Katkar, "Weather prediction and easy evacuation using satellite communication," *2019 International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2019*, pp. 66–68, 2019, doi: 10.1109/WiSPNET45539.2019.9032760.

[22] Z. Huaqing *et al.*, "Research on Early Warning and Monitoring Technology for Severe Weather Conditions on Highways," *2024 IEEE 2nd International Conference on Electrical, Automation and Computer Engineering, ICEACE 2024*, pp. 1562–1565, 2024, doi: 10.1109/ICEACE63551.2024.10898654.

[23] A. Arleiny, A. Z. Arfianto, E. Supriyanto, A. K. H. Maulidi, B. Suwandi, and J. R. Kurniawan Bokau, "Optimizing Battery Consumption for Android-based Automatic Identification System (AIS) Application for Indonesian Fishermen," *International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, no. September, pp. 434–439, 2024, doi: 10.1109/EECSI63442.2024.10776113.

[24] R. Wahyuni, Y. Irawan, A. Febriani, N. Nurhadi, H. Tri Saputra, and R. Andrianto, "Smart Egg Incubator Based on IoT and AI Technology for Modern Poultry Farming," *ILKOM Jurnal Ilmiah*, vol. 16, no. 2, pp. 134–144, 2024, doi: 10.33096/ilkom.v16i2.1957.134-144.

[25] A. Syarifuddin, H. Ma, H. Eren, and A. S. Alghamdi, "Optimizing THD in Modified Multilevel Inverters with IoT-Integrated MPPT Systems for Enhanced Efficiency," vol. 16, no. 2, pp. 198–209, 2024.

[26] R. Bindal, A. Yadav, P. Dewanzan, A. Shinde, Y. Salaria, and S. Deokar, "NodeMCU Based Weather Monitoring System," *Proceedings - International Carnahan Conference on Security Technology*, pp. 0–5, 2023, doi: 10.1109/ICCST59048.2023.10726856.

[27] C. H. Chen, Y. C. Wu, J. X. Zhang, and Y. H. Chen, "IoT-Based Fish Farm Water Quality Monitoring System," *Sensors*, vol. 22, no. 17, 2022, doi: 10.3390/s22176700.

[28] V. Ramakant, N. Khatri, S. Kumar, A. Shaddad, H. Abdul-qawy, and A. Kumar, "A systematic review of IoT technologies and their constituents for smart and sustainable agriculture applications," *Sci Afr*, vol. 19, p. e01577, 2023, doi: 10.1016/j.sciaf.2023.e01577.

[29] A. W. Al-Mutairi and K. M. Al-Aubidy, "IoT-based smart monitoring and management system for fish farming," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 3, pp. 1435–1446, 2023, doi: 10.11591/eei.v12i3.3365.

[30] R. Murgod Tejaswini, S. M. Sundaram, M. Sowmya, and K. S. Santhosh Kumar, "Automatic Fish Feeding and Water Quality Management System using Internet of Things," *MysuruCon 2022 - 2022 IEEE 2nd Mysore Sub Section International Conference*, pp. 1–5, 2022, doi: 10.1109/MysuruCon55714.2022.9972432.

[31] H. Pujiharsono and D. Kurnianto, "Mamdani fuzzy inference system for mapping water quality level of biofloc ponds in catfish cultivation," *Jurnal Teknologi dan Sistem Komputer*, vol. 8, no. 2, pp. 84–88, 2020, doi: 10.14710/jtsiskom.8.2.2020.84-88.

[32] A. Jalil and M. Matalangi, "Object motion detection in home security system using the binary-image comparison method based on robot operating system 2 and Raspberry Pi," *ILKOM Jurnal Ilmiah*, vol. 13, no. 1, pp. 1–8, 2021, doi: 10.33096/ilkom.v13i1.686.1-8.

[33] S. Yogarayan, S. F. A. Razak, M. F. A. Abdullah, A. Azman, V. Arultas, and S. S. Raman, "Comparative performance study of HC-12, nRF24L01, and XBee for vehicular communication," *International Journal of Informatics and Communication Technology*, vol. 12, no. 1, pp. 54–61, 2023, doi: 10.11591/ijict.v12i1.pp54-61.