Ministry of High Education

High Institute of Computer science & Information System

Culture and Science City, 6 October City

المعهد العالي لعلوم الحاسب و نظم المعلومات

Graduation Project

Endangered Animal protection System

Assistant

L.A. Ebtesam El-Hossiny

Supervised by

Prof. Dr. Sami Abd El-Moneim El-Dolil

Project No: N42201 Academic Year: 2021/2022

Ministry of High Education

High Institute of Computer science & Information System

Culture and Science City, 6 October City

المعهد العالي لعلوم الحاسب و نظم المعلومات

Graduation Project

Endangered Animal protection System

Prepared by

41034	إسراء أحمد محمد هندي	41043	آلاء حسن أحمد شاهين حسن
41044	أمنية بشندي عبده سيد	41090	عبد الرحمن مجدي محمد علي
41094	عبدالرحمن محمد عبدالرحمن عبدالحكيم	41146	مصطفى أشرف حسن محمد

41166 يمني حاتم محمد حمدي

Assistant:

L.A. Ebtesam El-Hossiny

Supervised by

Prof. Dr. Sami Abd El-Moneim El-Dolil

Project No: N42201 Academic Year: 2021/2022

Acknowledgment

Firstly, we would like to thank Allah, then thank our institute members. Also, we express our deepest gratitude and appreciation to our Dear Prof. Dr. Sami Abd El-Moneim El-Doll and our project supervisor for his guidance, continuous encouragement, and support during our project. Working with you was, as our project supervisor for always, enlightening and rewarding.

Secondly, we would like to thank Eng. Ebtesam El-Hossiny for her assistance and support in every moment during the project and for providing us with the useful information that helped in our final project.

Finally, to all teaching staff, and all others who assisted us in this effort, thank you all. Also, we offer our appreciation and thanks to the ins Institute of Computer Science and Information System Institution that opened its doors to make us good people who serve their society.

Abstract

We develop a smart system to protect endangered animals. Through a collar that the animal wears around the neck that contains a monitoring system to combat illegal poachers. Also contains a system for monitoring the health of the animal.

This system is directed to animal rights organizations in countries that seek to maintain the balance of their ecosystem and preserve the diversity of their animal wealth from the threat of extinction.

This project solves the problem of animal extinction through two stages, the first is to monitor everything that happens around the animal through cameras that capture everything that happens in the animal's surroundings from natural phenomena such as torrents, storms, or the approach of an illegal hunter of the animal. The second is to monitor the animal's health and physical condition such as heart rate, temperature, etc.

This is through some sensors that collect data.

The way the system works is like this, each of the cameras and sensors is connected to the Arduino board to collect data and establish a connection between the wifi module and the IoT cloud platform and send the data to it to be stored and organized, and then a connection is established between the IoT cloud platform and the node-red to restructure and arrange the data. Then establish a connection between it and the database so that this data stored in the database is displayed on the interface of the website, through which all data is monitored to follow up on the condition of the animal without the need for the staff to mix daily with all the animals only they will have to monitor this smart system.

Table of Contents

List of Figures

List of tables

Chapter1: Introduction

Worldwide, 90% of all living creatures are on the brink of extinction. This project can help them. Many endangered animals are in danger of extinction. Therefore, the world can lose these animals completely. The world is spending a lot of time and money to save endangered animals. Although extinction is a natural process that would happen with or without humans, research shows that extinctions are happening quicker now than ever before. And, loss of habitat is by far the biggest cause. This is a problem that we need to address.

Everything in nature is connected. If one animal or plant is removed, it upsets the balance of nature and can change the ecosystem completely; and may cause other animals to suffer. For example, bees may seem small and insignificant, but they have a huge role to play in our ecosystem – they are pollinators. This means they are responsible for the reproduction of plants. Without bees, many plant species would go extinct, which would upset the entire food chain.

Possible suggestions to protect endangered animals:

1. Protect wildlife habitats. Habitat loss is one of the biggest causes of extinction.
2. Educate others.
3. Stay away from pesticides and herbicides.
4. Shop ethically.

1.1 Habitat destruction:

A habitat is any natural region where wildlife lives undisturbed e.g., forest, pond, marsh, or desert. Most animals that are endangered have become so, not because they have been killed on purpose but because their habitats are being destroyed.

Throughout the centuries humans have steadily cut down the trees to make room for the human population which has increased enormously particularly during the past 200 years. The trees have been replaced by houses, factories, etc. Hedgerows that were planted by humans as boundaries around fields have, replaced woodlands, providing homes for many animals and plants. However, even though many of these have been destroyed, mostly during the last forty years, this vast loss has affected the wildlife.

People all over the world are working to help save endangered animals from extinction. There are conservation organizations that try to make people aware of the problems facing wild animals. Some of how they are being saved include habitat protection, captive breeding, setting up nature reserves and parks, and using alternative products in place of products from rare animals.

The solutions that have been taken to reduce the destruction of Habitat:

- Governments can help by making international agreements between countries to protect animals (many countries, for example, have agreed to stop hunting the blue whale). There has been an agreement between several countries in June 2010 to protect the rainforests and prevent deforestation through financial backing.
- Scientists are setting up gene banks in which they keep an animal's genetic material (the building blocks of a living thing) in suspended animation. This technique may make it possible in the future to *grow* a new animal of the same species.

This project solves the problem of animal extinction through two stages. The first stage is to monitor everything that happens around the animal through cameras that capture everything happening in the animal’s surroundings from natural phenomena, such as torrents, storms, or the approach of an illegal hunter of the animal. The second stage is to monitor the animal’s health and physical condition, including heart rate, temperature, etc. This is carried out through some healthcare sensors that collect data.

1.2 The proposed project features

This project involves the development of a smart system that protects endangered animals, through a specific collar around the animal's neck. This collar contains a monitoring system to combat poaching, as well as monitoring animals' health.

1. The following are the attributes of measures for animal health:
    - Heart rate measurement.
    - Measurement SpO2.
    - Measuring the temperature of the animal.
2. Monitor the animal's surroundings with a camera.
3. Air quality measurement.
4. Weather measurement.
5. Humidity measurement.
6. animal health report.
7. Notification when a malfunction occurs in the animal's health functions.
8. Classification of the number of animals in the reserve.

1.3 Endanger animals' protection system Advantages

This section explores the main advantages of the project:

1. Full animal control.
2. Monitoring the health status of the animal.
3. Observe the animal's surroundings.
4. Protection from overfishing.
5. Monitor the climatic conditions surrounding the animal.

1.4 IoT Challenges

Here are some of the challenges I encountered in the project:

- **Developing Privacy-Preserving Security Protocols:** Although in literature, there are many anonymization techniques, which have been proposed for providing privacy and security, most of them consume resources, such as energy, memory, etc., considerably. So existing IP-based security solutions cannot be directly deployed in IoT applications.
- **Self-immunity:** Providing self-immunity to the nodes in IoT applications is an important aspect to be considered. Sine, the futuristic IoT applications are moving towards providing greater autonomy for the objects (things), they should also be able to perceive and able to reach the security attacks. Also, an object should be able to protect itself from physical intrusion or tampering.
- **Optimal Energy Utilization:** proper management of energy in IoT devices is another open problem to be addressed. Low-power communications is an important research area since the battery replacement process is costly, especially in large-scalable deployments such as IoT. Though there are energy-efficient solutions at various layers of the ISO model, there will be a need to replace batteries from time to time, which is a huge barrier to the adoption of IoT technology.
- **Efficient Memory Utilization:** Similar to the energy constraint, the nodes used in many of the IoT applications are storage-constrained, efficient utilization of the available storage space (memory) is a challenging task to be addressed. Although there are few buffer management schemes implemented for ad-hoc networks, there are so many issues to be addressed.
- **Scalability:** As the number of objects being interconnected in IoT is huge in number, a scalability issue arises. Address space should be large enough to accommodate all the devices. The data generated by the vast number of sensors is

enormous. Proper information extraction mechanisms must be used to extract useful information from the data, d to store data. Service discovery in IoT is difficult since we need to find out the node that proves ides required service among billions of devices.

- **Finding New Applications of IoT in DTN and CCN:** Since IoT connects physical devices by giving them sensing, computing, and communicating abilities. Incorporating delay tolerance to it can add to the communication part of traditional IoT so that objects can communicate in the scenario of disruption in connectivity. The initial efforts to develop the delay-tolerant version of IoT, called, opportunistic IoT are proposed by Guo et al. while connecting devices to form infrastructure networks by using short-term communication techniques such as Bluetooth or Wi-Fi. In such an opportunistic IoT, data dissemination and sharing among devices is formed based on movement and opportunistic contact among humans. Domingo et al. also outlined applying IoT in underwater networks, Internet of Underwater Things (IoT), which is a network formed by smart underwater objects. Al-Tudjman et al., have discussed a delay-tolerant approach for integrated RFID sensor networks. Although very few researchers looked at this perspective, designing delay-tolerant IoT applications is a promising area of research. Similarly, since, Internet applications are moving towards P2P Content-Centric Network (CCN) approaches, investigation of how IoT could fit into such a scenario is another challenge for the research community. Performance implications of IoT over ICN are discussed.

Conclusion:

This project tries to reduce the extinction of animals and preserve the ecosystem of the earth by monitoring the health of the animal and the environment around them to reduce poaching. This project is dedicated to countries that have wildlife, animal rights organizations, and natural reserves, and we hope that this will happen to preserve animals and our planet earth. Besides, this project aims to reduce between humans and animals because it is an intelligent system that can be managed remotely without the need for direct contact with the animal every time to monitor it.

Chapter2: Theoretical background and Tools

IoT technology is improving life for humans in everything from smart home, smart health, and smart city applications. Humans, however, are not the only ones benefitting from this technology. The almost endless applications of IoT have now been extended to wildlife protection. Conservationists are turning to this technology to track and monitor wildlife in an effort to save the world's most endangered species. Unfortunately, many species need protection.

IoT technology is being used to turn that trend around. Its global connectivity coupled with low-powered, long battery life sensors makes monitoring, tracking, and protecting endangered animals like the rhino easier and more effective.

2.1 IoT solution

An IoT solution is more than an embedded system that has a connectivity feature. Such an IoT solution is a set of devices and sensors that is connected to a cloud platform through a gateway.

The cloud platform provides the infrastructure that is necessary to manage, store, secure, and analyze a large amount of data to extract valuable information and insights from it. [1]

2.2 IoT platform

An IoT platform provides a set of ready-to-use features that greatly speeds up the development of applications for connected devices and provides scalability and cross-

device compatibility. As shown in Figure 1, an IoT platform is an integrated service that provides the capabilities needed to bring physical objects online.

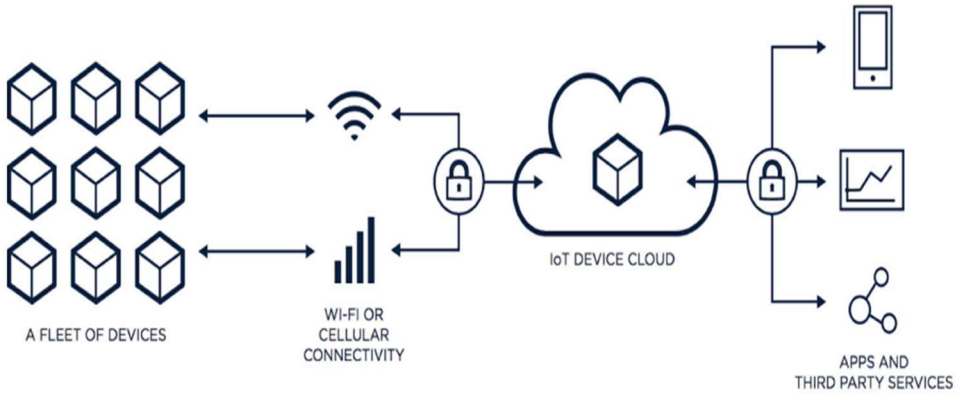


Figure 1 IoT platform

IoT platforms originated as IoT middleware, which functions as a mediator between the hardware and application layers. The primary tasks of this middleware include data collection from devices over different protocols and network topologies, remote device configuration and control, device management, and over-the-air firmware updates [1].

2.2.1 Watson IoT Platform solution:

As shown in Figure 2, the Watson IoT Platform imports device data and transforms that data into meaningful insights, which can optimize processes and guide new product design. [1]

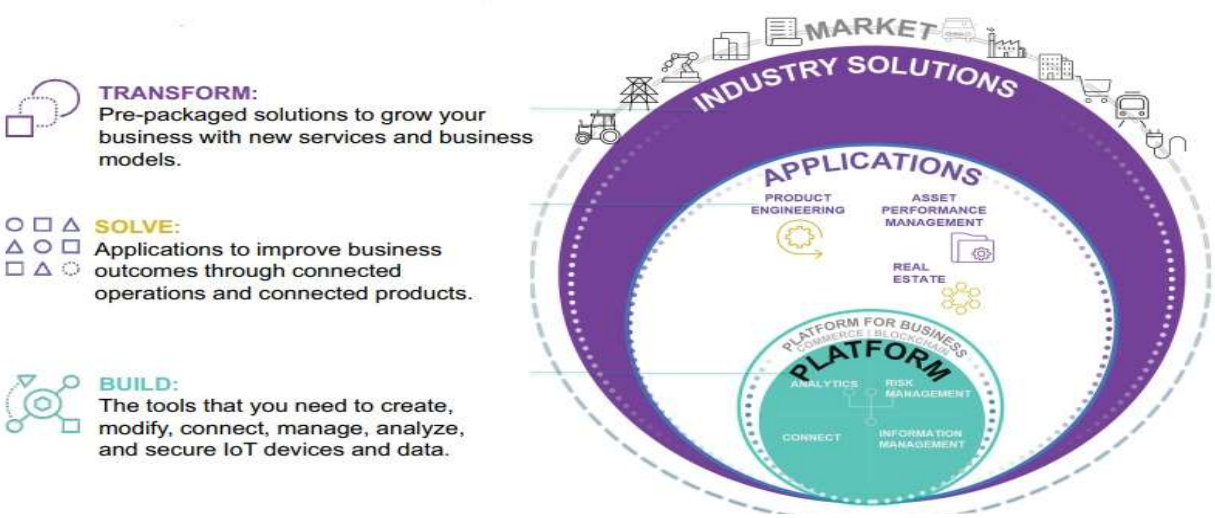


Figure 2 Watson IoT Platform solution

2.2.2 Watson IoT basic reference architecture

Figure 3 shows a basic reference architecture that shows the flow of data from the IoT device to the cloud:

1. You connect your device or gateway to the Watson IoT Platform.
2. The device communicates with the Watson IoT Platform by using the Message Queuing Telemetry Transport (MQTT) protocol.
3. Watson IoT Platform communicates with external applications and services for device management, data visualization, data storage, data analysis, and other useful services that are provided by the cloud platform [1].

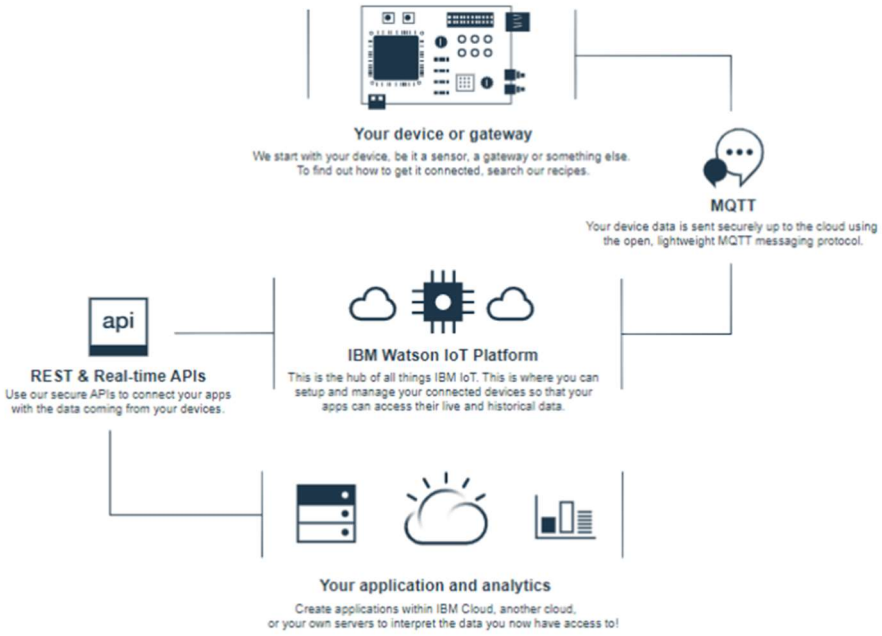


Figure 3 IBM IoT basic reference architecture

As shown in Figure 4, the four Watson IoT Platform quadrants:

1. **Connect:** connect and manage devices, networks, and gateways.
2. **Information Management:** integrating structured and unstructured data from devices, people, and the world around us.

3. **Analytics:** analyze and visualize real-time device data by using the Watson IoT Platform dashboards. Gain insights from data by using real-time data, predictive analytics, and AI by leveraging platform services.
4. **Risk Management:** You ensure that you leverage the correct information from trusted sources, that the correct software runs where you need it, and that your solution is visible end to end [1].

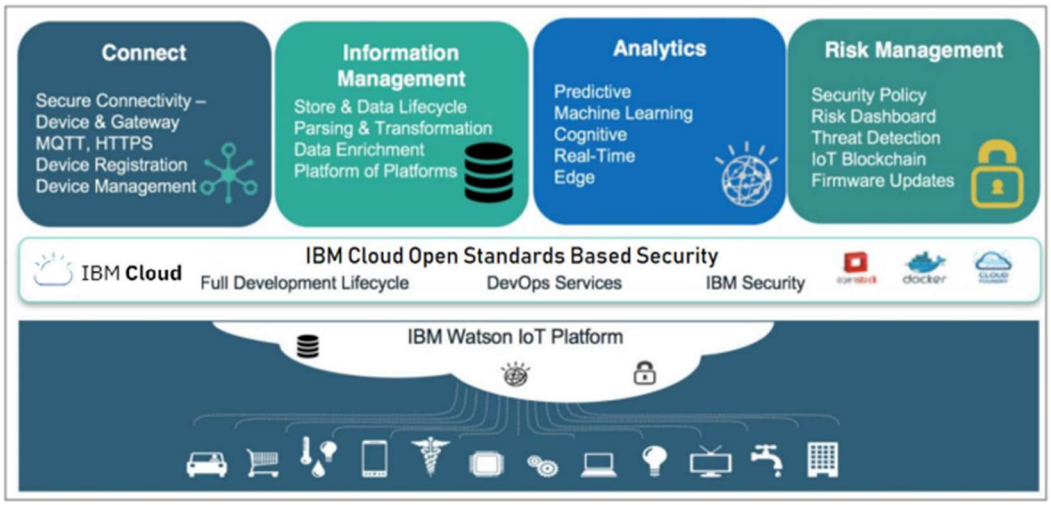


Figure 4 The four Watson IoT Platform quadrants

Techniques for connecting devices to the Watson IoT platform

Figure 5 visualizes the connecting stage in which the devices can be connected directly to the platform or through a gateway, depending on the following use case:

The Sensor Area Network is the network protocol that is used to communicate the IoT sensor devices to the gateway. Examples include Bluetooth or ZigBee.

Events represent certain sensor data value ranges that are measured. For example, an event can be a temperature sensor reading above 30 degrees. In many cases, devices do not have to send the sensor data to the cloud continuously but only when it matters, such as when an event occurs. Usually, the events are sent from the devices to the cloud.

Commands are sent from the Watson IoT Platform to the devices. Commands might be actuation commands that turn off some devices or they might be diagnostic commands so that devices can report malfunctions [1].

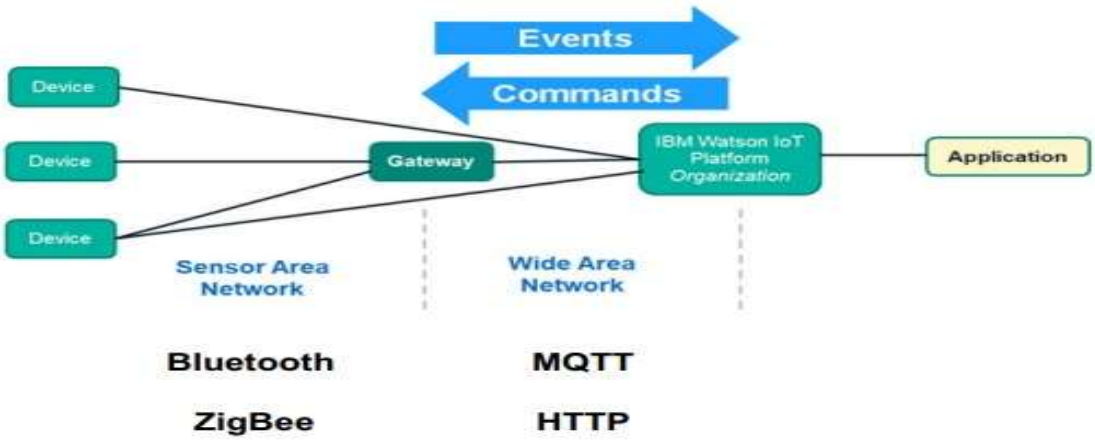


Figure 5 Techniques for connecting devices to Watson IoT platform

2.3 IoT devices

The "thing" in IoT is any device with embedded electronics that can transfer data over a network without any human interaction. The device that can be used in IoT is a computer system that contains the following components:

- Processor
- Memory
- Input/output peripheral devices
- Internet connection device

Examples of the most used embedded system boards for experimental and educational purposes are:

- Arduino boards
- Node Micro Controller Unit (NodeMCU)
- Raspberry Pi boards.[1]

As depicted in Figure 6, An IoT device has four main components [1]:

1. Input sensors to measure the physical environment variables.
2. Output actuators to control the physical environment, such as moving objects and raising the temperature.
3. The network connection device allows the device to share its collected data with the cloud. This device can be part of the IoT board like Raspberry Pi boards or an external add-on board like Arduino boards.
4. The IoT board collects the signals from the sensors, controls the actuators, and sends collected data and the device status to the network.

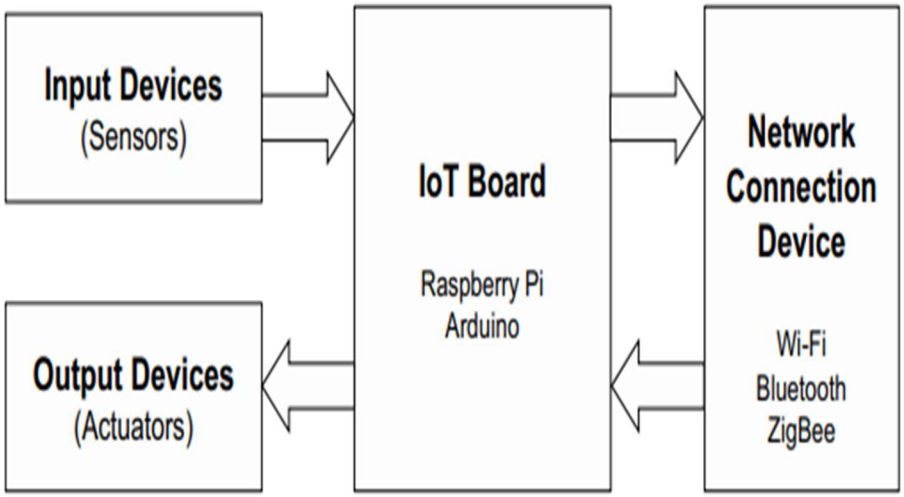


Figure 6 IoT device overview

2.4 Connecting IoT boards to the world

IoT boards can collect different types of signals from the environment by using different types of sensors, and they can make similar changes to the environment by using actuators, as shown in Figure 7 [1].

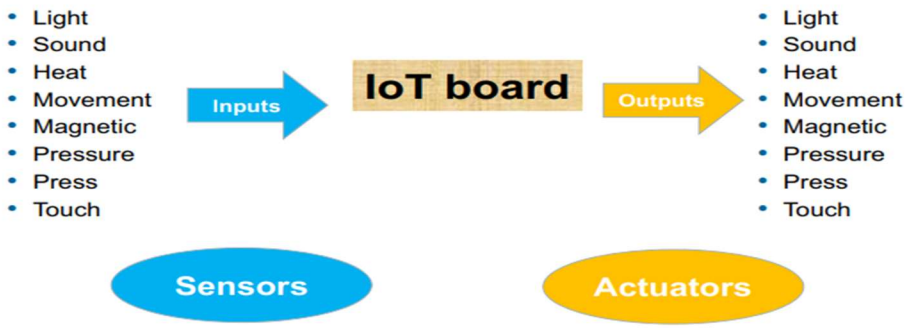


Figure 7 Connecting IoT boards to the world

2.5 IoT Platform communication protocols

Message Queuing Telemetry Transport (MQTT) is a machine-to-machine (M2M) IoT connectivity protocol:

- It follows the publish and subscribe protocol.
- It runs over Transmission Control Protocol/Internet Protocol (TCP/IP).

Hypertext Transfer Protocol (HTTP) is an application protocol that you use to enable communication between physically dispersed systems. It is one of the most used protocols to transfer data over the internet.

Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware.

Constrained Application Protocol (CoAP) is a specialized web transfer protocol that you use with constrained nodes and constrained networks for IoT devices [1].

Message Queuing Telemetry Transport (MQTT) definition:

MQTT is an M2M IoT connectivity protocol, it is a lightweight publish and subscribe messaging transport protocol, and it is useful for connections to remote locations where a small code footprint is required, or network bandwidth is at a premium [1].

2.6 MQTT components

A topic is a place to or from which a device wants to put or retrieve a message, a broker is a server that handles the data transmission between the clients.

Figure 8 MQTT components

As shown in Figure 8, In MQTT, the word topic refers to a UTF-8 string that the broker uses to filter messages for each connected client. The topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator). The client can subscribe to the exact topic or use a wildcard.

For example, a subscription to house//temperature results in all messages being sent to the topic house/living-room/temperature and any topic with an arbitrary value in the place of living-room, for example, house/kitchen/temperature. The plus sign is a single-level wildcard that allows arbitrary values for only one hierarchy. If you must subscribe to more than one level, for example, to the entire subtree, there is also a multilevel wildcard (#). You can use it to subscribe to all underlying hierarchy levels. For example, house/# subscribes to all topics beginning with the house.

An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients. There are many brokers that implement the MQTT protocol. One of the most popular and commonly used is the Mosquito broker [1].

MQTT three components definition

- Message is the data that a device receives "when subscribing" from a topic or sends "when publishing" to a topic.
- Publish is the process that a device uses to send its message to the broker.
- Subscribe is the process that a device uses to retrieve a message from the broker.

In a publish and subscribe system, a device can publish a message on a topic, or it can be subscribed to a topic to receive messages. For example, if Device 1 publishes on a topic and Device 2 is subscribed to the same topic as Device 1 is publishing in, Device 2 receives the message [1].

Publish and subscribe to pattern

As shown in Figure 9, the publish and subscribe pattern is composed of the following components:

- Publisher
- Subscriber
- Broker

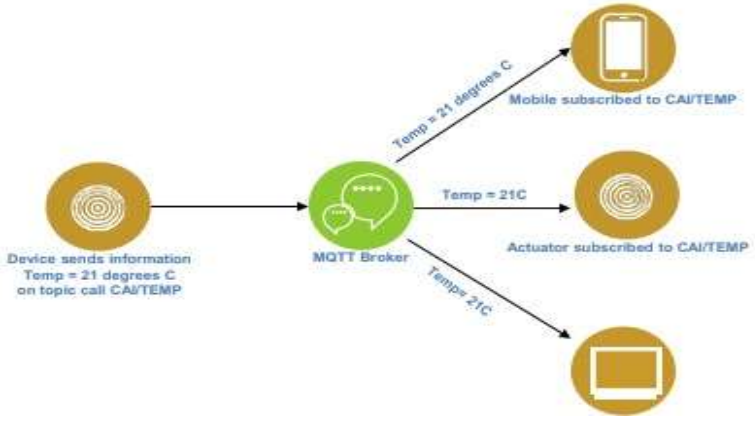


Figure 9 Publish and subscribe pattern

The publish and subscribe pattern (also known as publish/subscribe) provides an alternative to traditional client/server architecture. In the client-server model, a client communicates directly with an endpoint.

The publish/subscribe model decouples the client that sends a message (the publisher) from the client or clients that receive the messages (the subscribers). The publishers and subscribers never contact each other directly. They are not even aware that the other exists. The connection between them is handled by a third component (the broker).

The job of the broker is to filter all incoming messages and distribute them correctly to subscribers. So, let us dive a little deeper into some of the general aspects of publishing/subscribing. Figure 9 shows an example of the publish and subscribe pattern. The publisher sends the message to the broker, and all subscribers are connected to the same broker.

The publisher (in this case, a temperature sensor) sends temperature information to the MQTT broker.

There are three subscribers:

1. Mobile, which is controlled by an operation engineer.
2. An actuator that controls another machine.
3. A web dashboard that represents the control loop.

When the sensor sends its information to a predefined topic (in this case, CAI/TEMP), all subscribers that are subscribed to this topic receive a copy of this message [1].

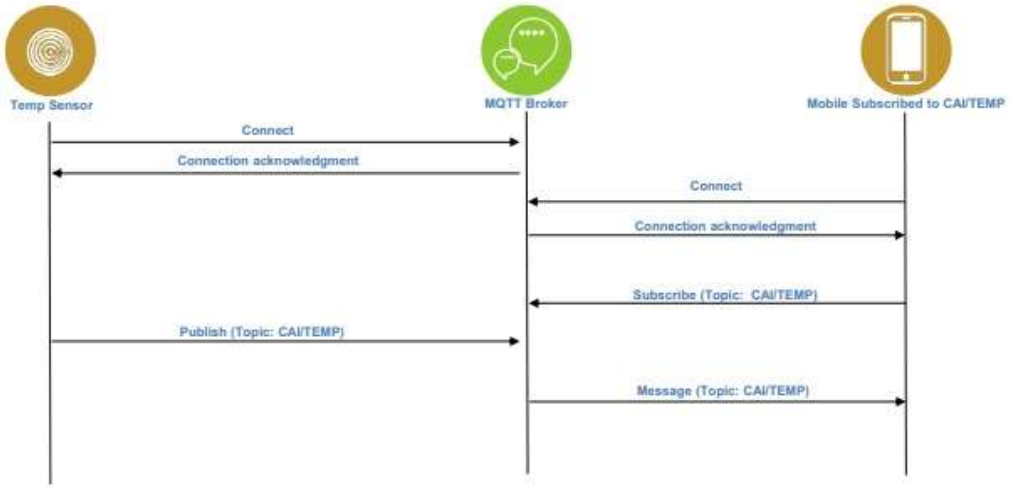


Figure 10 Publish and subscribe pattern (cont.)

Figure 10 shows the Temp Sensor connects to the MQTT broker, The mobile device connects to the same MQTT broker, and the mobile device is subscribed to the topic CAI/TEMP, so when the Temp Sensor publishes any data on the topic CAI/TEMP, the broker sends the message to the mobile device [1].

2.7 Message Queuing Telemetry Transport (MQTT) quality of service

An MQTT client provides three types of QoS for delivering publications to IBM MQ and the MQTT client:

1. At most once.
2. At least once.
3. Exactly once. When an MQTT client sends a request to IBM MQ to create a subscription, the request is sent with the "at least once": Quality of Service (QoS):

- At most once: QoS=0 is the fastest but is less reliable.
- At least once: QoS=1
- Exactly once: QoS=2 is the slowest but is more reliable [1].

The three types of Quality of Service (QoS):

Figure 11 shows the (QoS=0) state, this service level guarantees a best-effort delivery. There is no guarantee of delivery, the recipient does not acknowledge receipt of the message, the message is not stored and retransmitted by the sender, and QoS=0 is often called “fire and forget” and provides the same guarantee as to the underlying TCP protocol [1].

Figure 11 QoS=0

At most once (QoS=0), the message is delivered at most once or it is not delivered at all, its delivery across the network is not acknowledged, the message is not stored, the message might be lost if the client is disconnected or the server fails, and QoS=0 is the fastest mode of transfer. It is sometimes called "fire and forgets".

The MQTT protocol does not require servers to forward publications at QoS=0 to a client. If the client is disconnected when the server receives the publication, the publication might be discarded depending on the server. The telemetry (MQXR) service does not discard messages that are sent [1].

Figure 12 shows the (QoS=0) state, **(QoS=1),** guarantees that a message is delivered at least one time to the receiver, the sender stores the message until it gets a PUBACK packet from the receiver that acknowledges receipt of the message, and it is possible for a message to be sent or delivered multiple times [1].

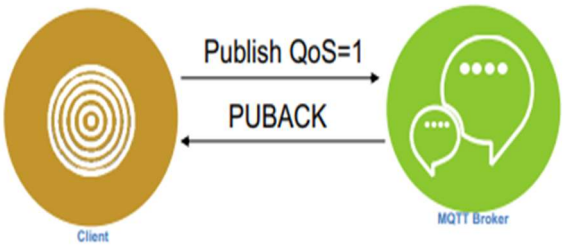


Figure 12 QoS=1

At least once (QoS=1), is the default mode of transfer, the message is always delivered at least once, If the sender does not receive an acknowledgment, the message is sent again with the DUP flag set until an acknowledgment is received, as a result, the receiver can be sent the same message multiple times and might process it multiple times, and the message must be stored locally at the sender and the receiver until it is processed.

The message is deleted from the receiver after it processes the message. If the receiver is a broker, the message is published to its subscribers, If the receiver is a client, the message is delivered to the subscriber application, After the message is deleted, the receiver sends an acknowledgment to the sender, and the message is deleted from the sender after it receives an acknowledgment from the receiver [1].

Figure 13 shows the **(QoS=2)**, is the highest level of service in MQTT, this level guarantees that each message is received only once by the intended recipients, QoS=2 is the safest and slowest QoS level, The guarantee is provided by at least two requests/response flows (a four-part handshake) between the sender and the receiver, and the sender and receiver use the packet identifier of the original PUBLISH message to coordinate the delivery of the message [1].

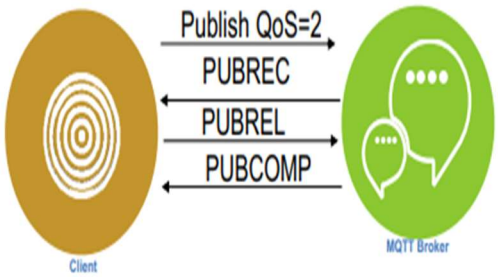


Figure 13 QoS=2

Exactly once (QoS=2), the message is always delivered exactly once, the message must be stored locally at the sender and the receiver until it is processed, QoS=2 is the safest but slowest mode of transfer, it takes at least two pairs of transmissions between the

sender and receiver before the message is deleted from the sender, and the message can be processed at the receiver after the first transmission.

In the first pair of transmissions, the sender transmits the message and gets acknowledgment from the receiver that it stored the message. If the sender does not receive an acknowledgment, the message is sent again with the DUP flag set until an acknowledgment is received.

In the second pair of transmissions, the sender tells the receiver that it can complete processing the PUBREL message. If the sender does not receive an acknowledgment of the PUBREL message, the PUBREL message is sent again until an acknowledgment is received. The sender deletes the message that it saved when it receives the acknowledgment to the PUBREL message.

The receiver can process the message in the first or second phases if it does not reprocess the message. If the receiver is a broker, it publishes the message to subscribers. If the receiver is a client, it delivers the message to the subscriber application. The receiver sends a completion message back to the sender that it finished processing the message [1].

2.8 Tools

Hardware tools:

Arduino Mega, Arduino Uno, NodeMCU, Servo Motor, Camera Sensor, PIR Sensor, Air Quality Sensor, Temperature and Humidity Sensor, Heart Rate, and Oximeter Sensor, and Temperature Sensor.

Software Programming languages and tools:

C++, MQTT Mosquitto broker, MQTT Protocol, IBM Watson IoT Platform, JavaScript, Node.js, Feather.js, Charts.js, Socket.Io, Node-Red, Node Media Server, Express, JWT, CASL, Apis, NoSQL, Cassandra, HTML, CSS, Bootstrap, Axios.

Conclusion

An IoT solution is more than an embedded system that has a connectivity feature, the cloud platform provides the infrastructure that is necessary to manage, store, secure, and analyze a large amount of data to extract valuable information and insights from it. MQTT is an M2M IoT connectivity protocol, it is a lightweight publish and subscribe messaging transport protocol. An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients. An MQTT client provides three types of QoS: at most once, at least once, and exactly once.

Chapter3: Analysis and Design

A system is a general set of parts, steps, or components that are connected to form a more complex whole. Systems analysis is a problem-solving method that involves looking at the wider system, breaking apart the parts, and figuring out how it works to achieve a particular goal. System analysis is used in every situation where something is developed.

There are many benefits why to analyzing a system. These include learning to use systems that somebody else created, planning new systems, and reducing errors when problem-solving. Systems analysis also leads to fewer mistakes when problem-solving. The better to understand the system, the less likely this is to happen. [2]

3.1 Software Requirements Specification

3.1.1 Functional Requirement:

An IoT-enabled system will monitor and reports toxic or explosive gasses and measure gas concentration with the help of an Air Quality Sensor. The system will display the temperature of the animal that is measured by a temperature sensor. Relative humidity becomes an important factor when looking for comfort.

The ratio of moisture in the air to the highest amount of moisture at a particular air temperature is called relative humidity. so IoT-enabled system measures and reports both moisture and air temperature with the help of temperature and humidity sensors. The system will monitor the change in the oxygen rate and heart rate of the target animal with the help of a medical sensor.

The system can display a live stream that detects a particular area around the animal and moves according to the PIR (Passive Infrared) sensor that detects any sudden action with the help of a servo motor. The administrator can print reports of the change in the

state of the animal at a specific time and be displayed them in the form of an understandable illustration (charts and diagrams) to the user.

3.1.2 Non-Functional Requirement:

In this section, we will show the non-functional requirement associated with our project.

Performance

- Appropriate response time, fast web loading, and update times.
- High accuracy in rates to produce useful reports.

Integrity

- Data integrity_ referential integrity in database tables and interfaces.

Usability

- user friendly and that is important to speed up the user response process in case of emergency.

Authentication and authorization

- Login requirements_ access level supported.
- Password requirements –length, special character.
- Manage users

3.2 Hardware Requirements Specification

Hardware Specification:

Table 1 describes the main hardware requirement specification used for our endangered animal protection system.

Table 1 Hardware requirements specification

Component	Model	Specifications
Microcontroller	Arduino Mega 2560 R3	• Microcontroller board based on the ATmega2560. • Flash Memory is 256 KB. • Clock Speed 16 MHz
NodeMCU	V1.0 ESP-12E	• Microcontroller board based on the ESP-8266 32-bit. • Flash Memory is 4 MB. • Clock Speed 80 MHz • Integrated HTTP, MQTT, TCP, FTP, etc.
Servo Motor	MG90S	• Rotational degree 180°. • Operating speed 0.12 sec/60.

Sensors Specification:

Table 2 describes the specification of the sensors used for our endangered animal protection system.

Table 2 Sensors specification

Component	Model	Specifications
Camera Sensor	ESP32 CAM	• Supports 2 Megapixels video recording with face detection feature. • Supports WIFI and Bluetooth. • Supports built-in flash lamp. • Supports microSD card slot. • Image transfer rate of 15 to 60 fps.
Motion Detection Sensor	HC-SR501	• Sensing range 7m. • Angle sensor 100°. • Lens size 23mm.
Heart Rate, Oxygen Ratio Sensor	MAX30100	• LED peak wavelength 660nm/880nm. • Board reserved assembly hole size: 0.5 × 8.5 mm.
Animal Temperature Sensor	DS18B20	• Sensing temperature: -55°C ~ +125°C.
Weather, Humidity Sensor	DHT11	• Temperature Range: 0°C to 50°C. • Humidity Range: 20% to 90%.
Air Quality Sensor	MQ135	• Detect NH3, NOx, Alcohol, Benzene, smoke, CO2, etc.

3.3 Requirement Analysis

In this section, we use the modeling system “UML diagrams”, based on requirements analysis, to describe the functions and processes that the system must perform.

Unified Modeling Language (UML) is a standardized graphical display format for the visualization, specification, design, and documentation of (software) systems. And it was created by the Object Management Group. One of the purposes of UML is to provide the development community with a stable and common design language that could be used to develop and build computer applications.[3]

In this section, we will introduce the UML diagrams of the project.

3.3.1 Use a case Diagram

A use case diagram is a graphic depiction of the interactions among the elements of a system, the key concept of use case modeling is that it helps us design a system from the end user's perspective.[4]

It provides all processes on the website.

User Goals:

1. Monitor Animals
2. Alert Urgent Case
3. Print Reports
4. Add Notes

Admin Goals:

1. Manage Animals
2. Manage Animals Category
3. Manage Users
4. Read Notifications

Figure 14 Represents the use case diagram design of a system from the end user's perspective of the system of the endangered animal protection system.

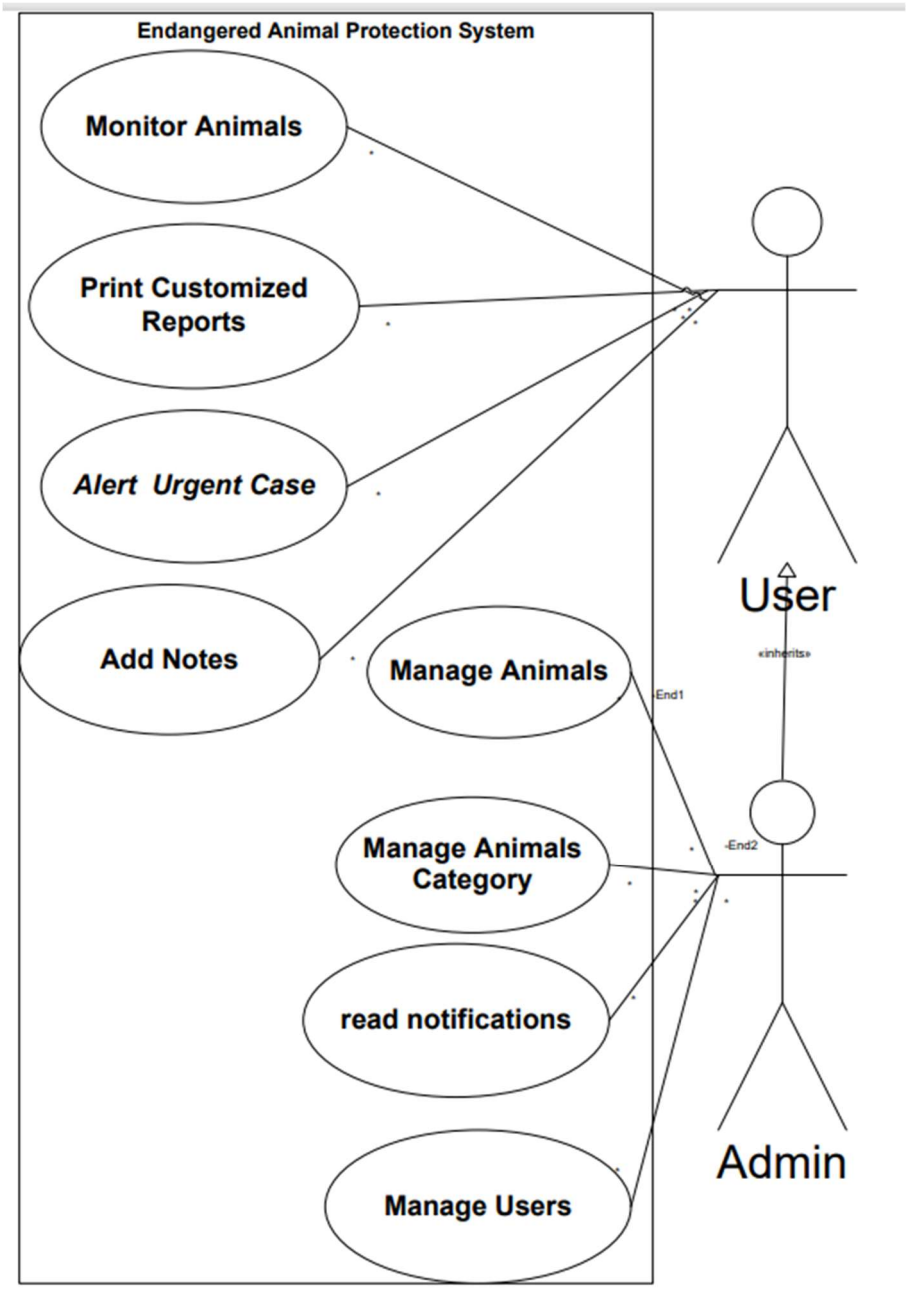


Figure 14 Use case of the endangered animal protection system

3.3.2 Sequence Diagram

The sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.[5]

Figure 15 Represents the sequence diagram for users of the endangered animal protection system.

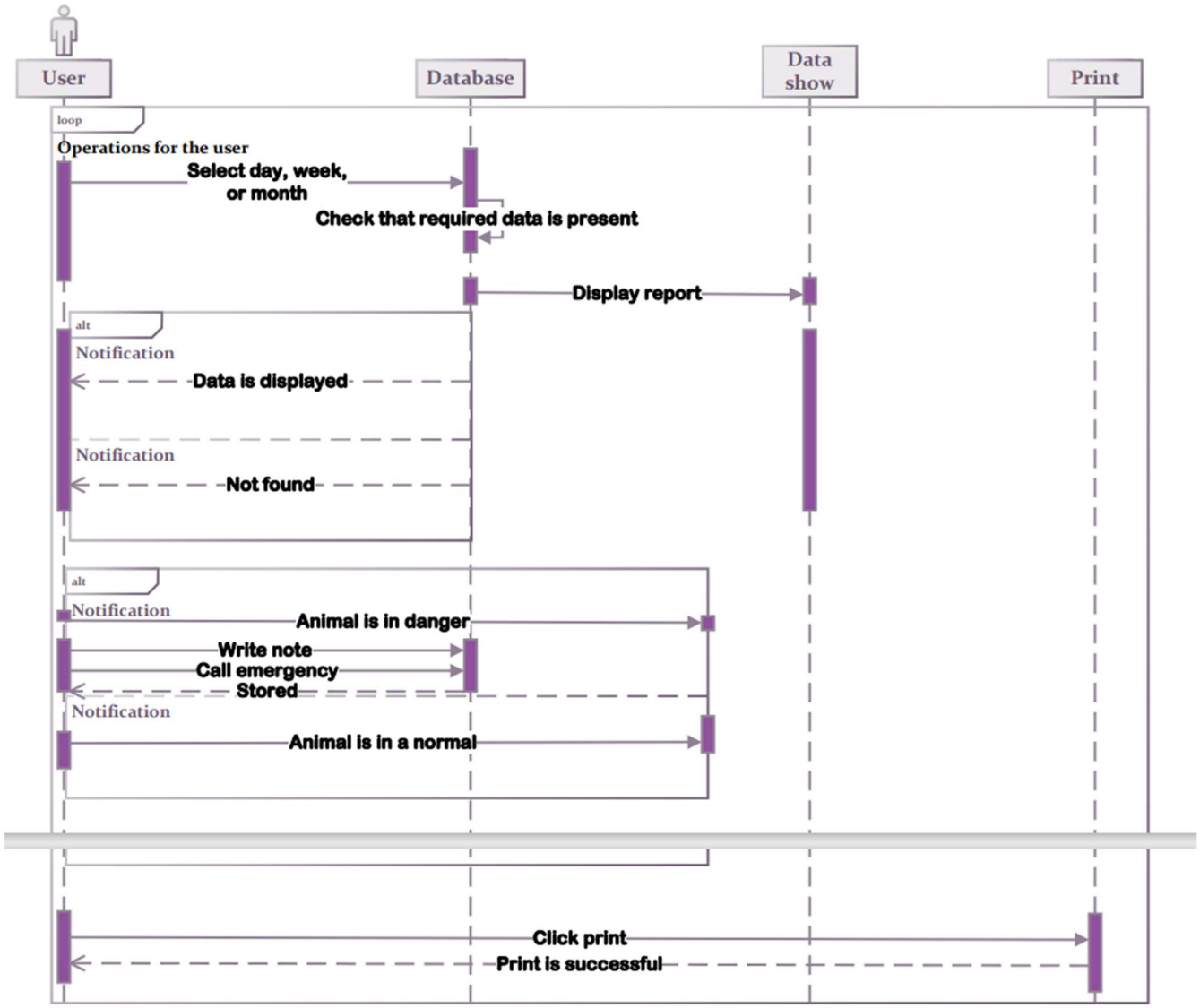


Figure 15 Sequence diagram for users of the endangered animal protection system

Figure 16 Represents the sequence diagram for sensors of the endangered animal protection system.

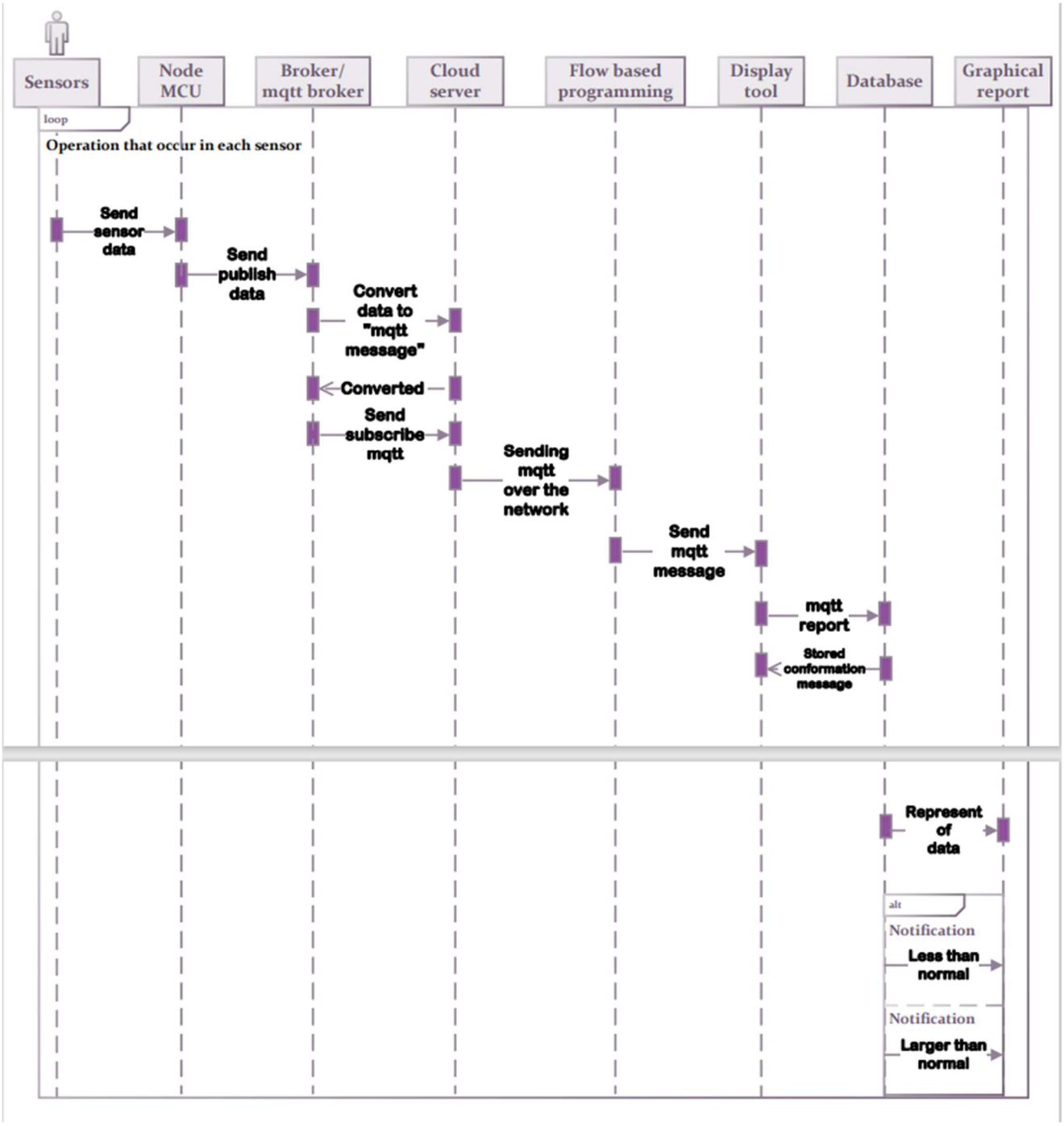


Figure 16 Sequence diagram for sensors of the endangered animal protection system

Figure 17 Represents the sequence diagram for the camera sensor of the endangered animal protection system.

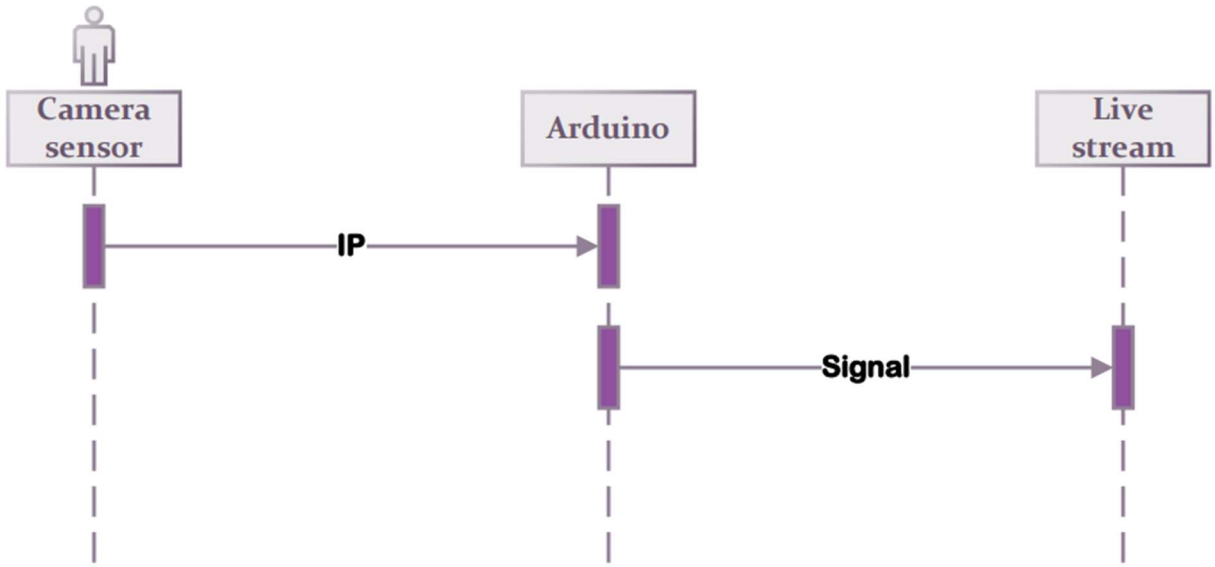


Figure 17 Sequence diagram for camera sensor of the endangered animal protection system

3.3.3 Activity Diagram

An activity diagram is a chart that captures the dynamic behavior of the system and shows the message flow from one activity to another.[6]

Figure 18 Represents the activity diagram for users of the endangered animal protection system.

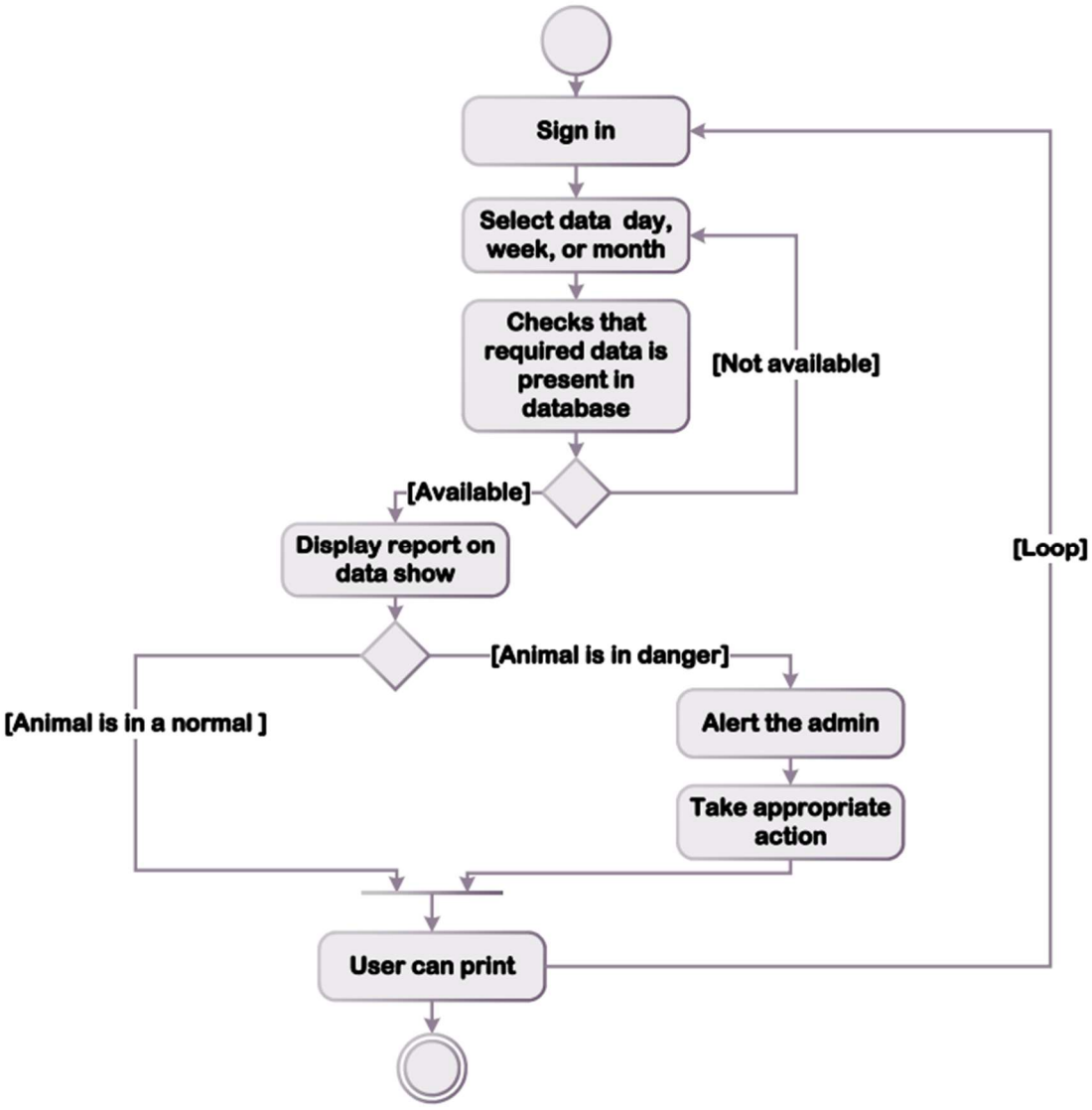


Figure 18 Activity diagram for user of the endangered animal protection system

Figure 19 Represents the activity diagram for sensors and shows animal data of the endangered animal protection system.

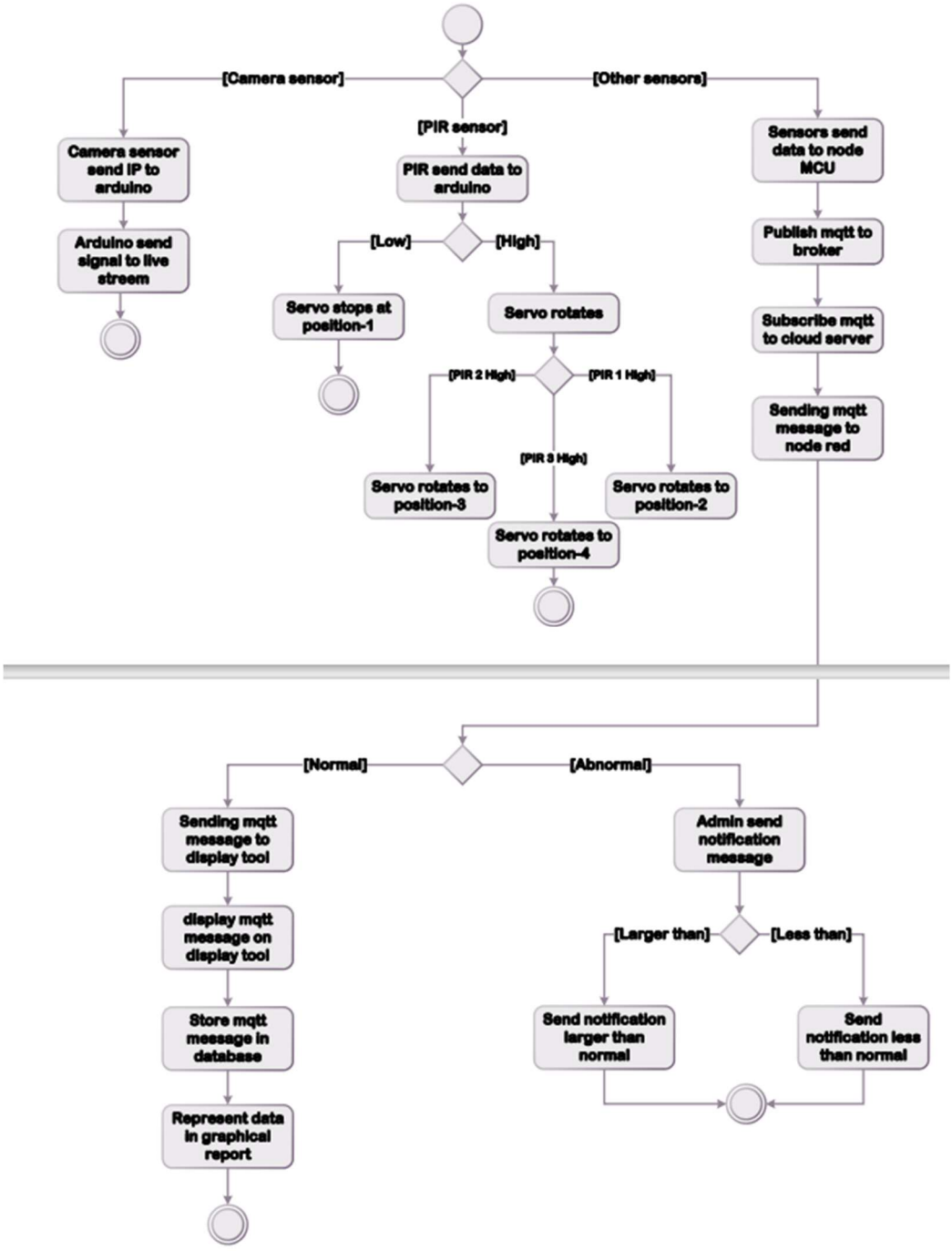


Figure 19 Activity diagram for sensors of the endangered animal protection system

3.3.4 Deployment Diagram

A deployment diagram displays the physical architecture of the system's hardware and software, that is, the software, processes, and devices that make up the system. The physical resources (computers, terminals, and various devices) are described as nodes (junctions) connected by communication lines. This diagram too is not relevant in the analysis and design phases, but it is especially useful in building a distributed system, where it is possible to divide the run units of the application among several interconnected servers. [7], Figure 20 Represents the deployment diagram and physical architecture of the system hardware and software of the endangered animal protection system.

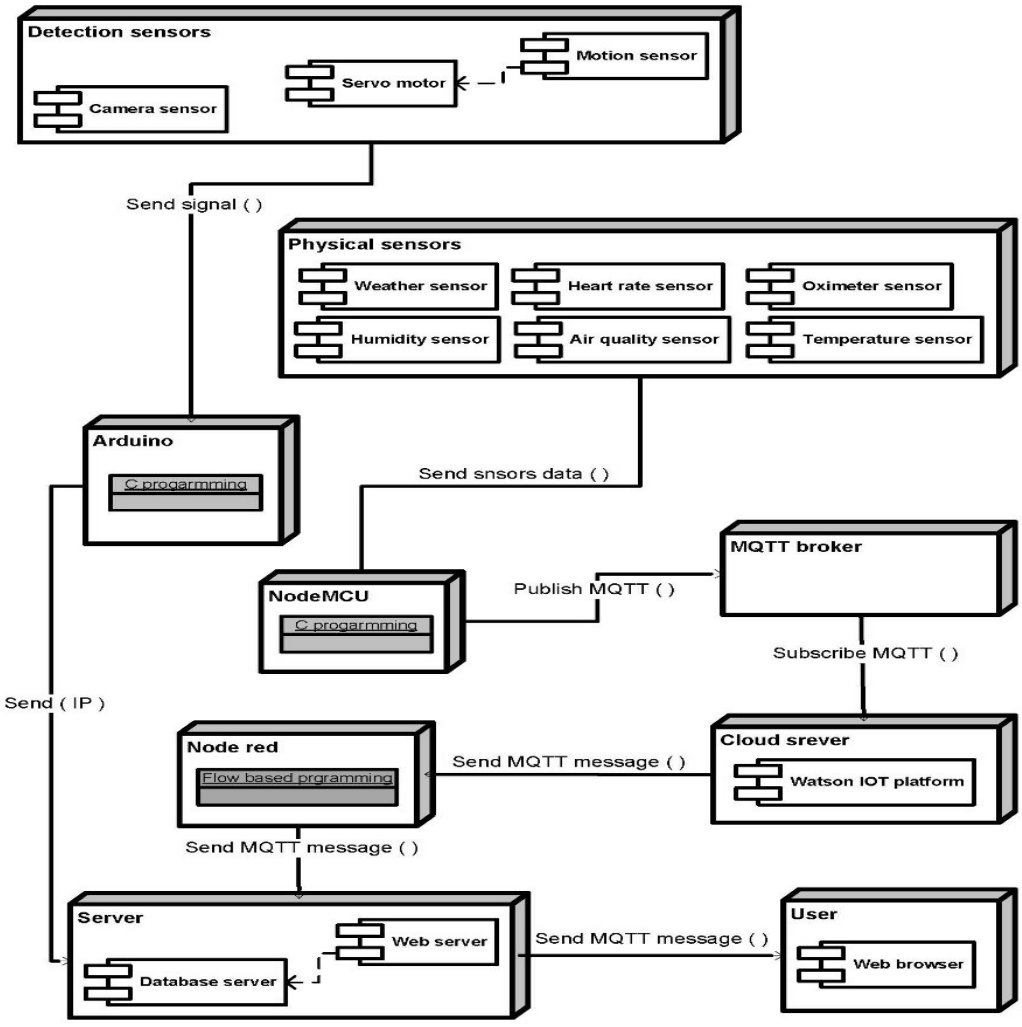


Figure 20 Deployment diagram of the endangered animal protection system

3.3.5 Communication Diagram:

A communication diagram is an extension of an object a diagram that shows the objects along with the messages that travel from one to another. In addition to the associations among objects, and communication the diagram shows the messages the objects send each other.

This diagram describes how the components communicate with each other.[8]

Figure 21 Represents the communication diagram objects along with the messages that travel from one to another in the endangered animal protection system.

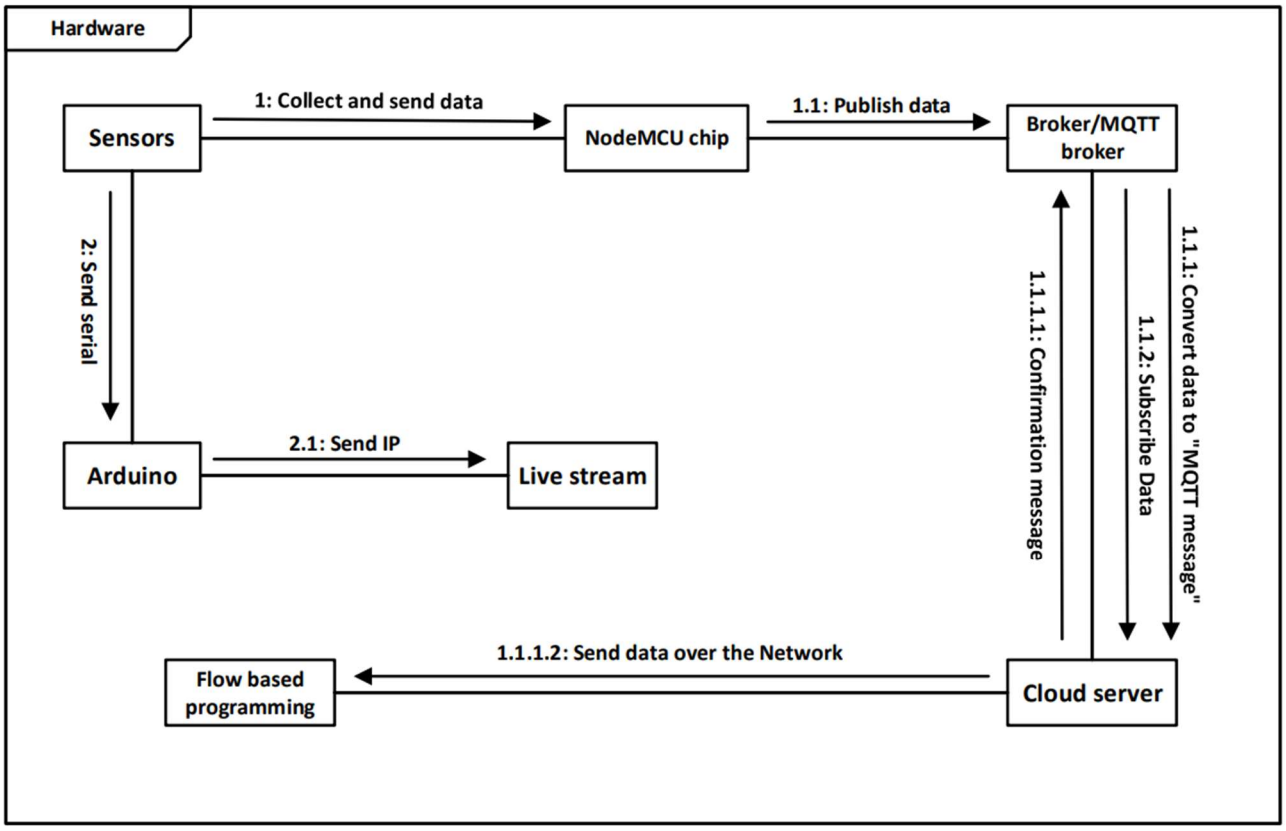


Figure 21 Communication diagram of the endangered animal protection system

3.4 The Design Phase

System Design: System design is the process of defining elements of a system like modules, architecture, components, and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing, and designing systems that satisfy the specific needs and requirements of the application.[9]

Physical Design: Design and implementation of a chip that takes physical placement, routing, and artifacts of those into consideration. Physical design is the process of turning a design into manufacturable geometries. It comprises several steps, including floor planning, placement, clock tree synthesis, and routing.

Floor planning is the first major step. It involves identifying which structures should be placed near others, taking into account area restrictions, speed, and the various constraints required by components.[10]

3.4.1 Logical Design

The logical design represents the data flow, inputs, and outputs of the system. Example: ER Diagram (Entity Relationship Diagrams).

Figure 22 ER Diagram (Entity Relationship Diagram) Represents the data flow, inputs, and outputs of the system of the endangered animal protection system.

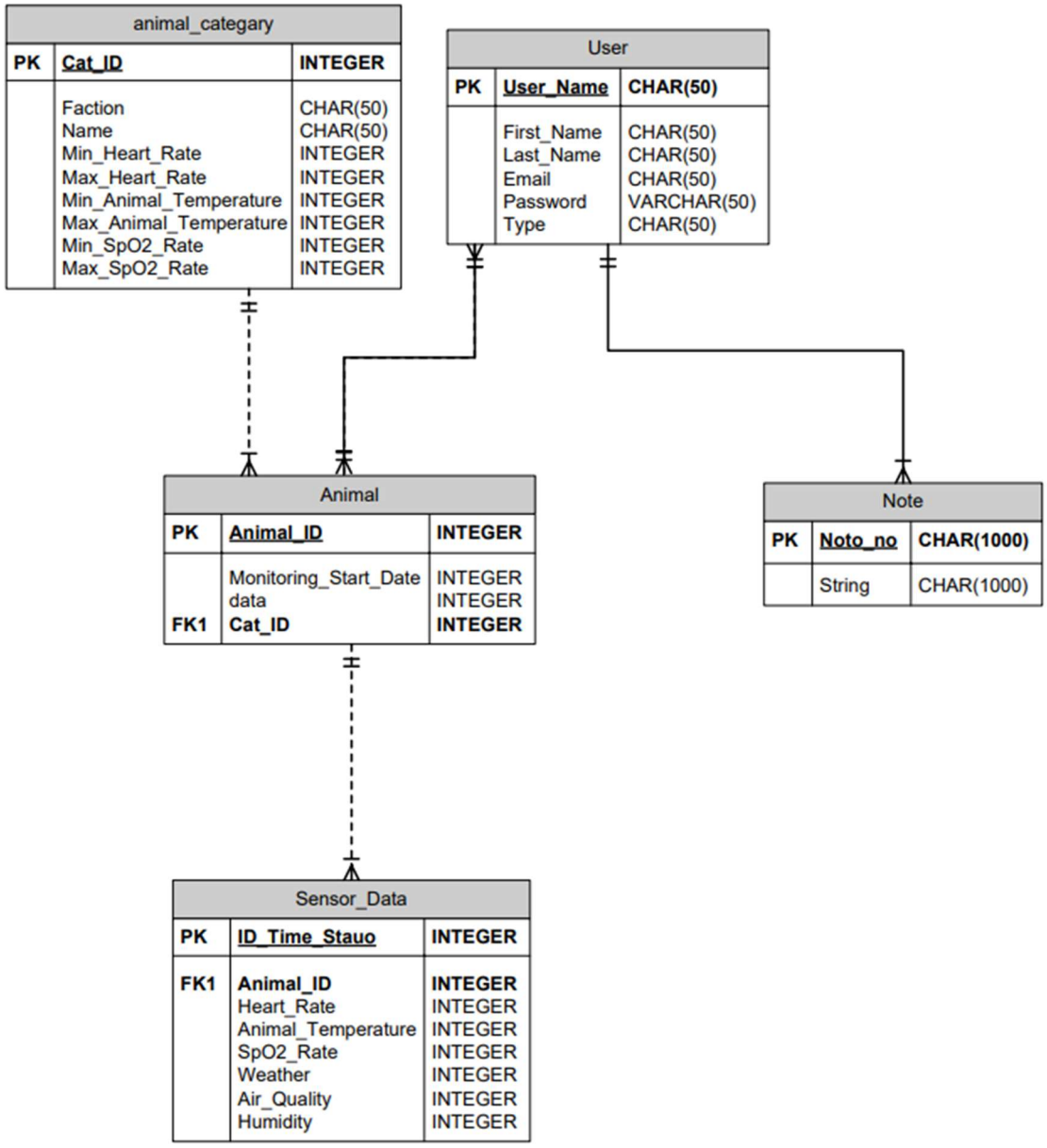


Figure 22 ER Diagram (Entity Relationship Diagram of the endangered animal protection system

3.5 Database Schema:

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them.[11]

Figure 23 Database scheme represents the logical view of the entire database of the system of the endangered animal protection system.

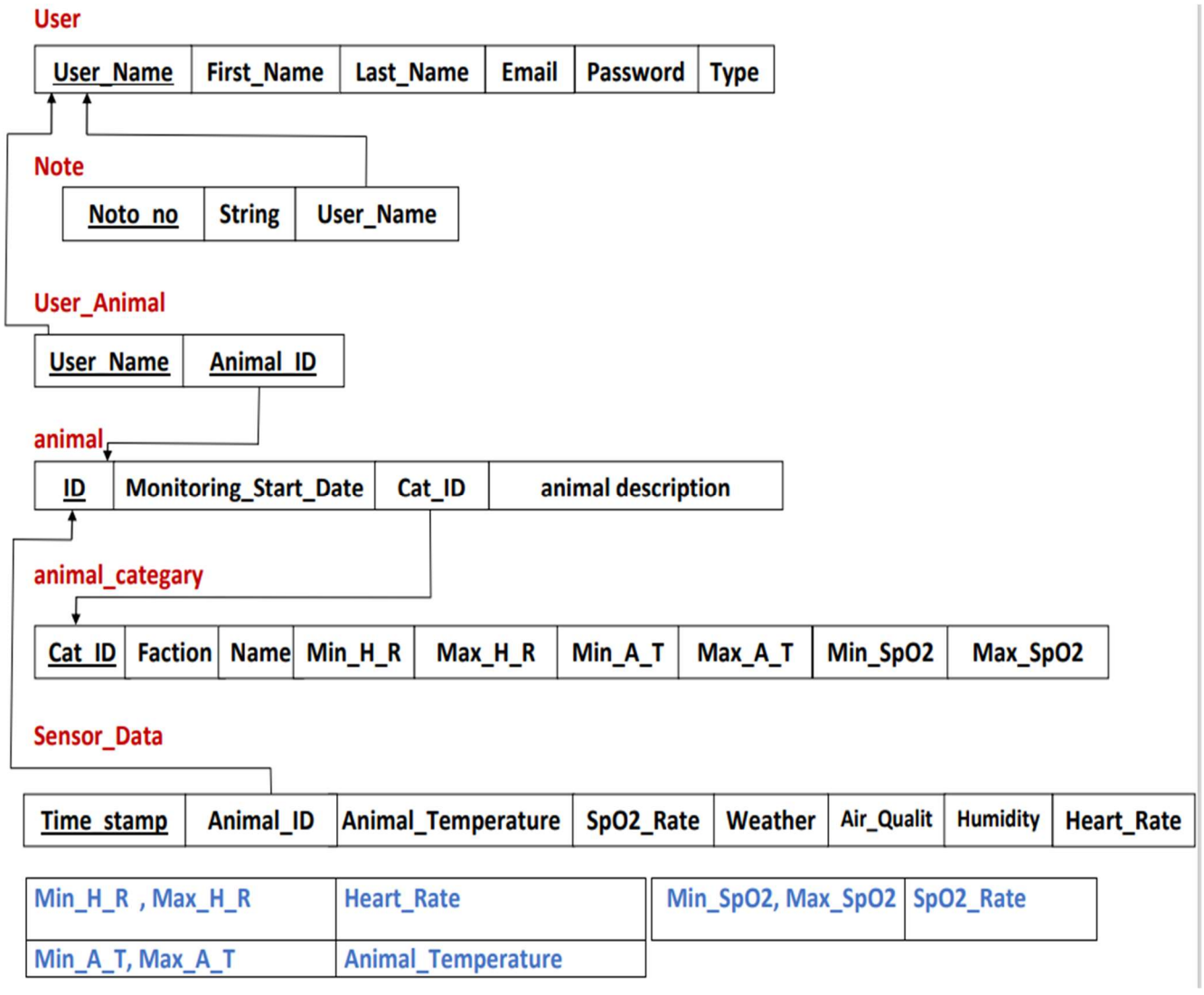


Figure 23 Database scheme of the endangered animal protection system

Because of the runtime data coming from the sensor, we faced a problem because of the large data, which we could not store in a regular Database table. We had to use something different that could handle the size of the large data and did not fall or stop, so the solution was to use Apache Cassandra.

3.5.1 What is Apache Cassandra?

Apache Cassandra is an open-source, distributed, NoSQL database. It presents a partitioned wide column storage model with eventually consistent semantics.

Cassandra is a free and open-source, distributed, wide-column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers support for clusters spanning multiple datacenters, with asynchronous replication allowing low latency operations for all clients. Cassandra was designed to implement a combination of Amazon's Dynamo distributed storage and replication techniques combined with Google's Bigtable data and storage engine model.[12]

Facebook developers named their database after the Trojan mythological prophet Cassandra, with classical allusions to a curse on an oracle.

Apache Cassandra was initially designed at Facebook using a staged event-driven architecture (SEDA) to implement a combination of Amazon's Dynamo distributed storage and replication techniques and Google's Bigtable data and storage engine model. Dynamo and Bigtable were both developed to meet emerging requirements for scalable, reliable, and highly available storage systems, but each had areas that could be improved. Cassandra was designed as a best-in-class combination of both systems to meet emerging largescale, both in data footprint and query volume, and storage requirements. As applications began to require full global replication and always available low-latency reads and write, it became imperative to design a new kind of database model as the

relational database systems of the time struggled to meet the new requirements of global-scale applications.

3.5.2 History of Apache Cassandra:

Avinashi Lakshman, one of the authors of Amazon's Dynamo, and Prashant Malik initially developed Cassandra at Facebook to power the Facebook inbox search feature. Facebook released Cassandra as an open-source project on Google code in July 2008. In March 2009 it became an Apache Incubator project. on February 17, 2010, it graduated to a top-level project.[13]

3.5.3 Systems like Cassandra are designed for these challenges and seek the following design objectives:

- Full multi-master database replication.
- Global availability at low latency.
- Scaling out commodity hardware.
- Linear throughput increases with each additional processor.
- Online load balancing and cluster growth.
- Partitioned key-oriented queries.
- Flexible schema.

3.5.4 Apache Cassandra Features:

Cassandra provides the Cassandra Query Language (CQL), an SQL-like language, to create and update database schema and access data.[12] CQL allows users to organize data within a cluster of Cassandra nodes using:

- **Keyspace**: Defines how a dataset is replicated, per data center. Replication is the number of copies saved per cluster. Keyspace contains tables.

- **Table**: Defines the typed schema for a collection of partitions. Tables contain partitions, which contain rows, which contain columns. Cassandra tables can flexibly add new columns to tables with zero downtime.
- **Partition**: Defines the mandatory part of the primary key all rows in Cassandra must have to identify the node in a cluster where the row is stored. All performant queries supply the partition key in the query.
- **Row**: Contains a collection of columns identified by a unique primary key made up of the partition key and optionally additional clustering keys.
- **Column**: A single datum with a type that belongs to a row.

CQL supports numerous advanced features over a partitioned dataset such as:

- Single partition lightweight transactions with atomic compare and set semantics.
- User-defined types, functions, and aggregates.
- Collection types including sets, maps, and lists.
- Local secondary indices.
- (Experimental) materialized views.

3.5.5 Apache Cassandra problems

However, despite these advantages, it has some disadvantages compared to normal databases.[12]

Cassandra explicitly chooses not to implement operations that require cross partition coordination as they are typically slow and hard to provide highly available global semantics. For example, Cassandra does not support:

- Cross partition transactions.
- Distributed joins.
- Foreign keys or referential integrity.

3.5.6 Cassandra Query Language

Cassandra introduced the Cassandra Query Language (CQL). CQL is a simple interface for accessing Cassandra, as an alternative to the traditional Structured Query Language (SQL). CQL adds an abstraction layer that hides implementation details of this structure and provides native syntaxes for collections and other common encodings. Language drivers are available for Java (JDBC), Python (DBAPI2), Node.JS (Datastax), Go (gocql), and C++.

The keyspace in Cassandra is a namespace that defines data replication across nodes. Therefore, replication is defined at the keyspace level. Below is an example of keyspace creation, including a column family in CQL.[14]

Figure 24 Database code represents an example of the logical view of the entire database of the system of the endangered animal protection system.

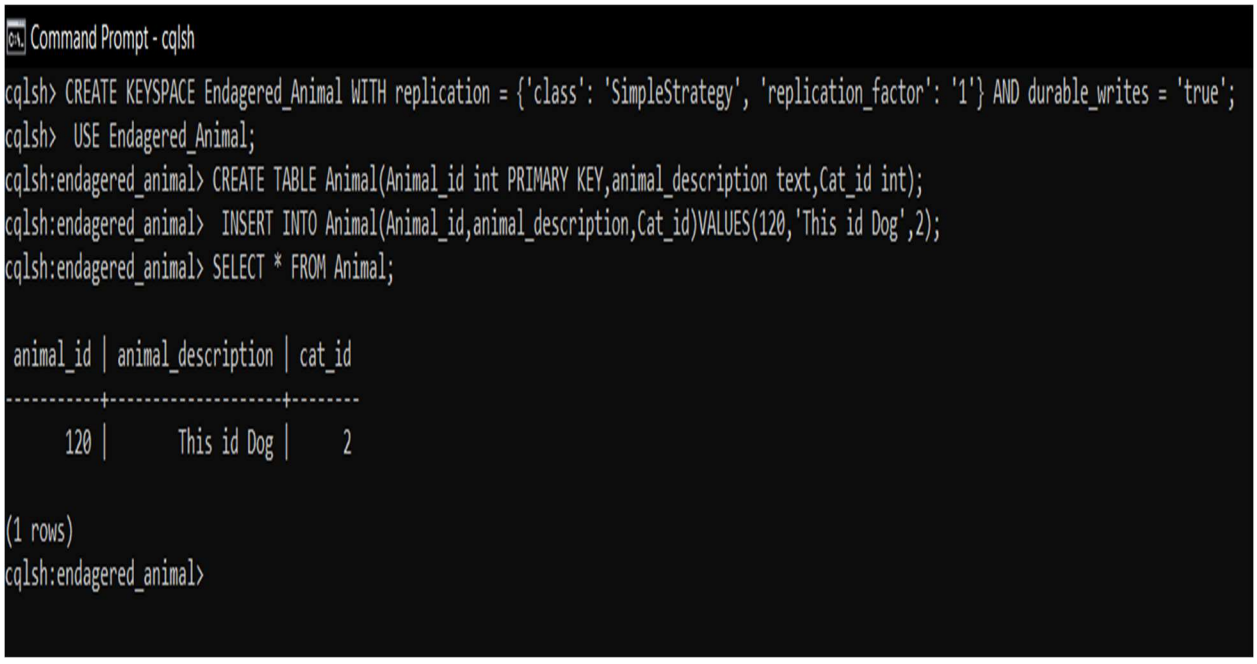


Figure 24 EX of CQL code

Conclusion:

In this chapter, we reviewed the concept of system analysis, we reviewed requirement specification which includes software requirements specification, hardware requirement specification, requirement analysis, and design phase.

In the software requirement specification, we talked about the functional requirement and the non-functional requirement, their performance, and their uses. And the hardware requirement specification shows the models used and describes each one of them. And mentioned the sensors that were used in the project by their names and the models and their description.

In part of the requirement analysis, we mentioned the concept of UML (Unified Modelling Language) and show diagrams the processing of each diagram by presenting each diagram separately.

Finally, showing the ER diagram (Entity Relationship Diagram) and database scheme concepts.

Chapter4: Implementation of Hardware and Software

4.1 Implementation of Hardware

4.1.1 Hardware Sensors Mechanism:

A sensor is a device that produces an output signal for the purpose of a physical phenomenon. In this section, we will explain the mechanism of our sensors.

4.1.1.1 DHT11 Humidity & Temperature Sensor:

DHT11 sensor shown in Figure 25 consists of a capacitive humidity sensing element and a thermistor for sensing temperature, the humidity sensing capacitor has two electrodes with a moisture-holding substrate as a dielectric between them, change in the capacitance value occurs with the change in humidity levels. The IC measure, process these changed resistance values and changes them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with an increase in temperature. To get a larger resistance value even for the smallest temperature change, this sensor is usually made up of semiconductor ceramics or polymers.

The temperature range of DHT11 is from 0 to 50 degrees Celsius with a 2-degree accuracy. The humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz.i.e., it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA. [15]

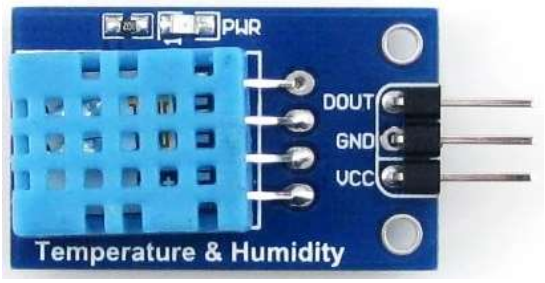


Figure 25 DHT11 Humidity & Temperature Sensor

4.1.1.2 DS18B20 Temperature Sensor Module

DS18B20 is a temperature sensor that can measure temperature from -55°C to +125°C with an accuracy of +- 5%. It follows 1 wire protocol which has revolutionized the digital world, Data received from the single wire is in the ranges of 9-bit to 12-bit.

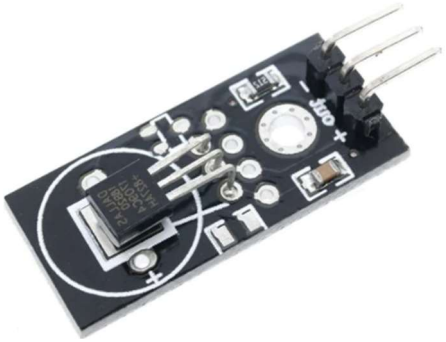


Figure 26 DS18B20 Temperature Sensor Module

Figure 26 shows the 1-wire protocol is an advanced level protocol and each DS18B20 is equipped with a serial code of 64-bit which helps in controlling multiple sensors via a single pin of the microcontroller.

It works on the principle of direct conversion of temperature into a digital value, its main feature is to change its bit numbers according to change in temperature. Like, it changes bits in 9. 10, 11, and 12 bits as temperature changes in values 0.5 °C. 0.25°C, 1.25 and, 0.0625°C respectively, its default bits value is 12 but it changes values according to temperature Change. [16]

4.1.1.3 Max30100 pulse oximetry and heart-rate monitor Sensor

The MAX30100, or any optical pulse oximeter and heart-rate sensor for that matter, consists of a pair of high-intensity LEDs as shown in Figure 27 (RED and IR, both of different wavelengths) and a photodetector. The wavelengths of these LEDs are 660nm and 880nm, respectively.

Figure 27 Max30100 pulse oximetry and heart-rate monitor Sensor

The MAX30100 works by shining both lights onto the finger or earlobe (or essentially anywhere where the skin isn't too thick, so both lights can easily penetrate the tissue) and measuring the amount of reflected light using a

photodetector. This method of pulse detection through light is called Photoplethysmogram, As shown in Figure 28.

The working of MAX30100 can be divided into two parts:

Heart Rate Measurement and Pulse Oximetry (measuring the oxygen level of the blood).

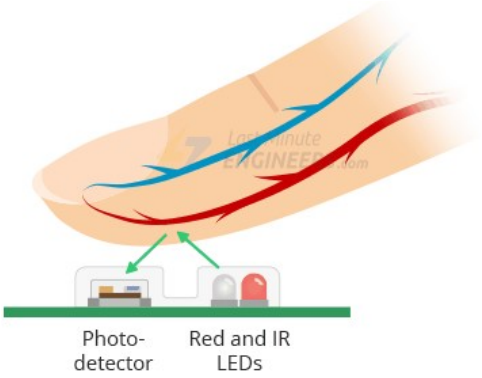


Figure 28 Shining both lights and measuring by the photodetector

4.1.1.3.1 Heart Rate Measurement

The oxygenated hemoglobin (HbO2) in the arterial blood has the characteristic of absorbing IR light. The redder the blood (the higher the hemoglobin), the more IR light is absorbed. As the blood is pumped through the finger with each heartbeat, the amount of reflected light changes, creating a changing waveform at the output of the photodetector. As you continue to shine light and take photodetector readings, you quickly start to get a heart-beat (HR) pulse reading.

4.1.1.3.2 Pulse Oximetry

Pulse oximetry is based on the principle that the amount of RED and IR light absorbed varies depending on the amount of oxygen in your blood.

Deoxygenated blood absorbs more RED light (660nm), while oxygenated blood absorbs more IR light (880nm). By measuring the ratio of IR and RED light received by the photodetector, the oxygen level (SpO2) in the blood is calculated. [17]

4.1.1.4 MQ-135 Air Quality and Hazardous Gas Sensor

The gas sensor module consists of a steel exoskeleton under which a sensing element is housed, As the sensor is shown in Figure 30. This sensing element is subjected to the current through connecting leads. This current is known as the heating current through it, the gases coming close to the sensing element get ionized and are absorbed by the sensing element. This changes the resistance of the sensing element which alters the value of the current going out of it.

When the target gas exists, then the sensor's conductivity increases more increasing more along with the gas concentration rising levels.

To get Gas Concentration from MQ-135 follow the next sensitivity curve:

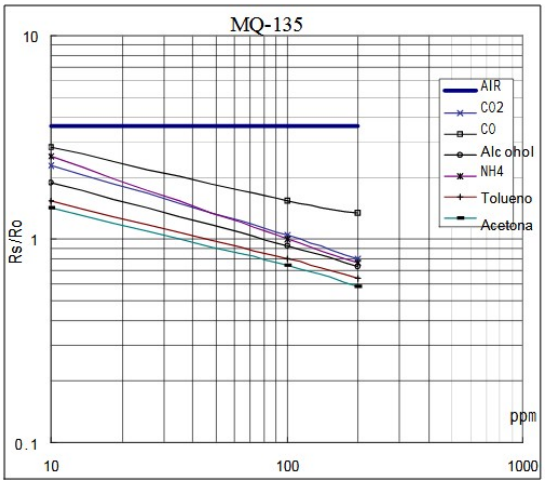


Figure 29 Sensitivity curve to MQ-135

The graph is a log-log graph and shows plots for multiple gases (Figure 29). The x-axis is the detected concentration of the gas in parts per million (PPM) while the y-axis is the RS/R0 ratio – the ratio of the sensor resistance in clean air over the resistance of the sensor in various gases. So, we can use this equation:[18]

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

for getting the concentration in PPM for every gas.

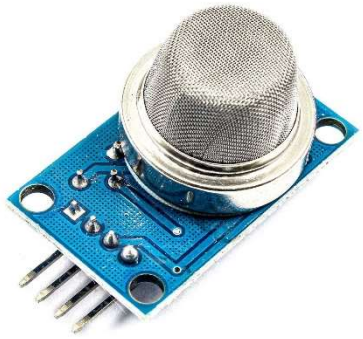


Figure 30 MQ-135 Air Quality and Hazardous Gas Sensor

The air quality sensor detects ammonia, nitrogen oxide, smoke, CO2, and other harmful gases. The air quality sensor has a small potentiometer that permits the adjustment of the load resistance of the sensor circuit. The operating voltage of this gas sensor is from 2.5V to 5.0V. [19]

4.1.1.5 PIR Motion Sensor

PIR is made of a pyroelectric sensor, which can detect different levels of infrared radiation, the detector itself does not emit any energy but passively receives it, and it detects infrared radiation from the environment. Once there is infrared radiation from the human body particle with temperature, focusing on the optical system causes the pyroelectric device to generate a sudden electrical signal.

Figure 31 PIR Motion Sensor

To lengthen the detection distance of the detector (shown in Figure 31), an optical system must be added to collect the infrared radiation. Usually, plastic optical reflection system or plastic Fresnel lens is used as a focusing system for infrared radiation. Indoor passive infrared: Detection distances range from 25 cm to 20 m. [20]

4.1.1.6 Esp32 Cam

The ESP32-CAM is a development board with an ESP32 camera, a microSD card slot for storing images taken with the camera or to store files, and several GPIOs to connect peripherals. The ESP32-CAM board has two pins of power (either 3.3 V or 5 V), The GPIO 1 and GPIO 3 act as the serial pins and are required for uploading the code. The function of the GPIO 0 pin is to identify whether the ESP32 is in flashing mode or not. Thus, the role of the GPIO 0 pin is very important. (Figure 32). [21]

Figure 32 Esp23 Cam

4.1.1.7 Servo sg90 Motor

It consists of three parts: Controlled device, Output sensor, and Feedback system, it is a closed-loop system where it uses a positive feedback system to control motion and the final position of the shaft. Here the device is controlled by a feedback signal generated by comparing the output signal and reference input signal.

Figure 33 Servo sg90 motor

Here reference input signal is compared to the reference output signal and the third signal is produced by the feedback system. And this third signal acts as an input signal to control the device. This signal is present as long as the feedback signal is generated or there is a difference between the reference input signal and the reference output signal. So the main task of servomechanism is to maintain the output of a system at the desired value in presence of noises; the Servo motor is shown in Figure 33. [22]

4.1.1.8 Node Microcontroller Unit (NodeMCU)

NodeMCU is an open-source IoT platform. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware that is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. (Figure 34)

Figure 34 NodeMCU

Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 Wi-Fi SoC, popularly called the "ESP8266 Core for the Arduino IDE", this has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs. [23]

4.1.1.9 Arduino Mega 2560 & Uno

Arduino Mega 2560 R3 and Arduino Uno R3 are microcontroller boards, The Arduino Mega 2560 is based on the ATmega2560. It has 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

The Arduino Uno is based on the ATmega328P. It has 14 digital input/output pins, 6 analog inputs, and a 16 MHz ceramic resonator, with the same USB connection, a power jack, ICSP header, and a reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. [24] [25] (Figure 36,Figure 35)

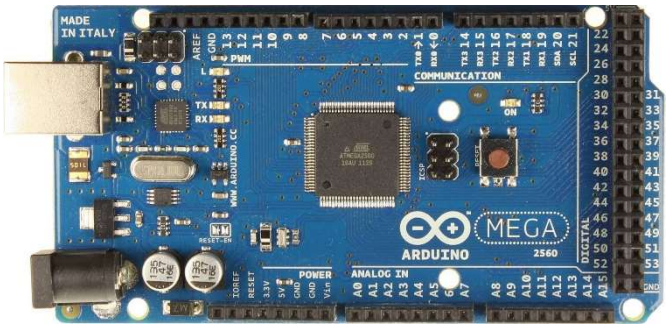


Figure 36 Arduino Mega 2560 R3

Figure 35 Arduino Uno R3

4.1.2 Circuit Diagram

4.1.2.1 Interfacing of ESP32-CAM with ARDUINO

The connections for interfacing the Arduino with ESP32 CAM are shown in Figure 37 Mainly, it consists of total of six connections

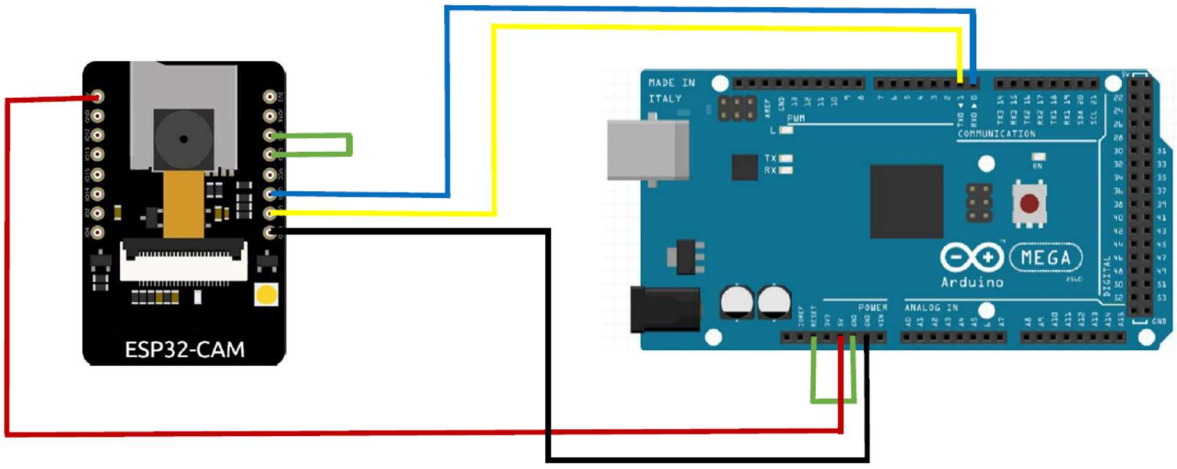


Figure 37 Interfacing of ESP32-CAM with ARDUINO

Identification camera pins to Arduino and what every pin does show in Figure 38.

```
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
```

Figure 38 Configuration camera pins

Figure 39 shows the image processing, first, it flips it back because the camera is capturing it upside down. second, to get the best saturation to the image, and finally, to get a higher initial frame rate.

```
if (s->id.PID == OV3660_PID) {
  s->set_vflip(s, 1);
  s->set_brightness(s, 1);
  s->set_saturation(s, -2);
}
s->set_framesize(s, FRAMESIZE_QVGA);
```

Figure 39 Initial camera image processing

4.1.2.2 Camera movement mechanism

It uses a servo motor to move the camera to different directions where the motion is felt by PIR, the mechanism system has 3 PIR sensors in different directions to have 270 degrees of view for the camera, and the camera has another default view, Arduino UNO controls this mechanism. (Figure 40)

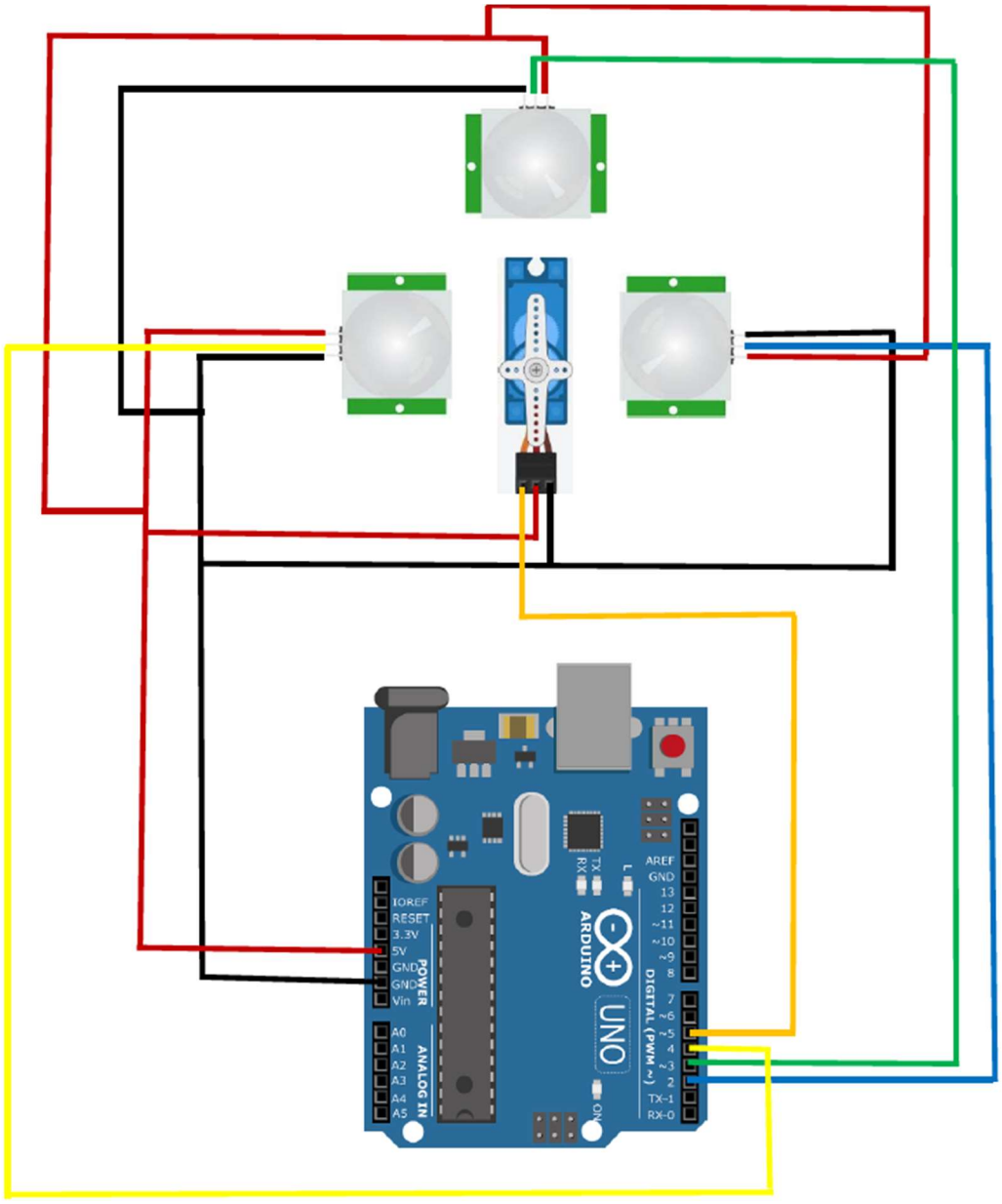


Figure 40 Camera Movement Mechanism

The servo moves the camera to the direction in which the PIR senses something, if it does not sense anything, it returns to the default that shown in Figure 41.

```
  SERVO.write(0);
if (PIR_STATE_EGY == HIGH) {
 SERVO.write(33);
  delay(1000);
  Serial.println("Hey I got you!!!_1");
  delay(5000);
}
else if (PIR_STATE_FRA == HIGH) {
 SERVO.write(65);
  delay(500);
  Serial.println("Hey I got you!!!_2");
  delay(5000);
}
 else if (PIR_STATE_ITA == HIGH) {
   SERVO.write(100);
  delay(500);
  Serial.println("Hey I got you!!!_3");
  delay(5000);
}
else {
 SERVO.write(0);
  delay(5000);
}
```

Figure 41 Servo movement depending on PIR

4.1.2.3 Interfacing of DS18B20 and DHT11 with NodeMCU

NodeMCU (Node Microcontroller Unit) is connected with DS18B20 Temperature Sensor Module and DHT11 Humidity & Temperature Sensor to read and handle data from sensors, it uses MQTT (Message Queuing Telemetry Transport) protocol to send data the real data to the server. (Figure 42)

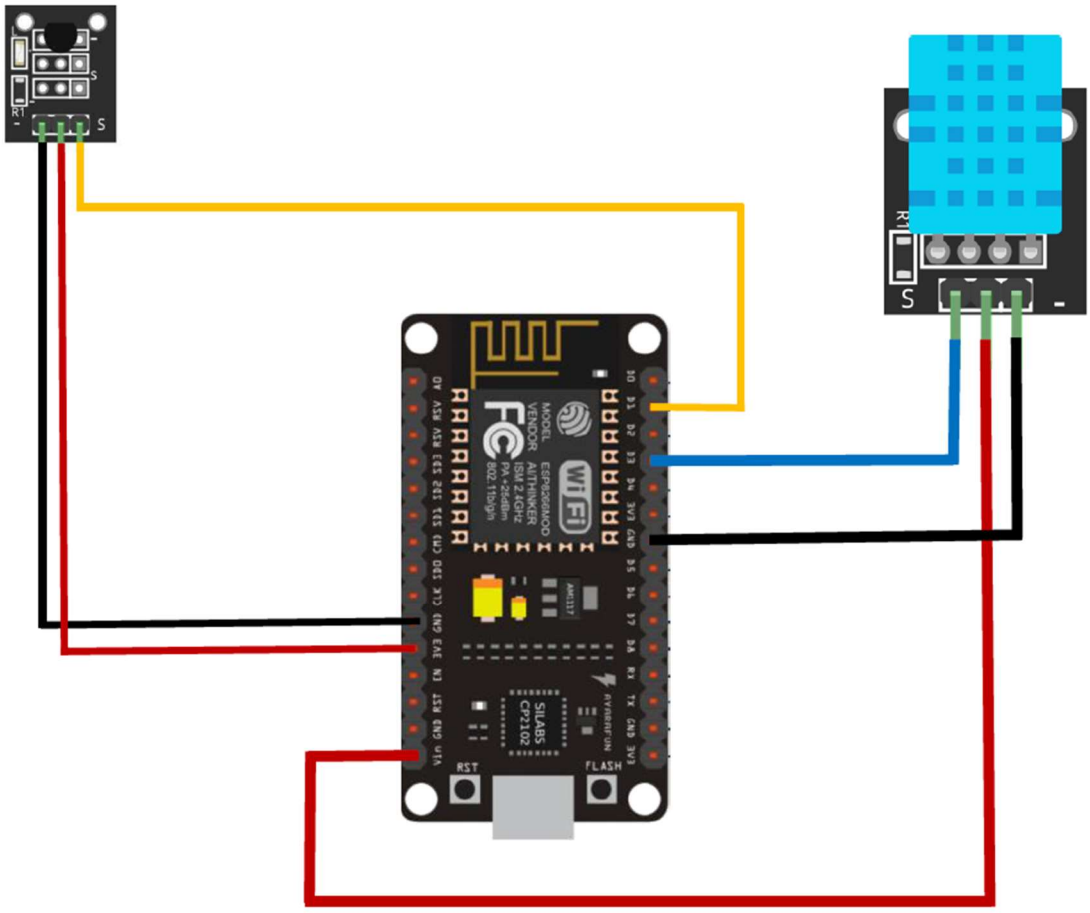


Figure 42 Interfacing of DS18B20 and DHT11 with NodeMCU

4.1.2.4 Interfacing of MQ-135 and Max30100 with NodeMCU

NodeMCU (Node Microcontroller Unit) is connected with MQ-135 Air Quality and Hazardous Gas Sensor and Max30100 pulse oximetry and heart-rate monitor Sensor to read and handle data from sensors (Note: MQ-135 connected to the only analog pin on NodeMCU), it uses MQTT (Message Queuing Telemetry Transport) protocol to send data the real data to the server. (Figure 43) [26]

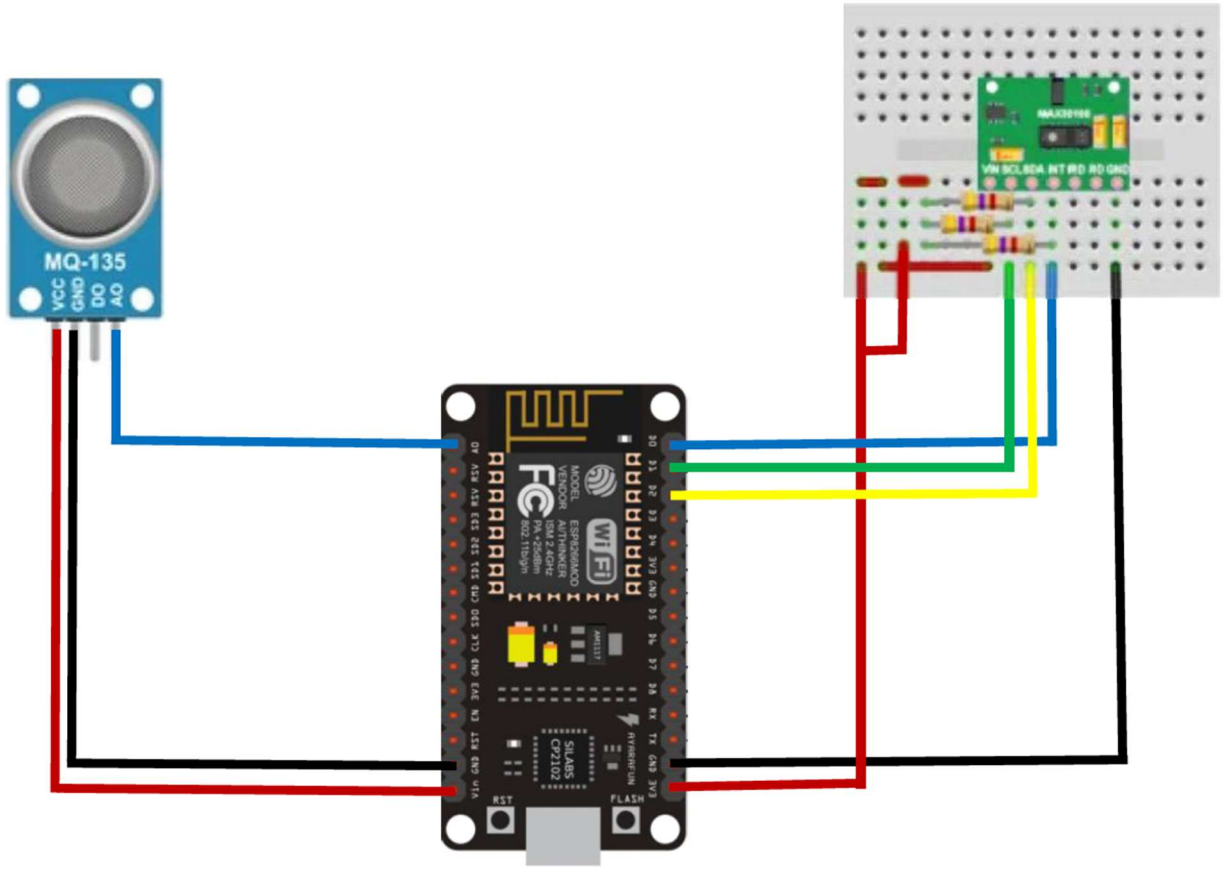


Figure 43 Interfacing of MQ-135 and Max30100 with NodeMCU

Figure 44 explains how to define the information of the cloud (IBM Watson server).

```
#define ORG "hkg2cn"
#define DEVICE_TYPE "NodeMCU_1"
#define DEVICE_ID "ESP-8266-E12-F1"
#define TOKEN "Wtk!a_rRmrAdO(ssaS"
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
```

Figure 44 Cloud Information

In Figure 45, provide a form to publish the data as a JSON file

```
char pubTopic_1[] = "iot-2/evt/status1/fmt/json";
char pubTopic_2[] = "iot-2/evt/status2/fmt/json";
char pubTopic_3[] = "iot-2/evt/status3/fmt/json";
char authMethod[] = "use-token-auth";
```

Figure 45 publish Form

Figure 46 explains trying to connect with wi-fi and sever

```
Serial.println();
Serial.print("Connecting to ");
Serial.print(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.print("WiFi connected, IP address: ");
Serial.println(WiFi.localIP());
if (!client.connected()) {
  Serial.print("Reconnecting client to ");
  Serial.println(server);
  while (!client.connect(clientId, authMethod, token)) {
    Serial.print(".");
    delay(500);
  }
```

Figure 46 Wi-fi Connection

Get a reading from sensors and convert the reading to percentage and real value shown in Figure 47.

```
float Sensor = analogRead (A0);
int Initial_Sensor = ((Sensor/1024)*100) ;
int Sensor_Value = (100-Initial_Sensor);
```

Figure 47 Get a reading from the sensor

Figure 48 shows the Publishing process to the Cloud and checks if it done

```
String payload_3 = "{\"d\":{\"Name\":\"" DEVICE_ID "\"";
payload_3 += ",\"Sensor\":";
payload_3 += Sensor_Value;
payload_3 += "}}";
Serial.print("Sending payload: ");
Serial.println(payload_3);
if (client.publish(pubTopic_3, (char*) payload_3.c_str())) {
  Serial.println("Publish ok");
} else {
  Serial.println("Publish failed");
}
```

Figure 48 Publish Data

4.1.3 IOT platform

Watson IoT platform is Subscribed JSON file (data) from MQTT Broker.

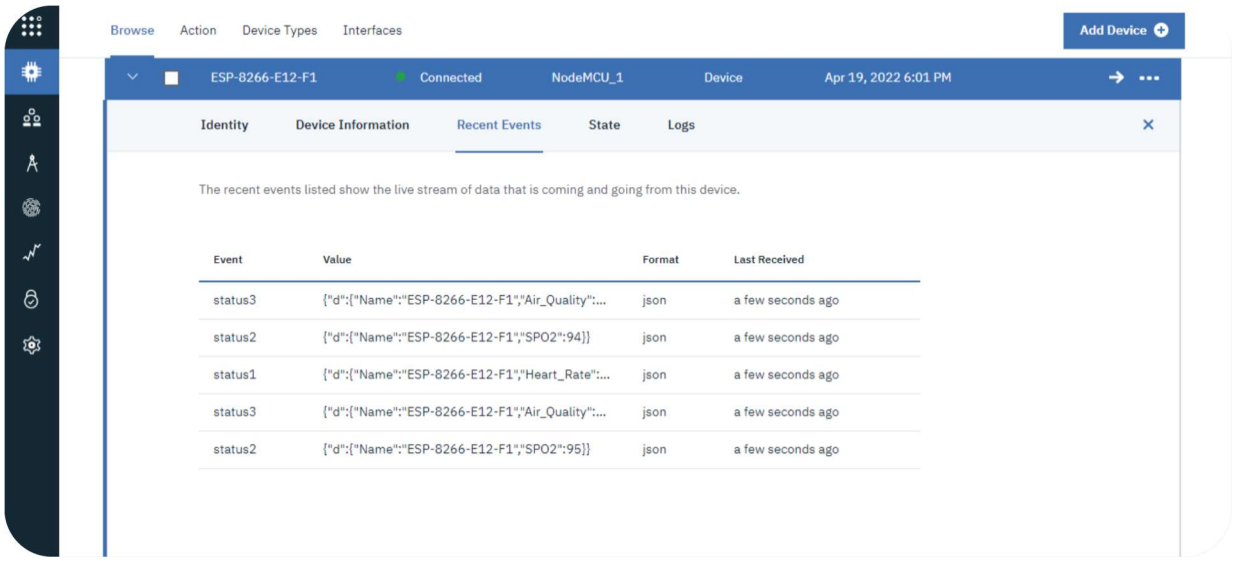


Figure 49 Recent Events

Show data on dashboards with the possibility of setting a specific time period.

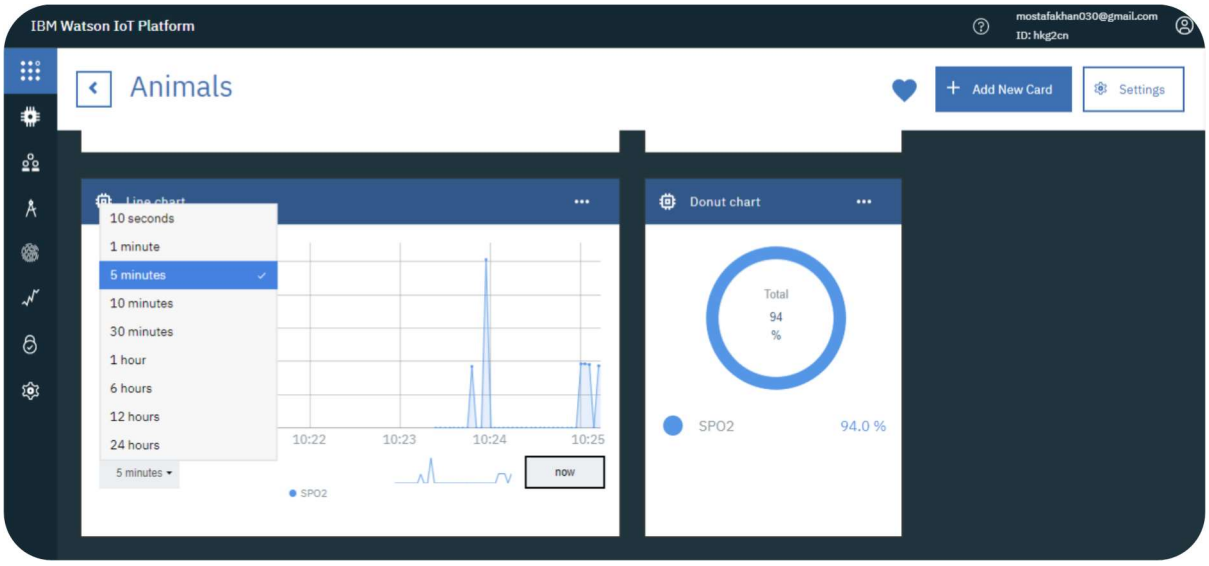


Figure 50 Displaying data

Generate API Key to use it in connecting Watson IoT platform to Node-Red to send JSON files (data).

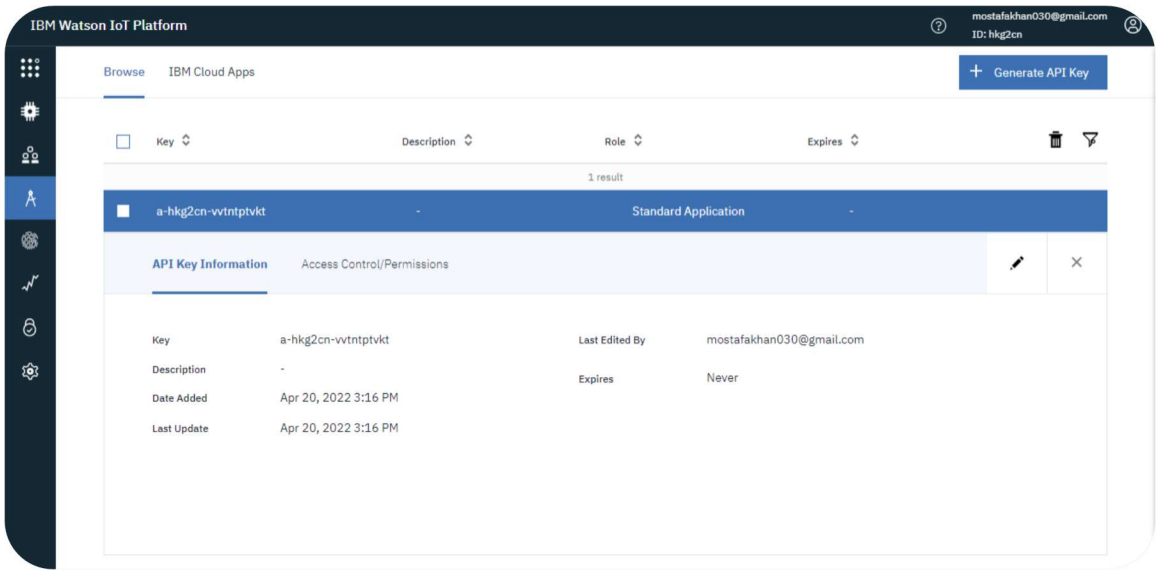


Figure 51 Generate API Key

4.2 Implementation of Software

4.2.1 Flow-based programming

Flow-based programming (FBP) views an application as a network of asynchronous processes that communicate by using streams of structured data chunks that are called information packets. In this view, the focus is on the application data and the transformations that are applied to it to produce the wanted outputs. The network is defined externally to the processes as a list of connections that is interpreted by a piece of software that is called the "scheduler". FBP is an effective way to produce reliable, maintainable, and large business applications. There is increasing interest in it worldwide. [27]

FBP is visual and component-oriented, which means that the processes inside an application can be reconnected endlessly to form different applications without having to be changed internally, is a “coordination language”, not a programming language, which makes it simple to reconfigure the application’s components, and is language-independent.

4.2.2 Node-Red

Node-RED is a Node.js-based run time at which you point a web browser to access the flow editor. Within the browser, you create your application by dragging nodes from your palette into a workspace and wiring them together. With a single click, the application is deployed back to the run time and ran. It is a way to build a fully functioning back end by using a drag-style interface and minimum coding experience. The palette of nodes can be extended by installing new nodes that are created by the Node-RED community. The flows that you create can be easily shared as JSON files. [28]

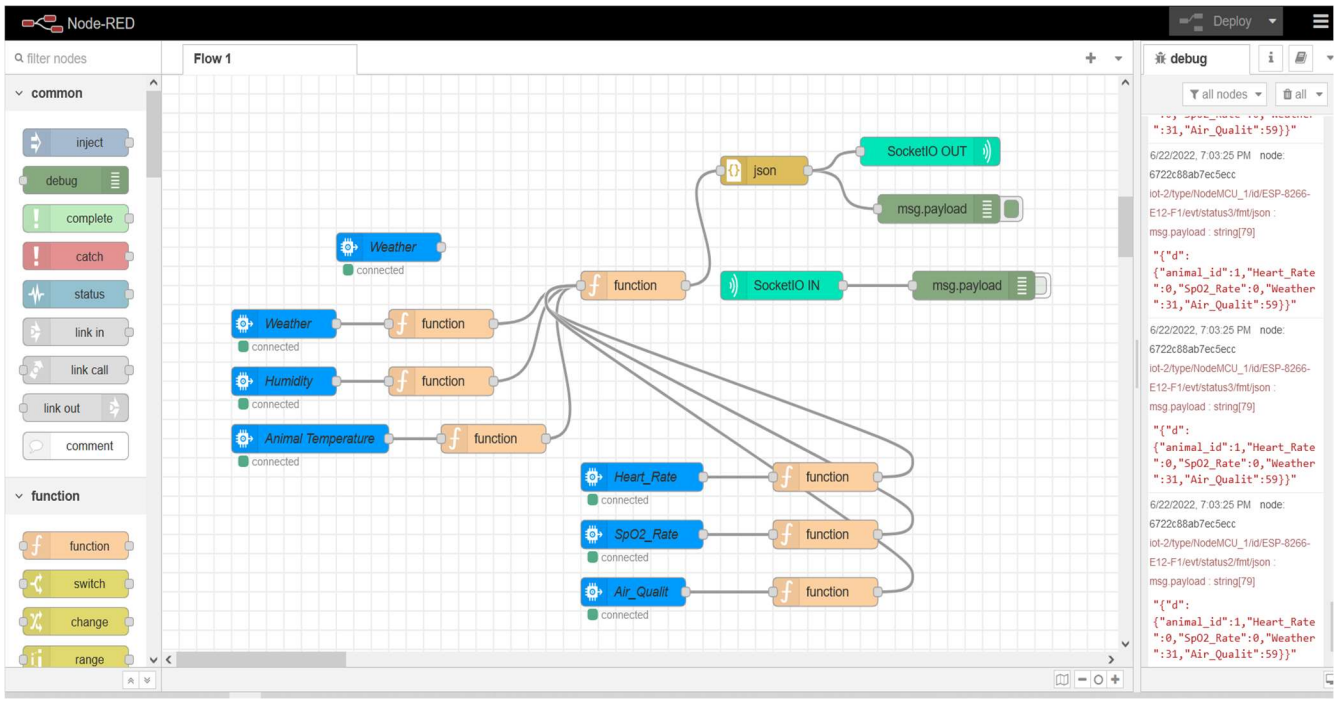


Figure 52 The flow of nodes

Figure 53 all Received sensors data from Arduino is transferred via Watson API and passes to the function node to set it as a global value then each function sends its data to the integrated function to get JSON data containing all collected sensor data (message payload) as shown in figure 4.6 then send it to JSON function for parsing and finally passes to socketio out to can get it in node js.

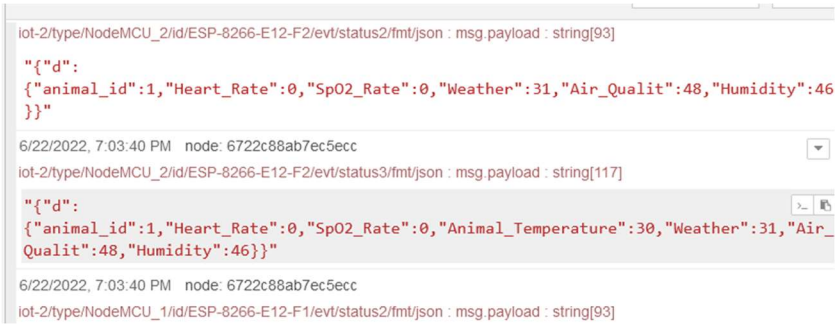


Figure 53 Message payload

Figure 54 all sensors data stream can be sent to backend via web socket connection that makes connection between node-red server and node js client to send the stream of data when user log into the website.

```
module.exports = function (app) {

  if (app.io) {  // if a client is connected to node js socketio

    app.io.emit('connected', {text: 'A client connected!'});
  }
  console.log('connecting to 1880....');

  socket2.on('connect', function () {
    console.log('ccccccccccccccccccccccccccccccccccccccccc');
  });
  socket2.on('stream', async (data) => {

    const looooll = JSON.parse(data);
    const {animal_id, Heart_Rate, SpO2_Rate, Animal_Temperature, Weather, Air_Qualit, Humidity} =
looooll['d'];

    const newRow = {
      animal_id: parseInt(animal_id),
      Heart_Rate: parseInt(Heart_Rate),
      SpO2_Rate: parseInt(SpO2_Rate),
      Animal_Temperature: parseInt(Animal_Temperature),
      Weather: parseInt(Weather),
      Air_Qualit: parseInt(Air_Qualit),
      Humidity: parseInt(Humidity)
    };

    await app.service('sensor-data').create(newRow);

    if (app.io !="undefined") {
      app.io.emit('stream', newRow)
    }

  });
  socket2.on('connect_error', (err) => {
    console.log(err);
    // console.log(err.connect);
    socket2.connect();
    if (err.message === 'invalid credentials') {
      socket2.auth.token = 'efgh';
      socket2.connect();
    }});
```

Figure 54 The function of sending data

The method of connecting between frontend and backend.

What is Axios?

Axios is used to communicate with the backend and it also supports the Promise API that is native to JS ES6. It is a library that is used to make requests to an API, return data from the API, and then do things with that data in our React application. Axios is a very popular (over 78k stars on Github) HTTP client, which allows us to make HTTP requests from the browser.

Why Axios?

Axios uses XMLHttpRequest under the hood, and it is widely supported by most browsers, including older ones like Internet Explorer 11. Fetch(), on the other hand, is only compatible with Chrome 42+, Firefox 39+, Edge 14+, and Safari 10.3.

web Application

In the following, we will show and identify EAPS system modules.

4.2.3 Sign-in Screen

On this page, the User and administrator can be authenticated regularly performed by entering a username and password combination as shown in Figure 55

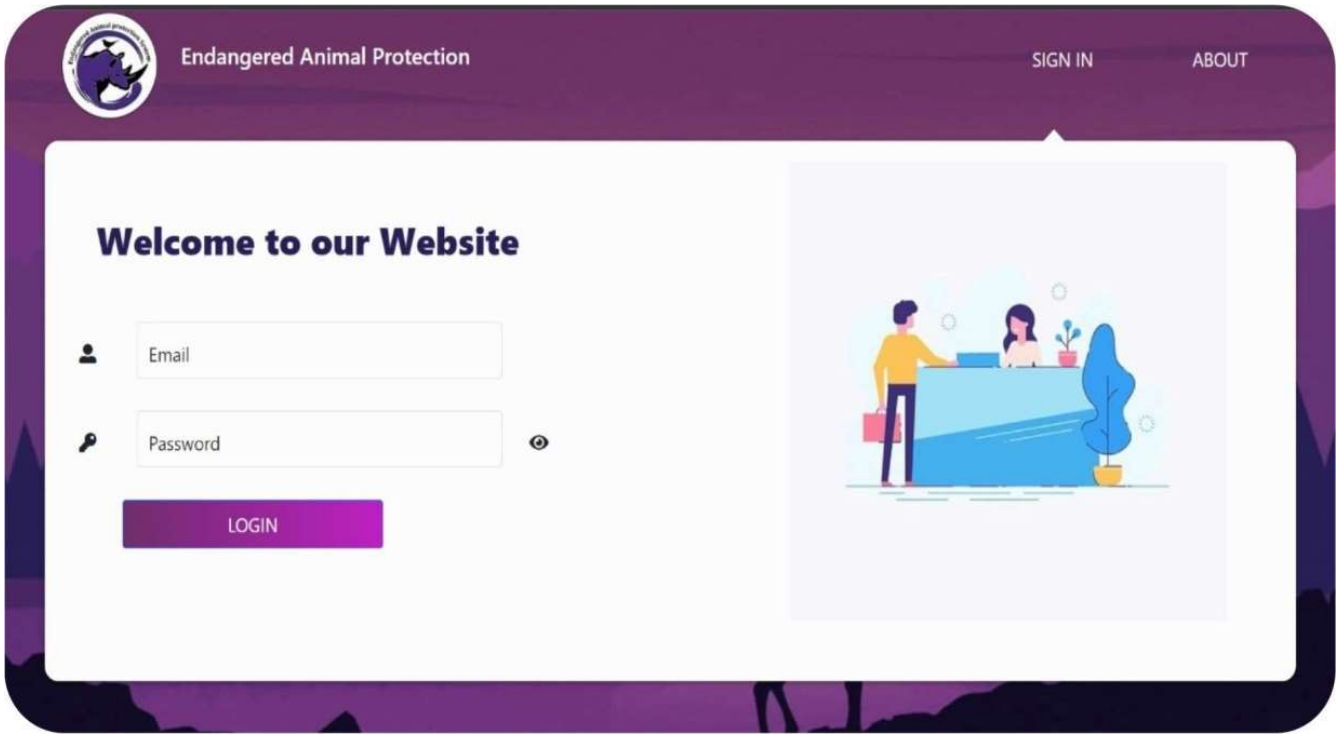
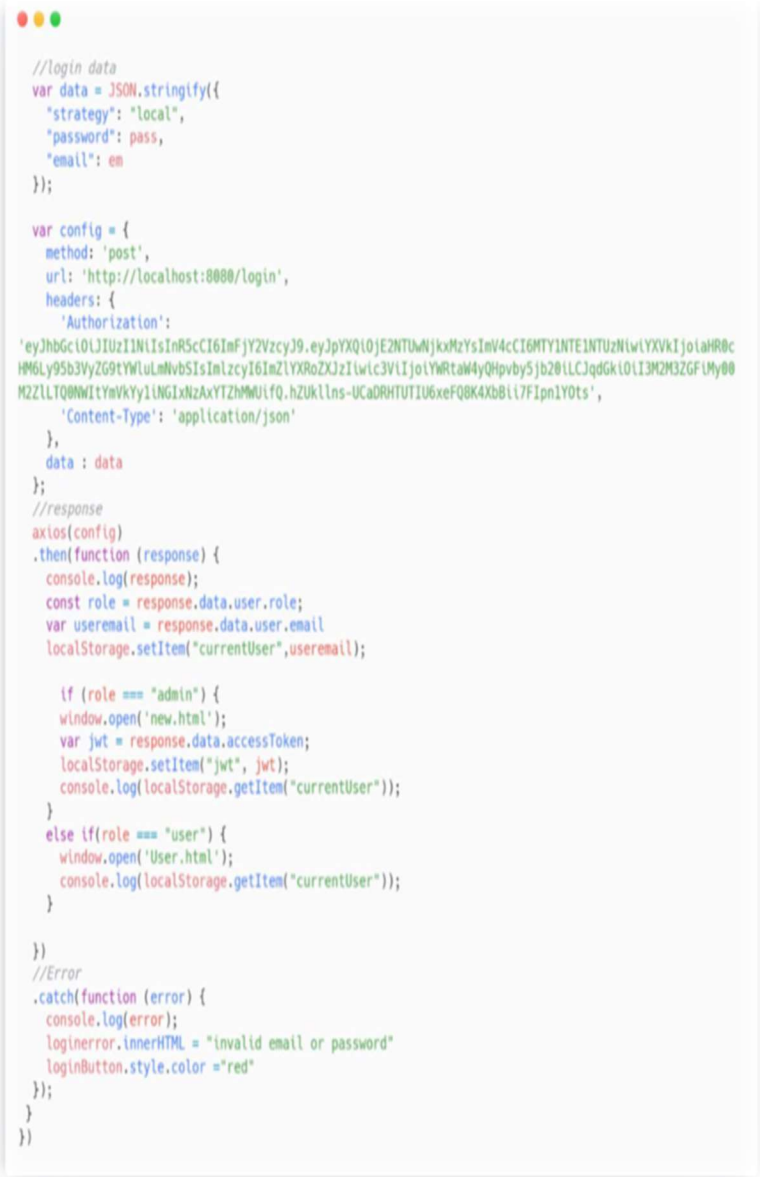


Figure 55 Sign in screen

Figure 56 and Figure 57 functions are used to verify the username and password from the API while logging in to the system.



```
//login data
var data = JSON.stringify({
  "strategy": "local",
  "password": pass,
  "email": em
});

var config = {
  method: 'post',
  url: 'http://localhost:8080/login',
  headers: {
    'Authorization':
'eyJhbGciOiJIUzI1NiIsInR5cCI6ImFjY2VzcyJ9.eyJpYXQiOjE2NTUwNjkxMzYsImV4cCI6MTY1NTE1NTUzNiwiYXVkIjoiaHR0c
HM6Ly95b3VyZG9tYWluLmNvbSIsImlzcyI6ImZlYXRoZXJzIiwic3ViIjoiYWRtaW4yQHpvby5jb20iLCJqdGkiOiI3M2M3ZGFlMy00
M2ZlLTQ0NWItYmVkYy1iNGIxNzAxYTZhMWUifQ.hZUkllns-UCaDRHTUTIU6xeFQ8K4XbBi7FIpn1YOts',
    'Content-Type': 'application/json'
  },
  data : data
};
//response
axios(config)
.then(function (response) {
  console.log(response);
  const role = response.data.user.role;
  var useremail = response.data.user.email
  localStorage.setItem("currentUser",useremail);

    if (role === "admin") {
    window.open('new.html');
    var jwt = response.data.accessToken;
    localStorage.setItem("jwt", jwt);
    console.log(localStorage.getItem("currentUser"));
  }
  else if(role === "user") {
    window.open('User.html');
    console.log(localStorage.getItem("currentUser"));
  }

})
//Error
.catch(function (error) {
  console.log(error);
  loginerror.innerHTML = "invalid email or password"
  loginButton.style.color ="red"
});
}
})
```

Figure 56 Login function in frontend side

```
module.exports = (app) => {
  return async function login(req, res, next) {
    const body = req.body;

    const user = {
      email: body.email,
      password: body.password
    };

    try {
      const service =  await app.service('users').find(user)

      const auth = await app.service('authentication')

      console.log('received body is ', user ,
        "user form system is " , service , " auth is " , auth
      );

      console.log('res is ',
      );

      res.status(201).json({
        status:"success",
        service

      })

    } catch (e) {
      console.log('catch', e, 'res is ', res);

      res.status(201).json({
        status:"failed",

      })})};
```

Figure 57 Login function in backend side

4.2.4 Stream video

Figure 58 the first screen for the administrator is the monitoring screen that shows the streaming video that reveals what is around the animal and the measures of the temperature sensors, the heart rate sensor, the air purity sensor, the air humidity sensor, the animal temperature sensor and the oximeter sensor that can be displayed all at the same time or any one of them.

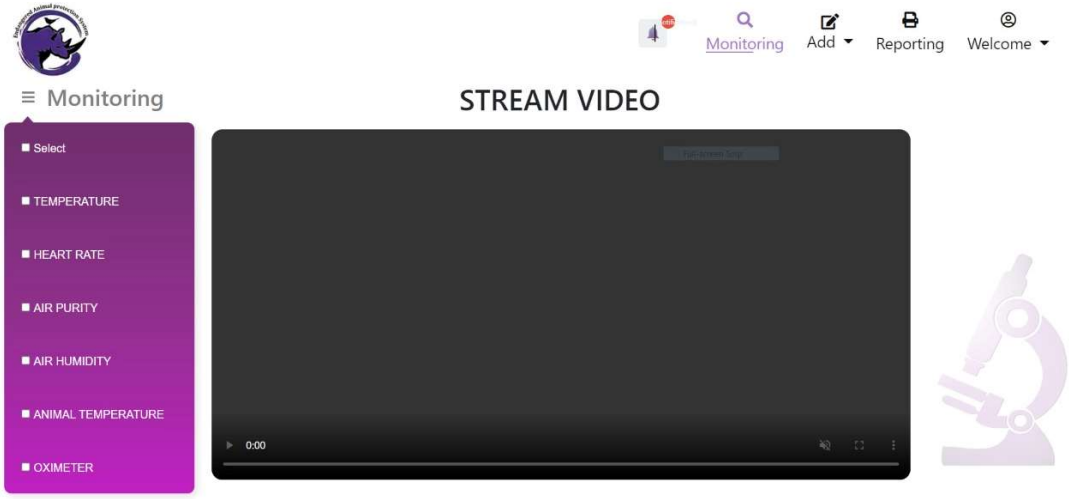


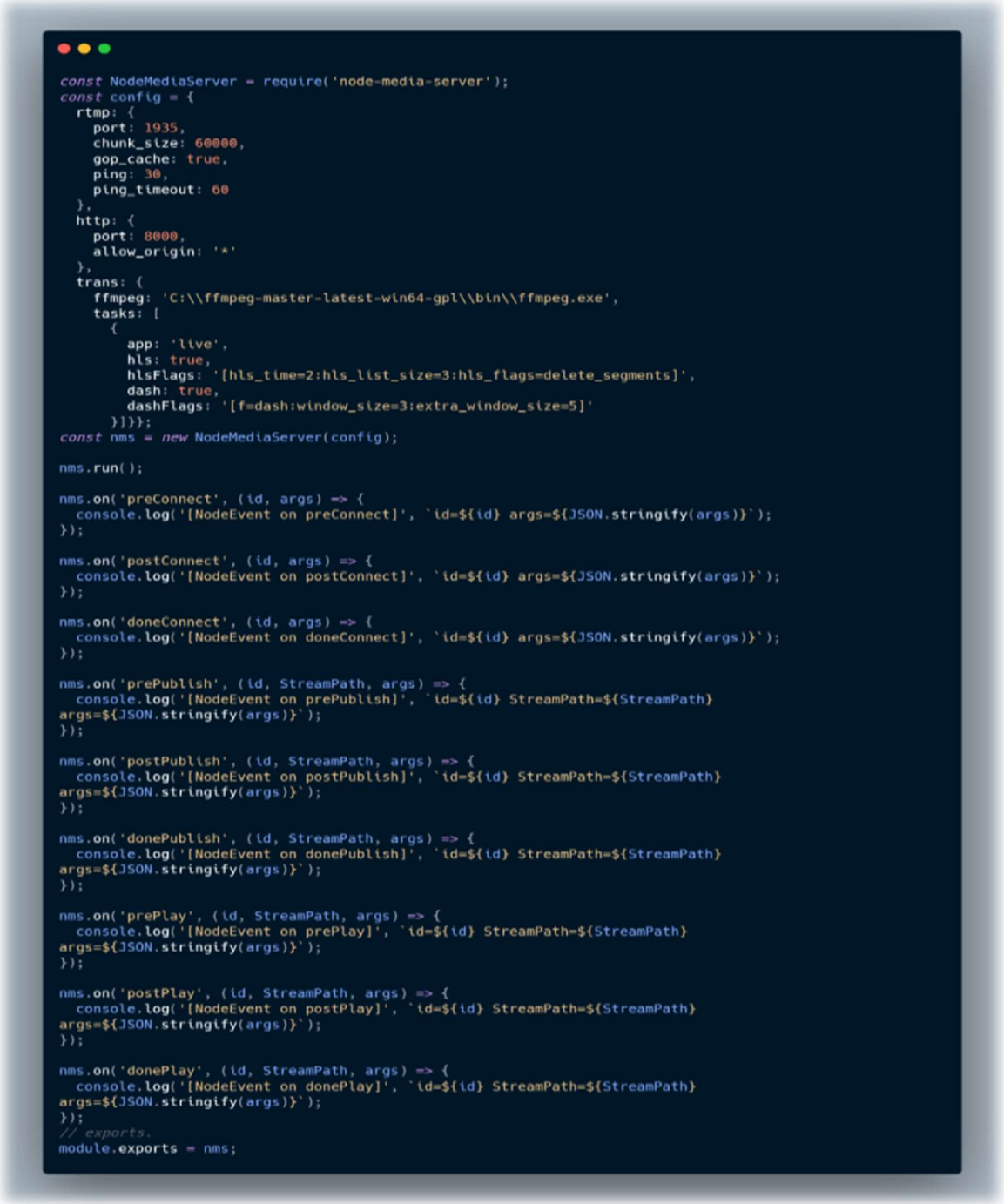
Figure 58 Stream video

Figure 59 element to display live stream video in frontend.

```
<video id="videoPlayer" width="90%" height="90%" controls muted="muted" autoplay>
        <source src="/liveStream" type="video/mp4" />
</video>
```

Figure 59 Element live stream function

Figure 60 function is used to Publishing live stream.



```
const NodeMediaServer = require('node-media-server');
const config = {
  rtmp: {
    port: 1935,
    chunk_size: 60000,
    gop_cache: true,
    ping: 30,
    ping_timeout: 60
  },
  http: {
    port: 8000,
    allow_origin: '*'
  },
  trans: {
    ffmpeg: 'C:\\ffmpeg-master-latest-win64-gpl\\bin\\ffmpeg.exe',
    tasks: [
      {
        app: 'live',
        hls: true,
        hlsFlags: '[hls_time=2:hls_list_size=3:hls_flags=delete_segments]',
        dash: true,
        dashFlags: '[f=dash:window_size=3:extra_window_size=5]'
      }]}};
const nms = new NodeMediaServer(config);

nms.run();

nms.on('preConnect', (id, args) => {
  console.log('[NodeEvent on preConnect]', `id=${id} args=${JSON.stringify(args)}`);
});

nms.on('postConnect', (id, args) => {
  console.log('[NodeEvent on postConnect]', `id=${id} args=${JSON.stringify(args)}`);
});

nms.on('doneConnect', (id, args) => {
  console.log('[NodeEvent on doneConnect]', `id=${id} args=${JSON.stringify(args)}`);
});

nms.on('prePublish', (id, StreamPath, args) => {
  console.log('[NodeEvent on prePublish]', `id=${id} StreamPath=${StreamPath}
args=${JSON.stringify(args)}`);
});

nms.on('postPublish', (id, StreamPath, args) => {
  console.log('[NodeEvent on postPublish]', `id=${id} StreamPath=${StreamPath}
args=${JSON.stringify(args)}`);
});

nms.on('donePublish', (id, StreamPath, args) => {
  console.log('[NodeEvent on donePublish]', `id=${id} StreamPath=${StreamPath}
args=${JSON.stringify(args)}`);
});

nms.on('prePlay', (id, StreamPath, args) => {
  console.log('[NodeEvent on prePlay]', `id=${id} StreamPath=${StreamPath}
args=${JSON.stringify(args)}`);
});

nms.on('postPlay', (id, StreamPath, args) => {
  console.log('[NodeEvent on postPlay]', `id=${id} StreamPath=${StreamPath}
args=${JSON.stringify(args)}`);
});

nms.on('donePlay', (id, StreamPath, args) => {
  console.log('[NodeEvent on donePlay]', `id=${id} StreamPath=${StreamPath}
args=${JSON.stringify(args)}`);
});
// exports.
module.exports = nms;
```

Figure 60 Publish live stream function

4.2.5 Add a new user screen

This function helps add a new user to the system user or admin rules and this information was not available before, and this page to the user in the add new user screen accepts username, email, and password to validate the addition user process in front side as shown in Figure 61.

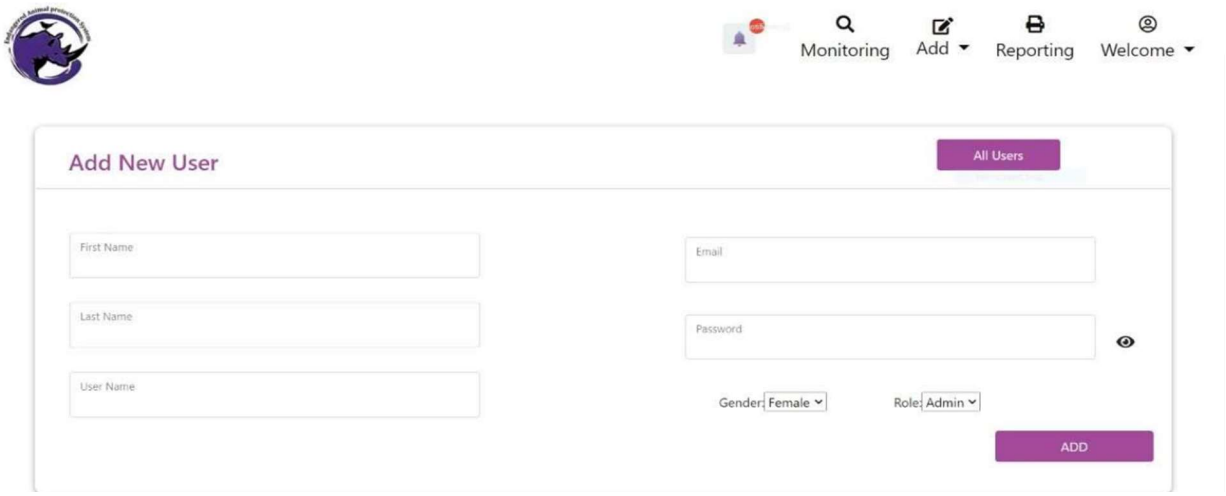


Figure 61 Add new user screen

```
exports.signup = (app) => {
  return async function signup(req, res, next) {
    const body = req.body;

    const user = {
      email: body.email,
      password: body.password,
      username: body.username,
      first_name: body.first_name,
      last_name: body.last_name,
      role: body.role || 'user',

    };


    try {
      const data = await app.service('users').find({
              query: {
                email: {
                  $eq: user.email,
                },
                $allowFiltering: true
              }
            })
      console.log('found a user with these crs',user.email, data.data.length )

      if (data.data.length != 0 ) {
        res.status(400).json({
          status: 'cannot create a user with these credentials as they already exist',

        });

      } else {
        const createdUser = await app.service('users').create(user);
        res.status(201).json({
          status: 'success',
          username: createdUser.username,
          email: createdUser.email,
          role: createdUser.role
        });}

    } catch (e) {
      console.log('catch', e,);

      res.status(400).json({
        status: 'error in data',
        e

      });};};
exports.restrictTo = (...roles) => {
  return (req, res, next) => {
    if (!roles.includes(req.user.role)) {
      return next(
        new AppError('You do not have permission to perform this action', 403)
      );}
    next();
  };};
```

Figure 63 Signe up function in backend side

```
      //Add New User
       var data = JSON.stringify({
         "password": password.value,
        "email": usemail.value,
        "username": uname.value,
        "first_name": fname.value,
       "last_name": lname.value,
       "role": role.value
      });

       var config = {
       method: 'post',
        url: 'http://localhost:8080/signup',
       headers: {
      'Authorization':jwt,
      'Content-Type': 'application/json'
      },
    data : data
};
//Response
axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
  errorMessage.innerHTML = "Done"

})
//Error
.catch(function (error) {
  console.log(error);
  errorMessage.innerHTML = "field"
});
```

Figure 62 Add a new user function in frontend side

4.2.6 Add a new animal screen

On this page the admin in the add new animal screen and accepts the id for the animal to validate the addition animal process as shown in Figure 64.

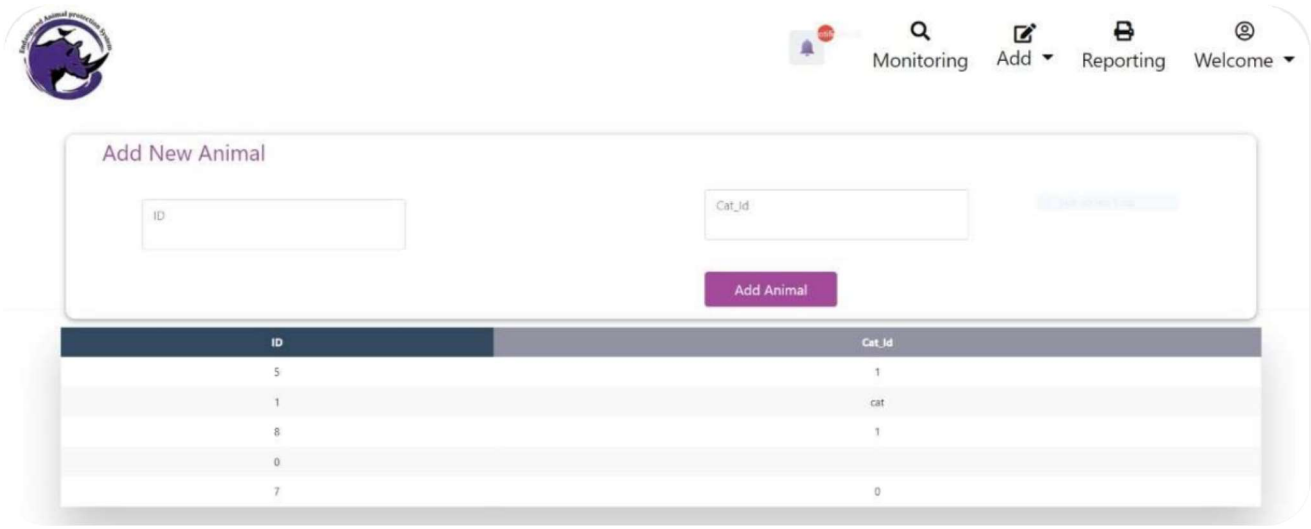


Figure 64 Add new animal screen

```
exports.addNewAnimalInfo = (app) => {
  return async function animalOperations(req, res, next) {
    const body = req.body;

    const animal = {
      id:body.id,
      cat_id : body.cat_id
    }
    try {
      await app.service('animals').create(animal);
      res.status(200).json({
        status: 'success',
      });
    } catch (e) {
      console.log('catch', e, 'res is ', res);

      res.status(400).json({
        status: 'error in data',

      });}};};
```

Figure 65 Add new animal function in backend side

```
//Add New Animal
 var data = JSON.stringify({
   "id":Number(id.value),
   "cat_id": cat_id.value
 });

 var config = {
   method: 'post',
   url: 'http://localhost:8080/addNewAnimalIfon',
   headers: {
     'Authorization': jwt,
     'Content-Type': 'application/json'
   },
   data : data
 };
 //Response
 axios(config)
 .then(function (response) {
   console.log(JSON.stringify(response.data));
 })
 //Error
 .catch(function (error) {
   console.log(error);
 });
```

Figure 66 Add new animal function in frontend side

4.2.7 Edit profile

Figure 67 the edit profile screen accepts the user can modify his data, except for the email.

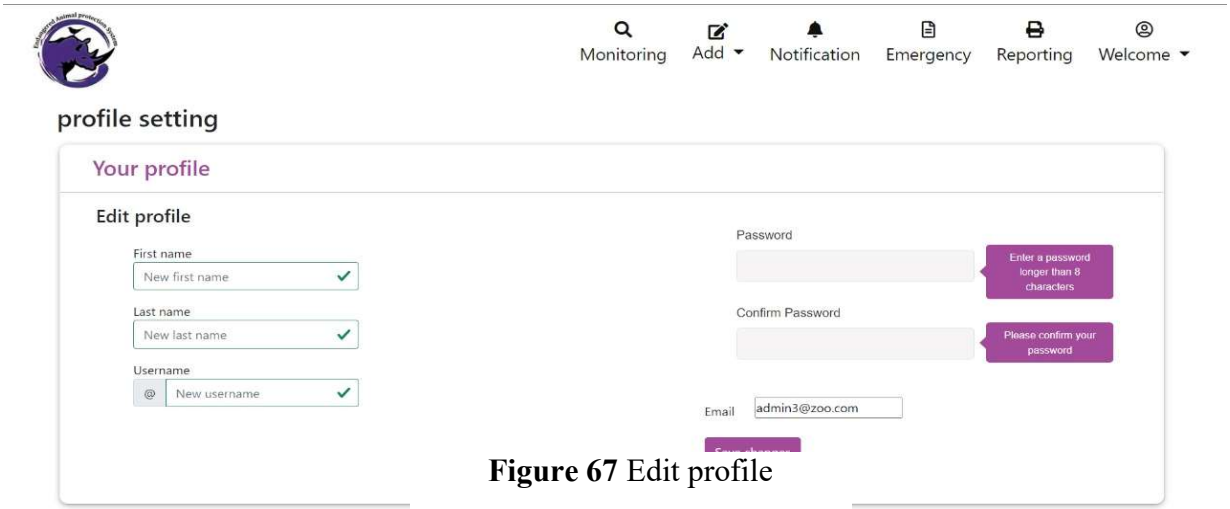
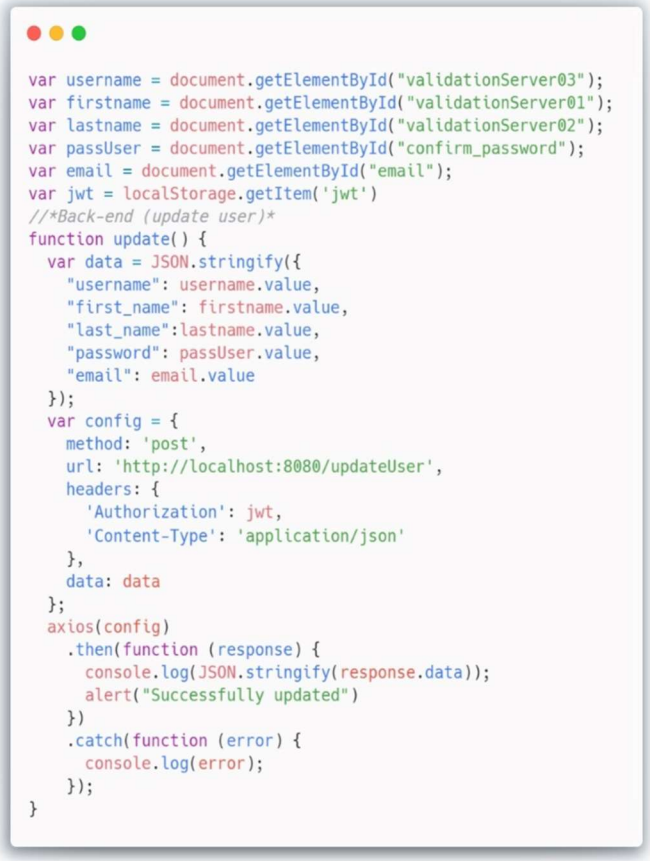


Figure 67 Edit profile

```
exports.updateUser = (app) => {
  return async function userOperations(req, res, next) {
    const body = req.body;
    const userFromReq = {
      email: body.email,
    };
    try {
      const data = await app.service('users').find({
        query: {
          email: {
            $eq: userFromReq.email,
          },
          $allowFiltering: true
        }
      });
      console.log('found a user with these crs', userFromReq.email, data.data.length);

      if (data.data.length == 0) {
        res.status(400).json({
          status: 'user doesn\'t exist',

        });

      } else {
        const service = await app.service('users');
        const user = await service.get(userFromReq);

        const newData = await service.patch(userFromReq, {
          username: body.username || user.username,
          first_name: body.first_name || user.first_name,
          last_name: body.last_name || user.last_name,
          password: user.password || body.password,
        });
        res.status(200).json({
          status: 'success',
          data:{
            email: newData.email,
            first_name: newData.first_name,
            last_name: newData.last_name,
            username: user.username,
            old: user.password,
            password: newData.password
          }});}
} catch (e) {
      console.log('ttttttttttttt', e);

      res.status(400).json({
        status: 'error in data',

      });}};};
```

Figure 69 Edit profile function in backend side



```
var username = document.getElementById("validationServer03");
var firstname = document.getElementById("validationServer01");
var lastname = document.getElementById("validationServer02");
var passUser = document.getElementById("confirm_password");
var email = document.getElementById("email");
var jwt = localStorage.getItem('jwt')
//*Back-end (update user)*
function update() {
  var data = JSON.stringify({
    "username": username.value,
    "first_name": firstname.value,
    "last_name":lastname.value,
    "password": passUser.value,
    "email": email.value
  });
  var config = {
    method: 'post',
    url: 'http://localhost:8080/updateUser',
    headers: {
      'Authorization': jwt,
      'Content-Type': 'application/json'
    },
    data: data
  };
  axios(config)
    .then(function (response) {
      console.log(JSON.stringify(response.data));
      alert("Successfully updated")
    })
    .catch(function (error) {
      console.log(error);
    });
}
```

Figure 67 Edit profile function in frontend side

4.2.8 Get all user screen

Figure 70 page performs three operations get all users, search, and delete users to validate the process.

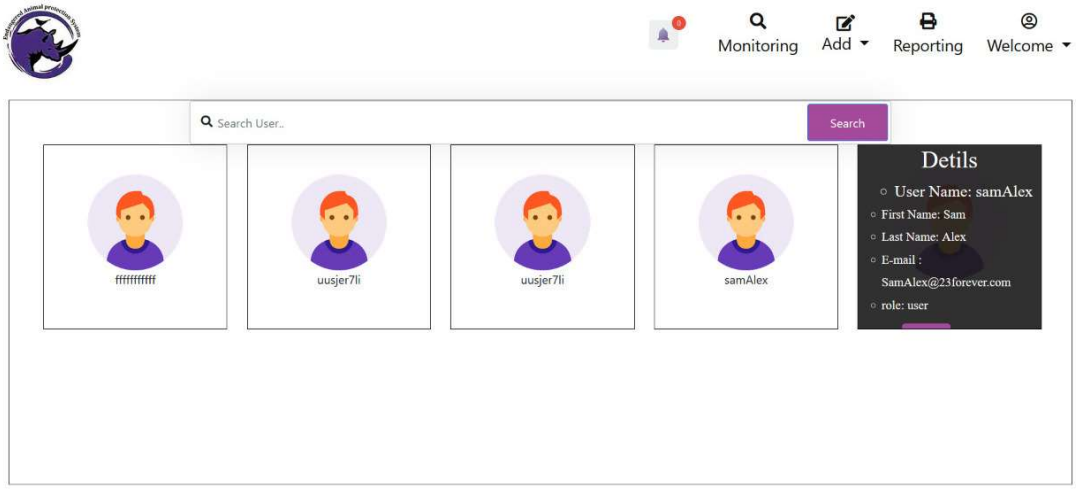


Figure 70 Get all user screen

```
exports.getUser = (app) => {
  return async function userOperations(req, res, next) {
    const body = req.body;
    const user = {
      email: body.email,
    };
    try {
      const service = await app.service('users');
      const data = await service.get(user);
      res.status(200).json({
        status: 'success',
        data: {
          email: data.email,
          first_name: data.first_name,
          last_name: data.last_name,
          role: data.role,
          username: data.username,
        }
      });
    } catch (e) {
      console.log('ttttttttttttt', e, 'res is ', res);

      res.status(400).json({
        status: 'no user associated with this info',

      });}};};
```

Figure 72 Search function in backend side

```
exports.getAllUsers = (app) => {
  return async function userOperations(req, res, next) {

    try {

      const service = app.service('users');

      const data = await service.find({
        query: {

          $select: ['email', 'username', 'first_name', 'last_name', 'role'],
          $allowFiltering: true
        }
      });
      res.status(201).json({
        status: 'success',
        data
      });

    } catch (e) {
      console.log('ttttttttttttt', e, 'res is ', res);

      res.status(400).json({
        status: 'error in data',

      });}};};
```

Figure 68 Get all users function in backend side

```
exports.deleteUser = (app) => {
  return async function userOperations(req, res, next) {
    const body = req.body;
    const user = {
      email: body.email,
    };
    try {
      const service = await app.service('users');
      const data = await service.remove(user);
      res.status(200).json({
        status: 'success',
      });
    } catch (e) {
      console.log('ttttttttttttt', e, 'res is ', res);

      res.status(400).json({
        status: 'error in data',

      });
    }
  };
};
```

Figure 69 Delete user function in backend side

4.2.9 Write the reported emergency

On this page, the User create a note (Report emergency) for a specific animal and determine the Danger level depending on the case that he detected as shown in Figure 70.

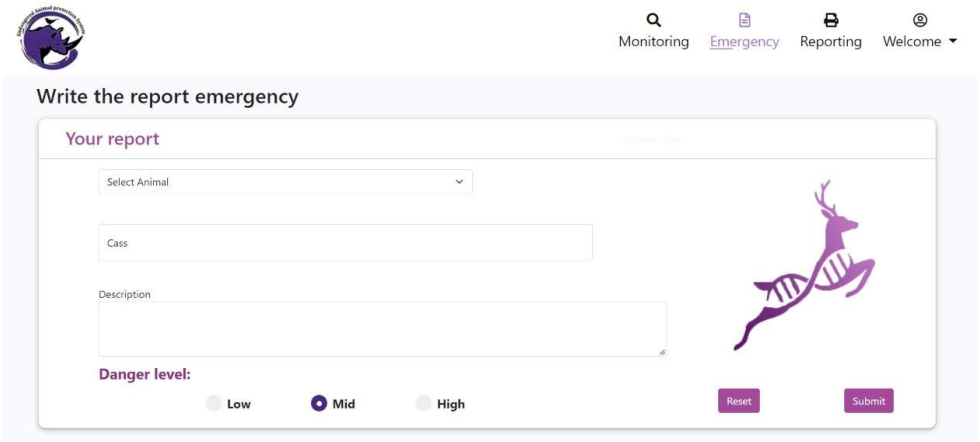
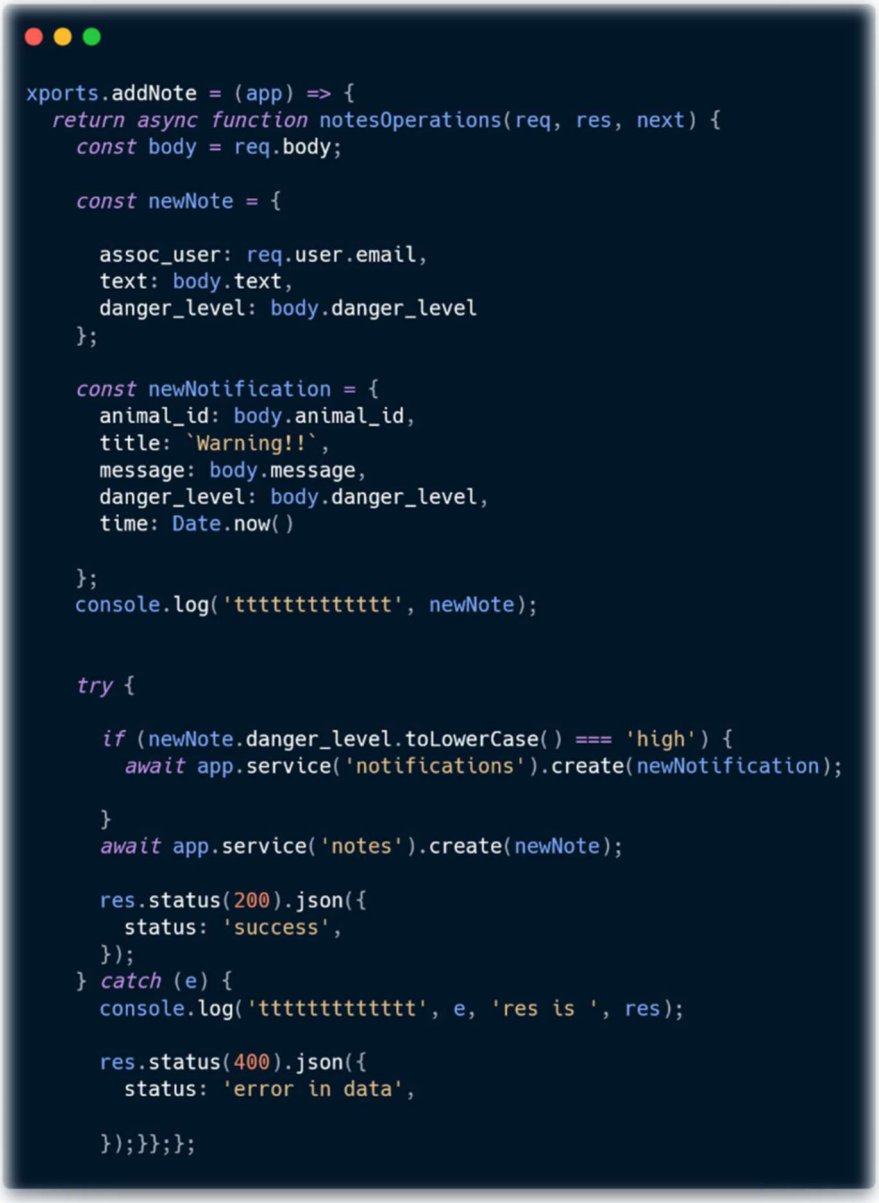


Figure 70 Write the report emergency



```
xports.addNote = (app) => {
  return async function notesOperations(req, res, next) {
    const body = req.body;

    const newNote = {

      assoc_user: req.user.email,
      text: body.text,
      danger_level: body.danger_level
    };

    const newNotification = {
      animal_id: body.animal_id,
      title: `Warning!!`,
      message: body.message,
      danger_level: body.danger_level,
      time: Date.now()

    };
    console.log('ttttttttttttt', newNote);


    try {

      if (newNote.danger_level.toLowerCase() === 'high') {
        await app.service('notifications').create(newNotification);

      }
      await app.service('notes').create(newNote);

      res.status(200).json({
        status: 'success',
      });
    } catch (e) {
      console.log('ttttttttttttt', e, 'res is ', res);

      res.status(400).json({
        status: 'error in data',

      });}};};
```

Figure 71 Function of Write the report emergency

4.2.10 Send a notification to the admin

Error! Reference source not found. the user with email (SamAlex@23forever.com) writes a note with a Danger level of "High", this note will be sent as a notification to the admin.

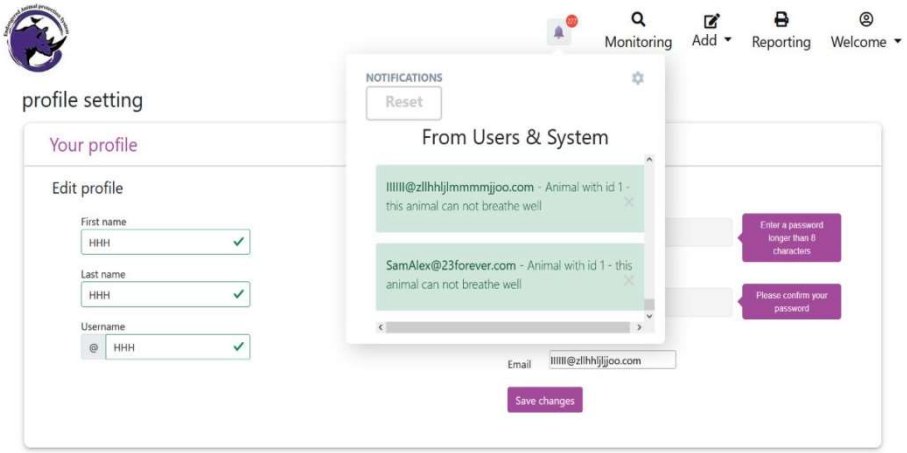


Figure 73 Send a notification to the admin

```
<div class="alert alert-success">
                    <strong>{{ item.title }}</strong> - Animal with id {{item.animal_id}} -
{{item.message}}
                    <button>
                      class="close">&times;
                    </button>
  </div>
```

Figure 72 Notification function in frontend

```
xports.addNote = (app) => {
  return async function notesOperations(req, res, next) {
    const body = req.body;

    const newNote = {

      assoc_user: req.user.email,
      text: body.text,
      danger_level: body.danger_level
    };

    const newNotification = {
      animal_id: body.animal_id,
      title: `Warning!!`,
      message: body.message,
      danger_level: body.danger_level,
      time: Date.now()

    };
    console.log('ttttttttttttt', newNote);


    try {

      if (newNote.danger_level.toLowerCase() === 'high') {
        await app.service('notifications').create(newNotification);

      }
      await app.service('notes').create(newNote);

      res.status(200).json({
        status: 'success',
      });
    } catch (e) {
      console.log('ttttttttttttt', e, 'res is ', res);

      res.status(400).json({
        status: 'error in data',
```

Figure 74 Notification function in backend

4.2.11 Report charts screen

When the admin want to show changes in data at of specific time for an animal from a specific category, select of time and show data on website as shown in Figure 77.

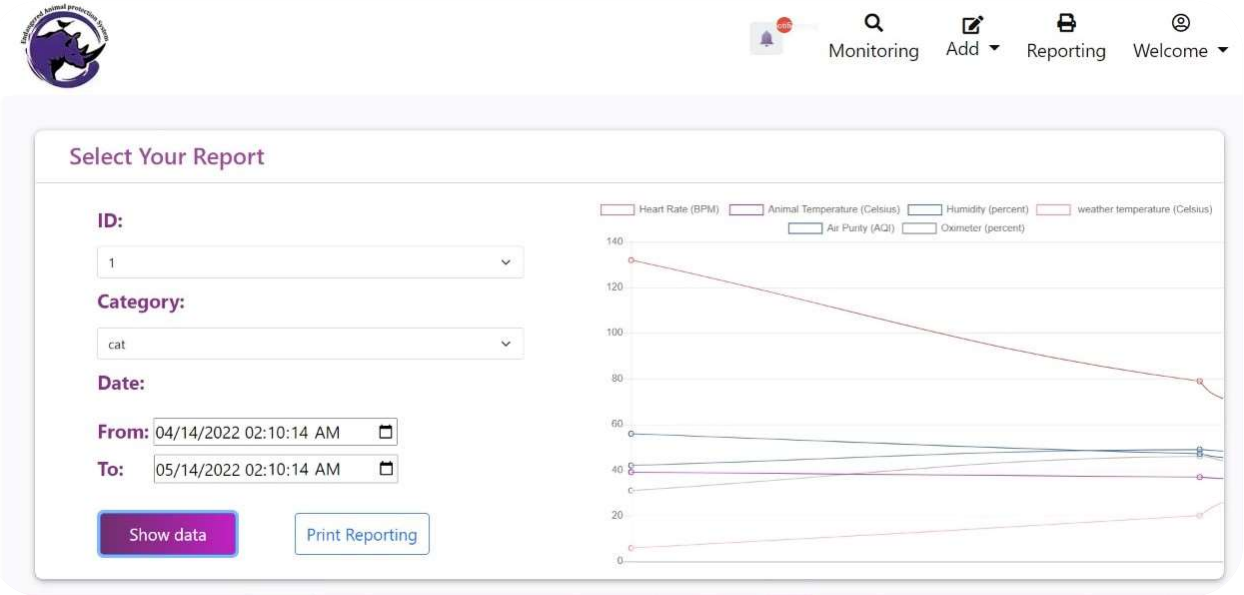


Figure 77 Report charts screen

```
exports.getFromTo = function (app) {
  return async function sensorDataStream(req, res, next) {
    const from = req.body['from'];
    const to = req.body['to'];

    let dateFrom = new Date(from);
    const dateTo = new Date(to);

    const diff = Math.abs(dateTo - dateFrom);

    let tempDate = dateFrom;

    console.log('2222222222222222222222222222 \n',
        '\n dateFrom ', dateFrom,
        '\n dateTo ', dateTo,
        '\n chunkCount ', tempDate
    );

    let dataArray = [];
    const data = [];

    try {
      while (tempDate < dateTo) {
        const newdata = await app.service('sensor-data').find({
          query: {
            time: {
              $gte: tempDate,
              $lte: dateTo
            },
            animal_id: {
              $eq: req.body.animal_id,
            },
            $allowFiltering: true,

          }
        });

        const dataArrayy = newdata['result'];
        const readableStream = Readable.from(dataArrayy);
        data.push(...dataArrayy);
        console.log('data added ', data.length);
        const dataArrayLength = dataArrayy.length;
        tempDate = new Date(dataArrayy[dataArrayLength - 1]);
        console.log('data added ', data.length, dataArrayy[dataArrayLength - 1]);
        console.log('data added ', data.length, tempDate);
```

Figure 75 Report charts function in backend side

```
function getLiveOrders() {
  // const event = new Event('DOMContentLoaded');
  axios.post('http://localhost:8080/sensorData/getAll/', {
    from: document.getElementById('start').value.toString() + '.000000+0000',
    to: document.getElementById('End').value.toString() + '.000000+0000',
    animal_id: "1"
  }).then(function (response) {
    // Element.dispatchEvent(event);
    liveOrders = Array(JSON.parse('[' + (response.data.slice(0, -1)) + ']'));
    console.log(liveOrders);
    remyChart.data.labels.push('');
    remyChart.data.datasets.forEach((liveOrders) => {
      dataset.data.push(data.Heart_Rate);
    });
    remyChart.data.labels.pop();
    remyChart.data.datasets.forEach((dataset) => {
      dataset.data.pop();
    });
    remyChart.update();
  })
    .catch(function (error) {
      console.log(error);
    });
}
```

Figure 76 Report charts function in frontend side

4.2.12 Heart rate sensor

Figure 78 is the heart rate sensor screen that measures the heart rate of animals and appears data on the website.

Animals	Pulse rate/minute
Cattle	60 – 90
Young calves	100-120
Horse	28 – 42
Foal up to 1 year	70-80
Sheep	68 - 90
Goat	68 - 90
Pig	60 - 90
Dog	90 - 130
Cat	110 - 130
Chicken	200 - 400

Figure 78 Heart rate sensor screen

```
if (app.io) {  // if a client is connected to node js socketio

    app.io.emit('connected', {text: 'A client connected!'});

  }
  console.log('connecting to 1880....');
  socket2.on('connect', function () {
    console.log('connecteddd');
  });

  socket2.on('stream', async (data) => {
    const looooll = JSON.parse(data);
    const {animal_id, Heart_Rate, SpO2_Rate, Animal_Temperature, Weather, Air_Qualit, Humidity} =
looooll['d'];

    const newRow = {
      animal_id: parseInt(animal_id),
      Heart_Rate: parseInt(Heart_Rate),
      SpO2_Rate: parseInt(SpO2_Rate),
      Animal_Temperature: parseInt(Animal_Temperature),
      Weather: parseInt(Weather),
      Air_Qualit: parseInt(Air_Qualit),
      Humidity: parseInt(Humidity)
    };
```

Figure 80 Heart rate sensor function in backend side

```
const socket = io("http://localhost:8080", { transports : ['websocket'] })
socket.on("connect",()=> {
console.log(`you connected with id: ${socket.id}`)
})

socket.on("stream",(data)=> {
    console.log(data)

    twomyChart.data.labels.push('');
    twomyChart.data.datasets.forEach((dataset) => {
        dataset.data.push(data.Heart_Rate);
    });
    twomyChart.update();
})
```

Figure 79 Heart rate sensor function in frontend side

References:

1. *<saiot3-courseguide-book.pdf>.*
2. Wasson, C.S., *System analysis, design, and development: Concepts, principles, and practices*. Vol. 22. 2005: John Wiley & Sons.
3. Haviluddin, H., *Memahami Penggunaan UML (Unified Modelling Language).* 2011.
4. Aleryani, A.Y., *Comparative study between data flow diagram and use case diagram.* International Journal of Scientific and Research Publications, 2016. **6**(3): p. 124-126.
5. Bell, D., *UML basics: The sequence diagram.* Retrieved July, 2004. **17**: p. 2015.
6. Bell, D., *UML basics Part II: The activity diagram.* IBM Global Services, Rational Software, 2003.
7. Bell, D., *UML basics: An introduction to the Unified Modeling Language.* The Rational Edge, 2003.
8. Merah, E., et al., *Design of ATL rules for transforming UML 2 communication diagrams into buchi automata.* International Journal of Software Engineering and Its Applications, 2013. **7**(2): p. 19-34.
9. Robey, D. and M.L. Markus, *Rituals in information system design.* MIS quarterly, 1984: p. 5-15.
10. Sherwani, N.A., *Algorithms for VLSI physical design automation*. 2012: Springer Science & Business Media.
11. Gawinecki, M., *How schema mapping can help in data integration?—integrating the relational databases with ontologies.* ITC School, Computer Science, XXIII Cycle DII, University of Modena and Reggio Emilia, Italy, 2008.
12. *Cassandra Documentation - Overview.* last visit: 22 jun 2022.
13. *Cassandra - Introduction.* last visit:: 22 jun 2022.
14. *Data Definition.* last visit: 22 jun 2002.
15. Gay, W., *DHT11 sensor*, in *Advanced Raspberry Pi*. 2018, Springer. p. 399-418.
16. Fezari, M. and A. Al Dahoud, *Exploring One-wire Temperature sensor "DS18B20" with Microcontrollers.* Badji Mokhtar Annaba University, University of Al-Zaytoonah Faculty of IT, Jordan, 2019.
17. *Interfacing MAX30100 Pulse Oximeter and Heart Rate Sensor with Arduino.* last visit : 13 jun 2022.
18. *MQ-135 Air Quality Sensor Tutorial.* last visit: 22 jun 2022.
19. Kasbe, M., et al., *An Electronic nose with LabVIEW using SnO2 Based Gas Sensors: Application to test freshness of the fruits.* International Journal of Scientific & Engineering Research, 2015. **6**(4): p. 1977.
20. *PIR Sensor Working Principle.* May 31, 2020.
21. Kumar, S., et al. *Arduino and ESP32-CAM-Based Automatic Touchless Attendance System*. in *Proceedings of the 3rd International Conference on Communication, Devices and Computing*. 2022. Springer.
22. Apoorve, *What is a Servo Motor? - Understanding the basics of Servo Motor Working.* August 1, 2015.
23. Vidhya, R. and S. Sugumar. *IoT based Hybrid energy system using nodeMCU module*. in *International Conference on New Scientific Creations in Engineering and Technology (ICNSCET-19)*. 2019.
24. *Arduino Mega 2560 Rev3.* last visit: 13 jun 2022.
25. *Arduino Uno Rev3.* last visit: 21 jun 2022.
26. Parida, D., *Performing MQTT Communication with ESP8266/NodeMCU using Arduino IDE.* 23 December 2020.
27. *Flow-based Programming.* last visit: 23 jun 2022.
28. Contributors, O.F., *node-RED.* last visit: 23 jun 2022.

ملخص المشروع باللغة العربية

في هذه الدراسة نعمل على إنشاء نظامًا ذكيًا لحماية الحيوانات المهددة بالانقراض، حيث فكرنا في وجود طوق يرتديه الحيوان حول رقبته يحتوي على نظام مراقبة لمكافحة الصيد غير المشروع، ويحتوي أيضًا على نظام لمراقبة صحة الحيوان.

يحل هذا المشروع مشكلة انقراض الحيوانات من خلال مرحلتين:

الأولى: مراقبة كل ما يحدث حول الحيوان من خلال كاميرات تلتقط كل ما يحدث في محيط الحيوان من الظواهر الطبيعية مثل السيول والعواصف أو اقتراب صياد غير شرعي للحيوان.

والثانية: مراقبة صحة الحيوان وحالته الجسدية مثل معدل ضربات القلب ودرجة الحرارة وما إلى ذلك. هذا من خلال بعض أجهزة الاستشعار التي تجمع البيانات.

وآليته بأن يتم توصيل كاميرات وأجهزة استشعار بلوحة Arduino لجمع البيانات وإنشاء اتصال بين وحدة wifi ومنصة iot cloud وإرسال البيانات إليها ليتم تخزينها وتنظيمها، ومن ثم يتم إنشاء اتصال بين منصة iot cloud وnode-red لإعادة هيكلة البيانات وترتيبها، ثم إنشاء اتصال بينها وبين قاعدة البيانات بحيث يتم عرض هذه البيانات المخزنة في قاعدة البيانات على شكل تقارير ورسوم بيانية بواجهة الموقع، وبحيث يتم جمع البيانات لمتابعة حالة الحيوانات المتواجدة في محميات أو غيره دون الحاجة إلى اختلاط الموظفين بشكل يومي مع جميع الحيوانات، وبحيث تنحصر مهمتهم فقط في مراقبة هذا النظام الذكي لتحقيق الأهداف المرجوة منه.

هذا وقد تم تقسيم البحث إلى أربعة مراحل على الوجه التالي:

1. ماهية المشروع، وأهدافه، ومميزاته، والتحديات التي واجهناها.
2. دراسة كيفية جمع واستقبال البيانات الواردة من أجهزة الاستشعار.
3. دراسة كيفية تخزين وحفظ البيانات الواردة في قاعدة البيانات.
4. كيفية عرض البيانات المخزنة على المتصفح.