Ministry of Higher Education
and Scientific Research
*** * ***
University of Carthage
*** * ***
National Institute of Applied Sciences and Technology

INSAT 1996
المعهد الوطني للعلوم التطبيقية و التكنولوجيا
Institut National des Sciences Appliquées et de Technologie

End-of-Year Project

Specialty : Industrial Computing and Automation

Level : 4th Year

Subject:

IoT System for Cow Health Monitoring

Realized by :

HADJ KHELIL Edriss

OUESLATI Niam

KAROUI Majd

KHECHINE Elyes

Supervised by :

Prof. Harbaoui Imen

Academic Year: 2022/2023

Acknowledgments

First and foremost, we would like to express our profound gratitude and humility as we acknowledge and extend our heartfelt appreciation to all those who have provided their invaluable support.

We would like to sincerely acknowledge the dedicated efforts and valuable time contributed by our professor, Ms. Harbaoui Imene, who served as our advisor at the National Institute of Applied Science and Technology. Her unwavering assistance and continuous encouragement have been instrumental in the success of our project.

Additionally, we would like to extend our heartfelt thanks to Ms. Youssef Rabaa for her guidance, constant supervision, and invaluable contributions to our project. Her advice and support have greatly contributed to its development.

Throughout this endeavor, their patience, kindness, and humility have consistently aided us at every step. Their unwavering support has been invaluable and deeply appreciated.

Furthermore, we would like to express our gratitude to Ali Gomri, Director of the APII international cooperation, for his kindness and assistance.

Lastly, we would like to convey our deepest and most sincere gratitude to the members of the jury for graciously accepting the responsibility of evaluating our project and honoring us with their presence.

Contents

List of Figures

List of Tables

Abbreviations

BLE Bluetooth Low Energy

CoAP Constrained Application Protocol

CSV Comma-Separated Values

HTTP HyperText Transfer Protocol

ICT Information and Communication Technologies

I2C Inter-Integrated Circuit

INAT Institut National Agronomique de Tunisie

INSAT Institut National des Sciences Appliquées et de Technologie

IP Internet Protocol

IoT Internet of Things

I/O Input/Output

LSB Least-Significant Bit

LoRa Long Range

ML Machine Learning

MQTT Message Queuing Telemetry Transport

SDK Software Development Kit

SRAM static random access memory

TCP/IP Transmission Control Protocol/Internet Protocol

Wi-Fi Wireless Fidelity

ZigBee Zonal Intercommunication Global-standard, where Battery life is long, which is Economical to deploy, and which exhibits Efficient use of resources

JSON JavaScript Object Notation

FFT Fast Fourier transform

General Introduction

The dairy industry has undergone significant transformations in recent decades, with a decrease in the number of family-owned small-scale farms and the rise of large-scale industrialized farms in response to growing demand. This shift has led to a greater focus on efficiency and profitability in the agriculture sector, facilitated by advancements in IoT, Sensor Technology, and Data Acquisition.

By harnessing these technological advancements, real-time monitoring of various health factors for each animal in a herd has become possible, enabling easy and accurate identification of their individual needs. One critical issue that farmers have always faced is the detection of estrus in milk cows, as it profoundly impacts production. This report aims to address this problem by presenting different factors that reliably predict estrus, introducing a built prototype for detecting these factors and showcasing a mobile application for real-time monitoring.

This project arises from the growing demand for a cost-effective estrus detection solution in the Tunisian market. The report, composed of four main chapters, will present the solution in detail.

Chapter 1: provides an overview of the cow's estrous cycle and estrus, discussing the dairy industry's interest in estrus detection and presenting existing market solutions.

Chapter 2: focuses on the selection of IoT technologies, including sensors, boards, power sources, and servers.

Chapter 3: delves into the design and development of the solution, covering the wiring of different components and showcasing relevant code snippets that ensure proper communication between the components.

Chapter 4: focuses on testing and validation, demonstrating how all the components work together seamlessly.

By addressing the need for an affordable estrus detection solution, this project aims to contribute to the dairy industry in Tunisia. Throughout the report, you will gain comprehensive insights into the solution's development, technology selection, and its practical implementation.

Chapter 1

Project context

1.1 Introduction

In this chapter, we introduced I-Cow-Care project that focuses on precision livestock farming techniques using ICT to provide comprehensive information for dairy farmers. The objective is to revolutionize Tunisian dairy farming by enhancing livestock performance and reducing labor. We discussed the importance of estrus detection in cows, its challenges, and its benefits in reproductive management. Additionally, a market research section compared different cow monitoring solutions to help us have an idea about this market and select the most suitable technological options to build a competitive product especially on the national level.

1.2 Host startup Presentation

I-Cow-Care is a spin-off from the National Institute of Applied Sciences and Technology in Tunis, that has established its offices at CID INSAT and is an integral part of the Trandairy project. Their mission revolves around the application of precision livestock farming techniques, leveraging the power of ICT to provide farmers with comprehensive information for making targeted and effective decisions, thus ensuring sustainable production practices.

The primary objective of the I-Cow-Care project is to revolutionize Tunisian dairy farming by reducing labor and enhancing livestock performance in crucial areas such as production, health, reproduction, and environmental impact. By utilizing modular monitoring solutions specifically tailored to the socio-economic context of Tunisia, I-Cow-Care aims to enable both small and large-scale farmers to track the health and well-being of their dairy cows while implementing sustainable development strategies.

At the core of the project lies the design and implementation of a precise and suitable IoT monitoring solution that is adaptable to the unique agricultural environment in Tunisia. This cutting-edge technology enables real-time data collection, analysis, and feedback, empowering farmers with actionable insights to optimize their farming practices and improve overall herd management.

The I-Cow-Care team comprises a group of exceptional researchers who are at the forefront of their fields. Rabaa Youssef and Imen Harbaoui, both accomplished researchers at INSAT,Naceur M'hamdi, a distinguished researcher at INAT and Sana Dhane, another esteemed researcher at

INAT. Together, this dynamic team drives I-Cow-Care's mission to revolutionize dairy farming practices in Tunisia and beyond.

1.3 Project Scope

1.3.1 General Context

In recent years, the agriculture industry has witnessed a significant transformation with the emergence of Agriculture 4.0 technologies, such as IoT and Machine Learning . These technologies have revolutionized traditional farming practices, paving the way for smart farming solutions that maximize efficiency, productivity, and sustainability.

Dairy is a universal agricultural production: people milk dairy animals in almost every country across the world, and up to one billion people live on dairy farms. It is a vital part of the global food system and it plays a key role in the sustainability of rural areas in particular.

It is a well-known fact that the dairy industry actively contributes to the economies of a number of communities, regions, and countries. An increasing demand worldwide is noticeably emerging, and the industry is globalizing, thus increasing the scope and intensity of the global dairy trade.

In 2011, milk production was estimated at 748.7 million tons, of which 620.7 million tons was cow's milk, produced by 260 million cows. The number of dairy farms depends on the countries and on the farming systems, but it can reach up to millions in some countries. [9]

1.3.2 Problem Statement

Here comes the need for cattle monitoring systems, able to collect data 24 hours a day, seven days a week, providing invaluable information that helps the farm better manage the herd.

"It's an incredibly valuable tool for us," says Chris Szydel, dairy herd manager for Pagel's. "Instead of looking for animals that might need attention, the computer comes up with a list and identifies those animals.The response time with the system is the fastest way that we can identify those animals."

As technology advances, data provides even more insight into the daily life of a cow and what producers can do to make that animal more productive. As a matter of fact, it relies more and more on Machine Learning techniques and Artificial Intelligence to help detect further significant details and give meaning to data that used to be insignificant before.

1.3.3 Estrous Cycle in Cows

The reproductive cycle in cows is known as the estrous cycle and encompasses a series of physiological and behavioral events. After reaching puberty, cows experience the estrous cycle approximately every 21 days, with a range of 17 to 24 days, except during pregnancy, reproductive disease, or hormonal disorders. This cycle prepares the reproductive tract for estrus (the period of sexual receptivity) and ovulation (release of an egg). The estrous cycle can be divided into four distinct parts:

- Proestrus.

- **Estrus.**
- **Metestrus.**
- **Diestrus.**

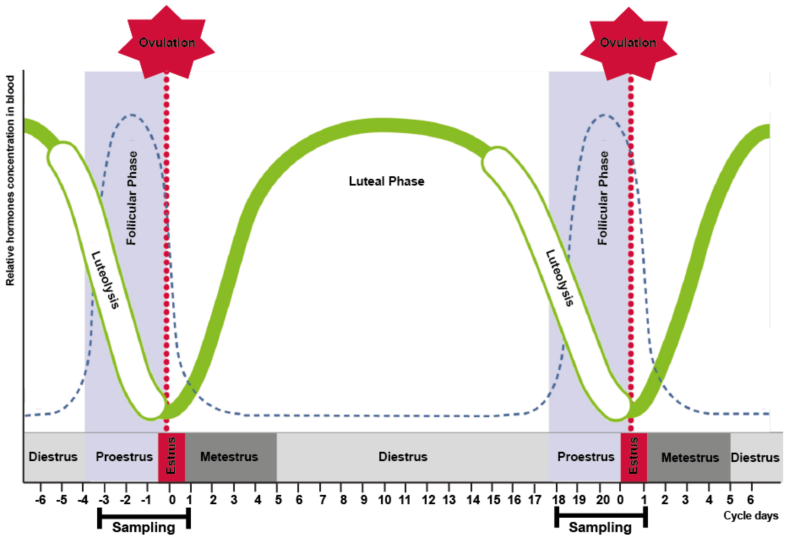


Figure 1.1: Estrous cycle. [7]

Proestrus is a stage that occurs after the regression of the previous cycle's corpus luteum, which is a reproductive gland required for the establishment and maintenance of pregnancy. This phase is characterized by the development of follicles and involves both anatomical and hormonal changes.

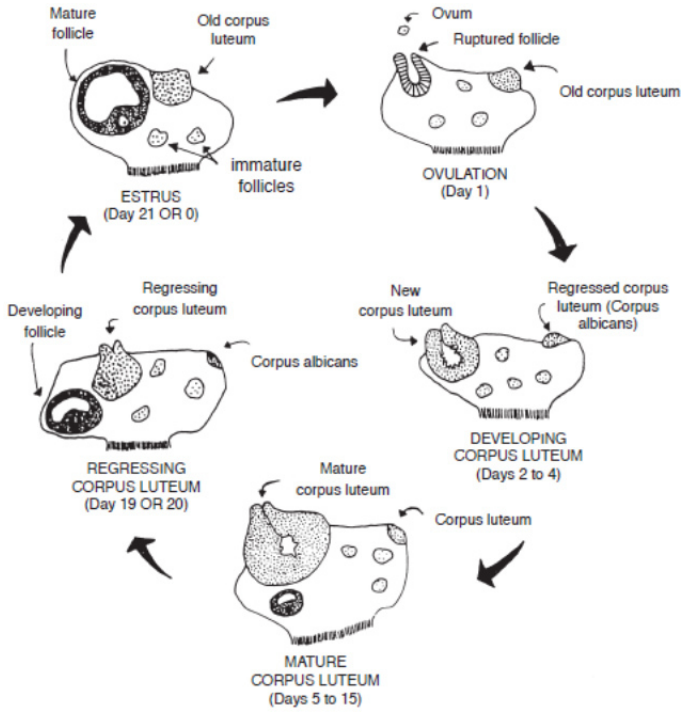


Figure 1.2: The ovarian changes during a normal estrous cycle (21 days) without pregnancy. [7]

Estrus, which follows proestrus, is marked by a significant increase in estrogen levels in the bloodstream. This hormone triggers behavioral signs of estrus, such as cows mounting each other, displaying receptiveness to being mounted, and showing increased activity.

After estrus, there is a 3 to 4-day period known as metestrus. During this stage, the corpus luteum develops under the influence of luteinizing hormone and starts producing larger amounts of progesterone.

1.3.3.1 Estrus

Estrus, also known as heat, is a monthly period in which female cows become receptive to reproduction, displaying specific behaviors that indicate their readiness for mating. This phase typically lasts for approximately 18 hours, although it can be shorter, ranging from 7 to 12 hours in some cases. The accurate detection of estrus is of utmost importance for the success of dairy operations.

1.3.3.2 Signs of Estrus

During estrus, cows experience significant hormonal fluctuations, leading to distinctive behavioral and physical signs. These signs can be easily recognized and interpreted by trained individuals. Some noticeable indicators of heat include:

- Attempting to mount other cattle.
- Displaying a receptive stance for being mounted by other cattle.
- An increase in body temperature.
- Exhibiting interest in and sniffing other females.
- Following other females.
- Decreased appetite.
- Heightened restlessness and activity.
- Swelling and reddening of the vulva.
- Presence of clear vaginal mucous discharge.

These evident changes in behavior and physical characteristics serve as valuable cues to identify when a cow is in estrus, allowing for appropriate timing of breeding activities and optimizing reproductive efficiency in dairy operations.

1.3.3.3 Challenges of Estrus Detection

Heat detection for cows presents significant challenges. Accurately identifying and detecting when cows are in heat is essential but can be quite difficult. Subtle behavioral changes or silent heats can make it challenging to detect estrus, while variations in individual cow behavior and environmental factors, such as temperature and social interactions, further complicate the process. Limited observation time and the presence of false indicators, such as illness or dietary changes, can result in missed or incorrect identification of estrus.
Moreover, the narrow time window for heat detection, typically lasting around 18 hours with variations of 7 to 12 hours, adds pressure to accurately time artificial insemination. To overcome these challenges, effective strategies including training programs, technological aids, and consideration of cow-specific factors and environmental conditions are necessary to optimize heat detection and maximize milk production in dairy farming operations.

1.3.3.4 Benefits of Estrus Detection

Estrus detection, or heat detection, plays a crucial role in reproductive management of dairy cows, offering several benefits to farmers and the overall herd management. Accurate and timely detection of estrus allows for optimal timing of insemination, which can significantly improve the success rates of artificial insemination. For example, a study in Florida found that using automated activity monitors to detect estrus increased the conception rate by 5.2% after first service. By identifying cows in heat, farmers can ensure the breeding process occurs at the most fertile period, maximizing the chances of conception and reducing the need for repeated insemination attempts.

According to a review, effective estrus detection can reduce the number of insemination attempts by 10 to 20% . Efficient estrus detection also helps in preventing reproductive problems, such as prolonged calving intervals and increased culling rates. Additionally, early detection of estrus enables proactive management, allowing farmers to closely monitor the cow's health and well-being during this critical reproductive phase. Overall, effective estrus detection empowers farmers to make informed decisions, improve breeding outcomes, enhance overall herd productivity, and ultimately contribute to the profitability and sustainability of the dairy farming operation.

1.4 Market Products Research

In order to make an informed decision regarding the selection of a cow monitoring solution, it is essential to compare and evaluate various options available in the market. The following table presents a comprehensive comparison of different cow monitoring solutions based on their features and criteria.

Each solution offers unique capabilities and functionalities, ranging from motion pattern monitoring, accessibility, detection sensitivity, sensor configuration, battery life, range, pricing, heat detection, rumen monitoring, calving detection, real-time monitoring, cloud-based systems, actionable recommendations, scalability, cost-effectiveness, and sustainability-focused approaches.

By examining the table, one can gain valuable insights into the strengths and limitations of each solution, enabling farmers and stakeholders to choose the most suitable option that aligns with their specific requirements and goals in smart cow monitoring.

Criteria	Afimilk	Nedap	Smartbow	Smaxtec Cowcontrol	MooMonitor+	CowManager
Motion Pattern Monitoring	Yes	No	No	No	No	Yes
Accessiblity	Yes	Yes	Yes	Yes	Yes	Yes
Detection Sensitivity	95%	95%	N/A	N/A	N/A	N/A
Battery Life	Multi-year	8-10 years	N/A	Lifetime	N/A	Long-lasting
Range	200-800m	Long-range	N/A	N/A	N/A	N/A
Price	Varies	Varies	$2,500 + $69/cow	$89 + $2.49/cow/month	140 (collar) + €4,500 (base station) + €5/cow/year	Varies
Heat Detection	No	Yes	Yes	Yes	Yes	Yes
Rumen Monitoring	No	No	Yes	Yes	Yes	Yes
Calving Detection	No	No	No	Yes	Yes	Yes
Real-time Monitoring	No	Yes	Yes	Yes	Yes	Yes
Cloud-based System	No	No	No	No	Yes	Yes
Scalability	Yes	Yes	No	Yes	Yes	Yes
Cost-Effectiveness	Varies	Varies	No	Yes	No	Yes

Table 1.1: Comparison of Cow Monitoring Solutions based on Features and Criteria

1.5 Gantt Chart

To ensure a well-planned and organized work plan, we used the GANTT chart presented in Figure 1.3, which will help break the project into defined steps for a clear overview of its development.

- The first phase is dedicated to the study of the current market and the feasibility of the project.
- The second phase focused on the study of different sensors and algorithms
- The third phase was the practical testing phase to narrow down our choices.
- The final phase was to validate our software on the real-life prototype.

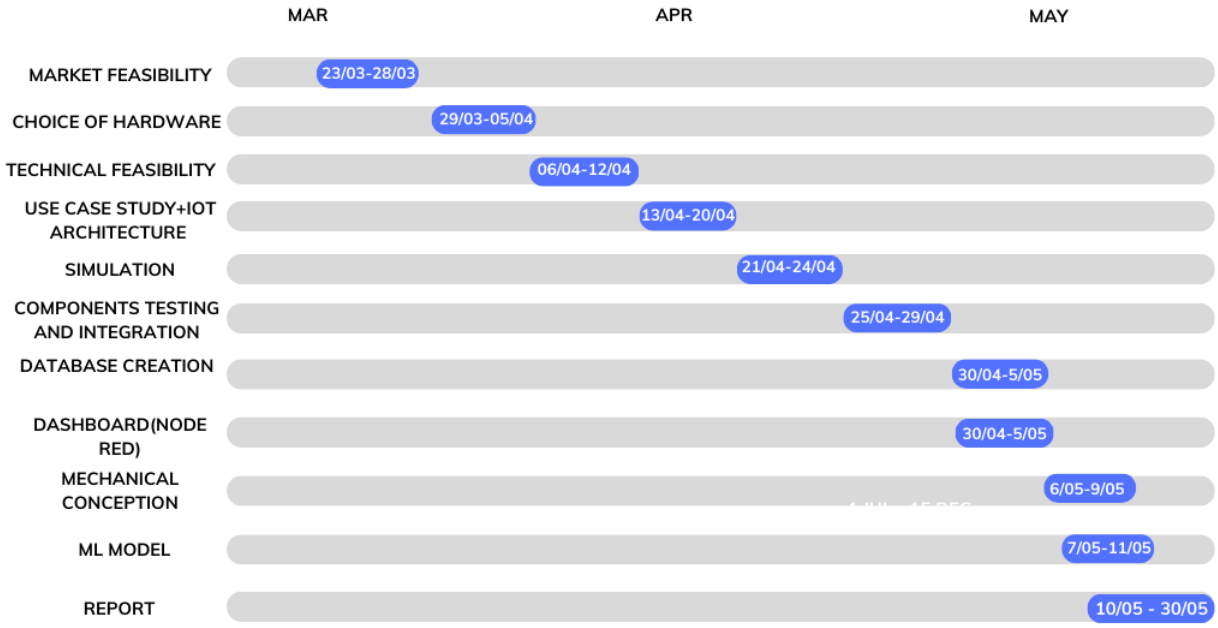


Figure 1.3: Gantt Chart

1.6 Conclusion

It is obvious that the identification of the estrus cycle in cows is of utmost importance and offers significant benefits to dairy farming operations. However, it is a challenging task due to subtle behavioral changes, variations in individual cow behavior, and the narrow time window for detection. These challenges led to the development of monitoring systems specifically designed to track and monitor the estrus cycle. The following chapter will present our monitoring solution that is low-cost and more in synch with the local market in Tunisia.

Chapter 2

IoT Technologies Selection

2.1 Introduction

Our cow monitoring IoT solution requires a careful selection of IoT technologies to ensure optimal performance, compatibility, and cost-effectiveness. In this chapter, we present the process and considerations involved in choosing the appropriate IoT technologies for our project. We begin by discussing the criteria used to evaluate development boards, including performance, price, and availability.

Then, we present our approach to selecting sensors, focusing on the temperature and accelerometer sensors. Factors such as sensor type, range of detectable values, sensitivity, and power consumption were considered to make informed decisions after the evaluation of various options from our research on the Tunisian market.

This chapter lays the foundation for the subsequent implementation and integration of IoT technologies into our cow monitoring system.

2.2 Developement Board

The choice of the development board was based on multiple criteria: Performance: We need a board with a multi-core microprocessor for simultaneous tasks, fair internal storage size, and I/O ports suitable for our sensors and power network. Price: We need a product with a competitive price to meet our budget and our scalability goals. Availability: The product must be available in the local electronics stores. The candidates for our benchmarking that fairly meet our criteria are resumed in this table:

Name	Pros	Cons	Price	Characteristics
Beetle ESP32-C3	Low power consumption, making it suitable for battery-powered applications. Dual-core microcontroller running up to 160 MHz. Compact size and low cost.	Single-band Wi-Fi (2.4GHz only). Limited availability of support and community resources compared to other ESP32-based boards.	52DT	Operating Voltage: 3.3V Type-C. Input Voltage: 5V DC. Operating Current: 25mA Maximum. Charging Current: 400mA. Operating Temperature: -40∼105℃. Size: 25 x 20.5mm.
ESP32 DevKitC V4 ESP32-WROOM-32D	Dual-core microcontroller running up to 240 MHz. More memory (520KB SRAM and 4MB flash memory). Dual-band Wi-Fi (2.4GHz and 5GHz). Wide availability of support and community resources.	Consume slightly more power compared to the Beetle ESP32-C3.	42DT	USB to Serial Port Chip: CP2102-GMR. Operating Voltage: DC 5V. Operating Current: 80mA. Current Supply: 500mA. Size: 55mm*26mm
ESP32 CH340C CP2102	Dual-core microcontroller running up to 240 MHz. More memory options (520KB SRAM and 4MB flash memory). Dual-band Wi-Fi (2.4GHz and 5GHz).	Limited availability of support and community resources. Consumes more power compared to the Beetle ESP32-C3.	39DT	USB to Serial Port Chip: CP2102. Operating Voltage: DC 5V. Operating Current: 80mA. Current Supply: 500mA. Size: 52*28mm. Weight: 9.5g.

Table 2.1: Comparison of Development Boards based on Features and Criteria

We finally ended up choosing the ESP32 DevKitC V4 [12] ESP32 WROOM-32D [4] because it met our needs from a fair processing power(Flash memory, SRAM, Dual Core..), great community support and resources, low energy consumption and the compatibility with our power source, sensors, and communication protocol.

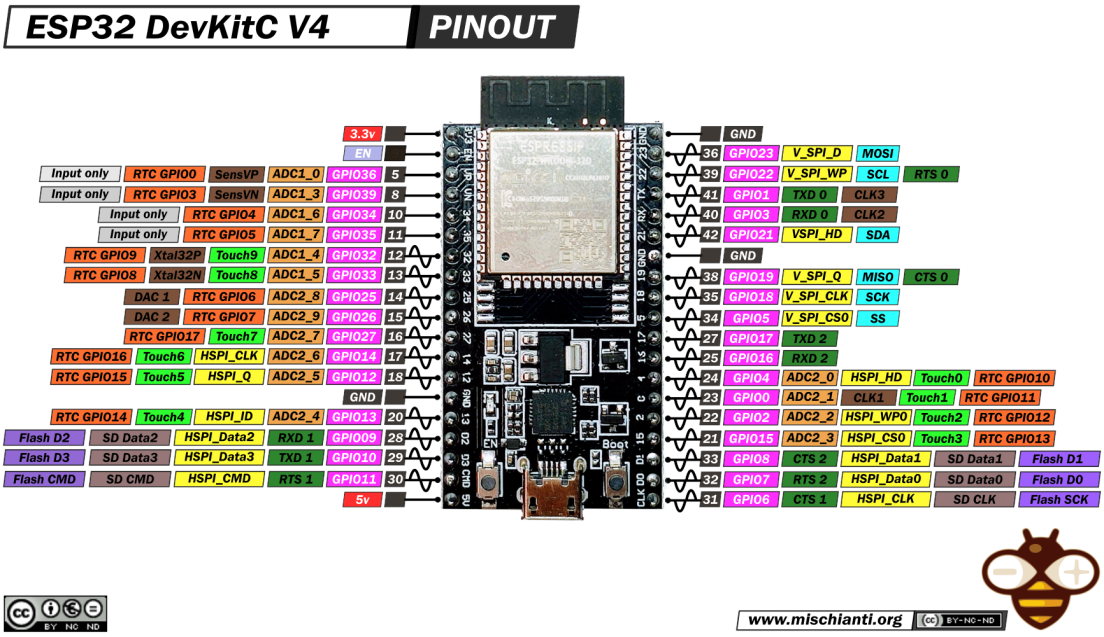


Figure 2.1: ESP32 DevKitC V4 [12] ESP32 WROOM-32D [4] Pinout

2.3 Sensors

2.3.1 Sensors Benchmarking

Our research in the global market of cow monitoring IoT products was essential in assessing and determining our specific needs and requirements for our project. We carefully examined a variety of products and their corresponding sensors and features. The comparison table provided offers a comprehensive overview of different cow monitoring solutions available worldwide.

Factors such as product weight, battery life, range, built-in sensors, output data, and frequency of data collection were thoroughly evaluated. This extensive analysis allowed us to make informed decisions regarding the selection of sensors and features that align with our project objectives. By understanding the existing market landscape, we gained valuable insights that will contribute to the successful implementation of our cow monitoring system.

Product	Company (Parent Company)	Battery Life	Range (m)	Built-In Sensors	Output Data	Frequency
CowScout Neck	GEA Farm Technologies, Inc., DE	5 years	100-1000	Accelerometer	Activity, Inactivity, Ruminating time, Eating time	Every 2 h
Axel Collar	Medria Inc., FR	-	1000	Accelerometer	High activity, Inactivity, Ruminating time, Eating time, Lying time, Standing time	-
Smart Collar	HerdInsights, IE	5 years	-	Accelerometer	Activity, Inactivity, Ruminating time, Eating time, Heat index	Every hour
MooMonitor+	Dairy Master, IE	10 years	1000	Accelerometer	High activity, Activity, Low activity, Inactivity, Ruminating time, Eating time	Every hour
SCR HR-LD/SCR HR-LDn	SCR Engineers Ltd. (Allflex Europe SA), IL	7 years	200 × 500*	Accelerometer, Microphone	Activity, Ruminating time, Heat index	Every 2 h
AfiCollar	Afimilk Agricultural Cooperative Ltd., IL	-	200–800	Accelerometer	Activity, Ruminating time, Eating time	-
Milkrite|InterPuls Neck Tag	milkrite | InterPuls, IT	-	75–500	Accelerometer	Activity, Ruminating time, Eating time, Location	-
Smarttag Neck	Nedap livestock management, NL	10 years	75	Accelerometer	Activity, Inactivity, Ruminating time, Eating time, Eating bouts, Location	Continuously
Smarttag Neck-/All in One	CRV international B.V., NL	-	-	Accelerometer	Inactivity, Ruminating time, Eating time, Not eating time, (Location)	Continuously
Activity meter system	DeLaval International AB Inc., SE	10 years	200	Accelerometer	Activity, Heat index	Every hour
HeatSeeker II Neck	BouMatic LLC, US	7 years	100–750	Accelerometer	Activity	Every 2 h

Table 2.2: Cow Monitoring IoT Products' Sensors Benchmark

2.3.2 Accelerometer Sensor

Our selection of the accelerometer sensor was based on a thorough evaluation of various factors, including sensor type, detectable acceleration values, sensitivity, output resolution, input voltage, supply current in operating mode, lowest power mode, sleep mode, and interface type. Our primary requirement was an accelerometer capable of detecting both ±2g and 4g acceleration values, with high sensitivity (low LSB/g values) in each range. Considering the significant impact on battery life, we carefully examined the full power mode current consumption.

Criteria	**ADXL335**	**MMA7455L**	**LIS2DH**	**MPU-6500**	**LIS3DH**
Manufacturer	Analog Devices	NXP	Gravity	TDK	Adafruit
Pricing (TND)	12,000	19,000	20,469	30,000	38,000
Operating Supply Current	140µA	490µA	11µA in 50 Hz	450µA (accel only)	11µA in 50 Hz
Lowest Power Mode Current	23 µA	-	2 A	3.4mA (all)	2 A
Sleep Mode Current	0.1 µA	10µA	0.5µA	6 A	0.5µA
Output Resolution	Fixed 10-bit	8-bit (2g,4g), 10-bit (8g)	8-bit (low power), 10-bit (normal power), 12-bit (high resolution)	16-bit	8-bit (low power), 10-bit (normal power), 12-bit (high resolution), 16-bit output available

Table 2.3: Comparison of Readily Available Accelerometer Sensors

Ultimately, we identified two suitable options: the LIS2DH [5] from Gravity and the LIS3DH [19] from Adafruit. Both sensors met our requirements, but the LIS3DH offered the additional advantage of 16-bit output, which we deemed unnecessary for our specific use case. Moreover, considering the price difference of 85% between the two, we decided to select the LIS2DH. However, for future production, we may consider switching to the LIS3DH due to its greater availability in the market.

Figure 2.2: Chosen Accelerometer Sensor: DFRobot Gravity LIS2DH [1]

2.3.3 Temperature Sensor

Our selection of the temperature sensor was based on several key factors, including the temperature measurement range, measurement accuracy and resolution, active and standby current consumption, and the sensor's capability to operate without physical contact.

Please refer to the following table for a comprehensive comparison of readily-available temperature sensors:

Part	**AHT10**	**AM2301**	**SHT20**	**MAX30205**	**MLX90614**
Sensor	Temperature + humidity	Temperature + Humidity	Temperature + Humidity	Human Body Temperature	IR Thermometer
Digital / Analog	Digital	Digital	Digital	Digital	Digital
Temperature Range	-40°C to +85°C	-40°C to +80°C	-40°C to +125°C	0°C to +50°C	-40°C to +125°C
Temperature Accuracy	±0.3°C	±0.3°C	±0.3°C	±0.1°C	0.5°C
Temperature Resolution	0.01	0.1 / 16 bit	0.01 / 14 bit	16-Bit (0.00390625°C)	0.02
Active Current Cons. (max)	23 µA	500 µA	330 µA	925A	2.5mA
Standby Current Cons. (max)	0.25 µA	15µA	0.4µA	3.5µA	2mA
Interface	I2C	1-Wire	I2C	I2C	I2C
Power supply Range	1.8V to 3.6V	3.3V to 5.2V	2.1V to 3.6V	2.7V to 3.3V	2.6V to 3.6V
Pricing (TND)	9.100	36.000	45.000	49,000	49,000

Table 2.4: Comparison of Readily-Available Temperature Sensors

After careful evaluation, we decided to adopt the MLX90614 [6] as our temperature sensor of choice. One of the primary reasons for this selection is its ability to accurately measure object temperature (within +0.5°C) without the need for physical contact. Moreover, the MLX90614 offers the added advantage of measuring ambient temperature, which is highly relevant for our specific use case.

Figure 2.3: Chosen Temperature Sensor: GY-906 MLX90614 [6]

It is worth mentioning that the sensor's current consumption and price were potential drawbacks; however, we plan to address these concerns through effective energy management and optimization techniques, while also exploring the potential for reduced pricing when purchasing in bulk.

2.4 Server

Our choice of server was based on three essential factors: scalability, security, and reliability. We prioritized scalability to accommodate our growing IoT network, selected a server with robust security measures to protect sensitive data, and ensured uninterrupted connectivity through a reliable server choice.
The Dell PowerEdge R230 [17] was our final choice and these are its main specifications:

- Performance: The server is powered by an Intel Xeon E3-1220 v6 processor, which offers good performance for tasks such as cow monitoring.
- Storage: The server comes with 2TB of storage.
- RAM: With 8GB of RAM, the server possesses sufficient memory to efficiently run the required applications. required.

2.5 Cloud

We evaluated various cloud services and thoroughly studied their pricing structures and options:

- AWS provider: starts at around $10 per month for a basic t2.micro instance with 1vCPU and 1GB RAM.
- Azure: starts at around $15 per month for a basic B1s instance with 1 vCPU and 1GB RAM.
- Google Cloud Platform: starts at around $10 per month for a basic f1-micro instance with 1 vCPU and 0.6GB RAM.

This analysis has provided us with valuable information for potential future migration to the cloud. By considering the pricing and available options, we will be able to make the right decisions regarding the feasibility and benefits of transitioning to cloud services.

2.6 Power Source

After careful consideration of various power supply options for our project, we have selected three final contenders: the CR123A 3V 1300mAh [2], ANSMANN 3V 1500mAh [16], and Huiderui 3V 1600mAh [18] batteries. These options have been chosen for their consistent voltage output of 3V, ensuring reliable and stable performance throughout our project's operation. The utilization of lithium batteries offers several advantages, specifically their ability to effectively utilize over 90% of their capacity until their voltage drops below 2.7 volts. Once the voltage reaches 2.55 volts, the battery is considered nearly depleted, indicating the need for replacement.

Moreover, lithium batteries excel in meeting the high short-term power demands of Wi-Fi operations, guaranteeing seamless functionality without encountering significant issues. For instance, lithium batteries can remain in standby mode for several years, depending on the frequency of wake-ups and the usage of Wi-Fi or Bluetooth. Furthermore, these batteries exhibit minimal self-discharge and demonstrate reliable performance even under extreme temperatures, such as hot

weather conditions. These characteristics establish lithium batteries as the preferred and optimal choice for our project's power supply requirements. We have chosen the CR123A 3V 1300mAh [2] battery as it best aligns with our project's requirements.

Figure 2.4: CR123A 3V 1300mAh battery [2]

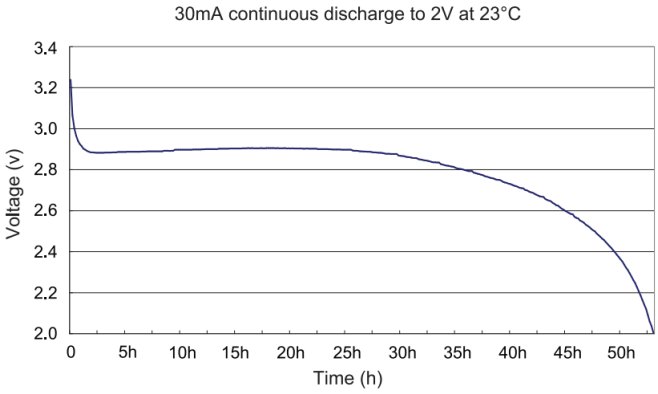


Figure 2.5: CR123A 3V 1300mAh battery discharge chart. [2]

2.7 Conclusion

At the end of our careful selection, we were able to collect the tools we need to make the cow monitoring system with the most adequate equipment combining performance, stability, low cost, and low energy consumption. In the next chapter, we'll be developing our IoT solution while tackling both hardware and software aspects.

Chapter 3

Design and Development

3.1 Introduction

In this chapter, we present the design and development of our Cow Monitoring IoT solution. Our solution incorporates various components and technologies to enable efficient monitoring and analysis of cow behavior and health. This chapter provides an overview of the key aspects of our design and development process, highlighting important considerations and choices made along the way.

We begin by discussing the selection of Firebase as our data storage and retrieval platform, leveraging its real-time database feature. We also outline the integration of the Firebase Arduino Client Library for ESP32, which facilitated the smooth and efficient transmission of accelerometer and temperature sensor data. Additionally, we describe the extraction of data from Firebase as a structured CSV file, which serves as the foundation for training machine learning models. Furthermore, we elaborate on the selection of Node-RED as our dashboard development tool, emphasizing its intuitive visual programming interface and seamless integration with MQTT.

Lastly, we touch upon the role of machine learning in enhancing cow monitoring devices and outline our approach to training and deploying machine learning models using the Edge Impulse platform. Through this chapter, we present a high-level overview of the design and development process, setting the stage for detailed explanations in subsequent sections.

3.2 Communication Protocols

3.2.1 Network Protocols

The choice of the Network Protocol was based on criteria related to the environment, stability, and data speed transfer. We also needed a range of approximately 100m radius to cover the whole barn space.

In the following table, we have collected the features of the most used network communication technologies used in IoT.

Technology	Frequency	Data Rate	Range	Power Usage	Cost
3G/4G	Cellular Bands	10 Mbps	Several miles	High	High
Bluetooth/BLE	2.4GHz	1,2,3 Mbps	∼120 m	Low	Low
Wi-Fi	2.4GHz, 5GHz	0.1-54 Mbps	∼100 m	Medium	Low
LoRa	subGHz	<50 kbps	1-5 Km	Low	Medium
ZigBee	2.4GHz	250 kbps	∼150 m	Low	Medium

Table 3.1: Network Protocol Technologies Comparison

After taking into consideration multiple constraints, we ended up choosing Wi-Fi at our network protocol as it meets our needs of range (>100m), data rate reaching 54 Mbps, Moderately low power usage, compatibility with our development board, and low cost.

3.2.2 Data Protocols

When evaluating data protocols for our cow monitoring device, we must consider their specific characteristics and how they align with our requirements. Here, we examine the differences between MQTT, CoAP, and HTTP protocols, focusing on their suitability for cow monitoring applications.

MQTT is a messaging protocol based on a publish-subscribe model. It offers lightweight and efficient data transmission, with low overhead and bandwidth requirements. This makes it ideal for devices with limited power and bandwidth resources. It is commonly used in scenarios where low power consumption and efficient communication are crucial.. However, MQTT lacks built-in security features and requires additional encryption measures for data protection

CoAP is an application-layer protocol designed for resource-constrained devices in IoT applications. It follows a **RESTful!** architecture, enabling easy integration with web-based systems. CoAP is characterized by its lightweight nature and efficient data exchange. While it offers similar advantages to MQTT in terms of low overhead and power consumption, its range and scalability might be more limited. CoAP is well-suited for scenarios that require integration with web services and efficient resource management.

HTTP is a widely used protocol for web communication. It operates over TCP/IP and provides reliable and secure data transmission. However, HTTP is not specifically designed for resource-constrained devices and can be relatively resource-intensive. It might not be the most suitable choice for cow monitoring applications where power efficiency and low data overhead are essential.

After careful consideration of these data protocols, we have selected MQTT as the most suitable option. MQTT aligns closely with our project's specific requirements and objectives. Its lightweight and efficient nature make it ideal for devices with limited power and bandwidth resources, ensuring

optimal performance in our cow monitoring application. Additionally, MQTT's publish-subscribe model allows for seamless data transmission and easy integration with other systems. By choosing MQTT, we are confident that our cow monitoring device will operate efficiently and effectively, meeting our project's goals.

3.2.3 MQTT Basic Concepts

MQTT consists of several key components that work together to facilitate efficient communication between devices and applications. Understanding these components is essential for harnessing the power of MQTT effectively.

Client: A client in MQTT refers to any device or application that can connect to an MQTT broker. Clients can act as publishers, subscribers, or both. They establish a connection with the MQTT broker and interact with it by sending and receiving messages.

Broker: The MQTT broker serves as the central hub in the MQTT messaging system. It is responsible for receiving messages published by clients and delivering them to interested subscribers. The broker manages client connections, handles subscriptions, and ensures reliable message delivery.

Topic: Topics play a crucial role in MQTT communication. They are string identifiers that categorize messages into specific subjects or categories. Publishers send messages to specific topics, and subscribers express their interest in receiving messages from particular topics. Topics follow a hierarchical structure, allowing for organized and flexible message routing.

Publish-Subscribe Model MQTT operates based on a publish-subscribe messaging model. Publishers are clients that send messages to specific topics without needing knowledge about the subscribers. Subscribers, on the other hand, express their interest in specific topics and receive messages published to those topics. This decoupling of publishers and subscribers enables scalable and dynamic communication.

Messages: Messages are the units of information exchanged in MQTT. They consist of a payload (the actual data) and optional metadata, such as the topic to which the message belongs. Publishers publish messages to topics, and subscribers receive messages from the topics they have subscribed to. Messages can carry various types of data, such as sensor readings, control commands, or application-specific information.

Clients connect to the MQTT broker using a unique client identifier. Publishers send messages to specific topics of their choice, without knowing who the subscribers are. The broker receives the published messages and distributes them to subscribers who have subscribed to those topics. Subscribers receive the messages and process them accordingly. The broker manages client connections, handles subscriptions, and ensures reliable message delivery between clients.

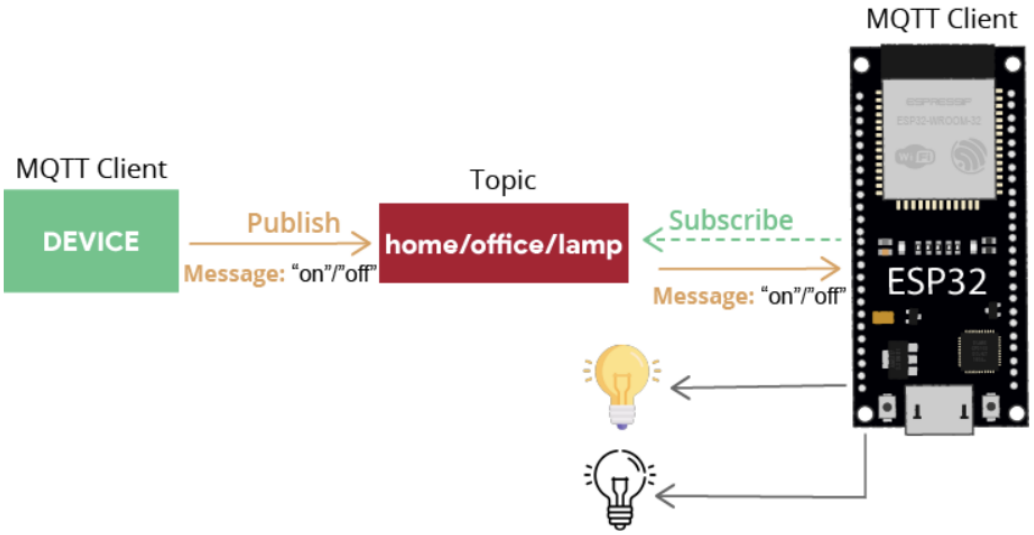


Figure 3.1: MQTT scenario of turning on a lamp in your home office . [14]

3.2.3.1 MQTT Broker Choice

In our quest to select the most suitable MQTT broker for our project, we have carefully evaluated various options and ultimately decided to adopt Mosquitto. Mosquitto stands out as an exemplary choice due to its open-source nature, well-established community, and widespread adoption. These factors guarantee extensive documentation and a wealth of knowledge that can significantly aid in configuring and utilizing the broker effectively.

One key attribute that solidified our decision is Mosquitto's reputation for being lightweight and highly efficient. Its implementation has been optimized to accommodate both resource-constrained devices and demanding scenarios with remarkable efficiency. With the ability to handle a large number of concurrent connections and messages seamlessly, Mosquitto ensures smooth and reliable communication among MQTT clients.

3.3 Hardware Developement

3.3.1 Electronic Scheme

Here is the simplified electronic scheme of the project using the ESP32 WROOM 32D, the temperature sensor MLX90614, the accelerometer LIS2DH and the lithium battery CR123A. We used the I2C protocol to connect both sensors' data directly to the internal memory of the ESP32.

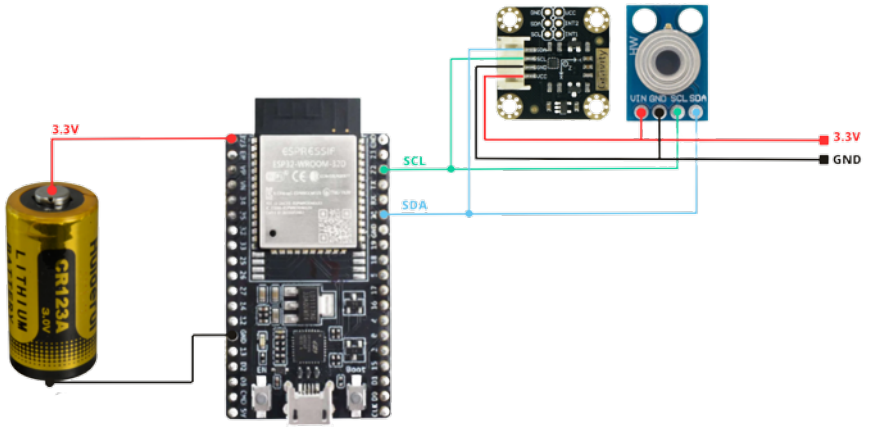


Figure 3.2: IoT Cow Health Monitoring Solution Electronic Scheme

3.3.2 Power Management

Optimizing energy consumption is one of the most important aspects of IoT projects. Thus, after choosing the CR123A Lithium battery, we simulated the power usage of our solution using an online battery life calculator. As a matter of fact, For battery-powered IoT sensor systems that transmit data wirelessly, there are often four unique operating modes: Sleep, Data Collection, Data Processing, and Data Transmission. The system will cycle through each of these modes with each mode requiring a different amount of power.

This calculator will take your project's battery capacity and determine its lifetime based on the following parameters:

- The amount of time the device spends in Sleep Mode
- How much current it consumes in Sleep Mode
- The amount of time the device spends transmitting data
- How much current it consumes when transmitting data
- The amount of time the device spends collecting data
- How much current it consumes when collecting data
- The amount of time the device spends processing data
- How much current it consumes when processing data

To find battery lifetime, divide the battery capacity by the average device current consumption over time. You can then break it down into where:

$$B_l = \frac{B_c}{\frac{I_s t_s}{t_s + t_t + t_c + t_p} + \frac{I_t t_t}{t_s + t_t + t_c + t_p} + \frac{I_c t_c}{t_s + t_t + t_c + t_p} + \frac{I_p t_p}{t_s + t_t + t_c + t_p}}$$

- B_l: Battery lifetime in hours
- B_c: Battery capacity in mAh
- I_s: Device current consumption when in Sleep Mode in mA
- I_t: Device current consumption when transmitting data in mA
- I_c: Device current consumption when collecting data in mA
- I_p: Device current consumption when processing data in mA
- t_s: Time spent in Sleep Mode in seconds (per cycle)
- t_t: Time spent transmitting data in seconds (per cycle)
- t_c: Time spent collecting data in seconds (per cycle)
- t_p: Time spent processing data in seconds (per cycle)

The calculations show that the battery could power our collar for more than 3400 hours equivalent to more than 4 months.

3.3.3 3D Mechanical Design

Before making our first collar prototype, using the 3D mechanical modeling software SolidWorks, we prepared a case that represents a support for the development board, the sensors, and the belt as well as to uncover some mechanical constraints related to space and accessibility after assembling all the parts and components.

Figure 3.3: Collar 3D Design

3.4 Software Development

3.4.1 Embedded C

We chose to work with Embedded C since it's a widely adopted programming language in the embedded systems domain that offers several advantages for IoT projects like its low-level approach, which enables direct hardware access and efficient resource utilization. This is essential for optimizing the performance and power consumption of the ESP32.

We also developed our library to help us modulate our script, for easier tractability and debugging and to make it easier to share and modify. The library contains the functions and the variables needed for Wi-Fi and MQTT connectivity, Sensors data access, and Database publishing...

3.4.2 Wi-Fi Connectivity

To allow the ESP32 to connect to Wi-Fi, we used the following function that uses the Wi-Fi credentials to connect to Wi-Fi and print out the host name and the IP address.

```
1
2  void setup_wifi() { //const char* ssid,const char* password
3    delay(10);
4    Serial.print("\nConnecting to ");
5    Serial.println(ssid);
6
7    WiFi.mode(WIFI_STA);
8    WiFi.begin(ssid, password);
9
10   while (WiFi.status() != WL_CONNECTED) {
11     delay(500);
12     Serial.print(".");
13   }
14   randomSeed(micros());
15   Serial.println("\nWiFi connected\nIP address: ");
16   Serial.println(WiFi.localIP());
17 }
18 }
```

3.4.3 MQTT Connectivity

To connect to the MQTT broker and be able to publish data to the dashboard or database through topics for further analysis as well as subscribe and receive information, we used the functions as cited below.

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "Vache No : ";   // Create a random client ID
    clientId += String(1);
    // Attempt to connect
   if (client.connect(clientId.c_str(), MQTT_username, MQTT_password)) {
      Serial.println("connected");
      client.subscribe("test1");   // subscribe the topics here

    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");   // Wait 5 seconds
         before retrying
      delay(5000);
    }
  }
}
```

```

void callback(char* topic, byte* payload, unsigned int length) {
  String incommingMessage = "";
  for (int i = 0; i < length; i++) incommingMessage+=(char)payload[i];

  Serial.println("Message arrived ["+String(topic)+"]"+incommingMessage);
}
```

```

void publishMessage(const char* topic, String payload , boolean
   retained){
  if (client.publish(topic, payload.c_str(), true))
      Serial.println("Message publised ["+String(topic)+"]: "+payload);
}
```

3.4.4 Sensors Library Tools

For data acquisition, we used some libraries to facilitate access to data in the sensors. As we are using the I2C protocol, we need to access the ESP32 data register using the convenient addresses. Below we cited some of the functions we used to extract data.

```
int8_t DFRobot_LIS2DH12::init(uint8_t range, uint8_t sensorAddress)
{
    int8_t ret = 0;
    this->sensorAddress = sensorAddress;

    setRange(range);
    Wire.beginTransmission(sensorAddress);
    ret = Wire.endTransmission();
    if(ret != 0){
       ret = -1;
    }else{
        uint8_t ctrl_reg_values[] = {0x2F, 0x00, 0x00, range, 0x00,
           0x00};
        ret = (int8_t)writeReg(0xA0, ctrl_reg_values,
           sizeof(ctrl_reg_values));
    }
    return ret;
}
```

```
bool Adafruit_MLX90614::begin(uint8_t addr, TwoWire *wire) {
  _addr = addr; // needed for CRC
  if (i2c_dev)
    delete \acs{I2C}_dev;
  \acs{I2C}_dev = new Adafruit_I2CDevice(addr, wire);
  return \acs{I2C}_dev->begin();
}
```

```
void DFRobot_LIS2DH12::readXYZ(int16_t &x, int16_t &y, int16_t &z)
   //read x, y, z data
{
    uint8_t sensorData[6];
    readReg(0xA8, sensorData, 6);
    x = ((int8_t)sensorData[1])*256+sensorData[0]; //return values
    y = ((int8_t)sensorData[3])*256+sensorData[2];
    z = ((int8_t)sensorData[5])*256+sensorData[4];
}
```

```
double Adafruit_MLX90614::readObjectTempC(void) {
  return readTemp(MLX90614_TOBJ1);
}

double Adafruit_MLX90614::readAmbientTempC(void) {
  return readTemp(MLX90614_TA);
}
```

3.4.5 MQTT Broker Configuration

Configuration plays a pivotal role in tailoring Mosquitto to our specific requirements. Here are the aspects we focused on during the configuration process:

Installation: Mosquitto can be effortlessly installed across various operating systems, including Linux, Windows, and macOS. In our case, we have selected mosquitto version 2.0.15 for windows x64.

Modifying the configuration file: Mosquitto v2 has brought forth significant changes that have impacted the configuration process. To ensure the proper functioning of the Mosquitto broker, we needed to manually modify the configuration file. One crucial modification we made was to the listener configuration for port 1883.

In the earlier versions of Mosquitto, the default configuration allowed anonymous connections, meaning that clients could connect to the broker without providing any credentials. However, in Mosquitto v2, the default behavior has changed, and anonymous connections are no longer permitted by default. To allow anonymous connections, we explicitly set the "allow_anonymous" parameter to true in the configuration file.

This modification ensures that clients without authentication credentials can still establish connections with the Mosquitto broker. It can be particularly useful in scenarios where certain clients or devices do not require authentication but still need to communicate using MQTT.

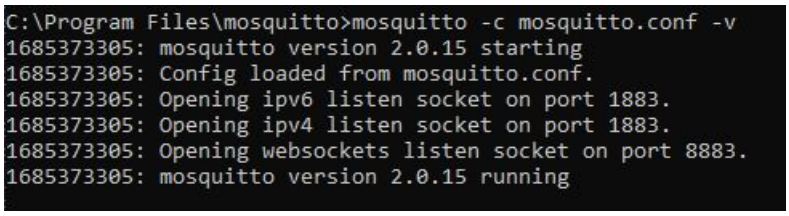


Figure 3.4: Mosquitto start command

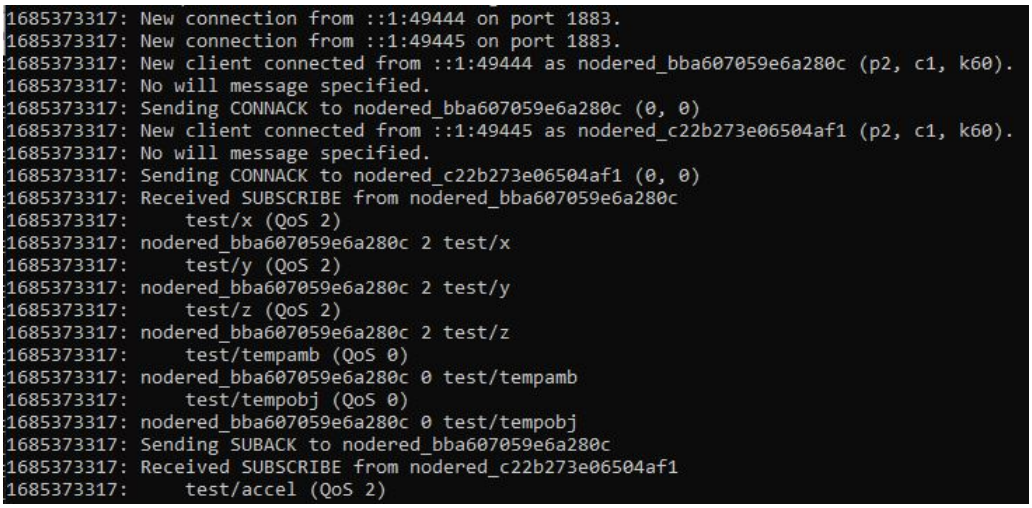


Figure 3.5: Node-Red Subscribed to Topics

3.4.6 Data Storage and Retrieval

We have chosen to work with Firebase due to its remarkable real-time database feature. To send accelerometer and temperature sensor data, we incorporated the Firebase Arduino Client Library for ESP32 [8]. To accomplish this, we assigned each sensor reading to a specific path within the database.

The library facilitated direct communication with the database, ensuring smooth and efficient data transmission. By assigning the sensor data to their respective paths in the database, we achieved organized storage and easy retrieval. This implementation demonstrated the simplicity and effectiveness of using the Firebase Arduino Client Library [8] and Firebase Realtime Database [10] to handle and store sensor data in our project.

```
void loop(void){
    while (millis() - startTime >10000){
        Serial.println("done");}
    for(i=1;i<=5;i++){
     // Send new readings to database
        if (Firebase.ready() && (millis() - sendDataPrevMillis >
           timerDelay || sendDataPrevMillis == 0)){
            sendDataPrevMillis = millis();
             //Get the current timestamp
            timestamp = getTime();
            parentPath= databasePath + "/" + String(timestamp)+" val num
               : "+String(i);
            //Get the acceleration in the three directions of xyz, in
               addition to the ambient and body temperature
            double ax,ay,az,tempamb,tempobj;
            ax = acce.readAccX();
            //Get the acceleration in the x direction
            ay = acce.readAccY();
            //Get the acceleration in the y direction
            az = acce.readAccZ();
            //Get the acceleration in the z direction
            tempamb=mlx.readAmbientTempC();
            tempobj=mlx.readObjectTempC();
            // sending data the the Firebase DB
            json.set(axPath.c_str(), String(ax));
            json.set(ayPath.c_str(), String(ay));
            json.set(azPath.c_str(), String(az));
            json.set(tempambPath.c_str(), String(tempamb));
            json.set(tempobjpath.c_str(), String(tempobj));
            json.set(timePath,String(timestamp)+ "/" + " val num "
               +String(i));
            Serial.printf("Set json... %s\n",
               Firebase.RTDB.setJSON(&fbdo,parentPath.c_str(), &json) ?
               "ok" fbdo.errorReason().c_str());
            } }
  i=0;}
```

We then extracted the data collected from Firebase as a CSV file to use for training machine learning models. This process involves retrieving the relevant data from Firebase Realtime Database and transforming it into a structured CSV format. This approach allows us to leverage the rich data collected and utilize it effectively in building and training machine learning models.

```
import firebase_admin
from firebase_admin import credentials, db
import pandas as pd

# Fetch the service account key JSON file path
cred = credentials.Certificate("/content/icowcare-firebase-
adminsdk-hws0v-c4f510403e.json")
# Initialize the Firebase app with the service account key
firebase_admin.initialize_app(cred, {
    'databaseURL':
       'https://icowcare-default-rtdb.europe-west1.firebasedatabase.app/'
})

# Fetch the reference to the RTDB node that you want to read
ref = db.reference('/UsersData/mEsvfduYLqg8Ax6a7F4hV7eeeTG2/readings')

# Read the data from the RTDB node as a dictionary
data = ref.get()

# Convert the dictionary to a Pandas DataFrame
df = pd.DataFrame.from_dict(data, orient='index')

# Print the resulting DataFrame
print(df)
#export dataframe
df.to_csv('data.csv', index=False)
```

3.4.7 Dashboard

For the implementation of our project's dashboard, we carefully evaluated various options and ultimately chose Node-RED as our preferred tool. Node-RED is a powerful visual programming platform that offers a user-friendly interface for developing IoT applications and data flows. Its selection was based on several compelling reasons that align with our project requirements and objectives.

One of the primary factors driving our choice is Node-RED's intuitive and visual development environment. With its drag-and-drop interface, we can easily create and configure dashboard components, such as charts, gauges, and data visualizations, without the need for extensive coding. This feature significantly streamlines the development process and allows us to focus more on enhancing the dashboard's functionality and user experience.

Moreover, Node-RED seamlessly integrates with MQTT, the messaging protocol we have chosen for our IoT infrastructure. MQTT compatibility enables efficient communication between the MQTT broker and the dashboard. Node-RED's built-in MQTT nodes facilitate subscribing to topics, receiving real-time messages, and updating the dashboard based on the incoming data.

This robust integration ensures smooth data flow and synchronization between our IoT devices and the dashboard.

Additionally, Node-RED benefits from a vibrant and supportive community. This active community provides access to a wealth of resources, tutorials, and examples, making it easier for us to leverage best practices, troubleshoot issues, and stay updated with the latest advancements in Node-RED. The availability of this community-driven support ensures that we have the necessary guidance and assistance throughout the development process, enabling us to achieve optimal results.

In summary, Node-RED emerged as the ideal choice for our project's dashboard implementation due to its user-friendly interface, seamless MQTT integration, extensibility through pre-built nodes, and strong community support. By leveraging Node-RED's capabilities, we can create visually appealing and interactive dashboards that effectively present the data collected from our IoT devices. This strategic decision aligns with our goal of developing a professional and comprehensive monitoring and visualization platform for our project.

3.4.7.1 I-Cow-Care Node-Red Dashboard

Our Node-RED flow consists of several components to read and process data from our MQTT broker. Here is a breakdown of the flow:

MQTT In Nodes: There are multiple MQTT In nodes in the flow, each responsible for reading a specific type of data. The MQTT In nodes are configured to subscribe to MQTT topics and receive messages published on those topics. The topics are related to ambient temperature, body temperature, acceleration (specifically x, y, and z values).

Gauges and charts : The flow includes gauges and chart nodes that are used to visualize the received data allowing you to monitor and interpret the data in a more intuitive way.

Data Conversion: After receiving the MQTT messages, we need some processing steps to convert and format the data appropriately.

File Conversion: Once the data is processed and formatted, the flow includes a node or set of nodes to convert the data into a file format. This is done using nodes that write the data to a file on the local system or send it to a database or cloud storage service.

ID Cow Node: The flow also incorporates an "ID Cow" node, which is used to display or log the ID of the cow.

In summary, the flow receives MQTT messages from different topics related to outside temperature, body temperature, acceleration (x, y, z), and cow ID. The data is processed, visualized using gauge and chart nodes, and converted into a file format for further analysis or storage. The ID Cow node is used to display or log the ID of the cow associated with the data.

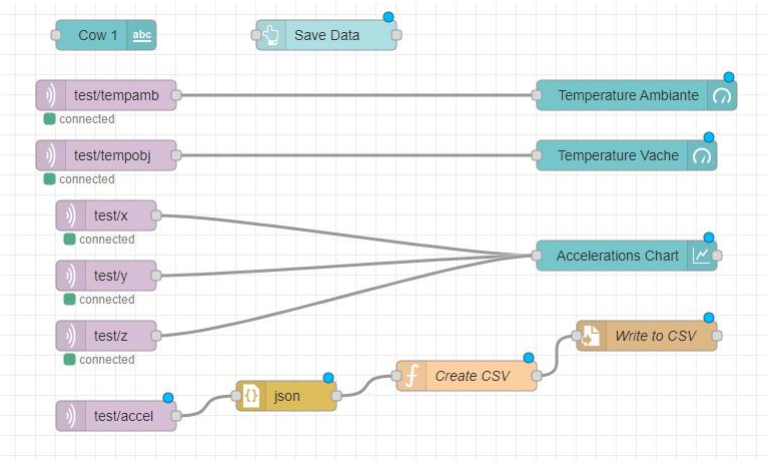


Figure 3.6: I-Cow-Care Node-RED Dashboard Flow

3.4.8 Machine Learning

Machine learning plays a crucial role in enhancing cow monitoring devices, revolutionizing the way we track and manage the health and behavior of cows. By leveraging algorithms and data analysis techniques, machine learning enables these devices to extract valuable insights from the collected data, leading to improved farming practices and better animal welfare.

By training models on datasets of health records, sensor data, and environmental factors, machine learning algorithms can learn to identify patterns that indicate heat and potential health problems. This can include detecting, temperature, rumination patterns, or other sensor measurements.

We will be using the Edge Impulse platform to train our machine-learning model and deploy it on the ESP32 for real-time inference.

Deploying a final model is a comprehensive process consisting of three major steps: data acquisition, feature engineering, and classification.

3.4.8.1 Data Acquisition

We will utilize two distinct procedures to monitor and analyze animal behavior:
Tracking movement: By using a triaxial accelerometer attached to the cow's neck, we capture detailed movement values That enable us to identify behavioral patterns.
Measuring body temperature: To gather information about the cow's body temperature, we employ a temperature sensor that periodically records and transmits values. This enables us to monitor temperature changes over time and predict the animal's physiological state.

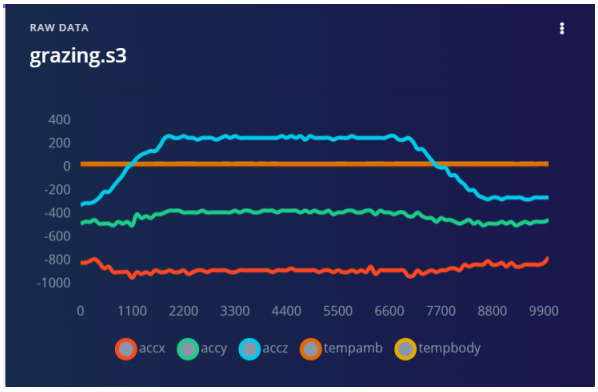


Figure 3.7: Raw data

3.4.8.2 Data Preprocessing

The first step consists in dividing each time signal(on each axis) into consecutive time windows of 10 seconds each. As a result, each interval consists of 100 samples and a sampling rate of 10 Hz produces 10 samples per second.

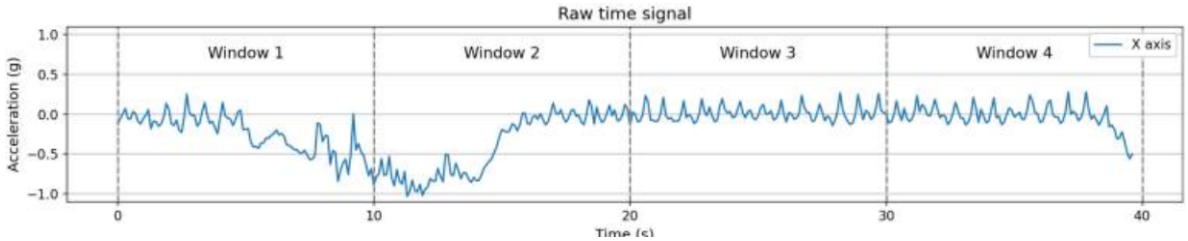


Figure 3.8: Window slicing. [1]

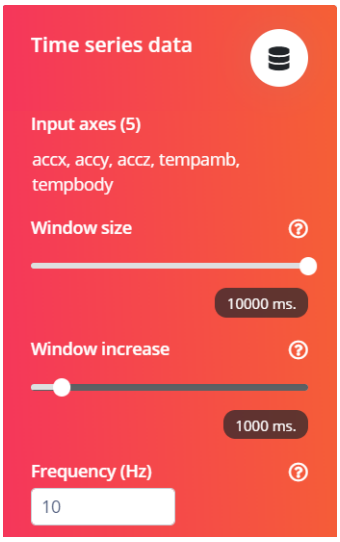
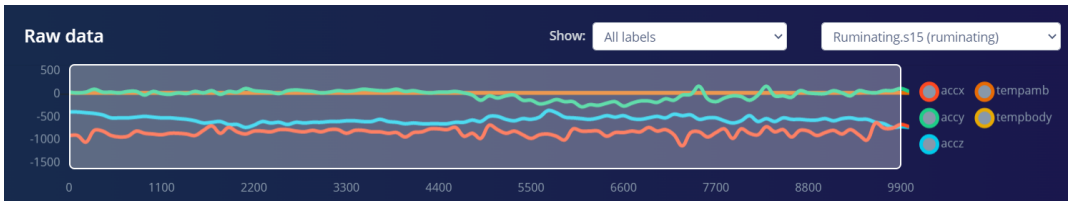
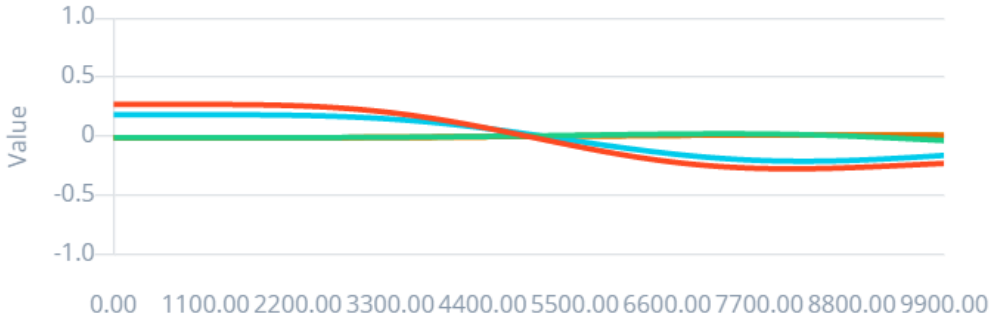


Figure 3.9: Window Slicing Block

We can filter the time-series data within the sample window to smoothen the signal or remove unwanted artifacts.



(a) Data Before Filtering



(b) Data After Filtering

Figure 3.10: Accelerometer Data Filtering

3.4.8.3 Feature Engineering:

The procedure for accelerometer raw data processing consists of different steps, these steps are shown in the figure below :

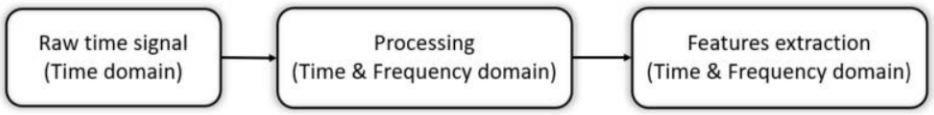


Figure 3.11: Feature extraction process. [1]

After filtering, the mean is subtracted from the signal. Several statistical features are calculated from the filtered signal which include:

- RMS: Root mean square of the signal.
- Skewness: Indicates the symmetry of the probability density function of the amplitude of a time series.
- Kurtosis: Measures the peakedness of the probability density function of a time series.

This filtered signal is passed to the Spectral power section, which computes the FFT, this can help us capture significant frequency components within the data.

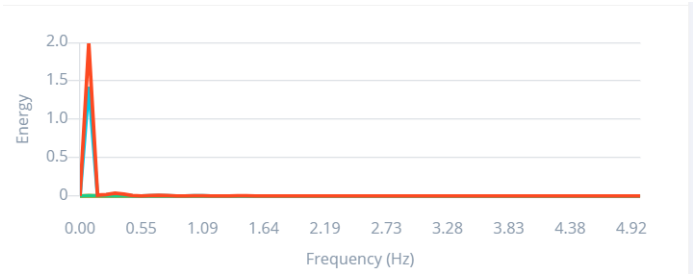


Figure 3.12: Spectral Power Graph

The generated features are listed below in descending order of importance, from the most significant to the least:

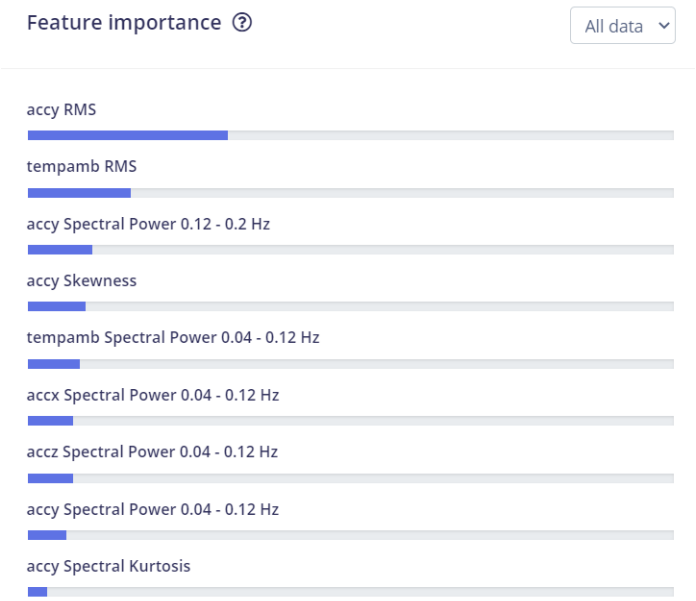


Figure 3.13: Feature importance ranking

3.4.8.4 Classification

Features extracted from accelerometer signals are used to train a supervised ML algorithm for behavioral pattern classification into three classes: Grazing, Ruminating, and Walking.

- Grazing: Regularly lowering and raising its head to eat pasture, while standing or walking slowly.
- Ruminating: Ruminating previously eaten food, while standing or laying.
- Walking: Walking at a normal pace with calm steps.

  To classify these behaviors, we chose to implement a neural network. The architecture of the network is depicted below, illustrating the arrangement of its layers and connections.

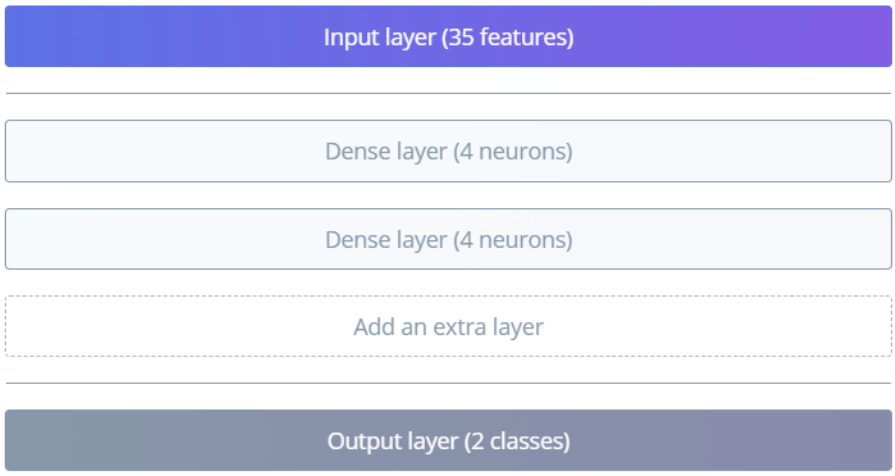


Figure 3.14: Neural Network Architecture

The neural network implemented utilizes specific hyperparameters that significantly impact its performance and learning capabilities. The hyperparameters chosen for this network are as follows:

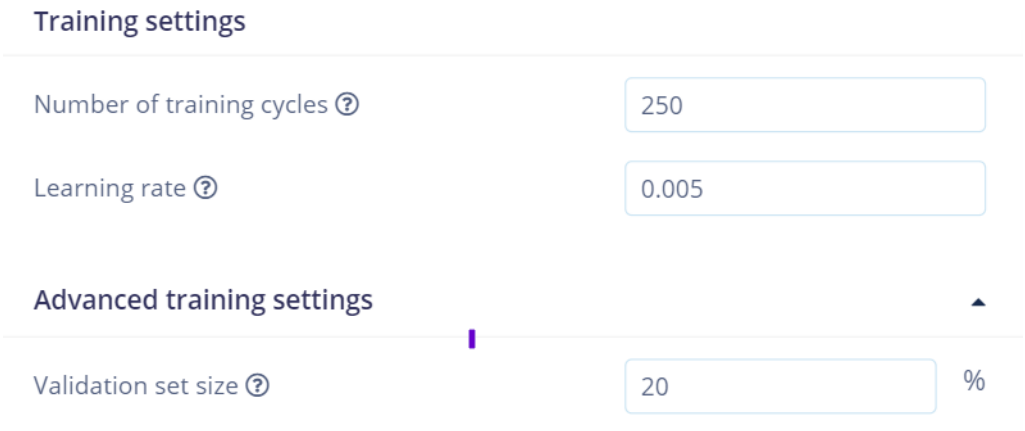


Figure 3.15: Neural Network Architecture

3.4.8.5 Evaluation Metrics

We need to understand the following terms first :

- True Positives (TP): Instances where the model correctly predicts positive outcomes.
- False Positives (FP): Instances where the model incorrectly predicts positive outcomes when the actual outcome is negative.
- True Negatives (TN): Instances where the model correctly predicts negative outcomes.
- False Negatives (FN): Instances where the model incorrectly predicts negative outcomes when the actual outcome is positive.

When evaluating the performance of the neural network, Two common metrics can be used which are accuracy and F1 score.

Accuracy: measures the overall correctness of the model's predictions by calculating the ratio of correctly classified instances to the total number of instances.

F1 Score : takes into account both precision and recall to provide a more complete evaluation. Precision measures how many of the predicted positive instances are actually correct, while recall measures how many of the actual positive instances were correctly predicted.

$$F_1\text{−score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

Figure 3.16: F1 score

3.5 Use Case

In our use case, the cow monitoring IoT solution involves the installation of a collar securely fastened around the cows' neck using a strap. To power the system, we rely on a CR123A 3V 1300MAH battery, which provides an estimated battery life of approximately 143 days or 4.8 months. The ESP32-DevKitC V4 serves as the central control unit, which needs to be turned on and connected to the local farm's Wi-Fi network.

Real-time data is acquired from our accelerometer and temperature sensors via I2C communication. The data is collected in batches of 10 readings per second, with each reading consisting of three values of X, Y, and Z accelerometer readings, as well as ambient and body temperature values. This results in a transmission frequency of 10 Hz. The data is transferred using Wi-Fi and MQTT protocol. The ESP32 acts as an MQTT client, publishing the acquired data in CSV format to an external MQTT broker. A Node-RED client subscribes to the MQTT broker, processes the data, and presents it in a well-organized Web dashboard.

Simultaneously, the ESP32 sends data in JSON format to a Firebase database via Wi-Fi. This database will be utilized for training machine learning models using the Edge Impulse platform. The process involves retrieving relevant data from the Firebase Realtime Database and transforming it into a structured CSV format. Once the machine learning models are trained, they will be deployed on the ESP32 board for testing and real-world application.

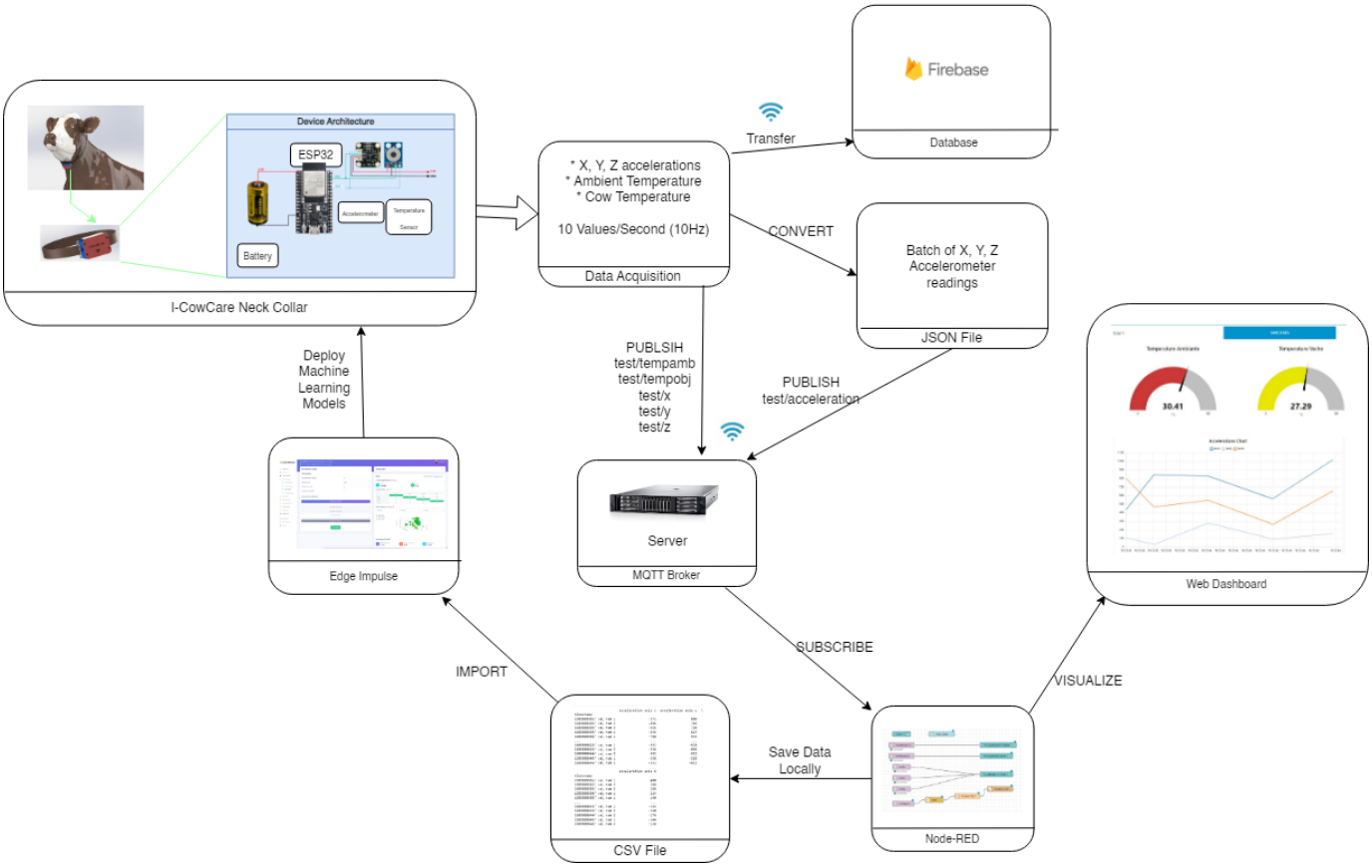


Figure 3.17: I-CowCare Use Case Flow Diagram

3.6 Conclusion

This step was the most important of the project as it allowed us to see our solution developing in all its aspects from data acquisition, data sharing, visualization through the dashboard, saving data into the database, connecting multiple devices to the broker, and exploring machine learning algorithms to optimize our data analyzing and help us monitor the cow state in real-time.

Chapter 4

Test and Validation

4.1 Introduction

This chapter our report focuses on test and validation, which is crucial to ensure the accuracy and reliability of our system. In this final phase, we conducted tests on the sensors we integrated, starting with the accelerometer sensor (DFRobot Gravity LIS2DH) and the temperature sensor (GY-906 MLX90614). The readings from these sensors were found to be correct and consistent, validating their reliability.

We also tested the connectivity of our IoT architecture, including Wi-Fi and MQTT, and successfully established connections with the desired outputs. Furthermore, we verified that the acquired data was being published to the MQTT broker and could be accessed by various devices and platforms. Lastly, we utilized Node-RED Dashboard to visualize the received data, enabling real-time monitoring and analysis of temperature and acceleration trends. These validation tests and visualization capabilities provide us with valuable insights for effective cow monitoring and care.

4.2 Sensors Testing

After cabling our sensors and power source to the ESP32, we now need to test the sensors and check the data acqisition.

4.2.1 Accelerometer Sensor : DFRobot Gravity LIS2DH

To test the accelerometer, we use a code to read values from sensor using I2C protocol.

```
Serial.print("Acceleration x: "); //print acceleration
    Serial.print(x);
    Serial.print(" mg \ty: ");
    Serial.print(y);
    Serial.print(" mg \tz: ");
    Serial.print(z);
    Serial.println(" mg");
```

The output shows that the readings are correct for the three accelerometer variables.

```
Acceleration x: 1015 mg      y: -46 mg    z: 156 mg
```

4.2.2 Temperature Sensor : GY-906 MLX90614

In the same way we want to test the MLX90614 to ESP32 data acquisition.

```
Serial.print("Ambient Temperature is");
Serial.print(mlx.readAmbientTempC());
Serial.println("*C");
Serial.print("Object Temperature is");
Serial.print(mlx.readObjectTempC());
Serial.println("*C");
```

We get the output as below which validates that the sensor's readings are reliable and stable.

```
Ambient Temperature is 26.23 *C
Object Temperature is 26.09 *C
```

4.3 Connectivity

In order to connect the different compartments of the IoT architecture (ESP32, MQTT Broker, Dashboard, Database), we need to assure

4.3.1 Wi-Fi

After writing the code related to Wi-Fi connection, we tested it and got the following output validating the established connection:

```
..
WiFi connected
IP address:
10.13.0.244
```

4.3.2 MQTT

After that, we need to test the connection to the MQTT broker and verify that our data is published to the corresponding topics.

```
if (!client.connected()) reconnect(); // check if client is connected
```

The output is:

```
Attempting MQTT connection...connected
```

We also tested the publishing of acquired data

```
publishMessage("test/cowid", "Cow No 1", true);
publishMessage("test/x", xacc, true);
publishMessage("test/y", yacc, true);
publishMessage("test/z", zacc, true);
publishMessage("test/tempamb", tempAmb, true);
publishMessage("test/tempobj", tempObj, true);
publishMessage("test/timestamp", timestamps, true);
```

We validate with the confirmation :

```
Message published [test/cowid]: Cow No 1
Message published [test/x]: 1015
Message published [test/y]: -46
Message published [test/z]: 156
Message published [test/tempamb]: 26.23
Message published [test/tempobj]: 26.11
Message published [test/timestamp]: 1685376148
```

This means that the data is published to the broker so it will be open to any node or device subscribing to the topics including Online MQTT servers, PCs, other microprocessors, Node-RED, smartphone, Web Page, Database...

4.4 Node-Red Dashboard

We utilized our MQTT subscription to receive data for various topics, including ambient temperature, object temperature, and acceleration values in the X, Y, and Z axes. Whenever the sensors read new values, they publish these readings, which are then received by the Node-RED client. The client processes and visualizes these values using gauges and charts, providing real-time insights into the data. The figures below showcase the graphical representation of the received data, enabling us to monitor and analyze the temperature and acceleration trends effectively.

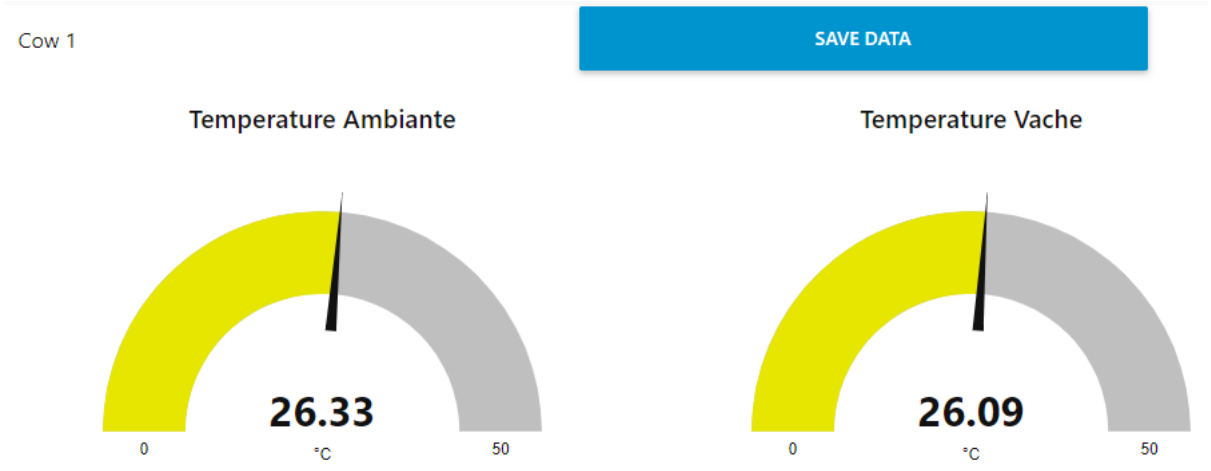


Figure 4.1: I-Cow-Care Temperatures Dashboard

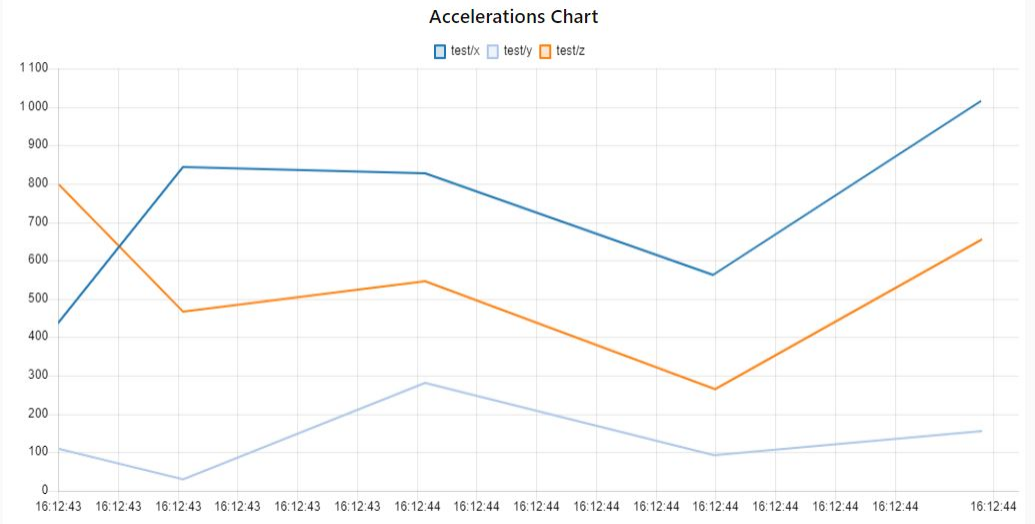


Figure 4.2: I-Cow-Care Accelerations Dashboard

4.5 Firebase

The data collected from the accelerometer and the temperature sensor is transmitted to the Firebase in real time. The code shown below confirms the connection to the database and the transmission of the data including acceleration values on the x,y, and z-axis and both the ambient and body temperature.

```
Token info:
type Token info: type= id token (GITKit token) ,status= on request
type Token info: type= id token (GITKit token) ,status= ready
Getting User UID
User UID: mEsvfduYLqg8Ax6a7F4hV7eeeTG2
1024.00,-32.00,144.00,29.19,26.97
Set json. . . ok
```

The resulting database is shown in the figure below.

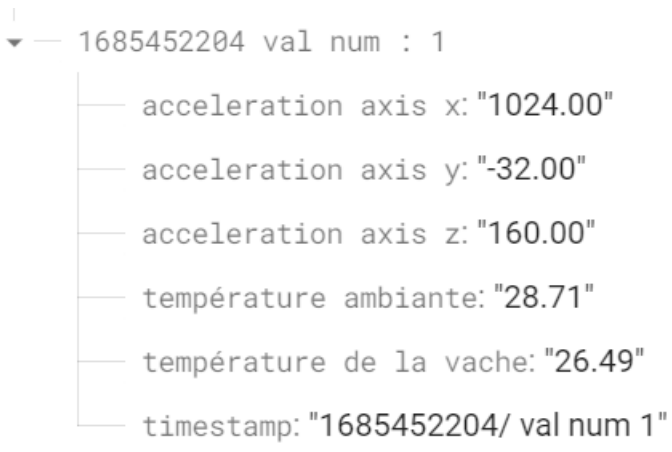


Figure 4.3: Firebase Realtime Database Interface

4.6 Machine Learning

4.7 Model Evaluation

4.7.0.1 Validation set

The confusion matrix presented below summarizes the validation results of our model, showcasing its performance in classifying various samples. The matrix reveals the percentage of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) obtained during the evaluation.

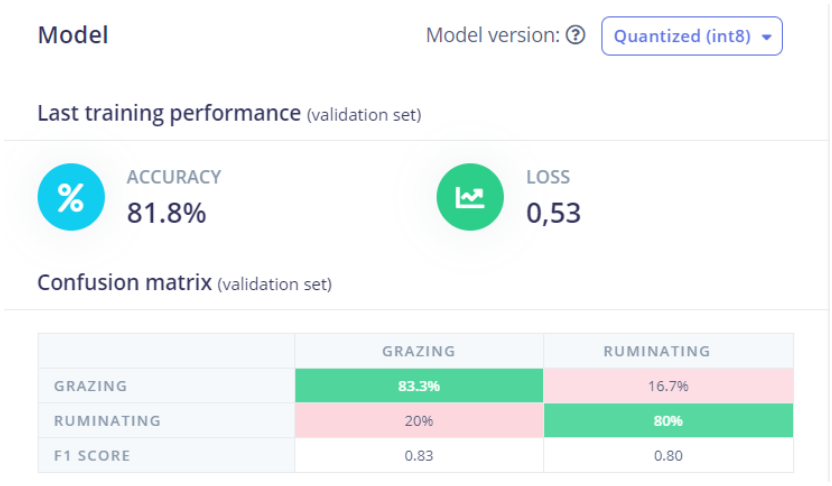


Figure 4.4: Model evaluation on Validation Set

4.7.0.2 Testing Set

The figure below shows the accuracy of the model to predict the classes of the testing set, in addition to the confusion matrix and f1 scores of each label.

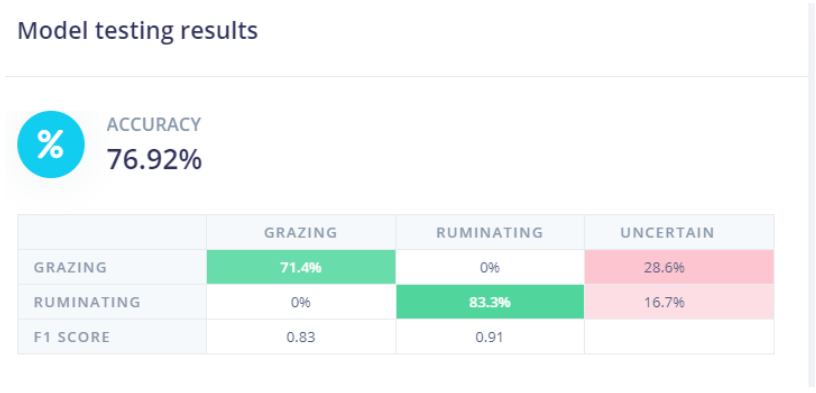


Figure 4.5: Model evaluation on testing set

4.8 Conclusion

In conclusion, the test and validation phase of our project ensured the accuracy and reliability of our system. Our integrated sensors, including the accelerometer and temperature sensor, provided consistent and reliable data capture. We established and tested the connectivity of our IoT architecture, enabling seamless data transmission. The visualization through Node-RED Dashboard offered real-time monitoring and valuable insights for cow monitoring. Integration with Firebase facilitated secure storage and accessibility of the acquired data.

Despite the challenge of a small dataset, our machine learning model achieved an 80% accuracy in classifying and predicting cow behaviors. Overall, this successful test and validation phase confirmed the robustness of our solution, ready for real-world implementation and Proof of Concept validation.

General Conclusion

In conclusion, this report has showcased significant advancements in the field of estrus detection of dairy cows, presenting a comprehensive solution that harnesses the power of IoT technologies. Our innovative approach addresses the critical problem faced by farmers in optimizing breeding practices and overall herd management. By accurately identifying estrus, our solution has the potential to revolutionize the dairy industry, leading to enhanced productivity, improved profitability, and sustainable growth for farmers.

Through in-depth research and practical implementation, we have developed a sophisticated system that integrates IoT technologies to enable real-time monitoring and precise data collection. By leveraging advanced sensors, intelligent boards, reliable power sources, and robust servers, our solution provides an effective and reliable method for detecting estrus. This technological integration empowers farmers with timely and accurate information, allowing them to take proactive actions to maximize breeding outcomes.

The test and validation phase played a crucial role in assessing the accuracy and reliability of our system, particularly in the context of machine learning-based classification and prediction. We conducted rigorous tests on integrated sensors, validating their consistent and accurate data capture. Additionally, the connectivity of our IoT architecture, including Wi-Fi and MQTT, was successfully tested, ensuring seamless data transmission. The use of Node-RED Dashboard facilitated real-time monitoring and analysis, offering valuable insights into cow behavior trends. Integration with Firebase enabled secure storage and accessibility of the acquired data.

Despite the challenge of a limited dataset, our Machine Learning model demonstrated its effectiveness in classifying and predicting different cow behaviors, achieving an accuracy of 80%. These successful test and validation results validate the potential of our solution for effective cow monitoring and care. By providing a cost-effective estrus detection tool, we aim to empower farmers and contribute to their success in the dairy industry.

In summary, the report has presented a comprehensive solution that combines IoT technologies with machine learning to address the crucial aspect of estrus detection in milk cows. By improving breeding practices and enhancing overall herd management, our solution holds the potential to transform the dairy industry, leading to increased productivity, improved profitability, and sustainable growth for farmers.

Bibliography

[1] Analysis of accelerometer and gps data for cattle behaviour identification and anomalous events detection. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8947510/?fbclid=IwAR0DSRsS0LPBjRMz9k19rR6_2lcHEQteRkwxHxfhFX0vMezc243Tc--PL3I.

[2] Cr123a 3v 1300mah battery datasheet. URL: https://www.tme.eu/Document/4dfa436d3d693704b7dd8b1095903d70/BAT-CR123A.pdf.

[3] Datasheet for dell poweredge r230. URL: https://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell_PowerEdge_R230_SpecSheet_final.pdf.

[4] Datasheet for esp32 wroom 32d. URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf.

[5] Datasheet for lis2dh. URL: https://www.st.com/resource/en/datasheet/lis2dh.pdf.

[6] Datasheet for mlx90614. URL: https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90614.

[7] Estrous cycle. URL: https://www.mdpi.com/2306-7381/9/12/688.

[8] Firebase arduino client library for esp32. URL: https://github.com/mobizt/Firebase-ESP-Client.

[9] Firebase realtime database.

[10] Firebase realtime database. URL: https://firebase.google.com/docs/database.

[11] Firk et al., 2002. automation of oestrus detection in dairy cows: a review. livestock production science, 75(3), pp.219-232. URL: https://www.sciencedirect.com/science/article/pii/S0301622601003232.

[12] Getting started guide for esp32-devkitc v4. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html.

[13] Marques et al., 2020. effect of automated activity monitors on reproductive performance of lactating holstein cows. journal of dairy science, 103(11), pp.10430-10441. URL: https://pubmed.ncbi.nlm.nih.gov/32418687/.

[14] Mqtt scenario of turning on a lamp in your home office. URL: https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/.

[15] Ovarian cycle. URL: https://www.groupe-esa.com/ladmec/bricks_modules/brick03/co/ZBO_Brick03_4.html#:~:text=The%20estrous%20cycle%20of%20the,and%20ovulation%20(ovum%20release).

[16] Product page for ansmann 3v. URL: https://brico-direct.tn/piles-batteries/2366-pile-speciale-cr123a-ansmann-4013674020010.html.

[17] Product page for dfrobot gravity lis2dh. URL: https://www.google.com/search?client=opera-gx&q=gravity+LIS2DH&sourceid=opera&ie=UTF-8&oe=UTF-8.

[18] Product page for huiderui 3v. URL: https://www.jumia.com.tn/huiderui-pile-3v-batterie-au-lithium-primaire-cr123a-688849.html.

[19] Product page for lis3dh. URL: https://www.st.com/en/mems-and-sensors/lis3dh.html.