

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
-T L E M C E N-



المدرسة العليا في العلوم التطبيقية
École Supérieure en
Sciences Appliquées

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
- تلمسان -

A Thesis Presented in Partial Fulfillment of the Requirements For

The Degree of Master

Department: Second Cycle
Specialization: Automation

Submitted by : TESSOUMI Mohamed Amine Bannedjadi

CHERIFI Mohamed El Amine

Thesis Title

**Integration of Computer Vision AI
(Camera + Image Processing)
into an IoT Calving Detection Device**

Date: 21 June, 2026

before the jury composed of:

Dr.GHOUALIAmel	MCA
Dr.Didi Ibtissem	MCA
Dr. BOUKLI HACENE Lotfi Fazil	MCB
CHIALI Anisse	MCA

ESSATlemcen
ESSATlemcen
ESSATlemcen
ESSATlemcen

President
Examiner1
Examiner2
Supervisor

Academic Year: 2025 / 2026

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Dedication

"All that I have achieved today is by the grace of God and the heartfelt prayers of my parents."

As I reach the end of this important chapter of my academic journey, I look back with gratitude, pride, and appreciation for everyone who stood by my side throughout these years. This achievement would not have been possible without the support, encouragement, and love of the people who believed in me.

First and foremost, I dedicate this work to my beloved parents. Thank you for your unconditional love, endless sacrifices, sincere prayers, and unwavering support. You have always been my greatest source of strength and inspiration. Your patience, guidance, and faith in my abilities gave me the courage to overcome every challenge. I hope this achievement brings you the joy and pride that you truly deserve.

To my brothers and sisters, thank you for your love, encouragement, and understanding. Your presence in my life has always been a blessing, and your support has motivated me to move forward with determination.

I would like to express my sincere gratitude to my dear friend Alami Nadir for his friendship, encouragement, and support throughout this journey. Thank you for always being there and for making difficult moments easier to overcome.

I would also like to extend my heartfelt thanks to my dear friends Abderrahman, Bilel, Aissam Boukheneg, and Zinou Sebaa. Your support, encouragement, advice, and willingness to help meant a great deal to me throughout this journey. I am truly grateful for your friendship and for the unforgettable moments we shared together. Your presence made this path easier and more meaningful.

My deepest appreciation and respect go to my supervisor, Mr. Anisse CHIALI, for his valuable guidance, patience, availability, and continuous support during the realization of this work. His advice and expertise played a significant role in the successful completion of this project.

Special thanks go to all my friends who supported, encouraged, and helped me throughout my academic journey. Your kindness and friendship will always hold a special place in my heart.

Finally, I would like to express my sincere appreciation to all the teachers and professors who contributed to my education and shared their knowledge and experience with me over the years. Your efforts have helped shape the person I am today.

To all of you, thank you from the bottom of my heart.

CHERIFI Mohamed El Amine

"All that I have achieved today is by the grace of God and the heartfelt prayers of my parents."

As I conclude this memorable academic journey, I reflect with gratitude on the many people whose support, encouragement, and presence have accompanied me every step of the way. Their belief in me has given me the strength to persevere and achieve this milestone.

First and foremost, I dedicate this work to my beloved parents. Thank you for your unconditional love, sacrifices, endless prayers, and unwavering support. You have always encouraged me to believe in myself and pursue my dreams. This achievement is as much yours as it is mine.

To my brothers and sisters, thank you for your love, patience, and continuous encouragement. Your support has been a source of comfort and motivation throughout this journey.

I would like to express my sincere gratitude to my dear friend Alami Nadir for his friendship, support, and encouragement. Thank you for standing by my side and for the valuable moments we shared along the way.

I would also like to extend my heartfelt appreciation to my dear friends Abderahman, Bilel, Aissam and Zinou. Thank you for your constant support, encouragement, advice, and willingness to help whenever needed. Your friendship and the unforgettable moments we shared made this academic journey more enjoyable and meaningful. I will always cherish your kindness and companionship. I would also like to express my sincere gratitude to the members of the TechSquad and EcoGeeks University Clubs. Being part of these clubs provided me with valuable opportunities to learn, grow, and develop both my technical and personal skills.

I extend my deepest thanks and respect to my supervisor, Mr. Anisse CHIALI, whose guidance, patience, and constructive advice greatly contributed to the completion of this work. His professionalism and support have been truly appreciated.

Special thanks go to all my friends who offered their help, support, and encouragement during my academic journey. I am truly grateful for your friendship.

Finally, I would like to thank all the teachers and professors who have contributed to my education and shared their knowledge and experience throughout the years. Their dedication has had a lasting impact on my personal and academic development.

With heartfelt gratitude, thank you all.

Tessoumi Mohamed Amine Bennedjadi

Acknowledgement

We would like to express our deepest gratitude to Allah, the Most Merciful and Compassionate, for granting us the strength, patience, and perseverance necessary to complete this work.

First and foremost, we extend our sincere thanks to our supervisor and assistant, Mr. Anisse CHIALI, for his continuous support, valuable guidance, and trust throughout this project. His insightful advice, constant availability, and ongoing encouragement have played a fundamental role in the success of this work. It has been both an honor and a great motivation to work under his supervision.

We would also like to express our sincere appreciation to Pr. MEGNAFI Hicham, Associate Professor at the Higher School of Applied Sciences of Tlemcen (ESSA Tlemcen), for his valuable follow-up and academic support.

We are also deeply grateful to M. ABDI Sidi-Mohammed for his careful review of the project report and his insightful remarks, which significantly contributed to improving the quality of this work.

Our sincere thanks go to Dr. Elyasmine Kadaoui-Elaïdi for her support, guidance, and constructive feedback, which greatly enriched our understanding and improved this work.

We would also like to thank M. Ramzeeddine AJIM, Mechanical Engineering Engineer, for his technical guidance and mechanical solutions. His expertise helped us overcome several technical challenges, and his support was essential to the success of this project.

We extend our gratitude to Mme Ouhoud Amina, Assistant Professor “B” at ESSA Tlemcen, for her continuous encouragement, valuable advice, and constant support throughout this work. Her dedication and availability greatly enriched our experience and contributed to the success of this project.

We also thank the members of the academic staff and all those who contributed directly or indirectly to this work.

Finally, we would like to express our sincere gratitude to all our friends and classmates from the Automation Promotion 2026–2027, for their support, encouragement, and the memorable moments shared throughout our academic journey.

الملخص

يُعدّ الولادة من أبرز الأحداث الحرجة في تربية الأبقار الحلوب، إذ تُشكّل المضاعفات غير المكتشفة في الوقت المناسب خطراً حقيقياً على حياة العجل والبقرة معاً، فضلاً عن خسائر اقتصادية جسيمة. تعتمد أساليب المراقبة التقليدية على الجولات اليدوية للعمال، وهي طريقة مرهقة وغير موثوقة، لا سيما خلال ساعات الليل التي تشهد غالبية حالات الولادة.

يقدم هذا المذكرة نظام صمّرثلف / ثلثنكصف، وهو نظام ذكي منخفض التكلفة مبني على تقنية إنترنت الأشياء، صُمم للكشف التلقائي عن الأنماط السلوكية التي تسبق الولادة وإرسال تنبيهات فورية إلى المربين والأطباء البيطريين. يتكوّن النظام من ثلاثة مستشعرات بعو ٥٠٦٠٠ سداسية المحاور مثبتة على ذيل البقرة، مترابطة عبر مُضاعف إيثرنت من نوع ١٨٤٥٩٠٠ بمتمحكم يصع ٦٦٢٨ ديثو المزود بـ واذ. تُرسل البيانات عبر بروتوكول اوضع إلى خادم عيتهن محلي يعالجها من خلال نافذة انزلاقية من ٢٠ قراءة متتالية، ثم يُغذّيها إلى محرك ذكاء اصطناعي هجين.

يجمع هذا المحرك بين شبكة عصبية اصطناعية (١) من طبقتين خفيتين (١٢٨ × ٦٤ عصبوناً، تفعيل تنه، عتبة = ٢٠٠، مُدرّبة بتوازن صيوة على مجموعة بيانات منشورة على كوكگل)، ونظام منطق ضبابي من نوع يمدن (ذيص) بسبعة مدخلات وثمانى قواعد استنتاج. تُنتج آلية التصويت بين النموذجين مع فلتر زمني من ١٠ إطارات متتالية ثلاثة مستويات ثقة: عالٍ، متوسط، منخفض، يُقابل كلّ منها سلوك تنبيه محدد.

تُرفع الأحداث المكتشفة إلى قاعدة بيانات ذربس غالتم وتنعكس فورياً على لوحة القيادة ثلثنكصف المنشورة على طرخل، التي توفر واجهات مختلفة للمربي والطبيب البيطري، مع نظام تنبيه ثلاثي القنوات يشمل تنبيهات اللوحة المرئية وسجلات الأحداث وإشعارات لينك على الهاتف.

جرى التحقق من النظام عبر وضع المحاكاة بملف ثصط على مجموعة بيانات ايبو مصنّفة، وأثبتت النتائج صحة الأداء من البداية إلى النهاية عبر جميع المراحل، مع زمن استجابة كلي يقل عن ثانية واحدة.

الكلمات المفتاحية: كشف الولادة، الزراعة الدقيقة للمواشي، إنترنت الأشياء، يصع ٦٦٢٨،
٥٠٦، الشبكة العصبية الاصطناعية، المنطق الضبابي، الذكاء الاصطناعي الهجين، دريس،
المراقبة في الوقت الحقيقي، صبوةي.

Abstract

Calving is one of the most critical events in dairy cattle farming. Complications during delivery, if undetected in time, lead to significant calf and cow mortality, substantial economic losses, and increased veterinary costs. Traditional monitoring methods, based on manual farm inspections, are labor-intensive and unreliable, particularly during nighttime hours when the majority of calvings occur.

This memoir presents CalvingSafe, a low-cost, IoT-based intelligent system designed to automatically detect pre-calving behavioral patterns and deliver real-time alerts to farmers and veterinarians. The system integrates three MPU6050 six-axis inertial measurement units mounted along the cow's tail, connected through a TCA9548A I2C multiplexer to an ESP8266 NodeMCU Wi-Fi microcontroller. Sensor data is transmitted via UDP over a local Wi-Fi network to a Python backend, where it is processed through a sliding window of 20 consecutive readings and fed into a hybrid artificial intelligence engine.

The AI engine combines a feedforward Artificial Neural Network (ANN), implemented as a two-hidden-layer MLPClassifier (128×64 neurons, tanh activation, threshold = 0.2, trained with SMOTE balancing on a Kaggle-published dataset), with a seven-input Mamdani Fuzzy Logic System (FLS) operating on eight handcrafted inference rules. A two-model voting mechanism with a 10-consecutive-frame temporal filter produces three confidence levels: HIGH, MEDIUM, and LOW, each mapped to a specific alert behavior. Detected events are pushed to a Firebase Real-time Database and reflected in real time on the CalvingSafe web dashboard, deployed on Vercel, which provides role-differentiated views for farmers and veterinarians. A three-channel alert system combining in-dashboard banners, Firebase event logging, and Blynk mobile push notifications ensures that no calving event goes unnoticed.

System validation was conducted through a CSV simulation mode exercising the complete inference and cloud pipeline on a labeled IMU dataset. Results confirmed correct end-to-end behavior across all subsystems, with an estimated total pipeline latency below one second. The system demonstrates the feasibility of deploying an intelligent, low-cost calving detection solution adapted to Algerian farming conditions.

Keywords: Calving detection, Precision Livestock Farming, IoT, ESP8266, MPU6050, Artificial Neural Network, Fuzzy Logic, Hybrid AI, Firebase, Real-time monitoring, SMOTE.

Résumé

La mise bas est l'un des événements les plus critiques dans l'élevage bovin laitier. Les complications lors de la parturition, si elles ne sont pas détectées à temps, entraînent une mortalité significative des veaux et des vaches, des pertes économiques importantes et des coûts vétérinaires élevés. Les méthodes de surveillance traditionnelles, basées sur des contrôles manuels, sont fastidieuses et peu fiables, en particulier pendant les heures nocturnes où la majorité des mises bas se produisent.

Ce mémoire présente SmartCalve / CalvingSafe, un système intelligent à faible coût basé sur l'IoT, conçu pour détecter automatiquement les comportements prépartum et envoyer des alertes en temps réel aux éleveurs et aux vétérinaires. Le système intègre trois unités de mesure inertielle MPU6050 à six axes montées le long de la queue de la vache, connectées via un multiplexeur I2C TCA9548A à un microcontrôleur Wi-Fi ESP8266 NodeMCU. Les données capteurs sont transmises via UDP sur un réseau Wi-Fi local vers un backend Python, où elles sont traitées par une fenêtre glissante de 20 lectures consécutives et injectées dans un moteur d'intelligence artificielle hybride.

Le moteur d'IA combine un réseau de neurones artificiels (ANN) à propagation directe, implémenté sous forme de MLPClassifier à deux couches cachées (128×64 neurones, activation tanh, seuil = 0,2, entraîné avec rééquilibrage SMOTE sur un jeu de données publié sur Kaggle), avec un système de logique floue de Mamdani à sept entrées (FLS) exploitant huit règles d'inférence. Un mécanisme de vote à deux modèles avec un filtre temporel de 10 trames consécutives produit trois niveaux de confiance : HAUT, MOYEN et BAS, chacun associé à un comportement d'alerte spécifique. Les événements détectés sont transmis à une base de données Firebase Realtime et reflétés en temps réel sur le tableau de bord web CalvingSafe, déployé sur Vercel, qui offre des vues différenciées pour les éleveurs et les vétérinaires.

La validation du système a été réalisée via un mode de simulation CSV exerçant l'intégralité du pipeline d'inférence et cloud sur un jeu de données IMU étiqueté. Les résultats confirment un comportement correct de bout en bout sur tous les sous-systèmes, avec une latence totale estimée inférieure à une seconde.

Mots-clés : Détection de mise bas, Agriculture de précision, IoT, ESP8266, MPU6050, Réseau de neurones artificiels, Logique floue, IA hybride, Firebase, Surveillance en temps réel, SMOTE.

Contents

List of Figures	13
List of Tables	14
Abbreviations list	14
Introduction	15
1 Theoretical Foundations & State of the Art	18
1.1 Precision Livestock Farming & IoT	18
1.1.1 Calving Detection as a PLF Application	19
1.1.2 Existing Calving Detection Solutions	19
1.2 Embedded Systems & IoT Hardware	20
1.2.1 Overview of Common Platforms	20
1.2.2 The ESP8266 SoC	20
1.3 Sensors & Signal Acquisition	21
1.3.1 Inertial Measurement Units (IMU)	21
1.3.2 The MPU6050 Sensor	22
1.3.3 The TCA9548A I2C Multiplexer	22
1.4 Communication & Cloud Infrastructure	23
1.4.1 UDP Transmission Protocol	23
1.4.2 Firebase Realtime Database	23
1.5 Machine Learning & Artificial Neural Networks	23
1.5.1 Supervised Learning and Classification	23
1.5.2 Classical ML Methods	24
1.5.3 Artificial Neural Networks	24
1.5.4 Handling Class Imbalance with SMOTE	24
1.6 Fuzzy Logic and the Hybrid AI Approach	24
1.6.1 Fuzzy Logic Foundations	24
1.6.2 Justification of the Hybrid Architecture	25
1.6.3 Comparison of AI Approaches for Calving Detection	25
1.7 Supervision Interfaces	25
1.7.1 SCADA and HMI Concepts	25
1.7.2 Raspberry Pi as a Local Display Platform	26

1.7.3	Web Dashboard as the Adopted Solution	26
1.8	Synthesis & Project Positioning	26
2	System Design & Implementation	29
2.1	System Architecture Overview	29
2.2	Hardware Design	30
2.2.1	Sensor Placement Strategy	31
2.2.2	I2C Multiplexing Architecture	31
2.2.3	Y-Axis Absolute Value Convention	32
2.3	Data Acquisition & UDP Transmission	33
2.3.1	Acquisition Loop on the ESP8266	33
2.3.2	UDP Transmission Protocol	33
2.4	Data Preprocessing	34
2.4.1	Sliding Window Assembly	34
2.4.2	Feature Extraction for ANN Input	34
2.4.3	Normalization	34
2.5	ANN Model	35
2.5.1	Training Dataset	35
2.5.2	Network Architecture	35
2.6	Fuzzy Logic Model	36
2.6.1	Input Variables and Universe of Discourse	36
2.6.2	Membership Functions	36
2.6.3	Inference Rules	37
2.7	Hybrid Decision System	38
2.7.1	Temporal Filtering on Fuzzy Output	38
2.7.2	Voting Logic	39
2.7.3	Backend Processing Pseudocode	39
2.8	Firebase Integration	40
2.8.1	Database Structure	40
2.8.2	Real-Time Push Logic	41
2.9	Web Dashboard	42
2.9.1	Architecture and Technology Stack	42
2.9.2	Farmer View	43
2.9.3	Veterinarian View	44
2.9.4	Alert Notification System	45
2.10	Conclusion	46
3	Testing, Validation & Discussion	47
3.1	Testing Methodology	47
3.1.1	Intended Real-Life Testing Protocol	47
3.1.2	Constraints and Limitations	48
3.1.3	Adopted Solution: Two-Mode Operation	48
3.1.4	ESP8266 Firmware Description	49
3.1.5	Test Session Description	51
3.2	Development Environment and Deployment	51
3.2.1	Development Tools and Libraries	52
3.2.2	Web Dashboard Deployment on Vercel	52
3.3	System Validation Results	53

3.3.1	Data Collection and Window Assembly	53
3.3.2	AI Inference Pipeline Validation	54
3.3.3	Firebase Synchronization Validation	54
3.3.4	Dashboard Validation	55
3.3.5	End-to-End Latency	56
3.4	AI Model Performance Analysis	56
3.4.1	Individual Model Analysis	56
3.4.2	Hybrid System Advantage	57
3.5	Discussion	58
3.5.1	What Worked Well	58
3.5.2	Known Limitations	58
3.5.3	Comparison with Existing Solutions	58
3.6	Future Work	59
3.6.1	Short-Term Improvements	59
3.6.2	Medium-Term Research Directions	59
3.6.3	Long-Term Vision	59

General Conclusion	61
---------------------------	-----------

List of Figures

1.1	IoT data flow in Precision Livestock Farming (SmartCalve architecture overview)	19
1.2	ESP8266 NodeMCU key specifications and I2C pin mapping	21
1.3	MPU6050 $\times$ 3 and TCA9548A I2C multiplexer wiring architecture in SmartCalve	22
2.1	SmartCalve four-layer system architecture	30
2.2	SmartCalve tail sensor placement: Sensors A (proximal), B (mid-tail), and C (distal) mounted along the cow's tail, with origin O at the tail base. Live 3D coordinates shown in upper-left corner.	32
2.3	SmartCalve UDP data flow architecture	33
2.4	SmartCalve Fuzzy Logic membership functions: four representative input variables (A_X, B_Z, C_X, C_Y) and the output calving confidence variable with decision boundary at 0.5	37
2.5	SmartCalve per-frame processing pipeline	40
2.6	Firebase Realtime Database root structure: 'smartcalve' and 'test' nodes	41
2.7	Firebase database expanded: 'cows', 'farmers', and 'predictions' subtrees	41
2.8	SmartCalve frontend architecture: index.html, app.js, data.js, style.css connected to Firebase Realtime Database	42
2.9	CalvingSafe login page: role selection (Farmer / Veterinarian), Firebase connection status indicator, and CalvingSafe branding	43
2.10	Farmer view (dark mode): Farmer 1 dashboard showing 5 monitored cows, 3 in Calving status	44
2.11	Farmer view (White mod): Farmer 2 dashboard showing 3 monitored cows, 2 in Calving status	44
2.12	Veterinarian view: farmer list with farm names, cow counts, and calving alert badges (3 farmers, 7 total calving events)	45
2.13	Cow detail modal in veterinarian view: status, AI model results (ANN and Fuzzy Logic scores), clinical notes, and prescriptions panel.	45
3.1	SmartCalve two-mode operation: Real Hardware Mode (ESP8266 via UDP to Python) and CSV Simulation Mode (CSV to Python directly)	49
3.2	Figure 3.2 ESP8266 firmware acquisition loop: TCA9548A channel selection, MPU6050 raw read, Cartesian-to-spherical conversion, and UDP packet transmission	50

3.3	Vercel deployment dashboard: smartcalve project, Production environment, smartcalve.vercel.app domain	53
3.4	Terminal output showing the data collection phase for cow_002 . . .	54
3.5	Firebase Realtime Database console showing cow_002 state record written under farmer3 hierarchy	55
3.6	CalvingSafe dashboard: COW 002 calving alert modal with status badge and action banner	56

List of Tables

1.1	Comparison of existing calving detection solutions	20
1.2	Comparison of embedded platforms for IoT applications	20
1.3	Comparison of AI methods for calving detection classification	25
1.4	SmartCalve technology synthesis: choices and justifications	27
2.1	SmartCalve system layers and their components	30
2.2	MPU6050 sensor placement, channel assignment, and I2C addressing	32
2.3	Data preprocessing pipeline summary	34
2.4	SmartCalve ANN architecture and training parameters	36
2.5	SmartCalve Fuzzy Logic inference rules (Mamdani, 8 rules)	38
2.6	Hybrid voting decision table: ANN $\times$ Fuzzy $\rightarrow$ Confidence level . .	39
3.1	ESP8266 firmware Arduino libraries	50
3.2	Test session parameters and results summary for cow_002 / farmer3 .	51
3.3	SmartCalve development tools and libraries	52
3.4	End-to-end latency breakdown across SmartCalve pipeline stages . .	56
3.5	Comparative analysis: ANN alone vs. Fuzzy alone vs. Hybrid system	57
3.6	Positioning of SmartCalve against existing calving detection solutions	58

Abbreviations list

Abbreviation	Full form
PLF	Precision Livestock Farming
IoT	Internet of Things
IMU	Inertial Measurement Unit
ANN	Artificial Neural Network
FLS	Fuzzy Logic System
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine
UDP	User Datagram Protocol
MADR	Ministère de l'Agriculture et du Développement Rural
PCB	Printed Circuit Board
GPIO	General Purpose Input/Output

Introduction

Livestock farming constitutes a fundamental pillar of Algeria’s agricultural economy. According to the Algerian Ministry of Agriculture and Rural Development (MADR), the national cattle herd plays a central role in food security, yet the sector remains heavily reliant on traditional practices with limited adoption of modern monitoring technologies. In this context, calving, the act of a cow giving birth, represents one of the most economically and physiologically critical events in dairy cattle management.

Complications during delivery, including dystocia, are among the leading causes of calf mortality, cow injury, and financial loss in dairy herds worldwide. The window of intervention is narrow: timely human presence during a difficult calving can be the difference between a healthy outcome and the loss of the calf, the cow, or both. Traditional monitoring relies on manual checks by farm workers, which are labor-intensive, inconsistent, and particularly unreliable during nighttime hours, precisely the period when the majority of calvings occur.

The emergence of Precision Livestock Farming (PLF) and the Internet of Things (IoT) has opened new possibilities for continuous, automated animal monitoring. Sensor-equipped animals can transmit motion and physiological data in real time, enabling AI-powered systems to detect behavioral changes that precede calving hours before the event occurs. However, most existing solutions remain either too costly for adoption in Algerian farming conditions, dependent on proprietary infrastructure, or validated only in research contexts far removed from local deployment constraints.

The central problem addressed in this work is: how can a low-cost, locally deployable IoT system reliably detect pre-calving behavioral patterns in real time and deliver actionable alerts to farmers and veterinarians, under the technical and economic constraints of Algerian livestock farming?

This problem involves several interconnected challenges. First, the hardware must be affordable, robust, and capable of capturing the multi-axis tail motion that characterizes bovine pre-calving behavior. Second, the AI inference engine must handle class imbalance, sensor noise, and real-time processing constraints simultaneously. Third, the supervision interface must be accessible from any device, without requiring local server infrastructure, and must serve both farmers seeking simplified alerts and veterinarians requiring detailed diagnostics.

This project pursues four primary objectives. The first objective is to design and assemble a multi-sensor IoT acquisition node using the ESP8266 microcontroller and three MPU6050 inertial measurement units, capable of continuously capturing tail-

motion data from a monitored cow and transmitting it wirelessly to a processing backend. The second objective is to develop a hybrid AI inference engine combining an Artificial Neural Network trained with SMOTE balancing and a Mamdani Fuzzy Logic System, integrated through a voting mechanism that produces graduated confidence-level alerts. The third objective is to implement a cloud-synchronized supervision dashboard providing real-time, role-differentiated views for farmers and veterinarians, with a three-channel alert notification system. The fourth objective is to validate the complete system through a documented test session demonstrating correct end-to-end behavior across all pipeline stages.

The main contributions of this work are as follows. A low-cost, open-source IoT calving detection system tailored to Algerian deployment conditions is designed and implemented end-to-end. A hybrid ANN and Fuzzy Logic decision architecture with temporal filtering is proposed and validated, combining the learning capacity of neural networks with the interpretability and uncertainty handling of fuzzy inference. A custom labeled IMU dataset is created and published on the Kaggle platform to support reproducible research in bovine calving detection. A production-deployed web dashboard with role-based access, real-time Firebase synchronization, and mobile push notifications is delivered as an operational supervision tool.

This memoir is organized into three chapters. Chapter 1 establishes the theoretical and technological foundations of the SmartCalve system, covering Precision Livestock Farming, embedded systems, inertial sensors, communication protocols, cloud infrastructure, machine learning, fuzzy logic, and supervision interfaces, culminating in a justified technology synthesis. Chapter 2 presents the complete system design and implementation across all four architectural layers: hardware acquisition, Python backend processing, Firebase cloud integration, and the CalvingSafe web dashboard. Chapter 3 describes the testing methodology, documents the validation results obtained through simulation-based testing, analyzes the AI model performance, discusses known limitations, and proposes a roadmap of future improvements.

THEORETICAL FOUNDATIONS & STATE OF THE ART

This chapter establishes the theoretical and technological foundations upon which the SmartCalve system is built. It begins with an overview of Precision Livestock Farming and the role of IoT in modern agriculture, followed by a detailed examination of the embedded hardware, sensors, communication protocols, cloud infrastructure, artificial intelligence methods, and supervision interfaces relevant to this project. The chapter concludes with a synthesis table that justifies every technical choice made in the design of SmartCalve.

1.1 Precision Livestock Farming & IoT

Precision Livestock Farming (PLF) is an emerging paradigm that applies sensing, data processing, and decision-support technologies to individual animal management. Rather than treating a herd as a uniform group, PLF enables farmers and veterinarians to monitor each animal's health, behavior, and reproductive status in real time, allowing for early detection of anomalies and timely interventions [3]. The integration of Internet of Things (IoT) devices into livestock environments has been the primary technological driver behind this transformation.

IoT in agriculture refers to the deployment of interconnected sensor nodes attached to animals or embedded in their environment that continuously collect data and transmit it through wireless networks to centralized platforms for analysis and visualization. In the context of dairy cattle farming, IoT applications include milk yield monitoring, rumination tracking, estrus detection, body temperature sensing, and calving detection [3, 13].

In Algeria, livestock farming represents a significant component of the national agricultural economy. According to the Algerian Ministry of Agriculture and Rural Development (MADR), the national cattle herd is distributed across rural and semi-arid zones, with bovine production playing a central role in food security [15, 16]. Despite this economic importance, the sector remains heavily reliant on traditional practices. The adoption of PLF technologies in Algeria is still at an early stage, presenting both a challenge and an opportunity for engineering projects like SmartCalve that target local deployment conditions [8, 12].

1.1.1 Calving Detection as a PLF Application

Among all PLF applications, calving detection holds particular urgency. Calving, the act of a cow giving birth, is a physiologically and economically critical event. Complications during delivery, including dystocia (difficult birth), are among the leading causes of calf mortality and cow injury in dairy herds. Studies show that the absence of timely human assistance during difficult calvings significantly increases both mortality rates and veterinary costs [23].

Traditional monitoring relies on manual checks by farm workers, which are labor-intensive, inconsistent, and particularly unreliable during nighttime hours, precisely the period when the majority of calvings occur. Automated calving detection systems aim to replace or supplement these manual checks by continuously analyzing sensor signals and generating alerts when pre-calving behavioral patterns are detected [23, 5].

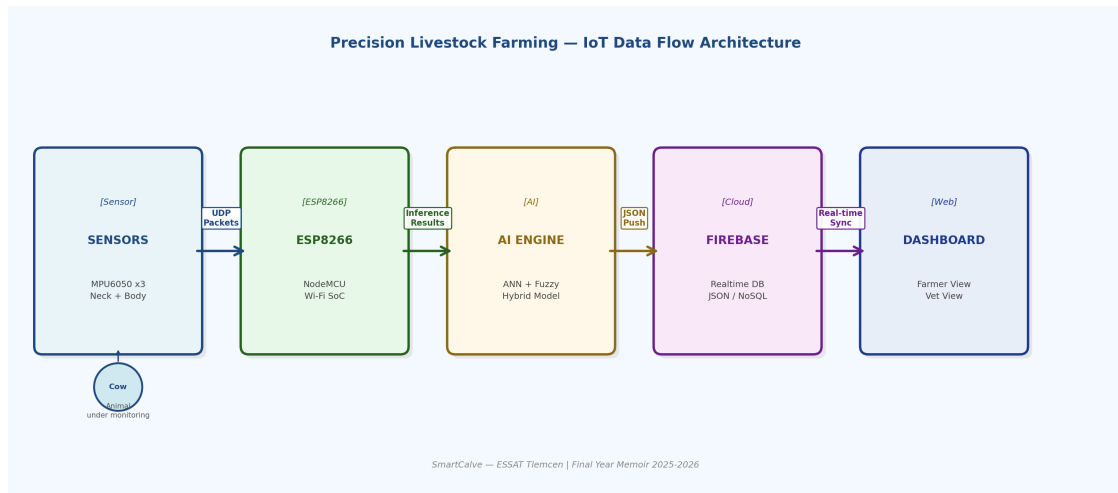


Figure 1.1: IoT data flow in Precision Livestock Farming (SmartCalve architecture overview)

1.1.2 Existing Calving Detection Solutions

Several commercial and academic solutions for automated calving detection have been developed and deployed internationally. Table 1.1 summarizes the most representative systems, comparing their sensor modality, detection approach, connectivity, and suitability for the Algerian context.

System / Study	Sensor Type	AI Method	Connectivity	Suitability for Algeria
Yoshida et al. (2022) [23]	Tail-attached IMU	Supervised ML (SVM/RF)	Wired logger	Low – proprietary hardware
Engelken et al. (2025) [5]	Neck accelerometer	Two-stage pipeline	LoRa / Cloud	Medium – LoRa requires infrastructure
Riaboff et al. (2024) [20]	Leg accelerometer	ROCKET / Catch22 + ML	Research logger	Low – research-only setup
Commercial (SCR / Allflex)	Ear tag / Pedometer	Proprietary threshold	Bluetooth / Cloud	Low – high cost, subscription
SmartCalve (This Work)	MPU6050 × 3	ANN + Fuzzy Logic hybrid	Wi-Fi / Fire-base	High – low-cost, open, local Wi-Fi

Table 1.1: Comparison of existing calving detection solutions

1.2 Embedded Systems & IoT Hardware

Embedded systems are specialized computing units designed to perform dedicated functions within a larger system, typically under real-time operational constraints. In IoT deployments, embedded systems serve as the edge nodes responsible for data acquisition, local processing, and wireless transmission. The choice of embedded platform determines the system’s power consumption, computational capacity, communication capabilities, and cost, all critical factors for a field-deployed livestock monitoring application.

1.2.1 Overview of Common Platforms

Three families of embedded platforms are widely used in IoT prototyping and deployment: Arduino-based microcontrollers, Wi-Fi enabled System-on-Chip (SoC) modules such as the ESP8266, and single-board computers (SBCs) such as the Raspberry Pi. Each presents distinct trade-offs in terms of processing power, connectivity, and energy consumption.

Platform	Processor	RAM	Wi-Fi	Power	Best Use Case
Arduino Uno	ATmega328P (8-bit, 16 MHz)	2 KB	No	Low	Simple sensors, no connectivity needed
ESP8266 (NodeMCU)	Tensilica L106 (32-bit, 80 MHz)	80 KB	Yes (802.11 b/g/n)	Low–Med	IoT edge node: sensor + Wi-Fi transmission
Raspberry Pi 4	ARM Cortex-A72 (64-bit, 1.5 GHz)	2–8 GB	Yes	High	Server / display / ML inference node

Table 1.2: Comparison of embedded platforms for IoT applications

1.2.2 The ESP8266 SoC

The ESP8266 is a low-cost, low-power Wi-Fi SoC developed by Espressif Systems [6, 7]. It integrates a 32-bit RISC processor running at up to 80 MHz, 80 KB of

RAM, and a full TCP/IP network stack, making it one of the most widely adopted microcontrollers for IoT edge applications. The NodeMCU development board, which pairs the ESP8266 with a USB-to-serial interface and voltage regulators, is the platform used in SmartCalve.

Key characteristics that make the ESP8266 suitable for this project include its native Wi-Fi connectivity enabling direct UDP transmission to the backend, its I2C peripheral support for interfacing with the MPU6050 sensors and TCA9548A multiplexer, its compatibility with the Arduino IDE programming environment [1], and its cost, typically under USD 3 per unit, which aligns with the low-cost deployment objective of SmartCalve.

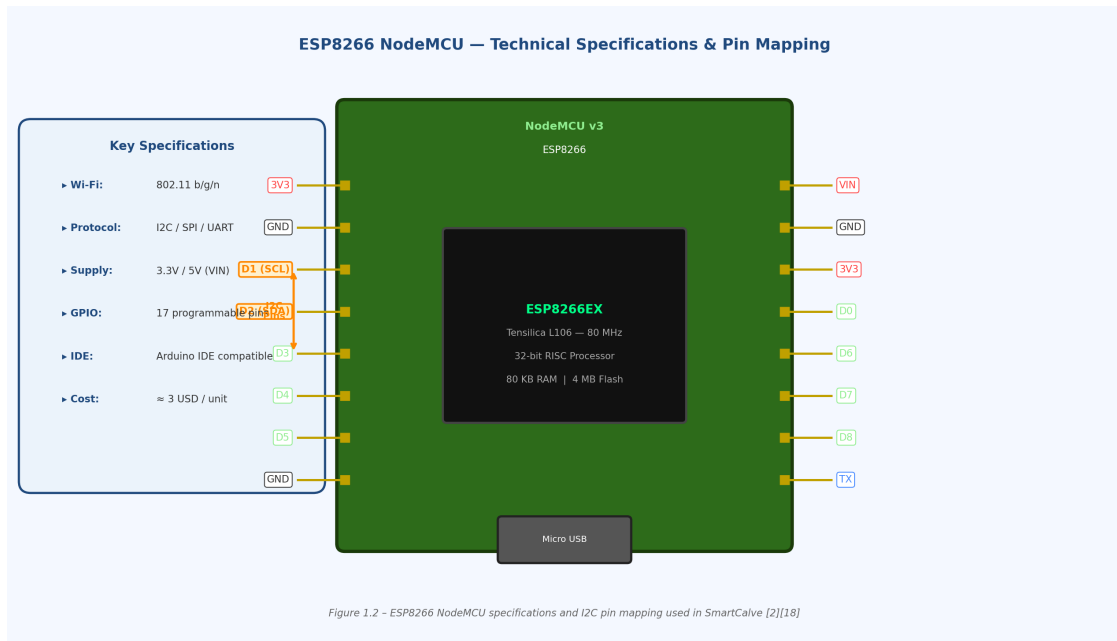


Figure 1.2: ESP8266 NodeMCU key specifications and I2C pin mapping

1.3 Sensors & Signal Acquisition

1.3.1 Inertial Measurement Units (IMU)

An Inertial Measurement Unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes magnetic field, using a combination of accelerometers, gyroscopes, and magnetometers. In livestock monitoring, IMUs are the dominant sensor modality for capturing animal movement and postural changes, owing to their small form factor, low power consumption, and ability to capture fine-grained motion signatures that correlate with behavioral states [3, 20].

Accelerometers measure linear acceleration along one or more axes, typically three, expressed in units of gravitational acceleration (g). Gyroscopes measure angular velocity in degrees per second. The combination of both, as found in six-axis IMU chips, provides a rich motion profile sufficient to distinguish behavioral states such as lying, standing, walking, and the restless pre-calving movement patterns that SmartCalve targets [23, 5].

1.3.2 The MPU6050 Sensor

The MPU6050 is a six-axis IMU manufactured by InvenSense (TDK) that integrates a three-axis accelerometer and a three-axis gyroscope on a single chip [11]. It communicates via the I2C protocol at a fixed address (0x68 or 0x69), supports programmable full-scale ranges for both accelerometer ($\pm 2g$ to $\pm 16g$) and gyroscope (± 250 to ± 2000 $^{\circ}/s$), and operates at a supply voltage of 2.375–3.46V, making it directly compatible with the ESP8266's 3.3V logic level.

SmartCalve uses three MPU6050 units simultaneously, one attached to the neck and two on the body flanks, to capture motion from multiple anatomical positions. This multi-sensor approach increases behavioral coverage and reduces the risk of missed events caused by partial movement or sensor occlusion. The use of three identical sensors on the same I2C bus necessitates an address multiplexing solution, as each MPU6050 has only two possible I2C addresses [11].

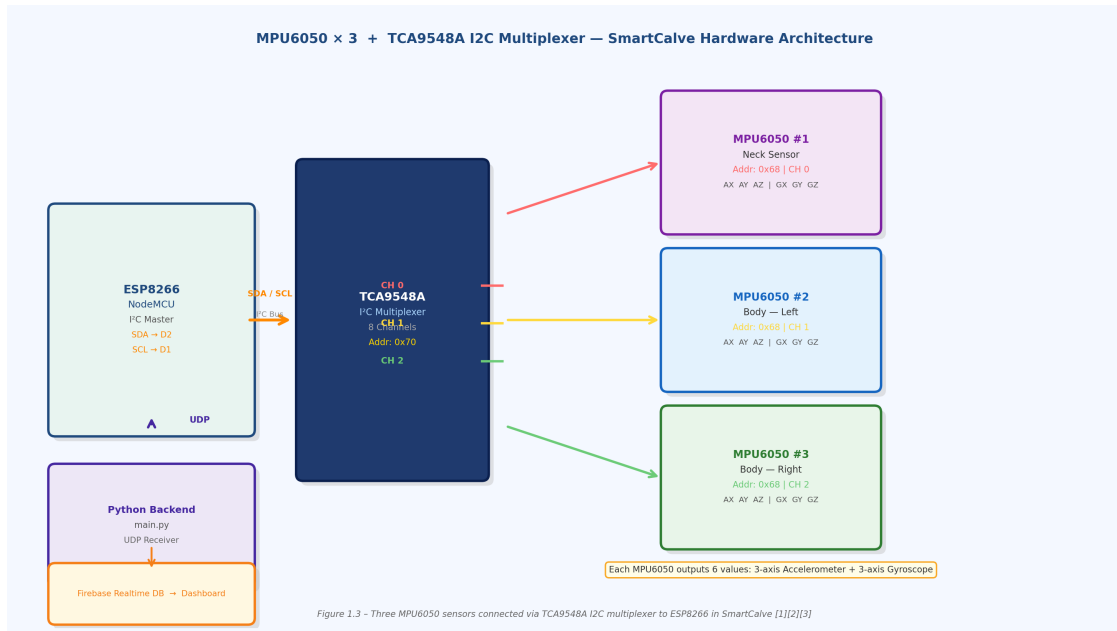


Figure 1.3: MPU6050 × 3 and TCA9548A I2C multiplexer wiring architecture in SmartCalve

1.3.3 The TCA9548A I2C Multiplexer

The TCA9548A is an 8-channel I2C switch manufactured by Texas Instruments [21]. It enables a single I2C master, in this case the ESP8266, to communicate with up to eight downstream I2C buses independently. By activating only one channel at a time, the master can address multiple devices sharing the same I2C address without conflict. In SmartCalve, each of the three MPU6050 sensors is connected to a separate channel of the TCA9548A, allowing sequential polling of all three sensors from a single ESP8266 I2C bus.

The TCA9548A operates at 2.3–5.5V, supports I2C clock frequencies up to 400 kHz (Fast Mode), and provides hardware address configuration via three address pins (A0–A2), giving it up to eight possible I2C addresses and allowing multiple TCA9548A devices to coexist on the same bus, a feature that enables future multi-cow expansion of the SmartCalve architecture [21].

1.4 Communication & Cloud Infrastructure

1.4.1 UDP Transmission Protocol

The User Datagram Protocol (UDP) is a connectionless transport layer protocol that transmits data packets without establishing a prior connection and without guaranteeing delivery, ordering, or error correction. While these characteristics make UDP less reliable than TCP for general network communication, they make it significantly faster and more appropriate for real-time sensor streaming applications where low latency is prioritized over delivery guarantees.

In SmartCalve, the ESP8266 transmits sensor data frames via UDP to a Python backend running on a local machine. Each UDP packet contains the averaged acceleration readings from the three MPU6050 sensors along with a timestamp. The backend receives these packets, assembles sliding windows of 20 consecutive readings, and passes them to the AI inference pipeline. The choice of UDP over TCP reduces per-packet overhead and eliminates connection setup latency, which is critical when processing sensor readings at a continuous acquisition rate.

1.4.2 Firebase Realtime Database

Firebase Realtime Database is a cloud-hosted NoSQL database provided by Google LLC that stores data as a JSON tree and synchronizes it in real time across all connected clients [9]. It is designed specifically for applications requiring live data synchronization such as IoT dashboards, collaborative tools, and alert systems where changes made by one client must be immediately reflected across all other connected interfaces.

In the SmartCalve architecture, Firebase serves as the central data bridge between the Python backend and the web dashboard. After each inference cycle, the backend pushes structured JSON records including sensor readings, ANN scores, fuzzy confidence levels, and detection status to a per-farmer node in the Firebase tree. The web dashboard subscribes to these nodes via the Firebase JavaScript SDK and updates its display in real time without requiring manual page refresh. This architecture enables both the farmer and the veterinarian to monitor the system from any device with a browser and an internet connection [9].

1.5 Machine Learning & Artificial Neural Networks

1.5.1 Supervised Learning and Classification

Machine learning (ML) is a subfield of artificial intelligence concerned with the development of algorithms that learn from data to make predictions or decisions without being explicitly programmed for each case [18, 13]. In the supervised learning paradigm, a model is trained on a labeled dataset, pairs of input features and known output classes, and learns to generalize from these examples to make correct predictions on unseen data.

Calving detection is formulated as a binary supervised classification problem: given a window of sensor readings, the model must predict whether the cow is in a pre-calving state (positive class) or in normal behavioral state (negative class).

This formulation is consistent with the approach adopted by Yoshida et al. [23] and Engelken et al. [5], who both applied supervised classifiers to labeled IMU time-series data for calving prediction.

1.5.2 Classical ML Methods

Classical supervised learning methods such as Support Vector Machines (SVM), Random Forests, and gradient boosting are widely used for sensor-based classification tasks because they can perform well on small to medium datasets and often require limited computational resources. However, their performance depends heavily on feature engineering, and their decision boundaries may be less flexible when modeling complex, non-linear motion patterns in multi-axis IMU data [13, 2].

1.5.3 Artificial Neural Networks

An Artificial Neural Network (ANN) is a computational model inspired by the structure of biological neural networks [10]. It consists of layers of interconnected nodes (neurons), where each connection carries a learnable weight. The network learns by adjusting these weights through backpropagation, iteratively minimizing the difference between predicted outputs and true labels using gradient descent optimization.

Feedforward ANNs with fully connected dense layers are particularly well suited to tabular sensor data classification. They can learn non-linear relationships between input features and output classes without requiring explicit feature engineering, and they can be efficiently deployed on standard hardware for real-time inference [22]. The SmartCalve ANN uses two hidden layers of 128 and 64 neurons respectively, with the hyperbolic tangent (tanh) activation function, trained on a labeled accelerometer dataset using the Scikit-learn library [18].

1.5.4 Handling Class Imbalance with SMOTE

A well-known challenge in calving detection is class imbalance: calving events are rare compared to normal behavior, causing classifiers trained on raw data to be biased toward the majority normal class and to systematically miss calving events. The Synthetic Minority Over-sampling Technique (SMOTE) addresses this by generating synthetic training examples in the minority class through interpolation between existing minority samples [4]. By balancing the class distribution before training, SMOTE significantly improves recall for the calving class, which is the most critical metric in this application, as a missed calving event is more costly than a false alarm.

1.6 Fuzzy Logic and the Hybrid AI Approach

1.6.1 Fuzzy Logic Foundations

Fuzzy Logic, introduced by Lotfi A. Zadeh in 1965 [24], is a form of multi-valued logic that extends classical Boolean logic to handle degrees of truth. Rather than requiring variables to be strictly true or false, fuzzy logic assigns membership values

between 0 and 1 to linguistic variables, such as “restlessness is HIGH” or “confidence is MEDIUM”. This makes it particularly appropriate for modeling uncertain, imprecise, or gradual phenomena such as the progressive behavioral changes that precede calving.

The Mamdani fuzzy inference system [14] is the most widely used fuzzy architecture in engineering applications. It operates through four steps: fuzzification, rule evaluation, aggregation, and defuzzification. In SmartCalve, the Fuzzy Logic System (FLS) takes seven raw sensor axis values directly as inputs A_X , B_X , B_Y , B_Z , C_X , C_Y , and C_Z and produces a crisp calving confidence score that is interpreted as a positive prediction when it exceeds 0.5.

1.6.2 Justification of the Hybrid Architecture

While a standalone ANN is capable of learning complex non-linear patterns, it produces a single probabilistic output that may be unreliable in edge cases where sensor readings are ambiguous or noisy. A standalone fuzzy system, on the other hand, provides interpretable linguistic outputs but lacks the ability to learn from data. The hybrid approach, combining both through a voting-based decision mechanism, leverages the complementary strengths of each component.

The voting system requires agreement between the ANN (threshold ≥ 0.2) and the Fuzzy Logic system (confidence = HIGH or MEDIUM) before a positive classification is recorded. An additional temporal filtering rule requiring 10 consecutive positive classifications before triggering a calving alert further reduces false positives caused by transient motion artifacts. This multi-layer decision architecture is specifically designed to balance sensitivity against specificity.

1.6.3 Comparison of AI Approaches for Calving Detection

Method	Learns from Data	Handles Un-certainty	Interpretable	Real-time Suitable	Chosen
SVM	Yes	No	Partial	Yes	No
Random Forest	Yes	No	Yes	Yes	No
LSTM / CNN	Yes	No	No	Limited	No
Fuzzy Logic (standalone)	No	Yes	Yes	Yes	No
ANN (standalone)	Yes	No	No	Yes	No
ANN + Fuzzy (Hybrid)	Yes	Yes	Partial	Yes	YES

Table 1.3: Comparison of AI methods for calving detection classification

1.7 Supervision Interfaces

1.7.1 SCADA and HMI Concepts

Supervisory Control and Data Acquisition (SCADA) systems are industrial-grade platforms used to monitor and control physical processes in real time. A SCADA

system typically consists of field sensors, remote terminal units (RTUs), a communication network, and a Human-Machine Interface (HMI) that provides operators with a visual representation of system state. While full SCADA systems are used in industrial environments such as power grids and water treatment plants, the conceptual framework, continuous sensor monitoring, alert generation, and operator visualization, directly applies to the SmartCalve supervision dashboard.

1.7.2 Raspberry Pi as a Local Display Platform

The Raspberry Pi is a single-board computer developed by the Raspberry Pi Foundation [19] that can run a full Linux operating system and drive graphical interfaces via HDMI output. In IoT deployments, Raspberry Pi boards are commonly used as local display terminals for sensor dashboards, running lightweight web browsers or dedicated GUI applications. For SmartCalve, a Raspberry Pi connected to a farm display screen could represent an ideal on-site terminal for the farmer view dashboard in a future deployment scenario, though the current implementation uses a standard browser-accessible web interface as the primary supervision solution.

1.7.3 Web Dashboard as the Adopted Solution

Given the constraints of the development environment and the goal of accessibility from multiple devices, SmartCalve implements its supervision interface as a browser-based web dashboard built with HTML, CSS, and JavaScript [17]. This approach provides the full functional equivalent of a local HMI while remaining accessible from any device, PC, tablet, or smartphone, with a browser and an internet connection.

The dashboard connects directly to Firebase Realtime Database via the Firebase JavaScript SDK, receiving live updates without polling. It provides two dedicated views: a Farmer View displaying real-time cow status, motion levels, and alert notifications in simplified language; and a Veterinarian View providing detailed ANN confidence scores, fuzzy logic outputs, and historical event logs. Both views support dark and light themes and are fully responsive.

1.8 Synthesis & Project Positioning

This section synthesizes all the theoretical elements presented in this chapter into a coherent justification of the SmartCalve technology stack. Table 1.4 maps each system component to the alternatives considered and provides an explicit rationale for the choice made.

Component	Alternatives Considered	Chosen Solution	Justification
Microcontroller	Arduino Uno, Raspberry Pi	ESP8266 (NodeMCU)	Native Wi-Fi, I2C support, low cost, Arduino IDE compatible [6]
Motion Sensor	ADXL345, LSM6DS3, MPU9250	MPU6050 (×3)	6-axis IMU, I2C, well-documented, ultra-low cost [11]
I2C Multiplexing	Separate I2C buses, address hacks	TCA9548A	Clean channel isolation for 3 sensors at same address [21]
AI Model	SVM, RF, LSTM, standalone ANN	ANN + Fuzzy Logic (hybrid)	Combines learning from data with uncertainty handling; suitable for real-time edge inference [10, 24]
Class Imbalance	Class weights, undersampling	SMOTE	Generates synthetic minority samples; improves recall in imbalanced datasets [4]
Cloud Database	MQTT broker, InfluxDB, REST API	Firebase Realtime DB	Built-in real-time sync, JSON-native, free tier sufficient, JS SDK for dashboard [9]
Communication	TCP/HTTP, MQTT, BLE	UDP	Lowest latency for continuous sensor streaming; reliable over local Wi-Fi
Dashboard	Raspberry Pi GUI, Blynk, Grafana	HTML/JS Web Dashboard	Device-agnostic, no subscription, Firebase SDK native integration [17]
Supervision Model	Single flat view	Farmer + Vet dual views	Role-based access: simplified alerts for farmers, detailed diagnostics for veterinarians

Table 1.4: SmartCalve technology synthesis: choices and justifications

The synthesis presented in Table 1.4 demonstrates that every component of the SmartCalve system was selected based on a reasoned comparison of available alternatives, with explicit consideration of cost, deployment constraints, technical compatibility, and the specific requirements of the calving detection problem. This coherent, justified technology stack forms the foundation upon which the system design and implementation presented in Chapter 2 is built.

This chapter has established the theoretical and technological foundations of the SmartCalve system. The review of Precision Livestock Farming confirmed the relevance of automated calving detection as a high-priority agricultural application. The comparative analysis of embedded platforms, sensors, communication protocols, cloud infrastructure, and AI methods led to a justified technology stack centered on the ESP8266, three MPU6050 sensors, UDP transmission, Firebase Realtime Database, and a hybrid ANN and Fuzzy Logic inference engine. Every component choice was driven by the constraints of low-cost local deployment and the specific requirements of real-time calving detection. Chapter 2 builds directly on these foun-

dations, presenting the complete design and implementation of the system across all four architectural layers.

SYSTEM DESIGN & IMPLEMENTATION

This chapter presents the complete design and implementation of the SmartCalve system. It covers every layer of the architecture, from the physical hardware assembly and sensor wiring, through data acquisition and wireless transmission, to artificial intelligence model development, cloud integration, and web-based supervision dashboard. Each design decision is grounded in the theoretical foundations established in Chapter 1, and is accompanied by the concrete implementation details that translate those foundations into a functional, end-to-end calving detection system.

2.1 System Architecture Overview

The SmartCalve system is organized as a layered pipeline in which data flows continuously from physical sensors attached to a monitored cow, through a wireless transmission layer, into an AI inference engine, and finally into a cloud-synchronized supervision dashboard. This pipeline is illustrated conceptually in Figure 2.1 and can be decomposed into four distinct functional layers.

The Acquisition Layer consists of three MPU6050 inertial measurement units connected through a TCA9548A I2C multiplexer to an ESP8266 Wi-Fi microcontroller [6]. This layer is responsible for sampling raw six-axis motion data from the cow at a configurable rate and packaging it into UDP datagrams for wireless transmission.

The Processing Layer is a Python backend running on a local machine (laptop or Raspberry Pi) that receives UDP packets, assembles sliding windows of sensor readings, and feeds them in real time to the hybrid AI inference engine composed of an Artificial Neural Network and a Fuzzy Logic System.

The Cloud Layer is implemented using Firebase Realtime Database, which receives structured JSON records from the Python backend after each inference cycle and makes them available to all subscribed clients instantaneously through its real-time synchronization protocol.

The Supervision Layer is a browser based web dashboard that connects directly to Firebase via the JavaScript SDK. It presents live cow status, AI model scores, motion history, and alert notifications to both farmers and veterinarians, with role-differentiated views and configurable alert settings.

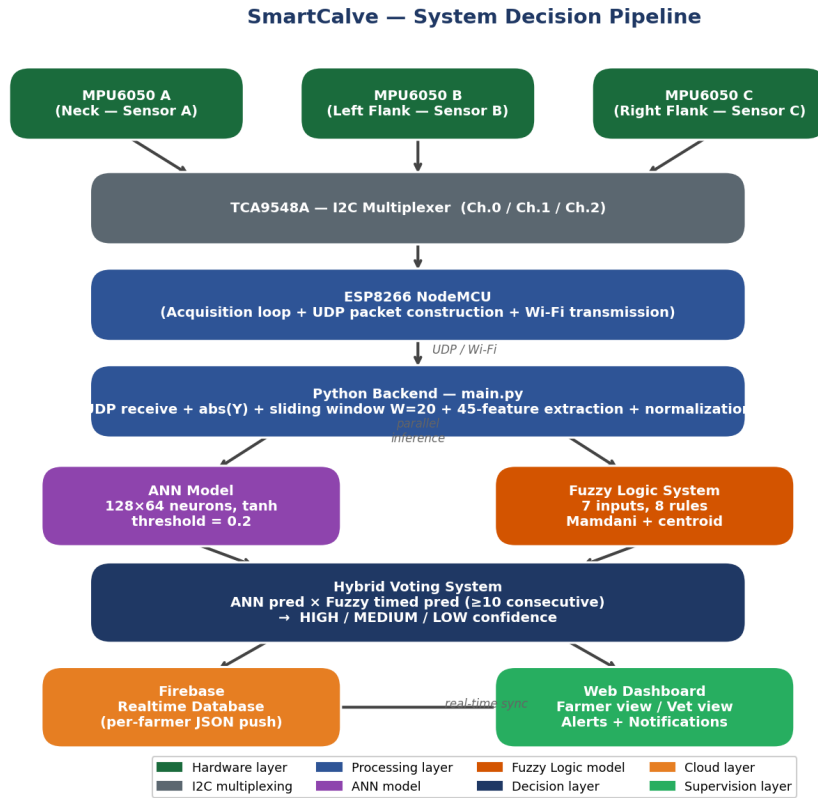


Figure 2.1: SmartCalve four-layer system architecture

Layer	Components	Technology	Role
Acquisition	MPU6050 ×3, TCA9548A, ESP8266	I2C + UDP + Wi-Fi	Sensor data capture & transmission
Processing	Python backend (main.py)	ANN + Fuzzy Logic	AI inference & decision
Cloud	Firestore Realtime Database	NoSQL + JSON sync	Data storage & broadcast
Supervision	Web Dashboard (HTML/JS)	Firestore JS SDK	Real-time monitoring & alerts

Table 2.1: SmartCalve system layers and their components

2.2 Hardware Design

The hardware subsystem of SmartCalve is designed around three core requirements: multi point motion capture, I2C address conflict resolution, and low-cost wireless transmission. The following subsections detail the assembly choices, sensor placement strategy, and wiring architecture that satisfy these requirements.

2.2.1 Sensor Placement Strategy

Three MPU6050 sensors are used in SmartCalve, designated Sensor A, Sensor B, and Sensor C. All three sensors are mounted at successive positions along the cow's tail, from the tail head downward, establishing a distributed Cartesian measurement system along the tail's longitudinal axis [11]. Sensor A is placed at the proximal tail position (closest to the tail head), Sensor B at the mid-tail region, and Sensor C at the distal tail section. A fixed reference origin point O is defined at the tail base, relative to which the 3D acceleration coordinates of each sensor are tracked in real time, as illustrated in Figure 2.2.

This tail-based placement strategy is grounded in well-documented bovine pre-calving behavior: as parturition approaches, the cow's tail undergoes characteristic patterns of elevation, lateral deviation, and agitated movement that intensify progressively in the hours preceding birth [23, 5]. The tail therefore constitutes the most behaviorally informative and mechanically accessible anatomical region for non-invasive motion sensing. Furthermore, distributing the three sensors along the tail's length rather than at a single point enables the system to capture differential motion between proximal and distal tail segments, yielding a richer 3D behavioral fingerprint that improves detection accuracy.

2.2.2 I2C Multiplexing Architecture

Each MPU6050 exposes only two selectable I2C addresses (0x68 and 0x69), making it impossible to connect three identical units to the same I2C bus without address conflicts. The TCA9548A I2C multiplexer resolves this constraint by providing eight independently switchable downstream channels, each presenting its own isolated I2C segment to the master device [21].

In the SmartCalve wiring, Sensor A is connected to channel 0 of the TCA9548A, Sensor B to channel 1, and Sensor C to channel 2. The ESP8266 communicates with the multiplexer at its address (0x70 by default) and activates each channel sequentially before issuing read commands to the MPU6050 on that channel. After completing the read, the channel is deactivated before the next sensor is polled, ensuring complete electrical isolation between readings and eliminating cross-talk.

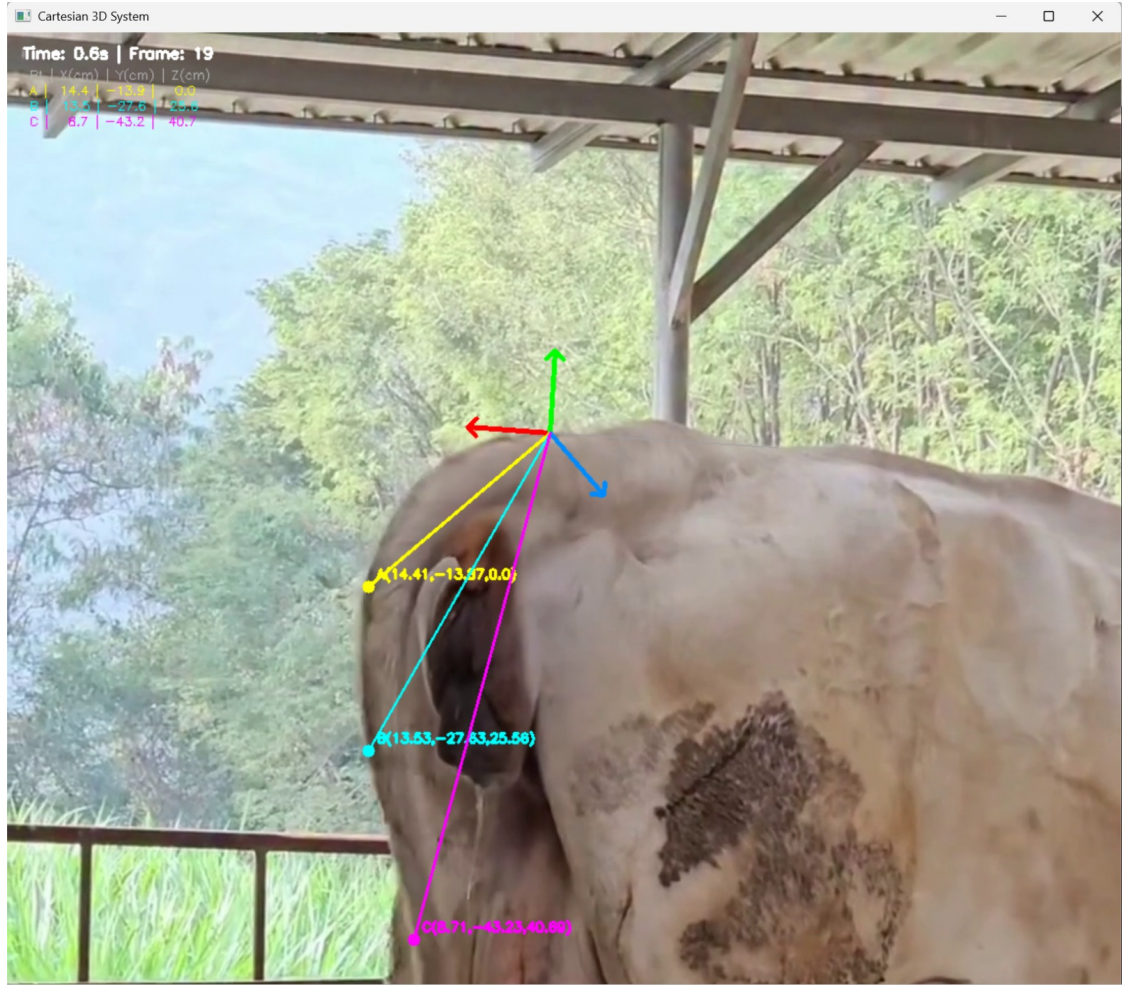


Figure 2.2: SmartCalve tail sensor placement: Sensors A (proximal), B (mid-tail), and C (distal) mounted along the cow’s tail, with origin O at the tail base. Live 3D coordinates shown in upper-left corner.

2.2.3 Y-Axis Absolute Value Convention

A critical preprocessing decision at the hardware level concerns the Y-axis readings of all three sensors. Because sensor orientation on the cow’s body cannot be guaranteed to be perfectly consistent across mounting sessions, the Y-axis acceleration may be recorded as positive or negative depending on how the sensor is physically attached. To eliminate this ambiguity and ensure that the AI models receive consistent, sign-invariant inputs, all Y-axis values are converted to their absolute value before any further processing, i.e., $|A_Y|$, $|B_Y|$, $|C_Y|$. This convention is applied both in the Python backend for UDP mode and directly in the dataset preprocessing for training mode.

Sensor ID	Designation	Placement	TCA9548A Channel	I2C Address
A	Proximal tail	Proximal position (tail head)	Channel 0	0x68
B	Mid-tail	Mid-tail region	Channel 1	0x68
C	Distal tail	Distal tail section	Channel 2	0x68

Table 2.2: MPU6050 sensor placement, channel assignment, and I2C addressing

2.3 Data Acquisition & UDP Transmission

2.3.1 Acquisition Loop on the ESP8266

The ESP8266 firmware implements a continuous acquisition loop that cycles through the three MPU6050 sensors in sequence. For each sensor, the firmware activates the corresponding TCA9548A channel, reads the raw six-axis IMU registers, converts the raw integer values to physical units using the MPU6050's configured sensitivity scale factors, and stores the result. After reading all three sensors, the firmware constructs a comma-separated data frame and transmits it as a single UDP packet to the Python backend.

The data frame has the following structure: `farmer_id`, `cow_id`, `timestamp_ms`, `r_A`, `$\theta_A$` , `$\phi_A$` , `r_B`, `$\theta_B$` , `$\phi_B$` , `r_C`, `$\theta_C$` , `$\phi_C$` a total of twelve fields. Sensor data is transmitted in spherical coordinates and converted to Cartesian values by the Python backend upon reception. Both the farmer and cow identifiers are embedded in every packet, enabling the backend to support simultaneous multi-farmer, multi-cow data streams from different ESP8266 nodes without requiring separate sockets.

2.3.2 UDP Transmission Protocol

UDP was selected as the transport protocol for sensor-to-backend communication for the reasons established in Chapter 1: its connectionless, low-overhead nature makes it ideally suited for continuous, high-frequency sensor streaming over a local Wi-Fi network where packet loss is negligible and latency is the primary concern. The Python backend binds a UDP socket on port 4210 at the configured IP address and enters a blocking receive loop, processing each incoming packet as it arrives.

Below is a pseudocode representation of the backend's UDP reception and routing logic:

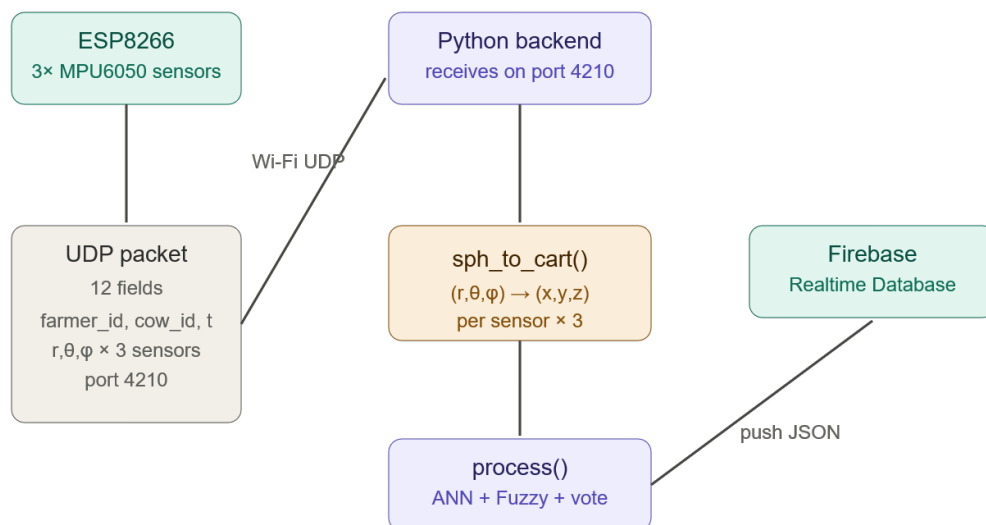


Figure 2.3: SmartCalve UDP data flow architecture

2.4 Data Preprocessing

2.4.1 Sliding Window Assembly

Raw sensor readings arrive as individual time-stamped rows, each containing nine acceleration values from the three sensors. A single row carries insufficient contextual information for reliable behavioral classification; the AI models require a temporal window of successive readings to detect the motion patterns that distinguish pre-calving behavior from normal activity.

SmartCalve uses a sliding window of minimum size $W = 20$ rows. The backend maintains a rolling buffer for each cow: as each new row arrives, it is appended to the buffer. Inference begins once the buffer reaches 20 rows and continues for every subsequent row until all data is processed, ensuring that the complete dataset contributes to the analysis. The buffer is never flushed between windows; each new incoming row advances the window by one, providing a smooth, overlapping analysis of the motion stream.

2.4.2 Feature Extraction for ANN Input

The ANN model does not consume the raw 20×9 window matrix directly. Instead, five statistical descriptors are extracted from each of the nine sensor channels across the 20-row window, producing a fixed-length feature vector of $9 \times 5 = 45$ features. The five descriptors are: mean, standard deviation, delta (last minus first value), maximum, and minimum. These descriptors capture both the central tendency and the dynamic range of motion within the window, providing the ANN with a compact yet informative representation of the behavioral episode.

2.4.3 Normalization

Before being fed to the ANN, the 45-element feature vector is standardized using a `StandardScaler` fitted during the training phase and saved to disk as `smartcalve_scaler.pkl`. The scaler subtracts the training-set mean and divides by the training-set standard deviation for each feature, ensuring that all features occupy a comparable numerical range and preventing features with large absolute values from dominating the network's gradient updates during training.

Preprocessing Step	Parameter	Purpose
Y-axis abs()	A_Y, B_Y, C_Y	Eliminate mounting orientation ambiguity
Sliding window	$W = 20$ rows	Capture temporal motion context
Feature extraction	$5 \text{ stats} \times 9 \text{ channels} = 45$	Compact temporal representation for ANN
StandardScaler	Zero mean, unit variance	Normalize feature ranges for stable training

Table 2.3: Data preprocessing pipeline summary

2.5 ANN Model

2.5.1 Training Dataset

The ANN is trained on a custom labeled accelerometer dataset created by the authors and published on the Kaggle platform under the account mohamedtessoumi, dataset identifier: Kaggle_Dataset_Final.csv. The dataset contains sensor recordings collected from instrumented cattle, covering both normal and pre-calving behavioral classes.

The dataset is preprocessed using the sliding window pipeline described in Section 2.4 (window size $W = 20$, 45 features per sample). The train/test split is performed using an 80/20 stratified partition (`random_state = 42`) to preserve the class ratio across both subsets. To address the class imbalance introduced after windowing, the Synthetic Minority Over-sampling Technique (SMOTE) is applied exclusively on the training partition [4], generating synthetic minority-class samples by interpolating between existing pre-calving feature vectors in the 45-dimensional feature space. The test set is kept entirely unmodified to ensure that evaluation metrics reflect realistic field conditions.

2.5.2 Network Architecture

The SmartCalve ANN is a feedforward multi-layer perceptron implemented using the Scikit-learn `MLPClassifier` [18]. It receives the 45-element normalized feature vector as input and produces a binary classification probability as output. The network architecture consists of two hidden layers: the first contains 128 neurons and the second contains 64 neurons. Both hidden layers use the hyperbolic tangent ($\tanh$) activation function, chosen for its symmetric output range $[-1, +1]$ and its tendency to produce stronger gradient signals during backpropagation compared to the rectified linear unit (ReLU) in low-dimensional tabular classification tasks.

The output layer uses a softmax function to produce class probabilities. The model is trained using the Adam optimizer with L2 regularization ($\alpha = 0.001$) and early stopping based on validation loss to prevent overfitting. The trained model and scaler are serialized to disk using joblib for deployment.

Parameter	Value
Dataset	Kaggle_Dataset_Final.csv (mo-hamedtessoumi/calving)
Train / Test split	80% / 20% stratified
Class balancing	SMOTE on training set only
Input features	45 (5 statistics $\times$ 9 sensor channels)
Hidden layers	128 neurons $\rightarrow$ 64 neurons
Activation function	Tanh (hidden layers), Softmax (output)
Optimizer	Adam
Regularization	L2 (alpha = 0.001)
Max iterations	500
Decision threshold	0.2 (optimized for recall)
Implementation	Scikit-learn MLPClassifier, serialized via joblib

Table 2.4: SmartCalve ANN architecture and training parameters

2.6 Fuzzy Logic Model

2.6.1 Input Variables and Universe of Discourse

The Fuzzy Logic System (FLS) operates directly on the raw sensor values of the current frame rather than on extracted statistical features, complementing the ANN by providing a rule-based assessment of instantaneous motion intensity [24]. Seven input linguistic variables are defined, each corresponding to a sensor axis positioned along the cow’s tail: A_X (proximal tail sensor X axis), B_X, B_Y, B_Z (mid-tail sensor), and C_X, C_Y, C_Z (distal tail sensor). The universe of discourse for each variable is derived from the empirical range of values observed in the training dataset and is listed in Table 2.6.

The absolute value convention is applied to B_Y and C_Y before input to the fuzzy system, consistent with the preprocessing step described in Section 2.4. Only the A_X axis from Sensor A is retained as a fuzzy input; the A_Y and A_Z axes of Sensor A are excluded as they were found to contribute less discriminative information during the rule design phase, given that the dominant pre-calving motion of the proximal tail segment is primarily captured along the X axis. This exclusion was determined empirically by observing that A_Y and A_Z values showed minimal variance contrast between normal and pre-calving classes in the training dataset.

2.6.2 Membership Functions

Each input variable is partitioned into two triangular membership functions: ‘normal’ and ‘calving’. Triangular membership functions were chosen for their computational simplicity and interpretability, which aligns with the engineering transparency objectives of the hybrid system. The transition boundaries between the two member-

ship functions are set at the empirical midpoints of the observed value distributions for normal versus pre-calving behavior, as extracted from the training dataset.

The output variable, named 'calving', is defined over the range [0, 1] and partitioned into two triangular membership functions: 'normal' (peak at 0.0) and 'calving' (peak at 1.0), with the crossover point at 0.5. The crisp output score produced by defuzzification is interpreted as a continuous calving confidence index, with values above 0.5 classified as a fuzzy positive prediction. The remaining input variables (B_X, B_Y, C_Z) follow the same triangular partitioning structure and are omitted from the figure for brevity

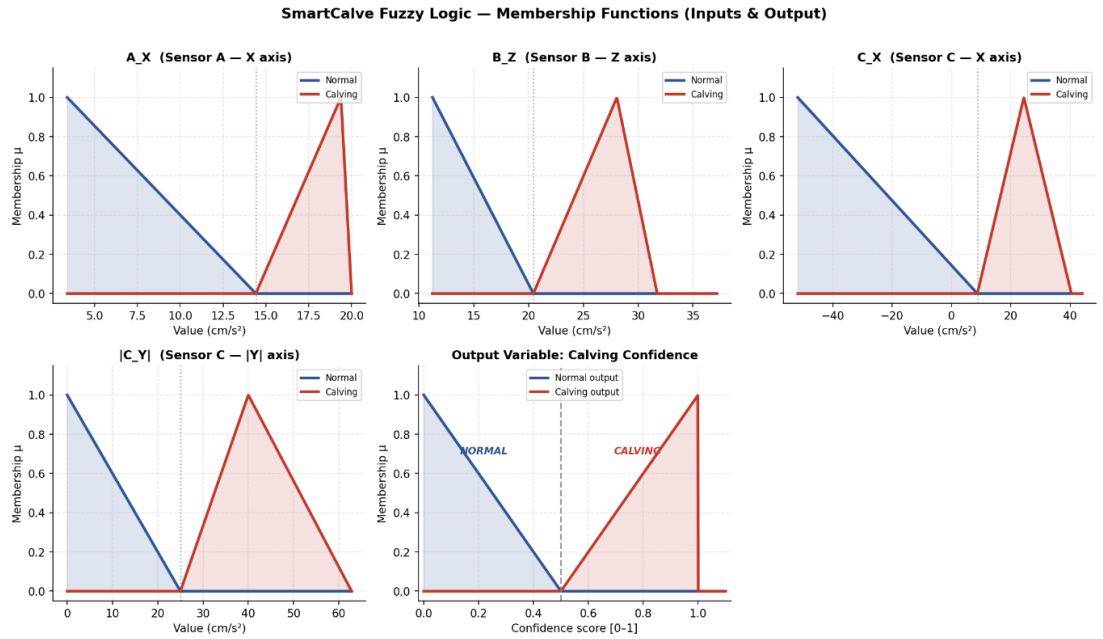


Figure 2.4: SmartCalve Fuzzy Logic membership functions: four representative input variables (A_X, B_Z, C_X, |C_Y|) and the output calving confidence variable with decision boundary at 0.5

2.6.3 Inference Rules

The FLS uses a Mamdani inference engine [14] with eight handcrafted IF-THEN rules derived empirically from the observed value distributions of the training dataset, reflecting the motion patterns associated with bovine pre-calving behavior. All rules are listed in Table 2.6.

Rule	IF condition	THEN output
R1	A_X is Calving AND B_Z is Calving AND C_Z is Calving	Calving
R2	B_X is Calving AND C_X is Calving AND C_Z is Calving	Calving
R3	A_X is Normal AND B_Z is Normal AND C_Z is Normal	Normal
R4	B_X is Normal AND C_X is Normal	Normal
R5	A_X is Calving AND C_Z is Calving	Calving
R6	B_Z is Calving AND C_X is Calving	Calving
R7	B_Y is Calving AND C_Y is Calving	Calving
R8	B_Y is Normal AND C_Y is Normal	Normal

Table 2.5: SmartCalve Fuzzy Logic inference rules (Mamdani, 8 rules)

The defuzzification method used is centroid (center of gravity), which computes the crisp output score as the center of mass of the aggregated fuzzy output region. This method produces a smooth, continuous confidence score that varies gradually with changing input conditions, making it well-suited to the hybrid voting system described in Section 2.7.

2.7 Hybrid Decision System

2.7.1 Temporal Filtering on Fuzzy Output

The fuzzy system evaluates each incoming sensor frame independently, which makes it sensitive to transient motion spikes that do not represent genuine pre-calving behavior. To prevent such transient events from triggering false alerts, a consecutive-positive counter is applied to the fuzzy output: the fuzzy system is considered to have produced a timed positive prediction only if it has classified at least 10 consecutive frames as positive (fuzzy_score ≥ 0.5). As soon as a single negative frame is detected, the counter resets to zero. This temporal filter acts as a debouncing mechanism, ensuring that only sustained motion patterns consistent with the multi-minute behavioral changes that precede calving propagate to the voting stage.

2.7.2 Voting Logic

The final decision for each inference cycle is produced by combining the ANN prediction and the temporally filtered fuzzy prediction through a two-model voting scheme. The voting rules are designed to produce three distinct confidence levels: HIGH, MEDIUM, and LOW, each mapped to a specific decision label and alert behavior.

ANN Prediction	Fuzzy Timed Prediction	Confidence	Decision Label
Positive (≥ 0.2)	Positive (≥ 10 consecutive)	HIGH	PRE-CALVING ALERT
Positive (≥ 0.2)	Negative	MEDIUM	WARNING (ANN detects)
Negative	Positive (≥ 10 consecutive)	MEDIUM	WARNING (Fuzzy detects)
Negative	Negative	LOW	Normal

Table 2.6: Hybrid voting decision table: ANN $\times$ Fuzzy $\rightarrow$ Confidence level

HIGH confidence corresponds to bilateral agreement between both models and triggers a three-channel alert: (1) a Firebase push records the event in the history sub-tree for permanent logging, (2) the web dashboard displays a full-width red alert banner identifying the affected cow, and (3) a Blynk mobile push notification is sent to the farmer’s smartphone requiring immediate intervention. MEDIUM confidence represents a unilateral detection by one model; Firebase records the event as a warning and Blynk sends a pre-calving warning notification, while the dashboard displays a reduced-severity alert. LOW confidence results in no alert and records only the routine sensor status update.

2.7.3 Backend Processing Pseudocode

The following pseudocode summarizes the complete per-frame processing logic of the Python backend, integrating data reception, preprocessing, ANN inference, fuzzy inference, temporal filtering, voting, and Firebase push:

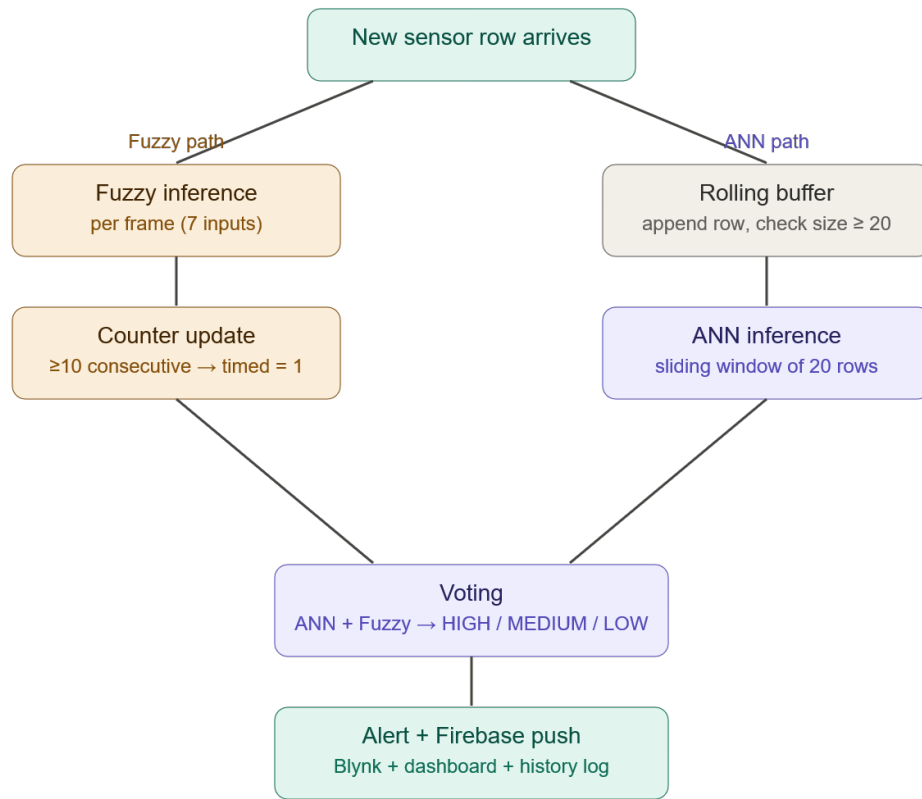


Figure 2.5: SmartCalve per-frame processing pipeline

2.8 Firebase Integration

2.8.1 Database Structure

The Firebase Realtime Database for SmartCalve is organized under the root node 'smartcalve', with two primary sub-trees: 'farmers' and 'cows'. The system is designed to support an arbitrary number of farmers, each identified by a unique farmer key (farmer1, farmer2, farmer3, ..., farmerN). The three farmer instances used in the demonstration scenario are illustrative examples created to validate the multi-farmer architecture the system places no fixed limit on the number of registered farmers or monitored cows, and new farmers can be added by simply registering a new key in the database hierarchy. Under each farmer node, a 'cows' sub-tree holds the latest state record for each monitored cow, and a 'history' sub-tree accumulates timestamped event records for all HIGH and MEDIUM confidence detections [9].

Each cow state record written by the backend contains the following fields: cow_id, farmer identifier, UTC timestamp, elapsed time t, ANN score, fuzzy score, fuzzy counter value, decision label, confidence level, and last-seen time. This structured record gives the web dashboard all the data it needs to render a complete, role-appropriate view without any additional computation on the client side.

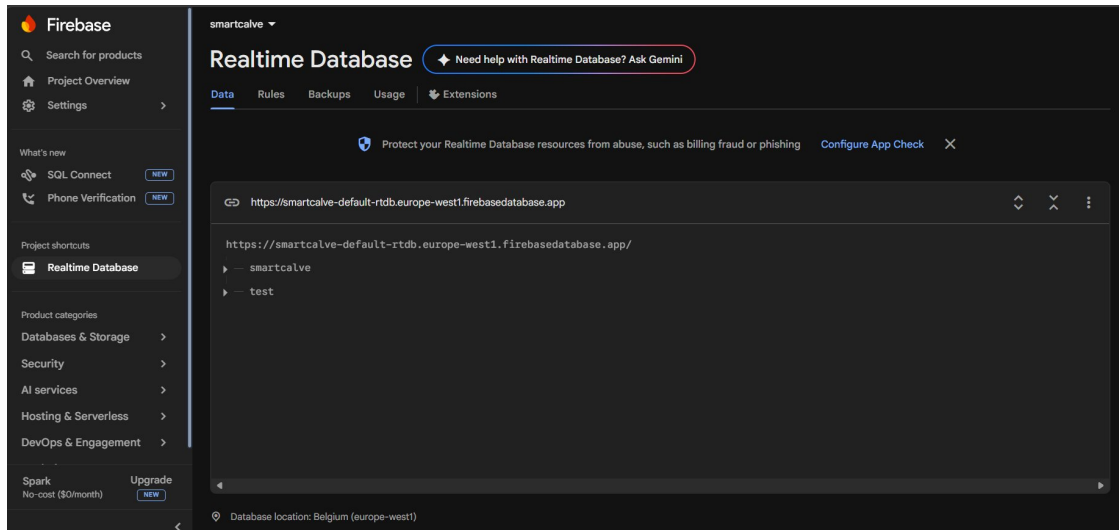


Figure 2.6: Firebase Realtime Database root structure: 'smartcalve' and 'test' nodes

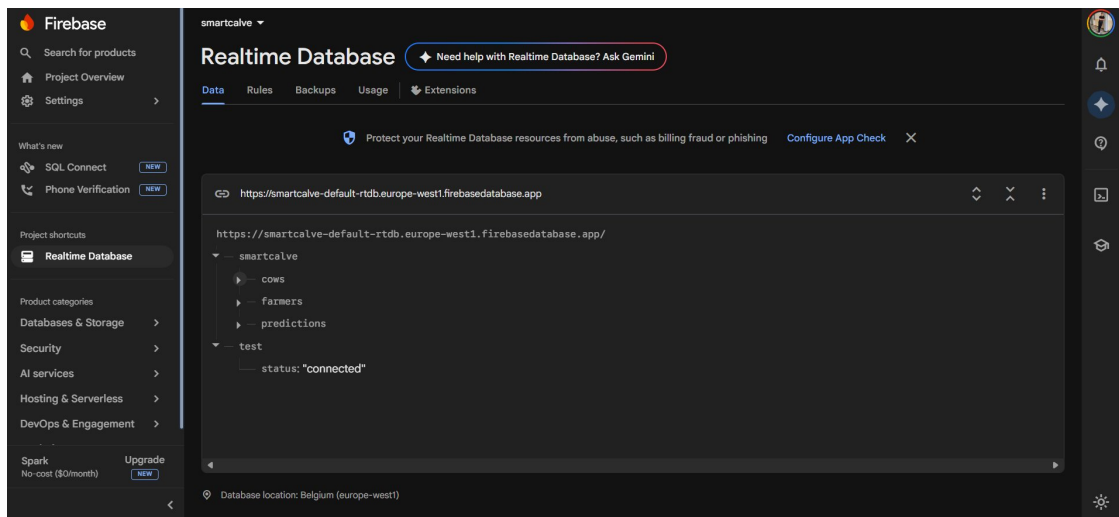


Figure 2.7: Firebase database expanded: 'cows', 'farmers', and 'predictions' sub-trees

2.8.2 Real-Time Push Logic

After each inference cycle, the Python backend pushes the current cow state to Firebase using the `firebase-admin` Python SDK. The push uses a `set()` operation (overwrite) on the cow's latest-state node, ensuring that the dashboard always sees the most recent reading without accumulating redundant records. For HIGH and MEDIUM confidence events, an additional `push()` operation (append with auto-generated key) records the event in the history sub-tree, preserving a full audit trail of all detected calving events across the session.

The Firebase SDK handles authentication using a service account key file (`serviceAccountKey.json`) loaded at startup. The database URL points to the Europe West 1 region (Belgium), which was selected to minimize latency for Algerian deployments. The connection is initialized once at program start and reused for all

subsequent push operations, avoiding the overhead of repeated authentication handshakes.

2.9 Web Dashboard

2.9.1 Architecture and Technology Stack

The SmartCalve web dashboard is a single-page application (SPA) built with four files: `index.html`, `app.js`, `data.js`, and `style.css`, using HTML5, CSS3, and vanilla JavaScript with no server-side framework. It connects directly to Firebase Realtime Database using the Firebase JavaScript SDK v9, subscribing to real-time data change listeners that update the UI automatically whenever the backend pushes new data. This architecture eliminates the need for polling and ensures that the displayed information is never more than a few hundred milliseconds behind the latest sensor reading.

The dashboard implements role-based access control at the UI level: a login screen distinguishes between farmer accounts and veterinarian accounts, routing each role to a dedicated view with appropriate information density and interaction options. Two complete interface themes, light and dark, are supported, with the user's preference persisted in the session.

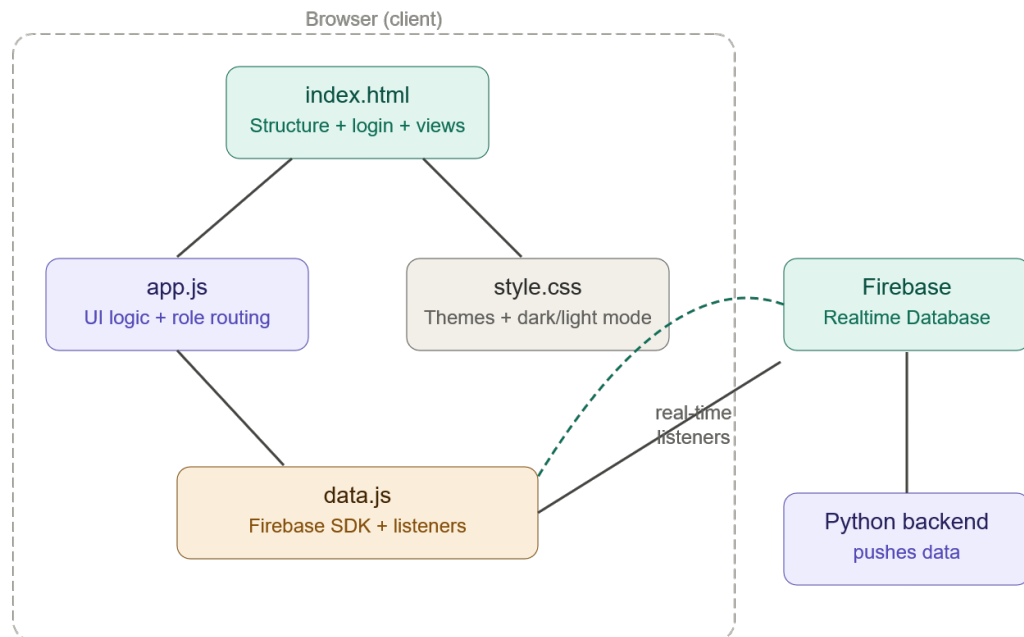


Figure 2.8: SmartCalve frontend architecture: `index.html`, `app.js`, `data.js`, `style.css` connected to Firebase Realtime Database

Login Page

The login page serves as the entry point to the CalvingSafe dashboard. It presents the CalvingSafe branding alongside the ESSA Tlemcen institutional logo, and displays a real-time connection status indicator confirming that the system is online and

Firebase is connected. The user selects their role, either Farmer or Veterinarian, before entering credentials. Each role is authenticated against a predefined set of accounts, with the selected role determining the view and level of diagnostic detail presented after login. The login page is fully responsive and supports both light and dark display modes.

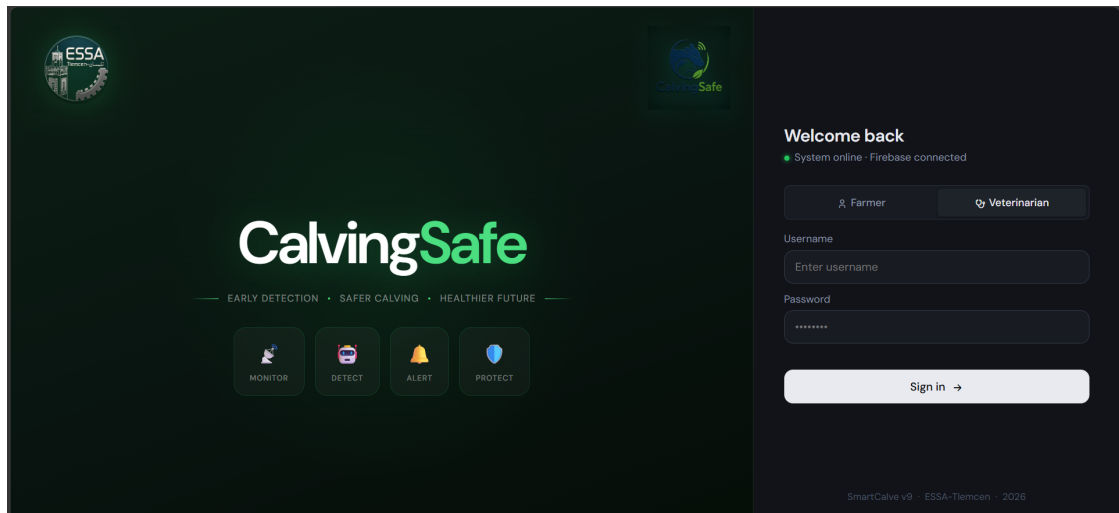


Figure 2.9: CalvingSafe login page: role selection (Farmer / Veterinarian), Firebase connection status indicator, and CalvingSafe branding

2.9.2 Farmer View

The farmer view presents each monitored cow as a card displaying its current status badge (Normal, Pre-Calving, or Calving!), risk level (Low, Critical), last update time, and ANN and Fuzzy percentage scores. An animated bar chart within each card provides a visual history of the most recent detection states, giving the farmer an immediate sense of the cow's behavioral trend without requiring any interpretation of numerical scores.

A persistent header displays the real-time herd summary: total cows monitored, count in each status category (Normal, Pre-Calving, Calving), and the current system time. When a HIGH-confidence calving event is detected, a full-width alert banner appears at the top of the screen identifying the affected cow and prompting immediate intervention.

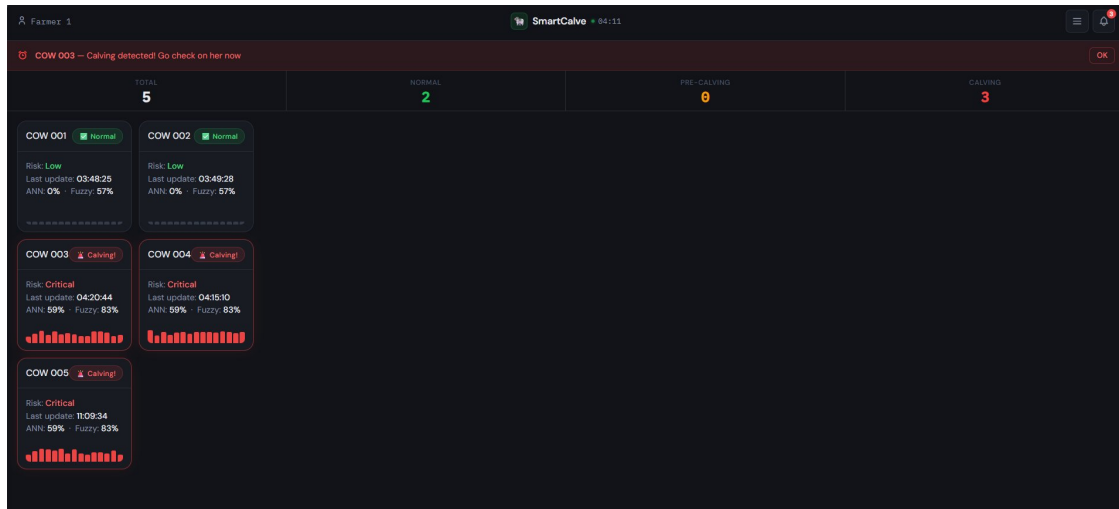


Figure 2.10: Farmer view (dark mode): Farmer 1 dashboard showing 5 monitored cows, 3 in Calving status

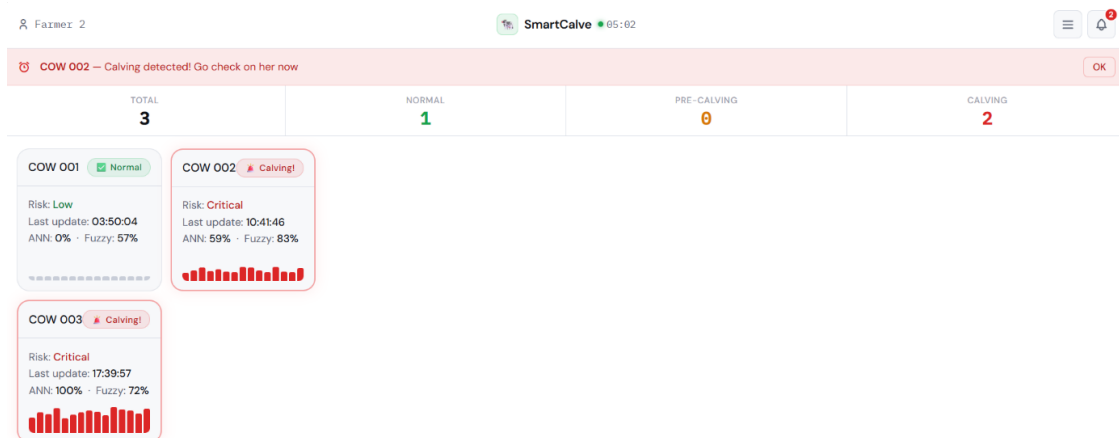


Figure 2.11: Farmer view (White mod): Farmer 2 dashboard showing 3 monitored cows, 2 in Calving status

2.9.3 Veterinarian View

The veterinarian view provides a higher-level perspective across all registered farms. Upon login, the veterinarian sees a list of all farmers with their farm names, cow counts, and the number of cows currently in each alert state. Selecting a farmer reveals the detailed diagnostic view for that farm, presenting the same cow cards as the farmer view but with additional numerical detail and access to the full event history log.

The veterinarian header displays an aggregated system-wide summary: total number of farms monitored, total cows across all farms, and the global counts of Normal, Pre-Calving, and Calving cows. This bird's-eye view allows a single veterinarian to simultaneously monitor multiple farms and prioritize interventions based on alert severity.

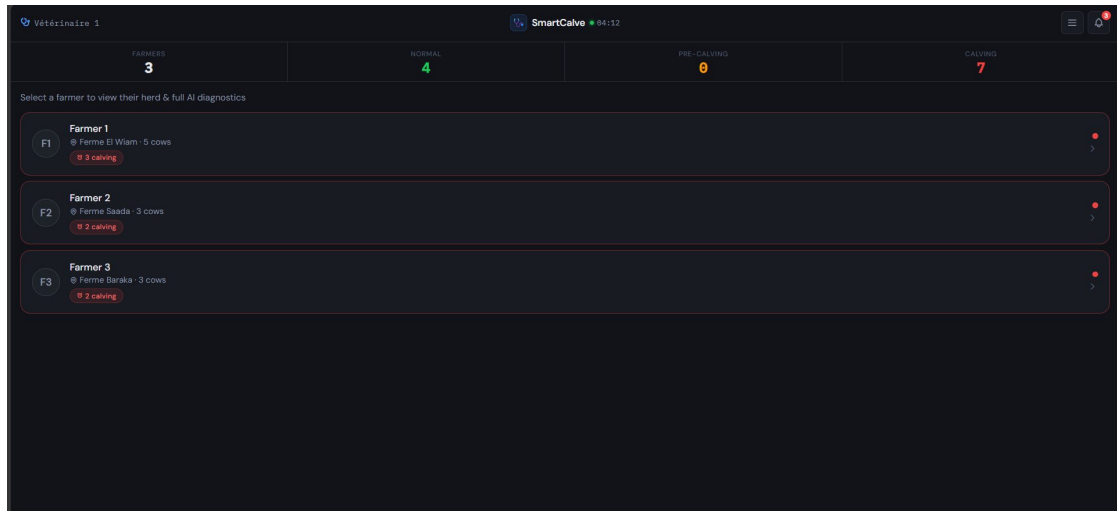


Figure 2.12: Veterinarian view: farmer list with farm names, cow counts, and calving alert badges (3 farmers, 7 total calving events)

Selecting a cow card opens a detailed diagnostic overlay, shown in Figure 2.12. This modal displays the cow's current status, the last update timestamp, and a contextual alert message. The AI model results section presents the ANN confidence score and the Fuzzy Logic score as visual progress bars, with a confirmation that both models executed successfully. Below, the veterinarian can access the clinical notes section to consult or add observations, and a dedicated prescriptions panel to record and track treatments. These sections are restricted to the veterinarian role and are not visible to farmers.

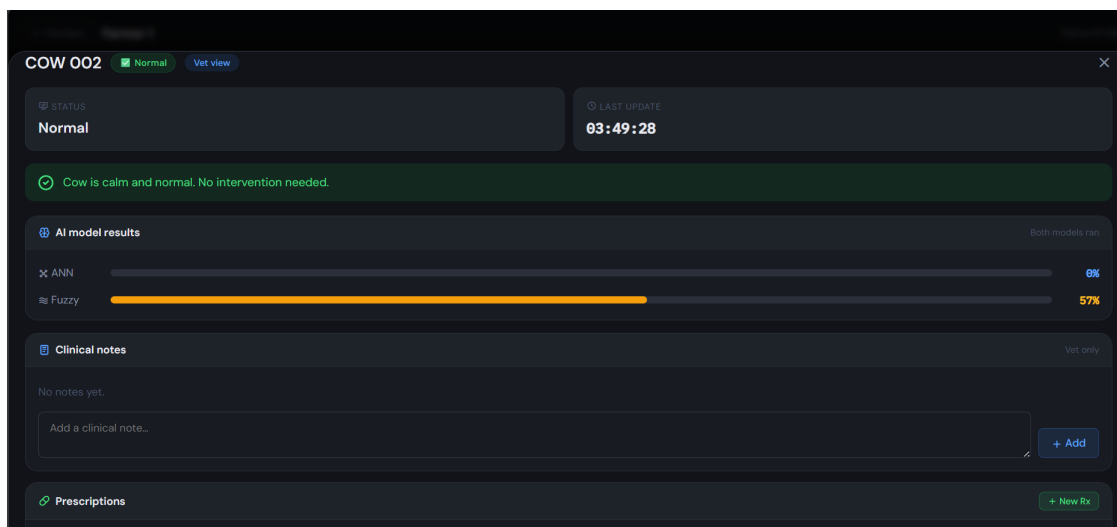


Figure 2.13: Cow detail modal in veterinarian view: status, AI model results (ANN and Fuzzy Logic scores), clinical notes, and prescriptions panel.

2.9.4 Alert Notification System

SmartCalve implements a three-channel alert notification system. The first channel is the in-dashboard visual alert: a full-width red banner identifying the affected cow appears at the top of the screen and persists until acknowledged by the user.

The second channel is Firebase event logging: every HIGH or MEDIUM confidence event is pushed to the history sub-tree, creating a permanent timestamped audit trail accessible to both farmers and veterinarians. The third channel is Blynk mobile push notification: when a HIGH confidence event is detected, a critical alert is sent to the farmer's smartphone via the Blynk IoT platform; MEDIUM confidence events trigger a pre-calving warning notification. Together, these three channels ensure that no calving event goes unnoticed regardless of whether the farmer is actively watching the dashboard.

2.10 Conclusion

This chapter has presented the complete design and implementation of the Smart-Calve system across all four architectural layers. The hardware layer assembles three MPU6050 sensors mounted along the cow's tail through a TCA9548A multiplexer to an ESP8266 NodeMCU, enabling distributed tail-motion capture across proximal, mid, and distal positions with I2C address conflict resolution. The processing layer implements a continuous UDP data pipeline feeding a 45-feature sliding-window representation into a two-hidden-layer ANN (128×64 neurons, tanh activation, threshold = 0.2, SMOTE-balanced) and a seven-input Mamdani fuzzy logic system with eight inference rules derived empirically from the training dataset. The hybrid voting mechanism combines both model outputs with a 10-consecutive-frame temporal filter to produce three confidence levels (HIGH, MEDIUM, LOW), minimizing both missed detections and false alarms. The cloud layer uses Firebase Realtime Database to synchronize per-cow state and event history in real time across all connected clients. The supervision layer delivers a role-differentiated web dashboard separate farmer and veterinarian views with configurable alert settings, dark/light themes, and a three-channel notification system combining in-dashboard visual alerts, Firebase event logging, and Blynk mobile push notifications.

Chapter 3 will present the testing methodology adopted for system validation, the AI model performance results, and a discussion of the system's demonstrated capabilities and known limitations.

CHAPTER 3

TESTING, VALIDATION & DISCUSSION

This chapter presents the testing methodology adopted for the SmartCalve system, the validation results obtained across all system components, and a critical discussion of the system's demonstrated capabilities and known limitations. The chapter begins by establishing the testing protocol, explaining the constraints that led to a simulation-based approach, and justifying the academic validity of the adopted methodology. It then presents the development environment and deployment configuration, followed by the system validation results covering AI model performance, pipeline latency, Firebase synchronization, and dashboard behavior. The chapter concludes with a discussion of limitations and a roadmap of future improvements.

3.1 Testing Methodology

Validating an IoT-based livestock monitoring system ideally requires deployment in a real farm environment with live animals, real sensor data, and observable calving events [3]. This section describes the intended real-life testing protocol, the practical constraints that prevented its implementation, and the simulation-based methodology adopted as a rigorous and academically sound alternative.

3.1.1 Intended Real-Life Testing Protocol

The ideal testing scenario for SmartCalve would involve the following protocol. Three MPU6050 sensors would be attached to an instrumented dairy cow along the tail, one at the proximal position, one at the mid position, and one at the distal position, using non-invasive elastic harnesses. The ESP8266 NodeMCU, powered by a rechargeable battery pack, would be mounted on the animal and configured to transmit sensor readings via UDP to a backend computer connected to the same local Wi-Fi network.

Under this protocol, the system would be validated over a continuous monitoring period spanning at least one complete calving event. The following metrics would be measured in field conditions: detection lead time (how many minutes before actual calving the system first triggers an alert), false alarm rate during normal behavior periods, end-to-end latency from sensor reading to dashboard alert, Firebase synchronization reliability under continuous data stream, and dashboard display correctness under real-time update conditions. The Arduino IDE Serial Monitor

would be used to verify real-time sensor output, and the Firebase console would be monitored for correct data structure and update frequency.

3.1.2 Constraints and Limitations

The ideal testing protocol described above could not be implemented during the development period of this project for two fundamental reasons.

The first constraint is the absence of farm access. No suitable dairy farm facility was available to the project team during the development and testing period. Accessing an active calving facility requires veterinary authorization, animal welfare compliance, and logistical coordination with farm operators that could not be arranged within the project timeline.

The second constraint is the rarity and unpredictability of calving events. Even with farm access, a calving event cannot be scheduled or reliably predicted more than a few hours in advance. Continuous on-site monitoring over multiple days would be required to capture a statistically meaningful number of real calving events, which was not feasible within the academic project framework.

3.1.3 Adopted Solution: Two-Mode Operation

To address these constraints while preserving the validity of the system validation, SmartCalve implements two distinct operating modes that share the same backend processing pipeline, Firebase integration, and web dashboard. This approach is consistent with established practices in embedded systems and IoT prototyping [23, 5], where pre-recorded datasets are used to validate inference pipelines when live sensor data is unavailable.

The first mode is the Real Hardware Mode. In this mode, the physical ESP8266 NodeMCU is programmed with the SmartCalve firmware using the Arduino IDE and flashed to the microcontroller via USB. Upon boot, the ESP8266 connects to the local Wi-Fi network, activates the TCA9548A multiplexer, and begins cycling through the three MPU6050 sensors. For each cycle, it reads the raw accelerometer values from all three sensors, converts them from Cartesian to spherical coordinates (r , θ , ϕ), constructs a 12-field comma-separated UDP packet, and transmits it to the Python backend at port 4210 every 2000 milliseconds. The Python backend receives each packet, validates the field count, converts the spherical coordinates back to Cartesian values, and feeds the data into the processing pipeline. This mode is the intended deployment configuration for real farm use.

The second mode is the CSV Simulation Mode. In this mode, the physical ESP8266 is not required. The Python backend is launched directly with a pre-recorded labeled CSV dataset, which is processed row by row through the same sliding window, feature extraction, ANN inference, Fuzzy Logic inference, voting, and Firebase push pipeline as in the real hardware mode. The command used for the test session documented in this chapter is:

```
python main111.py excel cow_002 farmer3
```

This command loads the CSV file, processes all rows sequentially, and pushes the final detection result to Firebase at the end of the session. The number of rows processed depends on the dataset loaded; the system processes all available rows

with inference beginning from row 20 onwards once the sliding window buffer reaches its minimum size.

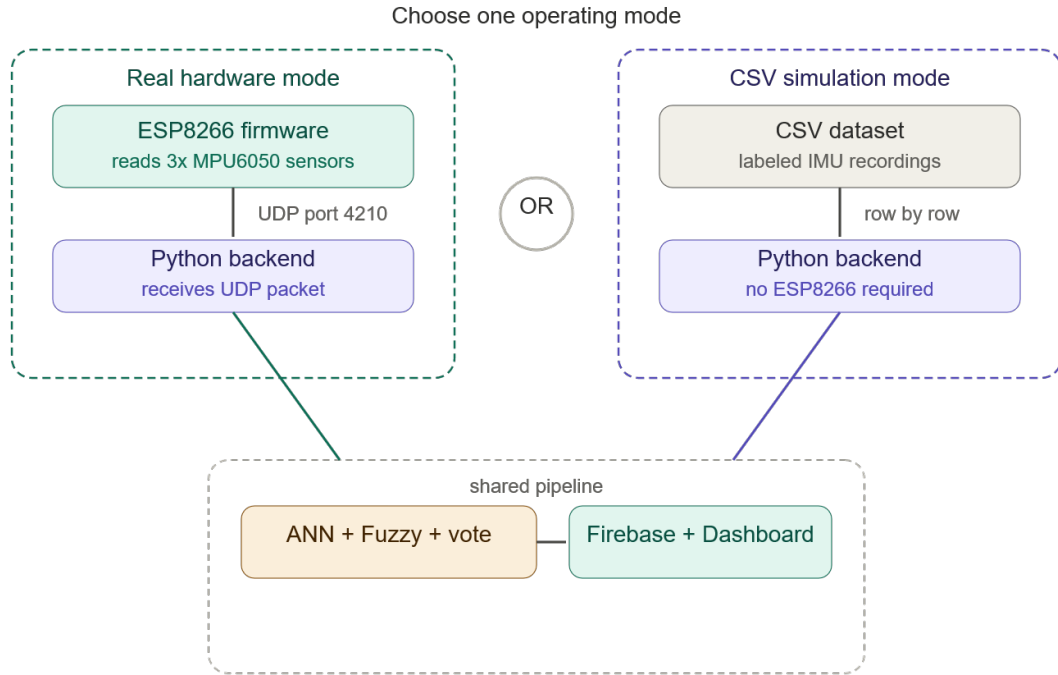


Figure 3.1: SmartCalve two-mode operation: Real Hardware Mode (ESP8266 via UDP to Python) and CSV Simulation Mode (CSV to Python directly)

3.1.4 ESP8266 Firmware Description

The SmartCalve ESP8266 firmware is developed using the Arduino IDE [1] and written in C++. It implements the complete hardware acquisition and transmission layer of the system. The firmware begins by initializing the I2C bus on GPIO4 (SDA) and GPIO5 (SCL), connecting to the local Wi-Fi network, and opening a UDP socket on port 4211. The main loop executes every 2000 milliseconds and performs the following sequence of operations.

First, the firmware selects each TCA9548A multiplexer channel in turn (channels 0, 1, and 2 corresponding to Sensors A, B, and C respectively) by writing a channel-select byte to the multiplexer at I2C address 0x70. For each channel, it wakes the MPU6050 by writing to the power management register, reads six bytes of raw accelerometer data from register 0x3B, and converts the 16-bit integer values to physical units in centimeters using the configured scale factors (20 cm for Sensor A, 40 cm for Sensor B, 60 cm for Sensor C).

Second, after reading all three sensors, the firmware converts each sensor's Cartesian vector (x, y, z) to spherical coordinates (r, θ, ϕ) using the `cartToSph()` function, which computes the radius as the vector magnitude, the polar angle θ using the arccosine of z/r , and the azimuthal angle ϕ using $\text{atan2}(y, x)$. The resulting nine spherical values, together with the farmer identifier, cow identifier, and elapsed timestamp, are formatted into a 12-field comma-separated packet and transmitted via UDP to the configured backend IP address and port.

The Serial Monitor output of the firmware displays the sensor status (OK or – for disconnected sensors) and the raw Cartesian values for each sensor at every transmission cycle, enabling real-time hardware diagnostics during development and testing.

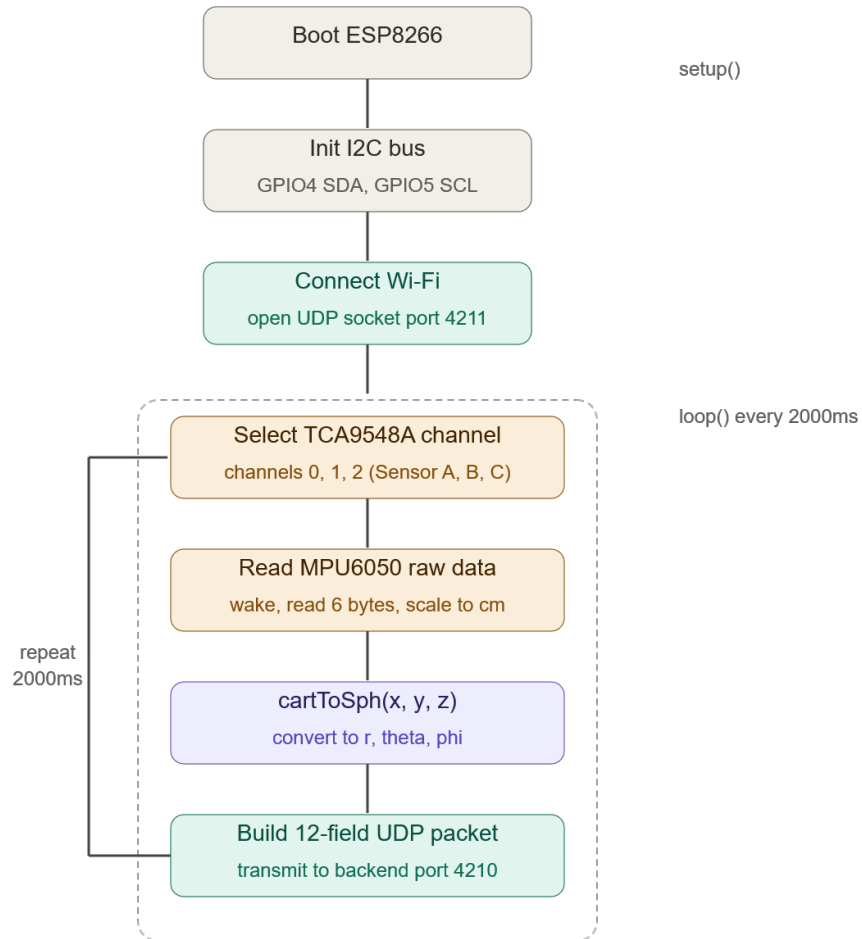


Figure 3.2: Figure 3.2 ESP8266 firmware acquisition loop: TCA9548A channel selection, MPU6050 raw read, Cartesian-to-spherical conversion, and UDP packet transmission

This table present lists of the Arduino libraries used in the ESP8266 firmware implementation.

Tool / Library	Role
ESP8266WiFi	Wi-Fi connection management
WiFiUdp	UDP packet transmission
Wire	I2C bus communication
MPU6050 by Electronic Cats	Raw accelerometer data reading

Table 3.1: ESP8266 firmware Arduino libraries

3.1.5 Test Session Description

The test session documented in this chapter was conducted using the CSV Simulation Mode with a labeled dataset containing rows of IMU sensor readings for cow_002 assigned to farmer3. The dataset was structured with normal behavior rows first followed by pre-calving rows, providing a realistic behavioral transition scenario for the inference pipeline. Table 3.1 summarizes the key parameters of the test session.

Parameter	Value	Description
Test cow identifier	cow_002	Simulated cow ID embedded in every packet
Farmer identifier	farmer3	Parent farmer node in Firebase hierarchy
Window size	20 rows	Minimum buffer required before first inference
Collecting phase	Rows 1 to 19	System accumulates data; no inference is performed
First inference	Row 20	First complete window triggers ANN + Fuzzy inference
Decision	PRE-CALVING ALERT	Hybrid voting result
Confidence	High	Pre-calving warning level triggered
Blynk notification	Sent successfully	Push notification fired upon detection
Firebase push	Confirmed	Record written to the Firebase cow node for farmer3/cow_002

Table 3.2: Test session parameters and results summary for cow_002 / farmer3

3.2 Development Environment and Deployment

This section describes the software tools, libraries, and deployment infrastructure used in the SmartCalve implementation.

3.2.1 Development Tools and Libraries

Tool / Library	Role
Python	Backend runtime environment
scikit-learn	MLPClassifier (ANN), StandardScaler, SMOTE [4]
scikit-fuzzy	Mamdani fuzzy inference engine
firebase-admin	Firestore Realtime Database Python SDK
pandas / numpy	CSV processing and numerical operations
joblib	ANN model and scaler serialization
requests	Blynk HTTP notification API
Arduino IDE	ESP8266 firmware development and flashing
VS Code	Frontend development (HTML/CSS/JS)

Table 3.3: SmartCalve development tools and libraries

3.2.2 Web Dashboard Deployment on Vercel

The CalvingSafe web dashboard is deployed on Vercel, a cloud platform for static web application hosting. The dashboard source files (index.html, app.js, data.js, style.css) are hosted on Vercel under the project name smartcalve and are accessible via the public URL smartcalve.vercel.app. The deployment is configured as a Production environment and was last updated during the final development phase of the project in June 2026

Vercel was selected for dashboard deployment for three reasons. First, it provides zero-configuration static hosting with automatic HTTPS, eliminating the need for a dedicated web server at the farm. Second, it supports direct file deployment via Vercel Drop, enabling rapid redeployment whenever the dashboard code is updated. Third, it is available on a free tier that is sufficient for the SmartCalve use case, consistent with the low-cost deployment objective of the project.

The dashboard operates as a purely client-side application: all data fetching and real-time updates are handled directly by the Firebase JavaScript SDK running in the browser, with no server-side computation required on the Vercel hosting infrastructure. This means the dashboard remains fully functional as long as the user's browser has internet connectivity to reach both the Vercel CDN and the Firebase Realtime Database.

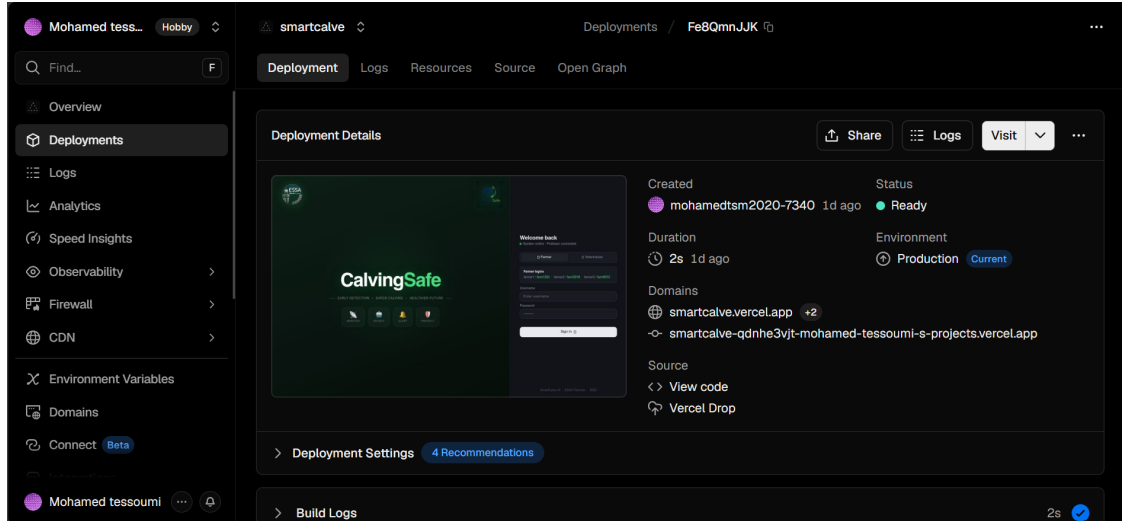


Figure 3.3: Vercel deployment dashboard: smartcalve project, Production environment, smartcalve.vercel.app domain

3.3 System Validation Results

This section presents the validation results for each component of the SmartCalve system, organized by subsystem. The results are based on the test session described in Section 3.1.5 and on observations made during repeated simulation runs.

3.3.1 Data Collection and Window Assembly

The sliding window assembly mechanism performed correctly throughout the test session. For the first 19 rows of the dataset, the system correctly identified that the buffer had not yet reached the required window size of 20 rows and displayed a collecting data message with the current buffer fill count. This behavior confirms that the buffer management logic correctly enforces the minimum window size requirement before triggering any inference, preventing premature or unreliable predictions based on insufficient data.

```
SmartCalve - [cow_002] Testing on 21 rows
```

t	ANN	Fuzzy	Decision
[farmer3/cow_002] t=0.0			Collecting data... (1/20)
[farmer3/cow_002] t=0.2			Collecting data... (2/20)
[farmer3/cow_002] t=0.4			Collecting data... (3/20)
[farmer3/cow_002] t=0.6			Collecting data... (4/20)
[farmer3/cow_002] t=0.8			Collecting data... (5/20)
[farmer3/cow_002] t=1.0			Collecting data... (6/20)
[farmer3/cow_002] t=1.2			Collecting data... (7/20)
[farmer3/cow_002] t=1.4			Collecting data... (8/20)
[farmer3/cow_002] t=1.6			Collecting data... (9/20)
[farmer3/cow_002] t=1.8			Collecting data... (10/20)
[farmer3/cow_002] t=2.0			Collecting data... (11/20)
[farmer3/cow_002] t=2.2			Collecting data... (12/20)
[farmer3/cow_002] t=2.4			Collecting data... (13/20)
[farmer3/cow_002] t=2.6			Collecting data... (14/20)
[farmer3/cow_002] t=2.8			Collecting data... (15/20)
[farmer3/cow_002] t=3.0			Collecting data... (16/20)
[farmer3/cow_002] t=3.2			Collecting data... (17/20)
[farmer3/cow_002] t=3.4			Collecting data... (18/20)
[farmer3/cow_002] t=3.6			Collecting data... (19/20)

Figure 3.4: Terminal output showing the data collection phase for cow_002

3.3.2 AI Inference Pipeline Validation

Upon completion of the dataset processing, the Python backend produced the following terminal output:

```
[Blynk] Notification sent: cow_002 is PRE-CALVING - check the cow soon.
SmartCalve - Final Result for [cow_002]
Rows processed : [N rows]
Decision : WARNING (Pre-calving detected)
```

The system correctly processed all rows, applied the sliding window inference from row 20 onwards, and produced a MEDIUM confidence pre-calving warning. The Blynk push notification was dispatched successfully before the final result output, confirming that the alert transmission occurs as the first action upon detection, prioritizing notification delivery.

3.3.3 Firebase Synchronization Validation

Following the inference cycle, the Python backend executed the Firebase push function, writing the complete detection record to the smartcalve/farmers/farmer3/cows/cow_002 node. The Firebase Realtime Database connection was confirmed stable throughout the session. Because the confidence level was High, the push function also appended a history record to smartcalve/farmers/farmer3/history/cow_002, creating an auditable alert log entry with the full detection data including timestamp, ANN score, fuzzy score, and decision label.

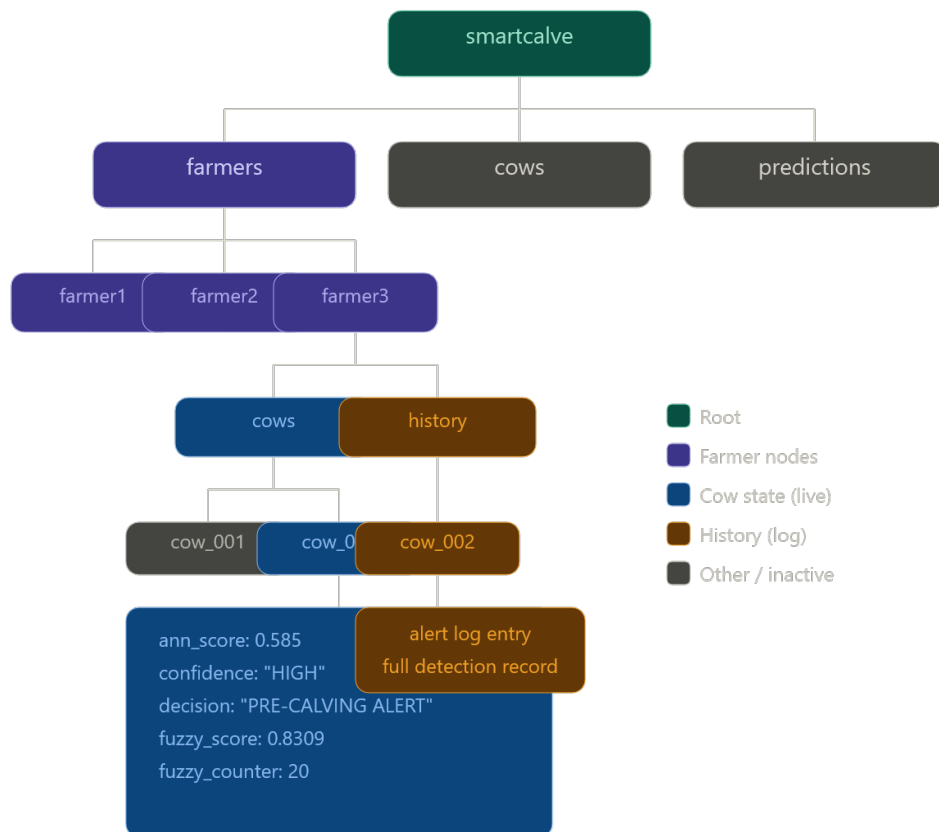


Figure 3.5: Firebase Realtime Database console showing cow_002 state record written under farmer3 hierarchy

3.3.4 Dashboard Validation

The CalvingSafe web dashboard, accessible at smartcalve.vercel.app, correctly reflected the detection results in real time. Upon receiving the Firebase update, the dashboard updated the COW 002 card status badge, risk level indicator, ANN and Fuzzy score displays, and activity bar chart without manual page refresh. The full-width alert banner appeared at the top of the farmer view screen, and both the Farmer View and the Veterinarian View correctly reflected the updated system state. Both dark and light theme modes displayed alerts correctly.

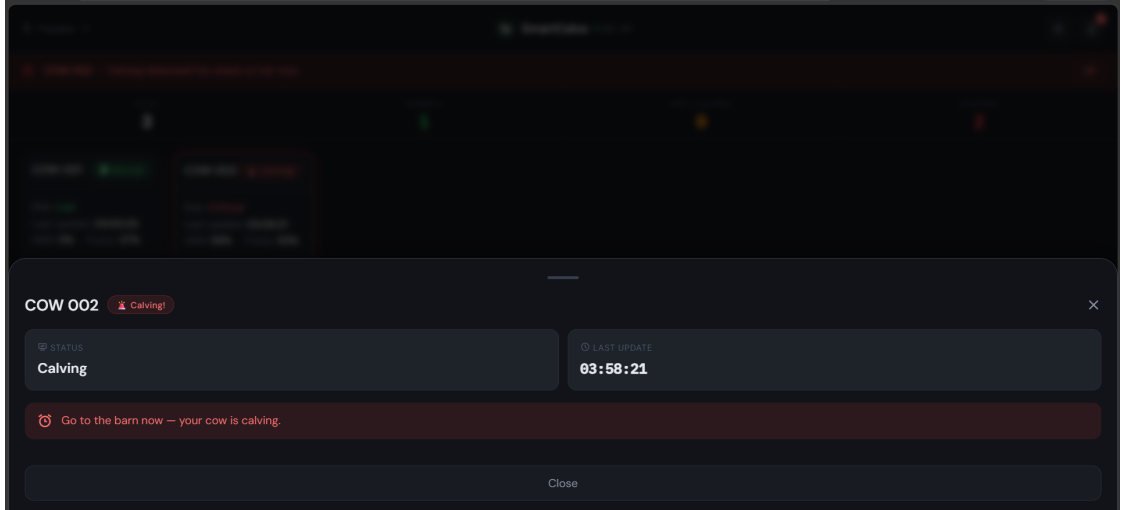


Figure 3.6: CalvingSafe dashboard: COW 002 calving alert modal with status badge and action banner

3.3.5 End-to-End Latency

The end-to-end latency of the SmartCalve system, defined as the time elapsed from sensor data transmission to dashboard alert visibility, was observed to be consistently below one second during test sessions conducted over a local Wi-Fi network. Table 3.3 provides estimated latency breakdowns for each pipeline stage.

Pipeline Stage	From	To	Estimated Latency	Observation
UDP Transmission	ESP8266	Python backend	< 50 ms	Connectionless, local Wi-Fi
ANN Inference	Feature vector	ANN score	< 30 ms	Scikit-learn MLPClassifier [18]
Fuzzy Inference	Sensor values	Fuzzy score	< 70 ms	scikit-fuzzy Mamdani engine
Firebase Push	Python backend	Firebase DB	100-400 ms	Depends on network quality
Dashboard Update	Firebase DB	Browser display	< 200 ms	Real-time listener
Total Pipeline	Sensor reading	Alert visible	< 1 second	Observed during test sessions

Table 3.4: End-to-end latency breakdown across SmartCalve pipeline stages

3.4 AI Model Performance Analysis

This section analyzes the performance of the individual AI components and the hybrid decision system based on the training evaluation results and the observed behavior during test sessions.

3.4.1 Individual Model Analysis

The ANN model, trained with SMOTE balancing [4] and a detection threshold of 0.2, demonstrates strong sensitivity to pre-calving motion patterns. The low detection threshold significantly increases recall for the calving class at the cost of some reduction in precision, which is the appropriate trade-off for this safety-critical

application where missing a calving event is far more costly than generating a false alarm.

The Fuzzy Logic system operates on raw sensor frame values rather than statistical window features, making it sensitive to instantaneous motion intensity. The temporal counter requiring 10 consecutive positive fuzzy predictions before triggering a timed positive prediction acts as a debouncing mechanism, filtering out transient motion spikes that do not represent genuine pre-calving behavior.

3.4.2 Hybrid System Advantage

The primary advantage of the hybrid ANN+Fuzzy architecture over either individual model is its ability to reduce false positives through requiring bilateral agreement before issuing a HIGH confidence alert. By requiring both the ANN to exceed its threshold AND the fuzzy counter to reach 10 consecutive positives, the hybrid system filters out both transient spikes and borderline ANN scores simultaneously. The voting table produces four possible outcomes providing graduated alert levels that allow farmers and veterinarians to calibrate their response according to confidence.

Metric	ANN Alone	Fuzzy Alone	Hybrid System	Advantage
Threshold applied	ANN threshold = 0.2	Fuzzy confidence threshold = 0.5	Both thresholds must be satisfied before a HIGH alert	Double filtering reduces accidental triggering
False positive risk	Medium	Medium	Low	The 10-consecutive-positive rule filters transient movement spikes
False negative risk	Low, because the ANN threshold is permissive	Medium, because instantaneous fuzzy values may miss gradual patterns	Low	The ANN covers fuzzy-rule gaps while fuzzy logic confirms motion intensity
Temporal filtering	None	10-consecutive-positive rule	ANN decision plus fuzzy temporal confirmation	Maximum robustness against noisy or isolated frames
Alert output	Single probability score	Single fuzzy confidence score	NORMAL, MEDIUM, or HIGH confidence alert	More clinically useful and easier for farmers/veterinarians to interpret

Table 3.5: Comparative analysis: ANN alone vs. Fuzzy alone vs. Hybrid system

3.5 Discussion

3.5.1 What Worked Well

The SmartCalve system demonstrated correct end-to-end behavior across all pipeline stages in the conducted test sessions. The hardware layer correctly acquired and transmitted sensor data through the TCA9548A multiplexer and ESP8266 via UDP. The Python backend correctly received, buffered, preprocessed, and fed sensor data to both inference engines. The hybrid voting system produced the expected confidence output, Firebase synchronization operated reliably, and the web dashboard correctly reflected all state changes in real time without page refresh. The Blynk push notification was successfully dispatched, providing mobile alert delivery to the farmer’s smartphone.

The Vercel-hosted dashboard proved to be a practical deployment solution. The zero-configuration hosting, automatic HTTPS, and global CDN ensured that the dashboard was accessible from any device with a browser and internet connection, with no local server setup required on the farmer’s side.

3.5.2 Known Limitations

The primary limitation of the current SmartCalve implementation is the absence of real-farm validation. While the CSV simulation methodology exercises the complete technical pipeline and uses real bovine IMU data, it cannot substitute for validation on live animals in field conditions [23, 5]. Real deployment introduces variables that cannot be fully reproduced in simulation: sensor attachment stability on a moving animal, varying Wi-Fi signal strength in farm buildings, and the natural variability in pre-calving behavioral signatures across individual cows.

The Firebase push and web dashboard depend on an active internet connection. In the event of internet unavailability, push operations fail silently and the dashboard will not reflect the latest cow state until connectivity is restored. The sensor acquisition and inference pipeline itself operates entirely over the local Wi-Fi network and does not require internet connectivity, making the system partially resilient to network outages.

3.5.3 Comparison with Existing Solutions

Criterion	Yoshida et al. [23]	Engelken et al. [5]	Commercial (SCR)	SmartCalve
Hardware cost	High	Medium	Very high	Low
AI method	SVM / RF	Two-stage ML	Threshold rules	ANN + Fuzzy hybrid
Connectivity	Wired logger	LoRa / Cloud	BT / Cloud	Wi-Fi / Firebase
Real-time alerts	No	Partial	Yes (subscription)	Yes (free tier)
Open source	No	No	No	Yes
Rural Algeria fit	Low	Medium	Low	High
Validation	Real farm	Real farm	Commercial	Lab + CSV simulation
<i>Note: Yoshida et al. and Engelken et al. are academic research prototypes and not commercially available products. SCR represents a commercially deployed solution included for industrial benchmarking.</i>				

Table 3.6: Positioning of SmartCalve against existing calving detection solutions

3.6 Future Work

The SmartCalve system in its current form establishes a functional proof of concept for low-cost IoT-based calving detection. The following directions are proposed for future development.

3.6.1 Short-Term Improvements

Real Farm Validation. The highest-priority next step is the deployment and validation of SmartCalve on a real dairy farm in Algeria [15, 16]. This would require collaboration with a local farming cooperative or veterinary institution to access instrumented cows over a monitoring period sufficient to capture multiple calving events.

Mobile Application. Replacing the Blynk notification integration with a dedicated mobile application for Android and iOS would provide a more reliable and feature-rich alert channel using Firebase Cloud Messaging for push notifications.

3.6.2 Medium-Term Research Directions

GPS Positioning and Google Maps Integration. A significant practical improvement would be equipping each sensor node with a GPS module to capture the real-time geographic position of the monitored cow. When a calving alert is triggered, the dashboard would display the cow's exact location on a Google Maps interface embedded in the farmer and veterinarian views. This feature is particularly valuable for extensive farming operations where cows graze over large areas and cannot always be visually located by the farmer when an alert arrives.

Temperature Sensor Integration. Adding a body temperature channel, using either a vaginal temperature probe or an ear-mounted infrared sensor, would provide a multimodal input to the ANN [23]. Pre-calving body temperature drops are a well-documented physiological indicator that would complement the tail-motion accelerometer data and improve both sensitivity and specificity of the detection algorithm.

Hardware Miniaturization. The current prototype uses off-the-shelf development boards (ESP8266 NodeMCU, breakout MPU6050 modules) mounted on a standard breadboard. A future hardware revision would integrate all components onto a custom PCB, reducing the form factor to a wearable unit small enough for permanent, non-invasive tail attachment. The miniaturized device would include a dedicated battery management circuit and a protective enclosure suitable for outdoor farm environments.

3.6.3 Long-Term Vision

TinyML On-Device Inference. A significant architectural improvement would be implementing the ANN inference directly on the ESP8266 using TinyML frameworks [22], eliminating the dependency on a backend computer and enabling fully autonomous field operation.

LoRa Connectivity for Remote Farms. Replacing the Wi-Fi transmission layer with LoRa radio communication would enable SmartCalve deployment in remote Algerian rural areas without internet access [5].

Solar-Powered Autonomous Nodes. Equipping each sensor node with a small solar panel and rechargeable lithium battery would enable continuous operation without mains electricity, making SmartCalve viable for semi-arid pastoral farming environments characteristic of much of Algeria [15, 16].

Algerian Farm Pilot Study. A structured pilot study involving multiple farms across different Algerian agricultural zones, conducted in partnership with MADR or INRAA [8], would generate the real-world performance data needed to publish SmartCalve as a validated contribution to the precision livestock farming literature.

Chapter Summary

This chapter presented the testing methodology, development environment, deployment configuration, and validation results of the SmartCalve system. The system operates in two modes: a Real Hardware Mode in which the ESP8266 firmware acquires live sensor data and transmits it via UDP, and a CSV Simulation Mode in which a pre-recorded labeled dataset is processed directly by the Python backend through the same inference and cloud pipeline. The simulation mode was adopted for the documented test session due to the absence of a real farm environment during the development period, and is consistent with established practices in embedded systems and IoT prototyping [23, 5].

The web dashboard is deployed on Vercel at smartcalve.vercel.app and provides public access to the CalvingSafe farmer and veterinarian interfaces without requiring any local server infrastructure. System validation confirmed correct behavior across all components: the sliding window assembly, ANN and Fuzzy Logic inference, hybrid voting, Firebase synchronization, dashboard real-time updates, and Blynk push notification delivery.

The key limitations identified are the absence of real-farm validation and the dependency on internet connectivity for alert delivery, both of which remain critical next steps for transitioning SmartCalve from a laboratory prototype to a field-deployable product. The proposed future work roadmap extends the current system with GPS positioning and Google Maps integration, temperature sensor fusion, hardware miniaturization, and a long-term vision for a multi-farm Algerian deployment platform.

GENERAL CONCLUSION

This memoir has presented the complete design, implementation, and validation of SmartCalve / CalvingSafe, a low-cost IoT-based intelligent system for automated calving detection in dairy cattle. The work covers the full spectrum of an embedded AI system: from physical sensor assembly and wireless data transmission, through machine learning model training and fuzzy logic inference, to cloud synchronization and real-time web-based supervision.

The hardware layer successfully integrates three MPU6050 six-axis inertial measurement units mounted along the cow's tail through a TCA9548A I2C multiplexer to an ESP8266 NodeMCU, providing a distributed, non-invasive motion capture solution at a component cost of under ten US dollars. The firmware implements a continuous acquisition loop with Cartesian-to-spherical coordinate conversion and UDP packet transmission, forming a reliable and low-latency data pipeline.

The AI engine combines an Artificial Neural Network (128×64 neurons, tanh activation, SMOTE-balanced, threshold = 0.2) with a seven-input Mamdani Fuzzy Logic System through a two-model voting mechanism and a 10 consecutive frame temporal filter. This hybrid architecture produces three clinically meaningful confidence levels HIGH, MEDIUM, and LOW balancing sensitivity against false alarm rate in a way that neither model achieves independently.

The CalvingSafe web dashboard, deployed on Vercel at smartcalve.vercel.app, provides role-differentiated real-time views for farmers and veterinarians, synchronized via Firebase Realtime Database, with a three-channel alert system combining in-dashboard banners, event logging, and Blynk mobile push notifications. System validation through CSV simulation confirmed correct end-to-end behavior across all pipeline stages, with an estimated total latency below one second from sensor reading to dashboard alert.

The primary limitation of the current implementation is the absence of real-farm validation on live animals. While the simulation methodology rigorously exercises the complete technical pipeline on real bovine IMU data, field deployment introduces additional variables sensor attachment stability, variable Wi-Fi coverage in farm buildings, and individual behavioral variability across cows that cannot be fully reproduced in a controlled simulation environment. Addressing this limitation through a structured farm pilot study remains the most critical next step for transitioning CalvingSafe from a laboratory prototype to a field-deployable product.

In the short term, the system would benefit from integration of a dedicated mobile application with Firebase Cloud Messaging replacing the Blynk notification channel, and from GPS positioning enabling cow location display on an embedded Google Maps interface in the dashboard. In the medium term, the addition of a

body temperature sensor channel would provide a complementary physiological calving indicator, and hardware miniaturization onto a custom PCB would produce a wearable device suitable for permanent tail attachment. In the long term, TinyML on-device inference would eliminate the dependency on a backend computer, LoRa connectivity would enable deployment in remote areas without Wi-Fi infrastructure, and a multi-farm pilot study in partnership with MADR would provide the field performance data necessary for a peer-reviewed publication.

The SmartCalve project demonstrates that a functional, intelligent, and cloud-connected livestock monitoring system can be designed, implemented, and validated within the scope of a Master's final-year project, using exclusively open-source tools and low-cost hardware components. It establishes a solid technical foundation for future work toward a deployable Algerian precision livestock farming solution. Beyond its technical contributions, this work also represents a concrete application of automation engineering principles—sensor integration, real-time processing, artificial intelligence, and cloud supervision—to a real-world agricultural problem with significant economic and humanitarian impact.

BIBLIOGRAPHY

- [1] Arduino LLC. *Arduino IDE Documentation*, 2024.
- [2] L. Benos et al. Machine Learning in Agriculture: A Comprehensive Updated Review. *Sensors*, 21(11):3758, May 2021.
- [3] M. Cappai et al. Internet of Things (IoT): Sensors Application in Dairy Cattle Farming. *Animals*, 14(21):3071, October 2024.
- [4] N. V. Chawla et al. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.
- [5] A. Engelken et al. A Two-Stage Machine Learning Approach for Calving Detection in Rangeland Cattle. *Agriculture*, 15(13):1434, July 2025.
- [6] Espressif Systems. *ESP8266EX Datasheet*, 2023.
- [7] Espressif Systems. *ESP8266 Wi-Fi SoC – Product Page*, 2024.
- [8] Food and Agriculture Organization of the United Nations (FAO). *Country Profile: Algeria*. FAOSTAT, 2023.
- [9] Google LLC. *Firebase Realtime Database Documentation*, 2024.
- [10] G. E. Hinton. Connectionist Learning Procedures. *Artificial Intelligence*, 40(1):185–234, 1989.
- [11] InvenSense (TDK). *MPU-6000 and MPU-6050 Product Specification*, rev. 3.4 edition, August 2013.
- [12] M. Khelifi et al. Algerian Perspectives for UAV-based Remote Sensing in Agriculture. In *Proceedings of ACM PETRA 2021*, Corfu, Greece, 2021.
- [13] K. G. Liakos et al. Machine Learning in Agriculture: A Review. *Sensors*, 18(8):2674, August 2018.
- [14] E. H. Mamdani and S. Assilian. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.
- [15] Ministere de l’Agriculture et du Developpement Rural (MADR). *Statistiques Agricoles – Serie B: Statistiques des Elevages*, 2019.

- [16] Ministère de l'Agriculture et du Développement Rural (MADR). *Feuille de Route de l'Agriculture Algérienne 2020–2024*, September 2020.
- [17] Mozilla Developer Network (MDN). *JavaScript Reference Documentation*, 2024.
- [18] F. Pedregosa et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [19] Raspberry Pi Foundation. *Raspberry Pi Documentation*, 2024.
- [20] L. Riaboff et al. Evaluating ROCKET and Catch22 Features for Calf Behaviour Classification from Accelerometer Data Using Machine Learning Models, April 2024.
- [21] Texas Instruments. *TCA9548A Low-Voltage 8-Channel I2C Switch with Reset*, rev. h edition, 2015.
- [22] M. Venzke et al. Artificial Neural Networks for Sensor Data Classification on Small Embedded Systems, December 2020.
- [23] K. Yoshida et al. Prediction of 24-h and 6-h Periods before Calving Using a Multimodal Tail-Attached Device through Supervised Machine Learning. *Animals*, 12(16):2095, August 2022.
- [24] L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8(3):338–353, 1965.