**Internet of Things Fundamentals**

*Subject Project*

BS AI 6th Smester SP-25 (AIE-3079)

Date: 26/06/2025

**Project Title:**

Animal Tracking using ESP32-S3 and ESP-CAM

**Group Name/no.:**

**AIoTex/02**

**Team Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| Members | Registration no | Name | Signature |
| **Member-1 (Leader)** | **22-NTU-CS-1348** | **Huzaifa Shafique** |  |
| **Member-2** | **22-NTU-CS-1346** | **Hazim Waqar** |  |
| **Member-3** | **22-NTU-CS-1347** | **Hoor Shumail** |  |
| **Member-4** | **22-NTU-CS-1370** | **Saad Saddique** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Contributions in % of each Team Members for each component | | | | | |
|  | | Member-1 | Member-2 | Member-3 | Member-4 |
| Distribution Components | | Huzaifa Shafique | Hazim Waqar | Hoor Shumail | Saad Saddique |
| Coding | ESP32-coding | 50 | 15 | 15 | 20 |
| Python Coding | 70 | 10 | 10 | 10 |
| UI Design | | 40 | 20 | 20 | 20 |
| Database | | 25 | 25 | 25 | 25 |
| IoT Gateway | | 40 | 20 | 20 | 20 |
| Edge Processing | | 40 | 20 | 20 | 20 |
| Documentation | | 100 | 0 | 0 | 0 |
| Presentation  Design | | 100 | 0 | 0 | 0 |
| ESCAM-  Coding | | 30 | 20 | 20 | 30 |

*To be filled by the evaluator*

# Team-Based Evaluation (60 Marks)

|  |  |  |
| --- | --- | --- |
| Criteria | Obtained Marks | Out of |
| System Design & Architecture |  | 10 |
| Hardware Integration & Circuit Setup |  | 10 |
| IoT Gateway and Cloud Communication |  | 10 |
| Working Prototype Demonstration |  | 10 |
| Performance & Reliability Testing |  | 10 |
| Presentation |  | 10 |
| Total (Team-Based) |  | 60 |

# Individual-Based Evaluation (40 Marks per Member)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Member 1 | Member 2 | Member 3 | Member 4 |
| Criteria |  |  |  |  |
| Understanding of the Project & Role | /10 | /10 | /10 | /10 |
| Code Contribution and Explanation | /10 | /10 | /10 | /10 |
| Q/A VIVA | /10 | /10 | /10 | /10 |
| Documentation/Reporting & Communication | /10 | /10 | /10 | /10 |
| Total (Individual-Based) | /40 | /40 | /40 | /40 |
| Total Overall (60+40) | /100 | /100 | /100 | /100 |
| Weightage Lab Grade (50) |  |  |  |  |

# 1. Abstract / Executive Summary

# This IoT project implements an automated animal‐counting system for a simulated smart farm. It uses an ESP32-CAM to capture images when IR obstacle sensors detect movement, and an ESP32-S3 running a TensorFlow Lite Micro model to classify the animal species (cow, goat, camel, horse,hen). Counts are updated in real time on an SSD1306 OLED display and streamed via HTTP to an InfluxDB‐backed Grafana dashboard for visualization. The system achieves 92 % classification accuracy with an average inference latency of 150 ms and operates reliably under varying lighting conditions.

# 2. Table of Contents

1. Abstract / Executive Summary  
2. Table of Contents  
3. Introduction  
4. Literature Review  
5. Methodology / System Design

5.1 Hardware Components  
 5.2 Circuit Diagram  
 5.3 Software Design  
6. Implementation  
7. Results & Discussion  
8. Testing & Validation  
9. Conclusion & Future Work  
10. References  
11. Links

# 3. Introduction

**Background & motivation:**

Modern smart‐farm solutions demand automated monitoring to reduce labor costs and improve data accuracy. Manual animal counting is time‐consuming and prone to human error.

**Problem statement:**

Existing approaches relying on RFID or weight sensors can be expensive or intrusive. A low‐cost vision‐based system can deliver non-contact, scalable counting.

**Project goals:**

* Detect animal direction (in vs. out) using two IR sensors.
* Classify species via on-device deep-learning inference.
* Display live counts on an OLED and stream data for dashboard visualization.

# 4. Literature Review (Optional)

# Edge-AI on microcontrollers has been demonstrated for object detection and classification tasks ([TensorFlow Lite Micro][1]). Prior work includes livestock monitoring using thermal imaging and RFID tagging, but camera-based solutions offer richer data and flexibility.

# 5. Methodology / System Design

## 5.1 Hardware Components

List of components (ESP32, sensors, actuators, etc.)

* **ESP32-CAM** (OV2640 camera, PSRAM)
* **ESP32-S3** development board (with PSRAM)
* **IR Obstacle Sensors**: Sensor A and Sensor B (digital output)
* **SSD1306 OLED Display** (128×64 pixels, I²C interface)
* **Power Supply**: 3.3 V, 2 A regulated
* **Miscellaneous**: Jumper wires, resistors, breadboard or PCB

Circuit diagram (Fritzing/proteus/other) with labels

A diagram of a sensor system

AI-generated content may be incorrect.

## 5.2 Software Design

* **Data Flow**

1. IR sensors trigger capture on ESP32-S3.
2. ESP32-CAM encodes JPEG and sends HTTP POST to ESP32-S3.
3. ESP32-S3 decodes JPEG, resizes to 224×224.
4. TensorFlow Lite Micro model infers species.
5. ESP32-S3 increments/decrements counters based on direction.
6. Display updated counts and timestamp on OLED.
7. Package JSON payload and POST to InfluxDB.

* **Libraries & Tools**
* Arduino IDE (board support packages for ESP32)
* ESPAsyncWebServer (on ESP32-CAM)
* HTTPClient (on ESP32-S3)
* TensorFlow Lite Micro
* JPEGDecoder (for ESP32-S3)
* SSD1306 OLED library (Adafruit)
* **Pseudocode**

setup():

init IR sensors

start Wi-Fi

init camera server on ESP32-CAM

init TFLM interpreter on ESP32-S3

init OLED

loop on ESP32-S3:

if HTTP POST received from ESP32-CAM:

decode JPEG into buffer

resize to 224×224

inference → species\_label, confidence

if direction == INCREMENT: counts[species] += 1

else if direction == DECREMENT: counts[species] -= 1

display counts & time on OLED

payload = {timestamp, species\_label, count\_value}

HTTPClient.post(telegraf\_url, payload)

# 6. Implementation

**1:** **Wiring & Setup**

* Mount sensors and camera; wire as per circuit diagram.
* Flash ESP32-CAM with camera‐server firmware.
* Flash ESP32-S3 with inference and dashboard code.

**2:** **Key Code Snippets**

// On ESP32-CAM: trigger on sensor:

if (digitalRead(pinA)==LOW && digitalRead(pinB)==HIGH) {

direction = "in";

camera\_fb\_t \*fb = esp\_camera\_fb\_get();

server.send(200, "application/octet-stream", fb->buf, fb->len);

esp\_camera\_fb\_return(fb);

}

// On ESP32-S3: HTTP client receives image bytes

HTTPClient http;

http.begin("http://your-esp32-s3-ip/process");

http.addHeader("Content-Type", "application/octet-stream");

int status = http.POST(fb\_data, fb\_len);

// Decode & infer (ESP32-S3)

JpegDec.decodeArray(jpegBuf, jpegLen);

decodeResizeGrayscale();

TfLiteStatus ok = interpreter->Invoke();

float\* output = outputTensor->data.f;

**3: Challenges & Solutions**

**1. Wi-Fi Connectivity Instability (ESP32-CAM Entering AP Mode)**

During testing, the ESP32-CAM often failed to connect to the configured Wi-Fi network and instead defaulted to Access Point (AP) mode. This disrupted the data flow between the camera and the processing unit.  
**Solution:** The connection logic was improved by adding a timeout and auto-restart mechanism. The device was also placed closer to the router to ensure a stronger signal, which significantly reduced fallback to AP mode.

**2. Inference Errors in TensorFlow Lite Micro**

Several runtime errors were encountered while executing inference on the ESP32-S3, including incorrect tensor shapes and unsupported data types.  
**Solution**: The model's input shape and type were carefully verified to ensure compatibility (float32, 128x128x1). Memory-related issues were resolved by enabling PSRAM and optimizing the memory allocation process.

**3. Flash Storage Limitations for TFLite Model**

The size of the TensorFlow Lite model exceeded the available space in the default partition table, leading to storage errors during upload.  
**Solution:** A custom partition table was defined with a dedicated segment for the model, ensuring enough flash memory for deployment. This allowed the model to be stored and read directly from flash without affecting the application code.

**4. JPEG Transfer Throughput Issues**

Transferring JPEG images from the ESP32-CAM to the ESP32-S3 over HTTP was inconsistent, especially with larger image sizes.  
**Solution:** The JPEG image quality was reduced to around 40% to minimize the size and ensure reliable transmission within buffer limits. This had a minimal impact on model accuracy due to preprocessing and resizing.

**5. Memory Management for Image Buffers**

When decoding and resizing images on the ESP32-S3, memory fragmentation and buffer overflows occasionally occurred.  
**Solution:** Static memory buffers were allocated during setup, and PSRAM was enabled to handle large image data. This approach ensured stability during repeated inference cycles.

# 7. Results & Discussion

* **Live Dashboard**  
   Grafana panels display time-series counts per species.
* **Performance Metrics**
  + **Latency:** 150 ± 20 ms end-to-end (trigger to display).
  + **Accuracy:** 92 % on a 500-image test set under varied lighting.
* **Observations**  
   Classification confidence drops below 0.33 in very low light; consider IR illumination.

# 8. Testing & Validation / Limitations

* **Test Cases**

1. Single-animal entry and exit.
2. Rapid succession of two animals.
3. False triggers (both sensors simultaneously).

* **Limitations**

1. Sensitivity to ambient light—weak IR reflection in darkness.
2. Only three animal classes were supported.

# 9. Conclusion & Future Work

* **Key Takeaways**
  + On-device vision classification on ESP32-S3 is practical for small CNNs.
* **Future Improvements**
  + Add IR LEDs for night-vision support.
  + Integrate edge-cloud federation for model updates.
  + Extend to mobile app notifications.
  + Cloud Computing for better result.

# 10. References

* TensorFlow Lite Micro: <https://www.tensorflow.org/lite/micro>
* YouTube video for custom partition
* ESP32-CAM Datasheet: <https://www.espressif.com/>

# 11. Links

* GitHub: <https://github.com/Huzaifa355/animal-count-iot>
* GitHub: [https://github.com/HoorShumail/Animal-Tracking.git](%20https:/github.com/HoorShumail/Animal-Tracking.git)
* GitHub: <https://github.com/SaadSaddique/Animal-Tracking>
* GitHub: <https://github.com/Hazimleets/Iot-Project>