**Problem Statement & Objectives**  
Small farms and enclosures rely on manual head-counts to monitor livestock movement, which is labor-intensive, error-prone, and yields no historical record. This project aims to build an automated system that:

* Detects when an animal crosses a gate (in or out)
* Captures an image via **ESP32-CAM / USB Webcam**
* Classifies the animal species at the edge (cow, buffalo, camel, goat)
* Tags each event with a Pakistan Standard Time (UTC+5) timestamp via NTP
* Uploads both image and metadata to Firebase for live dashboards and archival logs

**Scope of the Project**

* **Gate Deployment**: Single entry/exit point per pen; scalable to multiple gates
* **Capture Modes**: Option to use either the onboard ESP32-CAM module **or** a higher-quality USB Webcam on ESP32-S3
* **Edge Inference**: One still frame per crossing event; no continuous video upload
* **Cloud Backend**: Firebase Storage for images and Realtime Database for event records
* **Time Synchronization**: Device clock synced via NTP to Pakistan Standard Time
* **Visualization**: Real-time and historical charts through Firebase console or custom UI

**Required Components / Sensors / Modules**

* **Microcontrollers**
  + ESP32-CAM module (OV2640) **or** ESP32-S3 Dev Board with USB-OTG (for USB Webcam)
* **Camera**
  + **Option A:** ESP32-CAM (OV2640)
  + **Option B:** UVC-compliant USB Webcam (e.g., Logitech C980)
* **Motion & Direction Sensing**
  + Two IR Break-Beam sensors per gate (emitter + receiver)
* **Power**
  + Regulated 5 V, 2 A supply (to power board + camera)
* **Prototyping**
  + Breadboard, jumper wires
* **Display**
  + OLED I²C display (SSD1306) for local counts

**Software, Cloud & Database**

* **Firmware Platform**
  + ESP-IDF (for USB-Host on ESP32-S3) or Arduino core (for ESP32-CAM)
* **Edge AI**
  + TensorFlow-Lite Micro model for four-class classification
* **Time Sync**
  + <time.h> NTP client (UTC+5 offset)
* **HTTP Client**
  + esp\_http\_client or Arduino HTTPClient for HTTPS calls
* **Cloud Services**
  + **Firebase Storage**: JPEG image uploads
  + **Firebase Realtime Database**: JSON event logs (species, direction, timestamp, image URL)

**Subjects Used Other Than IoT**

* **Embedded Machine Learning** (TinyML)
* **USB-Host & UVC Protocol** (for external webcams)
* **Network Time Protocol** (synchronizing to Pakistan Standard Time)
* **Web API Integration** (secure HTTPS POST to Firebase)

**Detailed Description of the Approach**

1. **Gate Crossing Detection**
   * IR sensors A & B yield a direction: A→B = OUT, B→A = IN.
2. **Capture Trigger**
   * On an IN/OUT event, snap a single frame from either ESP32-CAM or USB Webcam.
3. **Edge Classification**
   * Run the TinyML model on the captured JPEG buffer to identify species.
4. **Timestamping**
   * Use NTP to sync the ESP32’s clock (UTC+5), then format via strftime() (“YYYY-MM-DD HH:MM:SS”).
5. **Cloud Upload**
   * **Storage**: Upload the JPEG to Firebase Storage → obtain public URL.
   * **Database**: POST { species, direction, timestamp, imageUrl } to Realtime DB.
6. **Visualization**
   * Firebase console charts for counts and an optional custom dashboard for image review.

**System Architecture / Block Diagram**

plaintext

CopyEdit

[ IR Beam A ]─────────┐

│ ┌──────────────────┐ USB host ┌──────────┐

[ IR Beam B ]──Trigger──▶ [ESP32-S3] ──stream──────▶│ USB Webcam │

│ └──────────────────┘ │

│

└────Trigger & capture──▶ [ESP32-CAM module]─────▶ esp\_camera.h

│

▼

TinyML Classification

│

▼

Timestamp (NTP, UTC+5, time.h)

│

┌─────────────HTTP POST───────────────┐

▼ ▼

Firebase Storage (JPEG) Firebase Realtime Database (JSON)