MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

Project Progress Report for CNG 476

Smart Agriculture Animal Tracking and Protection System

Project Members:

1. Ömercan Kahveci 2453298,
2. Dorukhan Doğan

Project Proposal:

Project Description:

The Smart Agriculture Animal Tracking and Protection System is an advanced IoT solution prepared to transform monitoring and security protocols within the cattle farming industry. By integrating advanced GPS tracking, camera surveillance, and sensor technology into cattle straps, the system offers comprehensive real-time monitoring and real-time threat detection.

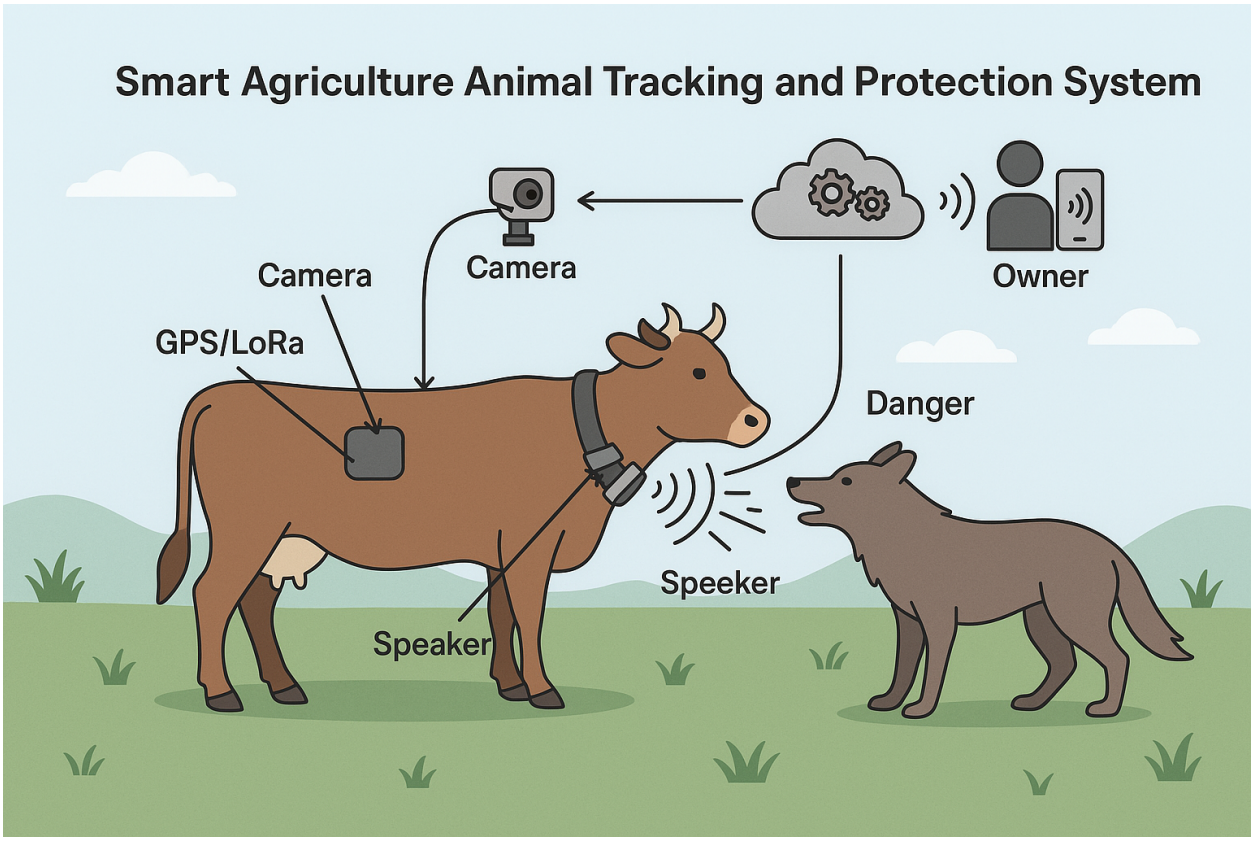

Figure 1 Smart Agriculture Animal Tracking and Protection System General Image(Canva)

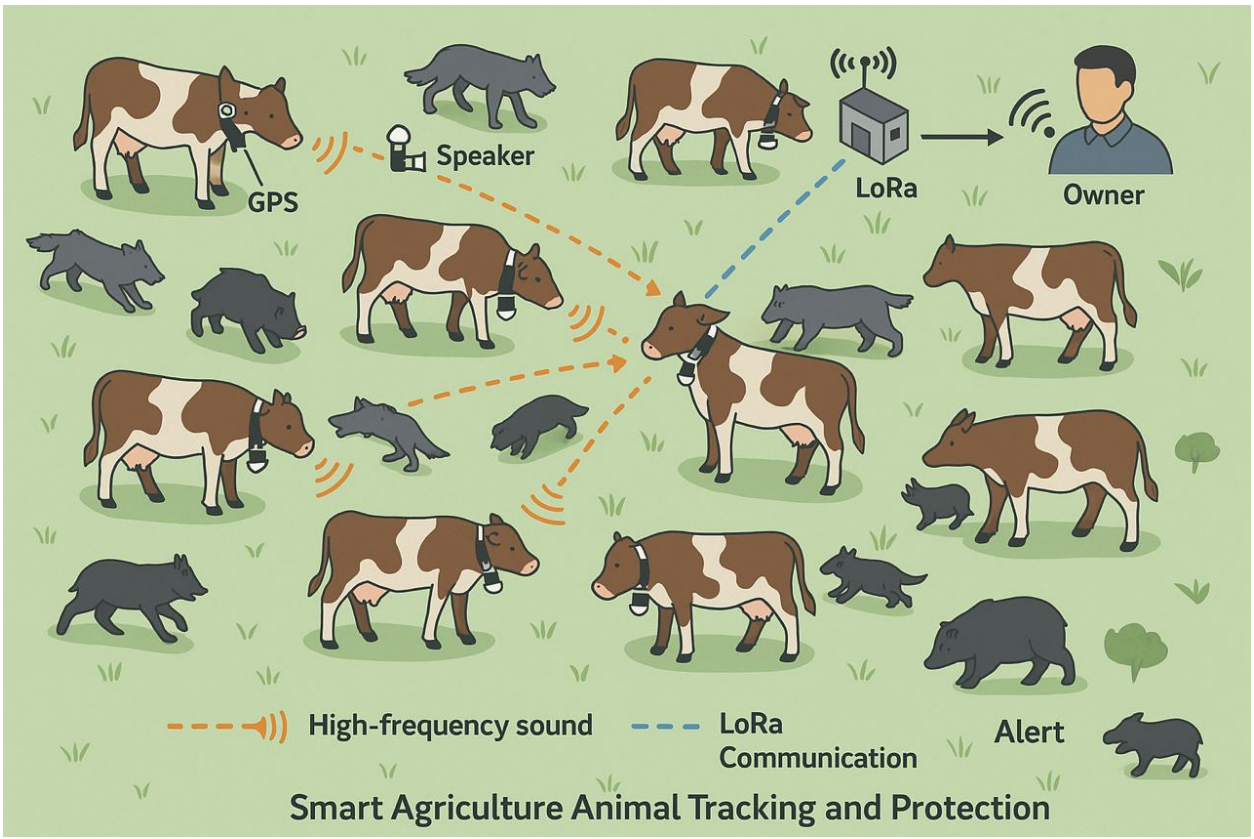

Figure 2 Smart Agriculture Animal Tracking and Protection System General Image with Herd of Cattles and Predators.

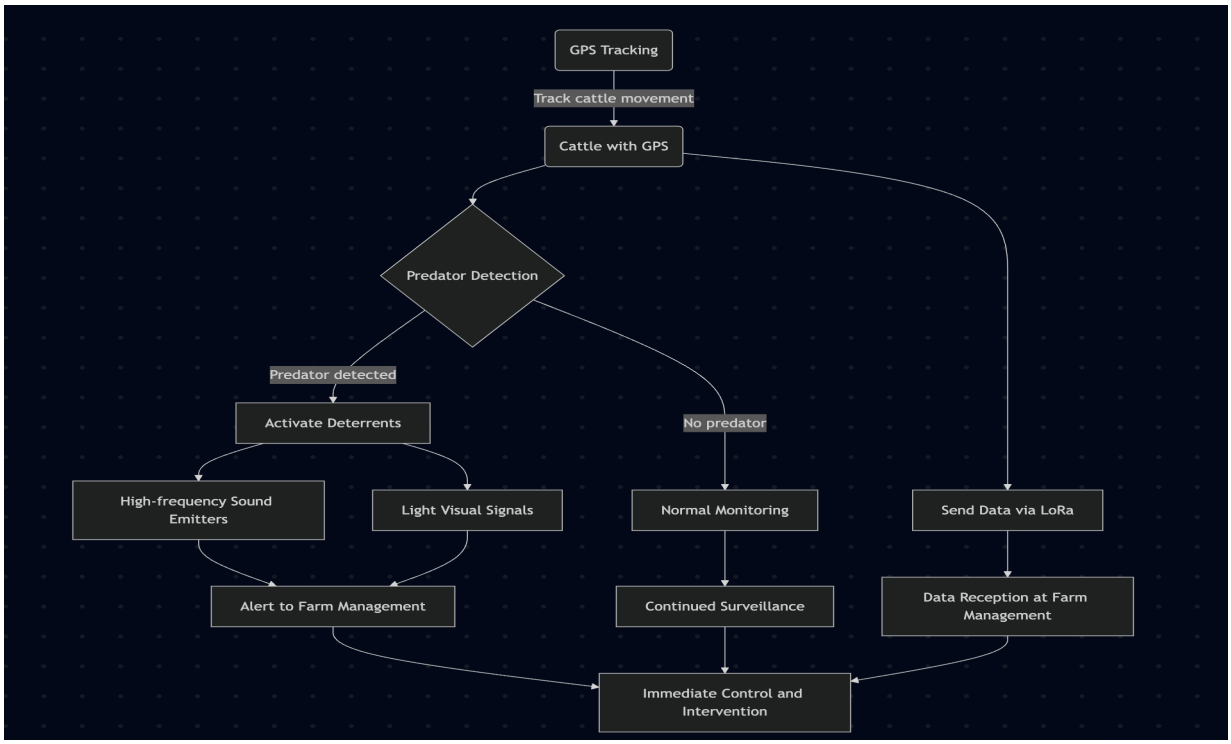

Figure 3 Smart Agriculture Animal Tracking and Protection System General Flowchart

Background and Challenges:

Cattle farmers can lose significant revenues because of predator attacks, especially wolves, which are prevalent in most pastoral regions. Researchers have found that such incidents have far-reaching economic consequences, such as livestock mortality, reduced productivity, and high management expenses (Smith et al., 2020). Solving such problems efficiently requires innovative approaches that provide real-time surveillance along with precautionary threat management.

Objectives:

1. **Enhanced Real-Time Monitoring:** It aims to apply GPS technology in real-time monitoring of the cattle as they move through the grazing fields. Its goal is to ensure that all the animals remain in safe areas, thus preventing them from wandering into hazardous areas.
2. **Advanced Threat Detection and Response**: The system incorporates sensors and cameras into the cattle straps, allowing them to recognize the presence of predators and other dangers. Upon such identification, the system automatically triggers the activation of particular deterrents, such as high-frequency sound emitters and, as necessary, light visual signals aimed at scaring away the predators. Concurrently, it sends real-time reports to farm management for immediate control and intervention.

3. **Proactive Defense Mechanisms**: The goal is to integrate harmless defense mechanisms in the cow straps that activate automatically as soon as dangers are detected, promoting the safety of the animals without causing harm to either the animals or the predators.

Technologies and Implementation:

- **LoRa Communication**: Use LoRa technology to enable efficient long-distance communication needed for real-time data transmission over large rural areas (The Things Network, 2025).

- **OMNeT++ and FLoRa Framework**: Model and test the sensor and network communications of the system with the OMNeT++ framework that has the FLoRa tool integrated, which is utilized to simulate LoRa-based networks (Aalto University, 2025).

Simulation and Modeling:

- **Random Number Generation and Monte Carlo Simulation**: Model random cattle movement and potential predator approaches to determine system responsiveness and accuracy of detection.

- **Probability and Stochastic Models**: Apply Poisson models in predator encounter frequency prediction and stochastic processes in sensor data analysis for predictive accuracy improvement and system reliability.

Assumptions:

- **Sensor Accuracy**: Assume high accuracy of sensors with minimal false positives or negatives in predator detection to focus on system response efficiency.
- **Environmental Factors**: Simulate under ideal environmental conditions; factors such as weather or terrain that might affect sensor performance are not considered.
- **Predator Behavior**: Assume typical predator behavior patterns. Eat, drink, move, sleep, and run.
- **System Components Reliability**: Assume all system components (GPS, sensors, cameras) function optimally without hardware failures.
- **Sensor Detection Range for the Simulation code**: Assume 80 meters is the optimal detection range.

Expected Impact and Benefits:

The use of this system should considerably decrease the attacks of predators on farm animals, because of that minimizing economic losses greatly. Furthermore, the system encourages animal welfare and improves farm management by providing efficient monitoring and protection tools.

Milestones Achieved:

Week	Date Range	Milestone	Team Member(s)	Details
Week 1	March 9 – March 15	Team formation & topic brainstorming	Ömercan Kahveci, Dorukhan Doğan	Chose partners and started researching IoT use cases for agriculture.
Week 2	March 16 – March 22	Finalized project idea & scope	Ömercan Kahveci	Outlined the cattle tracking + predator detection concept and documented high-level system goals.
Week 3	March 23 – March 29	Project proposal writing & submission	Dorukhan Doğan (draft), Ömercan Kahveci (finalized + visuals)	Wrote and submitted the proposal with detailed objectives, technologies, assumptions, and system flowchart.
Week 4	March 30 – April 5	OMNeT++ & FLoRa framework installation and environment setup	Ömercan Kahveci	Installed OMNeT++ 6.0 and FLoRa on Ubuntu VM. Verified with a base simulation (LoRa end-devices, gateways).
Week 5	April 6 – April 12	Designed system architecture and node types	Dorukhan Doğan	Reviewed LoRa node simulation examples; noted .ned, .ini, and .cc file structures.
Week 6	April 13 – April 19	Random movement logic for cattle nodes	Ömercan Kahveci	Developed initial movement pattern (Monte Carlo) for cattle within a fenced simulation area.
Week 7	April 20 – April 26	Implemented basic predator appearance model (Poisson process)	Dorukhan Doğan	Wrote Poisson-based threat generation logic and linked to predator node spawns in simulation.
Week 8	April 27 – May 2	Set up GitHub repository and project structure	Both	Created public GitHub repo. Pushed proposal PDF, README draft, and placeholder folders for simulation files.

Figure 4 Achieved Milestones Table

GitHub Repository Link:

Milestones Remained:

Week 9	May 3 – May 9	Implement message sending from cattle to base station via LoRa	Ömercan Kahveci	Use LoRa modules in OMNeT++ to simulate packet transmission when a threat is detected
Week 10	May 10 – May 16	Model predator approach and detection radius triggering	Dorukhan Doğan	Implement range-based detection logic (Euclidean distance check), triggering alerts
Week 11	May 17 – May 23	Add packet delay, message loss rate & basic reliability metrics	Both	Simulate delay and loss in communication; start recording success/failure rates
Week 12	May 24 – May 26	Final simulation runs & metric collection	Both	Run full simulation scenarios with multiple cows/predators to analyze: Avg. dangers encountered ,Detection success rate, Message loss rate, Delay statistics.
Week 13	May 26 – May 28	Project demonstration preparation & GitHub update	Both	Finalize code + README in GitHub, prepare demo for class presentation
Week 14	May 26 – May 28	Final Report writing	Both	Created public GitHub repo. Added proposal, diagrams, README, and first code files with early commits.

Figure 5 Remained Milestones Table

Code Snippets:

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

// Uniform random number between min and max
double uniform(double min, double max) {
    return min + ((double)rand() / RAND_MAX) * (max - min);
}

// Simulate cattle movement inside fenced field
void simulateCattleMovement(double *x, double *y) {
    double maxX = 500.0;// in meter
    double maxY = 300.0;

    *x = uniform( min: 0,  max: maxX);
    *y = uniform( min: 0,  max: maxY);
}

// Simulate predator appearance using Poisson distribution
int simulatePoisson(double lambda) {
    int k = 0;
    double p = 1.0;
    double L = exp(-lambda);

    do {
        k++;
        p *= uniform( min: 0.0,  max: 1.0);
    } while (p > L);

    return k - 1;
}
```

Figure 6 Code Snippet 1

```c
// Simulate predator random position and check proximity
int isPredatorDetected(double cowX, double cowY, double detectionRadius) {
    double predatorX = uniform( min: 0,  max: 500);
    double predatorY = uniform( min: 0,  max: 300);

    double distance = sqrt(pow(predatorX - cowX, 2) + pow(predatorY - cowY, 2));

    printf("Predator appeared at: (%.2f, %.2f)\n", predatorX, predatorY);
    printf("Distance to cattle: %.2f meters\n", distance);

    return distance <= detectionRadius;
}
```

Figure 7 Code Snippet 2

```c
int main() {
    srand(time(NULL));

    double cowX, cowY;
    simulateCattleMovement( x: &cowX,  y: &cowY);
    printf("Cattle moved to position: (%.2f, %.2f)\n", cowX, cowY);

    // Simulate predator encounter using Poisson model
    double lambda = 0.6; // average predator arrival rate
    int predators = simulatePoisson(lambda);
    printf("Number of predators encountered: %d\n", predators);

    // Detection radius ( camera/sensor range on the strap) it will be done using mobility positions in omnet++
    double detectionRadius = 80.0; // meters

    for (int i = 0; i < predators; i++) {
        printf("\n Checking predator #%d proximity...\n", i + 1);
        if (isPredatorDetected(cowX, cowY, detectionRadius)) {
            printf("!!! Predator detected !!! Sending alert...\n");
            printf(" Sound & light deterrent activated on cattle strap.\n");
        } else {
            printf("Relax Predator too far. No action needed.\n");
        }
    }

    return 0;
}
```

Figure 8 Code Snippet 3

Code Explanation:

This C program implements a simplified prototype of our IoT-based animal tracking and protection system. It simulates the movement of a cattle node and the random appearance of predators in a field. It checks if any predator is close enough to trigger an alert and deterrent measure, simulating how the actual smart strap system would respond in the field.

Even though written in pure C, the logic behind this simulation is designed for later integration with OMNeT++ via the FLoRa framework, where real wireless nodes and their communication patterns will be simulated.

Note that detectionRadius in the code is defined as 80 meters, which is an assumed threshold selected for simulation purposes. Since the OMNeT++-based simulation environment is still under development, this value was derived from existing research on real-world sensor capabilities. Specifically, LoRa-based proximity beacons typically offer detection or communication ranges between 50 to 300 meters, depending on transmission power and environmental conditions. Thus, 80 meters serves as a balanced, conservative estimate that reflects realistic sensing capability without overestimating the detection system's reach (The Things Network, 2025).

Also we used the idea:

- If the radius is **too small** (10–20m), predators would almost never be detected making the system seem ineffective.
- If the radius is **too large** (200m+), detect every predator, even if it's far away leading to too many false alerts.

So that 80 meters gives a realistic balance between useful reaction time, reducing false positives and keeping the simulation meaningful.

Technologies Used:

Component	Technique / Concept	Why Used
Random movement	**Uniform distribution**	To simulate unbiased cattle movement
Predator arrival	**Poisson distribution**	To simulate irregular, rare threat events
Proximity check	**Euclidean distance**	For basic spatial detection logic
Alerting logic	**Conditional statements**	Mimics reaction mechanisms of real system
Print output	**Descriptive logs**	Represents IoT system actions (e.g., alarms, LoRa message)

Figure 9 Table that Shows Technologies Used in Code Snippet

Example Code Outputs:

```
/Users/omercankahveci/Desktop/CNG476/projectCode/cmake-build-debug/projectCode
Cattle moved to position: (78.90, 13.20)
Number of predators encountered: 0

Process finished with exit code 0
```

Figure 10 Code Output that Shows No Predator Detected

```
/Users/omercankahveci/Desktop/CNG476/projectCode/cmake-build-debug/projectCode
Cattle moved to position: (81.28, 5.69)
Number of predators encountered: 1

 Checking predator #1 proximity...
Predator appeared at: (489.83, 88.13)
Distance to cattle: 416.79 meters
Relax Predator too far. No action needed.
```

Figure 11 Code Output Shows Encountered Predator with No Reaction due to Distance

```
/Users/omercankahveci/Desktop/CNG476/projectCode/cmake-build-debug/projectCode
Cattle moved to position: (85.13, 214.57)
Number of predators encountered: 2

 Checking predator #1 proximity...
Predator appeared at: (93.77, 264.49)
Distance to cattle: 50.67 meters
!!! Predator detected !!! Sending alert...
 Sound & light deterrent activated on cattle strap.

 Checking predator #2 proximity...
Predator appeared at: (385.83, 61.14)
Distance to cattle: 337.57 meters
Relax Predator too far. No action needed.
```

Figure 12 Code Output Shows Both Possibilities of the Predator Encountered Scenario

As can be seen from Figure 12, the cattle was randomly placed at the coordinates (85.13, 214.57) within the defined grazing field. In this specific simulation run, two predators were generated based on a Poisson distribution ($\lambda = 0.6$), reflecting the rare but possible appearance of threats in a short interval. The first predator appeared at (93.77, 264.49) and was located only 50.67 meters away from the cattle. Since this distance falls within the predefined detection radius of 80 meters, the system successfully identified the predator, simulated an alert being sent, and activated sound and light deterrents on the cattle's strap.

The second predator appeared at a much farther location, (385.83, 61.14), with a distance of 337.57 meters from the cattle. This distance exceeds the sensor's effective detection range, so the system made no response, correctly filtering out distant, non-threatening activity. This demonstrates that the detection logic operates correctly by only responding to close-proximity threats, aligning with the system's goal of reducing false alerts and preserving energy and bandwidth. This distance-based filtering mechanism is later planned to be modeled in OMNeT++ using mobility modules and custom detection logic, with alert messages sent through the LoRa communication channel when real threats are detected.

References:

- Aalto University. (2025). FLoRa: Framework for LoRa. Retrieved from https://flora.aalto.fi/
- Smith, J. K. et al. (2020). Economic impacts of predator attacks on cattle farms. Journal of Agricultural Economics, 65(2), 485-500.
- The Things Network. (2025). What is LoRaWAN?. Retrieved from https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/
- www.canva.com