A Pest Detection and Deterrence Sensor and Actuation System for Organic Farms

Rat Wranglers: Anna Finkelstein, Yash Gangavarapu, Ella Hamer, Flaviana Keller, Ishan Madan, Brandon Ramirez, Palo Silva, Ishan Vyas

ECE 449: Sensors and Sensor Interface Design

Table of Contents

Introduction

Farming plays a crucial role in the food chain for people all around the world. Particularly, organic farms, which represent 5.5% [1] of the U.S.'s $1 trillion [2] total food sales, are vulnerable to disruption from animals causing physical damage and waste contamination. While inorganic farms can deploy a number of pesticide products to protect crops, organic farms are left without a tangible defence.

We aim to address this gap by developing a smart, IoT-based system that first detects mid-size farm pests such as raccoons, groundhogs, squirrels, and rats. Using a tangential detection system, the device will sense an animal's presence, determine whether the intruder is a person or a pest, and, if a pest is confirmed, log the time, store an image, and communicate via WiFi to activate a targeted deterrence response. Because these animals are prey and respond strongly to motion, sound, and light stimuli, the deterrence module will deploy only when needed. This approach reduces crop losses, improves yields for organic farms, and helps address high food costs and demand-driven shortages for lower-income communities.

Design Criteria

Problem Statement: Organic farms face financial and physical damage due to large pests such as raccoons, groundhogs, squirrels, and rats. Developing an energy-efficient system to detect these large pests without harming them or the environment is critical to help organic farms combat such damages while complying with regulations set by the National Organic Standards Board.

The design criteria below cover an extensive set of goals we set out to achieve, a benchmark for each goal, and a justification for the selection of each goal.

Table 1: Design Criteria

Design Criteria	**Target Value**	**Justification**
Weather-Proof <u>*Constraint*</u>	Classifies as IP64 water resistant	If this device will be used in a field, especially in North Carolina, then it should be resistant to water and rain.

Photo storage *Constraint*	Photographs ≥95% of the triggers, including the date and time stamp	Provides additional information for the customers to determine which pests the farm is facing and when and where they are active.
Deterrence Strength over Protected Area *Constraint*	The deter system protects an area of 100ft (length) x 8ft (width) x 6ft (height).	This is the size of a standard row in the farm.
WiFi Connectivity *Constraint*	Communicates successfully with other system using WIFI for ≥95% of occurrences	The systems need to be able to communicate through WIFI so that the detection system can trigger the deterrence system.
Detection *Constraint*	≥95% of target pests are detected, day and night	The detection system needs to be able to accurately detect nearly all pests for the device to be useful
Compliant Material *Constraint*	0 Materials fail to meet NOP standard	NOP standardization ensures organic farms, the product's main customers, can use it.
Manipulatability *Constraint*	The system contains 1 on/off switch for the 3 methods: light, sound, motion.	Users should be able to turn on and off the deterrent system
Solar Powered *Constraint*	Charges battery to >= 80% in 24 hours with solar power	Charge battery with the sun
Animal-Friendliness *Constraint*	0 animals harmed while system implemented in field	The intention is to deter, not harm, the animals. Large pests are needed for the ecosystem.
Multi-level detection system *Objective*	System should have 3 levels of detection to save energy and detect and identify pests, without triggering response for humans	Have 3 layers of detection, one low power, one medium power, and one high power. This will reduce the total energy consumption of our system.
Energy Consumption *Objective*	Detect: Uses ≤1 Watt of power Deter: Uses < 7 Watt of power	Do not want the battery to die overnight.
Time to Detect	Detect in ≤3 seconds	Detect fast enough to trigger the system before an animal damages crops.

Objective		
Detection Radius *Objective*	Detect an animal within 20ft of the row crop	Squirrels move fast, so we want to detect fast enough to trigger the system before the squirrel reaches the crops
Bill of Materials Cost *Objective*	Total BoM cost <$150	Highly expensive products are out of reach for organic farmers.
Lifetime *Objective*	Withstand >1 year in field without human intervention	The product should not add additional work for a farmer during the farming season.
Portability *Objective*	<30 lbs total weight	An adult should be fully capable of transporting this object.

Final Design

Full System: The full system design consists of detection and deterrence sub-systems communicating through a WiFi network and sharing power from a 12V 25AH LiFePO4 battery. When a detection is made, a GET request must be sent via the ECE449 network to an HTTP server with static IP belonging to a ESP8266MOD to commence the deterrence process. Power is delivered to the detect sub-system via 7 feet of wire that carry 12V and 5V. Deutsch connectors were utilized to modularize battery load, detect load, deter load, and connections to deter components like lights, sound, and motion.

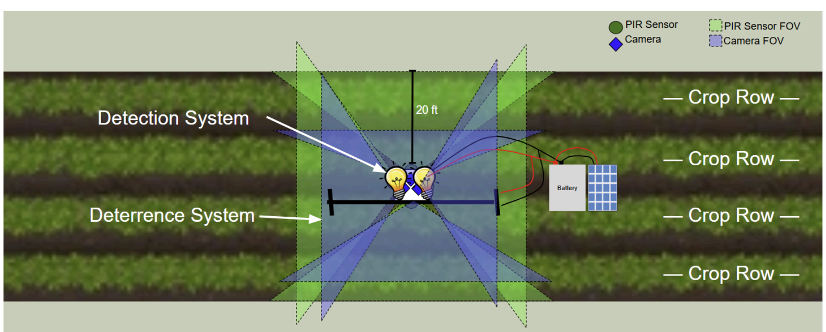


Figure 1: Full System Design

Figure 2: Implemented System at Duke Campus Farm

Detection System

Compute, WiFi, and Battery

We use a Raspberry Pi 4B as the primary computation platform for our detection subsystem. All system logic, including GPIO sampling, camera activation, image processing, and ML inference, executes locally on the Raspberry Pi. The Pi continuously monitors GPIO inputs from the four PIR sensors and interfaces with each of the four Arducam camera modules.

For wireless communication, we leverage the Raspberry Pi's built-in 2.4/5 GHz WiFi module. This enables low-latency communication with the deterrence subsystem, which is responsible for physical deterrence behaviors. Our detection subsystem operates on the same battery supply used by the deterrence subsystem; the detailed electrical design and battery implementation are described within the deterrence subsystem section of this report.

Multi-Level Detection

Our system is designed around a two-stage detection pipeline to minimize energy usage while maintaining high accuracy. Low-level detection (PIR sensors) serves as the continuous, low-power trigger mechanism, while high-level detection (camera + ML inference) provides species-specific classification when required.

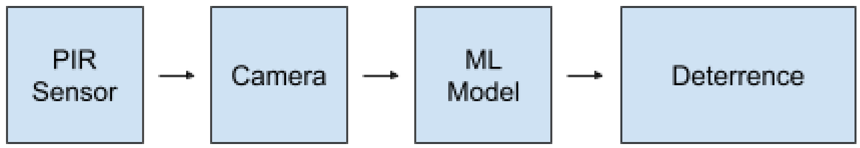


Figure 3: Overview of two-stage detection pipeline

Low-Level Detection (PIR Sensors)

We employ four Adafruit PIR motion sensors positioned at 90-degree intervals around the system enclosure. Each PIR sensor provides approximately a 120-degree sensing arc with a 20 ft range; arranging four in a circular pattern yields a full 360-degree detection field.

Each PIR outputs a digital HIGH when it detects a moving heat signature. The Raspberry Pi 4B continuously polls these PIR inputs via GPIO. When any PIR sensor is triggered:

1. The corresponding high-level detection camera is activated.
2. If the trigger occurs at night, the detection subsystem issues a WiFi signal to the deterrence subsystem, requesting a brief activation of their lighting system to serve as a camera flash.

Because PIR sensors draw minimal current, this architecture ensures the system remains energy-efficient by avoiding unnecessary camera or model activation during idle periods.

High-Level Detection (Camera + ML Model)

For high-level detection, we utilize four 12 MP Arducam IMX708 camera modules, each with a 120-degree field of view. Each camera is mounted directly above a corresponding PIR sensor, forming four PIR–camera pairs.

When PIR sensor *i* is triggered, camera *i* is activated. Upon activation, the system turns on a brief flash (provided by the deterrence subsystem's lighting when necessary) and captures an image. This image is then passed directly to the onboard ML model for inference. The model determines whether the trigger was caused by a pest (species-level), a human, or a false activation.

The model is designed for detection distances up to 20 ft, matching the PIR sensors' effective range. System behavior following high-level detection:

- If no animal is detected: The system returns to PIR polling.

- If a target pest species is detected: The Raspberry Pi sends a WiFi signal to the deterrence subsystem to activate deterrence.

ML Model

The species classification model is based on the YOLOv8 architecture. We trained a lightweight YOLOv8 variant using a custom dataset curated specifically for three target pest species: raccoon, squirrel, and groundhog.

Figure 4: Example annotated training images (Blue = raccoon and Green = groundhog)

To ensure robustness, the dataset contains images captured at various distances (including the full 20 ft), angles, and environmental backgrounds. The model was trained on a total of 600+ images taken in the lab, around campus, and on the farm. This resulted in a model that is background-invariant, angle-invariant, and distance-invariant.

During inference, a minimum confidence threshold of 0.25 is required for a positive detection. In practice, confidence scores typically fall between 0.6 and 0.8, indicating reliable model performance.

Once inference is complete, the detection result is passed to the system logic, which determines whether the deterrence subsystem should be activated. If the model has detected an animal with the minimum required confidence, it will send a WiFi ping to the deterrence system.

Figure 5: YOLOv8 model inference (Demo Day)

Deterrence System

WiFi, Compute, and Battery

We use an ESP8266MOD as an HTTP server. It's capable of receiving GET requests for "on" and "lights" functions. When 'on', we trigger the full deterrence. For 'light', we aim to trigger only the lights to provide lighting at nighttime for detection picture taking. ESP8266MOD code is shown in appendix B.

The WiFi module sends a HIGH signal to Arduino Mega 2560, on which the main computation occurs. Primarily, we check the validity of the signal, and then if necessary, trigger each of the three subsystems (Light, Sound, Motion). Because the Arduino Mega is a single core, simultaneous execution requires non-blocking code, using frequent time measurements to determine execution length. The code running on the Arduino Mega is shown in appendix A.

A provided ERYY 12V 25AH LiFePO4 battery, attached to a solar panel, is used to power the system. The battery outputs a steady 12V signal; thus, we use a standard voltage regulator to convert the 12V into a 5V supply used by the ESP8266MOD.

Sound

Our sound system consists of an SD card and reader, an audio amplifier, a speaker, and a speaker holder. When the system receives the requisite WiFi signal, the Arduino begins reading a random audio file (high pitched screams or wolf sounds) from the SD card. We use a generic 8GB SD card and a HiLetGo-supplied Micro SD Reader Module (part 3-01-0038) to read the card. The audio file is read into the Arduino, and then an audio signal is generated and sent to the amplifier. We use the Adafruit Stereo 20W Class D Audio Amplifier (MAX9744) operating at 12V to amplify the signal. It is then connected to a single waterproof speaker: MISCO OEM Part 90ON04-2WP. This portable, waterproof speaker is held to the zipline post via a custom 3D-printed speaker holder, designed in Autodesk Fusion.

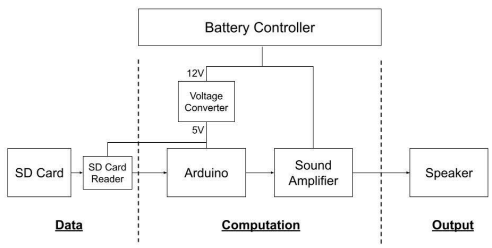


Figure 6: Sound System Diagram.

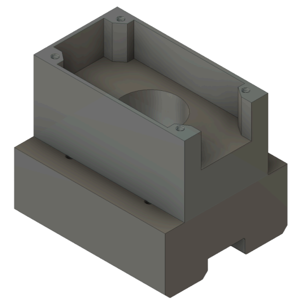


Figure 7: Custom 3D-printed speaker holder

Lights

Our simple light system consists of a IRL2703PBF N-channel mosfet, FR302 Flyback Diode, 2 QUANS LED Flood Lights, 150Ω and 100kΩ Resistors. The mosfet is used as a switch that when activated can allow the lights to receive power. The activation comes from an output pin on deter's Arduino MEGA 2560.

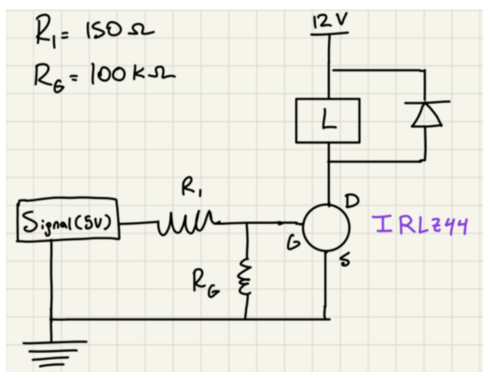


Figure 8: Lights Circuit Diagram

Motion

Our motion system is a zipline, which moves a scarecrow back and forth over the crop row. The zipline is suspended between two 7 foot metal posts (1 foot in the ground) spaced 15 feet apart. It consists of two lines, a motion line and a load line. Our system utilizes a lot of 3D printed components to house and drive our motion line. To start, the line is driven over a 3D printed wheel by a DC motor. The load line is static, carrying a trolley which holds our deterrence object. The zipline trolley is fastened to the motion line with a short piece of cable. This design reduces stress on the motor, since the weight of our deterrence object is not supported by the moving line but rather the static load line. Both lines are tensioned with steel turnbuckles. The lines are ⅛" diameter steel cable, clamped and reinforced with rope thimbles to prevent wear.

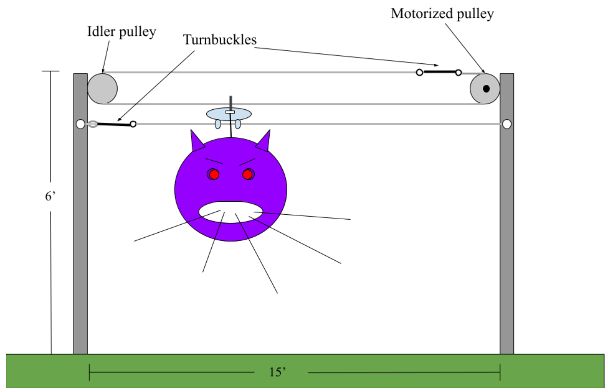


Figure 9: Initial sketch of our motion system

We 3-D printed a wheel which fastens to the motor shaft, a motor housing to waterproof the motor, and triangular mounting pieces with four M5 screw holes to fasten the motor and idler pulley on the far side of the system to the two posts.

Limit switches are used to reverse the direction of the motor when the object reaches one of the posts. To ensure our deterrence object has a reliable contact with the limit switch, we 3-D printed a limit switch housing which extends several inches from the post and a trolley attachment which increases the contact surface area. The signals from the two limit switches are routed to pins on our arduino. When our trolley contacts a limit switch, the corresponding pin reads high. When this happens, we reverse the direction of the motor by flipping the left and right PWM signals to our motor driver in our software.

Motion Components

Figure 10: Post with Motor Setup

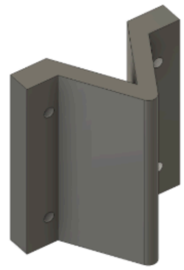


Figure 11: Rail Fastener Piece with M5 holes

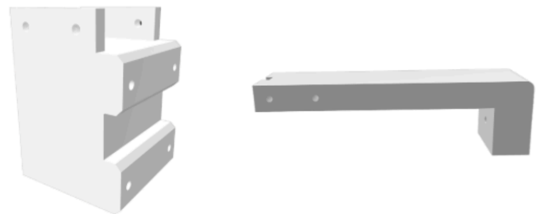


Figure 12: Limit Switch Post attachment and holder

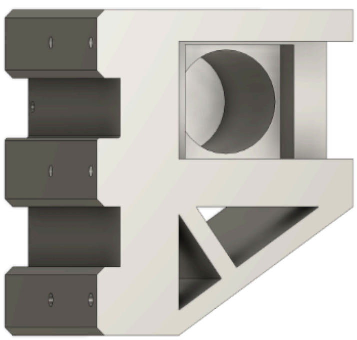


Figure 13: Motor House

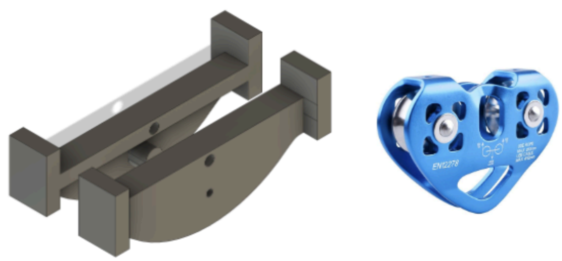


Figure 14: Zipline Trolley and Attachment

Tests

Table 2: Tests and Results

Design Criteria	Target Value	Test Overview	Results
WiFi	≥90% messages transmitted within 3 seconds	Send 50 messages from the detection system to the deterrence system and measure success rate.	Success. The detect team was able to successfully send 92% of messages to deter with a working mesh node.

Photo Storage	≥95% of trigger events with correct timestamp	Trigger the system multiple times and verify that photos are stored with accurate date and time.	Success. 99% of 200 images taken were successfully saved.
Weather-Proof	IP64 water resistant	Place the enclosure in a shower for 15 minutes and check that the interior remains dry.	Success. We placed pieces of the paper in the interior of our boxes and they remained dry after the outside was exposed to water.
Detection	≥90% of target pests detected	Use heated warmies to simulate pests and measure the recall of the detection system in day and night conditions.	Fail, detection correctly identified warmies in the farm with an average recall of 86%.
Analysis	≥90% species correctly identified	Trigger the analysis system with 50 animal photos and record whether species are correctly classified.	Fail, detection correctly identified the species of warmies in the farm with an average recall of 81%.
Ease of User Access	Average score ≥4.5/5	Have 10 participants set up the system and rate its ease of use on a 0–5 scale.	Failed; averaged 3.5, claimed zipline not easy to set up
Lightweight	<40 lbs	Place the fully assembled system on a scale and measure its weight.	Success, weighs 38 pounds
Bill of Materials Cost	≤$400	Sum unit cost × quantity of all components to verify total BoM cost.	Success, full system cost $350.27 for project, $483.31 for system to be reproduced
Detection Energy Consumption	≤3 W average power	Measure total energy consumed during a typical workload cycle using a DC wattmeter.	Success; although it was difficult to accurately measure power consumption, our battery never went below 50% over the four days our system was at the farm.
Time to Detect	≤3 seconds	Measure the interval between a trigger event and LED activation indicating deterrence.	Success, average of 2.96 seconds.

Detection Radius	20 ft	Test detection at distances up to 20 ft using a heated plushie and LED confirmation.	Success, achieved 9/12 true positives when measured at the edge of our detection radius
Compliant Material	100% NOP compliant	Inventory all materials used and verify compliance with NOP standards.	Success, uses wooden box to ensure 3D-printed components that were not NOP-compliant did not make contact with the ground or surrounding crops.
Deterrence Strength over Protected Area	The deter system protects an area of 100ft (length) x 8ft (width) x 6ft (height).	User feedback survey assessing scariness of the deterrence system.	Success, five friends tasked with evaluating our system qualitatively reported that it seemed effective (4.8/5 scariness).
Manipulability	The system contains 1 on/off switch for the 3 methods: light, sound, motion.	Test the deterrence system when the switch is off and when the switch is on, ensuring that the switch deactivates the system.	Success, deterrence includes functioning off/on switch.
Animal-Friendliness	0 animals harmed while system implemented in field	Record damage to heated plushie due to deterrence motion system when dragging it within 20 foot radius of detection system.	Success, zero animals were harmed by our system.
Deter Energy Consumption/ Power	Use ≤7 W of power	Measure total energy consumed by the deterrence system during a typical workload with a DC wattmeter.	Success; although it was difficult to measure instantaneous power (upon activation deterrence runs for 5 seconds), our battery charge never went below 50%.

Discussion

Challenges

Detection Challenges

During this project we faced several major hardware and system-level challenges that required iterative debugging and redesign. One early issue was an unexpected short on our protoboard. During our first power-up test, the Raspberry Pi and components briefly turned on before a burning smell came from the step-down transformer. During the debugging process the LM7805 voltage regulator failed and briefly sent about 10 V to the Pi's 5 V pin, forcing us to rebuild and resolder everything on a new protoboard.

We also struggled to power the Raspberry Pi through its 5 V pin. Even with decoupling capacitors, the Pi did not reliably turn on. Our tests showed the Pi would not power up at 4.9 V but would at 5.1 V. Combined with the Pi's varying current draw, the LM7805 proved unreliable, so we reverted to USB-C power. We spliced a USB-A to USB-A cable and used a USB-A to USB-C adapter to provide stable power from the solar charge controller.

Another significant challenge was SD card corruption. The first instance occurred when the Pi stopped booting, and we later realized power had been cut while the system was writing data. Despite performing proper shutdowns afterward, additional power issues caused the SD card to corrupt again. The accidental 10 V surge also seemed to affect HDMI output, and we relied on Raspberry Pi Connect until discovering that our mini-HDMI cable was defective. After replacing it, we confirmed the SD card had corrupted again.

To reduce future corruption, we added a physical button read by a GPIO pin which triggered a shutdown command that finally gave us a reliable way to shut down safely and helped keep the SD card stable for the rest of the project.

Additionally, during our final demo at the farm we faced another short which occurred when the pins in the deutsch connecter became crossed and the 5V power and ground became connected. Smoke came out of the Raspberry Pi, deutsch connecter, and wires near the battery.

Deterrence Challenges

In designing and building the deterrence system, we faced many challenges ranging from quick fixes to days-long detours. Wire management was a major issue as our project increased in complexity. Soldering all the electronics onto one board was not possible; if any issues occurred in the field, diagnosing and then changing the electronics would be immensely difficult. Furthermore, simply using a breadboard would result in loose connections. Our solution was employing deutsch connectors to ensure solid connections.

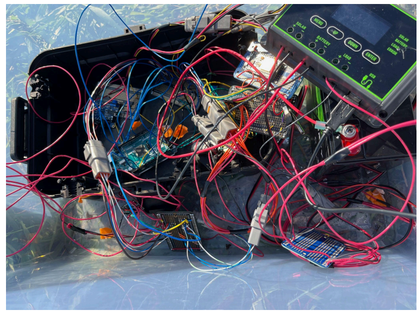


Figure 15: Electronics for deterrence system with deutsch connectors

The motion system's original motor was also an issue as our required torque exceeded our original estimate. As a solution, we bought a new motor and driver. Configuring these new parts midway through the project was a challenge as it would change our electronics and power consumption design.

Furthermore, we had challenges with components overheating and frying. Our 12 to 5 V regulator fried at some point, which sent 12V to the ESP8266 and broke it as well. We had to reevaluate how we converted voltages to ensure the regulator was rated for the correct voltage and current. After this incident, we also decided to use the Arduino to power the ESP8266, since there was no risk of the Arduino sending more than 5V to the ESP.

Analysis

Our system did not pass all of the tests that we initially planned for it, but overall we had some key successes that we believe still make the system a useful product for organic farms.

With regards to the detection system, we built a multi-stage detection scheme utilizing PIR sensors, cameras, and a lightweight ML model for analyzing photos. Our PIRs were accurately able to sense heat signatures without drawing much power. Only a positive trigger from the PIRs triggered the more power intensive camera and ML system. Our cameras worked efficiently, and the ML model was able to differentiate between pests and other things with heat signatures, and was also able to differentiate between the target pests with 81% confidence. The system accurately saved all of the taken photographs. The main hiccups involved integration of the detect and deter systems together in the farm. When sending the WiFi signals, we seemed to get network connection errors when sending messages to the deter team both for a positive detection and when the PIRs sensed something at night and transmitted a signal to turn on the lights so that a photo could be taken at night. For the entire system to work efficiently on a farm, the detect and deter subsystems should be able to communicate.

With regards to the deterrence system, we passed nearly all of our design criteria and testing. The deterrence effectiveness was quite good, averaging 4.8/5 on a user defined scale for scariness. The motion component—moving a scarecrow back and forth across a 14 foot zipline at approximately 5 miles per hour—was a strong design decision which improved the effectiveness of our deterrence. We were able to exceed coverage of the 10 foot by 10 foot square our system was intended to protect. The nature of the zipline structure also means our design is easily scalable, in both length and height. If installed over taller crops (e.g. corn), taller posts could be used to ensure no crops are harmed by our design. Because the entire system is elevated, dirt, leaves or other debris is less likely to accumulate and cause issues with the system. However, this more complicated motion design compromised our system's portability and ease of user access. Setting up a zipline is not straightforward, and much of the assembly has to be done in the field. We 3-D printed all of the pieces to mount the motion and load lines, limit switches, and speaker so that each component could be fastened to the poles with screws. However, the device used to stake the posts into the ground requires that no attachments are mounted to the top of the post during installation, which means that all of these components had to be attached in the field. This made the installation process quite time intensive. This caused us to fail our ease of user access design criteria, averaging a 3.5/5 on our user defined scale for ease of setup.

Regarding tests that our system passed, our system is capable of photo storage, weatherproofing, lightweight, low bill of materials total cost, had low detection energy consumption, had a short time to detect, had a good detection radius, was made with compliant materials, deterred over our protective area, was easily manipulatable, didn't cause physical harm to the animals, and was overall power efficient through both of the subsystems.

Regarding tests that our system failed, our system fell short with the detection, analysis, and ease of user access. Our system was able to detect with 86% accuracy, but this was a lower percentage than we were aiming for at the start of the project. Similarly, our system accurately differentiated between pest types 81% of the time, which fell short of our desired 90% accuracy at the beginning of the project.

Conclusion and Further Improvements

Although we were happy with the overall performance of the deter system, some aspects and design criteria were suboptimal such as system portability. Setting up the zipline in concept was simple and modular with our attachment pieces, however in real implementation it was time consuming and would be difficult for the average user to configure. For example, we could redesign the way in which the zipline, limit switches, and speaker attach to the posts so that they are simpler to put on and take off; in our current design, screws would get stripped, and we would spend an excessive amount of time trying to unscrew stripped screws.

Another area of potential improvement would be looking into different WiFi modules for passing signals between the detect and deter subsystems, as we ran into multiple issues with the ESP8266 in the field. We had to use another team’s WiFi module for the final demo. Additionally, the WiFi signaling efficiency went down in the field. We averaged 2.96 seconds for detection and sending the WiFi signal to deter (this included the PIR sensors, cameras, and lightweight ML analysis), but it took closer to 5 seconds out in the field due to poor connectivity with the ESP8266, so another module has the potential for faster system activation.

Our detection system also had an average recall of detecting animals at 81%, rather than our desired 90%. More fine-tuning of the model could be beneficial, although our model was able to accurately differentiate between animals. One thing that would have made testing at the farm easier would have been implementing a way to check the photos taken at the farm. With our current model, we need to bring the Raspberry Pi to a location where it could connect to the Internet to view the photos taken via Raspberry Pi Connect. We considered saving photographs to a mounted USB for easier viewing at the farm, but saving to the USB was slower than saving to the SD card.

Overall, our system successfully demonstrated a fully functional integration of detection and deterrence subsystems. Despite facing multiple hardware setbacks within both subsystems, we were able to deliver a design that met nearly all of our primary objectives of protecting at least a 10’ by 10’ crop area up to a 100’ by 10’ row. It efficiently detects the desired pests, deters humanely, uses low power consumption, and passes environmental compliance for organic farming. With continued improvements to connectivity, portability, and machine learning performance, this device could become a simple and deployable solution for real world pest mitigation.

Appendix A: Arduino Code

```
// Libraries
#include <SPI.h>
#include <SD.h>
#include <TMRpcm.h>

// Pin definitions
#define P_in A8               // wifi: receive signal to trigger
#define P_in_light A9         // wifi: receive signal to trigger light only

#define signalPin_light 2     // light
#define signalPin_sound 11    // sound: output
#define chipSelect 8          // sound: SD card reader

// Motion pins
#define IN1 12                // DRV8871 IN1
#define IN2 13                // DRV8871 IN2
#define LIM1 10               // limit switch HIGH when hit
#define LIM2 9                // limit switch 2 HIGH when hit

// constants
const int flashRateHz = 4;
const int periodMs = 1000 / flashRateHz;

// --- sound ---
TMRpcm audio;
char sounds[][15] = {
  "FOX.WAV",
  "HPITCH.WAV",
  "SCREAM1.WAV",
  "SCREAM2.WAV"
};

// variables
bool sys_triggered = false;

// --- light ---
int blinkCount = 0;
unsigned long lastBlinkTime = 0;
unsigned long blinkStart = 0;
unsigned long blinkDuration = 4000;
unsigned long blinkStartCamera = 0;
unsigned long blinkDurationCamera = 2000;
bool lightState = false;
bool blinkerDone = false;

// --- sound ---
bool soundDone = false;

// ---------------------------------------------------------------
//          MOTION STATE MACHINE VARIABLES  <<< ADDED >>>
// ---------------------------------------------------------------
```

```
bool motionDone = false;
unsigned long motionStartTime = 0;
int motionStage = 0;
// 0 = idle, 1 = forward1, 2 = backward, 3 = forward2, 4 = done

// --------------------------------------------------------------
//          MOTION HELPER FUNCTIONS  <<< ADDED >>>
// --------------------------------------------------------------
void stopMotion() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
}

void moveForward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
}

void moveReverse() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
}

// --------------------------------------------------------------
//                  SETUP
// --------------------------------------------------------------
void setup() {
  randomSeed(analogRead(A0));

  pinMode(P_in, INPUT);
  pinMode(P_in_light, INPUT);
  pinMode(signalPin_light, OUTPUT);
  pinMode(signalPin_sound, OUTPUT);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(LIM1, INPUT);
  pinMode(LIM2, INPUT);

  Serial.begin(9600);

  // SD card setup
  pinMode(chipSelect, OUTPUT);
  Serial.println("Initializing SD card...");
  if (!SD.begin(chipSelect)) {
    Serial.println("SD initialization failed!");
    while (1);
  }
  Serial.println("SD initialization done.");

  // Audio setup
  audio.speakerPin = signalPin_sound;
  audio.setVolume(4);
  audio.quality(1);
```

```
  stopMotion();   // <<< ADDED >>>
}

// ---------------------------------------------------------------
//                    MAIN LOOP
// ---------------------------------------------------------------
void loop() {

  // Main full-system trigger
  if (!sys_triggered && digitalRead(P_in) == HIGH) {
    sys_triggered = true;
    Serial.println("Trigger received! Starting systems...");
    startAllProcesses();

    // Start motion
    motionStage = 1;            // <<< ADDED >>>
    motionStartTime = millis();
    motionDone = false;
  }

  // Light-only trigger
  if (!sys_triggered && digitalRead(P_in_light) == HIGH) {
    Serial.println("Light-only trigger received!");
    blinkStartCamera = millis();
    handleBlinkerCamera();
  }

  // Run processes while active
  if (sys_triggered) {
    if (!blinkerDone) handleBlinker();
    if (!soundDone) handleSound();
    if (!motionDone) handleMotion();   // <<< ADDED >>>

    // System resets only after all three subsystems finish
    if (blinkerDone && soundDone && motionDone) {   // <<< MODIFIED >>>
      Serial.println("All systems complete. Resetting...");
      stopAllOutputs();
      resetState();
    }
  }
}

// ---------------------------------------------------------------
//                    START PROCESSES
// ---------------------------------------------------------------
void startAllProcesses() {
  blinkStart = millis();
  blinkerDone = false;

  soundDone = false;

  blinkCount = 0;
  lightState = false;
```

```
  digitalWrite(signalPin_light, LOW);
  lastBlinkTime = 0;
}

// ---------------------------------------------------------------
//                    LIGHT CONTROL
// ---------------------------------------------------------------
void handleBlinker() {
  unsigned long currentMillis = millis();

  if (currentMillis - blinkStart >= blinkDuration) {
    digitalWrite(signalPin_light, LOW);
    blinkerDone = true;
    return;
  }

  if (currentMillis - lastBlinkTime >= periodMs / 2) {
    lastBlinkTime = currentMillis;
    lightState = !lightState;
    digitalWrite(signalPin_light, lightState);

    if (lightState == LOW) {
      blinkCount++;
    }
  }
}

void handleBlinkerCamera() {
  unsigned long currentMillis = millis();

  if (currentMillis - blinkStartCamera >= blinkDurationCamera) {
    digitalWrite(signalPin_light, LOW);
    return;
  } else {
    digitalWrite(signalPin_light, HIGH);
  }
}

// ---------------------------------------------------------------
//                    SOUND CONTROL
// ---------------------------------------------------------------
void handleSound() {
  static bool started = false;
  static unsigned long startTime = 0;

  if (!started) {
    unsigned int rand = millis() % 4;
    if (SD.exists(sounds[rand])) {
      Serial.print("Playing: ");
      Serial.println(sounds[rand]);
      audio.stopPlayback();
      audio.play(sounds[rand]);
      startTime = millis();
      started = true;
```

```
    } else {
      Serial.println("Sound file not found!");
      soundDone = true;
      return;
    }
  }

  if (!audio.isPlaying() || millis() - startTime > 6000) {
    soundDone = true;
    started = false;
  }
}

// ----------------------------------------------------------------
//                    MOTION CONTROL
// ----------------------------------------------------------------
void handleMotion() {
  unsigned long now = millis();

  switch (motionStage) {

    case 1:  // Forward #1
      moveForward();
      if (now - motionStartTime >= 5000 || digitalRead(LIM1) == HIGH) {
        stopMotion();
        motionStartTime = now;
        motionStage = 2;
      }
      break;

    case 2:  // Reverse
      moveReverse();
      if (now - motionStartTime >= 5000 || digitalRead(LIM2) == HIGH) {
        stopMotion();
        motionStartTime = now;
        motionStage = 3;
      }
      break;

    case 3:  // Forward #2
      moveForward();
      if (now - motionStartTime >= 5000 || digitalRead(LIM1) == HIGH) {
        stopMotion();
        motionStage = 4;
      }
      break;

    case 4:
      stopMotion();
      motionDone = true;
      break;
  }
}
```

```
// ------------------------------------------------------------
//                    RESET LOGIC
// ------------------------------------------------------------
void stopAllOutputs() {
  digitalWrite(signalPin_light, LOW);
  noTone(signalPin_sound);

  stopMotion();  // <<< UPDATED >>>
}

void resetState() {
  sys_triggered = false;
  blinkerDone = false;
  soundDone = false;
  motionDone = true;
  lightState = false;

  digitalWrite(signalPin_light, LOW);
  Serial.println("System reset — ready for next trigger.");
}
```

Appendix B: ESP 8266 Code

```
#include <ESP8266WiFi.h>

const char* ssid = "ECE449deco";
const char* password = "ece449$$";

int P_out = D6;          // Deter signal pin
int Light_out = D5;      // Light signal pin

WiFiServer server(80);

// ---- Static IP configuration ----
IPAddress local_IP(192, 168, 68, 106);
IPAddress gateway(192, 168, 68, 1);
IPAddress subnet(255, 255, 255, 0);
// ----------------------------------

void setup() {
  pinMode(P_out, OUTPUT);
  digitalWrite(P_out, LOW);

  pinMode(Light_out, OUTPUT);
  digitalWrite(Light_out, LOW);

  Serial.begin(9600);

  // Apply static IP BEFORE WiFi.begin
  if (!WiFi.config(local_IP, gateway, subnet)) {
    Serial.println("Static IP Failed!");
  }

  WiFi.begin(ssid, password);

  Serial.println("Connecting to WiFi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nWiFi connected!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  server.begin();
}

void loop() {
  WiFiClient client = server.available();
  if (!client) return;

  Serial.println("Client connected");
  String req = client.readStringUntil('\r');
```

```
  Serial.print("Request: ");
  Serial.println(req);
  client.flush();

  if (req.indexOf("/on") != -1) {
    Serial.println("Deter System ON");
    sendDeterPage(client);
    gotSignal();
  } else if (req.indexOf("/lights") != -1) {
    Serial.println("Lights ON");
    sendLightPage(client);
    lightSignal();
  } else {
    Serial.println("Unknown request");
  }

  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close\r\n");
  client.println("<!DOCTYPE HTML><html>");
  client.println("<h2>ESP8266 → Mega LED Control</h2>");
  client.println("<a href=\"/on\">Turn Deter System On</a><br>");
  client.println("<a href=\"/lights\"> Turn Lights On</a>");
  client.println("</html>");

  delay(1);
  Serial.println("Client disconnected");
}

void gotSignal() {
  Serial.println("Sending deter signal");
  digitalWrite(P_out, HIGH);
  delay(2000);
  digitalWrite(P_out, LOW);
}

void lightSignal() {
  Serial.println("Sending light signal");
  digitalWrite(Light_out, HIGH);
  delay(2000);
  digitalWrite(Light_out, LOW);
}

void sendLightPage(WiFiClient &client) {
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close\r\n");
  client.println("<html><body>");
  client.println("<h2>Lights turning on now</h2>");
  client.println("</body></html>");
}

void sendDeterPage(WiFiClient &client) {
  client.println("HTTP/1.1 200 OK");
```

```
  client.println("Content-Type: text/html");
  client.println("Connection: close\r\n");
  client.println("<html><body>");
  client.println("<h2>Deter system activating now</h2>");
  client.println("</body></html>");
}
```

Appendix C: Bill of Materials

The following bill of materials was created with the assumption that our solution would be made in bulk in the order of 1000s to 10000s.

Item	Quantity (per solution)	Cost
Detection Subsystem		
Raspberry Pi 4B	1	$38.08
12MP IMX708 Arducam	4	$184.16
Multi Camera Adapter Module V2.2 for Raspberry Pi Camera Module 3 12MP IMX708 / 5MP OV5647 / 8MP IMX219 / 12MP IMX477 Cameras	1	$49.99
PIR Sensors	4	$4.80
USB A Cable	1	$1.73
USB Type A to Type C Adaptor	1	$1.40
PLA for enclosure	1 kg	$2.31
Wood	1.36 kgs	$2.70
Aluminum Heat Sink for Raspberry Pi 4	1	$1.95
	Detection Subsystem Total	$287.12
Deterrence Subsystem		
12 V DC Motor	1	$20.00
DRV8871 Motor Driver	1	$6.00
M8 hook-hook steel turnbuckles	2	$5.00
⅛” Steel Cable	50 feet	$9.42

⅛" cable clamps	10	$2.10
⅛" rope thimble	4	$1.88
Zipline Trolley	1	$9.66
HiLetGo MicroSD Reader Module	1	$1.40
8 GB SD Card	1	$4.30
Adafruit Class D 20W Amplifier	1	$15.96
Waterproof Full Range Speaker, 4 Ohm	1	$15.00
4-inch Square LED Flood Lights	2	$59.99
IRL2703PBF MOSFET N	1	$1.43
Fly Back Diode	1	$0.08
Arduino Mega 2560	1	$36.99
ESP8266	1	$4.67
PLA	1 kg	$2.31
	Deterrence Subsystem Total	$196.19
	Total System	**$483.31**

Appendix D: Gantt Chart

Task		10/6/2025	10/9/2025	10/13/2025	10/16/2025	10/20/2025	10/23/2025	10/27/2025	10/30/2025	11/3/2025	11/7/2025
WiFi Subsystem	Test Raspberry Pi Wifi Capabilities										
	Test Arduino/ESP8266 Wifi Capabilities										
Power Subsystem	Verify Solar Panel Functionality										
	Verify Battery Functionality										
	Test Battery Capacity										
	Set up battery controller										
	Connect battery controller to Microcontrollers										

Task		10/6/2025	10/9/2025	10/13/2025	10/16/2025	10/20/2025	10/23/2025	10/27/2025	10/30/2025	11/3/2025	11/7/2025
Motion Deterrence Subsystem	Time to receive orders										
	3-D print gears, shaft connectors										
	Assemble motor-gear system, wire arduino-controller-motor, test basic motion										
	3-D print motor enclosure										
	Suspend zipline cable, pulley system between posts										
	Verify motion of trolley (no suspended deterrence object)										

	Add deterrence object, verify motion										
	Construct wiring system to mount speaker on deterrence object (spooling/unspooling of wire)										
	Finalize control code										

Task		10/6/2025	10/9/2025	10/13/2025	10/16/2025	10/20/2025	10/23/2025	10/27/2025	10/30/2025	11/3/2025	11/7/2025
Light Deterrence Subsystem	Time to receive orders										
	prototype/test circuit										
	make any changes/solder protoboard										
	build encasing for circuit										
Sound Deterrence Subsystem	Time to receive orders										
	Find Stockroom Parts										
	Set up SD card reader and store random noises										
	Set up audio amplifier										
	Connect amplifier to speaker										
	Write control logic for SD card reading and audio amplification										

Task		10/6/2025	10/9/2025	10/13/2025	10/16/2025	10/20/2025	10/23/2025	10/27/2025	10/30/2025	11/3/2025	11/7/2025
Detection	Set up PIR sensors + circuitry										
	Train ML model with dataset										
	Connect low and high power detection systems										
	Ensure detection system works as 1										
Enclosure	Design enclosure using CAD software										
	Print/create physical enclosure										
	Ensure that PIR can detect through enclosure										
	Ensure that camera can see in enclosure										
Analysis	Train ML model with dataset										
	Analysis accuracy										
Monitoring Photo	Connect cameras to Raspberry Pi										
	Connect monitoring photograph lights										
	Ensure accurate photograph triggering										
	Ensure reliable data storage										
	Ensure correct date and time recorded										

Appendix E: lightweightml.py (Raspberry Pi)

```
# lightweightml.py
from pathlib import Path
import cv2
from ultralytics import YOLO
import time
from datetime import datetime

# global vars
MODEL_PATH = "/home/ratwranglers/ece449_project/models/ishan_v4.pt"
CONF = 0.25
IOU = 0.1

model = YOLO(MODEL_PATH)

def detect_animal(image_path: str):
   start_time = datetime.now()

   img_path = Path(image_path)
   if not img_path.exists():
       raise FileNotFoundError(img_path)

   # Run inference
   results = model(str(img_path), conf=CONF, iou=IOU)[0]

   confidences = [float(b.conf[0]) for b in results.boxes]
   animal_detected = len(confidences) > 0

   # Save annotated image
   annotated = results.plot()
   out_path = f"{image_path}"
   cv2.imwrite(str(out_path), annotated)

   trigger_time = datetime.now() - start_time

   return {
       "animal_detected": animal_detected,
       "confidences": confidences,
       "annotated_path": str(out_path),
       "trigger_time": trigger_time
   }
# Optional: allow running directly as a script
if __name__ == "__main__":
   import sys
   if len(sys.argv) != 2:
       print("Usage: python3 lightweightml.py <image_path>")
       sys.exit(1)
   result = detect_animal(sys.argv[1])
  print(result)
```

Appendix F: photo_trigger.py (Raspberry Pi)

```
import RPi.GPIO as GPIO
import subprocess
import time
import requests
from datetime import datetime
import json
from lightweightml import detect_animal
import os

LOG_FILE = "/home/ratwranglers/Desktop/cronlog.txt"

with open("/home/ratwranglers/Desktop/cronlog.txt", "a") as f:
   f.write("SCRIPT IS SCRIPTINGGG!!!" + "\n")

#check if the current hour is between 5pm and 7am
def check_nighttime():
   now = datetime.now()
   now_hour = now.hour
   return ((now_hour >= 17) or (now_hour <= 7))

def print_text(text):
   with open(LOG_FILE, "a") as f:
       f.write(text + "\n")

"21: Front PIR(1),26: PIR(2),20: 3, 6: Back PIR(4)"
# store gpio pins to listen to and assign cameras to those pins
GPIO_PINS = [20, 21, 26, 6]
POWER_PIN = 13
BUTTON_PIN = 19
GPIO.setmode(GPIO.BCM)

for pin in GPIO_PINS:
   GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

GPIO.setup(POWER_PIN, GPIO.OUT)
GPIO.output(POWER_PIN, GPIO.HIGH)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# GPIO_PINS = [20]
CAM_ASSIGN = {6:'0', 26:'1', 21:'3', 20:'2'}

# the amount of time we will wait after a pin goes low before allowing it to scan
again
DEBOUNCE_TIME = 0.2
```

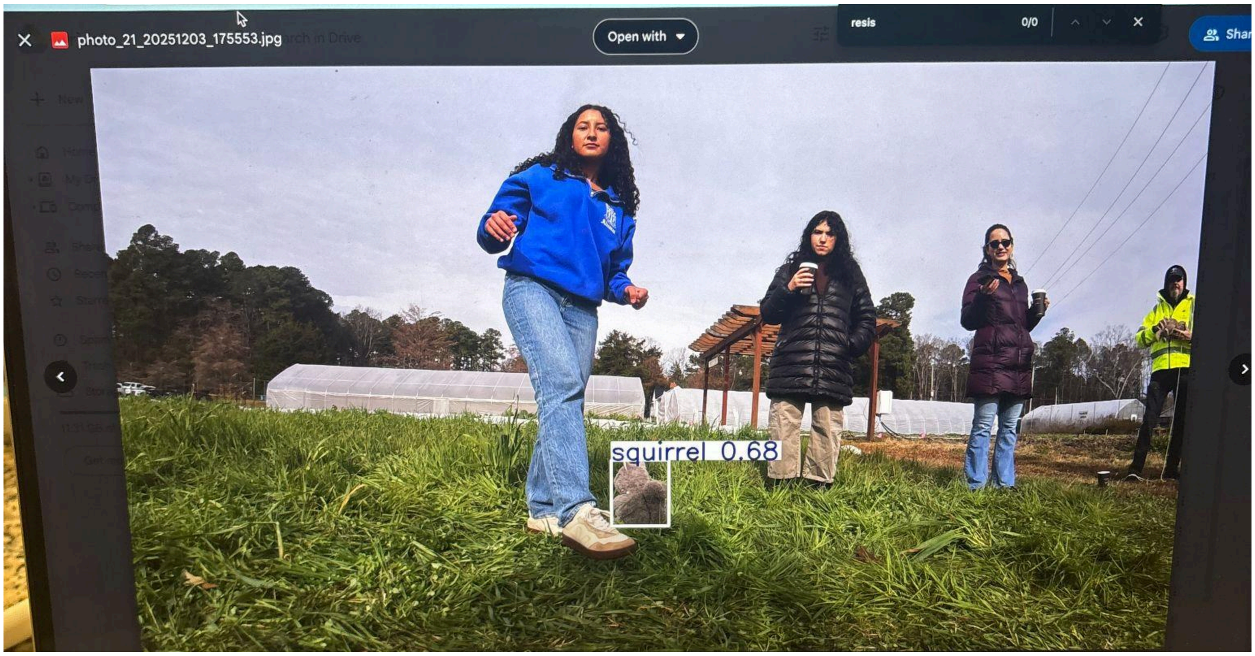
```
# the maximum amt of time we wait for a pin to be scanning high before allowing it to
scan again
MAX_HIGH_DURATION = 30.0

# photo directory to save photos into
PHOTO_DIR = "/home/ratwranglers/ece449_project/test_photos/"

# track which pin is triggered and when it was triggered and what time it goes low
triggered_pin = None
trigger_time = None
low_time = None

# take photo method takes a photo and stores it. Outputs the result of the ML model
def take_photo(pir_sensor):
   timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')

   #photo_path = f"{PHOTO_DIR}photo_{pir_sensor}_{timestamp}.jpg"
   filename = f"photo_{pir_sensor}_{timestamp}.jpg"
   photo_path = os.path.join(PHOTO_DIR, filename)
   subprocess.run(["rpicam-still", "--camera", CAM_ASSIGN[pir_sensor], "-o",
photo_path], stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)

   print_text(f"Photo takesn and saved to {photo_path}")
   return detect_animal(photo_path)

# returns if any of the scannable GPIO pins are active
def scan_gpio_pins(pins_to_scan):
   for gpio_pin in pins_to_scan:
       if GPIO.input(gpio_pin) == GPIO.HIGH:
           return gpio_pin
   return -1

# checks to see if any given pin should be unblocked
# returns a boolean for whether all pins should be unblocked or an array of the pins
to be unblocked
def check_unblock_conditions():
   global triggered_pin, trigger_time, low_time

   if triggered_pin is None:
       return True, GPIO_PINS

   # store the time since trigegring and the current state of the pin
   elapsed_since_trigger = time.time() - trigger_time
   triggered_pin_state = GPIO.input(triggered_pin)

   # track when the pin goes low
   if triggered_pin_state == GPIO.LOW and low_time is None:
       low_time = time.time()
       print_text(f"Pin {triggered_pin} went LOW at {low_time}")

   # if a pin goes HIGH again, reset the low_time
   if triggered_pin_state == GPIO.HIGH and low_time is not None:
       print_text(f"Pin {triggered_pin} went HIGH again, resetting the low_time")
       low_time = None
```

```
    # condition 1: triggered pin went low and debounce time has passed since going low
-> all pins now unblocked
    if triggered_pin_state == GPIO.LOW and low_time is not None:
        elapsed_since_low_time = time.time() - low_time
        if elapsed_since_low_time > DEBOUNCE_TIME:
            print_text(f"Pin {triggered_pin} has been LOW for
{elapsed_since_low_time:.2f}s (>= {DEBOUNCE_TIME}s) - unblocking ALL pins")
            triggered_pin = None
            trigger_time = None
            low_time = None
            return True, GPIO_PINS

    # condition 2: max duration has been exceeded while the same pin has remained HIGH
    if triggered_pin_state == GPIO.HIGH and elapsed_since_trigger >= MAX_HIGH_DURATION:
        print_text(f"Pin {triggered_pin} still HIGH after {MAX_HIGH_DURATION}s -
unblocking all pins EXCEPT {triggered_pin}")
        # reset all pins EXCEPT the triggered pin
        return False, [p for p in GPIO_PINS if p != triggered_pin]

    # if neither condition is met, return false and keep all pins blocked
    return False, None

try:
    print_text(f"Monitoring GPIO pins {GPIO_PINS}...")
    print_text(f"Debounce time: {DEBOUNCE_TIME}s, Max high duration:
{MAX_HIGH_DURATION}s")

    while True:

        if GPIO.input(BUTTON_PIN) == GPIO.HIGH:
            print_text("shutdown button has been pressed, shutting down")
            subprocess.run(["sudo", "shutdown", "-h", "now"])
            time.sleep(0.5)
            continue

        # get the pins which can be scanned/unblocked
        all_unblocked, scannable_pins = check_unblock_conditions()

        # only scan if we have pins available to scan
        if scannable_pins is not None and len(scannable_pins) > 0:
            # get active pin
            active_pin = scan_gpio_pins(scannable_pins)

            # if there is an active pin
            if active_pin != -1:
                # output that the pin is HIGH
                print_text(f"HIGH DETECTED FROM PIN {active_pin}")

                # block ALL pins immediately
                triggered_pin = active_pin
                trigger_time = time.time()
                low_time = None
                print_text(f"ALL pins blocked at {time.time()}")

                #checking if lights need to be turned on
```

```
                if not check_nighttime():
                    print_text("not nighttime, no light needed")
                if check_nighttime():
                    print_text("nighttime sending wifi signal")
                    try:
                        resp = requests.get("http://192.168.68.106/lights",
timeout=0.5)
                        print_text(f"sent light signal")
                        #time.sleep(0.5)
                    except Exception as e:
                        print_text(f"Error sending light signal: {e}")

                # take photo and process results
                detectionResults = take_photo(active_pin)
                print_text(json.dumps(detectionResults, indent=2, default=str))

                # if animal is detected, trigger wifi call to deter system
                if detectionResults["animal_detected"] == True:
                    print_text(f"DETER!!! RUN!!!")
                    try:
                        resp = requests.get("http://192.168.68.106/on", timeout=5)
                        print_text(f"sent deter signal")
                    except Exception as e:
                        print_text(f"Error sending deter signal: {e}")

            time.sleep(0.05)

except KeyboardInterrupt:
   print_text("Exiting program..")

finally:
   GPIO.cleanup()
```

Appendix G: Example of Positive Detection in the Field

```
SCRIPT IS SCRIPTINGGG!!!
Monitoring GPIO pins [20, 21, 26, 6]...
Debounce time: 0.2s, Max high duration: 30.0s
HIGH DETECTED FROM PIN 21
ALL pins blocked at 1764802606.035721
nighttime sending wifi signal
Error sending light signal: HTTPConnectionPool(host='192.168.68.106', port=80): Max retries exceeded with url: /lights (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f668fd940>: Failed to establish a new connection: [Errno 101]
Network is unreachable'))
Photo takesn and saved to /home/ratwranglers/ece449_project/test_photos/photo_21_20251203_175646.jpg

image 1/1 /home/ratwranglers/ece449_project/test_photos/photo_21_20251203_175646.jpg: 576x1024 1 raccoon, 4886.1ms
Speed: 107.3ms preprocess, 4886.1ms inference, 80.3ms postprocess per image at shape (1, 3, 576, 1024)
{
  "animal_detected": true,
  "confidences": [
    0.7100984454154968
  ],
  "annotated_path": "/home/ratwranglers/ece449_project/test_photos/photo_21_20251203_175646.jpg",
  "trigger_time": "0:00:07.045196"
}
DETER!!! RUN!!!
sent deter signal
Pin 21 went LOW at 1764802621.244125
Pin 21 has been LOW for 0.20s (>= 0.2s) - unblocking ALL pins
```



References

1. USDA, “USDA ERS - Organic Agriculture,” Usda.gov, Jan. 23, 2025. https://www.ers.usda.gov/topics/natural-resources-environment/organic-agriculture
2. “Food Industry Facts,” www.fmi.org. https://www.fmi.org/our-research/food-industry-facts
3. S. Mondal and R. Paily, "Efficient Solar Power Management System for Self-Powered IoT Node," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 9, pp. 2359-2369, Sept. 2017, doi: 10.1109/TCSI.2017.2707566.
4. S. K. Ram, S. R. Sahoo, K. Sudeendra and K. Mahapatra, "Energy Efficient Ultra Low Power Solar Harvesting System Design with MPPT for IOT Edge Node Devices," 2018 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), Hyderabad, India, 2018, pp. 130-133, doi: 10.1109/iSES.2018.00036.
5. R. L. R. Singh, V. Thirumalaivasan, N. N. Kumar and R. Adharsh, "Design and Development of Smart System for the Identification and Diversion of Wild Animal in Agricultural Field," 2024 10th International Conference on Advanced Computing and

Communication Systems (ICACCS), Coimbatore, India, 2024, pp. 585-590, doi: 10.1109/ICACCS60874.2024.10717270.

6. R. Arun, L. J. Usha, A. Sharma and L. Jha, "Integrating Camera Technology and Sound Frequencies to Mitigate Wildlife-Caused Crop Damage," 2025 International Conference on Emerging Technologies in Engineering Applications (ICETEA), Puducherry, India, 2025, pp. 1-5, doi: 10.1109/ICETEA64585.2025.11099772.
7. A. Arnita, C. Cahayahati, and M. Mirzazoni, “The development of APIO: Solar Power Plant based technology for automatic fish dryer equipment,” Andalas Journal of Electrical and Electronic Engineering Technology, vol. 3, no. 1, pp. 30–34, May 2023, doi: 10.25077/ajeeet.v3i1.36
8. N. N. Serge and F. D. Husserl, "IoT and Arduino Based Design of a Solar, Automated and Smart Greenhouse for Vegetable," E3S Web of Conferences, vol. 354, 2022.
9. D. Ramesh, M. Chandrasekaran, R. P. Soundararajan, P. P. Subramanian, V. Palled, and D. P. Kumar, “Solar-Powered Plant Protection Equipment: Perspective and Prospects,” Energies, vol. 15, no. 19, p. 7379, Oct. 2022, doi: https://doi.org/10.3390/en15197379.
10. P. Srivastava, M. Bajaj and A. S. Rana, "Overview of ESP8266 Wi-Fi module based Smart Irrigation System using IOT," 2018 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, India, 2018, pp. 1-5, doi: 10.1109/AEEICB.2018.8480949.
11. M. F. Shaik, R. Mounika, A. D. Prasad, I. R. Raja, B. P. Sekhar and D. Sampath, "Intelligent Secure Smart Crop protection From Wild Animals," 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2022, pp. 321-325, doi: 10.1109/ICACCS54159.2022.9785265.
12. L. Baumgärtner, A. Penning, P. Lampe, B. Richerzhagen, R. Steinmetz and B. Freisleben, "Environmental Monitoring Using Low-Cost Hardware and Infrastructureless Wireless Communication," 2018 IEEE Global Humanitarian Technology Conference (GHTC), San Jose, CA, USA, 2018, pp. 1-8, doi: 10.1109/GHTC.2018.8601883.
13. M. F. Akorede, J. J. Fatigun, J. A. Opaluwa and E. Pouresmaeil, "Efficient remote control system using SMS and WiFi technology for outdoor security lighting applications," 2017 IEEE PES PowerAfrica, Accra, Ghana, 2017, pp. 512-517, doi: 10.1109/PowerAfrica.2017.7991278.

14. K. Gomez, R. Riggio, T. Rasheed and F. Granelli, "Analysing the energy consumption behaviour of WiFi networks," 2011 IEEE Online Conference on Green Communications, (Online Conference) Piscataway, NJ, USA, 2011, pp. 98-104, doi: 10.1109/GreenCom.2011.6082515.
15. S. Heo, N. Baumann, C. Margelisch, M. Giordano and M. Magno, "Low-cost Smart Raven Deterrent System with Tiny Machine Learning for Smart Agriculture," 2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Kuala Lumpur, Malaysia, 2023, pp. 1-6, doi: 10.1109/I2MTC53148.2023.10175902.
16. V. Atchaya, V. Kowsalya, K. P. Dhivya Bharathi, and M. Arunkumar, "Implementation of Crop Protection System Against Wild Animals Attack," Sengunthar Engineering College, Tiruchengode, India, 2021.
17. N. Ananya, D. Chandrika, A. Bhavani, and T. V. Reddy, "Protection of Crops from Wild Animals," International Journal of Creative Research Thoughts (IJCRT), vol. 8, no. 8, pp. 1–6, Aug. 2020. ISSN: 2320-2882.
18. M. A. M. Fisol and W. M. Jubadi, "Ultrasonic and infrared repelling device for controlling the population of rat in paddy field," 2010 IEEE Asia Pacific Conference on Circuits and Systems, Kuala Lumpur, Malaysia, 2010, pp. 359-361, doi: 10.1109/APCCAS.2010.5774839.
19. N. Abed, R. Murgun, A. Deldari, S. Sankarannair, and M. V. Ramesh, “IoT and AI-driven solutions for human-wildlife conflict: advancing sustainable agriculture and biodiversity conservation,” Smart Agricultural Technology, p. 100829, Feb. 2025, doi: https://doi.org/10.1016/j.atech.2025.100829.
20. Geerthik S, S. G. A, Vishal B, and K. T. Rao, “IoT-Based System for Humane Animal Deterrence and Sustainable Crop Management,” pp. 770–775, Feb. 2025, doi: https://doi.org/10.1109/idciot64235.2025.10915026.
21. R. A. Kastelein et al, "Behavioral Responses of a Harbor Porpoise (Phocoena phocoena) to Sounds from an Acoustic Porpoise Deterrent," Aquat. Mamm., vol. 43, (3), pp. 233-244, 2017
22. R. A. Kastelein et al, "Behavioral Responses of a Harbor Porpoise (Phocoena phocoena) to Sounds from an Acoustic Porpoise Deterrent," Aquat. Mamm., vol. 43, (3), pp. 233-244, 2017

23. A. JOSHI, N. PANDEY, MANOJ DIWAKAR, P. SINGH, A. SHANKAR, and F. ALQAHTANI, “WBD-YOLO-AM: YOLOv8 with attention module and IoT-based wild boar detection and deterrence system for safeguarding small millets,” TURKISH JOURNAL OF AGRICULTURE AND FORESTRY, vol. 49, no. 4, pp. 769–786, Aug. 2025, doi: https://doi.org/10.55730/1300-011x.3303.
24. A. Muminov, Y. C. Jeon, D. Na, C. Lee, and H. S. Jeon, “Development of a solar powered bird repeller system with effective bird scarer sounds,” IEEE Xplore, Nov. 01, 2017. https://ieeexplore.ieee.org/document/8188587/authors#authors
25. K. Chitra et al., “Animals Detection System In The Farm Area Using Iot,” Jan. 2023, doi: https://doi.org/10.1109/iccci56745.2023.10128557.
26. M. M., L. A. Kurup, S. Gopinath and A. C. P., "An Efficient and Smart Animal Repulsion Technology using Laser Sensors and Acoustic Signals," 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), Kannur, India, 2022, pp. 105-110, doi: 10.1109/ICICICT54557.2022.9917646.
27. J. P. Pecaso, "Farm bird-deterrent device: A prototype," Design Engineering, pp. 530–539, Jun. 29, 2021.
28. “CONTENTdm,” Oclc.org, 2025. https://nwrc.contentdm.oclc.org/digital/collection/NWRCPubs1/id/31200 (accessed Sep. 12, 2025).
29. G. M. Linz, H. J. Homan, S. J. Werner, H. M. Hagy, and W. J. Bleier, “Assessment of Bird-management Strategies to Protect Sunflowers,” BioScience, vol. 61, no. 12, pp. 960–970, Dec. 2011, doi: https://doi.org/10.1525/bio.2011.61.12.6.
30. R. K. Megalingam et al., "Design and Implementation of Junction Boxes for Preventing Animal Attacks in Farms," 2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS), Bhopal, India, 2020, pp. 1-5, doi: 10.1109/SCEECS48394.2020.119.
31. A. Ramkumar, A. Deniston, K. Kishore and R. Faizuddin, "IOT solution for crop protection from wild boar attack," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 2021, pp. 1-6, doi: 10.1109/ICAECA52838.2021.9675503.

32. S. K, S. S, S. S and S. S, "Intelligent IoT System for Animal Detection and Crop Defense Using Acoustic and Visual Repellents," 2025 3rd International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA), Namakkal, India, 2025, pp. 1-5, doi: 10.1109/AIMLA63829.2025.11040691.
33. A. E. Koehler, R. E. Marsh, and T. P. Salmon, "Frightening methods and devices/stimuli to prevent mammal damage--a review," Proceedings of the Vertebrate Pest Conference, vol. 14, no. 14, 2024, Available: https://escholarship.org/uc/item/6mg4f2db
34. P. N. Karunanayake, N. De, J A C L C Jayasundara, Y. G. Wanniarachchi, and A P I Karunarathne, "INTELLIGENT PEST REPELLENT SYSTEM FOR SRI LANKAN FARMING INDUSTRY," IESL 111th Annual Sessions of the Institution of Engineers Sri Lanka, Dec. 2017
35. J. M. Gilsdorf, S. E. Hygnstrom, and K. C. VerCauteren, "Use of frightening devices in wildlife damage management," Integrated Pest Management Reviews, vol. 7, no. 1, pp. 29–45, Mar. 2002, doi: 10.1023/A:1025760032566
36. P. J. Robinson and W. E. Easterling, "Solar Energy Climatology of North Carolina," *Journal of applied meteorology*, vol. 21, no. 11, pp. 1730–1738, Nov. 1982, doi: https://doi.org/10.1175/1520-0450(1982)021%3C1730:seconc%3E2.0.co;2.
37. M. Hasan Sajib Md. and S. Abu Sadat Md., "Innovations in Sensor-Based Systems and Sustainable Energy Solutions for Smart Agriculture: A Review," *Encyclopedia*, vol. 5, no. 2, pp. 67–67, May 2025, doi: https://doi.org/10.3390/encyclopedia5020067.
38. Y. Li and J. Wu, "Optimum Integration of Solar Energy With Battery Energy Storage Systems," *IEEE Transactions on Engineering Management*, vol. 69, no. 3, pp. 1–11, 2020, doi: https://doi.org/10.1109/TEM.2020.2971246.
39. M. S. Mahmoud and A. A. H. Mohamad, "A Study of Efficient Power Consumption Wireless Communication Techniques/ Modules for Internet of Things (IoT) Applications," *Advances in Internet of Things*, vol. 06, no. 02, pp. 19–29, 2016, doi: https://doi.org/10.4236/ait.2016.62002.
40. Suh Young Lee, Seon Jeong Kim, and S. Kim, "Implementation of a low power motion detection camera processor using a CMOS Image Sensor," *2004 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2004, doi:

https://doi.org/10.1109/iscas.2004.1329377.

41. B. Martinez, M. Monton, I. Vilajosana, and J. D. Prades, “The Power of Models: Modeling Power Consumption for IoT Devices,” *IEEE Sensors Journal*, vol. 15, no. 10, pp. 5777–5789, Oct. 2015, doi: https://doi.org/10.1109/jsen.2015.2445094.
42. T. Wang, S. Ren, and H. Zhang, “Nighttime wildlife object detection based on YOLOv8‐night,” *Electronics Letters*, vol. 60, no. 15, Aug. 2024, doi: https://doi.org/10.1049/ell2.13305.
43. H. Li and X. Wu, “Pest-YOLOv8: A Novel Lightweight Algorithm for Pest Detection,” *2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pp. 626–630, Mar. 2024, doi: https://doi.org/10.1109/ainit61980.2024.10581830.
44. T. S. Delwar et al., “Real-Time Farm Surveillance Using IoT and YOLOv8 for Animal Intrusion Detection,” *Future Internet*, vol. 17, no. 2, p. 70, Feb. 2025, doi: https://doi.org/10.3390/fi17020070.
45. A. Mane, A. Mane, P. Dhake, and S. Kale, “Smart Intrusion Detection System for Crop Protection,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 9, pp. 2921–2925, 2022.
46. S. Yadahalli, A. Parmar and A. Deshpande, "Smart Intrusion Detection System for Crop Protection by using Arduino," *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2020, pp. 405-408, doi: 10.1109/ICIRCA48905.2020.9182868.
47. V. Bavane, A. Raut, S. Sonune, A. P. Bawane, and P. M. Jawandhiya, “Protection of Crops from Wild Animals Using Intelligent Surveillance System,” *International Journal of Research in Advent Technology (IJRAT)*, Special Issue National Conference “CONVERGENCE 2018,” Apr. 9, 2018.
48. K.-R. Li, L.-J. Duan, Y.-J. Deng, J.-L. Liu, C.-F. Long, and X.-H. Zhu, “Pest Detection Based on Lightweight Locality-Aware Faster R-CNN,” *Agronomy*, vol. 14, no. 10, pp. 2303–2303, Oct. 2024, doi: https://doi.org/10.3390/agronomy14102303.
49. J. Miao, Dasari Rajasekhar, S. Mishra, S. K. Nayak, and R. Yadav, “A Microservice-Based Smart Agriculture System to Detect Animal Intrusion at the Edge,”

Future Internet, vol. 16, no. 8, pp. 296–296, Aug. 2024, doi: https://doi.org/10.3390/fi16080296.

50. E.O. Amuta, G.O. Sobola, O. Eseabasi, H. N. Dike, and S. T. Wara, “Motion Detection System Using Passive Infrared Technology,” *IOP Conference Series Earth and Environmental Science*, vol. 1342, no. 1, p. 012001, May 2024, doi: https://doi.org/10.1088/1755-1315/1342/1/012001.
51. I. Salehin et al., “IFSG: Intelligence agriculture crop-pest detection system using IoT automation system,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 24, no. 2, p. 1091, Nov. 2021, doi: https://doi.org/10.11591/ijeecs.v24.i2.pp1091-1099.
52. K. A. Steen, A. Villa-Henriksen, O. R. Therkildsen, and O. Green, “Automatic Detection of Animals in Mowing Operations Using Thermal Cameras,” *Sensors*, vol. 12, no. 6, pp. 7587–7597, Jun. 2012, doi: https://doi.org/10.3390/s120607587
53. S. Pedersen and C. B. Pedersen, “Animal Activity Measured by Infrared Detectors,” *Journal of Agricultural Engineering Research*, vol. 61, no. 4, pp. 239–246, May 2002, doi: https://doi.org/10.1006/jaer.1995.1051.
54. M. Djawdan and T. Garland, “Maximal Running Speeds of Bipedal and Quadrupedal Rodents,” *Journal of Mammalogy*, vol. 69, no. 4, pp. 765–772, Nov. 1988, doi: https://doi.org/10.2307/1381631.
55. R. J. Batka, T. J. Brown, K. P. Mcmillan, R. M. Meadows, K. J. Jones, and M. M. Haulcomb, “The Need for Speed in Rodent Locomotion Analyses,” *The Anatomical Record*, vol. 297, no. 10, pp. 1839–1864, Jun. 2014, doi: https://doi.org/10.1002/ar.22955.
56. Tai-Hsien Ou Yang, M. Tsai, C. Yen, and T. Lin, “An infrared range camera-based approach for three-dimensional locomotion tracking and pose reconstruction in a rodent,” *Journal of Neuroscience Methods*, vol. 201, no. 1, pp. 116–123, Sep. 2011, doi: https://doi.org/10.1016/j.jneumeth.2011.07.019.
57. P. A. Burns, M. L. Parrott, K. C. Rowe, and B. L. Phillips, “Identification of threatened rodent species using infrared and white-flash camera traps,” *Australian Mammalogy*, vol. 40, no. 2, p. 188, 2018, doi: https://doi.org/10.1071/am17016.

58. Z. K. Wu and T. Zeng, “Design of On-Off Control System for Electric Equipment Based on WiFi,” *Applied Mechanics and Materials*, vol. 536–537, pp. 1157–1160, Apr. 2014, doi: https://doi.org/10.4028/www.scientific.net/amm.536-537.1157.
59. J. Brinkhoff and J. Hornbuckle, "Characterization of WiFi signal range for agricultural WSNs," *2017 23rd Asia-Pacific Conference on Communications (APCC)*, Perth, WA, Australia, 2017, pp. 1-6, doi: 10.23919/APCC.2017.8304043.
60. Celil Serhan TEZCAN and Kemal Sulhi GÜNDOĞDU, “The Effect of Some Meteorological Parameters on Wireless Data Transmission,” *Kahramanmaraş Sütçü İmam Üniversitesi Tarım ve Doğa Dergisi*, vol. 25, no. 5, pp. 1127–1133, May 2022, doi: https://doi.org/10.18016/ksutarimdoga.vi.910496.
61. Vojtěch Novák, M. Stočes, E. Kánská, J. Pavlík, and J. Jarolímek, “Monitoring of Movement on the Farm Using WiFi Technology,” *Agris on-line Papers in Economics and Informatics*, vol. 11, no. 4, pp. 85–92, Dec. 2019, doi: https://doi.org/10.22004/ag.econ.303927.
62. T. Sakamoto *et al.*, “Application of day and night digital photographs for estimating maize biophysical characteristics,” *Precision Agriculture*, vol. 13, no. 3, pp. 285–301, Sep. 2011, doi: https://doi.org/10.1007/s11119-011-9246-1.
63. J. Kriesel and N. Gat, “True-color night vision cameras,” *Proceedings of SPIE, the International Society for Optical Engineering/Proceedings of SPIE*, vol. 6540, pp. 65400D65400D, Apr. 2007, doi: https://doi.org/10.1117/12.719902.
64. P. Guo, M. S. Asif and Z. Ma, "Low-Light Color Imaging via Cross-Camera Synthesis," *in IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 4, pp. 828-842, June 2022, doi: 10.1109/JSTSP.2022.3175015.
65. J. Ellison, “Shielding Your Camera: The Ultimate Guide to Weather Sealing - BlinksAndButtons,” *BlinksAndButtons*, Dec. 25, 2024. https://blinksandbuttons.net/how-to-weather-seal-camera/ (accessed Sep. 11, 2025).
66. Z. Xu, B. Zhang, W. Wang, L. Guo, H. Li and J. Zhang, "Lightweight Region Extraction in Small Animal Images Based on Salient Object Detection," *2024 6th International Academic Exchange Conference on Science and Technology Innovation (IAECST)*, Guangzhou, China, 2024, pp. 650-654, doi: 10.1109/IAECST64597.2024.11117387.
67. K. Shim, A. Barczak, N. Reyes and N. Ahmed, "Small mammals and bird detection using

IoT devices," *2021 36th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, Tauranga, New Zealand, 2021, pp. 1-6, doi: 10.1109/IVCNZ54163.2021.9653430.

68. Rajesh Kumar Dhanaraj, Md. Akkas. Ali, A. K. Sharma, and A. Nayyar, “Deep Multibranch Fusion Residual Network and IoT-based pest detection system using sound analytics in large agricultural field,” *Multimedia Tools and Applications*, vol. 83, no. 13, pp. 40215–40252, Oct. 2023, doi: https://doi.org/10.1007/s11042-023-16897-3.
69. P. Venkatasaichandrakanth and M. Iyapparaja, "A survey on pest detection and classification in field crops using artificial intelligence techniques," *International Journal of Intelligent Robotics and Applications*, vol. 8, (3), pp. 709-734, 2024. Available: https://login.proxy.lib.duke.edu/login?url=https://www.proquest.com/scholarly-journals/survey-on-pest-detection-classification-field/docview/3101841983/se-2. DOI: https://doi.org/10.1007/s41315-024-00347-w.
70. B. Guo, J. Wang, M. Guo, M. Chen, Y. Chen, and Y. Miao, “Overview of Pest Detection and Recognition Algorithms,” *Electronics*, vol. 13, no. 15, pp. 3008–3008, Jul. 2024, doi: https://doi.org/10.3390/electronics13153008.
71. R. Tian, M. Chen, D. Dong, W. Li, L. Jiao, Y. Wang, M. Li, C. Sun, and X. Yang, “Identification and counting method of orchard pests based on fusion method of infrared sensor and machine vision,” *Nongye Gongcheng Xuebao*, vol. 32, no. 20, pp. 195–201, Oct. 2016.
72. C. J. Cohen, D. Haanpaa, and J. P. Zott, “Machine vision algorithms for robust animal species identification,” *2015 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–7, Oct. 2015, doi: https://doi.org/10.1109/aipr.2015.7444526.
73. A. Yi and Nantheera Anantrasirichai, “A Comprehensive Study of Object Tracking in Low-Light Environments,” *Sensors*, vol. 24, no. 13, pp. 4359–4359, Jul. 2024, doi: https://doi.org/10.3390/s24134359.
74. R. Dufour, “Sustainable Agriculture How Organic Pest Management Works,” Jul. 2015, https://www.ams.usda.gov/sites/default/files/media/Organic%20Pest%20Management_FINAL.pdf
75. S. Raszap, A. Carlson, and A. Spalding, “Rising consumer demand reshapes landscape for U.S. organic farmers,” Rising Consumer Demand Reshapes Landscape for U.S.

Organic Farmers | Economic Research Service,
https://www.ers.usda.gov/amber-waves/2023/november/rising-consumer-demand-reshapes-landscape-for-u-s-organic-farmers (accessed Sep. 3, 2025).

76. J. Knight and J. Jenks, Wildlife Damage Control for Organic Farmers. Bozeman, MT: Montana State University Extension, 2015, https://animalrange.montana.edu/documents/extension/WILDLIFEDAMAGECONTROLFORORGANICFARMERS.pdf(accessed Sep. 3, 2025).
77. G. C. Clifford, “Plywood weight calculator,” CTRL Calculator, https://ctrlcalculator.com/construction/plywood-weight-calculator/ (accessed Dec. 5, 2025).
78. Density of wood species: Data & Material Guide, https://www.engineeringtoolbox.com/wood-density-d_40.html (accessed Dec. 5, 2025).
79. Filamatrix, “Pla 3d printer filament 10 pound,” Filamatrix, https://filamatrix.com/products/pla-10lb?variant=47742511481144&country=US&currency=USD&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gad_source=1&gad_campaignid=23254963892&gbraid=0AAAAAClwVnVUgakYt1nQVYsSRbpRvXUDA&gclid=Cj0KCQiAosrJBhD0ARIsAHebCNrFqm53591z6195WwKukN6haDE6WHAdWEC-4Zfrnm_bwg569fvallMaAk_VEALw_wcB (accessed Dec. 5, 2025).
80. Sua-A-B bel power solutions | power supplies - external/internal (off-board) | DigiKey, https://www.digikey.com/en/products/detail/bel-power-solutions/SUA-A-B/25578429?gclsrc=aw.ds&gad_source=1&gad_campaignid=20232005509&gbraid=0AAAAADrbLlhKICeRgDNFs6KZOJOLjb_BJ&gclid=Cj0KCQiAosrJBhD0ARIsAHebCNpPtIcmvmRQvSsMtmfdZLZ-u7H1mQxaY9v9bajA_oHu6YpxF33PUzAaAksUEALw_wcB (accessed Dec. 5, 2025).
81. “Qualtek 3021059-01M,” DigiKey, https://www.digikey.com/en/products/detail/qualtek/3021059-01M/7795309?gclsrc=aw.ds&gad_source=1&gad_campaignid=20232005509&gbraid=0AAAAADrbLlhKICeRgDNFs6KZOJOLjb_BJ&gclid=Cj0KCQiAosrJBhD0ARIsAHebCNrvLA1HeB4rDVnkVN5YAEGWCPV7_JtplD6z1sGQmYBCqiiQ2xsVbzwaAtbpEALw_wcB (accessed Dec. 5, 2025).

82. “Pir motion sensor module,” elecrow.openhardware, https://www.elecrow.com/pir-motion-sensor-module-p-315.html?gad_source=1&gad_campaignid=23136944328&gbraid=0AAAADph4ppih2b-y5-LmPBHP1ohwnLhW&gclid=Cj0KCQiAosrJBhD0ARIsAHebCNrFAiK1X9eafEeuzJVGzz-m1qj2KTQlukgMxrN8EXlgC4GLor0RNtgaAuqdEALw_wcB (accessed Dec. 5, 2025).
83. Multi Camera Adapter Module V2.2 for Rasperry Pi Camera module ..., https://www.arducam.com/multi-camera-v2-1-adapter-raspberry-pi.html (accessed Dec. 5, 2025).
84. Arducam | DigiKey, https://www.digikey.com/en/products/detail/arducam/B0310/19116507?gclsrc=aw.ds&gad_source=1&gad_campaignid=20232005509&gbraid=0AAAADrbLlhKICeRgDNFs6KZOJOLjb_BJ&gclid=Cj0KCQiAosrJBhD0ARIsAHebCNoej0pKrh_-1kr0UqfmDH_Ur5DO8LzupBvEMitcl2V-Pn5HJ3q3ynYaApaCEALw_wcB (accessed Dec. 5, 2025).
85. SC0192(9) Raspberry Pi | Embedded Computers | DigiKey, https://www.digikey.com/en/products/detail/raspberry-pi/SC0192(9)/18634938?gclsrc=aw.ds&gad_source=1&gad_campaignid=20228387720&gbraid=0AAAADrbLlgXsFbsfma99EMgt1hjWlxG0&gclid=Cj0KCQiAosrJBhD0ARIsAHebCNqkDCZej7WpWIXaCg5aYSpejjWZz490IGUuVQghgow47ERYVbC3slMaAmIeEALw_wcB (accessed Dec. 5, 2025).
86. “Plywood,” Home Depot, https://www.homedepot.com/b/Lumber-Composites-Plywood/1-4/N-5yc1vZbqm7Z1z0mcqp?utm_ (accessed Dec. 5, 2025).