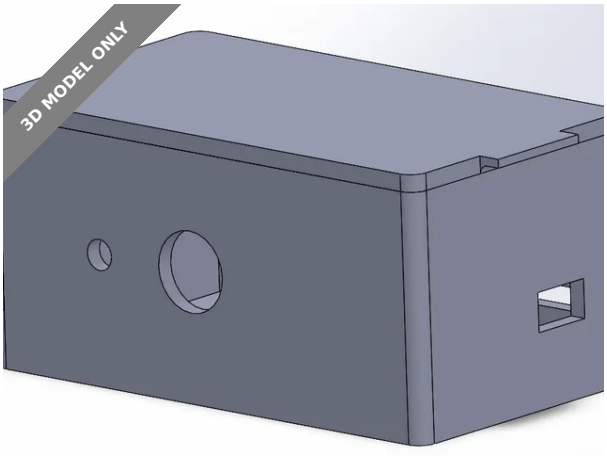


Pet Safety Zone Monitor

Juven Cardiz

VIEW IN BROWSER

updated 4. 4. 2025 | published 4. 4. 2025

Summary

Detects dogs or cats using AI with YOLO, Raspberry Pi, and ESP32 to trigger LEDs and buzzer based on the pet.

3D Printers > Test Models

Description

This project is a smart pet monitoring system designed to detect the presence of dogs or cats in specific areas using artificial intelligence. Detection is performed in real time through a webcam connected to a Raspberry Pi running the YOLO model. When a pet is detected, the Raspberry Pi sends the data to Firebase, and a Flask-based intermediary server relays it to an ESP32.

The ESP32 then activates a visual signal (LED) and an audio signal (buzzer) depending on the detected animal. Detection events are also logged on ThingSpeak for historical tracking, which is displayed through a web interface. This system combines computer vision, microcontrollers, and IoT, all integrated into a 3D-printed case, offering an efficient, affordable, and easy-to-implement solution for monitoring and protecting pets in hazardous or restricted zones.

Functions

- The main functions of this project include:
- Real-time pet detection (dog or cat) using a webcam and YOLO model running on a Raspberry Pi.
- Sending detection data to Firebase from the Raspberry Pi.
- A Flask-based intermediary server that fetches data from Firebase and forwards it to the ESP32.
- The ESP32 receives the detection results and responds by triggering a buzzer and lighting an LED with different colors depending on the detected animal (blue for dogs, red for cats).
- All detection events are logged to ThingSpeak for data visualization and monitoring.
- A web interface displays the historical presence of cats and dogs in the monitored space, including images and graphs.
- Simple integration of all components using 3D-printed enclosures for housing the electronics.

Hardware and Software Used

- 1x Raspberry Pi 4 (with power supply and microSD card)
- 1x USB webcam (compatible with Raspberry Pi)
- 1x ESP32 (Wi-Fi module)
- 1x RGB LED (common cathode or anode)
- 3x Resistors (220Ω for RGB LED channels)
- 1x Buzzer (piezoelectric)
- 1x Breadboard (or custom PCB)
- Jumper wires
- 1x 3D-printed enclosure (base and top with ventilation and ports)

Software:

- **Raspberry Pi OS** – Operating system for the Raspberry Pi
- **Python** – Used for computer vision and communication scripts
- **OpenCV** – For image capture and processing
- **YOLOv5 or YOLOv4-tiny** – Object detection model
- **Firebase Realtime Database** – For cloud-based data exchange
- **Flask** – Python web framework used for the intermediary server
- **Arduino IDE** – To program the ESP32 microcontroller
- **ThingSpeak** – For data logging and visualization
- **CATIA** – 3D modeling of the enclosure and support parts
- **Cura / PrusaSlicer** – Slicing software to prepare 3D models for printing
- **Web Browser** – To view the web interface and ThingSpeak dashboard

Wiring Diagram

This diagram shows the connection between the ESP32, an RGB LED, and a buzzer:

RGB LED:

Red pin → GPIO 25 (with a 220Ω resistor)

Green pin → GPIO 26 (with a 220Ω resistor)

Blue pin → GPIO 27 (with a 220Ω resistor)

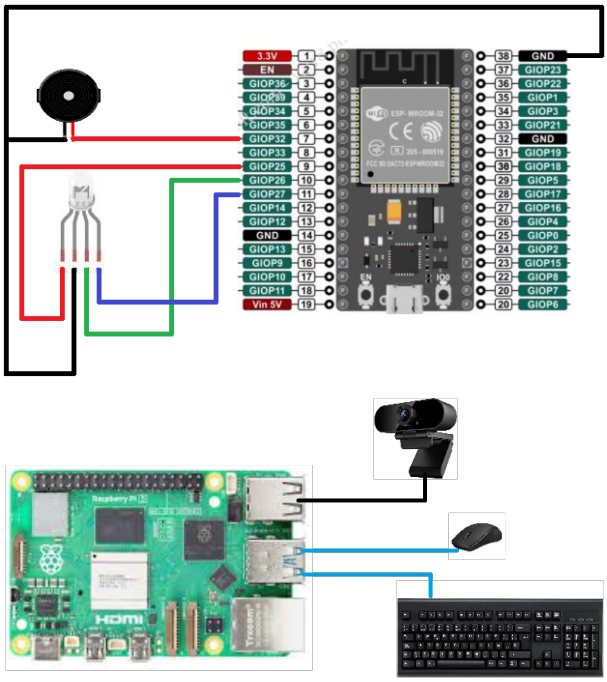
Common cathode → GND

Buzzer:

Positive pin → GPIO 14

Negative pin → GND

The RGB LED lights up in different colors depending on the animal detected, and the buzzer emits a specific tone.
Additionally, a USB webcam is connected to the Raspberry Pi, which processes the video input and performs real-time detection using the YOLO model.



Code

- The Arduino code running on the ESP32 is structured in the following way:

- **Wi-Fi Setup and Server Connection**
  In the setup() function, the ESP32 connects to a Wi-Fi network using WiFi.begin(ssid, password). Then, it starts a connection to the Flask server (intermediary) which receives data from Firebase.
- **Listening for Detection Events**
  Inside the loop(), the ESP32 periodically sends a GET request to the Flask server using the HTTPClient library. It checks the response to see whether a **dog** or a **cat** was detected.
- **Controlling the RGB LED**
  Based on the detection result:
- If a **dog** is detected → set the RGB LED to **blue** (e.g., digitalWrite(bluePin, HIGH)).
- If a **cat** is detected → set the RGB LED to **red** (digitalWrite(redPin, HIGH)).
- If nothing is detected → turn off the LED (digitalWrite(...) LOW on all channels).
- **Activating the Buzzer**
  The buzzer is also controlled depending on the detected animal:
- For **dog**, it plays a specific tone (e.g., 500 Hz for 500 ms).
- For **cat**, it plays a different tone (e.g., 1000 Hz for 500 ms). The tone() function is used for this purpose.
- **Delay and Loop**
  After each detection cycle, the ESP32 waits a few seconds before checking again. This avoids flooding the server and keeps buzzer/LED signals consistent.

Printing Instructions

Material: PLA or PETG

Layer Height: 0.2 mm

Infill: 30%

Supports: Yes — enable support generation, especially under the servo shaft mount area

Adhesion: Recommended — use a brim or raft to ensure proper bed adhesion for the base part

Dimensions:
The enclosure base measures **94.4 mm x 53 mm**, with a height of **46 mm**.

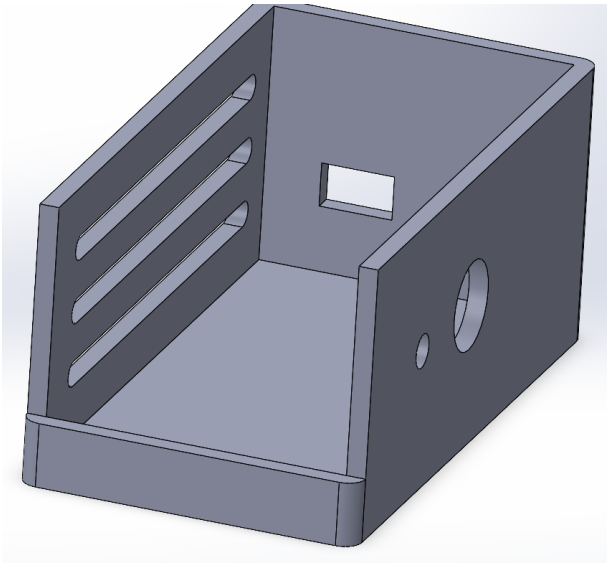
Hole Sizes:

The circular hole for the RGB LED has a diameter of **5.3 mm**

The larger hole for the buzzer is **14.4 mm** in diameter

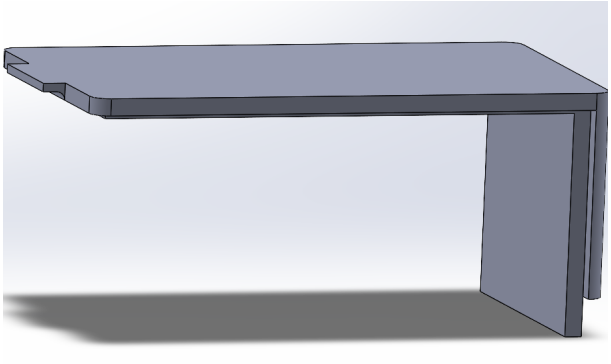
Tips:
Ensure that your slicer settings are configured for small overhangs and tight tolerances, especially around the LED and buzzer ports. Printing with a slower speed on the outer walls will improve accuracy for component fitting.

Assembly Instructions

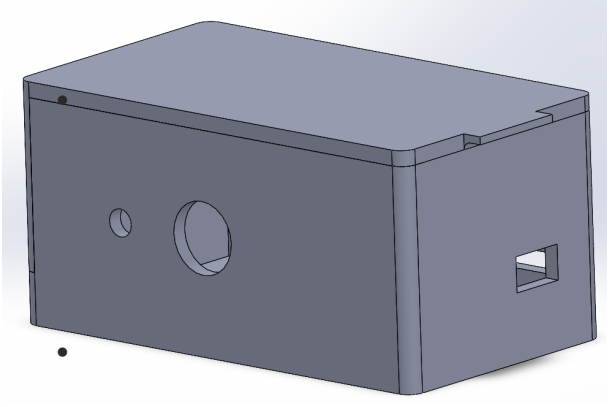


Base and Main Body (Image 1):
Place the main body upright. This is the base enclosure that includes ventilation slots and openings for the LED, buzzer, and USB/power connections.

Top Lid (Image 2):
Slide the top lid into the slot grooves on the sides of the main body. Make sure the notches align correctly with the body for a secure fit.



Final Enclosure (Image 3):
Once the top is in place and fits flush with the rest of the box, verify that all external ports (holes) are aligned with your components (e.g., buzzer, LED, USB cable).
Your enclosure is now assembled and ready for electronics installation.



Demo and Usage Instructions

Power Up the System:
Connect the Raspberry Pi and ESP32 to their power supplies. Ensure the webcam is properly connected to the Raspberry Pi via USB.
Detection Process:
The Raspberry Pi runs the YOLO

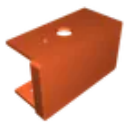
object detection model and continuously analyzes the video stream. When it detects a **dog** or **cat**, it sends the result to Firebase.

- **Data Transmission:**
  A Flask server reads the detection result from Firebase and forwards it to the ESP32 via HTTP.
- **ESP32 Response:**
  Based on the received data:
- If a **dog** is detected → The RGB LED lights up **blue** and a specific tone is played on the buzzer.
- If a **cat** is detected → The RGB LED lights up **red** and a different tone is played.
- **Monitoring Interface:**
  You can view detection history (date, time, and type of pet) on the **ThingSpeak** dashboard, along with images and graphs.
- **Enclosure Usage:**
  Ensure all components are placed correctly inside the 3D-printed enclosure. The LED and buzzer should align with the designated front-facing holes for proper visibility and sound.

Additional Notes or Tips

- Place the camera where pets cannot reach it, but with a clear and stable view of the monitored area.
- Ensure proper alignment between the servo motor and the door for precise and smooth movement.
- Consider using a rechargeable battery to make the system portable and independent from wall power.
- Align the RGB LED and buzzer accurately with the 3D-printed enclosure openings for better visibility and sound output.
- Test the full system with real pets to fine-tune sensor response, servo timing, and detection reliability.

Model files



caja.stl

tapa.stl

Other files

box.zip

License

This work is licensed under a
Creative Commons (International License)

Public Domain

✔ | Sharing without ATTRIBUTION
✔ | Remix Culture allowed
✔ | Commercial Use
✔ | Free Cultural Works
✔ | Meets Open Definition