ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Actions to Mosquitoes

Team #4

XIANGMEI CHEN (xc47@illinois.edu)
PEIQI CAI (peiqic3@illinois.edu)
YANG DAI (yangdai2@illinois.edu)
LUMENG XU (lumengx2@illinois.edu)

TA: Guo Hao Thng
Sponsor: Said Mikki

May 28, 2024

Abstract

This project presents an automated machine designed to detect, localize, and capture mosquitoes. The detection subsystem leverages audio analysis to identify mosquito-specific wingbeat frequencies using MFCC and CNN techniques. It is complemented by a localization subsystem with a high-resolution camera and advanced imaging model for real-time mosquito tracking. The attack subsystem employs a fan unit and a mechanical structure for precise capture, while the power and control subsystem, managed by a Raspberry Pi, oversees sensory data processing and motor control. The machine's design is ethically grounded, emphasizing privacy, humane mosquito elimination, and strict safety protocols.

Contents

1 Introduction

1.1 Problem and Solution Overview

Mosquitoes are not just a source of irritation due to their itchy bites; they are also public health threats, as documented by the World Health Organization (WHO), which identifies them as vectors for diseases like malaria and dengue [1]. The challenge of controlling these agile insects is compounded by the limitations of current methods, which can be less effective and potentially harmful, as noted in studies on the environmental impact of mosquito control. To address these issues, we've developed an innovative device that actively positions and captures mosquitoes. It operates by moving through the environment and sucking up mosquitoes upon detection, offering a more targeted and safer alternative to traditional repellents and swatters.

We design our project by four subsystems: a detection subsystem, a localization subsystem, an attack subsystem, and a power and control subsystem. The detection subsystem serves as the trigger, using audio cues to activate the machine when mosquitoes are present. The localization subsystem employs a camera to locate the mosquito and provides real time location data to the attack subsystem, which then mobilizes to capture the mosquitoes using a powerful fan. The power and control subsystem is strategically divided to supply continuous energy to the detection subsystem and activated power to the localization and attack subsystems, optimizing energy usage, and ensuring sustained operations.

1.2 Functionality

1.2.1 Detection Subsystem

The detection subsystem is designed to identify the presence of mosquitoes through their unique wingbeat frequency. This subsystem operates using a microphone that captures and processes sound waves within the 300 to 600 Hz range, which is specific to mosquitoes, leveraging the Mel-Frequency Cepstral Coefficient (MFCC), spectrogram and a machine learning model [2].

By accurately identifying mosquito presence, the device can operate efficiently, triggers the localization subsystem for the camera to work once the mosquito is detected.

1.2.2 Localization Subsystem

Localization subsystem is responsible for visually localizing the mosquito's position via a high-resolution USB camera mounted on the Raspberry Pi. The Roboflow Train 3.0 model [3], one of the best algorithms for real-time object detection is used by this subsystem, which gives the mosquito's position in the camera's field view.

The localization subsystem's precision is essential for the device's ability to navigate towards and capture mosquitoes. By accurately determining the mosquito's location, the device can go and suck the mosquito using attack subsystem.

1.2.3 Attack Subsystem

The attack subsystem is tasked with the physical capture of mosquitoes. It is equipped with a fan capture unit that generates an airflow to draw in mosquitoes and a mechanical structure that allows for precise positioning. The subsystem's mechanical structure, including the chassis, wheels, and lead screw, enables 360-degree rotation and accurate movement towards the mosquito's location.

The attack subsystem's efficiency and precision are paramount to the device's eradication capabilities. By receives signals from control subsystem, the subsystem effectively capturing mosquitoes.

1.2.4 Power and Control Subsystem

The power and control subsystem is the central part of the device, ensuring that all other subsystems function harmoniously. It comprises a control unit based on the Raspberry Pi, which processes sensory data and formulates commands for the motor control system. The motor control system, in turn, generates PWM signals to manage the motors, enabling the precise movement and positioning of the device.

The power and control subsystem is indispensable for the device's autonomous functionality. It ensures the power supply unit provide a stable and regulated power source. This subsystem's stability and efficiency are foundational to the project's success, powering and controlling all other subsystems.

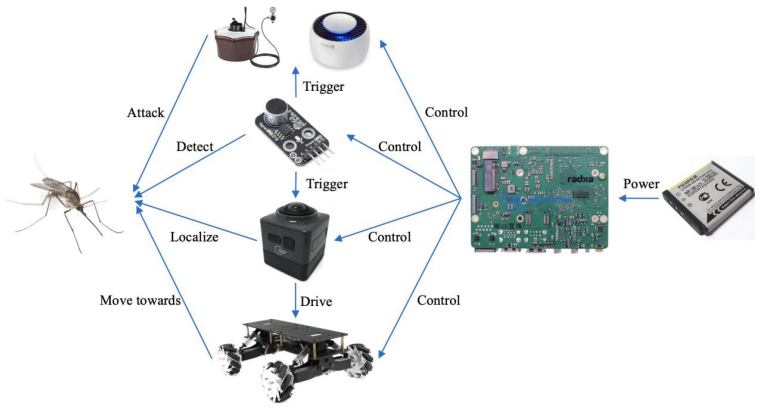


Figure 1: The overall visual graph of the design: One motor chip powers and controls all other parts, the microphone works as a trigger to enable the camera and the attacker, then the main part starts to localize and move to attack the mosquitoes.

1.3 Subsystem Overview

This machine is designed to detect, locate, and attack mosquitoes autonomously. It comprises four main subsystems, each playing a crucial role in the machine's operation and interacting seamlessly. See Figure1 for the overall visual graph of the design.

1.3.1 Detection Subsystem

The detection subsystem identifies mosquitoes through their characteristic wingbeat frequencies between 300 to 600 Hz. It uses a specialized microphone and the MFCC technique to process audio signals, also CNN is applied to train the data, enabling the system to efficiently detect and respond to mosquito presence. Figure2 is The frequency plot of some mosquitoes wingbeats noise.

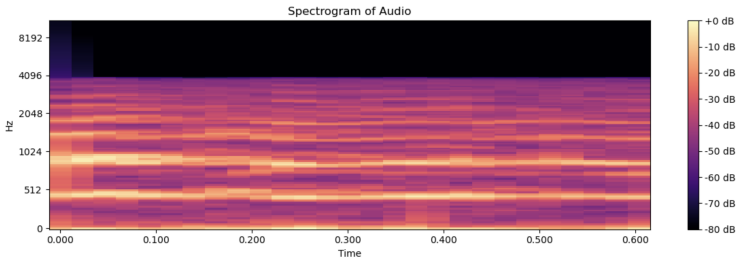


Figure 2: The frequency plot of some mosquitoes wingbeats noise.

Once a mosquito is detected, the detection subsystem activates the localization subsystem, initiating the camera for mosquito tracking.

1.3.2 Localization Subsystem

To aid our mosquito attack unit, we have developed the localization subsystem which is capable of detecting and localizing the mosquito visually. A high resolution USB camera is connected to a Raspberry Pi to capture images. Using the state-of-the-art Roboflow Train 3.0 model, the pipeline performs real-time object detection and is able to precisely localize the mosquito within the camera view.

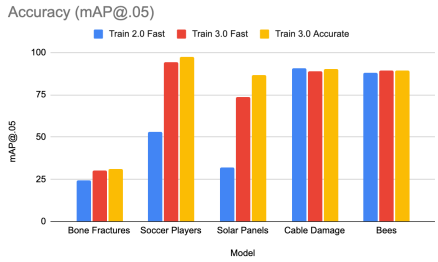


Figure 3: Accuracy of Roboflow 3.0.

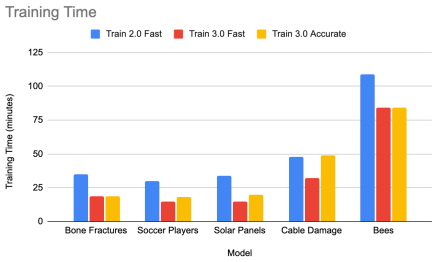


Figure 4: Training time of Roboflow 3.0.

The localization sub-system is always enabled and ready to provide the coordinates of the detected mosquito to the attack subsystem.

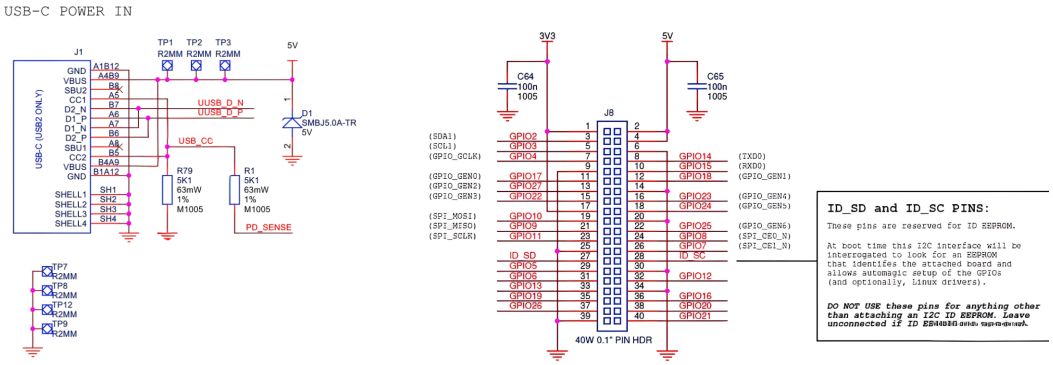
1.3.3 Attack Subsystem

The attack subsystem, built around a Raspberry Pi control unit, integrates inputs from a microphone and camera to manage a motor system for precise mosquito capture. It

features a fan for capturing mosquitoes and a mechanical structure that allows for 360-degree rotation and movement. This subsystem works with the detection and localization subsystem to receive real-time data, guiding it to the mosquito's location. The control unit processes this data and uses PWM signals to keep the mosquito in view for capture.

1.3.4 Power and Control Subsystem

The power and control subsystem is the heart of our machine, with a Raspberry Pi control unit that processes real-time data to distinguish mosquito sounds and identify their positions. It commands the motor control system to generate PWM signals for precise movement and positioning of components like the camera and capture mechanism. The Power Supply Unit (PSU) ensures stable 12V power and converts it for all components. This subsystem coordinates with others, receiving input for mosquito capture and managing a seamless flow of commands and power for efficient operation.



(a) USB circuit of Raspberry Pi.

(b) GPIO Pins expansion of Raspberry Pi.

Figure 5: Circuits of USB and GPIO

2 Design

2.1 Audio Detection Module

2.1.1 Design Ideas and Algorithm

The detection subsystem forms the first line of our mosquito attack device, leveraging acoustic data to detect the presence of mosquitoes. Utilizing an microphone to record sounds, the subsystem captures audio signals within a specific frequency range known to be characteristic of mosquito wingbeats, which typically lie between 300Hz to 600 Hz.

The detection process starts with audio signal capture via the microphone. The algorithm processes these audio inputs to extract relevant features that help differentiate mosquito noises from other ambient sounds. The primary feature extraction method used here is Mel-Frequency Cepstral Coefficients (MFCC). MFCCs are crucial in this context as they efficiently represent the power spectrum of audio signals, capturing the essential charac-

teristics needed for mosquito identification. Its calculation formula is as follows:

$$\text{MFCCs} = 20 \cdot \log_{10}\left(|\text{FFT}\left(\text{Window} \cdot \text{Signal}\right)|^2\right)$$

The extracted MFCC features are then utilized to determine the presence of mosquitoes through a classification process. While the specifics of the model used for classification are detailed in a subsequent section, it's important to note that the chosen model processes these features to accurately identify mosquito-related audio.

2.1.2 Neural Network Description

The detection subsystem utilizes a Convolutional Neural Network (CNN) to classify audio features extracted as Mel-Frequency Cepstral Coefficients (MFCCs), distinguishing mosquito sounds from other ambient noises.

The CNN architecture comprises:

- **Input Layer:** Processes input MFCCs, a time-series representation of audio.
- **Convolutional Layers:** Multiple layers with ReLU activation functions extract patterns from the audio data.
- **Pooling Layers:** Max pooling layers follow convolutional layers to reduce dimensionality and prevent overfitting.
- **Fully Connected Layers:** One or more layers that finalize the classification process.
- **Output Layer:** A softmax activation function provides the probabilities for each class, enabling a clear classification decision.

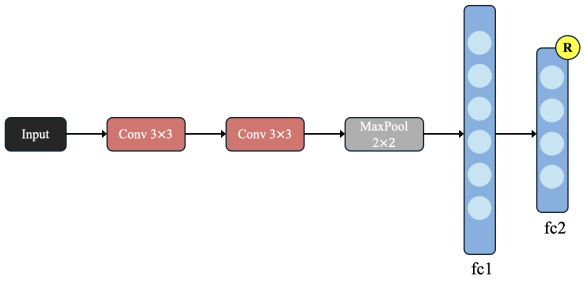


Figure 6: The structure of the CNN.

The CNN is trained using a dataset of labeled mosquito and non-mosquito sounds, adjusting weights and biases through backpropagation to minimize cross-entropy loss. Optimization is performed using algorithms like Adam. The mathematical expression of the calculation of the CNN can be expressed as:

$$y = \sigma\left(W_2 \cdot \text{ReLU}\left(W_1 \cdot x + b_1\right) + b_2\right)$$

Where:

- x is the input vector (MFCCs).
- W_1, W_2 are the weights of the first and second layers.
- b_1, b_2 are biases for the first and second layers.
- ReLU is the Rectified Linear Unit activation function.
- σ is the softmax function applied at the output layer.

2.2 Computer Vision Module

2.2.1 Design Ideas

The vision subsystem is a pivotal element of our mosquito attack device, employing the Roboflow 3.0 model for enhanced object detection capabilities, which is trained on a dataset of annotated images, where it learns to recognize and localize mosquitoes within the frames. The process initiates with the camera capturing video frames, which are then fed into the Roboflow 3.0 model. The model utilizes advanced algorithms to process the video inputs and extract features that are instrumental in distinguishing mosquitoes from their surroundings[4]. The Roboflow 3.0 model is particularly adept at handling complex visual patterns and providing high accuracy rates, thanks to its improved training infrastructure1.

2.2.2 Data Augmentation

The input dataset of the model during training will be the images of 640×640 pixels, which will be further augmented based on several strategies to improve training performance shown as below.

1. 90° Rotate: This step add 90-degree rotations to help the model be insensitive to camera orientation. As stated, the environment of the dataset is set to be constant, while the real situation could be different since our design requires the whole structure to rotate and move to capture the mosquito. Hence, the step could improve the robustness of the model.

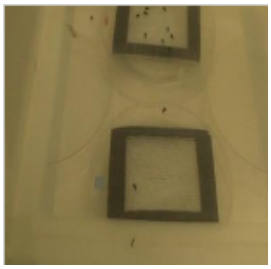


Figure 7: Preprocessed.

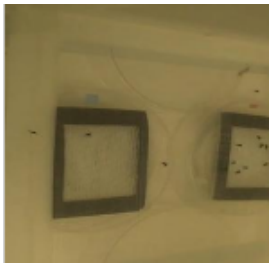


Figure 8: Clockwise.

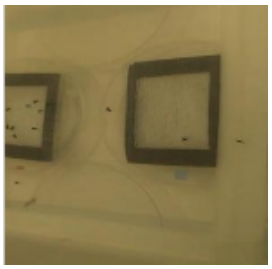


Figure 9: Counter-clockwise.

2. Brightness: This step add variability to image brightness to help the model be more

resilient to lighting and camera setting changes, since the time as well as environment may affect the brightness of the captured image. Both brighten and darken images are considered and brightness is set to -15% to 15%.

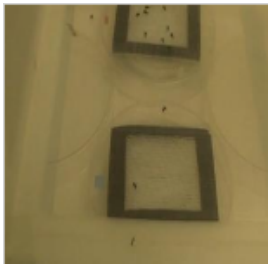


Figure 10: 0%.

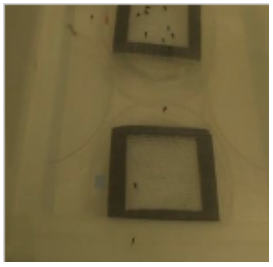


Figure 11: -15%.

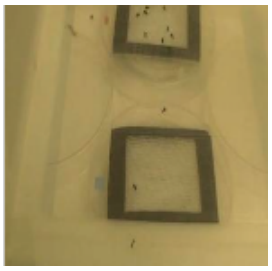


Figure 12: 15%.

After data augmentation, the amount of the dataset is doubled to 2258 in total, and is further split to the fraction: $85\%, 10\%, 5\%$, as the train-valid-test set.

2.2.3 Inference Acceleration

Design issue:

One of the greatest challenging we are facing in Localization subsystem is the time taken for processing each frame was higher than desired, leading to delays in mosquito detection. This was quantified by the latency equation.

$$L = \frac{1}{N} \sum_{i=1}^{N} (t_{\text{response},i} - t_{\text{request},i})$$

where L is the average latency, N is the number of requests, and $t_{\text{response},i}$ and $t_{\text{request},i}$ are the response and request times for the i-th inference. Due to the performance constraint of Raspberry Pi 4B, object detection models with large number of parameters such as YOLOv8[5], which we desired to use at the beginning, are abandoned.

Corrective Actions Taken:

The first measurement is to adopt model with fewer parameters, which we have mentioned above as Roboflow Train 3.0, which has faster speed in training and inference.

Another way is to use inference server service using Python package `inference` and set up our own Self-Hosted Inference Server. The basic principle diagram of inference server is illustrated as below.

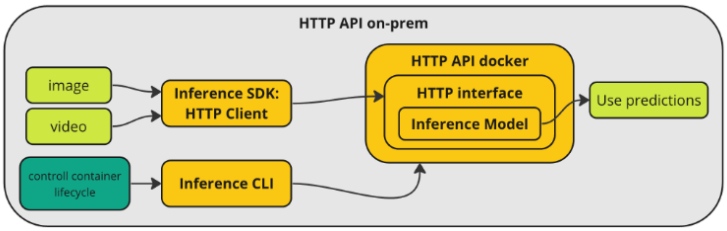


Figure 13: The diagram of using inference over HTTP.

Our linux server works on a Ubuntu 22.04 server with 64bit and a dual-core CPU of 2 threads and is set up in Shanghai, so that there will not be too much latency due to internet connection. The overall latency can be quantified as:

$$L_{\text{server}} = \frac{1}{N}\sum_{i=1}^{N}(t_{\text{response\_server},i} - t_{\text{request\_server},i} + \text{Internet latency})$$

Since the direct distance from ZJUI to Shanghai is within 100 km, the internet latency can be calculated as

$$L_{\text{internet}} = \frac{s}{v} = \frac{100\text{km}}{300,000\text{km/second}} \approx 0.4\text{ms}$$

.

Although, according to the real situation the latency may not be ideal, it's still acceptable comparing to deploy the model locally on Raspberry Pi.

2.3 Movement and Rotation Module

The Movement and Rotation Module is pivotal to our mosquito detection and elimination system, ensuring precise maneuvering for effective mosquito tracking and capture. It integrates the Roboflow object detection and custom bounding box detection to accurately pinpoint mosquito locations in images, guiding the system's movements. Utilizing PWM signals, it dynamically adjusts motor speeds for optimal approach patterns, controlling three omnidirectional wheels for advanced maneuvers in complex environments.

2.3.1 Configuration Space Calculation

The primary goal is to keep the mosquito centered within the camera's view, facilitating effective tracking and eventual capture.

Design Procedure:

In the mosquito tracking system, the configuration space is defined by the camera's field of view, which measures 640×480 pixels. The motor adjustments are determined based on the detected position of the mosquito within this field:

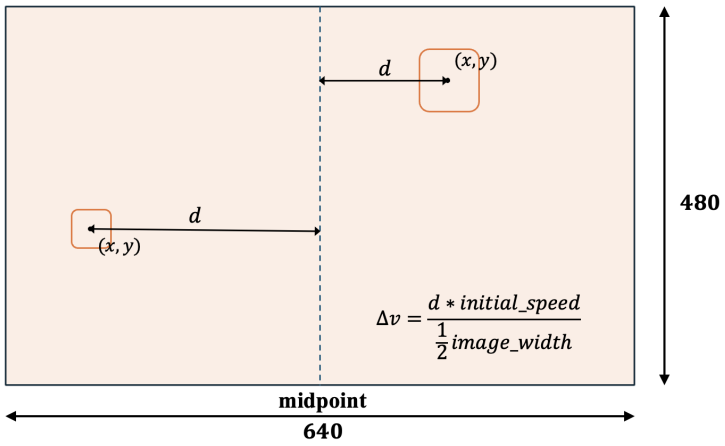


Figure 14: Visual Detection Boundary of Camera View.

- The mosquito detection system utilizes a Raspberry Pi-connected USB camera to capture live video, which is then processed by the high-precision Roboflow model to identify and locate mosquitoes. The system extracts the mosquito's position through bounding box coordinates, calculating the necessary motor speed adjustments to center the mosquito in the camera's field of view for enhanced tracking and positioning.

Design Details:

The operational logic is as follows:

Mosquito Position Analysis

- **Center Extraction:** The center (x, y) of the bounding box is used to determine the mosquito's current position.

```
    x, y, w, h = int(prediction['x']), int(prediction['y']),
            int(prediction['width']), int(prediction['height'])
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.putText(frame, f"{'mosquitoes'} {prediction['confidence']:.2f}",
            (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)
```

Listing 1: Bounding Box Coordinates.

- **Relative Position Calculation:** Calculates how far x is from the image's midpoint to determine necessary motor adjustments.

Motor Speed Adjustment

- **Adjustment Factor Calculation:**

$$\text{Speed Adjustment} = \text{Specific Speed} - \left(\frac{|x_{\text{position}} - x_{\text{mid}}|}{x_{\text{mid}}} \times \text{Specific Speed} \right)$$

Where x_{mid} is the midpoint of the image width, and x_{position} is the horizontal position of the mosquito. This notation clearly defines the adjustment needed based on the mosquito's position relative to the center of the camera's field of view.

- **Motor Control:**
  - If $x_{\text{position}} <$ mid_point, decrease left motor speed and increase right motor speed to turn left.
  - If $x_{\text{position}} >$ mid_point, increase left motor speed and decrease right motor speed to turn right.
- **Speed Application:** Adjustments to the PWM signals are dynamically applied to control motor speeds.

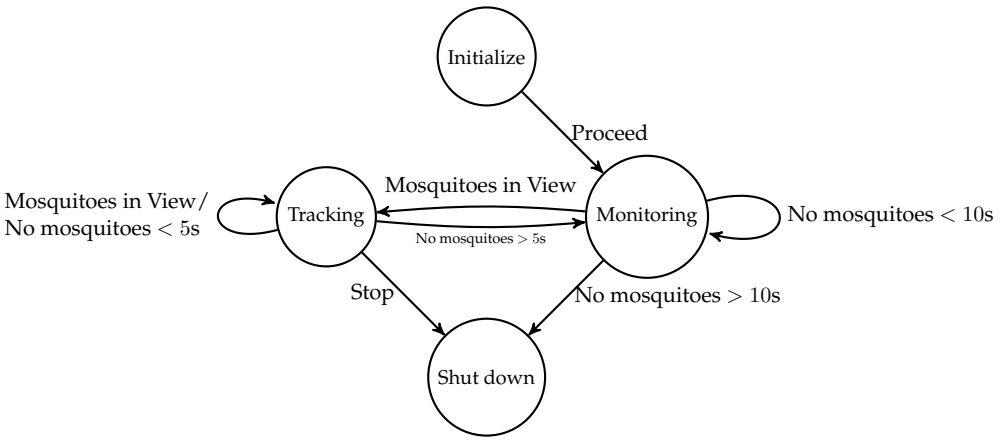
Real-Time Feedback Loop

- **Continuous Monitoring and Adjustment:** The device continuously adjusts based on real-time video and mosquito movement, enhancing tracking accuracy.

This balance between speed and precision ensures effective tracking and response to mosquito movements within the visual field.

2.3.2 Finite State Machine Design

The different states of the machine dealing with different situations can be expressed as a Finite State Machine.



2.3.3 PWM Design

Overview:

In the Movement and Rotation Module, Pulse Width Modulation (PWM) is utilized to control the speed and direction of motors based on the detected position of mosquitoes. By varying the duty cycle of the PWM signals sent to each motor, the system dynamically controls the device's movement, achieving precise positioning for optimal tracking.

Design Procedure:

The duty cycle in PWM is controlled by adjusting the duration of the high signal within each pulse relative to the total pulse duration. A higher duty cycle increases the motor speed, enabling quicker turns and faster reaction to mosquito movements.

Design Details:

Calculation of PWM Duty Cycle

The duty cycle for each motor is adjusted based on the deviation of the mosquito from the center of the camera's view. The formula used is:

$$\text{Duty Cycle}(\%) = \left(\frac{\text{Specific Speed} - \text{Adjustment Factor} \times \text{Specific Speed}}{\text{Maximum Speed}} \right) \times 100$$

where:

- **Specific Speed** is the speed required based on the mosquito's movement.
- **Adjustment Factor** is calculated from the positional deviation:

$$\text{Adjustment Factor} = \frac{|x_{\text{mid}} - x_{\text{position}}|}{\frac{1}{2} \times \text{Image Width}}$$

Sending PWM to Motors

Once the duty cycle is determined, it is applied to the motors using:

```
GPIO.PWM(pin, frequency).ChangeDutyCycle(duty_cycle)
```

This command controls the motor speed by adjusting the PWM duty cycle. The precise control of PWM signals to the motors ensures effective tracking of mosquitoes. This capability allows the device to align perfectly with the mosquito's position, optimizing the tracking and actions taken by the system.

2.4 Control and Integrated Module

2.4.1 Deployment of the audio detection to Raspberry Pi

The deployment of the neural network model to a Raspberry Pi involves transferring the pre-trained model and setting up real-time audio processing. The following steps and corresponding code illustrate how this is achieved:

The pre-trained model is transferred to the Raspberry Pi using PyTorch's functionality. The weights are saved in a `.pth` or `.pt` file and loaded on the Raspberry Pi as follows:

- `torch.load`: Loads the model, ensuring compatibility with the CPU environment of the Raspberry Pi.
- `model.eval()`: Sets the model to evaluation mode, which is necessary for inference as it disables training-specific layers like dropout.

Audio data is processed in real time, and the extracted features are used for classification. The function below handles the real-time prediction:

- `load_and_extract_features`: Extracts MFCC features from the audio file.

- `torch.no_grad()`: A context manager that disables gradient computation to speed up predictions and reduce memory usage.
- `torch.max(outputs.data, 1)`: Determines the predicted class by identifying the class with the highest probability.

The following equation models the real-time feature extraction and prediction pipeline, which is crucial for deployment on the Raspberry Pi.

$$\hat{y} = f_{\text{CNN}}(f_{\text{MFCC}}(\text{audio\_signal}))$$

Where:

- f_{MFCC} is the function extracting MFCC features from the audio signal.
- f_{CNN} is the CNN model function for prediction.
- audio_signal is the input audio data.
- $\hat{y}$ is the predicted output.

These steps encapsulate the process of deploying a neural network model on a Raspberry Pi, emphasizing efficient model loading and real-time processing for mosquito detection.

2.4.2 Deployment object detection model to Raspberry Pi

Adaptive Frame Processing

The system dynamically calculates and displays the frames per second (FPS)[6], which not only provides a real-time performance metric but also allows for adaptive adjustments. For instance, the processing detail or frequency could be scaled based on the current FPS, thus maintaining a balance between speed and accuracy, is essential for varied real-world scenarios.

```
while ret:
    counter += 1
    if (time.time() - start_time) != 0:
        cv2.putText(frame, "FPS {0}".format(float('%.1f' % (counter / (time.
    time() - start_time)))), (30, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
    255), 2)
        ret, frame = capture.read()
        print("FPS: ", counter / (time.time() - start_time))
        counter = 0
        start_time = time.time()
```

Listing 2: Video Capture and Frame Processing

Real-time Interactivity and Feedback Loop

The system is designed to provide immediate visual feedback through an annotated video stream, which is essential for user interaction and for tasks requiring instant decision-making.

```python
for prediction in srcimg.json()['predictions']:
    x, y, w, h = int(prediction['x']), int(prediction['y']), int(prediction['
    width']), int(prediction['height'])
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.putText(frame, f"{'mosquitoes'} {prediction['confidence']:.2f}",
                (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)
cv2.imshow("video", frame)
```

Listing 3: Model Prediction and Annotation

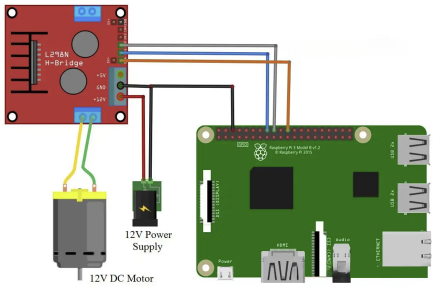
The 'predict' method of the local model is called with each frame, demonstrating how real-time detection is implemented. The system annotates detected objects in the video stream and displays these annotations in real time, highlighting the application's interactivity and immediacy.

2.4.3 Connection and Control of Motors

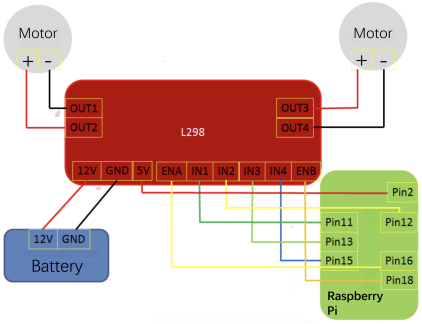
The L298N motor driver is commonly used for controlling motors in robotics due to its ability to drive two motors simultaneously and support motor directions with a high current output. In this design, a Raspberry Pi is used to control the motors through the L298N, enabling precise manipulation of motor speeds and directions based on real-time data processing from a camera.

Circuit Configuration

The GPIO (General Purpose Input/Output) pins of the Raspberry Pi are utilized to interface with the L298N motor driver[7]. The motor driver's input pins (INT1 to INT8) are connected to specified GPIO pins on the Raspberry Pi[8], which allows for controlling up to four motors (two motors with bidirectional control).



(a) Schematic diagram of circuit connection



(b) L298N and Raspberry PI connection diagram

Figure 15: Two kinds of circuit diagrams of L298N

2.5 Mechanical Structure

2.5.1 Design description and drawings

The mechanical structure has been greatly improved since the design document. We need it to be concise and easy to manufacture while implementing functions such as self-rotation, straight movement, and turning. As shown in Figure 16, we use acrylic layers and copper pillars to build the body of the machine and three omni wheels to control its movement, based on the principle of force balance. For self-rotation, all three wheels rotate at the same speed and in the same direction. For straight movement, one wheel's speed is set to zero, while the other two wheels rotate at the same speed but in opposite directions.

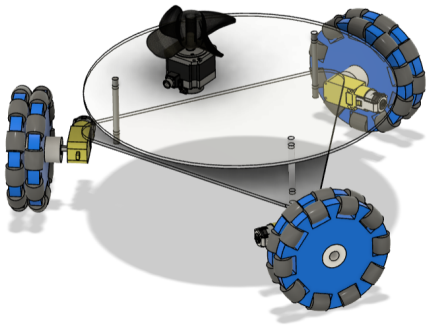
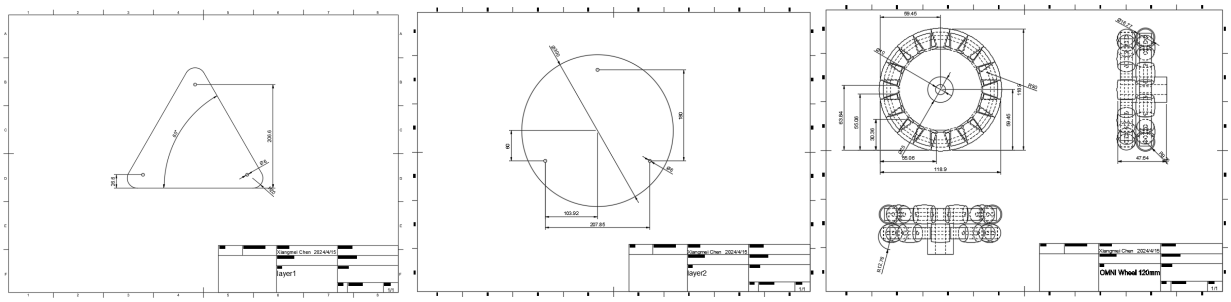


Figure 16: CAD model.

The current design significantly reduces mechanical and electrical complexity. Once the mosquito is detected to exist, the machine will rotate by itself to provide the camera with a 360-degree view; once the mosquito is localized, the machine will move towards it by issuing different commands to the three wheels.

Figures 17 depict CAD drawings of the mechanical system. The system has few components. The top and bottom layers of the cart are made by laser-cutting acrylic boards. All electrical components will be taped to the acrylic boards.



(a) Drawing for the bottom layer. (b) Drawing for the top layer. (c) Drawing for the omni wheel.

Figure 17: Drawing for the layers and wheels.

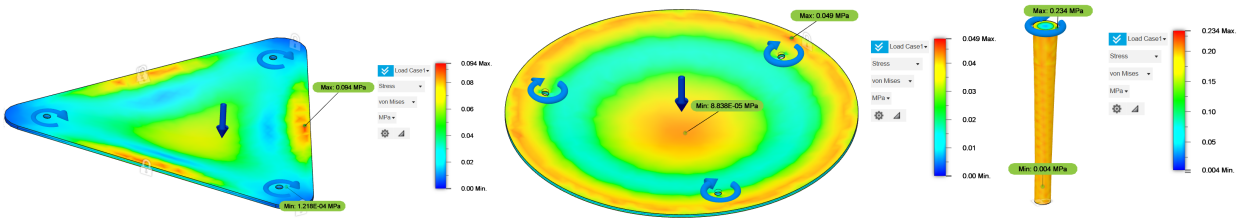
2.5.2 Simulation results

In summary, simulation results indicate that our mechanical design is theoretically safe. The stress distribution provides suggestions on where to place weight during physical testing.

The material for the top and bottom layers is 3 mm acrylic. For the bottom layer, we assume a downward force of 5 N at each hole and a moment of 1 N*mm. The maximum stress observed is 0.094 MPa, which is below the yield stress of acrylic (40 MPa). It is notable that the maximum stress occurs at three edges; therefore, if weight is to be added to the board, it is advisable to place it in the dark blue area.

Regarding the top layer, we assume a downward force of 2.5 N and a moment of 1 N*mm. While the overall stress is higher than that of the bottom layer, it still does not exceed the yield stress. Interestingly, the different geometry provides insights into where weight should be placed. Unlike the triangular bottom layer, the circular top layer exhibits a distinct stress distribution.

I also conducted a simulation analysis for the shaft, as shown in Figure 13. The material used is stainless steel, with a yield stress of 250 MPa. This analysis is for the scenario involving a quick turn, resulting in a torque of 5 N*mm being exerted on it. The maximum stress observed is 0.234 MPa, indicating that it is well within the safe range.



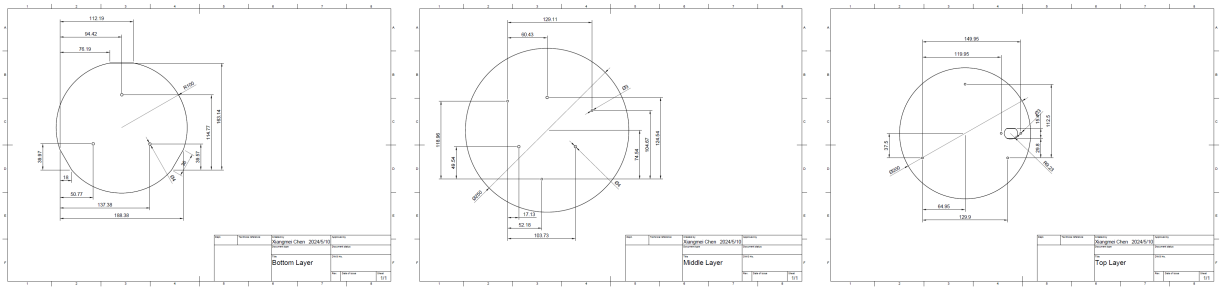
(a) Simulation results for the bottom layer. (b) Simulation results for the top layer. (c) Simulation results for shaft.

Figure 18: Simulation results for the components.

2.5.3 Design alternatives

As seen in Figure 22, we manufactured and built the first physical model. During testing, we found that electrical components were crowded at the bottom layer, which was very messy. Another issue was that the bottom layer was large compared to those three wheels. Though the acrylic board can withstand the weight, it was deformed, resulting in the three wheels not holding a vertical angle with the ground. There were also issues with the way the motor was fixed because screws and nuts would loosen during moving.

To address these issues, we improved the design. We made the machine in three layers while reducing the size of the bottom layer. The drawings for the three acrylic layers are shown in Figures 21.



(a) Drawing for the bottom layer. (b) Drawing for the middle layer. (c) Drawing for the top layer.

Figure 19: Drawing for the layers.

The connection between the motor and acrylic board is made using 502 glue. The overall appearance of the machine is shown in Figure 20.

(a) Physical model during the first test.

(b) Appearance of the design alternative.

Figure 20: Physical models.

2.6 Power Supply Module

2.6.1 Design Ideas

The Power Supply Module, a cornerstone of our mosquito attack device, is meticulously crafted to cater to the power demands of the Raspberry Pi in Control Subsystem. The design philosophy is rooted in the principle of efficiency, where we harness the potential of a 12V lithium battery to power the system. This choice is pivotal for several reasons:

Firstly, the 12V lithium battery is renowned for its high energy density, which translates to longer operational times without the need for frequent recharging.

Secondly, by stepping down this 12V supply to 5V specifically for the Raspberry Pi rather than using another 5V power supply, it significantly reduces the weight of the device, which is critical for enhancing portability.

2.6.2 RT8279 for Raspberry Pi

The RT8279 [9] is a highly efficient and compact step-down voltage regulator designed to power the Raspberry Pi in the Control Subsystem of our mosquito attack device. It employ feedback control mechanisms to regulate output voltage, ensuring stable power supplies. The components such as feedback resistors, inductors, capacitors, and soft-start capacitors are crucial for achieving the desired output voltages and protecting the devices.

Feedback resistor dividers R_1 and R_2:

For the resistor dividers, the formula that calculates the values of resistors is given by

$$V_{OUT} = V_{REF}(1 + \frac{R_1}{R_2})$$

where V_{REF} is the reference voltage with typical value $1.222V$. At the same time, the value of R_1 is restricted to $100k\Omega$ rather than giving a wide range to select. Therefore, the value of R_2 can be calculated as

$$R_2 = \frac{R_1}{\frac{V_{OUT}}{V_{REF}} - 1} = \frac{100k\Omega}{\frac{5V}{1.222V} - 1} \approx 32.3k\Omega$$

Output inductor L:

The formula the determines the value of inductor that operates the ripple current is shown as below. During calculation, the oscillator frequency is the same as $500kHz$, while the maximum output current reaches to $3A$.

$$L = \frac{V_{OUT}(1 - V_{OUT}/V_{IN,MAX})}{F_{SW} \times I_{OUT,MAX} \times 20\%} = \frac{5V(1 - 5V/12V)}{500kHz \times 3A \times 20\%} \approx 9.7\mu H$$

Input capacitor C_{IN} and Output capacitor C_{OUT}:

For the capacitors that connects to VIN pin, it's shown above in the application circuit with two capacitors with value $4.7\mu F$. And C_{OUT} is designed to be two capacitors with value $22\mu F$.

Enable Operation:

The enable pin should be driven higher than 1.4V to turn on the IC, where we chose to connect it to V_{IN} in series with a resistor of $100k\Omega$ since the EN pin can also be externally pulled to High by adding a $100k\Omega$ or greater resistor from the VIN pin.

The circuit diagram of RT8279 we designed is shown as below, which takes 12V as input and output voltage of 5V without the heating issue running continuously at 3A.

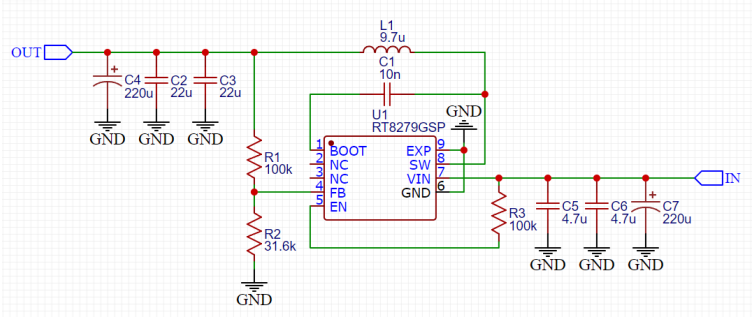


Figure 21: Designed Circuit Diagram Figure of RT8279.

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 Cost of labor

We take the average salary of UIUC graduates as our hourly wage, which is $20 per hour. Assume our team works three hours a day and five days a week, and there are 13 weeks to work. So, the total labor is $\$20 \times 3 \times 5 \times 13 \times 2.5 \times 4 = \$39000$.

3.1.2 Cost of parts

Part #	Description	Manufacturer	Quantity	Cost
1	Raspberry Pi 4B plus camera	Raspberry Pi Foundation	1	589 RMB
2	Arduino Development Board ATMEGA16U2	ArduinoLLC	1	80 RMB
3	Microphone for Raspberry Pi 4B	ArduinoLLC	1	9 RMB
4	Small fan	Telesky	1	7 RMB
5	Single chip small car	Beikemu	1	30 RMB
6	Bogie	Boxi	1	90 RMB
7	L298N	STMicroelectronics	1	6 RMB
8	Delipow 18650 lithium battery pack	Delipow	1	46 RMB
9	1080P Camera	Linboshi	1	196 RMB
10	printed PCB & components	JLC Technology Group	1	57 RMB
11	ball bearing	Tao Factory	1	3 RMB
Total				1113 RMB

Table 1: Cost Table.

3.1.3 Sum of Costs

The grand total costs is approximately $40000.

3.2 Schedule

Week	Xiangmei Chen	Peiqi Cai	Yang Dai	Lumeng Xu
3/25	Finish CAD model version 1. Finish purchases. Laser cut major parts.	Set up the running environment of Yolov8 on raspberry pi, and deploy the Yolov8 model.	Improve the YOLOv8 algorithm and start work on PCB design.	Research and identify a suitable dataset for mosquito wingbeat sounds, download and organize the dataset for further processing.
4/1	Assemble things together and test stability.	Ensure that the microphone is properly connected to the Raspberry Pi and set up the audio processing software to capture sound data.	Research on the codes and algorithms to perform data augmentation on the visual and audio datasets.	Begin coding the neural network architecture for sound classification and implement preliminary training using a portion of the dataset.
4/8	Test motor to see if the cart can move, revolute, and move up and down. Refine CAD if needed. Add features by 3D printing. Work on individual progress report.	Develop or adapt existing software to process the audio input from the microphone and detect mosquito sounds.	Collect and organize actual image and audio datasets and perform data augmentation on them.	Continue training the neural network with the full dataset. Validate the neural network's performance using a separate validation dataset.
4/15	Cooperate with detection subsystem and localization subsystem.	Create a program that can generate PWM signals to control the speed and direction of the motors connected to the servos.	Label the new datasets with a more complex background and train the model on them.	Connect the microphone to the Raspberry Pi. Set up the audio processing software to capture sound data accurately and test microphone functionality and ensure high-quality audio input.
4/22	Test the entire system. Add features by 3D printing or laser cutting if needed.	Build a PWM generation program on Raspberry Pi, responsible for sending PWM signals to the servos based on the mosquito's position in the camera's field of view.	Research on the codes and algorithms to locate the target based on the images feedback from Raspberry Pi.	Conduct tests with new, unseen sounds to verify the neural network's accuracy and begin integrating the neural network into the real-world environment for initial testing.
4/29	Make sure that the entire system is robust.	Calibrate the system for mosquito tracking, ensure that the camera and motors work in harmony.	Build the interfaces for the Roboflow 3.0 to control the rotation information contained in PWM, based on the target position.	Collaborate with the team on the design of the attacking subsystem and begin manufacturing or prototyping components for the subsystem.
5/6	Work on final report draft.	Develop or implement an algorithm that uses the mosquito's position data to control the motors, keeping the mosquito centered in the camera's view.	Test the performance of the localization subsystem in normal cases.	Finalize the design of the attacking subsystem, integrate the neural network into the system for real-time mosquito detection and response.
5/13	Work on demo and revision of individual report.	Test the system components individually.	Collaborate with other teammates with the testing on the components.	Test each component of the system individually to ensure proper functionality and identify and address any issues that arise during testing.
5/20	Prepare presentation and final report.	Ensure that the Raspberry Pi and all connected components are properly powered, and optimize power usage for efficient operation, especially if the system is battery-operated.	Ensure the performance of the device and make refinement on logic bugs.	Ensure the code can be run correctly and the connection of all parts are correct, check for overall integration.

Table 2: Schedule of the project.

4 Requirements and Verification

4.1 Audio Detection Module

4.1.1 Model Training Accuracy

Requirements: The machine learning model must correctly identify mosquito sounds with an accuracy of at least 85%.

Verification: Prepare some test datasets consisting of non-mosquito sounds. Executing the model and recording outcomes to calculate accuracy (A):

$$A = \frac{Correct}{Total};$$

where Correct is the number of labels where the outcome equals to the predicted, and Total is the number of all outcome labels. We need to ensure $A \geq 85\%$.

Results: We chose three datasets containing different pronunciations of English words "cat", "dog" and "bird" and imported them into our model to test, and all the three accuracy are 90% or so, which is above our requirement.

Figure 22: Testing Accuracy Result of Audio Detection Module.

4.1.2 Audio Processing and Detection Latency

Requirements: The total time from mosquito sound detection by the microphone to the identification of the sound by the software should not exceed 25 seconds. This ensures timely activation of the subsequent subsystems for effective mosquito targeting and eradication.

Verification:

1. Record the timestamp t_0 when mosquito sound is detected by the microphone.
2. Record the timestamp t_1 when the audio capture finishes.
3. Record the timestamp t_2 when mosquito detection is confirmed.

4. Calculate $T_{\text{total}} = t_2 - t_0$ for each trial and ensure $T_{\text{total}} \leq 10$ seconds.

We need:

$$\frac{1}{n}\sum_{i=1}^{n} T_{\text{total},i} \leq 10 \text{ seconds}$$

where n is the number of trials.

Results: We tested the code with different audio files, containing mosquito sounds and non-mosquito sounds, for 15 times, and the average timestamp duration is 22.4s.

4.2 Computer Vision Module

4.2.1 Model Metrics

Requirements: The Roboflow 3.0 model must correctly identify at least 90% of mosquitoes in the validation dataset with a precision of 85% or higher and a recall of 85% or higher.

Verifications: Process the captured images with the Roboflow 3.0 model. Calculate the precision and recall with different confidence based on the documented data and the formula

$$Precision = \frac{TP}{TP + FP}, \text{ and } Recall = \frac{TP}{TP + FN}$$

Also, use other metrics such as Mean Average Precision (mAP) to determine the performance and the value of confidence, where $mAP = \frac{1}{n}\sum_{k=1}^{n} AP_k$

Results: The training results shows sufficient performance of the model, where precision and recall reach 88.2% and 87.4% for the best checkpoint. The map also reaches 90.5% in this version.

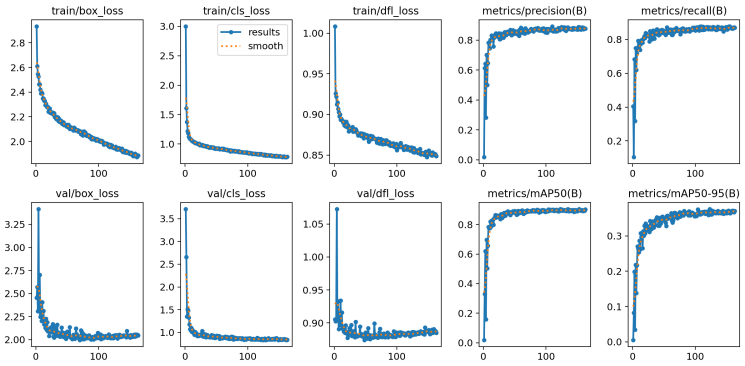


Figure 23: Training Metrics Result of Roboflow 3.0 Object Detection model.

4.2.2 FPS and Inference Latency

Requirements: The inference speed should be fast enough so that the control unit is able to receive 3-4 frames per second.

Verifications: Build the SSH tunnel connection between Raspberry Pi and inference server and transfer the image to the server to infer. Calculate the frames per second (FPS) by:

$$FPS = \frac{1}{end\_time - start\_time}$$

where $end\_time - start\_time$ is the time interval for each time the control unit reads a frame. Calculate the average FPS by averaging the processing time across 15-20 frames and verify if the average FPS meets the real-time processing constraint.

Results: After transferring the image to the inference server, received the inferred result and calculate the inference time, we get the FPS result for the 18 samples as follows.

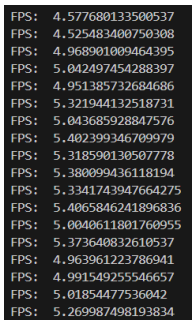


Figure 24: FPS of the image captured for 18 samples.

where we get the average FPS $FPS_{avg} = \sum FPS_i \approx 5.11\ frames\ per\ second$, which is above our requirements.

4.3 Movement and Rotation Module

4.3.1 Servo Motor Response

Requirements: Servo motors must respond within 0.5 seconds of a control signal.

Verification: Servo motors must respond to control signals within 0.5 seconds, ensuring immediate adjustment of camera positioning,

$$T_{\text{res}} \leq T_{\text{res,max}} = 0.5\text{s}$$

where T_{res} is the servo motor response time .

Results: After testing our model for ten times, we measured the time duration between the signal received moment at the Raspberry Pi from the state at the camera and the moment when the motor started to act accordingly, and calculated the average time duration by the formula

$$T_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} T_{\text{res}} = 0.48\text{s}$$

where n is 10, and the result shows that our motor response time meets the requirement.

4.4 Power and Control Module

4.4.1 PCB Functionality and Soft Start Time

Requirements: The PCB should be able to regulate the 12V input to the required 5V and the output voltage deviation is within ±5%, with the soft start time at around 4.5-5.5ms.

Verification: Connect the PCB to a 12V power source and measure the output voltage using an oscilloscope. Observe the output voltage waveform and calculate the soft start time using the oscilloscope data.

$$\text{Soft Start Time} = t_2 - t_1$$

Analyze the output voltage waveform and ensure that it remains stable at 5V with minimal fluctuations after the soft start period and verify that the output voltage is within the specified tolerance of ±5% from the 5V target:

$$\text{Voltage Deviation} = \frac{|V_{out} - 5V|}{5V} * 100\%$$

Where V_{out} is the measured output voltage.

Results:

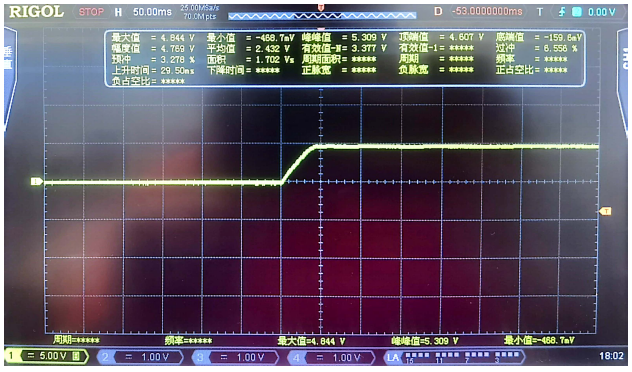


Figure 25: Oscilloscope data to measure peak value and soft start time.

The output voltage of the PCB took around 4.8ms to reach from 0V to 5V and can be identified from the oscilloscope data. According to the measurement, the minimum value of voltage is -468.7mV, which is the background voltage of the oscilloscope, and the maximum value of voltage is 4.844V. The voltage deviation is calculated as:

$$\text{Voltage Deviation} = \frac{|4.844V - 5V|}{5V} * 100\% = 3.12\%$$

which satisfied the requirements.

5 Conclusion

5.1 Accomplishments

Our machine successfully detects, localizes, and captures mosquitoes within a 2-meter range. It utilizes a microphone and camera, guided by machine learning models, to detect and track mosquitoes. The machine features a mechanical structure with 360-degree rotation capability and a fan unit for effective mosquito capture. These capabilities are made possible by a power and control subsystem with a useful PCB that provides stable power and coordinates operations through a Raspberry Pi control unit. Overall, our design almost meets our requirements and can effectively deal with mosquitoes.

5.2 Uncertainties

5.2.1 Uncertainty in Localization Subsystem

Camera Resolution and Field of View (FoV): The high-resolution USB camera's performance may be affected by lighting conditions and lens cleanliness. For instance, if there are reflections in the camera's FoV or dust on the lens, it could reduce localization accuracy.

Assuming optimal conditions, pixel density might be 300 PPI (Pixel Per Inch). Under adverse conditions (e.g., lens dirt or poor lighting), effective pixel density could drop to 150 PPI, potentially doubling the error margin in localization. The mathematical expression of the effective pixel density is:

$$P_{\text{effective}} = P_{\text{optimal}} - \Delta P$$

Where $P_{optimal}$ is the optimal pixel density and ΔP is the decrease in pixel density due to adverse conditions.

5.2.2 Uncertainty in Power Subsystem

Input Voltage Fluctuations:

Although lithium battery is stable, the input voltage to the power supply module may vary due to battery discharge or environmental factors. Assuming the battery discharge follows an exponential decay model, the input voltage can be expressed as:

$$V_{\text{in}}(t) = V_{\text{initial}} * exp(-t/\tau)$$

where $V_{\text{in}}(t)$ is the input voltage at time t, V_{initial} is the initial battery voltage, t is the elapsed time, and τ is the time constant of the battery discharge.Fluctuations in input voltage can lead to variations in the output voltage, since RT8279 has an input constraint of 5.5V to 36V operating input range. If the voltage of the battery drops to lower than 5.5V, the output of PCB may not be stable anymore.

5.3 Future Work / Alternatives

For future improvements, we can examine the subsystems one by one.

For the detection subsystem, future work should focus on real-time detection. Sounds made by mosquitoes are actually very small compared to environmental noise. Additionally, the frequency of the sound varies with temperature and humidity, making it a very complex issue.

For the localization and attack subsystems, future work should aim at reducing time lag and improving resolution so that our machine can accurately differentiate mosquitoes from other objects. When it comes to sucking in mosquitoes using a fan, we must find an easier way to clean the container after mosquitoes are captured.

As for the power and control subsystem, it can be enhanced by using a lighter-weight battery.

5.4 Ethical Considerations

A qualified project must adhere to the ethics codes outlined in IEEE Policies and ACM [10], [11]. As stipulated in the team contract, the four of us will collaborate to ensure mutual respect and fairness, committing to upholding these codes collectively and making requisite risk mitigation plans accordingly.

Our project seeks to manage mosquitoes for a healthier public environment, addressing the spread of diseases caused by them. It consists of mosquito detection, localization using a camera, and elimination. While there are no ethical concerns in detection, using a camera raises privacy issues. We respect life and ensure humane mosquito elimination without endorsing cruelty.

To mitigate risks, we inform individuals about monitoring in the experimental area to protect privacy. We use audio files and mosquito images for testing, avoiding real mosquitoes to prevent cruelty and harm to people during testing.

References

[1] WHO. "Mosquitoes." (2021), [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/mosquitoes (visited on 03/27/2024).

[2] H. Mukundarajan, F. J. H. Hol, E. A. Castillo, C. Newby, and M. Prakash, "Using mobile phones as acoustic sensors for high-throughput mosquito surveillance," *eLife*, vol. 6, 2017. DOI: https://doi.org/10.7554/eLife.27854.

[3] J. Gallagher. "Announcing roboflow train 3.0." (2023), [Online]. Available: https://blog.roboflow.com/roboflow-train-3-0/ (visited on 05/10/2024).

[4] R. Parthasarathi, *Computer Architecture*. INFLIBNET Centre, Jul. 2018, Licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. [Online]. Available: https://www.cs.umd.edu/~meesh/411/CA-online/chapter/computer-architectureintroduction/index.html (visited on 04/15/2024).

[5] J. Solawetz. "What is yolov8? the ultimate guide." (2023), [Online]. Available: https://blog.roboflow.com/whats-new-in-yolov8/ (visited on 03/05/2024).

[6] Lenovo, *Lenovo 500 fhd webcam - overview and service parts*, https://support.lenovo.com/us/en/accessories/acc500143-lenovo-500-fhd-webcam-overview-and-service-parts, [Online; accessed 15-April-2024], 2024.

[7] Raspberry Pi Foundation, *Raspberry pi 4 model b datasheet*, [Online; accessed 15-April-2024], 2024.

[8] Components101, *L293n motor driver module*, https://components101.com/modules/l293n-motor-driver-module, [Online; accessed 15-April-2024], 2024.

[9] *5a, 36v, 500khz step-down converter*, RT8279, Richtek Technology Corporation, Dec. 2011.

[10] IEEE. "IEEE Code of Ethics." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/05/2024).

[11] ACM. "ACM Code of Ethics and Professional Conduct." (2018), [Online]. Available: https://www.acm.org/code-of-ethics (visited on 03/05/2024).