SASTA OLIPHANT SCIENCE AWARDS

SOUTH AUSTRALIAN SCIENCE TEACHERS ASSOCIATION

Prize Winner

Programming, Apps & Robotics

Year 7-8

Nivaan Sardana

St Peter's College

ROWE SCIENTIFIC PTY. LTD. www.rowe.com.au

Australian Government Department of Defence

Government of South Australia Department for Education

SASTA South Australian Science Teachers Association

```
// Mosquito Decimator Code by Nivaan Sardana
// Credits to references mentioned in main report

// Include requiered libraries

#include <Servo.h>
// Create pan and tilt instances of servo
Servo panServo;
Servo tiltServo;

const int laserPin = 8;        // Digital pin for laser control
const int panPin = 9;          // PWM pin for pan servo
const int tiltPin = 10;        // PWM pin for tilt servo

String inputString = "";       // data from serial monitor
bool newData = false;

void setup() {
```

Connection lost. Cloud sketch actions and updates won't be available.

You appear to be offline. Without an Internet connection, the Arduino CLI might not be able to download the required resources and could cause malfunction. Please connect to the Internet and restart the application.

ArduinoTrial2 | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
ArduinoTrial2.ino
Output
Offline
Ln 60, Col 31
Arduino Uno on COM9 [not connected]
OneDrive > Nivaan - Personal > Desktop > OLIPHANT - 25- CLEANED > Mosquito Trial mirrored > ArduinoTrial2
Search ArduinoTrial2
Name
Status
Date modified
Type
Size
ArduinoTrial2.ino
7/24/2025 10:13 AM
INO File
2 KB
Share
Details
Type INO File
Size 1.81 KB
File location C:\Users\Nivaa\OneDrive\De...
Date modified 7/24/2025 10:13 AM
Properties
LG

MOSQUITO DECIMATOR

Prototype Laser Turret for Disease Control and Agricultural Pest Control

A Mosquito Decimator is a Laser Turret which is used to find, track and then eliminate mosquitoes. It works by using Deep Machine Learning detection system called YOLO (You Only Look once) which is linked to a camera that detects the mosquito, sends the coordinates of the detected object to the Arduino which then moves the laser mounted on servos. This device is important because it can help control mosquito spread diseases like malaria and the ross river virus. If it can do this, it will save numerous lives across the world especially in regions like Indian subcontinent and Africa, where it is a major concern due to hot, humid and unhygienic environment. It can also be tweaked to detect other types of insects, which can then be useful in agricultural pest control in an organic manner.

HOW I GOT THE IDEA?

I got the idea of the mosquito laser turret when we recently moved to our new home. Our new house has a big grassy backyard, but above the grass, there are often hundreds of hovering mosquitoes. This was frustrating me as I was not able to enjoy the grassy yard with my dogs as much as I wanted to.

Then, one day when I was leaving for school, the sprinklers popped up in the grass and began spraying water! When I saw this, I got the idea for my project! What if I made a machine like a sprinkler, which sits in the yard, but instead of water, it shoots laser beams that can kill the mosquitoes hovering over grass! This machine can then kill mosquitoes and protect lawns and houses from pests. This can also be very useful in underdeveloped countries where people often sleep outside in summer. With right type of strong motors, this could be scaled up for use in pest control of large agricultural farms.

HOW THE MODEL WORKS?

The very first step for this project is to enable object detection. With advent of Machine Learning, there are very smart and accurate ways to do this. The process involves training a machine on a large dataset of images of the object (mosquitoes) so that it learns to recognise them. YoloV8 is one such Deep Machine Learning model.

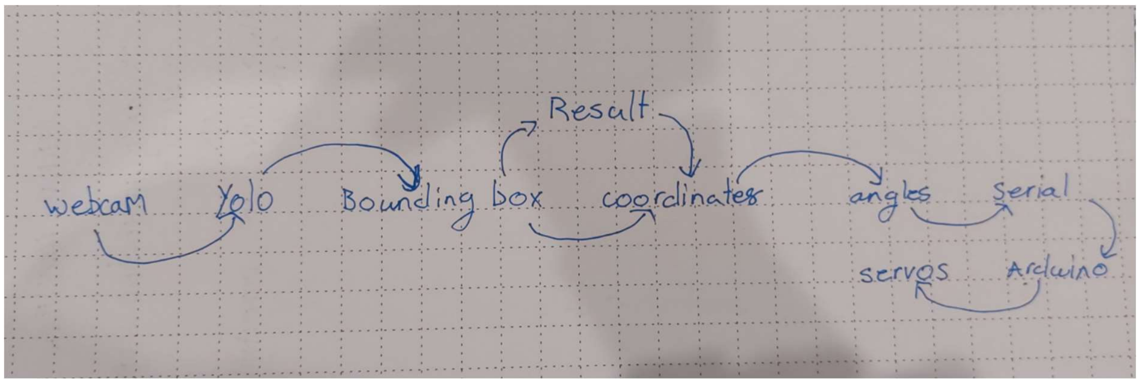
I found the most appropriate good sized dataset of mosquito images on Roboflow. I then sent it over to Google Colab, which used YoloV8 to train the model to view all these images several times, so it can begin to recognise them when they see something similar. This trained model was then downloaded to my PC and linked to my Visual Studio Python Code.

The project uses two programming platforms:

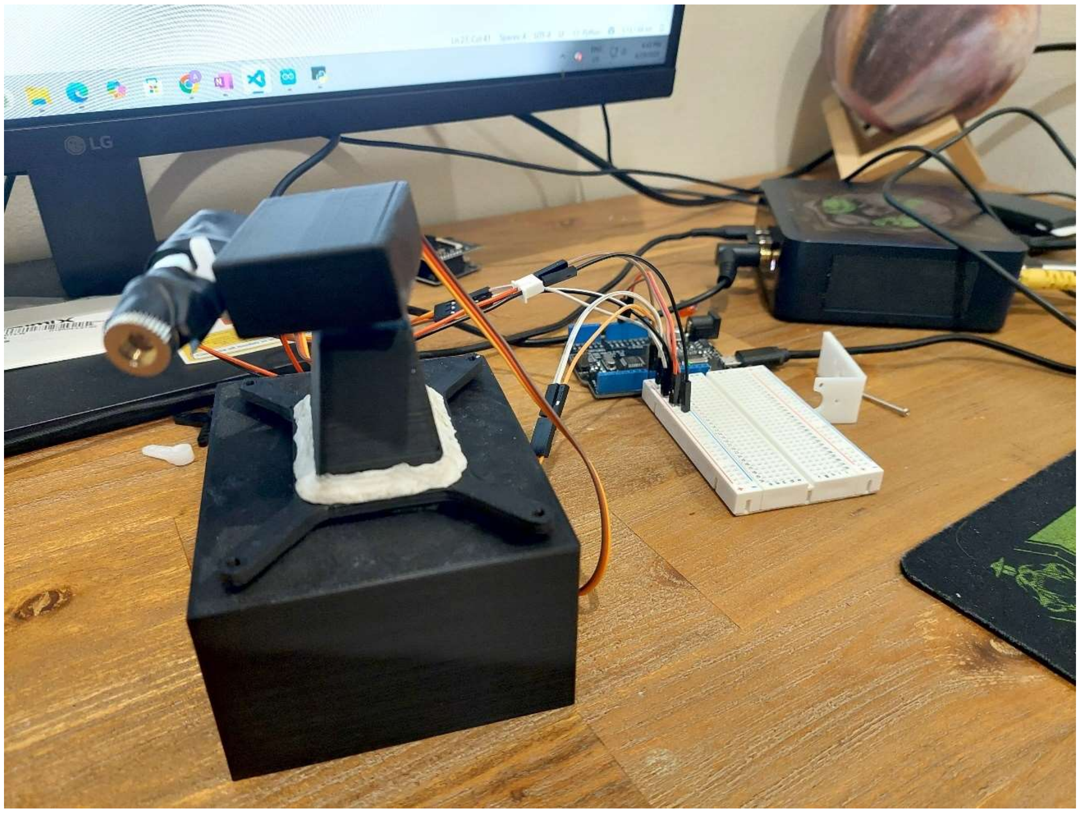
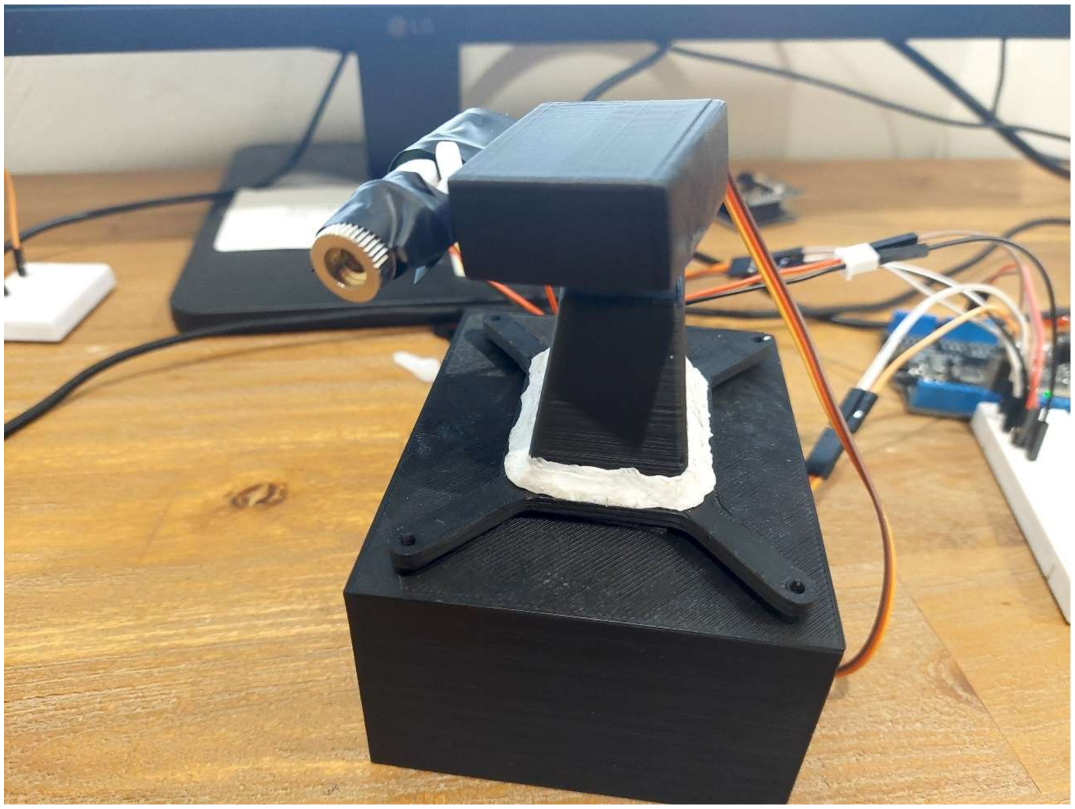
1. **Visual Studio program** for activating the webcamera, viewing images from it, inferring what the machine can see in its field of view, detecting mosquito in that field. It also generates a box around the detected mosquito and pushes its coordinates on to the serial monitor of COM port Arduino is connected to.
2. **Arduino IDE Code** for picking up the coordinates data from the serial monitor that then engages the servo motors to track the detected object and fire the laser beam.

   The Coding has 6 steps in one loop, which run across both programming platforms. Please refer to codes at the end of the report.

1. The YOLO model in Visual Studio takes in the webcam output and runs inference on it to try to find an object that it has been trained to find- in my case, the mosquito.
2. Once it finds the object, the python code draws a bounding box around it
3. Then it sends the coordinates of the center of the box over serial monitor to the Arduino
4. The Arduino Uno uses these coordinates and sends commands to the servos to adjust to those coordinates
5. The laser activates and shoots the target, in this case the mosquito
6. It repeats the process



Sketch Image- Flow of information



Images showing Model set up with turret and laser

KEY COMPONENTS:

How does YOLO object detection work?

YOLO works by taking screenshots of your screen or webcam every millisecond and runs inference on them and then a bounding box is drawn around the target. Inference is the process YOLO performs on an image to determine what the target is in the photo. The last and optional setting of YOLO is prediction. YOLO will predict where the target will go based on its previous movements. However, we won't use this setting in this model because mosquitoes are unpredictable when they move. In our case, we rely on the coordinates of the center of bounding box for identifying location of mosquito.

What is a Serial Monitor?

A serial monitor in an Arduino project is a medium that provides a way to communicate between the microcontroller and the computer. There is only ONE serial monitor that is linked to a particular COM port in a computer. We use this property of one single serial monitor in our project to connect the two programs. While running our codes, we first need to identify what COM port is named as and set our Arduino to link to the port and also update the COM port name in the VS code. This way both programs refer to the same COM port and hence connection is established over the serial monitor.

What is a servo and how it works?

A servo is a type of electrical motor which is used for accurate speed and torque. Inside a servo is a DC motor and a potentiometer for precise position input and output. A servo motor is part of a 'closed loop system' that uses input to control rotational linear speed and position. This is the best type of motor for this project as it provides precise direction control. The problem though is that economical servo has very low weight carrying capacity and does not have very smooth movement.

How do wires work?

Wires are found everywhere in our daily lives from a toy to the fridge. Wires work by using tiny strips of highly conductive wire to transmit energy. These small strips are usually of metal such as, copper, aluminium and sometimes even gold! However, the small strips can also be a mix of different metals depending on the object it's being used for. But because wires transmit electricity it means that the person could get electrocuted when handling wires, so they are most likely wrapped in rubber or nonconductive materials to reduce chance of shock. In our project, similar to most robotics project, wires are heavily used for communicating between different components and sending current, receiving sensor data, etc.

How does an Arduino work?

An Arduino is a miniature computer which works on a microcontroller. When you write the code in Arduino IDE and you flash it onto the Arduino, it gets stored on the onboard chip designated to retain sketches. Then, it will endlessly loop through it, doing what it is instructed to do until it is either turned off, or has completed its task. This microcontroller is very popularly used in making hobby models and even some advances study models.

SCIENCE CONCEPTS:

The project makes use of several scientific concepts:

Mosquito Detection: The project relies on a trained machine that recognizes biological or physical features of mosquitoes – their wings shape, body shape, colour, legs, etc that define the particular specie of mosquito. This biology is the backbone of the whole system.

Servo Coordinates and Tracking use mechatronics and robotics to convert the location of mosquito into coordinates that the servo uses.

The laser itself uses principles of physics and optical science where the beam is focused optically to produce a strong laser beam. This beam then kills the mosquito or damages its body/ wings etc. using principles of biology again.

CHALLENGES AND DIFFICULTIES:

The whole process of making this model to programming presented several challenges, with the key ones mentioned below:

Programming Challenges

First was when I started training the model in Google Colab. Colab is a cloud-based environment where we can run python codes using Googles strong servers and see its results. It is a popular platform for training models. I got hit with a challenge at the start. On the free Colab environment, getting the runtime to last longer to even attempt to train the YOLO model became difficult. I overcame this by getting Colab plus, which extends your runtime and allows you to train faster.

After that, the next problem which occurred was that I couldn't export the YOLO model into an ONNX (Open Neural Network Exchange, a very popular format for transferring trained model between different platforms) format, meaning that I couldn't run it on my PC. However, I overcame this by finding out how to use YOLO models on PC's without having to export it to ONNX.

The last problem which occurred during the training process was that the models I trained kept getting corrupted after a few days and they weren't detecting anything. However, I realised they weren't detecting since in the Colab I wasn't downloading the file correctly. So, I learnt how to do this after a few trials.

Secondly, when I was uploading the code to my Arduino and running it, while running it on python, I noticed that the Arduino was not moving the turret since the serial monitor didn't work. I fixed this by enabling an option in Arduino IDE which allowed the Arduino to receive and send writing in serial monitor, thus allowing it to move to the correct coordinates.

Other minor problems were that the image from camera as well as the coordinates from the detection had to be mirrored as originally the laser would go left when mosquito was going right! This was easier to fix by just updating the code mirror command.

The last problem I encountered was that I needed to transfer all the code and program files from my computer (Windows based) over to my MacBook so I could take it to the interview, since my setup (PC with monitor) I use to code is too big to take. During the transfer, the code did not work because the YOLO model has to be in the same directory as the python code. However, when I did so, it still didn't work. I realized that there were problems because of iCloud. I overcame this problem by using my hard drive's storage instead of iCloud. This resolved the problem and allowed me to continue on my MacBook.

The hardware for this project was simpler than my past projects. It had a small turret model. I made this using my 3D printer at home. The most difficulty in physical model has been with limited strength of motors as they are sometimes unable to take load of the laser beam which is made of brass. Finding the suitable laser beam that would be good for a demonstration model and not create risks for the judges and me was also a challenge.

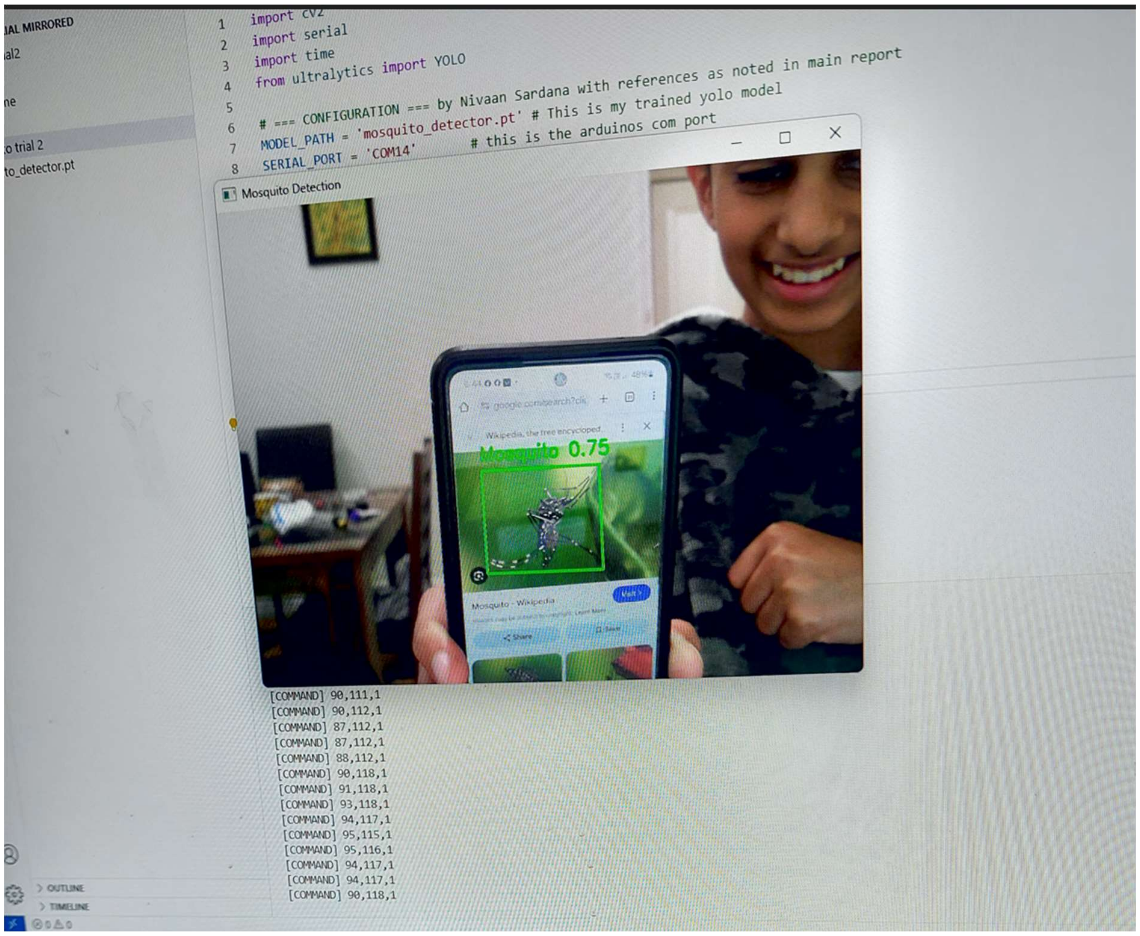
STEPS AHEAD:

As a person who wants to keep developing on their earlier models, some steps after this that I will take would be making the tracking more precise and faster, and I would do this by training the YOLO model for longer and making the dataset bigger, thus making it more advanced. Mosquitoes are very small creatures and very fast moving in random patterns. So, the system has to be better trained and stronger tools to match speed of mosquito. Also, dataset has to cover lots of mosquito breeds on different types of background.

I would also make the pan and tilt system be based on NEMA motors or stepper motors instead of servos to make it more powerful and precise, like a true targeting system would be.

Additionally, I would make the system run entirely on a microcontroller instead of needing additional processing power from a pc, so it can become more portable and process the object detection locally, on itself. This would also need a powerful battery like a LiPo battery to provide the required energy.

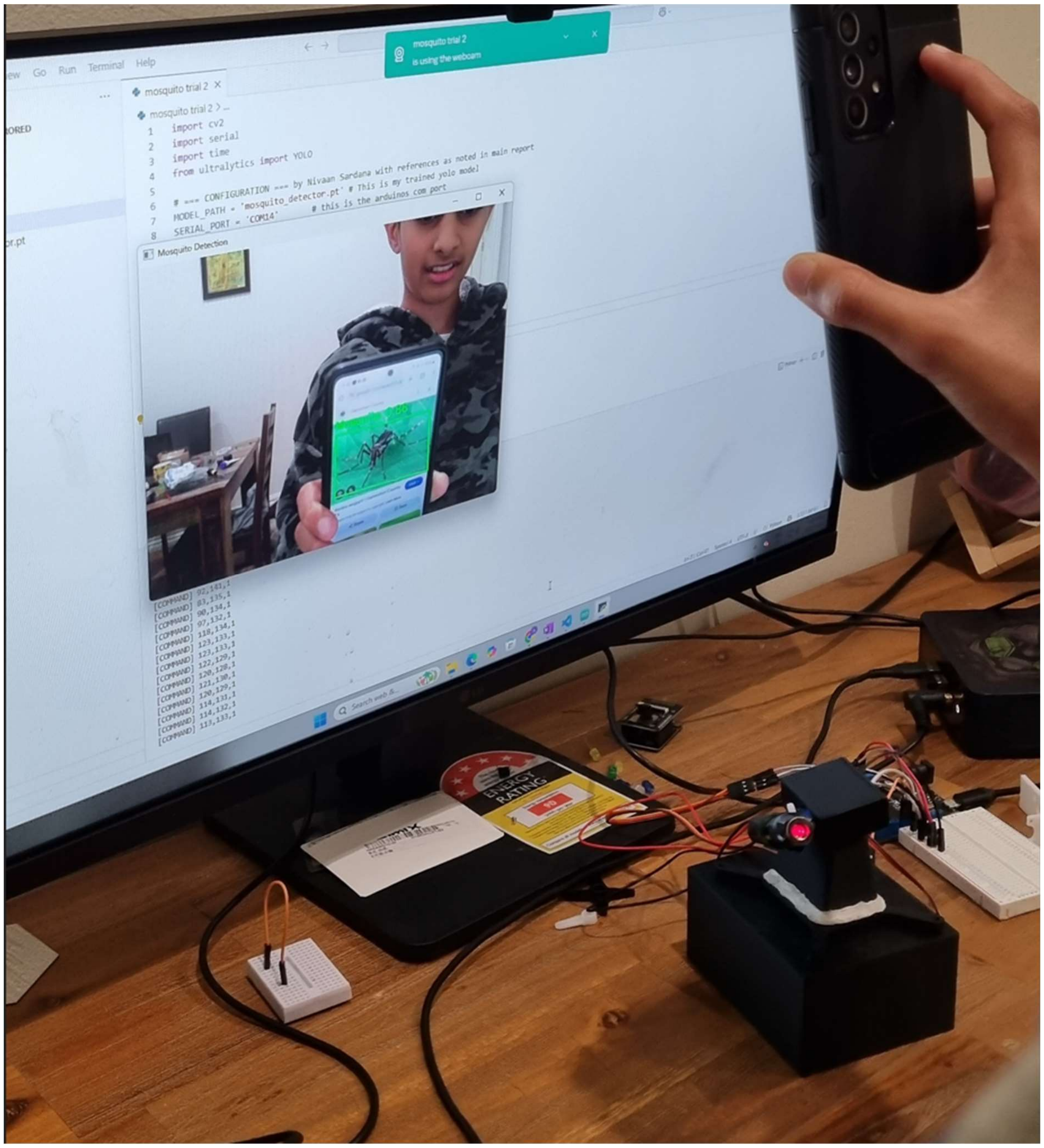
A major challenge was in ordering the components for the system. The hardest component to find was a safe laser, which can't hurt anyone, but powerful enough for the demonstration. This took a lot of time because we needed something affordable, safe and light enough to be seated on the turret without weighing it down. I eventually found out a laser which met the criteria on AliExpress.



```
import cv2
import serial
import time
from ultralytics import YOLO

# === CONFIGURATION === by Nivaan Sardana with references as noted in main report
MODEL_PATH = 'mosquito_detector.pt' # This is my trained yolo model
SERIAL_PORT = 'COM14'      # this is the arduinos com port
```

```
[COMMAND] 90,111,1
[COMMAND] 90,112,1
[COMMAND] 87,112,1
[COMMAND] 87,112,1
[COMMAND] 88,112,1
[COMMAND] 90,118,1
[COMMAND] 91,118,1
[COMMAND] 93,118,1
[COMMAND] 94,117,1
[COMMAND] 95,115,1
[COMMAND] 95,116,1
[COMMAND] 94,117,1
[COMMAND] 94,117,1
[COMMAND] 90,118,1
```

Image showing successful detection of mosquito in the camera feed



```
import cv2
import serial
import time
from ultralytics import YOLO

# === CONFIGURATION === by Nivaan Sardana with references as noted in main report
MODEL_PATH = 'mosquito_detector.pt' # This is my trained yolo model
SERIAL_PORT = 'COM14'      # this is the arduinos com port
```

```
[COMMAND] 92,141,1
[COMMAND] 83,135,1
[COMMAND] 90,134,1
[COMMAND] 97,132,1
[COMMAND] 118,134,1
[COMMAND] 123,133,1
[COMMAND] 123,133,1
[COMMAND] 122,129,1
[COMMAND] 120,128,1
[COMMAND] 121,130,1
[COMMAND] 120,129,1
[COMMAND] 114,131,1
[COMMAND] 114,132,1
[COMMAND] 113,133,1
```

Image showing mosquito detection in camera and laser getting activated (red light)

APPLICATIONS

- As per World Health Organization, globally there were 263 million cases of malaria and 597,000 deaths. The project can be developed at commercial scale at a reasonable cost and provided to nations that have lots of these cases. One Mosquito Decimator can be placed amongst a family of sleeping people and they can sleep more peacefully as mosquitoes will be killed by the turret.
- This project can also be used in home environment and outdoor barbeques or get together to keep people safe from mosquito
- Similarly, elderly and disabled can sleep more peacefully without mosquitoes biting them.
- The turret's dataset can also be expanded to include other pests like locusts, making it optimal for pest removal, and good for farmers (to save their crops from locust attack). This outcome does not rely on any harmful chemicals.
- The turret can be used to eliminate mosquito, and some of those mosquitoes could be carriers of parasites causing diseases like malaria, thus making it a good disease controller, especially in outdoor settings.
- This system can also be adapted to for data-collection on mosquito population, and if used with IoT can benefit public health surveillance.
- Likewise, the system can be adapted to sense and record various types of mosquitoes, butterflies, etc to better understand biodiversity.

The possible applications of this system are limitless.

REFERENCES:

https://www.who.int/news-room/fact-sheets/detail/malaria

STL model from https://www.thingiverse.com/thing:5782107

https://www.instructables.com/Python-Controlled-Arduino-Laser-Turret/

https://www.instructables.com/Arduino-Laser-Turret/

https://docs.ultralytics.com/modes/train/

https://www.youtube.com/watch?v=r0RspiLG260

https://www.digitalocean.com/community/tutorials/train-yolov5-custom-data

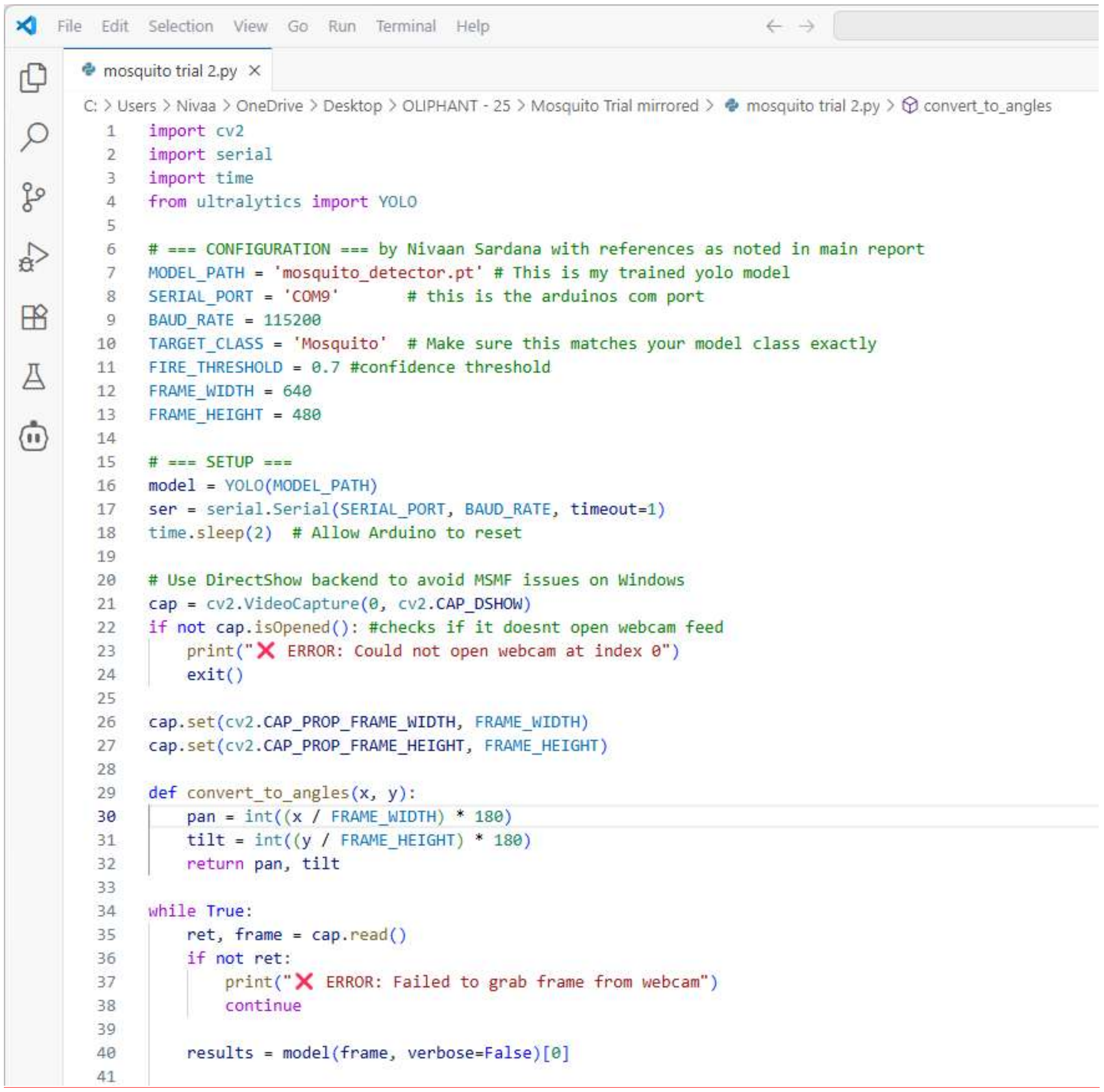
https://universe.roboflow.com/mosquitos-u6ipx/mosquito-detection-dataset

https://universe.roboflow.com/search?q=class%3Amosquito+detection+-+v5+2023-10-18+12%3A12am

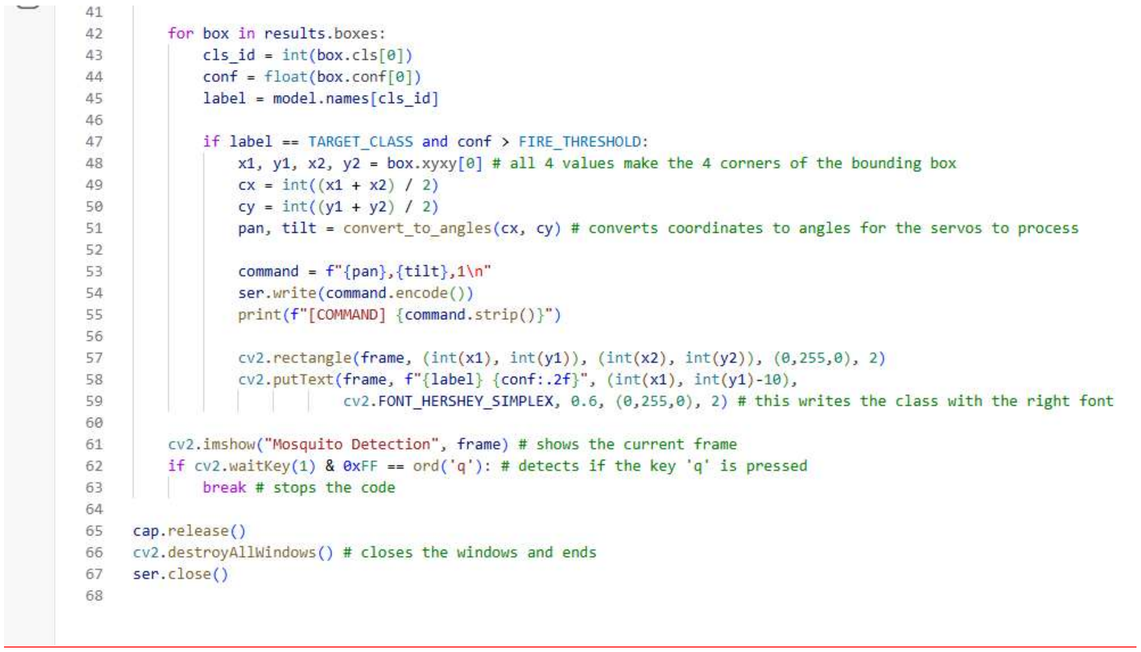
CODE PICS:

For codes I researched various websites to learn about them. As I could not find online example of mosquito turret that I was trying to make so getting codes to make it work was difficult. I was still able to find some websites that talked of different aspects of mosquito turret. So, I decided to take early help from those codes and then further develop it.

<u>VISUAL STUDIO CODE:</u>


File Edit Selection View Go Run Terminal Help
mosquito trial 2.py
C: > Users > Nivaa > OneDrive > Desktop > OLIPHANT - 25 > Mosquito Trial mirrored > mosquito trial 2.py > convert_to_angles

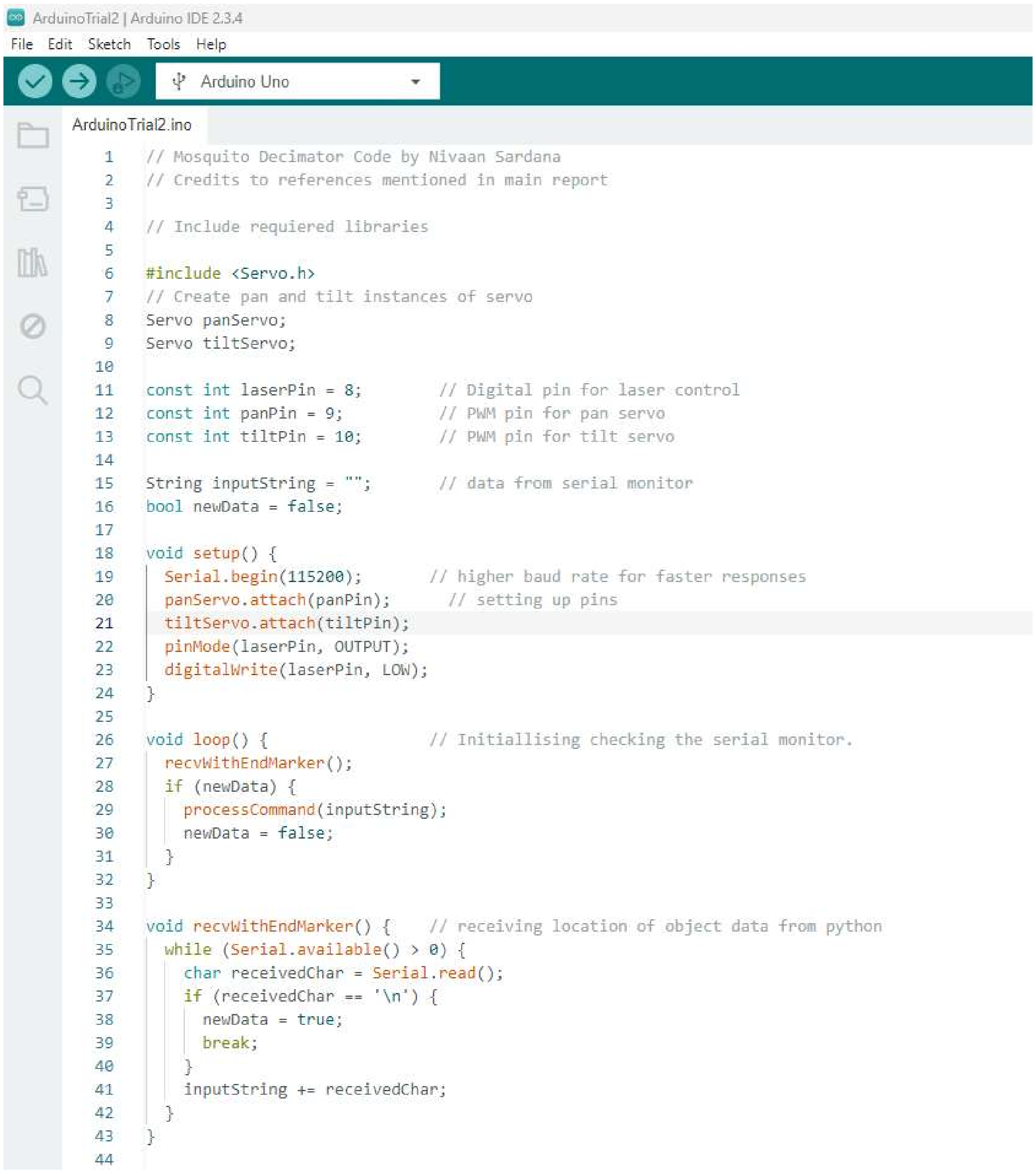
```python
import cv2
import serial
import time
from ultralytics import YOLO

# === CONFIGURATION === by Nivaan Sardana with references as noted in main report
MODEL_PATH = 'mosquito_detector.pt' # This is my trained yolo model
SERIAL_PORT = 'COM9'       # this is the arduinos com port
BAUD_RATE = 115200
TARGET_CLASS = 'Mosquito'  # Make sure this matches your model class exactly
FIRE_THRESHOLD = 0.7 #confidence threshold
FRAME_WIDTH = 640
FRAME_HEIGHT = 480

# === SETUP ===
model = YOLO(MODEL_PATH)
ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
time.sleep(2)  # Allow Arduino to reset

# Use DirectShow backend to avoid MSMF issues on Windows
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
if not cap.isOpened(): #checks if it doesnt open webcam feed
    print("❌ ERROR: Could not open webcam at index 0")
    exit()

cap.set(cv2.CAP_PROP_FRAME_WIDTH, FRAME_WIDTH)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT)

def convert_to_angles(x, y):
    pan = int((x / FRAME_WIDTH) * 180)
    tilt = int((y / FRAME_HEIGHT) * 180)
    return pan, tilt

while True:
    ret, frame = cap.read()
    if not ret:
        print("❌ ERROR: Failed to grab frame from webcam")
        continue

    results = model(frame, verbose=False)[0]

```



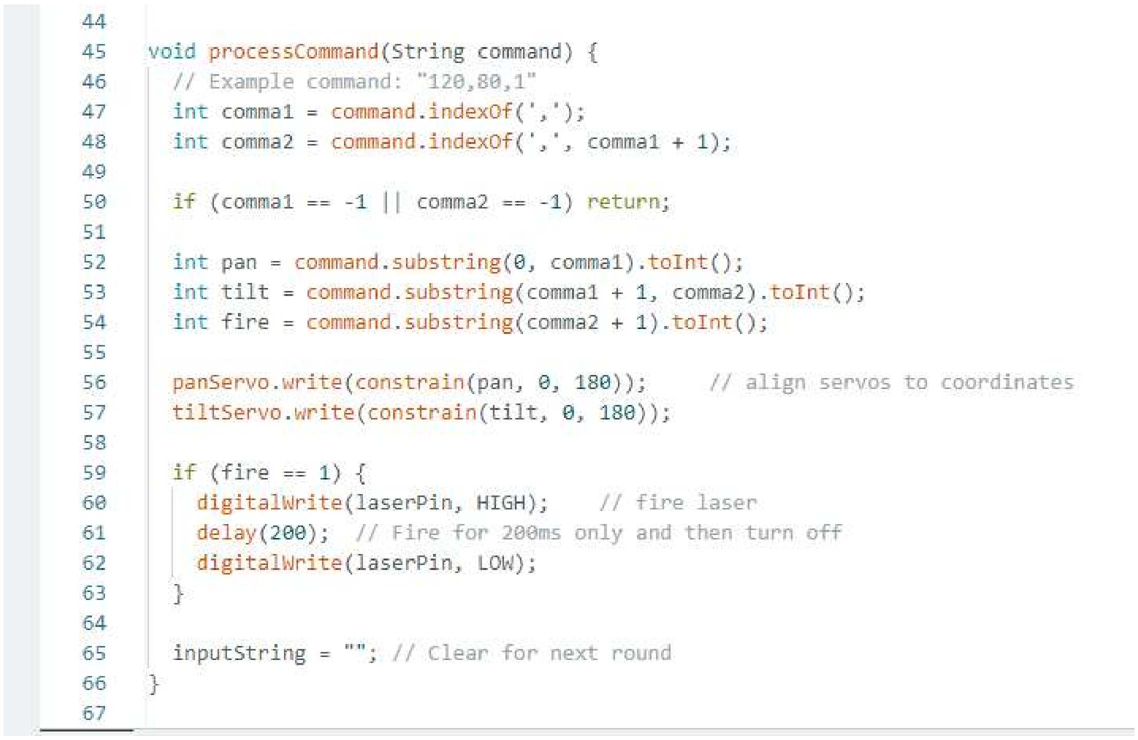
```python

    for box in results.boxes:
        cls_id = int(box.cls[0])
        conf = float(box.conf[0])
        label = model.names[cls_id]

        if label == TARGET_CLASS and conf > FIRE_THRESHOLD:
            x1, y1, x2, y2 = box.xyxy[0] # all 4 values make the 4 corners of the bounding box
            cx = int((x1 + x2) / 2)
            cy = int((y1 + y2) / 2)
            pan, tilt = convert_to_angles(cx, cy) # converts coordinates to angles for the servos to process

            command = f"{pan},{tilt},1\n"
            ser.write(command.encode())
            print(f"[COMMAND] {command.strip()}")

            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0,255,0), 2)
            cv2.putText(frame, f"{label} {conf:.2f}", (int(x1), int(y1)-10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0), 2) # this writes the class with the right font

    cv2.imshow("Mosquito Detection", frame) # shows the current frame
    if cv2.waitKey(1) & 0xFF == ord('q'): # detects if the key 'q' is pressed
        break # stops the code

cap.release()
cv2.destroyAllWindows() # closes the windows and ends
ser.close()

```

Arduino code on below pages

ARDUINO IDE CODE:


ArduinoTrial2 | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
ArduinoTrial2.ino

```
// Mosquito Decimator Code by Nivaan Sardana
// Credits to references mentioned in main report

// Include requiered libraries

#include <Servo.h>
// Create pan and tilt instances of servo
Servo panServo;
Servo tiltServo;

const int laserPin = 8;        // Digital pin for laser control
const int panPin = 9;          // PWM pin for pan servo
const int tiltPin = 10;        // PWM pin for tilt servo

String inputString = "";       // data from serial monitor
bool newData = false;

void setup() {
  Serial.begin(115200);       // higher baud rate for faster responses
  panServo.attach(panPin);      // setting up pins
  tiltServo.attach(tiltPin);
  pinMode(laserPin, OUTPUT);
  digitalWrite(laserPin, LOW);
}

void loop() {                  // Initiallising checking the serial monitor.
  recvWithEndMarker();
  if (newData) {
    processCommand(inputString);
    newData = false;
  }
}

void recvWithEndMarker() {    // receiving location of object data from python
  while (Serial.available() > 0) {
    char receivedChar = Serial.read();
    if (receivedChar == '\n') {
      newData = true;
      break;
    }
    inputString += receivedChar;
  }
}

```



```

void processCommand(String command) {
  // Example command: "120,80,1"
  int comma1 = command.indexOf(',');
  int comma2 = command.indexOf(',', comma1 + 1);

  if (comma1 == -1 || comma2 == -1) return;

  int pan = command.substring(0, comma1).toInt();
  int tilt = command.substring(comma1 + 1, comma2).toInt();
  int fire = command.substring(comma2 + 1).toInt();

  panServo.write(constrain(pan, 0, 180));     // align servos to coordinates
  tiltServo.write(constrain(tilt, 0, 180));

  if (fire == 1) {
    digitalWrite(laserPin, HIGH);    // fire laser
    delay(200);  // Fire for 200ms only and then turn off
    digitalWrite(laserPin, LOW);
  }

  inputString = ""; // Clear for next round
}

```

Thank you for taking time to read my report.