Laser Pointer Turret Based Mosquito Air Defence System

Final Report

A. Hartman
20475323

Submitted as partial fulfilment of the requirements of Project EPR402
in the Department of Electrical, Electronic and Computer Engineering
University of Pretoria

November 2023

Study leader: Prof. P. de Villiers

Part 1. Preamble

This report describes work that I did *<to be completed>*.

Project proposal and technical documentation
This main report contains an unaltered copy of the approved Project Proposal (as Part 2 of the report).

Technical documentation appears in Part 4 (Appendix).

All the code that I developed appears as a separate submission on the AMS.

Project history
This project makes extensive use of existing algorithms on ... Some of the algorithms I used were adapted from ... Where other authors' work has been used, it has been cited appropriately, and the rest of the work reported on here, is entirely my own.

Language editing
This document has been language edited by a knowledgeable person. By submitting this document in its present form, I declare that this is the written material that I wish to be examined on.

My language editor was _____.

_____ _____
Language editor signature *Date*

Declaration

I, A. Hartman understand what plagiarism is and have carefully studied the plagiarism policy of the University. I hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else.

Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

I certify that the Project Proposal appearing as the Introduction section of the report is a verbatim copy of the approved Project Proposal.

A. Hartman Date

TABLE OF CONTENTS

Part 1. Preamble i

Part 2. Project definition: approved Project Proposal vi

1. Project description
2. Technical challenges in this project
3. Functional analysis
4. System requirements and specifications
5. Field conditions
6. Student tasks

Part 3. Main Report vii

1 Literature study 1

2 Approach 2

3 Design and implementation 4

3.1 Things I made 4

3.2 Design summary 4

3.3 Theoretical analysis and modelling 5

3.4 Simulation and Prototyping 10

3.5 Hardware design 10

3.6 Hardware implementation 14

3.7 Software design 22

3.8 Software implementation and optimisation 28

3.9 Final system integration and testing 30

4 Results 31

4.1 Summary of results achieved 31

4.2 Qualification tests 31

5 Discussion 37

5.1 Critical evaluation of the design 37

5.2 Considerations in the design 37

6 Conclusion 38

6.1 Summary of the work completed 38

6.2 Summary of the observations and findings 38

6.3 Contribution 38

6.4 Future work 38

7 References 39

Part 4. Appendix: technical documentation 40

HARDWARE part of the project 41

Record 1. System block diagram 41

Record 2. Systems level description of the design 41

Record 3. Complete circuit diagrams and description 41

Record 4. Hardware acceptance test procedure 41

Record 5. User guide 41

SOFTWARE part of the project 41

Record 6. Software process flow diagrams 41

Record 7. Explanation of software modules 41

Record 8. Complete source code 41

Record 9. Software acceptance test procedure 41

Record 10. Software user guide 41

EXPERIMENTAL DATA 41

Record 11. Experimental data 41

LIST OF ABBREVIATIONS

CAD	computer aided design
CCL	connected components labelling
CPU	central processing unit
CSI	camera serial interface
CUDA	compute unified device architecture
GPIO	general purpose input/output
GPU	graphics processing unit
LED	light emitting diode
MADS	mosquito air defence system
MIPI	mobile industry processor interface
PID	proportional-integral-derivative
PWM	pulse width modulation
RGB	red, green, blue
RPM	revolutions per minute
SORT	simple online and realtime tracking
USB	universal serial bus

Part 2. Project definition: approved Project Proposal

This section contains the problem identification in the form of the complete approved Project Proposal, unaltered from the final approved version that appears on the AMS.

Part 3. Main Report

1. Literature study

Malaria is still one of the leading causes of death in low-income countries according to the World Health Organisation [1]. Mosquitoes that do not carry diseases are also a general nuisance in the everyday life of people living in mosquito-prone areas. Therefore, it is necessary to pursue improvement in our defence against mosquitoes.

To be able to design a laser pointer turret-based mosquito air defence system it is necessary to understand the principles of computer vision object detection and real-time tracking.

One approach towards tracking is to perform pattern matching. In general pattern matching is searching and checking images for the presence of other given images (patterns) to find and mark the patterns' locations (if any) within the given images. However, the study conducted by Hurtik et al. [2] presents results that indicate the best frame rate they achieved was 0.43 frames per second. This is far too slow to be used in a real-time tracking application.

Another approach is to perform particle filter-based tracking. This considers the proximity and behaviour of other targets. In the case of social insect tracking, it is known that two targets cannot occupy the same space, and targets will actively avoid collisions. Unfortunately, the joint particle tracker proposed in [3] suffers from exponential complexity.

A popular approach is to separate the detection and tracking functions. While numerous deep learning algorithms can detect objects based on appearance, it is worth noting that mosquitoes, particularly when not filmed up close, prove too minute to be reliably detected using appearance-based methods. A viable alternative is to detect objects by isolating the background and foreground of the image [4]. The foreground of the image contains the objects of interest. In [5] objects that are too close to one another are split into two and abnormally small objects are merged.

A proposed tracking method is the Simple Online Real-time Tracking (SORT) algorithm [6]. The algorithm is composed of an estimation model which makes use of a Kalman filter and a data association system that is solved optimally using the Hungarian algorithm.

In the proposed mosquito air defence system mosquito detection will be based on background and foreground isolation. This method is suitable because the system will operate in a known test environment where the background will change minimally. The online real-time nature of this system makes the case for pattern matching and particle filtering unfavourable because of the computationally intensive nature of these techniques. The methods in [4], [5], and [6] will be further investigated for the proposed system.

2. Approach

The aim of this project was to develop a system that would illuminate mosquitoes with a laser turret. Throughout the remainder of this document this system will be referred to as the mosquito air defence system (MADS). The project can be viewed in terms of a set of integrated subsystems. The subsystems are as follows:

Laser Turret Control System
This subsystem is responsible for controlling the laser turret. The laser control system is discussed in

Laser Detection System
This subsystem is responsible for detecting the actual position of the laser with the camera. The laser detection is required to provide feedback to the laser turret control system. The laser detection system is discussed in

Mosquito Detection System
This subsystem is responsible for detecting the position of the mosquitoes. The mosquito detection system is discussed in

Mosquito Tracking System
This subsystem is responsible for tracking the mosquitoes and predicting their future positions. The mosquito tracking system is discussed in

The subsystems were integrated on a real-time embedded system and will be discussed in detail in the relevant sections. The system is designed to operate in a controlled environment constructed specifically for this project. The functional block diagram of the MADS is shown in Figure 1.

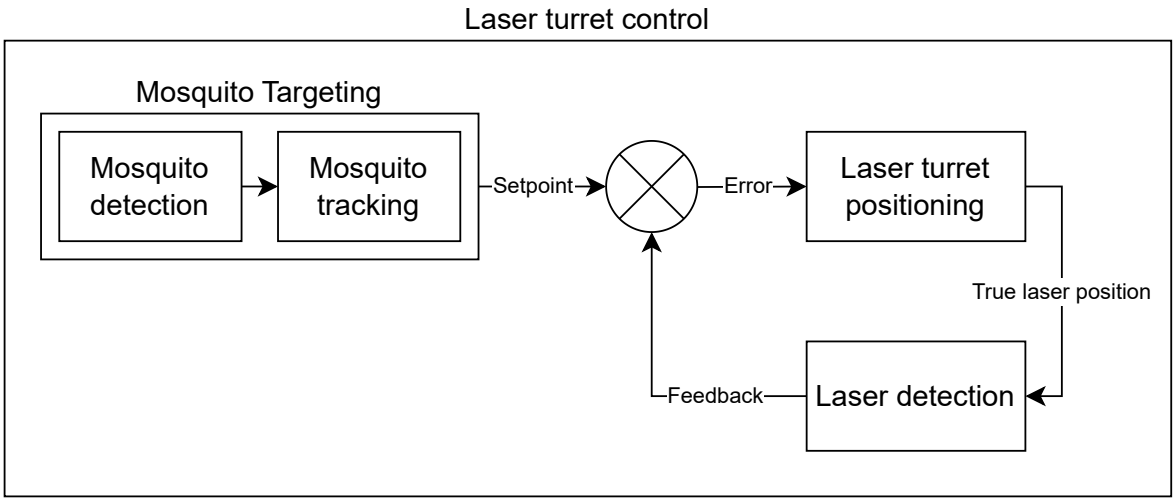


Figure 1.
Functional block diagram of the MADS.

The design choices for the MADS were made with careful consideration of the implicit real-time operation requirement of the system. All aspects of the MADS were designed to be as lightweight as possible in terms of computational complexity.

3. Design and implementation

3.1 Things I made

- Laser turret housing.
- Worked out the speed, torque, and step size required for the stepper motors.
- Interfaced with stepper motors using GPIO pins to drive the stepper motor drivers.
- Mapping between camera pixels and real-world co-ordinates. Had to do camera calibration and account for different perspectives of camera and turret.
- Calculate steps from angle required based on distance using Pythagoras.
- Distinguish between laser reflections using the geometry of the laser turret.
- Laser detection from first principles optimised with GPU kernels. 1. Gaussian smoothing, 2. Binarise with threshold, 3. Morphological operations (closing and opening), 4. Connected components labelling, 5. Find centroids of connected components, 6. Distinguish laser detections with turret geometry.
- Mosquito detection. Same image processing steps expect step 2 is either a less than threshold or background subtraction.
- Mosquito tracking using SORT algorithm. (Kalman filter and Hungarian algorithm).
- Laser turret PID controller. Must still be tuned because the error calculated is inaccurate. Should tuning be done without developing a model? Or should a model be developed? How do you develop a model?
- Feedback for turret. The current steps are saved at the instance that the frame is captured. The frame is then processed using the laser detection system. The pixel co-ordinates are converted to steps. The step error is calculated and added to the current steps.
- System integration on embedded system running in real-time on multiple threads.

3.2 Design summary

This section summarises the project design tasks and how they were implemented (see Table 1).

Deliverable or task	**Implementation**	**Completion of deliverable or task, and section in the report**
The mosquito detection subsystem had to be designed and implemented by the student.	The mosquito detection subsystem was designed and implemented from first principles.	Completed.
The laser detection subsystem had to be designed and implemented by the student.	The laser detection subsystem was designed and implemented from first principles.	Completed.
The laser turret control subsystem had to be designed and implemented by the student.	The laser turret control subsystem was designed and implemented from first principles.	Completed.
The mosquito tracking subsystem had to be designed and implemented by the student.	The mosquito tracking subsystem was designed and implemented from first principles.	Completed.
The various subsystems had to be integrated on a real-time embedded system.	The various subsystems were integrated on a real-time embedded system.	Completed.
Appropriate motors needed to be selected for the laser turret.	The stepper motors were selected based on the requirements of the laser turret.	Completed.

Table 1.
Design summary.

3.3 Theoretical analysis and modelling

3.3.1 Mapping pixel co-ordinates to metric co-ordinates

To control the laser, its position must be known in the world co-ordinate frame. The laser position was measured using a camera, thus the camera's pixel co-ordinate frame must be mapped to the world co-ordinate frame. To perform this mapping a camera model is required. The forward imaging model of a camera is shown in Figure 2.

Using the forward imaging model the pixel distance was mapped to the metric distance for the x-axis with

$$X = Z \times \left(\frac{x - x_{ref}}{f_x} \right), \tag{1}$$

where Z is the depth camera with respect to the world co-ordinate frame, x is the pixel of interest, x_{ref} is the reference pixel, and f_x is the effective focal length of the camera. Similarly,

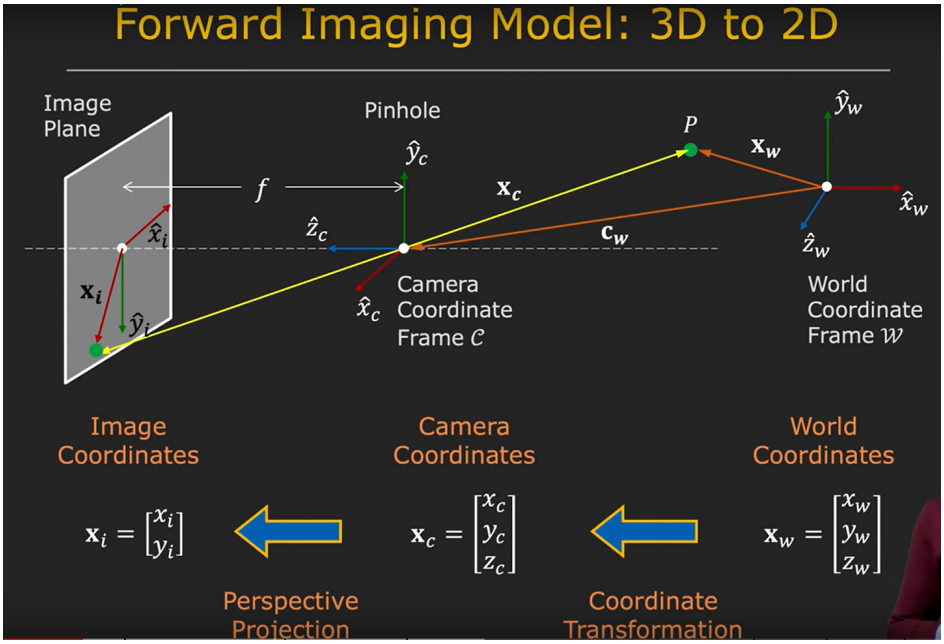


Figure 2.
Forward imaging model of a camera. CITE

the pixel distance was mapped to the metric distance for the y-axis.

The effective focal length of the camera f_x was determined through camera calibration.

3.3.2 Camera calibration

a. Intrinsic parameters

These parameters are inherent to the camera and remain constant unless the camera's internal settings (like focus) are changed. They are typically found by calibrating the camera using multiple views of a known pattern (like a checkerboard).

- **Camera Matrix (K)**: Defines the camera's internal characteristics. The principal components are:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

  where f_x and f_y are the focal lengths in pixels and c_x, c_y are the co-ordinates of the principal point. The principal point is the pixel co-ordinate where the camera's principal axis intersects the image plane.
- **Distortion Coefficients (D)**: Captures lens distortion. This is a vector with up to 5 elements in the common "plumb-bob" model of OpenCV

$$D = [k_1, k_2, p_1, p_2, k_3]$$

  where k_1, k_2, k_3 are radial distortion coefficients and p_1, p_2 are tangential distortion coefficients.

b. Extrinsic Parameters
These parameters capture the camera's orientation and position concerning the world or an external reference frame.

- **Rotation Vector (rvec)**: Represents the orientation of the camera. It's a 3x1 vector used to derive the rotation matrix R.
- **Translation Vector (tvec)**: Represents the position of the camera. It's a 3x1 vector capturing the translation in X, Y, and Z directions.

They represent the position and orientation of the camera relative to the world (or in your case, the turret's frame). Every time you move the camera or change the scene, these parameters would change. These are found using functions like `solvePnP` in OpenCV, which computes the pose of an object given some known 3D points on the object and their corresponding 2D projections in the image.

3.3.3 Morphological operations

Erosion is a fundamental morphological operation used to remove small structures or details from a binary image. It is defined as the basic set operation of moving a structuring element (usually a smaller binary matrix) over the input binary image and finding the intersection of the structuring element with the image. This operation can be mathematically expressed as

$$(A \ominus B)(x,y) = \bigcap \{A(x+i, y+j) | (i,j) \in B\}, \tag{3}$$

where

- A is the input binary image.
- B is the structuring element.
- $\ominus$ represents the erosion operation.
- (x,y) are the pixel co-ordinates in the resulting image.

Dilation is another fundamental morphological operation, but it is used to enhance or grow the features in a binary image. Dilation can be defined as the set operation that moves the structuring element over the input image and computes the union of the element with the parts of the image where the structuring element "hits". The mathematical expression for dilation is as follows:

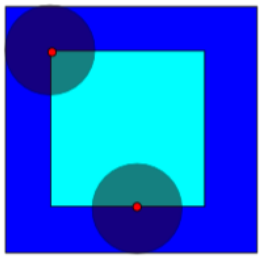
$$(A \oplus B)(x,y) = \bigcup \{A(x+i, y+j) | (i,j) \in B\} \tag{4}$$

where

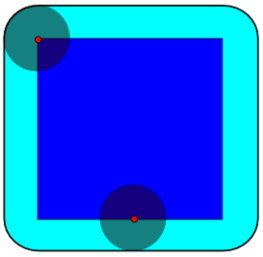
- A is the input binary image.
- B is the structuring element.
- $\oplus$ represents the dilation operation.
- (x,y) are the pixel co-ordinates in the resulting image.

The erosion and dilation operations are illustrated in Figure 3.

Opening is a compound morphological operation that consists of an erosion followed by a dilation using the same structuring element. It is primarily used to remove noise and small objects from a binary image. The opening of an image A by a structuring element B is defined



(a) The erosion of the dark-blue square by a disk, resulting in the light-blue square.



(b) The dilation of a dark-blue square by a disk, resulting in the light-blue square with rounded corners.

Figure 3.
The figure shows an example of erosion and dilation. This figure was modified from Renatokeshet at the English Wikipedia (2008).

as

$$A \circ B = (A \ominus B) \oplus B, \tag{5}$$

where

- A is the input binary image.
- B is the structuring element.
- $\circ$ represents the opening operation.

Closing is another compound morphological operation that consists of dilation followed by an erosion using the same structuring element. It is used to close small holes and gaps in objects and to connect nearby objects. The closing of an image A by a structuring element B is defined as

$$A \bullet B = (A \oplus B) \ominus B, \tag{6}$$

where

- A is the input binary image.
- B is the structuring element.
- $\bullet$ represents the closing operation.

Opening and closing are both idempotent operations, meaning that repeated openings or closing have no effect on the image. Opening and closing is illustrated in Figure 4.

3.3.4 Kalman filter

The Kalman filter is a recursive algorithm that estimates the state of a system from a series of noisy measurements. It is a powerful tool for tracking the state of a system over time and is widely used in control systems and robotics. The Kalman filter is based on a linear dynamical system model, which is defined by the following equations:

(a) The opening of the dark-blue square by a disk, resulting in the light-blue square with round corners.

(b) The closing of the dark-blue shape (union of two squares) by a disk, resulting in the union of the dark-blue shape and the light-blue areas.

Figure 4.
The figure shows and example of opening and closing. This figure was modified from Renatokeshet at the English Wikipedia (2008).

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1} \\ z_k &= Hx_k + v_k \end{aligned} \tag{7}$$

where

- x_k is the state vector at time k.
- z_k is the measurement vector at time k.
- A is the state transition matrix.
- B is the control matrix.
- u_k is the control vector at time k.
- w_k is the process noise vector at time k.
- H is the observation matrix.
- v_k is the measurement noise vector at time k.

3.3.5 SORT tracking

SORT (Simple Online and Realtime Tracking) is a popular algorithm for multi-object tracking in video sequences. The main goal of the SORT algorithm is to associate detection boxes across frames in a video sequence to form trajectories for each individual object. Here is a theoretical background on the SORT tracking algorithm: 1. Overview

SORT is designed to be both simple and efficient, achieving real-time performance. It is based on the tracking-by-detection paradigm, which means it relies on an external object detector to provide bounding boxes around objects of interest in each frame. The algorithm then associates these detections across frames to create tracks. 2. Detection

Before tracking, an object detection algorithm (such as Faster R-CNN, YOLO, or SSD) is applied to each frame to detect objects of interest. The output is a set of bounding boxes along

with their associated confidence scores. 3. Association

The core of the SORT algorithm is the association step, where detections are linked across frames to form tracks. This is done based on the intersection over union (IoU) metric, which measures the overlap between two bounding boxes. The steps involved are:

Prediction: For each existing track, the Kalman filter is used to predict the new position of the object in the current frame. Matching: The predicted positions of existing tracks are matched with the current frame's detections based on the IoU metric. A bipartite graph matching algorithm (such as the Hungarian algorithm) is used to find the optimal assignment. Update: The Kalman filter for each matched track is updated with the current detection. Creation and Deletion: New tracks are created for detections that could not be matched to existing tracks. Tracks that have not been matched for a certain number of frames are deleted.

4. Kalman Filter

The Kalman filter is a critical component of the SORT algorithm, providing a way to predict and update the state of the tracks based on the detections. The state of a track is represented as $[x, y, \dot{x}, \dot{y}]$, where x,yx,y are the coordinates of the center of the bounding box, ss is the scale/area of the bounding box, rr is the aspect ratio, and $\dot{x}, \dot{y}$ are the respective velocities. 5. Challenges and Limitations

Occlusions: SORT can struggle with objects that are occluded or interact closely with each other, as the IoU metric may not be sufficient for correct association. Identity Switches: In cases where objects cross paths or are close together, SORT can suffer from identity switches. Dependence on Detection: The performance of SORT is highly dependent on the quality of the detections provided by the external object detector.

3.4 Simulation and Prototyping

Detection and tracking done in python with videos.

3.5 Hardware design

The hardware overview is presented in Figure 5. The hardware consists primarily of a laser turret, a camera, a mosquito enclosure, and an embedded processing platform. The laser turret is used to move the laser beam across the mosquito enclosure. The camera is used to detect the laser beam and the mosquitoes. A bracket is used to position the laser turret and the camera relative to the mosquito enclosure. The embedded processing platform is used to control the laser turret and perform the image processing required for the laser detection and mosquito detection.

3.5.1 Embedded platform selection

To select an embedded platform for the project the following requirements were considered:

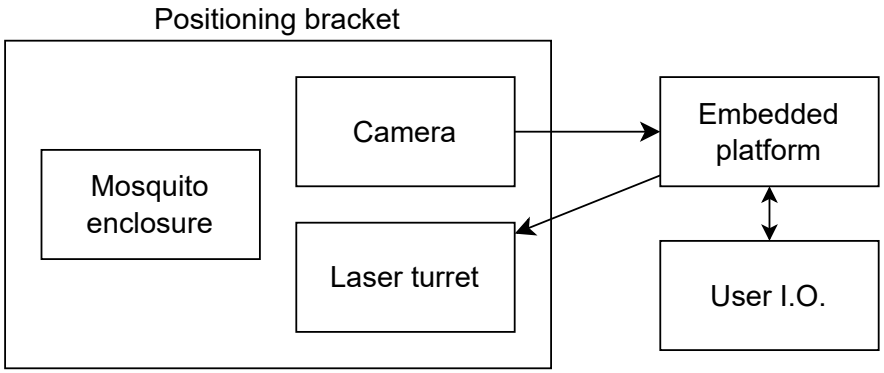


Figure 5.
Hardware overview.

Processing power
The embedded platform must be able to perform the image processing required for the laser detection and mosquito detection in real-time. Image processing is computationally intensive, but can largely benefit from parallelisation. Therefore, an embedded platform with a graphics processing unit (GPU) will be preferred.

Memory
The embedded platform must have sufficient memory to store the multiple images along with the required algorithms. A 1080p image with 3 8-bit colour channels requires 6 MB of memory.

Hardware interfaces
The embedded platform must be compatible with camera interfaces. It must also have sufficient general purpose input/output (GPIO) pins to interface with the stepper motor drivers. The embedded platform must also have support for a display output and a keyboard and mouse input to provide a user interface to control the system.

Availability and support
It is import to select an embedded platform that is readily available. It is important to select an embedded platform with large body of knowledge available and sufficient scientific support community since this is an individual project without access to people with expertise with the embedded platform.

The Nvidia Jetson Nano was chosen as the embedded platform for the project. The Nvidia Jetson Nano is a single-board computer with a GPU and a quad-core central processing unit (CPU). The Nvidia Jetson Nano has a mobile industry processor interface (MIPI) camera serial interface (CSI)-2 camera interface and has 40 GPIO pins. The Jetson Nano has a quad-core ARM Cortex-A57 MPCore processor, an Nvidia Maxwell GPU with 128 compute unified device architecture (CUDA) cores, and 4 GB of LPDDR4 memory. It also features an HDMI display output and a universal serial bus (USB) 3.0 port. The Nvidia Jetson Nano is readily available and has a large body of knowledge available and sufficient scientific support community.

3.5.2 Camera selection

To minimise the computational complexity of the image processing required for the laser detection and mosquito detection a low resolution will be used. The video resolution will not need to exceed 1080p. The camera must be able to capture images at a frame rate of at least 30 frames per second. The camera must also be compatible with the MIPI CSI on the Nvidia Jetson Nano.

The Raspberry Pi Camera Module V2 was chosen as the camera for the project. The Raspberry Pi Camera Module V2 is an 8 MP camera that can capture images at a frame rate of 30 frames per second at 1080p. The Raspberry Pi Camera Module V2 is compatible with the MIPI CSI on the Nvidia Jetson Nano.

3.5.3 Mosquito enclosure

The mosquito enclosure will be a rectangular prism as shown in Figure 6. The width of the enclosure must be a minimum length of 1 metre. The front facing surface will be transparent and the rest of the surfaces will be white. The enclosure will be fitted with internal lighting to ensure contrast between the background and the mosquitoes and to minimise the camera noise.

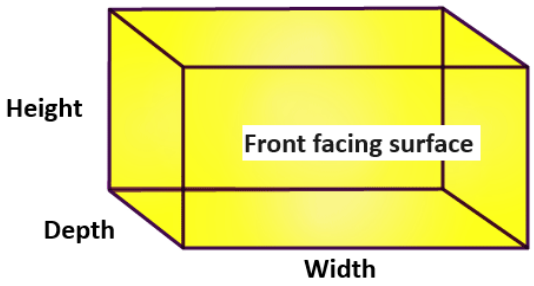


Figure 6.
Mosquito enclosure dimensions.

3.5.4 System positioning

The laser turret and the camera will be placed outside the mosquito enclosure in known positions relative to the enclosure. The bracket, shown in Figure 7, will be built to hold the laser turret and the camera in place relative to the mosquito enclosure.

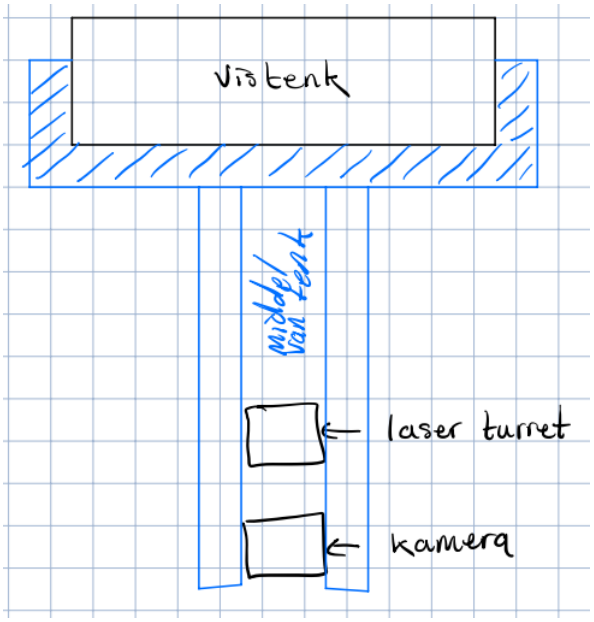


Figure 7.
System positioning bracket.

3.5.5 Laser turret

The laser turret design was inspired by commercial two-axis laser scanners. The position of the laser beam is controlled by adjusting the angles of two mirrors. This is illustrated in Figure 8. The mirrors are connected directly to the output shaft of the motors. The laser turret will be positioned orthogonally to the *xy*-plane of the mosquito enclosure. The point where the laser beam shines orthogonal to the *xy*-plane of the mosquito enclosure is considered the origin of the laser. This will occur when the two mirrors are at 45° relative to the laser beam.

When a single axis of the laser turret is considered it can be seen that a right triangle is formed between the laser turret and the *xy*-plane mosquito enclosure as shown in Figure 9. Using the properties of a right triangle the mirror angle θ required to shine the laser a distance d relative to the origin of the laser can be calculated using

$$\theta = \arctan\left(\frac{d}{z}\right), \tag{8}$$

where z is the distance between the turret and the back wall of the mosquito enclosure.

The lateral step size of the laser must be a maximum of 2 mm. This will ensure that the laser will be able to illuminate every position in the mosquito enclosure since the laser itself will be a few millimetres wide.

The required lateral speed is determined in accordance with the field specifications. The longest length of the mosquito enclosure d_{max} will be 1 m. To ensure that the laser can illuminate any set point inside the mosquito enclosure within 2 s of receiving a step input, it was assumed that the laser must be able to move with a velocity v_{laser} of least $1\,\mathrm{m\,s^{-1}}$ opposed

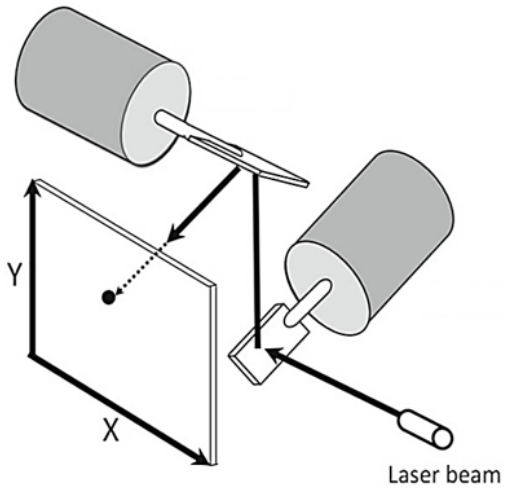


Figure 8.
Two-axis laser scanner schematic. This figure was modified from [7].

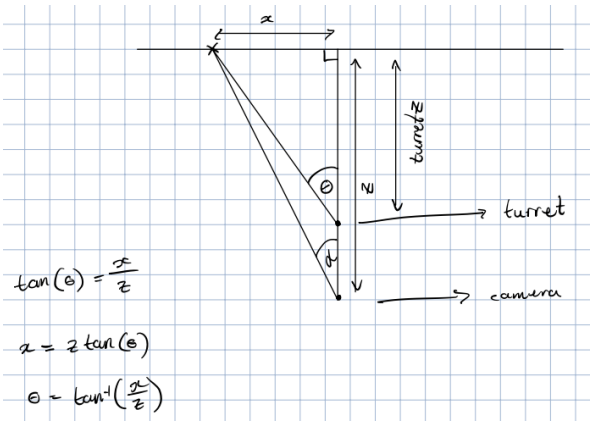


Figure 9.
Mirror angle calculation.

to the $0.5\,\mathrm{m\,s^{-1}}$ required to move the 1 m length of the mosquito enclosure within 2 s. This was done to accommodate for the settling time of the laser turret control system.

3.6 Hardware implementation

3.6.1 Mosquito enclosure

The mosquito enclosure was constructed from a second hand glass fish tank with dimensions $90 \times 38 \times 32$ cm. All the surfaces other than front facing surface was retrofitted with a white lining. Internal light emitting diodes (LEDs) were added to the enclosure, shining from the top and bottom surfaces to minimise shadows on the back surface the enclosure, to ensure

contrast between the background and the mosquitoes and to minimise the camera noise.

3.6.2 Laser turret

The specific geometry of the laser turret was designed with the goal to practically obtain a sufficiently small lateral step size of the laser on the back wall of the mosquito enclosure, while maintaining sufficient speed. The dimensions of the mirrors were chosen for practicality as $30 \times 15 \times 3$ mm.

a. Angular step size

The turret axis on which the laser beam is first incident on will be referred to as the first axis and the other axis will be referred to as the second axis for the remainder of this discussion. The rotation range of the first axis is bounded by the geometry of the turret since the laser beam reflected from the first axis must be incident on the second axis. With the chosen mirror dimensions, the maximum angle through which the first axis can rotate is 40°. This was conservatively determined geometrically in the to scale drawing in Figure 10. The rotation range of the second axis is not bounded by the geometry of the laser turret since the laser beam reflected from the second axis is incident on the mosquito enclosure. Therefore, the laser turret will be oriented such that the first axis moves the laser beam parallel to the shorter width of the mosquito enclosure and the second axis moves the laser beam parallel to the longer length of the mosquito enclosure.

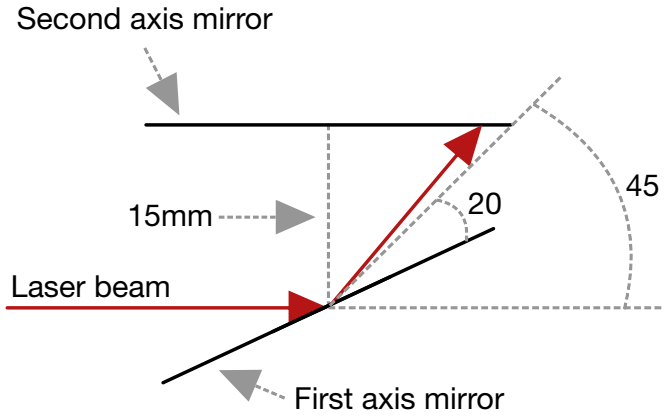


Figure 10.
Rotation range of first axis with chosen mirror dimensions.

The minimum distance z_{min} between the laser turret and the mosquito enclosure can be determined by rearranging Equation 8 to give

$$z_{min} = \frac{d}{\tan(\theta)} = \frac{38\text{ cm}}{\tan(40°)} = 45.3\text{ cm}, \tag{9}$$

where $d = 38$ cm is the height of the mosquito enclosure and $\theta = 40°$ is the maximum angle through which the first axis of the laser turret can rotate. The maximum step resolution θ_{step}^{max} of the motors can be calculated by substituting z_{min} and the required lateral step size d_{min} into

Equation 8 to produce

$$\theta_{step}^{max} = \arctan\left(\frac{x_{min}}{z_{min}}\right) = \arctan\left(\frac{2\text{mm}}{45.3\text{cm}}\right) = 0.253^\circ . \tag{10}$$

b. Angular speed

The maximum angular speed ω_{max} required by the motors can be determined with d_{max}, v_{laser}, and z_{min}. The maximum angle through which the turret must rotate is

$$\theta_{max} = \arctan\left(\frac{d_{max}}{z_{min}}\right) = \arctan\left(\frac{1\,\text{m}}{45.3\ \text{cm}}\right) = 65.6^\circ . \tag{11}$$

The maximum angular speed required to move the laser beam at the lateral speed $v_{laser} = 1\,\text{m}\,\text{s}^{-1}$ is

$$\omega_{max} = v_{laser} \times \theta_{max} = 65.6^\circ\,\text{s}^{-1} = 1.14\,\text{rad}\,\text{s}^{-1} . \tag{12}$$

The results in a required motor revolutions per minute (RPM) of

$$\omega_{max} = \frac{1}{2\pi} \times 1.14\,\text{rad}\,\text{s}^{-1} \times \frac{60}{1} = 10.9\ \text{RPM} . \tag{13}$$

c. Torque

The required torque τ was calculated using

$$\tau = I\alpha , \tag{14}$$

where I is the moment of inertia of the load, which is the mirror, and α is the required angular acceleration.

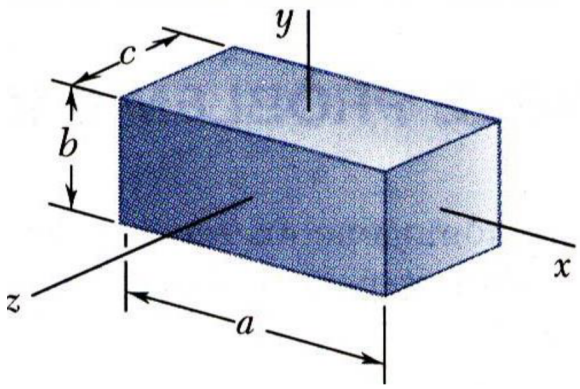


Figure 11.
Axes and dimensions of a rectangular prism.

The moment of inertia of the mirror was calculated using the equation for the moment of inertia of a rectangular prism with rotation about its x-axis as seen in Figure 11 given by

$$I = \frac{1}{12} M\left(b^2 + c^2\right) , \tag{15}$$

where M is the mass of the rectangular prism and b and c are the dimensions of the rectangular prism. The mass of the rectangular prism M will be calculated using the density of glass $\rho = 2500\,\text{kg}\,\text{m}^{-3}$ and the volume of the rectangular prism V given by

$$V = abc , \tag{16}$$

where a, b, and c are the dimensions of the rectangular prism. The dimensions of the rectangular prism is given by the chosen mirror dimensions as $a = 30\,\text{mm}$, $b = 15\,\text{mm}$, and $c = 3\,\text{mm}$. The mass of the mirror M is

$$M = \rho V = 2500\,\text{kg}\,\text{m}^{-3} \times (30\,\text{mm})(15\,\text{mm})(3\,\text{mm}) = 3.38\,\text{g}. \tag{17}$$

Thus, the moment of inertia of the mirror is

$$I = \frac{1}{12}M\left(b^2 + c^2\right) = \frac{1}{12}(3.38\,\text{g})\left((15\,\text{mm})^2 + (3\,\text{mm})^2\right) = 6.59 \times 10^{-8}\,\text{kg}\,\text{m}^{-2}. \tag{18}$$

The required angular acceleration of the motor α is calculated with

$$\alpha = \frac{\Delta\omega}{\Delta t}, \tag{19}$$

where $\Delta\omega = \omega_{max} - 0$ is the change in angular velocity and Δt is the change in time. The change in time Δt was determined by assuming the laser must be able to accelerate from a stand still to $v_{laser} = 1\,\text{m}\,\text{s}^{-1}$ extremely rapidly to ensure that the motors could respond to the irregular flight pattern of a mosquito. It was assumed that this acceleration must occur within 10 ms. Thus, the required angular acceleration α is

$$\alpha = \frac{\Delta\omega}{\Delta t} = \frac{1.14\,\text{rad}\,\text{s}^{-1}}{0.01\,\text{s}} = 114\,\text{rad}\,\text{s}^{-2}. \tag{20}$$

Given the calculated moment of inertia and angular acceleration the required torque τ is

$$\tau = I\alpha = 6.59 \times 10^{-8}\,\text{kg}\,\text{m}^{-2} \times 114\,\text{rad}\,\text{s}^{-2} = 7.51 \times 10^{-6}\,\text{N}\,\text{m}. \tag{21}$$

d. Motor and driver selection

Typical stepper motors have a basic step angle of 1.8°. This does not meat the required step resolution $\theta_{step}^{max} = 0.253°$, however the step angle can be reduced by using microstepping. Microstepping works by interpolating the pulse width modulation (PWM) signal to the stepper motor driver to produce intermediate step positions between the basic step positions. Stepper motor drivers that support up to 16 microsteps are readily available. Drivers that support up to 256 microsteps are also available. The step angle θ_{step} of a stepper motor using 8 microsteps is

$$\theta_{step} = \frac{1.8°}{16} = 0.1125°, \tag{22}$$

which is sufficient to meet the maximum required step resolution $\theta_{step}^{max} = 0.253°$. Therefore, a typical stepper motor with a basic step angle of 1.8° operated using at least 8 microsteps will be sufficient.

A drawback of using microstepping is that the effective torque of the stepper motor is reduced with each increase in microstepping. The torque reduction due to microstepping can be calculated with

$$\tau_{eff} = \tau_{rated} \sin\left(\frac{\pi S}{2}\right), \tag{23}$$

where

- τ_{eff} is the effective torque of the stepper motor for a specific amount of microstepping.
- τ_{rated} is the rated torque of the stepper motor.
- S is the reciprocal of the number of microsteps.

The effective torque of a stepper motor for the full range of microsteps available is tabulated

in Table 2. It can be seen that microstepping has a drastic impact of the effective torque of the stepper motor. Thus, the required torque will need to be calculated for the chosen microstepping operation.

Microsteps	Effective torque
1	100%
2	70.71%
4	38.27%
8	19.51%
16	9.8%
32	4.91%
64	2.45%
128	1.23%
256	0.61%

Table 2.
Effective torque of stepper motor for full range of microsteps.

The DRV8825 stepper motor driver was chosen since it is readily available and supports up to 32 microsteps. The DRV8825 stepper motor driver has a maximum current rating of 2.5 A and a maximum voltage rating of 45 V. The maximum microstepping mode will be utilised for increased resolution in the control of the laser. The required torque with 32 microsteps is

$$\tau_{required} = 7.51 \times 10^{-6}\,\mathrm{N\,m} \times \frac{100\%}{4.91\%} = 1.53 \times 10^{-4}\,\mathrm{N\,m}. \tag{24}$$

A bipolar NEMA8 stepper motor with specifications shown in Table 3 and a pull-out torque

Basic step angle	1.8°
Holding torque	$180\,\mathrm{g\,cm} = 1.77 \times 10^{-2}\,\mathrm{N\,m}$
Rated voltage	3.9 V
Rated current	0.6 A

Table 3.
Specifications of the NEMA8 stepper motor.

curve at half step shown in Figure 12 was chosen as the motors for the laser turret. From the pull-out torque curve in Figure 12, the maximum rated speed of the motor is given as 13 000 points per second at half step 0.9° which is

$$13\,000 \div \frac{360}{0.9} = 32.5\,\mathrm{RPM}, \tag{25}$$

with a pull-out torque greater than $100\,\mathrm{g\,cm} = 9.8 \times 10^{-3}\,\mathrm{N\,m}$. The effective torque of the stepper motor at maximum speed with 32 microsteps is

$$\tau_{eff} = 9.8 \times 10^{-3}\,\mathrm{N\,m} \times 4.91\% = 4.81 \times 10^{-4}\,\mathrm{N\,m}. \tag{26}$$

The effective torque of the stepper motor at maximum speed is greater than the required torque for the laser turret, therefore the NEMA8 stepper motor is suitable for the laser turret and was also chosen for its compactness which can be seen in Figure 13.

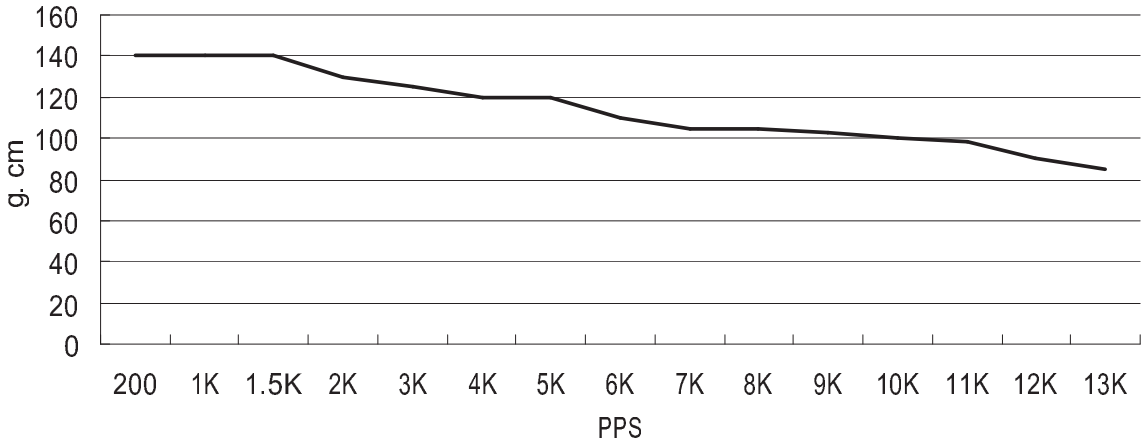
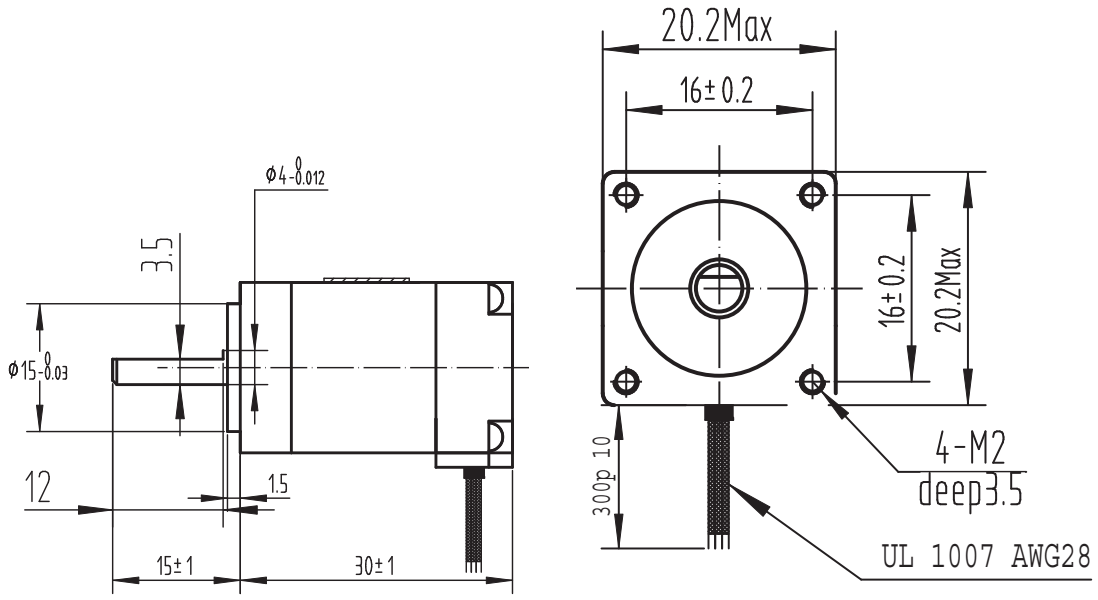


Figure 12.
Pull-out torque curve of the stepper motor at half step with 24 V and 0.6 A.

e. Laser selection and activation circuit
A low-powered 5 mW red laser with adjustable focus was chosen for the laser turret. The laser has an operating voltage range of 2.8 – 5.2 V. A low-powered laser was chosen for safety. The adjustable focus enabled the laser beam to be focused to a small spot size on the back wall of the mosquito enclosure. The laser must be red since the image processing will be done using on the red channel of the red, green, blue (RGB) video frame.

The 3.3 V GPIO of the Nvidia Jetson Nano cannot supply enough current to power the laser. A transistor circuit was designed to enable the laser to be turned on and off from the Nvidia Jetson Nano GPIO.

f. Laser turret computer aided design (CAD) The laser turret was designed in CAD using Autodesk Fusion 360. The laser turret was designed to be mounted on the bracket shown in Figure 7. The main component and subcomponents of the laser turret can be seen in Figure 14 and Figure 15 respectively.



(a) Side view. **(b) Front view.**

Figure 13.
Dimensions of the NEMA8 stepper motor.

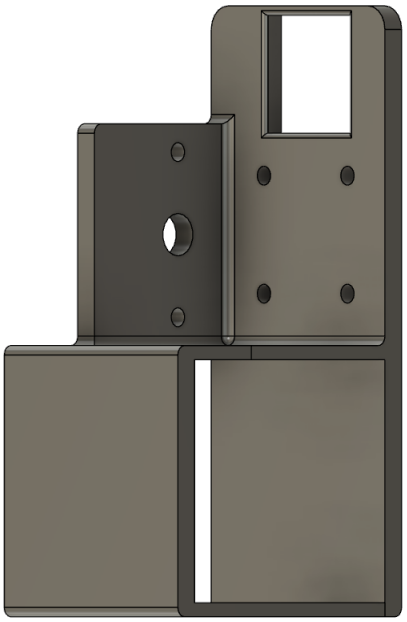
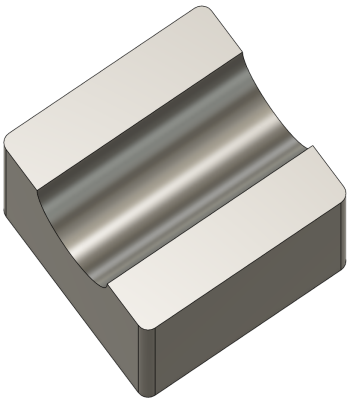
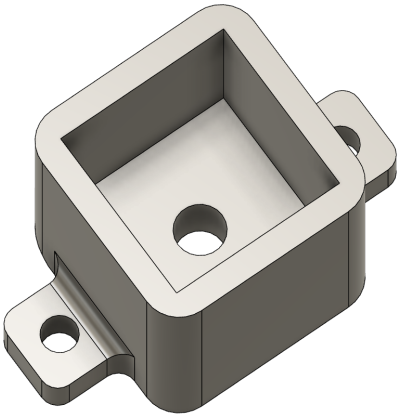


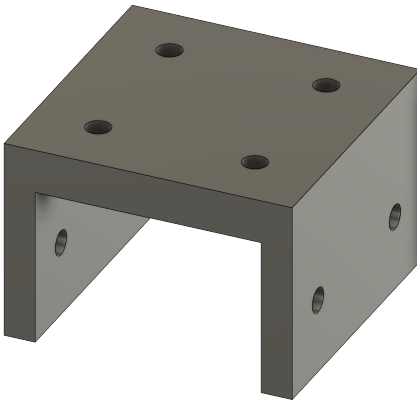
Figure 14.
Laser turret main component.



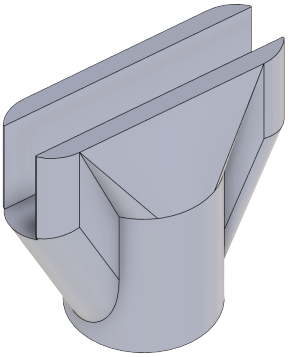
(a) Laser mount.



(b) Motor mount.



(c) Laser turret clamp.



(d) Mirror holder.

Figure 15.
Laser turret subcomponents.

3.7 Software design

Mapping between pixels and real distance. Mosquito detection and tracking. Laser detection and tracking. proportional-integral-derivative (PID) controller. Feedback for turret. Distinguishing between laser reflections.

3.7.1 Mosquito and laser detection

It was assumed that the mosquitoes would be the only dark blobs in the enclosure and that the laser and its reflections would be the only bright blobs in the enclosure. The video frame will be cropped such that only the white back surface of the enclosure is visible. To reduce the computational complexity of the image processing only the red channel of the RGB frame will be used, hence, the necessity of the red laser.

The mosquito detection was designed to detect dark blobs in the enclosure and would not be able to distinguish between mosquitoes and other dark blobs. The laser detection was designed to detect bright blobs in the enclosure and would not be able to distinguish between the laser's reflections and other bright blobs. The mosquito and laser detection processes are similar in nature. The basic detection process flow is shown in Figure 16. The red channel of

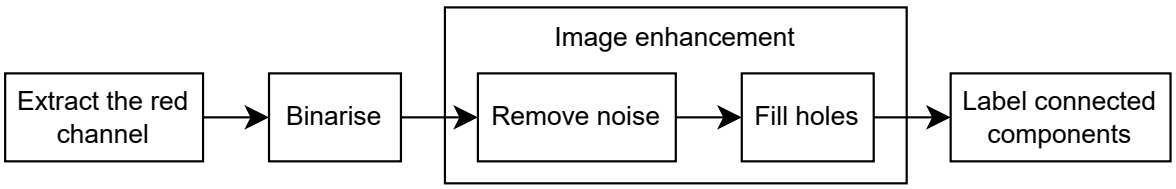


Figure 16.
Detection process flow.

the RGB frame will be extracted, which will be referred to as the frame for the rest of this discussion. The difference between the mosquito and laser detection occurs in the binarisation step. A copy of the frame is made before binarising so that both the mosquito and laser detection can be performed since the frame will be overwritten in the image processing steps in pursuit of efficiency.

a. Binarisation
The frame will be binarised using a threshold value. The exposure of the camera will be fixed and the lighting in the mosquito enclosure will be controlled thus the threshold value will be fixed. To detect mosquitoes the frame will be binarised with a less than threshold since the mosquitoes will be darker than the white background. To detect the laser's reflections the frame will be binarised with a greater than threshold since the laser and its reflections will be brighter than the white background. The ideal result of the binarisation is an image where the mosquitoes and the laser's reflections are the only foreground pixels in their respective copies of the frame. This will not be the case since the mosquitoes and the laser's reflections can not

be perfectly isolated using a threshold value in frame that will contain noise from the camera sensor.

b. Image enhancement

The binarised frames are subject to noise, holes, and erroneously joint or disjointed sections. This can be resolved using morphological operations. The goal of the morphological operations is to ensure that all the foreground pixels that correspond to a detection are connected, that there are no foreground pixels that do not correspond to a detection, and that there are no erroneously joint or disjoint foreground pixels. There must be a one to one correspondence between the groups of connected foreground pixels and the objects that must be detected. The connectivity of pixels will be explained in paragraph c. Morphological operations work by passing a structuring element over the pixels in an image and performing a logical operation on the pixels in the image that are covered by the structuring element.

The noise and erroneously joint sections are removed using the morphological opening operation. Opening has been formally defined in subsubsection 3.3.3. The opening of A by B is the union of all translations of B that are completely contained in A as illustrated in Figure 4a. The holes and erroneously disjointed sections are removed using the morphological closing operation. Closing has been formally defined subsubsection 3.3.3. The closing of A by B is the complement of the union of all translations of B that do not overlap A as illustrated in Figure 4b.

The structuring element must be identical independent of orientation in a 2D image since the orientation of the mosquitoes will be unknown. The structuring element will therefore be a disc. The size of the disc will be adjusted independently for the opening and closing operations for both the mosquito and laser detection.

c. Connected components labelling

From the enhanced binary frames the co-ordinates of the connected foreground pixels must be obtained. This is done using connected components labelling (CCL). CCL is the process of assigning a label to each foreground pixel in a binary image such that connected pixels have the same label. The connectivity of pixels are defined using either four or eight connectivity. Four connectivity defines the neighbourhood of a pixel as the pixels to the left, right, top, and bottom of the pixel. Eight connectivity defines the neighbourhood of a pixel as the pixels to the left, right, top, bottom, top left, top right, bottom left, and bottom right of the pixel. The CCL connectivity is shown in Figure 17.

(a) Four connectivity. **(b) Eight connectivity.**

Figure 17.
Connected components labelling connectivity.

Various algorithms exist to perform CCL. Two of the most commonly used algorithms are the two-pass algorithm and the breadth-first single-pass algorithm. The two-pass algorithm is generally faster for images with large connected components and the breadth-first single-pass algorithm is generally faster for images with small connected components. The nature of the mosquito and laser detection is such that the connected components will be small, thus the breadth-first single-pass algorithm was chosen. The breadth-first single-pass algorithm is presented in Algorithm 1.

Algorithm 1 Breadth-First Single-Pass Connected Component Labelling

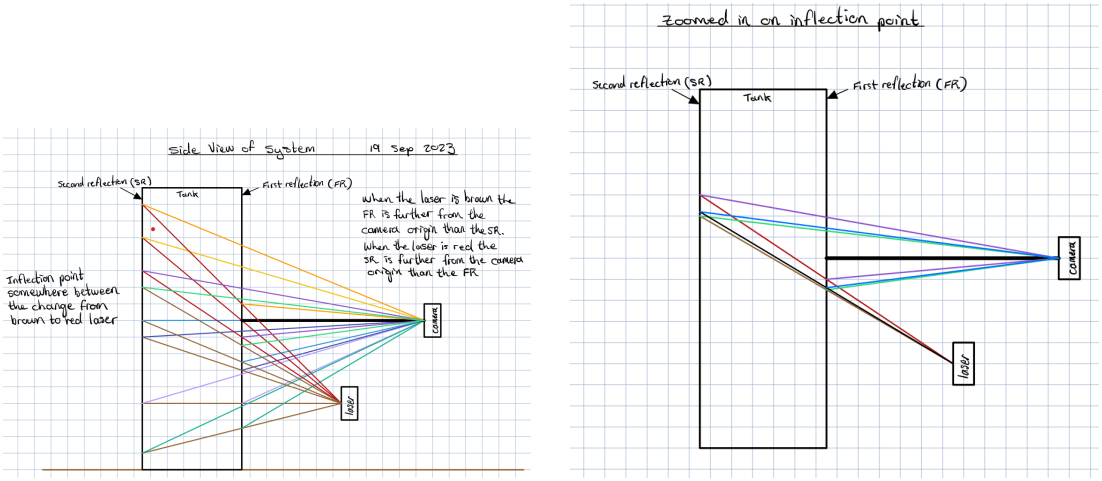
```
1:  procedure SINGLEPASSCCL(image)
2:      rows, cols ← dimensions of image
3:      label ← 1
4:      labels ← 2D array of zeros of size rows × cols
5:      for y ← 0 to rows − 1 do
6:          for x ← 0 to cols − 1 do
7:              if image[y][x] = 1 and labels[y][x] = 0 then
8:                  ExploreBlob(image, labels, x, y, label)
9:                  label ← label + 1
10:             end if
11:         end for
12:     end for
13: end procedure
14:
15: procedure EXPLOREBLOB(image, labels, x, y, label)
16:     queue ← empty queue
17:     Enqueue(queue, (x, y))
18:     while not empty(queue) do
19:         (x, y) ← Dequeue(queue)
20:         if x < 0 or x ≥ cols or y < 0 or y ≥ rows or image[y][x] = 0 or labels[y][x] > 0
    then
21:             continue
22:         end if
23:         labels[y][x] ← label
24:         for each neighbor (nx, ny) of (x, y) do
25:             Enqueue(queue, (nx, ny))
26:         end for
27:     end while
28: end procedure
```

3.7.2 Distinguishing between the laser's reflections

The laser must shine through the front glass of the mosquito enclosure. This results in various reflections of the laser beam. There are three predominate reflections that can not be distinguished using a brightness threshold. These reflections are:

- The reflection of the laser beam off the front glass at the origin of the laser turret, resulting from scattered light off the turret mirrors.
- The reflection of the laser beam incident on the front glass.
- The reflection of interest, which is the reflection of the laser beam incident on a mosquito or the back surface of the enclosure.

The reflection off the front glass at the origin of the laser turret remains stationary since the laser turret itself is stationary with only the angles of the mirrors changing. The remaining two reflections are distinguished using the geometric properties of the system positioning. The yz-plane of the system is shown in Figure 18. The laser turret is slightly below the camera which is exaggerated in Figure 18 for illustration purposes. It can be seen in Figure 18 that if both detections are at or below the camera origin the reflection of interest is closer to the camera origin. If both reflections are at or above the camera origin then the reflection of interest is further from the camera origin. If the reflections are on either side of the camera origin the reflection of interest is the reflection above the camera origin. This logic is easily extended to three dimensions by considering the yz-plane and xz-plane independently.



(a) Laser reflections.

(b) Laser reflections zoomed in on the inflection point.

Figure 18.
Distinguishing between the laser's reflections.

3.7.3 Mosquito tracking

The mosquito tracking was performed using the simple online and realtime tracking (SORT) algorithm. The SORT algorithm is based on the tracking-by-detection paradigm, which means it relies on an external object detector. The object detector used was the mosquito detection algorithm described in subsubsection 3.7.1. The SORT algorithm associates the detections across frames to create tracks.

a. Tracks

The mosquito tracks are created using the Kalman filter. The theoretical background of Kalman filter is discussed in subsubsection 3.3.4. The kinematic equation representing the flight of a mosquito must be defined to determine the state space of the Kalman filter. The kinematic equation used to model the flight of a mosquito is

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \dot{\mathbf{x}}_{k-1}\Delta t + \frac{1}{2}\ddot{\mathbf{x}}_{k-1}\Delta t^2, \tag{27}$$

where $\mathbf{x}_k$ is the position vector defined by the 2D pixel co-ordinate of a mosquito $\begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}$. A constant velocity model with acceleration as noise will be used since mosquitoes have an erratic flight pattern. Therefore, the set of differential equations describing the state space of the Kalman filter is

$$\begin{gathered} \mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k \\ \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix}, \end{gathered} \tag{28}$$

where $\mathbf{u}_k$ is the input vector, which is the acceleration of the mosquito according to its kinematic flight model defined in Equation 27. However, this acceleration is unknown and will be compensated for by the process noise $\mathbf{w}_k$, thus $\mathbf{u}_k = \mathbf{0}$. The process noise $\mathbf{w}_k$ is assumed to be drawn from a zero mean multivariate normal distribution $\mathcal{N}$ with covariance $\mathbf{Q}_k$. Hence, $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$, thus, $\mathbb{E}[w_k] = 0$ and $\mathbb{E}[w_k w_k^{\mathrm{T}}] = \mathbf{Q}_k$. The process noise covariance matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = \begin{bmatrix} \sigma_x^2 & 0 & \sigma_x\sigma_{\dot{x}} & 0 \\ 0 & \sigma_y^2 & 0 & \sigma_y\sigma_{\dot{y}} \\ \sigma_{\dot{x}}\sigma_x & 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & \sigma_{\dot{y}}\sigma_y & 0 & \sigma_{\dot{y}}^2 \end{bmatrix}, \tag{29}$$

where σ_x and $\sigma_{\dot{x}}$ are the standard deviations of the position and velocity, respectively. The standard deviation of the position is defined as the standard deviation of the acceleration $\sigma_{\ddot{x}}$ multiplied by $\frac{1}{2}\Delta t^2$ since this is the effect that the acceleration will have on the position as shown in Equation 27. Similarly, the standard deviation of the velocity is defined as $\Delta t\sigma_{\ddot{x}}$. Therefore, the process noise covariance can be written as

$$\mathbf{Q} = \sigma_{\ddot{x}}^2 \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix}. \tag{30}$$

The position of a mosquito is the only component of the state that is measured by the detection system. Thus, the observation matrix $\mathbf{H}$ is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{31}$$

The measurement noise covariance matrix $\mathbf{R}$ is

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}, \tag{32}$$

where σ_x^2 and σ_y^2 are the variances of the position detected. The initial state covariance matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \dot{x}_{max}^2 & 0 \\ 0 & 0 & 0 & \dot{y}_{max}^2 \end{bmatrix}. \tag{33}$$

The Kalman filter is initialised with the position of the mosquito as the initial state $\mathbf{x}_0$. The position of the mosquito in the next frame $\hat{\mathbf{x}}_{k|k-1}$ is predicted using the adapted predict equation of the Kalman filter $\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_k \mathbf{x}_{k-1|k-1}$. This prediction will be used in the association step before the update step. The update step of the Kalman filter is performed using the update equation of the Kalman filter $\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}_k \mathbf{x}_{k|k-1} \right)$, where $\mathbf{K}_k$ is the Kalman gain and $\mathbf{z}_k$ is the measurement vector. The measurement vector $\mathbf{z}_k$ is the position of the mosquito detected in the current frame.

b. Association

The association of tracks and detections will be done using the Hungarian algorithm, which is an optimised optimal assignment algorithm. The Hungarian algorithm has a time complexity of $O(n!)$ compared to the time complexity naïve approach $O(n^3)$. The Hungarian algorithm works as follows:

Generate a cost matrix.

Represent the problem with a square $n \times n$ matrix $\mathbf{C}$ where the element $\mathbf{C}_{i,j}$ represents the cost of assigning track i to detection j. The cost function is the euclidean distance between the predicted location and the detected location of the mosquito.

Row and column reduction.

Subtract the minimum value in each row from each element in the row. Perform the same operation for each column. This will produce a matrix with at least one zero in each row and column.

Cover zeros and check for optimality.

Cover all the zeros in the matrix using the minimum number of horizontal and vertical lines. If the minimum number of covering lines is n, an optimal assignment exists among the zeros, and the algorithm can proceed to the assignment phase. If not, you need to adjust the matrix and repeat from this step.

Adjust the matrix.

Find the smallest element that is not covered by any line. Subtract this element from all uncovered elements, and add it to all elements at the intersections of the covering lines. Return to *cover zeros and check for optimality*.

Assignment.

Once an optimal assignment is found, read the assignment from the zeros in the matrix. ~~Each row will have exactly one zero, and each column may have one zero. If a column has no zeros, the track corresponding to that column is unassigned. If a column has more than one zero, the track corresponding to that column is assigned to the detection corresponding to the zero that is covered by a star. If a row has more than one zero, the~~

~~track corresponding to that row is assigned to the detection corresponding to the zero that is covered by a star.~~

3.7.4 Laser turret control system

The laser turret control system is a closed-loop PID controller. A PID controller is represented by the following equation

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}, \tag{34}$$

where $u(t)$ is the control signal, $e(t)$ is the error signal, K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain. The goal of the laser turret control system is to position the laser beam at the target co-ordinates. The target co-ordinates is defined by the mosquito detection and tracking system in pixels. The stepper motors are controlled in steps, thus the target pixels and must be mapped to the number of steps required to move the laser beam to the target co-ordinates. The steps are defined in reference to the origin of the laser defined in subsubsection 3.5.5, which is the position when the laser beam is perpendicular to the *xy*-plane of the mosquito enclosure. The block diagram of the laser turret control system can be seen in Figure 19. The two axes of the laser turret are controlled independently each having their own PID controller. The PID controller is implemented in software and the control signal is sent to the stepper motor drivers using GPIO.

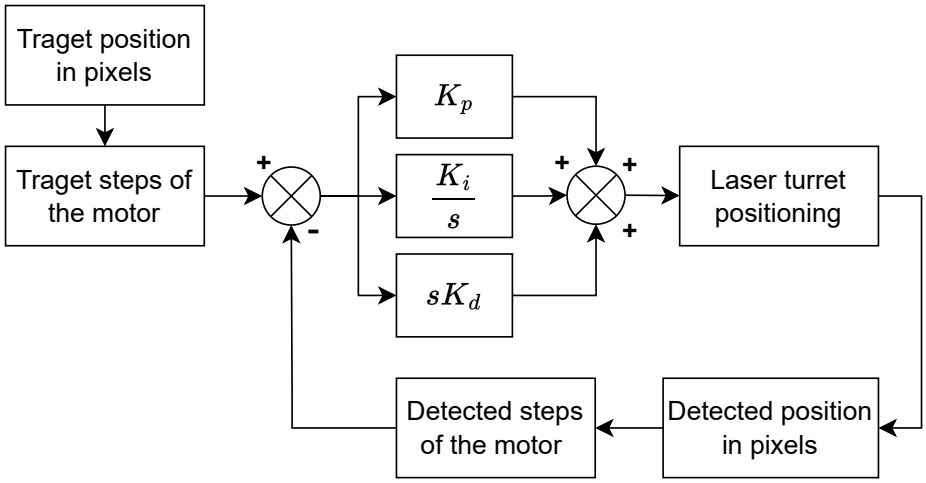


Figure 19.
Laser turret control system.

3.8 Software implementation and optimisation

Only using red channel, GPU, low resolution, etc. GPIO interfacing with stepper motor driver.

The Raspberry Pi camera v2 was chosen. The low resolution was 700 x 300 was chosen to reduce computational complexity.

3.8.1 *Image processing*

The binarisation and image enhancement processes described in subsubsection 3.7.1 are suitable for parallelisation. Therefore, custom CUDA kernels were written to perdform these operations on GPU.

```
__global__ void erosion(uint8_t* input,
                        uint8_t* output,
                        uint8_t* struct_elem,
                        int struct_elem_radius) {
  int x = blockIdx.x * blockDim.x + threadIdx.x;
  int y = blockIdx.y * blockDim.y + threadIdx.y;

  if (x < d_COLS && y < d_ROWS) {
    int min_val = 255;
    for (int i = -struct_elem_radius; i <= struct_elem_radius; i++) {
      for (int j = -struct_elem_radius; j <= struct_elem_radius; j++) {
        if (y + i >= 0 && y + i < d_ROWS && x + j >= 0 && x + j < d_COLS)
        {
          if (struct_elem[(i + struct_elem_radius) *
                          (2 * struct_elem_radius + 1) +
                          j + struct_elem_radius] == 1)
          {
            int idx = (y + i) * d_COLS + (x + j);
            min_val = min(min_val, (int)input[idx]);
          }
        }
      }
    }
    output[y * d_COLS + x] = min_val;
  }
}
```

Listing 1.
Erosion GPU kernel.

3.8.2 *Laser turret control*

The laser turret control consists of various independent subcomponents that required meticulous attention to detail to successfully integrate. The integration will be discussed in subsection 3.9. The focus in this section will be on the subcomponents.

a. Interfacing with stepper motor drivers

The stepper motors drivers were connected to the Nvidia Jetson Nano GPIO according to the diagram in Figure 20. The motors are enabled by setting the `ENABLE PIN` to a logical high. The direction of rotation is determined by the logic level of the `DIRECTION PIN`. The stepper motors are driven by pulsing the `STEP PIN` at the desired frequency for the desired number of steps.

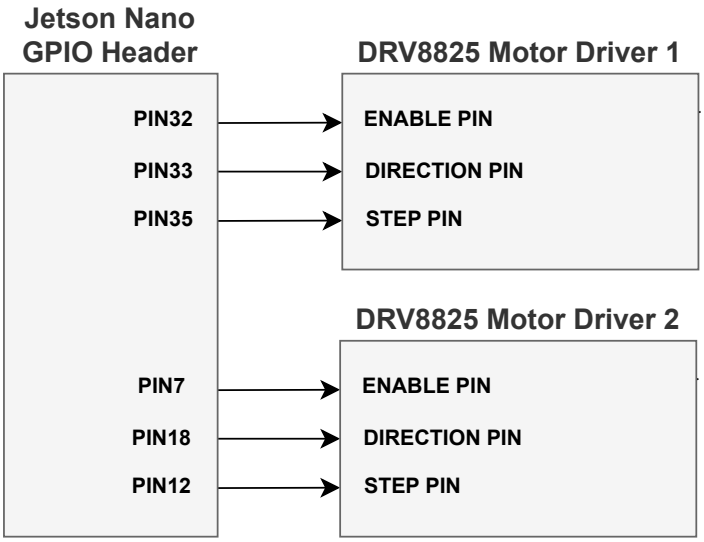


Figure 20.
Stepper motor driver connections.

Algorithm 2 Stepper Motor Control

1: $ENABLE\ PIN \leftarrow$ logical high
2: $DIRECTION\ PIN \leftarrow direction$
3: **for** $step \leftarrow 0$ to $steps - 1$ **do**
4: $\quad STEP\ PIN \leftarrow$ logical high
5: $\quad$ delay$(period)$
6: $\quad STEP\ PIN \leftarrow$ logical low
7: $\quad$ delay$(period)$
8: **end for**

b. Converting pixels to steps

The laser turret control system must be able to position the laser beam at any pixel in the enclosure. The laser turret control system must therefore be able to convert pixels to steps. The conversion from pixels to steps is done using the following equation

$$\text{steps} = \frac{\text{pixels}}{\text{pixels per step}}, \tag{35}$$

3.9 Final system integration and testing

Real-time multi threading.

4. Results

4.1 Summary of results achieved

Intended outcome	Actual outcome	Location in report
Core mission requirements and specifications		
The system must track mosquitoes in a mosquito enclosure and illuminate a mosquito every 5 seconds.	The system tracks mosquitoes in the enclosure and illuminates a mosquito when it is stationary or moves predictably.	**??**
The laser must be able to illuminate a set point within 2 seconds accurate to within 1 millimetre.	Yes	sec
The system must be able to detect mosquitoes with a 90% accuracy and 5% false positive rate. The detection must be updated at least every 500 milliseconds.	Weet nie. The detection is updated every 500 milliseconds.	sec
The system must be able to track mosquitoes with 90% accuracy of correct association between frames after 5 seconds.	Weet nie.	sec
Field condition requirements and specifications		
Mosquitoes should be in an enclosure with controlled lighting and white lining on all the sides except the front facing side. The enclosure should be at least 1 metre wide.	The enclosure has LEDs to control the lighting and white lining on all the sides except the front facing side. The enclosure is 0.9 metres wide.	sec
If mosquitoes cannot be obtained a suitable substitute will be used. The substitute will be a similar flying insect.	Mosquitoes and similar flying insects were obtained. Dead mosquitoes were also used.	sec

Table 4.
Summary of results achieved.

4.2 Qualification tests

Qualification test 1: Test of tracking and illuminating a mosquito

Objectives of the test or experiment
The objective of this experiment is to determine whether the system can track and illuminate a mosquito in the mosquito enclosure. The requirement states that the system must track mosquitoes in the enclosure and illuminate a mosquito every 5 seconds.

Equipment used
The Nvidia Jetson Nano was used as the embedded platform to control the system. The Raspberry Pi Camera Module V2 was used to capture the video feed. The laser turret was used to position the laser. The Nvidia Jetson Nano was connected to user peripherals for user input and output.

Test setup and experimental parameters
Mosquitoes were simulated with a small black dot attached to a white stick. The stick was inserted into the controller and moved around.

Results or measurements

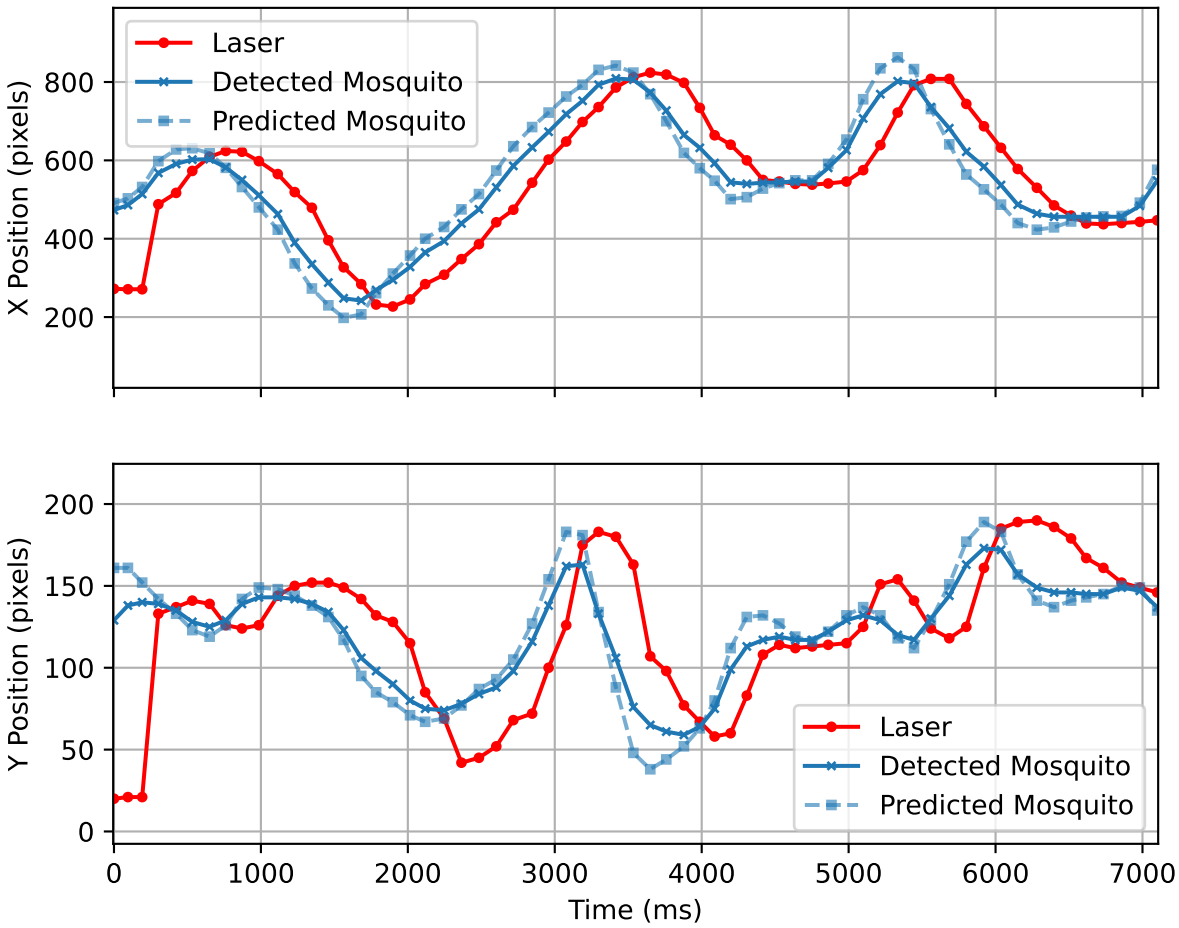


Figure 21.
q1 1mos detection 5fps

Observations

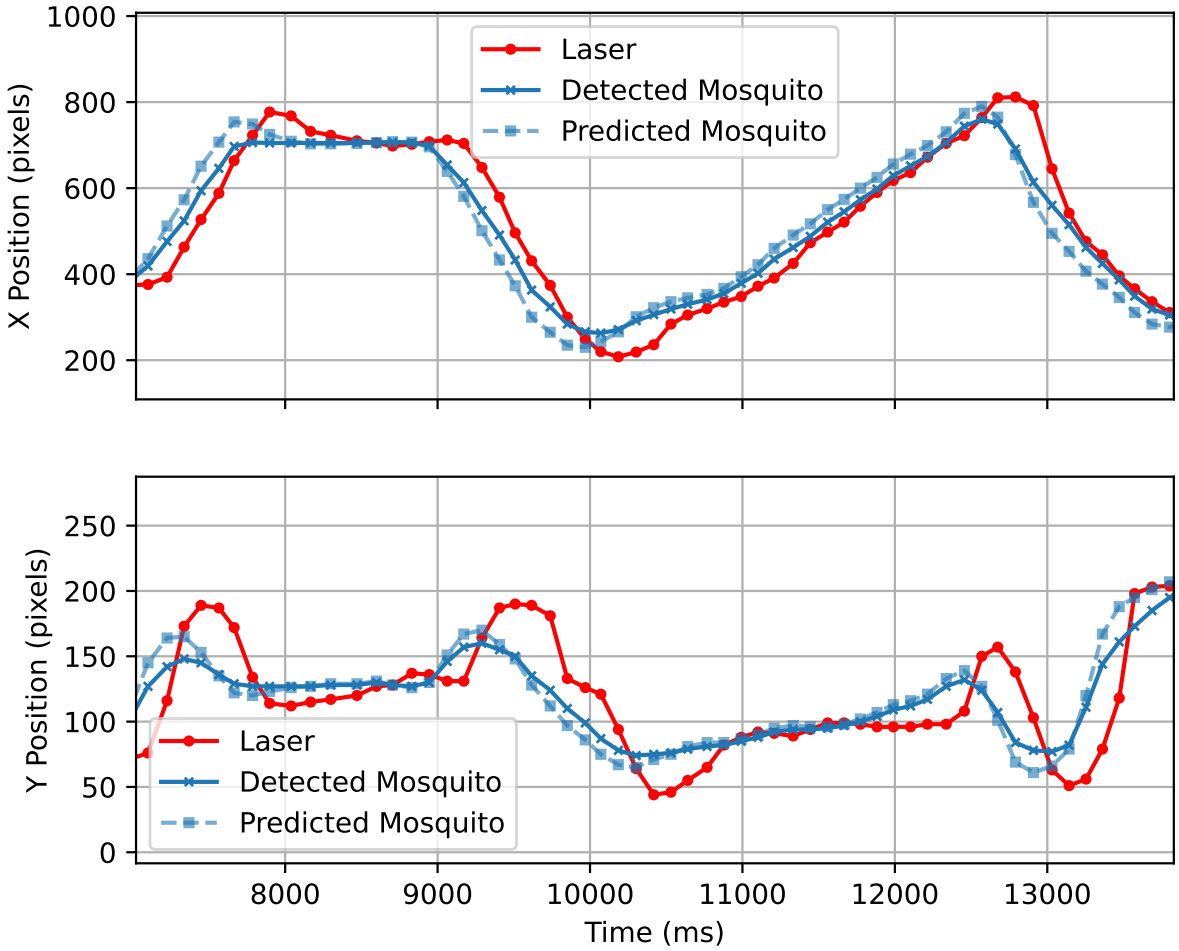


Figure 22.
q1 1mos prediction 5fps

Qualification test 2: Measurement of the time taken for the laser to reach a set point

Objectives of the test or experiment
The objective of this experiment is to measure the time it takes for the laser to reach a set point. The requirement states that the laser must be able to illuminate a set point within 2 seconds accurate to within 1 millimetre.

Equipment used
The Nvidia Jetson Nano was used as the embedded platform to control the system. The Raspberry Pi Camera Module V2 was used to capture the video feed. The laser turret was used to position the laser. The Nvidia Jetson Nano was connected to user peripherals for user input and output.

Test setup and experimental parameters
Multiple iterations was performed. The experiment will be performed using both a single step input and a step input with a ramp.

For a few runs align the set point with a circle or square with radius of 1mm and take a picture of the laser setting point to see if it is accurate to 1mm. Also measure the amout of pixels for 1mm at a few points then this can be used for the rest of the results.

1. The laser is manually positioned in an arbitrary location in the mosquito enclosure.
2. The set point is set to an arbitrary location in the mosquito enclosure.
3. The system is set to set the set point with either a step input or a ramp function.
4. The turret is started.
5. The time and position of the laser is recorded for each frame captured by the system until the laser reaches a settling point.
6. The results are saved, and the experiment is repeated.

Results or measurements

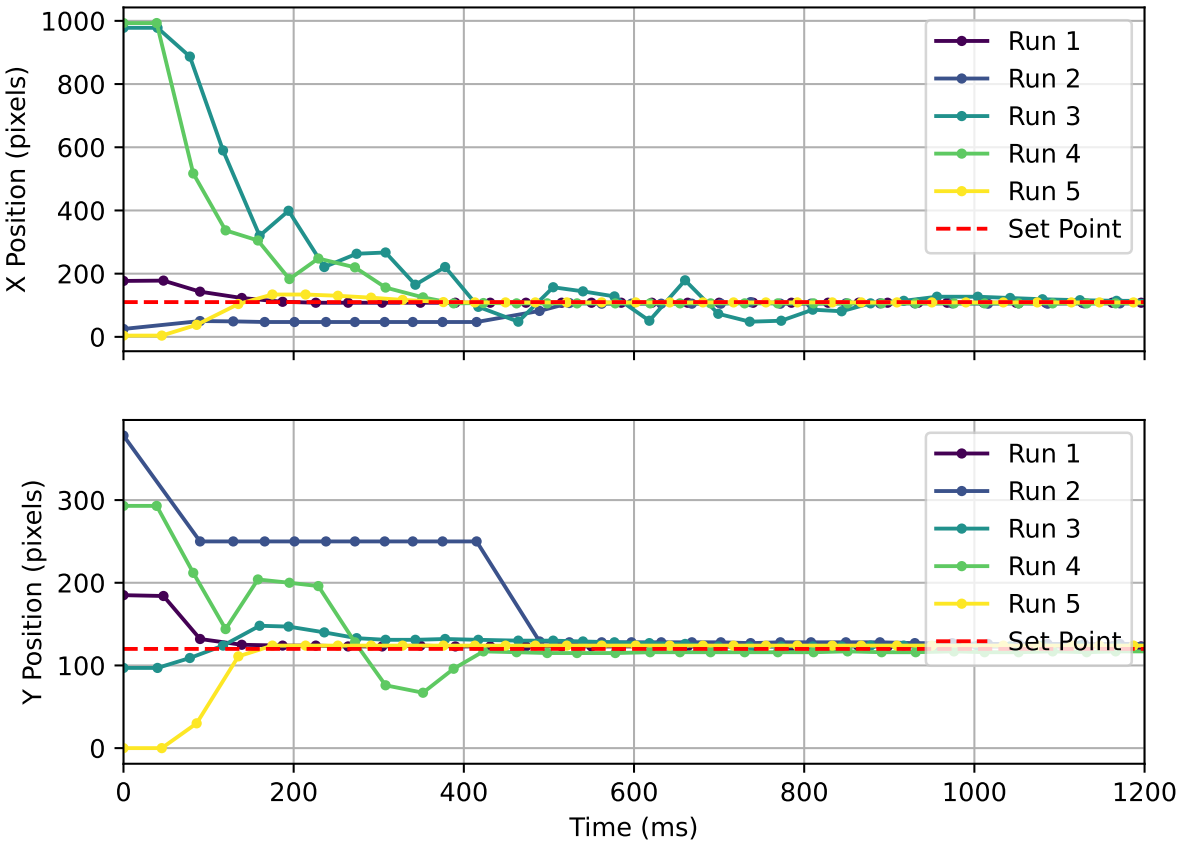


Figure 23.
q2

Observations

Qualification test 3: Test of mosquito detection

Objectives of the test or experiment
The objective of this experiment is to determine whether the system can detect mosquitoes in the mosquito enclosure. The requirement states that the system must be able to detect mosquitoes with a 90% accuracy and 5% false positive rate. The detection must be updated at least every 500 milliseconds.

Equipment used
The Nvidia Jetson Nano was used as the embedded platform to control the system. The Raspberry Pi Camera Module V2 was used to capture the video feed. The Nvidia Jetson Nano was connected to user peripherals for user input and output.

Test setup and experimental parameters
Keep the mosquitoes in the tank constant and vary the threshold for detection and plot the accuracy and false positive rate against the threshold. Either two plots or one plot with two y-axes or a 3d plot.

Create a function that takes in the known number of mosquitoes in the enclosure and saves this, the threshold, and the number of mosquitoes detected for each frame in a file. The first line must be the known num mossies and threshold. The second line must be the number of mosquitoes detected for each frame seperated by commas. The function must also save the average frame rate. The function must run for 1 min.

To determine the update frequency of the detection the following experiment was performed.

1. The mosquito enclosure was populated with a varying amount of mosquitoes.
2. The system was left to run for x time.
3. The average frame rate was recorded for the duration of the experiment.
4. The experiment was repeated for different amounts of mosquitoes in the enclosure and with tracking enabled and disabled.

To determine the accuracy of the detection the following experiment was performed.

1. The mosquito enclosure was populated with a varying amount of mosquitoes.
2. The system was left to run for x time.
3. The number of mosquitoes detected was recorded for the duration of the experiment.
4. The experiment was repeated for different amounts of mosquitoes in the enclosure.

Kan ek net dooie muskiete plak in die tenk? Kan ek hulle plak of my stokkies en rotbeweeg? Kan ek net kolle gebruik? As ek regte muskiete vang dan vlieg hulle die heel tyd in hoekies waar my kamera hulle nie kan sien nie of hulle gaan sit still.

Results or measurements

Observations

Qualification test 4: Test of mosquito tracking

Objectives of the test or experiment
The objective of this experiment is to determine whether the system can track mosquitoes in the mosquito enclosure. The requirement states that the system must track mosquitoes in the enclosure with 90% accuracy after 5 seconds.

Equipment used
The Nvidia Jetson Nano was used as the embedded platform to control the system. The Raspberry Pi Camera Module V2 was used to capture the video feed. The Nvidia Jetson Nano was connected to user peripherals for user input and output.

Test setup and experimental parameters
Compare number of actual mosquitoes to number of tracked mosquitoes.

Plot predicted position vs actual position.

Results or measurements

Observations

5. Discussion

5.1 Critical evaluation of the design

5.1.1 Interpretation of results

5.1.2 Critical evaluation

5.1.3 Unsolved problems

5.1.4 Strong points of the design

5.1.5 Expected failure conditions

5.2 Considerations in the design

5.2.1 Ergonomics

5.2.2 Health and safety

5.2.3 Environmental impact

5.2.4 Social and legal impact

5.2.5 Ethics clearance

6. Conclusion

6.1 Summary of the work completed

6.2 Summary of the observations and findings

6.3 Contribution

6.4 Future work

7. References

[1] W. H. Organisation. (2020, 12) The top 10 causes of death. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

[2] P. Hurtik, D. Číž, O. Kaláb, D. Musiolek, P. Kočáek, and M. Tomis, "Software for visual insect tracking based on f-transform pattern matching," in IEEE Second International Conference on Data Stream Mining & Processing, Lviv, Ukraine, 2018.

[3] Z. Khan, T. Balch, and F. Dellaert, "Efficient particle filter-based tracking of multiple interacting targets using an mrf-based motion model," in Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Las Vegas, Nevada, USA, 2003.

[4] W. Liang, H. Wang, and H. Krim, "A behaviour-based evalutation of product quality," in International Conference on Acoustics, Speech, and Signal Processing, 2016.

[5] Y. Bao and H. Krim, "Video tracking of insect flight path: Towards behavioral assessment," in IEEE, 2018.

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online realtime tracking," Queensland University of Technology, University of Sydney, Tech. Rep., 2017.

[7] "Polarization effects of galvos in SLA 3D printing," https://henryquach.org/galvopolarization.html, accessed: 2023-10-14.

Part 4. Appendix: technical documentation

HARDWARE part of the project

Record 1. System block diagram

Record 2. Systems level description of the design

Record 3. Complete circuit diagrams and description

Record 4. Hardware acceptance test procedure

Record 5. User guide

SOFTWARE part of the project

Record 6. Software process flow diagrams

Record 7. Explanation of software modules

Record 8. Complete source code

Complete code has been submitted separately on the AMS.

Record 9. Software acceptance test procedure

Record 10. Software user guide

EXPERIMENTAL DATA

Record 11. Experimental data