NWU®

FACULTY OF ENGINEERING

AUTONOMOUS PAINTBALL SENTRY TURRET

EERI474

by:
Dewald Krynauw 26013835

Submitted in pursuit of the degree

BACHELOR OF ENGINEERING

In

COMPUTER AND ELECTRONIC ENGINEERING

North-West-University Potchefstroom Campus

Supervisor: Prof J. Holm
Potchefstroom
2018

Abstract

The Autonomous Paint-ball Sentry Turret is a legal defence system that can efficiently remove intruders from the owner's property. The system uses motion detection and tracking algorithms to locate moving targets, and fires either autonomously or manually. An effective motion detection system has been developed using Python along with OpenCV on a Raspberry Pi 3 Single Board Computer that controls three servo motors (pan, tilt, trigger). Motion tracking was implemented using the periodic background estimation subtraction algorithm.

The Turret had a hit rate rating of up to 40% on live humans and up to 60% on controlled non-human tests. Hardware and software was optimised for simplicity and future updates.

Keywords - Background Subtraction, OpenCV, Turret, Raspberry Pi, Paintball.

Declaration

I, **Dewald Krynauw**, declare that this report is a presentation of my own original work.

Whenever contributions of others are involved, every effort was made to indicate this clearly, with due reference to the literature.

No part of this work has been submitted in the past or is being submitted, for a degree or examination at any other university or course.

Signed on, October 28, 2018, in Potchefstroom.

DD Krynauw

Acronyms

BS	Background Subtraction
PWM	Pulse Width Modulation
FPS	Frames Per Second
IR	Infra-Red
IP	Internet Protocol
I/O	Input output
GPIO	General Purpose Input Output

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

"*No weapon that is formed against you will prosper;*" - Isaiah 54:17

Crime rates in South Africa are on a raging increase, causing a dire need for personal security. More people are looking to alternative security solutions that would remove intruders from their property without putting themselves at risk.

In these modern times, an intruder cannot be shot or killed if he/she intrudes onto a property, because the law of South Africa, the owners of the property will be arrested and taken to jail. If the intruder can be removed from the property without killing him/her before a personal encounter even occurs would reduce the number of innocent people going to jail and unnecessary murder.

Table 1 below reveals some statistics on the crime rates in South Africa in 2017.

Table 1: Crime Statistics (South Africa - 2017)

Burglary at residential premises	246 654
Robbery at residential premises	22 343
Murder	19 016
Reported Farm Attacks	446
Farm Murders	49

http://www.crimestatssa.com

From the statistics, only 7% of all entries are with lethal intent. This means that 93% of these burglaries can be stopped. From Figure 1.1 the murder rates of South Africa is one of the largest in the world.

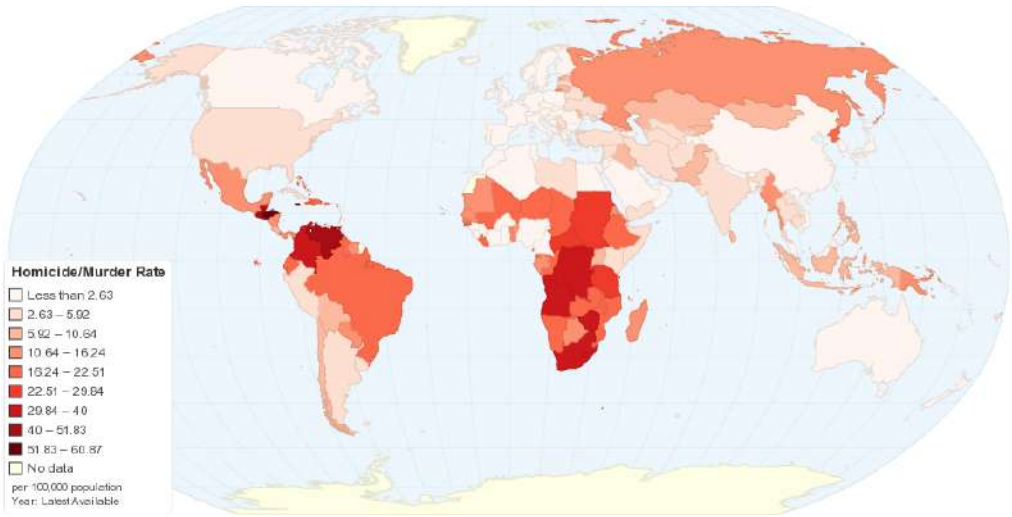


Figure 1.1: World Murder Rates

1.2 Problem Statement

A legal sentry turret defence system needs to be developed, to prohibit intruders from entering a property with ill intent.

1.3 Project Scope

For the sentry turret, the following aspects need to be covered:

- A physical frame that can holster a paintball gun needs to be designed and constructed
- Electronics to control the gun should be integrated into the frame
- Software must be written to control the electronics.

Regarding the Software, it should comply with the following:

- Manual Control
- Autonomous Control
- Motion Tracking

For the whole of the design and implementation process, the focus should be on the performance and hit rate of the turret.

Any project management related subjects like budget and time-management are not within the scope of this project.

1.4 Project Objectives

The objective of the paintball sentry turret is to shoot intruders autonomously that enter a property without the owner being there. Paintball bullets would ensure that the intruders are not killed but maimed and would rather flee than entering the property any further. This could help reduce the number of burglaries in the country.

1.5 Methodology

The following method will be used for constructing the mechanical structure:

1. Plan and design concept structure
2. Design a structure with all the electronics in place
3. Draw up a budget, double check the work
4. Purchase and construct the structure
5. Integrate, Test and improve the structure
6. Finalise details

The following method will be used for developing the software:

1. Plan and research motion tracking, and motor control.
2. Design operational flow of software
3. Do preliminary testing on possible algorithms
4. Design and integrate parts of software and electronics
5. Test and evaluate software
6. Integrate with structure
7. Test and evaluate again
8. Finalise details

1.6 Document Overview

This document covers all the section of the final report for the Paintball Sentry Turret. It begins with Section 1 as the introduction and continues to Section 2 as the literature study. Section 3 is the preliminary design. Section 4 is the detail design. Section 5 provides the implementation. Section 6 shows the testing and evaluation. Section 7 is the improvement and lastly, Section 8 is the conclusion.

The introduction covers, the background, problem statement, project scope, project objectives, methodology and document overview.

The literature review gives al the literature and research needed to complete the project. It consists of previous sentries, algorithms and hardware.

The preliminary design portrays the conceptual design. It illustrates the architecture, flow and trade-off studies.

Detail design includes all the planning and designing of the software, hardware and electronics. All the calculations and different algorithms are analysed.

The implementation is all the integration of parts and completed turret.

Testing and evaluation consist of preliminary testing in the design phase, as well as the tests conducted on the finished product. Both human and non-humans tests.

Improvements show all the apparent issues and possible resolutions to them. Another general improvement to enhance the system is also inserted.

Lastly, the conclusion wraps up the project gives an overview, conclusion and recommendations for future additions to the project.

2 Literature Review

2.1 Sentry Turrets

A Sentry Turret or Sentry Gun is a weapon that automatically aims and fires at targets that are detected with sensors. Where Sentry means to guard, and Turret means tower, i.e. a guard tower.

Over the past few decades, security and military applications have moved to more autonomous control. A few previously developed sentry guns are examined below.

2.1.1 Low-Cost 2-DOF Sentry Gun

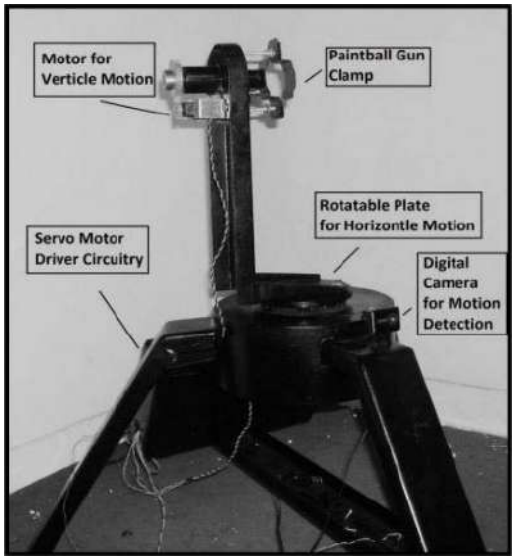


Figure 2.1: Example of 2-DOF Sentry Gun [1]

Figure 2.1 above proposed a low-cost solution to a sentry turret that is not military grade. The system tracks motion using an embedded microcontroller and MATLAB interface. Motion is detected using periodic background estimation subtraction.

2.1.2 Autonomous sentry gun platform

Figure 2.2: Example of Sentry Gun platform [2]

The "Nerf" sentry gun (Figure 2.2) is one of the most aesthetically pleasing designs with a very futuristic look and feel. It uses the **Pandaboard** for running the main system program. A custom shield termed: Kung Fu Shield, was designed for the actuation of pan and tilt servo motors. The software was developed in C for the ARM-cortex of the shield. A java program runs as a server on the Pandaboard that uses JavaCV, a wrapper for OpenCV. Tracking the targets is done using the *Lucas-Kanade* method to calculate the optical flow.

This Sentry gun is extremely complex, requires a large budget, very specific tools and a wide knowledge of different fields.

2.1.3 Hybrid Defence System

Figure 2.3: Example of Hybrid Defence System [3]

Figure 2.3 above has excellent software and electronic features. Not only does it have motion detection, but also face detection and recognition. They had two versions, one with a Raspberry Pi 2 and one with a mini ATX desktop. The two versions are to enable the software to run on an SBC and high-end computer. The Arduino Uno was used as the microcontroller to move the servos.

Their hardware included a Nerf gun that could be used semi-automatic. They had the very innovative idea of current induction which transfers power so that they do not have cables limiting the rotation.

They also used the Asus Xtion Pro Live, which has a 3D sensor, which takes stereoscopic images. This incorporates distance into the motion tracking. They used is the OpenCV MOG2 algorithm [10]. They also had a web interface that alerted users if there are intruders.

2.1.4 Semi-Autonomous Sentry Robot

Figure 2.4: Example of Semi - Autonomous Robot [4]

The Autonomous Robot (Figure 2.4) uses cheap infra-red sensors that track targets but come with many unanticipated challenges. It made use and an Arduino Pro, Xbee and motor controller. This a very inexpensive method of tracking target compared to LIDAR, but drawback include not being able to discern between friend or foe.

2.1.5 Motion Tracking Gun Turret

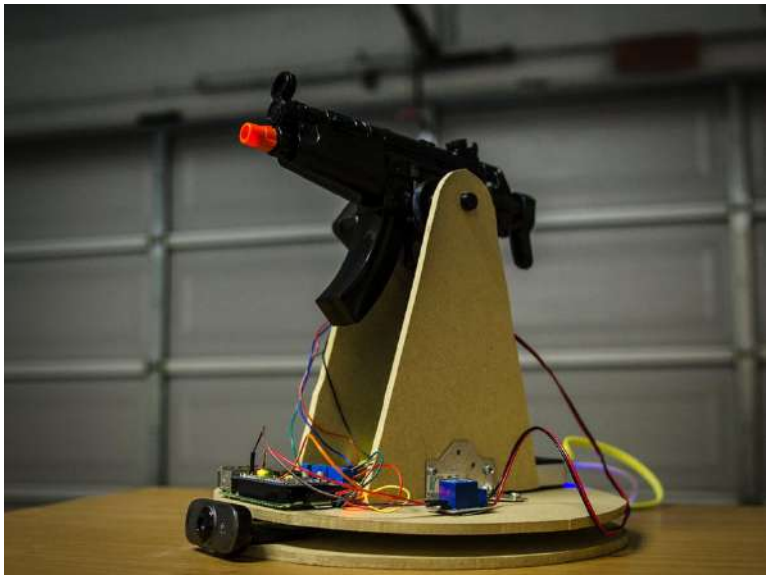


Figure 2.5: Example of Motion Tracking Turret by [5]

This Motion tracking turret (Figure 2.5) illustrates that it is also possible to use stepper motors for this use case, but also uses a Nerf gun with foam darts, which is for entertainment and not security.

The turret uses a Raspberry Pi 3 and a stepper shield to easily control the motors. The code is written in Python along with the OpenCV library to do computer vision.

2.1.6 Project Sentry Gun

Figure 2.6: Example of Gladiator 2 [6]

Project Sentry Gun is probably one of the most popular sentry guns on the internet today due to its open source code. The Gladiator 2 (Figure 2.6) uses a dedicated PC controlling an Arduino, along with a webcam as visual input. The software consists of two parts: The Computer Vision code and the Microcontroller code.

The Computer Vision code (GUI) is written in *Processing* and is a JAVA based application that sends commands to the Arduino via a serial interface. The GUI comes with extensive features including anticipation and smoothing, as well a colour tracking and calibration.

2.1.7 Sentry Turret with Dlib

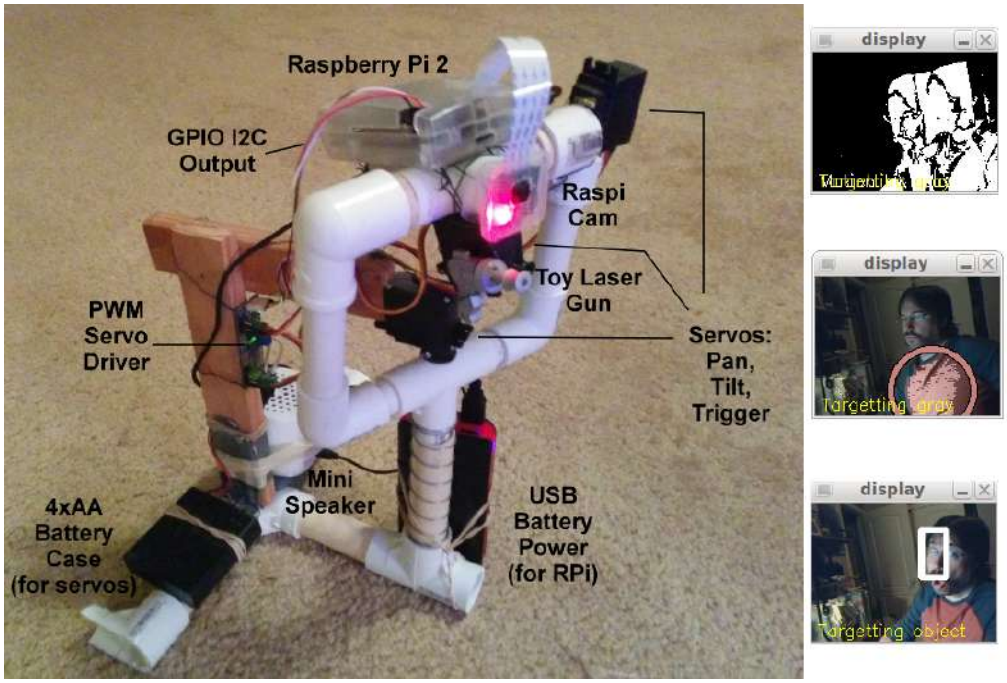


Figure 2.7: Example of Sentry Turret with Dlib [7]

Figure 2.7 is another great example of a sentry turret with open source code. It uses a Raspberry Pi, OpenCV, Python and Dlib. Dlib is an additional library that includes pre-trained neural models for facial recognition and motion tracking for humans.

2.2 Software

The software is responsible for locating a moving target with some visual/sensing input and then issue the right commands to aim the paintball marker in the direction of movement. The software is the largest contributing factor to the efficiency of the turret. This is due to the time it takes processing these large chunks of data and then calculating where the gun should move to and then compensate for the time lost.

2.2.1 Possible Visual Input Solutions

Motion detection can be done either by:

- *Infra Red (IR) Proximity Sensors* like in the low cost sentry robot [4]
- *Image Processing* like in the low cost motion UAVs [11]
- *LIDAR* (Radar with light)

LIDAR is a very good way of detecting motion but is extremely expensive and very difficult to implement.

The *IR sensors* provides a very low-cost way of detecting motion while also allowing for a moving sentry. But like [4] stated is that IR sensing cannot be made smart e.g. detecting whether something is an animal, pet or friendly neighbour.

Image Processing is mostly used with these types of applications, because it provides reliable and visual (to humans) feedback on what the system is doing. The field of image processing and available library packages already exist that makes development easier.

To know if Image processing is a viable option; more information is listed in the following section.

2.2.2 Motion Detection and Tracking Algorithms

There are three common methods to segment the motion of an object out of a video which are, background subtraction, frame differencing, and optical flow segmentation. The simplest being background subtraction [8].

Background Subtraction is used in the [1, 11, 3] sentries for motion detection algorithms to determine if there is motion in video. This algorithm does not include motion tracking, for this, another algorithm needs to be applied such as the *Kalman Estimates* [12].

The *Lucas-Kanade* method was used in the auto sentry gun platform [2] to calculate the optical flow of the frames to track and shoot targets. This is a very processing efficient method of object tracking [13], although there are not many open source libraries available showing how to implement it, making this method a high risk.

2.2.3 Background Subtraction

Background Subtraction (BS) in itself is very simple, a background frame is compared to the incoming frame and the difference in intensity gives the possible motion. However, the background frame should remain unchanged, but due to natural changes in lighting and movement, this becomes a problem.

There are multiple techniques addressing BS and the lighting problem, one of these techniques is Periodic Background Estimation Subtraction (Figure 2.8) as proposed by [8]. The algorithm describes how objects remaining static in the frame over a period become part of the background frame.

Other algorithms include Multi-Layer Background Subtraction Based on Color and Texture [10], and probably the algorithm that gained the most popularity Mixture of Gaussian (MoG) [14]. More information on this algorithm in the OpenCV section.

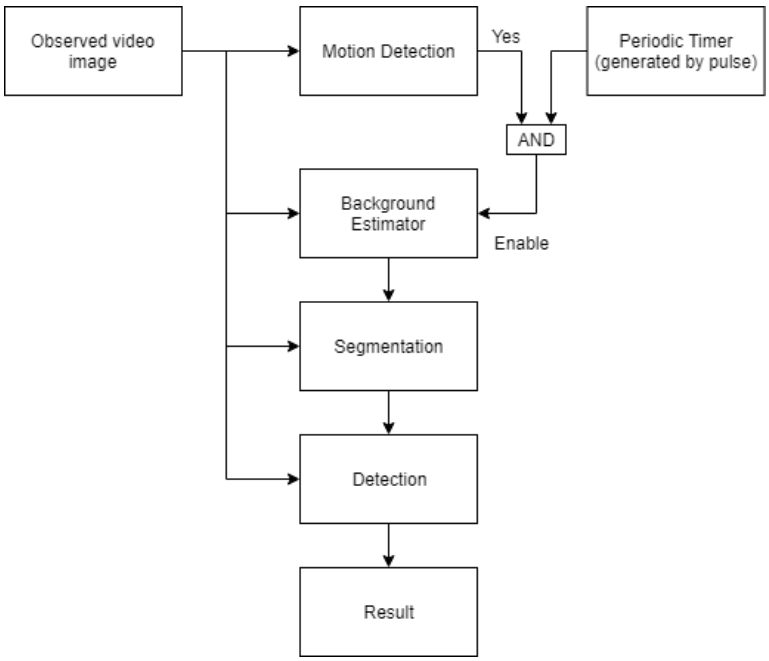


Figure 2.8: Periodic Background Estimation Subtraction [8]

2.2.4 Software Languages

There are multiple software languages that can do computer vision and image processing, but the most common remains:

- C++
- Python
- Java
- MATLAB

MATLAB is certainly one of the easier languages to code in, due to it extensive documentation and support. Although a major drawback is licensing and cannot be made open source and will be purely academical. A good example of the *background subtraction* method with MATLAB is done in [1].

C++, Python and Java all use OpenCV to do their respective image processing, which means that any of these can be chosen, each having their own pros and cons.

2.2.5 OpenCV

Figure 2.9: OpenCV Logo

OpenCV is an Open Source Computer Vision Library which means that it is a free library that is purposed for all forms of graphics processing. It is designed for computational efficiency and real-time applications.

It supplies a multitude of pre-written background subtraction including but not limited to algorithms from the published journals [14], [15] and [16]. These algorithms each tackle the problems that lighting and shadow issues background subtraction has.

The most notable of these are the Gaussian Mixture-based Background/Foreground Segmentation (MOG2) [15] and the Background subtraction based on counting (CNT) [17] which is specifically optimized for devices such as the Raspberry Pi for background subtractions.

These algorithms are not necessary to accomplish background subtraction, it also provides lower level tools to achieve it manually.

2.3 Electronics

The electronic system is responsible for interpreting the visual software and then controlling the motors with embedded software.

The electronic system usually consists of some microcontroller that controls the motors, and a single board computer (SBC) or Dedicated PC that can interface with the visual software with a relatively good speed. The motors should then rotate the paintball marker in the correct direction. There can be some intermediary hardware to the motors via some shield/breakout board.

2.3.1 DC Motors

Choosing motors will largely affect the rest of the electronic system, but the motors are again dependent on the physical structure specifications. Meaning that the motors should be strong enough to move a heavy paintball marker. The two types of motors that can easily be controlled is: *Stepper motors 2.10 and Servo motors 2.11*

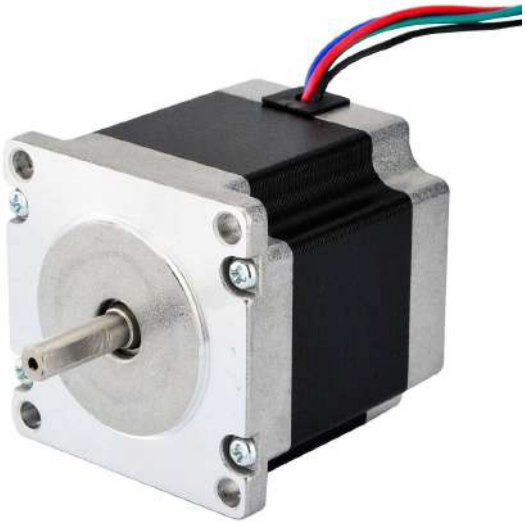


Figure 2.10: Stepper Motor

Figure 2.11: Servo Motor

From [18] Table 2 a few of the most relevant aspects are tabulated. A servo can provide a high torque for the heavy paintball marker, as well as a high speed, meaning a more accurate sentry. A stepper has very high acceleration which would make delays much shorter.

Table 2: Servo vs. Stepper

Requirement Application	Stepper	Servo
High Torque		✓
High Speed		✓
High Acceleration	✓	
Load Mass and Inertia	✓	
Cost		✓
Size		✓

A standard *Servo Motor* works with Pulse Width Modulation (PWM) rotating it to the

required location. Servos work by applying a certain frequency at a specified duty cycle to determine its position. Not all servos work with the same frequencies and values but are similar in how they are controlled. As seen in Figure 2.12 below, the servo's position is determined by the duty cycle, in this case, 1 - 2ms for 0° and 180° respectively [9].

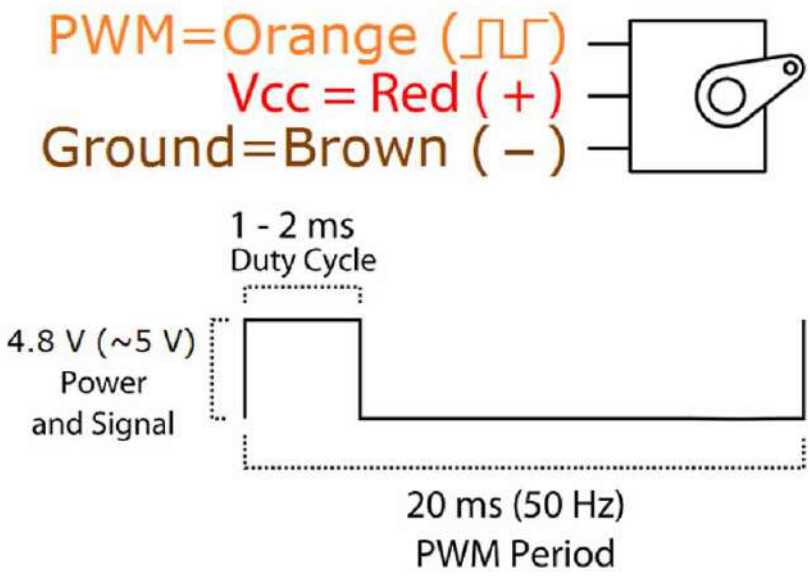


Figure 2.12: Servo Motor PWM [9]

2.3.2 Motor Controller Unit

This is the embedded device that is responsible for the interpretation of commands and movement of motors with PWM.

Servo control can either be done by:

- SBC
- MCU
- FPGA

Each of the previously mentioned devices would work in controlling a DC motor. Although the largest determining factor is how fast and easy it can be to take the given input and rotate the motors in specific directions.

SBCs like the Raspberry Pi are of the best embedded systems to do multiple tasks. They provide a wide range of features on top of basic I/O control. These features include External

RAM, Ethernet, Wi-Fi, I2C, SPI, HDMI display and an Operating System. Software scripts can be developed in languages like Python and C++ to control the GPIO. These extra features allow for easier future expansion of the project e.g. Remote control from a web application, or a single deployed unit without an external image processor like a laptop.

SBCs have significantly more processing power and memory to do complex tasks such as image processing, but it comes at a cost of higher power consumption. A major drawback for *SBCs* are their limited hardware PWM pins, this implies that either a dedicated PWM breakout board (that usually works with I2C) is needed, or PWM needs to be defined in software. Software-defined PWM can cause jitter in servos.

MCUs are programmable integrated controllers that can do simple electronic tasks such as analogue-to-digital conversion or PWM these devices are the most cost effective and power efficient in controlling something as simple as DC motors, but can not do any complex decision-making as an SBC can.

MCUs are also better at hardware PWM than Software PWM, this makes servo movement much smoother and perhaps even faster.

FPGA is a great device for handling multiple parallel I/O tasks simultaneously at a fast speed. An FPGA is not programmed in the traditional sense, it uses a Hardware Description Language (HDL) that changes its internal circuitry to handle I/O [19].

2.3.3 Electronic Power Supply

An external power supply for the electronics is definitely necessary to power the SBC or MCU, as well as the DC motors.

The sentry gun in [6] uses an HS-805BB Servo, which has a Current Drain - no-load (6V) of 830mA. This equates to a Power Dissipation of 5W per servo. The I/O pins of the MCU and SBC will not be able to handle this high current, therefore an external power supply is surely needed.

The scope of this does not need a large battery pack to make it a single deployed unit. Mains voltages with a simple DC step down converter can be used.

2.4 Mechanical System

The mechanical system is responsible for allowing the physical movement of the paintball marker, as well as being an enclosure for the electronic components.

2.4.1 Paint-Ball Marker

The **Valken SW-1 paint-ball marker** in Figure 2.13 will be used for this project. The weight of this gun + a full ammunition container: $1.7kg$.

By choosing this marker with a relatively heavy weight and rotational inertia, the choices of possible DC motors are being constrained. This means that a high torque is needed for rotating this gun. Which also implies that, if the motors can not supply enough torque, a sort of gear system might be needed.

Figure 2.13: Paint-ball Marker

2.4.2 Gearing System

Gears are a simple way of providing higher torque but at a reduction of speed. Equation 2.1 below gives the Gear Output Torque for a certain ratio. [20]

$$M_o = M_i r \mu \tag{2.1}$$

where,

Mo = output torque (Nm)

Mi = input torque (Nm)

r = gear transmission ratio

μ = gear efficiency (%)

And the Gear Output Speed [20],

$$S_o = S_i/r \tag{2.2}$$

where,

So = output speed (rad/s, rpm)

Si = input speed (rad/s, rpm)

Two most common methods to increase torque are either by: Timing Belts (Figure 2.14) or Gears (Figure 2.15)

Figure 2.14: Timing Belt with Pulley

Figure 2.15: Gears

Problems with gears and pulleys are their high cost and difficulty in finding the correct fit.

Addressing the cost problem; gears and pulleys can be 3D printed at a much lower cost. 3D printed parts are not very strong compared to aluminium. 3D printed parts might strip and shear due to the high torque. 3D printed parts can definitely be used for preliminary testing to check torque calculations and shear strengths.

Timing belts are certainly cheaper than normal gears, but take up more space and are not made to fit on servo motors.

Gears are relatively expensive but are designed to fit servos well.

2.4.3 Gearing Implications

Gears/Timing-belts will not only reduce the speed of the motors but also the degree of rotation. If a standard servo is used with a 180° rotation and there is a gearing ratio of 2:1, it implies that the movement is reduced to 90°.

If a stationary camera is used with a specific viewing angle, there is a respective maximum gear ratio,

$$r_{max} = \frac{\text{Servo Degree}}{\text{Camera View Angle}} \tag{2.3}$$

Exceeding this maximum ratio will result in losing available locations to shoot.

3 Preliminary Design

3.1 System Architecture

System architecture shows a conceptual model of the Turret; divided into sub-systems or functional units in this case. It numbers the hardware, software and electronic units as well as the interfaces that connect them.

3.1.1 Level 0

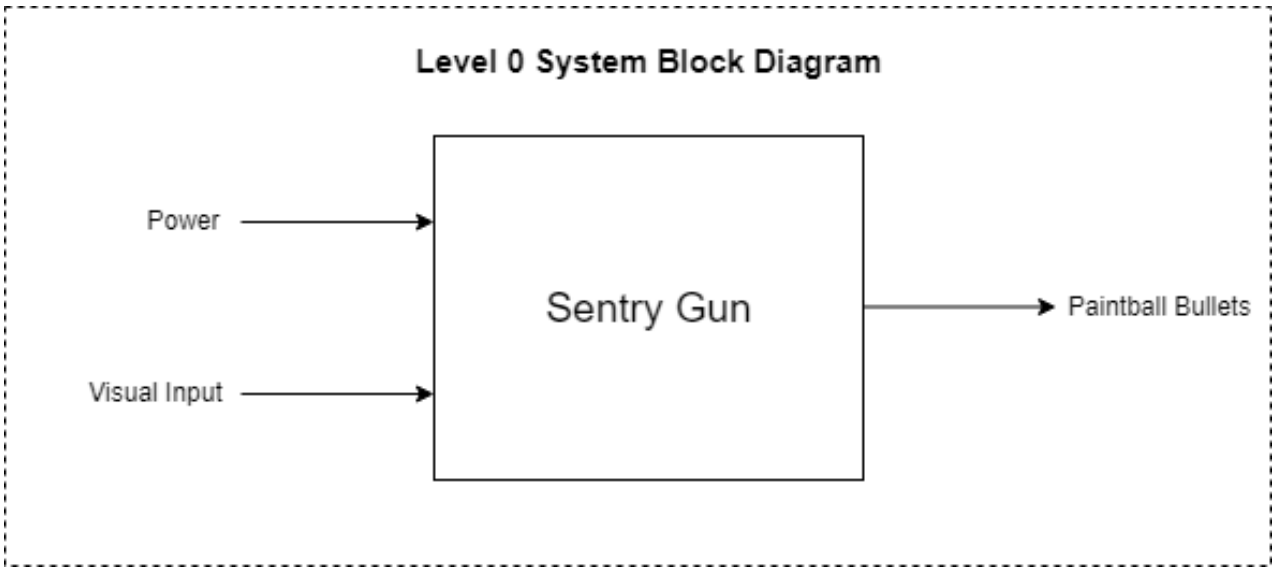


Figure 3.1: Level 0 System Architecture

Level 0 system architecture shows the most basic input and output of the system as seen in Figure 3.1. Power is given to the system, motion is detected and bullets leave the system. The intruder was left out of this level of the architecture.

3.1.2 Level 1

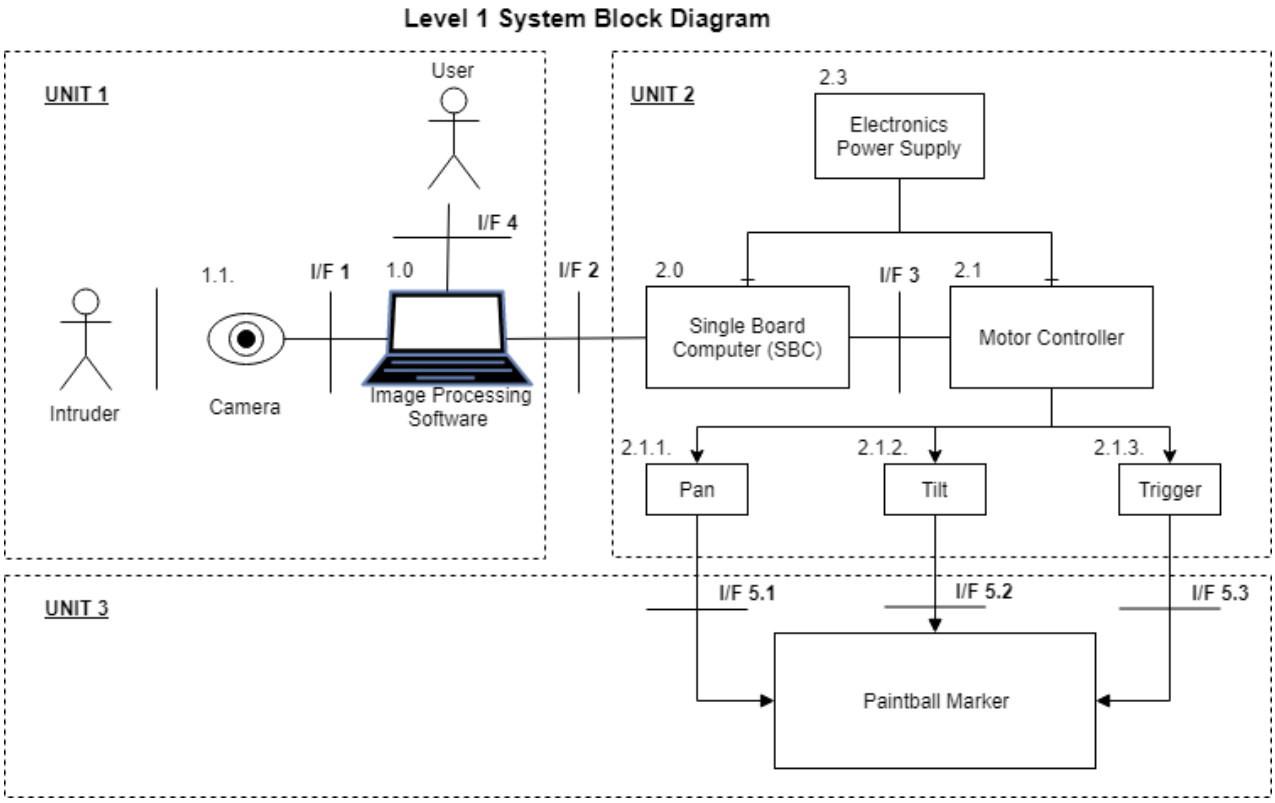


Figure 3.2: Level 1 System Architecture

Level 1 is slightly more complex. It breaks up the Turret into its most integral parts. Each of these units and sub-units will be numbered and explained.

Functional Units (F/U)

1. Software
   1.1. Computer/Software to do image processing
   1.2. Camera for visual input.
2. Electronics
   2.1. Single Board Computer to for software to hardware commands
   2.2. Motor Controller
   2.3. Power Supply
3. Hardware
   3.1. Paintball Marker
   3.2. Structure

Interfaces (I/F)

1. Camera to PC interface: Universal Serial Bus (USB)
2. PC to SBC interface: Ethernet or Serial cable
3. SBC to Motor Controller: From GPIO pins of SBC
4. GUI to User interface: Terminal or Generated GUI
5. Motor to Structure interface
   5.1. Pan motor to swivel motion structure: Gears or timing belts
   5.2. Tilt motor to swivel motion structure: Gears or timing belts
   5.3. Trigger motor to trigger structure

3.2 Operational Flow

Operational Flow describes the operation of the turret from the deployment until the storage.

3.2.1 1st Level

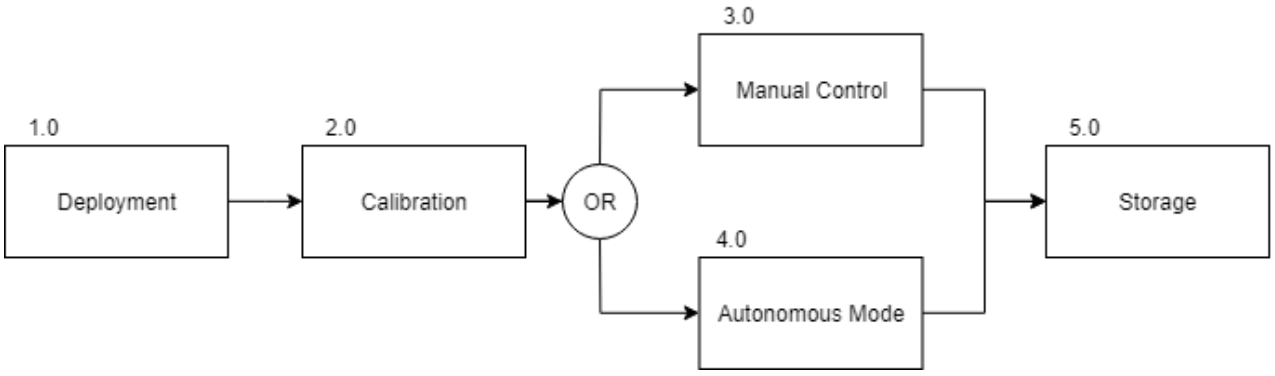


Figure 3.3: Turret Operational Flow, 1st Iteration

Level 1 shows how the turret is deployed i.e. connecting power and starting up the software. The turret is calibrated to the user's camera and frame interface. Users can choose between either manual and autonomous control. After the purpose of the turret has been fulfilled; it can be shut-down and stored.

3.2.2 2nd Level

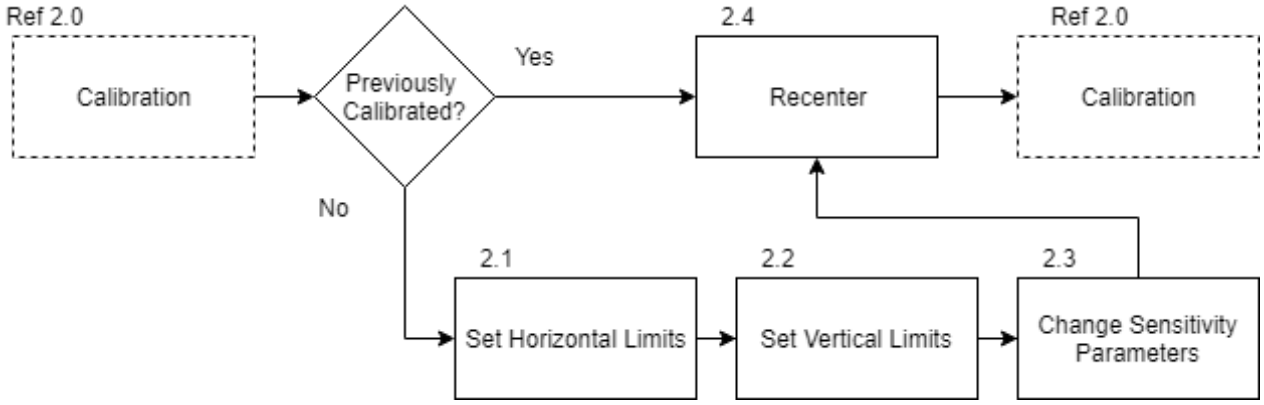


Figure 3.4: Turret Operational Flow, 2nd Iteration

Level 2 demonstrates how the calibration process goes. Calibration of the turret is the most crucial part concerning the hit rate. Calibration aligns the pixel coordinates with the physical degrees of freedom of the motors. These limits also provide an extra safety feature for the mechanical frame itself, should the turret move wildly, it will remain within its bounds.

Sensitivity parameters (2.3) provides the user to tweak some variables to make the turret perform better according to desire. The parameters include Smoothness, Vertical and Horizontal Anticipation and the even modes. Parameters should be adjustable at any time.

It will be good practice for the turret to recentre (2.4) itself and make it safe to reload the gun with bullets to prepare for the following step.

3.2.3 3rd Level

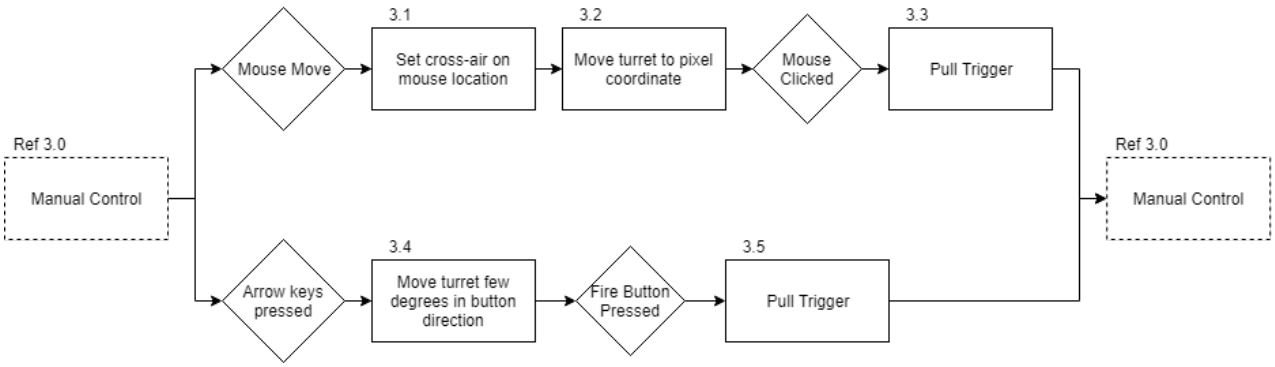


Figure 3.5: Turret Operational Flow, 3rd Iteration

Level 3 allows the paintball gun to be controlled via manual control, either keyboard or with a mouse. When the turret displays what the camera sees, the mouse can be moved to that pixel coordinate, and when clicked; a bullet is fired.

Having a display active could slow the performance of the turret, therefore an alternative would be to control the turret with the keyboard. This method would also be more accurate by using small increments to finely adjust the position.

Manual control is crucial to make the calibration process easier.

3.2.4 4th Level

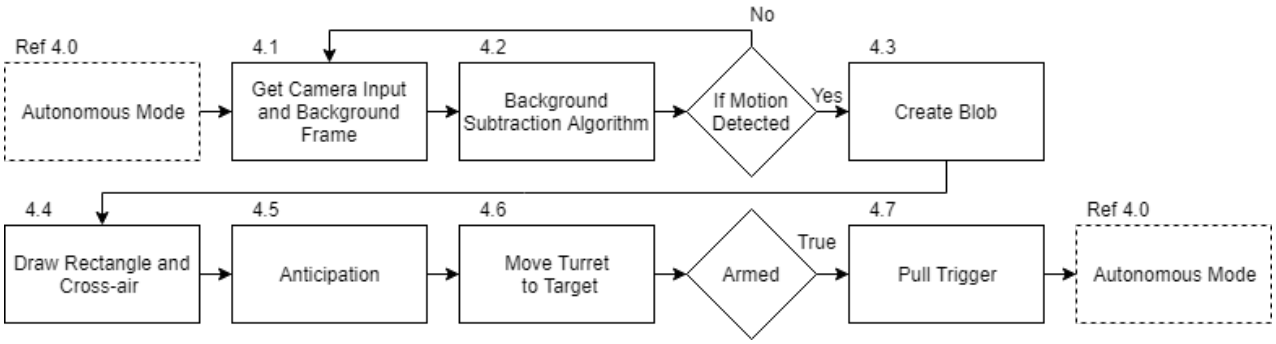


Figure 3.6: Turret Operational Flow, 4th Iteration

Level 4 is the most critical of them all. Firstly a background image has to be obtained (4.1). After which any incoming frame will be compared to, by using whichever background subtraction algorithm decided upon. The algorithm (4.2) is very subject to change. The algorithm to use can also become mode selectable, but the rest of the operation will stay the same. After the algorithm has detected motion, a rectangle is drawn over the area where there is motion and a cross-air is drawn.

To compensate for processing times and moving targets, the turret should anticipate the movement, and slightly adjust the turret to point slightly ahead in the direction the target is moving in.

If the turret is in the armed state, it will fire.

3.3 Trade-off Studies

To make logical and unbiased choices for the turret on which software, electronic and hardware to use, a decision matrix is drawn to calculate the best choices based on specified criteria. The tables below states only critical choices that have to be made in order to continue with any further design process.

Motors	1	2	1	1	1	6
	17%	33%	17%	17%	17%	100%
Option	Torque	Speed	Acceleration	Cost	Ease of use	Score
Servo	80	80	20	80	70	68
Stepper	20	60	80	40	60	53

SBC	2	1	1	1	1	6
	33%	17%	17%	17%	17%	100%
Option	Speed	Cost	Ease of Use	Peripherals	Extras	Score
Raspberry Pi 3	80	80	80	90	90	83
Raspberry Pi ZeroW	50	90	80	45	45	60
Up board	90	50	80	80	90	80
Nano pi Neo	50	100	80	45	45	62

Materials	2	1	1	1	5
	40%	20%	20%	20%	100%
Option	Cost	Strength	Weight	Usability	Score
Wood	80	70	80	80	78
Steel	40	100	40	50	54
Plastic	90	50	95	90	83

Visual Input	1	1	1	2	5
	20%	20%	20%	40%	100%
Option	Cost	Implementation Ease	Speed	Applicable	Score
Infra Red	100	80	80	40	68
Image Processing	70	60	50	90	72
LIDAR	35	30	100	80	65

Software Language	2	2	1	1	6
	40%	40%	20%	20%	100%
Option	Experience	Practicality	Documentation	Libraries	Score
C++	80	80	70	90	96
Python	80	90	60	80	96
Java	10	80	50	60	58
MATLAB	80	40	100	100	88

Figure 3.7: Trade Off Studies

4 Detail Design

The aim of the detail design is to explain how each aspect of the turret: Software, electronics and hardware would need to be implemented.

4.1 Software Design

The software of the turret is the most important aspect for functionality and performance. For the turret to work within a reasonable hit rate and precision; a few coding techniques will need to be implemented. In the following software sub-section, it will be explained how the software will go from visual input from a camera to the firing of a paintball bullet.

4.1.1 Raspberry Pi

Figure 4.1: Raspberry Pi 3 Model B

From the trade-off study (Figure 3.7) in the previous section, the Raspberry Pi has been chosen for the SBC to control the motors as well as run the image processing software. All of the image processing can be done by the pi but comes at a loss of processing power compared to a traditional laptop due to budget constraints. This has a negative effect on performance but comparatively saves physical space and set-up time. This makes the turret easier to deploy and more cost effective, but means that the software should make up for the processing loss.

4.1.2 Interactive Development Environment (IDE)

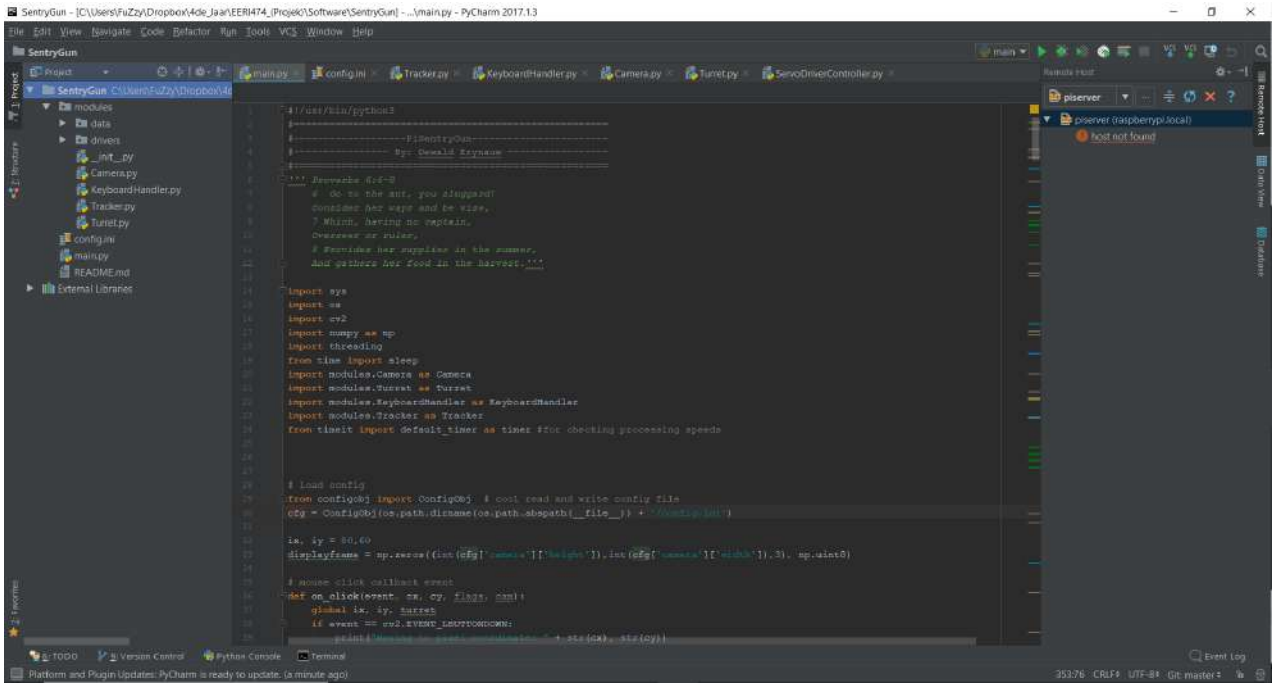


Figure 4.2: PyCharm IDE

IDE for development is JetBrains PyCharm. It provides easy to use programming environment for Python with smart sense and auto-completion. The IDE also allows remote compilation of code onto a pi, this is the reason why Python is chosen as the developing language and not C++. The remote compilation increases developing speeds and reduces cable usage, as it syncs and builds over the Wi-Fi.

Other features include Github version control, SSH commands, live python scripting, file managing, and debugging with processing times and threading capabilities.

4.1.3 OpenCV

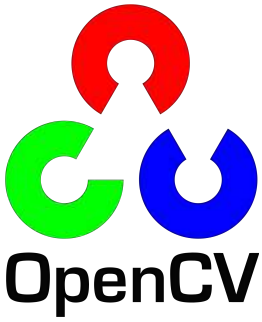


Figure 4.3: OpenCV Logo

OpenCV (Open Source Computer Vision Library) will be used for all the computer vision needs. It is compatible with both C++ and Python, therefore it would fit the purpose of this project perfectly. It is aimed at real-time computer vision, which reduces the computational overhead on the pi and alleviates the burden of programming from first principles.

OpenCV also includes a few built-in background subtraction algorithms such as the: Gaussian Mixture-based Background/Foreground Segmentation Algorithm (MOG2) and Background subtraction based on counting (CNT) which would work very well with a pi according to the OpenCV documentation.

A personal algorithm will also be written such as in *Figure* 4.4. All of the algorithms will be tested and compared; the best will then be incorporated into the turret's software. The different algorithms could also become mode selectable for personal preference.

OpenCV also includes neural networks and facial recognition (dlib) libraries that can make the turret more personalised. It even includes advanced motion tracking systems, but these options are out of the scope for this project and would only be "nice to have" features.

4.1.4 Motion Tracking Algorithm

To detect and track an intruder; a form of motion tracking is needed. The most common algorithm for motion tracking is *Background Subtraction* (BS). There are a plethora of methods for doing BS, but the basic form is: There are a background frame and the new frame where there is motion. The two frames are subtracted from each other, and that difference represents where the motion is.

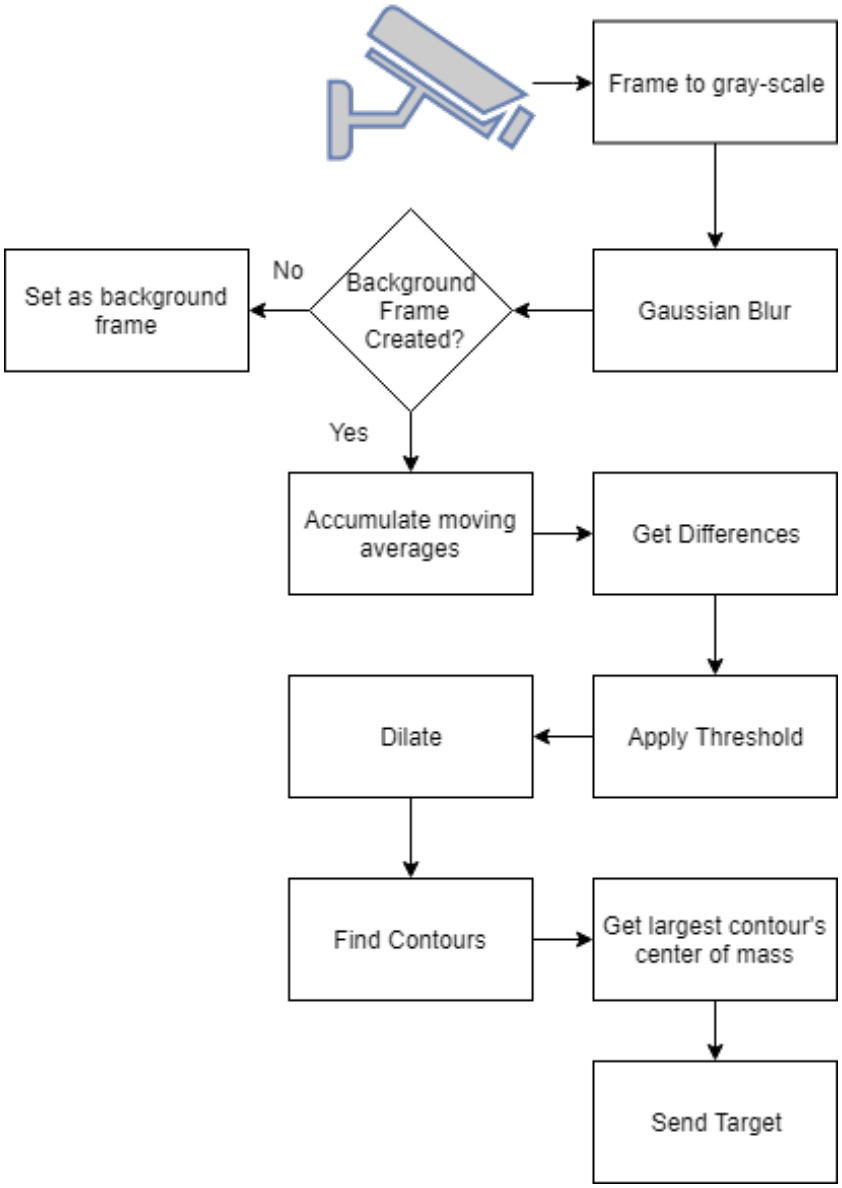


Figure 4.4: Detailed Background Subtraction Algorithm

Figure 4.4 above shows how background subtraction will be implemented in the software. This flow diagram iterates continuously in the autonomous mode. Parts of the algorithm is based on the *PyImageSearch.com* example [21]

Firstly the image/frame is captured from the camera, resized to fewer pixels and converted to grey-scale. This enables the pi to process more frames at a faster rate.

Next, a Gaussian blur is applied to the current frame, this enlarges the possible movement area.

If this is the first iteration a new background frame is set, which will be referred to as the average frame.

Now a moving average is accumulated by the current and previous frames. This addresses the number one issue with background subtraction: The changing of the environment and lighting conditions. By getting a moving average, the background frame changes with respect to time.

The absolute difference between current and average frame is calculated, and all values above a threshold are made white and others black.

To make the area of motion more clear; a dilation is applied, to make a nice blob of the object in motion. This ensures that the entire object is captured and not just the edges.

Finally, the largest contour of the blob is found along with its centre of mass. These pixel coordinates from the centre of mass are sent to the software part that controls the motion of the paintball gun.

4.1.5 Turret Control

The turret controller is an object in the software that is in charge of getting the pixel coordinates and moving the paintball gun via motors to the corresponding location in reality. It is also responsible for the firing of bullets.

The Turret Controller thread is responsible for the following:

- Initialize all the configuration variables
- Translating coordinates to pulses for the servo driver
- Setting/Resetting calibrations
- Anticipation
- Smoothing
- Moving the servos

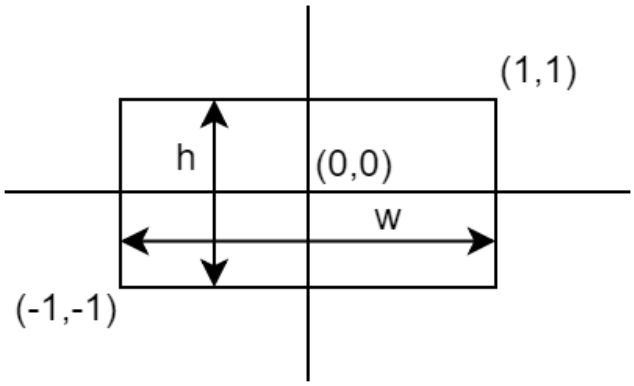


Figure 4.5: Coordinate Representation

Rather than working with servo pulses widths and degrees, a better coordinate system like represented above in Figure 4.5 will be used. This allows for floating point precision no matter the pixel size of the image. It is also humanly more comprehensive that millisecond values. This type of coordinate system comes from the sentry turret with dlib [7]. The turret controller will convert the pixel coordinates to the pulse coordinates system and send it to the servo driver.

Now a driver should be written for the servo to understand the aforementioned pulses. This driver again translates the pulse values to PWM values that the servo can understand.

4.1.6 Threads

Threading allows for the simulation execution of pieces of code.

To optimise processing speeds threads can be created for:

- Turret Controller
- Camera
- Trigger
- Keyboard commands

Implementation of threading can increase the speed at which the software runs. Problems when using threading is resource management. In order to avoid *deadlocks*; there should be no shared variables between these threads that are critical to functionality. The pi also has a few cores to allow for threading, consequently, threading must be implemented.

4.1.7 Calibration

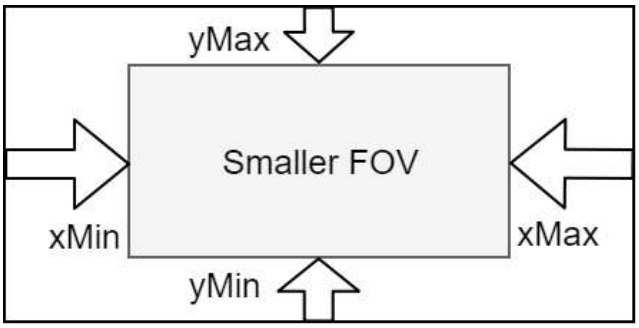


Figure 4.6: Calibration Mapping Illustration

Not all cameras have the same lenses or field of view (FOV), this means that the turret should be able to adapt to different viewing angles relatively easy. This can be achieved by a simple mapping technique.

New upper and lower limits are set to the specified value, and a new ratio is determined from them,

$$Ratio = \frac{\text{Actual Height/Width (pixels)}}{\text{new Max} - \text{new Min}} \tag{4.1}$$

4.1.8 Anticipation

Due to the natural delays in processing and moving parts, if the turret shot at the location where it detects motion, the target could have already moved past that point and the turret shoot behind it. So to compensate for the lost time and the motion of the target, some anticipation is needed.

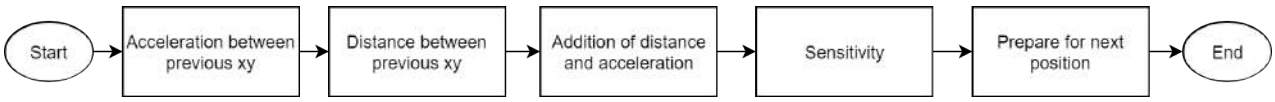


Figure 4.7: Anticipation Flow Diagram

4.1.9 Smoothing

A problem that might arise is if a target is sent to the turret and it needs to move from one end of the screen to the other in an instant. The turret could possibly move too fast jump a few gears or damage the servos. In order to prolong the longevity of the turret, some smoothing will be done in the software.

The solution lies in *Zeno's Paradox*. As illustrated in Figure 4.8, if the starting point is 0 and the end point is 1, if we move closer by some fraction (smoothing factor) over and over again, the target approaches 1 over a few iterations, smoothing the motion, by forcing it to move at increments.

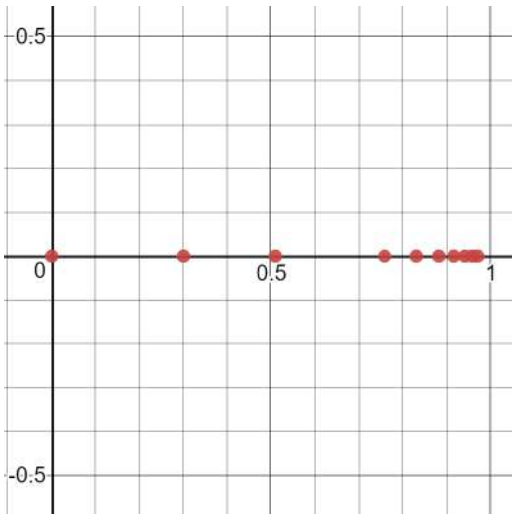


Figure 4.8: Smoothing Algorithm

4.2 Electronics Design Set-up

4.2.1 Adafruit PCA9685

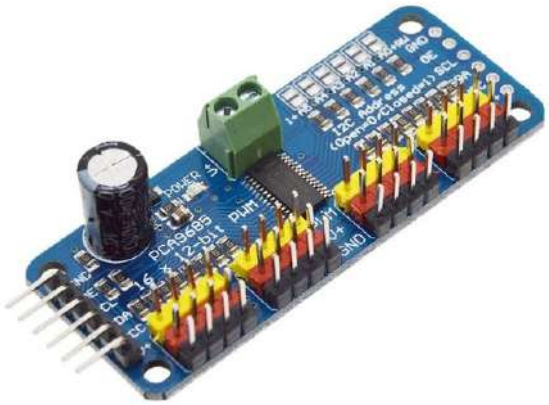


Figure 4.9: Servo Motor Controller (I2C)

Servo Motor Controller is a product of Adafruit, which is a 16-Channel 12-bit PWM/Servo Driver with an I2C interface. It comes with a Python library to easily communicate with the servos with high-level commands. This board negates the designing of a PCB and already

has a mutual ground. Instead of wasting 8 GPIO pins on the pi, only 4 is needed, making the electronics much neater.

The Adafruit PCA9685 is the most important design choice of all. Most of the other design is dependant on this controller, such as the use of Python and the use of Servos instead of Stepper Motors.

4.2.2 Power Supply

If each servo draws about 1A at 6V. That equates to a **power dissipation** of $18W$. This can be achieved by a large battery but adds a hassle of charging it and unreliability. Another option is just to use mains voltage and step it down to the correct DC voltage.

Figure 4.10: 12 Volt DC Power Supply

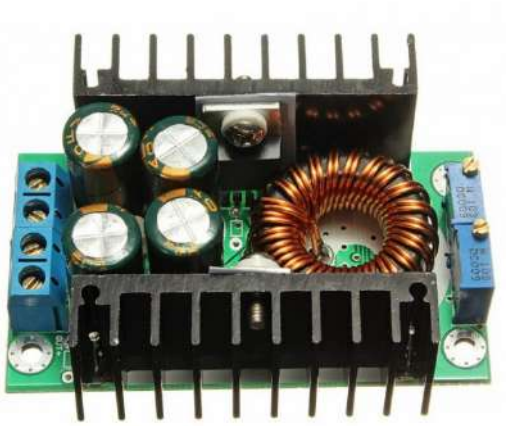


Figure 4.11: Step Down Buck Converter

Figure 4.10 depicts the most general AC to DC converters. It takes 220V AC to 12V DC and is generally rated at 10A.

Unfortunately, the PCA9685 operates between 4-6V, this necessitates the need for voltage to be stepped down to 6V. Not just any voltage regulator would be able to handle the high current. A step-down buck converter that can handle at least 5A is needed, as presented in Figure 4.11.

This specific converter is rated at:

- Input voltage range:7 to 40VDC
- Output voltage range:1.25-36VDC adjustable
- Output current: 0-9A
- Output power: 280W
- High efficiency up to 96
- Built in thermal shut down function
- Built in output short protection function
- Input reverse polarity protection: None

4.2.3 Web-cam

Figure 4.12: Webcam (Logitech S7500)

Software needs to get video images from some sort of camera. The most suited for this project would be a webcam. Other options like an IP cam and Pi Cam are either too expensive or too much additional wiring. Webcams simply interface via USB to the raspberry pi.

4.2.4 Servos

Figure 4.13: Servo Motor (20 Kg.cm)

Three of the standard **JX PDI 5521MG** servos will be used for the pan, tilt and trigger motion on the paintball gun. A standard servo means that it operates between 0 and 180 degrees.

Table 3: JX PDI 5521MG Specifications

Dead band	2μsl
Working frequency	1520μs / 330hz
Operating Voltage	DC4.8∼6.0 V
Operating Speed (4.8V)	0.18sec/60 degr
Operating Speed (6V)	0.16 sec/60 degre
Stall Torque (4.8V)	17.25kg.cm (239.55
Stall Torque (6.0V)	20.32 kg.cm (281.8
Dimensions	40.5X20.2X44.2mm /1.59 X0.8
Weight	55.6 g (1.96oz)
Connector Wire Length	JR 265 mm (10.43

These servos are not too expensive for the amount of torque they provide. A 20kg.cm servo should easily be able to rotate a 2 Kg paintball gun.

4.3 Mechanical Structure

Mechanical Structure is tasked in converting the servos' motion to the rotation of the paintball gun. All of the following designs were created using *Solidworks*.

Firstly a basic frame was designed that could house the paintball gun. A model paintball gun was imported from *GrabCAD.com* and inserted into the design for rough estimations.

The servo model was also imported from *GrabCAD.com*. The servo gears and worm gears were acquired from *ServoCity.com*.

Figure 4.14: Detail Desgin (Left View)

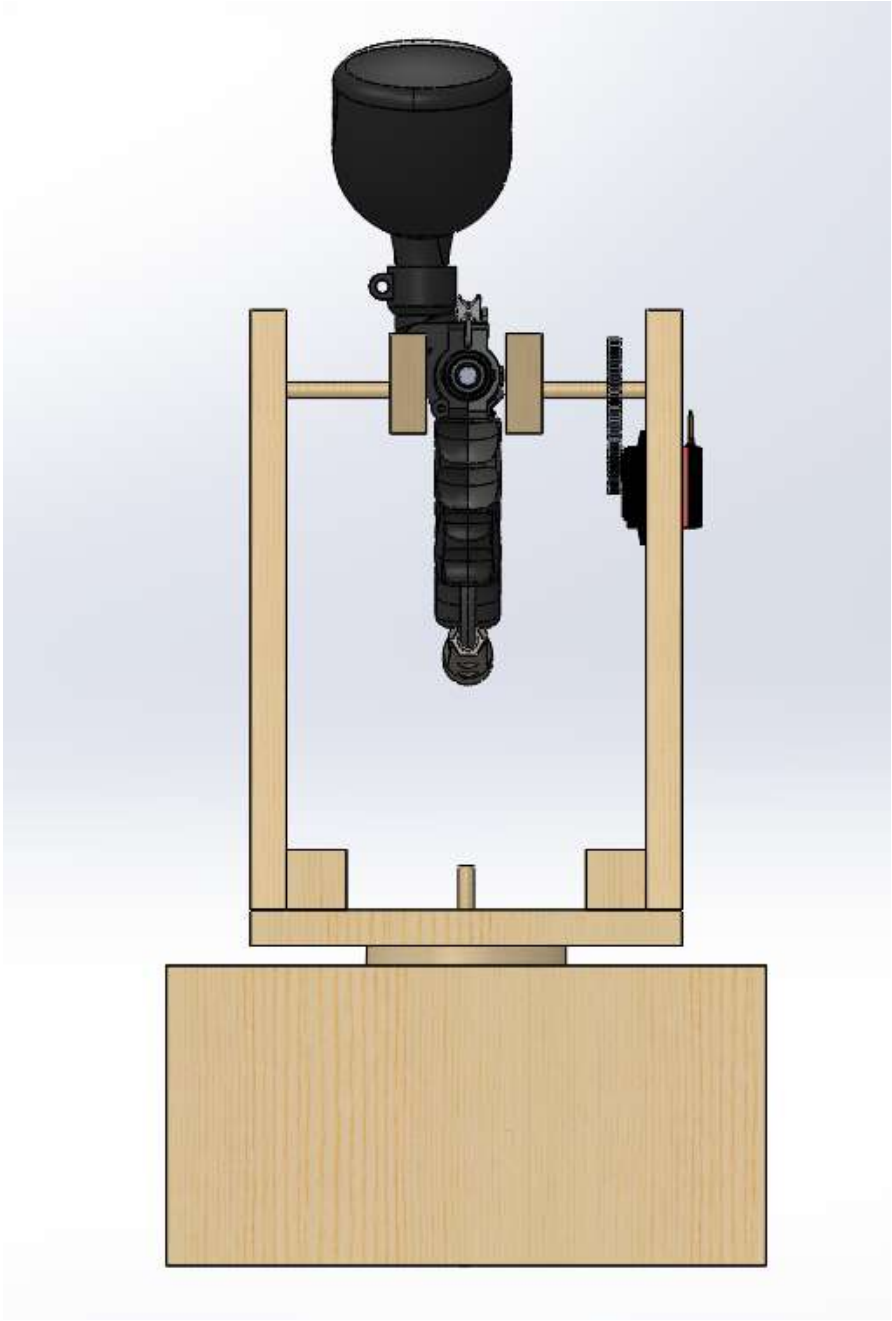


Figure 4.15: Detail Desgin (Front View)

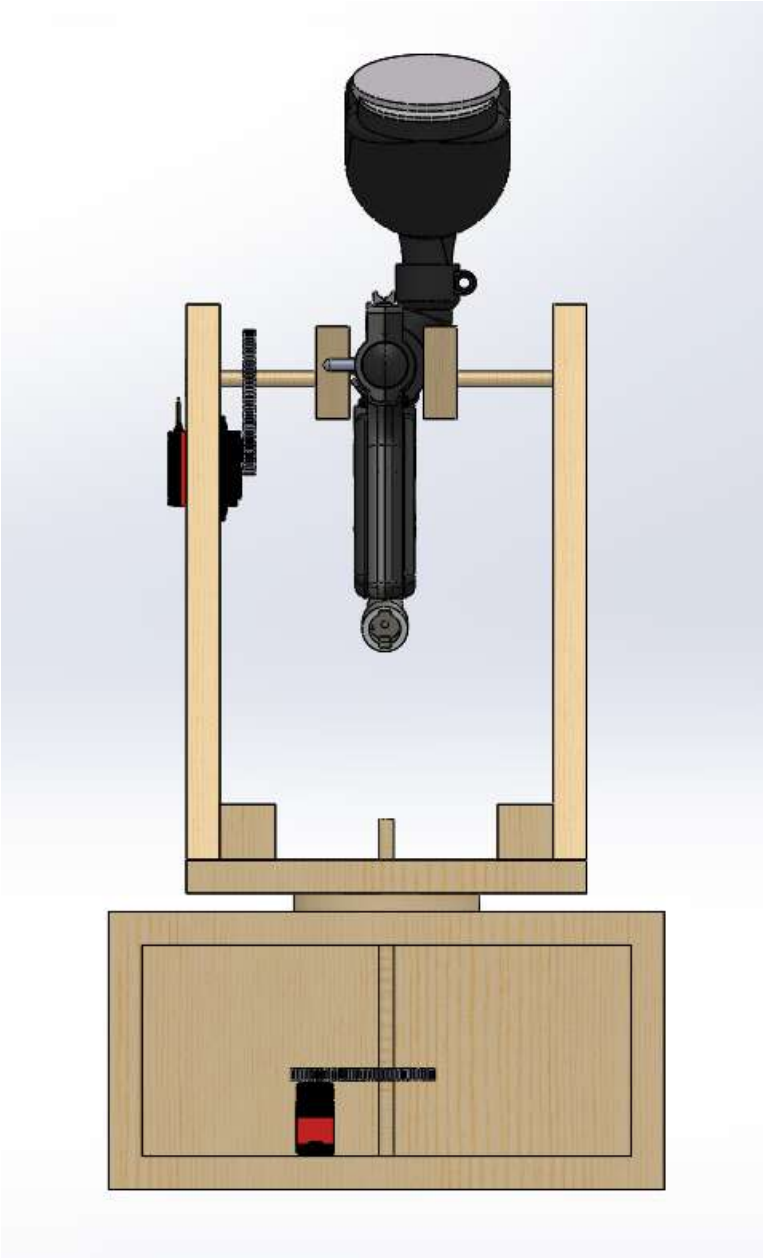


Figure 4.16: Detail Desgin (Back View)

In Figure 4.16 the gears and servos can be seen nicely how they are fitted.

Figure 4.17: Detail Desgin (Side View)

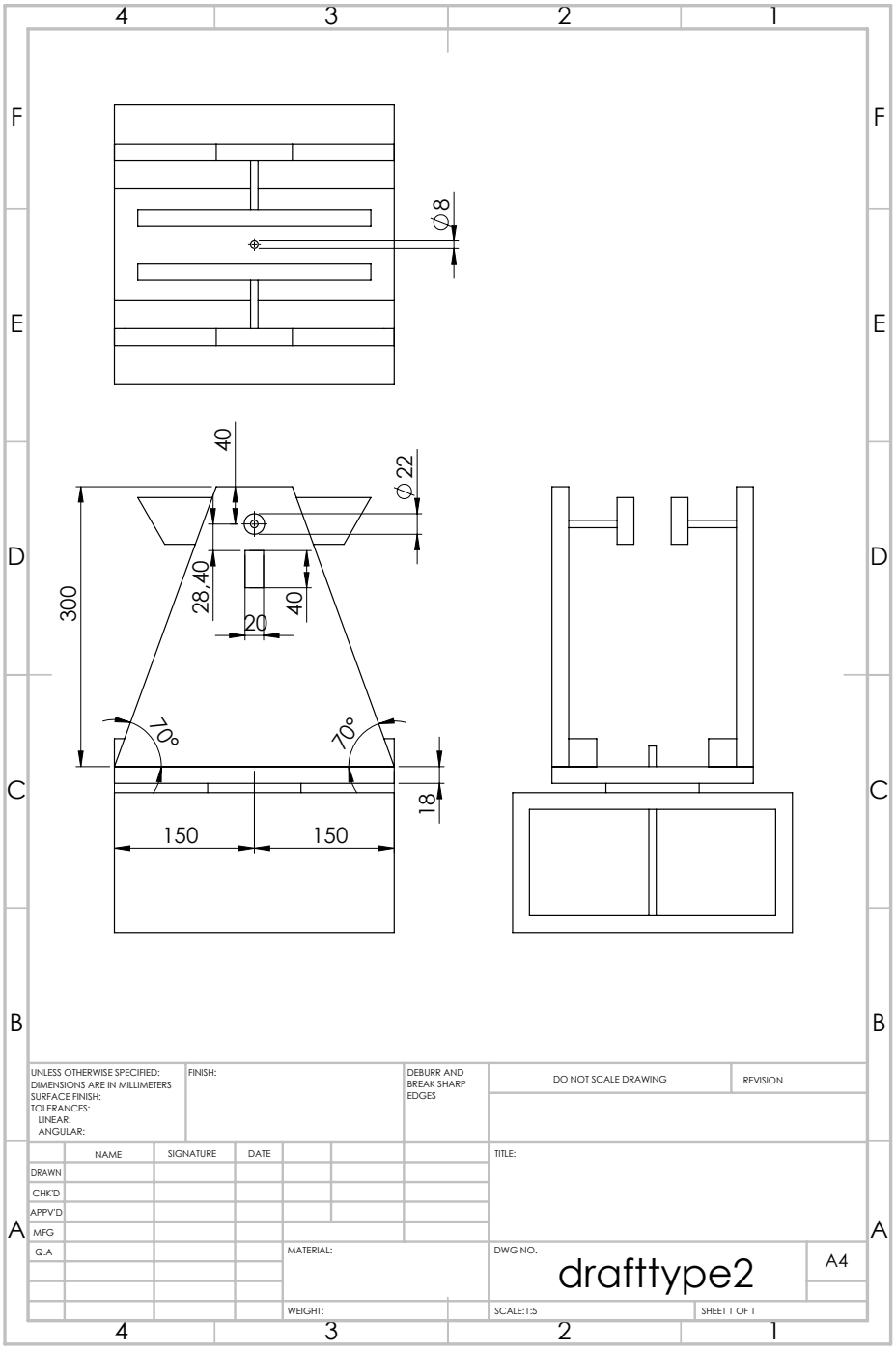


Figure 4.18: Detail Schematic

5 Implementation

5.1 Mechanical Construction

Mechanical Construction is the riskiest aspect of the project, it can break and be difficult to fix. It had to be designed very precisely for all the small gears to fit and cut to the exact sizes. The material chosen for the construction is **plywood**, as it is cheap and lightweight.

Figure 5.1: Frame Construction - Mount

The Mount (Figure 5.1) is responsible for the panning motion of the turret and housing the gun holder. It should be rigid to support the paintball gun during rotation. Later some additional brackets are inserted to fixate it more.

Figure 5.2: Frame Construction - Box

Figure 5.3: Frame Construction - Gun Holder

The box in Figure 5.2 will contain all the electronics. The gun holder frame in Figure 5.3 differs from the detail design, as not all paintball guns are the same. This frame allows for multiple different types of paintball guns to fit but it purposefully made to fit this specific gun. Gas is supplied to the gun via a remote line. This takes off the canister from the gun reducing strain on the servos and giving it more freedom of movement.

Figure 5.4: Frame Construction - Unassembled

The unassembled frame construction (Figure 5.4) marks the first half of the mechanical construction. Next, all the holes need to be drilled to fit all the gears, shafts and servos.

5.1.1 Centre of Mass

To alleviate unnecessary stress from the servos, the centre of mass for the frame holding the gun is determined from hanging it on a string a few locations with a weight at the bottom as illustrated in *Figure* 5.5. A line is drawn on the frame where the string is. The moment of the gun and frame at that location is represented by this line. This is repeated until a clear point arises where the lines cross each other. This point is the centre of rotation where the holes for the shafts are made to have the least strain on the servos.

Figure 5.5: Centre of mass of tilt frame

5.1.2 Moving parts

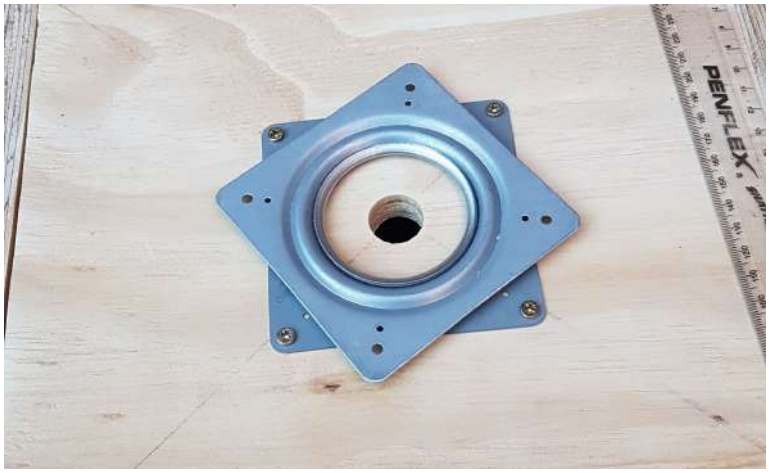


Figure 5.6: Swivel Bearing

The Swivel (Lazy Susan) Bearing is just a simple bearing that will allow the pan frame to rotate on top of the box. But to use this bearing, and additional hole needed to be drilled in die panning frame to insert the screws.

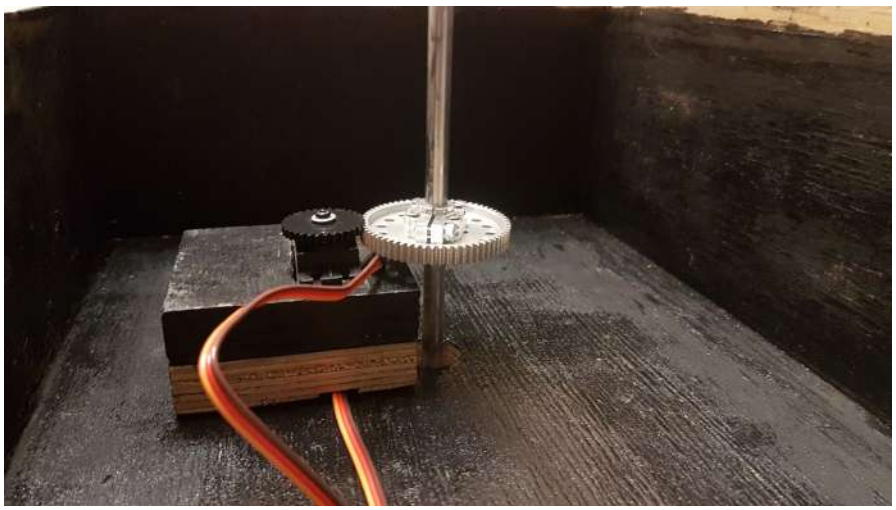


Figure 5.7: Pan gears implementation

The additional block in Figure 5.7 was inserted to secure the servo tightly to fit into the shaft's gear.

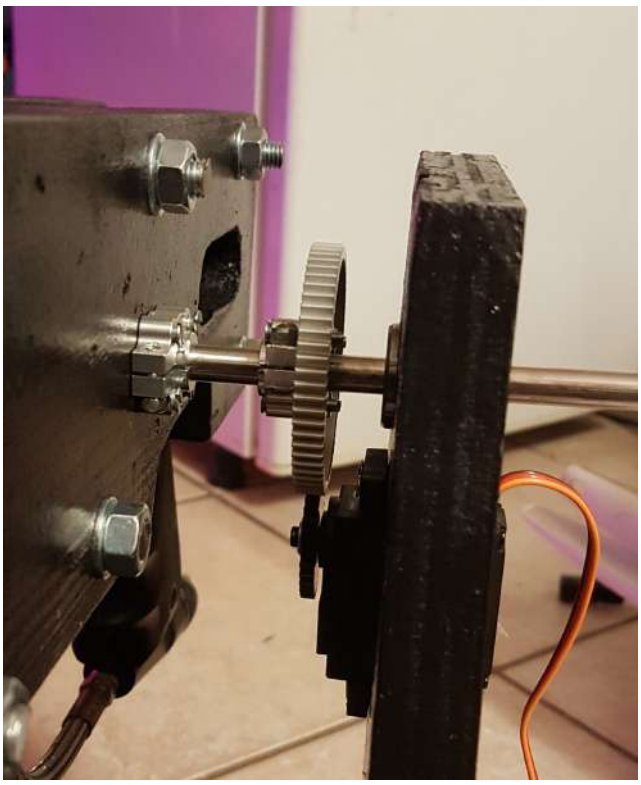


Figure 5.8: Tilt gear implementation

Figure 5.8 illustrates how the tilt gear is fitted after everything has been painted.

Cutting the holes for the servos to fit in, is an extremely exhausting process. A chisel worked really well to precisely and neatly make these holes. Standard screws included with the servos are small and prone to breaking. Therefore these needed to be hand screwed to avoid more screws breaking.

5.2 Electronics Implementation

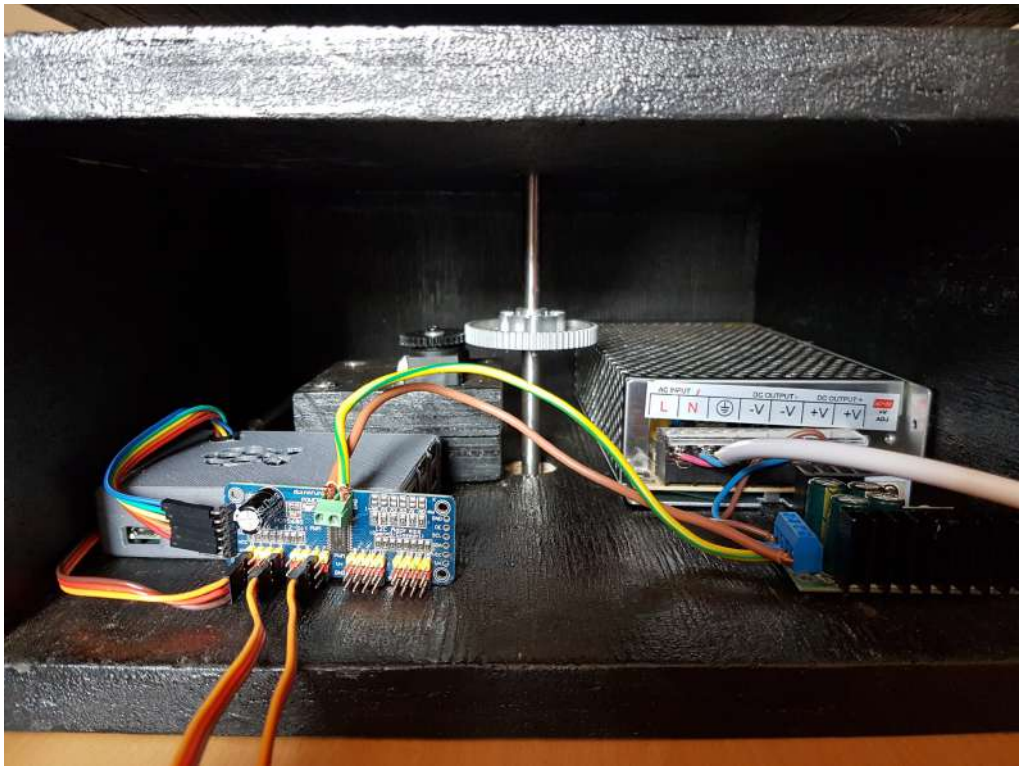


Figure 5.9: Electronics Implementation

All the electronics are connected and placed inside the box (Figure 5.9). From the aforementioned figure, it is clear to see that the electronic set-up is not complex and neat due to the Adafruit servo controller.

For the webcam, the Logitech S7500 was implemented, as it was already owned. Any webcam will do the job well, as long as it is more than 160x120 pixels.

5.3 Software Implementation

The software were implemented with the following specifications:

- Python 3.5
- OpenCV 3.4.1
- OpenCV open-contrib-library
- Adafruit PCA9685 library
- Raspbian Stretch

5.3.1 Development Set-up

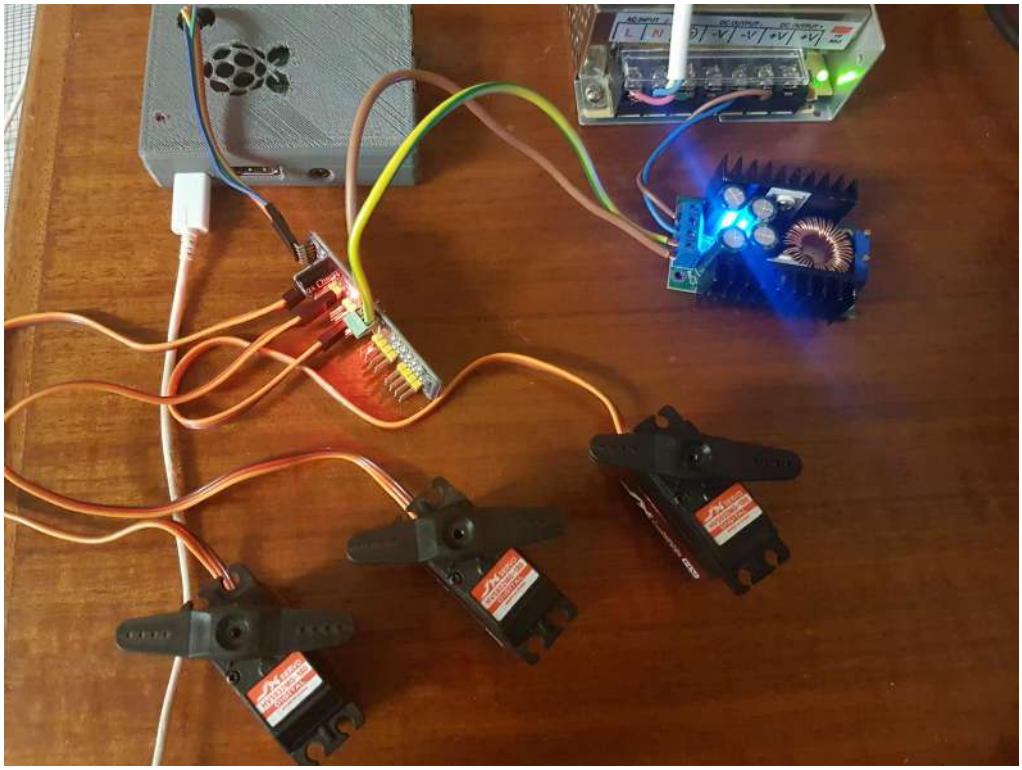


Figure 5.10: Software development set-up

Before the code was written and tested on the mechanical construction, the code was first developed as shown in Figure 5.10. This was not to damage the construction during development phases. Luckily there were three additional servos available, so there was no need to adjust anything on the physical construction.

5.3.2 Graphical User Interface (GUI)

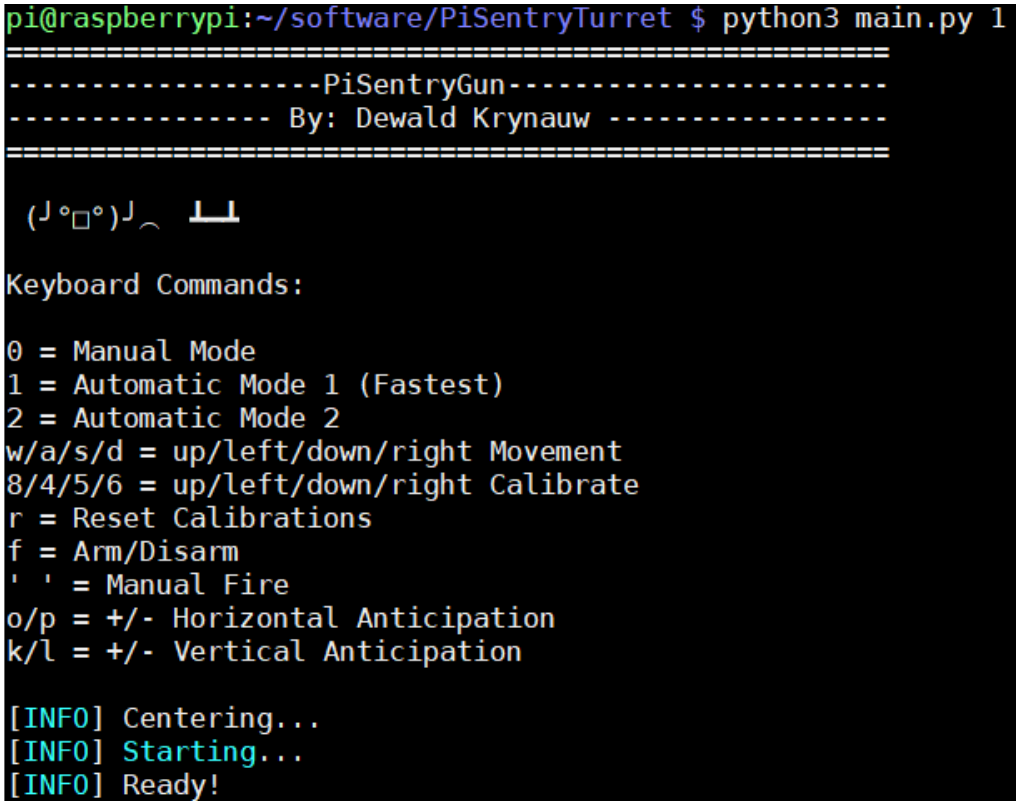


Figure 5.11: Software Terminal GUI

PiSentryGun software (Figure 5.11) can be run by executing the python3 command on the main.py file. If no argument is given the program executes without a display (headless), if the argument of 1 is passed, it will give a small display.

The program starts up by centring the turret and waiting a few seconds for everything to start up. It starts up in manual mode, after which the user can use the keyboard command to switch to other modes.

All the other available commands are displayed terminal GUI. Settings from calibrations are all saved in the config.ini file, for later retrieval even after restarting the software.

5.4 Integration

Now that all the functional units are completed the integration can commence.

Figure 5.12: Integration (Left View)

Figure 5.13: Integration (Front View)

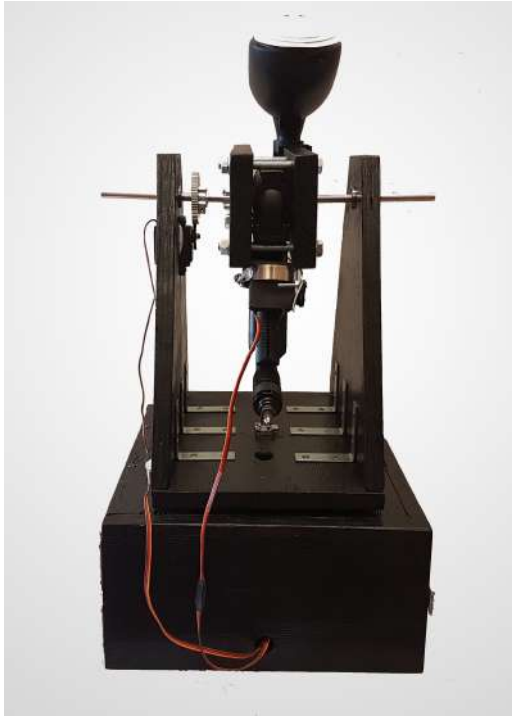


Figure 5.14: Integration (Rear View)

Figure 5.15: Integration (Side View)

6 Test and Evaluation

The following section provides a more quantitative analysis of the tests run, before and after the implementation.

6.1 Motion Tracking Algorithms Evaluation

Purpose Testing OpenCV's built in background subtraction algorithm speed.

Set-up OpenCV provides a multitude of code and libraries for motions tracking. Before just choosing an algorithm and implementing it; the different types of BS where compared.

opencv_contrib/bgsegm library on *GitHub* provides some pre written scripts that can evaluate these different algorithms opencv has built in. But firstly some test data is needed.

changedetection.net [22] [23] provides large datasets regarding motion, from acquisition at night to PTZ cameras. The pedestrian dataset (Figure 6.1) will prove the most useful in our scenario. It contains labelled data that the *evaluation.py* script can use to determine its performance and mimics the type of video input the turret can expect.

Figure 6.1: Dataset 2014 - Pedestrian

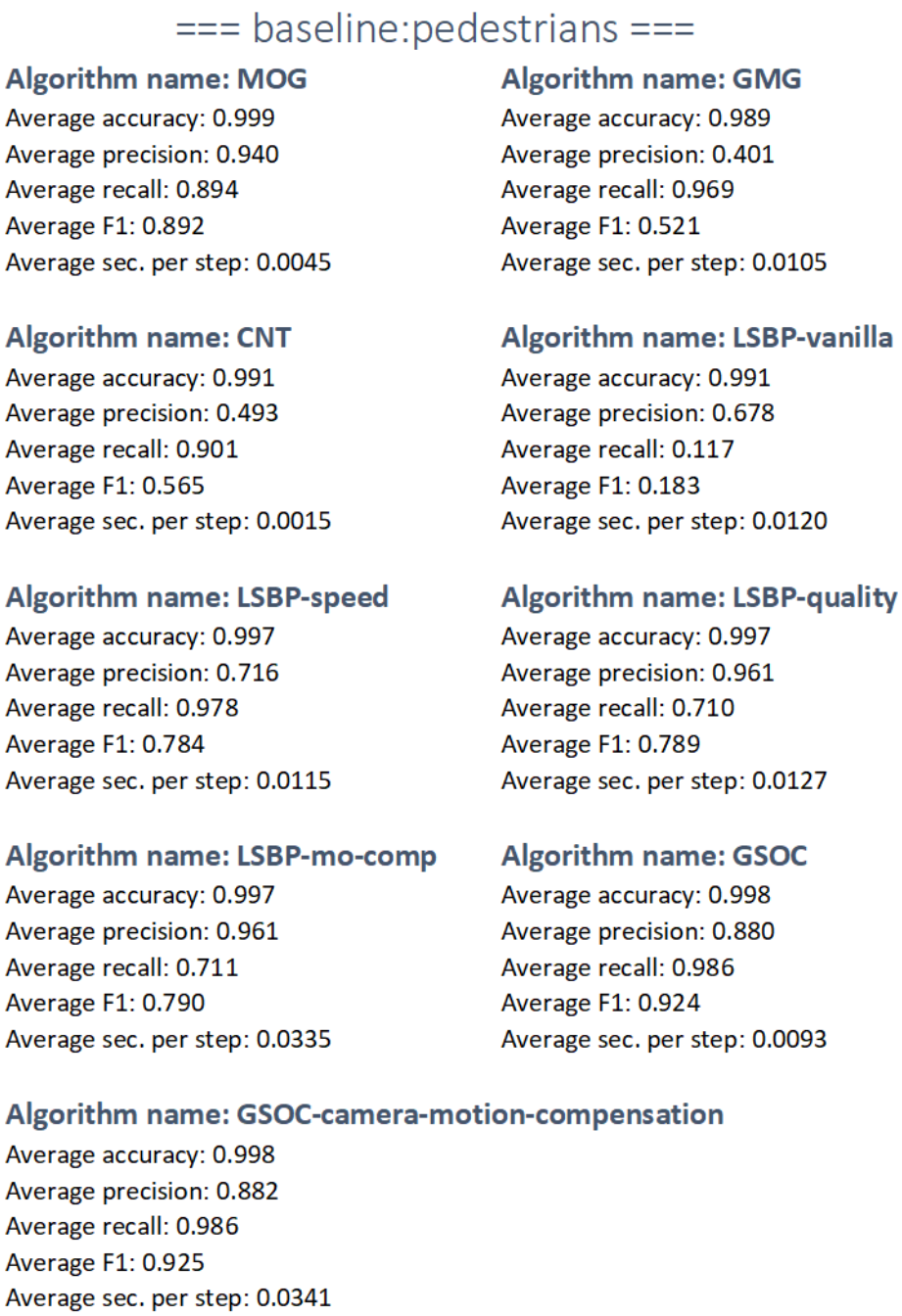


Figure 6.2: Algorithms Evaluation Results

Results At first glance looking at the Algorithm Evaluation results (Figure 6.2), it is seen that the *MOG* and *GSOC* was the most accurate based on the F_1 score [24]. But they

compared quite slow to the *CNT* algorithm. The *CNT* speed would be even more than twice as fast as the *MOG* on cheap hardware (Raspberry Pi) based on the OpenCV documentation [17]. These algorithms can be implemented after the custom BS algorithm has been applied.

6.2 Manual Background Subtraction Test(Mode 1)

Purpose Creating a basic background subtraction algorithm that captures motion in a rectangle.

Set-up This specific manual background subtraction method is the basic algorithm based on the PyImageSearch.com example [21]. It is the algorithm described in Figure 4.4 of the detail design. Software was written using OpenCV and Python.

Results Window 1 from Figure 6.3 is the captured image with the bounding boxes drawn over it of all the contours. Window 2 is the grey scale difference between the current and average background frame. Window 3 is the threshold and diluted frame of which the contours are drawn.

Figure 6.3: Manual Background Subtraction Test

6.3 Rolling Frame Subtraction Test (Mode 2)

Purpose Creating another background subtraction algorithm that captures motion in a rectangle.

Set-up Rolling Frame is much like the former algorithm, but instead of having an average background frame, the previous frame is the background frame.

Results Window 1 (Figure 6.4) is the input as well as the output image with a bounding box. Windows 2 and 3 is the difference and blurred images in RGB and grey scale. Windows 4 and 5 are the eroded and diluted operations.

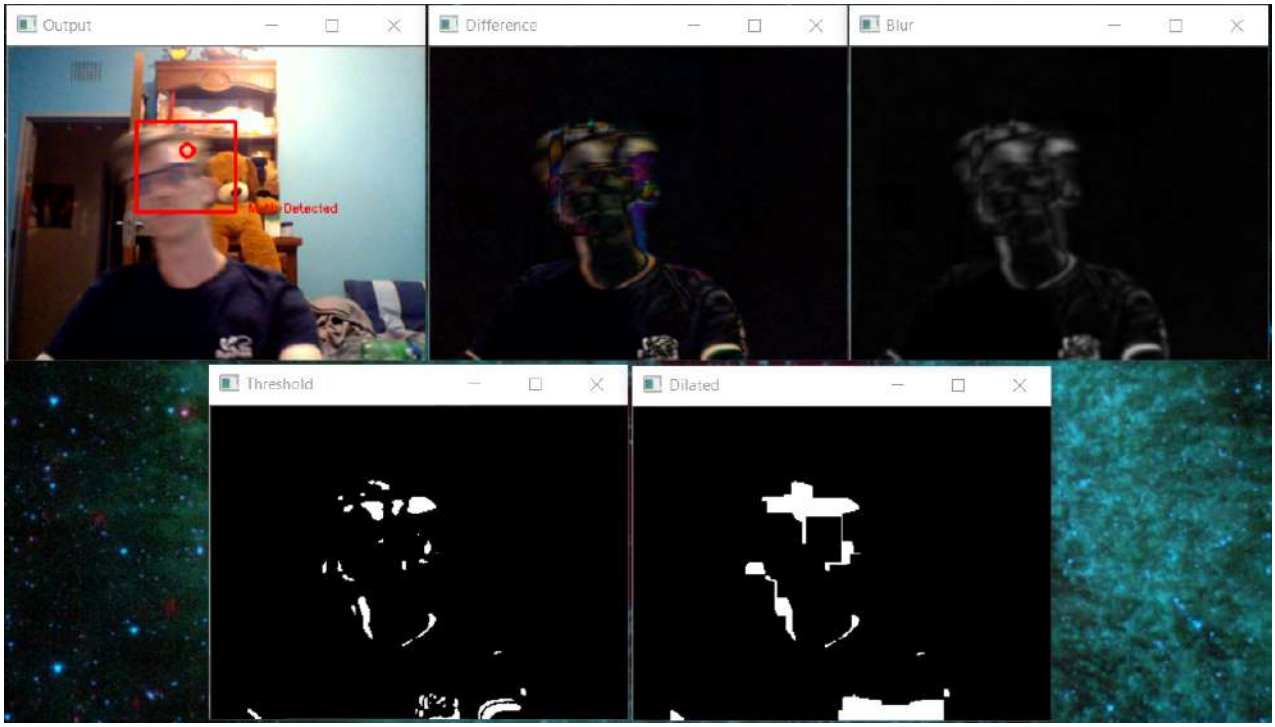


Figure 6.4: Rolling Frame Background Subtraction Test

6.4 Built-in Background Subtraction Test(Experimental Mode 3)

Purpose Comparing OpenCV's built in BS algorithm speeds to the preceding results.

Set-up Each test was run with a timer logging the average time it takes for one iteration of the algorithm to complete.

Results All the built-in functions were tested, but only the *MOG* algorithm performed as expected. Results can be seen in Figure 6.5.

The averages times are displayed in Table 4 below.

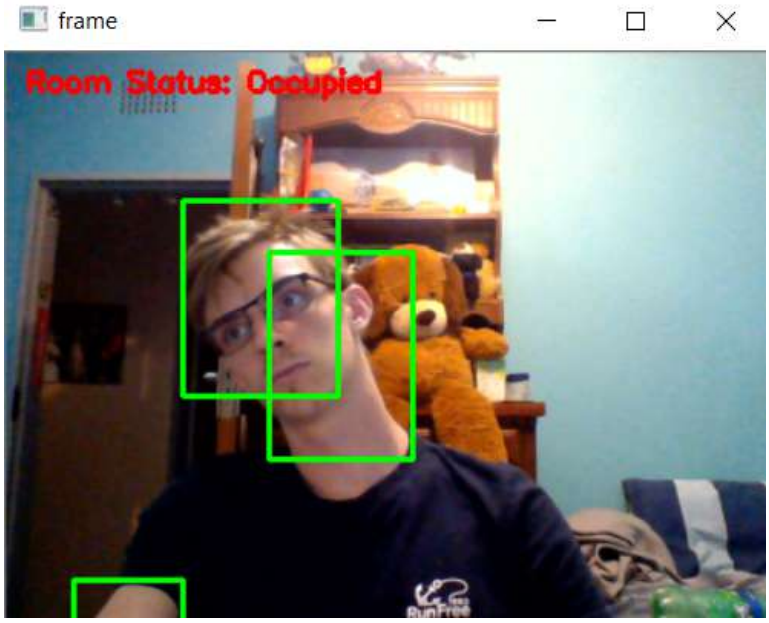


Figure 6.5: Built-in Background Subtraction Test

Table 4: Motion Tracking Speeds Comparison

Motion Tracking Method	Time per loop (s)
Built-in BS	0,04895
Manual BS	0,00143
Rolling Frame Subtraction	0,00052

* pixel width = 160, run on PC

Surprisingly the Rolling Frame is the fastest for each iteration. Both the manual algorithms greatly outperform the *MOG* algorithm. It should be noted that these values will change when implemented on the pi.

6.5 Threading Comparison Evaluation

Purpose Testing if threading is needed to incorporate into the software.

Set-up From *pyimagesearch.com* the FPS of the webcam can be increased by making a thread in the software that handles the incoming frame. From the script in [25], the FPS with and without threading can be determined quantitatively.

Results The table below shows the percentage increase with the use of threading for different image sizes and whether the display is sent over the Wi-Fi or not.

The smallest size with no display makes the greatest difference when using threading.[1]

Webcam Threading Speeds			
Pi - With Display		Pi - Without Display	
Width: 120px		Width: 120px	
pi@raspberrypi:~/examples $ python3 fps_demo.py -d 1		pi@raspberrypi:~/examples $ python3 fps_demo.py	
[INFO] sampling frames from webcam...		[INFO] sampling frames from webcam...	
[INFO] elasped time: 6.93		[INFO] elasped time: 5.46	
[INFO] approx. FPS: 14.44		[INFO] approx. FPS: 18.30	
[INFO] sampling THREADED frames from webcam...		[INFO] sampling THREADED frames from webcam...	
[INFO] elasped time: 4.84		[INFO] elasped time: 2.66	
[INFO] approx. FPS: 20.64		[INFO] approx. FPS: 37.56	
Increase	**143%**		**205%**
Width: 400px		Width: 400px	
pi@raspberrypi:~/examples $ python3 fps_demo.py -d 1		pi@raspberrypi:~/examples $ python3 fps_demo.py	
[INFO] sampling frames from webcam...		[INFO] sampling frames from webcam...	
[INFO] elasped time: 28.32		[INFO] elasped time: 6.23	
[INFO] approx. FPS: 3.53		[INFO] approx. FPS: 16.06	
[INFO] sampling THREADED frames from webcam...		[INFO] sampling THREADED frames from webcam...	
[INFO] elasped time: 8.49		[INFO] elasped time: 4.19	
[INFO] approx. FPS: 11.78		[INFO] approx. FPS: 23.88	
Increase	**42%**		**149%**

Figure 6.6: Webcam Threading Comparison Table

[1]The code started getting irregular behaviour when threading the camera, therefore further tests are conducted without it, but with a smaller pixel width

6.6 Test Set-up/Deployment

Figure 6.7: Hardware testing set-up

Before the testing can commence, the turret first needs to be deployed (Figure 6.7). This includes getting power from mains voltage via an extension cord, powering the servos and the pi.

Using *MobaXterm*, the PC can SSH to the pi via a Wi-Fi/Ethernet. The shell can then be used to run the program with the command line. MobaXterm is used specifically due to its X functionality. X meaning explorer. This implies that the visual output can be display on the PC from the pi, this is necessary for calibration.

The turret is calibrated for the area by setting upper and lower limits; then the testing can commence.

6.7 Human Target Test (Anticipation)

Purpose Measuring hit rate with a human target with anticipation enabled.

Set-up The target moved left to right on the screen at a relatively slow but constant velocity. The distance between the turret and the target is approximately 4 meters.

Results Two out of five shots fired hit the target. This gives a **hit rate up to 40%**. This test should be repeated for a few iterations to give a more precise value, but too many variables that can not be controlled becomes problematic. Which is why the following Tyre test comes into play. [2]

Figure 6.8: Human Target Test Video *(Click the image)*

6.8 Rolling Tyre Test

Purpose To eliminate some human factors apparent in previous testing.

Set-up A tyre is placed at a distance of 2m from the turret to have the same surface area representation as a person at 4 metres. By rolling a tyre from the left view to the right as depicted in Figure 6.8, a more standardized test is constructed. The test ensures a constant velocity and distance from the turret. This gives a general performance should any future comparison be made.

[2] At first the software would've been compared anticipation vs. no anticipation, but no anticipation was not even viable, unless at very close distances.

Results The test was repeated 4 times to get a rough estimated on how precise the turret is. These results are tabulated in Table 5 below. An average of all the hit rates from the 4 iterations gives a **precision of 60%**. This is a 20% increase from the human tests.

Figure 6.9: Rolling Tyre Target Test Video *(Click the image)*

Table 5: Tyre Test Results

Shots Hit	**Shots Fired**	**Hit Rate**
4	5	0,8
2	4	0,5
2	4	0,5
3	5	0,6
	Precision:	0.6

6.9 Cellphone Control Test

A unique feature of the turret is the ability to control the turret with a smart-phone. Because the turret uses a Raspberry Pi Model 3, it includes Wi-Fi capabilities. This means

that a smart-phone with an SSH terminal such a *Dataplicity* can connect to the turret from anywhere in the world over the internet. Provided that both devices are connected to the internet/network.

This enables the turret to be started and controlled without the hassle of a large laptop and additional wires. The terminal (Figure 6.10) on the phone allows for Linux bash commands to be sent.

The only downfall of using this terminal is that it does not have X functionality, meaning the display cannot be visualised on the phone, making any form of calibration difficult.

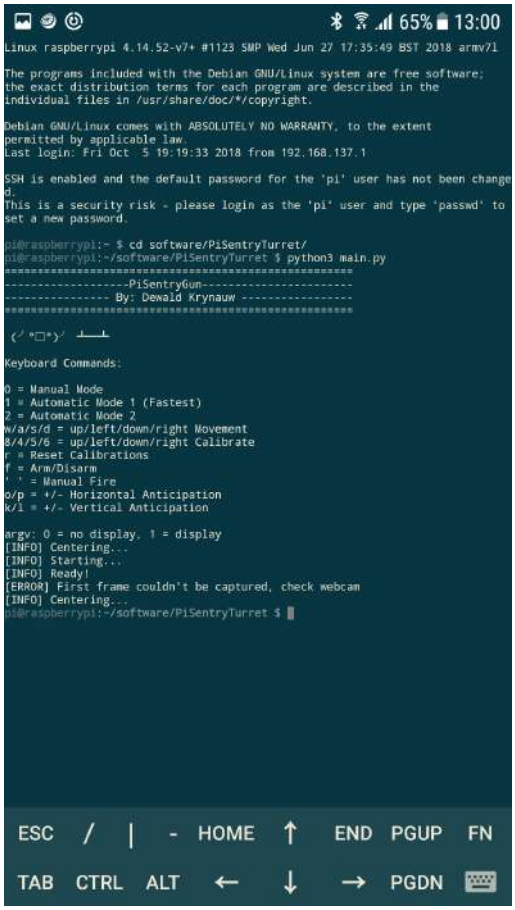


Figure 6.10: Cellphone Wireless SSH

6.10 Overall Performance Evaluation

The most crucial aspect of the turret's performance is that it must hit a target at least once if the target is in the area of fire for it to work as intended, or else the intruder will enter the property without being adequately warned. As the prior tests have concluded is that this can be expected from the turret under certain conditions.

The performance of the turret can be summarised as follows:

Table 6: Overall Functional Performance

Function	**Performance**
Maximum Range	5m
Rate of Fire	2 rounds/s
Human Hit Rate	40% at 4m
Test Precision	60% at 2m
Field of view	50 degrees

It should be noted that the tests had a very specific set of constraints to adhere to which is not always applicable in real life.

These real-life factors include:

- Distance from turret (further than 5m)
- If the target is running to or from the turret
- How large the person is (Turned sideways or not)
- Calibration and sensitivity parameters of the turret
- Changes in speed or acceleration of the target

The turret functions well enough for it to be a viable solution, but can still greatly improve in certain aspects, as covered in the next section.

7 Improvements

Just as no system is without flaws, the turret will have a few shortcomings. So long as the turret does not have a 100% hit rate it could always improve more.

For the hit rate to be improved, why the turret is not accurate should first be narrowed down to a few factors. The largest contributor to the hit rate is *latency*. In other words; the time it takes since the visual input is received, to hitting the target. During that latency period, the target might have moved a relative distance.

In this section, some possible changes that can be made to improve the hit rate and overall performance of the turret will be elaborated upon.

7.1 Mechanical Improvements

The mechanical construction is the most exhaustive contributors in the latency aspect.

Issues Some mechanical factors that negatively affect the hit rate include:

- Small tolerances on the gears causing even larger offsets the further the target is.
- Gears jumping teeth.
- The weight of paintball gun
- The slow firing rate of paintball gun

Improvements Gears with larger teeth size (pitch) would reduce the tolerances as well avoid the gears jumping over each other when the servo moves too fast. But this is a very costly improvement because the gears would then need to be custom-made and the frame should be adjusted to fit new gears.

Additions The best way to ensure hitting the target more, would be to get an automatic paintball gun or an electric trigger. The automatic feature would give a faster rate of fire, increasing the number of bullets hitting the target, but will consume more ammo.

An e-trigger would replace the trigger servo and reduce the time it takes for the trigger to be pulled. Currently, the trigger takes around 100ms to pull. A person walking at $1.4m/s$ would already displace 14cm just in the time to pull the trigger. This is compensated with anticipation, but would still be a better solution if an automatic gun is not available.

7.2 Electronics Improvements

Issues A few electronic issues that can be addressed include:

- Slow image processing
- No depth perception
- No night vision

Improvements These issues can not be addressed unless there is a larger budget, therefore they are regarded as additions.

Additions Processing can be increased with a faster CPU inside the Pi, this is not possible to change, unless entirely different SBC is used, such as a UP Board. Incoming frames from the webcam have already been resized to an extent that reduces the range the turret shoots.

Currently the web-cam is unable to see depth, making it difficult for the software to track motion if a target is moving towards and away from it. One improvement would be to use a Stereoscopic Camera (Figure 7.1) that is able to see depth and calculate distance. This solution would make the software more complex, but could enhance the performance.

Figure 7.1: Stereoscopic Camera (Xbox Kinect)

There are plenty ways to give the turret night vision capabilities, these include: Infra-red sensors (Figure 7.2), removing the IR filter on the web-cam, thermal cameras (Figure 7.3 and even IP cameras.

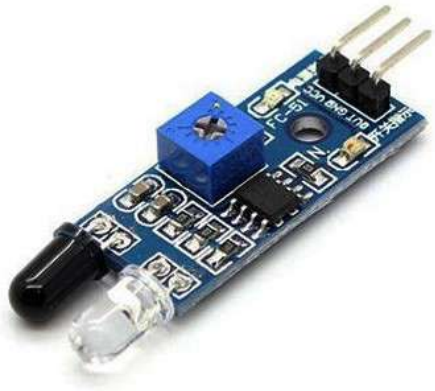


Figure 7.2: Infra-red Sensor

Figure 7.3: Thermal Camera

7.3 Software Improvements

Software is functional within design and working constraints. Additions would only feature as ”nice to have”.

Additions

- A trained neural network that can only track humans
- Facial recognition for trusted users
- Better algorithms that track motion, such as using HSV instead of gray-scale
- Joystick control
- Self/Assisted calibration
- Linking the turret to cloud, that notifies owner if the system motion has been detected
- Save video snapshots to the cloud server
- Bullet counter, to know when to refill.
- Additional camera for the gun point of view

8 Conclusion

As rampant crimes rates increase in South Africa and criminals break entry with lethal intent more regularly; a dire need to defend ourselves has arisen.

The Paintball Sentry Turret was proposed as a solution. It provides a non-lethal way of safeguarding a person/property without putting its owner at risk.

The turret tracks motion via a webcam, processes the image information on a Raspberry Pi 3 which in turn translates the pixel coordinates to servo positions. Servo motors (pan, tilt and trigger) rotate the paintball gun into position and fires if the system is armed.

Anticipation had to be implemented for compensation due to time lost in processing and mechanical movement.

The turret delivered a 60% hit rate during in a controlled test environment. Up to 40% in human target tests, making the turret functional given its constraints.

Improving the amount of paintballs that hit a target; one solution would be to get an automatic paintball gun that increased the rate of fire and the area it shoots. Another improvement would be to make the turret function at night as well because most crimes are committed at night.

9 Adherence to ECSA Exit Level Outcomes

The project has to address various Engineering Council of South Africa (ECSA) Exit Level Outcomes (ELO) as described below,

ELO 1: Engineering problem solving Problem solving was applied throughout the entire project. The whole project designed to address the issue of crime rates in South Africa. During the building process, other shortcomings also needed to be solved for the turret to be functional like, bugs in the code and power supply problems.

ELO 3: Engineering Design and Synthesis A customised systems engineering design approach was taken which included preliminary and detail design of the mechanical, electronics and software aspects of the turret. Available resources were used to implement the solution.

ELO 4: Investigation and Data Analysis Investigations were conducted to provide feasible solutions during the literature study, preliminary and detail design. Data was acquired from testing and the results were analysed.

ELO 5: Engineering Methods, skills, tools and information technology Tools such as Github was used for version control of software. Smart IDEs like Pycharm created a much easier development environment. Computer aided design software like *Solidworks* was used to visualise viable solutions.

ELO 6: Professional and Technical Communication Professional communication was used for presentations, technical documentation and the poster.

ELO 8: Individual Work The research, design and implementation of the Autonomous Sentry Turret was completed individually.

ELO 9: Independent Learning Ability A literature study was performed to possible sentry gun solutions. Usage of the Python programming language and OpenCV library was learned independently of the university.

ELO 10: Engineering Professionalism Having weekly meetings, submissions, feedback and communication with the supervisor were addressed professionally. Presentations were also given during the course.

References

[1] H. Ahmad, S. M. Ali, U. A. Sheikh, Z. Murtaza, and M. Rizwan, "Design and manufacturing of a low-cost 2-DOF autonomous sentry gun," in *2016 2nd International Conference on Robotics and Artificial Intelligence, ICRAI 2016*. IEEE, nov 2016, pp. 196–201. [Online]. Available: http://ieeexplore.ieee.org/document/7791253/

[2] A. Gagliardi, D. Gagliardi, and S. Project, "Autonomous/remote pilot sentry gun platform," Ph.D. dissertation, 2013. [Online]. Available: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1243{&}context=eesp

[3] M. Tsourma and M. Dasygenis, "Development of a Hybrid Defensive Embedded System with Face Recognition," *Proceedings - 19th IEEE International Conference on Computational Science and Engineering, 14th IEEE International Conference on Embedded and Ubiquitous Computing and 15th International Symposium on Distributed Computing and Applications to Business, Engi*, pp. 154–157, 2017.

[4] S. Shue, C. Hargrove, and J. Conrad, "Low cost semi-autonomous sentry robot," in *Conference Proceedings - IEEE SOUTHEASTCON*. IEEE, mar 2012, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/6196937/

[5] Hacker House, "Raspberry Pi Motion Tracking Gun Turret," 2016. [Online]. Available: https://www.hackster.io/hackerhouse/raspberry-pi-motion-tracking-gun-turret-77fb0b

[6] R. Rudolph, "Project Sentry Gun," 2014. [Online]. Available: https://sites.google.com/a/rudolphlabs.com/project-sentry-gun/products/gladiator-ii-paintball-turret

[7] S. Vincent, "SentryTurret." [Online]. Available: https://github.com/steve-vincent/SentryTurret

[8] S. T. Shazali, W. L. Cheong, S. Mohamaddan, A. M. Kamaruddin, A. Yassin, and K. Case, "Motion detection using periodic background estimation subtraction method," in *2011 7th International Conference on Information Technology in Asia: Emerging Convergences and Singularity of Forms - Proceedings of CITA'11*. IEEE, jul 2011, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/5999523/

[9] D. Sawicz, "Hobby Servo Fundementals," 2007. [Online]. Available: https://www.princeton.edu/{~}mae412/TEXT/NTRAK2002/292-302.pdf

[10] J. Yao and J.-M. Odobez, "Multi-Layer Background Subtraction Based on Color and Texture," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2007, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/4270495/

[11] Z. Wu, W. Keatts, and A. Davari, "Low-cost motion detection and counter attacking test bed for swarm UAVs," in *Proceedings of the Annual Southeastern Symposium on System Theory*, vol. 37. IEEE, 2005, pp. 362–366. [Online]. Available: http://ieeexplore.ieee.org/document/1460937/

[12] B. Sarala, B. Swathi, A. S. N. Rani, and V. Maik, "Object tracking using block motion estimation with adaptive Kalman estimates," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, may 2017, pp. 751–754. [Online]. Available: http://ieeexplore.ieee.org/document/8256697/

[13] M. Zhao, J. Zhang, and R. Figueiredo, "Distributed file system support for Virtual Machines in Grid computing," *IEEE International Symposium on High Performance Distributed Computing, Proceedings*, vol. 130, no. x, pp. 202–211, 2004. [Online]. Available: http://cseweb.ucsd.edu/classes/sp02/cse252/lucaskanade81.pdfhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.2019{&}rep=rep1{&}type=pdf

[14] P. Kaewtrakulpong and R. Bowden, "An Improved Adaptive Background Mixture Model for Real- time Tracking with Shadow Detection," in *Advanced Video Based Surveillance Systems*. Kluwer Academic Publishers, 2001, pp. 1–5. [Online]. Available: http://personal.ee.surrey.ac.uk/Personal/R.Bowden/publications/avbs01/avbs01.pdf

[15] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2004, pp. 28–31 Vol.2. [Online]. Available: http://ieeexplore.ieee.org/document/1333992/

[16] A. B. Godbehere and K. Goldberg, "Algorithms for visual tracking of visitors under variable-lighting conditions for a responsive audio art installation," pp. 181–204, 2014. [Online]. Available: http://goldberg.berkeley.edu/pubs/acc-2012-visual-tracking-final.pdf

[17] S. Zeevi, "BackgroundSubtractorCNT." [Online]. Available: https://github.com/sagi-z/BackgroundSubtractorCNT

[18] H. Gill, "Stepper Motor or Servo Motor : Which should it be ?" pp. 1–8. [Online]. Available: https://www.kollmorgen.com/uploadedFiles/kollmorgencom/Service{_}and{_}Support/Knowledge{_}Center/White{_}Papers/ServoorStepper{_}08{_}05{_}16{_}WhitePaper{_}FINAL.pdf

[19] Michael Parks, "MCU, SBC or FPGA?" 2014. [Online]. Available: https://www.mouser.com/blog/picking-the-right-tool-for-the-job-mcu-sbc-or-fpga

[20] Engineering ToolBox, "Gears," 2008. [Online]. Available: https://www.engineeringtoolbox.com/gears-d{_}1307.html

[21] A. Rosebrock, "Basic motion detection and tracking with Python and OpenCV," 2015. [Online]. Available: http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/

[22] D. Liang and S. Kaneko, "Improvements and Experiments of a Compact Statistical Background Model," may 2014. [Online]. Available: http://arxiv.org/abs/1405.6275

[23] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier, "Background subtraction based on Local Shape," apr 2012. [Online]. Available: http://arxiv.org/abs/1204.6326

[24] Y. Sasaki, "The truth of the F-measure," *Teach Tutor mater*, pp. 1–5, 2007. [Online]. Available: http://www.cs.odu.edu/{~}mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf

[25] A. Rosebrock, "Increasing Raspberry Pi FPS with Python and OpenCV," 2015. [Online]. Available: http://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/

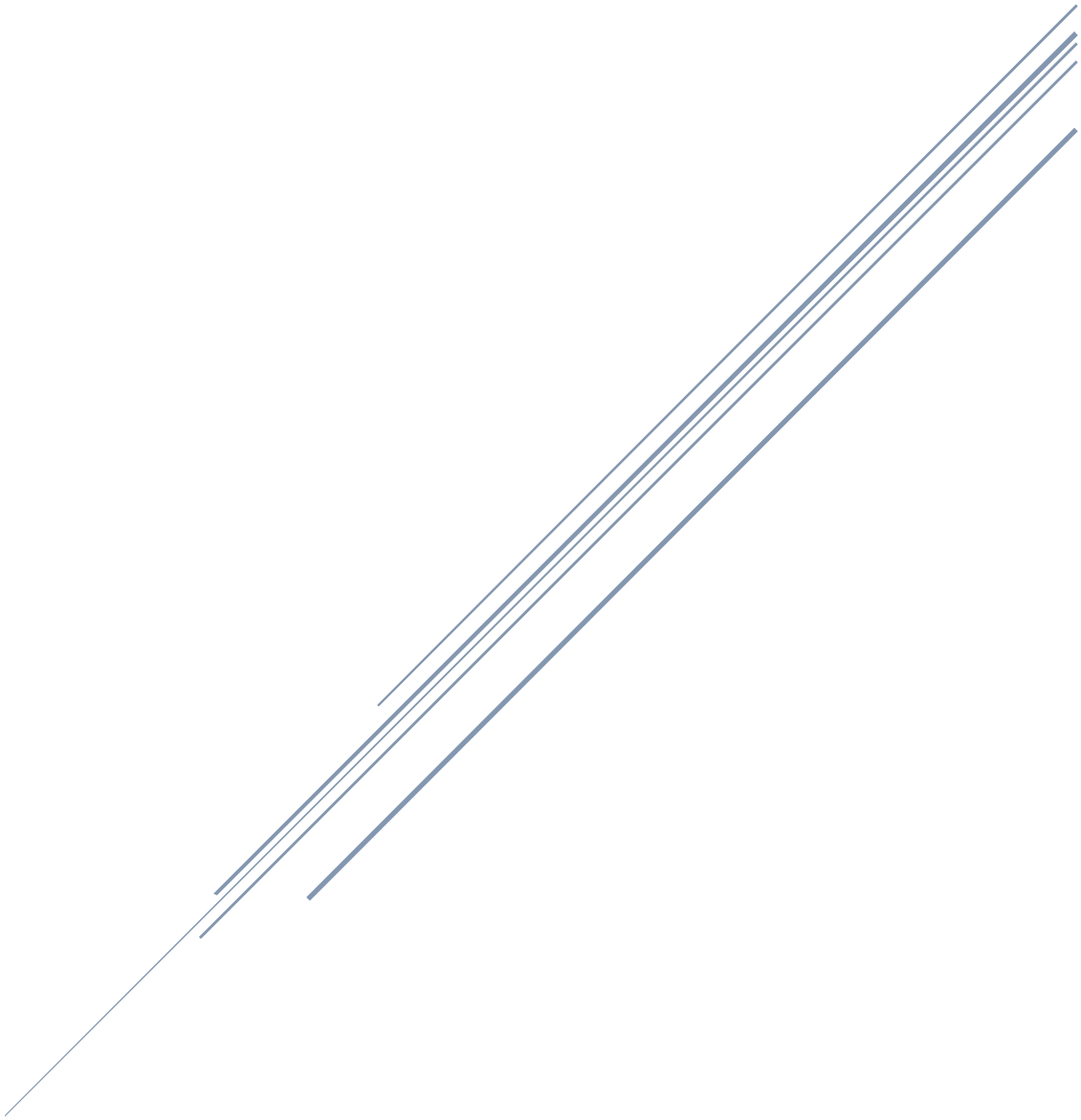
[26] R. S. Wolcott and B. E. Bishop, "Cooperative decision-making and multi-target attack using multiple sentry guns," in *2009 IEEE International Symposium on Sustainable Systems and Technology, ISSST 2009*. IEEE, mar 2009, pp. 261–265. [Online]. Available: http://ieeexplore.ieee.org/document/4806801/

[27] Raspberry Pi, *DATASHEET Raspberry Pi Compute Module (CM1) Raspberry Pi Compute Module 3 (CM3) Raspberry Pi Compute Module 3 Lite (CM3L)*, 2016, vol. 3, no. October. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/computemodule/RPI-CM-DATASHEET-V1{_}0.pdf{%}0Ahttps://www.sparkfun.com/products/13825

A Requirement Specification

SYSTEM REQUIREMENTS SPECIFICATION

Autonomous Paintball Sentry Gun



Dewald Krynauw
26013835

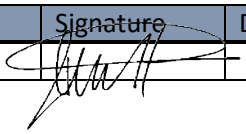
DOCUMENT IDENTIFICATION

Project Title:	Turret Requirements Specification – Autonomous Paintball Sentry
Document Number:	SyRS-SS_v1.1
Turret / Sub-Turret:	Autonomous Paintball Sentry Gun
Document Issue Date:	2018-03-17
Client:	Prof J. Holm – NWU
Client Reference:	Autonomous Paintball Turret

ORIGINATION AND APPROVAL

Checked by Party	Individual Name	Signature	Date
Author:	Mr. Dewald Krynauw		2018-03-17
Quantity Assurance:			
Technical Approval:	Prof J. Holm		
Project Manager:	Dr Leenta Grobler		

ACCEPTANCE

Checked by	Individual Name	Signature	Date
Approved by:	Prof J. Holm		

DISTRIBUTION LIST

Company	Individual Name	Date
NWU	Dr Leenta Grobler	
NWU	Prof J. Holm	

SECURITY LEVELS AND RESTRICTIONS

Level	Description	Applicable Level
1	Strictly Confidential – not to be distributed	
2	Company Confidential – distributed inside company	
3	Client Confidential – distributed to limited clients and contractors	X
4	Public Domain – distributed freely	

CONTACT INFORMATION

Contact Person	Mr. Dewald Krynauw
Company	NWU – EERI474 2018
Street Address	348 Patria Manskoshuis, 11 Hoffman Street, Northwest
Telephone Number	0737092927
Email address	Dewaldkrynauw123@gmail.com
Web site	None

DOCUMENT REVISION HISTORY

Date	Responsible person	Description	Revision No.
2018-03-17	Dewald	Document creation	1.0
2018-03-25	Dewald	Change the wording of a few statements	1.1

Table of Contents

1 Introduction and scope

1.1 Identification

This turret specification pertains to the Autonomous Paintball Sentry Turret being developed by the North West University (NWU).

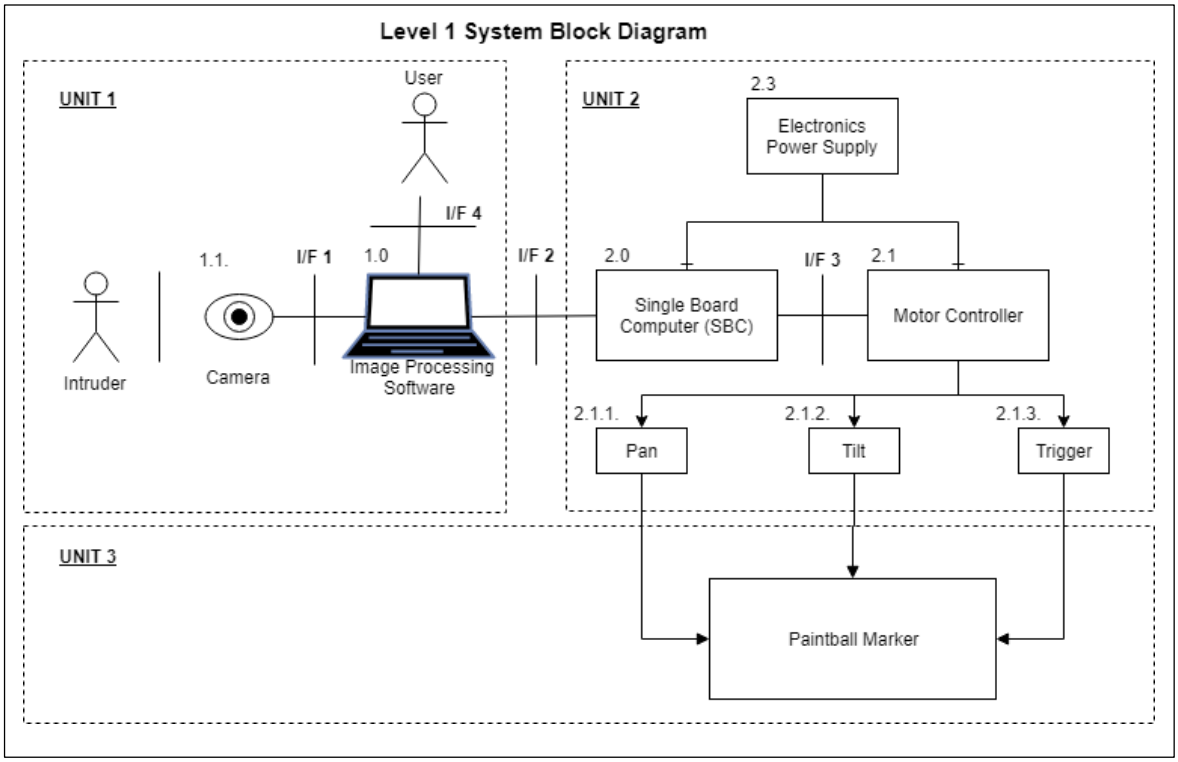
1.2 Intended use

The Sentry Gun is intended to be deployed on a privately-owned property to keep intruders with ill intentions from further entering the premises. Should the perpetrator continue to make his/her pursuit to enter the real estate the Sentry Gun should detect the intruder and fire non-fatal bullets continuously to maim the intruder until he/she leaves.

The Sentry Gun is intended to alleviate real guards from protecting a property and not having to risk lives trying to defend it. It also poses as an extra line of defence on any premises.

1.3 Background

1.4 Turret Overview



1.5 Document Overview and Use

This SyRS is intended to be used by the client and their appointed contractors to develop the Autonomous Paintball Sentry. Unless explicitly stated herein all contents of this SyRS is to be treated as client confidential by any contractors. At the discretion of the client this SyRS may be distributed to any party deemed to have a stake in the development of this turret or the management of the turret development.

2 Applicable and other referenced documents

2.1 Applicable documents

N/A

2.2 Other referenced documents

Unless explicitly states any requirement in this specification that is found to be in conflict with the referenced standards shall be considered to be subservient to said standard.

DOCUMENT IDENTIFIER	DOCUMENT DESCRIPTION

3 Meanings, Acronyms, and Abbreviations

3.1 Meanings

Unless otherwise explicitly states here all words and terms shall be interpreted as per the latest edition of the United Kingdom variant of the Oxford English dictionary.

TERM	DEFINITION
SHALL	Expresses a characteristic which must be present in the item of specification, thus a binding requirement
SHOULD	Expresses a goal or target to be pursued but not necessarily achieved
MAY	Expresses permissive guidance
WILL	Expresses a declaration of intent on the part of a party
STATE	The state of a turret refers to a state of being of the turret.
MODE	The mode of a turret refers to the state of doing of a turret. Typically modes are encapsulated within states.
SENTRY TURRET/ SENTRY GUN	Refers to the entire turret being constructed. Meaning: “guardian tower” or “guardian gun”. Can be used interchangeably.
PAINTBALL MARKER/GUN	The gun that is mounted inside the device.
SBC/MOTOR CONTROLLER	These are the same entities that control the motors. Although they are two parts in the turret overview, they are one entity. The SBC is a motor controller
OFF	No power is applied to the turret
DISARMED	The turret is powered on but will have no response
ARMED	The turret is tracking motion or being controlled manually by a user
FIRING	The paintball gun is being fired
IMAGE PROCESSOR/SOFTWARE	Is the physical computer or device that does the digital image processing

3.2 Acronyms

ACRONYM	DEFINITION
NWU	North West University
SYRS	Turret Requirements Specification

TBD	To Be Defined
SBC	Single Board Computer

3.3 Abbreviations

ABBREVIATION	EXPLANTATION
E.G.	For example
REQID	Requirement Identifier

4 Requirements

4.1 Identification of External Interfaces

4.1.1 Intruder Camera interface

The interface through which the intruder is detected and data sent to a processor.

4.1.2 Sentry-Intruder interface

The interface with which the gun shoots the intruder.

4.1.3 User to software interface

The interface which the user can adjust and control the behaviour of the turret.

4.2 Identification of States and Modes

The turret shall have the following states and modes as defined in Section 3.1

- State – Off
- State – Armed
- Mode – Autonomous
- Mode – Manual
- State – Disarmed
- State – Firing
- Mode – Automatic
- Mode – Semi-Automatic

4.3 Turret Function and Performance Requirements

4.3.1 Process digital camera input.

The turret must detect and track any movement received from camera and should send coordinates to motor controller. It is not required to only detect humans, but rather any large movement. **REQID 1**

4.3.2 Move and fire paintball marker

The turret should move the paintball marker in the direction of the coordinates and fire the bullets. **REQID 2**

4.4 Relationships between States and Modes

TRANSITION	REQUIREMENT	RESPONSE
OFF – DISARMED	On switch pressed	Start bootup sequence
DISARMED – ARMED	If a mode is selected	Enter autonomous or manual mode
ARMED.MANUAL	If manual mode button clicked	Gives control to user

ARMED.AUTONOMOUS	If auto mode button clicked	Start image processing algorithms
ARMED – FIRING	If large motion is detected or fire button clicked	Start pulling trigger in the specified direction

4.5 Turret External Interface Requirements

4.5.1 Intruder Camera interface

The camera shall provide the image processor with a stream of digital images at a high frame rate. **REQID 3**

The camera may take visible images during night and day. **REQID 4**

The camera may have a low resolution. **REQID 5**

4.5.2 Sentry-Intruder interface

The gun-intruder interface shall provide the means of maiming the intruder with paintball bullets **REQID 6**

The turret will provide the means to automatically shoot moving targets. **REQID 7**

4.5.3 User to software interface

The user-software interface shall provide the user with the means to select between an autonomous or manual mode. **REQID 8**

The user-software interface shall provide the user with the means to select between an automatic or semi-automatic firing mode. **REQID 9**

The user-software interface shall provide the user with the means to control the paintball gun from the software. **REQID 10**

4.6 Turret Environmental Requirements

4.6.1 Classes of environment

For the purposes of this SyRS only the operational environment is defined, with transportation and storage environments being contained within the parameter envelopes of the operational environment.

4.6.2 Operational Environment

The Sentry will be operated outdoors under natural of artificial light. **REQID 11**

The Sentry must not be operated in rainy weather. **REQID 12**

The Sentry should not be moved or receive high speed winds while being operated. **REQID 13**

The Sentry should not be operated in an area with a constant movement. **REQID 14**

The Sentry should be mounted on a fixed and relatively flat surface **REQID 15**

4.7 External Resource Utilization Requirements

The turret shall consume a maximum of 130 W of electrical power. **REQID 16**

The turret will hold a standard paint ball gun. **REQID 17**

The turret shall expend paintball bullets at a rate of 2 bullets per seconds. Some may be recovered if solid bullets are used. **REQID 18**

The turret shall consume 20oz CO2 gas per 1000 shots. **REQID 19**

4.8 Turret Physical Requirements

The turret must not weigh more than 15kg.

The turret may be black to be more hidden at night.

The turret should look neat and professional

The turret should not be taller than 1.5m.

The turret should be stable as to not move the camera.

4.9 Other Turret Qualities

Version control of software will be done in git by the developer.

The Sentry shall exhibit high quality workmanship insofar as cabling and wiring is concerned as per the NASA workmanship standards.

For all software configuration items (Application, Server, or Embedded) the developer shall make use of a developer selected coding standard.

4.10 Design and Construction Requirements

4.10.1 General Design and Construction Requirements

The Turret should be optimised for speed and accuracy up to 10m. **REQID 20**

4.10.2 Characteristics of sub-ordinate elements

The turret shall not make use of advanced computer vision.

The turret may not have any IOT functionality

The turret is inherently dangerous and should be used with caution.

4.11 Precedence of requirements

All requirements stated herein are subservient to requirements of safety. Should the satisfaction of a requirement lead to the safety requirement being violated the contractor is required to notify the stakeholder.

5 Verification requirements

The following requirements will be subject to verification:

- The turret will provide the means to automatically shoot moving targets. `REQID 21`
- The user-software interface shall provide the user with the means to control the paintball gun from the software. `REQID 22`
- The Turret should be optimised for speed and accuracy up to 10m. `REQID 23`

5.1 Accuracy in Manual mode up to 10m.

The turret must have a manual mode. The person being shot will be stationary.

PERFORMANCE LEVEL	GRADE
IF THE TURRET CAN SHOOT A BULLET IN A DIRECTION VIA MANUAL CONTROL	40%
30% ACCURATE	50%
50% ACCURATE	60%
70% ACCURATE	75%
90%ACCURATE	90%
100% ACCURATE	100%

*under specific test conditions.

5.2 Accuracy in Autonomous mode up to 10m.

The sentry must have an autonomous mode. The accuracy measurement is sensitive to the time spent and the speed of the moving target, traversing over the field of view.

PERFORMANCE LEVEL	GRADE
IF MOTION CAN BE TRACKED IN SOFTWARE AND BULLET FIRED FROM TURRET	40%
10% ACCURATE	50%
20% ACCURATE	60%
30% ACCURATE	70%
40% ACCURATE	75%
60% ACCURATE	90%
>70% ACCURATE	100%

*under specific test conditions.

5.3 Constraints

All tests will be measured at constant speeds of the average person's walking speed.

The time spent in autonomous mode should be equal to the time for a large enough sample e.g. 30 bullets to be shot.

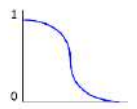
Tests may be conducted anywhere from 3 to 10 metres.

Due to the inaccurate nature of paintball guns, the size of the grouping should also be considered when determining the accuracy. This will be the control measurements.

5.4 Final Performance Mark

$$Final\ performance\ Mark = \frac{Autonomous\ \% + Manual\ \%}{2}$$

6 Value Model

Measure of effectiveness	Minimum acceptable	Maximum acceptable	Relative Importance	Utility function
Cost Sentry Turret excluding PC	R 3 000	R 7 000	100	
Reliability of Sentry Turret turret	60 %	99.99%	70	

B Code Archive