GENE121/MTE100 Final Project Report

Expected December 3, 2018

Group 865

James Lin 20785133

Michael Nasello 20772924

Brian Windhaus 20762950

Jing Hao Yao 20761481

Abstract

In the final months of the Fall 2018 term, the 1A Mechatronics class was tasked to design and build a system which solved a real-world problem. This report details the project which tackled invasion of personal space using an automated deterrence system, with a design focused on detection, controlled fire, autonomy, and stability. This design consists of a rectangular base supporting the *LEGO MINDSTORMS EV3®.* The system relies on a front-wheel drive-based rotation. 2 motors are placed at the front, with high-traction wheels. The rear component is lifted by a ball caster, allowing it to follow in rotation. Centered above this is a long barrel dimensioned to loosely fit a regular store-bought 1-inch marshmallow. Attached atop the barrel at a slant is gravity-reliant magazine for loading projectiles. This system can monitor a 180 ° arc while ignoring common desk objects such as water bottles. Additionally, it is capable of firing a marshmallow over 250 cm, although only requires detection around 120 cm. The software uses 6 main functions. A startup function aims to improve user experience. A rotation function works in tandem with an intrusion detection function and firing function. An ammunition check function prevents common errors such as blank-fires and running when not required. Lastly, a discharge function to prevent any further accidents. The project went through 4 major design changes to better meet the criteria mentioned above, and the tradeoffs and design decisions of each are mentioned below. The system overall performed well on testing and proved equal to the challenge it aimed to tackle.

Table of Contents

Table of Figures

1 Introduction

1.1 Report Overview

This report details the purpose and functionality of the proposed solution. It analyzes the timeline of the project, with details regarding the tests and revisions that were made to the design to ensure that it can perform its desired tasks effectively. The constraints and criteria that were used to evaluate the product are also discussed alongside the mechanical and software designs and decisions for the solution.

1.2 Problem Description

The targeted problem for the solution is the need to defend one's own personal space. This need exists for the user as they may not always be aware of their surroundings, and even if they were, they may not have the effective means to prevent others from entering their personal space.

1.2.1 Stakeholders

The main stakeholders involved with the project is the user, the approaching person and the bystanders who may be observing the situation. In terms of the user, they are benefited by the solution by having a more secure personal space. The intruder on the other hand serves as the main point of interaction between the solution and the environment, where they are affected negatively through a means of deterrence that the solution offers. Additionally, any bystanders can be considered stakeholders as they may be alerted by the robot and act upon any dangerous situation as they see necessary.

1.2.2 Problem Importance

Maintaining personal space is vital for a variety of reasons. Firstly, breaching personal space may lead to anxiety and stress, or if the person is focusing on work, a breach can lead to lower productivity as the user is distracted. Defending personal space can also be a matter of safety, especially if the person is approaching with malicious intent. In this case, the product could serve as a self-defense tool, providing deterrence and lowering any risks. Even if there is no immediate threat to the user's mental and physical health, the solution can provide a means of assurance to the user, which can lead to a greater level of comfort for the user in public spaces.

2 Project Scope

2.1 Main Functionality

The product has three main tasks that it must complete to protect the user: area surveillance, intruder detection, and intruder deterrence. When these tasks are completed, the user is protected and can proceed without the worry of threats of his/her environment.

2.1.1 Area Surveillance

To protect the user, the robot will observe a predetermined area that is controlled in the main program of its source code. Horizontal rotation is necessary so that the robot can observe an arc rather than a single field of view, allowing for a larger area of the client's surroundings to be protected.

2.1.2 Intruder Detection

While the robot rotates, it will observe the field of view and search the area for intruders. When an object is detected, the robot will abruptly halt. To differentiate between stationary objects and those that are approaching, an algorithm will run. If it is determined that the object indeed is approaching the client, the firing process will begin. Otherwise, the program will continue its detection.

2.1.3 Intruder Deterrence

When a target has been identified, the ultrasonic sensor will activate the firing process. Marshmallows will periodically and consistently be launched from the barrel of the launching apparatus. Upon an initial firing and after the release of subsequent rounds, marshmallows will automatically be reloaded into the firing barrel until the ammunition is depleted.

2.1.4 Additional Tasks

The robot performs a discharge of its ammunition when the program begins or terminates. If the robot's program has ended or the the user decides to discharge the robot upon starting the robot, the gear with which rubber bands are drawn back onto will complete a full revolution to launch all remaining marshmallows in the magazine.

2.2 Measurements and Detection

The robot requires several sources of data in order to protect the client. It requires human feedback to control the start and end of the program. There is a continuous input from the gyrometer, colour sensor, motor encoder, and ultrasonic sensor that help the *LEGO MINDSTORMS EV3*® brick adjust to its environment.

2.2.1 Human Input

As the system itself consists of a majority of autonomous functions that are operated by various input values from sensors, it requires very little human interaction during operation. The initial start-up requests that the user selects a mode of operation via the *LEGO MINDSTORMS EV3*® brick. It will then run until one of the stopping flags is triggered. The only inputs that are mandatory from the user are the powering of the brick and starting the program.

2.2.2 Gyrometer

The gyrometer is used to control the rotation of the robot's chassis. The robot is responsible for observing an area set by the user. In order to do this, the robot must rotate

in a controlled manner and run a complete sweep of that area periodically. The gyrometer assists this task by inputting the angle at which the chassis points relative to its starting position. Using this data, the *LEGO MINDSTORMS EV3*® brick can control the robot's rotation so that the product properly surveys that area.

2.2.3 Ultrasonic Sensor

The ultrasonic sensor is responsible for detecting objects in the area that it surveys. This is done so by continuously computing the distance of the farthest object. If the sensor inputs any changes to this data, the system assumes there is an intruder and calls the appropriate functions.

2.2.4 Colour Sensor

The colour sensor is used to determine when the magazine is depleted of its ammunition. It functions by looking for the colour white at the bottom of the magazine. If no white material is found, the system assumes there is no more marshmallows and ends the program.

2.2.5 Motor Encoder

The encoder on the firing apparatus' motor is used to set the angle at which the gear turns when given power. This is used to release rubber bands that are drawn onto the pegs of the gear individually. This differs from the original use of encoders as specified on the Interim Report and was changed to accommodate a new mechanical design.

2.3 External Interactions

Apart from functions within the robot, the system requires interactions with the rest of its environment to carry out its desired functions.

2.3.1 Mobility

The robot interacts with the surface on which it resides via two wheels that grip onto the surface and rotate upon a single axis. A ball caster on the back end of the chassis is used to support the weight of the robot.

2.3.2 Ballistics and Projectile Use

To deter any intruders, the robot fires a marshmallow by releasing a rubber band that is drawn back behind the round. In this case, the interaction is the collision between the intruder and the marshmallow. Additionally, the robot interacts with marshmallows both by launching them and storing them in a magazine on the top of the firing assembly.

2.4 Program Termination

The program should terminate under two conditions. The user is able to halt the program at any time by pressing the enter button if they no longer require its services. Additionally, the robot will continuously check the magazine for marshmallows. If the

colour sensor determines that the system is depleted of ammunition, after a brief waiting period, it will complete the discharge task and then end the program.

2.5 Changes in Scope

Originally, the robot was to deter any intruders by the rapid launch of marshmallows. Due to restraints of motor power, the original design could not meet the criteria outlined in the Interim Report [1]. It then became necessary to change the scope of the project so that the robot was only responsible for intruder deterrence by short bursts of fire, like a shotgun. This was more within the range of capabilities of the hardware provided.

3 Constraints and Criteria

3.1 Summary of Constraints

The section below outlines the constraints that were applied to the design of the system during planning.

3.1.1 Consistency of Motion

For the robot to perform surveying tasks accurately, the distance between the robot's initial position measured at the ball caster and its position after a rotation of 180 ° forwards and backwards had to remain within ± 2.0 cm. This constraint was established to ensure that the system can consistently guard the user.

3.1.2 Accuracy of Detection

To prevent misfires and wasted ammunition on non-threatening objects, the robot is to only fire at detected objects that are approaching the robot. Stationary objects or objects that are moving away should not be fired at.

3.2 Summary of Criteria

The section below outlines the criteria that were applied to judge the design of the system. Each had been set during the planning of the system as a core component in tackling the problem of personal space invasion.

3.2.1 Detection

For an effective design, the robot must be able to detect approaching objects accurately so that the firing mechanism can more accurately launch a projectile. This is categorized in two main measurable factors. The first factor is how accurately the system can rotate 180 ° consistently when completing half-revolutions in a surveying arc; this is quantified as the angle deviation from the endpoints of the 180 ° arc. The second factor is how accurately the system can stop upon detecting an object, which is measured as the angle between the orientation of the robot when an object is detected and where the robot stops its rotation.

3.2.2 Controlled Fire

Upon the ultrasonic sensor detecting an object and categorizing it as a threat, the robot should fire a projectile at the object and the projectile should travel a minimum distance of 1.2 meters before hitting the ground. This is to ensure that any object within the entire circle determined by the given radius will be deterred. The firing apparatus should also be able to launch within the general vicinity of the object to maximize potential deterrence. This area will be defined as 20 centimeters to the right and left of the intrusion.

3.2.3 Autonomy

Ideally, the robot will be fully autonomous and require no human interaction. The robot serves the need of an individual who needs protection from their surroundings as their attention is focused on their work. Constant maintenance of the robot requires the user to leave their work which defeats the purpose of the robot. Thus, it is a vital part of the robot's functionality that it be autonomous. The more feasible design will require the least human interference and never require the client to leave their during its program.

3.2.4 Stability

The mechanical system of the robot should be able to support the LEGO EV3 Brick along with the other assemblies on the robot. Ideally, the robot should not collapse under any circumstances and a design that performs better when judged against this criterion will have a greater number of successive revolutions and rubber band releases before breaking or collapsing. For the robot to perform all its tasks effectively, the robot should be able to complete 20 revolutions and 3 successive rubber band releases before collapsing.

3.3 Evaluation of Constraints and Criteria

The constraint and two criteria that guided the design process most significantly were accuracy of detection, stability, and detection respectively. Every task was dependent on balance and support from the main body of the system, so consequently, every early iteration of the mechanical design had its focus on stability. The system's ability to detect approaching objects accurately was also critical to its main task, the deterrence method. This resulted in much time and effort spent on developing, testing, and refining the software to ensure the robot could effectively differentiate approaching objects without impeding its further tasks.

Controlled fire was another difficult criterion to meet due to the overall problem this system would tackle, intrusion deterrence. Making sure the robot could fire 1.2 m was vital for the robot to guard the entirety of the user's personal space. The criterion also encompassed the sustainability and efficiency of the robot. Ensuring that the robot only fired a single projectile preserved ammunition and prolonged service time, which was important because of the magazine limit.

The remaining constraints and criteria played a smaller role in our robot design. The constraint for consistency of motion was easily met with simple design options in both

software and mechanical design. Specifically, in the final design, the choices were to incorporate a 2-wheel design while applying opposite motor powers to each respective wheel to rotate it upon a single pivot. In terms of autonomy, this criterion was met to a high standard with basic software decisions. For example, to automate the robot so that it would continue rotating after launching a marshmallow, a simple while loop was added so that the robot would return to rotating and not require another startup.

4 Mechanical Design

The mechanical design of the system proved to be the most challenging, due to the criteria and constraints stated above. The following section describes the design of out robot using sketches, drawings, and photos as appropriate.

4.1 Overall Design

The design of the robot consists of a large base that holds the LEGO EV3 brick at the rear along with the rest of the assemblies needed for the robot to function. Two front wheels and a ball caster support the base and prohibit movement of the robot.

The firing mechanism consists of a long barrel with three rubber bands tied to a rod at its front. These rubber bands are drawn onto the pegs of a gear at the back of the barrel, which is connected to a motor. As the motor is given power, the gear spins towards the front of the barrel causing the rubber bands to release successively. This is the fundamental operation of the firing mechanism.

The magazine is a flexible Lego structure fastened onto the barrel near the backside of the robot. Marshmallows are stacked on top of each other within the cartridge, with the bottom marshmallow sitting in the barrel. As the rubber bands are released and the bottom marshmallow is launched, the stack drops. This action replaces the marshmallow round that was fired.

The ultrasonic sensor is mounted onto the front of the robot. The color sensor is attached to the back of the magazine; it is facing the barrel so that it can detect any marshmallows sitting in the barrel. The gyrometer is placed near the back and bottom of the robot’s chassis.

4.2 Design Elements

Each component was refined many times to satisfy the criteria outlined in section 4.2. They had to be altered on numerous occasions so that when assembled, the parts would work in harmony. This section outlines the specifics of the robot’s mechanical design.

4.2.1 Chassis

The chassis was built to provide support to vital components of the upper body design, including the firing mechanism, magazine, and detection apparatus. It comprises of several cross beams attached to rectangular support structures. This foundation is

responsible for assisting with the rotation of the robot via two motors mounted to the front lower-body. When necessary, the motors would receive an appropriate power and rotate the robot about a center axis.

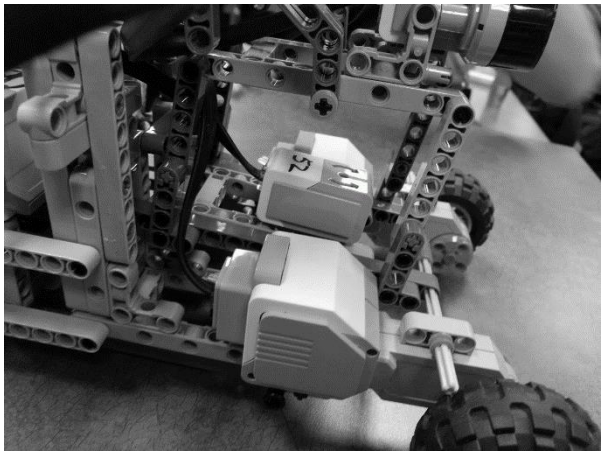


Fig 4.0 Chassis Front Side View The picture portrays the assembly of the motor wheels on the front end of the robot, as well as part of the mechanical composition of the chassis.

A pivot ball is centered at the rear. This allowed for the base of the robot to remain suspended for rotational movement. Resting on the rear of the chassis is the LEGO EV3 brick. This is a central location where the brick is within close enough proximity to each sensor; this allows the cables to connect without stress put on the terminals.

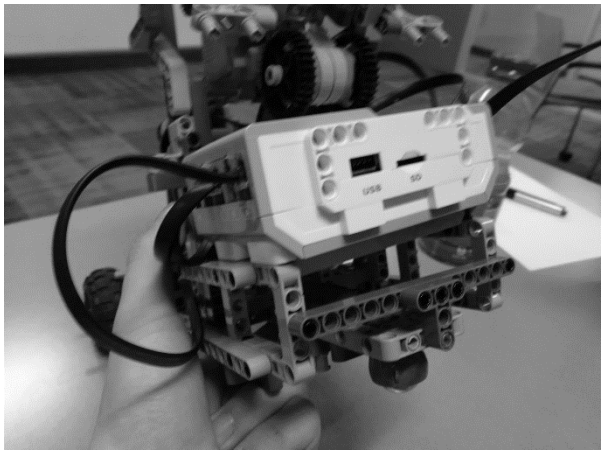


Fig 4.1 Back View of Chassis This displays the back assembly of the robot, specifically how the LEGO EV3 brick is mounted and the ball caster used to support the back weight of the robot.

4.2.2 Barrel

The barrel of the robot consists of multiple long LEGO pieces forming a platform confined by walls on either side. The barrel itself is raised at an angle of 45° to the plane of the surface and is supported by multiple vertical support beams.

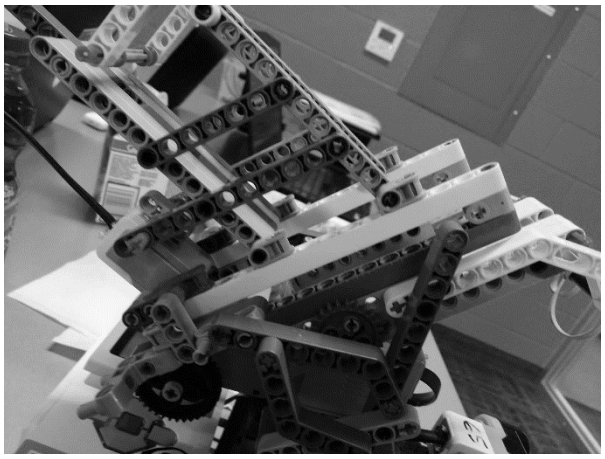


Fig 4.2 Barrel View
The side view of the whole barrel along with its composition of Lego pieces that were used to assemble it.

A small ridge was placed near the inside of the end of the barrel to prevent marshmallows from rolling down through the back. Due to the elasticity of the rubber bands, the barrel experiences a considerable amount of tension when loaded. To hold the barrel in place and to make the structure more rigid, clamps made using multiple interlocking LEGO pieces were placed at the front and back of the barrel.

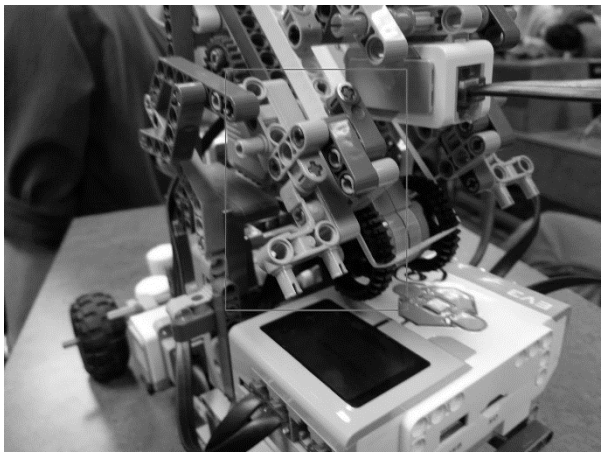


Fig 4.3 Back Clamps
The picture shows the design of clamps outlined as well as the back end of the barrel.

4.2.3 Firing Assembly

The firing assembly is attached to the base of the robot chassis and consists of three elastic bands and two gears on an axle through the motor. They are fixed onto the gears at five teeth intervals, which spanned 45°. At the other end, the bands are fastened to a rod at the front of the barrel, angled 45° below the firing plane.

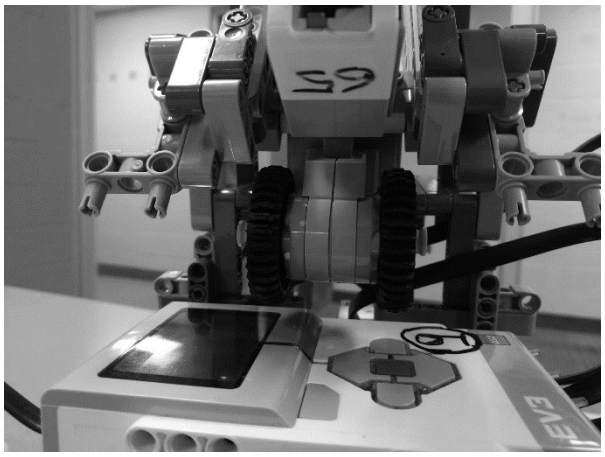


Fig 4.4 Firing Gears
The figure depicts the gears positioned at the back of the barrel which have the rubber bands drawn back onto.

A signal is sent from the detection code to the motor encoder, engaging a rotation of 45°. This releases one cord and uses the elastic potential energy to propel the ammunition forward. Since the path of the cord was designed to be angled downward, it contracts beneath the firing plane, out of the way of the next projectiles path. As the cord is now out of the trajectory of the next projectile, consecutive firing is achieved without interference of other components.

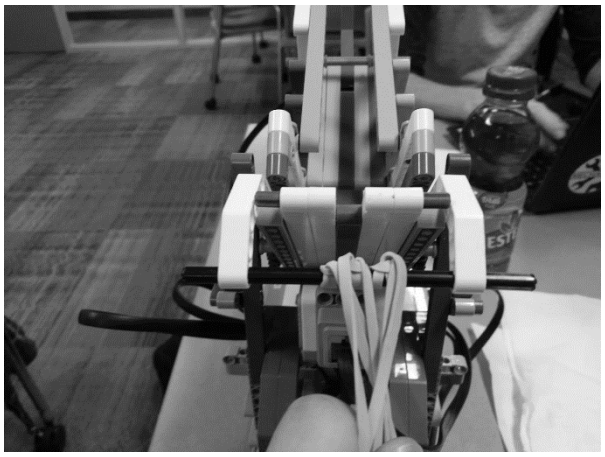


Fig 4.5 Front of Firing Assembly
As shown above, the rubber bands are tied to a rod angled downwards from the firing trajectory of the projectiles as they leave the barrel.

4.2.4 Magazine

The magazine is held up by multiple vertical beams connected to each other with various cross braces fastening the sides. The bottom of the beams was designed to be fastened through hinges, to which the magazine could be slanted to provide an angle at which the marshmallows were aligned for when they fall into the barrel. The magazine itself is mounted on top of the firing mechanism, and marshmallows are stacked on top of each other. As a bottom marshmallow is fired, another drops into place via gravity, and this process occurs continuously until either the marshmallows are depleted or the firing process halts.

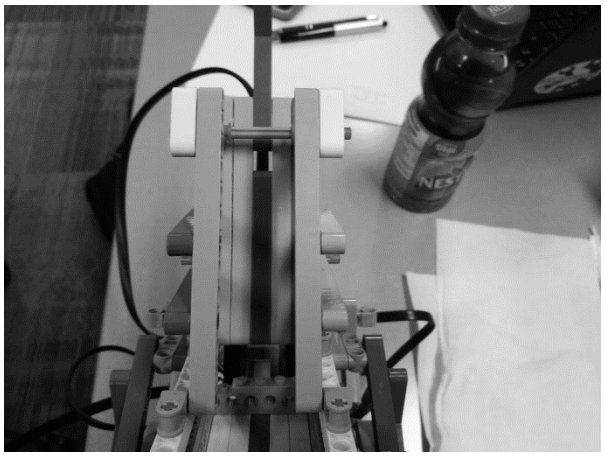


Fig 4.6 Magazine View
The magazine is shown from the front in the figure above. The multiple beams composing the magazine are shown along with the hinges at the bottom which connect the magazine to the firing barrel.

4.2.5 Sensor Layout

There are three sensors that send input to the controller: the ultrasonic, gyro, and colour sensor. The ultrasonic sensor is responsible for detecting intruders in a radius around the robot, thus it must have an unobstructed view of the field. This was achieved by mounting the sensor at the front of the chassis. From early testing, the ultrasonic laid at a slight depression consistently detecting the table and deeming it an intruder. This error was accounted for by considering a higher mounting point directly underneath of the firing barrel..

The functionality of the gyro does not depend on its location, only its vertical orientation. It was placed underneath of the barrel, in the main structure of the supporting chassis as this was considered a convenient location: unobtrusive and within proximity of the EV3 brick.

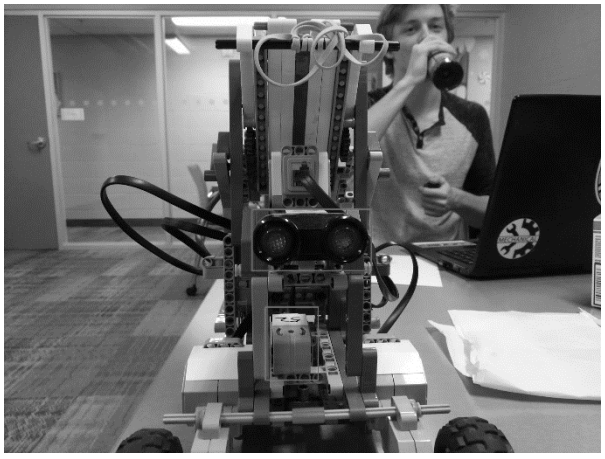


Fig 4.7 Ultrasonic and Gyro Position
This shows the position of ultrasonic sensor the gyrometer when mounted on the front of the robot.

The colour sensor is responsible for determining if the magazine is empty, and so the magazine was designed to allow for an extension. The location of this extension had to be unobtrusive of the projectile's path, within proximity of the EV3 brick, and in a position close enough to the cartridge so that it can detect. This extension was accomplished through a thin beam placed behind the magazine which gave reach to the projectile resting position.

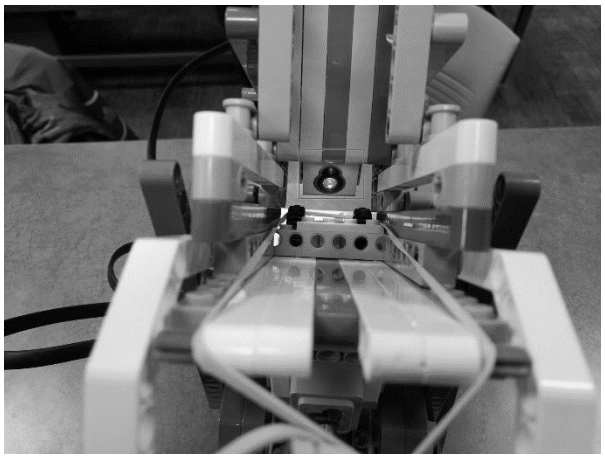


Fig 4.8 Color Sensor Position
The down-sight view of the barrel is shown along with the position of the color sensor.

4.3 Design Decisions

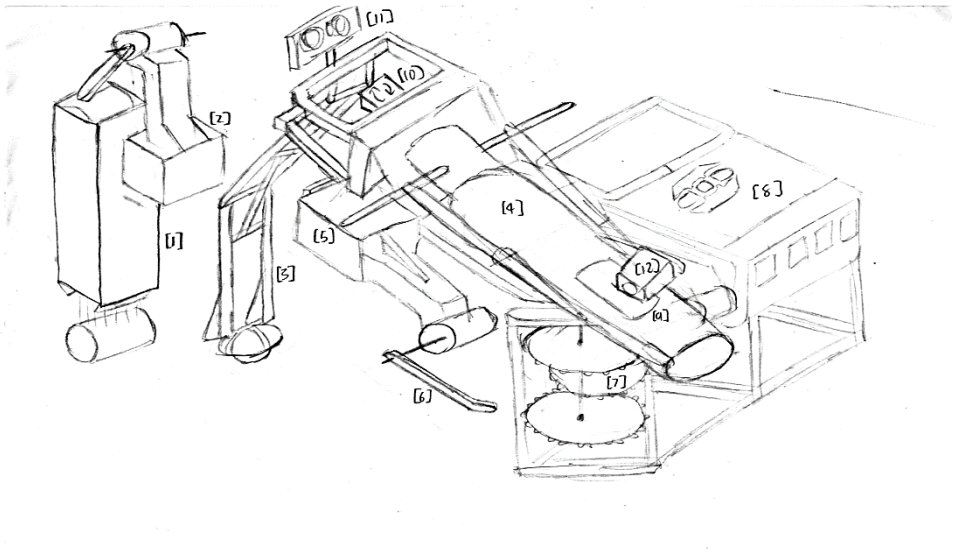


Fig 4.9 Design Alternative 1 depicts a sketch of the first design that was considered before the final design was built.

Our original design, as depicted in figure 5.9, consisted of a stationary base with a moving barrel and magazine. The LEGO EV3 Brick (8) was mounted using multiple Lego beams and a gearbox was made by interlocking three gears with each other (7). Resting on the axle was the cardboard barrel of the firing mechanism (4), with which would rotate gearbox powered by a side-mounted motor. A motor fastened to the underside of the firing assembly (5) related to an arm (6) which would pull back the sling of the barrel when rotated to fire a marshmallow. The magazine (1), would be mounted to the top of the barrel and consisted of a cardboard box with a motor mounted to the top (2). This pushed marshmallows beyond a flap that supported the stack of marshmallows. A supporting structure (3) made using multiple vertical beams connected to the ball

caster was designed to attach to support the backside of the barrel. This would follow the barrel as the system rotated.

One of the first design decisions was the design of the firing mechanism. A cardboard tube was initially chosen, as the cylindrical shape perfectly matched the diameter and size of a marshmallow. Rubber bands were used as the launching mechanism instead of other alternatives (ex. using a motor with an arm to fling marshmallows) to provide adequate power for our criteria and could fit within size constraints on the robot.

The reasoning behind the use of gears to rotate the barrel was based off an assumption of rigidity and support. By using gears, the rotation system could be supported by a wider base and the gears could be stress fitted together to provide further rigidity.

As for the magazine, a cardboard box was initially considered for adding paper flaps to the bottom. A cardboard based magazine could also be cut and designed to match the dimensions of a typical marshmallow. The flaps would allow the loaded marshmallow to launch without the friction of other marshmallows, while the motor loaded the next marshmallow.

The chassis was designed as multiple structures interconnected with each other so that the entire assembly could remain stationary, as intended with its use.

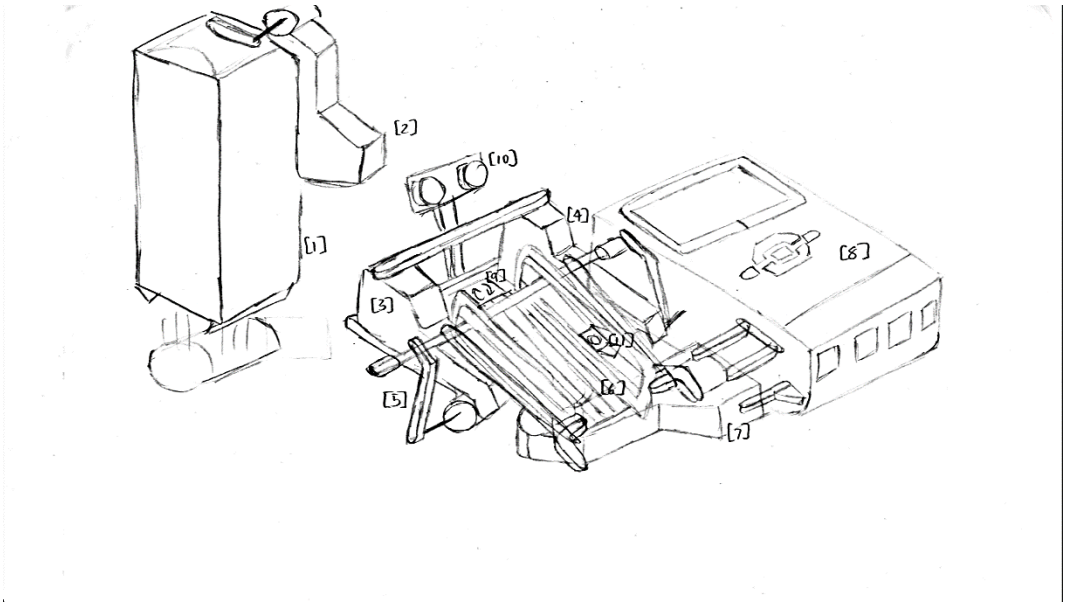


Fig 4.10 Design Alternative 2 depicts a sketch of the second design alternative that was considered after adjustments to the first alternative.

Some deviations from the first design included the exception of the gearbox, and redesign of the barrel and firing mechanism. Firstly, the motor that rotated the barrel is propped onto the ground facing up (7), and the firing mechanism (6) rotates directly by the axis of the motor instead of by the original gearbox design. Additionally, the barrel rebuilt using Lego pieces, and two motors, each with an arm attached to their axle, were strapped to

the side of the barrel. Those were used to pull the sling back to launch the marshmallows. The design of the magazine remained the same.

The gearbox was removed due to consistent gear misalignment. This led to an inability to support the weight of the rest of the structure, including both the firing assembly, magazine and supporting structure. As a result, the motor was placed facing up and the the upper assembly was lowered for increased stability.

The barrel's material was rebuilt using Lego pieces instead of cardboard due to the cardboard's high tendency to deform and tear from tension caused by rubber bands. With the rigidity of interlocking Lego pieces, the barrel was more suited to handle the tension from drawn back rubber bands.

A dual-motor drawback mechanism was implemented as opposed to the previous single motor. The second motor contributed much needed strength and balance to pull back the sling, increasing not only the drawback distance but also accuracy and consistency.

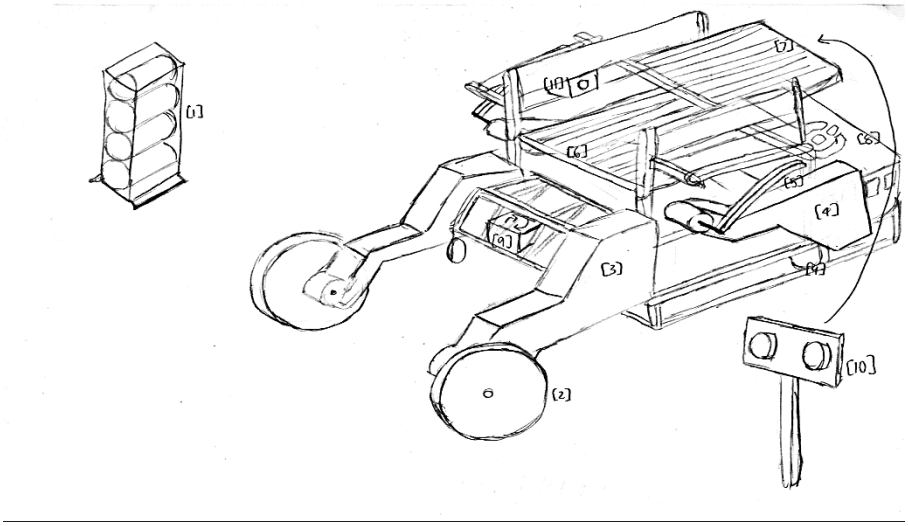


Fig 4.11 Design Alternative 3 The third design alternative that was designed after considerations to the previous two is shown in the sketch above.

Overall, the two main changes and decisions that had to be made involved the overall chassis design with the wheel motors as well as the design of the magazine.

The most significant change that was made in the third design alternative was the entire the chassis. Instead of a stationary base, the entire structural component would be supported by two wheels and a rigid chassis. This was done to maximize stability in a way the previous design couldn't. By supporting the entire structure of the robot on one stable moving base, more components could be connected fixed against each other using multiple Lego beams.

Since the chassis required two motors and the firing assembly required two motors, the motor from the magazine was relocated to be used on the chassis. Having the

marshmallows stacked on top of each other in the barrel was not a huge drawback as long as the surface of the sling was small enough so only one marshmallows was launched at a time.

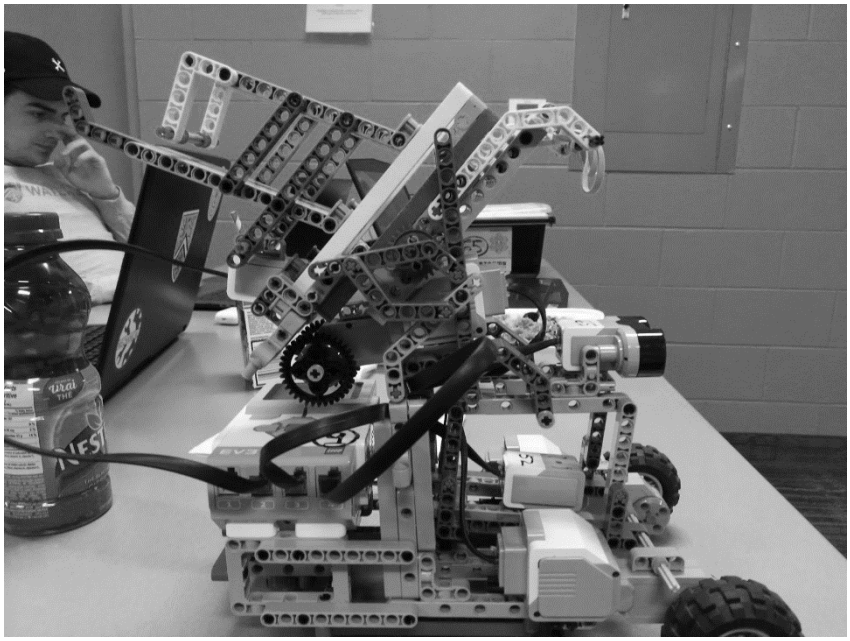


Fig 4.12 The picture above depicts the final design that was chosen for the project. The elements of the chosen design were discussed in section 5.2 of the report.

Although the chassis was kept the same, the main design decisions made from the third alternative to the final design were based around the firing assembly and the magazine. The firing mechanism was redesigned to gears and pre-drawn rubber bands because the double arm drawback system still carried inconsistencies. During testing, the sling would often be stuck or the arms would not draw the sling far enough to provide sufficient power. By using gears, releasing the rubber bands was more consistent.

The rubber bands were tied to a bar located at an angle 45° below the front of the barrel. This allowed the rubber bands to rest out of the line of fire after release.

The magazine was designed to be aligned at a slant so that the marshmallows sitting in the magazine would only partly contact the bottom marshmallow that was to be fired. This allowed for less friction from the upper marshmallows as the bottom marshmallow is launched from the barrel.

4.4 Trade-Offs

As a revision of the firing design, one motor had been allocated to the firing mechanism instead of the original choice of having two motors actuating the motion. The second motor from the original design has been moved to the base of the chassis allowing for a more accurate rotation of the robot. This allowed the firing mechanism to still be capable

of firing multiple projectiles consecutively but at a limited number before having to draw the rubber bands back.

5 Software Design

As with the mechanical design, meeting criteria while staying within constraints proved a difficult task with the hardware given. Many iterations of the program were tested both alone and in conjunction with one another. All the while, alterations both slight and significant were made. The following section describes the components and design of our software, including how it is implemented to perform the tasks necessary for the robot to protect its user.

5.1 Functions

Functions were designated through the careful dividing of planned tasks based on importance and reliance. The following outlines how functions were used to complete the tasks listed in section 6.2.

5.1.1 Startup

The startup function is used as an introduction to the program for the user, providing easy access to the system's main function. It prompts the user with the option to initially discharge ammunition from the magazine. It has function prototype:

void startUp(tSensors colorSens)

Additionally, it performs the initial check for marshmallows in the cartridge by receiving input from the colour sensor. Then it waits for confirmation from the user before it begins operation. As a user-based product, the user interface as well as ease of use dictates the success of the product, thus the startup had its own function. This function was written by Brian Windhaus.

5.1.2 Body Rotation

The rotate function assists the ultrasonic sensor with its intruder detection. It controls the rotational movement for the robot to observe a field of view rather than limited area directly in front of the robot. It has function prototype:

void rotateAngle(int motorPower, int arcAngle, tSensors gyroMeter)

The function allocates a given power to the two motors controlling the wheels under normal circumstances to create a arc of magnitude *arcAngle*. Using the gyrometer, the robot remains spatially aware of its direction, and will adjust its rotation direction appropriately to stay within the arc angle. When an object is detected, *rotateAngle* stops the chassis' movement and allows the ultrasonic sensor to take its readings. The rotation will continue once the system no longer detects intruders. Since the robot's surveillance

required more than just a motor power but multiple readings, it was written as a function. This function was written by Michael Nasello.

5.1.3 Intruder Detection

This function is responsible for recognizing and reacting to intruders through the main program. It has function prototype:

bool isApproaching(float waitDuration, int detectDist, int approachDist, tSensors ultraSonic, bool rotateDirection)

Upon initial detection, the ultrasonic sensor stores a reading of how far away that object is. After a waiting period of *waitDuration,* another distance is measured, and the two measurements are compared. If the second of two measurements is less than the first by the set *approachDist*, the function categorizes the object as a threat. A problem was encountered during testing, where, if a stationary object was detected the robot would remain fixed on the object. Controlled frames were set up to investigate this, whereby certain objects would be introduced or removed, and different scenarios simulated. It was found that as the ultrasonic sensor recognized the item was not an intruder, it would rotate for a very brief amount of time, then stop again as it detected the same object. To solve this problem, a wait period between detection times was added. The function *isApproaching* is important to intruder detection and is in use throughout the duration of the program. Thus, it was deemed necessary that it have its own function, written by Jing Hao Yao.

5.1.4 Firing

The firing function is the line of defense the robot provides for the user. It performs the actual intruder deterrence. In the source code, it has function prototype:

void releaseSling(float motorPower, float incrementAngle)

The method of firing involves elastic bands hooked on to a gear which *releaseSling* controls. The function allocates power to the motor so that each rubber band is released and launches forward. This function is responsible for accurately releasing one sling at a time using the motor encoders. The encoders regulated the rotation to forty-five degrees backwards. The need for this action came up many times during the program, and it intertwined with the isApproaching function, thus making it much simpler as its own function, written by James Lin.

5.1.5 Ammunition Check

The function is required to continuously check the magazine for ammunition and activate the appropriate actions if deemed necessary. It has function prototype:

bool isLoaded(float waitDuration, tSensors colorSens)

The colour sensor is mounted to the magazine and continuously checks the cartridge for ammunition. If the sensor does not detect the colour white, the ammunition is considered to be depleted. It will then display the appropriate message on the EV3 brick. This process is used continuously, and consequently was given its own function, written by Brian Windhaus.

5.1.6 Discharge

This function is offered to the user at the beginning of the program and automatically implemented at its end. It has function prototype:

void dischargeAmmo(int maxAmmo)

The function works firing any remaining projectiles in an ordered manner. It will gradually release any left-over elastic bands by rotating the gear they are hooked on to. The discharge of ammunition is used twice throughout the program; for this reason, the process was given a block of code dedicated to it. This function was written by Michael Nasello.

5.2 Task List

This section delves into how the overall functionality of the robot was divided in individual task. A description of each is included.

5.2.1 Final Task List

Startup	Program starts by user uploading the code or starting the code on the EV3 brick.
	The robot completes a check using the color sensor to see if there are any marshmallows in the magazine. It waits until marshmallows are loaded until starting the rest of the tasks.
	Once marshmallows are loaded, users are given an option to choose between two startup options. One option is to simply begin the robot's detection by pressing the enter button, and the second option is a discharge button, which releases all the marshmallows before continuing.
Main	Maintain an arc of 180°(±10°) during rotation.
	When an object enters the detection field (1.2 m) the robot is to stop.
	If the object moves out of the detection field or does not approach the robot, the system will continue to rotate.

	Fires marshmallows upon detecting that the object is approaching, using the rubber bands drawn back to the gear.
	Stops and waits for further marshmallows to be loaded once no more marshmallows are in the magazine.
Shutdown	Begins the shutdown procedure when the enter button is pressed during its detection.
	Begins shutdown procedures once 20 seconds have passed without more marshmallows being loaded into the magazine.
	The option to discharge the magazine (launch all remaining marshmallows) or simply to end the program by pressing the up and enter button respectively.
	The program shuts down after the user presses one of the two buttons mentioned in the previous task.

Fig 5.0 A task list that details the decisions and performance of the robot

5.2.2 Modifications

There was one change made between the original task list and final presented in this report: the option to discharge remaining ammunition. This was an added feature provided to the user upon program start and automatically implemented at program end.

5.3 Design

Due to the constant rebuild of the mechanical design, the program needed rewriting as well. This section details the final build of the program, along with flowcharts, data storage, decisions, and tradeoffs.

5.3.1 Flow Chart

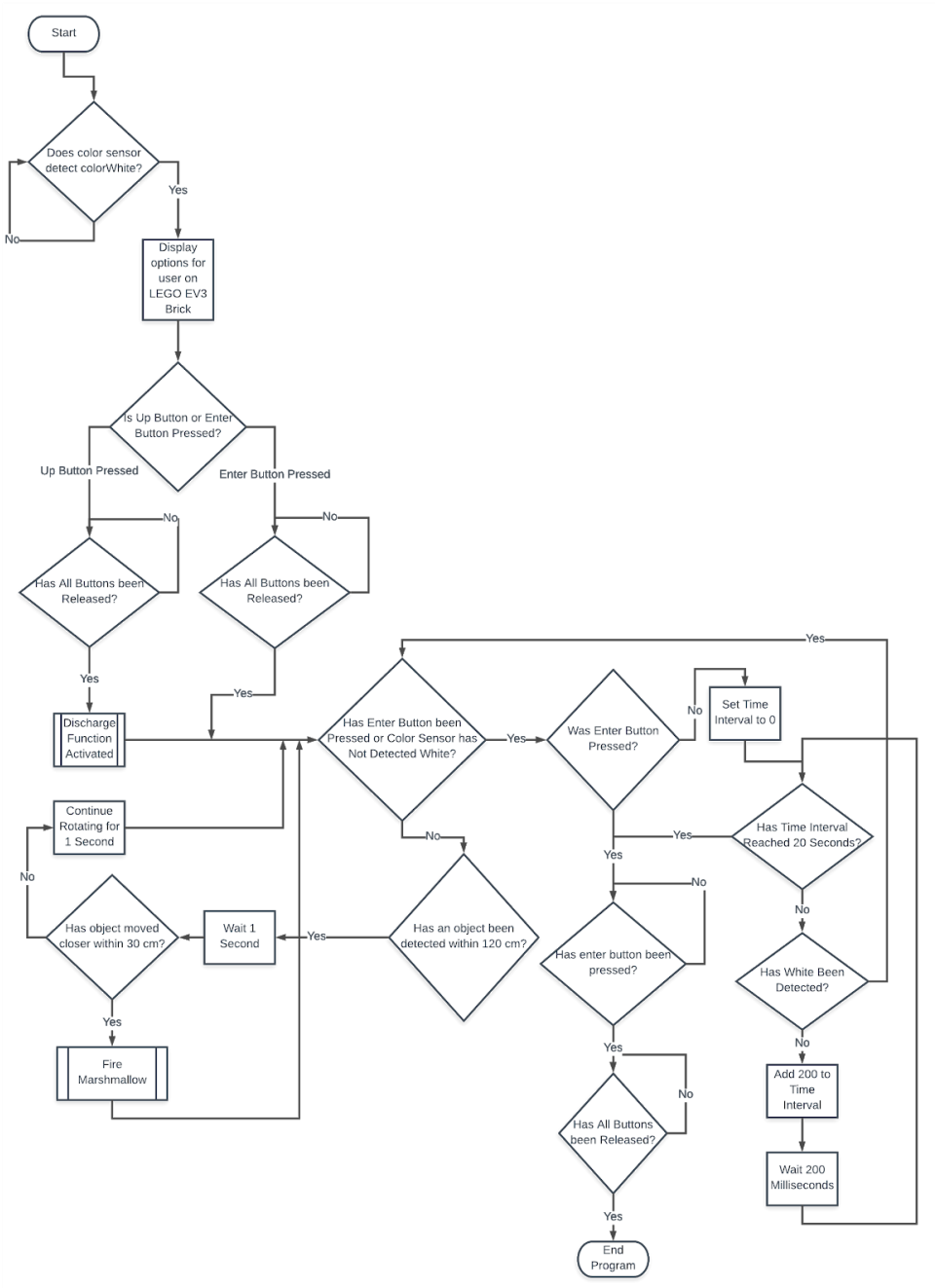


Fig 5.1 The main flowchart for the program is shown above.

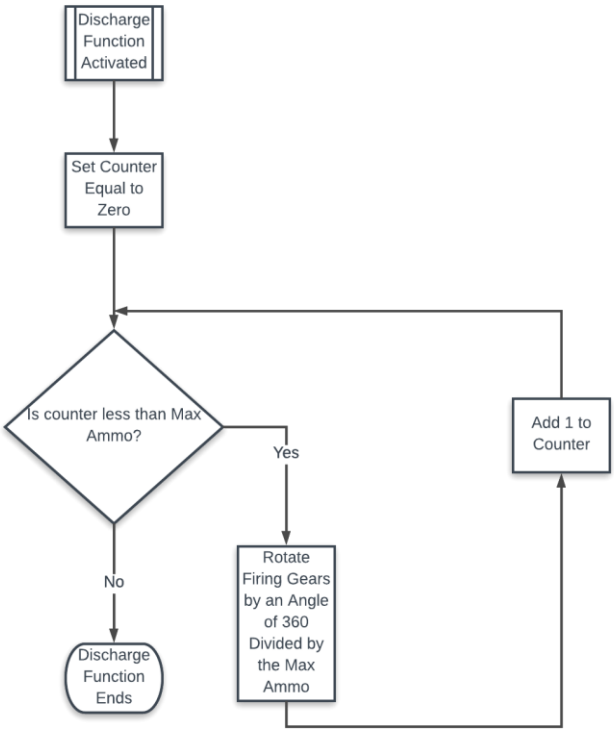


Fig 5.2 shows the flowchart for the discharge function in the code.

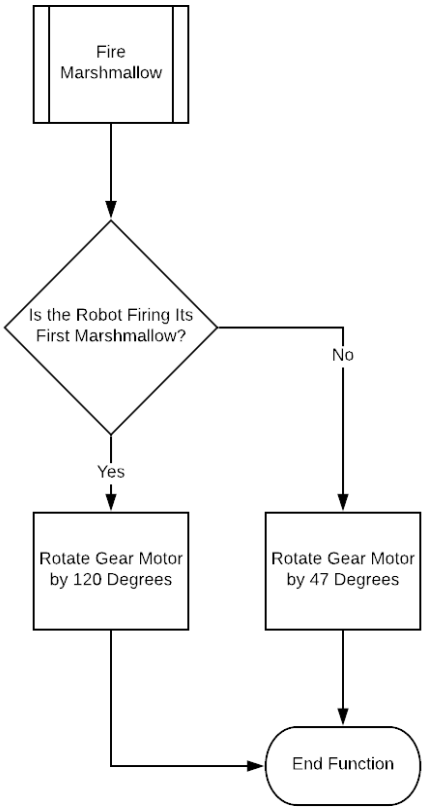


Fig 5.3 shows the flowchart for the fire marshmallow function

5.3.2 Data Storage

The first data piece stored in our code is the distance from the robot to the object, which was stored as an integer type. In this case, two integers were declared and updated whenever a new object is detected using the ultrasonic sensor. This is used to judge whether or not an object is approaching.

The second data piece stored is the color that the color sensor detected. This is stored as an integer and is continuously updated as the color sensor continuously checks the color within the barrel to judge if the magazine still had ammunition.

Other types of data that needed to be stored within our code were parameters that were used to modify tasks that the robot completed. The angle at which the drawback gears rotate to release the rubber bands is stored as an integer. Other integers include designated motor power which determined how fast the robot rotated, and the detection distance which the robot deemed the obstacle to be in the range of the user's personal space and thereby start the tasks following detection. Another data type that was stored is a boolean that indicated whether the robot was launching its first marshmallow, and this

was used to indicate that the drawback gears need to rotate a larger angle on the first release to accomodate for where the rubber bands are positioned when the robot is initially initiated. The final data that needed to be stored is a boolean that indicated whether or not the robot was rotating clockwise or counterclockwise, which guided the direction that the robot would rotate when reaching both ends of its designated arc.

5.3.3 Design Decisions

The first software design decision that was made was the creation of the "IsApproaching" function, which had a return type of a bool. Initially, the plan was to have the check for whether the object was approaching as a part of the main code, where the rest of the tasks would branch off the result. However, it was decided that making the "check" a function would streamline the code and the function can instead be used as a condition in a while loop that contained the function used to rotate the robot.

Another decision that was made was the inclusion of an action where after the robot detects a stationary object, it would proceed to rotate for a small duration so that it could pass the detected obstacle. This decision was made upon observation that the robot would stop and remain stuck facing a stationary object as it continuously performed checks, and so the addition of the action allowed for this correction.

Some software decisions involved the parameters used to call the functions. One particular instance is the decision to make the wait time between taking the first and second distance measurement by the ultrasonic sensor one second. This was to accommodate for a person's walking speed. As a result, one second was chosen as a suitable wait time so that the distance readings could reflect the distance covered by an intruder. The next decision was to have the detecting distance (i.e. the distance between the first and second distance readings at which the object is considered to be approaching) be 30 cm. Ideally, the ultrasonic sensor would give the same readings when detecting an object that is stationary, and the detecting distance could be set as 1 cm. Realistically however, the ultrasonic has imprecision in its readings shown later in testing and therefore a larger detecting distance was given to account for the sensor's error, while reflecting the distance a human would reasonably cover in the 1.2 radius.

The next set of design decisions involve the angle that the draw back gears would rotate upon. The first inclusion is a bool that indicates whether the robot is firing the first marshmallow in its operation or not. Its purpose is described in section 6.3.2 and was included to improve the consistency that the first marshmallow is launched. This is because the angle between each peg that the rubber bands were drawn onto is much smaller than the angle that the gear has to travel on the first rotation to release the first rubber band.

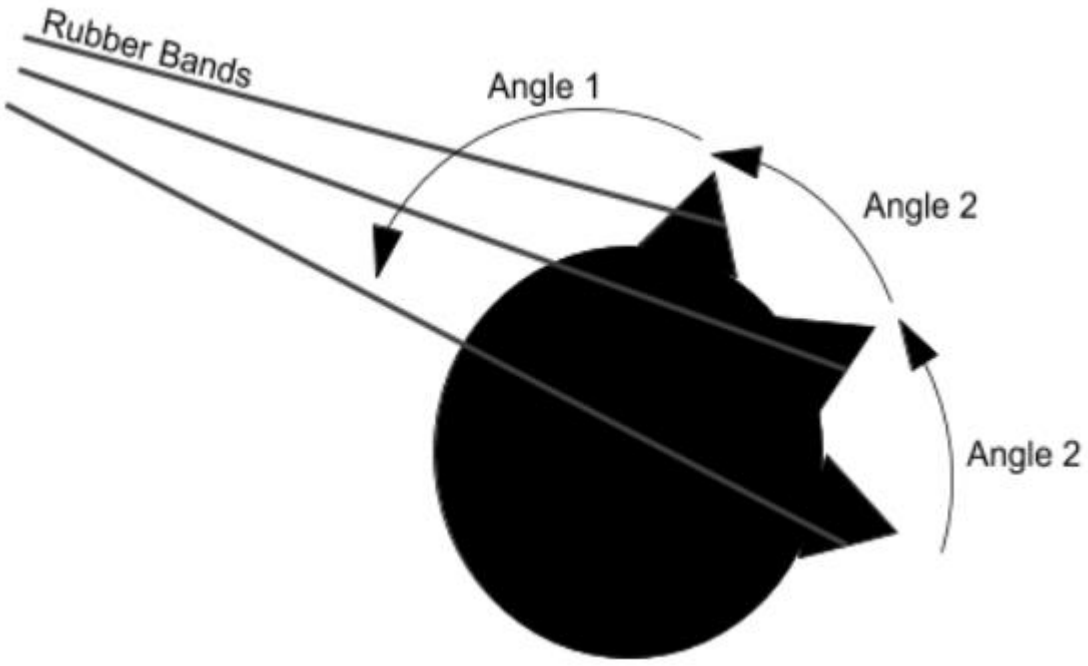


Fig 5.4 This visual interpretation of firing angles shows a visual description of the angles that the pegs that the rubber bands are drawn onto need to rotate in order to release their rubber bands. As shown, the first rotation of Angle 1 is larger than the rotation needed for the second two pegs to release their rubber bands (Angle 2).

Another decision that had to be made were the angles that the pegs themselves had to rotate. After testing with the firing gears through multiple trials, it was decided that the best angle for the gears to rotate is 47◦ as it provided the best firing conditions where the rubber bands wouldn't release consecutively immediately or jam on the pegs.

5.3.4 Trade Offs

As mentioned in section 6.5, there was a problem with our *isApproaching* function where the ultrasonic sensor would continually detect the same object and not advance from it. To solve this, a three second period of rotation was coded between subsequent checks so that the robot could pass stationary objects and continue its area surveillance. This created a problem at the rotation endpoints where, if an object was detected near its rotation endpoint, the chassis would continue its rotation past the 180-degree marker until the three second period was over. At this time, it would change its direction of rotation and continue normal activity. The tradeoff for this error had to be made in order for the robot to be able to patrol the entire designated area and not experience deadlock upon detection of a stationary object.

5.4 Testing

Extensive testing was done to assist the programming of the software as well as understand the limitations of the robot. Tests mostly included distance, rotation, and

consistency of detection. Due to equipment constraints, straight distance measurements were done with a reading error of ± .5 mm and rotational measurements ± 1°. A field for measurements was designated and distances were marked with a long piece of tape roughly 3 m. A foot ruler was used to measure every 10 cm interval, and another piece of tape was crossed the start to indicate a perpendicular placement point.

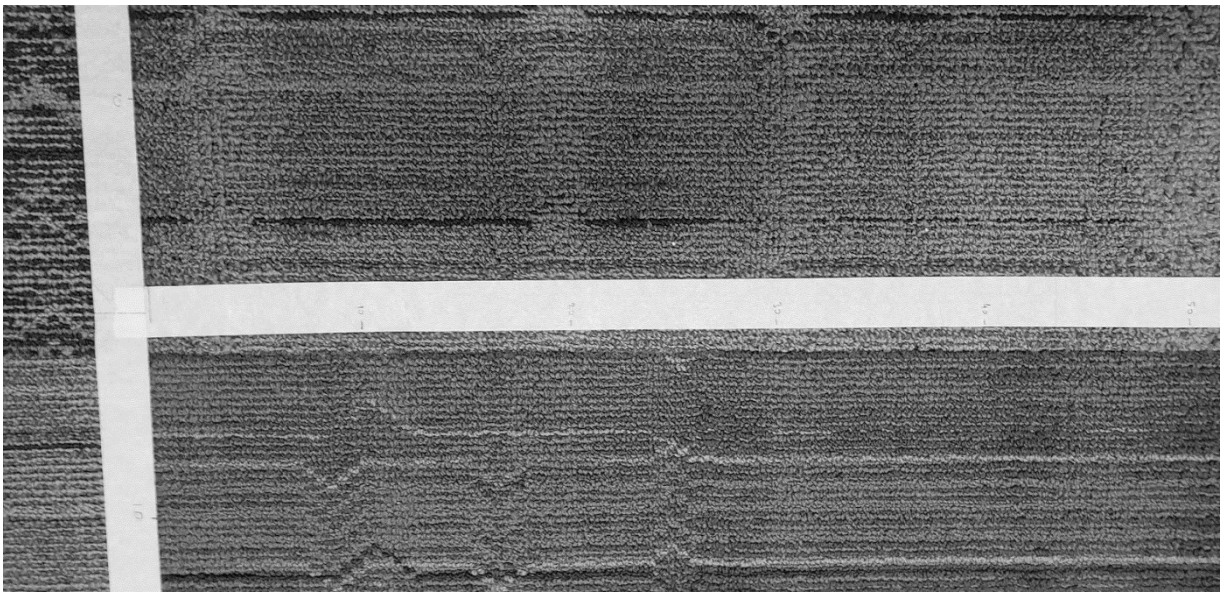


Fig 5.5 The testing field was set up using 2 perpendicular pieces of tape. The robot's initial position for each test would be parallel to the horizontal piece, which is marked in 10 cm intervals, and thus perpendicular to the vertical tape.

Just to note, the vertical tape will be further referenced as the perpendicular tape, and the horizontal tape the parallel tape, due to the designed standard position of the robot during all tests, as well as the different viewing angles present in future figures.

5.4.1 Ultrasonic

The ultrasonic was programmed to stop if anything was detected within 120 cm. To test this, distance and consistency was considered. To test the distance, a binder was made to move closer by roughly 1 mm every second. This was made to be as continuous as possible. When the ultrasonic detected the binder, it was dropped, and a ruler was measured from the 10 cm interval before up until the binder.

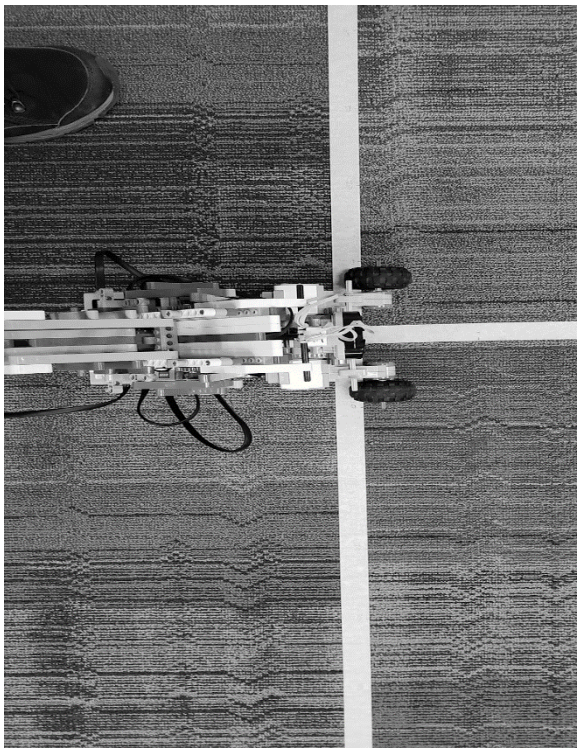


Fig 5.6 The ultrasonic was lined up at the center of the cross with its lenses against the end width of the perpendicular tape. Each interval of 10 cm was marked along the parallel tape.

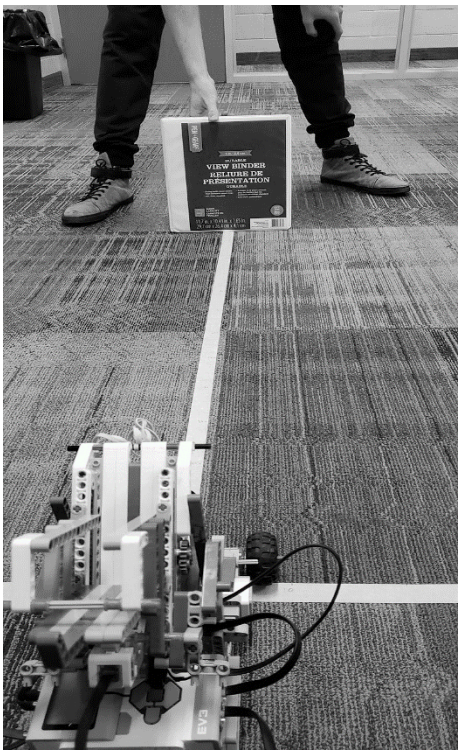


Fig 5.7 A binder was made to inch closer until the robot detected it. Detection was indicated by a slight movement of the firing motor.

The average distance detected by the ultrasonic was (124±3) cm, roughly 4 cm higher than the programmed distance. As this wasn't a big difference, and the range indicates a inconsistent measurement, the software wasn't adjusted for this error. The scope of the project also allows for leeway, as 120 cm was an arbitrary distance used to define deterrence from personal space. It was agreed that 124 cm equally matched that description.

Another test was done to check the consistency of this detection while running the robot's program. A binder was placed at (120±1) cm mark, and the robot was made to turn 180° back and forth. From the data, the ultrasonic detected the binder about 97% of the time, missing once in 30 trials.

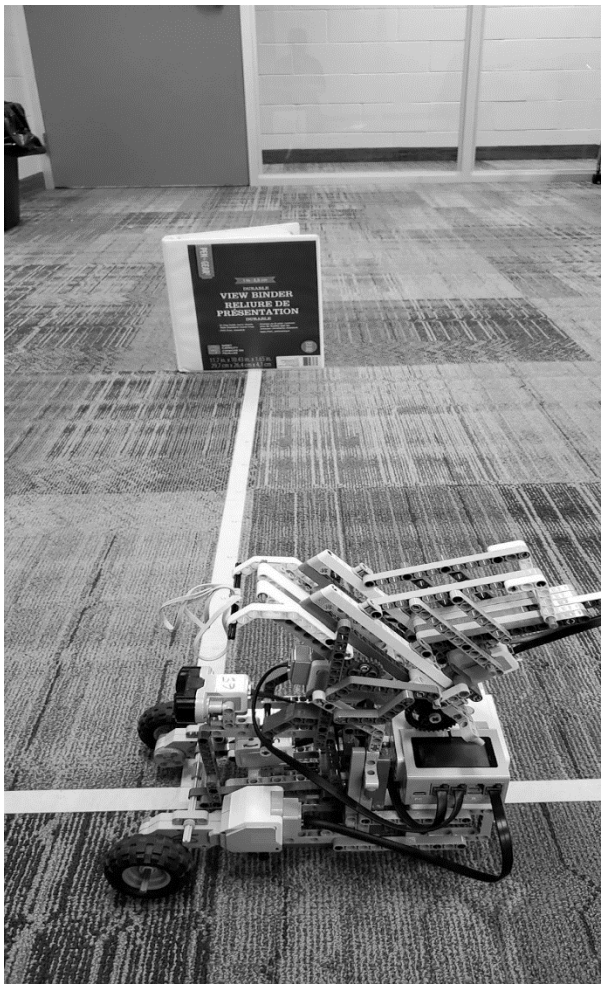


Fig 5.8 This shows the initial position of the test. The robot is lined up similarly to the rotation test below and the binder is left at 120 cm.

Fig 5.9 This shows a successful detection of the binder. The angle is due to the ultrasonic detecting in a sort of cone manner for farther away objects.

5.4.2 Motor Rotation

Another important aspect of precision was the rotation of the robot. Testing this was done by using the perpendicular tape line as a baseline for a 180°. Since the axis of rotation of the robot is between the 2 front wheels, the wheels were centered as best as possible along the width of the perpendicular tape. The front of the robot was lined up with the parallel tape as best as possible, to mark the starting position, and the program was run to rotate it's 90° clockwise, then 180° counterclockwise. The counterclockwise end position was then measured. This was done by lifting the rear ball caster slightly and marking that position, then the position was measured relative to the tape.

An important note is that the test was done on a carpeted surface. While this caused some instabilities in the robot's movement, it didn't affect the angle of rotation, as that's done by the gyro sensor. Thus, the robot is easily shifted back in line with the tape without

altering its ending angle, allowing for a measurement irrespective of the surface of movement.

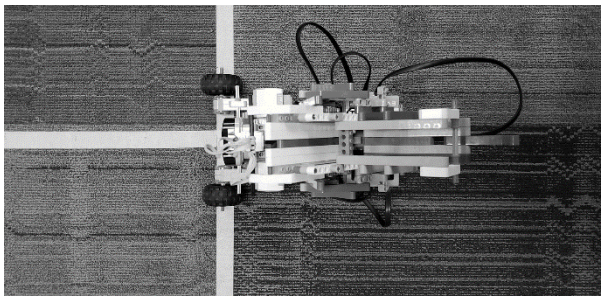


Fig 5.10 Position 1, the robot is lined up to the parallel tape as the starting position. The wheels are best fitted down the width of the perpendicular tape to center the robot's rotation axis.

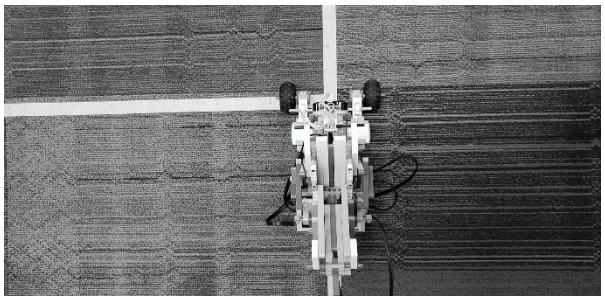


Fig 5.11 Position 2, the robot has done its first 90° rotation clockwise and is now lined up with the perpendicular tape. If offset, the robot is shifted back into line with the tape with minimal alteration of its angle.

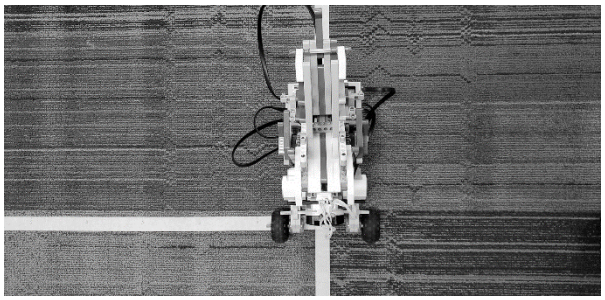
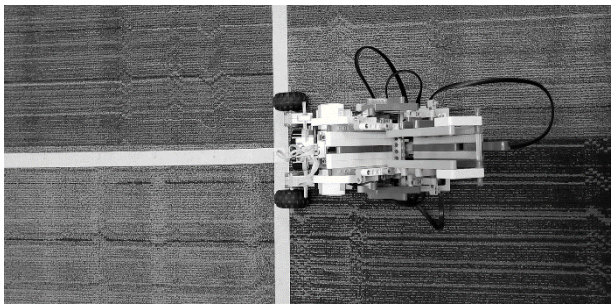


Fig 5.12 Position 3, the robot has turned 180° counterclockwise, finishing one loop of the program. Any offset from the tape is adjusted and now the angle can be measured.

The robot rotated (180±1) ° on average with an overall range of (4±2) ° over the 30 trials. The gyrometer of the robot is fairly consistent, able to stay on its rotation axis and not overshoot or undershoot (although it is discussed elsewhere in the report that a gyrometer earlier in the project build phase had separate issues). One last test was done by running the program continuously to see if any error would propagate, but it can be said that the robot on average stayed within (182±2) ° after 5 full loops.

5.4.3 Ballistic Firing Mechanism

To test the distance the robot would be able to fire, the same tape configuration was used but with farthest front edge of the robot lined up with the start of the parallel tape measure.



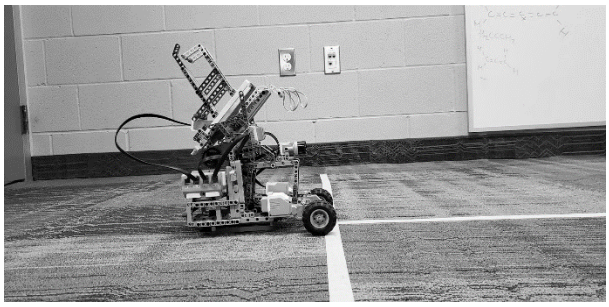


Fig 5.13, 5.14 Two pieces of tape were crossed perpendicularly with the wheels lined up to the end width of the horizontal piece. This means the actual firing distance is slightly farther, but the effective distance from the robot was prioritized.

The robot was then made to fire and the marshmallow's final position would be measured. This means that, inevitably, the measured distances are further than the actual firing range due to bounce, but the carpeted surface as well as the marshmallow's physical softness limited the distance gained. An estimate pits the distance gained between 1 and 10 cm, depending on the original distance. To measure the position of the marshmallow, a ruler would be placed collinear to the closest marking before the marshmallow, and another ruler perpendicular to the first. Then the closest point of the marshmallow would be recorded.

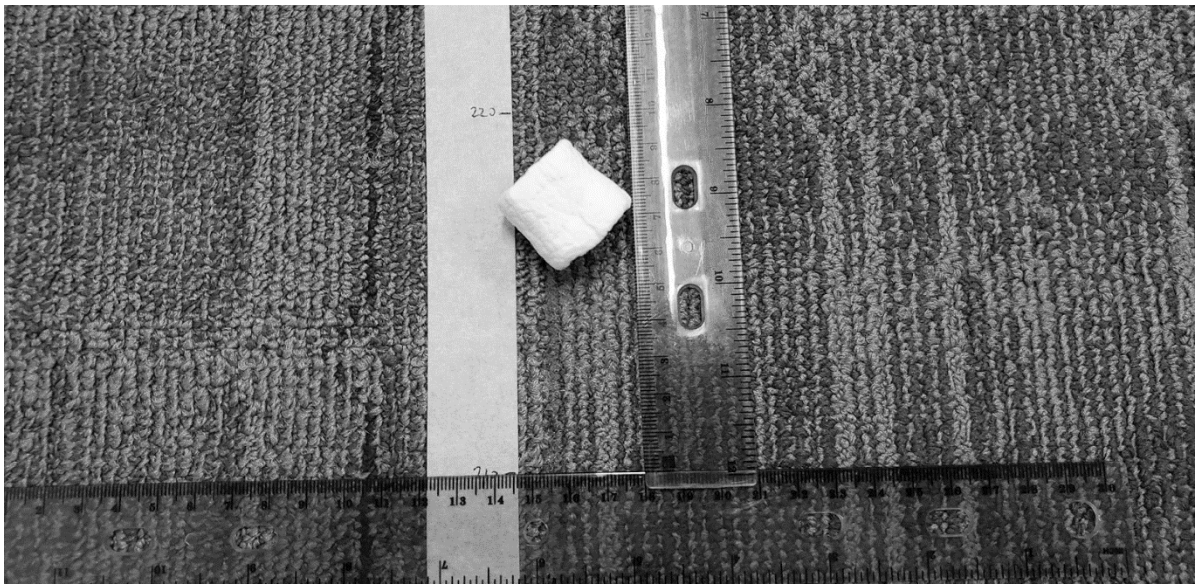


Fig 5.15 The closest point of the marshmallow is measured using 2 rulers, one lying perpendicular, and one lying parallel to the tape. The offset from the tape is mostly negligible.

It's noted that this method ignores the offset of the marshmallow due to its irregular shape and some inconsistencies from the ballistic mechanism. It was noticed however,

that the offset due to this was insignificant, and thus each measurement is done as if the marshmallow traveled straight down the firing field.

Another test for consistency of fire was done by recording successful fires. A successful fire was defined as the marshmallow traveling at least 10 cm, but a large amount of failures was simply the marshmallow failing to exit the firing chamber. From 30 trials, the marshmallow was successfully launched 60% or 18 times.

5.4.4 Data Excerpt

Below shows a summary of the aggregate data of the different measured tests. Additional tests included success rate of detection and success rate of firing.

	Ultrasonic Detection	Gyro Rotation	Rotation after 5 loops	Projectile Fire Distance
Average Measured	(124±3) cm	(180±1) °	(182±2) °	(80±70) cm
Median Measured	(123.55±.5) cm	(180±1) °	(182±1) °	(259.3±.1) cm
Range of Trials	(10±1) cm	(4±2) °	(5±2) °	(63.65±.05) cm

Fig 5.16 A summary of each test is shown. Each test was done with 30 trials. Median and range values use reading error, while the error in the average measured values is calculated using the standard population deviation formula.

It should be noted that measured projectile fire distances are heavily rounded due to the significant error. Also, the large range in rotation is a product of a few datums that weren't quite outliers but was still infrequent, and the inability to measure at a higher degree of precision.

5.4.5 Testing Error

Aside from hardware and equipment error, other errors lie in the design of the test as well. Due to the approach of the detected surface being done by human hand, it may be irregular causing the detected distance to be less when the ultrasonic actually detected the surface. Additionally, the robot signals a successful detection by running its firing gears, thus stopping movement of the binder is dependent on human reaction time. All this amounts to a higher expected distance of detection than what was measured.

An error affecting the rotation tests was the carpeted surface. While the correction was deemed satisfactory, it still inevitably leads to error, thus this was accounted for with the high degree of reading error allocated, being 1 full degree. Also, the test was set up with the robot centered on the width of the perpendicular, which forced measurements from the center of the same width. Finding the center required additional measurements, and thus produced more room for error.

Lastly, it was previously mentioned that the horizontal offset of the marshmallow during the fire test was negligible, but the existence of such still renders the measurements erroneous. Additionally, a normal foot ruler was used to mark each 10 cm interval and marks further than 30 cm were based of the most recent multiple of 30 cm. This causes further measurements to have a propagated equipment error.

5.5 Software Challenges

When testing the robot in a controlled environment, many tasks were not completed adequately due to flaws in our source code. Problems were experienced during chassis rotation and intruder detection.

After frequent testing of the first gyrometer, it was apparent that the sensor had a constant bias of approximately 2 meters per second. This bias altered the arc path of the robot, no longer observing a 180-degree arc. Initially, the solution to this error was accounted for by forcing a literal adjustment factor in the code, but it eventually became obsolete, when another gyro with no bias was obtained.

Another source of error was consequent of our intruder detection function. The function is instructed to stop when it detects an object and determine if it is approaching. At this time, it was supposed to continue its surveillance of the given area, however after the ultrasonic sensor took its second reading, it would immediately detect the same object and repeat the process. This resulted in deadlock, where the robot was fixed on the same area. The solution to this problem involved setting a three second rotation period after consecutive checks returned no change in distance. This solved the deadlock problem; however, this caused the robot to overturn if it detected an object three seconds before its set stopping point.

6 Verification

The following section provides a description of how constraints established for demo day deliverables were met, as well as discusses changes to constraints throughout the entirety of the project.

6.1 Meeting Constraints

Two constraints were set on the robot during demonstrations of robot functionality on demo day: consistent motion and accurate detection. These constraints are discussed below.

6.1.1 Consistency of Motion

As listed on the updated list of constraints for demo day:

“The robot should pivot about its central axis without moving from the initial starting point. Given a tolerance of motion, the robot should not move away from its center point more than ± 2 cm in a radius around the center axis.”[2]

This was achieved through the *rotateAngle* function. The function was designed so that the chassis could only rotate about its center axis and could not deviate from it with any horizontal motion. Throughout the program, the two motors work in tandem. At no time does one motor receive power without its other receiving a power of equivalent magnitude but in the opposite direction. This ensures that only rotational movement and no deviation from the robot's initial position.

6.1.2 Accuracy of Detection

As listed on the updated list of constraints for demo day:

"The robot is not to fire upon any person or object that is either stationary or moving away from the system. Measured quantitatively, the number of stationary people or objects and people moving away from the robot will be -1 and every time they are not fired upon is +1. Having no non-targets being hit is considered a success (positive score)." [2].

This was achieved by implementation of the *isApproaching* function, which differentiates between approaching and stationary objects by taking two distance readings, separated by a brief waiting period. The difference between the first and second reading is computed and, if this value is greater than a predetermined *detectDist*, the object is deemed approaching and dealt with appropriately. To ensure a series of accurate detections, the *detectDist* variable was used. Due to imprecision of the ultrasonic sensor, a variance of readings for a stationary object may occur. As mentioned in section 6.3.3, a threshold of 30 cm was established so no errors were experienced during runtime due to the sensor.

6.2 Changes to Constraints

There were few changes to project constraints made between the interim and final report. A third constraint, labelled 'Resource Limitations' was left out of the final report. The constraint outlined:

"In terms of resource restrictions, a budget was constructed for the expenses that would come up during the process of building, testing, and displaying the robot. The budget is set to be thirty dollars which will be distributed for constructing the parts that must be custom made, buying the ammunition, and any other costs that may occur during the procession of this project. The budget is flexible but not to exceed an upper limit of forty-five dollars." [2].

This constraint was left out of the report by unanimous decision as expenses during the project were very little and not a focus. All constraints were met on demo day.

7 Project Planning

The planning discussed below shows a linear workflow, but, the planning direction was very adaptive. General ideas were laid out and attempted with alterations made to maximize the efficiency and consistency of each component.

7.1 Task Distribution

The tasks were given to group members based on proficiency with the material required to complete the software and mechanical design aspects of the project.

Jing Hao Yao	Wrote the “isApproaching” function for the detection of objects coming towards the system.
	Wrote the main code to call the functions and run the system protocols.
James Lin	Wrote the “releaseSling” function for the firing mechanism actuation.
	Developed and built the firing mechanism design for use in final demonstration.
Michael Nasello	Wrote the “rotateAngle” function for the rotation of the robot chassis. rotateAngle.
	Wrote the “discharge” function to deploy all extra ammunition in the firing mechanism when called upon.
Brian Windhaus	Wrote the “startUp” function which began the call for the other functions and begins the program
	Wrote the “isLoaded” to check if ammunition was loaded and end the program if none was detected.
Group	Developed and built the robot chassis for use in the final demonstration.
	Wrote and developed all deliverables for project benchmarks.

Fig 7.0 A list of all individual tasks is shown, as well as tasks designated to the entire group.

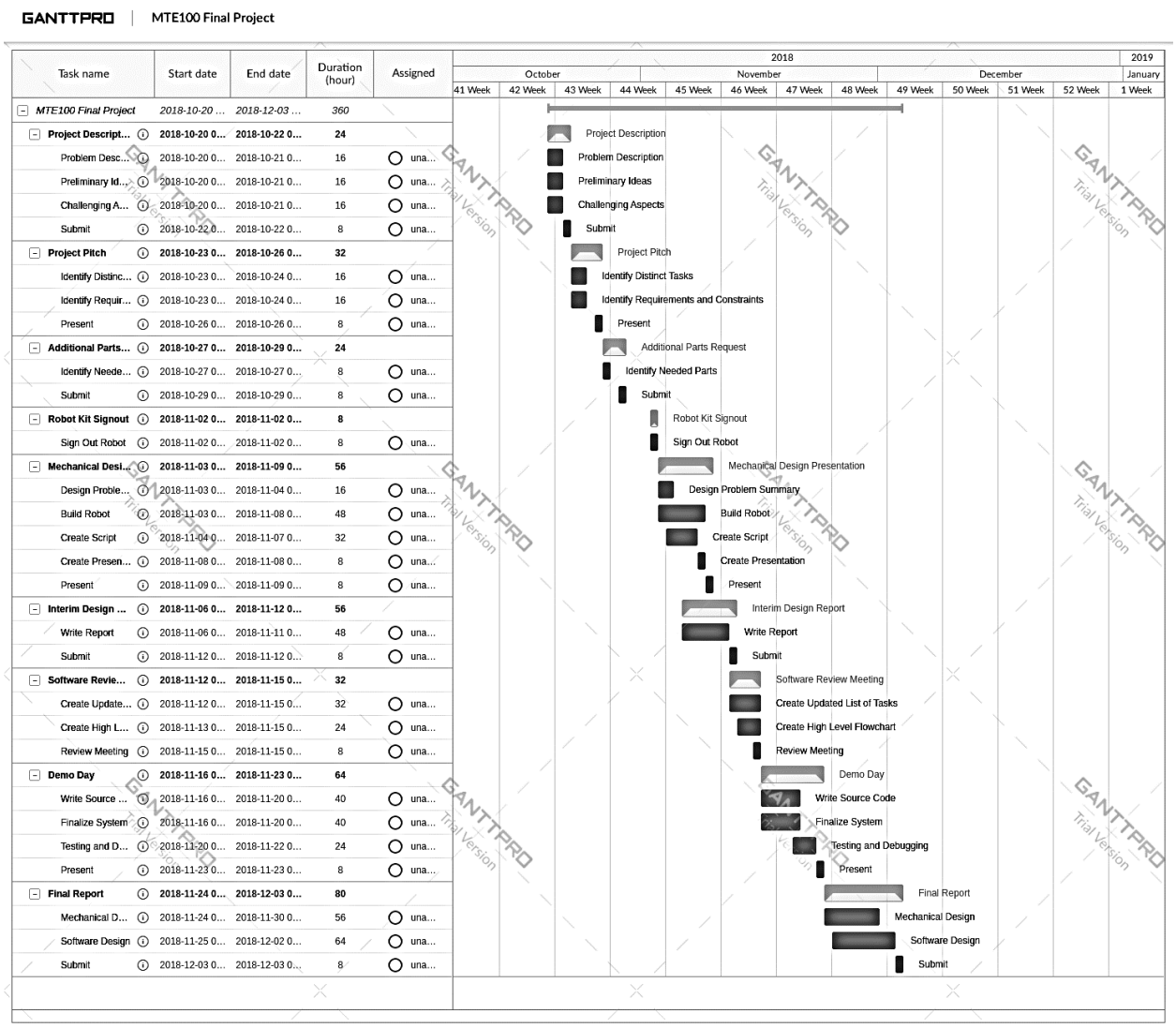


Fig 7.1 The project planning GANTT chart displays the timeline and distribution of the tasks and deadlines to meet for the project milestones

7.2 Project Plan Revisions

The overall project plan was followed very closely to the layout that had been set in Figure 8.1. The only modification to the project plan was during the mechanical design and implementation period as the firing mechanism had to be redesigned twice. Adding the extra hour to redesign and build a working firing mechanism ultimately did not change our scheduling which allowed for the project to be finished without taking time away from other aspects of the project plan.

7.3 Project Plan Comments

Aside from the redesign process for the firing mechanism, The GANTT chart was followed throughout every stage of the project without having any other deviation from

the outline. All milestones and deliverables were finished ahead or on schedule and no other modifications or delays were encountered during the project timeline.

8 Conclusion

A device was needed to provide protection of personal space while performing tasks that require one's attention to be in front of them. The objective was achieved by the design and implementation of a sentry robot composed of a *LEGO MINDSTORMS EV3*® kit, ammunition, and a source code to provide autonomous detection and firing during the systems operation.

The mechanical system comprised of the *LEGO MINDSTORMS EV3*® brick and subsequent sensors, along with multiple elastic bands. It provides object detection via the ultrasonic sensor, ammunition detection via the colour sensor, and rotation along a horizontal plane via the gyrometer, allowing for the proper implementation of the sentry robot code. The firing mechanism makes use of the motor encoders which allow for the elastic bands to be released sequentially and at a specified interval firing the ammunition. Using two more motors attached to the base of the chassis as well as the readings from the gyro sensor, motor power is applied in a way such that the system rotates 180° about the center axis of the robot.

The sentry software consists of startup, rotation, and detection functions allowing for the mechanical system to operate and achieve the objectives set by the need statement. During the startup procedure, the center button is pressed to start the robots main program and call all the functions to proceed in detection. Likewise, the up button would discharge all the ammunition on startup and proceed to start the program to detect and then end. The robot's distance detection function checks for an approaching object during its rotation iteration of 180° and stops if an object is detected to be within 1.2 m. It will continue to proceed on its survey if the object is not moving towards the sentry unit or if they object moves away from it, however, if the object moves toward the unit will then proceed to fire upon the object. After the robot has discharged all ammunition the colour detection function checks for white and displays a message to the brick; the robot will continue its survey if white is detected within 20 seconds of the discharge and the robot will end its program if white is not detected within the given interval.

The culmination of both the mechanical system and main program achieved the criteria and constraints set for the sentry robot to be successful. The robot moves with a consistency of motion within the set tolerance of a 2 cm drift from the central axis of rotation. The robot detects an object at the range of 1.2 m and proceeds to fire if it comes within the detection field. The robot completed all required tasks for the demonstration and met all constraints for the overall design.

9 Recommendation

The following section discusses changes that could be made to either the mechanical or software design in order to improve the product.

9.1 Mechanical

A source of error during operation involves the design of the magazine and how it interacts with the firing barrel. Occasionally, the marshmallow resting on the round to be fired would obstruct its launch. This obstruction could be eliminated with a cartridge that automatically dispenses ammunition and does not depend on the round in the firing barrel to support additional marshmallows in the magazine.

9.2 Software

With more time, the *isApproaching* function would be refined and implement a more advanced algorithm for intruder detection. Instead of taking two readings with a brief period in between, the function would collect a consecutive distance reading within proximity in small time intervals, where the values would be stored in an array. The difference between two consecutive elements in the array would be divided by the time interval at which the readings were taken at, and those values would be averaged out and compared to a range that corresponds to the average walking/running speed.

10 Back Matter

10.1 References

[1] M. Nasello, J. Hao Yao, J. Lin, and B. Windhaus, “MTE100 Project Interim Design Report,” rep.
[2] M.Nasello, J.Hao Yao, J.Lin, and B.Windhaus, “MTE100 Demo Day Deliverables,” rep.

10.2 Appendix

The following is source code used to operate the robot:

```
bool isApproaching(float waitDuration, int detectDist, int
approachDist, tSensors ultraSonic, bool rotateDirection)
{
	int firstDetection = 0, secondDetection = 0;
	if(SensorValue(ultraSonic) > detectDist){
		return false;
	}
	else{
		motor[motorA] = motor[motorD] = 0;
		firstDetection = SensorValue(ultraSonic);
		wait1Msec(waitDuration);
		secondDetection = SensorValue(ultraSonic);
		if((firstDetection - secondDetection) > approachDist){
			return true;
		}
		else
		{
			if(rotateDirection == false)
			{
				motor[motorA] = -15;
				motor[motorD] = 15;
			}
			else
			{
				motor[motorA] = 15;
				motor[motorD] = -15;
			}
			clearTimer(T1);
			while(time1[T1] < 3000)
			{
			}
			return false;
		}
```

```
	}
}

void releaseSling(float motorPower, float incrementAngle)
{
	nMotorEncoder(motorC) = 0;
	while(nMotorEncoder(motorC) > incrementAngle){
		motor[motorC] = motorPower;
	}
	motor[motorC] = 0;
}

bool isLoaded(float waitDuration, tSensors colorSens)
{
	if(SensorValue(colorSens) == (int)colorWhite)
	{
		return true;
	}
	else
	{
		motor[motorA] = motor[motorD] = 0;
		eraseDisplay();
		displayString(4, "Checking for loaded
Marshmallows...");
		for(int timeInterval = 0; timeInterval <= waitDuration
			- 200; timeInterval += 200)
		{
			if(SensorValue(colorSens) == (int)colorWhite)
			{
				eraseDisplay();
				displayString(4, "Marshmallows loaded.");
				wait1Msec(3000);
				return true;
			}
			wait1Msec(200);
		}
		return false;
	}
}

void dischargeAmmo(int maxAmmo)
{
	for(int shotCount = 0; shotCount < maxAmmo; shotCount++)
	{
		releaseSling(-25,-360/maxAmmo);
		wait1Msec(1500);
```

```
	}
	eraseDisplay();
	displayString(4,"Discharged.");
}

void startUp(tSensors colorSens)
{
	if(SensorValue(colorSens) != (int)colorWhite)
	{
		displayString(3,"No Marshmallows!");
		while(SensorValue(colorSens) != (int)colorWhite)
		{
			//the program waits
		}
	}
	eraseDisplay();
	displayString(3,"Marshmallows loaded.");
	displayString(5,"Press up button to discharge");
	displayString(6,"and then start, or just");
	displayString(7,"press enter to start");
	while(!getButtonPress(buttonAny))
	{
	}
	if(getButtonPress(buttonEnter))
	{
		while(getButtonPress(buttonEnter))
		{
			//it's like that dr seuss book
		}
	}
	else if(getButtonPress(buttonUp))
	{
		while(getButtonPress(buttonUp))
		{
		}
		dischargeAmmo(4);
	}
	eraseDisplay();
	displayString(3,"program started");
}

void rotateAngle(int motorPower, int arcAngle, tSensors
gyroMeter)
{
	int endAngle1 = arcAngle/2;
	int endAngle2 = -arcAngle/2;
```

```
	bool rotateRightDirection = false;

	while(isApproaching(1000,120,10,S4,rotateRightDirection) ==
false
		&& isLoaded(20000,S2) == true &&
	getButtonPress(buttonEnter)
		== false){
		displayString(10, "gyro is: %d",
getGyroDegrees(gyroMeter));
		if(rotateRightDirection)
		{
			motor[motorA] = motorPower;
			motor[motorD] = -motorPower;
			if(getGyroDegrees(gyroMeter) >= endAngle1)
			{
				rotateRightDirection = false;
			}
		}
		else
		{
			motor[motorA] = -motorPower;
			motor[motorD] = motorPower;
			if(getGyroDegrees(gyroMeter) <= endAngle2)
			{
				rotateRightDirection = true;
			}
		}
	}
}


task main()
{
	SensorType[S2] = sensorEV3_Color;
	wait1Msec(50);
	SensorType[S4] = sensorEV3_Ultrasonic;
	wait1Msec(50);
	SensorType[S3] = sensorEV3_Gyro;
	wait1Msec(50);
	SensorMode[S2] = modeEV3Color_Color;
	wait1Msec(50);
	SensorMode[S3] = modeEV3Gyro_RateAndAngle;
	wait1Msec(50);
	bool firstFire = false;

	startUp(S2);
```

```
	resetGyro(S3);

	while(isLoaded(20000,S2) == true &&
getButtonPress(buttonEnter) == false)
	{
		rotateAngle(15,180,S3);
			if(firstFire == false)
			{
				releaseSling(-30,-120);
				firstFire = true;
			}
			else
			{
				releaseSling(-30,-47);
			}
			wait1Msec(500);
	}
	motor[motorA] = motor[motorD] = 0;
	while(getButtonPress(buttonAny))
	{
	}
	eraseDisplay();
	displayString(7,"press enter to discharge and end");
	while(!getButtonPress(buttonEnter))
	{
	}
	while(getButtonPress(buttonEnter))
	{
	}
	dischargeAmmo(4);
	eraseDisplay();
	displayString(3,"Program ended.");
	wait1Msec(2000);
}
```

Below is the testing data

Detection Tests		Reading error = ± .5mm | ± 1°					
Still measurement		Stopped at 120±1cm	Rotation of 180°		Firing Consistency (Single Shot Past 10cm)		
Trials	Distance (cm)	Stopped	Degrees offset	After 5 loops	Distance	Successfully shot 10 cm	
1	123.05	1	178	181	43.05	1	
2	127.65	1	179	182	121.85	1	
3	119.2	1	179	184	76.7	0	
4	121.65	1	179	179	20.45	0	
5	123.55	1	179	180	25.6	0	
6	129.35	1	180	180	49.85	1	
7	122.05	1	181	181	10.4	0	
8	126.6	1	179	183	162.55	1	
9	120.15	1	180	180	12	1	
10	128.05	1	181	182	29.95	1	
11	125.8	1	180	182	69.25	0	
12	129.1	1	180	180	190.2	0	
13	123.45	0	179	183	157.15	1	
14	125.65	1	178	185	75.75	0	
15	123.25	1	181	183	122.3	1	
16	127.75	1	181	184	35.6	1	
17	122.5	1	180	182	58	1	
18	123.35	1	182	182	15.45	1	
19	124.9	1	180	183	214.25	1	
20	126.15	1	179	182	17.95	0	
21	124.8	1	180	182	135.75	0	
22	121.65	1	180	183	129.25	1	
23	122.5	1	180	184	56.45	0	
24	125.25	1	179	180	81.05	0	
25	127.6	1	179	185	92.95	1	
26	129.5	1	180	182	25.9	1	
27	123.55	1	178	182	271.35	1	
28	122.25	1	180	184	147	0	
29	120.95	1	181	184	53.5	1	
30	122.4	1	181	184	13.05	1	
Average:	124.455	0.966667	179.7667	182.2667	83.81833	0.6	
STD:	2.771923	0.179505	0.989388	1.590248	66.55301	0.489898	
Range:	10.3		4	5	259.35		
Median	123.55		180	182	63.625		