`

Team Myanmar

Engineering Notebook for 2019 First Global Competition (Dubai)

Ocean Opportunities

`

This page is intentionally left blank.

`

Contents

Introduction 3
Engineering Design Process.....3
Theme of FGC 2019 (Dubai) 4
How we understand about Ocean Opportunities...4
Brainstorming Process 4
Mechanical Design 9
Design thinking 9
Shooting Mechanism (Flywheel) 10
Macro-pollutants 15
Problems Faced 19
Software Algorithm and Design 21
Background 21
Problems.....21
Motor Control.....22
Velocity PIDF 23
PID tuning.....25
Differential drive function 26
Twin motor synchronization.....27
Automatic Aiming and Shooting 28
Measurements 28

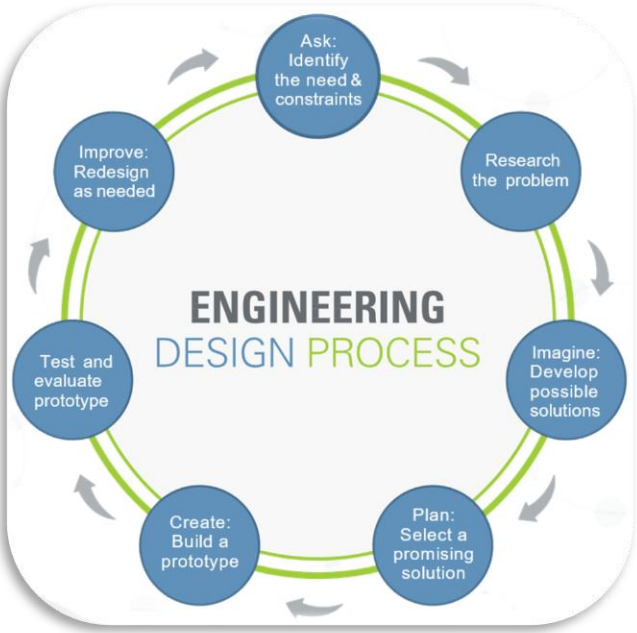
`

Output functions 34
User Friendly Gamepad Interface 36
Control Manual for Flying Dutchman Robot ..37

Introduction

Engineering Design Process

The Engineering design process is a series of steps that engineers follow to come up with a solution to a problem. Many times, the solution involves designing a product (like a machine or computer code) that meets certain criteria and/or accomplishes a certain task.



Theme of FGC 2019 (Dubai)

The name of this year's competition is "Ocean Opportunities". First, we try to understand the Goal of FGC as well as this year's title.

How we understand about Ocean Opportunities

As our mother earth is covered with 71 percent of water which is known as the Ocean. We humans keep on evolving while on the other hand we harm the Earth by means of pollution.

The worst of this is polluting the Oceans with plastics and other unbiodegradable wastes. Thus, in other to make the youths and citizens around the Myanmar as well as the World to raise the awareness and value the ocean, this competition is held.

Brainstorming Process

In this year, the game is to throw micro-pollutants and macro-pollutants into the processing barge which is located at the center of the field. In order to throw the balls, we must build a throwing mechanism as well as carefully calculate the projectile motion of the ball including the drag resistance.

First, we derived the mathematical model of the projectile motion with the air resistance. Even our university taught us about projectile motion, the air

resistance is neglected. We can't find the air resistance included projectile nowhere in our textbook and even hard to find it on the internet. Then we decided to derive it ourselves.

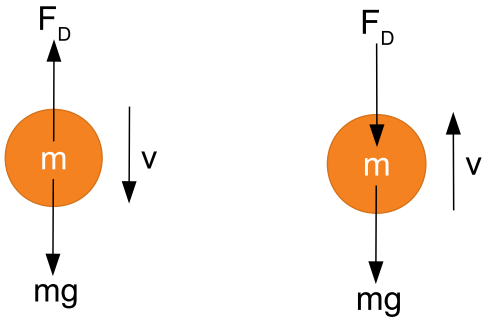


Fig: Free body diagram of pure down(left) and up(right) motion

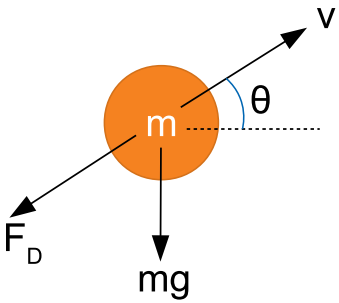


Fig: Free body diagrams related to the derivation of the equation. (where FD is the air resistance, $\frac{1}{2}\rho ADv^2$)

After a series of derivations, the mathematical model and the graph of our motion is ready. But one thing left to know is the initial velocity of the pollutant(ball).

We calculated the initial velocity as follows.

At first, we tried to calculate it by measuring the total flight time of the ball.

$$\sum t = \frac{\mathcal{V}}{g}\left[\cosh^{-1}\left\{\frac{1}{\cos\left\{\tan^{-1}\left(\frac{v\sin\theta}{\mathcal{V}}\right)\right\}}\right\} + \tan^{-1}\left(\frac{v\sin\theta}{\mathcal{V}}\right)\right]$$

Where, $\sum t =$ total time

$$\mathcal{V} = \sqrt{\frac{gm}{k}} \text{ where } k = \frac{1}{2}\rho AD \text{ (Drag Const)}$$

$$v = initial\ velocity$$

$$g = acceleration\ due\ to\ gravity (9.81 ms^{-2})$$

$$\theta = 60°$$

The above equation can only determine the motion that start from a certain height (or ground) and end up exactly **in the same elevation**.

As the flywheel mechanism is set up on the robot, it has initial height of **0.35m**, that should not be neglected in the calculation. Then, the upgraded time equation of the motion with initial height and comes up with

$$\sum t = \frac{v}{g}[\cosh^{-1}\left\{e^{\frac{k}{m}(\frac{m}{2k}[\ln|\frac{v^2\sin\theta^2}{v^2}|]+0.31)}\right\} + \tan^{-1}(\frac{v\sin\theta}{v})\,]$$

It is not exact enough to measure the time, but what we can exactly measure is the range of the ball. So, we then determined to derive a range equation and eliminate the time function in it.

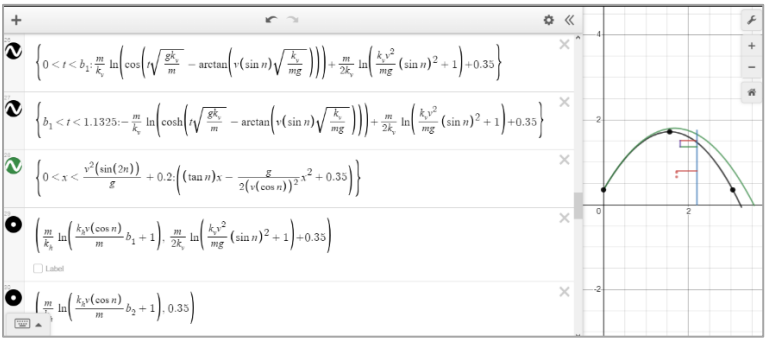
$$R = \frac{m}{k}\ln\{\left(\frac{kv(\cos\theta)}{m}.(\sum t)\right) + 1\}$$

By replacing the $\sum t$ function,

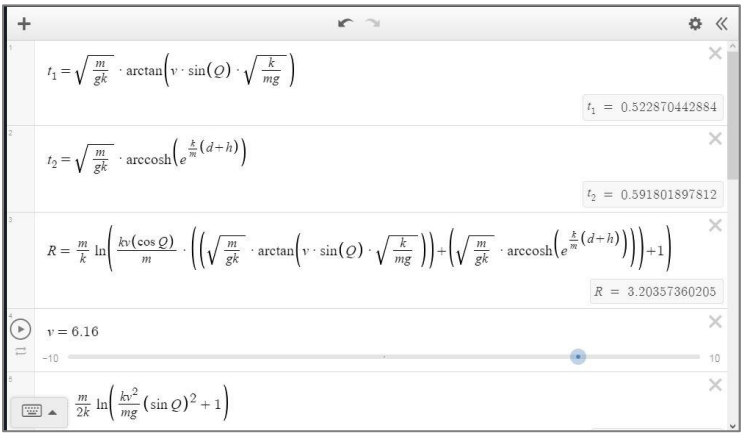
$$R = \frac{m}{k}\ln\{\left(\frac{kv(\cos\theta)}{m}.\left(\frac{v}{g}[\cosh^{-1}\left\{\frac{1}{\cos\left\{\tan^{-1}\left(\frac{v\sin\theta}{v}\right)\right\}}\right\} + \tan^{-1}\left(\frac{v\sin\theta}{v}\right)\right)\right) + 1\}$$

Finally, we now have a function that only include the range and the initial velocity of the ball, the others are known values. After a series of numerical analysis, we got the initial velocity of the ball that come out from our flywheel is **6.16m/s**.

By this velocity, the ball can go up to 1.557m of maximum height that is acceptable value for the reuse level of the processing budge.



The trajectory of the ball (black: real motion, green: neglect air resistance)



Numerical Analysis

Mechanical Design

Design Thinking

For the first design, we decided to make the frame as light as possible since the robot must make the docking at the end of the game. We worried that the motors could not be strong enough to hold it for a certain amount of time or the extensions or components given are not strong enough to maintain the lifting condition. But after reading the specs about the robot kit and reading precise data about motors, they have incredibly high torque.

The other thing we brain-stormed is that how to place the shooting mechanisms for micro and macro pollutants. One of the members suggested to make top and bottom for micro and macro pollutants respectively. But the mechanism at the top will interfere with the bottom one while building the robot. So, placing the mechanisms side by side is suggested. The rest of the members were concerned about the space taken when placing them side by side.

Shooting Mechanism (Flywheel)

For shooting micro-pollutants, the pollutants will be placed between the two wheels which are rotating very high speed. When the pollutants get into those wheels, they will be fired to the desired location. The two wheels which act as shooters will be controlled by a single motor and connected through gears. They will be rotating with same speed so that the pollutants won't change direction but move straight while shooting. We also have to increase the gear ratio by connecting them with the external gears which means the speed will be increased up to that of the output speed of motor.

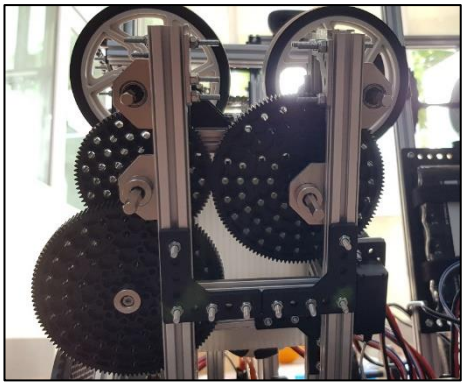


Fig: Bottom view of shooting mechanism which will indicate the idea of increasing gear ratio

Collecting mechanism:

So, as for the micro-pollutants, they will be collected and move up to the shooting mechanism with belts and there is a servo at the top to get them into the shooting mechanism one by one. They are collected by the continuous rotating servo with the elastic bands at the end. The purpose of the elastic bands is to reduce the bounce effect. Then the ball is collected through the belt. The belt has two motion, vertical and horizontal motion. The unique feature is

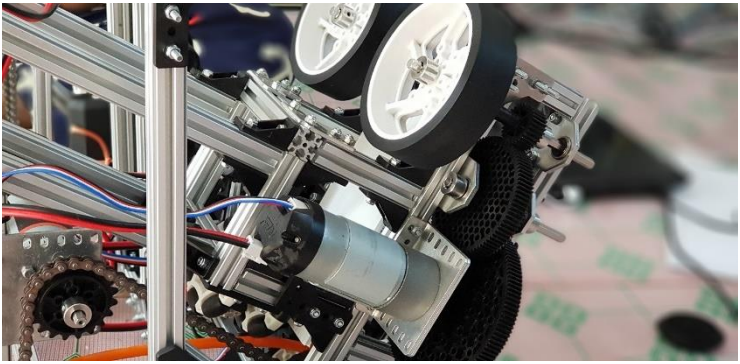


Fig: Shooting mechanism for micro-pollutants. (The pollutants will be fired through the wheels shown above)

that both motions are run by a single motor. The transfer of motion is done by the chains and gears. There is no container in this construction, but the pollutants can be saved along the belt. We must build it as compact as possible so that it can leave space for

design related to macro-pollutants as it is limited with 50cmx50cmx50cm dimensions.

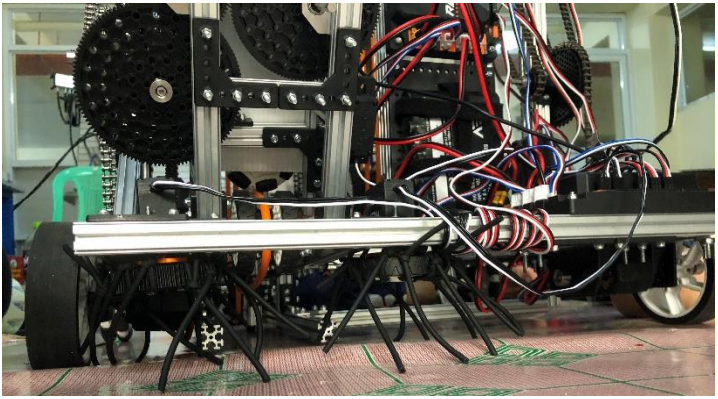


Fig: Elastic bands at the front in order to reduce the bounce effect

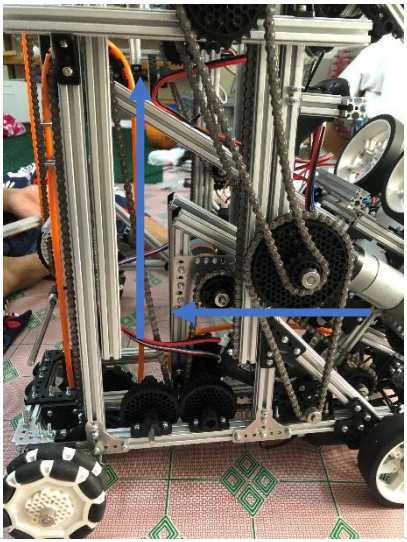


Fig: The motion driven by a single motor (the arrows indicate the direction of the motion)

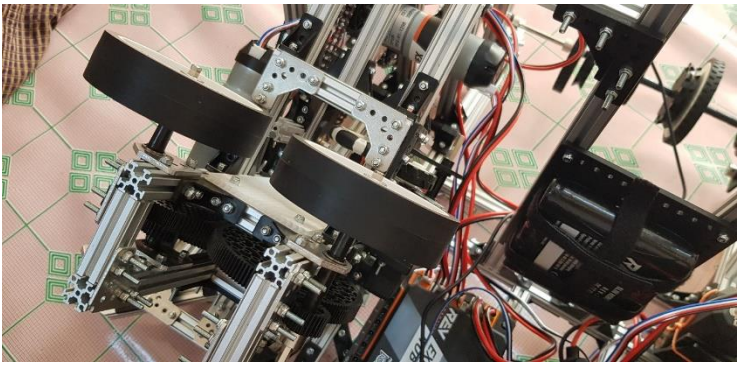


Fig: Top view of shooting mechanism

Theory: We used the concept of conservation of energy in this process. That is, the kinetic energy in the rotating two wheels will transfer into the kinetic energy of the micro-pollutants which will make them speed up for shooting.

$$KE_{wheels} = KE_{ball} + \text{Other energy}$$

KE_{wheels} = Kinetic energy in flywheels

KE_{ball} = Kinetic energy in micro-pollutant

The other energy in the equation refers to heat, sound and other effects which are caused by friction force and some other interferences when the pollutants are shot. Also, the micro-pollutants are elastic and will make the bounce effect which support them to increase the speed. The bounce effect occurs as the micro-pollutants are contracted by the rotating

wheels and then released. To increase the speed of rotating wheels, we connect them with a single motor through gears. As we constructed, the gears are joined as <1>125 teeth gear to 30 teeth gear,

<2>30 teeth gear to 90 teeth gear (same axis) and

<3>90 teeth gear to 30 teeth gear which will end up connecting with the wheel.

The formula of calculating gear ratio for speed is

$$\frac{number\ of\ teeth\ on\ driver\ gear}{number\ of\ teeth\ on\ driven\ gear}$$

The ratio will be

<1> $\frac{125}{30} = \frac{25}{6}$

<2>$\frac{25}{6}$ (the gears are on the same axis)

<3> $\frac{90}{30} = \frac{3}{1}$

The total gear ratio for speed will be $\frac{25}{6}$ x $\frac{3}{1} = \frac{25}{2}$ or 25:2 (12.5) which means that the speed will be 12.5 faster than the original speed generated by the motor. Increasing the speed makes the micro-pollutants to push with greater kinetic energy.

Macro-pollutants

Even shooting of micro-pollutants need that much speed, shooting of macro-pollutants is nearly impossible, so instead of throwing, we decided to lift them up to that “recycle level” which will join with docking mechanism. The macro-pollutants are collected and moved to the top of the robot with belts and linear motion mechanism. At the end, there is a servo motor for placing the pollutant to the “recycle level”. There is also no container in this mechanism and the pollutants will be saved in the belt way. The maximum number of pollutants can be saved is three before taking them to the “recycle level”. The actual number of pollutants can be saved in the belt will be less if we don’t make the sliding mechanism with extensions. So, the belt is also connected with extensions and will be go up along through it so will get more space for pollutants. It also helps to get higher enough for “recycle level” to put the macro-pollutants.

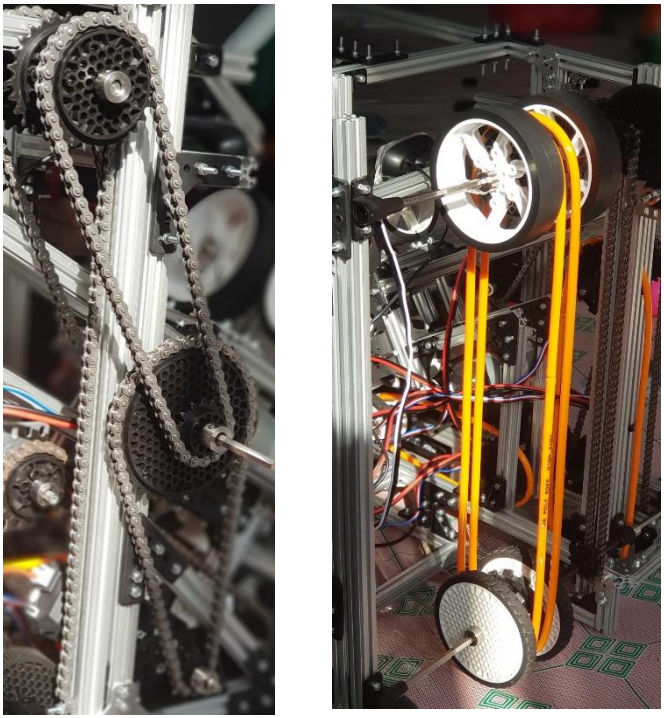


Fig: Mechanism for collecting macro-pollutants which is jointed with the docking mechanism

By the end of construction for macro-pollutants and docking mechanism, we got the problem of motors not strong enough for vertical linear motion since it will heavy with components. We solved this by increasing torque for the output of motors, but the speed of the process will be decreased. We decided to use this technique since the speed will not be that necessary compared to torque

in this process. Increasing the torque with chains and sprockets which are connected to the mechanism for docking and for macro-pollutants is not that easy but we managed to do this which took many hours.

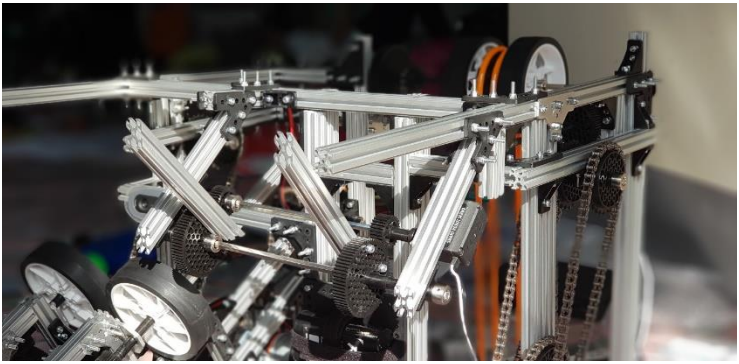


Fig: Mechanism for putting macro-pollutants on to the "recycle-level"

Theory: Since we must increase the torque of the mechanism, the teeth on sprockets must be calculated and connected with chains. The concept of increasing torque is the opposite to that of increasing speed. We use sprockets instead of gears but there is no difference between formulae. As we constructed, the sprockets are joined as

<1>10 teeth to 40 teeth,

<2>40 teeth to 15 teeth (same axis) and

<3>15 teeth to 26 teeth which will end up connecting with the rack gear (we will review about this later).

The formula of calculating gear ratio for torque is

$$\frac{number\ of\ teeth\ on\ driven\ gear}{number\ of\ teeth\ on\ driver\ gear}$$

The ratio will be

<1> $\frac{14}{10} = \frac{4}{1}$

<2> $\frac{4}{1}$ (the gears are on the same axis)

<3> $\frac{26}{15}$

The total gear ratio for speed will be $\frac{4}{1}$ x $\frac{26}{15} = \frac{104}{15}$ or 104:15 (6.933)

Which means the torque will be increased 6.933 times to that of produce by the motor and the speed of it will be reduced. At the end of this torque increasing mechanism, there is 26 teeth attached with the chains on extension which acts as a rack gear. It is the main component for liner motion which moves vertically. The main point of using this system is that they provide higher load capacity. By increasing the torque solves the problem of motors not strong enough to carry the heavy mechanisms.

Problems Faced

Having insufficient amount of provided 8mm screws becomes the main problem when construction had been taking place for a while. We solved this by cutting other very long screws included in the kit which are not necessary as much as 8mm screws. Also, other components such as spacers and bearings are also not enough in the kit which we have to consider since the design will be limited according to that.

Limiting the robot dimension was also the other problem we faced since we must make it as compact as possible. We must make the frame strong by using brackets and extensions but at the same time we should not use them much since we have to consider the docking operation and it might weigh too much for motors to hold it for a certain amount of time. But the motors are strong enough and gear ratio can be raised so this problem was solved easily.

Due to the overweight, the bending of the frame at the back of robot becomes a problem as there is nothing that helps to keep it in position. We solved this by adding an additional mini-omni wheels at the back of the frame, as shown in the following figure.

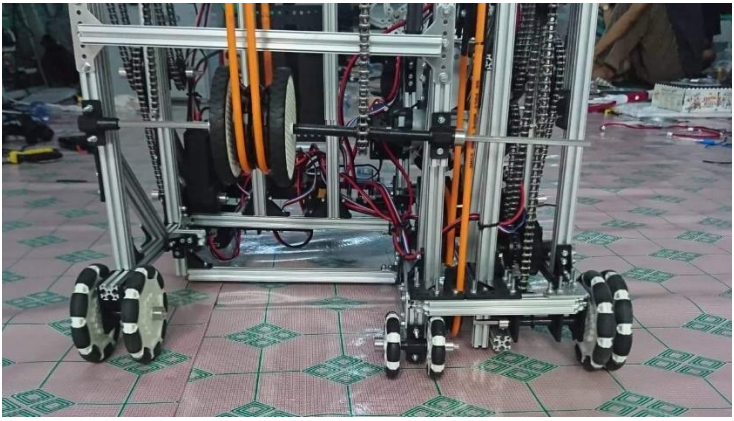


Fig: Showing of mini-omni wheels in the middle at the back of the frame

Software Algorithm and Design

Background

Running on a *Java* based *Android OS, REV Control Hub* mainly supports *Java*. It also supports block programming. Our team prefers text-based programming other than graphic-based programming. Most of the classes written for the program are customized versions of FTC classes. For example, our motor class has an extra feature of power transition, servo class has a non-blocking sequential control.

Problems

One of the objectives of the game is to collect the small balls (micro pollutants) as many as they can be and shoot them to the highest level (the reuse level of the processing barge). This seems easy to collect and get many points. Due to the limitation of number of balls, the game could be a little bit aggressive. The mechanisms are designed to collect and place the balls instantly. The robot can be controlled with a gamepad controller. The mechanism contains a lot of actuators and controlling them by a player could be difficult. The part of software interfacing the user inputs should be designed to be user friendly. It should use less buttons for controlling a mechanism.

One big problem is even if the manual control is user friendly, it has still a high chance of missing the balls. It is very difficult to aim from the player's perspective. So, an automatic function for aiming is used. This is where we put a lot of effort in software designing. This uses extra sensors, a webcam and sensor fusion algorithms.

Motor Control

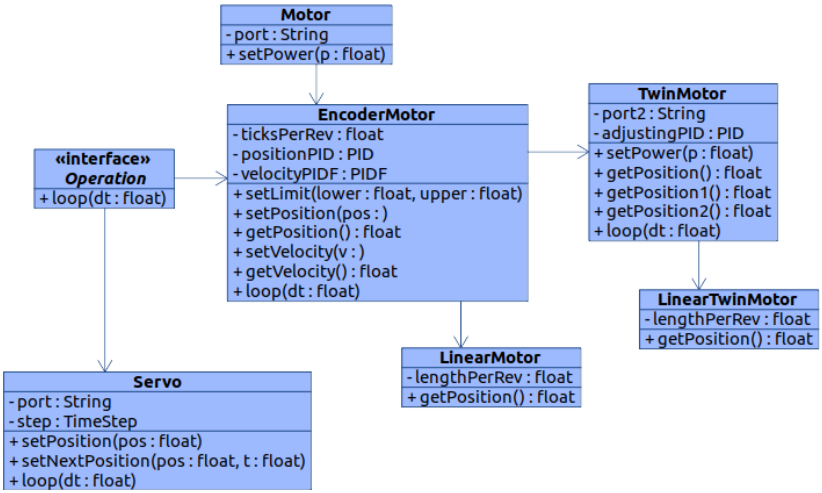


Fig: Motor class hierarchy

The encoder motors in our robot use closed loop control systems. In order to have low error with high efficiency in actuating the motors, we mainly

use PID controllers as a closed loop control. PID control is an error-based control system and it is calculated based on proportional, differential and integral errors in order to obtain the precise output power. The coefficients for these 3 errors are adjusted using the real-time tuning. (See The following fig).

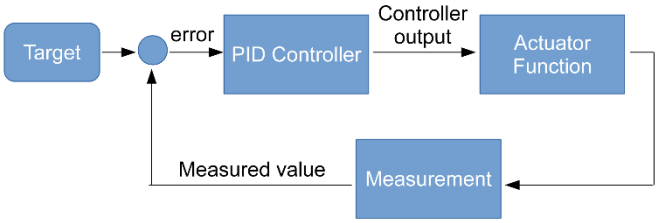


Fig: Simple PID algorithm

Velocity PIDF

Velocity PIDF is a control system used in flywheel and motors used for driving wheels. Unlike position, it is impossible to obtain the desired velocity using PID controller alone. When the target is reached, a constant feed is still needed to keep it moving. So, PIDF, which has an open loop feed-forward controller, a target-based formula estimating the output for the target is used.

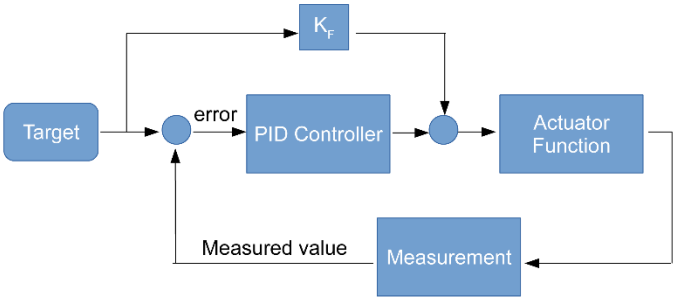


Fig: PIDF algorithm

Since the data obtained is very oscillated, we used a signal smoothening algorithm, *Moving Average* in order to get average velocity curve which is smoothened. Moving average is an algorithm which connects a series of averages of the data to smoothen the data curve. First, we determine the average of data at a set amount of time repeatedly and the averages are connected and thus, the data is filtered.

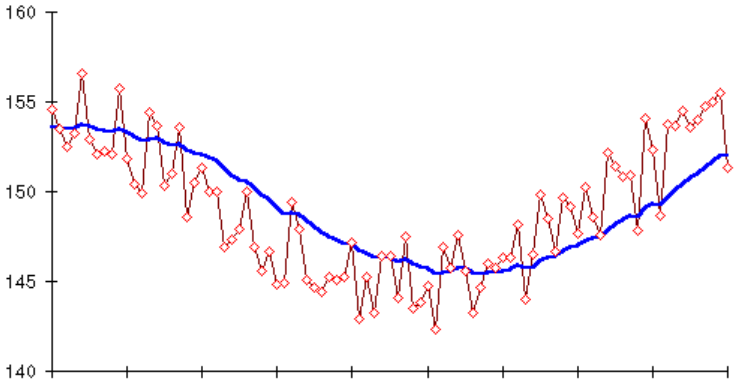


Fig: Oscillated values filtered by moving average algorithm

PID tuning.

As it is pretty handful to deal with PID/PIDF variables and compile again and again, we decided to use real-time tuning using the analog potentiometer inputs so that we could adjust the variables without compiling the code repeatedly.

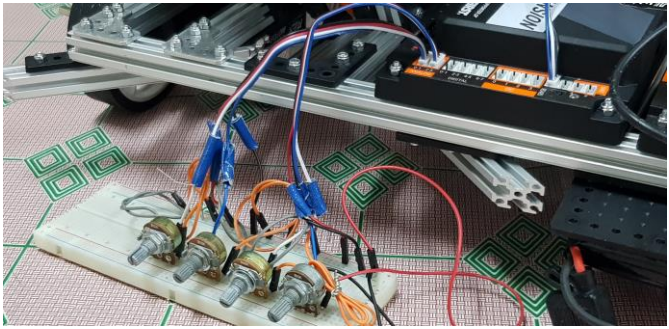


Fig: Circuit for PID tuning

Differential drive function

For differential drive wheels, a smooth output of the motors is considered. First a discrete function with four states front, left, right, back is noted. In order to get a continuous function with respect to direction, a cosine function is used for transition from one state to another. Multiplying with a speed value with that function output which is between -1 and 1, we get a function of speed and direction. This is useful for smooth joystick control. Output function for feedback control systems is much better with continuous functions other than discrete functions. So, this differential drive function suits such control systems like moving the robot to a specified coordinate.

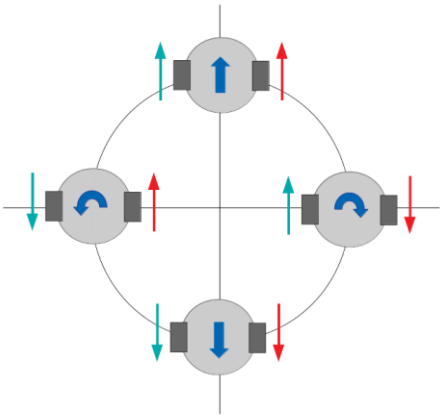
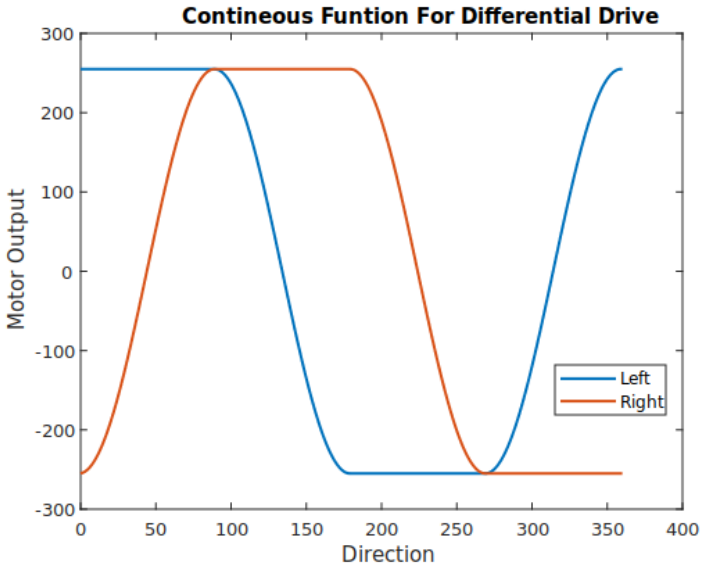


Fig: Discrete differential drive function



Twin motor synchronization

The robot contains a elevator-like mechanism which is used to lift the macro pollutants and perform docking. The mechanism contains two linear motors, and these two motors need the same velocity in order to gain efficient performance. Unless the motors are synchronized, the lift alignment may become oblique and if it is too much, it might cause serious damage. The two motors have separate position or velocity PID blocks. The extra PID block gives the controller output depending on the position difference which is then added together with the former PID output.

Automatic Aiming and Shooting

The manual control is not suitable for aiming and shooting as there can be human error, so we decided to have and autonomous mode for aiming and shooting. There were numerous errors but after going through a lot of rough times, the result was very satisfactory.

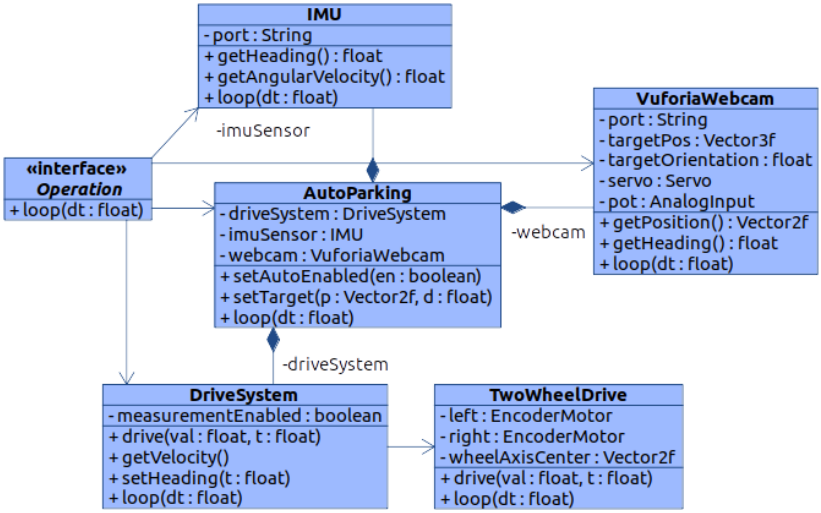


Fig: Classes concerning automatic aiming and shooting

Measurements

The *AutoParking* class uses the measurements of IMU, webcam and encoder motors and drives the robot to a precise distance from the specified coordinate with the heading towards it. For

measurement functions, IMU and webcam are fused to obtain heading and for the coordinate, encoders and webcam are fused. The sensors are fused using *The Complimentary Filter*. The sensor which is accurate for a short amount of time yet drifting over time passes through the high pass filter and the sensor which is reliable for a long amount of time but is not very accurate for short usage passes through the low pass one.

Measuring robot heading

In order to obtain the robot's current heading, we use the gyroscope for prediction and the magnetometer and webcam for measurement. The IMU class uses a super-efficient BNO*055* chip with gyroscope, accelerometer and magnetometer. The chip has a built-in sensor fusion algorithm and is implemented.

The webcam is used together with *Vuforia* library. The direction of the image with respect to camera is calculated by bending and vanishing point of the square frame.

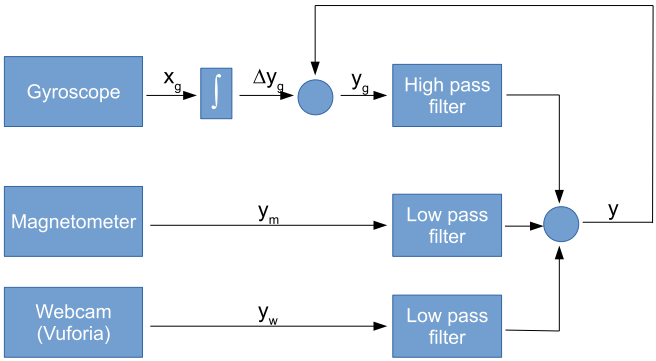


Fig: Complementary filter for robot heading

Measuring robot coordinate

Odometry with motor encoders

Odometry is the field of collecting data which calculates the robot current position by the use of motion sensors. It tends to have errors due to the integration of distance change and angle change in order to get the velocity and the angular velocity. It also can have non-systematic errors caused by the interference factors of the environment we cannot see such as uneven floor or slipping.

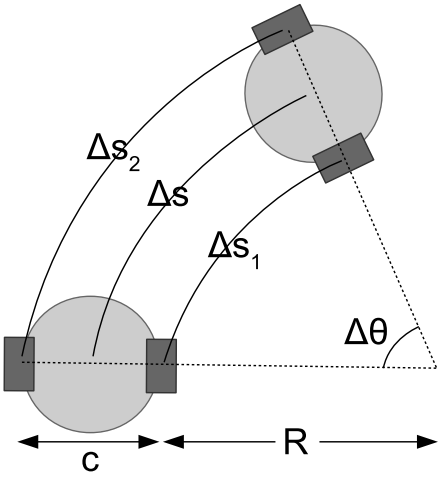


Fig: Odometry kinematics

Deriving from the arc length formula,

$$s = r\theta$$

$$\Delta s = \frac{(\Delta s_1 + \Delta s_2)}{2}$$

$$R = \frac{c \,.\, \Delta s_1}{(\Delta s_2 - \Delta s_1)}$$

$$\Delta\theta = \frac{(\Delta s_2 - \Delta s_1)}{c}$$

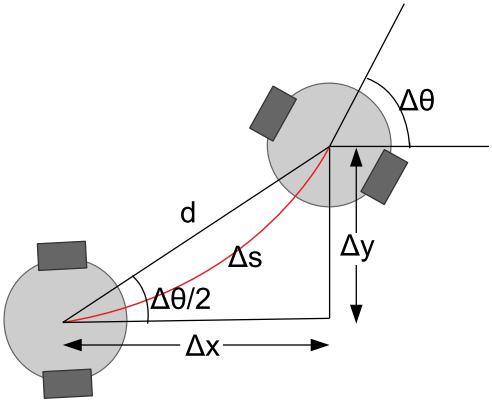


Fig: Odometry kinematics

When the motion is very small, we can assume that $\Delta s \approx d$ and by applying trigonometry equations,

$$\Delta x = \Delta s \cos(\theta + \frac{\Delta\theta}{2})$$
$$\Delta y = \Delta s \sin(\theta + \frac{\Delta\theta}{2})$$
$$\Delta\theta = \frac{(\Delta s_2 - \Delta s_1)}{c}$$

Since odometry is not very reliable on its own, we decided to use webcam and IMU (Inertial Measurement Unit) to cover the errors. The heading of the robot car is obtained from the fusion of IMU

and webcam and the coordinates are gathered by the use of encoders and webcam.

Sensor fusion with webcam data

Since odometry is not very reliable on its own, we decided to aid it with webcam and IMU to cover the errors. The heading of the robot car is obtained from the previous system and the coordinates are gathered using encoders and webcam.

The odometry calculation is taken as a prediction and the webcam the measurement. Webcam measures coordinate position with respect to image by size and translation of the image appeared in the frame. Like the complementary filter in the previous system, the encoders go through high pass filter and the webcam goes through low pass.

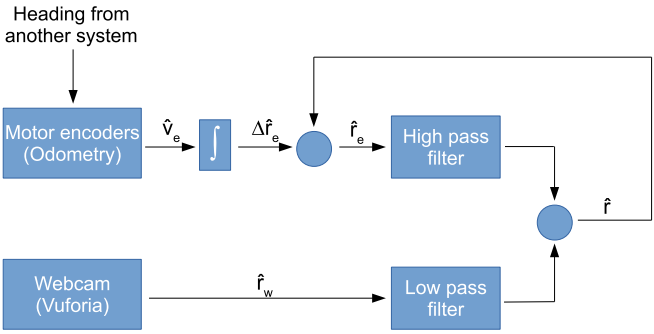


Fig: Complementary filter for robot heading

Output functions

After the measurement step, the distance and direction between the robot coordinate and target coordinate is calculated and passed into two PID controllers, distance PID and direction PID. The distance is determined by the projectile motion formula. The robot can park in anywhere on the circle with the heading towards the target.

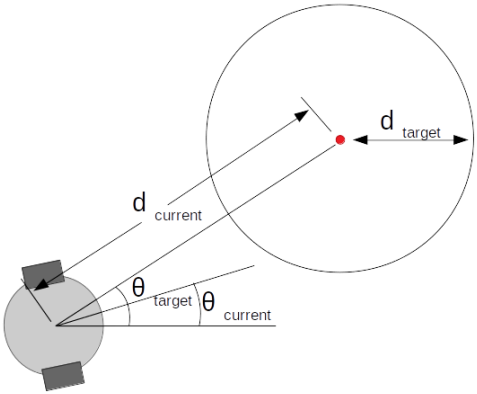


Fig Distance and heading target and measurement

Circular quantity problem

The quantity for the angles will be over 360 degrees when the robot has done fully 360 degrees turn so in order to reset the heading, the reminder of it to 360 is taken. The minimum value is selected from the remainder or the addition of 360 to the remainder or the subtraction of 360 to the remainder. The solutions are implemented in *DirectionPID* class.

User Friendly Gamepad Interface

Getting friendly user-interfaced gamepad control is a must to obtain great efficiency at games. So, we put quite an effort on this aspect as well. The combo pairs are added since there are limited number of buttons. We also made the buttons similar to the video games we are familiar with. For instance, when we press R1 button, the robot will run with high speed just like in FIFA or PES.

Control Manual for Flying Dutchman Robot

Buttons on gamepad	Functions
A	Shoot
↑ + A	Forced shoot
B	Flywheel switch on/off
X	Belt 1 switch on/off
Y	Belt 2 switch on/off
↑ + X	Move belt 1 forward
↓ + X	Move belt 1 backward
↑ + Y	Move belt 2 forward
↓ + Y	Move belt 2 backward
→ + X	Collector servo switch
L1	Inverse driving direction
L2	Push ball from belt to container
R1	Speed up
R2	Drop the macro pollutants
↑ + R2	Quick ball drop
L3 + R3	Auto-aim for projectile
Left joystick	Drive
Right joystick	Lift control

Appendix

Playing Field and Coordinate System

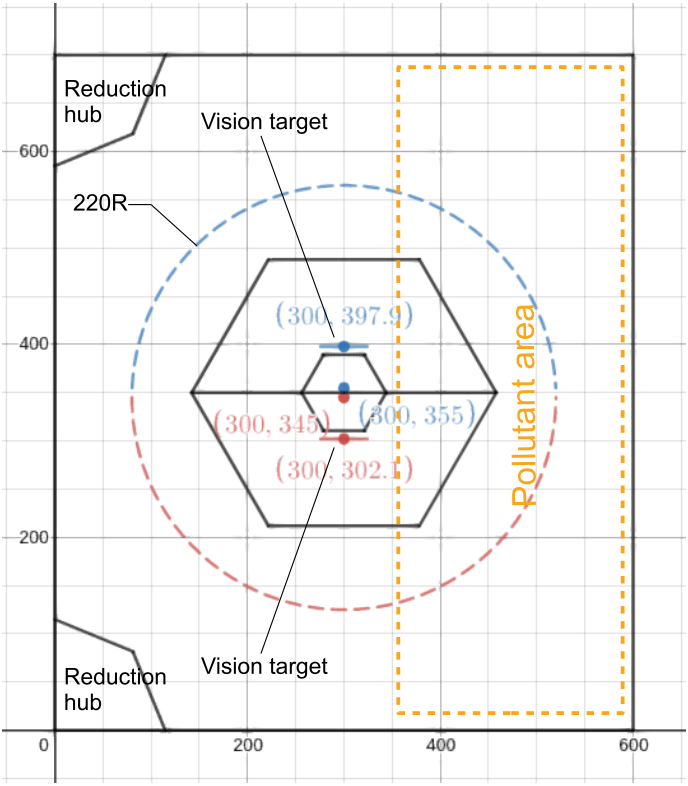
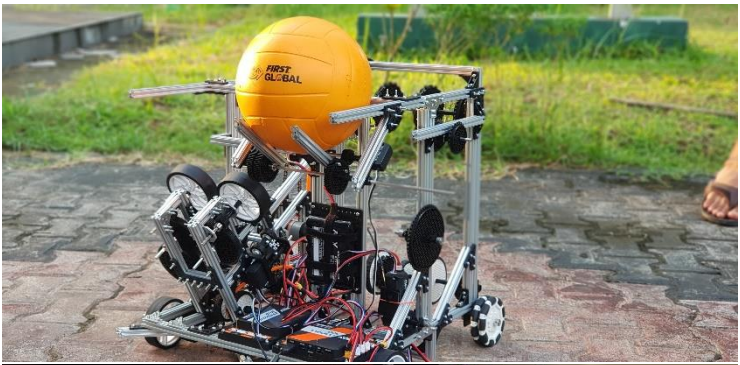
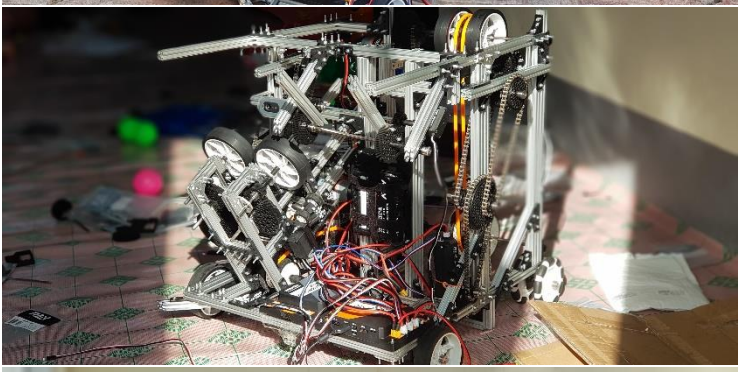


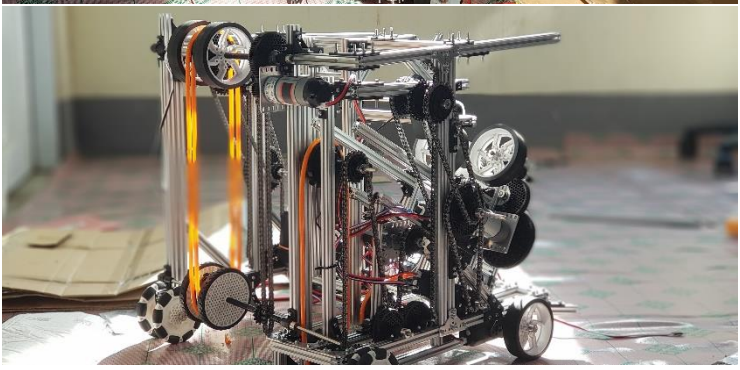
Fig: Playing field and coordinate system

The red and blue circular arcs are the location on which the robot is supposed to parked in order to shoot the projectile where the range is 220cm.

Photos of The Robot







Contributed by

-Khant Kyaw Khaung (Team Captain)

-Thiha Zaw (Mechanical Designer)

-Thaw Dar San (Mechanical Designer)

-Hein Htut Zaw (Programmer)

-Win Naing Kyaw (Programmer)