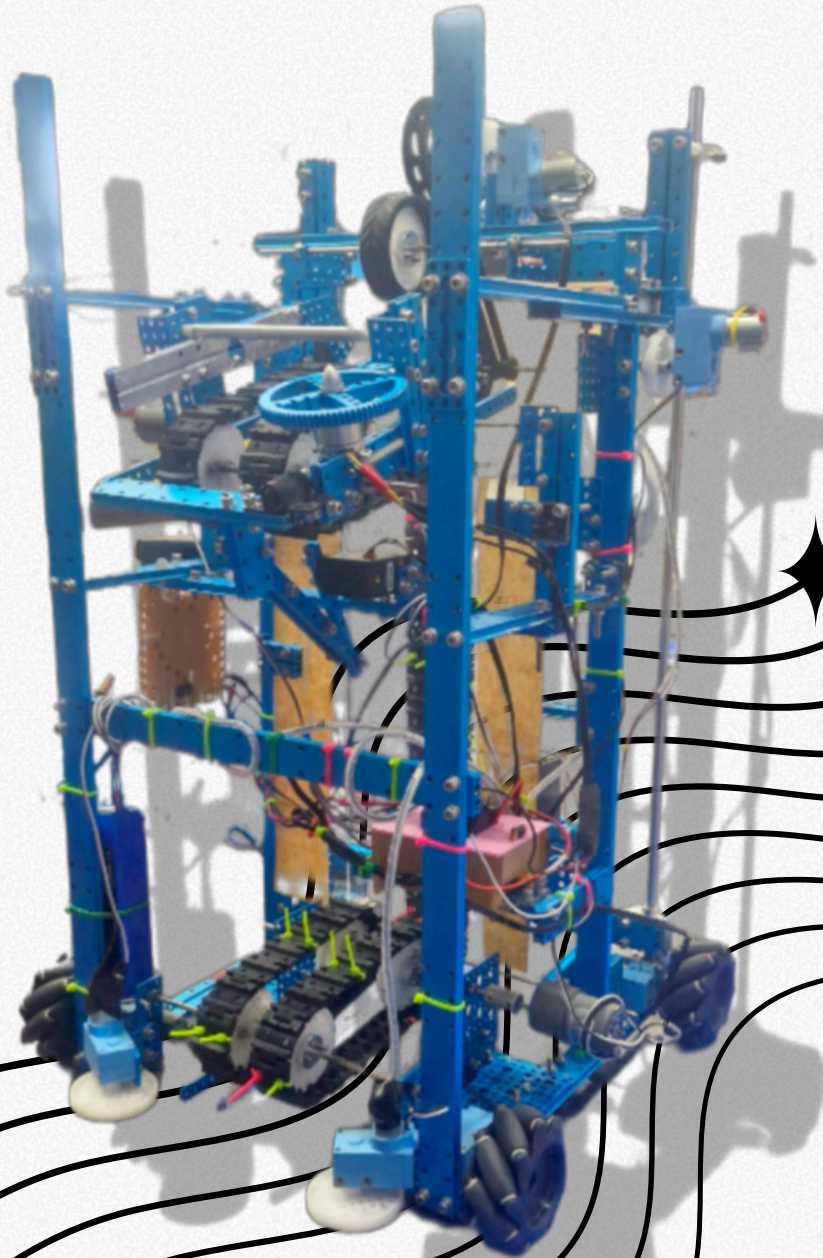
GRAVITYSHIFT

ENGINEERING BOOK

CTHE032

GravityShift

SUKSANAREEWITTAYA SCHOOL



For more Quality of our Engineering book

PREFACE

Build the Robot for competition has many benefits to all the people in the world, the competition can be use as teaching media for the children who are interested in Robotics and Engineering and the Robotics competition is also has create the new inventions to the world by every experience of the contestant.

The MakeX Robotics Competition is a competition that is enjoyable and has a lot of benefits for the contestant, the contestant doesn't come to the competition just for the reward. But they can grained new skills, new friend, create a new conection with each other even a new idea for there own invention in the future.

Finally, In the name of the team that have participated in the competition. We would like to thanks The MakeX and MakeX Thailand for giving all of us the opportunity to be participate in the competition and organize such a useful and amusing competition.

TABLE OF CONTENTS

TABLE OF CONTENTS

TEAM INFOMETION

Mentor

Name : Mr.Yossawee Longtong
Email : yossawee@snws.ac.th
ID line : nongmayhub
Tel. : 0894501023

TEAM INFOMETION

Nick name Shi

Name : Thanadech Lapassirikul

Age : 16

Team role : Team Programers ,
Subport-Mechanic,
Team Technician

Nick name Seven

Name : Tanapat Phinnok

Age : 17

Team role : Pilot ,
Head-Mechanic,
Quality Control

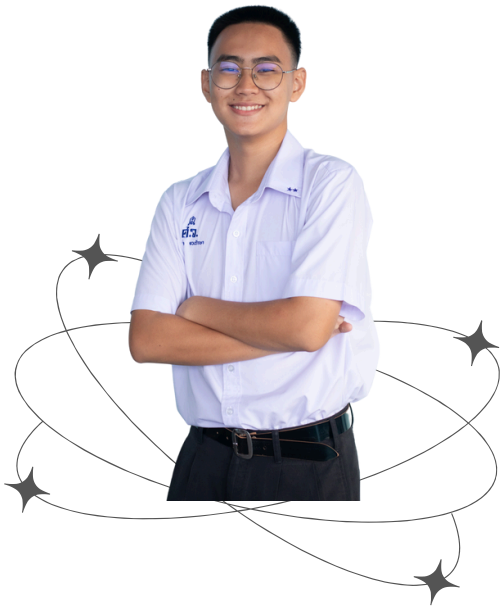
TEAM INFOMETION

Nick name Nut

Name : Natdanai Duangraksa

Age : 16

Team role : Designer ,
Subport-Mechanic



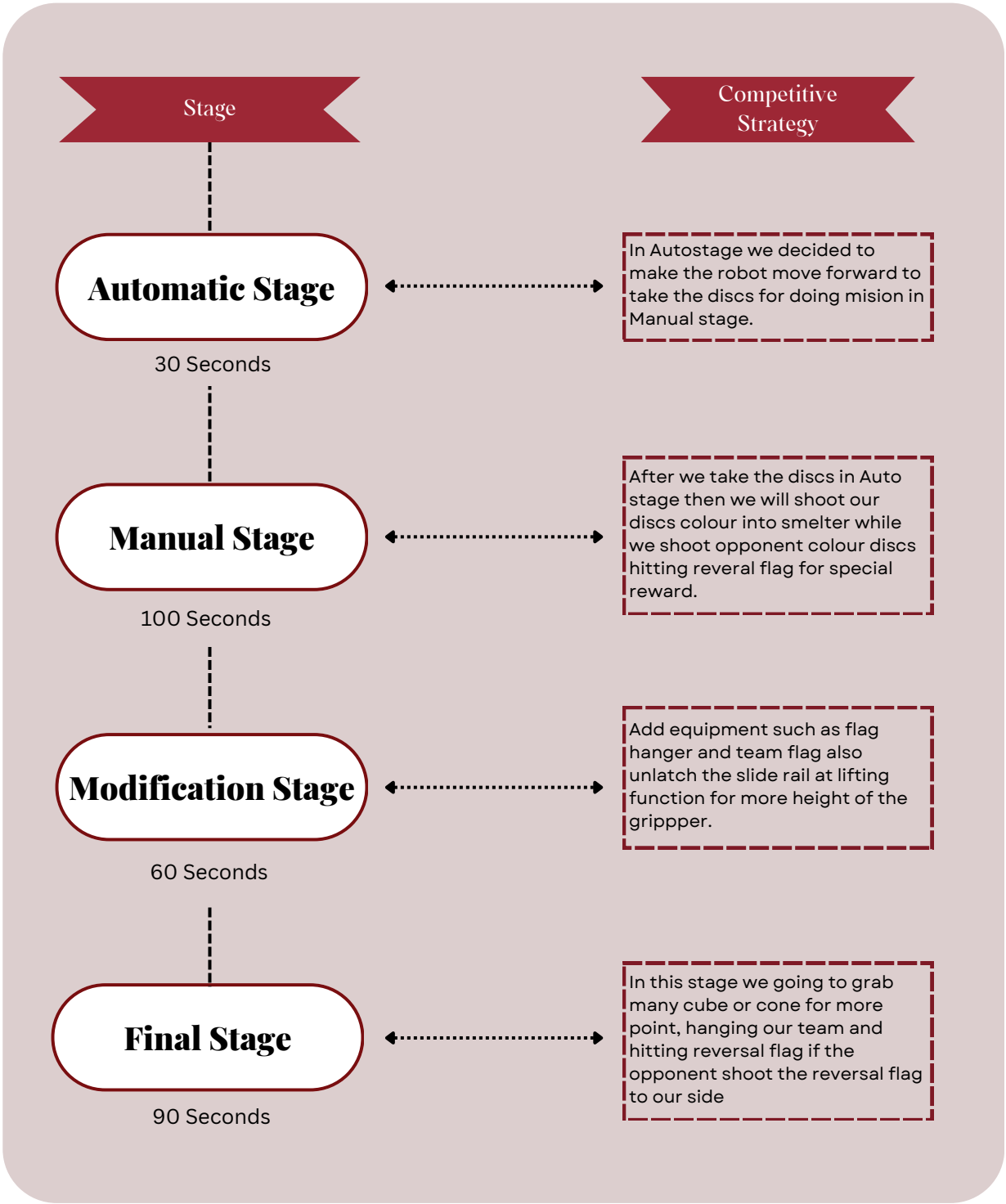
Nick name Tong

Name : Pranutsma Lhwanrueang

Age : 18

Team role : Co-pilot ,
Team Strategist,
Subport-Mechanic

COMPETITIVE STRATEGY



The Competiton include: Automatic Stage, Manual Stage, Modification Stage, Final Stage. The detail of Competitive Strategy for our team is down below.

**** After all these strategy can be changed depend on each match of the competiton ****

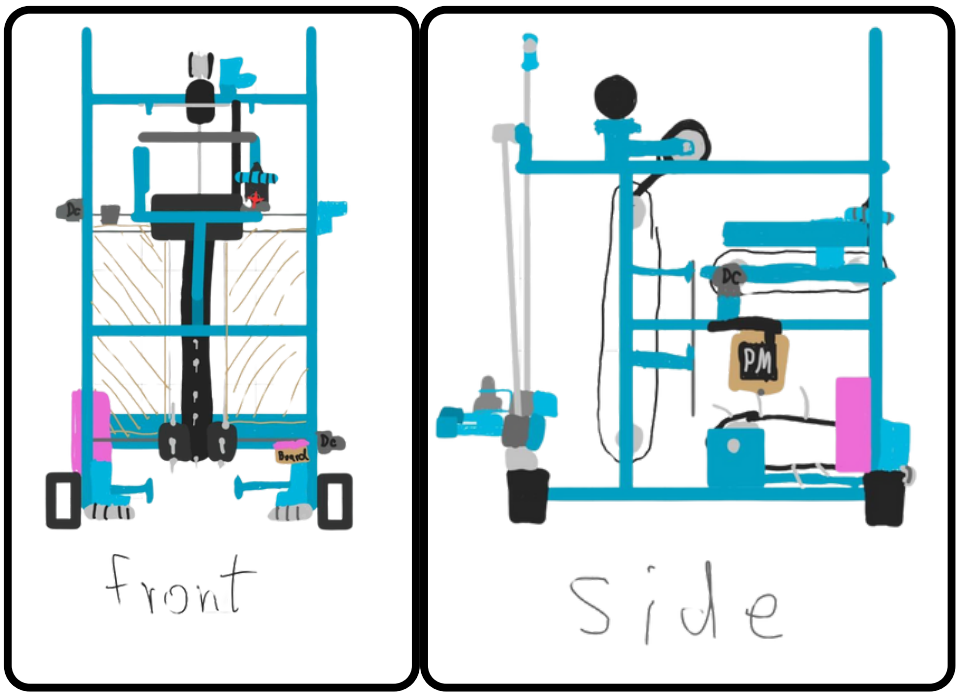
Concept

The concept is to make the robot that has the ability to do every mission efficaincy and works fast.

We had tried a lot of design in each function so that can can discoverd the best design of every funcion.

The design of the robot is refered to other teams and we improved the design so that the system are compateble to our robot.

Overall Design



PERFOMANCE AND PURPOSE

Performance

Drive System	The robot can move in every direction of machanum wheels
Shooting	The robot can shoot discs into the smelter, hitting reversal flag and opponent pins
Storaging Cube,Cone	The gripper of robot can grab the cube and cone not just one
Hanging Flags	The robot can hang our team flag on the flagpole that locate in our own flag-hanging area.
Unique	The robot has installed the cooling system such as DC Blushless Fan to the NovaPi Case to prevent the overheat of the NovaPi and it's expansion board, The heat sink at the encoder motors also to cooling the motors from overheat

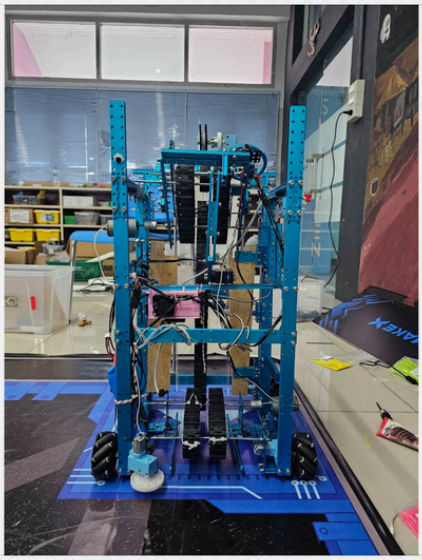
Purpose

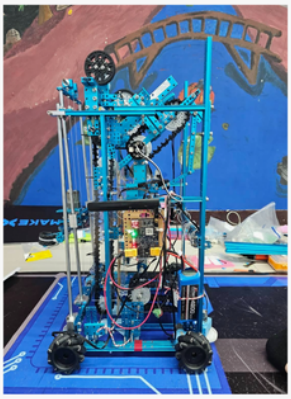
The robot has the ability to do every mission fast and efficiency, the robot can shoot into the smelter, shoot pin and reversal flag, can hold the cube and cone tight

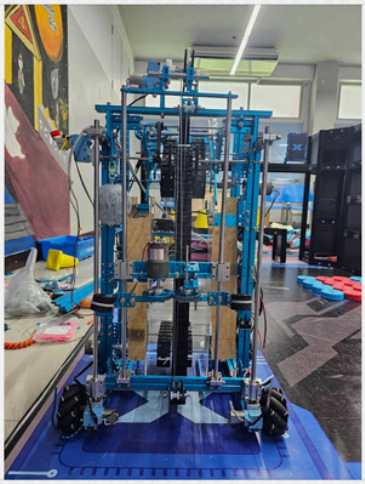
To the Best for the winter.

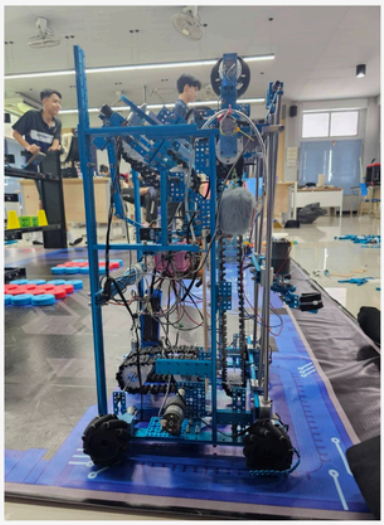
OUR ROBOT

ALL ANGEL VIEW



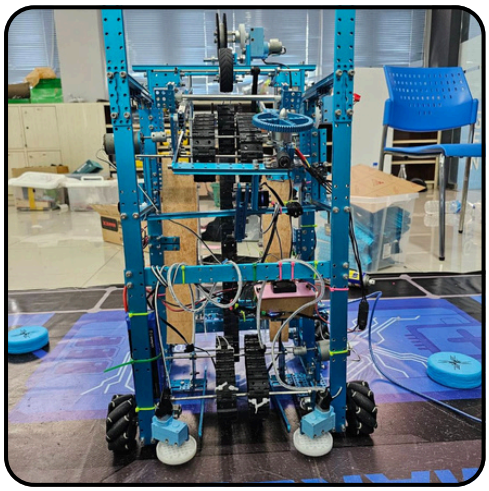


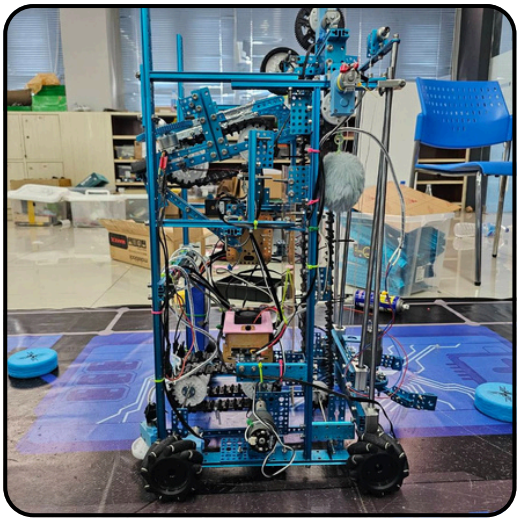




MATERIALS

ALUMINIUM ALLOY



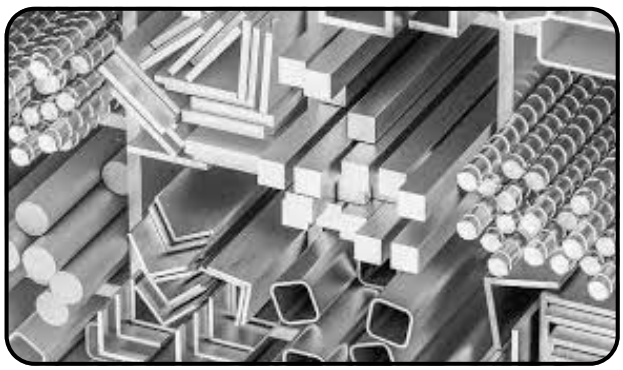


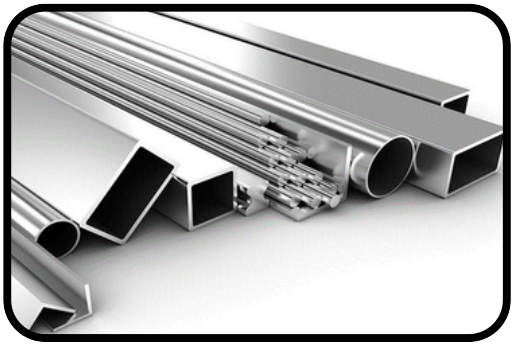
Aluminium alloys are a type of metal that is made by mixing aluminum with other elements, such as copper, magnesium, silicon, and zinc. This process creates a material that is stronger, lighter, and more durable than pure aluminum.

Aluminum alloys are used in a variety of applications, including:

- Aerospace: Aircraft and spacecraft
- Automotive: Cars, trucks, and buses
- Construction: Buildings and bridges
- Consumer products: Electronics, appliances, and packaging

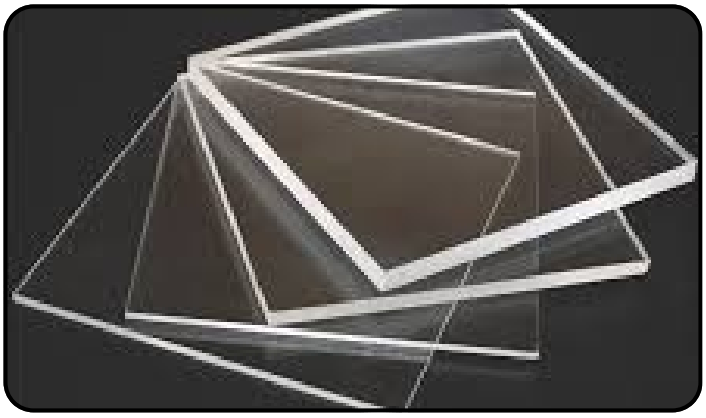
Aluminum alloys are a versatile and valuable material that is used in many different industries.





MATERIALS

ACRYLIC



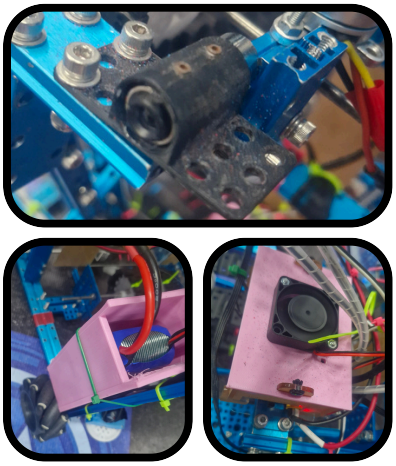
Example of Acrylic sheets.
Referenced : https://setshop.com/4-x-8-x-1-thick-clear-acrylic-sheet-plexiglas/

Acrylic is a versatile material that can be used in a variety of ways. It is a type of plastic that is known for its clarity, strength, and durability. Acrylic is often used as a substitute for glass because it is lighter and more shatter-resistant.
Acrylic is a safe and non-toxic material that is easy to clean and maintain. It is a great choice for a variety of applications, both indoors and outdoors.

Example of Acrylic sheets.
Referenced : https://www.delviesplastics.com/p/clear_acrylic_sheet.html

MATERIALS

3D PRINTTER



Our 3D Printed Part.
Referenced : Team GravityShift.

Our 3D Printer : FlashForce Finder 3.
Referenced : https://www.print3dd.com/product/flashforge-finder3/

3D Printer is a machine that creates physical objects from digital designs. It works by layering materials, one thin layer at a time, until the entire object is formed. This process is called additive manufacturing, as it adds material to build the object, unlike traditional methods that remove material.

The materials that we use in 3D Printer ;
- PLA filaments
- ABS filaments
- PETG filaments

Our 3D Printer Machine ;
- FlashForce Finder 3

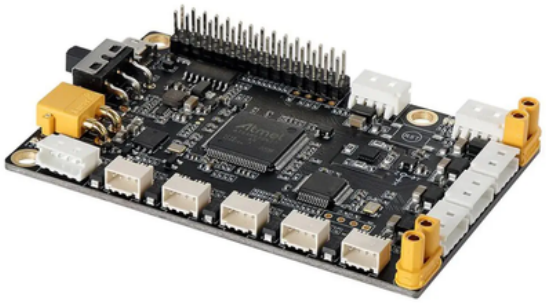
PLA (Polylactic Acid)
- Pros: Biodegradable, easy to print, low odor, low warping.
- Cons: Less durable than ABS or PETG, prone to breaking.
- Best for: Prototypes, models, and non-structural parts.

ABS (Acrylonitrile Butadiene Styrene)
- Pros: Durable, impact-resistant, good for detailed prints.
- Cons: Higher printing temperature, prone to warping, strong odor.
- Best for: Functional parts, enclosures, and toys.

PETG (Polyethylene Terephthalate Glycol)
- Pros: Strong, durable, good layer adhesion, less odor than ABS.
- Cons: Can be more difficult to print than PLA, prone to stringing.
- Best for: Durable parts, containers, and outdoor applications

Example of 3D Printer Filaments.
Referenced :
https://www.crealityofficial.co.uk/products/creality-pla-filament-6kg-packs-mixed-color-bundles

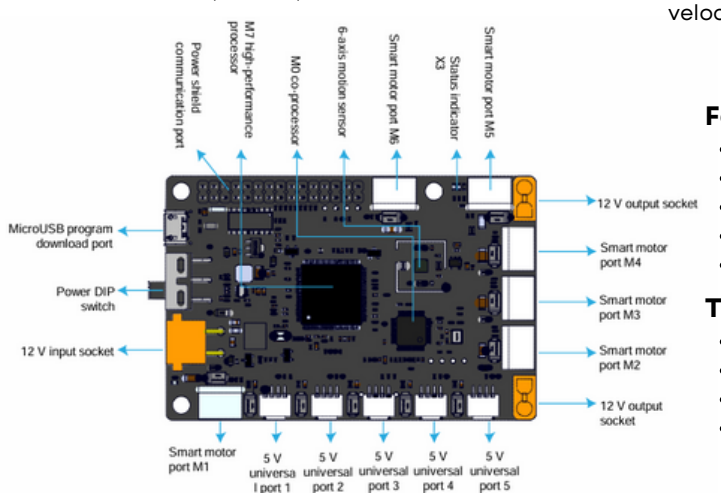
ELECTRONICS
COMPUTING BOARD



NovaPI Computing Board.
Referanced : https://th.aliexpress.com/i/1005004418087757.html

NovaPi Computing Board

NovaPi is the new generation of main control board used in the **MakeX Robotics Competition** (Challenge Program), featuring a M7 high-performance processor (ATSAMS70N20A-AN), one STM32 co-processor (STM32F030CCT6), five 5 V universal ports (For sensors, Bluetooth Module and Power Management Board.), and six smart motor interfaces (For Encoder Motors and Smart Servo). The 5 V universal ports output 5 V voltage. The smart motor ports output 12 V voltage, also the NovaPi has a 6-axis motion sensor integrated that can measure the acceleration velocity and angular velocity along the X, Y, and Z axes.



NovaPI Computing Board Structure.
Referanced : https://www.creativakids.com/images/makex/tutorialchallenge.pdf

Features

- Support the Makeblock metal parts
- Five 5 V universal ports
- Six smart motor ports
- 6-axis motion sensor
- Support power shields

Technical Specifications

- Module dimensions: 85 mm x 56 mm x 13 mm
- Operating voltage: 6 V - 13 V
- Operating current: 60 mA
- Communication port and protocol: serial port/dedicated protocol

NovaPi Expansion Board (Power Shield)

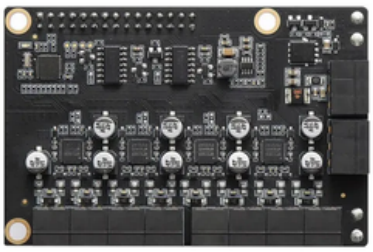
NovaPi Expamsion Board (Power Shield) contains an MCU with the ARM Cortex-M4 core, uses serial interfaces to communicate with NovaPi, and uses eight DC output ports and two brushless motor ports to control the powered devices such as DC motors, mechanical grabbers, electromagnetic valves, and brushless motors.

Features

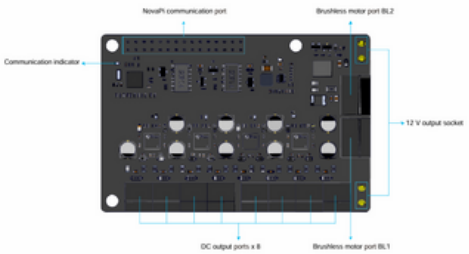
- Work with NovaPi.
- Can control Makeblock series. powered devices such as DC motors and brushless motors.
- Can control common powered devices including DC motors.
- 8 DC output interfaces and 2 Brushless motor interfaces.
- Embedded overload protection circuits to prevent exceptional conditions from damaging the shield.
- Require no extra external power supply.

Technical Specifications

- Module dimensions: 85 mm x 56 mm x 19.5 mm
- Input voltage: 6 V - 13 V • DC port voltage: 12 V
- Maximum DC port current: 3 A
- Brushless motor port voltage: 12 V
- Maximum brushless motor port current: 5 A



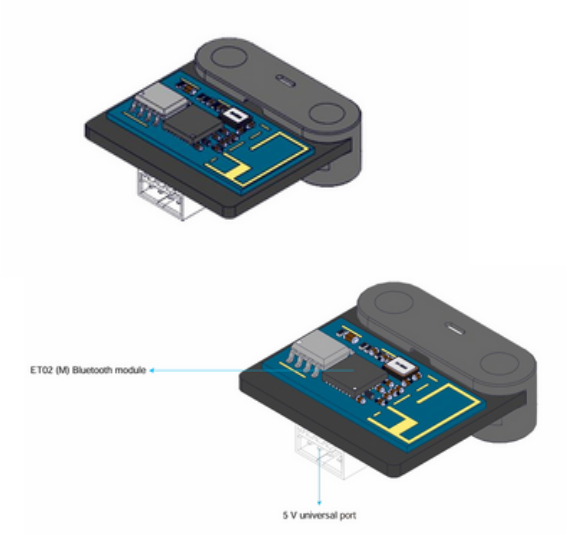
NovaPI Expansion Board.
Referanced : https://th.aliexpress.com/i/1005004418087757.html



NovaPI Expansion Board Structure.
Referanced : https://www.creativakids.com/images/makex/tutorialchallenge.pdf

ELECTRONICS

CONNECTION DEVICES



Bluetooth Module and Bluetooth Module Structure.
Referanced : https://www.creativakids.com/images/makex/tutorialchallenge.pdf

Bluetooth Module

Bluetooth Module implements system connection via Bluetooth to Bluetooth joysticks and Bluetooth adapters. connect to the NovaPi using the 5V universal port comunicate using serial port.

Features

- Support Makeblock metal parts and Lego pins
- Support Bluetooth devices such as joysticks and dongles
- Small size suitable for various projects

Technical Specifications

- Module dimensions: 24 mm x 23 mm x 6 mm
- Operating voltage: 2.5 V - 5.5 V
- Operating current: 20 mA
- Communication port and protocol: serial port communication

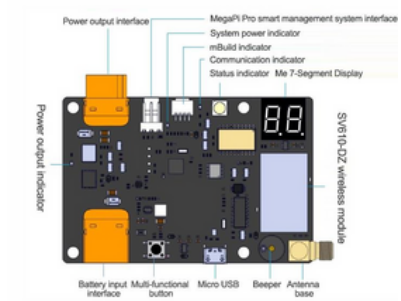
Bluetooth Controller.

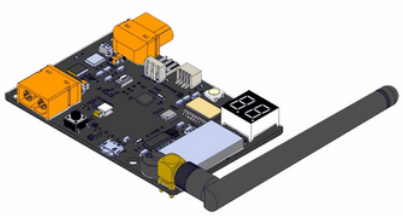
Bluetooth Controller use as a Remote Controller to control the robot connected via Bluetooth to Bluetooth Module that has connected to NovaPi Computing Board, The Bluetooth controller has 15 buttons and two joysticks. The functions of the buttons can be customized and can be used for multiple programs to facilitate operations as soon as the programs are uploaded.

Bluetooth Controller.
Referanced : https://imagineering.co.th/th/shop/bluetooth-controller, https://www.creativakids.com/images/makex/tutorialchallenge.pdf

Technical Specifications

- Material: ABS
- Bluetooth version: 4.0+
- Transmission distance: 20 m
- Operating current: ≤ 25 mA
- Transmit power: 4 dBm
- Frequency range: 2,402 MHz - 2,480 MHz
- RF transmit range: 4 dBm
- Maximum transmission distance: 20 m
- Bluetooth version: 4.0
- Data transmission: Data packets can be obtained by Bluetooth devices within 100 ms (low delay).
- Battery: two AA dry cell batteries
- Supported OS: MacOS / Windows
- External package dimensions: 170 mm x 110.5 mm x 54 mm (L x W x H)
- Product dimensions: 149 mm x 88 mm x 46 mm (L x W X H)





Power Managemengt Module

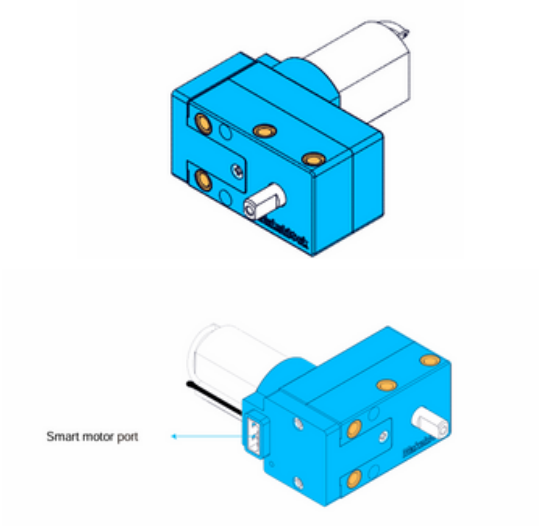
The power management module is an important competition-oriented device. It is designed mainly for wireless connection to the MakeX competition system to monitor the power status of participating teams' robots and send competition instructions (such as switching to manual/automatic control).

Technical Specifications

- Module dimensions: 85 mm x 56 mm x 11.5 mm
- Operating voltage: 6 V - 13 V
- Operating current: 100 mA
- Communication port and protocol: wireless transparent transmission module/serial port/dedicated protocol

ELECTRONICS

MOTORS AND SERVO



Encoder Motors.
Referanced : https://www.creativakids.com/images/makex/tutorialchallenge.pdf

Encoder Motor

Encoder Motors a Motors that has connect with Incremental Encoder so that we can measured the velocity of each motors. It works with NovaPi to serve as the chassis motor for MakeX Challenge.

Features

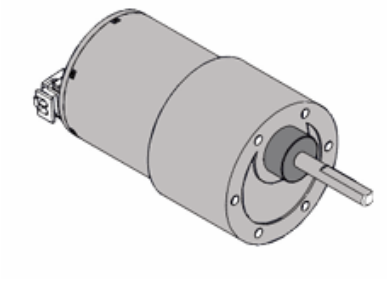
- Sheer metal gear set for excellent durability
- Embedded encoder for precise control of robotic motions
- Mounting holes for bracket-free motor mounting
- Various types of installation holes to facilitate installation
- High torque and power
- Smart motor port

Technical Specifications

- Reduction-gear ratio: 39.43
- Rated voltage: 12 V
- No-load current: 350 mA
- Communication port and protocol: serial port communication

37 DC Motors

37 DC Motors A DC motor is driven by direct current and its rotational speed can be easily controlled by tuning the voltage.
In MakeX Robotics Competition has divided the 37 DC Motors into 2 kind : 200 rpm and 50 rpm DC motors



37 DC Motors.
Referanced : https://www.creativakids.com/images/makex/tutorialchallenge.pdf

Features

Its rotational speed can be changed, but its position and rotational speed cannot be accurately controlled. It can generate strong torque and bear a certain level of overload.

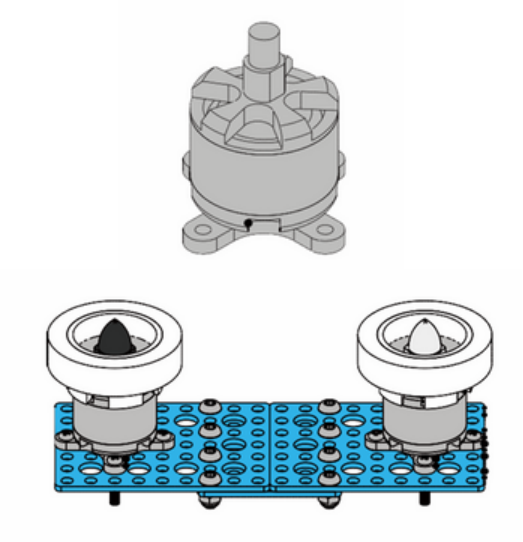
Technical Specifications

- 37 DC motor: 50 rpm :
  1. Rated voltage: 12.0 V (DC)
  2. Temperature range: +25±3°C
  3. Humidity range: 60%±5%
  4. Reduction-gear ratio: 1:90
  5. No-load current: ≤ 80 mA
  6. No-load rotational speed: 50±12% rpm
  7. Rated torque: 2.4 kg.cm
  8. Rated current: ≤ 300 mA
  9. Rated rotational speed: 41±12% rpm
  10. Locked-rotor torque: 9 kg.cm
  11. Locked-rotor current: ≤ 1.8 A

- 37 DC motor: 200 rpm :
  1. Rated voltage: 12.0 V (DC)
  2. Temperature range: +25±3°C
  3. Humidity range: 60%±5%
  4. Reduction-gear ratio: 1:30
  5. No-load current: ≤ 120 mA
  6. No-load rotational speed: 200±13% rpm
  7. Rated torque: 2.0 kg.cm
  8. Rated current: ≤ 450 mA
  9. Rated rotational speed: 160±13% rpm
  10. Locked-rotor torque: 7.5 kg.cm
  11. Locked-rotor current: ≤ 1.8 A

ELECTRONICS

MOTORS AND SERVO



Brushless Motors.
Referanced : https://www.creativakids.com/images/makex/tutorialchallenge.pdf

Brushlees Motors

Brushless Motors is provided by Makeblock exclusively for the launching mechanism. It features a much higher rotational speed than other DC motors to launch “bullets” to accomplish various missions.

Technical Specifications

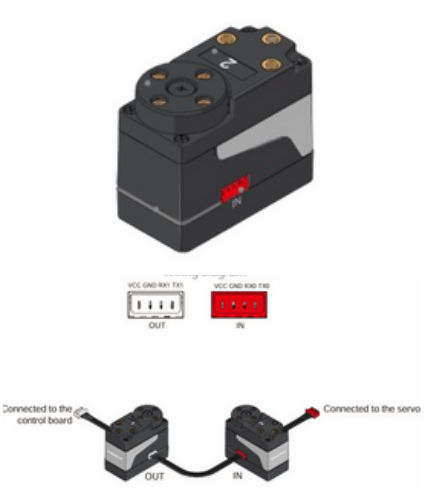
- Rated voltage: 11.1 V
- Rated rotational speed: 7,300 rpm
- Rated current: 10,000 mA MAX

Smart Servo MS-12A

The Smart Servo MS -12A is a next-generation smart servo that Makeblock has independently developed. It focuses on resolving traditional servos' problems, such as complex control, small angular range, inability to perform continuous rotation, and single accessories

Technical Specifications

- Gear: Sheer metal gear (reduction-gear ratio 1:305)
- Velocity: 0.18s/60°, 7.4
- Operating voltage: 6 V - 12.6 V
- Angular resolution: 4096
- Rated torque: 12 kgf.cm
- Locked-rotor current: 2 A



Smart Servo and Wiring Diagram.
Referanced : https://www.creativakids.com/images/makex/tutorialchallenge.pdf

ELECTRONICS

COOLING SYSTEM AND LASER MODULE

Brushless DC Fan.
Referanced : From AEI.th Shop (Shopee)

Heatsink.
Referanced : From AEI.th Shop (Shopee)

Cooling System

Our team have made **The Coling System** by using these 2 device :

- Brushless DC Fan
- Heatsink

Brushless DC Fan

Our team have install the Brushless DC Fan 12V on top of the Computing Board by creating a new Top case with for the Computing Case that has a Brushless DC Fan Holder.

Heatsink

Our team have install the Heatsink at the encoder motors and on top of the chip of the computing board to conduct heat from these devices.

The Cooling System is used for cooling down the important devices so these devices wouldn't get overheating and be broked.

Laser Module

Laser Module is used to help the pilot to aim easier and faster so when we control the robot more efficiency.

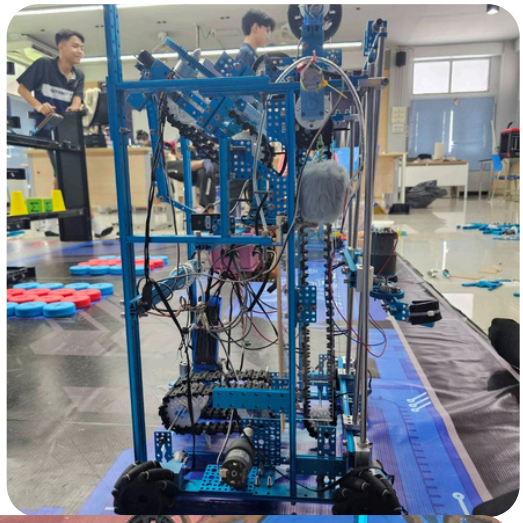
Technical Specifications

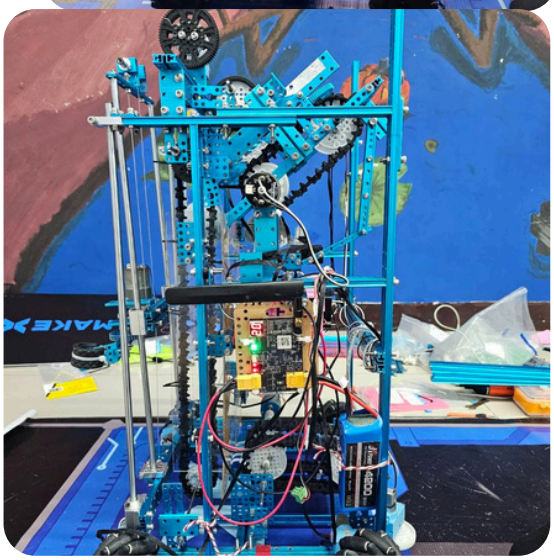
- Output Power : 5mw
- Wavelength : 650nm
- Working Voltage : 3~5V
- Working Temperature : +10 ~ +40C
- Laser Shape Line Focusable : Yes
- Material and Color Metal
- Lens Glass
- Dimensions : 12*12*40mm

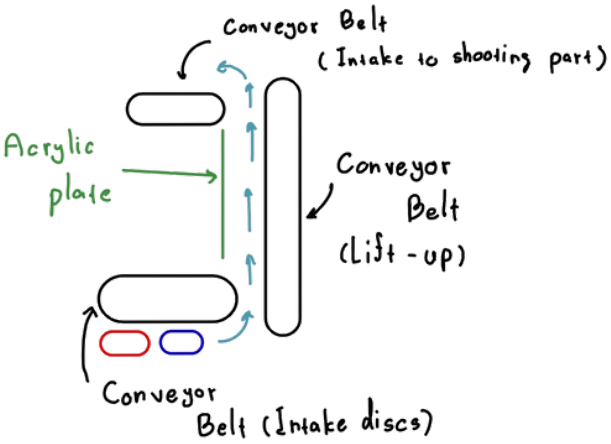
Laser Module.
Referanced : iT.in.Store (Shopee)

MECHANISM

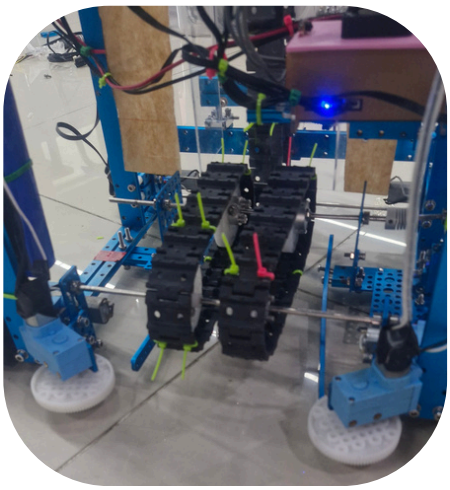
CONVEYOR FEED MECHANISM







Design Sketch
Referanced : GravityShift (Designer)



```
if gamepad.is_key_pressed("N3"):
    entrance_feed.set_reverse(False)
    entrance_feed.on()
    power_expand_board.set_power("DC7", -65) #feeder
    conveyer.set_reverse(False)
    conveyer.on()
elif gamepad.is_key_pressed("N2"):
    entrance_feed.set_reverse(True)
    entrance_feed.on()
    power_expand_board.set_power("DC7", 65) #feeder
    conveyer.set_reverse(True)
    conveyer.on()
elif gamepad.is_key_pressed("N2") or gamepad.is_key_pressed("N3") == False:
    entrance_feed.off()
    feeder.off()
    conveyer.off()
```

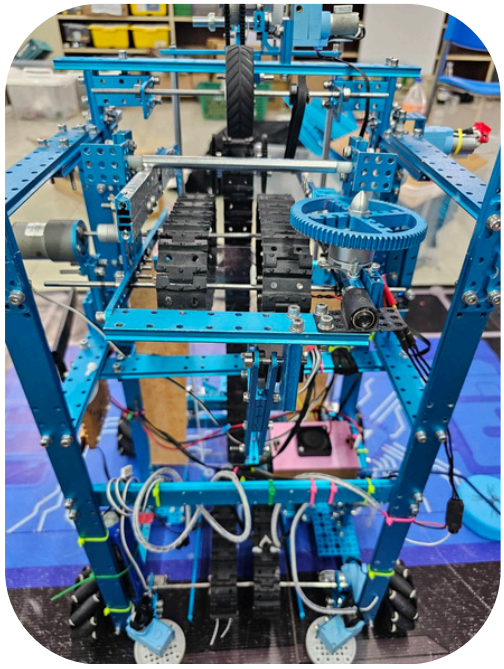
Conveyor Feed Mechanism is used to take all the discs into the robot and transport into the shooting part.

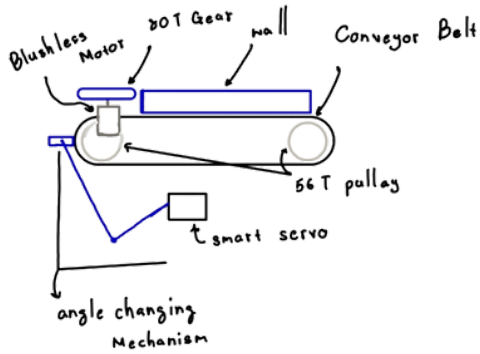
This Conveyor belt part work by changing rotation of the discs that have taken into the robot and bring the disc up forward.

We used the **200 rpm DC Motors** at the Intake part and **encoder motors** that has the high speed so we see the opportunity and benefits, so we can use these motor to drive our lift-up part.

MECHANISM

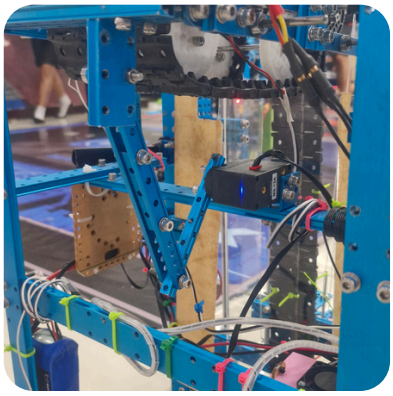
SHOOTING MECHANISM

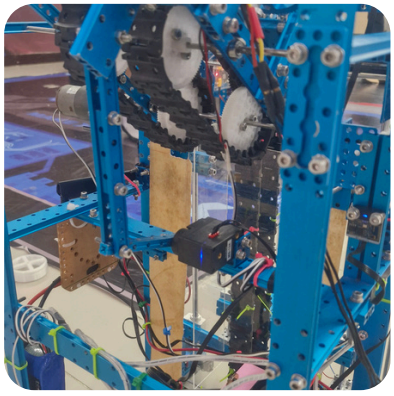


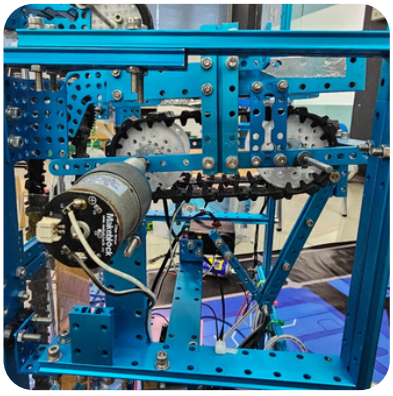


Design Sketch
Referanced : GravityShift (Designer)

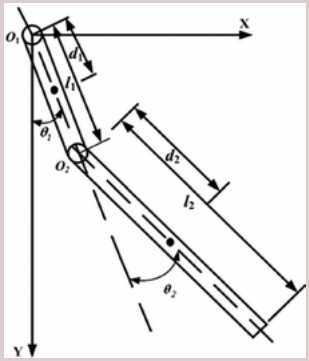
Shooting Mechanism used for shoot the discs out of the robot by using Blushless Motor that has a high speed and force, this machanism refered to ball shooting machine by the distance of the wall and blushless wheel is about 90 milimeters or less than that also we use the laser to help the pilot to aim for pins, smelter holes, reversal flag.





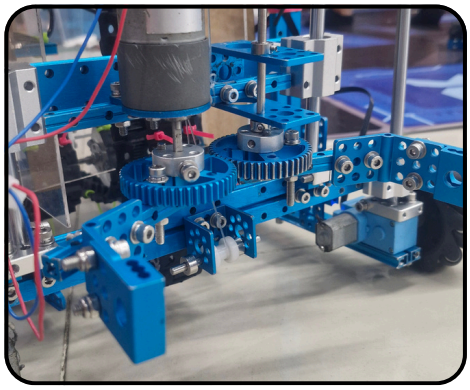


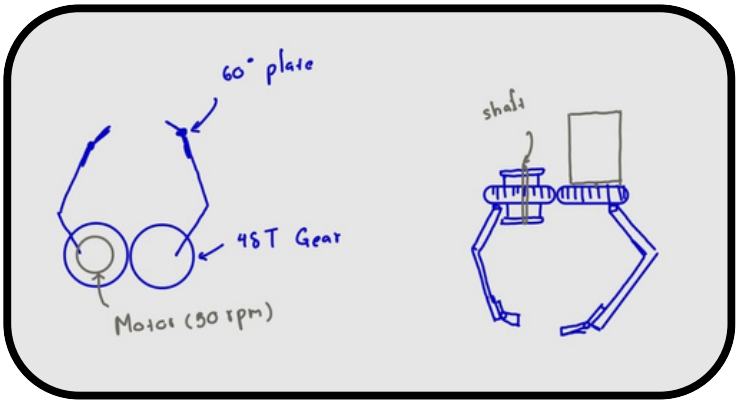
Angle Changing System for shooting Mechanism is for changing angle of shooter for shooting the discs into the smelter and hitting the reversal flag. This Machanism we refered to **two bar linkage mechanism**, we using smart servo as the actuator to control the angle of shooter.



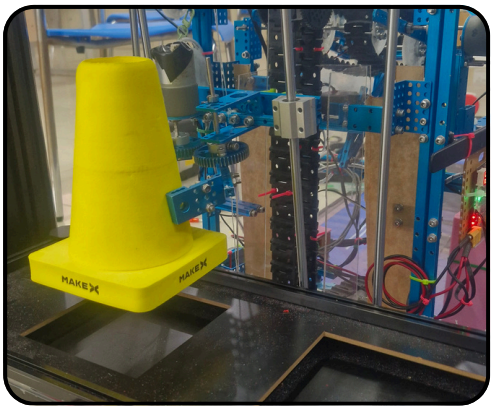
MECHANISM

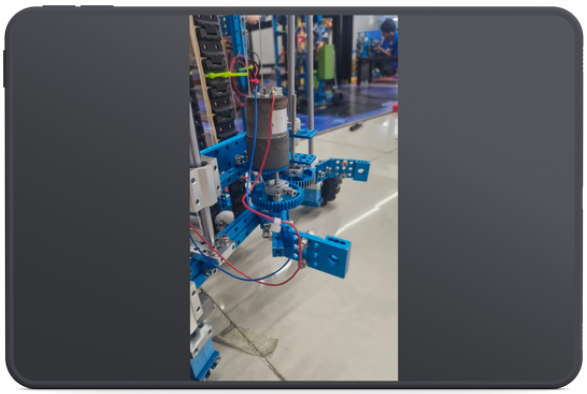
GRIPPER MECHANISM

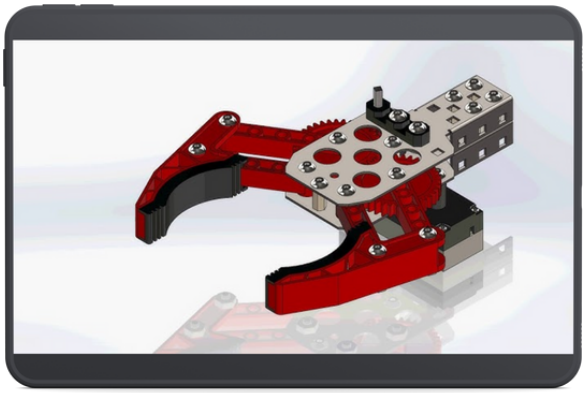




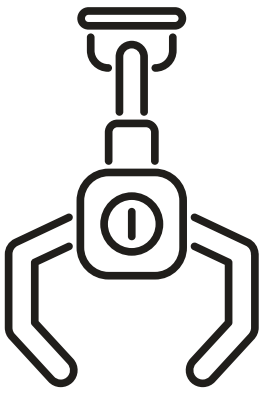
Design Sketch
Referanced : GravityShift (Designer)





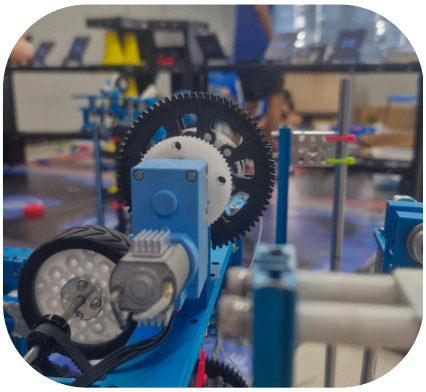


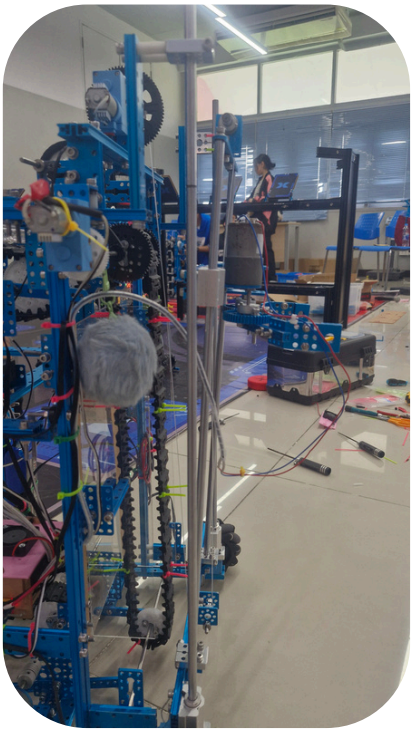
Gripper Mechanism is using for grab the cube and cone, we using the 50 rpm DC motor in this machanism, we refered to mechanical gripper for robot. The gripper will be use together with our lifting mechanism.

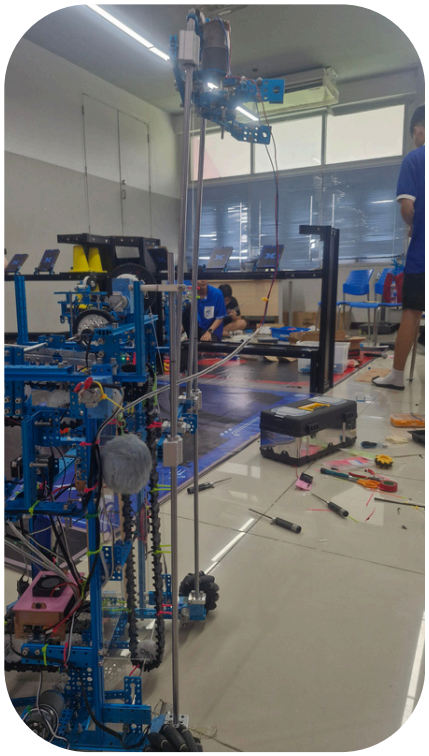


MECHANISM

LIFTING MECHANISM

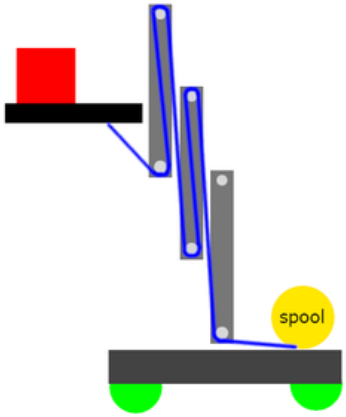






Lifting Mechanism using to bring the gripper up or down so that the pilot can hold the cube or cone to place on the resource tray

The referance of our lifting mechanism we use the **Continuous Rigging mechanism** as the base of our lifting mechanism

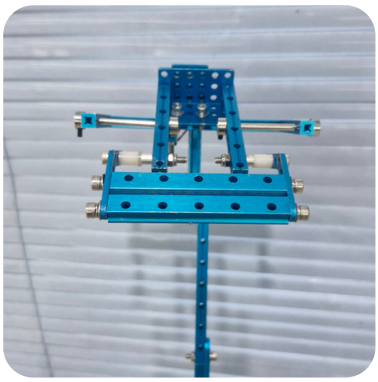


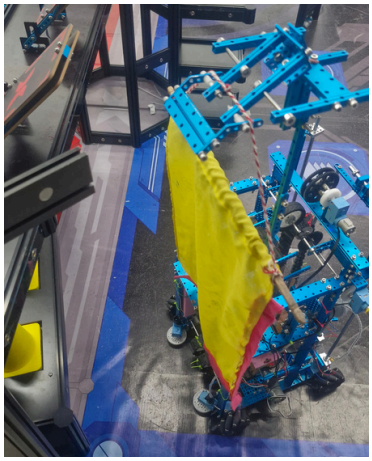
MECHANISM

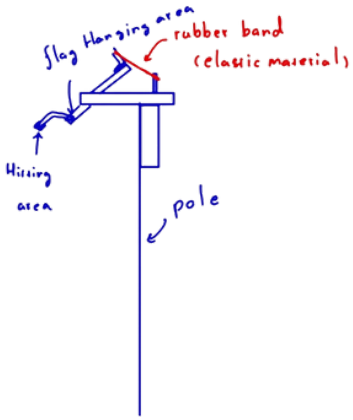
FLAG. HANGING. EQUIPMENT

The Flag Hanging Equipment is use as the accessories to modified the robot in Modification stage so that the pilot can hang our team flag on the flagpole.

Our concept of our hanging equipment is when we want to hang the team flag, we can just walk to the flagpole and push the head of the hanging equipment then the flag would go off the hanging equipment and hang on the flagpole.







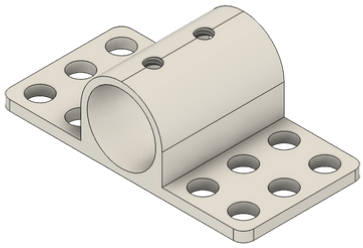
Design Sketch
Referanced : GravityShift (Designer)

MECHANISM

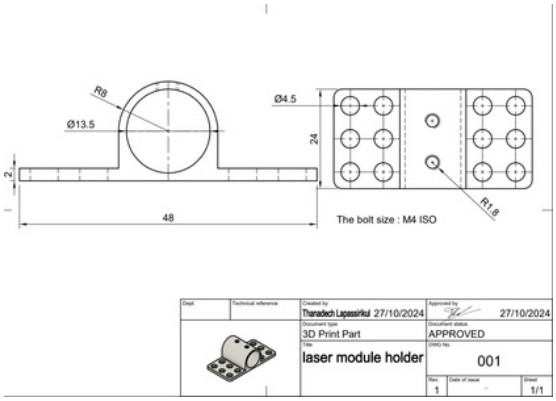
3D PRINT PART

Laser Module Holder

We create a 3D Model for Laser Module Holder for holding and lock the laser Module, so that the laser module can be install to our robot more easier and the laser would be stable to see as it's locked.



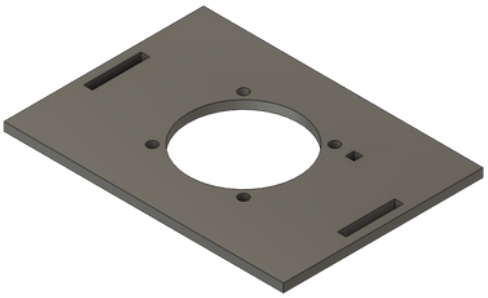
Laser Module Holder.
Referanced : GravityShift



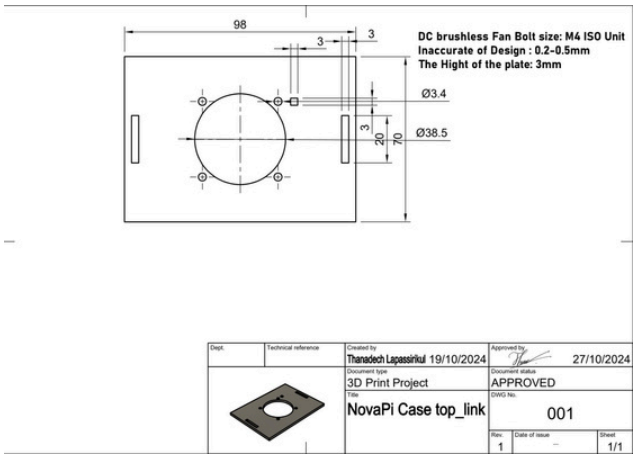
Laser Module Holder Drawing.
Referanced : GravityShift

NovaPi Top Part Case With Fan Holder

We create a new 3D Model of NovaPi Case at the top of the case with the Brushless DC Fan Holder So we can install the Cooling Fan for the NovaPi to prevent the NovaPi gets overheating problem.



NovaPi Top Part Case With Fan Holder.
Referanced : GravityShift



NovaPi Top Part Case With Fan Holder Drawing.
Referanced : GravityShift

Simple Lipo battery enclosure v2.x

a very simple battery enclosure for robots participating in MakeX Challenge robotic competition
This Model we import from the Outsource because we don't have much time left before the Point-Race #3 Competition

Simple Lipo battery enclosure v2.x.
Referanced : https://www.printables.com/model/550473-simple-lipo-battery-enclosure-v2x

PROGRAMMING

OVERALL EXPLANATION

Programming part we choose to program the robot using MicroPython language because the MicroPython is a lean and efficient implementation of the **Python 3** programming language which is optimised to run on microcontrollers. So we thought that when we program the robot it would be easy to analyze, Debuging and has more effective.

The program of our team include:

- PID controller.
- Joysticks movement control.
- Mode changing. (Change betwen shooting mode and gripper mode)
- Closed loop control lifting function.

Our Team's Github Repository

PROGRAMMING

LIBRARY AND VARIABLE EXPLANATION

```
# Import necessary modules
import novapi
import time
import math
from mbuild.encoder_motor import encoder_motor_class
from mbuild import power_expand_board
from mbuild import gamepad
from mbuild.led_matrix import led_matrix_class
from mbuild.smart_camera import smart_camera_class
from mbuild.ranging_sensor import ranging_sensor_class
from mbuild.smartservo import smartservo_class
from mbuild import power_manage_module
import time
```

1-13 we **import every neccessary library** and modules that is required for NovaPi to compile using MicroPython into our program.

15-18 we **define every robot wheels** that are using encoder motor as the actuators

20-23 we **define the variables** that is going to use in the calculation , these variable are define as a globle variable so in every class and function can use these variable.

```
left_forward_wheel = encoder_motor_class("M2", "INDEX1")
right_forward_wheel = encoder_motor_class("M3", "INDEX1")
left_back_wheel = encoder_motor_class("M5", "INDEX1")
right_back_wheel = encoder_motor_class("M6", "INDEX1")

MAX_SPEED = 255
SPEED_MULTIPLIER = 2.1
PID_SPEED_MULTIPLIER = 0.6
BL_POWER = 100
```

```
# Instantiate DC motors
lift = encoder_motor_class("M4", "INDEX1") # using encoder for spacific position of lift functions
cooling = dc_motor("DC2")
gripper1 = dc_motor("DC1")
conveyer = dc_motor("DC5")
entrance_feed = dc_motor("DC6")
feeder = dc_motor("DC7")
bl_1 = brushless_motor("BL1")
bl_2 = brushless_motor("BL2")
shooter = smartservo_class("M1", "INDEX1") # only for angles
laser = dc_motor("DC8")
front_input = dc_motor("DC3")
```

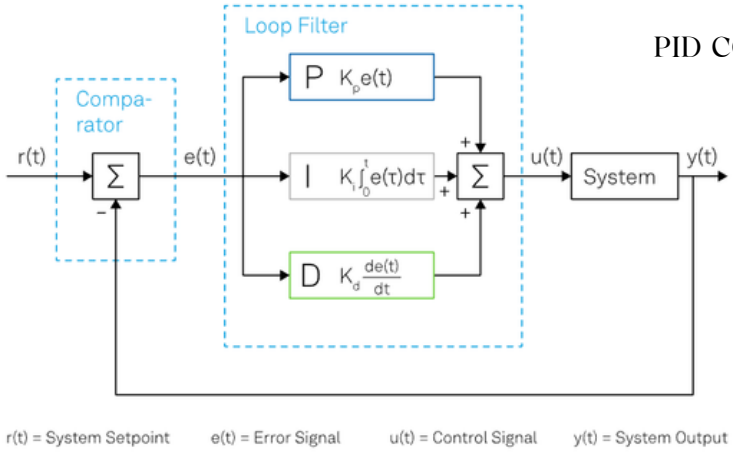
190-227 we create the Class dc_motor for instantiate what dc motors is used for and to control motors

326 we **instantiate the encoder motor** for lifting function for the specific position using closed loop control

327-336 we **instantiate every DC motor** in every part of the robot

PROGRAMMING

PID CONTROLLER AND MOTORS



The PID Controller Explanation

PID CONTROLLER EXPLANATION

A PID controller is an instrument that receives input data from sensors, calculates the difference between the actual value and the desired setpoint, and adjusts outputs to control variables such as temperature, flow rate, speed, pressure, and voltage. It does this through three mechanisms: proportional control, which reacts to current error; integral control, which addresses accumulated past errors; and derivative control, which predicts future errors. The PID controller sums those three components to compute the output. This architecture allows PID controllers to efficiently maintain process control and system stability. Before we start to define the parameters of a PID controller, let's discuss what a closed loop system is and some of the terminologies associated with it.

26-61 Class PID use to calculate the differance betwen the actual value from encoder and the desired setpoint then adjusts the output value to meet the desired setpoint real-time because the actual value can be change anytime but the calculation is always working while the program is running.

```
26  class PID:
27      def __init__(self, Kp,  Ki, Kd, setpoint=0):
28          self.Kp = Kp  # Proportional gain
29          self.Ki = Ki  # Integral gain
30          self.Kd = Kd  # Derivative gain
31          self.setpoint = setpoint  # Desired value (target)
32          self.integral = 0  # Sum of errors over time
33          self.previous_error = 0  # Previous error (used for derivative)
34
35      def update(self, current_value):
36          # Calculate the error (setpoint - current value)
37          error = self.setpoint - current_value
38
39          # Proportional term
40          P = self.Kp * error
41
42          # Integral term
43          self.integral += error
44          I = self.Ki * self.integral
45
46          # Derivative term
47          derivative = error - self.previous_error
48          D = self.Kd * derivative
49
50          # Calculate the output
51          output = P + I + D
52
53          # Save the current error for the next update
54          self.previous_error = error
55
56          return output
57
58      def set_setpoint(self, setpoint):
59          self.setpoint = setpoint
60          self.integral = 0  # Reset the integral to avoid wind-up
61          self.previous_error = 0  # Reset previous error to avoid a large derivative spike
```

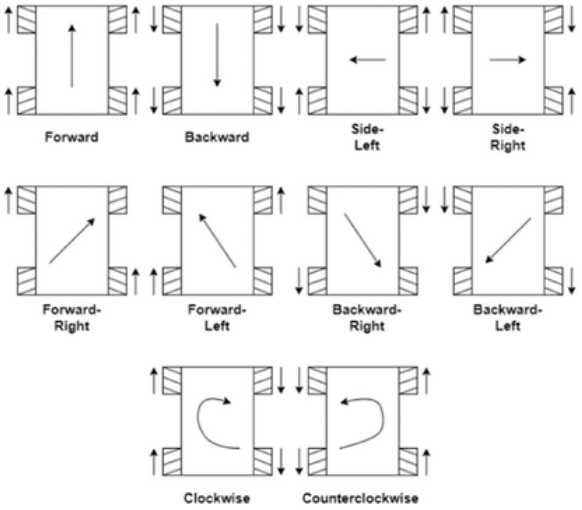
```
64  class motors:
65
66      def drive(lf: int, lb: int, rf: int, rb: int):
67          left_back_wheel.set_speed(lb) # left back :DDDDD
68          right_back_wheel.set_speed(-rb)  # RIGHT BACK
69          right_forward_wheel.set_speed(-(rf))      # RIGHT FORWARD
70          left_forward_wheel.set_speed(lf)            # LEFT BACK
71
72      def stop():
73          motors.drive(0, 0, 0, 0)
```

64-73 Class motors use to control encoder motors that we use as the robot wheels, the drive function is use to set the speed of each motors, stop function use to set the speed of the motors to 0 rpm so that the robot will stop moving.

PROGRAMMING

UTILS AND HOLONOMIC

DRIVE SYSTEM



Mecanum wheels movement direction

HOLONOMIC DRIVE SYSTEM EXPLANATION

A holonomic drive system refers to a drive system that allows a robot to move in any direction, independent of its orientation. Holonomic systems are important because they allow for efficient mathematical modeling and control of physical systems that have constraints on their motion.

<u>The Calculation of each wheel's velocity</u> ;
Vx : The desired x-axis velocity
Vy : The desired y-axis velocity
ωL : The desired angular velocity

$$V_{FL} = V_x + V_y + \omega_L$$

$$V_{FR} = -V_x + V_y - \omega_L$$

$$V_{BL} = -V_x + V_y + \omega_L$$

$$V_{BR} = V_x + V_y - \omega_L$$

The wheel's velocity calculation formula

```
75    class util:
76        def restrict(val, minimum, maximum):
77            return max(min(val, maximum), minimum)
```

```
134           # Velocity
135           vFL = util.restrict(vFL, -MAX_SPEED, MAX_SPEED)
136           vFR = util.restrict(vFR, -MAX_SPEED, MAX_SPEED)
137           vBL = util.restrict(vBL, -MAX_SPEED, MAX_SPEED)
138           vBR = util.restrict(vBR, -MAX_SPEED, MAX_SPEED)
```

75-77 Class util is use to create a method to restrict the velocity of each wheels

134-138 These are the variable that the velocity is restricted by the **class util**, <u>these variable is defined in class holonomic.</u>

79-94 Class holonomic use to create a method to control the movement, the rotation of each motors, tune the motor's velocity to match the robot weight in each wheels by creating a variables to set the Kp, Ki and Kd value

96-140 drive function is created in the class holonomic to calculate the velocity of each wheels, the condition that make the program update the value of each wheels to the calculation

142-158 is a function that use to control the movement of the robot in Automatic stage.

```
79    class holonomic:
80
81        pids = {
82            "lf": PID(Kp=1, Ki=0, Kd=0),
83            "lb": PID(Kp=1, Ki=0, Kd=0),
84            "rf": PID(Kp=0.9, Ki=0, Kd=0),
85            "rb": PID(Kp=0.9, Ki=0, Kd=0),
86        }
87
88        # motor tune
89        tune = {
90            "fl": 1,
91            "fr": 0.7,
92            "bl": 1,
93            "br": 0.5,
94        }
```

```
142       def move_forward(power):
143           holonomic.drive(0, power, 0, pid=True)
144
145       def move_backward(power):
146           holonomic.drive(0, -power, 0, pid=True)
147
148       def slide_right(power):
149           holonomic.drive(power, 0, 0, pid=True)
150
151       def slide_left(power):
152           holonomic.drive(-power, 0, 0, pid=True)
153
154       def turn_right(power):
155           holonomic.drive(0, 0, power, pid=True)
156
157       def turn_left(power):
158           holonomic.drive(0, 0, -power, pid=True)
```

```
96        def drive(vx, vy, wL, deadzone=5, pid=False):
97            global SPEED_MULTIPLIER, PID_SPEED_MULTIPLIER
98            if math.fabs(vx) < math.fabs(deadzone):
99                vx = 0
100           if math.fabs(vy) < math.fabs(deadzone):
101               vy = 0
102           if math.fabs(wL) < math.fabs(deadzone):
103               wL = 0
104
105           # Ensure the correct speed multiplier
106           multiplier = PID_SPEED_MULTIPLIER if pid else SPEED_MULTIPLIER
107
108           # Calculation for the wheel speed
109           vFL = (vx + (vy * 1.2) + wL) * multiplier
110           vFR = (-(vx) + (vy * 1.2) - wL) * multiplier
111           vBL = (-(vx) + (vy * 1.2) + wL) * multiplier
112           vBR = (vx + (vy * 1.2) - wL) * multiplier
113
114           # Sliding check to not interfere with the normal movement, incase of tuning specific power
115           if math.fabs(vx) > math.fabs(vy) and vx > 0:
116               vFL *= holonomic.tune["fl"] # หน้าซ้าย
117               vFL *= holonomic.tune["fr"] # หน้าขวา
118               vBL *= holonomic.tune["bl"] # หลังซ้าย
119               vBR *= holonomic.tune["br"] # หลังขวา
120           if pid:
121               # Left Forward
122               holonomic.pids["lf"].set_setpoint(vFL)
123               vFL = holonomic.pids["lf"].update(-left_forward_wheel.get_value("speed"))
124               # Left Back
125               holonomic.pids["lb"].set_setpoint(vBL)
126               vBL = holonomic.pids["lb"].update(-left_back_wheel.get_value("speed"))
127               # Right Forward
128               holonomic.pids["rf"].set_setpoint(vFR)
129               vFR = holonomic.pids["rf"].update(right_forward_wheel.get_value("speed"))
130               # Right Back
131               holonomic.pids["rb"].set_setpoint(vBR)
132               vBR = holonomic.pids["rb"].update(right_back_wheel.get_value("speed"))
```

PROGRAMMING

AUTOSTAGE PROGRAM

```
160  class Auto:
161      def right():
162          pass
163
164      def left():
165          entrance_feed.set_reverse(True)
166          entrance_feed.on()
167          front_input.set_reverse(True)
168          front_input.on()
169          power_expand_board.set_power("DC7", 16)
170          holonomic.move_forward(60)
171          time.sleep(2.2)
172          motors.stop()
173          holonomic.turn_left(60)
174          time.sleep(2.1)
175          motors.stop()
176          holonomic.slide_left(45)
177          time.sleep(2)
178          motors.stop()
179          holonomic.move_forward(45)
180          time.sleep(2.5)
181          motors.stop()
182          holonomic.turn_right(60)
183          time.sleep(2.1)
184          motors.stop()
185          time.sleep(50000)
```

160-185 These are the Autostage Program use to make the robot move to take the discs for doing mission in Manual Stage.

PROGRAMMING

JOYSTICKS CONTROL AND MODE CHANGING

MODE CHANGING AND JOYSTICKS CONTROL

231-258 Class runtime we create a method to change between shoot_mode (default) and gripper_mode, also in this class has method of joysticks control, we has seperate the control into 2 function is move_1 and move_2

move_1 : control in default mode

move_2 : reverse control for gripper_mode

260-303 Class Shoot_mode we create a class to define each button on Bluetooth controller to control the feed system, shooting system and shooter angular changing.

Control button;

- N2 : move feed system forward
- N3 : move feed system backward
- R1 : control shooting system
- + : turn on laser
- ≡ : turn off laser
- Up : increase shooter angular system
- Down : decrease shooter angular system

```
class gripper_mode:
    # Method to control various robot functions based on button inputs
    def control_button():
        if gamepad.is_key_pressed("N2"):
            lift.set_power(-90)
        elif gamepad.is_key_pressed("N3"):
            lift.set_power(45)
        else:
            lift.set_speed(0)
        if gamepad.is_key_pressed("N1"):
            gripper1.set_reverse(True)
            gripper1.on()

        elif gamepad.is_key_pressed("N4"):
            gripper1.set_reverse(False)
            gripper1.on()
        else:
            gripper1.off()
```

305-322 Class Gripper_mode we create a class to define each button on Bluetooth controller to control the lifting system and Gripper system.

Control button;

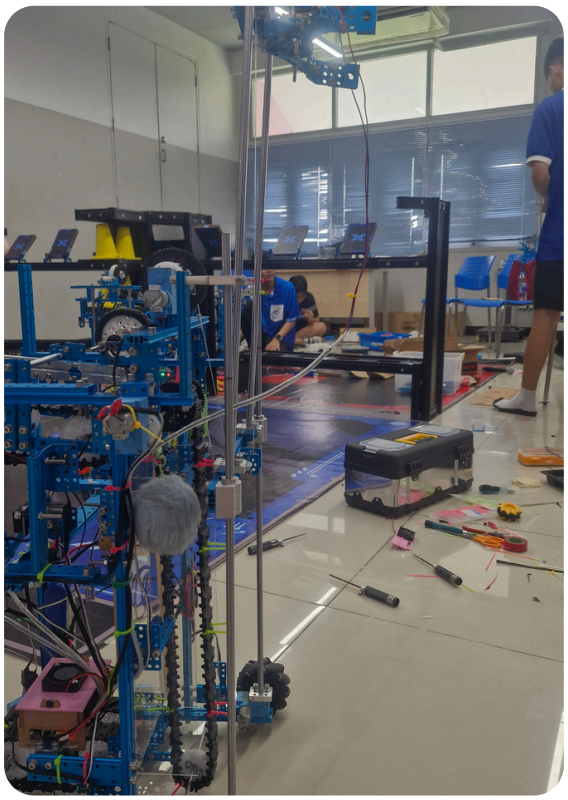
- Up : move lifting system forward
- Down : move lifting system backward
- N1 : make the gripper hold items
- N4 : make the gripper release items

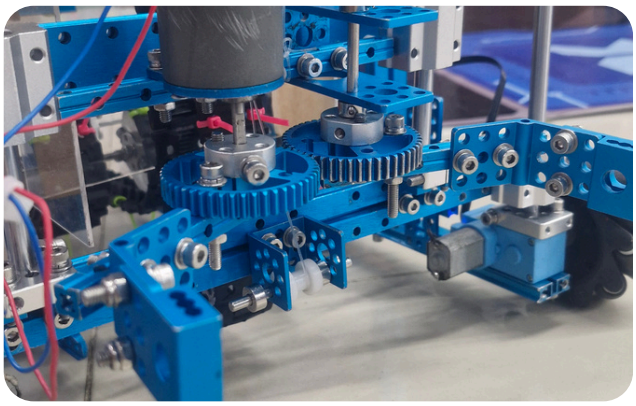
338-352 Condition of Power management the condition that make the robot to run the auto program in Automatic Stage then if it's a Manual or Final Stage the the robot would run the Manual control which is a shoot_mode or gripper_mode where we can pressed L2 and R2 button in the same time to change between these 2 mode.

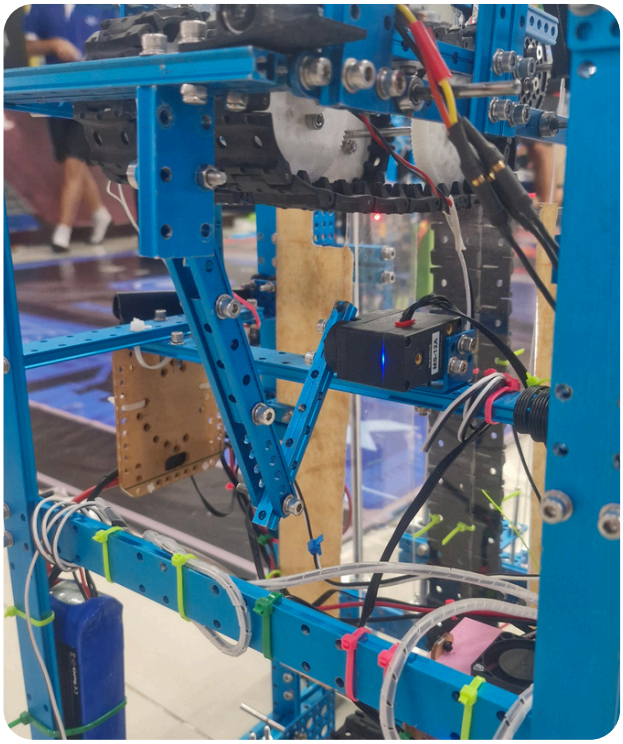
```
while True:
    cooling.set_reverse(True)
    cooling.on()
    if power_manage_module.is_auto_mode():
        Auto.left()
    else:
        if gamepad.is_key_pressed("L2") and gamepad.is_key_pressed("R2"):
            runtime.change_mode()
        else:
            if runtime.CTRL_MODE == 0:
                shoot_mode.control_button()
                runtime.move_1()
            else:
                gripper_mode.control_button()
                runtime.move_2()
```

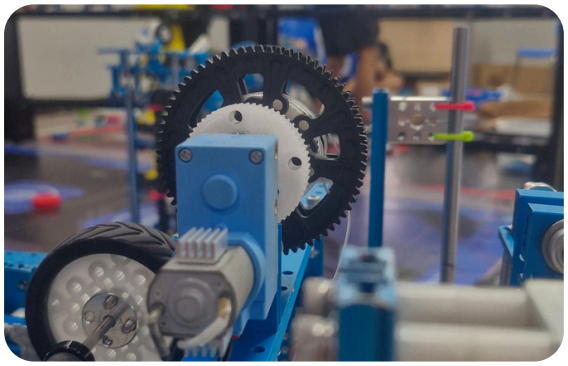
GALLERY

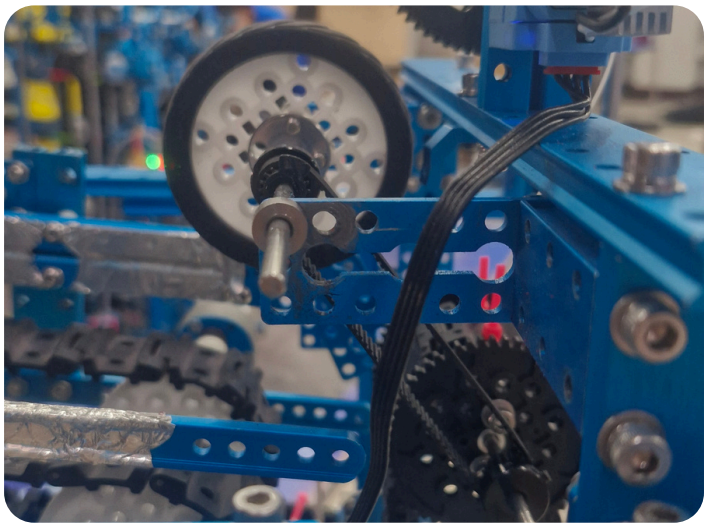
GALLERY

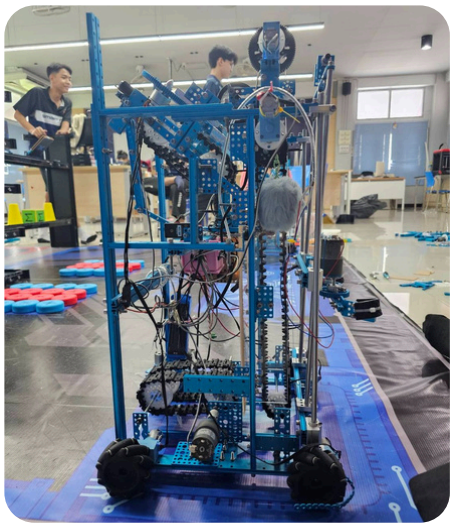












MAKEX

Suksanareewittaya School

Smart Gen Robotics Teams

ENGINEERING BOOK

MakeX Thailand Championship Robotics Competition 2024