

& TECNOLOGIA ENGENHARIA

Apresentado pela equipe:

ALPHA TBR

TBR TORNEIO BRASIL DE ROBOTICA

Firjan SENAI SESI

SUMÁRIO

INFORMAÇÕES DO TIME ALPHA TBR

NOME DA EQUIPE:
ALPHA TBR

CATEGORIA:
High

TÉCNICO/MENTOR:
Marco Antonio, Fabiane Martins e Paulo Vitor.

INTEGRANTES:
Ana Luíza, Flora Maria, Gabriel Paiva, João Pedro, Karoline Balbino, Marina Biscaia, Marcella Vianna, Matheus Cardoso, Natan Santiago e Pedro Douglas

PATROCINADORES:
Firjan SENAI SESI, Calara Software, Elétrica Raiz, Ferazi, Meninas Digitais, Manitshow, Socorrista Polo Curicica e Bombeiro Civil Polo Curicica.

REDES SOCIAIS:
Instagram: @alphatbr

ORGANIZAÇÃO DA NOSSA EQUIPE



MARCO ANTONIO

Técnico

Professor de Física no SESI e técnico da equipe. Responsável pela **coordenação geral** da equipe, fornecendo suporte na **construção do robô** e ajudando em questões **técnicas do projeto.**



FABIANE MARTINS

Técnica

Professora de Química no SESI e técnica da equipe. Encarregada da **gestão e liderança**, estabelecendo metas, mantendo a **motivação da equipe** e oferecendo suporte em aspectos **técnicos do projeto.**



PAULO VITOR

Mentor

Ex-membro da equipe, contribui para os **processos internos**, auxiliando no desenvolvimento de documentações, **definição de metas** e otimização das **tarefas de organização.**

TBR ALPHA

EQUIPE

MÉRITO CIENTÍFICO

- Ana Luíza
- Flora Maria
- João Pedro

TECNOLOGIA E ENGENHARIA

- Gabriel Paiva
- Matheus Cardoso
- Pedro Douglas
- Natan Santiago

ORGANIZAÇÃO E MÉTODO

- Karoline Balbino
- Marcella Viana
- Marina Biscaia

GERENCIAMENTO

- Fabiane Martins
- Marco Antonio
- Paulo vitor

OS MEMBROS DO NOSSO TIME

ORGANIZAÇÃO E MÉTODO (OM)



KAROLINE BALBINO

16 anos;
Técnico em PCP;
1ª temporada no TBR.



MARCELLA VIANA

16 anos;
Técnico em logística;
1ª temporada no TBR.



MARINA BISCAIA

17 anos;
Técnico em logística;
1ª temporada no TBR.

MÉRITO CIENTÍFICO (MC)



ANA LUÍZA

17 anos;
Técnico em PCP;
1ª temporada no TBR.



FLORA MARIA

17 anos;
Técnico em logística;
3ª temporada no TBR.



JOÃO PEDRO

17 anos;
Técnico em logística;
2ª temporada no TBR.

TECNOLOGIA E ENGENHARIA (TE)



GABRIEL PAIVA

18 anos;
Técnico em eletrotécnica;
1ª temporada no TBR.



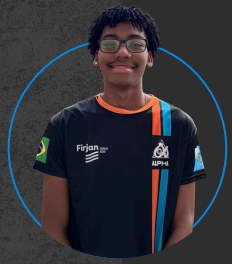
MATHEUS CARDOSO

17 anos;
Técnico em logística
2ª temporada no TBR



PEDRO DOUGLAS

16 anos;
Técnico em logística;
2ª temporada no TBR.



NATAN SANTIAGO

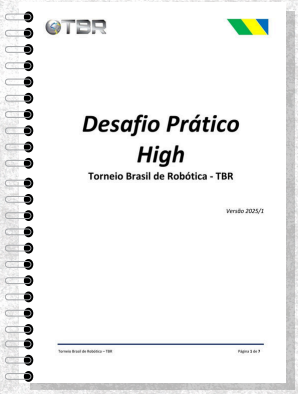
17 anos;
Técnico em eletrotécnica;
2ª temporada no TBR.

IDEAÇÃO DO ROBÔ

TIMELINE DO PROCESSO

LEITURA E ANÁLISE DOS MANUAIS

Com o objetivo de compreender melhor a nova temporada, iniciamos a leitura e análise dos manuais, com destaque para o “Vade Mecum”, documento que reúne de forma geral as normas e diretrizes do TBR, e para o “Desafio Prático”, que apresenta os critérios e exigências específicos dessa etapa da competição.



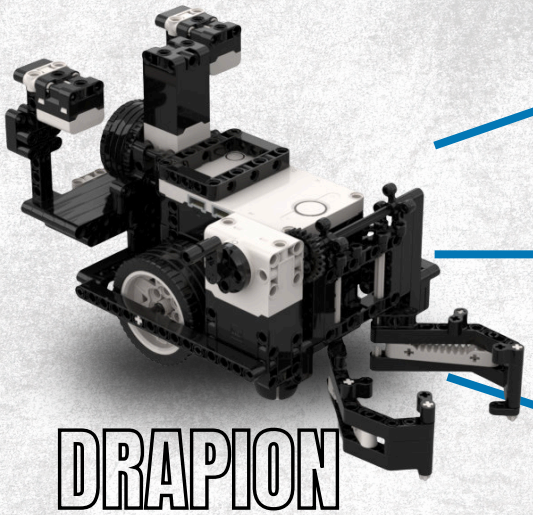


DESAFIO PRÁTICO



Mesa da temporada

Para a análise dos manuais, concentramos nossa atenção no "Desafio Prático" com o objetivo de compreender todas as regras e conhecer a mesa da temporada. A partir disso, começamos a desenvolver possíveis estratégias e estabelecer metas que contribuíssem para o melhor desempenho do nosso robô na mesa.



DRAPION

COMPACTO

Design simples e eficiente.

VERSÁTIL

Realizar duas missões simultaneamente.

CONSTANTE

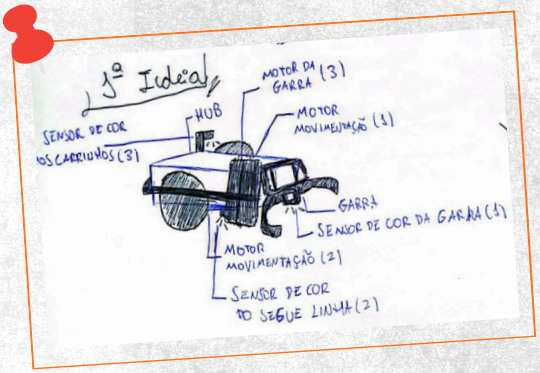
Desvio pequeno nas pontuações da mesa

IDEAÇÃO DO ROBÔ
TIMELINE DO PROCESSO

BRAINSTORMING

Iniciamos uma nova etapa no desenvolvimento do robô, focada no brainstorming de possíveis ideias a serem implementadas. Durante as discussões em grupo, organizamos cada proposta com base em critérios como a quantidade de motores e sensores necessários, os mecanismos a serem utilizados e a estratégia prevista para seu funcionamento.

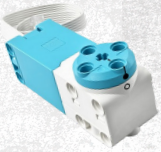
1ª IDEIA



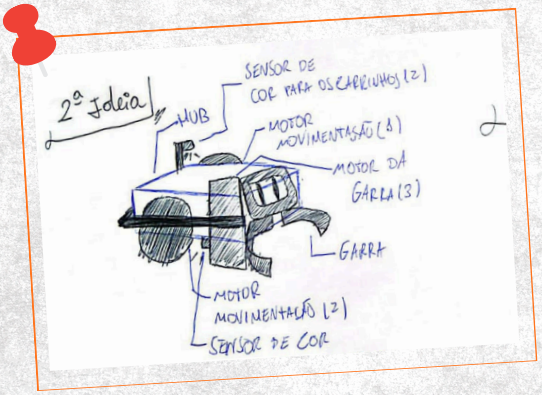
A primeira ideia se baseia em um robô com um mecanismo: o "grab and lift"; ele consegue tanto pegar as caixas através da garra, quanto levá-las com apenas um motor. Além disso, com um sensor de cor localizado atrás do robô, a própria garra empurraria os carrinhos até a sua cor destinada.


X 1


X 2


X 2

2ª IDEIA



A segunda ideia se difere da primeira por apenas um sensor de cor, onde antes era usado na garra para detecção das caixas. Entretanto, as caixas já tem posições específicas na mesa, o que tornaria a utilização desse sensor não muito inteligente. Sem esse sensor, sobraria ainda mais um espaço no hub.


X 1

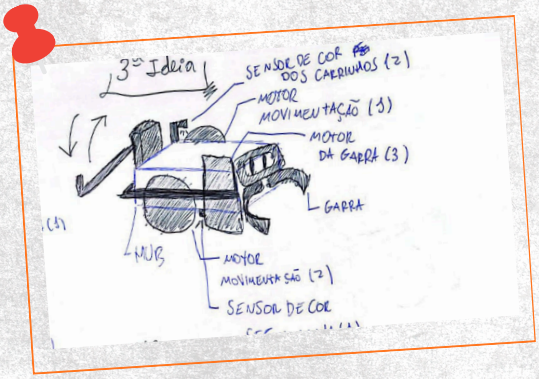

X 1


X 2

IDEAÇÃO DO ROBÔ
TIMELINE DO PROCESSO

BRAINSTORMING

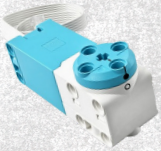
3ª IDEIA



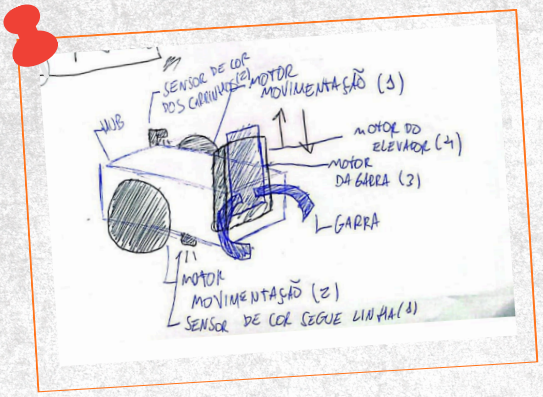
A terceira ideia já busca trazer outro mecanismo, com o intuito de que o robô consiga fazer duas missões simultaneamente. Para isso, uma espécie de cancela é adicionada na parte de trás do robô, para que ele consiga empurrar os carrinhos, e ao mesmo tempo a garra pode segurar a caixa das sementes.


X 1


X 2


X 3

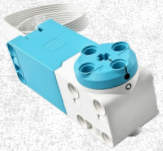
4ª IDEIA



A quarta ideia busca substituir o movimento de levantar do mecanismo "grab and lift". Para isso, uma espécie de elevador seria adicionado, fazendo com que a caixa das sementes fosse elevada apenas verticalmente, deixando o sistema mais estável e seu funcionamento mais simples e eficaz.


X 2


X 2


X 2

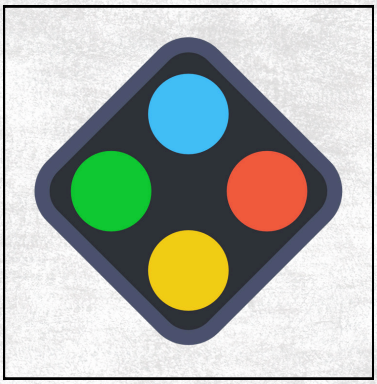
PROTOTIPAGEM DO ROBÔ

TIMELINE DO PROCESSO

Nesta fase, iniciamos os testes com as ideias que a equipe considerou mais urgentes e impactantes para o desempenho do robô. São soluções que apresentam baixa complexidade de construção, mas alto potencial de eficiência na realização das missões na mesa. Para isso, são criados "protótipos sujos", ou seja, versões simples e improvisadas de cada mecanismo, com o objetivo de avaliar sua eficácia na prática.

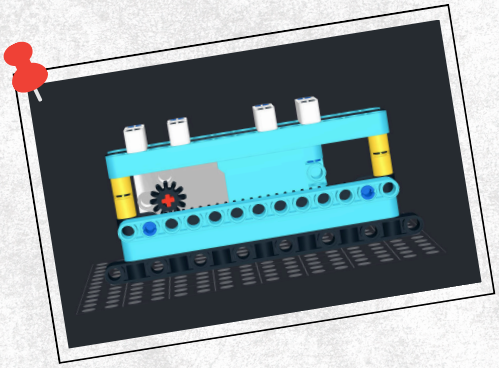
TESTES E CÁLCULOS DO ROBÔ

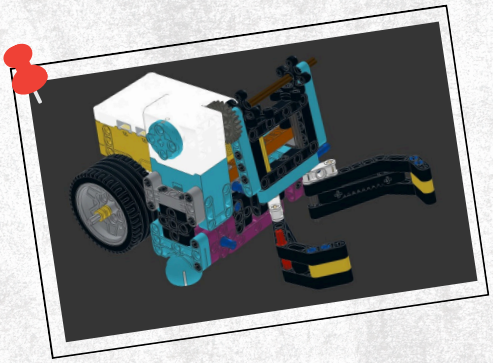
Foram criados diversos protótipos que acompanharam a evolução do robô. Além disso, foram feitos alguns cálculos para garantir a eficiência dos mecanismos.

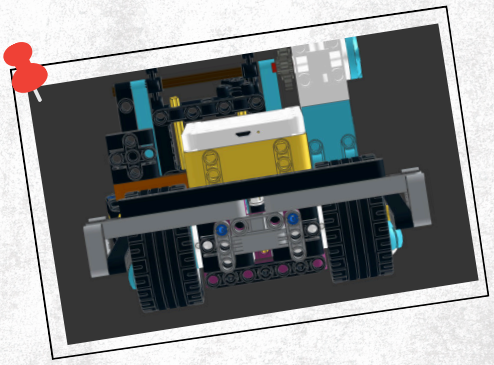


MODELAGEM NO STUDIO 2.0

Além da modelagem utilizada ao longo da evolução do robô definitivo, também aplicamos esse processo em alguns protótipos mais complexos e em outros cuja eficiência foi tão alta que o Studio 2.0 foi empregado para realizar ajustes finos.







PROTOTIPAGEM DO ROBÔ

TIMELINE DO PROCESSO

SISTEMA DE ELEVAÇÃO

Esse mecanismo é composto por um motor acoplado a uma pequena engrenagem, que se mantém em contato com uma cremalheira. À cremalheira, pode-se acoplar uma base, permitindo a instalação de outros sistemas sobre ela, possibilitando sua elevação. Após alguns testes, identificamos pontos positivos e negativos em seu funcionamento.

BAIXA COMPLEXIDADE

Devido a seu funcionamento simples, não apresentou muitas dificuldades em relação a sua montagem para os testes.



INSTABILIDADE

Durante os testes o sistema apresentou uma certa instabilidade na base que ficava conectada a cremalheira, o que poderia ocasionar em um mau funcionamento durante as missões da mesa.

ESPAÇO BEM LIMITADO

O espaço que se pode conectar na cremalheira é pouco, o que dificulta fixar um motor para acionar a garra.

PROTOTIPAGEM DO ROBÔ

TIMELINE DO PROCESSO

GRAB AND LIFTING

Esse mecanismo já havia sido utilizado pela nossa equipe na temporada passada, apresentando um excelente desempenho na execução das missões. Por isso, optamos por usá-lo novamente nesta temporada, realizando apenas alguns ajustes na fase de prototipagem para adaptá-lo ao robô atual.

VERSÁTIL

O mecanismo faz dois movimentos e utiliza somente um motor, o que o tornou prioridade para utilização no robô.

EFICIENTE

O sistema apresentou ótimo desempenho na realização das missões de "reflorestamento".



CONTATO ENTRE ENGRENAGENS

Durante os testes o sistema apresentou algumas problemas, como o mau contato na relação das engrenagens no eixo do motor.

PROTOTIPAGEM DO ROBÔ

TIMELINE DO PROCESSO

SISTEMA DE CANCELA MÓVEL V1 E V2

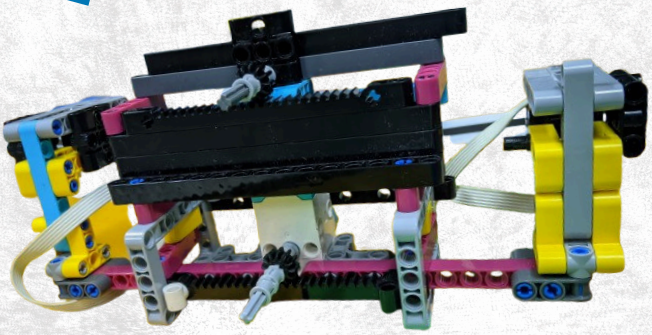
Para garantir a versatilidade do robô na execução simultânea de duas missões, foram desenvolvidos protótipos de um mecanismo inovador voltado à missão "Expresso Ecológico". Na primeira versão, um motor acoplado a uma pequena engrenagem movimenta-se ao longo de uma estrutura por meio de uma cremalheira. Já na segunda versão, um motor fixo aciona uma roda que, ao girar, desloca uma estrutura responsável por empurrar os carrinhos.

FUNCIONAL

Apesar de seu funcionamento complexo, ele se apresentou eficiente na realização das missões.

MECANISMO GRANDE

O sistema apesar de ser funcional era muito grande, o que impedia o robô de estar totalmente dentro da área de estacionamento.



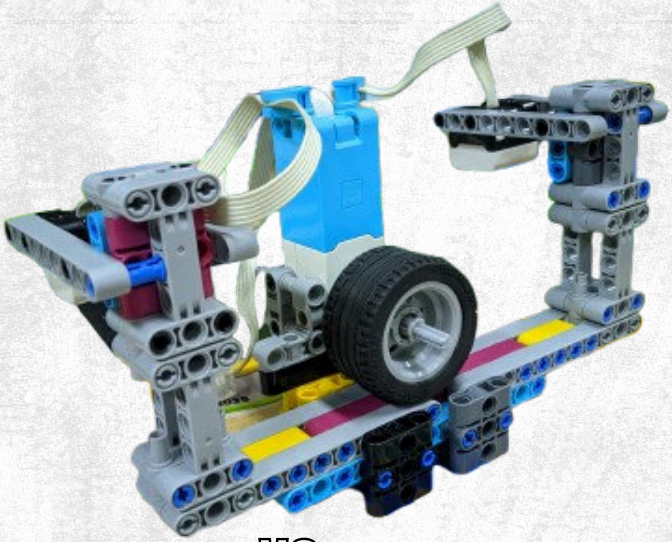
V1

MECANISMO PEQUENO

Em comparação com a primeira versão, ele não impede que o robô esteja totalmente dentro da área de estacionamento.

EFICIENTE

O mecanismo apresentou um melhor desempenho até do que a primeira versão, além de seu funcionamento mais simples.



V2

PROTOTIPAGEM DO ROBÔ

TIMELINE DO PROCESSO

BASE DO ROBÔ V1 E V2

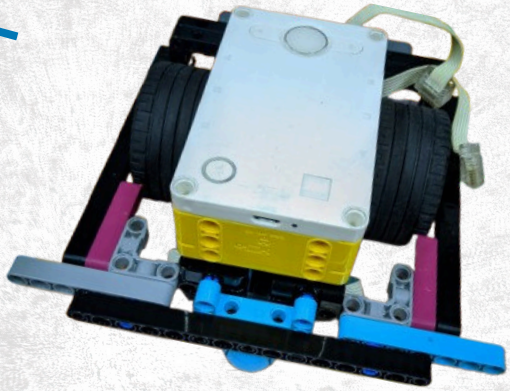
Outra parte fundamental do robô é a sua base, pois é nela que está localizado o hub e onde todos os demais mecanismos são fixados. Na primeira versão, foi utilizada uma estrutura simples, com apenas uma bilha, o que resultou em instabilidade. Além disso, algumas partes não estavam devidamente fixadas, devido à falta de suporte adequado na base. Para corrigir esses problemas, a segunda versão foi projetada com uma estrutura mais robusta e com múltiplos pontos de fixação, proporcionando maior estabilidade e eficiência na integração dos mecanismos.

SIMPLES

A sua montagem é de baixa complexidade, o que o tornou um protótipo rápido e prático de se fazer.

PARTES SOLTAS

A base apresentava partes soltas, devido a falta de pontos de fixação.



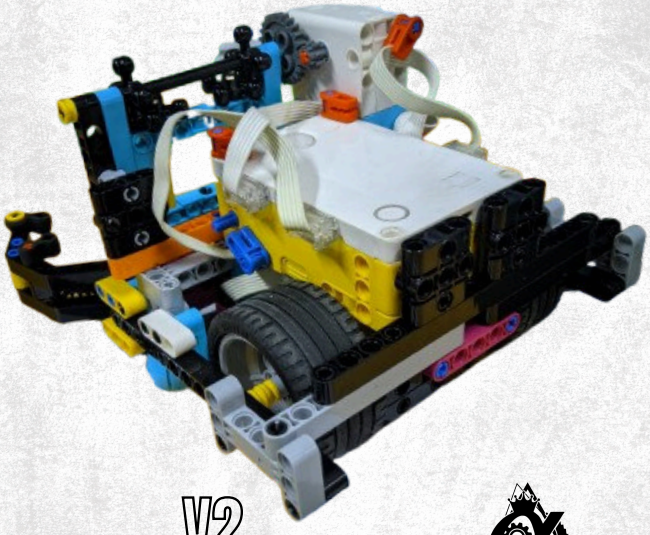
V1

ROBUSTA

A base foi planejada para ter as rodas tampadas e o eixo delas fixadas na base, além de ter o espaçamento mínimo entre elas, garantindo que o robô tenha o menor tamanho possível.

ESTRUTURADA

A estrutura foi feita para ter diversos pontos de fixação para os mecanismos, garantindo que nada fique solto.



V2

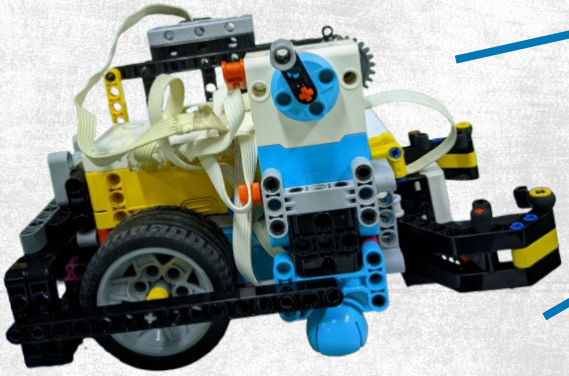
DESIGN DO ROBÔ

TIMELINE DO PROCESSO

O design do robô é uma das partes mais importantes, pois define sua identidade e também contribui para sua eficiência. Durante o processo de construção, foram desenvolvidas e testadas três versões, sendo a primeira utilizada como base para as demais, servindo como um molde inicial.

PRIMEIRA VERSÃO DO ROBÔ

A primeira versão do robô foi desenvolvida com um design extremamente simples, com uma estrutura básica formada por poucas peças e equipada com apenas um mecanismo: o "grab and lift" (agarrar e levantar). Um sensor de cor, localizado no lado esquerdo, era responsável por identificar a cor dos carrinhos e, a partir dessa leitura, a garra os empurrava até a área correspondente. Mesmo sendo uma versão simples, ela serviu de base para o desenvolvimento das outras duas que vieram em seguida, sendo uma escolha essencial para a construção da versão final do robô.



SIMPLES

Sua montagem é simples e compõe de poucas peças

EFICIENTE

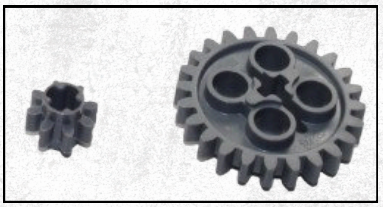
Em seus testes, o grab and lift teve ótimo desempenho.

RELAÇÃO DE TRANSMISSÃO

Com o uso do mecanismo "grab and lift", foi realizado um cálculo para determinar a relação de transmissão entre as engrenagens do eixo do motor e do eixo da garra. Esse cálculo é simples, consistindo apenas na divisão entre o número de dentes da engrenagem maior, que está no eixo da garra, e o da engrenagem menor, localizada no eixo do motor.

Í = 24/8

Í = 3



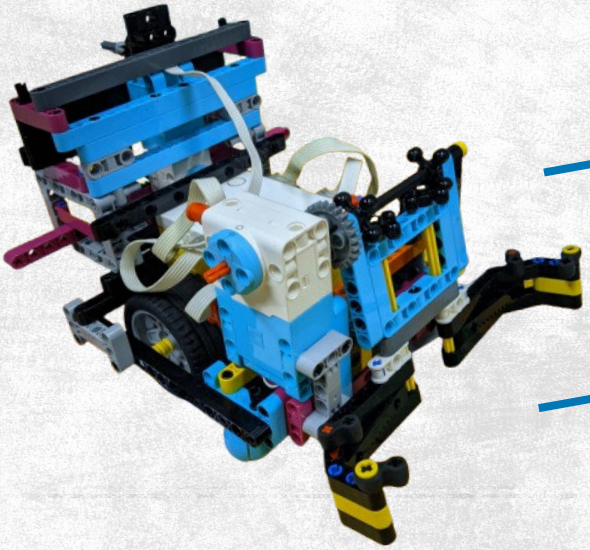
Para a engrenagem maior completar uma volta, a menor precisar completar 3 voltas.

+ Torque

DESIGN DO ROBÔ
TIMELINE DO PROCESSO

SEGUNDA VERSÃO DO ROBÔ

A segunda versão foi desenvolvida com o objetivo de permitir que o robô realizasse duas missões simultaneamente: o “Reflorestamento” e o “Expresso Ecológico”. Para isso, foi criado um mecanismo que permitiria se movimentar lateralmente, tanto para a esquerda quanto para a direita, a fim de empurrar os carrinhos até suas áreas designadas. Embora o mecanismo tenha se mostrado bastante eficiente e tenha cumprido bem sua função, o tamanho da estrutura acabou ultrapassando os limites da área de estacionamento, o que comprometeu o desempenho do robô na arena devido ao design excessivamente volumoso.



EFICIENTE

Todos os mecanismos tiveram um ótimo desempenho.

VERSÁTIL

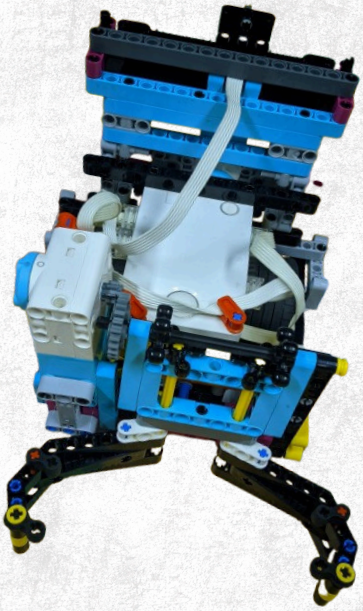
Consegue fazer duas missões simultaneamente.

TAMANHO EXCESSIVO

O robô não fica totalmente dentro da área de estacionamento.

INSTABILIDADE

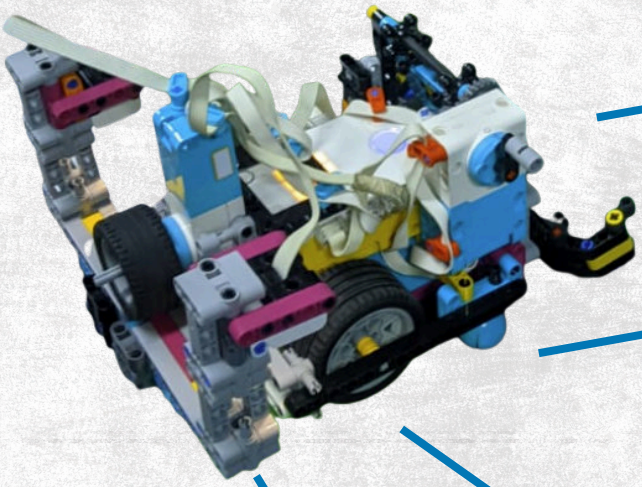
O robô apresenta certa instabilidade no mecanismo da cancela.



DESIGN DO ROBÔ
TIMELINE DO PROCESSO

TERCEIRA VERSÃO DO ROBÔ

A terceira e última versão é, sem dúvida, a melhor entre as três. Ela apresenta pouquíssimos pontos a serem aprimorados, os quais não comprometem seu desempenho durante o uso. O robô possui uma estrutura compacta, com base e mecanismos bem definidos. Além disso, seu design contribui significativamente para seu excelente desempenho, eliminando as limitações de tamanho observadas anteriormente. Essa versão concretiza os objetivos estabelecidos na fase de ideação: um robô compacto, versátil e consistente.



EFICIENTE

Seus mecanismos são eficientes na realização das missões

VERSÁTIL

Consegue fazer duas missões simultaneamente.

COMPACTO

Possui uma base bem estruturada e compacta.

CONSISTENTE

Durante os rounds apresenta pontuações constantes.

EVOLUÇÃO

DESAFIOS NA CONSTRUÇÃO DO ROBÔ

Durante a construção do robô, enfrentamos diversas dificuldades. Uma das principais foi idealizar e desenvolver um mecanismo eficiente e eficaz para empurrar os carrinhos. Ao analisarmos os manuais, percebemos que nossa estratégia inicial – utilizar apenas a garra e um sensor posicionado em um único lado do robô – não seria viável, já que um dos carrinhos possui a posição invertida em relação aos demais. Isso exigiu a criação de um novo mecanismo capaz de superar esse desafio, o que representou uma etapa particularmente complexa do projeto.

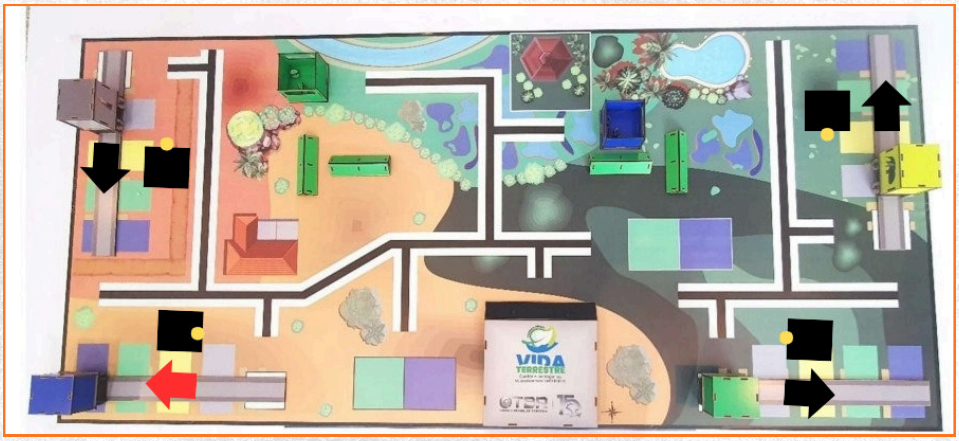
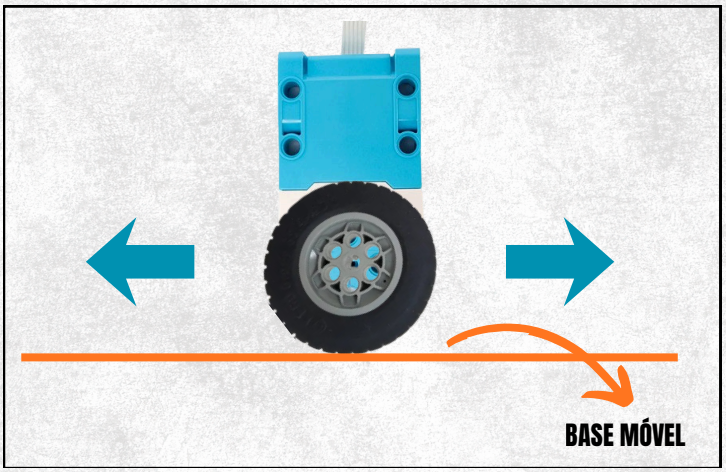


Foto retirada do caderno de engenharia.

SOLUÇÃO ENCONTRADA



Um mecanismo que tem amplitude suficiente para ambos os lados e mesmo assim estar dentro da área de estacionamento do robô. Seu funcionamento consiste em um motor que aciona uma roda que através do atrito faz uma base se mover para ambos os lados.

EVOLUÇÃO

DESAFIOS NA CONSTRUÇÃO DO ROBÔ

Durante a missão "Expresso Ecológico", observamos que o uso dos sensores de cor para a leitura dos carrinhos pode ser comprometido, uma vez que há elementos nos carrinhos – como as figuras dos animais e partes feitas de MDF – que podem dificultar a detecção precisa das cores, prejudicando assim o desempenho do robô na mesa.

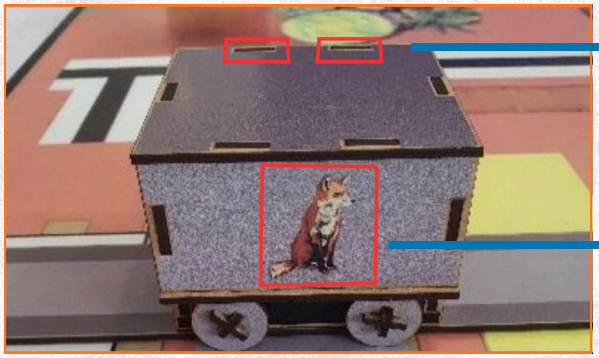


Foto retirada do caderno de engenharia.

Em todo o carrinho existem partes de encaixe de mdf que podem prejudicar a leitura correta do sensor de cor.

Nas partes laterais do carrinho existem as figuras dos animais, que podem prejudicar a leitura.

SOLUÇÃO ENCONTRADA

Como solução, propusemos que os sensores de cor fizessem a leitura pela parte superior dos carrinhos, onde se concentra a maior área visível da cor, facilitando a identificação correta.

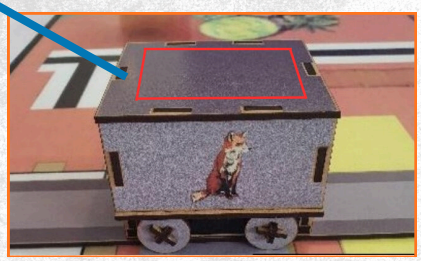
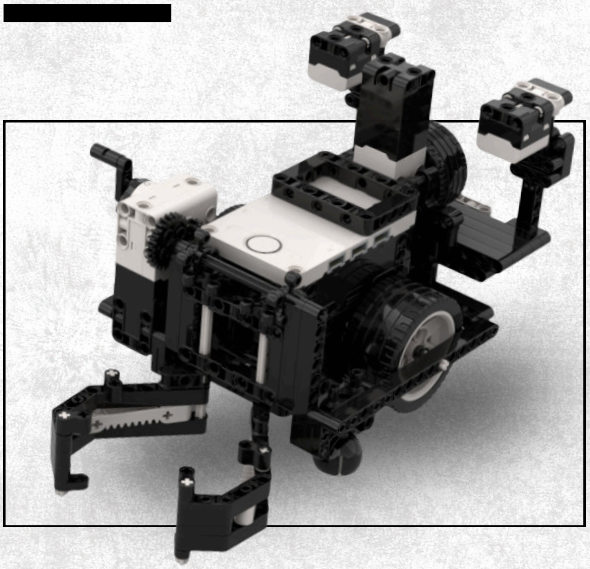


Foto retirada do caderno de engenharia.



EVOLUÇÃO

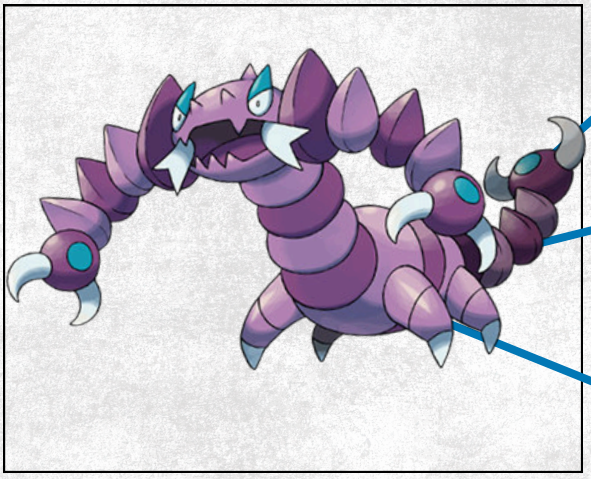
FICHA TÉCNICA DO ROBÔ



Motores 4
Sensores de cor 2
Giroscópio 1
Mecanismos 2
Relação de engrenagem 3x1

O robô possui dois mecanismos que permitem a realização de duas missões simultaneamente. Além disso, conta com um sistema de engrenagens que aumenta o torque disponível. Sua base é compacta e fechada, proporcionando firmeza e estabilidade à estrutura do robô.

DRAPION INSPIRAÇÃO



ÁGIL

Ambos possuem rapidez e precisão em seus movimentos.

RESISTENTE

Skorupi é conhecido por sua defesa, assim como o robô apresenta uma estrutura robusta e estável.

VERSÁTIL

O Pokémon pode se adaptar a diferentes batalhas, e o robô executa múltiplas tarefas simultaneamente, mostrando flexibilidade funcional.

ESTRATÉGIA DA MESA

Para garantir uma organização na ordem da realização das missões, dividimos a rota em 6 partes para executá-las. Dessa forma, cada parte o robô pode executar 2 missões simultaneamente ou somente uma.

PARTE 1
NA MESA

O robô sai da base em direção ao primeiro carrinho para realizar uma das missões de "expresso ecológico".



X 1

EXPRESSO ECOLÓGICO

Esperado um ganho de +55 pontos.

PARTE 2
NA MESA

O robô sai do primeiro carrinho e vai em direção a caixa verde e a pega. Logo depois vai em direção ao segundo carrinho e o coloca na sua posição destinada.



X 1

EXPRESSO ECOLÓGICO

Esperado um ganho de +55 pontos.

ESTRATÉGIA DA MESA

PARTE 3
NA MESA

O robô sai do segundo carrinho e vai em direção a área de plantio da cor verde para pontuar a primeira caixa de sementes.



X 1

REFLORESTAMENTO

Esperado um ganho de +50 pontos.

PARTE 4
NA MESA

Logo depois, o robô sai da área de plantio da caixa de sementes verde e vai em direção para pegar a caixa azul e depositar ela na área de plantio da cor azul.



X 1

REFLORESTAMENTO

Esperado um ganho de +50 pontos.

ESTRATÉGIA DA MESA

PARTE 5
NA MESA

O robô sai da área de plantio da cor azul e vai em direção para pontuar o terceiro carrinho da missão "expresso ecológico".



X 1

EXPRESSO ECOLÓGICO

Esperado um ganho de +55 pontos.

PARTE 6
NA MESA

E por último, o robô vai em direção ao quarto carrinho para pontuar, e depois vai para área de estacionamento, fechando assim a rota completa do robô na mesa.

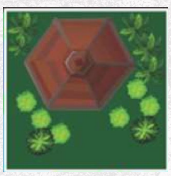


X 1

EXPRESSO ECOLÓGICO

Esperado um ganho de +55 pontos mais 50 pontos extras.

✚



DEPARTAMENTO ECOLÓGICO

Esperado um ganho de +80 pontos.

ORGANIZAÇÃO

PLANO A
NA MESA

MISSÃO	QUANTIDADE	PONTOS
Reflorestamento	2 caixas de sementes	100 pts.
Expresso Ecológico	4 vagões	220 pts + 50 pts.
Departamento Ecológico	Robô estacionado	80 pts.
Total		450 pts.

PLANO B
NA MESA

MISSÃO	QUANTIDADE	PONTOS
Reflorestamento	1 caixas de sementes	50 pts.
Expresso Ecológico	4 vagões	220 pts + 50 pts.
Departamento Ecológico	Robô estacionado	80 pts.
Total		400 pts.

Fizemos um plano B, caso haja algum imprevisto na estratégia A, realizando 11% menos pontos na arena.

ORGANIZAÇÃO

PLANO C
NA MESA

MISSÃO	QUANTIDADE	PONTOS
Reflorestamento	0 caixas de sementes	0 pts.
Expresso Ecológico	4 vagões	220 pts + 50 pts.
Departamento Ecológico	Robô estacionado	80 pts.
Total		350 pts.

Elaboramos um plano C para o caso de imprevistos na estratégia B, considerando uma redução de 22% na pontuação da arena. Tanto na estratégia B quanto na C, optamos por não pontuar uma das caixas de sementes, devido a um fator estratégico: a bonificação concedida quando todos os vagões estão corretamente posicionados. Dessa forma, caso o tempo disponível não seja suficiente para que o robô execute todas as ações previstas, a prioridade será completar a missão “Expresso Ecológico”, pois ela oferece uma pontuação maior em comparação à missão de “Reflorestamento”, além de garantir o bônus adicional.

INOVAÇÕES NA PROGRAMAÇÃO

NOVA LINGUAGEM DE PROGRAMAÇÃO

Iniciamos a temporada programando com o software do Kit Spike, mas, após alguns testes, percebemos a necessidade de mudar o foco para uma nova linguagem: o Pybricks. Essa escolha se mostrou muito mais vantajosa, já que a linguagem oferece maior precisão e mais recursos, permitindo que o nosso código ficasse estruturado exatamente da maneira que planejávamos.

Além disso, contamos com o apoio de alguns integrantes da nossa própria equipe e também de um membro da equipe de TE, que já possuíam experiência com Python. Essa base de conhecimento foi fundamental para nos auxiliar na adaptação e acelerar o processo de construção do código.

INTEGRAÇÃO DE NOVOS MÉTODOS

Integramos ao nosso trabalho novos métodos de organização e visualização no código: a Programação Orientada a Objetos (POO) e o conceito de Clean Code.

A POO é uma abordagem de programação voltada para partes específicas do projeto. Na prática, ela se baseia no uso de imports e na separação de cada subsistema em sua própria classe, o que facilita a manutenção, aumenta a eficiência ao realizar reparos e garante uma estrutura mais organizada.

Já o Clean Code atua em conjunto com a POO, aproveitando essa separação entre classes para tornar o código mais claro e fácil de compreender. Essa clareza não só agiliza o trabalho da equipe de Tecnologia e Engenharia, como também facilita a integração com membros de outras áreas e até mesmo com pessoas externas que precisem ler ou entender o código.

DIFICULDADES NA PROGRAMAÇÃO

ADAPTAÇÃO À NOVA LINGUAGEM

Um dos maiores desafios que enfrentamos na programação foi na migração para uma nova linguagem. Até então, utilizávamos os Blocos como principal, mas decidimos adotar o Pybricks, o que exigiu da equipe um período de adaptação para atender às novas demandas e aprender a utilizá-lo de forma eficiente.

Apesar da dificuldade inicial, conseguimos superar esse obstáculo trabalhando em conjunto, com destaque para o empenho dos programadores. Graças a esse esforço, alcançamos o resultado que queriamos e esperávamos, tornando o robô extremamente preciso.

LOCALIZAÇÃO NA MESA

Um dos momentos em que nos deparamos com um novo obstáculo foi na questão da localização do robô em relação à mesa.

A dificuldade estava no fato de que o robô não possuía um ponto fixo para se alinhar, o que acabava gerando pequenos desvios em sua rota, que posteriormente vinha a alterar a rota por completo. Além disso, a própria linguagem que utilizávamos contribuía para o problema, já que apresentava bastante instabilidade em várias funções, prejudicando a precisão nas missões.

A solução que encontramos para superar esse desafio foi trocar a linguagem de programação por uma mais confiável. Optamos pelo Pybricks, que não apenas atendeu às nossas necessidades de precisão e eficiência, como também facilitou a adaptação da equipe, já que alguns membros já possuíam conhecimento prévio sobre essa linguagem, ajudando a acelerar o processo.

PROGRAMAÇÃO

Recursividade de código

USO DO POO

Com a utilização da Programação Orientada a Objetos (POO), nosso código se tornou muito mais limpo, organizado e fácil de compreender. Essa abordagem trouxe uma estrutura mais clara à programação, permitindo que cada parte do código tivesse uma função bem definida e que as alterações fossem feitas de maneira mais prática e segura.

Um bom exemplo disso pode ser visto na própria programação, onde as classes foram divididas em diferentes partes, cada uma responsável por uma tarefa especifica. Essa separação facilitou a identificação de possíveis erros e tornou o processo de manutenção muito mais ágil, já que conseguimos trabalhar em seções isoladas sem comprometer o restante do código.

- O uso de “imports” é a maneira de utilizar o POO, onde nós usamos funções e variáveis de uma classe em outra

```
from pybricks.hubs import PrimeHub
from pybricks.pupdevices import Motor, ColorSensor, UltrasonicSensor, ForceSensor
from pybricks.parameters import Button, Color, Direction, Port, Side, Stop
from pybricks.robotics import DriveBase
from pybricks.tools import wait, StopWatch
from consts import Open, close, low, mid, deposit, cancela_esquerda, cancela_direita, volta_direita, volta_esquerda
from Drive_Train import claw, andar, turn, rotate, cancela
from functions import detect_train_I, detect_train_II, detect_train_III, detect_train_IV
from Rotas import route_I, route_II, route_III, route_IV, train_I, train_III, train_IV, boxG, boxB, park
```

```
from Drive_Train import claw, andar, turn, rotate, cancela
from functions import detect_train_I, detect_train_II, detect train_III, detect_train_IV
```

PROGRAMAÇÃO

Recursividade de código

USO DO HUE

Durante nossos testes, percebemos que o robô confundia as cores azul, cinza e verde, já que a porcentagem de luz refletida por elas era muito semelhante. Diante disso, surgiu a necessidade de desenvolver um código capaz de diferenciá-las. Após diversas pesquisas e experimentos, descobrimos o hue (matiz), um dos componentes do modelo de cor HSL (Hue, Saturation, Lightness). Com novos testes, verificamos que azul, cinza e verde apresentam índices de hue bastante distintos, o que nos permitiu diferenciá-los na mesa e evitar erros de reconhecimento por parte do robô.

```
PRETO_MAX = 15
AMARELO_MIN = 60
HUE_SATURACAO_MIN = 20

HUE_AZUL_MIN = 200
HUE_AZUL_MAX = 240
HUE_VERDE_MIN = 120
HUE_VERDE_MAX = 180

def get_cor_confiavel(sensor: ColorSensor) -> Color:
    intensidade_reflexao = sensor.reflection()
    hsv = sensor.hsv() # (Hue, Saturation, Value)
    hue = hsv[0]
    saturation = hsv[1]

    cor_detectada = Color.NONE

if intensidade_reflexao >= AMARELO_MIN:
    if hue > 40 and hue < 70 and saturation > HUE_SATURACAO_MIN:
        cor_detectada = Color.YELLOW
    else:
        cor_detectada = Color.GRAY

elif intensidade_reflexao <= PRETO_MAX:
    cor_detectada = Color.BLACK

elif hue >= HUE_AZUL_MIN and hue <= HUE_AZUL_MAX and saturation > HUE_SATURACAO_MIN:
    cor_detectada = Color.BLUE
```

PROGRAMAÇÃO

Recursividade de código

UTILIZAÇÃO DO IMU

Começamos a utilizar o IMU após a etapa regional, onde percebemos que a orientação do robô ficava instável por complicações na montagem da mesa.

Para fazermos com que qualquer erro pudesse ser mitigado, utilizamos o sensor interno do robô, o IMU. Ele é usado como um método de orientação para o robô, fazendo com que os erros, em geral na movimentação, sejam consertados pelo próprio robô.

```
hub.imu.up()

while True:
    roll, pitch, yaw = hub.imu.orientation()

    print(roll, pitch, yaw)

    if abs(roll) > 20 or abs(pitch) > 20:
        hub.light.on(255, 0, 0)
    elif abs(yaw) > 170:
        hub.light.on(0, 0, 255)
    else:
        hub.light.on(0, 255, 0)
```

PROGRAMAÇÃO

Recursividade de código

MOVIMENTAÇÃO

A forma como configuramos a movimentação no Pybricks é um pouco diferente do que estávamos acostumados. Primeiro, precisamos indicar quais motores serão responsáveis pelo movimento, depois definir os valores das rodas e escolher se vamos utilizar ou não o Gyro, uma função do próprio sistema que permite ao robô girar em torno do próprio eixo. Só então configuramos a função;

- Definição da DriveBase (motores de movimentação, diâmetro das rodas, em milímetros, e distância entre as rodas, em milímetros);
- Configuração para permitir com que o robô possa girar em seu próprio eixo;
- Função para andar, essa função calcula automaticamente a distância, que naturalmente é em milímetros, para que o robô ande em centímetros.

```
drive_base = DriveBase(me, md, 54, 115)
drive_base.use_gyro(True)

def andar(dist):
    drive_base.settings(350, 350, 220, 220)
    drive_base.straight(dist*10)
```

PROGRAMAÇÃO

Recursividade de código

ROTATE & TURN

- Define o quanto o robô vai girar baseado no angulo e no raio que indicamos para ele, podendo girar uma quantidade especifica e parar orientado para outra;

- Função para o robô girar em torno de seu próprio eixo, sem se mover para frente ou para trás.

```
def turn(radius, angle):
    drive_base.settings(200, 200, 200, 200)
    drive_base.curve(radius, angle)

def rotate(graus):
    drive_base.settings(220, 220, 220, 220)
    drive_base.turn(graus)
```

PROGRAMAÇÃO

Recursividade de código

CONSTANTES(GARRA)

Todas os valores que a nossa garra pode assumir, elas possuem uma definição diferente para cada posição;

- Valor da GARRA aberta;
- Valor da GARRA fechada;
- Valor da GARRA fechada e baixa;
- Valor da GARRA fechada e no meio;
- Valor da GARRA alta.

```
def Open():
  claw.run_target(2000, 0)

def close():
    claw.run_target(2000, 100)

def low():
    claw.run_target(2000, 130)

def mid():
    claw.run_target(2000, 220)

def deposit():
    claw.run_target(1800, 600)
```



PROGRAMAÇÃO

Recursividade de código

CONSTANTES(CANCELA)

Todas os valores que a nossa cancela pode assumir, elas utilizam de graus(diferente da garra) para uma maior precisão, por ser um motor menor ele tem uma menor rotação causando uma diferença nos valores;

- Valor da CANCELA para ir à esquerda;
- Valor da CANCELA para voltar da esquerda para o centro;
- Valor da CANCELA para ir à direita;
- Valor da CANCELA para voltar da direita para o centro.

```
def cancela_esquerda():
    cancela.run_angle(1800, 180)

def volta_esquerda():
    cancela.run_angle(1800, -105)

def cancela_direita():
    cancela.run_angle(1800, -180)

def volta_direita():
    cancela.run_angle(1800, 150)
```

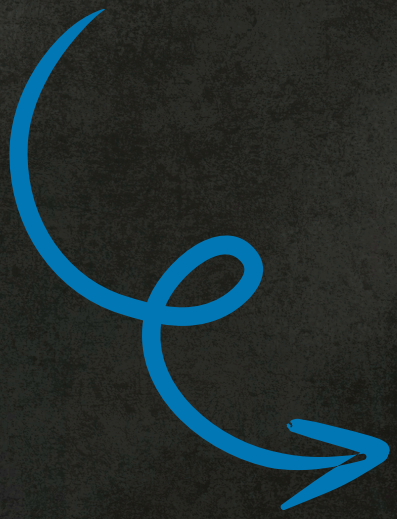
PROGRAMAÇÃO

Recursividade de código

ROTAS

Dividimos as nossas rotas em uma classe separada para podermos fazer ajustes direcionados, sem o risco de causar interferências em outras partes e para uma maior eficiência ao fazer adaptações caso necessário.

- Sai da base inicial
- Se direciona para o primeiro trem
- Faz o caminho até a caixa de sementes verde
- Faz o caminho para o segundo trem
- Entrega a caixa verde



```
# SAI DA BASE
def route_I():
    andar(25)
    close()
    rotate(-90)
    andar(30)

# VAI PARA O TREM
def train_I():
    rotate( 90)
    andar(16)
    rotate(90)
    andar(18)

#caminho até a caixa verde
def route_II():
    rotate(90)
    andar(45)
    rotate(90)
    andar(20)


def route_III():
    andar(-40)
    rotate(90)
    cancela_direita()

#caminho para depositar a caixa verde
def boxG():
    rotate(-90)
    andar(30)
    rotate(90)
    andar(5)
```

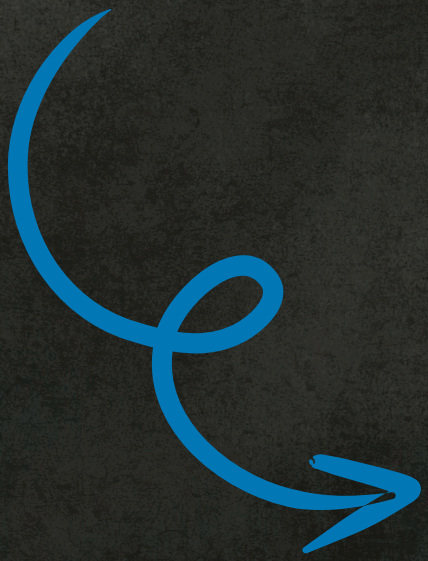
PROGRAMAÇÃO

Recursividade de código

ROTAS

Continuação e finalização das rotas;

- Caminho até a caixa azul;
- Depoisto da caixa azul;
- Caminho e execução do terceiro trem;
- Estacionamento.



```
#caminho até a caixa azul
def route_IV():
    andar(-30)
    rotate(-90)
    andar(60)
    rotate(-90)

#caixa azul
def boxB():
    andar(15)
    low()
    andar(-15)
    rotate(90)
    Open()

def train_III():
    rotate(90)
    andar(20)
    rotate(90)
    cancela_direita()

def train_IV():
    rotate(-90)
    andar(20)

#estacionar
def park():
    rotate(-90)
    andar(65)
    rotate(90)
    andar(-5)
    deposit()
```

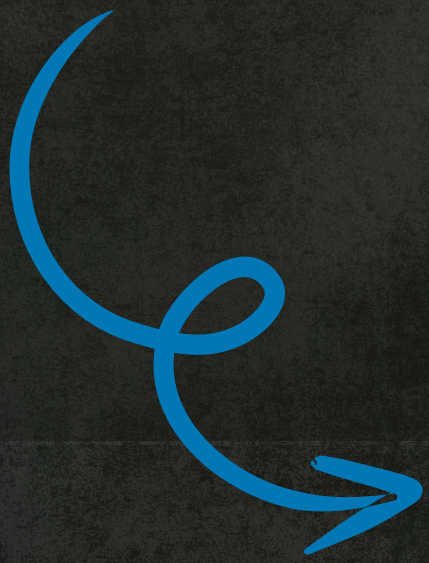
PROGRAMAÇÃO

Recursividade de código

IDENTIFICAR AS CORES DOS TRENS

Para identificarmos as cores dos trens de maneira automática, criamos funções que movem a cancela e andam de acordo com a cor identificada pelo sensor. Cada cor identificada tem um valor especifico para se mover;

- A primeira cor é o azul que, quando identificada, ativa a cancela e anda a quantidade especifica;
- A segunda é a amarela;
- A terceira a cinza;
- A quarta o verde
- A função que assimila as cores às suas variáveis;
- A função que faz com que a cor seja identificada.



```
def azul_TI():
    andar(45)
    wait(500)
    andar(-25)
    volta_esquerda()

def amarelo_TI():
    andar(22)
    volta_esquerda()

def cinza_TI():
    andar(13)
    volta_esquerda()

def verde_TI():
    andar(34)
    volta_esquerda()

train_I = {
    Color.BLUE: azul_TI,
    Color.YELLOW: amarelo_TI,
    Color.RED: cinza_TI,
    Color.GREEN: verde_TI
}

def detect_train_I():
    train_I[colors_esq]()
```

PROGRAMAÇÃO

Recursividade de código

ROTA COMPLETA

Utilizando esse método para manter o nosso código limpo e de fácil entendimento, conseguimos desenvolver um programa final que ficou bem curto e objetivo. Ele atendeu plenamente ao que tínhamos planejado, além de se manter organizado e simples de interpretar. Essa clareza foi essencial para que todos da equipe pudessem compreender o funcionamento do código sem dificuldades, facilitando tanto a programação quanto eventuais ajustes. Dessa forma, garantimos não apenas eficiência no robô, mas também uma base sólida para evoluir nas proximas etapas.

```
route_I()
train_I()
detect_train_I()
route_II()
low()
route_III()
detect_train_II()
boxG()
route_IV()
boxB()
train_III()
cancela_esquerda()
rotate( 90)
volta_esquerda()
detect_train_IV()
park()
```

SOFTWARES

UTILIZADOS NO PROCESSO

RECURSOS HUMANOS

Para que a equipe de engenharia do robô pudesse fazer um trabalho colaborativo mesmo distante do local de treino e desenvolvendo habilidades para o mercado de trabalho e nosso desenvolvimento humano, utilizamos a ferramenta em nuvem OneDrive. Dessa forma, temos um registro contínuo de tudo que fizemos no torneio além do caderno de engenharia que também fica disponível na nossa pasta compartilhada



DIVULGAÇÃO

Como estamos programando em bloco o robô, decidimos armazenar nossa programação no GitHub, que é um serviço baseado em nuvem que hospeda códigos e permite os desenvolvedores colaborem e façam mudanças em projetos compartilhados enquanto mantêm um registro detalhado do seu progresso. Disponibilizamos o link do repositório no nosso Instagram.

Como ferramenta para divulgar o nosso design de robô, nós utilizamos a plataforma do BrickLink, criadora do Studio 2.0 que utilizamos para modelar, dessa forma podemos deixar o modelo do nosso robô disponível para download e visualização online. Disponibilizamos o link do modelo no nosso Instagram.





CONTATOS

Nossos meios de comunicação!

(21) 98518-6299

alphaftc20858@gmail.com
alphatechjac@gmail.com

@alphatbr

Firjan SESI/SENAI - Jacarepaguá