**Desenvolvimento de software para explorar funcionalidades dos robôs móveis de tração diferencial e omnidirecional**

Atividades para o mês de setembro, outubro, novembro e dezembro:

1. DESENVOLVIMENTO DOS CÓDIGOS DE AQUISIÇÃO DE DADOS E ACIONAMENTO DOS MOTORES PARA O MICROCONTROLADOR.
2. IMPLEMENTAR A COMUNICAÇÃO ENTRE O COMPUTADOR DE PLACA ÚNICA E O MICROCONTROLADOR.
3. INSTALAR E FAZER TESTES COM O SENSOR IMU.
4. INVESTIGAR E IMPLEMENTAR O CÓDIGO DE ODOMETRIA.

* DIA 14/SET *(Em Casa)*
  + *Instalação do Ubuntu Desktop 24.04 LTS no cartão SD do omnidirecional*
    - O sistema foi instalado através da ferramenta [*Raspberry Pi Imager*](https://www.raspberrypi.com/software/)*.*
  + *Verificação e teste do processo de instalação do* [*ROS*](https://www.ros.org/)
    - Para o teste, a versão [*ROS 2 Jazzy Jalisco*](https://docs.ros.org/en/jazzy/Installation/Ubuntu-Install-Debs.html) foi instalada em uma máquina virtual com o Ubuntu.
  + *Revisão dos primeiros passos com o ROS*
    - Criação do workspace, criação de pacotes, testes com nós e tópicos, etc.
* DIA 15/SET *(Em Casa)*
  + *Verificação da necessidade de instalação do SSH para acesso remoto.*
    - O comando sudo apt install openssh-server openssh-client instala o SSH. [*Referência*](https://guidetux.com.br/como-instalar-ssh-ubuntu-24-04-22-04-ou-20-04/).
    - O SSH é essencial para conseguir acessar e executar comandos no UBUNTU remotamente.
  + *Pesquisa sobre a comunicação serial com o Raspberry*
    - Foi pesquisado a respeito da conexão física, habilitação e uso da comunicação Serial UART. Mais informações e referências podem ser encontradas neste [*ARQUIVO*](https://drive.google.com/file/d/1JJx3BCYrpgJANMMoZAd9y-k9G8v92Jo3/view?usp=drive_link).
* DIA 17/SET *(COLAB)*
  + *Instalação do Ubuntu Desktop 24.04 LTS no cartão SD do diferencial*
    - A imagem “Preinstalled desktop image” do sistema foi baixada neste [link](https://cdimage.ubuntu.com/releases/noble/release/) e carregada usando a ferramenta [*Raspberry Pi Imager*](https://www.raspberrypi.com/software/)*.*
  + *Configuração do ubuntu nos Raspberry*
    - O ubuntu foi devidamente instalado e foi definido o nome de usuário para o omnidirecional “**colab**” e senha “**ifalpibicjr**”, e para o diferencial foi definido usuário “**diferencial**” e senha “**pibicjr**”.
    - O ssh também foi instalado em ambos.
    - O vscode foi instalado em ambos. Foi baixado através deste [link](https://code.visualstudio.com/download) o arquivo “.deb”, [instalado via terminal](https://help.ubuntu.com/kubuntu/desktopguide/pt_BR/manual-install.html) e devidamente configurado.
  + *Instalação e configuração do ROS*
    - o Ros foi devidamente instalado seguindo as [instruções](https://docs.ros.org/en/jazzy/Installation/Ubuntu-Install-Debs.html).
    - Também foi [instalado o colcon](https://docs.ros.org/en/jazzy/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html) para que seja possível compilar pacotes.
    - Além disso, foram criados os workspaces.
    - Foram realizados testes com um [pacote exemplo](https://docs.ros.org/en/jazzy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html) onde dados eram enviados de um raspberry até o outro através do ROS usando a rede.
* DIA 19/SET *(COLAB)*
  + *Habilitação da porta serial física (falhado):*
    - Alguns testes foram realizados para a habilitação da porta serial do Raspberry, no entanto nenhum foi bem sucedido.
    - Pesquisas posteriores apontaram que aparentemente esta porta está sendo usada pelo Bluetooth ou por um chamado “console serial”
* DIA 20/SET *(Em Casa)*
  + *Pesquisas mais aprofundadas sobre a habilitação da serial física:*
    - Será necessário desabilitar o Bluetooth e o serviço responsável pelo “console serial”.
    - é necessário habilitar a porta serial no arquivo ‘config.txt’ em /boot/firmware/ adicionando ‘enable\_serial=1’
    - Para desativar o bluetooth é necessário [adicionar uma linha](https://community.umbrel.com/t/how-to-turn-off-bluetooth-and-wifi-on-rasberry-pi4/584/2) no arquivo config.txt e desabilitar o serviço **bluetooth**.
* DIA 23/SET *(COLAB)*
  + *Instalação do Arduino IDE:*
    - O IDE do arduino foi instalado no raspberry. A versão Legacy IDE foi instalada no [site oficial](https://www.arduino.cc/en/software). O arduino pode ser facilmente gravado via USB, no entanto foi necessário mudar as permissões de acesso através do comando sudo chmod a+rw /dev/ttyUSB0 onde ttyUSB0 se refere a porta a qual o arduino está conectado.
  + *Habilitação da serial física:*
    - O arquivo config.txt foi devidamente alterado, o Bluetooth foi desabilitado e o “console serial” também.
    - Ao desabilitar o bluetooth, a porta serial já ficou disponível em /dev/ttyAMA0 em ambos Raspberry.
    - Não é possível gravar um programa no arduino usando a UART física do Raspberry (Através do IDE).
* DIA 24/SET *(COLAB)*
  + *Comunicação serial entre Raspberry e Arduino.*
    - *Por parte do arduino, o código de comunicação e controle via serial já havia sido desenvolvido em um outro projeto. Ele foi reutilizado aqui. Para a conexão foi usado um divisor de tensão com 2 resistores de 10k.*
    - *Por parte do Raspberry, foi usada a biblioteca* [*pyserial*](https://pyserial.readthedocs.io/en/latest/) *para conseguir se comunicar com o arduino.*
    - *Existia a necessidade de, a cada reconexão,* mudar as permissões de acesso através do comando sudo chmod a+rw /dev/ttyAMA0 ou então era necessário rodar o código python com permissão de superusuário.
    - Após ser feita várias requisições na comunicação, houve um erro apontando que múltiplos dispositivos estavam usando a comunicação serial, foi notado que **o “console serial”, mesmo sendo desabilitado, é reativado a cada reinicialização**, e este interfere na comunicação serial.
* DIA 26/SET *(COLAB)*
  + *Controlador:*
    - O [controlador](https://drive.google.com/file/d/1y5kUdiDnSBbAk7n1IZYTRkGPu_Ou_zMF/view?usp=sharing) (interface gráfica) desenvolvido no outro projeto foi baixado/configurado/testado no raspberry pi do omni.
    - No outro raspberry pi ele foi baixado, mas não configurado
  + *ROS - os testes:*
    - A biblioteca pyserial foi implementada em um nó Ros e foram realizados testes onde o controlador e o nó estavam rodando no Raspberry do omni e o nó conseguia receber dados do controlador e enviar para o arduino.
* DIA 28/SET *(Em casa)*
  + *Pesquisa: Desabilitação permanente do serviço do “console serial”*
    - Usar sudo systemctl mask <nome\_servico> vai impedir que o serviço inicie na inicialização. [Referência.](https://sobrelinux.info/questions/12208/what-is-the-difference-between-systemctl-mask-and-systemctl-disable#google_vignette)
  + *Pesquisa: Permissões (chmod e chown)*
    - Para conseguir usar a porta seria /dev/ttyAMA0 sem a permissão de super usuário, ou seja, dentro de um nó ROS, será necessário [alterar o proprietário](https://www.hostinger.com.br/tutoriais/como-alterar-permissoes-e-proprietarios-via-linha-de-comando) através do comando sudo chown user /dev/ttyAMA0 onde user é o nome do usuário.
* DIA 30/SET *(COLAB)*
  + *Desativação permanente do console serial:*
    - O “console serial” foi desativado permanentemente usando o comando já visto.
  + *Configurando o Controlador:*
    - O controlador foi configurado no robô de tração diferencial
    - O nó ros responsável pelo recebimento de dados e envio ao arduino também foi criado.
* DIA 02/OUT *(COLAB)*
  + *Teste do Robô:*
    - Ambos os robôs foram testados sendo controlados um pelo outro (um de cada vez) navegando pelo chão.
    - Este teste tinha como objetivo verificar a troca constante de dados e a velocidade de resposta.
    - A troca constante de dados estava okay, não havendo nenhum erro em nenhum momento.
    - No entanto, o tempo de resposta, tanto no envio, quanto na passagem do Raspberry para o arduino apresentaram-se limitantes.
* DIA 04/OUT *(COLAB)*
  + *Instalação do Dropbox:*
    - Foi averiguado como instalar o Dropbox no Ubuntu (Raspberry). Ele ainda não foi instalado.
    - Com o Dropbox ficará mais fácil a troca de códigos entre os robôs bem como trabalhar com os mesmos em outros computadores.
  + *Aplicação dos Resistores na placa:*
    - O robô de tração diferencial foi desmontado e o divisor de tensão, bem como os conectores, foram soldados em sua placa.
    - Anotações das conexões também foram feitas na placa.
* DIA 07/OUT *(COLAB)*
  + *Remontagem do robô de tração diferencial*
  + *Criação/aprimoramento do código do arduino*
    - Todos os dados que o arduino precisa receber são enviados de uma única vez e ele responde com todos os dados que precisa responder.
    - O tempo para ele receber e responder é de aproximadamente 1.5s
* DIA 09/OUT *(COLAB)*
  + *Elaboração da Estrutura ROS*
    - Um único nó por robô, publicando/escutando de 4 tópicos chamados motor, lcd, sensor, velocidade.
  + *Criação/Aprimoramento do código python que se comunica com o Arduino:*
    - O código agora envia uma informação e espera a resposta.
    - Antes o código tentava fazer isso, mas por algum motivo, antes de retornar um dado real o Arduino retornava um byte vazio e o código se confundia achando que era um dado
    - Agora o código verifica se o byte recebido é diferente de vazio, se for vazio ele espera.
* DIA 11/OUT *(COLAB)*
  + *Correção da montagem do Robô:*
    - Foi verificado um erro quanto à montagem do robô.
    - O encoder não está funcionando devido a inversão da polaridade de alimentação.
    - O erro foi identificado e será corrigido no dia 14/OUT.
    - O robô será desmontado e remontado novamente.
  + *Criação do código Ros*
  + *Tentativa de criação de Serviço:*
    - Houve a tentativa da criação de um serviço que daria a permissão de uso da porta seria quando o Raspberry Iniciasse.
    - A criação falhou aparentemente por causa da necessidade da permissão sudo no serviço.
* DIA 13/OUT *(Em casa)*
  + *Verificação do problema na criação do serviço:*
    - O teste foi um sucesso. É possível criar serviços que executem códigos root e também normais.
    - O problema é que para a execução do serviço root o código python terá que ter a senha escrita no corpo. [Referências.](https://drive.google.com/file/d/18uweMQUlYYfOwvjXx2gyKKNYsvv6AHoy/view?usp=drive_link)
* DIA 14/OUT*(COLAB)*
  + *Criação do serviço:*
    - Mais testes foram feitos a respeito da criação de serviços. O serviço que torna a porta serial acessível foi criado.
    - Para isso o usuário root foi habilitado com a mesma senha que o usuário normal.
    - Testes e configurações feitas no Diferencial.
  + *Correção do problema de conexão da placa:*
    - O erro foi corrigido, no entanto ao corrigir esse erro houve uma inversão de polaridade no divisor de tensão que vai do ardu para o Rasp.
    - O erro será corrigido futuramente.
* DIA 15/OUT
* *(COLAB)*
  + *Correção final das conexões:*
    - A placa agora está 100% funcional.
    - Foi testada a comunicação entre o arduino e o Raspberry.
  + *Problema:*
    - O tempo para o arduino receber, reconhecer, tratar e responder é muito grande, algo em torno de 1.5 segundos. Tentarei diminuir reduzindo a quantidade de dados enviados e os simplificando para diminuir o tempo de reconhecimento por parte do arduino.
    - O código está acendendo e apagando o lcd a cada dado recebido. Mudar isso.
    - Durante a iniciação, dados são enviados pela serial, fazendo com que o lcd ligue ou até os motores girem. A ideia é colocar uma palavra-chave que inicie a comunicação entre eles. Assim os dados recebidos durante a inicialização não serão tratados.
* *(Em Casa)*
  + *Ajuste do código do Ardu:*
    - O código foi ajustado para esperar a palavra “manoel” antes de ler e reagir a qualquer informação recebida via serial.
* DIA 16/OUT*(COLAB)*
  + *Adaptação e teste final do código:*
    - O código foi adaptado para garantir que o arduino só receba dados dele.
* DIA 18/OUT*(COLAB)*
  + *Elaboração da nova forma de enviar/receber dados:*
    - O arduino ficará enviando os dados dos sensores periodicamente.
    - Em “paralelo” a isso ele ficará sempre esperando uma informação na porta serial.
    - Para tal feito é necessário usar o timer do arduino.
  + *Pesquisa sobre o timer:*
    - Foi pesquisado e desenvolvido um código de teste a respeito dos timer no arduino.
* DIA 20/OUT*(Em casa)*
  + *Pesquisa a respeito dos timer do Arduino.*
    - Foi feita uma pesquisa mais profunda. [Ref 1](https://www.aranacorp.com/pt/uso-dos-timers-do-arduino/#google_vignette). [Ref 2](https://embarcados.com.br/timers-do-atmega328-no-arduino/).
  + *Teste Py - recebendo IP e Nome do Access Point.*
    - O IP pode ser obtido através do comando hostname -I sendo executado juntamente com a biblioteca subprocess.
    - Consegui achar uma biblioteca capaz de listar as redes wifis disponíveis ou conectar-se a uma rede. [Referência](https://ronaldonunez.wordpress.com/2015/06/28/listando-as-redes-wi-fi-em-python/).
    - Foi encontrada outra biblioteca que tem a capacidade de indicar qual rede você está conectado. [Referência](https://pypi.org/project/python-wifi/).
    - Não foram realizados testes, apenas pesquisas,
  + *Pesquisa LCD no Raspberry:*
    - [Referência 1](https://medium.com/@thedyslexiccoder/how-to-set-up-a-raspberry-pi-4-with-lcd-display-using-i2c-backpack-189a0760ae15).
    - Uma outra forma mais complexa e detalhista [Aqui](https://www.arduinoecia.com.br/como-usar-display-lcd-i2c-raspberry-pi/).
    - a
* DIA 21/OUT*(COLAB)*
  + *Uso dos timers:*
    - Os timers do arduino são usados para coisas como PWM, se forem dedicados a outra tarefa essa função não será mais possível.
  + *LCD (exibindo informações):*
    - O lcd passou a exibir informações como a rede e o ip.
* DIA 23/OUT*(COLAB)*
  + *Serviço:*
    - Um Serviço foi criado para o lcd já iniciar exibindo a informação.
  + *Tratamento de dados:*
    - *Os dados dos sensores são separados e repassa através de topicos*
* DIA 25/OUT*(COLAB)*
  + *Serviço de inicialização Com nó ROS:*
    - A criação de um serviço Linux que inicie um no ROS não foi bem sucedida.
  + *Soldagem dos Terminais dos STM32:*
    - Os terminais foram soldados.
* DIA 26/OUT*(Em casa)*
  + *Pesquisa (STM32 no arduino):*
    - Foi achado uma forma bem simples de programar o stm32 na IDE do arduino.
    - Testes não foram feitos.
* DIA 30/OUT*(COLAB)*
  + *Teste e Funcionamento:*
    - Foi necessário alterar o modo de boot para BOOT1=0, BOT0=0
    - O STM32 foi programado pelo Arduino IDE
    - Houve um teste de PWM
    - Houve uma pesquisa sobre a possibilidade de trabalhar com entradas de 5V. Os resultados foram positivos
* DIA 01/NOV*(COLAB)*
  + *Comunicação UART:*
    - Foi testado a comunicação entre o STM32 e o Raspberry via serial.
  + *Teste final diferencial:*
    - O teste final (ligar e funcionar sem a necessidade de uma tela foi feito.
    - Em algumas situações, o nó ros não era iniciado. O problema será corrigido.
  + *Desenvolvimento do código:*
    - Foi elaborado o início do código do stm32 que vai funcionar de forma semelhante ao do Arduino.
* DIA 02/NOV*(Em casa)*
  + *Interrupções STM32:*
    - Existem 16 pinos disponíveis, no entanto todos apontam para a mesma interrupção.

***Próximos passos:***

* *Ajustar/Revisar o código que coleta os dados e controla os motores via Serial no Arduino.*
* *Criar a estrutura (pacotes) do ROS para controle externo*
* *Fazer os nós serem executados ao iniciar o sistema..*
* *Adicionar a conexão física da comunicação na placa de distribuição do robô omnidirecional.*
  + *Desmontar e remontar o robô.*
* *Calibrar os Encoder - Definir experimentalmente a Resolução (pulsos por volta).*
* *Usar o LCD para exibir informações importantes como IP e rede a qual o Raspberry está conectado para fácil acesso via SSH.*
* *Implementar uma verificação e correção de erro nos pontos de comunicação.*