

THOMAZ AKIRA FURUKAWA

**DESENVOLVIMENTO DE EQUIPAMENTOS
PARA HOSPITAL UNIVERSITÁRIO DA USP**

São Paulo
2022

THOMAZ AKIRA FURUKAWA

**DESENVOLVIMENTO DE EQUIPAMENTOS
PARA HOSPITAL UNIVERSITÁRIO DA USP**

Trabalho apresentado à Escola Politécnica
da Universidade de São Paulo para con-
clusão de Iniciação Científica .

São Paulo
2022

THOMAZ AKIRA FURUKAWA

**DESENVOLVIMENTO DE EQUIPAMENTOS
PARA HOSPITAL UNIVERSITÁRIO DA USP**

Trabalho apresentado à Escola Politécnica
da Universidade de São Paulo para con-
clusão de Iniciação Científica .

Área de Concentração:
Engenharia Mecatrônica

Orientador:
Leopoldo Rideki Yoshioka

Co-orientador:
Oswaldo Horikawa

São Paulo
2022

Prof. Dr. Leopoldo Rideki Yoshioka

Prof. Dr. Oswaldo Horikawa

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Catálogo-na-publicação

da Silva dos Santos, Vanderson

Robô Hospitalar: Hardware e Software / V. da Silva dos Santos --
São Paulo, 2021.

p.

Trabalho de Iniciação Científica - Escola Politécnica da
Universidade de São Paulo. Departamento de Engenharia de Sistemas
Eletrônicos.

1.hardware 2.software 3.robótica 4.sistemas embarcados
5.ROS I.Universidade de São Paulo. Escola Politécnica.
Departamento de Engenharia de Sistemas Eletrônicos II.t.

Dedico esse trabalho aos atuais
e futuros integrantes da ZIMA -
Soluções Médico Hospitalares

AGRADECIMENTOS

Primeiramente, à minha família, que me forneceram inspiração, suporte emocional e financeiro para que eu chegasse até a universidade.

Ao meus professores orientadores, Leopoldo Yoshioka, pela a oportunidade de participar do grupo , fazer a manutenção do projeto e sempre dar suporte, tanto em ajuda, quanto em ensinamentos através de boas conversas e alguma risadas. Ao Professor Oswaldo Horikawa, que foi um forte auxílio técnico no desenvolvimento dos projetos e me inspirou a ser um engenheiro melhor.

Aos coordenadores de área e antigos membros, Vanderson Santos, Lucas Boccia, Lucas Grob, Wu Kam Long, Pedro Croso, Caio Oliveria, Lucas Junji e Joao Magano que sem essa equipe nada do que será apresentado seria possível.

“Para ganhar, tem que jogar bem e dar sorte.”

- Josué Ramalho da Silva

RESUMO

Desde 2020, a Escola Politécnica e Hospital Universitário da USP colaboram juntos para o desenvolvimento de soluções tecnológicas na área da saúde. O trabalho de incentivo a inovação foi o responsável pelo projeto Robô Hospitalar, robô autônomo para entregas de exames laboratoriais, que chamou atenção dos colaboradores do Hospital para as oportunidades de inserir a tecnologia para inovar na saúde. Com o desenvolvimento do projeto e a entrega de resultados concretos, novas oportunidades surgiram, sendo duas delas sementes de dois novos projetos: o Dispensador de Remédios da Farmácia do HU e o aparelho de reabilitação Ciclo Ergômetro. Com a aproximação da Escola Politécnica e o Hospital Universitário da USP o incentivo a inovação se deu de diversas maneiras, não se limitando a entrega de projetos mas também a criação de grupos focados em inovação na saúde.

Palavras-Chave – Automação, Algoritmos de Controle, Programação de Embarcados, Robótica, Saúde, Eletromédicos, Hospitalar, Inovação.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

PARTE I

INTRODUÇÃO

A Escola Politécnica e o Hospital Universitário vem se aproximando através de projetos como O Robô Hospitalar, um Robô capaz de se movimentar autonomamente dentro do hospital e realizar as entregas de exames laboratorias. Esse primeiro projeto foi promovido pela iniciativa do Professor Leopoldo e Dr. Oscar Fugita, responsável pelo Núcleo de Inovação e Tecnologia do HU(INTEC)[FOTO]. Com o sucesso da colaboração se concretizando através da evolução do projeto, que já se encontrava na Segunda Versão em 2021[FOTO], foi proposto a resolução de dois novos problemas do HU para esse grupo de alunos responsáveis pelo Robô. No segundo semestre de 2021 o grupo de pesquisa que antes apenas desenvolvia o robô, iniciou a concepção de duas novas máquinas, uma que operaria na Farmácia do HU e outra que seria usada pela equipe de Fisioterapia da UTI Adulto. Como apoio financeiro, o grupo contava com o Amigos da Poli, fundo patrimonial que financiou, desde o início, os projetos propostos por Professor Leopoldo. Além do aumento no número de projetos, dos recursos disponíveis veio também um crescimento no número de alunos interessados em inovação na saúde, politécnicos atraídos pelas mídias sociais de divulgação do Robo Hospitalar. Em agosto de 2021, tive a oportunidade de, juntamente com meus colegas, fundar a ZIMA - Soluções Médico Hospitalares[FOTO], que sintetiza todo esse apoio docente, financeiro e institucional em um grupo de extensão de alunos motivados a impactar a saúde através de tecnologia. Indo além do desenvolvimento de pesquisa, esse documento contempla a criação de um instituição criada por alunos que perpetuará o seu impacto na comunidade USP e na graduação dos Engenheiros Politécnicos oferecendo um espaço de experimentação e aprendizado em inovação da saúde.

Figura 1: Robô Hospi

Fonte: Business Wire

1 OBJETIVO

1.0.1 Robo Hospitalar / Hema-Bot

Dar continuidade ao Projeto piloto da parceria HU X POLI, Robô Hospitalar, fazendo melhorias técnicas e estéticas ao modelo proposto e fabricado anteriormente. Isso seria feito através da Terceira Versão do Robo, apelidado de Hema-BOT, nome inspirado célula sanguínea responsável pelo transporte de oxigênio e gás carbônico no corpo humano.

1.0.2 Ciclo Ergômetro

Fazer a concepção e prototipação de um equipamento de reabilitação capaz de ser usado por paciente acamados com saúde debilitada. O equipamento deve auxiliar o movimento ou resisti-lo dependendo da saúde do paciente, além disso o aparelho irá coletar dados como velocidade de rotação, quantidade de ciclos realizados e esforço médio. Em suma, será um ciclo ergometro com sistema de tração controlado(auxílio/resistência) que coleta e processa parâmetros importantes para mensurar a progressão do tratamento fisioterapêutico do paciente.

1.0.3 Automação da Farmácia / Golgi-Bot

Realizar a concepção e fabricação de uma máquina selecionadora de remédios. Tal equipamento seria semelhante a uma vending-machine[FOTO] porém de remédios. Ela contaria com um sistema que permite realizar o controle de estoque farmacológico, análise de estoque, registro de entrada e saída, autenticação via chave única e a programação de retirada automática dessas receitas. De maneira resumida, o equipamento faria a seleção dos remédios de maneira automática, mantendo-se assim registro de todas as operações feitas pela máquina assim como os responsáveis por recarregar ou retirar medicamentos, o que garante a segurança do processo e a rastreabilidade de falhas na seleção, além do ganho de segurança a ela poderá operar 24 horas permitindo assim que a equipe da farmácia

possa se dedicar integralmente ao cuidado do paciente.

2 MOTIVAÇÃO

A dinâmica de hospitais são tão complexas quanto a de fábricas ou indústrias. Existem uma rede grande de colaboradores, fornecedores(remédios, equipamentos e utilitários médicos), controle de estoque(material cirúrgico e farmacos), logística de transporte(ambulância) e prestação de serviço(consultas, exames e procedimentos cirúrgicos). Se tratando de Hospitais Públicos a otimização desses processos é fator determinante da qualidade do serviço e bom uso dos recursos(limitados) públicos. No HU-USP foi reportado, através do INTEC, a necessidade de tecnologias que resolvessem três problemas: Contaminação por contato com material laboratorial; Fisioterapia para paciente de mobilidade reduzida e Seleção de medicamentos.

2.1 Problemas

2.1.1 Contaminação por contato com amostras laboratoriais

Evidenciado pela pandemia, o risco de contaminação por inúmeras doenças pelo contato com amostras é algo que ocorre no Hospital Universitario, no qual um colaborador é designado especificamente ao trabalho de transportar as amostras [FOTO], do ponto de coleta(Pronto Socorro) até o laboratório. O envolvimento de uma terceira pessoa no processo, aquele que transporta o exame, aumenta o risco de espalhamento de patógenos e diminui a segurança sanitária do Hospital

2.1.2 Fisioterapia para paciente de mobilidade reduzida

Com o avanço da pandemia de Covid-19, inúmeros pacientes foram internados devido ao processo de entubação ou outras decorrências da doença. Essa imobilização induz perder massa muscular em 48 horas sem o estímulo motor, além de iniciar rápido esse processo se dá de forma aguda ao ponto de paciente que passam 1 ou 2 semanas acamados não conseguirem sustentar o peso do próprio corpo. Para resolver isso é necessário um

processo de fisio terapia lento, que primeiro depende da recuperação da consciência do paciente para iniciar tratamento, que recupera pouco a pouco a massa muscular, que irá durar semanas. Ou seja, para um paciente reber alta, é necessário que se recupere não só a patologia/lesão que o levou à imobilização como também a perda de massa muscular ocasionada por ela. Quanto mais tempo um paciente permanece na UTI mais gastos por paciente é empregado e menos pacientes são contemplados pelo sistema de Saúde público.

2.1.3 Seleção de medicamentos

Dentro do HU-USP, todos os dias são dedicados 5 horas de trabalho de 4 colaboradores para selecionar os remédios que serão administrados aos pacientes naquele dia. Esses profissionais são enfermeiros ou farmaceuticos que realizam o trabalho mecânico de retirar remédios de gavetas e inserí-los em sacos plasticos [FOTO]. Além de um desperdício de mão de obra qualificada, dado a exaustão do trabalho repetitivo, existe o risco do erro humano na medicação o que pode ser motivo de piora de quadro clinico do paciente ou até óbito.

2.2 Solução proposta

Após a indentificação desses problemas, a equipe de alunos(ZIMA) com os professores Leopoldo Yoshioka e Oswaldo Horikawa, foram até o hospital universitário[FOTO] à uma visita guiada pelo responsável de cada um dos desafios indenticados. Dr oscar fugita(Diretor do Núcleo de Inovação e Tecnologia (INTEC) e Urologista no Hospital Universitário da USP - SP), Dra. Alexandra siqueira(Coordenadora do Serviço de Fisioterapia no Hospital Universitário da USP) e Dra Valentina Porta(Diretora Técnica da Divisão de Farmácia do HU eProfessora da Faculdade de Ciências Farmacêuticas) guiaram os alunos e explicaram a dinâmica de cada um dos sistemas envolvidos e suas peculiaridades. Com essas informação os alunos, professores e profissionais da saúde chegaram em três conclusões sobre qual abordagem de engenharia usar em cada situação problema. Para a coleta de exames, seria desenvolvido um robô de mobilidade autônoma, para fisioterapia de mobilidade reduzida, um ciclo êrgometro para leitos de UTI e para a seleção de remédios, um selecionador automático de remédios.

Figura 2: 2ª Versão Robô Hospitalar - incompleta



Fonte: Escola Politécnica da Universidade de São Paulo

3 METODOLOGIA

3.0.1 Projetos

Na parte de concepção, foram realizadas reuniões quinzenais entre alunos, professores e colaboradores do HU, as quais serviam para a definição dos módulos eletrônicos, modelo mecânico e software a ser usado nos projetos. O grupo trabalhou com melhorias incrementais e versionamentos dos projetos, consolidando as melhorias técnicas e corrigindo falhas de projeto. Essas versões foram fabricadas e testadas em laboratório para validação de hipótese e avaliação da funcionalidade do protótipo. Foi parte do processo de concepção o contato constante com a equipe do Hospital Univesitário para garantir o desenvolvimento de um equipamento útil e funcional.

Toda parte de concepção e fabricação foi realizada pelos alunos e guiada pelos professores. Foram utilizados softwares de modelagem 3D para o desenvolvimento do CAD de cada projeto, Softwares como Altium Designer para concepção de Placas de Circuito Impresso(PCBs) para os módulos eletrônicos e linguagens como C++, python e frameworks como ROS para o desenvolvimento do software responsável pelo funcionamento inteligente dos equipamentos.

Para a aquisição de materiais, contratação de serviços e todos os custos relacionados ao projeto foram financiados pela Fundo Patrimonial Amigos Da Poli, o qual forneceu cerca de XXXX Reais através dos seus editais.

3.0.2 Gestão

Durante todo o processo de desenvolvimento, foi dos professores toda responsabilidade fiscal/financeira bem como a comunicação entre Escola Politécnica e HU, enquanto os alunos possuíam liberdade para desenvolver as atividades de projeto em termos de concepção, fabricação e teste, porém com feedback constante dos professores, médicos, farmacêuticos e Fisioterapeutas. Foram estabelecidas reuniões quinzenais entre Escola Politécnica e HU e reuniões semanais para cada área, mecânica, eletrônica e computação

feita entre alunos. Além dos encontros oficiais, reuniões extras de alinhamento, visitas técnicas e eventos foram realizando entre os alunos, professores e equipe do HU durante o processo.

4 ARQUITETURA DO PROJETO

O projeto da robô hospitalar, desde a seus primórdios, tem uma arquitetura de projeto dividida em grandes três áreas de atuação, Sistemas Mecânicos, Eletrônicos e Computacionais. Para essa segunda versão do robô hospitalar, a sua estrutura de organização foi mantida, dessa forma, cada uma das áreas têm seus respectivos objetivos, funções e deveres para o projeto como um todo. Por mais que tenha essa divisão, os membros da equipe, que cursam entre engenharia elétrica, mecânica e mecatrônico, sempre acabam atuando em mais de uma área, o que contribui para a dinamicidade do projeto.

A área **Sistemas Mecânicos** é responsável pelo desenvolvimento, fabricação de modelos físicos, usinagem das peças, além da manutenção do robô como um todo.

A área **Sistemas Eletrônicos** é destinada para a elaboração de todos os projetos da eletrônica embarcada, firmware, garantindo a alimentação, elétrica e segura, do robô e funcionamento dos componentes elétricos, em conjunto da transmissão de informação entre tais componentes e os sistemas computacionais.

A área **Sistemas Computacionais** tem como objetivo desenvolver algoritmos de controle, tomadas de decisão, construção de simuladores robóticas, visão computacional, desenvolvimento web e mobile.

Além dessas três áreas de atuação, há uma área não oficial de **design**, focada em divulgar a equipe e realizar a manutenção das redes sociais. (?)

4.1 Hardware

Os sistemas eletrônicos do projeto tem como missão, antes de tudo, garantir a locomoção, iluminação, sensoramento, telemetria modularizada e alimentação elétrica do robô hospitalar, sempre visando a precisão e a segurança. De maneira geral, os sistemas eletrônicos são divididos em duas frentes: Distribuição de energia e Módulos eletrônicos.

A frente de **distribuição de energia**, até Agosto de 2021, foi pouco explorada.

Como um todo no projeto, há somente uma bateria com uma tensão contínua elevada para alimentação total do robô e um regulador de tensão chaveado, que é necessário para diminuir a alta tensão da bateria e poder alimentar corretamente os módulos eletrônicos.

Por outro lado, a área dos **módulos eletrônicos** foi bem trabalhada no último ano. De maneira geral, é dividida em cinco grandes modelos eletrônicos: Sinalização, Percepção Externa, Telemetria, Controle e Interface com Usuário. Cada um desses módulos tem uma parte totalmente eletrônica, que é composta por uma placa de circuito impresso, e uma parte do firmware do ESP32 (?), o microcontrolador utilizado em cada uma das placas.

Por mais que cada placa tenha seu código único, muitas funções e bibliotecas são reutilizadas, por isso, há uma parte significativa dos códigos que foram feitos como bibliotecas e são usadas em todos os módulos.

4.2 Software

A área de computação da equipe, foi sem dúvidas a que mais se desenvolveu nos últimos seis meses. Na primeira versão do robô Hospitalar somente uma pessoa trabalhava, em contrapartida, na segunda versão três pessoas trabalham no software do robô, sendo duas delas inclusivamente para isso. De maneira geral, a confecção do software do robô hospitalar é dividida em cinco grandes frentes: Simulação, Algoritmos de Controle, Integração de Componentes, Visão Computacional e Desenvolvimento web e mobile.

A área de **Simulação**, feita com gazebo (?) sem dúvida a melhor desenvolvida e consolidada nos últimos seis meses, que já foi quase que concluída, foi subdividida em três grandes problemas: modelo simplificado do robô, universo de simulação e Implementação de sensores. Esses três problemas já foram devidamente desenvolvidos e hoje em dia sofrem poucas alterações.

A área de **Desenvolvimento de algoritmos de controle** pode ser subdividida em desenvolver dois grandes problemas: Odometria e Navegação. A odometria consiste em pegar os dados corretamente dos encóderes do robô e navegação em conseguir fazer o robô ir de um ponto a outro de forma autônoma. Ainda está sendo desenvolvida essa parte da equipe, porque é sem dúvida o core de todo o robô.

A área de **Integração de Componentes** consiste em utilizar a Jetson Nano (?) com os algoritmos desenvolvidos e estabelecer uma conexão com os sistemas embarcados.

A área de **Visão computacional** da equipe visa fazer o robô entender alguns objetos

de ambientes hospitalares a partir de uma câmera. Por conta de muito trabalho nas outras áreas e a pouca aplicabilidade dela para o projeto hoje, essa parte de desenvolvimento não está com pessoas alocadas oficialmente.

A área de **Desenvolvimento Web e Mobile**, uma área nova na equipe, que foi adicionada com a entrada de uma nova integrante para o grupo. Essa área está em um momento de pesquisas constantes, porém é bem promissora para um futuro breve.

Figura 3: Sistema completo - Robô Hospitalar (V2)

sistema_robo.png

Fonte: Elaborada pelo autor

Figura 4: Sistema Eletrônico - Robô Hospitalar (V2)

sistema_eletronico.png

Fonte: Elaborada pelo autor

Figura 5: Sistema Computacional - Robô Hospitalar (V2)

sistema_computacional.png

Fonte: Elaborada pelo autor

PARTE II

HARDWARE

5 DISTRIBUIÇÃO DE ENERGIA

Para o robô hospitalar, levando em consideração que tem uma série de equipamentos sensíveis, como a Jetson Nano (?), o computador embarcado do projeto, é de extrema importância ter uma fonte de energia estável e segura, para assim, não queimar eventualmente algum equipamento ou lesionar algum dos membros.

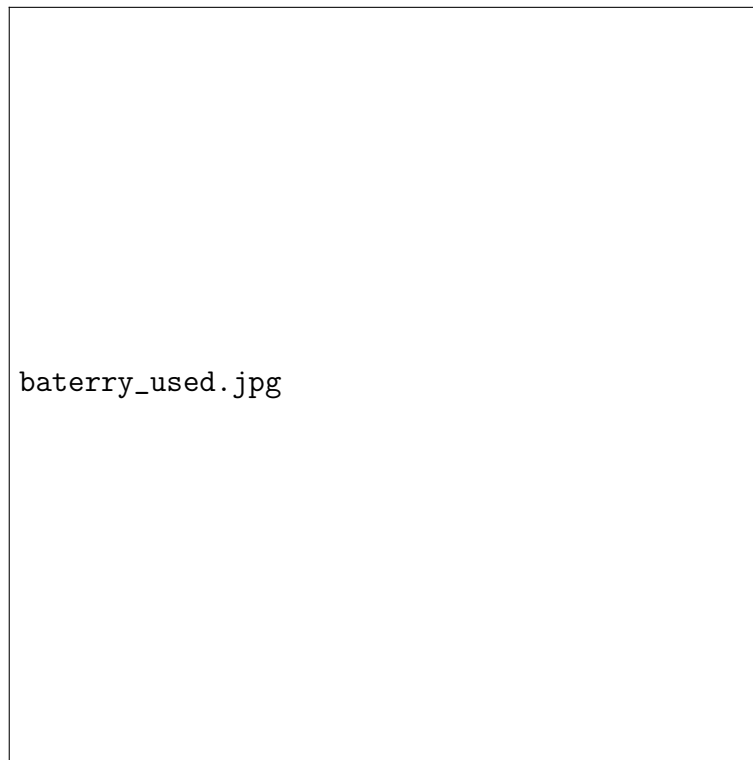
Para o projeto todo, somente uma bateria de alta tensão é utilizada e a partir de um regulador de tensão as tensões necessárias para todos os equipamentos é feita.

5.1 Bateria DC e Regulador de Tensão

No geral, como driver do motor usado é feito para um Hoverboard, para facilitar a montagem do robô, utilizamos baterias universais para Hoverboard e partir dela diminuimos a tensão para os demais parelho, como os ESP32 (?), que recebem 12V e a Jetson que recebe 5V.

Para diminuir a tensão de 36V baterias universais para Hoverboard, foi utilizado um regulador de tensão chaveado buck (?), que abaixaria a tensão da bateria para 12V e 5V, que é o suficiente para tanto os meus embarcado, quanto a Jetson Nano.

Figura 6: Bateria de Bateria Universal Hoverboard



Fonte: retirado de (?)

6 MODULOS EMBARCADOS

Figura 7: Módulos Embarcados - Robô Hospitalar (V2)



Fonte: Elaborada pelo autor

Os módulos embarcados, ou eletrônicos do Robô Hospitalar (V2) foram divididos por função específica, dessa forma, cada módulo tem um pequeno trabalho diante o robô todo. Dentre esses cinco módulos, um deles, como se fosse um líder ou mestre, tem função de

receber as informações dos outros módulos e se comunicar com o computador embarcado no robô, a Jetson Nano (?), Cada um desses módulos recebeu um nome específico, sendo eles: Sinalização, Telemetria, Percepção Externa, Interface com Usuário e Controle.

Nos primórdios, ela foi feita para facilitar a manutenção do robô e troca eventual de alguns dos módulos. Por mais que essa organização exige mais microcontroladores e aumenta a complexidade da confecção dos códigos embarcados, ainda se mostra bem interesse em termos de manutenção.

Por mais que cada módulo tenha sua característica que o torne único, há diversos parâmetros e componentes que todos os módulos tiveram ou têm em comum entre todos. Dentre eles, tantos nos protótipos e versões oficiais, destaca-se o microcontrolador, as dimensões da placa, encapsulamento e protocolos de comunicação.

O microcontrolador padrão escolhido para todos os módulos foi ESP32 (?), que é dentre os mais famosos, como Arduino (?), o que apresenta a maior capacidade de processamento e ainda é mais barato que os convencionais (?), além de já ter um módulo bluetooth embutido. Por conta desses motivos que adotamos o ESP32 como micro controlador padrão.

Para realizar a comunicação entre os ESP32 (?), a princípio, seria utilizado somente o protocolo I2C (?) para haver a troca de mensagem entre os módulos. Todas as placas fresadas, ou seja, que ainda são protótipos, só tinham esse protocolo de comunicação. No entanto, por conta da instabilidade do protocolo I2C, foi adicionado conexões SPI (?), que é muito mais rápido que o convencional e ainda tem mais instabilidade no esquemático.

Além disso, todas as placas têm a sua dimensão constante. Todos os protótipos tinham, dimensões de 120x60mm, porém, conforme fomos adicionando coisas novas no projeto, como componentes e conectores, se viu a necessidade de mudar essas dimensões, por conta disso o tamanho padrão das placas oficiais foram 100x65mm.

Figura 8: Microcontrolador ESP32



Fonte: Foto disponibilizada por saravati

6.1 Controle

A placa de controle é a principal do robô. Ela foi idealizada, desde da primeira versão do robô hospitalar, para realizar o controle dos drivers dos motores, receber e enviar mensagens para os outros dispositivos embarcados e se comunicar diretamente com a Jetson (?), um computador embarcado no robô feita para processar dados.

6.1.1 Placa

Como módulo principal, existem muitas minúcias que precisamos tomar ao projetá-las. A placa de controle, para segunda versão do robô hospitalar, com objetivo de evitar problemas e realizar testes, teve duas versões: um protótipo, que já foi finalizado, e uma versão oficial, ainda em desenvolvimento.

6.1.1.1 Protótipo

O protótipo das placas foi refeito três vezes. Como se trata de uma placa fresada, pode ser refeita no próprio laboratório do professor orientador. O projeto da placa eletrônica, assim como o de todos os módulos, foi dividido em esquemático e Printed Circuit board (PCB) .

Para design do hardware do módulo de controle foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo está disponibilizado em (?) e todos os componentes usado nesse protótipo podem ser visto na tabela ??.

Tabela 1: Componentes Utilizados na placa de Controle - Protótipo

Component	Description	Designator	Footprint	Quantity
4.7K	RES 4.7K OHM 1/4W 5% CARBON FILM	R6, R7	RES 4.7K 1/4W CARBON FILM	2
47K	RES 4.7K OHM 1/4W 5% CARBON FILM	R10	RES 4.7K 1/4W CARBON FILM	1
68k	RES 4.7K OHM 1/4W 5% CARBON FILM	R11	RES 4.7K 1/4W CARBON FILM	1
BC549	TRANS NPN 30V 0.1A TO-92	Q2	TO92	1
IRFZ44NPBF	MOSFET (N-Channel)	Q1	TO254P483X1016X1994-3P	1
KK_2.54_2vias	Conector KK 2.54mm 2 vias	CON1, CON3, CON6, CON7	KK_2VIAS_180°	4
KK_2.54_3vias	Conector KK 2.54mm 3 vias	CON5	KK_3vias_180°	1
KK_2.54_4vias	Conector KK 2.54mm 4 vias	CON4	KK_4vias_180°	1
KK_2.54_5vias	Conector KK 2.54mm 5 vias	CON2, CON8	KK_5vias_180°	2
LED 5MM RED	LED 5MM RED	D1	LED 5MM RED	1
LM741IN	Integrated Circuit	IC1	DIP762W56P254L920H508Q8N	1
microcontrolador	microcontrolador com moculo bluetoth e wifi	U1	ESP32_Desvikit_v1	1
RES 470R 1/4W CARBON FILM	RES 470R OHM 1/4W 5% CARBON FILM	R1, R2, R4, R5, R8, R9	RES 470R 1/4W CARBON FILM	6

Fonte: Elaborado pelo autor

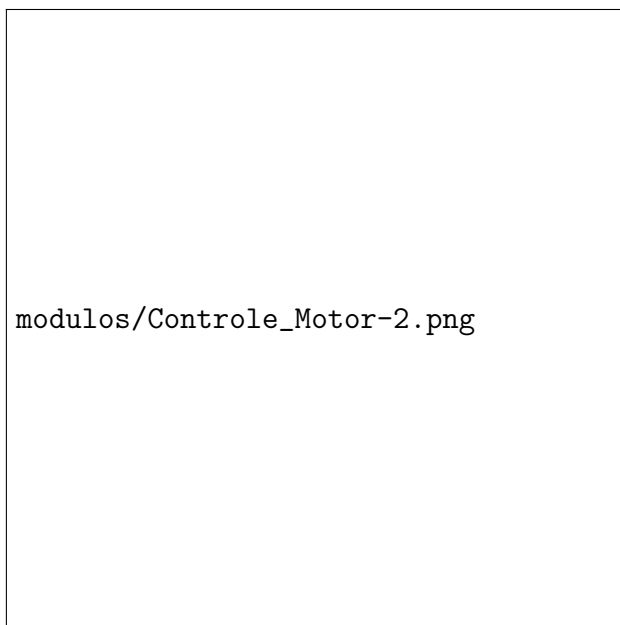
Dentre os componentes usados, destaca-se o transistor mosfet IRFZ44NPBF (?), utilizado no circuito da chave do driver e o amplificador operacional LM741 (?), usado para envio de pequenos sinais no circuito para o driver.

Figura 9: Protótipo placa de Controle - Esquemático principal



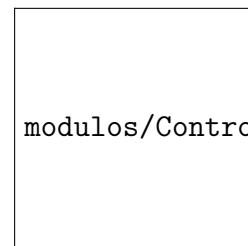
Fonte: Elaborado pelo autor no software Altium Design(?)

Figura 10: Protótipo Controle - PCB 2D



modulos/Controle_Motor-2.png

Figura 11: Protótipo Controle - PCB 3D



modulos/Controle_Motor.png

Fonte: Elaborado pelo autor no software Altium Design(?)

6.1.1.2 Oficial

A placa oficial de controle ainda não foi fabricada. Por se tratar de uma placa mais profissional, ela será mandada para ser feita para uma empresa privada ainda não escolhida, não na própria Universidade São Paulo. Assim como o protótipo, o projeto como um todo foi dividido em um esquemático e uma PCB.

Figura 12: Placa de Controle - Esquemático principal



Fonte: Elaborado pelo autor no software Altium Design(?)

Para design do hardware do módulo de controle foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo está disponibilizado em (?) e todos os componentes usado nessa versão oficial podem ser visto na tabela ??.

Tabela 2: Componentes Utilizados na placa de Controle

Component	Description	Designator	Footprint	Quantity
KK_2.54_5vias	Conector KK 2.54mm 5 vias	CON1, CON7, CON10, CON11	KK_5vias_180°	4
KK_2.54_4vias	Conector KK 2.54mm 4 vias	CON2	KK_4vias_180°	1
KK_2.54_2vias	Conector KK 2.54mm 2 vias	CON3, CON5, CON6, CON8	KK_2VIAS_180°	4
KK_2.54_6vias	Conector KK 2.54mm 6 vias	CON4	KK_6vias_180°	1
KK_2.54_3vias	Conector KK 2.54mm 3 vias	CON9	KK_3vias_180°	1
LED 3MM RED	LED 3MM RED	D1	LED RED	1
LM741IN	Integrated Circuit	IC1	DIP762W50P254L712H450Q6N	1
Trans BC817	Transistor BJT NPN BC817-25-7-F	Q1	SOT96P240X110-3N	1
IRFZ44NPBF	MOSFET (N-Channel)	Q2	TO254P483X1016X1994-3P	1
4k7	RES 1206 5%	R1, R2	RESC3216X60N	2
680R	Resistor	R3	RESC3216X60N	1
2k2	Resistor	R4, R5	RESC3216X60N	2
47K	RES 1206 5%	R6	RESC3216X60N	1
68k	0805	R7	RESISTOR 0805	1
22R	RES 1206 5%	R8	RESC3216X60N	1
100k	RES 1206 5%	R9, R10	RESC3216X60N	2
microcontrolador	microcontrolador com modulo bluetooth e wifi	U1	ESP32_Desvikit_v1	1

Fonte: Elaborado pelo autor

Em relação ao protótipo, pouco mudou da lógica por trás do circuito, a principal diferença é que o modelo oficial utiliza todos componentes com encapsulamento SMD (?). Além disso, foi adicionado o protocolo de comunicação SPI também, como já foi comentado no início do capítulo.

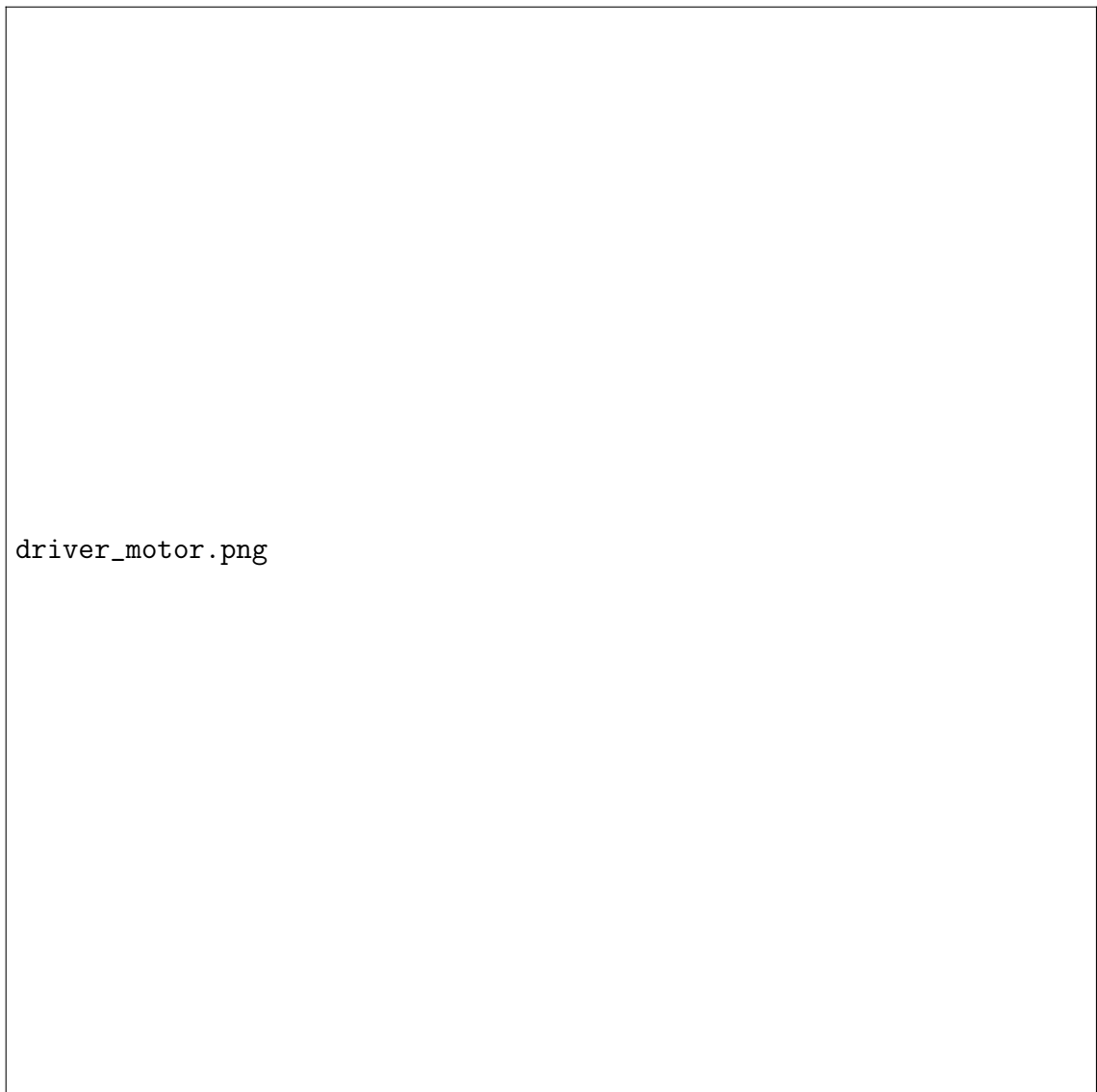
A placa de circuito impresso ainda está em desenvolvimento.

6.1.2 Driver do Motor

O motor do robô hospitalar é trifásico. A construção de uma ESC para controle adequado e preciso do motor é um trabalho muito minucioso e preciso. Por conta disso, como decisão de projeto, os integrantes do grupo preferiram comprar um driver que conseguisse realizar o controle de segurança. A imagem do driver pode ser vista na figura ??.

Entretanto, ainda há necessidade de controlar adequadamente esse driver, o qual é a função da placa de controle.

Figura 13: Driver de Controle do Motor



driver_motor.png

Fonte: imagem retirada de (?)

6.2 Percepção Externa

A placa de Percepção Externa é a mais complexa dentre todas. Ela foi idealizada, desde da primeira versão do robô hospitalar, para realizar a leitura de todos sensores de distância e de mensurar o estado do robô a partir de um sensor de inércia. Além de claro, como já mencionado antes, enviar e receber informações da placa de controle principal. De maneira geral, ela foi pensando em coletar dados do mundo externo para o robô.

6.2.1 Placa

Como módulo, existem muitas minúcias que precisamos tomar ao projetá-las. A placa de Percepção Externa, para segunda versão do robô hospitalar, com objetivo de evitar problemas e realizar testes, teve duas versões: um protótipo, que já foi finalizado, e uma versão oficial, ainda em desenvolvimento.

6.2.1.1 Protótipo

O protótipo das placas foi refeito duas vezes. Como se trata de uma placa fresada, pode ser refeita no próprio laboratório do professor orientador. A primeira placa não funcionou porque a impressão ficou ruim. O projeto da placa eletrônica, assim como o de todos os módulos, foi dividido em esquemático e Printed Circuit board (PCB) ou placa de circuito impresso.

Para design do hardware do módulo de controle foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo está disponibilizado em (?) e todos os componentes usado nesse protótipo podem ser visto na tabela ??.

Tabela 3: Componentes Utilizados na placa de Percepção Externa - Protótipo

Component	Description	Designator	Footprint	Quantity
100nF	CAP CER DISK 100NF 50V	C1	CAP CER DISK 100NF 50V	1
Cap. Elet. 1uF 50V	Aluminum Organic Polymer Capacitors 16volts 470uF ESR 10mohm	C2	Cap. Elet. 470uF 16V	1
KK.2.54,6vias	Conector KK 2.54mm 6 vias	CON1, CON2, CON3, CON4, CON5, CON6, CON7, CON8, CON9, CON15	KK.5vias,180°	10
KK.2.54,5vias	Conector KK 2.54mm 5 vias	CON10	KK.5vias,180°	1
KK.2.54,4vias	Conector KK 2.54mm 4 vias	CON11	KK.4vias,180°	1
KK.2.54,8vias	Conector KK 2.54mm 8 vias	CON12	KK.8vias,180°	1
KK.2.54,2vias	Conector KK 2.54mm 2 vias	CON13, CON14	KK.2vias,180°	2
LED 5MM RED	LED 5MM RED	D1	LED 5MM RED	1
LM317	Integrated Circuit	IC1	TO254P467X1016X1971-3P	1
BC549	TRANS NPN 30V 0.1A TO-92	Q1	TO92	1
RES 470R 1/4W CARBON FILM	RES 470R OHM 1/4W 5% CARBON FILM	R4, R5, R8, R9, R10	RES 470R 1/4W CARBON FILM	5
4.7K	RES 4.7K OHM 1/4W 5% CARBON FILM	R6, R7	RES 4.7K 1/4W CARBON FILM	2
microcontrolador	microcontrolador com modulo bluetooth e wifi	U1	ESP32-Devkit-v1	1

Fonte: Elaborado pelo autor

Dentre os componentes usados, destaca-se o regulador de tensão LM317 (?). Além disso, como já mencionado antes, também usamos o ESP32 (?) como microcontrolador

Figura 14: Protótipo placa de Percepção Externa - Esquemático principal

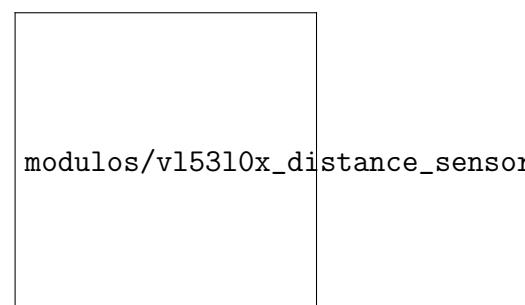
modulos/Percepo_Externa-1.png

Fonte: Elaborado pelo autor no software Altium Design(?)

do módulo.

Os sensores de distância escolhidos para o robô foram os vl53l0x (?), e para fins de simplificação do circuito, foi escolhido um módulo que já tivesse um circuito pré realizado. Esse sensor consegue alcançar até dois metros de distância e é bem pequeno, por isso foi escolhido. Além disso, o sensor de inércia, que é um módulo com dois

Figura 15: Sensor de distância vl53l0x



sensores que consegue agir como um acelerômetro. O módulo usado foi o Mpu-9250 (?).

Figura 16: Protótipo Percepção Externa - PCB 2D

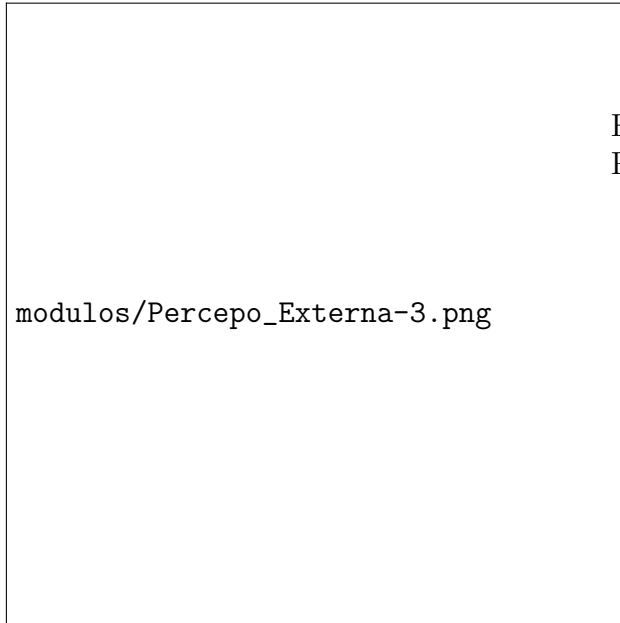
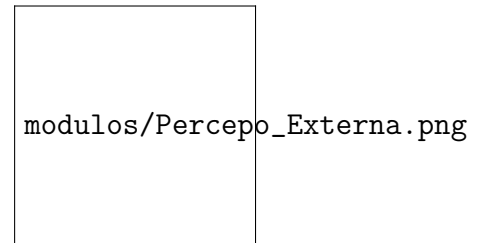


Figura 17: Protótipo Percepção Externa - PCB 3D



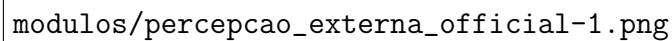
Fonte: Elaborado pelo autor no software Altium Design(?)

A partir do esquema elétrico que foi feito e do desenho da PCB, a placa de circuito impresso, em uma única camada (Single Layer) e dimensões de 120x60mm. As visões 2D e 3D podem ser vista na figura ?? e ??.

6.2.1.2 Oficial

A placa oficial de Percepção Externa ainda não foi fabricada. Por se tratar de uma placa mais profissional, ela será mandada para ser feita para uma empresa privada ainda não escolhida, não na própria Universidade São Paulo. Assim como o protótipo, o projeto como um todo foi dividido em um esquemático e uma PCB.

Figura 18: placa de Percepção Externa - Esquemático principal



modulos/percepcao_externa_oficial-1.png

Fonte: Elaborado pelo autor no software Altium Design(?)

Para design do hardware do módulo de Percepção Externa foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo

Tabela 4: Componentes Utilizados na placa de Percepção Externa

Component	Description	Designator	Footprint	Quantity
100nF	CAP CER DISK 10uF 50V	C1	CAP CER DISK 10uF 50V	1
Cap. Elet. 1uF 50V	Aluminum Organic Polymer Capacitors 16volts 470uF ESR 10mohm	C2	Cap. Elet. 470uF 16V	1
KK_2.54_2vias	Conector KK 2.54mm 2 vias	CON1, CON3, CON4	KK_2VIAS_180°	3
KK_2.54_6vias	Conector KK 2.54mm 6 vias	CON2, CON8, CON9, CON10, CON11, CON12, CON13, CON14, CON15, CON16, CON17	KK_6vias_180°	11
KK_2.54_5vias	Conector KK 2.54mm 5 vias	CON5	KK_5vias_180°	1
KK_2.54_4vias	Conector KK 2.54mm 4 vias	CON6	KK_4vias_180°	1
KK_2.54_8vias	Conector KK 2.54mm 8 vias	CON7	KK_8vias_180°	1
LED 3MM RED	LED 3MM RED	D1	LED RED	1
LM317	Integrated Circuit	IC1	TO254P467X1016X1971-3P	1
Trans BC817	Transistor BJT NPN BC817-25-7-F	Q1	SOT96P240X110-3N	1
240k	RES 1206 5%	R1	RESC3216X60N	1
390k	RES 1206 5%	R2	RESC3216X60N	1
680R	Resistor	R3	RESC3216X60N	1
2k2	Resistor	R4, R5	RESC3216X60N	2
4k7	RES 1206 5%	R6, R7	RESC3216X60N	2
microcontrolador	microcontrolador com modulo bluethoth e wifi	U1	ESP32_Desvikit_v1	1

Fonte: Elaborado pelo autor

está disponibilizado em (?) e todos os componentes usado nessa versão oficial podem ser visto na tabela ??.

Em relação ao protótipo, pouco mudou da lógica por trás do circuito, a principal diferença é que o modelo oficial utiliza todos componentes com encapsulamiento SMD (?) e no lugar de 6 sensores de distância, serão usados 10. Além disso, foi adicionado o protocolo de comunicação SPI também, como já foi comentado no início do capítulo.

A placa de circuito impresso ainda está em desenvolvimento.

6.3 Sinalização

A placa de Sinalização é a mais importante esteticamente para o robô. Ela foi idealizada, desde da primeira versão do robô hospitalar, para realizar a iluminação e comunicação por luz (como faróis). De maneira geral ele controla uma série de leds rgbs endereçáveis e strobos automotivos. Além de, claro, como já mencionado antes, enviar e receber informações da placa de controle principal.

6.3.1 Placa

Como módulo, existem muitas minúcias que precisamos tomar ao projetá-las. A placa de Sinalização, para segunda versão do robô hospitalar, com objetivo de evitar problemas e realizar testes, teve duas versões: um protótipo, que já foi finalizado, e uma versão oficial, ainda em desenvolvimento.

6.3.1.1 Protótipo

O protótipo da placa de Sinalização foi refeito uma única vez. Como se trata de uma placa fresada, pode ser refeita no próprio laboratório do professor orientador. O projeto da placa eletrônica, assim como o de todos os módulos, foi dividido em esquemático e Printed Circuit board (PCB) ou placa de circuito impresso.

Para design do hardware do módulo de Sinalização foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo está disponibilizado em (?) e todos os componentes usado nesse protótipo podem ser visto na tabela ??.

Tabela 5: Componentes Utilizados na placa de Sinalização - Protótipo

Component	Description	Designator	Footprint	Quantity
Conversor de nível lógico	Conversor de nível lógico	U2	Conversor de nível lógico	1
Cap. Elet. 470 uF 16V	Aluminum Organic Polymer Capacitors 16volts 470uF ESR 10mohm	C1, C2	Cap. Elet. 470uF 16V	2
KK.2.54.3vias	Conector KK 2.54mm 3 vias	CON1, CON3, CON7, CON8	KK_3vias_180°	4
KK.2.54.5vias	Conector KK 2.54mm 5 vias	CON2	KK_5vias_180°	1
KK.2.54.4vias	Conector KK 2.54mm 4 vias	CON4, CON10	KK_4vias_180°	2
KK.2.54.2vias	Conector KK 2.54mm 2 vias	CON5, CON6, CON9	KK_2VIAS_180°	3
LED 5MM RED	LED 5MM RED	D1	LED 5MM RED	1
BC549	TRANS NPN 30V 0.1A TO-92	Q1	TO92	1
RES 470R 1/4W CARBON FILM	RES 470R OHM 1/4W 5% CARBON FILM	R2	RES 470R 1/4W CARBON FILM	1
RES 1/4W CARBON FILM	RES ? OHM 1/4W 5% CARBON FILM	R3, R4	RES 10K 1/4W CARBON FILM	2
4.7K	RES 4.7K OHM 1/4W 5% CARBON FILM	R5, R6	RES 4.7K 1/4W CARBON FILM	2
microcontrolador	microcontrolador com modulo bluetooth e wifi	U1	ESP32_Desvikit_v1	1

Fonte: Elaborado pelo autor

Dentre os componentes usados, destaca-se os conversores de nível lógico bidirecional, sendo que cada conversor é um módulo com 8 transistores BSS138 (?) . Além disso, como já mencionado antes, também usamos o ESP32 (?) como microcontrolador do módulo.

Figura 19: Protótipo placa de Sinalização - Esquemático principal

modulos/Sinalizacao-1.png

Fonte: Elaborado pelo autor no software Altium Design(?)

Para a iluminação do robô, foram escolhidos os leds rgb endereçável ws2812b (?), que, com um conjunto de três fitas com 80 leds em casa, iluminam o robô por completo. Além disso, para enviar sinais mais de sinalização, foram usados 4 strobos automotivos Ajk 4 (?).

A partir do esquema elétrico que foi feito e do desenho da PCB, a placa de circuito impresso, em uma única

Figura 20: Sensor de distância vl53l0x

modulos/led_rgb_end.png

Fonte: Foto disponibilizada por botnroll

Figura 21: Protótipo Sinalização - PCB 2D

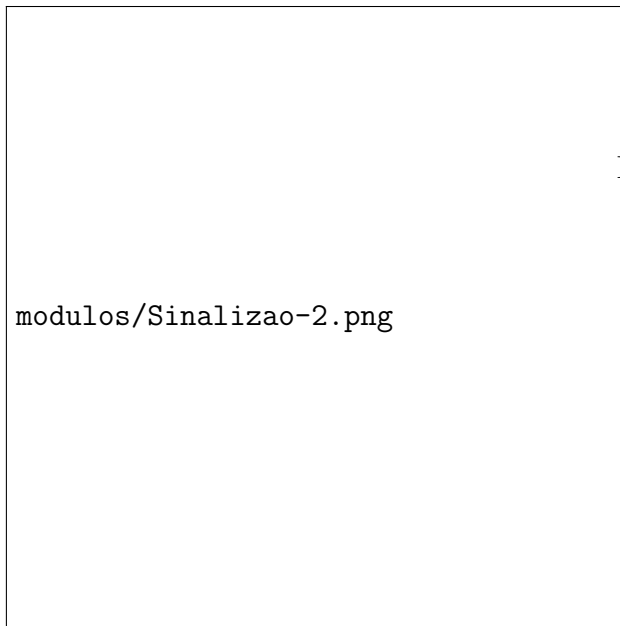
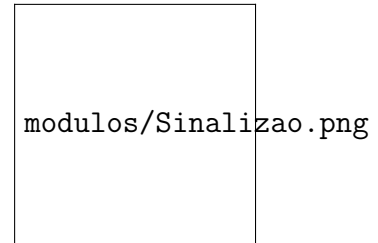


Figura 22: Protótipo Sinalização - PCB 3D



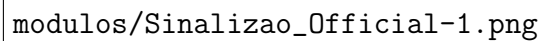
Fonte: Elaborado pelo autor no software Altium Design(?)

camada (Single Layer) e dimensões de 120x60mm. As visões 2D e 3D podem ser vista na figura ?? e ??.

6.3.1.2 Oficial

A placa oficial de Sinalização ainda não foi fabricada. Por se tratar de uma placa mais profissional, ela será mandada para ser feita para uma empresa privada ainda não escolhida, não na própria Universidade São Paulo. Assim como o protótipo, o projeto como um todo foi dividido em um esquemático e uma PCB.

Figura 23: placa de Sinalização - Esquemático principal



modulos/Sinalizao_Official-1.png

Fonte: Elaborado pelo autor no software Altium Design(?)

Para design do hardware do módulo de Sinalização foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo está

disponibilizado em (?) e todos os componentes usado nessa versão oficial podem ser visto na tabela ??.

Tabela 6: Componentes Utilizados na placa de Sinalização

Comment	Description	Designator	Footprint	Quantity
1000uF 50V	Capacitor Eletrolitico 1000uF 50V	C1, C2, C3	CAPPRD500W62D1300H2650	3
KK_2.54.5vias	Conector KK 2.54mm 5 vias	CON1	KK_5vias_180°	1
KK_2.54.4vias	Conector KK 2.54mm 4 vias	CON2, CON7	KK_4vias_180°	2
KK_2.54.2vias	Conector KK 2.54mm 2 vias	CON3, CON9, CON11, CON12	KK_2VIAS_180°	4
KK_2.54.3vias	Conector KK 2.54mm 3 vias	CON4, CON5, CON6, CON8	KK_3vias_180°	4
KK_2.54.6vias	Conector KK 2.54mm 6 vias	CON10	KK_6vias_180°	1
LED 3MM RED	LED 3MM RED	D1	LED RED	1
Trans BC817	Transistor BJT NPN BC817-25-7-F	Q1, Q2	SOT96P240X110-3N	2
4K7	Resistor	R1, R2	RESC3216X60N	2
1K	Resistor	R3	RESC3216X60N	1
2K2	Resistor	R4, R7	RESC3216X60N	2
680R	Resistor	R5	RESC3216X60N	1
Resistor 1206	Resistor	R6	RESC3216X60N	1
microcontrolador	microcontrolador com moculo bluetoth e wifi	U1	ESP32_Desvikit_v1	1
Conversor de nível logico	U2	Conversor de nível lógico	Conversor de nível logico	

Fonte: Elaborado pelo autor

Em relação ao protótipo, pouco mudou da lógica por trás do circuito, a principal diferença é que o modelo oficial utiliza todos componentes com encapsulamento SMD (?). Além disso, foi adicionado o protocolo de comunicação SPI também, como já foi comentado no início do capítulo.

Figura 24: Sinalização - PCB 2D

modulos/Sinalizao_Official-5.png

Figura 25: Sinalização - PCB 3D

modulos/Sinalizao_Official.png

Fonte: Elaborado pelo autor no software Altium Design(?)

A placa de circuito impresso está completa, porém ainda não foi feito o pedido oficial de fabricação.

6.4 Telemetria

A placa de Telemetria foi idealizada, desde da primeira versão do robô hospitalar, para realizar a avaliação funcional do robô, ou seja, ela tem como missão coletar dados, como temperatura, corrente que circula nos motores, tensão da bateria etc, para fins de conseguir mensurar o estado do robô, identificar problemas e principalmente facilitar a manutenção.

6.4.1 Placa

Como módulo, existem muitas minúcias que precisamos tomar ao projetá-las. A placa de Telemetria, para segunda versão do robô hospitalar, com objetivo de evitar problemas e realizar testes, teve duas versões: um protótipo, que já foi finalizado, e uma versão oficial, ainda em desenvolvimento.

6.4.1.1 Protótipo

O protótipo da placa de Telemetria foi refeita uma vez. Como se trata de uma placa fresada, pode ser refeita no próprio laboratório do professor orientador. O projeto da placa eletrônica, assim como o de todos os módulos, foi dividido em esquemático e Printed Circuit board (PCB) ou placa de circuito impresso.

Para design do hardware do módulo de Telemetria foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo está disponibilizado em (?) e todos os componentes usado nesse protótipo podem ser visto na tabela ??.

Tabela 7: Componentes Utilizados na placa de Telemetria - Protótipo

Component	Description	Designator	Footprint	Quantity
KK.2.54.2vias	Conector KK 2.54mm 2 vias	CON1, CON4, CON7, CON8	KK_2VIAS_180°	4
KK.2.54.4vias	Conector KK 2.54mm 4 vias	CON2, CON6	KK_4vias_180°	2
KK.2.54.3vias	Conector KK 2.54mm 3 vias	CON3, CON9	KK_3vias_180°	2
KK.2.54.5vias	Conector KK 2.54mm 5 vias	CON5	KK_5vias_180°	1
LED 5MM RED	LED 5MM RED	D1	LED 5MM RED	1
BC549	TRANS NPN 30V 0.1A TO-92	Q1	TO92	1
RES 470R 1/4W CARBON FILM	RES 470R OHM 1/4W 5% CARBON FILM	R1, R2, R3, R8	RES 470R 1/4W CARBON FILM	4
4.7K	RES 4.7K OHM 1/4W 5% CARBON FILM	R4, R7	RES 4.7K 1/4W CARBON FILM	2
1M6	RES 1.6M OHM	R5	RESISTOR 0603	1
140K	RES 140k OHM	R6	RESISTOR 0805	1
microcontrolador	microcontrolador com modulo bluetoth e wifi	U1	ESP32_Desvikit_v1	1

Fonte: Elaborado pelo autor

Dentre os componentes usados, nenhum muito diferente foi utilizado. Para mensurar a tensão, somente um divisor de tensão foi usado. Além disso, como já mencionado antes,

Figura 26: Protótipo placa de Telemetria - Esquemático principal

modulos/Telemetria-1.png

Fonte: Elaborado pelo autor no software Altium Design(?)

também usamos o ESP32 (?) como microcontrolador do módulo.

Para medir a temperatura e umidade dentro do robô foi usado o sensor DHT22 (?) e o sensor de corrente linear ACS712 (?). Sensores, como o de bateria, foram feitos na com resistores e capacitores.

A partir do esquema elétrico que foi feito e do desenho da PCB, a placa de circuito impresso, em uma única

Figura 27: Sensor de temperatura e umidade dht 22

modulos/dht22_sensor.jpg

Figura 28: Protótipo Telemetria - PCB 2D

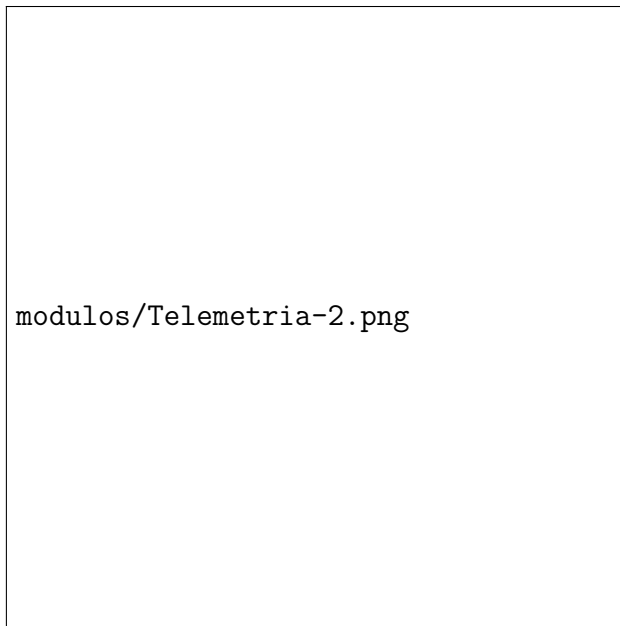
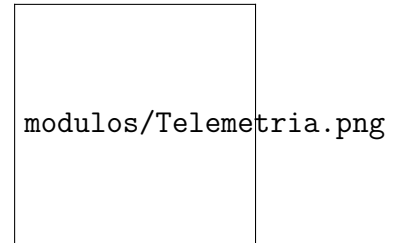


Figura 29: Protótipo Telemetria - PCB 3D



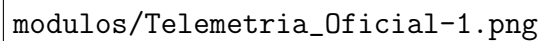
Fonte: Elaborado pelo autor no software Altium Design(?)

camada (Single Layer) e dimensões de 120x60mm. As visões 2D e 3D podem ser vista na figura ?? e ??.

6.4.1.2 Oficial

A placa oficial de Telemetria ainda não foi fabricada. Por se tratar de uma placa mais profissional, ela será mandada para ser feita para uma empresa privada ainda não escolhida, não na própria Universidade São Paulo. Assim como o protótipo, o projeto como um todo foi dividido em um esquemático e uma PCB.

Figura 30: Placa de Telemetria - Esquemático principal

A imagem mostra o esquemático principal da placa de Telemetria. O conteúdo da imagem não é visível, apenas o nome do arquivo é exibido.

modulos/Telemetria_Oficial-1.png

Fonte: Elaborado pelo autor no software Altium Design(?)

Para design do hardware do módulo de Telemetria foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo está

disponibilizado em (?) e todos os componentes usado nessa versão oficial podem ser visto na tabela ??.

Tabela 8: Componentes Utilizados na placa de Telemetria

Component	Description	Designator	Footprint	Quantity
KK_2.54_2vias	Conector KK 2.54mm 2 vias	CON1, CON3, CON4, CON5	KK_2VIAS_180°	4
KK_2.54_6vias	Conector KK 2.54mm 6 vias	CON2	KK_6vias_180°	1
KK_2.54_5vias	Conector KK 2.54mm 5 vias	CON6	KK_5vias_180°	1
KK_2.54_4vias	Conector KK 2.54mm 4 vias	CON7, CON8, CON11	KK_4vias_180°	3
KK_2.54_3vias	Conector KK 2.54mm 3 vias	CON9, CON10, CON12, CON13	KK_3vias_180°	4
LED 3MM RED	LED 3MM RED	D1	LED RED	1
Trans BC817	Transistor BJT NPN BC817-25-7-F	Q1	SOT96P240X110-3N	1
680R	Resistor	R1	RESC3216X60N	1
2k2	Resistor	R2, R3	RESC3216X60N	2
4k7	Resistor	R4, R5	RESC3216X60N	2
10k	Resistor	R6, R7	RESC3216X60N	2
1M6	0603	R8, R10	RESISTOR 0603	2
140K	0805	R9, R11	RESISTOR 0805	2
microcontrolador	microcontrolador com modulo bluethoth e wifi	U1	ESP32_Desvikit_v1	1

Fonte: Elaborado pelo autor

Em relação ao protótipo, pouco mudou da lógica por trás do circuito, a principal diferença é que o modelo oficial utiliza todos componentes com encapsulamento SMD (?). Além disso, foi adicionado o protocolo de comunicação SPI também, como já foi comentado no início do capítulo.

Figura 31: Telemetria - PCB 2D

modulos/Telemetria_Oficial-5.png

Figura 32: Telemetria - PCB 3D

modulos/Telemetria_Oficial.png

Fonte: Elaborado pelo autor no software Altium Design(?)

A placa de circuito impresso está completa, porém ainda não foi feito o pedido oficial de fabricação.

6.5 Interface com Usuário

A placa de Interface com o usuário é uma das que deram mais problemas. Ela foi idealizada, desde da primeira versão do robô hospitalar, para realizar a leitura de cartões, para retirada de entregas do robô hospitalar, abertura e controle do motor das quatro portas do robô hospitalar, além, é claro, de enviar informações para o módulo de controle.

6.5.1 Placa

Como módulo, existem muitas minúcias que precisamos tomar ao projetá-las. A placa de Interface com o usuário, para segunda versão do robô hospitalar, com objetivo de evitar problemas e realizar testes, teve duas versões: um protótipo, que já foi finalizado, e uma versão oficial, ainda em desenvolvimento.

6.5.1.1 Protótipo

O protótipo das placas foi refeito três vezes. Como se trata de uma placa fresada, pode ser refeita no próprio laboratório do professor orientador. A primeira placa não funcionou porque a impressão ficou ruim. O projeto da placa eletrônica, assim como o de todos os módulos, foi dividido em esquemático e Printed Circuit board (PCB) .

Para design do hardware do módulo de Interface com Usuário foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo está disponibilizado em (?) e todos os componentes usado nesse protótipo podem ser visto na tabela ??.

Tabela 9: Componentes Utilizados na placa de Interface com Usuário - Protótipo

Component	Description	Designator	Footprint	Quantity
KK.2.54.8vias	Conector KK 2.54mm 8 vias	CON1, CON4	KK_8vias_180°	2
KK.2.54.3vias	Conector KK 2.54mm 3 vias	CON2	KK_3vias_180°	1
KK.2.54.5vias	Conector KK 2.54mm 5 vias	CON3	KK_5vias_180°	1
KK.2.54.2vias	Conector KK 2.54mm 2 vias	CON5, CON8, CON10, CON13	KK_2VIAS_180°	4
KK.2.54.4vias	Conector KK 2.54mm 4 vias	CON6, CON9, CON11	KK_4vias_180°	3
LED 5MM RED	LED 5MM RED	D1, D4	LED 5MM RED	2
1N4007-E3.53	Diode	D2, D3	DIOAD1405W89L465D235	2
Fuse 5x20	Fuse	F1	Fuse 5x20	1
4N25	Integrated Circuit	IC1	DIP762W50P254L712H450Q6N	1
rele T73	Relay or Contactor	K1	JQC3FT731Z12VDC	1
BC549	TRANS NPN 30V 0.1A TO-92	Q1, Q2, Q3	TO92	3
RES 470R 1/4W CARBON FILM	RES 470R OHM 1/4W 5% CARBON FILM	R1, R2, R5, R6, R7, R8, R9, R10, R11	RES 470R 1/4W CARBON FILM	9
4.7K	RES 4.7K OHM 1/4W 5% CARBON FILM	R3, R4	RES 4.7K 1/4W CARBON FILM	2
microcontrolador	microcontrolador com modulo bluetoth e wifi	U1	ESP32_Desvikit_v1	1

Fonte: Elaborado pelo autor

Dentre os componentes usados, destaca-se o rele T73 (?) e o optoacoplador 4N25 (?), utilizado nas solenoides da porta. Além disso, como já mencionado antes, também usamos o ESP32 (?) como microcontrolador do módulo.

Figura 33: Protótipo placa de Interface com Usuário - Esquemático principal



Fonte: Elaborado pelo autor no software Altium Design(?)

A partir do esquema elétrico que foi feito e do desenho da PCB, a placa de circuito impresso, em uma única camada (Single Layer) e dimensões de 120x60mm. As visões 2D e 3D podem ser vista na figura ?? e ??.

Figura 34: Protótipo Interface com Usuário
- PCB 2D

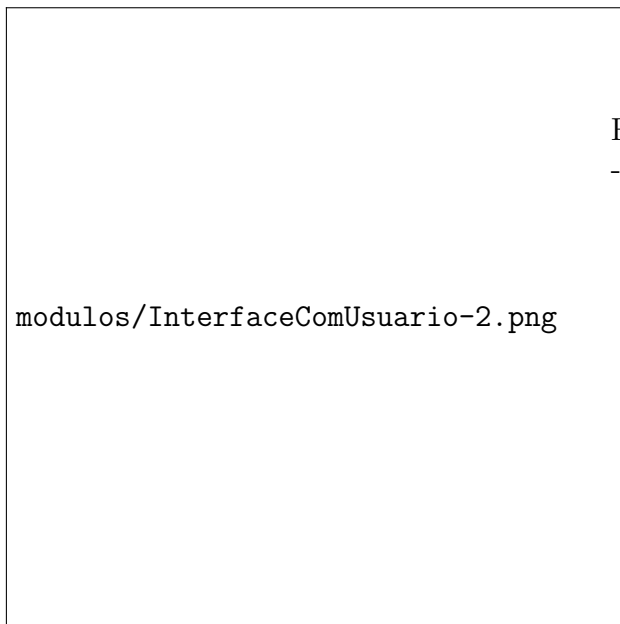
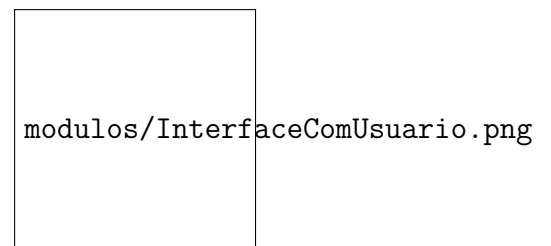


Figura 35: Protótipo Interface com Usuário
- PCB 3D

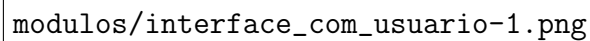


Fonte: Elaborado pelo autor no software Altium Design(?)

6.5.1.2 Oficial

A placa oficial de Interface com o usuário ainda não foi fabricada. Por se tratar de uma placa mais profissional, ela será mandada para ser feita para uma empresa privada ainda não escolhida, não na própria Universidade São Paulo. Assim como o protótipo, o projeto como um todo foi dividido em um esquemático e uma PCB.

Figura 36: placa de Interface com Usuário - Esquemático principal

The image area is mostly blank, with the text 'modulos/interface_com_usuario-1.png' located in the lower-left corner. This likely represents the schematic diagram of the user interface module.

modulos/interface_com_usuario-1.png

Fonte: Elaborado pelo autor no software Altium Design(?)

Para design do hardware do módulo de Interface com o usuário foi utilizado o CAD e software Altium Designer 21 (?) a partir de uma licença estudantil. O projeto completo

está disponibilizado em (?) e todos os componentes usado nessa versão oficial podem ser visto na tabela ??.

Tabela 10: Componentes Utilizados na placa de Interface com Usuário

Component	Description	Designator	Footprint	Quantity
KK_2.54_8vias	Conector KK 2.54mm 8 vias	CON1, CON2	KK_8vias_180°	2
KK_2.54_3vias	Conector KK 2.54mm 3 vias	CON3	KK_3vias_180°	1
KK_2.54_2vias	Conector KK 2.54mm 2 vias	CON4, CON6, CON7, CON12	KK_2VIAS_180°	4
KK_2.54_6vias	Conector KK 2.54mm 6 vias	CON5	KK_6vias_180°	1
KK_2.54_5vias	Conector KK 2.54mm 5 vias	CON8	KK_5vias_180°	1
KK_2.54_4vias	Conector KK 2.54mm 4 vias	CON9, CON10, CON11	KK_4vias_180°	3
LED 3MM RED	LED 3MM RED	D1	LED RED	1
1N4007 smd	Diode smd	D2, D3	DIOM5327X242N	2
rele T73	Relay or Contactor	K1	JQC3FT731Z12VDC	1
Trans BC817	Transistor BJT NPN BC817-25-7-F	Q1, Q2, Q4	SOT96P240X110-3N	3
4N25-X007T	Phototransistor	Q3	SOP254P1030X440-6N	1
47k	RES 1206 5%	R1, R2	RESC3216X60N	2
680R	Resistor	R3	RESC3216X60N	1
2k2	Resistor	R4, R5	RESC3216X60N	2
4k7	RES 1206 5%	R6, R7	RESC3216X60N	2
1k	RES 1206 5%	R8, R10	RESC3216X60N	2
100R	RES 1206 5%	R9	RESC3216X60N	1
microcontrolador	microcontrolador com modulo bluetoth e wifi	U1	ESP32_Desvikit_v1	1

Fonte: Elaborado pelo autor

Em relação ao protótipo, pouco mudou da lógica por trás do circuito, a principal diferença é que o modelo oficial utiliza todos componentes com encapsulamento SMD (?). Além disso, foi adicionado o protocolo de comunicação SPI também, como já foi comentado no início do capítulo.

A placa de circuito impresso ainda está em desenvolvimento.

6.6 Firmware

Cada módulo eletrônico tem o microcontrolador ESP32 (?) para realizar o processamento de dados e comunicação com outros dispositivos embarcados, como a Jetson Nano (?), o computador embarcado do Robô hospitalar. Cada microcontrolador, para realizar toda e qualquer função precisa ser devidamente programado, sem levar em consideração o circuito por trás dele, é claro. Por conta disso, cada um dos cinco módulos tem um firmware associado ao seu ESP32 específico, ou seja, cada módulo tem um código específico.

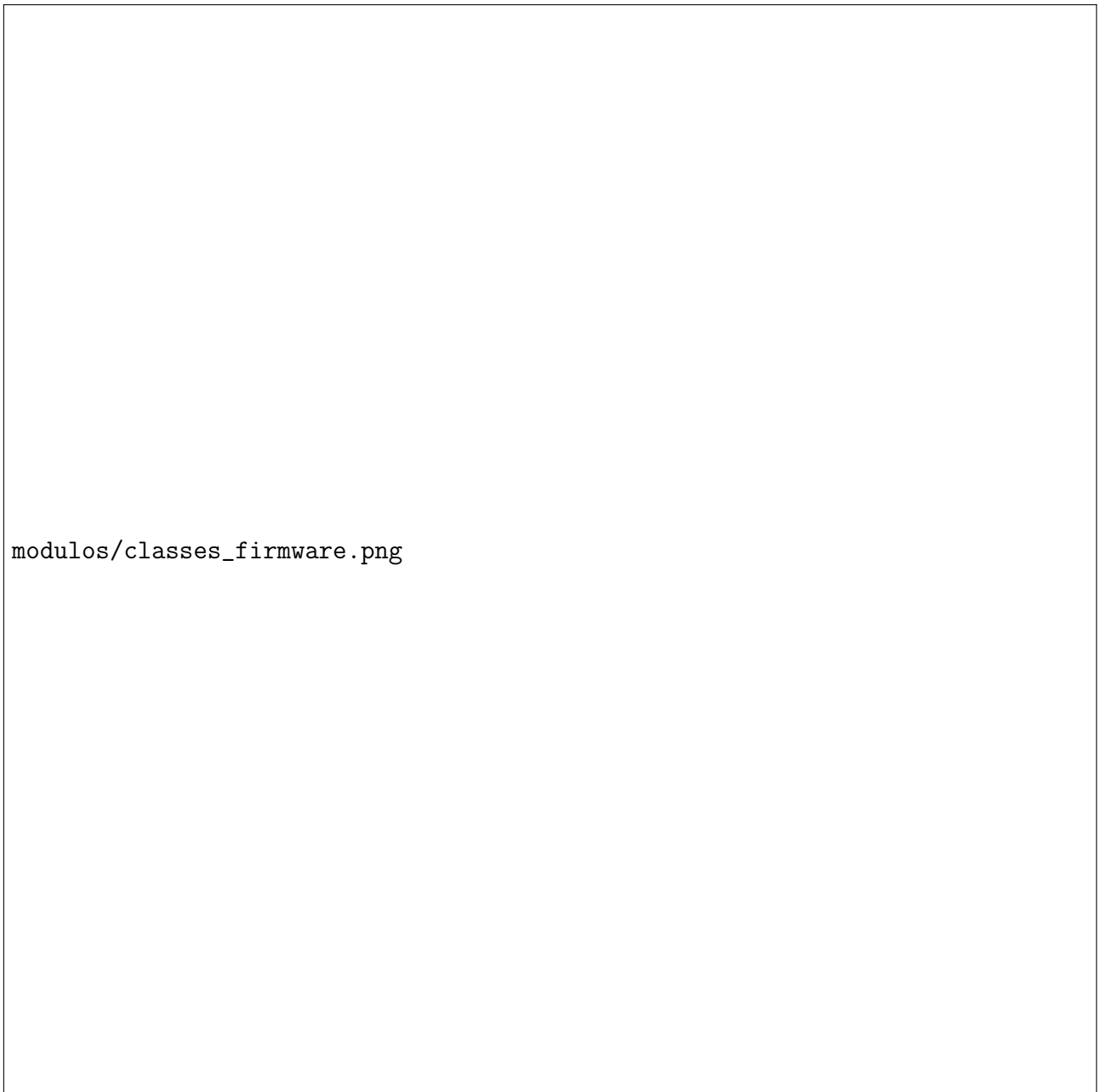
Nos primórdios do projeto, ainda na primeira versão do robô hospitalar, os códigos embarcados eram feitos em um arquivo só para cada módulo e de forma desorganizada. O maior problema disso era que isso tornava difícil compreender e adaptar os códigos. Dessa forma, há pouco mais de um mês começamos a reestruturar toda a arquitetura do firmware do projeto.

Com a nova arquitetura, os códigos associados a cada um dos módulos estão altamente relacionados com Orientação a Objeto (OO) e foram construídos com a linguagem de programação C++ (?), é muito recomendada para aplicações como essa, pois apresenta um baixo nível abstração, que é necessário para um firmware, e sintaxes típicas de OO. Além disso, C++ é uma linguagem que os membros da equipe do robô hospitalar já estavam familiarizados.

De maneira geral, os códigos foram feitos com base em princípios de Orientação a Objetos. Dessa forma, para cada módulo tem um único arquivo associado a ele, no qual esse arquivo utiliza uma série de bibliotecas que foram produzidas com os princípios mencionados.

Por fim, essa organização mais padronizada e associada a orientação a objetos, além de permitir uma maior portabilidade e vida útil do código, ainda facilita a compreensão dos scripts como um todo para os futuros membros do robô hospitalar.

Figura 37: Arquitetura geral do Firmware dos módulos eletrônicos embarcados



Fonte: Elaborado pelo autor

PARTE III

SOFTWARE

A primeira versão do software do robô hospitalar não era muito convidativa. Ela, por mais que fosse funcional e tivesse inúmeras qualidades, ainda era um código desorganizado, não tinha nenhuma documentação e quase nenhum padrão. De fato, o era evidente que foi um software feito de forma corrida, individualmente e sem nenhuma intenção de deixar fácil de se entender.

Para a segunda versão do robô hospitalar, todo o software do robô foi refeito por completo, porém, dessa vez, tomando muito cuidado com parametrização de códigos, separação de longos scripts e vários arquivos organizados menos com nomes bem intuitivos, documentação de tudo que foi feito, adição de orientação a objetos para construção dos algoritmos e por fim e mais importante, a adição do Git (?) e GitHub (?), junto ao Gitflow, para armazenar os códigos desenvolvidos e conciliar o trabalho em grupo.

Essa formulação foi feita com intuito de tornar o software do robô hospitalar organizado, padronizado e de fácil compreensão para os eventuais membros novos que irão entrar para o projeto do robô hospitalar, dessa forma, podendo dar continuidade ao trabalho sem muitas dificuldades.

Assim como a primeira versão, para a segunda versão foi utilizado o framework ROS (Robotic Operational System) (?) associado ao simulador Gazebo (?) para construção de todos os algoritmos de controle de controle do robô hospitalar.

7 AMBIENTE DE SIMULAÇÃO

Quando falamos em ambiente de simulação emular algum robô de maneira geral, a primeira dúvida que surge é “**por que simular?**”.

Quando falamos de emular o robô hospitalar, existem dois motivos principais que justificam o esforço de simular um sistema autônomo. O primeiro é a capacidade de pôr o projeto à prova sem a necessidade dos recursos físicos para isso. Tornamos o desenvolvimento de código mais dinâmico, pois não estamos limitados pelo meio físico. Outra vantagem menos óbvia é a garantia de repetibilidade dos testes, ou seja, podemos testar o sistema diversas vezes com as exatas mesmas condições, de modo a entender melhor o comportamento e as respostas do sistema simulado.

Pensando na grande área da engenharia mecânica, é muito mais fácil, rápido e barato simular o comportamento de uma peça no computador do que fabricá-la e aplicar um teste de impacto nela. O processo de projetar uma peça se torna mais eficiente e dinâmico, pois podemos testar diversas configurações e encontrar entre elas qual a mais adequada às nossas necessidades.

Por conta desses dois motivos, foi construído um ambiente de simulação para o robô hospitalar, pois os integrantes do projeto poderiam construir algoritmos de controle e visão computacional sem necessariamente estar com robô físico para realizar os teste, o que era de extrema importância, levando em consideração que toda a segunda versão do projeto foi construído em um período de isolamento social. A situação da simulação atualmente pode ser vista nas figuras ?? e ??.

Mesmo assim, vale lembrar que, por mais que a simulação seja convidativa e muito boa, não substitui o teste na realidade. Por melhor que a simulação seja, quem entra no hospital é o robô físico e por melhor que seja a lógica de controle, não existe algoritmo para corrigir parafuso solto ou falta de rigidez na estrutura.

Figura 38: Simulação Atualmente - Gazebo



simulacao_atual.jpg

Fonte: Elaborada pelo autor no Software Gazebo (?)

Figura 39: Simulação Atualmente - Rviz



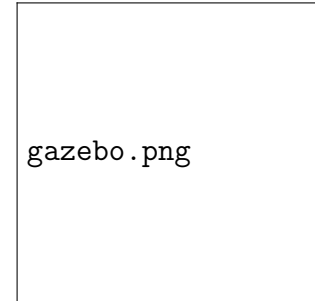
Fonte: Elaborada pelo autor no Software Rviz(?)

7.1 Gazebo

Dentro do conjunto de pacotes que constituem o ROS (?), um dos mais famosos no quesito de simulação de robôs autônomos e utilizados é o Gazebo. Nele podemos simular o comportamento dinâmico de um robô, seu movimento e a leitura de seus sensores em diferentes ambientes, cenários e situações programados, de modo a testar todo o algoritmo de controle do robô sem a necessidade ter o robô físico.

Por mais que o Gazebo (?) faça parte dos pacotes padrões do ROS, ele funciona de forma independente, ou seja, podemos utilizar o simulador sem outro software como requisito.

Figura 40: Simulador Gazebo



Fonte: Foto disponibilizada por gazebo.org

7.2 Modelo do Robô

Quando falamos em construir um modelo 3D de robô no Gazebo, precisamos descrever sucintamente como ele se comporta mecanicamente, como onde tem junções, e sensorial, como mencionando o como o sensor se comporta e onde se localiza.

O projeto em CAD do Robô hospitalar foi feito no SolidWorks (?) e Fusion 360 (?). Porém o gazebo não suporta arquivos de CAD diretamente na simulação, são suportados 3 tipos de arquivo: .urdf, .xacro e .sdf. Entretanto, tanto o SolidWorks quanto o Fusion têm a opção de exportar os modelos 3D construídos para esses formatos suportados pelo Gazebo, o que facilitou muito o nosso trabalho de criação de um modelo.

Os 3 formatos correspondem a arquivos com estrutura XML. O URDF é o mais simples do 3, o formato xacro é igual ao URDF com a adição ao suporte de macros dentro no arquivo, o que permite fazer a divisão de um código longo em vários pequenos e organizados. Para completar, o formato SDF é mais novo e possui suporte a uma maior quantidade de informações na descrição do modelo do robô, como fator de atrito e amortecimento nas juntas do robô.

Internamente, o Gazebo converte todas as entradas para SDF. Nós utilizamos o formato URDF para o modelo da carenagem porque existe uma extensão de SolidWorks que exporta um arquivo .urdf a partir de um CAD de Assembly.

7.2.1 Estrutura XML

No universo da engenharia de software da robótica, em geral, é muito comum o uso da estrutura XML para descrever as relações entre as diferentes peças e partes de um robô. Um robô descrito em URDF é composto por duas entidades: links e joints

Segundo a documentação oficial do Gazebo (?), as joints são a descrição de como juntamos duas estruturas diferentes em um URDF, assim, ela meio que descreve os graus de liberdade de determinada junção. O gazebo tem uma série de junções permitidas:

- revolute - Revolução ao redor de um eixo com limite de posição, como um servo;
- continuous - Revolução ao redor de um eixo sem limite de posição, como um motor DC ou motor de passo;
- prismatic - Vínculo deslizante ao longo de um eixo com limite de posição, como um carro em uma guia;
- fixed - Vínculo de travamento, no qual os 6 graus de liberdade estão fixos. Descreve o vínculo entre duas peças que não possuem movimento relativo;
- floating - Vínculo sem restrição em nenhum dos 6 graus de liberdade;
- planar - Vínculo de movimento em um plano perpendicular a um eixo especificado.

Os links estão organizados em uma estrutura de dados de árvore, na qual links pais (como o corpo) se conectam com links filhos (como as rodas ou sensor) por meio de joints, que descrevem o vínculo existente entre os links. Essa estrutura serve para relacionar os diferentes sistemas de coordenadas dos vários links, como pode ser visto na figura ?? e ??

7.2.2 Carenagem

O CAD completo da carenagem do robô era muito complexo. Havia um número muito grande de parafusos e outros detalhes mecânicos que tornaria a simulação muito custosa sem um bom custo benefício. Por conta disso foi elaborado um modelo simplificado da estrutura principal do robô.

Além disso, uma simulação muito custosa deixaria inviável a utilização da mesma nos computadores domésticos que a maioria dos membros do projeto tem. Por conta disso, deixar a simulação mais eficiente era necessário para todo mundo dar continuidade ao trabalho a distância.

Figura 41: Estrutura URDF - Visão Física

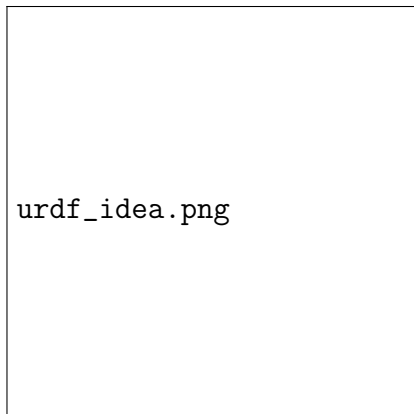
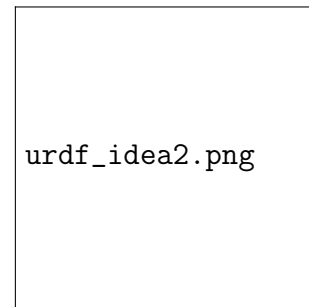


Figura 42: Estrutura URDF - Visão Geral



Fonte: retidado do site <http://library.isr.ist.utl.pt/>

Nessa estrutura principal, se deu prioridade principalmente ao formato geral do robô, peso e momento de inércia e principalmente na construção sensorial do robô. Dessa forma, podendo realizar um ótima simulação e não ter um custo computacional muito elevado.

Figura 43: Modelo Simulado da Robô Hospitalar (V2)



modelo_robo_hospitalar.jpg

Fonte: Elaborada pelo autor no Software Gazebo(?)

7.2.2.1 Estrutura do URDF - Carenagem

```

root Link: base has 14 child(ren)
  child(1): left_back_back_castor_mount_link
    child(1): left_back_castor_dummy_link
      child(1): left_back_castor_wheel_link
  child(2): left_front_castor_mount_link
    child(1): left_front_castor_dummy_link
      child(1): left_front_castor_wheel_link
  child(3): right_back_castor_mount_link
    child(1): right_back_castor_dummy_link
      child(1): right_back_castor_wheel_link
  child(4): right_front_castor_mount_link
    child(1): right_front_castor_dummy_link
      child(1): right_front_castor_wheel_link
  child(5): camera_link
  child(6): link_back_left_proximity
  child(7): link_back_proximity
  child(8): link_back_right_proximity
  child(9): link_front_left_proximity
  child(10): link_front_proximity

```

7.2.3 Sensores

Depois de importar corretamente o modelo 3D, é necessário adicionar a descrição funcional e física do sensor no seu modelo de sensor.

Para realizar descrição, é necessário adicionar um **plugin** no Gazebo, que consiste em criar um arquivo que contém toda a descrição física, lógica e conexão(relacionado com as joints).

No caso do robô hospitalar, era necessário adicionar sensores de distância, lidar, câmera e encoder. Para cada um desses sensores, com exceção do encoder, que ainda não foi implementado, todos os outros receberam um um arquivo individual no repositório individual da simulação.

O modelo do robô com os sensores funcionando pode ser visto na figura ??

Figura 44: Sensores Simulado da Robô Hospitalar (V2)



modelo_robo_sensores.jpg

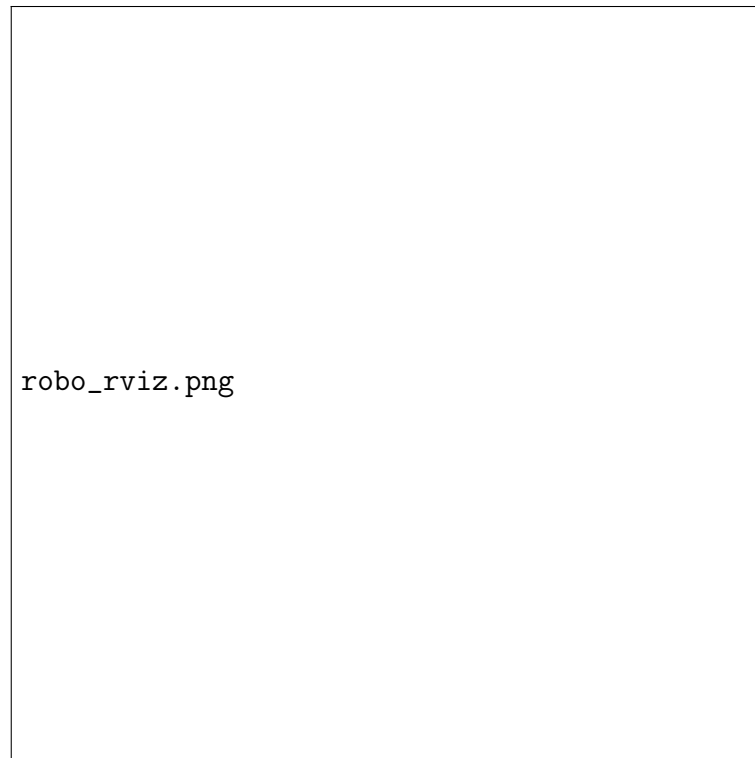
Fonte: Elaborada pelo autor no Software Gazebo(?)

7.2.3.1 Rviz

Para visualizar os dados coletados pelo robô e como processa essa informação, principalmente associada aos tópicos publicados, usamos outro framework associado ao ROS, o Rviz.

Com o auxílio do Rviz, podemos entender melhor como os dados obtidos pelos sensores do robô estão sendo processados e traduzidos para o ponto de vista do robô. Na figura ??, pode-se ver o modelo do robô hospitalar no Rviz. Na parte inferior, é exposto como os dados do sensor de distância estão sendo vistos pelo robô e na parte superior, a parte em vermelho, é como os dados do lidar estão sendo vistos.

Figura 45: Sensores Simulado no Rviz da Robô Hospitalar (V2)



Fonte: Elaborada pelo autor no Software Rviz(?)

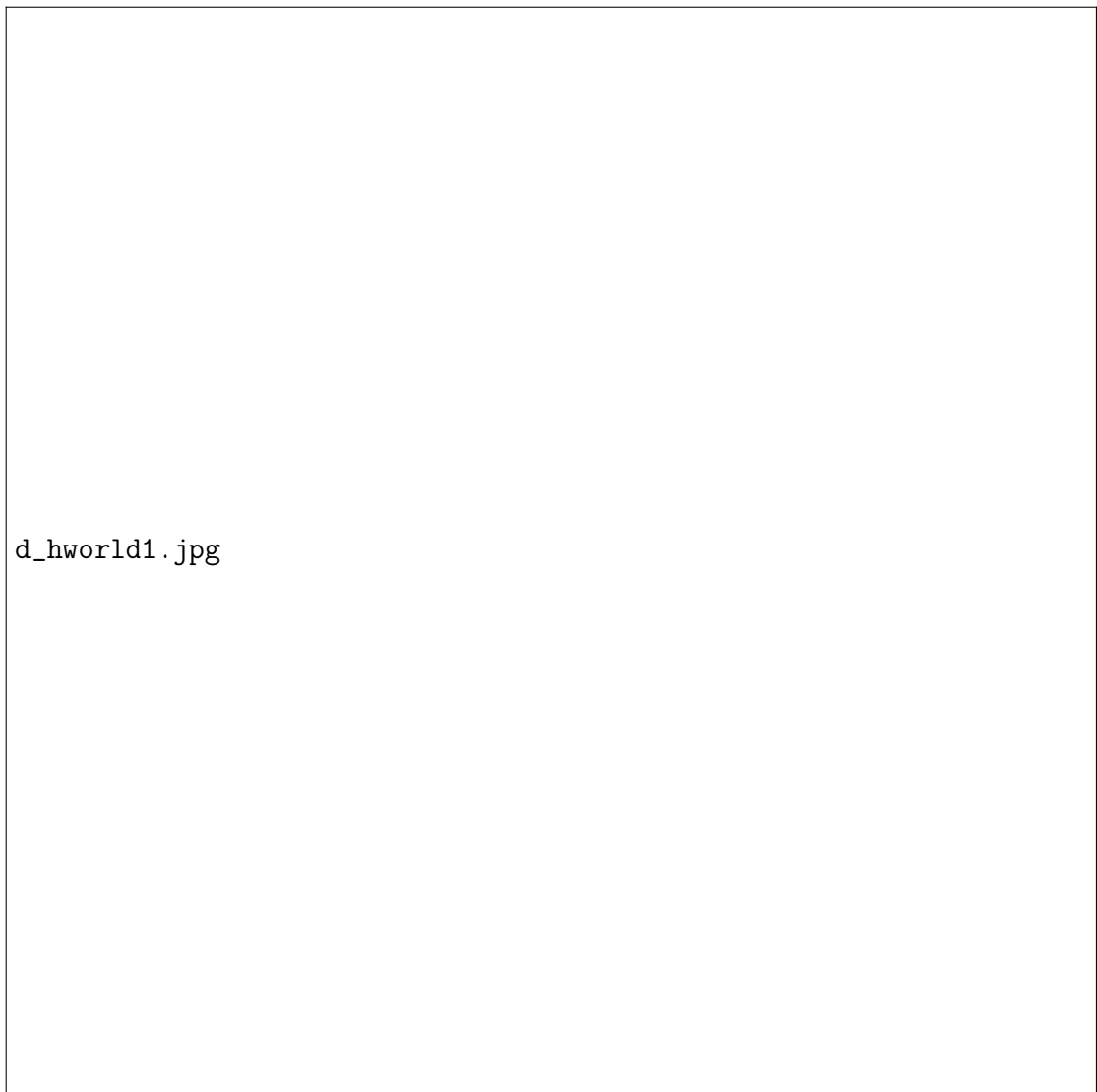
7.3 Modelo do mundo

Para uma boa simulação, não basta somente termos o modelo físico do robô bem estruturado, precisamos adicionar o nosso robô em um ambiente que ele realmente estará imerso na vida real. Para isso, é necessário criar um mundo (vindo do termo **world**) no caso para adicionarmos o nosso robô.

Por se tratar de um robô hospitalar, o melhor ambiente possível para deixar o robô imerso em um hospital ou algo similar a isso. Entretanto, criar um mundo do zero no gazebo é uma experiência muito demorada e minuciosa, por conta disso, o grupo optou por usar um mapa de hospital já pronto e disponibilizado pela Amazon para uso no AWS no GitHub, tal repositório pode ser visto aqui (?).

Com o mapa disponibilizado pela Amazon, que disponibiliza um ambiente ideal para simulação do robô hospitalar, podemos por fim completar todo o ambiente de simulação e começar a desenvolver os algoritmos básicos de controle do robô hospitalar.

Figura 46: AWS RoboMaker Hospital World - vista pela frente



Fonte: Elaborada pelo autor no Software Gazebo(?)

Figura 47: AWS RoboMaker Hospital World - vista por trás



Fonte: Elaborada pelo autor no Software Gazebo(?)

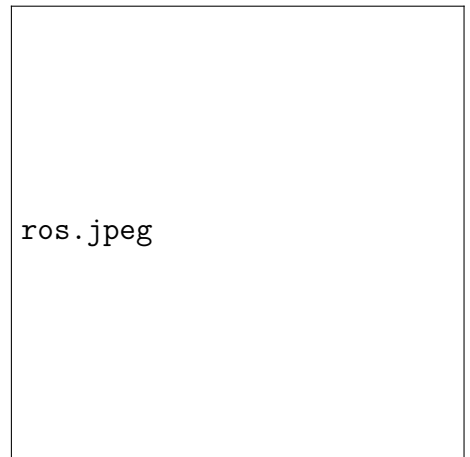
8 ALGORITMOS DE CONTROLE

8.1 ROS

A sigla significa Robotic Operating System, no entanto, o ROS não é exatamente um sistema operacional. O ROS é um meta-sistema operacional, ou seja é uma coleção de frameworks, voltado para o controle de robôs autônomos, que busca padronizar a forma como se organiza e se escreve código em robótica. Dessa forma, seguindo a ideia de padronizar com um conjunto s de convenções, é possível reutilizar códigos.

Em engenharia da computação, há a necessidade de padronizar códigos da mesma forma que na engenharia mecânica há a necessidade de padronizar parafusos. Por mais diferentes que possam ser dois robôs, existe uma série de atividades básicas e cotidianas que costumam aparecer em quase todos os projetos, como ler sensores, analisar e filtrar dados, acionar motores etc. Dessa forma, como uma série de pessoas usam a mesma convenção, a importação desse código se torna mais fácil e natural, assim, focar mais na unicidade do nosso robô do que uma série de implementações já feitas várias vezes.

Figura 48: Framework ROS



Fonte: Foto disponibilizada em ros.org

8.1.1 Funcionamento

Como um todo, ROS facilita o tráfego de análise de informações, os seus principais conceitos podem ser vistos abaixo:

- Node: Menor unidade de processos executáveis, ou seja a unidade mínima de um projeto no ROS. Um node é um pedaço de código que realiza uma determinada

tarefa. Cada node transmite e recebe informações por meio da message.

- **Package:** Principal unidade de organização do software no ROS e é composto de um ou mais nodes, informações para execução dos nodes, entre outros. O package é a menor unidade de construção e lançamento do ROS.
- **Message:** É a forma de comunicação entre os nodes. As messages possuem dados que fornecem informações para outros nodes. Um exemplo prático de message seria a leitura de um sensor de distância.
- **Topic:** Canal de comunicação entre nodes e identifica o conteúdo da message. Quando um node está enviando dados, dizemos que está publicando um topic, ou seja, ele é o publisher. Já quando um node está recebendo dados, ele está assinando um topic, ou seja, ele é o subscriber.
- **Master:** Núcleo do ambiente em ROS e sabe o que está acontecendo em todo o sistema. É responsável por inicializar o sistema e fornecer o registro de nomes e o serviço de pesquisa para os nodes. Além disso, ela também configura as conexões entre os nodes.

De maneira geral, tudo repassado aqui tem como base o guia oficial de utilização do ROS (?), que é disponibilizado de graça.

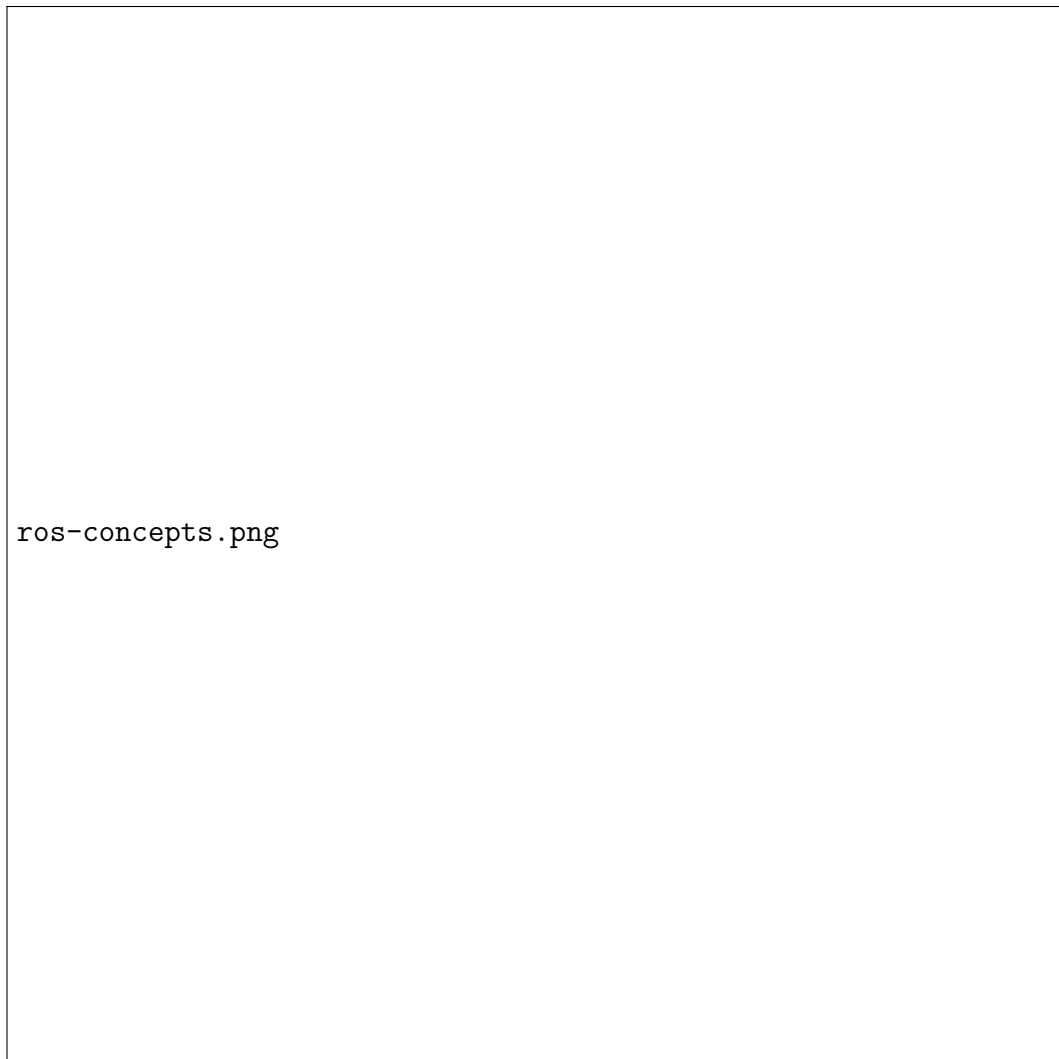
De certa forma, o que resta esclarecer é como é feita a integração dos algoritmos feitos em ROS com o simulador Gazebo. Como o Gazebo faz parte dos Frameworks associados ao ROS, a integração entre os dois sistemas é muito simples. Todos os tópicos publicados no ROS podem ser lidos no Gazebo, dessa forma, somente exigindo que associemos um determinado tópico a algum link ou joint do nosso robô.

8.2 Estrutura dos algoritmos

Quando pensamos na construção do software de um robô autônomo, é quase que imediatamente associamos aos algoritmos que são executados e dados que são processados. Dessa forma, realizar tomadas de decisão e reagir da melhor forma possível a alguma adversidade do ambiente.

No caso do robô hospitalar não é diferente. Na estrutura geral do software do robô hospitalar a ideia é constituída, até o momento, basicamente: mapeamento do ambiente, planejamento global e local e uma odometria ainda em desenvolvimento. Além dessa,

Figura 49: Funcionamento do ROS



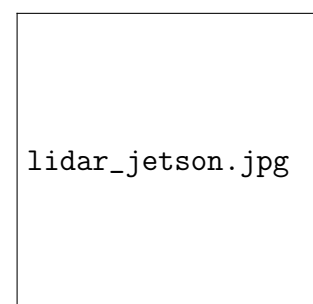
Fonte: Retirado de (?)

há uma parte muito importante de processamento de dados, o qual consiste em preparar os dados para enviar para os dispositivos embarcados, para assim realizar o envio da informação pela comunicação Serial (?). Para saber mais sobre isso, verificar a seção de Comunicação com Embarcados.

8.2.1 Mapeamento

O robô, para definir a sua rota para determinado ponto do mapa precisa ter noção macro e micro do ambiente que ele se encontra. Para haver a possibilidade do robô ter essas noções é necessário haver o mapeamento do ambiente que o robô se encontra e deixar isso registrado

Figura 50: RPLidar A1M8



em um mapa com formato suportado pelo ROS.

Para realizar o mapeamento do ambiente, foi feitos teste com RPLidar A1M8 R6 (?) e Slamtec Mapper M1M1 (?), ambos lidares feitos para mapeamentos. Ambos tiveram bons resultados, porém o Mapper não tem suporte para arquitetura computacional da Jetson, então ele foi usado somente em computadores e dispositivos mobile.

8.2.1.1 Mapas obtidos

A partir de simulações usando o Rviz (?), outro framework associado ao ROS, pudemos simular o mapeamento com o robô em um ambiente hospitalar. Simular o mapeamento no hospital simulado foi de extrema importância para o projeto, pois os integrantes do projeto não estavam permitidos a frequentar o laboratório na USP na época. o resultado pode ser vista na figura ??.

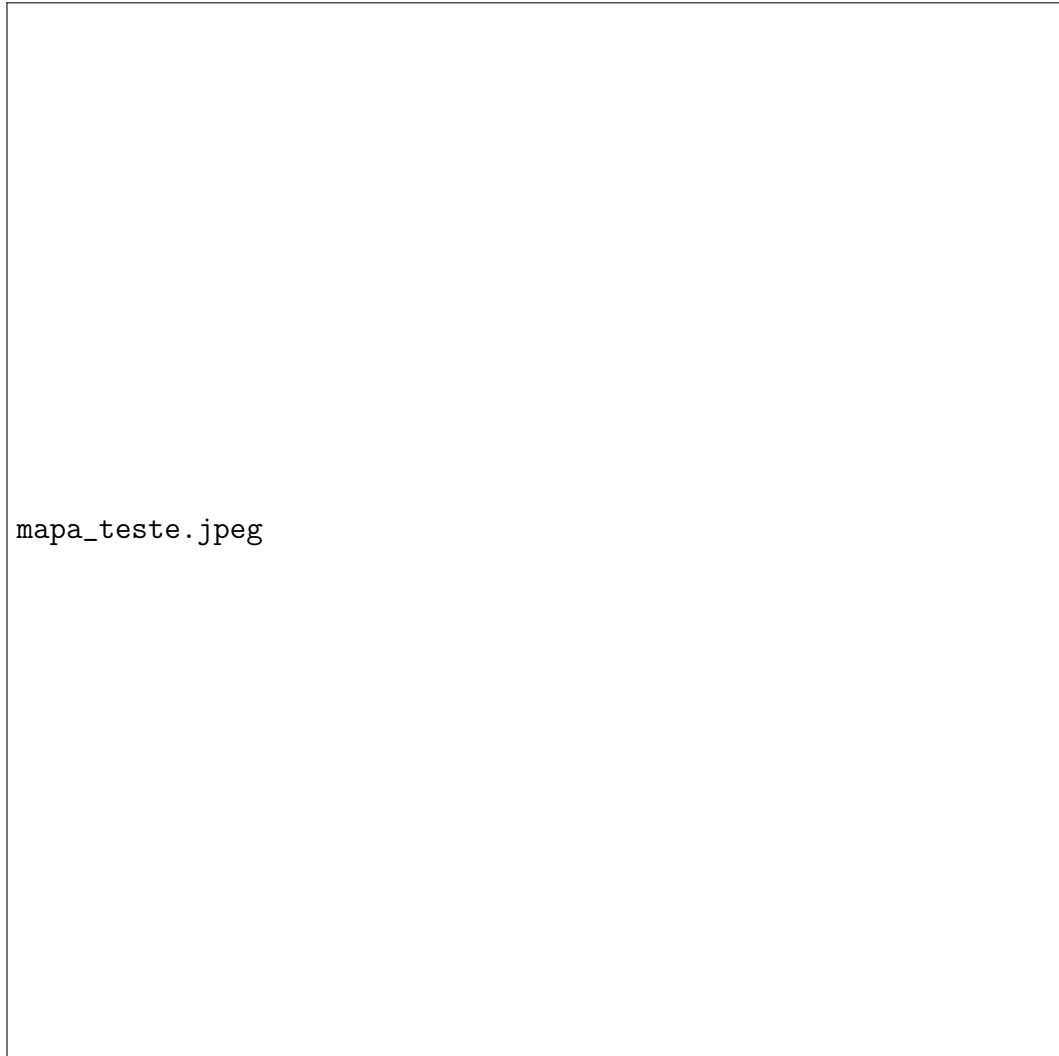
Figura 51: Hospital Simulado Mapeado



Fonte: Elaborada pelo autor pelo Software Rviz (?)

Depois de simulado, foi realizado o mapeamento usando o lidar Mapper na casa de um dos integrantes do projeto. O resultado pode ser visto na figura ??.

Figura 52: Mapeamento real - teste



Fonte: Elaborada pelo autor pelo Software Rviz (?)

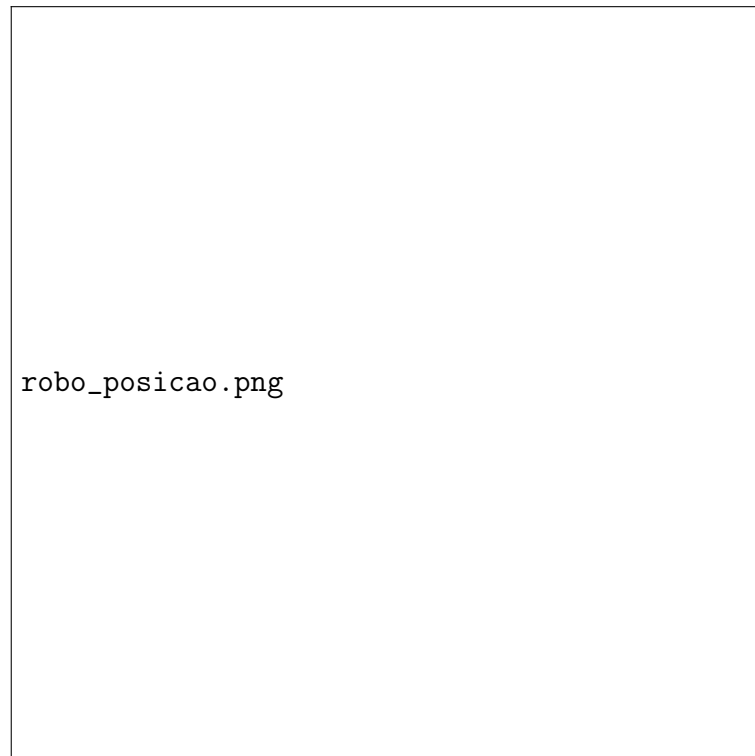
8.2.2 Navegação

Depois que todo o mapeamento tiver sido feito, pode-se estimar uma posição inicial para o robô começar a se locomover e a partir dele o robô e ir estimando a sua posição com parâmetros da sua odometria. Entretanto, quando se pensa na navegação do robô, é preciso mensurar a todo momento a posição atual, com seu erro, e a rota que o robô vai prosseguir naquele momento. Por conta disso, foi implementado um algoritmo de localização e de planejamento de rotas.

8.2.2.1 Localização

Assim como na primeira versão do robô hospitalar, para localizar o robô no mapa o módulo foi utilizado o algoritmo de Adaptive Monte Carlo Localization (?), que é mais conhecido como AMCL e já se encontra de forma nativa no ROS. Com esse algoritmo, podemos estimar a posição do robô e seus erros, que é indicado na figura ??.

Figura 53: Adaptive Monte Carlo Localization (?),



Fonte: Elaborada pelo autor no Software Rviz(?)

O algoritmo em questão trabalha junto a odometria, recebendo dados da posição do robô nos eixos x e y. A partir desses dados, gera uma nuvem de probabilidade para o robô estar, assim, conforme o robô se movimenta essa posição é corrigida e a nuvem de probabilidade diminui, o que indica a maior precisão da posição atual do robô. Na figura ??, pode-se ver a nuvem de posições.

8.2.2.2 Planejamento de Rotas

Para gerar o mapa de custos e calcular a melhor rota possível foi utilizado o pacote Move Base (?), que assim como o AMCL (?), é um algoritmo já implementado e nativo do próprio ROS. O funcionamento desse pacote do ROS pode ser visto na figura ??.

De maneira geral, o pacote move base fornece um auxílio para gerar rotas para robôs

Figura 54: Funcionamento do Move base



Fonte: Retidada do ROS Wiki (?)

no geral para os usuários de ROS. Assim, o Algoritmo “emprestado” gera um escopo de um plano local e global para o definir a rota que o robô precisará traçar para chegar no local desejado.

Para o planejar a rota local, o algoritmo não leva em consideração os mapas gerados, mas sim os dados coletados por sensores de distância e pelo lidar. O principal objetivo de planejar uma rota local é evitar imprevistos, como alguém entrar na frente do robô ou colocar um objeto em um lugar que não tinha antes. Assim que houver a detecção de algum desses imprevistos, o robô calcula uma nova rota que não tenha esse obstáculo. Para realizar tal mudança, é gerado um mapa de custo local (`local_costmap`). Vale ressaltar, que como é um plano que tem como função evitar problemas, ele está atualizando constantemente.

Graças ao Move Base, há uma série de parâmetros que se pode usar para configurar o mapa gerado usado e regular ele para atender melhor a estrutura do robô produzido. na figura ?? pode-se ver a rota local naquele ponto.

Figura 55: Planjamento de rota - Local



Fonte: Elaborada pelo autor no Software Rviz(?)

O mapa global, por outro lado, somente recebe o mapa do ambiente, e a partir de algoritmos nativos do ROS, calcula o mapa de custo global (`global_costmap`) e define a melhor rota até determinado ponto. De maneira geral, o mapa consegue mensurar as áreas que o robô pode circular, as que ele deve evitar e as que ele não pode chegar perto, como paredes. Dessa forma, as regiões próximas às que são proibidas tendem a ser regiões a ser evitadas.

O mapa de custo global, como um todo, se comporta como um campo escalar de duas dimensões, o qual as regiões proibidas tendem a ter um peso infinito e as regiões livres, peso zero. O peso da região tende a aumentar de forma exponencial conforme se aproxima da região proibida. No mapa de custo global gerado pelo próprio ROS, se pode alterar parâmetros do crescimento dessa exponencial para se adequar o melhor possível ao robô

produzido. O mapa global e a rota traçada podem ser visto na figura ?? e ??

Figura 56: Planjamento de rota - Global



Fonte: Elaborada pelo autor no Software Rviz(?)

Figura 57: Planjamento de rota - Trajetório



robo_rota2.png

Fonte: Elaborada pelo autor no Software Rviz(?)

9 INTEGRAÇÃO SOFTWARE HARDWARE

Todos os algoritmos complexos desenvolvidos para robô hospitalar são feitos com linguagem de alto nível, com Python (?), ou C++ (?) em algumas situações. Porém, como esses algoritmos requerem um poder computacional muito maior que um microcontrolador embarcado consegue fornecer, é necessário utilizar um computador embarcado para realizar essas diversas operações complexas e processar os dados.

Por outro lado, a integração de todo o Software do sistema tem uma série de desafios, porém os mais importantes no ponto que projeto esta é: A utilização , Conexão a distância e se comunicar com os microcontroladores com o Computador embarcado.

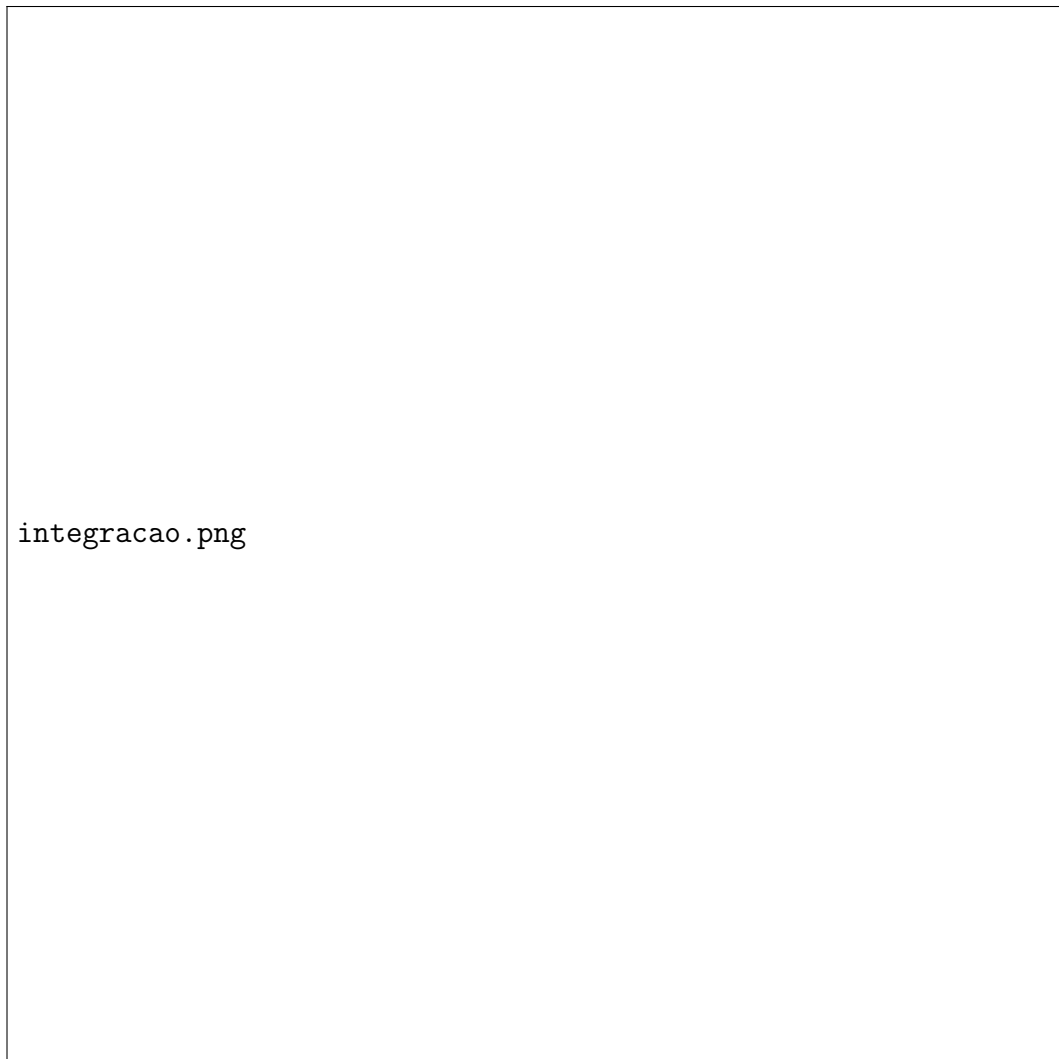
9.1 Computador Embarcado

A escolha do computador embarcado é fundamental para o projeto, pois o mesmo é o grande limitador do poder computacional do robô completo. A escolha errado desse computador corrobora para o mal funcionamento de todo o hardware e software, dado que conforme a complexidade aumentasse, a velocidade de processamento de dados cairia até um ponto que o robô entraria em colapso computacionalmente.

De maneira geral, ainda como herança da primeira versão do robô hospitalar, o computador embarcado utilizado no projeto foi a Jetson Nano Develop Kit (?). O motivo completo da escolha desse computador embarcado, a qual eu não presenciei, pode ser visto no relatório do João Pedro Secchim Sotto Maior (?). O motivo da escolha pode ser visto abaixo:

Nos últimos anos houve grandes avanços no desenvolvimento de processadores arm, principalmente por ser a arquitetura utilizada nos smartphones e tablets. Aproveitando esses avanços diversas empresas desenvolveram computadores que utilizam esses processadores, a mais conhecida é a fundação Raspberry Pi que produz e vende o Raspberry Pi, single-board computer (computador de placa única) que é um computador que possui todos os componentes que precisa soldados em uma única placa.

Figura 58: Integração da Jetson com Módulos Embarcados



Fonte: Elaborada pelo autor

Após pesquisar as opções de computadores disponíveis, foi necessário escolher entre três placas, o Raspberry Pi, o Labrador desenvolvido na Escola Politécnica da USP [4] ou uma placa da linha Jetson da Nvidia [5]. Uma das razões que motivou a decisão de utilizar uma placa da linha Jetson da Nvidia foi grande presença da empresa na área de Machine learning e Computer vision. Além disso, os computadores da linha Jetson possuem GPUs muito mais poderosas do que as outras duas opções apresentadas, o que acelera muito o processo de manipulação de imagens.

No mais, desde a escolha da Jetson Nano, não houve nenhuma necessidade até o momento de trocar o computador embarcado, muito pelo contrário, ele ainda está suprimindo muito bem as demandas computacionais do projeto.

Figura 59: Jetson Nano Developer Kit



Fonte: Nvidia

9.2 Conexão a distância

A Jetson Nano (?) é um computador embarcado que no projeto do robô hospitalar tem a missão de carregar os códigos de controle de todo o robô autônomo. No mais, quando surge necessidade mexer nessa parte do hardware do robô, não é muito interessante precisar desmontar uma série de equipamentos para ter acesso a Jetson. Tendo isso em mente, surge a necessidade de conseguir acessar o computador embarcado a distância.

Para resolver esse problema na Jetson Nano, foi implementado a comunicação Secure Shell (SSH) (?), que é um protocolo de rede criptográfico para operação de serviços de rede de forma segura sobre uma rede insegura. Além de ser extremamente seguro, o protocolo SSH não demanda de uma boa conexão de internet para ter uma respostas satisfatória, um dos requisitos mais importantes, levando em consideração que dentro do hospital universitário a conexão com a rede wireless não é boa no estabelecimento todo.

A implementação é relativamente simples, basta ter o ssh devidamente instalado na Jetson Nano e no computador que deseja acessar remotamente. No caso da Jetson, como se comporta como servidor, é necessário criar uma chave SSH, de resto, a implementação é idêntica em ambos os dispositivos.

Figura 60: Comunicação SSH



Fonte: Imagem obtida em hostinger.com.br

9.3 Comunicação com Embarcados

A comunicação serial é capaz de enviar e receber toda a informação sequencialmente, o que abre a possibilidade de que o número de fios seja menor. Em outras palavras, protocolos de comunicação baseados no serial, são síncronos, ou seja, dependem do clock do sistema. Por outro lado, diferente de protocolos de comunicação, só consegue se comunicar com um aparelho por vez.

Outros protocolos de comunicação, como I2C e SPI, os quais as conexões são organizadas de forma paralela, ou seja, é capaz de enviar e receber dados em mais de uma via de comunicação, assim, é capaz de enviar vários bits simultaneamente, resultando em maior rapidez na transmissão da informação.

Figura 61: Comunicação Serial



Fonte: Obtido em <https://www.robocore.net/>

No nosso caso, como só precisamos nos comunicar com um microcontrolador e como protocolo serial é mais simples de se implementar do que os outros protocolos, por decisão de projeto foi adotado a comunicação serial para haver essa ponte entre os dados processados na jetson para os módulos embarcados.

PARTE IV

RESULTADOS

No âmbito da computação e da eletrônica, o software e hardware, que andam muito próximos um do outro, os resultados foram satisfatórios, mas nada além disso. Foi bastante trabalho em pouco tempo, porém, não foram realizados tantos testes quanto foi idealizado por conta de períodos sem poder frequentar a USP.

No que diz respeito ao Hardware, os módulos oficiais ainda não foram fabricados, pois é necessário revisar melhor os esquemáticos e testar todos os protótipos por completo para mandar fazer. Porém, levando em consideração que quase toda a reestruturação do projeto tem pouco menos de 8 meses, e poucos mais de 10 placas de circuito impresso foram sintetizadas, ainda é um bom resultado.

No que diz respeito ao Software, em pouco menos de 6 meses, todo o ambiente de simulação e algoritmos de controle foram produzidos. Por mais que o código da primeira versão fosse funcional, muito pouco era reaproveitável pela falta de organização e documentação de tudo desenvolvido. Por conta disso, tudo foi refeito. Contudo, tiveram-se muitos resultados produzidos e garantia de um código que pode ser repassado para futuras gerações de membros do projeto.

APÊNDICE A – VISÃO COMPUTACIONAL

Por falta de tempo, ainda não foi possível dar continuidade às pesquisas que o João Pedro (?) começou e implementar algo concreto no robô.

De maneira geral, como todo o grupos estava utilizando ROS para a produção de algoritmos de controle, as pesquisas sobre visão computacional foram em cima de ferramentas que facilitassem a comunicação com o ROS.

A princípio, além do OpenCV, foi utilizado a ferramenta Darknet(?) e Darknet Ros (?), que já são grandes datas set com uma comunicação facilitada e pré estabelecida com ROS. De maneira geral os testes com essas ferramentas foram um sucesso nos computadores convencionais, como pode ser visto nas figuras ?? e ??.

Entretanto, os testes usando o computador embarcado, a Jetson Nano , não foram satisfatórios e a ideia acabou sendo deixada de lado para dar prioridade para outras áreas.

Figura 62: Teste de visão Computacional - Simulação



viso_computacional_simulado.jpg

Fonte: Elaborado pelo Autor no Software Gazebo (?)

Figura 63: Teste de visão Computacional - Real

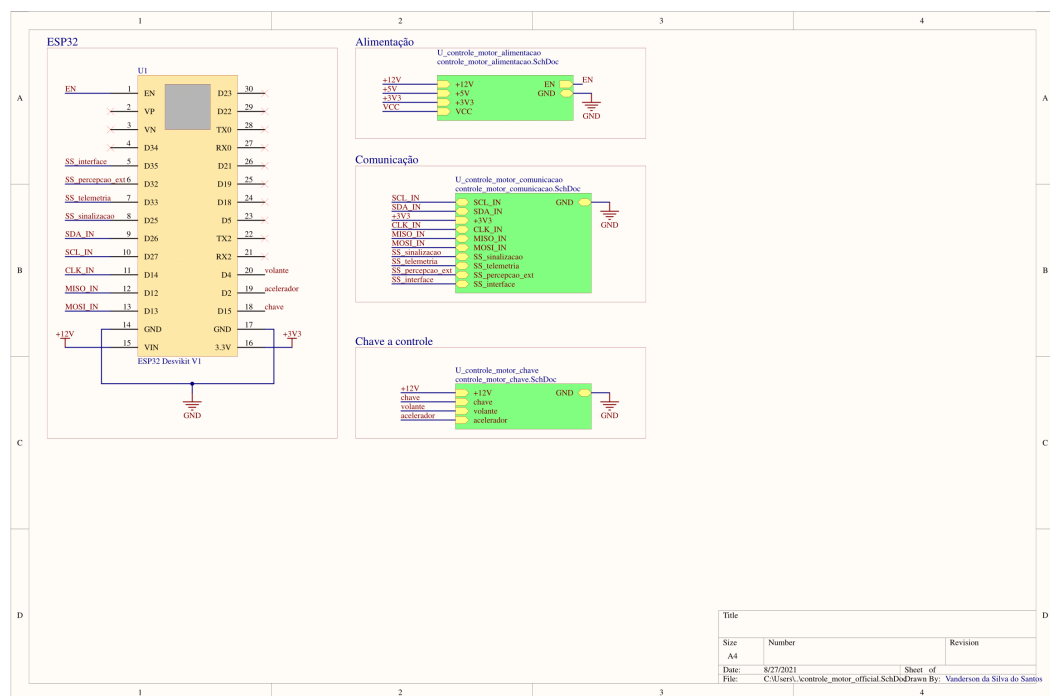


Fonte: Elaborado pelo autor

ANEXO A – ESQUEMÁTICO COMPLETO DOS MÓDULOS EMBARCADOS

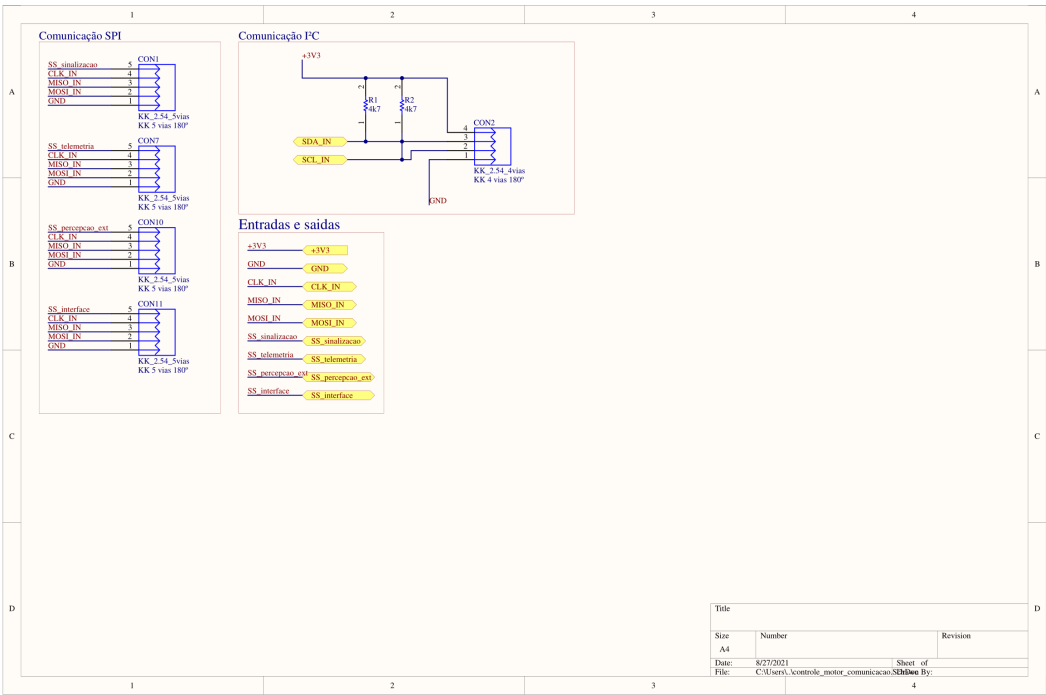
A.1 Controle

Figura 64: Placa de Controle - Esquemático 1



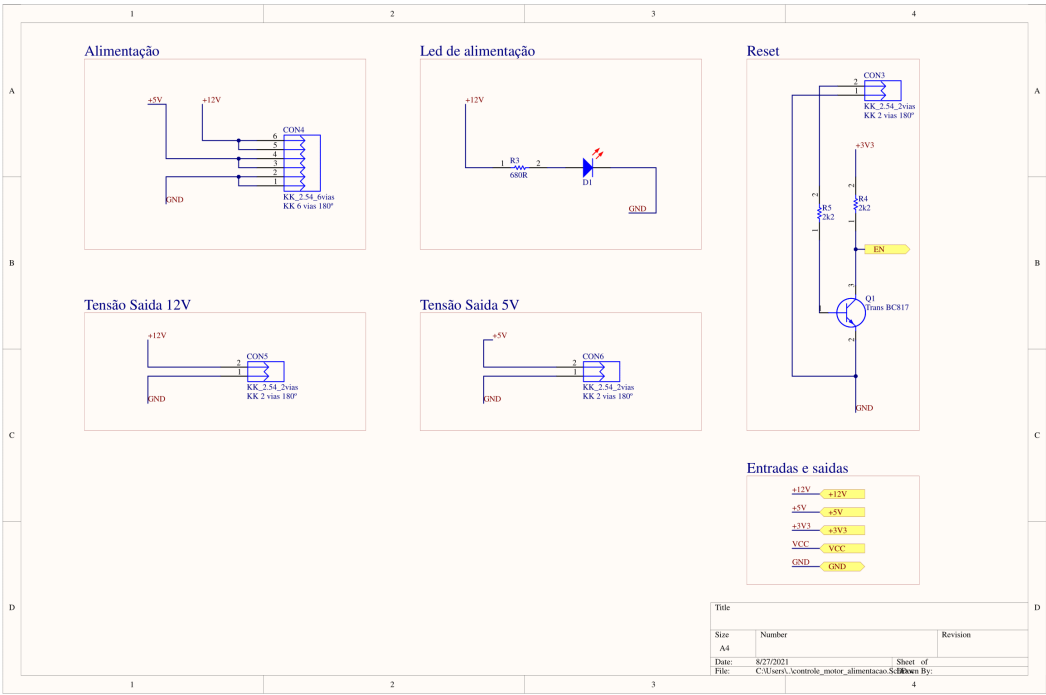
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 65: Placa de Controle - Esquemático 2



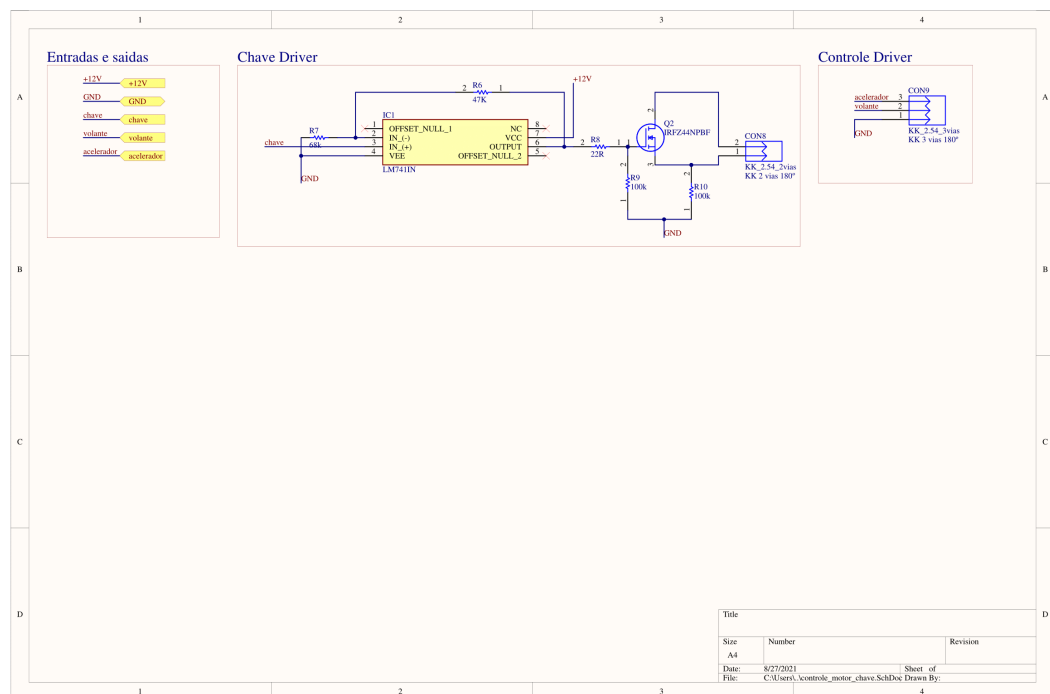
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 66: Placa de Controle - Esquemático 3



Fonte: Elaborada pelo autor no software Altium Design(?)

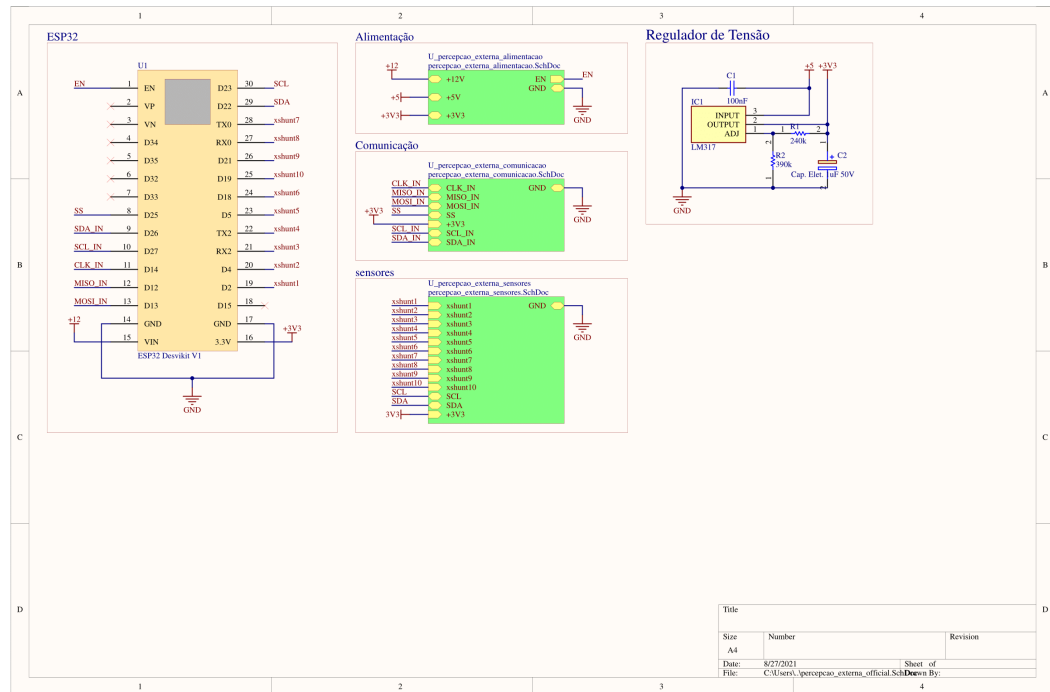
Figura 67: Placa de Controle - Esquemático 4



Fonte: Elaborada pelo autor no software Altium Design(?)

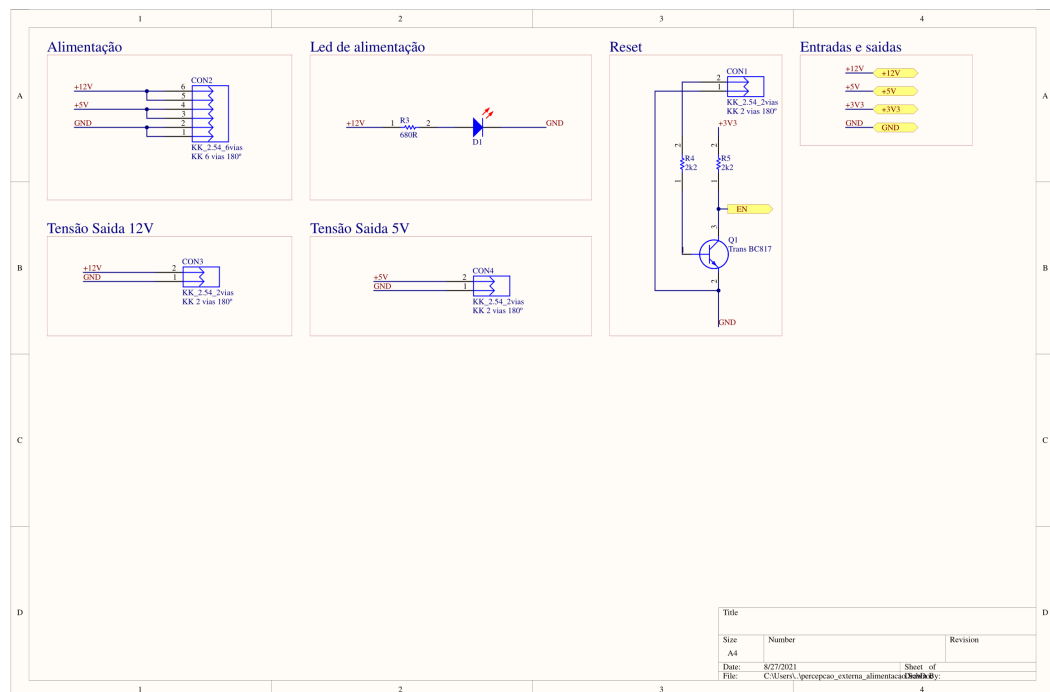
A.2 Percepção Externa

Figura 68: Placa de Percepção Externa - Esquemático 1



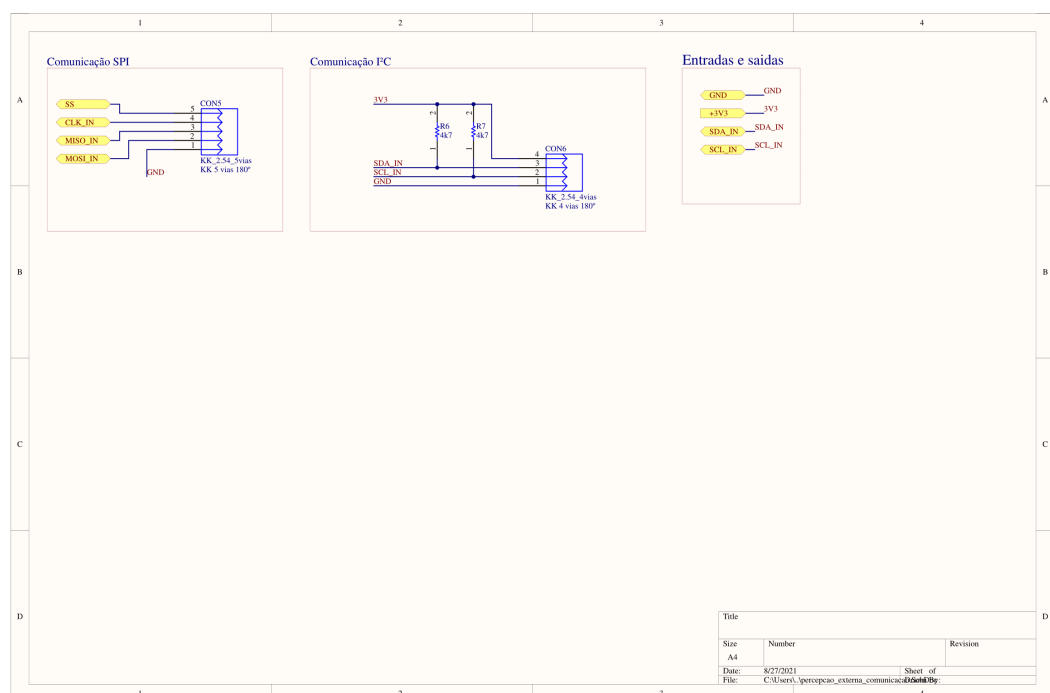
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 69: Placa de Percepção Externa - Esquemático 2



Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 70: Placa de Percepção Externa - Esquemático 3



Fonte: Elaborada pelo autor no software Altium Design(?)

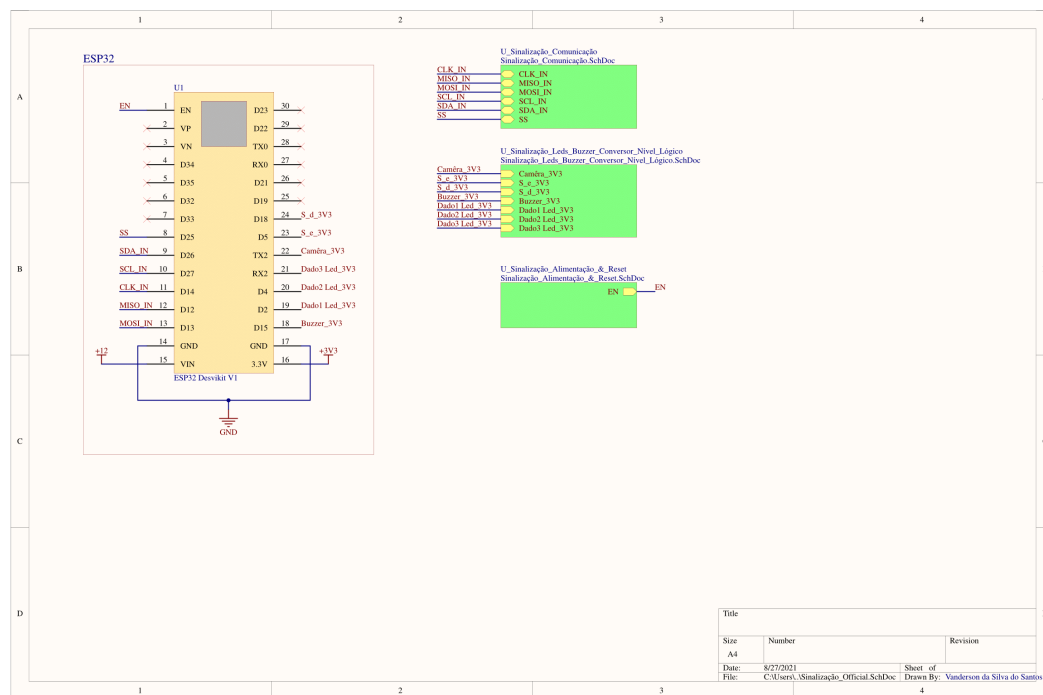
Figura 71: Placa de Percepção Externa - Esquemático 4



Fonte: Elaborada pelo autor no software Altium Design(?)

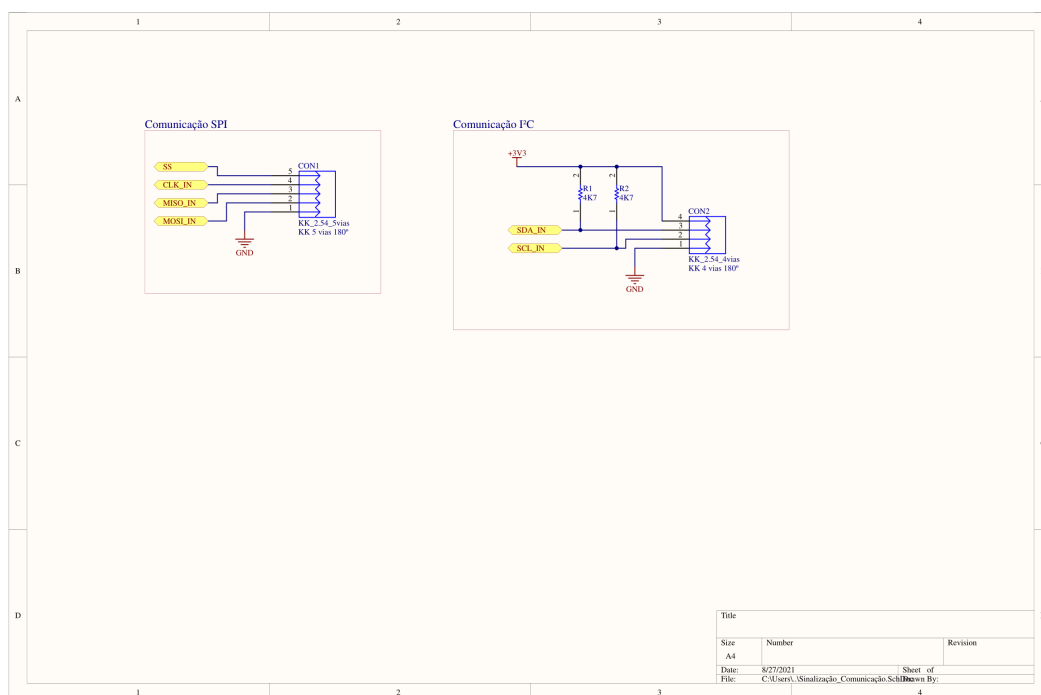
A.3 Sinalização

Figura 72: Placa de Sinalização - Esquemático 1



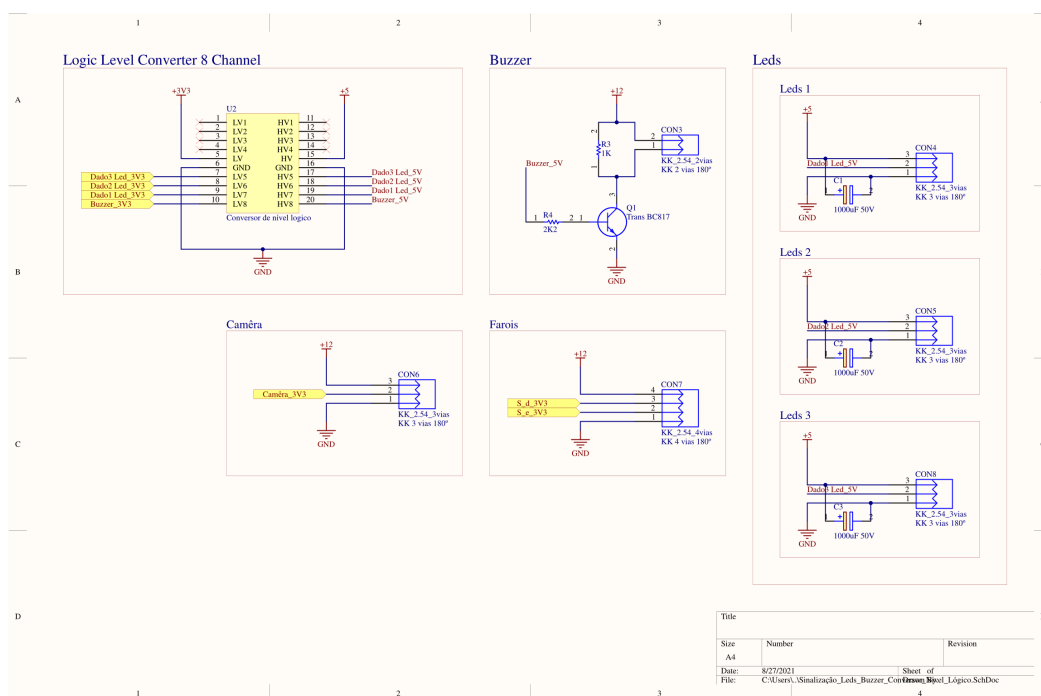
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 73: Placa de Sinalização - Esquemático 2



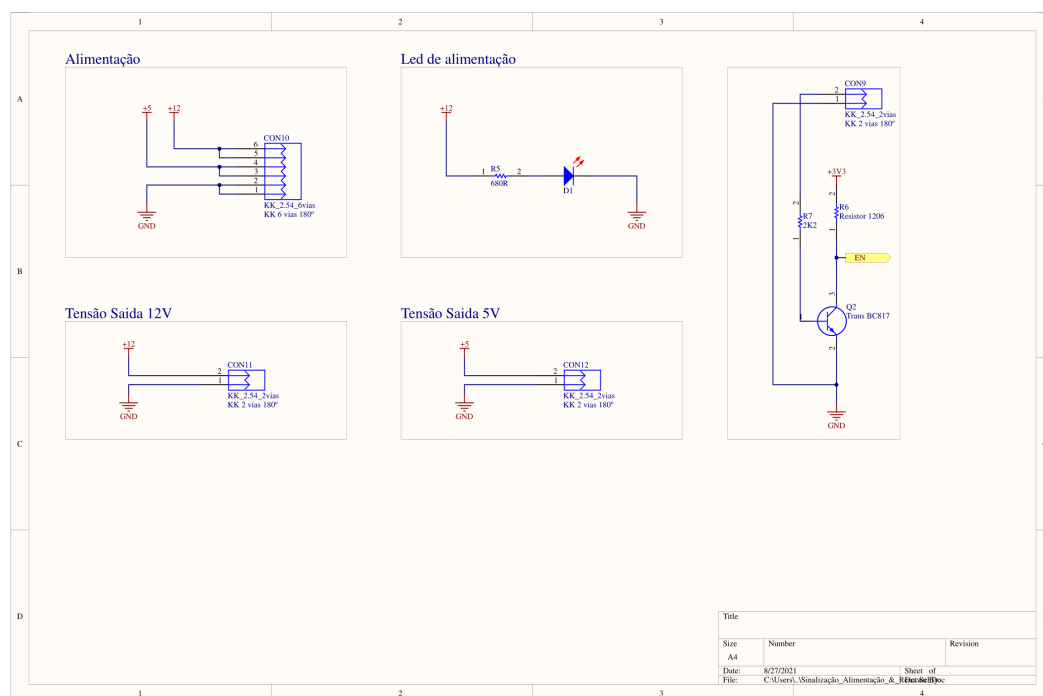
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 74: Placa de Sinalização - Esquemático 3



Fonte: Elaborada pelo autor no software Altium Design(?)

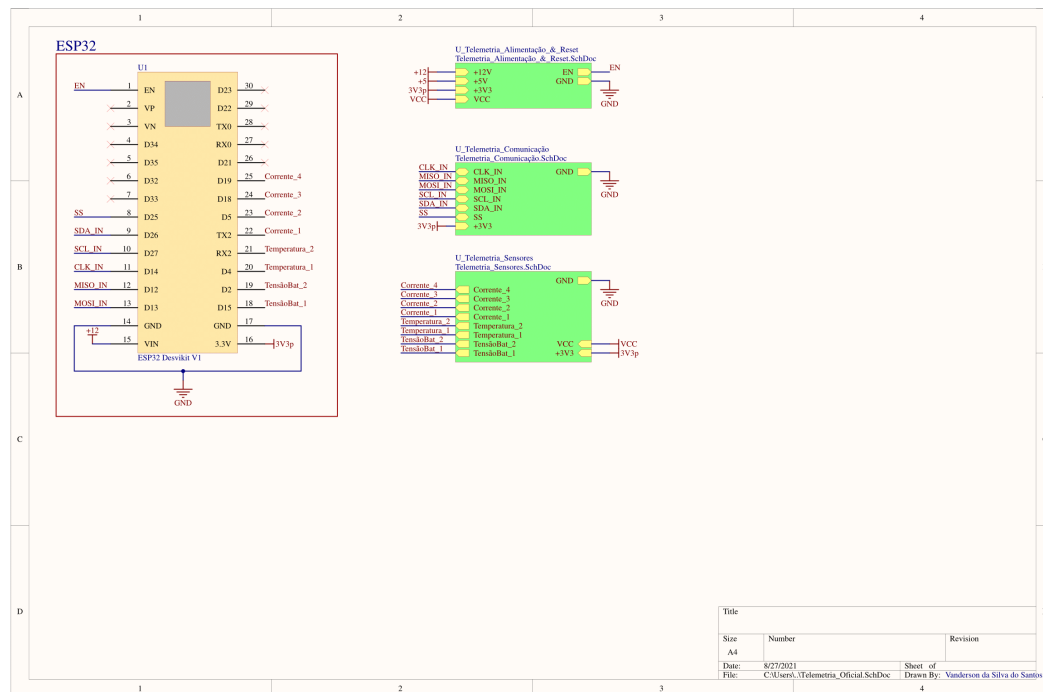
Figura 75: Placa de Sinalização - Esquemático 4



Fonte: Elaborada pelo autor no software Altium Design(?)

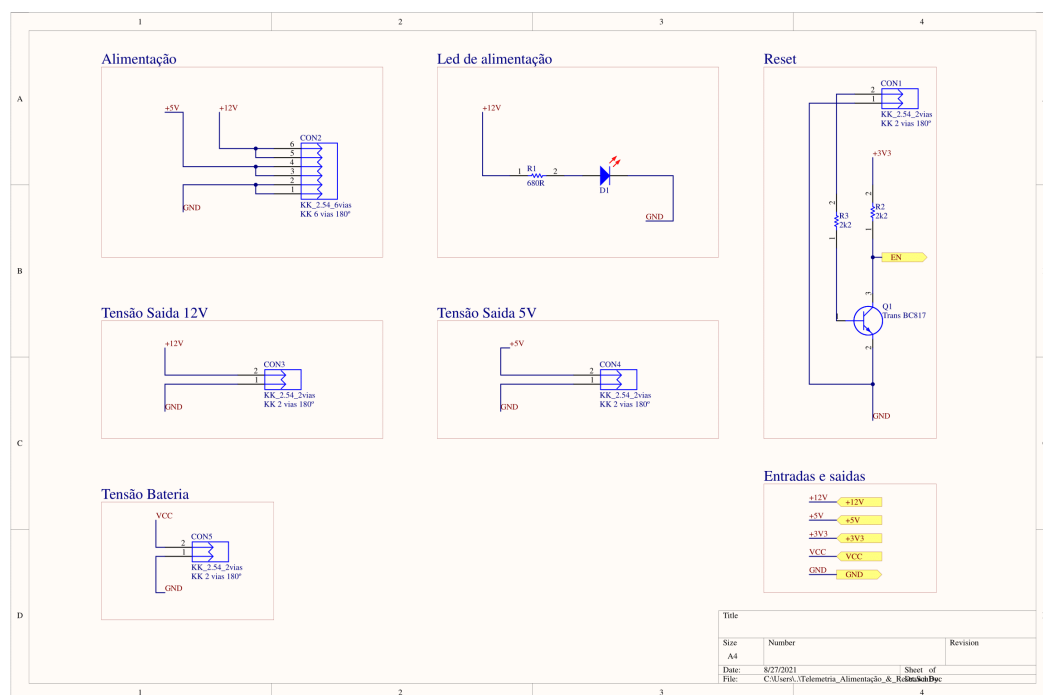
A.4 Telemetria

Figura 76: Placa de Telemetria - Esquemático 1



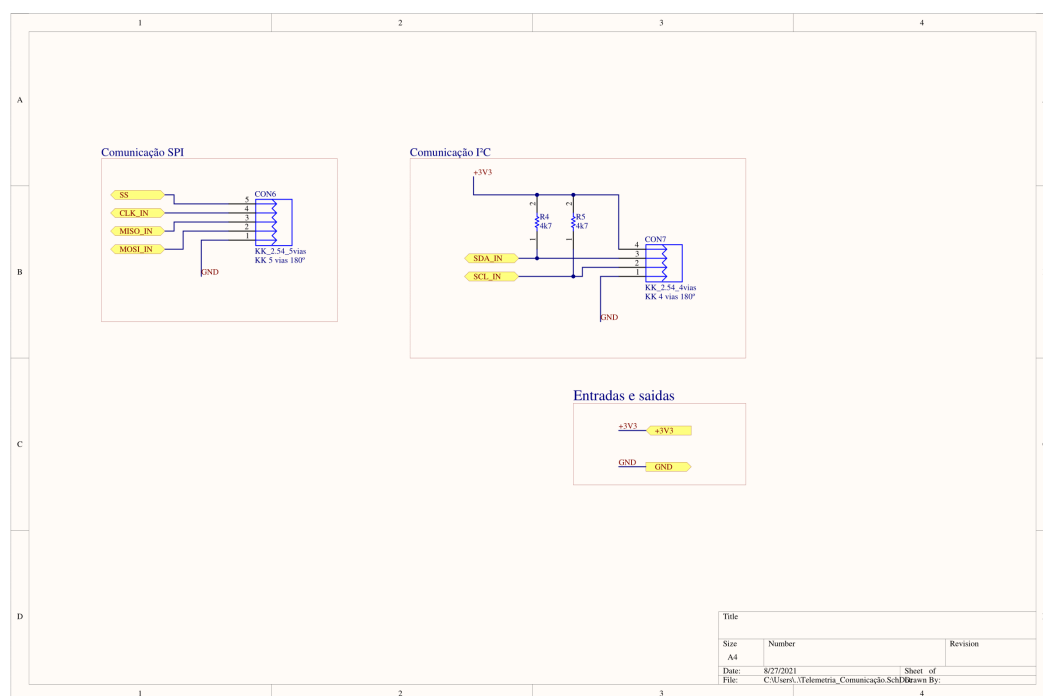
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 77: Placa de Telemetria - Esquemático 2



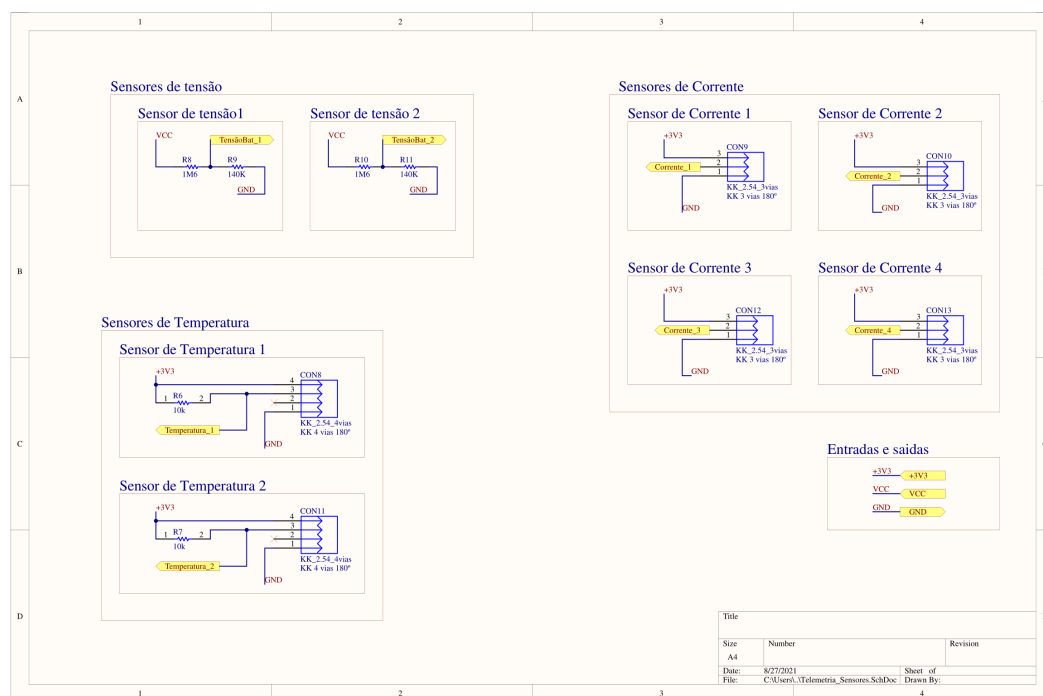
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 78: Placa de Telemetria - Esquemático 3



Fonte: Elaborada pelo autor no software Altium Design(?)

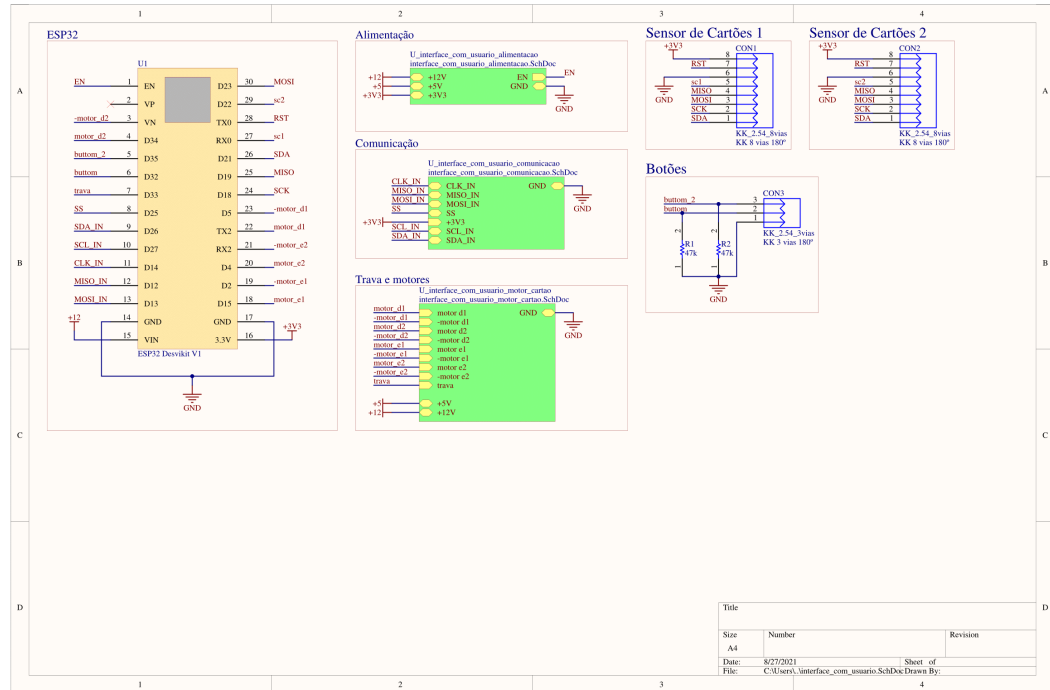
Figura 79: Placa de Telemetria - Esquemático 4



Fonte: Elaborada pelo autor no software Altium Design(?)

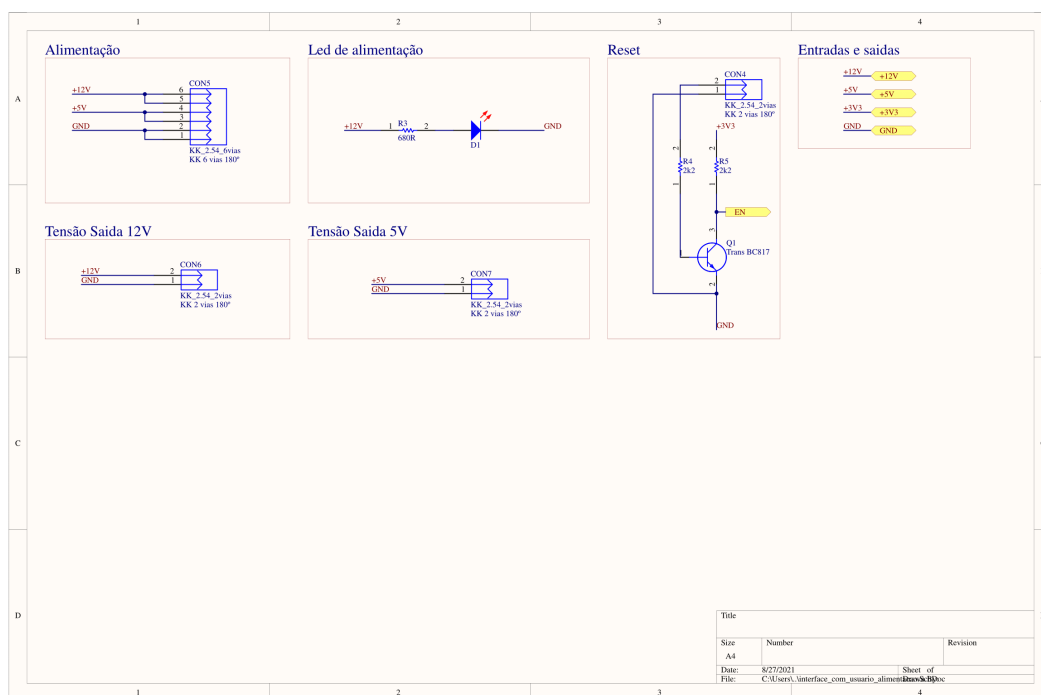
A.5 Interface Com Usuário

Figura 80: Placa de Interface Com Usuário - Esquemático 1



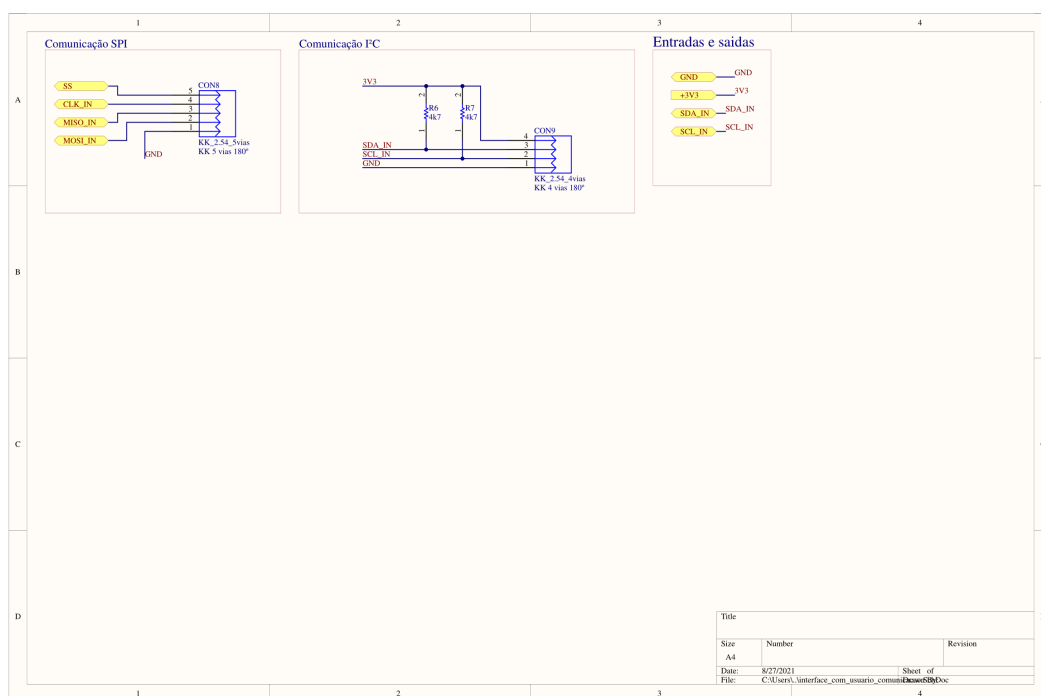
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 81: Placa de Interface Com Usuário - Esquemático 2



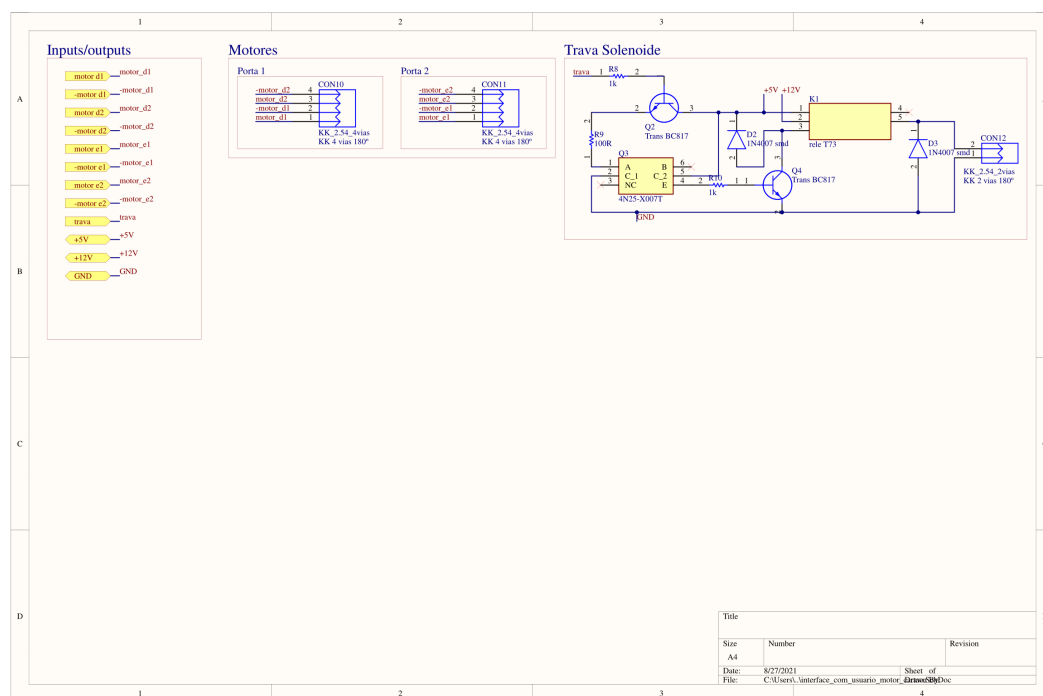
Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 82: Placa de Interface Com Usuário - Esquemático 3



Fonte: Elaborada pelo autor no software Altium Design(?)

Figura 83: Placa de Interface Com Usuário - Esquemático 4



Fonte: Elaborada pelo autor no software Altium Design(?)