UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LEONARDO ALVES DA FONTOURA

TELEOPERAÇÃO DE ROBÔS: COMANDO REMOTO POR IMAGEM E PERSPECTIVA PARA O DESENVOLVIMENTO DA ROBÓTICA COLABORATIVA

CURITIBA

2023

LEONARDO ALVES DA FONTOURA

TELEOPERAÇÃO DE ROBÔS: COMANDO REMOTO POR IMAGEM E PERSPECTIVA PARA O DESENVOLVIMENTO DA ROBÓTICA COLABORATIVA

Remote operation of robots: Remote control by image and perspective for development of collaborative robotics

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Controle e Automação do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Marco Antonio Busetti de Paula.

CURITIBA

2023

4.0 Internacional

Esta licença permite remixe, adaptação e criação a partir do trabalho, para fins não comerciais, desde que sejam atribuídos créditos ao(s) autor(es) e que licenciem as novas criações sob termos idênticos. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

LEONARDO ALVES DA FONTOURA

TELEOPERAÇÃO DE ROBÔS: COMANDO REMOTO POR IMAGEM E PERSPECTIVA PARA O DESENVOLVIMENTO DA ROBÓTICA COLABORATIVA

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Controle e Automação do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 24/Novembro/2023

_____

Marco Antonio Busetti de Paula
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Daniel Balieiro Silva
Mestrado
Universidade Tecnológica Federal do Paraná

_____

Walter Denis Cruz Sanchez
Doutorado
Universidade Tecnológica Federal do Paraná

CURITIBA

2023

Dedico este trabalho à minha família e amigos que me incentivaram e apoiaram durante toda a jornada deste curso e para o êxito deste trabalho.

AGRADECIMENTOS

Agradeço a Deus pela sabedoria, saúde e disposição para alcançar os resultados e poder concluir esta jornada.

Agradeço aos meus familiares e amigos que compartilharam comigo todos os momentos e que contribuíram com este trabalho e que me motivaram para concluí-lo nos momentos difíceis.

Agradeço ao meu orientador Prof. Marco Antonio Busetti de Paula pela dedicação e compartilhamento de seus conhecimentos e visão.

Enfim, agradeço a toda a comunidade acadêmica e tecnológica que foram fundamentais provendo ferramentas para este trabalho, e àqueles que contribuíram com esta pesquisa.

Sucesso é o estado de espírito resultante da consciência que você tem de haver se empenhado para ser o melhor que é capaz de ser.
(WOODEN, 2005)

RESUMO

A transformação tecnológica movida a partir do contexto de indústria 4.0, além dos aspectos produtivos, traz consigo a discussão da ressignificação da interação humano-máquina, visto a presença crescente de robôs e dispositivos inteligentes nos mais diversos âmbitos, de forma que funções mandatoriamente operacionais, e/ou repetitivas, são atribuídas a estes, e agentes humanos assumem funções estratégicas. Neste sentido, outro aspecto relevante é a preparação e capacitação destes profissionais, tanto em nível técnico quanto comportamental, num cenário tão dinâmico e disruptivo, sendo que ainda não há estabelecida uma base formal para tal. Entretanto, entende-se que é fundamental que conceitos de engenharia estejam devidamente difundidos para o sucesso na construção deste modelo de times híbridos. Assim este trabalho visa apresentar um modelo introdutório pelo desenvolvimento de um protótipo capaz de agregar diversas funcionalidades englobadas dentro da indústria 4.0 e discutir como tais podem ser integradas em novas soluções, sendo conduzido sob uma abordagem de projetos que permita identificar tecnologias em destaque, tal qual recursos de visão computacional que aumentam a autonomia de robôs e habilitam, por exemplo, a teleoperação destes por intermédio de inteligências artificiais e protocolos de comunicação, e que implementados proporcionam um modelo flexível para capacitação e treinamento de profissionais humanos. Além disso, há o levantamento da discussão de tópicos complementares, tais como a segurança humano-máquina, novos meios de comunicação e gestão de dados e seu potencial de escalabilidade, de maneira que se possa difundir e popularizar conhecimentos técnicos e incentivar o planejamento de um ecossistema voltado ao desenvolvimento da robótica colaborativa.

Palavras-chave: robótica; redes neurais (computação); visão por computador; aprendizagem industrial; automação industrial; grupos de trabalho.

ABSTRACT

The technological transformation driven by the context of industry 4.0, in addition to the productive aspects, brings with it the discussion of the resignification of human-machine interaction, given the growing presence of robots and intelligent devices in the most diverse areas, so that mandatorily operational and/or repetitive functions are assigned to them, and human agents assume strategic functions. In this sense, another relevant aspect is the preparation and training of these professionals, both at a technical and behavioral level, in such a dynamic and disruptive scenario, and there is still no formal basis established for this. However, it is understood that it is essential that engineering concepts are properly disseminated for the success in building this model of hybrid teams. Thus, this work aims to present an introductory model for the development of a prototype capable of aggregating several functionalities encompassed within industry 4.0 and discuss how these can be integrated into new solutions, being conducted under a project approach that allows the identification of highlighted technologies, such as computer vision resources that increase the autonomy of robots and enable, For example, the teleoperation of these through artificial intelligence and communication protocols, which implemented provide a flexible model for the qualification and training of human professionals. In addition, there is a survey of the discussion of complementary topics, such as human-machine security, new means of communication and data management and their scalability potential, so that technical knowledge can be disseminated and popularized and the planning of an ecosystem aimed at the development of collaborative robotics can be encouraged.

Keywords: robotics; neural nets (computer science); computer vision; industrial education; industrial automation; work groups.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ARM	*Advanced RISC Machines*
COBOT.	*Collaborative Robot*
CPU	*Central Processing Unit*
CV	*Computer Vision*
DL	*Deep Learning*
EAP	Estrutura Analítica de Projetos
ESP	*Espressif Systems*
EUA	Estados Unidos da América
GPU	*Graphic Processing Unit*
IA	Inteligência Artifical
ICW	*Institute for Collaborative Working*
IDE	*Integrated Development Environment*
IHR	Interação Humano-Robô
ISO	*International Organization for Standardization*
ML	*Machine Learning*
MQTT	*Message Queuing Telemetry Transport*
OpenCV	*Open Source Computer Vision*
PC	*Personal Computer*
PSAI	Poliestireno de Alto Impacto
QFD	*Quality Function Deployment*
RIA	*Robotics Institute of America*
SSC	Sistema, subsistemas e componentes
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

Nesta seção, visa-se contextualizar o tema em discussão, juntamente a suas justificativa e apresentar o formato de abordagem a partir dos objetivos estabelecidos e procedimentos e estruturas que serão utilizados ao longo do desenvolvimento.

1.1 Contexto

As revoluções industriais representam quebras de paradigmas no âmbito produtivo e social, trazendo sempre a inovação em sua essência, foi assim desde a máquina a vapor até as produções enxutas. De forma similar uma nova revolução ganha força propondo novas tendências a partir do desenvolvimento e combinação de tecnologias estimando que seus efeitos atinjam todas as atividades econômicas, mas principalmente reconfigurando o setor industrial e exigindo novas competências (INSTITUTO DE ESTUDOS PARA O DESENVOLVIMENTO INDUSTRIAL, 2019).

Concebida na Alemanha em 2011, mais precisamente em Hannover, a indústria 4.0 apresentava-se como uma estratégia de alta tecnologia (ZHOU, LIU, ZHOU, 2015), que abrangendo um vasto conjunto de tecnologias, propõe a conexão entre máquinas, pessoas e sistemas ao sistema produtivo. Desta forma, alterando a cadeia produtiva como um todo (SCHMIDT et al, 2015) e consolidando o conceito de indústria inteligente ao inserir altos níveis de automação aliados a computação que resultam em sistemas cibernéticos-físicos que atendam a uma manufatura mais dinâmica.

Essas características impulsionaram o estudo e desenvolvimento de inteligência artificial a ponto de podermos dizer que esta é a nova eletricidade (LYNCH, 2017). Sendo estudada desde meados de 1950, a IA começou investigando a natureza da inteligência através do uso de computadores simulando, e exibindo, de fato sinais de inteligência, tendo por fundamento base a capacidade de construção de sistemas inteligentes que tenham capacidade de aprender, seja pela execução de tarefas, por exploração da natureza ou mesmo pela demonstração de inteligência humana (SIMON, 1995). Estima-se que tais avanços em IA coloque em risco cerca de 47% dos empregos nos EUA, e uma elevada taxa também em outros países desenvolvidos, mudando significativamente a natureza do trabalho (FREY; OSBORNE, 2016).

Uma tendência que mostra o comprometimento com o conceito da indústria 4.0 é a do crescimento do número de robôs. Segundo relatório da Federação Internacional

de Robótica de 2018, as vendas de robôs fecharam o quinto ano de crescimento, apresentando um aumento de 30% em relação a 2016, este comportamento vem sendo notado desde 2010, quando iniciaram-se os desenvolvimentos de melhorias para robôs industriais, sendo estes os principais destaques devido à demanda dos setores da indústria metal mecânica, elétrica e eletrônica. Tais resultados certamente têm ligação com os novos recursos introduzidos aos robôs que lhes dão maior flexibilidade, autonomia e cooperatividade impulsionados principalmente por Machine Learning e diversas áreas de IA (SHEHADEH et al., 2017). O sentido de cooperatividade surge como destaque fomentando tecnologias de dispositivos assistivos inteligentes; dentro desta gama de dispositivos há uma nova classe de robô, os chamados *cobots* (nomenclatura que vem do inglês pelo jogo de palavras entre Collaborative e Robots), estes vêm a partir da nova proposta de ambiente de trabalho onde humanos e robôs compartilham o mesmo espaço (AKELLA et al., 1999).

No entanto, entre a migração de sistemas robóticos até a autonomia pretendida com os *cobots* há uma série de conceitos e estruturas a serem estabelecidas, dentre elas, e como ponto de partida, a teleoperação de robôs que envolve as dinâmicas de teleoperação e telepresença, que ampliam a abrangência de controle e manipulação autônoma (HAFIANE et al., 2013). Além das tecnologias de comunicação desenvolvidas a fim de viabilizar a telerobótica, outros meios que potencializam o valor e os resultados desta são relacionados ao desenvolvimento de reconhecimento por imagem por meio de visão computacional que permite comandos complexos e reduzem a necessidade de equipamentos ou manipuladores de controle (HAFIANE et al., 2013).

Neste cenário de modernização do ambiente de trabalho, enxerga-se, portanto, a colaboração humano-robô como um ativo de grande valor na fundação da indústria 4.0, onde novos times híbridos surgem cooperando de forma dinâmica e aprendendo com as experiências e um novo campo de pesquisa cresce, o de desenvolvimento de IA e robótica (RICHERT et al., 2016).

1.2 Problemas e Premissas

Para Zhong et al. (2017) uma nova geração de indústria surge no panorama da quarta revolução industrial com potencial de ampliação de flexibilidade de manufatura e melhorias de qualidade e produtividade, e estes seriam os pilares de uma manufatura inteligente, todavia para alcançarmos tal nível de maturidade

produtiva há necessidade de tecnologias que permitam aprendizagem e variações a diferentes situações que através de comunicação direta possam resolver problemas de forma ágil. Assim um novo cenário de interação humano máquina se estabelece com cooperatividade e alta integração.

Shedadeh et al. (2017) ressaltam que a formação de times híbridos no ambiente industrial não tem uma definição precisa dentro da indústria 4.0 e confunde-se aos conceitos de automação total, mas temos de relembrar o ímpeto de cooperação que carrega as premissas de confiança, troca de informações, medidas de desempenho, planejamento e previsão que nos remetem, juntamente a questão de segurança, aos desafios a serem enfrentados e colocam as seguintes perguntas para reflexão: “Como as máquinas seriam capazes de aprender e colaborar efetivamente com as pessoas dentro dos processos industriais?” e “Como as pessoas serão preparadas e se comportarão frente a robôs cada vez mais inteligentes?.

1.3 Objetivos

1.3.1 Objetivo Geral

Projetar um protótipo de robô educacional operado remotamente por visão computacional.

1.3.2 Objetivos Especifícos

- Integrar visão computacional ao robô proposto;
- Aplicar redes neurais para identificação de gestos e posição;
- Desenvolver um software de controle para o robô;
- Integrar o software desenvolvido a um robô já construído.
- Integrar robô e controle remoto por meio de protocolo MQTT

1.4 Justificativa

Em um momento no qual a automação vem crescendo e elevam os argumentos de um novo modelo produtivo, pensar em equipes híbridas amplia o ponto de vista sobre a manufatura inteligente e desenvolve um ponto chave de longo prazo para se atingir os objetivos de fábricas inteligentes e sistemas cyber-físicos.

Um árduo caminho ainda há de ser perseguido até a implementação da indústria 4.0 no ambiente fabril, nota-se cada vez mais a presença de máquinas e

robôs nesse sentido. Esta tendência é um passo importante, porém tal movimento se mostra muito mais ligado a terceira revolução industrial ao invés da quarta revolução. A indústria 4.0 tem fortemente arraigada a questão da digitalização e de inteligência em todos os níveis, a comunicação e as decisões assumem muito mais agilidade e confiabilidade e assim a inteligência artificial é posta quase como onipresente na organização. A partir disto, a pesquisa de tecnologias e sistemas adentra o chão de fábrica e vai além, rompendo os leiautes onde robôs e pessoas operam em passos paralelos ou seriados e colocando-os no nível de agentes dentro de um sistema de operação descentralizado.

Neste trabalho busca-se discutir os tipos de relações humano-máquina na indústria e como estas devem ser afetadas a partir da consolidação dos conceitos de indústria 4.0, principalmente em termos de robótica colaborativa e inteligência artificial, por meio do desenvolvimento de um protótipo teleoperado para fins educacionais e de capacitação, trazendo funcionalidades e recursos comuns à indústria integradas sob um sistema inteligente simplificado, flexível e seguro para a formação.

1.5 Procedimentos Metodológicos

A abordagem do tema será realizada de forma a construir um raciocínio linear relacionando os tópicos discutidos durante o trabalho para que ao fim tenha-se conceitos claros e consolidados embasando a elaboração de um modelo próprio.

A obtenção de informações será advinda da análise de artigos, relatórios, fóruns, teses, dissertações e manuais técnicos os quais serão a base de levantamentos estatísticos e das proposições de cenários futuros que motivam a discussão ao longo do trabalho. A relevância dos dados e informações será discutida com o orientador em reuniões e com profissionais da linha de pesquisa para que se tenha embasamento e credibilidade ao que se refere a presente análise.

A elaboração de um modelo consistirá na convergência de pontos reiterados nas bases teóricas analisadas, que suportarão a definição de prioridades direcionando as tecnologias e abordagens que mais se correlacionem a proposta em si.

Por fim, discutir-se-á os resultados obtidos e a importância do trabalho em concordância aos planos de indústria 4.0, e a realização de um feedback validando se os resultados esperados ao início do trabalho foram ratificados e alcançados e se novas discussões vieram a enriquecer e modificar a linha de pesquisa.

1.6 Estrutura do trabalho

O trabalho será subdividido buscando a construção fundamentada de raciocínio para a elaboração do modelo pretendido. Sendo assim dividido:

- Capítulo 1 - Introdução: Contextualização do tema a ser desenvolvido e síntese dos conhecimentos fundamentais do trabalho bem como breve descrição das motivações e objetivos.
- Capítulo 2 – Revisão Bibliográfica: Apresentação das bases teóricas relevantes aos conceitos de inteligência artificial, robótica e meios de comunicação orientadas ao desenvolvimento de robô de operação remota.
- Capítulo 3 – Modelamento: Definição das características do modelo em desenvolvimento, desde princípios construtivos a integração de sistema e processamento, e da metodologia de validação.
- Capítulo 4 – Conclusões: Apresentação e discussão sobre os resultados da pesquisa e fatos notáveis, juntamente a sugestões de temas correlatos e sequência de pesquisa.

Ao fim do trabalho são indicados as Referências, Anexos e os Apêndices envolvidos.

2 REVISÃO BIBLIOGRÁFICA

Nesta seção são apresentados contextos referentes a proposta principal do trabalho, bem como o estado da arte de desenvolvimento da área de conhecimento em discussão afim de suportar a compreensão da implementação final deste trabalho e desafios que visam ser superados.

2.1 Times Híbridos: Colaboração e eficiência na indústria 4.0

Os princípios da indústria 4.0 estão cada vez mais presentes no cotidiano das empresas, sendo notável pelos esforços direcionados a digitalização, automação e reestruturação organizacional. Para Kurt (2019) tal reestruturação e seus impactos frente ao mercado de trabalho e relações empregatícias ainda têm sido pouco estudadas, uma vez que tais adventos podem vir a desbalancear as forças de trabalho e as organizações, preocupações que segundo Ostergaard (2019) já motivam um novo movimento industrial, a priori denominado de indústria 5.0, o qual tem por motivação a revalorização do papel humano para a manufatura. A partir destes pressupostos é possível estabelecer o aprofundamento do tema compreendendo os efeitos em níveis organizacionais e como a colaboração como um todo seria afetada.

2.1.1 Colaboração e as estruturas organizacionais

Para traçar um panorama industrial e organizacional cabe resgatar significados e definições relacionadas principalmente a forma administrativa e a abordagem colaborativa intrínseca às organizações. O conceito mais simples de colaboração, segundo o ICW (*Institute for Collaborative Working*) é a atividade desenvolvida entre membros conjuntos com o intuito de atingir uma meta comum por meio de cooperação. Nesta realidade, os agentes passam a ter funções fundamentais e ainda mais amplas, inclusive os robôs que surgem como integrantes chave e trazem alternativas relevantes como os times híbridos.

As adaptações colocadas assumem o pressuposto de uma comunicação efetiva e interativa em ambientes comuns de trabalho a humanos e máquinas, condição ainda pouco explorada em relação aos modelos organizacionais tradicionais, este também começa a ser estudado como parte de uma nova revolução industrial que se constrói sendo denominada por autores, como Demir (2019), por indústria 5.0.

2.1.2 Relações Humano Máquina

Para compreensão dos efeitos 4.0 no ambiente de trabalho, tanto para humanos quanto para robôs, é fundamental recapitular as interações constituídas entre os agentes e processos envolvidos, em especial as de interface humano-robóticas, uma vez que estas são fator chave para habilitar a constituição de times híbridos no ambiente produtivo.

2.1.3 Interações humano-robóticas e os times híbridos

Goodrich e Schultz (2007) colocam a interação humano-robótica como o campo de estudo dedicado ao entendimento, desenho e levantamento dos sistemas robóticos para uso com, ou, por humanos através de uma relação de comunicação clara subdividida inicialmente em duas categorias: Interação remota e interação de proximidade que se diferem entre si em termos espaciais e/ou temporais, uma vez que, quando se trata de uma interação remota robôs e humanos não compartilham o mesmo espaço ao mesmo tempo, situação inversa quando em proximidade, na qual há relação de assistência efetivamente.

Sheridan (2016) expande a divisão ainda mais, segmentando afinal em quatro áreas de aplicação:

1. Controle supervisionado humano de robôs sob a execução de tarefas de rotina, o qual refere-se a maior gama de robôs aplicados até então pela variedade de aplicações, porém remetendo ao início da implementação robótica, visto que, caracterizam-se pela execução de tarefas repetitivas ou danosas aos humanos ao longo do tempo, que assim assumem papel de planejamento, controle e manutentores dos robôs.
2. Controle remoto de tarefas especiais em ambientes hostis ou inacessíveis, ou também chamados de telerrobôs, assumem funções inacessíveis ou perigosas aos humanos. Entretanto, tendem a possuir um controle supervisório operado por pessoas, pelo qual recebe os comandos e os reproduz.
3. Veículos automatizados, nessa categoria surgem os primeiros modelos de autonomia que utilizam recursos tecnológicos e de inteligência artificial para operarem sem interface humana direta, mas sim de forma assistencial.

4. Interação humano-robótica social, é a aplicação mais complexa para robôs na qual a inteligência artificial exigiria alto grau de autonomia e adaptação, permitindo que os robôs participem de tarefas de entretenimento, educacionais e assistência.

Yanco e Drury (2004) propuseram uma classificação utilizando conceitos taxonômicos para evidenciar a importância e os avanços em relação a comunicação e cooperação junto a sistemas inteligentes e tecnologias de IA. Tal modelo leva em conta onze classificações:

a) Tipo de tarefa: Caracterização específica da ação de trabalho que deve ser realizada que determina o desenho do sistema em geral e a real aplicação.

b) Criticidade da tarefa: Apesar de ser subjetivo o conceito de criticidade, em termos de IHR, refere-se a ações robóticas que possam representar risco a integridade e vida humana, e geralmente utiliza a escala entre Alto, Médio e Baixo.

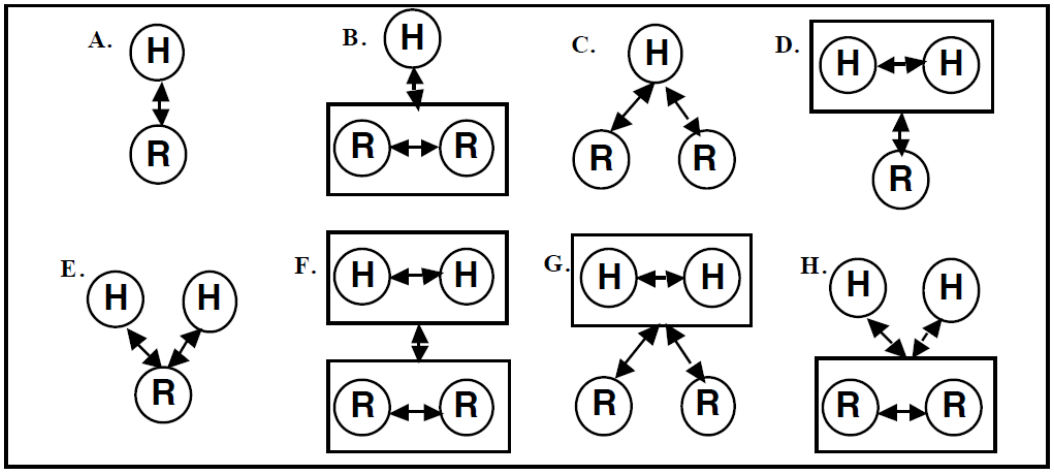
c) Morfologia robótica: Considera para definição a aparência física do robô classificando-a dentre antropomórfica, zoomórfica e funcional.

d) Razão de pessoas e robôs: A partir do número de indivíduos e robôs se estabelece uma proporção dadas as quantidades dos mesmos, ignorando inicialmente a interação dentre eles.

e) Composição robótica dos times: Dadas as diferentes aplicações e modelos robóticos um time, em geral, pode vir a apresentar mais de um tipo de robôs de forma heterogênea, ou até mesmo abranger mais de um robô de mesmo modelo colocados assim num formato homogêneo.

f) Nível de interação compartilhada entre o time: Além das quantidades em si, é necessário considerar o gerenciamento dentre todos os agentes e como eles interagem, priorizam e classificam as tarefas para execução. A partir da análise de interação obtém-se oito valores classificatórios: um humano e um robô, um humano e um time robótico, um humano e múltiplos robôs, um time humano e um robô, múltiplos humanos e um robô, equipe humana e equipe robótica, equipe humana e múltiplos robôs, múltiplos humanos e equipe robótica. Tais classificações podem ser representadas de forma visual conforme a figura 1.

Figura 1 - Combinações interativas humanos-robôs



Fonte: Yanco e Drury (2004)

g) Papéis de interação: O papel humano assumido ao longo da interação junto a robôs permite estabelecer cinco alternativas: supervisor, operador, companheiro de time, mecânico/programador e espectador, cada um com suas particularidades de interação e controle.

h) Tipo de proximidade: Muito intrínseca a interações físicas efetivamente, pelas quais é possível identificar cinco categorizações: alerta, passagem, acompanhamento, aproximação e contato direto.

i) Suporte a decisão operacional: Robôs podem também assumir papel importante ao fornecer informações ao longo do processo de trabalho, pois a variedade de sensores permite uma apresentação do status real de operação e mapeando o controle efetivo sobre tal.

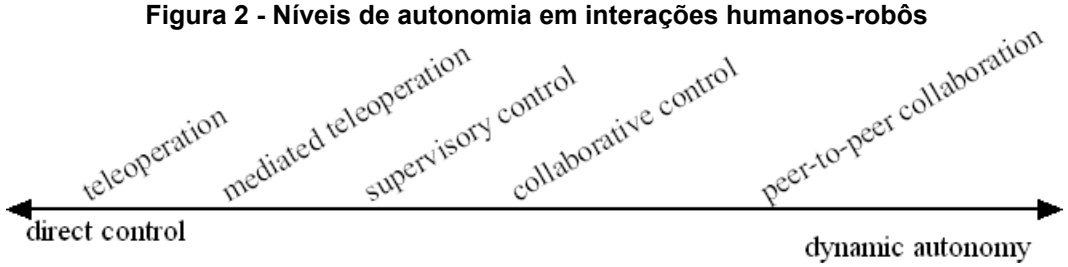
j) Tempo e espaço: Em efeito de trabalho, também se considera os fatores de tempo e espaço levantando a IHR em termos nos quais humanos e robôs compartilham o mesmo ambiente ao mesmo tempo, sendo que para tempo observa-se trabalhos síncronos e assíncronos enquanto para ambientação é estabelecido o formato de colocação ou não colocação.

k) Nível de autonomia e necessidade de intervenção: Autonomia e intervenção são conceitos conjuntos em IHR, dado que a autonomia de robô é a medida percentual de execução de tarefas sem a necessidade de ação humana de controle.

Através das classificações observadas, é perceptível a relevância da autonomia dos robôs, resumidas genericamente na figura 2, e de uma comunicação

clara e consistente para habilitação da formação de times híbridos, porém não são estes os únicos desafios a serem superados.

Figura 2 - Níveis de autonomia em interações humanos-robôs



Fonte: Goodrich e Schultz (2007)

2.1.4 Desafios

Em termos de comunicação a primeira barreira entre humanos e robôs é bastante significativa, pois como proporcionar percepção e entendimento às máquinas? A Indústria 4.0 traz recursos cada vez mais capazes de garantir tais capacidades, como IA, visão de máquina, realidade aumentada dentre tantas outras que reconheçam visão, sons, gestos e hábitos comuns e de linguagem natural que reconhecidos permitem a tomada de decisão e viabilizam um ambiente de trabalho saudável garantido com as devidas normas regulatórias e leis próprias e consolidam ainda mais a visão apresentada por Demir (2019) de indústria 5.0.

Todavia, não basta analisar apenas desafios técnicos e tecnológicos, uma vez que também temos a presença humana e sua contribuição na composição de times híbridos e cooperativos e que tal traz fatores comportamentais e subjetivos, pois humanos trazem consigo preferências, psicologias, aspectos sociais, ética e moral e que tantas variações podem carregar comportamentos resistivos e competitivos que precisarão ser trabalhados por um modelo educacional e de treinamento que inspirem confiança nas relações humano-robóticas.

2.2 Computação cognitiva e aprendizagem de máquina

2.2.1 Inteligência Artificial

Desde a proposição de Alan Turing de que as máquinas seriam capazes de pensar, cada vez mais estas são presentes no cotidiano e na sociedade em geral, de maneira que as chances de interação com tais inteligências virtuais são crescentes, como aponta Mou e Xu (2017), recordando o paradigma proposto por Nass et al. (1994) conhecido como Paradigma dos Computadores como Atores Sociais, que analisa o comportamento humano frente a interações com indivíduos virtuais.

O estudo conduzido por Mou e Xu, observou as variações do comportamento humano a interações comuns entre pessoas, porém realizadas frente a inteligências artificiais, notando que a diferença de interlocutor induzia uma resposta cognitiva afetiva diversa, nas quais constatou-se que ao primeiro contato os indivíduos apresentavam personalidades mais resistentes do que normalmente têm em relação ao contato humano social. Assim, mesmo com tantos avanços, como é possível que máquinas e robôs realmente aprendam? A resposta para este questionamento é ponto de partida para a disrupção no aprendizado e desenvolvimento de inteligências ainda mais elaboradas e complexas.

Para compreender melhor os avanços aplicados e como robôs e inteligências assistivas leem o ambiente em seu entorno e as expressões e gestos humanos e transformam tais informações em dados e sequências a serem executadas, retoma-se as definições de aprendizagem e cognição sob a ótica da psicologia em si. Bock et al. (2001) classificam as teorias de aprendizagem dentre dois grupos de estudo: teorias de condicionamento e teorias cognitivas. As teorias de condicionamento são orientadas pelo conceito de estímulo e resposta considerando fatores externos e de ambiente de alta relevância na construção do aprendizado, enquanto isso as teorias cognitivas são orientadas de forma mais introspectiva e definem a aprendizagem a partir da relação do indivíduo com o meio e como estas se desenvolvem no plano de organização do conhecimento. Ainda neste raciocínio, há uma tênue diferenciação entre cognição e aprendizagem, pela qual cognição é o conjunto de significados que o indivíduo atribui a realidade na qual se encontra e aprendizagem é a organização e integração deste conjunto de informações.

Wang (2009) elabora o conceito de computação cognitiva e como tal constrói o pensamento de máquinas a partir de inferências e percepções autônomas que visam

emular os mecanismos cerebrais baseados em informática cognitiva e inteligência abstrata constituindo um conjunto de processos e mecanismos do cérebro, trabalhados por engenharia, na compreensão da inteligência natural sob um formato universal matemático que transcreva conhecimentos e comportamentos. Resumidamente, é a ciência que reproduz a inteligência natural em semântica matemática e lógica que pode ser interpretada por máquinas.

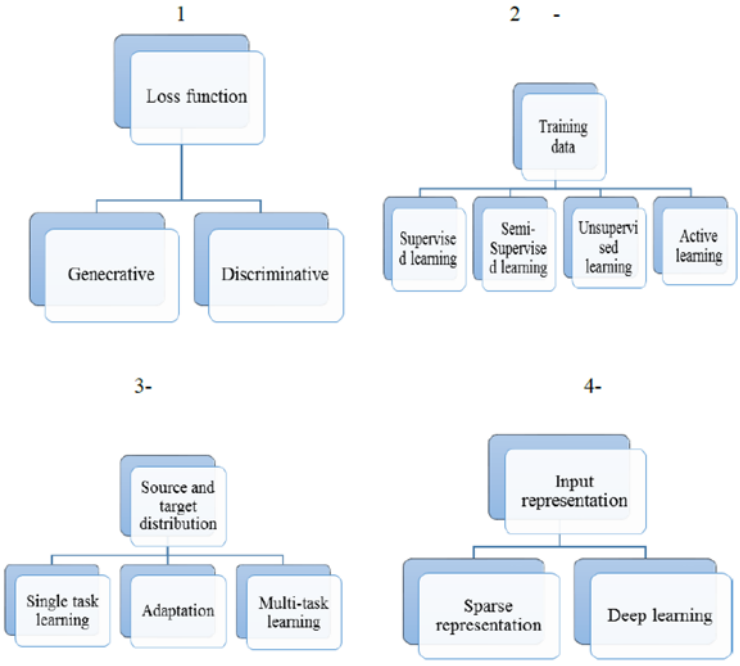
2.2.2 Aprendizagem de máquina (*Machine Learning*)

O conceito de *Machine Learning* (ML) é um derivado em si da definição de aprendizagem, que conforme Mishra e Gupta (2016) é a ciência que desenha máquinas inteligentes pela utilização de ferramentas como, por exemplo, redes neurais que permitem o aprofundamento e a interpretação da linguagem natural passando para um estágio conhecido como *Deep Learning*, que consiste num conjunto de técnicas de aprendizagem que se distinguem da aprendizagem convencional através dos processos de supervisão aplicados para a representação e classificação de modelos e vem se destacando por habilitar recursos como visão computacional e interpretação de voz. Tais técnicas aplicadas, geralmente, são constituídas por uma série de camadas de dados para processamento via arquiteturas niveladas para geração de conhecimento de forma não supervisionada.

Um breve modelo de como a área de *Machine Learning* se subdivide é mostrado na figura 3 e que representa a arquitetura de construção da inteligência utilizando os seguintes parâmetros:

1. Função de perda
2. Base de treinamento
3. Fonte e distribuição
4. Representação das entradas

Figura 3 – Classificação de *Machine Learning*



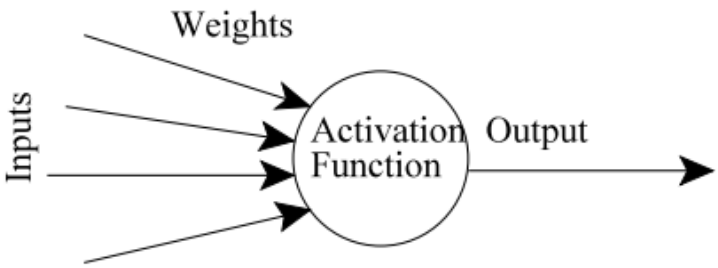
Fonte: Mishra e Gupta (2016)

2.2.2.1 Redes Neurais

Para a composição de uma rede em geral a ideia base é a desconstrução de um problema em partes reduzidas para simplificar a sua resolução, como coloca Gershenson (2003), de forma a se obter uma representação das relações entre tais elementos que são comumente chamados de nós e recebem sinais de entrada que resultarão em uma saída correspondente.

Um modelo bastante presente em aprendizagem de máquina e inteligência artificial é o de redes neurais que visa emular o processamento do cérebro humano de maneira computacional, utilizando nós que se assemelhariam muito aos neurônios humanos, porém estes utilizam processamento lógico e matemático a partir da ponderação dos sinais recebidos, que serão combinados por uma função de ativação que terá efetivamente o resultado de saída, comportamento que pode ser representado de forma simplificada pela figura 4.

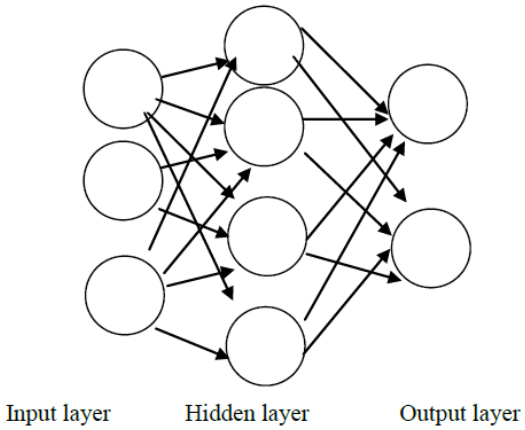
Figura 4 – Representação de neurônio artificial



Fonte: Gershenson (2003)

Entretanto, um nó em si não compõe uma rede como um todo, sendo, portanto, uma rede neural estabelecida pela conexão de vários destes elementos numa forma organizada que é, geralmente, posta sob três camadas: camada de entrada, camada oculta e camada de saída, Mishra e Gupta (2016), que podem ser observadas pela figura 5.

Figura 5 – Modelo geral de rede neural



Fonte: Mishra e Gupta (2016)

Cada uma das camadas é definida da seguinte maneira:

- Camada de entrada: Não é considerada propriamente uma camada de neurônios em si, uma vez que é caracterizada pela entrada de sinais do ambiente, sem a realização de qualquer processamento.
- Camada Oculta: Camada que define a topologia e as conexões entre nós, responsável pelo processamento das informações recebidas pela camada de entrada, porém sem conexão direta ao meio externo.
- Camada de saída: É a responsável pela transmissão da informação da camada oculta para o meio externo, ou seja, a resposta do sistema.

Um dos algoritmos comumente aplicados a redes neurais é o de propagação reversa por seu poder de aproximação dado por uma função de continuidade, porém muito direcionado a treinamento supervisionado devido a sua resposta computacional mais lenta e limitação de resolução de problemas de maior grau.

Gershenson (2003) explica que tal algoritmo se adequa as redes organizadas sob camadas, de maneira que o sinal siga propagado entre as mesmas e os erros sejam retroalimentados através de propagação reversa, convergindo a definição de Rosa (2013) em que tal mecanismo é descrito por dois passos: 1) Ativação propagada e 2) Propagação reversa de erros, tendo como objetivo a redução do erro geral.

Como mecanismo de treinamento supervisionado, há de ser considerada uma base de exemplo para comparação e ponderação, uma vez que se pode entender o algoritmo de propagação reversa como uma soma ponderada, Gershenson (2003), e assim torna-se possível a obtenção de uma função característica que calcula o erro resultante.

Redes de múltiplas camadas de processamento combinadas ao algoritmo de propagação reversa, compõem o que Mishra e Gupta (2016) definem como redes neurais convolucionais, e que são responsáveis pelo sucesso desta sistemática como meio de treinamento supervisionado de forma robusta. Além disso, tal viés multicamadas permite que redes deste tipo possam habilitar o processamento de dados bidimensionais, tais como imagens, sons e vídeos, a partir do processamento em regime de séries temporais, particionando a entrada e reduzindo as necessidades de cálculos de pré-processamento.

2.3 Visão computacional

Visão computacional tem sido um campo de estudo em ampla expansão, principalmente voltado a análise e tratamento de imagens que propiciem maior entendimento das informações registradas, permitindo o controle de computadores ou sistemas robóticos (PULLI et al, 2012). Sigut et al (2020) evidenciam tal crescimento pela presença de inteligência artificial além dos recursos industriais, tais como em redes sociais para identificação de usuários e modificação de imagens que demonstram o potencial e abrangência dos recursos visuais, porém que demandam o desenvolvimento de ferramentas e algoritmos para plena aplicação de seus recursos.

Um dos desafios enfrentados é o da eficiência energética que é trabalhado de dois modos:

1. Paralelização de dispositivos de computação, ou seja, replicação da unidade de processamento
2. Especialização de *hardware* que adiciona unidades aceleradoras de suporte

Este conceito é definido como computação paralela heterogênea e que é observado em sistemas de visão por arquiteturas utilizando CPUs e GPUs (PULLI et al, 2012).

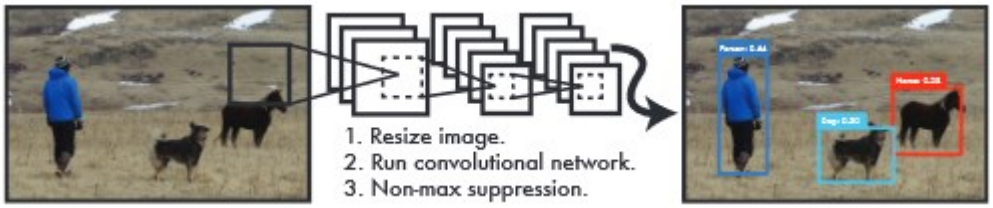
De outro ponto, há também a complexidade lógica para os algoritmos integrados presente no primeiro contato com esta tecnologia e que pode ser simplificado pela utilização de bibliotecas amigáveis à experiência de usuário, como, por exemplo, a *Open Source Computer Vision Library* (OpenCV) que é um conjunto de ferramentas de propósitos gerais e de ampla documentação para suporte ao desenvolvimento e aprendizagem prática de visão computacional, além de também ser compatível com dispositivos móveis (SIGUT et al, 2020).

2.4 Sistema de detecção YOLO (*You Only Look Once*)

O modelo de detecção YOLO foi desenvolvido em meados de 2016, por Redmond et al, como uma ferramenta de detecção rápida e com alta taxa de assertividade através do conceito base do sistema visual humano transformado em algoritmo (REDMOND et al, 2016). O escopo funcional inicial de YOLO pode ser sumarizado de maneira simples em três etapas, listadas a seguir e exemplificadas na figura 6:

1) Processamento e redimensionamento da imagem;
2) Aplicação de uma única rede neural convolucional a imagem;
3) Resposta resultante com as detecções e taxa de confiança.

Figura 6 – Algoritmo geral de YOLO

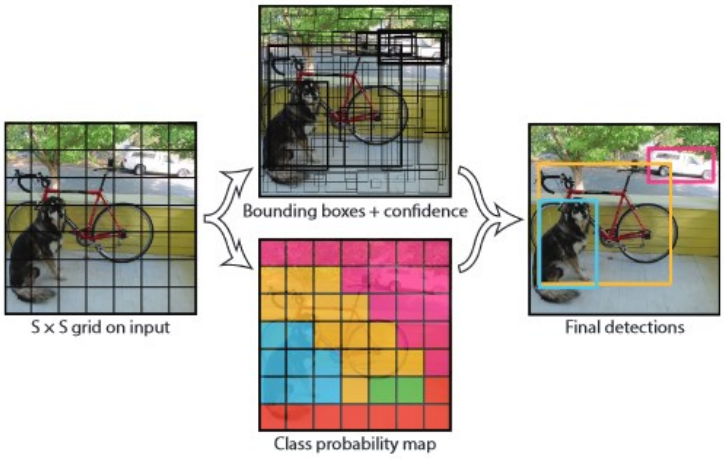


Fonte: Redmond et al (2016)

A partir delas se estabelece, o que Redmond et al definem como, detecção unificada que se caracteriza pela junção de diferentes componentes sob uma mesma rede neural.

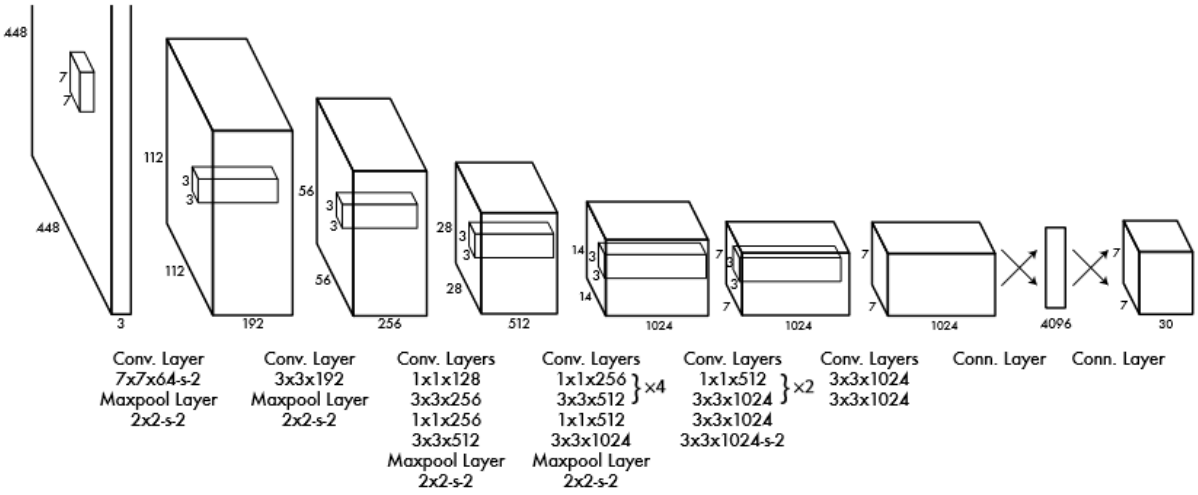
Tal rede neural utiliza dados de pré-treinamento para otimização e dados completos da imagem para predizer as caixas de detecção, taxa de confiança e o mapa de probabilidades resultantes, conforme exemplificado na figura 7, estando arquitetada inicialmente sob 24 camadas convolucionais mais duas camadas com conexão completa e que são representadas na figura 8.

Figura 7 – Representação de detecção YOLO



Fonte: Redmond et al (2016)

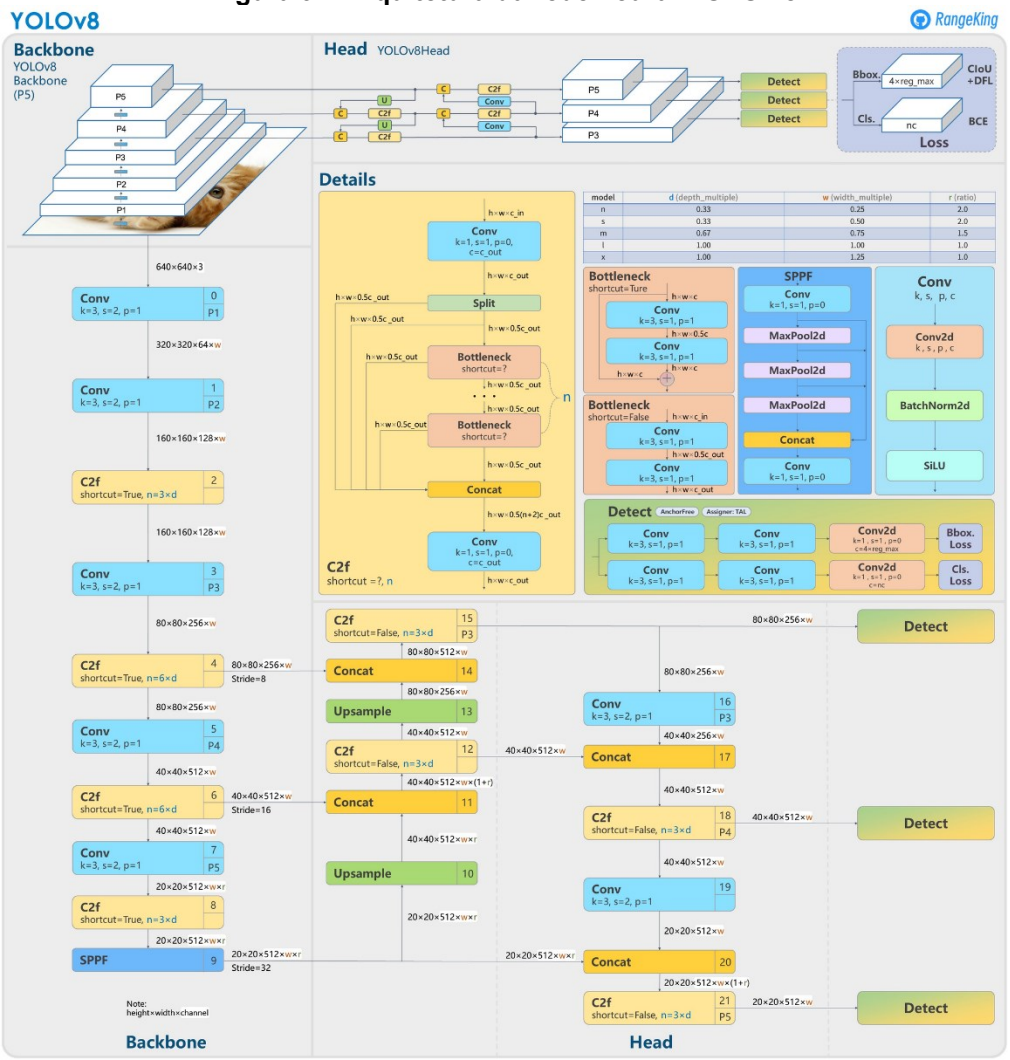
Figura 8 – Arquitetura da rede neural YOLO



Fonte: Redmond et al (2016)

Este modelo foi atualizado e encontra-se sob a versão 8, tendo passado desde atualizações de programação, como, por exemplo, compatibilidade a linguagem Python (YOLO v5), até reformatação de sua arquitetura (YOLO v7 e YOLO v8), a qual pode ser melhor representada pela figura 9.

Figura 9 – Arquitetura da rede neural YOLO v8



Fonte: RangeKing em Roboflow (2023)

2.5 Robôs – Definições e regulamentações

O termo “robô” tem origem tcheca introduzido por Karel Capek (1890 – 1938), em sua obra teatral *Rossum’s Universal Robots*, inicialmente associado ao conceito de um trabalhador escravo e que na robótica contemporânea visa substituir o trabalho humano em condições de risco (SANTOS et GORGULHO, 2015), dentro do paradigma das leis de Asimov, descritas abaixo:

1. Um robô não pode ferir um ser humano ou, por inação, permitir que um humano seja ferido;
2. Um robô deve obedecer às ordens dadas por humanos, exceto quando isto conflitar com a primeira lei;
3. Um robô deve proteger sua própria existência, a menos que isto conflite com a primeira ou segunda lei.

Em termos de definição do robô industrial, tem-se ainda o conceito estabelecido pelo *Robotics Institute of America* (RIA) de 1981, transcrita por Santos e Gorgulho (2015):

> Um robô industrial é um manipulador reprogramável, multifuncional, projetado para mover materiais, peças, ferramentas ou dispositivos especiais em movimentos varáveis programados para a realização de uma variedade de tarefas.

Esta definição, porém, foi ampliada pela *International Organization for Standardizatio*n (ISO) sob a norma ISO-10218 de 2011, e adaptada por Santos e Gorgulho (2015):

> Um robô industrial é uma máquina para manipulação, com vários graus de liberdade, controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial.

Quanto a aplicabilidade de robôs na indústria, tem-se dentre as atividades mais comuns controle de posicionamento e movimentação, que compreendem inclusive os processos de manipulação e medição em seu escopo, e que em termos técnicos apresentam resultados descritos no quadro 1 (SANTOS et GORGULHO, 2015).

Quadro 1 - Diferentes estruturas de organização

Característica	**Especificação**
Repetibilidade	décimos de milímetro
Velocidade	até 5 m/s
Aceleração	até 25 m/s²
Carga admissível	até centenas de kg
Relação Peso/Carga	em torno de 30 a 40
Número de eixos	6 (tipicamente)
Comunicação	Profibus, Ethernet, canais seriais (RS 232, 485)
Capacidades de E/S	Similares a um CLP para sinais analógicos e digitais

Fonte: Adaptado de Shedadeh et al. (2017)

2.6 Tele robótica

Dentre os modelos de comando de robôs na indústria, há os que se baseiam em comunicações sem fio e controle remoto, tal domínio é classificado como tele robótica por basicamente envolver duas áreas de estudo: Tele operação e telepresença, que visam o controle robusto a distância que formem um sistema robótico autônomo (HAFIANE, SALIH.et MALIK, 2013).

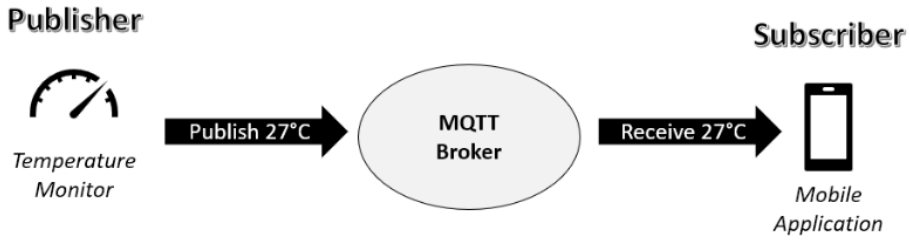
Contudo, mesmo com o desenvolvimento de serviços e tecnologias de comunicação remota, ainda não há um padrão estabelecido, ou uma unanimidade, quanto a fluxo de dados e confiabilidade, porém um modelo que se destaca para tal é o de comunicações *publish* e *subscribe* devido a sua simplicidade e escalabilidade (KIM, KANG et. PARK, 2013).

2.7 Protocolos de comunicação e MQTT

O desenvolvimento das tecnologias IoT *(Internet of Things*) teve grande crescimento recente dada a ampla demanda da indústria 4.0 em termos de comunicação, tais RFID (*Radio-frequency identification*) e sensores inteligentes, com o intuito de proporcionar maior inteligência a dispositivos e aproximá-los de camadas de tomada de decisão. Apesar disto, ainda há restrições técnicas que limitam a transmissão de dados entre dispositivos e redes, com isso um protocolo de comunicação popular é o MQTT (*Message Queuing Telemetry Transport*) (YASSEIN et ALJWARNEH, 2017).

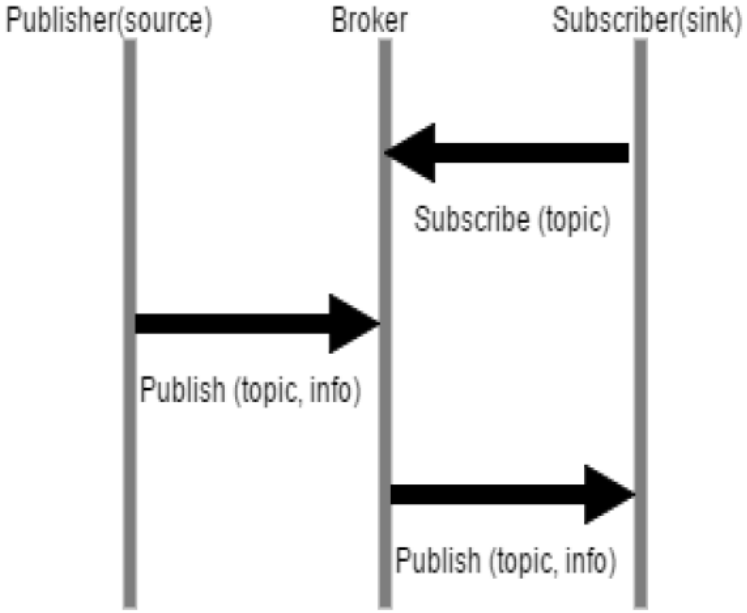
Tal aproveitamento do protocolo MQTT se dá por sua estrutura e modelo de comunicação baseada em publicação e inscrição representadas nas figuras 10 e 11 respectivamente.

Figura 10 – Exemplo de estrutura de comunicação MQTT



Fonte: Parikh (2022)

Figura 11 – Representação de comunicação MQTT



Fonte: Yassein et Aljwarneh (2017)

Neste modelo de comunicação pode-se identificar três agentes ativos:

- *Publisher*: Responsável pela coleta das informações enviando-os ao broker
- *Broker*: Responsável pela intermediação e conexão entre *Publishers* e *subscribers.*
- *Subscriber*: Responsável por monitorar as informações recebidas via *broker* e disparar ações programadas.

As informações transitam entre estes sob o formato tópico-mensagem, sob os quais *publishers* direcionam que informação está sendo compartilhada para a leitura dos *subscribers* que monitoram o tópico específico, permitindo alta escalabilidade com baixo consumo de banda.

3 MODELAMENTO

Para o desenvolvimento de um protótipo será utilizada a abordagem por meio do modelo de Estrutura Analítica de Projeto (EAP), que consiste em uma hierarquia do projeto desdobrada em entregas de trabalho orientadas ao objetivo do projeto sob níveis graduais detalhados (GAMA et al., 2015) e que permite a visualização do escopo geral de forma gráfica, além de abranger as cinco fases comuns a projetos: Iniciação, planejamento, execução, monitoramento e encerramento.

3.1 Estrutura Analítica de Projeto

Considerando o objetivo de construção de um robô de propósitos gerais e de funcionalidades múltiplas, operado de forma remota e voltado ao treinamento e capacitação de profissionais, elabora-se a EAP do projeto para estruturação do escopo de trabalho. Assim, têm-se a EAP constituída sob fases a serem desenvolvidas, sendo estas: Definição de requisitos, definição de especificações, implementação e testes.

3.1.1 Etapa de requisitos:

Nesta etapa serão levantadas as principais funcionalidades disponíveis no mercado e como estas podem ser abordadas no projeto através das seguintes ações:

- Pesquisa de mercado e soluções requeridas pelo público (Benchmarking)
- Obtenção de conceito de operação do produto (Protótipo)
- Obtenção de requisitos funcionais e não funcionais

3.1.2 Etapa de especificação:

Para a definição das especificações serão tomados os requisitos elencados na etapa anterior e realizada a estratificação deles em termos de controle e componentes, distribuídos sobre:

- Análise e estudo das variáveis do domínio do problema (velocidade / posição / massa)
- Configuração e programação do sistema
- Análise de comunicação e interface câmera / PC / robô
- Análise e elaboração do projeto do arranjo físico do produto

3.1.3 Etapa de implementação:

A partir das definições de requisitos e especificações, faz-se o trabalho de desenvolvimento considerando a conversão das funcionalidades em sistemas, subsistemas e componentes aplicados de forma discreta tendo soluções sob cada nível. Desta forma, os principais objetivos são:

- Análise e implementação do *firmware* relativo ao produto
- Análise e implementação do projeto físico do produto
- Análise e implementação da interface
- Montagem do protótipo físico

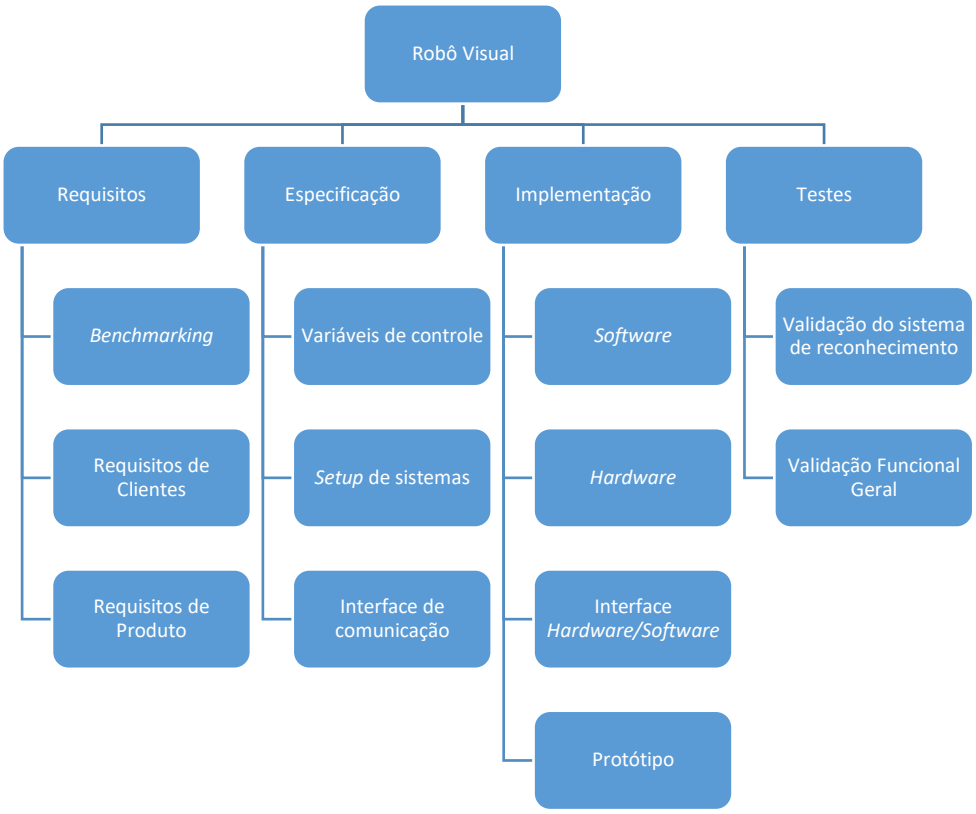
3.1.4 Etapa de testes:

Como etapa final do projeto têm-se a validação dos objetivos frente a comparação das metas esperadas com os indicadores obtidos, além disso, é validada a integração entre todos os módulos desenvolvidos, e assim podemos sumarizar as ações necessárias sob:

- Validação do sistema de reconhecimento;
- Validação funcional geral

Por fim, revisado todo o escopo do projeto pode-se representar a EAP em seu formato visual através de um diagrama que apresentado na figura 12.

Figura 12 – Diagrama EAP

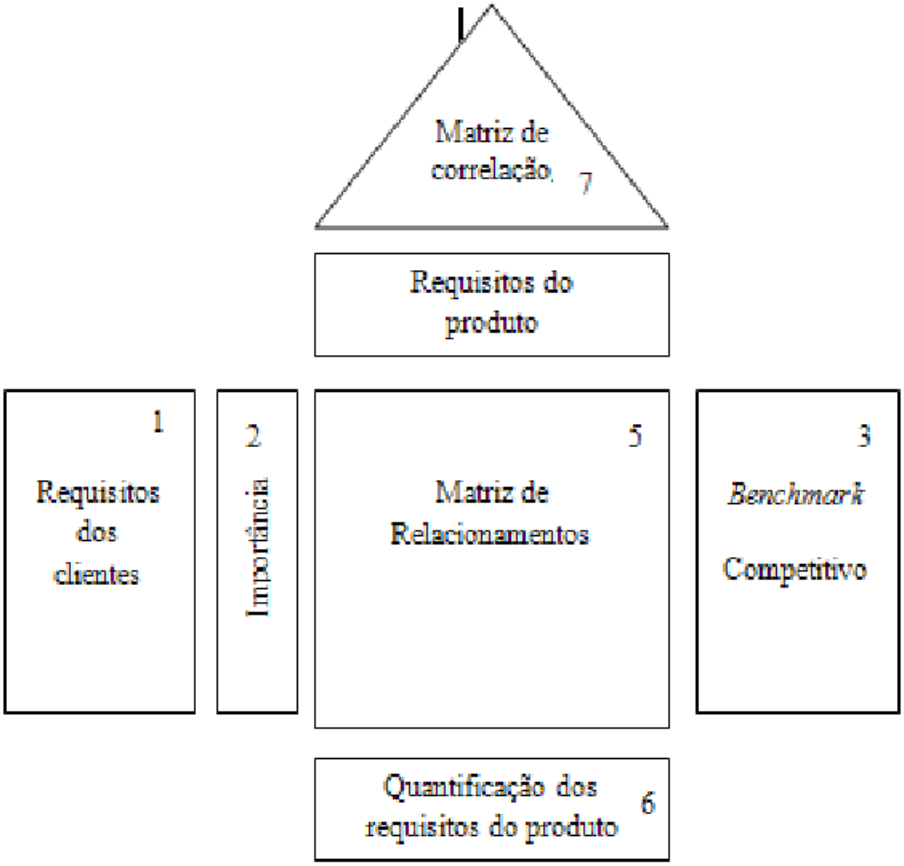


Fonte: Autoria própria (2022)

3.2 *Quality Function Deployment*

Para o mapeamento dos requisitos de forma estruturada e elaboração de suas correspondentes aplicações será utilizada a ferramenta de *Quality Function Deployment* (QFD), traduzida como desdobramento da função qualidade, e representada na figura 13. Este método permite identificar requisitos externos, relevantes ao mercado disponível ao que se insere o protótipo, quanto requisitos internos, relacionadas a funcionalidades específicas do projeto em termos de produto e processo (SALVADOR, 2015).

Figura 13 – Diagrama QFD



Fonte: Salvador (2015)

3.2.1 *Benchmarking*

Para o levantamento das características relevantes ao protótipo proposto, fez-se um comparativo com as principais características técnicas e recursos de *cobots* já existentes, a partir das fichas técnicas disponibilizadas pelos fabricantes, e que direcionariam as funcionalidades mandatórias dentro do espectro de desenvolvimento desde o contexto de teleoperação até o nível colaborativo para suportar o mapeamento e o plano de trabalho, sendo obtidos os resultados sumarizados no Quadro 2:

Quadro 2 – *Benchmarking* COBOTs

Característica	UR3E	LBR iiwa 7 R800	JAKA Zu 3s	HC10DTP	RV-5AS-D	TM5-700
Fabricante	Universal Robots	KUKA Robotics	JAKA	Yaskawa	Mitsubish Electric	OMRON
Capacidade de carga (kg)	3	7	3	10	5	6
Graus de liberdade	6	7	6	6	6	6
Alcance (mm)	500	800	626	1200	910	700
Repetibilidade (mm)	0,03	0,1	0,02	0,05	0,03	0,05
Temperatura de trabalho (°C)	0-50	-	0-50	0-40	0-40	0-50
Fonte de Energia (I/O)	24V (DC)	-	24V (DC)	-	24V (DC)	-
Velocidade Máx. (m/s)	1	-	1,5	-	1	1,1
Torque Máx. (Nm)	10	-	-	27,4	-	-
Programação	Interface Gráfica Licenciada	Interface Gráfica Licenciada	Interface Gráfica Licenciada Driver	Interface Gráfica Licenciada	Interface Gráfica Licenciada	Interface Gráfica Licenciada
Comunicação	500 Hz Control frequency Modbus TCP PROFINET Ethernet USB 2.0, USB 3.0	USB Ethernet DVI-I	TCP/IP Modbus TCP Modbus RTU	-	USB Ethernet RS-422	RS232 Ethernet Modbus TCP/RTU PROFINET
Mobilidade	Fixa	Fixa	Fixa	Fixa	Fixa	Fixa
Peso (kg)	11,2	22,3	12	48	32	22,1
Sistemas e interfaces de segurança	17 funções config.	KUKA Sunrise	MT (PAD/ Mobile) APP	Smart Pendant	Unidade de expansão de segurança	Câmera

Fonte: Autoria própria (2022)

3.2.2 Requisitos

A partir do *benchmarking* realizado, é possível efetuar a segmentação de requisitos dentre o que parte de demanda externa, funcionalidades desejadas pelos usuários e clientes, e o que se trata de requisitos específicos de desenvolvimento, ou seja, recursos e equipamentos que permitam o atendimento dos requisitos de cliente.

3.2.2.1 Requisitos de cliente

Tais requisites são levantados como oportunidades a atender critérios de satisfação dos usuários e clientes, sendo considerados principalmente em termos de aprendizagem e desenvolvimento comparados a produtos de maior maturidade já disponíveis como visto sob o *benchmarking*. Assim, têm-se em evidência os seguintes elementos sob o quadro 3:

Quadro 3 – Requisitos de cliente

ID	Requisito
RC1	Plataforma aberta
RC2	Mobilidade
RC3	Versatilidade de montagem
RC4	Baixa potência
RC5	Baixo custo
RC6	Conexão remota
RC7	Manutenção simplificada

Fonte: Autoria própria (2022)

3.2.2.2 Requisitos de produto

A partir da compreensão da necessidade dos usuários é possível trazer os requisitos em termos de elementos e componentes correlatos para atendê-los, sendo tais objetivos trabalhados como o detalhamento técnico/tecnológico das demandas percebidas e que são sumarizadas no quadro 4:

Quadro 4 – Requisitos de produto

ID	Requisito
RP1	*Firmware* em linguagem de plataforma aberta
RP2	Compatibilidade com microcontrolador ARM, Arduino, ESP
RP3	Memória ROM para armazenamento do *firmware*
RP4	Entradas e saídas digitais
RP5	Entradas e saídas analógicas
RP6	Mecanismo de *pick and place*
RP7	Câmera de reconhecimento de padrões - Visão computacional
RP8	Sensor ultrassônico - Sistema anticolisão
RP9	Conectividade remota - PC/*Smartphone*

Fonte: Autoria própria (2022)

3.2.3 Especificações meta do produto

Efetuando a estratificação dos requisitos e os reconhecendo em termos práticos, é possível delimitar o escopo do projeto e do desenvolvimento e com isto estabelecer parâmetros de validação da efetividade no atendimento do que foi proposto, de forma que tais metas são apresentadas no quadro 5 a seguir:

Quadro 5 – Especificações meta

ID	Especificação
EM1	Programação em *Python*
EM2	Alimentação por baterias Li-Ion 3,7V
EM3	Sistema de movimentação com motores e correias
EM4	Sistema de pega para objetos de até 100g
EM5	Identificação em tempo real via câmera
EM6	Sistema de conectividade *Wi-fi* local
EM7	Custo inferior a R$3000,00

Fonte: Autoria própria (2022)

3.2.4 Relacionamento e conversão de requisitos

A partir desta etapa são situadas as formas de implementação das especificações definidas previamente, relacionando requisitos com elementos técnicos de conversão e de maneira análoga obtendo-se os respectivos sistemas e subsistemas correspondentes ao modelo de desenvolvimento.

3.2.4.1 Tabela de conversão

Definida a abrangência do projeto a partir dos requisitos e metas, utiliza-se a matriz de conversão para apresentação dos elementos de sistema a serem desenvolvidos, sejam eles *software* ou *hardware*, sendo neste caso representados sob o quadro 6:

Quadro 6 – Conversão de requisitos

Req. Cliente	Elemento	Req. Produto
Plataforma aberta	*Software*	*Firmware* em linguagem de plataforma aberta
Plataforma aberta	*Hardware*	Compatibilidade com microcontrolador ARM, Arduino, ESP
Mobilidade	*Hardware*	Entradas e saídas analógicas
Mobilidade	*Hardware*	Sensor ultrassônico - Sistema anticolisão
Versatilidade de montagem	*Hardware*	Mecanismo de *pick and place*
Baixa potência	*Hardware*	Compatibilidade com microcontrolador ARM, Arduino, ESP
Baixo custo	*Hardware*	Compatibilidade com microcontrolador ARM, Arduino, ESP
Conexão remota	*Hardware*	Conectividade remota - PC/*Smartphone*
Manutenção simplificada	*Hardware*	Compatibilidade com microcontrolador ARM, Arduino, ESP

Fonte: Autoria própria (2022)

3.2.4.2 Matriz morfológica

Enfim, têm-se a sumarização em elementos físicos quanto a forma de cada sistema identificado, isso feito por meio de uma matriz morfológica, apresentada no quadro 7, que lista cada função chave com seus respectivos componentes e que permitirão a constatação e validação da aplicação ao fim do trabalho de desenvolvimento.

Quadro 7 – Matriz morfológica

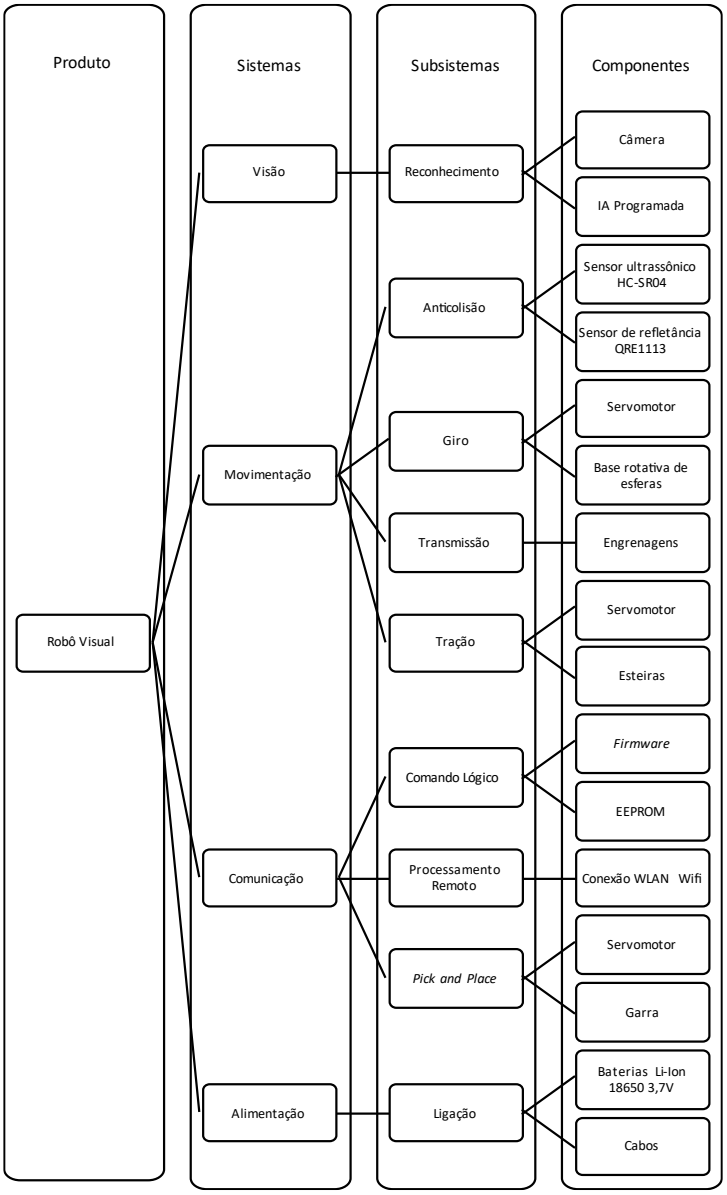
Função	Solução 1
Configuração	Cabo + EEPROM
Mobilidade	Servomotor + Rodízios + Correias
Pick and place	Braço + Garra
Reconhecimento	Câmera
Comunicação	Módulo *Wi-fi*

Fonte: Autoria própria (2022)

3.2.5 Sistemas, subsistemas e componentes

Como elemento final, têm-se o escopo de funcionalidades apresentado sob um diagrama de sistemas, subsistemas e componentes (figura 14), tais resultantes das etapas pregressas, e que permite reconhecer de maneira visual os relacionamentos gerais do produto e suas aplicações chave.

Figura 14 – Diagrama SSC



Fonte: Autoria própria (2022)

3.3 Implementação

3.3.1 Lista de materiais

Conhecidos os elementos de desenvolvimento, pode-se listar os recursos necessários e que serão utilizados para implementação, sendo inicialmente segmentados sob *software* e *hardware*.

3.3.1.1 Recursos de *software*

- Compilador *Python* (*Python* 3.10, Anaconda);
- Bibliotecas Python (OpenCV, NumPy, Pandas, *Ultralytics*, Streamlit etc.);
- *Aplicação Roboflow / Dataset base para treinamento de reconhecimento;
- Google *Collab* / Jupyter *Notebook*;
- Microsoft Visual Studio Code e/ou terminal de comando;
- IDE Arduino;
- Navegador *Web* (*Google Chrome, Microsoft Edge, Mozilla Firefox*)
- Servidor *broker* MQTT

3.3.1.2 Recursos de *hardware*

- *Webcam* e *Smartphone* com câmera;
- Estrutura de posicionamento de câmera;
- Computador portátil;
- Kit Robocore Robô Explorer
  - 06un placas de montagem em PSAI;
  - 04un tubos de Aço 6x23;
  - 02un motores elétricos;
  - 02un esteiras de borracha;
  - 01un suporte para baterias Li-Ion;
  - 02un bateria Li-Ion 18650 3,7V ;
  - 01un suporte para sensor ultrassônico;
  - 01un sensor ultrassônico HC-SR04
  - Parafusos M3 e M4 diversos tamanhos;
  - Porcas M3 e M4;
  - Espaçadores M3x10mm;
  - Jumpers para conexão.

- Kit de Expansão – Robocore *Rocket Tank*
  - 02un placas de montagem em PSAI 3mm;
  - 01un placa de montagem em acrílico 6mm;
  - 02un parafusos M3x6mm;
  - 02un parafusos M3x10mm;
  - 02un porcas M3.
- Braço Robótico Robocore RoboARM
  - 04un placas de montagem em PSAI;
  - 04un servomotores;
  - 06un esferas 6mm para sistema de rolamento;
  - 01un cabo extensor;
  - Parafusos M2, M2.5, M3 diversos;
  - Porcas M2.5 e M3;
  - *Jumpers* para conexão
- Placa Robocore Vespa

3.3.2 Sistema de detecção e reconhecimento

Em termos de informações e reconhecimentos algumas etapas são chave para efetuar a configuração do sistema de reconhecimento visando a efetiva performance deste, sendo listadas como: Preparação de dados, treinamento e validação.

3.3.2.1 Preparação dos dados para reconhecimento

O escopo de detecção e reconhecimento está definido sob duas categorias: comando e operacional, ambas tendo como base um conjunto de imagens levantadas para treinamento prévio do modelo YOLO e que constituirão os *datasets* do sistema.

Tais *datasets* serão definidos através da ferramenta de identificação *Roboflow*®, sob a qual será realizada a classificação das classes presentes em cada uma das imagens contidas nos *datasets*, este procedimento é realizado manualmente, desenhando as caixas delimitadoras de cada objeto conectada a respectiva classe, conforme pode ser visto na figura 15:

Figura 15 – Identificação das caixas delimitadoras *Roboflow*

Fonte: Autoria própria (2023)

3.3.2.2 Treinamento do modelo YOLO

O treinamento do modelo YOLO v8 é realizado por meio de código, apresentado no Apêndice A, utilizando os recursos da biblioteca *python ultralytics* a partir do carregamento do *dataset Roboflow* que contém as imagens pré-classificadas nas categorias de teste, treinamento e validação.

Desta forma, o algoritmo requer os parâmetros de tarefa, modelo base, informação da base de dados das classes, número de iterações, tamanho de imagem e configuração de plotagem. A partir destes, o sistema executa iterações comparando as imagens submetidas a treinamento com as imagens base do modelo efetuando a seguir a validação de acurácia.

3.3.2.3 Validação do sistema de reconhecimento YOLO

Para avaliação do sistema de reconhecimento o próprio sistema de detecção YOLO gera indicadores resultantes do seu treinamento, apresentando métricas como matriz de confusão, taxa de precisão média, velocidade de inferência entre outros que são obtidas pela execução do modelo de detecção sob uma base de imagens (*dataset*) e comparação entre as imagens de saída com as originais pré-treinadas.

3.3.3 Arquitetura física

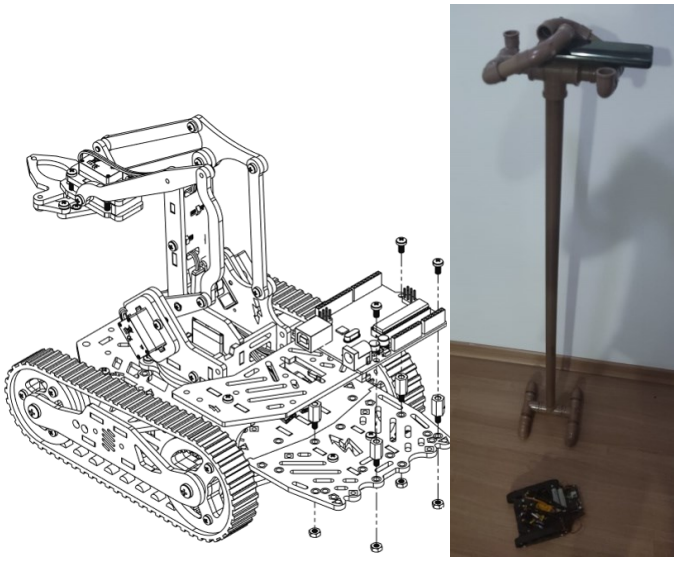
A estrutura física do protótipo é subdividida em três subsistemas:

1. Estrutura física;
2. Sistema elétrico e de sinais;
3. Sinais e informações

3.3.3.1 Estrutura mecânica

A estrutura mecânica remete ao conjunto a ser comandado remotamente, e consiste no robô a ser controlado e o conjunto de sustentação da câmera de monitoramento, que são basicamente representados na figura 16. Tal arranjo simplificado tem o intuito de permitir o amplo monitoramento das ações do robô em função dos comandos submetidos.

Figura 16 – Estrutura mecânica do robô e câmera

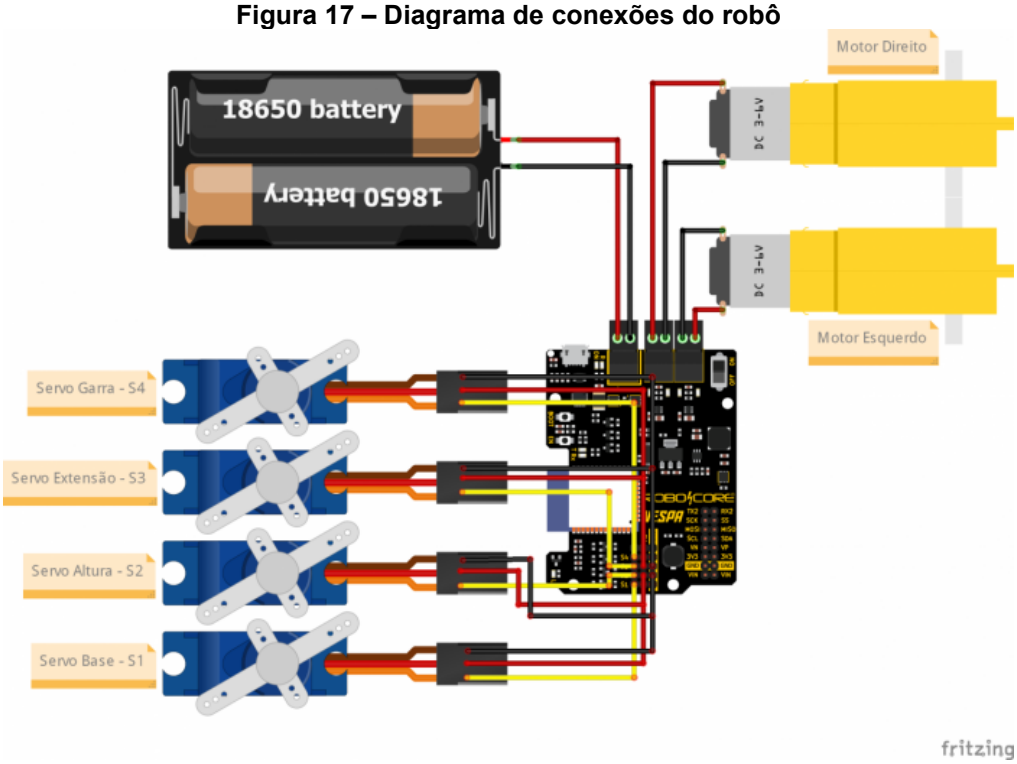


Fonte: Autoria própria (2023)

3.3.3.2 Sistema elétrico e de sinais

O sistema de controle do robô é configurado a partir da placa Vespa que contém um controlador ESP32 integrado já com conectores para controle de motores DC, baterias e sensores, sendo representados com suas devidas ligações na figura 17:

Figura 17 – Diagrama de conexões do robô



Fonte: Robocore © (2023)

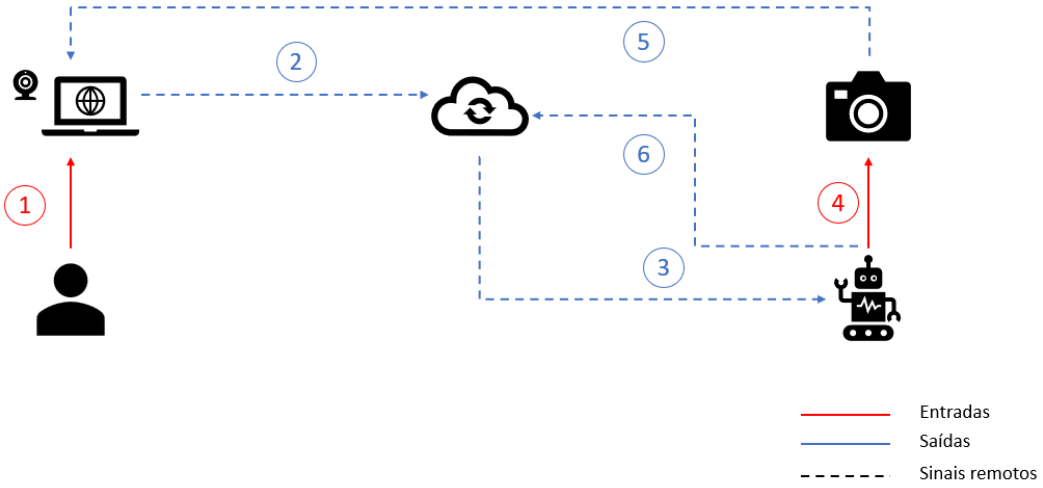
3.3.3.3Sinais e informações

Os sistemas de comunicação são descritos na figura 18, e demonstram as trocas de sinais entre os agentes inseridos no escopo de trabalho. A sequência lógica representada é a seguinte:

1. Sinal de entrada visual, a câmera 1 (*webcam* integrada ao computador) reconhece o padrão de comandos manuais do usuário, configurados conforme a matriz do Quadro 8;
2. O comando identificado é processado pelo algoritmo e enviado via sinal para o *broker* MQTT;
3. A mensagem recebida é monitorada e compreendida pelo robô inscrito ao *broker* MQTT e que dispara a realização do respectivo comando;
4. A câmera 2 (*IPCam*) acompanha e monitora a execução do robô para validação da ação correta;
5. A imagem da *IPcam* é enviada para a interface *web* e visualização do usuário;
6. O robô envia dados da sua bateria ao *broker*;

.

Figura 18 – Diagrama de sinais



Fonte: Autoria própria (2023)

Quadro 8 – Matriz de comandos manuais

Mão Direita	Mão Esquerda	Ação
		Movimento para a frente do robô
		Movimento para trás do robô
		Rotação sentido anti-horário
		Rotação sentido horário
		Parada comandada
		Aumento do nível de potência
		Redução do nível de potência
		Movimento para frente do braço robótico (1° Eixo)
		Movimento para trás do braço robótico (1° Eixo)
		Movimento para baixo do braço robótico (2° Eixo)
		Movimento para cima do braço robótico (2° Eixo)
		Abertura da garra
		Fechamento da garra
		Rotação do braço robótico sentido horário
		Rotação do braço robótico sentido anti-horário

Fonte: Autoria própria (2023)

3.3.4 Interface do usuário

A interface *web* contém logo abaixo da logo e do cabeçalho os campos de configuração do servidor MQTT e da *IPCam*. Nesta são solicitadas as seguintes informações:

1. Endereço URL do broker MQTT;
2. Porta de conexão do *broker*;
3. Nome de usuário do *broker*;
4. Senha de usuário do *broker*;
5. Endereço da *IPCam*

Além destes, há duas interfaces que mostram as visualizações da *webcam* e da *IPcam*.

Figura 19 – Página *Web*

Fonte: Autoria própria (2023)

4 RESULTADOS E CONCLUSÕES

Esta seção abrange a validação do trabalho e apresenta os resultados obtidos ao fim do desenvolvimento, relatando a experiência geral. Além disso, são apresentadas propostas de continuidade do trabalho desenvolvido para continuidade da pesquisa e fomento acadêmico do tema.

4.1 Validação do sistema de detecção

Para avaliação do modelo de detecção a própria função de treinamento YOLO provê algumas métricas de performance tais como:

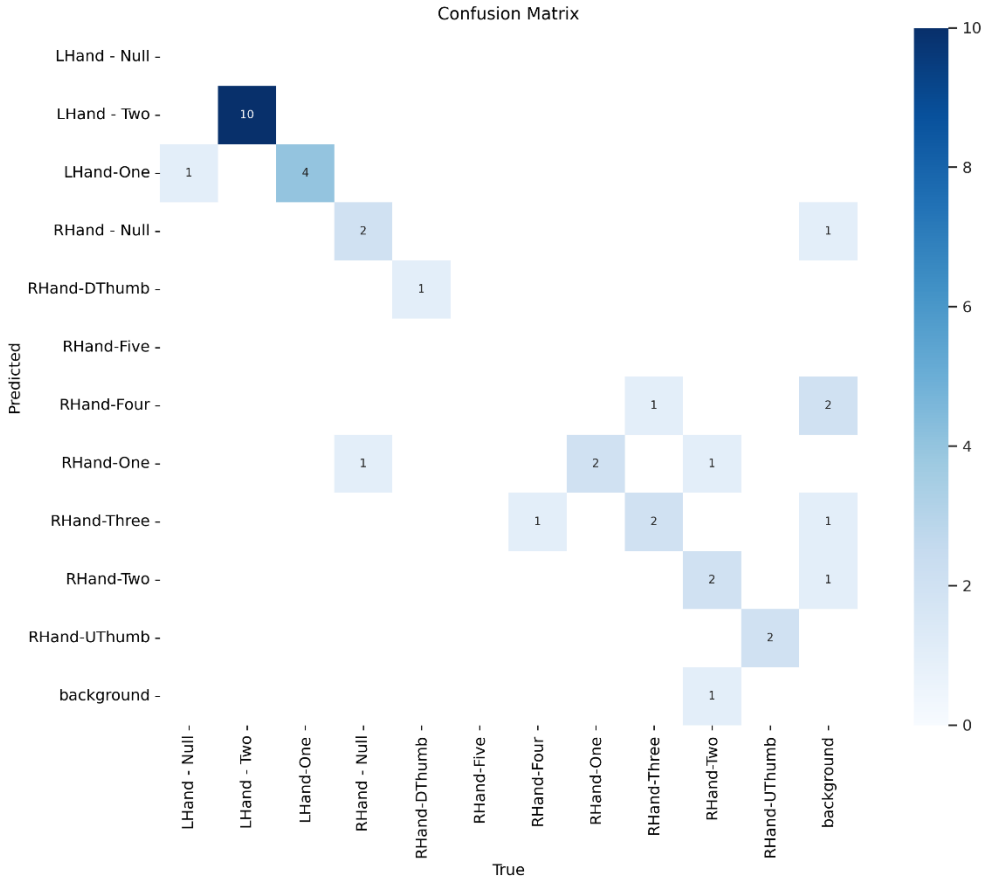
- Matriz de confusão: Representa visualmente a performance interna do modelo dentre as classes em que este é treinado, levando em conta grau de confiança e sobreposição de forma para cada predição efetuada.
- Gráficos de perdas:
  - *Box Loss*: Representa quão assertivo é o modelo na identificação do centro e das coordenadas da *bounding box* frente ao objeto exposto, e é calculada a partir da regressão do erro da predição versus a imagem referência, de forma que quanto menor, melhor é a performance. É representada em termos de erro versus o número de execuções treinadas.
  - *Seg Loss*: De maneira parecida ao *box loss* este índice é aplicado a segmentação de objetos na imagem, avaliando a máscara de contorno que identifica e destaca elementos classificados, e também é levantado a partir do erro e, portanto, quanto menor tem-se melhor resultado. É representada em termos de erro versus o número de execuções treinadas.
  - *Cls Loss*: Este índice mede o erro dentre a predição e a base de referência, sendo que quanto menor, mais assertivo é o modelo. É representado em termos de erro versus o número de execuções treinadas.
  - DFL *Loss*: Este índice avalia a perda da distribuição focal, que é um problema de desbalanceamento no treinamento de redes neurais que ocorre quando uma classe tem maior frequência de detecção do que outras, assim aplica-se um fator de compensação maior para classes

menos frequentes. É representado em termos de erro versus o número de execuções treinadas.

- Gráficos de performance:
  - Precisão: Em termos práticos, é a capacidade do modelo acertar positivamente, ou seja, interpretar corretamente com o menor fator de falsos positivos, e é medida de forma percentual versus o número de execuções treinadas.
  - *Recall*: Bastante similar ao conceito de precisão, contudo é o indicador para falsos negativos, ou seja, é a medida de quanto o modelo confunde elementos com o que não o são, medido de forma percentual versus o número de execuções treinadas.
  - mAP: É uma medida global da performance do sistema que leva em conta os índices de precisão, *recall*, intersecção sobre união (IoU) e precisão média de cada classe, geralmente é apresentado como mAP50 ou mAP50-95, que indicam sob um índice de intersecção sobre união, ou seja, considerando a sobreposição de classes, num fator específico de 0,5 e de um campo de 0,5 a 0,95 com um passo de 0,05 respectivamente. É representado em forma percentual versus o número de execuções treindas.

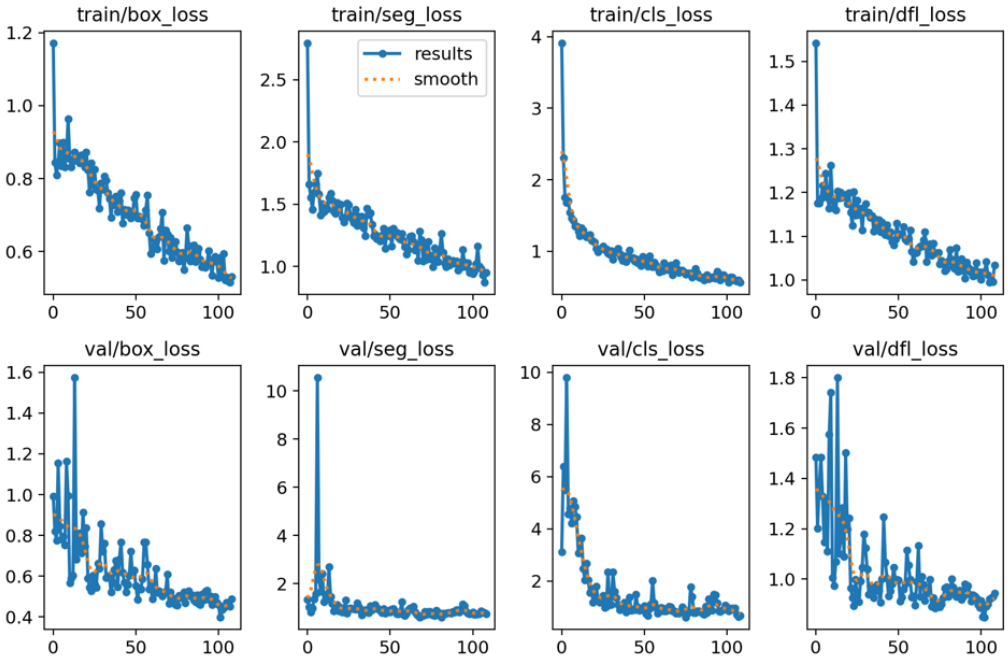
Tais métricas foram obtidas, a partir do código de treinamento do Apêndice A, para ambas as condições de detecção, comandos manuais e detecção do robô, com os resultados representados entre as figuras 20 e 25;

Figura 20 - Matriz de confusão obtida para comandos de mão



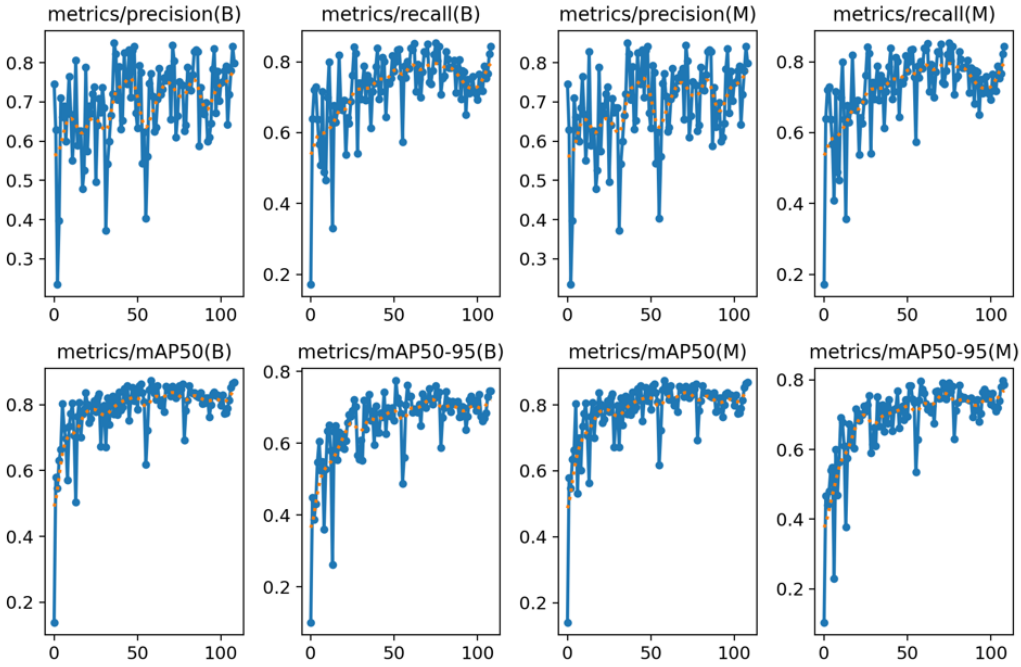
Fonte: Autoria própria (2023)

Figura 21 - Gráficos de perdas do modelo de comandos de mão



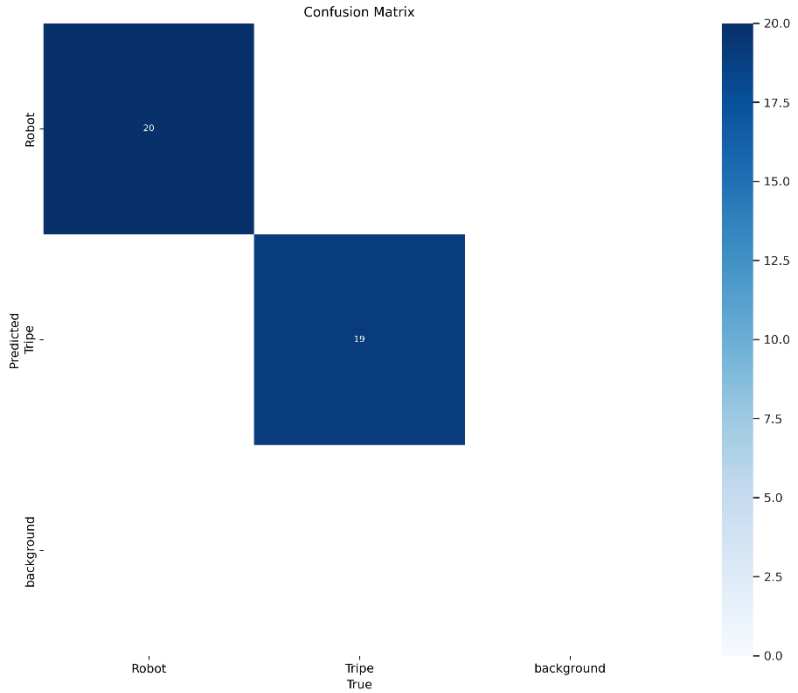
Fonte: Autoria própria (2023)

Figura 22 - Gráficos de métricas do modelo de comandos de mão



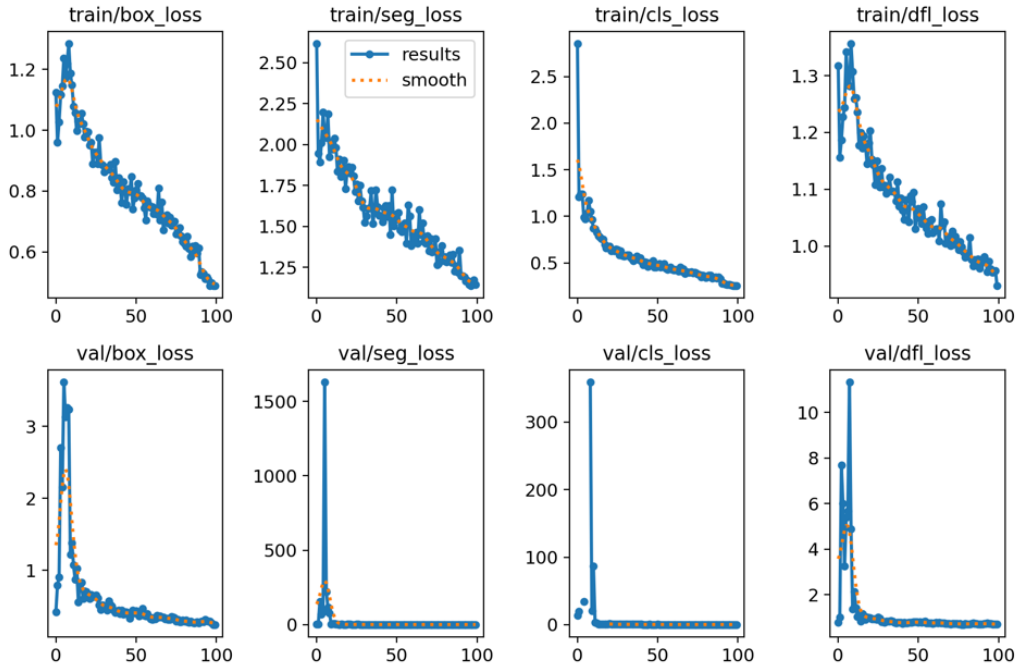
Fonte: Autoria própria (2023)

Figura 23 - Matriz de confusão obtida para detecção do robô



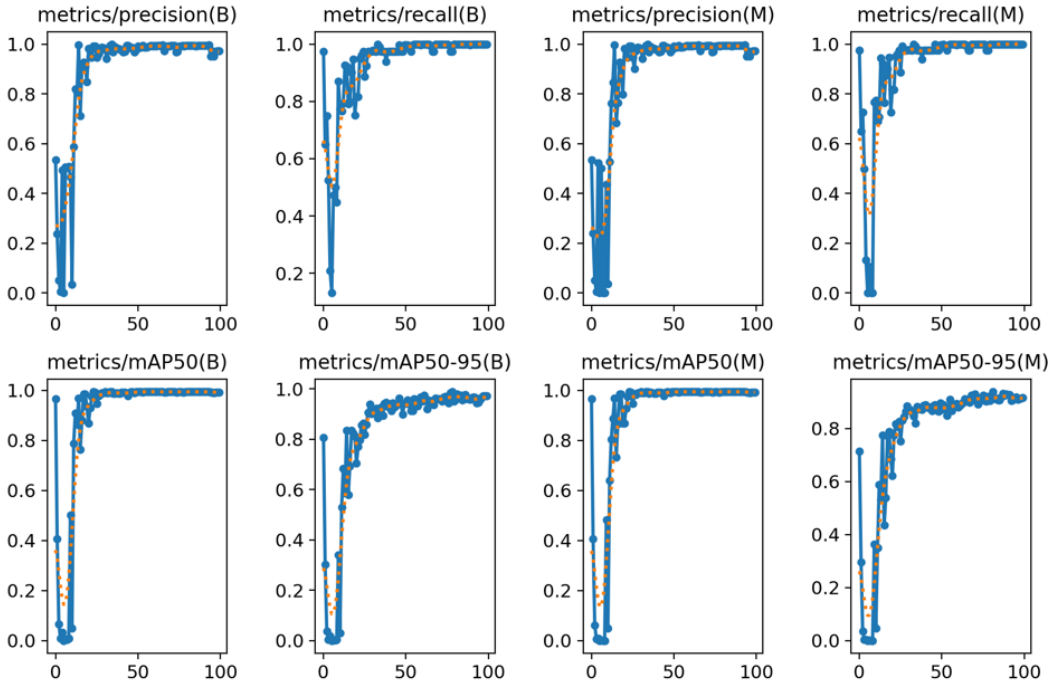
Fonte: Autoria própria (2023)

Figura 24 - Gráficos de perdas do modelo de detecção do robô



Fonte: Autoria própria (2023)

Figura 25 - Gráficos de métricas do modelo de detecção do robô



Fonte: Autoria própria (2023)

Os resultados obtidos evidenciam que o modelo de detecção tem um desempenho bastante apurado e estável, principalmente pelo nível mAP obtido muito próximo a 1, isto devido a base reduzida de categorias a serem identificadas que reforça o processo de treinamento com a redução de potenciais distúrbios. Já o modelo de detecção de comandos manuais apresenta indicadores com maior

oscilação, visto o número de classes relacionadas e o potencial de confusão entre estas, dada a similaridade que possuem, notando-se que este modelo tem um ajuste mais sensível do que o modelo robótico, porém que também se mostra estável entre um índice geral com um mAP na faixa de 0,7, o que representaria uma acuracidade média de aproximadamente 70%.

Assim, considera-se ambos os modelos válidos para aplicarmos ao protótipo em termos de execução, e que permita em desenvolvimentos futuros oportunidades de refinamento da performance de identificação.

4.2 Validação funcional geral

Em termos operacionais o protótipo se mostrou eficaz de acordo com a proposta de comando por reconhecimento visual, descrita em 3.3.3.3 – Sinais e Informações, executando as respectivas ações programadas a partir das informações recebidas via MQTT referentes aos resultados do processamento do modelo. No entanto, a performance foi bastante limitada exigindo restrição de parâmetros para atender aos requisitos funcionais mínimos, tais como limitação da taxa de quadros de comandos a um frame por segundo devido a taxa de processamento do modelo YOLO para a resposta à imagem recebida, ou parametrização de tempo para cada função programada de execução, sendo estes desafios advindos decorrentes da integração entre as plataformas. Apesar disso, o protótipo se mostra um bom modelo para a aprendizagem robótica e incentivador do desenvolvimento da robótica colaborativa, como um primeiro contato técnico/tecnológico acessível a novos desenvolvedores.

Estudos como este têm relevância quanto ao amadurecimento e preparação fundamentada de um ecossistema de tecnologias baseado nos conceitos de indústria 4.0, propiciando meios de disseminação e popularização dos conceitos industriais e expandindo tais tecnologias de forma disruptiva. Além disso, é importante para a preparação dos profissionais que tomarão postos terem acesso a conceitos e aplicações que permitam seu desenvolvimento e compreensão dos desafios que estão por vir.

4.3 Recomendações para trabalhos futuros

Para trabalhos futuros, as recomendações seguem no sentido de consolidar o aprendizado e direcionar o desenvolvimento de implementações voltadas a robótica colaborativa. Assim a partir dos resultados obtidos fatores direcionadores seriam:

a) Melhoria na acuracidade e performance de modelos YOLO ou redes neurais;
b) Soluções de identificação de orientação e rastreamento de objetos;
c) Desenvolvimento de um aplicativo remoto, ou plataforma de controle para o protótipo;
d) Implementação interativa entre robô/robô ou robô/humano;
e) Estudo de caso com aplicações voltadas a indústria.

REFERÊNCIAS

AKELLA, Prasad et al. *Cobots for the automobile assembly line.* ***Proceedings 1999 IEEE International Conference on Robotics and Automation***, Detroit, MI, USA, 1999, pp. 728-733 vol.1. Disponível em: https://ieeexplore.ieee.org/document/770061. Acesso em: 04 mai. 2019

ALMEIDA, Julio Sergio Gomes de; CAGNIN, Rafael Fagundes. **A INDÚSTRIA DO FUTURO NO BRASIL E NO MUNDO**. Disponível em: https://iedi.org.br/media/site/artigos/20190311_industria_do_futuro_no_brasil_e_no_mundo.pdf. Acesso em: 04 mai. 2019.

ALVES, Maria Bernardete Martins; ARRUDA, Susana Margareth. **Como fazer referências** (Bibliográficas, Eletrônicas e Demais Formas de Documentos). Florianópolis: UFSC, 2002. Disponível em: http://www. Bu.ufsc.br . Acesso em: 17 nov. 2002.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 6023:** informação e documentação: referências: elaboração. Rio de Janeiro: ABNT, 2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 6024:** Informação e documentação - Numeração progressiva das seções de um documento escrito - Apresentação. Rio de Janeiro: ABNT, 2012.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 6027:** Informação e documentação — Sumário — Apresentação. Rio de Janeiro: ABNT, 2012.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 6028:** Informação e documentação — Resumo, resenha e recensão — Apresentação. Rio de Janeiro: ABNT, 2021.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 6034:** Informação e documentação – Índice – Apresentação. Rio de Janeiro: ABNT, 2004

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 10520:** Informação e documentação: citações em documentos - apresentação. Rio de Janeiro: ABNT, 2002.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 14724:** Informação e documentação — Trabalhos acadêmicos — Apresentação. Rio de Janeiro: ABNT, 2011.

BOCK, Ana Mercês Bahia; FURTADO, Odair; TEIXEIRA, Maria de Lourdes Trassi. **Psicologias: Uma introdução ao estudo de psicologia.** 13. ed. São Paulo: Editora Saraiva, 1999.

CICHORSKI, G. **Estudo dos requisitos necessários de segurança em uma célula robotizada de soldagem de acordo com a norma NR-12**. 2015.

DEMIR, Kadir Alpaslan; DÖVEN, Gözde; SEZEN, Bülent. *Industry 5.0 and Human-Robot Co-working.* ***Procedia Computer Science***, v. 158, p. 688-695, jun. 2019.

Disponível em: https://doi.org/10.1016/j.procs.2019.09.104. Acesso em: 01 out. 2020.

DENG, Li; YU, Dong. ***Foundations and Trends in Signal Processing***, v.7, n.3–4, p. 197-387. jun. 2014. Disponível em http://dx.doi.org/10.1561/2000000039. Acesso em: 25 out. 2020.

FREY, Carl Benedikt; OSOBORNE, Michael A. *TECHNOLOGY AT WORK v2.0.* ***Citi GPS: Global Perspectives & Solutions***. 2016. Disponível em: https://www.citivelocity.com/citigps/technology-work-v2-0/. Acesso em: 04 mai. 2019.

GAMA, Priscila dso Santos, JACUBAVICIUS, Celso, FORMIGONI, Alexandre. PROPOSTA DE CONTROLE DE ESCOPO POR MEIO DA ESTRUTURA ANALÍTICA DO PROJETO (EAP): ESTUDO DE CASO. **SADSJ – *South American Development Society Journal***. v.1, n.1, p. 109-123. mar. 2017. Disponível em: http://www.sadsj.org/index.php/revista/article/view/8. Acesso em: 12 nov. 2023.

GERSHENSON, Carlos. ***Artificial Neural Networks for Beginners***. 20 ago. 2003. Disponível em: https://arxiv.org/pdf/cs/030803. Acesso em: 09 set. 2021.

GOODRICH, Michael A.; SCHULTZ, Alan C. ***Foundations and Trends® in Human–Computer Interaction***. *Human–Robot Interaction: A Survey*. v.1, n.3, p. 203-275. jan. 2008. Disponível em: http://dx.doi.org/10.1561/1100000005. Acesso em: 03 out. 2020.

HAFIANE, Saad, SALIH, Yasir, MALIK, Aamir S. *3D Hand Recognition for Telerobotics.* ***2013 IEEE Symposium on Computers & Informatics***. Langkawi, Malasia. 2013. p. 132-137. Disponível em: https://doi.org/10.1109/ISCI.2013.6612390. Acesso em: 24 set. 2023.

ICW. ***About ICW***. *What is Collaborative Working?*. Disponível em: https://instituteforcollaborativeworking.com/About-ICW/What-is-Collaborative-Working. Acesso em: 04 out. 2020.

IFR, *EXECUTIVE SUMMARY WORLD ROBOTICS 2018 INDUSTRIAL ROBOTS*. [Online]. Disponível em: https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf. Acesso em: 03 out. 2020.

KIM, Young Gyu, KANG, Jeong Seok, PARK, Hong Seong. *Publish/Subscribe model based Communication for Telerobotics*. ***2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)***. Jeju, Coréia do Sul. 2013. p. 305-308. Disponível em: https://doi.org/10.1109/URAI.2013.6677372. Acesso em: 24 set. 2023.

KURT, Resul. *Industry 4.0 in Terms of Industrial Relations and Its Impacts on Labour Life.* ***Procedia Computer Science***, v. 158, p. 590-601, jun. 2019. Disponível em: https://doi.org/10.1016/j.procs.2019.09.093. Acesso em: 01 out. 2020.

LYNCH, Shana. Andrew Ng: *Why AI Is the New Electricity.* ***Insights by Stanford Business***. 11 mar. 2017. Disponível em:

https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity. Acesso em: 04 mai. 2019.

MISHRA, Chandrahas; GUPTA, Dharmendra Lal. *Deep Machine Learning and Neural Networks: An Overview.* ***International Journal of Hybrid Information Technology***, v. 9, n. 11, p. 401-414, set. 2016. Disponível em: http://dx.doi.org/10.14257/ijhit.2016.9.11.34. Acesso em: 23 out. 2020.

MOU, Yi; XU, Kun. *The media inequality: Comparing the initial human-human and human-AI social interactions.* ***Computers in Human Behavior***, v.72, p. 432-440, jul. 2017. Disponível em: https://doi.org/10.1016/j.chb.2017.02.067. Acesso em: 03 out. 2020.

OSTERGAARD, Esben. *Welcome to Industry 5.0: The "human touch" revolution is now underway.* ***Quality Magazine : Management***, p. 36-39, 08 mai. 2019. Disponível em: https://www.qualitymag.com/articles/95450-welcome-to-industry-50. Acesso em: 30 set. 2020.

PARIKH, Dhairya. ***Raspberry Pi and MQTT Essentials****: A complete guide to help you build innovative full-scale prototype projects using Raspberry Pi and MQTT protocol*. 1 ed. Birmingham: Packt, 2022.

PULLI, Kari et al. *Real-Time Computer Vision with OpenCV.* ***Communications of the ACM***, v.55, p. 61-69, jun. 2012. Disponível em: https://dl.acm.org/doi/10.1145/2184319.2184337. Acesso em: 22 out. 2020.

REDMOND, Joseph et al. *You Only Look Once: Unified, Real-Time Object Detection.* ***Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*** **(CVPR)**, p. 779-788, mai. 2016. Disponível em: https://doi.org/10.48550/arXiv.1506.02640. Acesso em: 16 mai. 2023.

RICHERT, Anja et al. *Educating Engineers for Industry 4.0: Virtual Worlds and Human-Robot-Teams Empirical Studies towards a new educational age*. **2016 IEEE *Global Engineering Education Conference (EDUCON)*,** Abu Dhabi, 2016, p. 142-149. Disponível em: https://ieeexplore.ieee.org/document/7474545. Acesso em: 04 mai. 2019.

SALVADOR, Jocelei André. **FERRAMENTA DE GESTÃO DE RISCOS E PROBLEMAS DE PROJETO NO PROCESSO DE DESENVOLVIMENTO DE PRODUTOS**. 2015. Trabalho de conclusão da atividade de estágio II – Curso de graduação em Engenharia Mecânica, Universidade de Caxias do Sul, Caxias do Sul, 2015.

SANTOS, Winderson Eugenio dos, GORGULHO, José Hamilton Chaves. **Robótica Industrial: Fundamentos, tecnologias, programação e simulação**. 1. ed. São Paulo: Editora Erica, 2014.

SIGUT, Jose et al. OpenCV *Basics: A Mobile Application to Support the Teaching of Computer Vision Concepts*. **IEEE *Transactions on Education***, v.63, p. 328-335, mai 2020. Disponível em: https://ieeexplore.ieee.org/document/9103956. Acesso em: 23 out. 2020.

SIMON, Herbert A. *Artificial intelligence: An empirical science.* ***Artificial Intelligence***, v. 77, n. 1, p. 95-127, ago. 1995. Disponível em: https://doi.org/10.1016/0004-3702(95)00039-H. Acesso em: 04 mai. 2019.

SCHMIDT, R; MOHRING, M; HARTING, R; REICHSTEIN, C; NEUMAIER, P; JOZINOVIÉ, P. *Industry 4.0 – Potentials for Creating Smart Product: Empirical Research Results*. **Springer *International Publishing Switzerland*.** p. 16–27. 2015

SHEHADEH, Mohammad et al. *Hybrid Teams of Industry 4.0: A Work Place considering Robots as Key Players*. **2017 IEEE *International Conference on Systems, Man, and Cybernetics* (SMC)**, Banff, p. 1208-1213, out. 2017. Disponível em: https://ieeexplore.ieee.org/document/8122777. Acesso em: 04 mai. 2019.

SHERIDAN, Thomas B. *Human-Robot Interaction: Status and Challenges.* ***Human Factors the Journal of the Human Factors and Ergonomics Society***, v.58, n. 4, p. 525-532, jun. 2016. Disponível em: https://doi.org/10.1177/0018720816644364. Acesso em: 03 out. 2020.

WANG, Yingxu. *Cognitive Computing and machinable thought*. ***IEEE International Conference on Cognitive Informatics***, p. 6-8, jun. 2009. Disponível em: https://doi.org/10.1109/COGINF.2009.5250709. Acesso em: 24 set. 2019.

WOODEN, John R. ***Wooden on Leadership:*** *How to Create a Winning Organization*. Nova Jersey: McGraw-Hill, 2005.

YASSEIN, Muneer Bani, SHATNAWI, Mohammed Q., ALJWARNEH, Shadi AL-HATMI, Razan. *Internet of Things: Survey and open issues of MQTT Protocol*. ***2017 International Conference on Engineering & MIS (ICEMIS)***, Monastir, Tunisia, 2017, pp. 1-6, Disponível em: https://doi.org/10.1109/ICEMIS.2017.8273112. Acesso em: 24 set. 2023

ZHONG, Ray Y.; XU, Xun; KLOTZ, Eberhard; NEWMAN, Stephen T. *Intelligent Manufacturing in the Context of Industry 4.0: A Review. Engineering* v. 3, n. 5, Outubro 2017, p. 616-630. Disponível em: https://doi.org/10.1016/J.ENG.2017.05.015. Acesso em: 04 mai. 2019.

ZHOU, Keliang; LIU, Taigang; ZHOU, Lifeng. *Industry 4.0: Towards future industrial opportunities and challenges.* ***2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (fskd)***, Zhangjiajie, p. 2147-2152, ago. 2015. Disponível em: http://ieeexplore.ieee.org/document/7382284. Acesso em: 04 mai. 2019.

APÊNDICE A - Código de preparação dos dados (treinamento)

```
# PREPARE RESOURCES, LIBRARIES AND MODULES
from google.colab import drive
drive.mount('/content/drive')

!pip install -r '/content/drive/MyDrive/TCC/Program/COBOT Command/requirements.txt'

!nvidia-smi

import os
HOME = os.getcwd()
print(HOME)

#os.chdir('/content/drive/MyDrive/TCC/Program/Piece Block Detection')

#PATH = os.getcwd()
#print(PATH)

from roboflow import Roboflow
rf = Roboflow(api_key="YOUR KEY")
project = rf.workspace("YOUR WORKSPACE").project("YOUR PROJECT")
dataset = project.version(2).download("yolov8")

from IPython import display
display.clear_output()
import ultralytics
ultralytics.checks()

from ultralytics import YOLO
from IPython.display import display, Image

import cv2
import numpy as np
import yaml
from yaml.loader import SafeLoader
import mediapipe as mp

import streamlit as st
from streamlit_webrtc import webrtc_streamer
from PIL import Image as Img

# TRAINING
print(dataset.location)

!yolo mode=train task=segment model=yolov8l-seg.pt data={dataset.location}/data.yaml epochs=500 imgsz=640 plots=True

print({HOME})

#from IPython.display import display, Image
```

```
Image(filename=f'{HOME}/runs/segment/train/confusion_matrix.png', width=1280)

Image(filename=f'{HOME}/runs/segment/train/results.png', width=1280)

Image(filename=f'{HOME}/runs/segment/train/val_batch0_pred.jpg', width=600)

!yolo task=segment mode=predict model={HOME}/runs/segment/train/weights/best.pt conf=0.25 source={dataset.location}/test/images save=True

!yolo task=segment mode=val model={HOME}/runs/segment/train/weights/best.pt data={dataset.location}/data.yaml

!yolo task=segment mode=predict model={HOME}/runs/segment/train/weights/best.pt conf=0.25 source={dataset.location}/test/images save=True

!yolo export model={HOME}/runs/segment/train/weights/best.pt format=onnx opset=12

#from google.colab import files
#files.download('filepath/filename')

# Copying folders, format: !rsync -r --progress source_path destination_path
#!rsync -r --progress "./model" "/content/drive/My Drive/Colab Notebooks/my-project/model"
```

APÊNDICE B - Código *Python* – Biblioteca YOLO

```
from ultralytics import YOLO
from ultralytics.yolo.v8.detect.predict import DetectionPredictor
from PIL import Image
from paho import mqtt
from mdMQTT import my_mqtt


import tensorflow as tf
import numpy as np
import cv2
import streamlit as st
import streamlit_webrtc as webrtc
#import paho.mqtt.client as mqtt
import time
import paho.mqtt.client as paho
#from paho.mqtt.client import MQTTMessage
import json

#############################################################################
#############################################################

model_height = 640
model_width = 640
my_topic = "101"

class my_pred():

    def __init__(self,mymodel_address):
       self.model = YOLO(mymodel_address)
       self.broker_address = ""
       self.broker_port = ""
       self.broker_usr = ""
       self.broker_pwd = ""
       self.conf_lvl = 0.75

    def set_mqtt(self,broker_address, broker_port, broker_usr, broker_pwd):
        self.MQTT =
my_mqtt(usrname=broker_usr,pwd=broker_pwd,broker_address=broker_address,port=b
roker_port)
        #self.MQTT.connect()
        #self.MQTT.publish(topic="encyclopedia/",payload="Connected")
        #self.MQTT.loop_start()


    def video_to_yolo(self,frame: np.ndarray) -> tuple[np.ndarray, list[str],
list[int], list[float]]:
    # create a yolo model
        model = self.model
```

```
        # resize the frame to the model input size
        frame = cv2.resize(frame, (model_width, model_height))
        frame = cv2.flip(frame,1)

        cls_list = []
        names_list = []
        conf_list = []

        # predict the bounding boxes and class labels using the model
        results = model(source=frame, stream=True, conf=0.75, imgsz=(640,
640))
        for r in results:
            boxes = r.boxes
            coords = boxes.xywh
            vertix = boxes.xyxy
            labels = r.names
            frame = r.plot()
            # normalize the frame pixels to [0, 1]
            frame = frame / 255.0
            #frame = frame.astype(np.uint8)

            detection_count = boxes.shape[0]

            for i in range(detection_count):
                cls = int(boxes.cls[i].item())
                name = labels[cls]
                #self.MQTT.loop_start()
                if (cls < 3):# and model_address == "./02 - Training/Hand
Commands/runs/segment/train/weights/best.onnx"):
                    self.MQTT.publish(topic="HandCommands/LH", payload= name)
                elif (cls >= 3 and cls < 11):# and model_address == "./02 -
Training/Hand Commands/runs/segment/train/weights/best.onnx"):
                    self.MQTT.publish(topic="HandCommands/RH", payload= name)
                else:
                    pass
                #self.MQTT.loop_stop()
                confidence = float(boxes.conf[i].item())

                cls_list.append(cls)
                names_list.append(name)
                conf_list.append(confidence)

                names_list_json = json.dumps(names_list)

                print(cls_list)
                print(len(names_list))
                print(len(names_list_json))
                print(conf_list)
```

```
            # return the processed frame
            return frame, names_list, cls_list, conf_list
        return frame, names_list, cls_list, conf_list

################################################################################
#############################################################

    def video_to_streamlit(self, frame: np.ndarray) -> tuple[np.ndarray,
list[str], list[int], list[float]]:
    # create a yolo model
        #model = YOLO(mymodel)
        model = self.model

        # resize the frame to the model input size
        frame = cv2.resize(frame, (model_width, model_height))
        frame = cv2.flip(frame,1)

        cls_list = []
        names_list = []
        conf_list = []

        # predict the bounding boxes and class labels using the model
        results = model(source=frame, stream=True, conf=0.75, imgsz=(640,
640))
        for r in results:
            boxes = r.boxes
            coords = boxes.xywh
            vertix = boxes.xyxy
            labels = r.names
            frame = r.plot()
            # normalize the frame pixels to [0, 1]
            frame = frame / 255.0
            #frame = frame.astype(np.uint8)

            detection_count = boxes.shape[0]

            for i in range(detection_count):
                cls = int(boxes.cls[i].item())
                name = labels[cls]
                xywh = coords[cls]
                #self.MQTT.loop_start()
                self.MQTT.publish(topic="VIBot/Coordinates", payload= xywh)
                #self.MQTT.loop_stop()
                confidence = float(boxes.conf[i].item())

                cls_list.append(cls)
                names_list.append(name)
                conf_list.append(confidence)
```

```
                names_list_json = json.dumps(names_list)

                print(cls_list)
                print(len(names_list))
                print(len(names_list_json))
                print(conf_list)

            # return the processed frame
            return frame, names_list, cls_list, conf_list
        return frame, names_list, cls_list, conf_list

################################################################################
#############################################################
```

APÊNDICE C - Código *Python* – Biblioteca MQTT

```python
from paho import mqtt
#from paho.mqtt.client import MQTTMessage

#import paho.mqtt.client as mqtt
import paho.mqtt.client as paho
import time
import json

class my_mqtt():
    def __init__(self, usrname, pwd, broker_address, port):
        self.client = paho.Client(client_id=usrname, protocol=paho.MQTTv5)
        self.client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
        self.userdata = ""
        self.flags = ""
        self.broker_address = broker_address
        self.port = port
        self.client.username_pw_set(username=usrname, password=pwd)
        self.flag = 0
        self.message_topic = ""
        self.message_payload = ""


    def on_connect(self, userdata, flags, usrname, pwd,rc):
        if self.flag == 0: #28.10.2023
            print("Connected with result code "+str(rc))
        else:
            self.flag = 1 #28.10.2023

    def on_message(self, msg):
        # print(msg.topic+" "+str(msg.payload))
        self.message_topic = msg.topic
        self.message_payload = msg.payload
        m_decode = str(msg.payload.decode("utf-8", "ignore"))
        print("message received:", m_decode)


    def on_subscribe(self, userdata, mid, granted_qos, properties=None):
        print("Subscribed: " + str(mid) + " " + str(granted_qos))

    def on_publish(self, userdata, mid, properties=None):
        print("mid: " + str(mid))

    def connect(self):
        self.client.on_connect = self.on_connect
        self.client.on_message = self.on_message
        self.client.on_publish = self.on_publish
        self.client.on_subscribe = self.on_subscribe
        self.client.connect(self.broker_address,self.port)
```

```
    def subscribe(self, topic):
        self.client.subscribe(topic)

    def publish(self, topic, payload):
        self.client.publish(topic, payload,retain=False,qos=1)

    def loop_forever(self):
        self.client.loop_forever()

    def loop_start(self):
        self.client.loop_start()

    def loop_stop(self):
        self.client.loop_stop()
```

APÊNDICE D - Código *Python*/*Web*

```
# Libraries import
import streamlit as st
import numpy as np
import threading
import av
import cv2
import time

from test_pred_v9 import my_pred
from mdMQTT import my_mqtt

from PIL import Image as img
from ultralytics import YOLO
from ultralytics.yolo.v8.detect.predict import DetectionPredictor
from streamlit_webrtc import webrtc_streamer, VideoProcessorBase

################################################################################
############################################################

# Classes definition
class MyLocalVideoProcessor(VideoProcessorBase):
    def __init__(self, video_source):
        self.model_lock = threading.Lock()
        self.my_pred = h_yolo#my_pred(mymodel_address="path/to/your/model")
        self.cap = cv2.VideoCapture(video_source,cv2.CAP_DSHOW)
        self.names_list = []
        self.cls_numbers = []
        self.conf = []


    def recv(self, frame: av.VideoFrame) -> av.VideoFrame:
        # Read a frame from the video source
        #ret, frame = self.cap.read()
        #frame = frame.to_image()
        frame = frame.to_ndarray(format="bgr24")
        frame = cv2.resize(frame, (640, 640))
        #frame = cv2.flip(frame,1)

        if frame is not None:
            # if not ret:
            #     # Failed to read a frame from the video source
            #     return None
            with self.model_lock:
            # Call your video_to_yolo method here
                processed_frame, self.names_list, self.cls_numbers, self.conf
= self.my_pred.video_to_yolo(frame)
                #myMQTT.publish("101",detections)
                processed_frame = processed_frame * 255
                processed_frame = processed_frame.astype(np.uint8)
```

```
                # self.names_list.append(detections)
                # self.cls_numbers.append(class_number)
                # self.conf.append(conf_list)
                # Wait 1 sec
                #time.sleep(1)
                # Return the processed video frame
                return av.VideoFrame.from_ndarray(processed_frame,
format="bgr24")

class MyIPVideoProcessor(VideoProcessorBase):
    def __init__(self, video_source):
        self.model_lock = threading.Lock()
        self.my_pred = r_yolo#my_pred(mymodel_address="path/to/your/model")
        self.cap = cv2.VideoCapture(video_source)


    def recv(self, frame: av.VideoFrame) -> av.VideoFrame:
        # Read a frame from the video source
        ret, frame = self.cap.read()
        #frame = frame.to_image()

        if not ret:
            # Failed to read a frame from the IP camera stream
            return None

        # Read a frame from the video source
        #frame = frame.to_ndarray(format="bgr24")
        frame = cv2.resize(frame, (640, 640))
        frame = cv2.flip(frame,1)

        if frame is not None:
            # if not ret:
            #     # Failed to read a frame from the video source
            #     return None
            with self.model_lock:
            # Call your video_to_yolo method here
                processed_frame = self.my_pred.video_to_streamlit(frame)
                processed_frame = processed_frame * 255
                processed_frame = processed_frame.astype(np.uint8)
                # Wait 1 sec
                #time.sleep(1)
                # Return the processed video frame
                return av.VideoFrame.from_ndarray(processed_frame,
format="bgr24")

################################################################################
#############################################################
```

```
# Functions definition
def local_process_video(video_source):
    webrtc_streamer(key="Local Cam", video_processor_factory=lambda:
video_source, #MyLocalVideoProcessor(video_source),
                    media_stream_constraints={"video":True,"audio":False})

def ip_process_video(video_source):
    webrtc_streamer(key="IP Cam", video_processor_factory=lambda:
MyIPVideoProcessor(video_source),
                    media_stream_constraints={"video":True,"audio":False})

################################################################################
############################################################
h_yolo_flag = 0
main_yolo_flag = 0
r_yolo_flag = 0

lh_messages = []
rh_messages = []



################################################################################
############################################################

# Program Start
# load yolo models
h_yolo = my_pred(mymodel_address="./02 - Training/Hand
Commands/runs/segment/train/weights/best.pt")
r_yolo = my_pred(mymodel_address="./02 - Training/Robot
Detection/runs/segment/train/weights/best.pt")

# Page tab setup
st.set_page_config(page_title="CT-TCC DAELT",
                   layout='wide',
                   page_icon='./03 - WebApp/images/home.png')

# Header config
col1, col2, col3 = st.columns(3)
with col2:
    st.image('./03 - WebApp/images/utfbranco1.png', use_column_width = 'auto')
st.markdown("<h1 style='text-align: center; color: white;'>DAELT|CT - Trabalho
de Conclusão de Curso</h1>", unsafe_allow_html=True)

st.write("---")
broker_address = st.text_input('Broker Address: ', "broker.hivemq.com" )
broker_port = int(st.text_input('Broker Port: ',8883))
broker_usr = st.text_input('Insert your username for broker: ')
```

```
broker_pwd = st.text_input('Insert your password for broker:
',type="password")

st.write((broker_address != "") and (broker_usr != "") and (broker_pwd != ""))

if (broker_address != "") and (broker_usr != "") and (broker_pwd != ""):
    h_yolo.set_mqtt(broker_address=broker_address,broker_port=broker_port,brok
er_usr=broker_usr,broker_pwd=broker_pwd)
    if h_yolo_flag == 0:
        h_yolo.MQTT.connect()
        if h_yolo.MQTT.flag == 0 and h_yolo_flag==0:
            h_yolo_flag = 1
        else:
            pass
        h_yolo.MQTT.loop_start()
    else:
        pass

    #r_yolo.set_mqtt(broker_address=broker_address,broker_port=broker_port,bro
ker_usr=broker_usr,broker_pwd=broker_pwd)

    myMQTT = my_mqtt(broker_address=broker_address, port=broker_port,
usrname=broker_usr,pwd=broker_pwd)
    if main_yolo_flag == 0:
        myMQTT.connect()
        myMQTT.subscribe(topic="HandCommands/#")
        myMQTT.subscribe(topic="VIBot/#")
        myMQTT.publish(topic="Main",payload="myMQTT Connected")
        if myMQTT.flag == 0 and main_yolo_flag == 0:
            main_yolo_flag = 1
        else:
            pass
        myMQTT.loop_start()
        if myMQTT.message_topic == "VIBot/Battery":
            st.write(myMQTT.message_payload)

    else:
        pass

# Get information about the IP Cam address
url = st.text_input('IP Cam Address: ','https://127.0.0.1:8080')
url = url+"/video" #"http://192.168.0.16:8080"+"/video"
st.write('The current url is: ', url)

st.write("---")

# Setup and call the streamers to show the commands and robot moves
col4, col5 = st.columns([2,2])
```

```
with col4:
    #hconf = st.slider(label="HC
Confidence",min_value=0.0,max_value=1.0,value=0.75,step=0.01)
    st.write("Local Cam:")
    my_local_processor = MyLocalVideoProcessor(video_source=0)
    local_ctx = local_process_video(my_local_processor)
    #my_detections = my_local_processor.names_list
    #st.write(my_local_processor.names_list)
   # my_cls = my_local_processor.cls_numbers


with col5:
    st.write("IP Cam:")

    #webrtc_streamer(key="IPCam",
video_transformer_factory=OpenCVVideoTransformer)
    OpenCV_ctx = ip_process_video(video_source=url)

st.write("---")

# Show complementary information (Last Command Identified, Robot Battery)
col6, col7 = st.columns([2,2])


#st.image('./03 - WebApp/images/Hand Commands/RHand-Three.png',width=100,)


st.write("---")

###############################################################################
#############################################################

# Page footer
st.text('Trabalho elaborado por Leonardo Alves da Fontoura')

###############################################################################
#############################################################
```

APÊNDICE E - *Firmware* do robô

```
#include "WiFi.h"
#include "PubSubClient.h"
#include <WiFiClientSecure.h>
#include <RoboCore_Vespa.h>

//--------------------------------------------------------------------------

// Replace with your network credentials
const char* ssid = "YOUR WIFI SSID";
const char* password = "YOUR WIFI PWD";

// Add your MQTT Broker IP address or URL
const char* mqtt_server = "YOUR SERVER ADDRESS";
const int mqtt_port = 8883;
const char* broker_usr = "YOUR USER";
const char* broker_pwd = "YOUR PWD";

const char* lh_topic = "HandCommands/LH";
const char* rh_topic = "HandCommands/RH";

uint16_t velocidade = 90;
uint8_t bot_device=8, bot_function=8;
uint32_t mytime = millis();

//--------------------------------------------------------------------------

//Structs:

  struct servo_angulos_t {
    uint8_t min; // angulo minimo [0;180]
    uint8_t max; // angulo maximo [0;180]
    uint8_t atual; // posicao atual
  };

// Vespa variables
  VespaMotors motores;
  VespaServo servos[4];
  const uint16_t SERVO_MAX = 2500;
  const uint16_t SERVO_MIN = 500;
  enum Motor { Base = 0 , Alcance , Elevacao , Garra };
  servo_angulos_t sangulos[4] = { { 0, 180, 90 }, //0, 180, 90
                                  { 40, 180, 70 },  //40, 180, 90
                                  { 80, 180, 90 }, //80, 180, 140
                                  { 70, 160, 160 } }; //70, 160, 100

  VespaBattery vbat;
  uint8_t vbat_critico = 0xFF;
  const uint32_t INTERVALO_ATUALIZACAO_VBAT = 5000; // [ms]
  const uint32_t INTERVALO_LED_VBAT_HIGH = 1000; // [ms]
```

```
  const uint32_t INTERVALO_LED_VBAT_LOW = 500; // [ms]
  uint32_t timeout_vbat, timeout_led_vbat;

  // LED
  const uint8_t PIN_LED = 15;

//----------------------------------------------------------------------

// Callback function header
void callback(char* topic, byte* payload, unsigned int length);

WiFiClientSecure espClient;
PubSubClient client(espClient);

// HiveMQ Cloud Let's Encrypt CA certificate
static const char *root_ca PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh
cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4
WhcNMzUwNjA0MTEwNDM4WjBPMQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJu
ZXQgU2VjdXJpdHkgUmVzZWFyY2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBY
MTCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAK3oJHP0FDfzm54rVygc
h77ct984kIxuPOZXoHj3dcKi/vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+
0TM8ukj13Xnfs7j/EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U
A5/TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW
T8KOEUt+zwvo/7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/ELNKjD+Jo2FR3qyH
B5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC
B5iPNgiV5+I3lg02dZ77DnKxHZu8A/lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv
KBds0pjBqAlkd25HN7rOrFleaJ1/ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn
OlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3SzynTn
jh8BCNAw1FtxNrQHusEwMFxIt4I7mKZ9YIqioymCzLq9gwQbooMDQaHWBfEbwrbw
qHyGO0aoSCqI3Haadr8faqU9GY/rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI
rU7m2Ys6xt0nUW7/vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV
HRMBAf8EBTADAQH/MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq
hkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1XIpu+ILlaS/V9lZL
ubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ
3BebYhtF8GaV0nxvwuo77x/Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/KK
NFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5
ORAzI4JMPJ+GslWYHb4phowim57iaztXOoJwTdwJx4nLCgdNbOhdjsnvzqvHu7Ur
TkXWStAmzOVyyghqpZXjFaH3pO3JLF+l+/+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC
jNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc
oyi3B43njTOQ5yOf+1CceWxG1bQVs5ZufpsMljq4Ui0/1lvh+wjChP4kqKOJ2qxq
4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/9yi0GE44Za4rF2LN9d11TPA
mRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d
emyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/+jh5Xos1AnX5iItreGCc=
-----END CERTIFICATE-----
)EOF";
```

```
//--------------------------------------------------------------------

void setup() {
  // put your setup code here, to run once:

  // Initialize Serial Monitor
  Serial.begin(115200);

  // configura os servos
  servos[0].attach(VESPA_SERVO_S1, SERVO_MIN, SERVO_MAX);
  servos[1].attach(VESPA_SERVO_S2, SERVO_MIN, SERVO_MAX);
  servos[2].attach(VESPA_SERVO_S3, SERVO_MIN, SERVO_MAX);
  servos[3].attach(VESPA_SERVO_S4, SERVO_MIN, SERVO_MAX);
  // atualiza os motores para as posicoes iniciais
  for(uint8_t i=0 ; i < 4 ; i++){
    servos[i].write(sangulos[i].atual);
  }

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Connect to MQTT Broker
  espClient.setCACert(root_ca);
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);

}

void callback(char* topic, byte* message, unsigned int length) {
  String messageTemp;

  //int bot_device, bot_function;
  unsigned int newtime = millis();

  int k;

  for (int i = 0; i < length; i++) {
    messageTemp += (char)message[i];
  }

  // Print out received message
  Serial.print("Received message: ");
  Serial.println(messageTemp);
```

```
  Serial.printf("bot_device: %d \n",bot_device);
  Serial.printf("bot_function: %d \n",bot_function);

  // If 'ON' message received, turn flashlight on
  // if (messageTemp == "ON") {
  //   digitalWrite(FLASH_PIN, HIGH);
  //   Serial.println("Flashlight turned ON");
  // }

  // // If 'OFF' message received, turn flashlight off
  // if (messageTemp == "OFF") {
  //   digitalWrite(FLASH_PIN, LOW);
  //   Serial.println("Flashlight turned OFF");
  // }


  // if (topic == "HandCommands/LH"){
  //   if (messageTemp == "LHand-One"){
  //     bot_device = 1;
  //   }
  //   else if (messageTemp == "LHand-Two"){
  //     bot_device = 2;
  //   }
  // }

  if (strcmp(topic,lh_topic)==0){
    device_set(messageTemp);
  }

  //   if (strcmp(topic,rh_topic)){
  //     if (messageTemp == "RHand-Null"){bot_function = 0;}
  //     else if (messageTemp == "RHand-One"){bot_function = 1;}
  //     else if (messageTemp == "RHand-Two"){bot_function = 2;}
  //     else if (messageTemp == "RHand-Three"){bot_function = 3;}
  //     else if (messageTemp == "RHand-Four"){bot_function = 4;}
  //     else if (messageTemp == "RHand-Five"){bot_function = 5;}
  //     else if (messageTemp == "RHand-UThumb"){bot_function = 6;}
  //     else if (messageTemp == "RHand-DThumb"){bot_function = 7;}
  // }

    if (strcmp(topic,rh_topic)==0){
      function_set(messageTemp);
    }


  if (bot_device == 1 && bot_function == 1){motores.forward(velocidade);
delay(1500); motores.stop();}
  else if (bot_device == 1 && bot_function == 2){motores.backward(velocidade);
delay(1500); motores.stop();}
```

```
  else if (bot_device == 1 && bot_function == 3){motores.turn(0, velocidade);
delay(1500); motores.stop();}
  else if (bot_device == 1 && bot_function == 4){motores.turn(velocidade, 0);
delay(1500); motores.stop();}
  else if (bot_device == 1 && bot_function == 5){motores.stop();}
  else if (bot_device == 1 && bot_function == 6){if
(velocidade<=90){velocidade = velocidade +
5;}else{velocidade=100;}Serial.printf("Velocidade: %d",velocidade);}
  else if (bot_device == 1 && bot_function == 7){if (velocidade>80){velocidade
= velocidade - 5;}else{velocidade=10;}Serial.printf("Velocidade:
%d",velocidade);}
  else if (bot_device == 2 && bot_function ==
1){if(sangulos[1].atual<=(sangulos[1].max-
30)){sangulos[1].atual=sangulos[1].atual+30;servos[1].write(sangulos[1].atual)
;}
                                                    else{servos[1].write(sangulos[
1].max);}}
  else if (bot_device == 2 && bot_function ==
2){if(sangulos[1].atual<=(sangulos[1].min+30)){sangulos[1].atual=sangulos[1].a
tual-30;servos[1].write(sangulos[1].atual);}
                                                    else{servos[1].write(sangulos[
1].min);}}
  else if (bot_device == 2 && bot_function ==
3){if(sangulos[2].atual<=(sangulos[2].min+30)){sangulos[2].atual=sangulos[2].a
tual-30;servos[2].write(sangulos[2].atual);}
                                                    else{servos[2].write(sangulos[
2].min);}}
  else if (bot_device == 2 && bot_function ==
4){if(sangulos[2].atual<=(sangulos[2].max-
30)){sangulos[2].atual=sangulos[2].atual+30;servos[2].write(sangulos[2].atual)
;}
                                                    else{servos[2].write(sangulos[
2].max);}}
  else if (bot_device == 2 && bot_function ==
0){servos[3].write(sangulos[3].max);}
  else if (bot_device == 2 && bot_function ==
5){servos[3].write(sangulos[3].min);}
  else if (bot_device == 2 && bot_function ==
6){if(sangulos[0].atual<=(sangulos[0].max-
30)){sangulos[0].atual=sangulos[0].atual+30;servos[0].write(sangulos[0].atual)
;}
                                                    else{servos[0].write(sangulos[
0].max);}}
  else if (bot_device == 2 && bot_function ==
7){if(sangulos[0].atual<=(sangulos[0].min+30)){sangulos[0].atual=sangulos[0].a
tual-30;servos[0].write(sangulos[0].atual);}
                                                    else{servos[0].write(sangulos[
0].min);}}
```

```
  mytime = millis();
}

uint8_t device_set(String messageTemp){
    if (messageTemp == "LHand-One"){
      bot_device = 1;
      client.publish("VIBot/Device","Robot",false);
    }
    else if (messageTemp == "LHand-Two"){
      bot_device = 2;
      client.publish("VIBot/Device","Arm",false);
    }
  Serial.printf("Test_bot_device: %d \n",bot_device);
  return bot_device;
}

uint8_t function_set(String messageTemp){


      if (strcmp(messageTemp.c_str(),"RHand-Null")==0){bot_function = 0;}
      else if (strcmp(messageTemp.c_str(),"RHand-One")==0){bot_function = 1;}
      else if (strcmp(messageTemp.c_str(),"RHand-Two")==0){bot_function = 2;}
      else if (strcmp(messageTemp.c_str(),"RHand-Three")==0){bot_function =
3;}
      else if (strcmp(messageTemp.c_str(),"RHand-Four")==0){bot_function = 4;}
      else if (strcmp(messageTemp.c_str(),"RHand-Five")==0){bot_function = 5;}
      else if (strcmp(messageTemp.c_str(),"RHand-UThumb")==0){bot_function =
6;}
      else if (strcmp(messageTemp.c_str(),"RHand-DThumb")==0){bot_function =
7;}
      else if (strcmp(messageTemp.c_str(),"Stop/Reset")==0){bot_function = 8;}

      Serial.printf("Test_bot_function: %d \n",bot_function);
      return bot_function;
}

void bat_loop(){

  // le a tensao da bateria e envia para o cliente
  if(millis() > timeout_vbat){
    // le a tensao da bateria
    //uint32_t
    uint32_t tensao = vbat.readVoltage();
    char str_tensao[10];
    dtostrf(tensao, 4, 2, str_tensao);
    // envia o status da bateria via MQTT
    client.publish("VIBot/Battery",(str_tensao),false);
    // verificar se a tensao esta critica
    if((tensao < 7000) && (vbat_critico == 0xFF)){
```

```
      Serial.printf("Tensao critica (%u mV)\n", tensao);
      vbat_critico = LOW;
      digitalWrite(PIN_LED, vbat_critico);
      timeout_led_vbat = millis() + INTERVALO_LED_VBAT_LOW;
    }
    else{
      if((tensao >= 7000) && (vbat_critico < 0xFF)){
      vbat_critico = 0xFF; // reset
      }
    }
    timeout_vbat = millis() + INTERVALO_ATUALIZACAO_VBAT; // atualiza
  }

    // pisca o LED se a tensao estiver critica
    if(millis() > timeout_led_vbat){
    if(vbat_critico < 0xFF){
      if(vbat_critico == LOW){
        vbat_critico = HIGH;
        timeout_led_vbat = millis() + INTERVALO_LED_VBAT_HIGH;
      } else {
        vbat_critico = LOW;
        timeout_led_vbat = millis() + INTERVALO_LED_VBAT_LOW;
      }
      digitalWrite(PIN_LED, vbat_critico);
    }
    }
}

void loop() {
  // put your main code here, to run repeatedly:

    if (!client.connected()) {
    while (!client.connected()) {
      if (client.connect("ESP32Client", broker_usr, broker_pwd)) {
        client.subscribe("HandCommands/#");
        Serial.println("Connected to MQTT broker");
      } else {
        Serial.println("Failed to connect to MQTT broker... retrying");
        delay(5000);
      }
    }
  }

  client.loop();
  bat_loop();
}
```