INSTITUTO FEDERAL
Bahia
Campus Vitória da Conquista

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

REGULAMENTO DE TRABALHO DE CONCLUSÃO DE CURSO

ANEXO II
SÚMULA DE PROJETO DE PESQUISA

BRENDON SOUSA LIMA

Sistema de Visão Computacional Embarcado para Robótica Autônoma de Baixo Custo

Pré-projeto de pesquisa desenvolvido na linha de pesquisa de **desenvolvimento de sistemas embarcados**, sob orientação do(a) **Prof(ª). Andrique Figueirêdo Amorim**, como requisito de inscrição no componente curricular de Trabalho de Conclusão de Curso I, do Curso de Bacharelado em Sistemas de Informação, do Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Campus Vitória da Conquista

Vitória da Conquista, 2025

INSTITUTO FEDERAL
Bahia
Campus Vitória da Conquista

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

REGULAMENTO DE TRABALHO DE CONCLUSÃO DE CURSO

ANEXO II
SÚMULA DE PROJETO DE PESQUISA

1. Introdução e Contextualização do Problema de Pesquisa

1.1 A Ascensão da Inteligência Artificial na Borda (Edge AI) e o Paradigma da TinyML

A inteligência artificial tem se expandido para além dos servidores de alta potência e ambientes de nuvem, atingindo a fronteira dos dispositivos. Esse movimento é conhecido como *Edge AI* e tem sido impulsionado pela necessidade de processamento local de dados para aplicações que exigem baixa latência, alta privacidade e eficiência energética.[1] Nesse contexto, a disciplina do

Tiny Machine Learning (TinyML) surge como uma nova fronteira, focada em "espremer" modelos de aprendizado de máquina para que operem em dispositivos diminutos, como microcontroladores (MCUs), que possuem apenas algumas centenas de kilobytes de memória.[1]

A ascensão do TinyML não é apenas um avanço tecnológico, mas uma resposta a um imperativo de processamento de dados. Com bilhões de dispositivos IoT gerando volumes massivos de dados, a dependência de servidores na nuvem para a inferência se torna um gargalo de desempenho, latência e segurança.[1] O processamento na borda permite a tomada de decisões em tempo real sem a necessidade de comunicação de rede, o que é vital para aplicações como veículos autônomos e robótica.[1] O projeto aqui proposto se insere diretamente nessa tendência, posicionando a robótica autônoma no centro da evolução da Edge AI.

Contudo, a implementação de modelos de aprendizado profundo em dispositivos embarcados de recursos limitados não é trivial. A pesquisa no campo do TinyML indica que a principal restrição não é o número de parâmetros do modelo, mas sim a memória de ativação.[1] Isso significa que modelos originalmente projetados para plataformas móveis ou de nuvem, como os MobileNets ou ResNets, não se adequam de forma eficiente aos dispositivos minúsculos.[1] Dessa forma, o sucesso de uma aplicação de TinyML depende de um "co-design" — um design conjunto e otimizado do algoritmo e da arquitetura do sistema.[1] A presente proposta, ao focar na otimização de um modelo de visão computacional especificamente para as restrições e capacidades do ESP32-S3, adere a essa abordagem de pesquisa de ponta.

ANEXO II
SÚMULA DE PROJETO DE PESQUISA

1.2 A Robótica Autônoma em Ambientes Dinâmicos: O Caso do Futebol de Robôs

A robótica autônoma, especialmente em ambientes dinâmicos e imprevisíveis, como um campo de futebol, representa um desafio complexo e relevante no avanço da inteligência artificial e da mecatrônica.[4] A capacidade de um robô de detectar, rastrear e interagir com objetos em movimento, como uma bola, é central para o desenvolvimento de sistemas de percepção visual e controle de movimento. Competições como a

RoboCup Small Size League servem como um domínio de pesquisa crucial, impulsionando a inovação em coordenação multi-agentes e controle em tempo real.[6]

Uma distinção fundamental na pesquisa em robótica de futebol é a arquitetura do sistema de visão. Ligas tradicionais como a RoboCup SSL utilizam frequentemente um sistema de visão centralizado, externo ao campo, que processa a imagem de várias câmeras e transmite os dados de posição dos objetos aos robôs.[6] Essa abordagem oferece uma visão global e completa do ambiente de jogo.[7]

Em contraste, a presente proposta explora uma arquitetura de visão **embarcada e on-board**, onde o próprio robô é responsável por sua percepção. Essa abordagem enfrenta um desafio mais significativo e, de certa forma, mais realista.[8] A visão embarcada limita o campo de visão do robô, e o sistema de percepção deve lidar com movimentos bruscos, tremores e iluminação variável do ambiente.[8] A pesquisa proposta não apenas desenvolve um sistema de visão, mas aprofunda a autonomia de percepção a um nível mais complexo. Nesse cenário, a baixa latência proporcionada pelo TinyML e pelo processamento on-chip se torna não apenas uma vantagem, mas um requisito fundamental para a viabilidade do projeto, permitindo que o robô reaja a eventos em tempo real, sem o atraso da comunicação externa.

ANEXO II
SÚMULA DE PROJETO DE PESQUISA

1.3 Apresentação do Problema: A Necessidade de Sistemas de Visão Acessíveis e Eficientes

A viabilidade de projetos avançados de robótica e inteligência artificial tem sido historicamente limitada pelo alto custo de plataformas computacionais de alto desempenho, como as da linha NVIDIA Jetson, que, embora poderosas, representam um investimento substancial.[9] Embora o Raspberry Pi tenha democratizado o acesso a computação de placa única, ele ainda é um computador (SBC), não um microcontrolador, e sua arquitetura consome mais energia e é mais cara do que a de um MCU de baixo custo como o ESP32.[11] A principal barreira para a aplicação de sistemas de visão embarcados em projetos educacionais e para a comunidade

maker tem sido a ausência de uma plataforma que combine baixo custo, alta eficiência energética e capacidade técnica suficiente para inferência de visão computacional.

O presente projeto aborda diretamente essa lacuna. Ao demonstrar a aplicação de uma solução de menos de 15 dólares para um problema que normalmente exige hardware de centenas ou milhares de dólares, o projeto contribui para a democratização da robótica complexa e da inteligência artificial.[1] O uso do ESP32-S3 é um ponto crítico, pois versões anteriores do ESP32, apesar do baixo custo, não possuíam as capacidades necessárias para o processamento de visão complexo.[1] O modelo S3, com sua aceleração de IA e memória PSRAM, transforma o chip em uma plataforma viável para a execução de modelos de detecção de objetos de forma eficiente e local.

2. Fundamentação Teórica

2.1 Visão Computacional Embarcada: Conceitos Fundamentais e Arquiteturas

A Visão Computacional, como campo de estudo, visa permitir que máquinas "vejam" e interpretem o mundo a partir de dados visuais. A Visão de Máquina, por sua vez, é a aplicação industrial da visão computacional, focada em tarefas precisas e de alta velocidade em ambientes controlados. A **Visão Embarcada** (Embedded Vision) é um subcampo que se refere à integração dessas capacidades em sistemas compactos, como microcontroladores e SoCs.[14] Diferente dos sistemas tradicionais que dependem de computadores externos de propósito geral, a visão embarcada opera em hardware dedicado, permitindo análise e tomada de decisão em tempo real em dispositivos que podem ser móveis ou de baixo custo.[8]

ANEXO II
SÚMULA DE PROJETO DE PESQUISA

O fluxo de trabalho de um sistema de visão embarcada tipicamente envolve quatro etapas: 1) **Aquisição de Imagem**, onde uma câmera ou sensor óptico captura dados visuais; 2) **Pré-processamento**, que melhora a qualidade da imagem (ex: redução de ruído, ajuste de contraste); 3) **Inferência**, onde um modelo de aprendizado de máquina analisa a imagem e extrai informações relevantes; e 4) **Ação ou Saída**, onde os dados processados são usados para acionar um atuador ou gerar um comando.[8] A arquitetura do projeto aqui proposto segue precisamente esse modelo, com o ESP32-S3 servindo como o núcleo de processamento para a etapa de inferência.

2.2 O Universo do TinyML: Desafios e Estratégias de Otimização

O sucesso de um sistema TinyML depende da superação de desafios impostos por hardware com recursos estritamente limitados, como memória RAM na ordem de kilobytes e processadores de baixa frequência.[2] Para que modelos de aprendizado de máquina caibam e operem de forma eficiente nesses dispositivos, são aplicadas estratégias de otimização especializadas.[1] Uma das mais importantes é a

quantização, que reduz a precisão dos parâmetros do modelo de representações de ponto flutuante para inteiros de 8 ou 16 bits.[3] Essa técnica diminui drasticamente o tamanho do modelo e o consumo de memória, ao custo de uma pequena perda na precisão. Outras estratégias incluem a poda de redes neurais (removendo conexões não essenciais) e o uso de arquiteturas de rede otimizadas.

A necessidade de um design conjunto (co-design) entre o algoritmo e o sistema é a principal consideração no campo do TinyML. Ao contrário da computação na nuvem, onde a otimização pode se concentrar em um único aspecto, o TinyML exige que os modelos sejam criados com as restrições do hardware em mente.[1] Isso levou ao surgimento de arquiteturas como a MCUNet [1] e o TinyissimoYOLO [3], que são otimizadas para ambientes com menos de 500 KB de memória. O modelo FOMO (Faster Objects, More Objects) também se destaca, pois é uma versão de detecção de objetos otimizada para microcontroladores, projetada para executar com eficiência em ambientes com restrições de memória.[15] A escolha do ESP32-S3 como plataforma é crucial, pois sua arquitetura e memória PSRAM permitem a execução de modelos otimizados para detecção de objetos, como o FOMO, que seria inviável em microcontroladores mais básicos.[16]

INSTITUTO FEDERAL
Bahia
Campus Vitória da Conquista

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

REGULAMENTO DE TRABALHO DE CONCLUSÃO DE CURSO

ANEXO II
SÚMULA DE PROJETO DE PESQUISA

2.3 Plataformas de Hardware para Visão Embarcada: Análise Comparativa e Justificativa da Escolha

A escolha do ESP32-S3 como a plataforma central do projeto não se baseia apenas em seu baixo custo, mas em uma análise técnica que o posiciona como a plataforma idealmente balanceada para os objetivos do projeto. Para formalizar essa justificativa, a Tabela 1 a seguir compara o ESP32-S3 com outras plataformas comuns de robótica e visão embarcada, destacando as características que o tornam a escolha mais apropriada.

Tabela 1: Análise Comparativa de Plataformas de Visão Embarcada

Característica	ESP32-S3 + ESP-CAM	Raspberry Pi + Câmera	NVIDIA Jetson Nano
Tipo de Dispositivo	Microcontrolador (SoC)	Computador de Placa Única (SBC)	Computador de Placa Única (SBC)
Custo Aproximado	$10 - $15 [13]	$35 - $100+ [11]	$100 - $200+ [10]
Processador	Dual-core 32-bit LX7, 240 MHz [16]	ARM Cortex-A53 ou superior	NVIDIA Maxwell GPU (128 CUDA Cores)
Memória	512KB SRAM, 8MB PSRAM [16]	1GB, 2GB ou 4GB RAM [12]	4GB LPDDR4
Aceleração de IA	Aceleração de Rede Neural Integrada [16]	Não dedicada; CPU/GPU de propósito geral	Unidade de Processamento de IA (GPU) [9]
Consumo de	Milhares de miliwatts [2]	Vários watts [12]	5 - 10 watts ou mais [10]

ANEXO II
SÚMULA DE PROJETO DE PESQUISA

Energia			
Aplicações Típicas	IoT, automação, projetos de baixo consumo	Servidores, desktops, projetos educacionais	Robótica de alto desempenho, Edge AI industrial
Processamento de Visão	Inferência on-chip com TinyML [16]	Processamento local ou de rede	Processamento de alta performance para redes complexas [9]

A análise comparativa demonstra que, embora plataformas como o Raspberry Pi e o NVIDIA Jetson Nano ofereçam maior poder de processamento e memória, elas são fundamentalmente diferentes do ESP32-S3. O Raspberry Pi é um computador em miniatura, mais adequado para tarefas que exigem um sistema operacional completo e maior capacidade de processamento.[11] O Jetson Nano é uma solução de alto desempenho, voltada para projetos de pesquisa e aplicações industriais que requerem poder de processamento massivo para redes neurais complexas.[9]

Em contraste, o ESP32-S3 se destaca por ser uma solução de microcontrolador (MCU) que, com a adição de recursos específicos, atinge o ponto ideal de viabilidade para o projeto proposto. A memória **PSRAM (8MB)** é um diferencial crítico, pois mitiga o gargalo de memória de ativação dos modelos de TinyML, permitindo que modelos de detecção de objetos mais complexos sejam executados.[16] Além disso, a aceleração de hardware para computação de rede neural (

hardware neural network computation) e o custo reduzido tornam o ESP32-S3 a plataforma mínima viável para a tarefa de detecção de objetos em tempo real em um sistema embarcado de baixo custo. A decisão de usar esta plataforma reflete um design criterioso, que busca um equilíbrio entre capacidade técnica e acessibilidade, fundamental para a democratização da robótica.

ANEXO II
SÚMULA DE PROJETO DE PESQUISA

3. Proposta Metodológica e Arquitetura do Sistema

3.1 Visão Geral da Arquitetura do Sistema Embarcado

A arquitetura do sistema é composta por três camadas principais: a camada de percepção, a de processamento e a de controle. A camada de percepção, utilizando o módulo de câmera ESP-CAM (OV3660), captura o ambiente de jogo. O ESP32-S3 atua como a unidade de processamento central, executando o modelo de aprendizado de máquina para inferência on-chip e extraindo as coordenadas (x,y) dos objetos de interesse. A camada de controle, embora não seja o foco principal deste projeto, receberá os dados numéricos do processador de visão para comandar os atuadores do robô (motores, etc.). A interconexão entre essas camadas será crucial para o sucesso do projeto.

A seguir, a Tabela 2 apresenta os parâmetros técnicos e especificações do sistema, fornecendo uma visão clara do escopo do projeto e dos alvos de desempenho.

Tabela 2: Parâmetros e Especificações do Sistema

Categoria	Componente / Parâmetro	Especificação / Alvo	Fonte de Referência
Hardware	SoC	ESP32-S3	[16]
	Módulo de Câmera	OV3660 (3MP)	[16]
	Memória de Trabalho	512KB SRAM + 8MB PSRAM	[16]
Software	Framework de ML	TensorFlow Lite Micro, Edge Impulse	[3]

	Modelo de Detecção	Arquitetura FOMO (TinyML)	[15]
Parâmetros de Desempenho	Resolução de Imagem	Otimizada para o modelo (e.g., 96x96 pixels)	[15]
	Latência de Inferência	Mínima (tempo real, sem atrasos de rede)	[1]
	Taxa de Quadros (FPS)	Alvo mínimo de 15 FPS	Dados da pesquisa

3.2 Metodologia de Treinamento e Otimização do Modelo de Machine Learning

A metodologia de desenvolvimento do modelo de visão seguirá um fluxo de trabalho estruturado, com ênfase na utilização de plataformas que simplificam o processo de TinyML, como o Edge Impulse.[15] As etapas a serem seguidas são:

1. **Coleta de Dados:** Imagens da bola e das traves serão capturadas em diversas condições de iluminação, ângulos e distâncias, diretamente com o módulo de câmera ESP-CAM, no ambiente de jogo. O objetivo é criar um conjunto de dados robusto que represente as condições reais de operação do robô.[15]
2. **Rotulagem e Preparação do Conjunto de Dados:** O conjunto de dados coletado será enviado para a plataforma Edge Impulse, onde as imagens serão rotuladas com caixas delimitadoras (bounding boxes) que indicam a localização da bola e das traves. O processo de rotulagem é fundamental para o treinamento de modelos de detecção de objetos.[15]
3. **Design do Modelo (Impulse Design):** Na plataforma, será selecionado o pipeline de processamento de imagem (como conversão para escala de cinza) e, mais importante, a arquitetura de rede neural a ser utilizada para a detecção de objetos.

Uma arquitetura como o FOMO será a escolha preferencial devido à sua otimização para ambientes com recursos limitados.[15]

4. **Treinamento e Otimização:** O modelo será treinado na nuvem da plataforma, utilizando o conjunto de dados rotulado. Durante essa etapa, serão aplicadas as estratégias de otimização, como a quantização, para garantir que o modelo final caiba e execute eficientemente no ESP32-S3.[3]
5. **Implantação:** Após o treinamento e validação, o modelo será exportado como uma biblioteca de C++ ou Arduino, que é compatível com o ambiente de desenvolvimento do ESP32-S3. O código será gravado no chip, ativando a memória PSRAM para armazenar o modelo e dados de inferência em tempo real.[15]

3.3 O Processamento On-Chip: A Inovação Central

Uma das inovações centrais deste projeto é a execução de todo o processamento de visão diretamente no chip ESP32-S3, em contraste com arquiteturas alternativas que utilizam o ESP32 como um mero servidor de streaming de vídeo que envia os quadros para um PC para análise.[17] Essa abordagem on-chip é viável devido a duas características críticas do ESP32-S3: a memória

PSRAM e a **aceleração de IA** de hardware.[16]

A memória PSRAM é essencial, pois permite armazenar o modelo de aprendizado de máquina e os dados temporários de ativação da rede neural, superando a limitação de memória SRAM interna de versões anteriores.[16] O processamento on-chip garante uma latência mínima, eliminando qualquer atraso de comunicação de rede ou Wi-Fi que seria introduzido por uma arquitetura de streaming para um computador externo.[3] A baixa latência é um requisito para a operação do robô em um ambiente dinâmico, onde a tomada de decisão precisa ser instantânea para que o robô reaja a tempo.

4. Integração do Sistema de Visão com o Controle do Robô

4.1 Conversão de Coordenadas: A Transição do Espaço de Pixels para o Espaço Físico

A saída do sistema de visão é uma série de dados numéricos que representa a localização dos objetos detectados em coordenadas de pixels (x,y).[16] No entanto, para que o robô possa se mover em direção à bola ou se posicionar em relação às traves, essas coordenadas de pixels precisam ser convertidas para o sistema de coordenadas do mundo físico (por exemplo, em milímetros ou centímetros).[18]

O processo de conversão será realizado por meio de uma calibração geométrica. Utilizando uma metodologia similar à regressão linear, o sistema de visão será calibrado com base em pontos de referência conhecidos. Esses pontos terão coordenadas no espaço físico do campo e suas posições em pixels serão registradas pelo sistema de visão.[19] Com um conjunto de pelo menos três pontos de referência, será possível calcular os coeficientes de transformação que mapeiam as coordenadas de pixels para as coordenadas físicas.[19] A relação de transformação pode ser expressa através de equações como:

$$Rx = A \cdot Px + B \cdot Py + C$$

$$Ry = D \cdot Px + E \cdot Py + F$$

onde Rx,Ry são as coordenadas do robô no espaço físico e Px,Py são as coordenadas dos objetos em pixels na imagem.[19]

4.2 O Loop de Feedback e o Controle PID para o Movimento do Robô

A integração do sistema de visão com o controle de movimento é a etapa final e crucial para a autonomia do robô. Os dados de coordenadas físicas fornecidos pelo sistema de visão servirão como o **input** para um laço de controle de feedback.[5] A base desse sistema de controle será um controlador

PID (Proporcional, Integral, Derivativo), amplamente utilizado em robótica para minimizar o erro entre um valor desejado e um valor medido.[20]

A dinâmica do controle funcionará da seguinte forma:

- O **valor medido** será a posição atual da bola (ou trave) fornecida pelo sistema de visão embarcado.
- O **ponto de ajuste** (setpoint) será a posição ideal, por exemplo, o centro do campo de visão da câmera, que representa a direção para a qual o robô deve se mover.
- O **erro** é a diferença entre o ponto de ajuste e o valor medido.[21]
- O controlador PID calculará a correção necessária a ser aplicada aos motores do robô para minimizar esse erro, fazendo com que o robô se alinhe com o objeto de interesse.
- A parte **proporcional (P)** ajustará a resposta com base no erro atual, a **integral (I)** eliminará erros acumulados ao longo do tempo, e a **derivativa (D)** amortecerá o sistema, prevendo a tendência futura do erro para evitar overshoot.[20]

A baixa latência proporcionada pela arquitetura TinyML é o fator que torna o sistema de controle viável. Um atraso significativo na percepção visual resultaria em um erro de feedback defasado, levando a um controle instável e ineficaz do robô. Portanto, a união entre a percepção visual de baixa latência e o controle PID cria um sistema robusto para a tomada de decisões em tempo real em um ambiente dinâmico.[5]

5. Contribuições, Relevância e Impacto do Projeto

5.1 Otimização de Custo e Energia: A Viabilidade de Soluções de Baixo Custo

A principal contribuição deste projeto é a prova de conceito de que sistemas complexos de percepção visual e controle para robôs autônomos podem ser construídos com hardware de baixo custo. A análise comparativa demonstra que o ESP32-S3, com seu consumo de energia na ordem de miliwatts, é uma alternativa energeticamente mais eficiente a plataformas que consomem vários watts.[12] A viabilidade financeira e de energia abre as portas para a adoção de projetos de IA em contextos onde a limitação de orçamento é uma barreira.

5.2 Democratização da Robótica e Potencial de Escalabilidade

Ao provar a viabilidade de uma solução acessível, o projeto contribui para a democratização da robótica avançada.[1] Ele oferece um modelo escalável e replicável que pode ser adotado por estudantes, educadores e entusiastas, permitindo que a inovação em robótica autônoma prospere em um ecossistema mais amplo, fora de laboratórios de pesquisa de alto orçamento. Essa acessibilidade fomenta o desenvolvimento de projetos futuros e o aprendizado de conceitos avançados de Edge AI.

5.3 Posicionamento do Projeto em Relação ao Estado da Arte em Robótica Autônoma

O projeto aqui proposto se posiciona como um avanço na área de Edge AI e robótica autônoma. Diferente de soluções de alta potência que dependem de GPUs e processadores caros [9] ou de sistemas de visão externos, que simplificam o problema ao centralizar a percepção [7], este trabalho aborda o desafio de autonomia de forma mais holística. Ele demonstra a capacidade de um robô de baixo custo de perceber e interagir com seu ambiente de forma independente e em tempo real. Essa abordagem, que combina a eficiência do TinyML com a visão embarcada, é um campo de pesquisa de alta relevância, que reflete a tendência de levar a inteligência artificial para o mundo físico.

6. Referências Bibliográficas

- [1] https://arxiv.org/html/2403.19076v2
- [22] https://www.mdpi.com/2079-9292/13/17/3562
- [8] https://www.digitalsense.ai/blog/what-is-embedded-vision
- [14] https://plasticservicecompany.com/en/articles/machine-vision-vs-embedded-vision-vs-computer-vision/
- [23] https://www.routledge.com/Low-Power-Computer-Vision-Improve-the-Efficiency-of-Artificial-Intelligence/Thiruvathukal-Lu-Kim-Chen-Chen/p/book/9780367755287
- [24] https://www.mdpi.com/1424-8220/22/3/1280
- [3] https://medium.com/@CodeWithHannan/the-rise-of-tinyml-running-ml-models-on-microcontrollers-0d7a61929e65
- [2] https://thinkrobotics.com/blogs/learn/tinyml-applications-on-microcontrollers-revolutionizing-edge-ai
- [16] https://wiki.dfrobot.com/SKU_DFR1154_ESP32_S3_AI_CAM
- [13] https://www.hiwonder.com/products/esp32-cam-ai-vision-module
- [17] https://www.makerguides.com/object-detection-with-esp32-cam-and-yolo/
- [25] https://docs.espressif.com/projects/esp-dl/en/latest/esp-dl-en-master.pdf
- [15] https://eloquentarduino.com/posts/esp32-cam-object-detection

- [26]https://www.reddit.com/r/esp32/comments/1b0k6pn/esp32cam_is_this_a_reasonable_plan/
- [27]https://www.reddit.com/r/esp32/comments/1j4q9fm/ml_algorithms_on_esp32/
- [28]https://www.hackster.io/mjrobot/exploring-machine-learning-with-the-new-xiao-esp32s3-6463e5
- [18]https://www.solisplc.com/tutorials/robot-coordinate-frames-and-points
- [19]https://forums.ni.com/t5/Machine-Vision/Coordinate-transformation/td-p/3850950
- [4]https://fse.studenttheses.ub.rug.nl/36594/1/bAI2025RodrigoSierra.pdf
- [5]http://www.et.byu.edu/~beard/papers/preprints/JohnsonEtAl02.pdf
- [21]https://forum.dronebotworkshop.com/technology/pid-controller-for-robotics/
- [20]https://www.researchgate.net/publication/317556526_Vision-Based_Robot_Following_Using_PID_Control
- [29]https://www.unitxlabs.com/resources/robotics-actuators-machine-vision-system-precision-automation/
- [9]https://www.nvidia.com/en-us/industries/robotics/
- [11]https://qsmsemiconductores.com/esp32-vs-raspberry-pi/
- [12]https://www.elprocus.com/difference-between-esp32-vs-raspberry-pi/
- [10]https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetbot-ai-robot-kit/
- [30]https://thinkrobotics.com/blogs/learn/building-autonomous-robots-a-comprehensive-guide
- [31]https://www.rapidinnovation.io/post/computer-vision-in-sports-training
- [32]https://voxel51.com/blog/how-computer-vision-is-changing-sports
- [7]https://user.informatik.uni-bremen.de/tlaue/publications/SSL-Vision-RoboCup-2009.pdf
- [6]https://ssl.robocup.org/