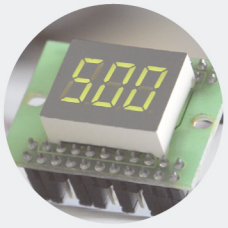
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Análise e Proposta de Melhorias do Pipeline de processamento do wiiGO

PEDRO MANUEL DA SILVA RIBEIRO
Outubro de 2017

POLITÉCNICO
DO PORTO

ISEP

Instituto Superior de Engenharia do Porto

Análise e Proposta de Melhorias do Pipeline de processamento do wiiGO

Mestrado em Engenharia Eletrotécnica e de Computadores -
Área de Especialização de Sistemas Autónomos

Pedro Manuel da Silva Ribeiro
Nº 1110489

31 de Outubro de 2017

Dissertação, para satisfação parcial dos requisitos do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Pedro Manuel da Silva Ribeiro
Nº 1110489

Orientação Cientifica: André Miguel Pinheiro Dias

Empresa: Follow Inspiration, S.A.
Supervisão: José Eduardo de Sampaio e Castro Xavier

Mestrado em Engenharia Electrotécnica e de Computadores -
Área de Especialização de Sistemas Autónomos

Agradecimentos

Gostaria de utilizar esta secção para agradecer ao meu orientador, Professor André Dias pela ajuda disponibilizada ao longo deste projeto.

Quero agradecer à *Follow Inspiration* por me proporcionar esta oportunidade de estágio curricular.

Agradeço também a todos os colegas da *Follow Inspiration* pela sua amabilidade e disponibilidade. Agradeço ao meu supervisor Engº José Xavier pela ajuda e acompanhamento que me deu ao longo deste projeto.

A todos os meus colegas e amigos de curso que participaram neste percurso comigo. A todos os meus amigos de longa data pela amizade e companheirismo.

A todos os membros do Laboratório de Sistemas autónomos que me acompanharam ao longo destes últimos anos.

Por fim, um agradecimento especial aos meus pais e ao meu irmão, pela paciência apoio e incentivo durante a elaboração desta dissertação.

Esta página foi intencionalmente deixada em branco.

Resumo

Este trabalho consiste na análise e proposta de melhorias do pipeline de processamento do wiigo. A presente dissertação foi integrada num estágio curricular na empresa *Follow Inspiration*. Um dos produtos desenvolvido por esta empresa é um carrinho de compras autónomo com o nome de wiigo que se integra na categoria de robô de serviço. Atualmente este robô tem todo o seu processamento centralizado num computador localizado na sua base.

Efetuou-se um estudo das características técnicas e funcionalidades dos robôs de serviço.

Foram usadas e testadas varias ferramentas de analise de performance de um sistema ROS, incluído o *rostopic statistics*, *rosprofiler* e ARNI. Com recurso a estas ferramentas foi caracterizado o desempenho da configuração atual de hardware e software do wiigo.

Foram propostas duas novas arquiteturas de software e uma nova arquitetura de hardware. A arquitetura de hardware irá consistir em dois computadores, um localizado na base e outro na cabeça do wiigo ligados por Ethernet. Será feita a divisão do processamento pela cabeça do robô onde estão localizadas as câmaras e a base onde estão localizados o resto dos sensores. As arquiteturas de software escolhidas visam balancear a execução dos *nodes* ROS e testar a performance de diferentes processadores.

De modo a aumentar a robustez do sistema foi testada a viabilidade do uso do *middleware* DDS (*data distribuition service*) para comunicação entre computadores. Foi criado um pacote ROS que faz uso do DDS para substituir a camada de comunicação do ROS. Como alternativa ao DDS foi também testado o pacote ROS *multimaster fkie*.

Todas as configurações foram testadas fisicamente no wiigo com sucesso. Os resultados dos diferentes testes vão ser apresentados sob a forma de gráficos e tabelas. Será feita uma analise das vantagens e desvantagens das diversas arquiteturas propostas em relação à arquitetura atual. Vai ser apresentada uma configuração de hardware e software final e justificada a sua viabilidade.

Palavras-Chave: Robótica, Robôs de serviço, processamento distribuído, *middlewares*, ROS, DDS, Linux.

Esta página foi intencionalmente deixada em branco.

Abstract

This work consists of the analysis and proposal of improvements in the wiiGO processing pipeline. The present dissertation was integrated in a curricular internship in the company *Follow Inspiration*. One of the products developed by this company is an autonomous shopping cart with the name of wiigo that is integrated into the category of service robot. Currently this robot has all its processing centralized in a computer located in its base.

A study of the technical characteristics and functionalities of the service robots was carried out.

Several performance analysis tools for a ROS system were used and tested, including rostopic statistics, rosprofiler and ARNI. The performance of the current wiigo hardware and software configuration was characterized using these tools.

Two new software architectures and a new hardware architecture were proposed. The hardware architecture will consist of two computers, one located at the base and the other at the head of the wiigo connected by Ethernet. The processing will be divided by the head of the robot where the cameras are located and the base where the rest of the sensors are located. The chosen software architectures are designed to balance the execution of ROS nodes and to test the performance of different processors.

In order to increase the robustness of the system it was tested the feasibility of using DDS (data distribution service) as a means of communication between computers. A ROS package was created for this that uses the DDS to communicate. As an alternative to DDS, the *multimaster fkie* package was also tested.

All architectures have been physically tested on wiigo successfully. The results of the different tests will be presented in the form of graphs and tables. An analysis was made of the advantages and disadvantages of the various proposed architectures in relation to the current architecture. A final hardware and software configuration will be presented and its viability justified.

Palavras-Chave: Robotics, Service Robots, distributed processing, robotics middleware, ROS, DDS, Linux.

Esta página foi intencionalmente deixada em branco.

Conteúdo

Esta página foi intencionalmente deixada em branco.

Lista de Figuras

Esta página foi intencionalmente deixada em branco.

Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

Acronyms

CAN	Controller Area Network
ISO	International Organization for Standardisation
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RAM	Random-access memory
USB	Universal Serial Bus
FPS	Frames per second
TTL	Time to Live
ROS	Robot Operating System
HDMI	High Definition Multimedia Interface
LCD	Liquid Crystal Display
GPU	Graphics Processing Unit
CPU	Central Processing Unit
SSH	Secure Shell
LRF	laser range finder
PC	Personal Computer
LED	Light-emitting diode
RGB	Red Green Blue
IMU	inertial measurement unit
DDS	Data Distribution Service
NTP	Network Time Protocol
LAN	Local Area Network
QoS	Quality of Service
XML	eXtensible Markup Language
API	Application Programming Interface

Esta página foi intencionalmente deixada em branco.

1 Introdução

1.1 Enquadramento e motivação

Esta dissertação foi efetuada no âmbito da unidade curricular Tese/Dissertação do segundo ano do Mestrado de Engenharia Eletrotécnica e Computadores no ramo de Sistemas Autónomos do Instituto Superior de Engenharia do Porto em contexto de estágio curricular na empresa *Follow Inspiration*.

Atualmente com a tecnologia a evoluir exponencialmente e o rápido desenvolvimento de aplicações robóticas, tornou-se possível a criação de sistemas autónomos que podem ajudar ou até substituir os humanos em tarefas repetitivas e monótonas.

Para além da sua importância na industria, os robôs começaram a entrar também em ambientes domésticos e comerciais com o objetivo de fornecer serviços e entretenimento aos seus utilizadores. Enquanto que os robôs industriais tradicionais realizam as suas tarefas em ambientes estruturados sem contacto direto com pessoas, os robôs de serviço trabalham geralmente em ambientes não estruturados e colaboram diretamente com humanos.

Por definição, um robô de serviço é um robô que executa tarefas úteis para os humanos mas que não é usado para aplicações industriais. Este tipo de robôs pode ser subdividido em duas categorias, robô de serviço pessoal e robô de serviço profissional[1]. Um robô de serviço pessoal é usado para aplicações não comerciais e um robô de serviço profissional para aplicações comerciais.

De acordo com um relatório da união europeia [2] é previsto que de 2015 a 2018 sejam

vendidas em todo o mundo cerca de 152.375 unidades de robôs de serviço para aplicações profissionais perfazendo um valor total de 19.6 biliões de dólares. No entanto é esperado que estes valores sejam excedidos, o que mostra que este mercado irá ter grande procura no futuro.

A *Follow Inspiration* é uma empresa que desenvolve tecnologias na área da robótica móvel, visão computacional e inteligência artificial, focando-se essencialmente em melhorar a experiência do utilizador.

O carrinho de compras autónomo *WiiGo*, figura 1.1, é um dos produtos atualmente em desenvolvimento pela *Follow Inspiration*. Foi criado para operar em superfícies comerciais de retalho enquadrando-se portanto na categoria de robô de serviço profissional.

Este robô foi desenvolvido com o objetivo de ajudar nas compras de pessoas com mobilidade reduzida. O WiiGo é compatível também com cadeiras de rodas, não sendo necessário qualquer tipo de adaptação ou marcador externo.

Figura 1.1: *Carrinho de compras autónomo wiiGO.*

O processo de funcionamento é simples e intuitivo bastando ao utilizador carregar num botão, esperar uns segundos para o robô fazer o reconhecimento da sua face e poderá começar as suas compras. Com este robô o utilizador consegue controlar o carrinho de compras sem ter de fazer esforço físico, podendo deslocar-se normalmente.

A versão atual do wiiGO tem um único computador responsável pela execução de todos os processos de software responsáveis pelo funcionamento do robô, (perceção, controlo, navegação, etc).

A dissertação visa a realização da conceção e desenvolvimento de um sistema distri-

buído de processamento utilizando dois computadores sendo que um deles ficará responsável pela perceção (módulo computacionalmente mais exigente) e o outro pelos restantes módulos. Pretende-se também desenvolver um sistema de comunicação entre os restantes módulos baseado no protocolo que cumpra a modularidade pretendida para o robô wiigo.

1.2 Objetivos

Nesta dissertação é abordada otimização da arquitetura de processamento distribuído do robô wiigo em termos de hardware e software. Assim sendo os objetivos que se pretendem alcançar com o desenvolvimento deste projeto podem ser resumidos nos seguintes pontos:

- Estudo da características técnicas/funcionalidades dos robôs de serviço;
- Caracterização da arquitetura atual de processamento do robô wiigo;
- Avaliação do desempenho atual dos blocos de processamento do robô wiigo;
- Identificar métodos/técnicas de processamento distribuído que possam ser integradas no robô wiigo;
- Avaliar métodos de comunicação que permitam melhorar a modularidade do wiigo;
- Avaliar os resultados obtidos comparando a performance anterior com a arquitetura implementada;

1.3 Estrutura do documento

Esta dissertação encontra-se organizada em 7 capítulos. No segundo capitulo é apresentado um estudo preliminar sobre a temática em questão. Serão descritas as tecnologias existentes e analisados alguns artigos que consistem no estado da arte do tópico em questão.

De seguida no capitulo 3 serão apresentados conceitos e fundamentos teóricos necessários à compreensão do funcionamento do software usado no trabalho.

O capitulo 4 contem uma descrição da arquitetura atual de software e hardware do wiigo e apresenta novas arquiteturas.

No capitulo 5 vão ser descritas as diferentes fases da implementação das arquiteturas propostas.

Dentro do capitulo 6 vão ser apresentados e comentados os resultados obtidos. Será também proposta uma configuração final de hardware e software.

Por ultimo no capitulo 7 são apresentadas algumas conclusões sobre o trabalho desenvolvido bem como o trabalho futuro a realizar.

Esta página foi intencionalmente deixada em branco.

2
Estado da Arte

Neste capitulo vão ser identificados robôs de serviço disponíveis comercialmente bem como as suas características de hardware e software. De seguida serão apresentados alguns projetos académicos sobre robôs de serviço.

2.1 Robôs de Serviço

Nos últimos anos cada vez mais robôs fazem parte do nosso dia à dia. Ao contrario dos robôs industriais, os robôs de serviço são flexíveis e conseguem reagir a eventos ou mudanças no seu ambiente [3]. A interação robô-humano é uma das características principais destes robôs. Um robô de serviço deve ser capaz de processar informação complexa em tempo real e ao mesmo tempo interagir com os humanos de forma natural [4]. Por esta razão o robô deve estar munido de múltiplos sensores para estudar o ambiente envolvente, entender o comportamento humano e atuar de maneira natural. Ambientes de operação incertos e diversas funções implicam o uso de uma arquitetura flexível para facilitar a reutilização do software e hardware. A capacidade de interação com humanos torna este tipo de robôs uma alternativa para os desafios de produção industrial e para aplicação em tarefas diárias.

2.1.1 AMIGO

O AMIGO, figura 2.1 (*Autonomous Mate for IntelliGent Operations*) [11] é um robô de serviços e cuidados que foi criado na *Eindhoven University of Technology*. Este robô foi criado

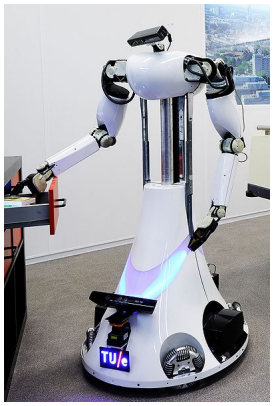


Figura 2.1: *AMIGO*

para assistir pessoas em atividades do dia à dia dentro de espaços fechados ou para assistir nos cuidados de pacientes em ambiente hospitalar. Participou também em diversas competições, *RoboCup@Home*, *Bobbie*, e *RoboEarth* como parte da equipa *Tech United*.

A base da plataforma foi inspirada nos robôs de futebol robótico da equipa *Tech United* sendo esta omnidirecional e holonómica. O seu corpo é extensível em altura e está ligado a dois braços robóticos que oferecem 7 graus de liberdade. Na ponta de cada braço encontra-se também uma garra para poder agarrar e manipular objetos.

No topo do robô está montada uma câmara RGB-D, *Microsoft XBox 360 Kinect* que assenta num sistema *PAN&TILT* baseado em dois servos *Dynamixel* RX-28 que permitem a câmara variar o seu ângulo de *pitch* e *yaw*. Esta câmara é usada para deteção e reconhecimento de pessoas e objetos.

Na base encontra-se um LRF (Laser Range Finder) *Hokuyo UTM-30LX* que é usado principalmente para localização e desvio de obstáculos. Além do LRF encontra-se também outra câmara RGB-D *kinect* que é usada para detetar obstáculos acima e abaixo do *Hokuyo*. O processamento do robô é feito por três mini-PC's cada um contem um processador Intel I5 e 8 GB de RAM. Os PC's estão ligados aos sensores e atuadores por uma rede *EtherCAT*. O sistema é alimentado por quatro baterias de 24V com capacidade de 3.3 Ah.

O software do AMIGO é baseado em ROS e *Orocos*. De modo a controlar o hardware é usado o *Orocos Real-Time Toolkit* que foi desenvolvido para garantir um escalonamento *real-time* entre os processos de navegação percepção e controlo.

2.1.2 PR2

O PR2, figura 2.2 foi criado pela *Willow Garage* com a finalidade de ser uma plataforma robótica para pesquisa e desenvolvimento. Este robô é o sucessor ao PR1 que foi desenvolvido pela universidade de *Standford* [12], PR significa *Personal Robot* (robô pessoal).

Figura 2.2: *PR2*

O software do PR2 é totalmente desenvolvido em ROS, e está disponível livremente em repositórios online.

O robô assenta numa base omnidirecional que contem quatro rodas castor direcionáveis [13]. Para manipulação de objetos tem dois braços robóticos com 7 graus de liberdade e capacidade de carga de 1.8 kg.

As placas de controlo dos motores (*Motor controller boards*) estão ligadas numa rede *EtherCAT*. Na cabeça encontra-se um sensor *Microsoft Kinect*, uma câmara RGB de 5 MP, um sistema *stereo* RGB com *Wide-Angle*, um sistema *stereo* monocromático com *narrow-Angle* e um projetor de textura LED.

Nos ombros e na base do robô encontra-se um módulo com atitude variável que contem um LRF *Hokuyo UTM-30LX* e um sensor IMU *Microstrain 3DM-GX2*. No antebraço temos uma câmara, e nas pinças um acelarómetro e uma matriz de sensores de pressão. Para o processamento o PR2 possui dois computadores (c1 e c2), cada um com um processador *Quad-Core i7* com 24 GB de RAM[14]. O computador 1 (c1) é apelidado de *master* porque é responsável por uma serie de funções chave para a infraestrutura computacional, como por exemplo o controlo dos motores. O computador 2 (c2) é normalmente apelidado de *slave* porque faz *netboot* a partir do c1. O *netboot*, abreviatura para *network booting*, é um processo de *boot* de um computador através da rede em vez de um disco rígido. O PR2 foi desenvolvido com recurso à framework ROS por isso todo o seu software segue uma estrutura baseada em *nodes* para cada subsistema. No entanto para controlo de movimento integra no sistema operativo um ciclo de controlo sob a forma de um processo que corre em tempo real. Para tal são usadas as extensões *RT PREEMPT* para o *kernel* do *Ubuntu* Linux de forma a correr um processo em *hard real time* com um período de 1 ms. Este ciclo faz a atualização e cálculos do controlador bem como a comunicação através de *EtherCAT* com as placas de controlo de motores. Para obter *timestamps* consistentes entre os dois computadores do robô foi usado o *chrony* de modo a sincronizar o tempo dos PC's.

2.1.3 KeJia Service Robot

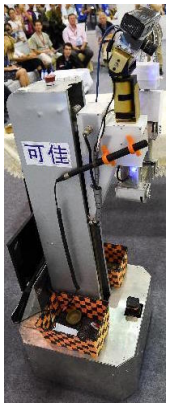


Figura 2.3: *Kejia Robot*

O robô *Kejia*, figura 2.3 foi criado pela Universidade de ciência e tecnologia da China para participar no desafio *RoboCup@Home*. Este robô participa nesta competição desde 2011 e foi primeiro classificado em 2014.

Para além da participação na *RoboCup@Home* outra das motivações foi a construção de um robô que integrasse técnicas de inteligência artificial avançadas como processamento natural de fala, planeamento de tarefas hierárquico e aprendizagem autónoma.

O *Kejia* [15] é construido sob uma base diferencial com duas rodas, tem cerca de 1.60 m e pesa 40 kg. Tem também um sistema de elevação para ajustar a sua altura e um braço robótico com cinco graus de liberdade. Para perceção está equipado com uma câmara RGB-D *Microsoft Kinect*, uma câmara RGB de alta resolução da *PointGrey*, um LRF e um microfone. A informação de profundidade do *kinect* é combinada com a imagem da câmara RGB para detetar e seguir pessoas bem como reconhecimento de objetos. O processamento é feito por um portátil integrado na traseira do robô.

Em termos de software o *Kejia* utiliza ROS como infraestrutura para apoiar a comunicação entre módulos. No modo de operação normal o robô é controlado por comandos de voz que funcionam como entradas no módulo de dialogo robô-humano. As palavras são traduzidas para uma representação interna no módulo de compreensão da fala. Estas representações são passadas ao módulo de planeamento de tarefas que toma as decisões autonomamente. Por sua vez o módulo de planeamento de tarefas gera planeamentos de alto nível para as tarefas do utilizador. A ação a tomar pelo robô é passada ao módulo de planeamento de movimento e por sua vez executada pelo módulo de controlo de hardware. Para tarefas simples ou pré-definidas é usada uma maquina de estados.

Figura 2.4: *Robot-Era Domestic robot*

2.1.4 Robot-Era domestic robot

Este robô doméstico, figura 2.4 faz parte de um projeto europeu chamado *Robot-Era* [16]. O objetivo deste projeto é desenvolver e demonstrar a viabilidade cientifica e técnica bem como a aceitação geral do utilizador final de uma pluralidade de serviços robóticos avançados que trabalharão em conjunto para melhorar a qualidade de vida e a eficiência no cuidado de pessoas idosas.

Este robô integra uma base diferencial da *Metralabs*, nomeadamente a *Scitos-G5*. Para localização e deteção de obstáculos o robô possui na frente um LRF *Sick S300* e um na traseira da plataforma (*Hokuyo URG-05*) bem como um anel de sonares na base. Além destes sensores tem também uma câmara RGB-D *kinect* e uma câmara de alta resolução. Estas câmaras estão localizadas na cabeça do robô que é controlada por motores *stepper* que dão a capacidade de *pan&tilt*. Este robô está também equipado com um braço robótico *Kinova Jaco* com seis graus de liberdade.

O software está organizado em três camadas de abstração, figura 2.5. A primeira camada consiste na interface do utilizador e o *PEIS*. O sistema PEIS mantém o estado de todos os sensores no ambiente de funcionamento do robô, gere a informação de alto nível dos objetos e tarefas e inclui um planeador simbólico que controla os diferentes robôs e sensores na rede. O segundo nível é formado por um conjunto de serviços que encapsulam as funções que correspondem às tarefas do robô. Por sua vez estas funções são implementadas na terceira camada através de *nodes* ROS interligados entre si. Nesta camada foi implementado um *node* supervisor que gere o escalonamento dos serviços e dá *feedback* sobre o progresso das tarefas. O quarto nível é constituído pelos drivers dos sensores e atuadores que por sua vez interagem com o *hardware* físico. A localização e navegação é feita pelo software *Mira/Cognidrive* da *Metralabs*.

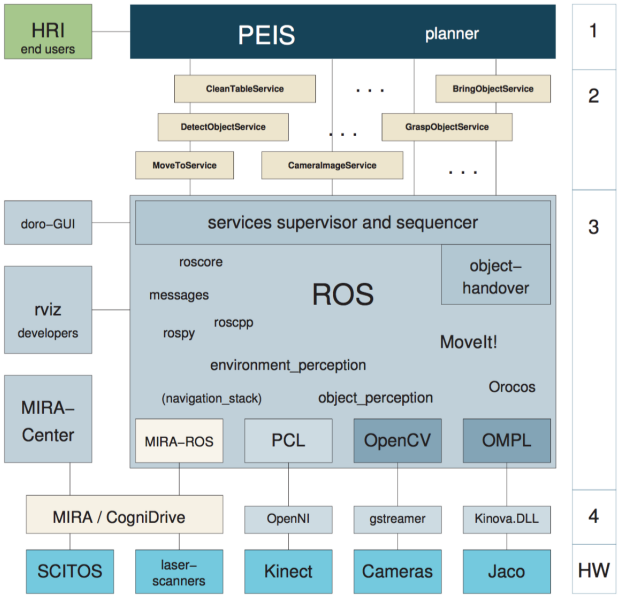


Figura 2.5: *Arquitetura de software do robot-era.*

2.1.5 Fetch & Freight

Figura 2.6: *Fetch & Freight*

Os robôs *Fetch & Freight* foram desenvolvidos pela *Fetch Robotics*. O *Freight* é uma base robótica para aplicações industriais de gestão de inventário e o *Fetch* um manipulador aplicado no robô. O *Fetch* [17] é um manipulador robótico desenhado para ser robusto, ter alta performance bem como baixo custo. Este manipulador foi desenvolvido para aplicações comerciais mas está também disponível para pesquisa e desenvolvimento. O *Freight* pode operar individualmente e tem disponível vários pontos de montagem que lhe permitem realizar um numero elevado de tarefas.

A base móvel com configuração diferencial (*Freight*) contem um scanner laser *SICK TIM571* com um FOV de 220 graus, alcance de 25 m e uma taxa de atualização de 15Hz. Esta

base tem também uma câmara RGB-D que ajuda na navegação e desvio de obstáculos. Cada motor tem uma placa de controlo dedicada baseada num *STM32* que corre o controlador dos motores a uma frequência de 17kHz. Para melhorar a odometria a base está equipada com um sensor inercial. Existem dois barramentos RS-485, um para os motores do braço e outro para a base, torso e cabeça.

O *Fetch* tem uma câmara RGB-D localizada na sua cabeça com um sistema *Pan & Tilt* que funciona a 15Hz. Tem também ao seu dispor um braço robótico com 7 graus de liberdade, capacidade de carga de 6Kg, e um torso elevatório. As duas baterias de *Sealed Lead Acid* fornecem uma autonomia de 8 a 10h de operação. O processamento do *Fetch* é feito por um *Intel I5* com 16 GB de RAM, placa de rede wireless e disco SSD.

O sistema operativo é *Ubuntu Linux* e a framework é o *middleware* ROS. As versões para pesquisa e desenvolvimento usam a *stack* de navegação do ROS que inclui os *planners* e *costmaps* que integram o ROS. além dos componentes que vêem com as *stacks* do ROS foram acrescentados ou melhorados outros módulos. Entre eles temos um *node* que faz uso da câmara localizada na cabeça do *fetch* para se desviar de obstáculos.

2.1.6 SPENCER

Figura 2.7: *SPENCER*

O SPENCER, figura 2.7 é um robô desenvolvido no âmbito de um projeto europeu com a finalidade de assistir, informar e guiar passageiros em aeroportos muito movimentados [18]. Um dos seus principais objetivos é ajudar nos voos de ligação até ao controlo de passaporte de maneira rápida e eficiente. No cenário de demonstração final este robô efetuou a gestão de fluxo de passageiros no aeroporto *Schiphol* em Amesterdão.

Os sensores do *SPENCER* consistem em dois LIDAR *SICK LMS500* com uma cobertura de 360 graus, quatro câmaras RGB-D (duas à frente e duas atrás) e um sistema de câmaras *stereo* posicionadas à altura dos ombros do robô. Para a interação com o utilizador a

plataforma tem um *touchscreen* e um leitor de passes de embarque.

Este robô tem ao todo seis computadores [19], dois *Intel Core i7-3520M* para planeamento e navegação, um *i3-2120* para interação com o utilizador, dois portáteis com *Intel i7-4700MQ* mais uma *Nvidia GTX 765M* para perceção e um sistema embebido *PowerPC* para controlo de baixo nível do robô, figura 2.8. Todos estes computadores comunicam entre si por ligação *gigabit* Ethernet e executam o ROS *Indigo* no *Ubuntu 14.04*.

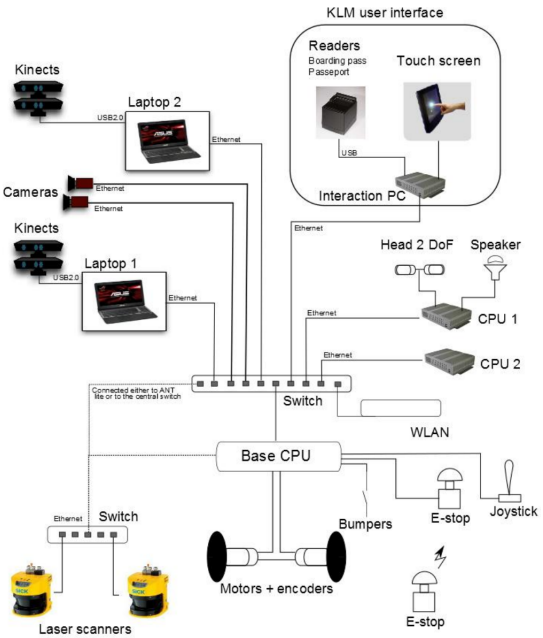


Figura 2.8: *Arquitetura de hardware do SPENCER.*

A arquitetura de software do SPENCER está dividida em diversos módulos e por motivos de flexibilidade o *middleware* escolhido foi o ROS [20]. Esta arquitetura consiste em múltiplos módulos de software:

- Interface com o utilizador;
- Supervisor - este módulo tem a função de orquestrar os diversos módulos do sistema de maneira a ser possível completar uma tarefa;
- Planeador de tarefas - este módulo é responsável por criar um plano para atingir o objetivo atual;
- Planeador de movimento e controlador - este módulo é dividido em planeador global e planeador local;
- Avaliação da situação - este sistema funcionará como intermediário entre os processos de baixo nível que são executados continuamente e os sistemas de alto nível que

tomam as decisões;

- Deteção do porta-voz - este módulo vai identificar a pessoa num grupo que pode ser abordada;
- Deteção de atividade social e analise de relações sociais;
- Deteção e seguimento de pessoas;
- Estimação de postura, da pose da cabeça e analise do tronco;
- Classificação de atributos humanos;
- Deteção e seguimento de grupos;
- Aprendizagem de objetos online;
- Localização e mapeamento socialmente anotado;
- Aprendizagem de comportamentos normativos simples e complexos;

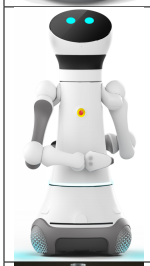
Imagem	Fabricante e Modelo	Aplicação	Dimensão	Peso	Velocidade Máxima	Câmaras	Tração	LIDAR	Outros Sensores
	Panasonic [5] AP-3020A003	Hospitais	630mm(L) 725mm(W) 1386mm(H)	170 Kg	1 m/s	Sim	Diferencial	Sim	RFID Impressão digital
	Savioke[6] Relay	Hotéis	510mm(L) 510mm(W) 920mm(H)	40.8 Kg	0.7 m/s	sim, 2 RGB-D	Diferencial	Sim, 1	Sonares IMU Bumper Sensor
	Fraunhofer[7] Care-O-bot 4	Doméstico	720mm(L) 720mm(W) 1580mm(H)	140 Kg	1.1 m/s	sim, 3 RGB-D	Omnidirecional	Sim, 3	-
	Knightscope[8] K5	Segurança	813mm(L) 914mm(W) 1524mm(H)	136 Kg	1.34 m/s	Sim, 3	Diferencial	Sim, 2	Câmara Termica GPS radar microfone
	Fellow[9] Robots Navii	Retalho	457.2mm(L) 685.8mm(W) 1525mm(H)	90.71 Kg	1 m/s	Sim, 2	Diferencial	Sim, 2	-
	PAL[10] Robotics StockBot	Retalho	500mm(L) 500mm(W) 1900mm(H)	0	1 m/s	Sim, 2	Diferencial	Sim, 1	Antenas RFID

Tabela 2.1: *Comparação de características de vários robôs de serviço.*

3

Fundamentos teóricos

Neste capitulo serão apresentados alguns conceitos e fundamentos necessários para a compreensão do trabalho realizado.

3.1 Arquiteturas de escalonamento para robôs de serviço

O artigo [21] propõem um método de medir a performance de um sistema ROS e sugere uma maneira de dividir os nós por múltiplos computadores.

O autor começa por fazer a caracterização do sistema ROS e usa uma analise de regressão linear para descobrir como a performance de cada nó varia em função dos seus parâmetros.

São abordados essencialmente dois problemas:

- Determinar os parâmetros e as alocações dos nós que maximizam a performance do sistema.
- Dados os parâmetros, determinar as alocações dos nós que minimizam o hardware necessário.

Estes problemas podem ser modelados como uma variante do MMKP (Multiple-Choice Multi-Dimensional Knapsack Problem). Depois será aplicado um *greedy algorithm* para resolver cada problema, sendo que o primeiro usa um gradiente de performance e o segundo é baseado nos requerimentos de CPU de cada um dos nós.

Os cálculos mostram que o algoritmo dá resultados com um desvio de apenas 1% da solução ótima. Um caso de estudo de uma aplicação real em ROS valida o modelo proposto com valores de performance observados com um desvio de 2.5% em relação aos esperados. O código foi desenvolvido em *python* e está disponível no *github*.

O artigo [22] expõem um conjunto de ferramentas que podem ser usadas para avaliar o esforço necessário para atualizar um software que opera num robô de serviço. Estas ferramentas são usadas para fazer uma comparação entre uma arquitetura *peer-to-peer* (ligações diretas entre os diferentes módulos) e uma arquitetura centralizada (*blackboard*) no contexto da robótica móvel. A analise vai considerar o custo de desenvolvimento ao fazer uma atualização bem como o tempo de resposta de ambas as arquiteturas. Os resultados mostram que é mais simples manter um software com a arquitetura *blackboard* do que quando *peer-to-peer* é usado. Estes resultados mostram também que não existe diferença considerável no tempo de resposta ou na performance entre ambas as arquiteturas.

3.2 Middlewares

O *middleware* em robótica é uma camada de abstração que reside entre o sistema operativo (OS) e as aplicações de software, figura 3.1. Foi criado para lidar com a heterogeneidade do hardware, melhorar a qualidade do software da aplicação, simplificar o design do software e reduzir custos de desenvolvimento [23]. O programador necessita apenas de construir a lógica ou algoritmo como um componente que pode ser depois combinado e integrado a componentes existentes.

Num design modular o *middleware* tem um papel importante pois funciona como a "cola"que liga os módulos, além de fornecer também os mecanismos necessários para a comunicação entre eles.

Este formato modular permite aos *developers* trabalhar em diversas partes da aplicação em paralelo, tornando a complexidade gerível e resultando num acoplamento solto dos módulos de software. Isto por sua vez facilita o desenvolvimento do software de maneira a suportar a sua manutenção e reusabilidade.

As principais vantagens do *middleware* são, a modularidade, abstração da arquitetura de hardware, acesso a API's de hardware estandardizadas, independência da plataforma e portabilidade.

No artigo [23] é apresentado um novo *middleware* para robótica chamado de *MIRA*. São também abordados diversos *middlewares* como o *CARMEN*, *Player*, *ROS*, *Urbi*, *LCM*, *MOOS* e *YARP*. De seguida será apresentado um pequeno pequeno resumo de cada um destes *middlewares*.

CARMEN - O *Carnegie Mellon Robot Navigation Toolkit* foi um dos primeiros *mid-*

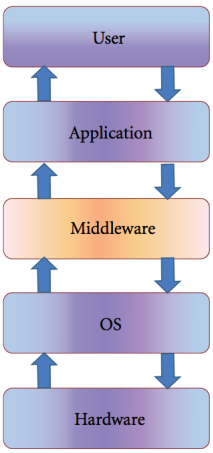


Figura 3.1: *Camadas do middleware.*

dlewares. Suporta diferentes módulos de software que são implementados como processos separados e que comunicam entre si por comunicação inter-processo baseada em *sockets* TCP/IP. Não recebe atualizações desde 2008 por isso está atualmente obsoleto.

Player - Este *middleware* procura fornecer interfaces simples para sensores e atuadores robóticos. Módulos de dispositivos (chamados de*drivers*) correm um processo dentro de um servidor numa *thread* por processo, este *driver* pode ser acedido por clientes via TCP. Utilizando interfaces é possível múltiplos dispositivos serem controlados por um cliente. Os clientes podem ser executados em diferentes plataformas e podem ser implementados em diversas linguagens de programação. No entanto a abordagem dum servidor centralizado expõe o sistema a um único ponto de falha.

Urbi - O *Urbi* está dividido em duas partes. A primeira parte *UObject* fornece uma API em C++ onde componentes como drivers e algoritmos podem ser desenvolvidos e expostos à segunda parte *urbiscript*. O *urbiscript* é uma linguagem de *script* baseada em eventos que é usada para interligar componentes numa aplicação. O *Urbi* permite ao utilizador decidir se quer correr os componentes dentro do mesmo processo ou em processos diferentes ou até em sistemas diferentes de forma transparente durante a execução do programa. Outra vantagem é que para componentes que estão a correr no mesmo processo suporta o modo *0-copying*, o que previne copia desnecessária de dados. No entanto tem as mesmas desvantagens do *Player* e do ROS porque usa uma abordagem centralizada onde os clientes se ligam a um servidor central para executar o código *urbiscript*.

LCM - O *Lightweight Communications and Marshalling* procura simplificar o desenvolvimento de sistemas de troca de mensagens com baixa latência, especialmente para aplicações robóticas em tempo real. Tal como no ROS ele utiliza o modelo *publish/subscribe* para transmitir mensagens entre processos usando um tipo de mensagens independente

da plataforma de funcionamento. O LCM baseia a sua comunicação em *multicast* UDP sem a necessidade de um ponto central de comunicações.

MOOS - O *Mission Oriented Operating Suite* é um *middleware* multi-plataforma para investigação na área da robótica. Ele usa uma rede de comunicação com uma topologia de estrela onde cada cliente tem um único canal de comunicação para um servidor central. Os dados são publicados pelo cliente sob a forma de mensagens, no formato *string* ou *double* e depois guardados numa base de dados. Outros clientes podem depois buscar não só os dados mais recentes mas também o histórico das alterações efetuadas desde a ultima leitura. Devido aos custos adicionais de *parsing* e da necessidade de guardar dados numa base de dados central antes de enviar a um cliente leva a problemas de performance.

YARP - O *Yet Another Robot Platform* procura minimizar o esforço de desenvolvimento de uma estrutura de software facilitando a reutilização de código e modularidade. Ele inclui suporte para comunicação inter-processo num modelo baseado em portos. Os portos podem gerir múltiplas ligações por unidade de dados como entrada ou saída. Cada ligação permite enviar e receber dados a diferentes taxas de transferência e usado diferentes protocolos como o TCP, UPD ou memoria partilhada. Para manter um lista com todos os portos e como se ligar a eles é usado um servidor central.

MIRA - O *Middleware for Robotic Applications* é uma *framework* escrita em C++ que inclui um *middleware* e ferramentas para desenvolver e testar módulos de software. Foca-se na criação fácil de aplicações complexas e dinâmicas, usando estes módulos como *plugins*. O *MIRA* foi desenvolvido essencialmente para robótica, no entanto a sua *framework* não está limitada a esta aplicação pois permite troca de dados entre módulos de software intra e inter-processo. Outra vantagem deste *middleware* é que ao contrario do *ROS* não necessita de um servidor central. O *MIRA* é completamente descentralizado e os processos ligam-se entre si numa arquitetura *peer-to-peer*.

3.3 ROS

O ROS [24] (Robotic Operating System) foi desenvolvido pela *Willow Garage* e pela Universidade de *Standford* como parte do projeto STAIR [25], ele consiste num *middleware* grátis e com código aberto para desenvolvimento de sistemas robóticos.

O ROS não é um sistema operativo no sentido tradicional de gestão de processos e escalonamento, mas em vez disso ele fornece uma camada de comunicações estruturada em cima do sistema operativo hospedeiro.

Este *middleware* fornece uma abstração do hardware, controlo de baixo nível e possui habilidade de passagem de mensagens intra-processo e gestão de pacotes. Fornece também ferramentas e bibliotecas para obter, compilar, escrever e executar código através de vários

computadores. Isto torna a programação dos robôs muito mais simples e deixa os *developers* focarem-se no problema especifico que desejam resolver em vez de necessitarem de implementar diversas partes do sistema que não estão diretamente relacionadas.

A sua principal vantagem é possibilitar a manipulação de dados de sensores sob a forma de um fluxo de dados abstrato chamado tópico, não sendo assim necessário lidar diretamente com drivers de hardware.

3.3.1 Resumo das funcionalidades

Os tópicos ROS são uma implementação do mecanismo *publish-subscribe* onde o *master* funciona como um ponto de entrada conhecido para nomeação e registo. Cada nó ROS anuncia os tópicos que vai subscrever ou publicar ao *master*, figura 3.2. Se existir publicação e subscrição para o mesmo tópico é criada uma ligação direta entre os nós que estão a publicar e subscrever. De seguida vai ser apresentado um glossário de algumas das terminologias do ROS:

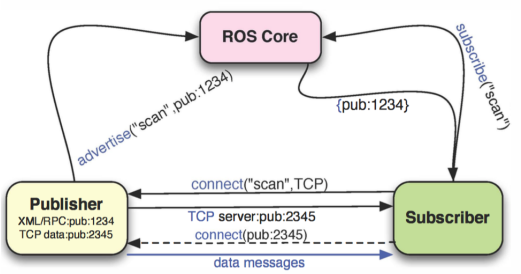


Figura 3.2: *Framework publish/subscribe ROS [26]*

- **Master** - Único no sistema (variável de ambiente ROS_MASTER_URI) fornece serviços de nomeação e registo a todos os nós. Também conhecido por *roscore*.
- **Nome** - Nome dum recurso (nó, tópico, serviço ou parâmetro) dentro do gráfico de computação ROS. O esquema dos nomes é hierárquico e tem vários aspetos em comum com os caminhos do sistema de arquivos UNIX.
- **Host** - Computador dentro da rede ROS, identificado pelo seu endereço IP (variável de ambiente ROS_IP).
- **Nó** - Qualquer processo que utilize a API do cliente ROS, identificado pelo nome do recurso gráfico.
- **Tópico** - Um canal de comunicação unidirecional, assíncrono, usado no mecanismo *publish-subscribe* e identificado pelo seu nome de recurso gráfico.

- **Mensagem** - Os nós comunicam entre si publicando mensagens em tópicos. A mensagem é uma estrutura de dados simples contendo múltiplos campos com diferentes tipos de variáveis.
- **Serviço** - Uma chamada de procedimento remoto síncrona, identificada pelo nome do recurso gráfico.
- **Parâmetros** - Um dicionário compartilhado e multi-variável acessível através de APIs de rede. São utilizados para dados de mudança lenta como argumentos de inicialização.
- **Roslaunch** [27] - O *Roslaunch* é uma ferramenta que permite executar múltiplos nós ROS localmente ou remotamente por SSH. Cada *roslaunch* usa um ou mais ficheiros XML que especificam os parâmetros que vão ser definidos nos nós que vão ser lançados.

3.3.2 Rostopic Statistics

Desde a versão *Indigo* que a possibilidade de ativar estatísticas dos tópicos vem incluída no ROS [28]. Depois de ativo com o comando *rosparam set enable_statistics true* o ROS passará a publicar um tópico com o nome */statistics*.

Os parâmetros medidos para cada *publisher* são, período de mensagens (media, máximo e desvio padrão), idade das mensagens (media, máximo e desvio padrão) baseada no *timestamp*, numero de mensagens perdidas e volume de trafico em bytes.

3.3.3 Rosprofiler

Este pacote contem ferramentas para adquirir e publicar informação e estatísticas sobre os nós que estão em execução e sobre as maquinas que os estão a executar [29]. O *Rosprofiler* funciona em conjunto com as ferramentas de estatísticas disponíveis no ROS.

Neste pacote estão incluídos dois nós, o *rosprofiler* e o *rosgrapher*. Num sistema de processamento distribuído com o ROS a correr em várias maquinas estes nós devem ser corridos em cada um dos PC's.

O *rosprofiler* publica dois nós, o */host_statistics* que fornece informações sobre a maquina e os seus recursos disponíveis e o */node_statistics* que fornece informação sobre cada um dos nós a ser executados. Cada valor destes tópicos é obtido com base em 20 amostras.

As informações mais relevantes fornecidas pelo */host_statistics* são, o *timestamp* da mensagem, *hostname* da máquina, o tempo de inicio e fim da amostra, a percentagem (media, máximo e desvio padrão) de cada um dos núcleos do CPU da máquina e a quantidade (media, máximo e desvio padrão) da memoria RAM livre.

Os parâmetros mais importantes publicados pelo */node_statistics* são o *timestamp* da mensagem, *hostname* da máquina, nome do nó, tempo de inicio e fim da amostra, numero de threads, percentagem (media, máximo e desvio padrão) da carga total do processador e a quantidade (media, máximo e desvio padrão) da memoria RAM real e virtual ocupada.

O *rosgrapher* publica informação obtida do *master* e de cada um dos nós sobre o estado do sistema ROS. Este nó só necessita de ser executado uma vez e publica o tópico */topology*.

3.3.4 Advanced ROS Network Introspection

O *Advanced ROS Network Introspection* (ARNI) [30] estende as funcionalidades do */statistics* e completa os dados adquiridos com aquisição de meta-dados dos *hosts*, nós e ligações de uma maneira descentralizada. Estes dados são obtidos através de um nó extra que corre em cada um dos computadores.

As medidas ou meta-dados podem ser comparadas com valores de referencia usando o *monitoring_node*. As estatísticas obtidas permitem executar contra-medidas quando um desvio da referencia é detetado, para resolver a falha ou trazer o sistema para um estado seguro [31].

Como são apenas necessários alguns *floats* de meta-dados para cada um dos tópicos o trafico de rede adicional consiste apenas uma pequena fração do trafico total não afetando significativamente a performance do sistema.

3.3.5 Multimaster fkie

Esta extensão fornece uma maneira não invasiva de gerir múltiplos *masters* numa rede unificada [32]. Adicionalmente este pacote fornece um GUI para monitorizar e gerir nós, tópicos, serviços, parâmetros e ficheiros *launch* numa rede ROS. As funcionalidades chave do *Multimaster_fkie* são:

- Todos os componentes ROS são executados sem serem modificados, logo não necessitam de configurações adicionais;
- O comportamento dos nós ROS não é mudado
- Os robôs ou estações de controlo podem ser desenhados como redes independentes com um único *master* e conectados conforme necessário;
- O *Multimaster_fkie* pode ser ligado ou desligado sem haver necessidade de reiniciar os nós que estão a ser executados;
- Cada ROS *master* retém uma visão consistente da rede local mesmo que partes da rede sejam removidas;

- Os ficheiros *launch* e configurações existentes podem ser usados sem modificações;

Este pacote foi desenvolvido em *python* e está disponível no *GitHub* [33]. A documentação da API e dos parâmetros ROS suportados está disponível na wiki do ROS [34].

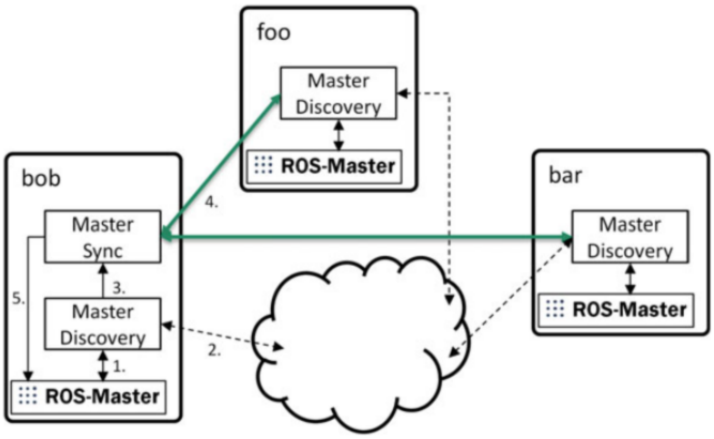


Figura 3.3: *Visão conceptual para descoberta e sincronização*

Cada nó *master_discovery* liga-se ao seu ROS *master* local e monitoriza continuamente por mudanças (1.). As mudanças são publicadas através da rede em *multicast* ou *unicast* (2.). As mudanças recebidas dos *master's* remotos são publicadas nos tópicos ROS locais (3.).

O nó *master_sync* liga-se a todos os nós *master_discovery* (4.) faz o pedido do estado atual do ROS e regista os tópicos e serviços remotos (5.).

Para simplificar a gestão do sistema ROS *multi-master* vem incluído o *node_manager* que é um GUI para monitorizar e gerir nós, tópicos, serviços, parâmetros e ficheiros *launch*. Este programa funciona também com sistemas ROS com um único *master*.

3.3.6 Multi-Threaded Spinning

Para chamar *callbacks* de múltiplas *threads* o *roscpp* duas opções, o *MultiThreadedSpinner* e o *AsyncSpinner* [35].

O *MultiThreadedSpinner* é bloqueante e similar ao *ros::spin()*. Permite especificar o numero de *threads* como parâmetro, sendo que por defeito este numero é igual ao numero de núcleos do processador.

Por sua vez o *AsyncSpinner* é não bloqueante tendo funções para inicio e fim e para automaticamente quando é destruído.

3.3.7 Algoritmo de *Nagle*

Para se obter alta eficiência de rede é necessário enviar pacotes TCP cheios. O algoritmo de *Nagle* diz que caso não se tenham dados suficientes para encher um pacote TCP inteiro e já existem dados *unacknowledged* em transmissão então espera-se até a aplicação disponibilizar mais dados para encher o pacote TCP ou até a outra ponta enviar o *acknowledge* de todos os dados.

Listing 3.1: *Algoritmo de Nagle*

```
When the application produces data to send
        if both the available data and the window >= MSS(
           maximum segment size)
                send a full segment
        else
                if there is unACKed data in flight
                        buffer the new data until an ACK
                           arrives
                else
                        send all the new data now
```

3.3.8 Transport Hints

A classe *ros::TransportHints* [36] é usada para o utilizador configurar como quer que a camada de transporte se comporte para um determinado tópico.

Permite especificar múltiplas opções de transporte diferentes como por exemplo:

- *tcp()* - Especifica o transporte de dados como sendo por TCP;
- *maxDatagramSize()* - Tamanho máximo do datagrama UDP;
- *udp()* - Especifica o transporte de dados como sendo por UDP;
- *tcpNoDelay()* - Caso esteja a ser usado TCP é configurada uma ligação de baixa latência (desativando o algoritmo de *Nagle* [37]) com o custo de aumento da largura de banda;

3.4 ROS2

Para resolver as limitações do ROS em sistemas de tempo real e atender às necessidades de uma comunidade em grande crescimento o ROS irá sofrer um *upgrade* significativo para ROS2.

O ROS2 vai incluir os seguintes casos de uso: sistemas em tempo real, sistemas embebidos, redes não ideais e multi-plataforma. Para satisfazer estes requerimentos a versão existente do ROS (ROS1) vai ser reconstruida para melhorar as API's e integrar novas tecnologias como o *Data Distribuition Service* (DDS), *Zero-Conf*, *Protocol Buffers*, *ZeroMQ*, *Redis*, e *WebSockets*.

A camada de transporte do ROS1 vai ser substituída pelo DDS que é um sistema de comunicação em tempo real padrão na industria e um *middleware* ponto a ponto. O DDS fornece um transporte *publish-subscribe* similar ao do ROS1 mas mais robusto e confiável.

No artigo [38] o autor demonstra uma prova de conceito para uma abordagem DDS do ROS. Será clarificada a performance do transporte de dados para o ROS1 e ROS2 em várias situações. Performance significa, características de latência, taxas de transferência, e modularidade. Dependendo dos vendedores do DDS e configurações foi explorado e avaliado o potencial e limitações de vários aspetos, latências, largura de banda, numero de *threads* e consumo de memória. A partir dos resultados experimentais são feitas diretrizes e foram propostas soluções para resolver as limitações atuais deste *middleware*.

No lado esquerdo da figura 3.4 está representada a implementação do ROS1 que inclui o protocolo de comunicação *TCPROS/UDPROS*. Esta comunicação requer um processo *master* que por sua vez tem de ser o mesmo para todo o sistema distribuído. No lado direito da figura temos o ROS2 que é construido sob o DDS e contem uma camada de abstração do DDS. Devido a esta camada os utilizadores não necessitam de ter conhecimento da API do DDS. Esta camada permite então configurações de alto nível e otimiza a utilização do DDS. Devido ao uso do DDS deixa de ser necessário haver um processo *master*, isto é um ponto importante no melhoramento da tolerância de falhas.

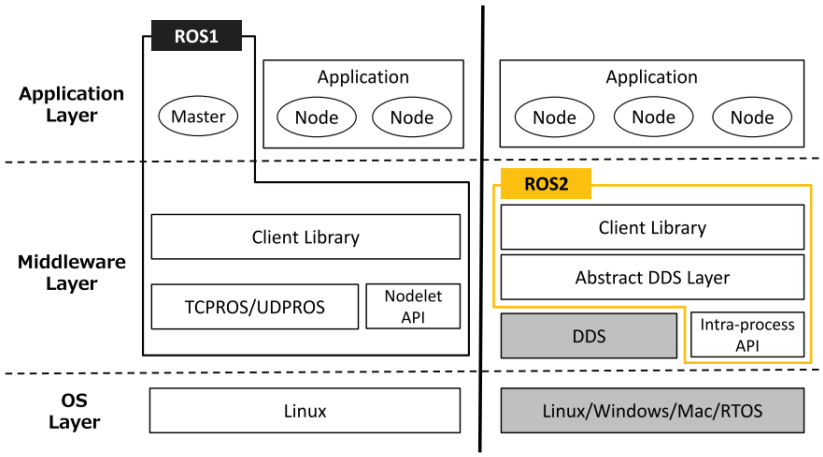


Figura 3.4: *Arquitetura ROS1/ROS2 [38].*

3.5 Data Distribution Service

O Data Distribution Service (DDS) é um middleware de comunicações cujo conceito foi estandardizado e é atualmente gerido pelo *Object Management Group (OMG)*[39].

O DDS especifica uma API (application programming interface) estandardizada pela qual uma aplicação distribuída pode usar um *DataCentric Publish-Subscribe* (DCPS) como mecanismo de comunicação.

Uma comunicação *DataCentric* permite especificar vários parâmetros como a frequência de publicação, frequência de subscrição, por quanto tempo são os dados válidos, entre outros. Estes parâmetros de qualidade de serviço QoS permitem aos *developers* construir uma aplicação distribuída baseada nos requisitos de cada conjunto de dados.

Dado que o standard do DDS contem uma API bem definida e com portabilidade entre plataformas, é possível que aplicações desenvolvidas usando diferentes implementações DDS de diferentes vendedores comuniquem e troquem dados entre si. Como o DDS é implementado como uma infraestrutura ele pode ser usado como interface de comunicação entre quaisquer aplicações de software. Tal como se pode ver na figura 3.5, o DDS fornece uma camada que permite diferentes tipos de aplicações comunicarem entre si.

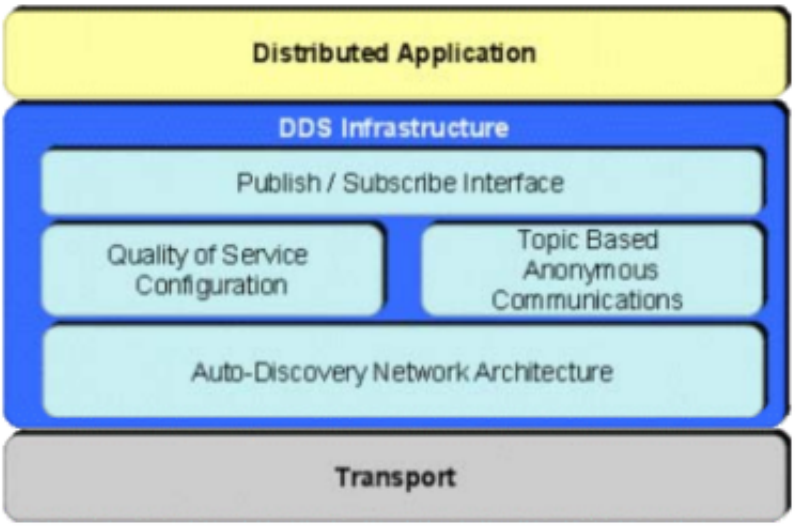


Figura 3.5: *Infraestrutura do DDS*

O modelo *publish-subscribe* do DDS liga produtores de informação (*publishers*) com consumidores de informação (*subscribers*). A aplicação distribuída global é composta por processos chamados de *participants*, sendo cada um executado num espaço de endereçamento diferente e possivelmente em diferentes computadores. Um *participant* pode simultaneamente subscrever e publicar *typed data-streams* fluxos de dados identificados por nomes e chamados de tópicos.

O DDS define a relação de comunicações entre *publishers* e *subscribers*. Estas comunicações são desacopladas no espaço (os nós pode estar localizados em qualquer lado) no tempo (a entrega pode ser feita imediatamente após a publicação ou mais tarde) e fluxo (a entrega pode ser feita eficientemente com largura de banda controlada).

Os parâmetros de QoS definem o grau de acoplamento entre os participantes (*participants*), propriedades do modelo geral e dos tópicos. De modo a aumentar a escalabilidade do sistema os tópicos podem conter múltiplos canais de dados independentes identificados por *chaves*. Isto permite aos nós subscrever muitos fluxos de dados semelhantes com uma única subscrição.

O DDS foi fundamentalmente desenhado para funcionar sob meios de transporte pouco eficientes como o UDP ou redes *wireless*.

Em suma DDS é uma tecnologia estandardizada de forma ubíqua, interoperável, segura, independente da plataforma, eficiente no tempo e no espaço, para partilha de dados através de dispositivos interligados por rede.

Principais funcionalidades do DDS:

- Baseado no paradigma de comunicação *publish-subscribe*
- Abstração de alto nível para partilha de dados;
- Poliglota e independente da plataforma;
- *Peer-To-Peer* por natureza, funciona eficientemente em ligações com pouca largura de banda com latência mínima;
- Eficiente no tempo e no espaço;
- Filtro temporal e de conteúdo;
- *Querries*;
- Descoberta dinâmica de *publishers*, *subscribers* e tópicos;
- Mais de 20 definições de *QoS*;
- *Real time* ou próximo de *Real time*;
- Entrega de dados determinística;
- Alta performance e escalável.

3.5.1 OpenSplice

O DDS da *OpenSplice* é um produto desenvolvido pela *PrismTech* que inclui um *middleware* que implementa o modelo *publish subscribe*, uma API e um protocolo para interoperabilidade. Estão disponíveis API's para *C, C++, Java e C#* . Benefícios do *OpenSplice*:

- **Centrado nos Dados** - Permite que as aplicações sejam construidas à volta de um modelo de dados extensivo, segurança ponto-a-ponto e eficiência.
- **Tempo Real** - A informação certa é entregue no local correto na altura desejada.
- **Confiável** - Garante disponibilidade, fiabilidade, segurança e integridade no caso de falhas de hardware ou software.
- **Alta Performance** - Consegue distribuir altos volumes de dados com baixas latências.
- **Escalável** - Compatível com sistemas simples a sistemas de larga escala, bem como sensores inteligentes e servidores de alta performance.
- **Seguro** - Permite manter confidencialidade, integridade e autenticidade dos dados trocados.

3.6 Chrony

O *chrony* [40] é uma implementação do *Network time protocol* (NTP). Este software consegue sincronizar o relógio do sistema com servidores NTP, relógios de referencia e por definição manual. Foi feito para funcionar em condições variadas, como por exemplo ligações de rede intermitentes, redes congestionadas e sistemas que não correm continuamente. A precisão normal entre duas maquinas sincronizadas pela Internet é de alguns milissegundos, em LAN é de décimas de micro segundos e com relógio por hardware pode atingir precisão abaixo de micro segundos.

Com a instalação do *chrony* veem incluídos dois programas, *chronyd* que é o *deamon* que inicia com o *boot*, e o *chronyc* que é uma interface na linha de comandos que pode ser usada para monitorizar a performance do *chronyd* e alterar parâmetros do mesmo.

3.7 GNU Project Debugger (GDB)

O GDB [41] permite ao utilizador inspecionar o que está a acontecer dentro de um programa quando ele está a ser executado ou o que um programa estava a fazer no momento em que foi abaixo.

As principais funcionalidades do GDB são:

- Iniciar o programa, especificando parâmetros que afetem o seu comportamento;
- Fazer o programa parar em situações especificas;
- Examinar o que aconteceu quando o programa parou;
- Mudar parâmetros no programa.

O GDB suporta programas escritos em *Ada, C, C++, Objective-C, Pascal*, entre outros. Estes programas podem ser executados nativamente na maquina local ou remotamente.

3.8 Protocolo CAN

O protocolo *Controller Area Network* Controller Area Network (CAN) é um protocolo de comunicação serie robusto que é comummente usado em automóveis e industrias devido à sua resistência a efeitos eletromagnéticos e de temperatura. O seu sistema de barramento tem capacidades multi-mestre, isto é, vários nós podem pedir acesso ao meio de transmissão em simultâneo. Este protocolo comporta também o conceito de multicast, isto é, permite que uma mensagem seja transmitida a um conjunto de recetores simultaneamente.

Numa rede CAN todos os nós tem acesso ao mesmo barramento e cada mensagem contem um identificador único e um nível de prioridade. Com base no identificador recebido cada nó decide se deve ou não processar a mensagem. Se dois nós tentarem usar o barramento simultaneamente a mensagem com prioridade mais alta ganha acesso ao mesmo enquanto que a de prioridade mais baixa tem de abortar. As comunicações *CAN* são aplicáveis em redes que tenham volumes de dados relativamente baixos (1Mbps ou menos). Assim, cada mensagem CAN pode conter um máximo de 8 bytes de informação útil, sendo no entanto possível transmitir blocos maiores de dados recorrendo a segmentação.

Propriedades do *CAN* [42]:

- Padrão ISO;
- Prioridade de mensagens;
- Considerável imunidade ao ruído;
- Garantia de tempos de latência;
- Flexibilidade de Configuração;
- Capacidade *multicast* com sincronização de tempo;

- Consistência de dados;
- *Multimaster*;
- Deteção de erros e sinalização;
- Retransmissão automática de mensagens corrompidas quando o barramento estiver livre;
- Distinção entre erros temporários e falhas permanentes de nós;
- Taxas de transferência até 1 Mbit/s;
- *Hardware standard*.

Benefícios de um sistema de rede CAN em relação a um sistema centralizado [43]:

- Menos fios no sistema, o que torna a cablagem mais simples e barata;
- Necessidade de ligações curtas a sensores sensíveis a ruído elétrico antes dos sinais seres convertidos em mensagens digitais imunes a ruído;
- Flexibilidade, as unidades funcionais podem ser adicionadas ou removidas de forma simples;
- Facilidade de manutenção, pois a eletrónica de varias unidades pode ser idêntica;
- Cada unidade pode ser desenvolvida e testada individualmente de acordo com os requisitos do sistema;
- Partilha de dados entre as várias unidades pode eliminar redundância de informação.

Esta página foi intencionalmente deixada em branco.

4

Case Study : WiiGo

Nesta capitulo será detalhada a arquitetura atual de hardware e software do WiiGo. Com base na analise da arquitetura atual iremos endereçar algumas linhas de trabalho com o objetivo de permitir uma maior modularidade do wiigo.

4.1 Modo de funcionamento

Sendo um carrinho de compras autónomo, projetado para acompanhar pessoas com ou sem mobilidade reduzida em superfícies comerciais, o Wiigo enquadra-se na categoria de robô de serviço [44]. Graças aos seus sensores (câmaras RGB-D e LRF) o robô deteta e identifica o utilizador em menos de dois segundos sendo depois apenas necessário carregar no botão iniciar e o wiigo começará a seguir-lo. Estes sensores permitem também ao robô identificar e evitar obstáculos ao longo do seu caminho. Na figura 4.1 é possível visualizar a imagem RGB (janela superior esquerda) e a imagem de profundidade (janela inferior esquerda).

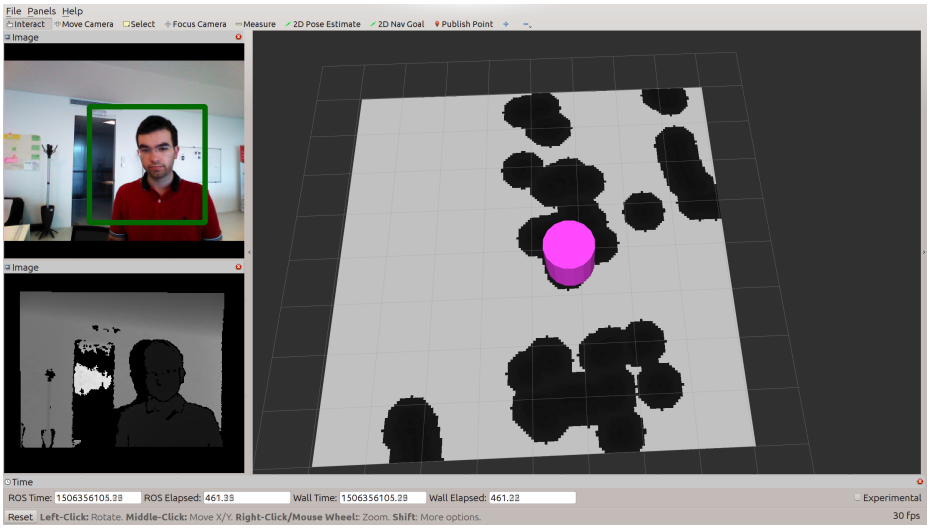


Figura 4.1: *Interface gráfica do rviz com a deteção do utilizador na imagem RGB, imagem de profundidade, costmap e posição do utilizador (cilindro rosa) no mapa.*

4.2 Arquitetura de Hardware

Nesta secção iremos detalhar a arquitetura de hardware do *WiiGo*, figura 4.2. Nesta figura encontra-se um diagrama onde estão representadas as ligações entre os vários sensores e o computador principal (MCU), bem como as linhas de alimentação de todo o sistema. As ligações a vermelho correspondem à alimentação e as ligações a preto às de sinal.

Lista completa de hardware do wiigo:

- 1x PC com um Intel I5-4460 com 8GB de RAM;
- 3x Orbbec Astra RGB-D câmara;
- 1x Hokuyo UST-10LX;
- 1x Roboteq SDC2130;
  - 2x HEDM-5500 Encoder;
  - 2x Bosch F 006 B20 093;
- Sensor board baseada num STM32F072x8;
  - 8x SRF05 Sonar;
  - 1x Push Button;
  - 1x Emergency Button A01ES+A0154B;
  - 1x Current sensor ACS711EX;

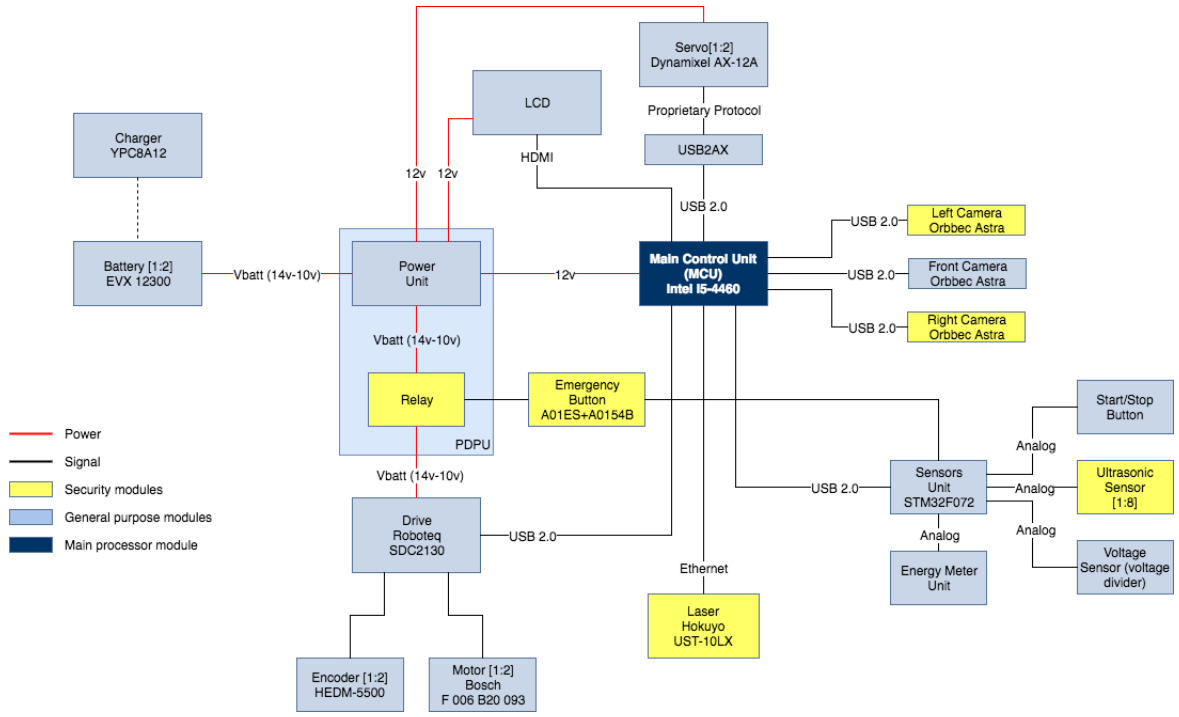


Figura 4.2: *Arquitetura de Hardware do WiiGo*

- 2x Dynamixel AX-12A;
- 1x Adaptador USB2AX;
- 1x Custom power unit (PDPU);
- 2x Bateria EVX 12300;
- 1x 10.1 Inch 1280x800 IPS LCD;

4.3 Arquitetura Software

O diagrama da figura 4.3 apresenta uma arquitetura simplificada de software do *wiigo*.

Este software está dividido em diversos módulos, *Vision* (Visão), *Sensors* (sensores), *Behaviour* (comportamento), *Executing* (execução) e *Control system* (sistema de controlo).

O modulo de visão adquire e processa a imagem RGB e a informação de profundidade de cada uma das câmaras. Alem disso efetua também a deteção de pessoas, redução de falsos positivos e tarefas de identificação.

O modulo de sensores é responsável pela aquisição dos dados laser e sonares para verificar a existência de obstáculos. O *Behaviour* inclui os módulos de *tracking* do utilizador detetado e planeamento de trajetória. Para o planeamento da trajetória é usado um método

de localização baseado na odometria (que resulta da leitura dos *encoders*). De maneira a evitar obter uma trajetória evitando obstáculos detetados é usado o algoritmo de procura A*. Para obter a localização do robô no mundo é feita a fusão da visão com os dados do laser e sonares. O modulo de execução recebe a trajetória gerada e as deteções dos obstáculos e de acordo com a informação providenciada pelo *Behaviour* envia comandos para o modulo de controlo que move o robô num caminho pre-definido.

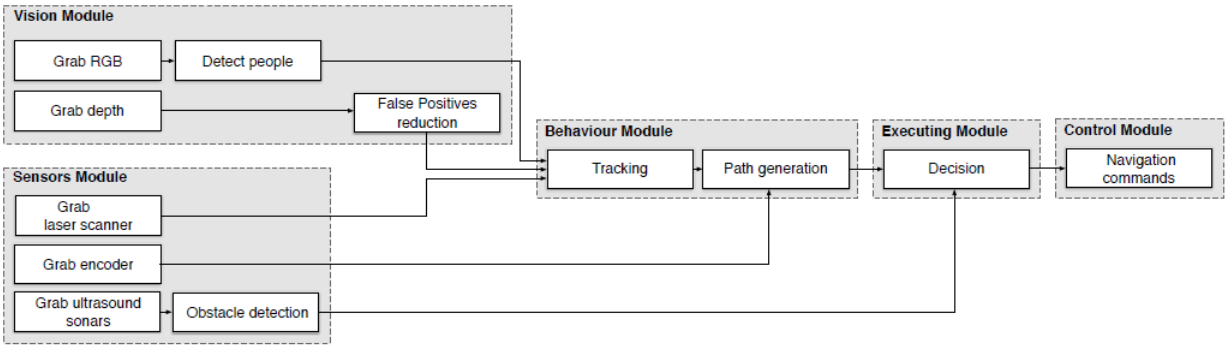


Figura 4.3: *Arquitetura de software de alto nível do WiiGo.*

4.4 Analise de performance do software

Nesta secção vai ser descrito o processo de aquisição dos resultados usados para a medição da performance do software do *WiiGo*.

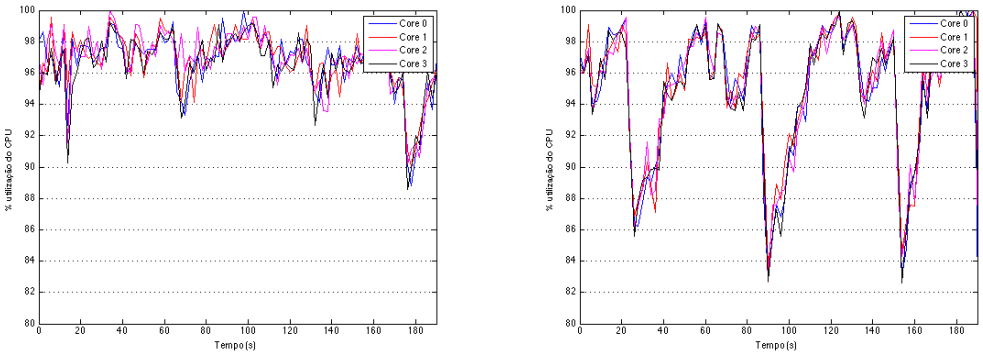
Após a inicialização do software, o robô começa a efetuar deteções de pessoas até que seja premido o botão para iniciar a operação. Enquanto o wiigo se encontra parado neste estado pode-se afirmar que está em modo de espera (*idle*).

Carregando no botão é feita uma identificação do utilizador mudando o robô para o modo de identificação. Se a identificação tiver sido feita com sucesso o wiigo irá começar a seguir o utilizador identificado. Quando o robô se encontra a seguir o utilizador considera-se que está em modo de seguimento (*tracking*).

No evento de perder o utilizador o robô entra automaticamente em modo perdido e tenta fazer a recuperação do seguimento do utilizador.

De modo a analisar e medir performance global do sistema vai ser registada a percentagem de utilização de cada um dos núcleos do processador. Foi medida também a influencia individual de cada um dos *nodes* ROS na utilização do processador. Estes dois resultados foram obtidos durante um intervalo de tempo de 120 segundos utilizando o *rosprofiler*.

Depois vão ser apresentadas as frequências médias de alguns tópicos ROS e finalmente a percentagem de mensagens ROS perdidas. Estas duas ultimas foram registadas num intervalo de 60 segundos utilizando o *rostopic statistics*.



(a) Modo de seguimento I5-4460

(b) Modo de espera I5-4460

Figura 4.4: *Gráfico de utilização de CPU vs o tempo dos quatro núcleos da máquina*

Os resultados vão ser apresentados sob a forma de gráficos e tabelas que servirão de base para que depois seja possível chegar a conclusões.

Os seguintes testes foram realizados apenas no computador central do Wiigo (*I5-4460*), sem recurso a processamento distribuído.

4.4.1 Utilização do CPU do sistema

A utilização do CPU foi obtida com recurso ao *rosprofiler*. Tal como foi descrito anteriormente nos fundamentos teóricos este pacote ROS tem a capacidade de medir a utilização do CPU do sistema e de cada um dos nós individuais.

O gráfico da figura 4.4(a) mostra a utilização total de cada um dos núcleos do CPU em relação ao tempo em modo de *tracking*.

Quando o robô está em espera continuam a ser efetuadas deteções, na figura 4.4(b) é apresentado o gráfico de utilização do processador em *idle*.

Nestes gráficos é possível ver que em ambos os modos a utilização do CPU anda relativamente próximo de 100%.

Outra das funcionalidades do *rosprofiler* é a medição da memoria RAM livre e ocupada. Com base nos resultados obtidos verificou-se que o que a memoria RAM ocupada no sistema em modo de seguimento tem um valor médio de **3208 MB** e um valor máximo de **3417 MB**.

4.4.2 Utilização do CPU pelo *nodes* ROS

O *rosprofiler* permite obter também a percentagem de CPU que um *node* utiliza durante um intervalo de tempo. Na Figura 4.5 são apresentados os seis que mais CPU recorrem para processamento, em modo de seguimento.

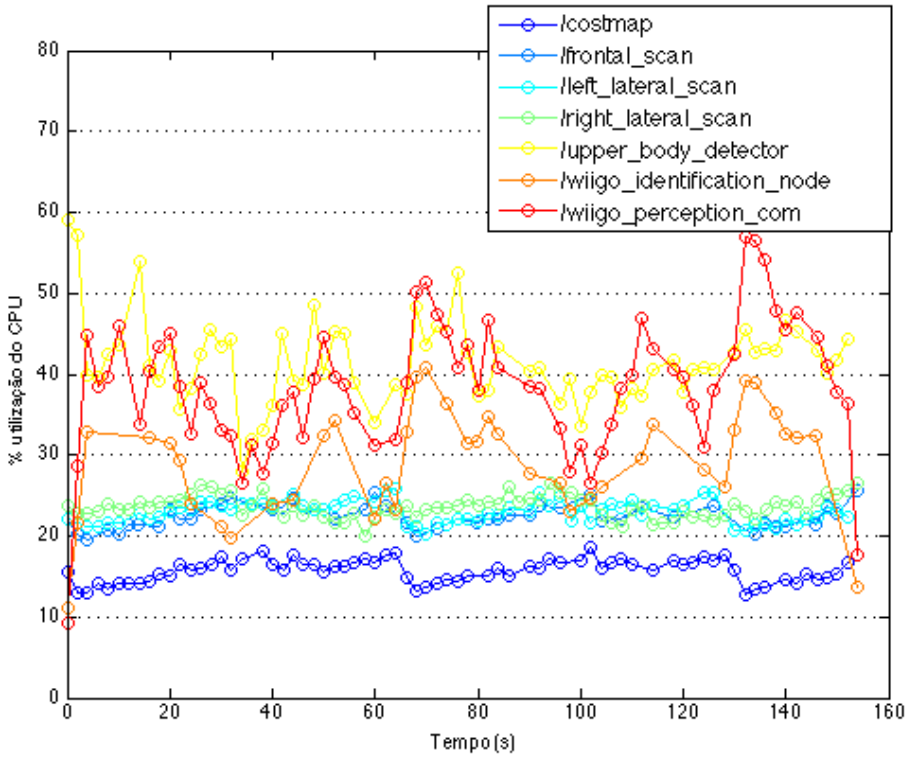


Figura 4.5: *Gráfico de utilização de CPU em relação ao tempo dos seis nodes ROS computacionalmente mais intensivos, em modo de seguimento.*

Estes resultados foram depois processados e calculou-se o valor médio e desvio padrão da utilização do processador por cada um destes nós, tabela 4.1.

Uma outra analise da performance do sistema consiste em medir algumas estatísticas dos tópicos. Foi decidido registar a frequência média e número de mensagens perdidas dos tópicos mais importantes. Quando o sistema está com mais carga e não consegue executar todos os pedidos perde mensagens, esse é um dos motivos para frequência dos tópicos diminuir. Outro dos motivos são os atrasos na rede que levam a que algumas mensagens tenham de ser descartadas por serem muito antigas.

Embora se tenham adquirido dados de todos os tópicos do sistema, foram apenas escolhidos alguns da perceção e o *costmap* porque são os mais computacionalmente exigentes. Por esta razão sofrem uma variação maior do que os restantes.

O gráfico apresentado na Figura 4.6 apresenta uma comparação das frequências de alguns dos tópicos mais importantes, nomeadamente os que fazem parte do *perception* e do *costmap*.

As mensagens perdidas dependem do tamanho definido para as filas de mensagens nos *publishers* e *subscribers*, da frequência de publicação e do tempo que demora a processar a

Tabela 4.1: *Percentagem media utilização pelos nós e desvio padrão*

	% média de utilização do CPU	σ
/costmap	15.723	5.113
/frontal_scan	22.422	5.850
/left_lateral_scan	22.989	5.763
/right_lateral_scan	23.627	6.150
/upper_body_detector	41.559	11.329
/wiigo_identification_node	29.115	10.352
/wiigo_perception_com	38.398	13.161

mensagem. Um numero elevado de mensagens perdidas é geralmente mau porque significa que que existe um atraso no processamento. A percentagem de mensagens perdidas é calculada com a seguinte formula, $\%lost = \frac{perdidas}{entregues} * 100$.

Na tabela 4.2 encontra-se o numero das mensagens perdidas sob a forma de percentagem.

Tabela 4.2: *Percentagem media e desvio padrão das mensagens perdidas nos tópicos.*

	% média de mensagens perdidas	σ
/person_stopped	37.509	0.334
/scan	19.341	7.665
/vision/ground_plane	33.892	6.602
/wiigo_active_camera/tilt_current_pos	22.847	16.958
/wiigo_odometry/odom	33.252	0.376
/wiigo_velocity_control/cmd_vel	18.888	3.329

Estes valores obtidos vão ser usados como referencia de comparação (*baseline*) entre as outras soluções com múltiplos computadores.

4.5 Definição do problema

Observando o WiiGo ao pormenor é possível ver que as três câmaras, os servos e o monitor se encontram localizados na sua cabeça. De modo a diminuir as dimensões de transporte a parte superior é separável da base, isto significa que existem conectores para ligar todos estes sensores ao computador principal. Dado que o computador central do robô está montado na sua base, é necessário passar múltiplos cabos de cima para baixo para efetuar as ligações. Como os cabos destes sensores são curtos é necessário usar extensões USB, isto aumenta o tempo de montagem e origina mais ruído eletromagnético.

Os sonares e os botões estão ligados a um micro-controlador que por sua vez está ligado

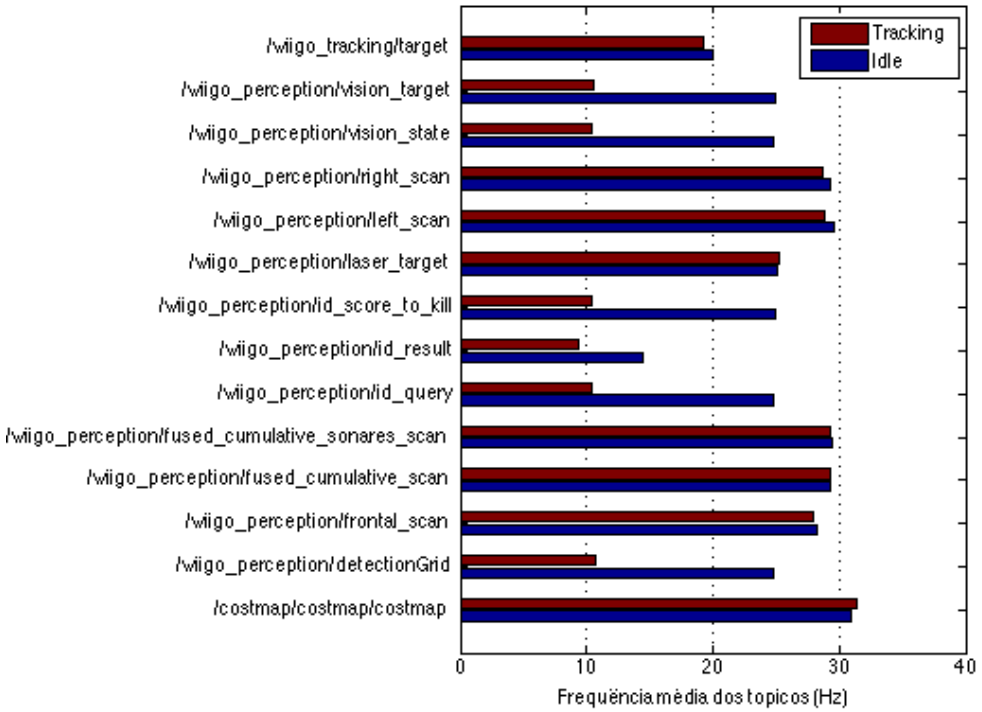


Figura 4.6: *Comparação das frequências em modo de seguimento com modo parado.*

por USB ao computador central. Os servos que controlam a câmara central e o controlador *roboteq* dos motores DC estão também ligados por USB.

Este elevado numero de USB's em uso (6 sem contar com o adaptador wi-fi) ocupa muitas portas do computador central e limita a largura de banda disponível para as câmaras que tem de obrigatoriamente usar este barramento.

No futuro poderão ser adicionadas novas funcionalidades ao software do wiigo, como por exemplo localização. Por este motivo poderá ser necessário mais poder de processamento pois atualmente o CPU existente já se encontra perto do seu limite. De modo a colmatar esta necessidade poderá ser adicionado um segundo computador.

Deverá ser implementada uma arquitetura de software que otimize o desempenho desta nova configuração com dois computadores.

5

Arquitetura proposta/Implementação

Neste capitulo vão ser descritas as diferentes fases da implementação das arquiteturas propostas.

5.1 Hardware

Para a arquitetura de hardware utilizou-se a solução já existente e dividiu-se a *Main Control Unit* (MCU) em duas partes.

A primeira parte *Top Control Unit* (TCU) vai ser responsável pela perceção, fica localizada na cabeça do robô e consiste num computador que vai estar ligado às três câmaras, ao *USB2AX* e ao LCD. Este PC vai ser responsável pelo processamento mais intensivo, que corresponde aos nós contidos no *launcher* do *preception_astra*. A segunda parte do sistema *Bottom Control Unit* vai ser responsável pelo resto do processamento menos intensivo.

De modo a ligar os dois computadores e o LIDAR vai ser adicionado um *switch* ao sistema. Isto permite que estes três dispositivos fiquem na mesma rede.

A comunicação da *sensor board* e o *roboteq* com o computador principal irá ser substituída por uma rede CAN. Esta rede poderá implementar um protocolo CAN de alto nível como por exemplo o *CANOpen* que seja suportado pelo *roboteq*. O uso deste protocolo simplificará o código e aumentará a robustez do sistema. A *sensor board* deverá ser modificada de maneira a integrar um *transciver* CAN para adaptar o sinal. Finalmente para ligar esta rede ao computador é necessário um adaptador CAN-USB, outra alternativa seria escolher uma motherboard que já integrasse entrada CAN.

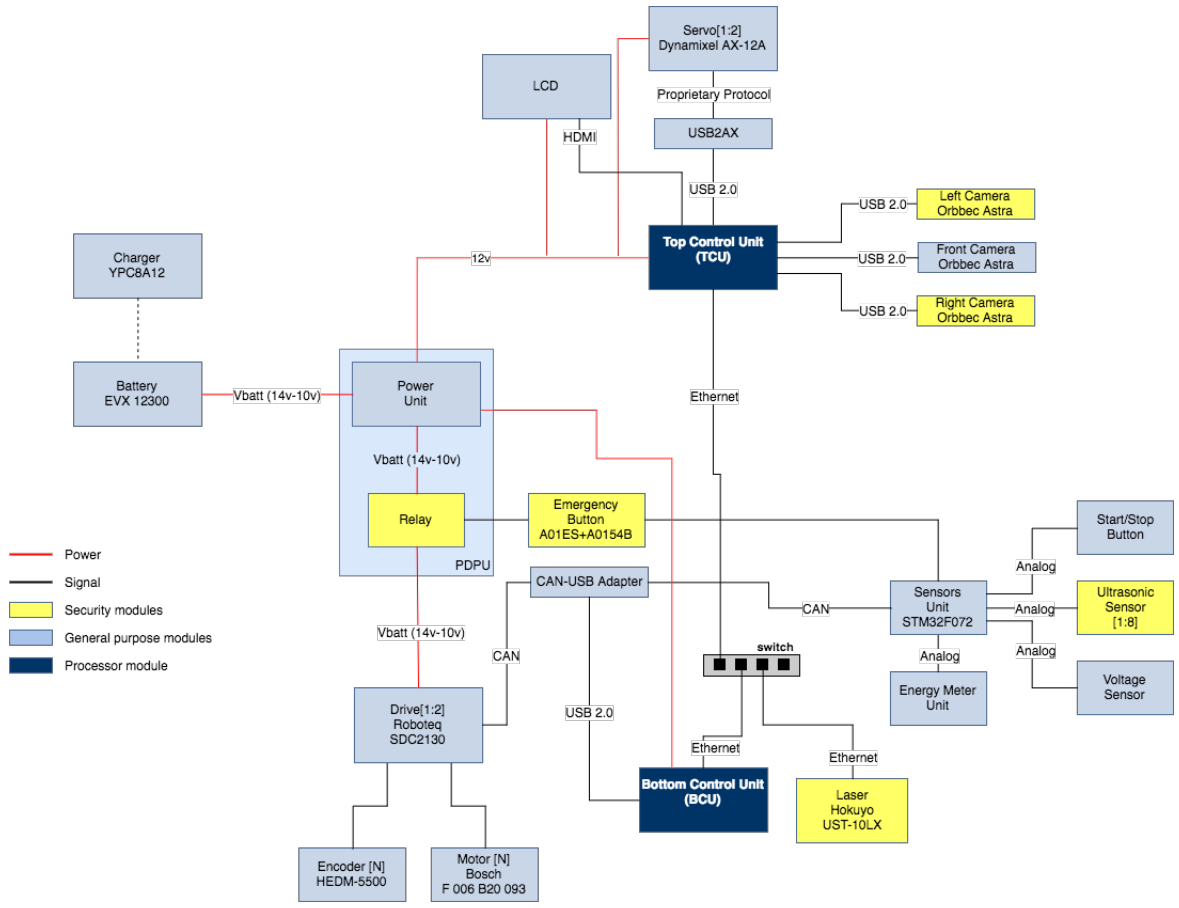


Figura 5.1: *Arquitetura modular proposta*

5.1.1 Software

Para o software vão ser propostas três arquiteturas diferentes de maneira a avaliar diferentes configurações do sistema.

A primeira arquitetura consiste em executar apenas a perceção, (*wiigo_perception_astra* no computador localizado na cabeça do wiigo (TCU), isto inclui todo o processamento de imagem e deteção de obstáculos, o resto dos nós serão executados no computador da base.

A segunda arquitetura passa por correr apenas o driver do *hokuyo*, o *costmap* e outros nós relacionados com a navegação no computador localizado na base e o resto do processamento no computador localizado na cabeça. Esta arquitetura simula a hipótese de colocar apenas um computador de baixa gama na base.

A terceira arquitetura é uma variação da segunda mas neste caso a comunicação entre os computadores é efetuada via DDS e não por ROS.

5.1.2 Arquitetura 1 - processamento da perceção no PC da cabeça

Neste caso serão executados os *nodes* responsáveis pelo processamento de imagem no computador que seria localizado na cabeça. O computador da base será o *master* e vai ser responsável pelo resto do processamento.

No entanto, preciso salientar que os drivers da câmara (*camera, left right*) só podem ser lançados no PC onde as câmaras estão ligadas fisicamente. Os nós do *launcher* da perceção (*wigo_perception_astra.launch*) que vai correr neste computador são os seguintes:

- **camera** - Driver da câmara central;
- **left** - Driver da câmara esquerda;
- **right** -Driver da câmara direita;
- **wiigo_identification_node** - Calcula um valor que corresponde à confiança no utilizador que está a seguir;
- **wiigo_ground_plane** - Carrega a calibração e calcula a altura do utilizador e a sua projeção no chão;
- **detection_2d** - Caso a câmara central perca os dados de profundidade faz o seguimento só com a imagem RGB;
- **wiigo_acquisition_astra** -Aquisição da imagem da câmara central;
- **upper_body_detector** - Associação de modelos para deteção de utilizadores;
- **wiigo_perception_com** - Efetua toda a comunicação entre os nós da perceção;
- **left_lateral_scan** - Deteção de obstáculos pela câmara esquerda;
- **right_lateral_scan** - Deteção de obstáculos pela câmara direita;
- **wiigo_fused_scan** - Funde as leituras do laser com as leituras das câmaras laterais;
- **wiigo_detection_filtered** - Filtra deteções do laser;
- **wiigo_laser_tracker** - Seguimento das deteções do laser;

5.1.3 Arquitetura 2 - Navegação no PC da base

Nesta arquitetura o PC principal (*master*) será o da cabeça e o da base ficará apenas responsável pela navegação. Esta hipótese é a que consome menos largura de banda pois não há envio de imagens entre computadores. O conjunto de nós foi escolhido de maneira a que o *costmap* pudesse ser executado independentemente no segundo computador, evitando a transferência de quantidades elevados de dados pela rede, como por exemplo imagens. Neste computador da base serão executados os seguintes nós:

- **urg** - Este nó lança o driver do *Hokuyo*;
- **wiigo_nav** - navegação;
- **wiigo_person** - Calcula a velocidade da pessoa;
- **wiigo_nav_astar** - Executa o algoritmo A*;
- **sensor_fusion** - Funde as leituras do lidar com as leituras das câmaras e sonares;
- **wiigo_battery_indicator** - Calcula a percentagem de bateria restante;
- **wiigo_zone** - Define as zonas de segurança;
- **laser_filter** - Filtra as leituras do laser;

5.1.4 Arquitetura 2 com comunicação por DDS

Como a implementação da comunicação por DDS vai ser feita apenas com fins de comparação foi feita uma abordagem mais simplista do problema. O objetivo desta experiência é descentralizar o sistema tornado assim cada uma das maquinas independente da outra. Assim cada um dos PC's vai correr um *roscore* que apenas terá conhecimento dos nós que vão ser executados nele próprio. Desta forma caso haja uma falha num computador e por exemplo um *roscore* for abaixo, os nós que estão a correr no segundo PC não serão afetados. Outra vantagem do DDS são as diferentes configurações de QoS e durabilidade que podem influenciar muito a transferência de dados, especialmente em redes pouco confiáveis.

A solução escolhida passou por criar um nó para cada uma das duas máquinas. Este nó terá a função de fazer de ponte para todos os tópicos que são necessários subscrever ou publicar localmente.

As mensagens do DDS têm de ser definidas previamente numa estrutura IDL. De maneira a evitar a necessidade de criar esta estrutura para cada um dos tipos de mensagens optou-se por fazer a serialização de cada mensagem com recurso a classe *ros::serialization*. Desta forma é apenas necessária uma estrutura IDL que contenha um vetor com tamanho

suficiente para guardar a *string* de caracteres da mensagem. A desvantagem deste método é que cada mensagem vai passar pelo processo de serialização e deserialização duas vezes, uma internamente pelo do ROS por causa da ligação TCP entre tópicos e outra para ser enviada por DDS.

A arquitetura proposta deste sistema está representada no diagrama da figura 5.2, onde os nós criados vão ser o *Talker* e o *Listner* cuja finalidade será subscrever aos tópicos que se deseja enviar para o outro PC e publicar os tópicos resultantes dos nós que estão a correr nessa outra máquina. Por cada um dos tópicos que se deseja enviar ou receber será criado um tópico no *opensplice* que publica ou subscreve dentro do *DDS global data space*.

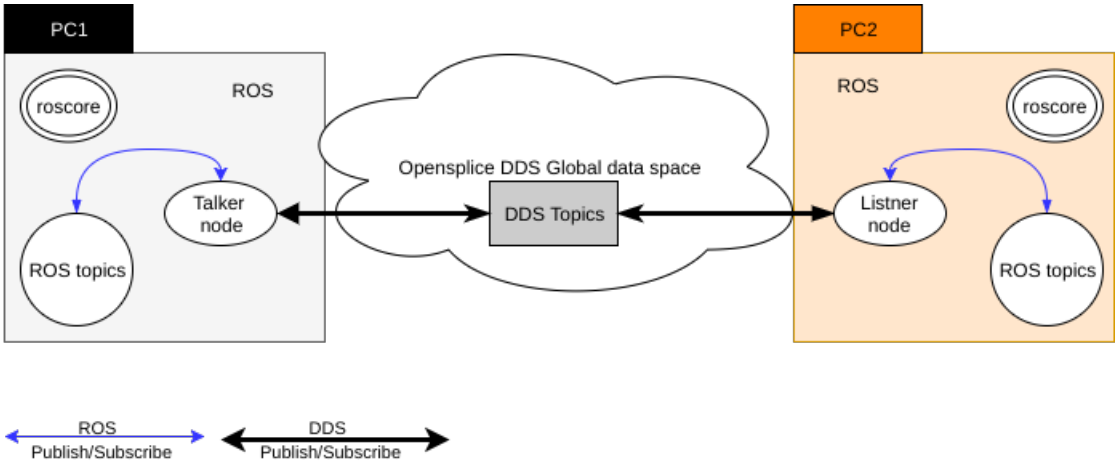


Figura 5.2: *Arquitetura do sistema usando DDS*

Para publicar um tópico de ROS para DDS será necessário criar um subscritor em ROS que irá subscrever ao tópico em questão. Cada vez que no ROS chega uma mensagem nova é automaticamente chamado o *callback* deste subscritor onde a mensagem é serializada e publicada no correspondente tópico DDS. Chegando uma nova mensagem pelo DDS é chamado o *callback* correspondente onde ela é desirializada e publicada num tópico ROS.

Os nós *talker* e *Listner* fazem essencialmente o mesmo processo mas para tópicos diferentes.

5.2 Escolha de Processadores

A escolha foi feita essencialmente com base no material disponível na empresa e porque os processadores disponíveis englobam as varias gamas à venda no mercado. Na tabela 5.1 encontra-se a lista dos processadores usados neste trabalho.

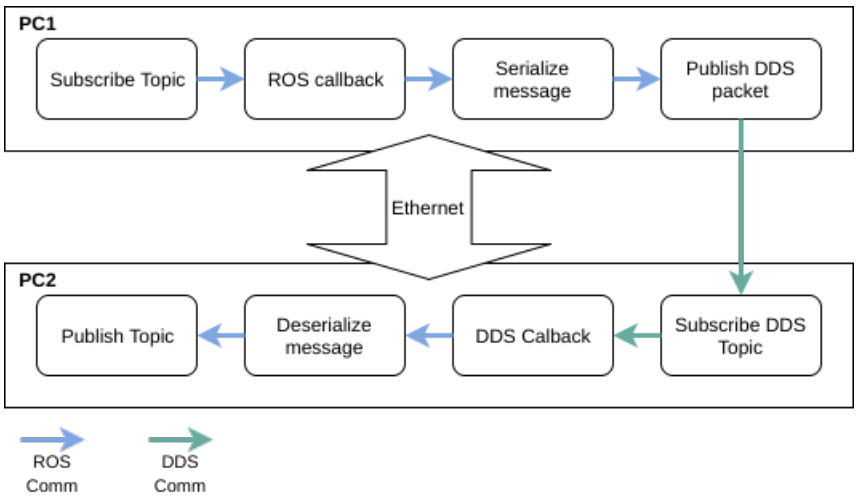


Figura 5.3: *Arquitetura de software da implementação do DDS*

Tabela 5.1: *Lista de processadores usados.*

	Intel I5-4460	**Intel I5-4258U**	**Intel i3-6100U**	**Intel Celeron J1900**
Nº Núcleos	4	2	2	4
Nº Threads	4	4	4	4
Clock base	3.2 GHz	2.4 GHz	2.3 GHz	2.0 GHz
Clock Turbo	3.4 GHz	2.9 GHz	-	2.42 GHz
Memoria Cache	6 MB	3 MB	3 MB	2 MB
Litografia	22 nm	22 nm	14 nm	22 nm
TDP Máx.	84 W	28 W	15 W	10 W
Gráficos	Intel HD Graphics 4600	Intel Iris Graphics 5100	Intel HD Graphics 520	Intel HD Graphics
Data de Lançamento	Q2'2014	Q3'2013	Q3'2015	Q4'2013

5.2.1 Teste dos processadores

Para obter uma classificação numérica da performance de cada computador usou-se o *Geekbench*. O *Geekbench* [45] é um software multi-plataforma para teste de processadores. Estes testes simulam cargas de trabalho reais e os resultados são divididos numa pontuação para tarefas que fazem uso de um ou múltiplos núcleos.

A empresa que desenvolve este software fornece uma versão grátis e disponibiliza no seu site resultados de múltiplos processadores para comparação com o sistema atual. Estas pontuações são usadas para quantizar a performance dos processadores usados e permitem também sugerir outras configurações. Na tabela 5.2 é possível observar os resultados do teste para os quatro processadores usados neste trabalho.

5.3 Configuração do sistema Operativo

De modo a aumentar a performance do sistema sistema e dado que o segundo PC não tem necessidade de mostrar imagem foi desativado o seu ambiente gráfico. Para tal editou-se o

Tabela 5.2: *Resultados do teste GeekBench*

	Single-Core Score	Multi-Core Score
Intel I5-4460	2980	9551
Intel I5-4258U	2715	5862
Intel i3-6100U	2479	5208
Intel Celeron J1900	895	2867

ficheiro localizado em **/etc/default/grub** 5.1 e alterou-se a variável *GRUB_CMDLINE_LINUX_DEFAULT* de *quiet splash* para *text*.

Listing 5.1: *Ficheiro /etc/default/grub*

```
GRUB_CMDLINE_LINUX_DEFAULT="text"
```

Desta forma cada vez que se inicia o computador ele arranca em modo de texto. Caso seja necessário lançar o ambiente gráfico a partir da consola basta fazer login e executar o seguinte comando 5.2.

Listing 5.2: *Iniciar o GUI*

```
sudo service lightdm start
```

5.4 Configuração de rede

Dado que o *Hokuyo* ocupa uma porta Ethernet cria-se um problema de falta de portas disponíveis para ligar o segundo PC. Face a este problema existiam duas hipóteses, ligar um adaptador USB-Ethernet ou usar um *switch*. Foi escolhido o *switch* devido à sua capacidade de ligar mais maquinas em rede e por possibilitar que o Lidar fique acessível a ambos os PC's.

Após ligar fisicamente as maquinas ao *switch*, é necessário configurar a rede através do menu *Network connections*. Em cada um dos computadores foi criada uma ligação de rede com IP estático, figura 5.4.

Foi escolhida a gama de IP's *192.168.0.xx* para ambos os PC's ficarem configurados na mesma sub rede do Lidar *Hokuyo*.

Depois de definido um IP fixo para cada um dos computadores presentes na rede é associado um *hostname* a cada um dos computadores, sendo neste caso o *master* chamado de pc1.

Listing 5.3: *Ficheiro etc/hosts do wiigo*

```
127.0.0.1       localhost
```

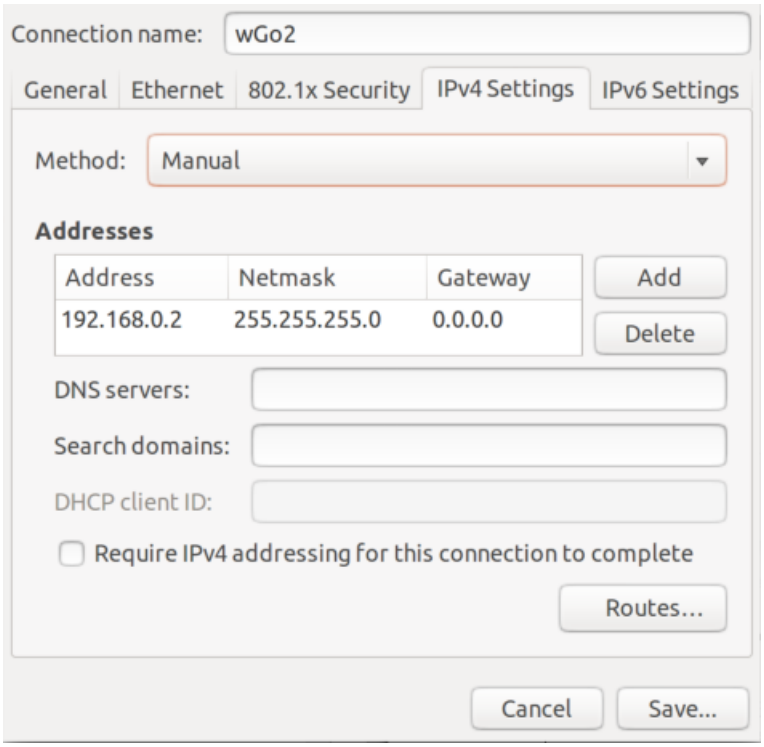


Figura 5.4: *Ligações de rede no Ubuntu.*

```
127.0.1.1       wiigo
192.168.0.2     pc2
192.168.0.1     pc1
```

5.5 Configuração do Chrony

De modo a evitar discrepâncias no tempo devido a diferenças nos relógios internos das maquinas configurou-se o *chrony* de modo a sincronizar o segundo PC com o primeiro.

Depois de instalado é necessário editar no PC secundário o ficheiro **/etc/chrony/chrony.conf**

Listing 5.4: *Ficheiro chrony.conf do pc2*

```
server pc1 minpoll 0 maxpoll 5 maxdelay .0005
initstepslew 0 pc1
makestep 0.005 -1
```

Depois de reiniciar o Chrony *deaemon* é possível consultar o estado do chronyd com o comando **chronyc tracking**.

5.6 Processamento distribuído em ROS

O ROS foi desenhado desde o inicio com processamento distribuído em mente. Este *middleware* oferece a possibilidade de alocar o processamento de cada nó a computadores diferentes. Para tal é necessário executar apenas numa das máquinas um processo principal apelidado de *master* que gere as ligações do sistema. Todos os nós devem estar configurados com o mesmo *master* através da definição da variável *ROS_MASTER_URI*. Deve também haver comunicação bidirecional entre cada par de máquinas e cada uma delas deve anunciar o seu nome de maneira a que todos computadores na rede tomem conhecimento da sua existência.

5.6.1 Configuração do Roslaunch

Uma das maneiras mais simples de executar nós ROS em computadores diferentes é através da criação de um ficheiro *.launch* com os nós desejados para cada uma das maquinas e através dos terminais correspondentes executar manualmente este *launcher*.

Inicialmente por motivos de teste começou-se por comentar no *launcher* principal *wigo_launcher.launch* os nós que se desejavam correr na segunda máquina. Estes nós foram depois incluídos num segundo *launcher* que era lançado manualmente a partir da consola do segundo computador.

De modo a funcionar foi necessário definir algumas variáveis de ambiente ROS editando cada um dos ficheiros *.bashrc* das respetivas máquinas. Isto permite que o *master* tenha conhecimento de todas as maquinas da rede e que as maquinas o conheçam a si.

Na máquina 1 que corresponde ao *master* é definida a variável *ROS_MASTER_URI* que contem o seu próprio IP e a variável *ROS_HOSTNAME* que contem o seu *hostname*.

Na maquina 2 é definido o mesmo *ROS_MASTER_URI* do *master* e o *ROS_HOSTNAME* desta mesma maquina. Existe também a possibilidade de realizar esta configuração usando a variável *ROS_IP* em vez da *ROS_HOSTNAME*.

O processo de edição manual dos *.launch* é pouco eficiente pois requer que vários ficheiros *roslaunch* tenham de ser editados e que pelo menos dois terminais sejam abertos (um em cada PC) de modo a ser possível lançar todos os nós.

5.6.2 Configuração do Roslaunch Machine

Como já foi referido anteriormente no estado da arte o *roslaunch* permite também lançar nós ROS a partir de apenas uma máquina e definir em quais dos computadores é executado cada nó.

De modo a ser possível lançar todos os nós automaticamente nas duas máquinas foi necessário criar um *script* para carregar todas as variáveis de ambientes necessárias. Estes

scripts definiem as variaves de ambiente ROS necessárias.

Ao todo foram criados dois scripts *host_env.sh* para o PC principal e *slave_env.sh* para o PC secundário. Estes ficheiros foram colocados na diretoria */opt/ros/indigo*

Listing 5.5: *Ficheiro host_env.sh*

```
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:/opt/wiigo/instal
export ROSLAUNCH_SSH_UNKNOWN=1
. /opt/wiigo/install/setup.sh
exec "$@"
```

Por sua vez foi acrescentado ao final do ficheiro *setup.sh* 5.6 a definição das variáveis ROS_MASTER_URI e ROS_IP da maquina respetiva.

Listing 5.6: *Final do ficheiro setup.sh*

```
export ROS_MASTER_URI=http://pc1:11311
export ROS_IP=192.168.0.2
```

Para funcionar em conjunto com o *launcher* existente foi adicionado o código em 5.7 ao inicio ficheiro *wiigo_launcher.launch*.

Deste modo, antes de lançar todos os nós é definida a variável *ROS_MASTER_URI*, que corresponde ao endereço do PC onde está a correr o *roscore* (pc1). Seguidamente é feita uma ligação SHH à segunda maquina que tem o endereço *pc2*, sendo depois carregadas as suas variáveis de ambiente através do *script* correspondente. Finalmente é definido o endereço do PC principal (pc1), que vai ser a maquina escolhida por defeito para lançar os nós e são depois também carregadas as variáveis de ambiente contidas no seu *script*.

Listing 5.7: *Launch das maquinas*

```
<env name="ROS_MASTER_URI" value="http://pc1:11311"/>

<machine name="c2" address="pc2" env-loader="/opt/ros/indigo
   /slave_env.sh" user="wiigoindustria" password="wiigo"/>
<machine name="c1" address="pc1" default="true" user="wiigo"
    env-loader="/opt/ros/indigo/host_env.sh"/>
```

Para definir os nós que vão ser lançados na segunda máquina basta apenas escrever o nome da maquina desejada a seguir à *tag* <*node*>.

5.6.3 Rostopic Statistics Debug

Nos testes em que os nós do *perception* são executados num segundo PC verificou-se que muitas vezes os nós *wiigo_identifaction* e *perception_com* iam abaixo. Dado que estes dois

nós são *required* a sua terminação significa que todo o *launcher* vai abaixo.

De modo a descobrir a origem deste problema utilizou-se a funcionalidade de *debug* do *roslaunch*. O *debuger* escolhido foi o GDB, para tal foi passado o parâmetro *launch-prefix="xterm -e gdb –args"*. Desta maneira cada vez que é lançado o nó abre uma consola à parte com o GDB, para iniciar o nó basta escrever *run* na consola. Após vários testes verificou-se que quando um dos nós ia abaixo lançava sempre o mesmo erro *std::runtime_error what(): Duration is out of dual 32-bit range.*

Constatou-se então que este erro só ocorria quando as estatísticas do ROS estavam ligadas. Numa pesquisa *online* descobriu-se que no *ROS answers* [46] outros utilizadores reportaram este problema quando ativaram as estatísticas no ROS. Após uma analise mais detalhada concluiu-se se seria um bug no *statistics.cpp* . Na data de 9 de Agosto de 2017 foi efetuado um *commit* no *github* do ROS por um membro da comunidade com o titulo de *ignore headers with zero stamp in statistics* que aparentemente resolvia o erro em questão. Infelizmente à data da realização dos testes esta atualização ainda não estava disponível no repositório *apt-get* logo não foi possível confirmar se o problema ficou resolvido.

5.7 ARNI

Antes de instalar este pacote é necessário instalar duas bibliotecas de python, *pyqtgraph, pysensors* e fazer atualização à biblioteca *psutil*. A versão do *psutil* instalada por defeito é a 1.2.1, no entanto o *ARNI* necessita pelo menos da versão 2.1.

Ao atualizar com o *–upgrade* vai mudar para a versão mais recente (5.2) que apesar de funcionar não calcula as frequências dos tópicos. Para fins de testes as frequências não serão necessárias mas caso sejam precisas no futuro devera-se testar com uma versão do *psutil* mais antiga.

Listing 5.8: *Instalação das bibliotecas python*

```
pip install --user --upgrade psutil
pip install --user pysensors
pip install --user pyqtgraph
```

Estando as bibliotecas instaladas é possível então fazer o download do pacote dentro do diretório */catkin_w/src/WiiGo_sw* e depois compilar com o *catkin_make* ou *wiigo_compile*.

Listing 5.9: *Comando de instalação*

```
git clone https://github.com/ROS-PSE/arni.git
```

Após os dois computadores estarem corretamente configurados em rede, lança-se o

roscore. Antes de lançar os nós e de maneira a inicializar os parâmetros para gerar estatísticas executa-se o seguinte *roslaunch*.

Listing 5.10: *Roslaunch das inicializações*

```
roslaunch arni_core init_params.launch
```

Finalmente, de modo a obter a informação pretendida é necessário executar o comando da listagem 5.11 em cada um dos computadores.

Listing 5.11: *Rosrun do ARNI*

```
rosrun arni_nodeinterface arni_nodeinterface
```

Para abrir a interface gráfica do *ARNI* é necessário executar o *rqt* e depois selecionar *plugins-> Arni-Detal* e *plugins-> Arni-Overview*.

5.8 ROS Multimaster Extension

O pacote *multimaster_fkie* pode ser obtido através dos repositórios apt-get ou do *GitHub*. O download do pacote foi feito dentro do *catkin workspace catkin_ws/src* com o comando da listagem 5.12.

Listing 5.12: *Comando de instalação*

```
git clone -b indigo-devel https://github.com/fkie/
   multimaster_fkie.git
```

Para compilar é preciso apenas usar o *catkin_make* ou o *wiigo_compile*. Este processo tem de ser efetuado em cada uma das máquinas pois o pacote instala algumas bibliotecas localmente, logo não basta enviar os executáveis pelo *script sendToWiigo*.

Antes de iniciar é necessário verificar que a interface de rede está configurada como *multicast*, que todos os *hosts* estão definidos no ficheiro */etc/hosts*.

Para cada um dos computadores a variável *ROS_MASTER_URI* deve ser configurada para *localhost* e o *ROS_IP* para o IP do PC correspondente, listagem 5.13.

Listing 5.13: *Variaveis de sistema do ROS*

```
export ROS_MASTER_URI=http://localhost:11311
export ROS_IP=192.168.0.2
```

Estando o *ROS_MASTER_URI* corretamente definido e o *roscore* a correr em cada um dos PC's, iniciam-se os nós *master_discovery* e *master_sync*. Opcionalmente é possível também lançar estes dois nós usando uma *launch file*.

Listing 5.14: *Execução dos nós*

```
rosrun master_discovery_fkie master_discovery &
rosrun master_sync_fkie master_sync
```

Após estes dois nós estarem a correr em ambos os computadores é possível lançar os *launchers* do wiigo nos respetivos PC's.

Para monitorizar e configurar os nós nas varias maquinas este pacote inclui uma interface gráfica chamada de *Node Manager*. De modo a abrir este programa basta escrever na consola **node_manager**.

5.9 Configuração e Instalação do OpenSplice DDS

Após o download da versão desejada é extraído o ficheiro para um diretório à escolha e depois atualizado o diretório de instalação que corresponde à variável *OSPL_HOME* com o comando *sed*, listagem 5.15.

Listing 5.15: *Comandos de instalação*

```
gtar -xzf OpenSpliceDDSV4.1.090513-x86.linux2.6-gcc412-
   gnuc25-HDE.tar.gz
sed -i "s|@@INSTALLDIR@@|$PWD|g" HDE/x86.linux2.6/release.
   com
```

Para direcionar o *opensplice* para o ficheiro de configuração de rede é definida a variável *OSPL_URI*. Como foi utilizada a versão *comunity* é apenas possível usar a configuração de processo único, sendo que a configuração multi-processo com memoria partilhada está disponível apenas na versão paga.

Configuração de processo único significa que o domínio do serviço, administração da base de dados e outros serviços necessários são iniciados com o processo da aplicação DDS quando esta invoca a operação *create_participant*. Para edição dos ficheiros XML de configuração é recomendado usar a ferramenta *osplconf* que vem instalada com o *opensplice*.

Neste cenário de aplicação foi usado o ficheiro padrão *ospl_sp_ddsi.xml* que é usado para a configuração de processo único *standalone* e configuração de rede DDS padrão.

Como o *opensplice* tem bibliotecas dinâmicas, cada vez que se deseja executar programas que fazem uso do *opensplice* é necessário fazer *source* no terminal do *script release.com*

Listing 5.16: *Source do script opensplice*

```
source /home/wiigo/HDE/x86_64.linux/release.com
```

5.10 Arquitetura 2 com comunicação por DDS

De modo a simplificar a implementação da camada DDS com o ROS foi decidido criar dois nós, um em cada computador. Após pesquisa pela Internet verificou-se que não existia nenhum *wrapper* em código aberto de DDS para ROS1. No entanto encontrou-se uma biblioteca que integra o DDS *opensplice* no pacote ROS para enxames de robôs chamado *micros_swarm_framework* [47]. Esta biblioteca implementa uma classe que faz uso dos ficheiros gerados a partir da estrutura IDL. Deste modo é possível criar e subscrever a um tópico DDS apenas pelo seu nome. Cada um destes nós irá converter as mensagens ROS para DDS e vice versa.

O DDS usa os ficheiros IDL para gerar bibliotecas na linguagem de programação desejada.

Criou-se então a seguinte estrutura que consiste numa *string* com 64 caracteres, uma variável do tipo *long* e um *array* de 8192 *octets* que é o equivalente a um *array* de 8192 bytes. Este tamanho é o suficiente para guardar uma mensagem de um varrimento laser completo.

De maneira a manter compatibilidade com o código existente manteve-se o nome do modulo e da estrutura. Desta forma é possível manter a estrutura do *cmake* da biblioteca existente.

As variáveis do tipo *prefix* e *sampleId* acabaram por ser usadas apenas com fins de *debug* porque toda a informação necessária vem na *payload*.

Listing 5.17: *Ficheiro IDL*

```
module opensplice_dds_comm{
        struct GSDFPacket {
         string<64>             prefix;
         long                   sampleId;
         sequence<octet, 8192>  payload;
        };
        #pragma keylist GSDFPacket
};
```

Listing 5.18: *Comando para gerar ficheiros cpp*

```
idlpp -S -l c++ HelloWorld.idl
```

Dentro dos *callbacks* ROS é feita a serialização das mensagens e por sua vez

Após uma primeira implementação notou-te uma diferença significativa entre as frequências publicadas e a frequência recebia do outro lado. Dado que a frequência recebida era

significativamente inferior à frequência de publicação original pensou-se inicialmente que poderia ser devido ao tempo de serialização das mensagens ou perda de pacotes pelo DDS.

De modo a testar se o problema era do DDS colocou-se o nó *talker* a enviar a mesma mensagem para o *listner* a uma frequência fixa usando o *ros::Rate*. Verificou-se neste teste que a frequência do tópico do *listner* era igual à *Rate* definida no *talker*. Com este resultado concluiu-se que o problema não era do DDS nem do tempo de serialização pois mesmo a 200 Hz não houve perdas. Após uma pesquisa mais exaustiva descobriu-se que esta latência era devido ao ROS usar o algoritmo de *Nagel* para reduzir o numero de pacotes que são enviados pela rede. O efeito deste algoritmo é mais notável no tópico */tf* pois este tem uma frequência mais elevada e o tamanho de cada mensagem é menor.

Para resolver este problema foi usada a classe *ros::TransportHints* e a função booleana *tcpNoDelay()*. Esta função "desliga"o algoritmo de *Nagel* [37] para o subscritor em questão e assim as mensagens passam a chegar na altura em que são enviadas.

De modo a diminuir ainda mais a latência das usou-se o *ros::MultiThreadedSpinner* para dividir os *callbacks* por *threads*.

5.11 Teste final com dois computadores de baixa gama

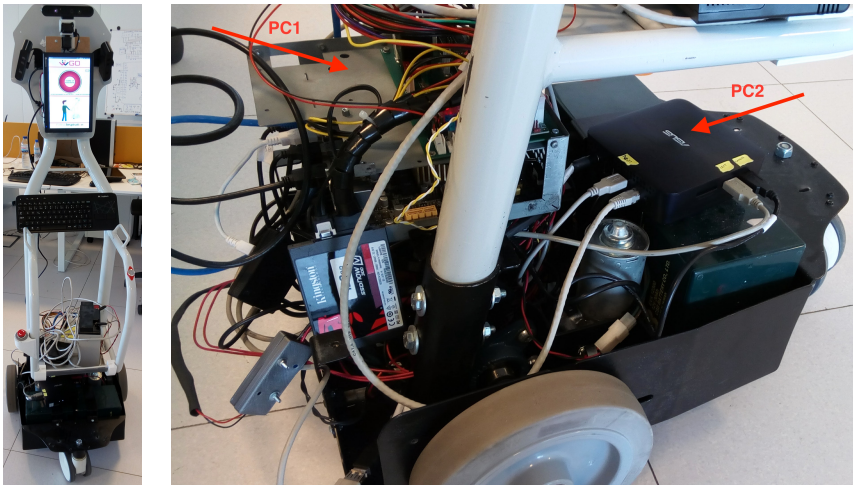
Este teste foi realizado para avaliar a viabilidade do uso de dois PC's de baixa gama como substitutos do PC original do *Wiigo*.

O computador 1 era uma *Motherboard Asus J1900i-c* com o CPU integrado *Intel Celeron Quad-Core J1900*, este conjunto tem um consumo de apenas 15W. De modo a alimentar este sistema usou-se uma placa de alimentação ATX alimentada por uma bateria de 12V.

O computador 2 foi um *Asus VivoMini UN65H* com um processador *Intel i3-6100U* este mini-PC tem um formato reduzido de 131 x 131 x 52 mm e um peso de apenas 0.7kg. O objetivo seria colocar este computador de dimensões reduzidas na cabeça do *WiiGo* para realizar o processamento das três câmaras.

A escolha destes PC's em particular deveu-se ao facto de que dentro da lista dos computadores disponíveis estes eram os dois mais fracos e também os mais fáceis de transportar.

Para lançar o programa foi usado o *launcher* global sendo que o computador 2 foi responsável pela execução dos nós de perceção e diagnósticos e o computador 1 pelo resto do processamento. Desta forma foram ligadas as três câmaras ao computador 2, que teoricamente ficaria localizado na cabeça e o resto dos dispositivos (ecrã, *USB2AX*, *Roboteq* e *Sensor Board*) ao computador 1 que fica na base. Finalmente o *Hokuyo* e os dois computadores ficam ligados ao *switch*, figura 5.6.



(a) wiiGO (b) Computador 1 e computador 2 na base do wGO

Figura 5.5: *Configuração da experiência*

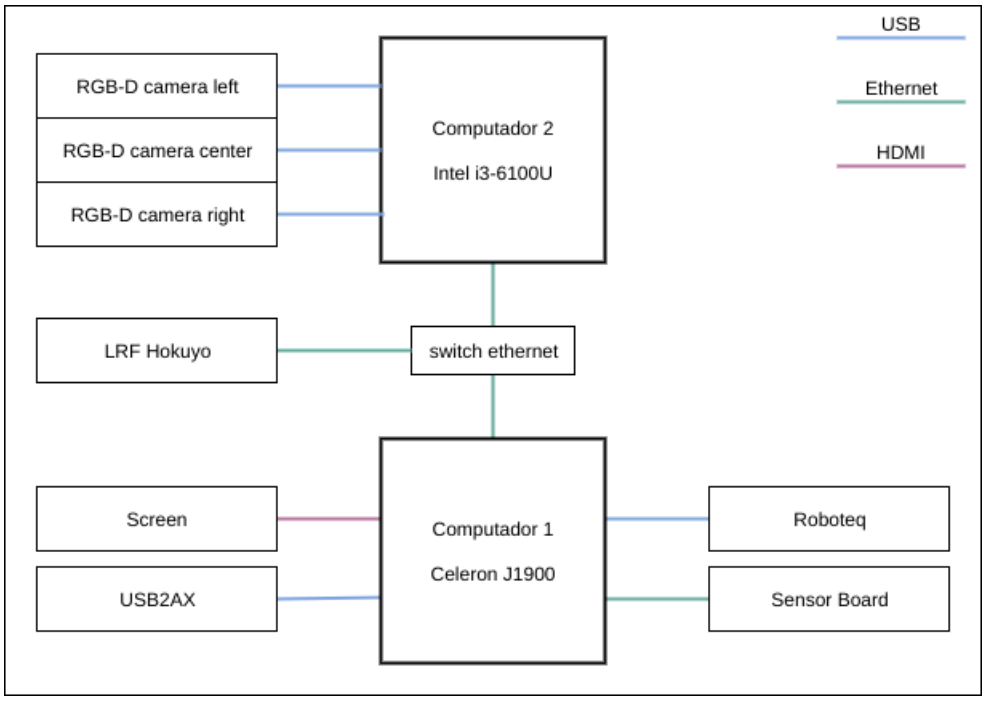


Figura 5.6: *Arquitetura do novo Sistema*

6 Resultados

Neste capitulo vão ser apresentados os resultados das experiências realizadas com uma ou múltiplas maquinas. Os dados de utilização do processador foram obtidos através dos tópicos publicados pelo *rosprofiler*, este teste teve uma duração de 120 segundos. As frequências dos tópicos e mensagens perdidas vieram do tópico */statistics*, este teste teve uma duração de 60 segundos.

Todos os resultados foram obtidos com o *wiigo* a funcionar em modo de seguimento. Devido a isto foi necessário o robô seguir uma pessoa durante a duração de cada um dos testes.

Os resultados foram guardados em ficheiros *.csv* com o comando *rostopic echo -p*. Para apresentar estes resultados foram desenvolvidos vários *scripts* em *Matlab* para ler, processar dados e desenhar os gráficos.

6.1 Tabela de Comparação de tempos

Inicialmente na empresa foi efetuada uma tarefa para documentação do tempo de processamento de cada *node*. Esta medição consistia em registar o tempo despendido pelo CPU e o tempo do sistema numa determinada função ou bloco de código à escolha.

Como o código de teste original não estava disponível foi criado um novo código que calculava estes tempos e guardava-os num ficheiro.

Terminando este teste verificou-se que o método de medição utilizado não era o ideal porque requer que todos os ficheiros fonte dos *nodes* ROS a medir sejam editados indivi-

dualmente. Numa base de código que está em constante mudança este processo seria uma tarefa morosa e acabaria por gerar resultados pouco precisos.

Estes problemas levaram ao uso das ferramentas ROS existentes que foram desenvolvidas com a finalidade de introspeção do sistema (*/statistics* e *rosprofiler*). Os resultados obtidos com estas ferramentas foram mais fáceis de reproduzir e comparar.

A tabela de comparação de tempos de execução de algumas das funções que fazem parte dos *nodes* ROS encontra-se no anexo A.

6.2 ARNI

O ARNI foi executado com o *wiigo* a funcionar numa configuração com dois computadores. Na figura 6.1 é possível observar algumas das informações fornecidas por este programa. Dentro do retângulo (1) encontra-se o indicador do estado do sistema, neste caso está *offline* pois não foram executadas nenhumas contra-medidas. No (2) encontra-se informação geral (utilização de CPU, RAM etc) e gráficos do sistema. Os dados do */statistics* do ROS estão na janela (3), é possível visualizar em tempo real as estatísticas de cada *node* e tópico para cada *host* ligado na rede.

Com base nos dados obtidos pelo ARNI foi possível saber que no total o software do *wiigo* tem **52 *nodes***, **257 tópicos** e **440 ligações publish-subscribe**.

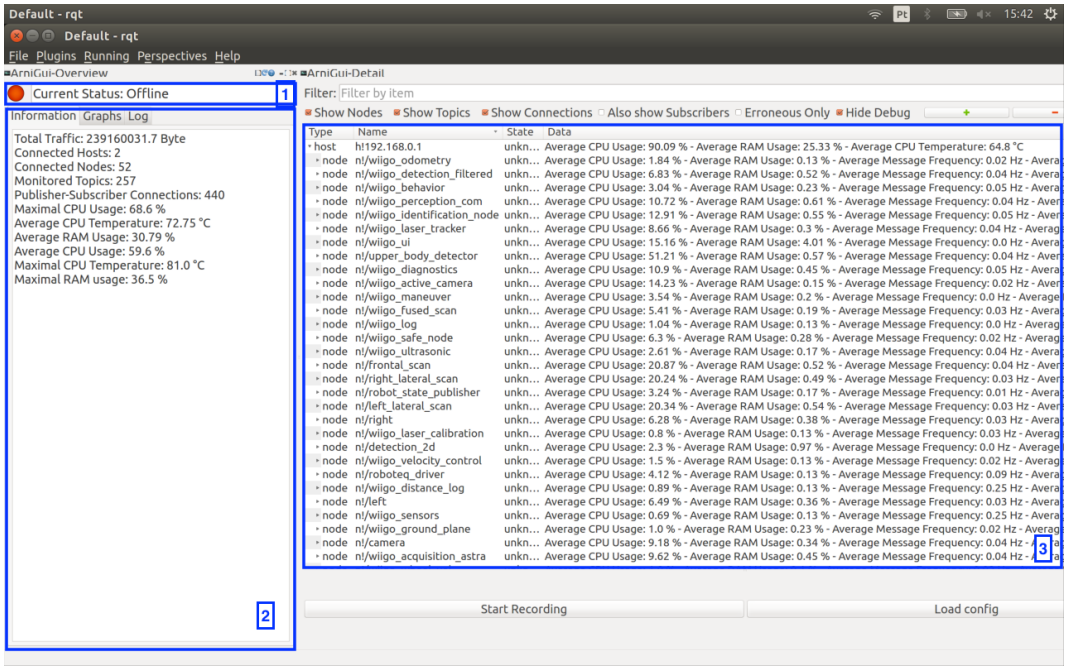


Figura 6.1: *Interface gráfica do ARNI.*

6.3 Multimaster fkie

Este pacote ROS foi testado na arquitetura 2, com a navegação a ser executada no segundo computador *I5-4258U* e o resto dos *nodes* no computador do WiiGo *I5-4460*.

Na figura 6.2 é possível ver-se *GUI* do *node manager*. Esta interface gráfica fornece informações sobre os *nodes* ROS que estão a ser executados a cada computador e os tópicos associados a cada um destes *nodes*.

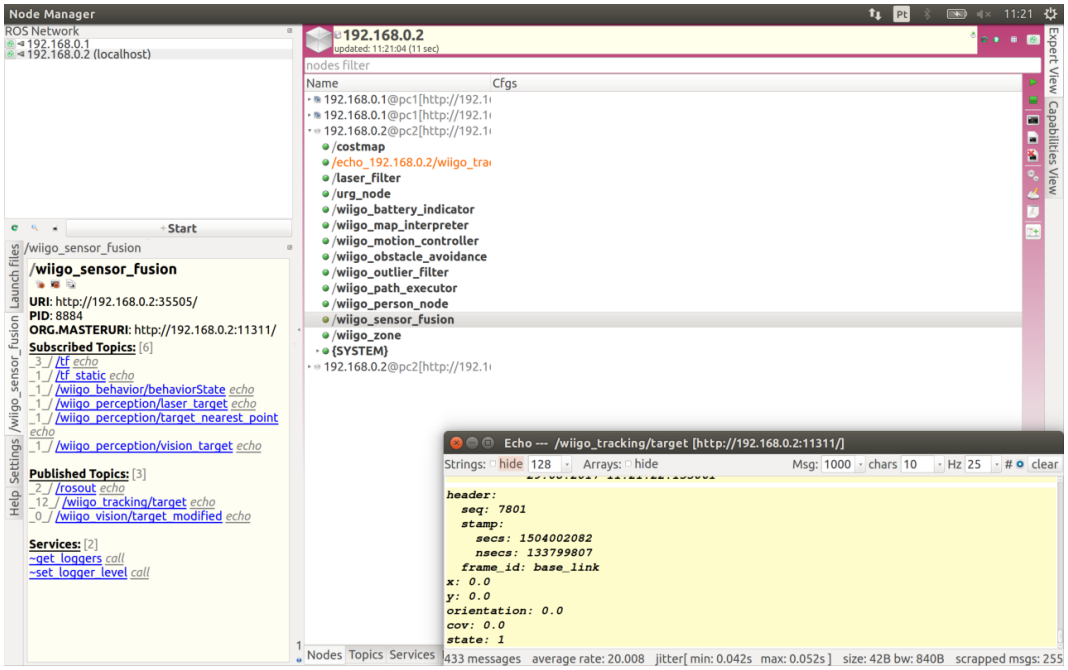


Figura 6.2: *Interface gráfica do Multimaster fkie.*

6.3.1 Utilização do processador do sistema

Nesta situação foi ligado o computador principal do wiigo (I5-4460) ao *MacBook Pro* (I5-4258U). A figura 6.3 apresenta um gráfico da percentagem de utilização de cada um dos núcleos dos dois processadores usados na experiência.

Observando o gráfico verifica-se que tal como na experiência sem o *multimaster* o CPU do wiigo continua perto dos 100%.

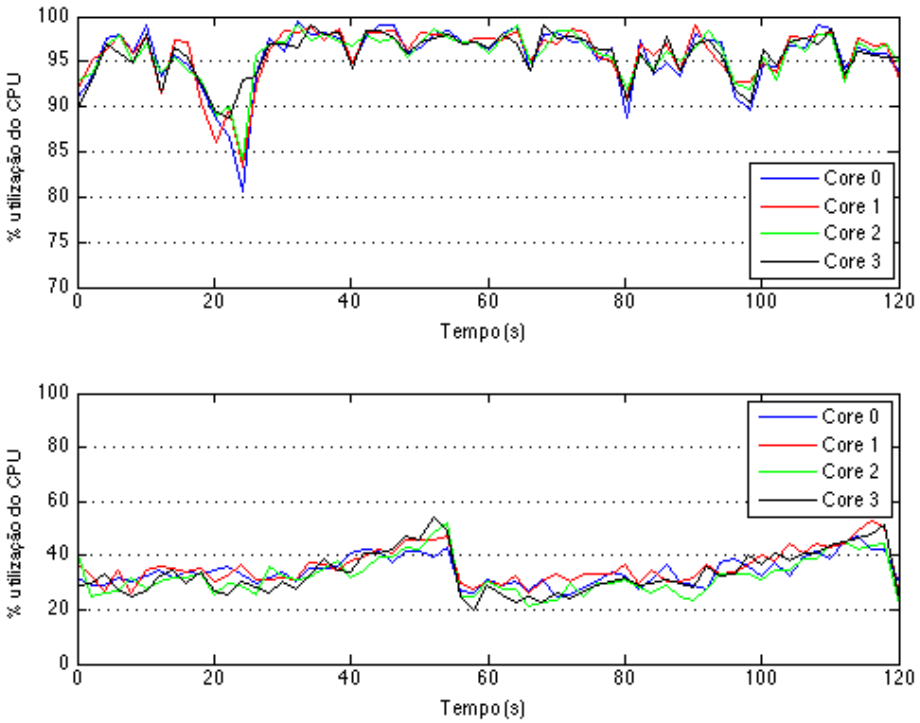


Figura 6.3: *Percentagem de carga em cada um dos núcleos do processador, gráfico superior I5-4460 e inferior I5-4258U, usando o pacote multimaster.*

Utilização do processador pelos *nodes*

Na Figura 6.4 encontra-se um gráfico com os *nodes* ROS que gastam mais CPU em cada um dos computadores.

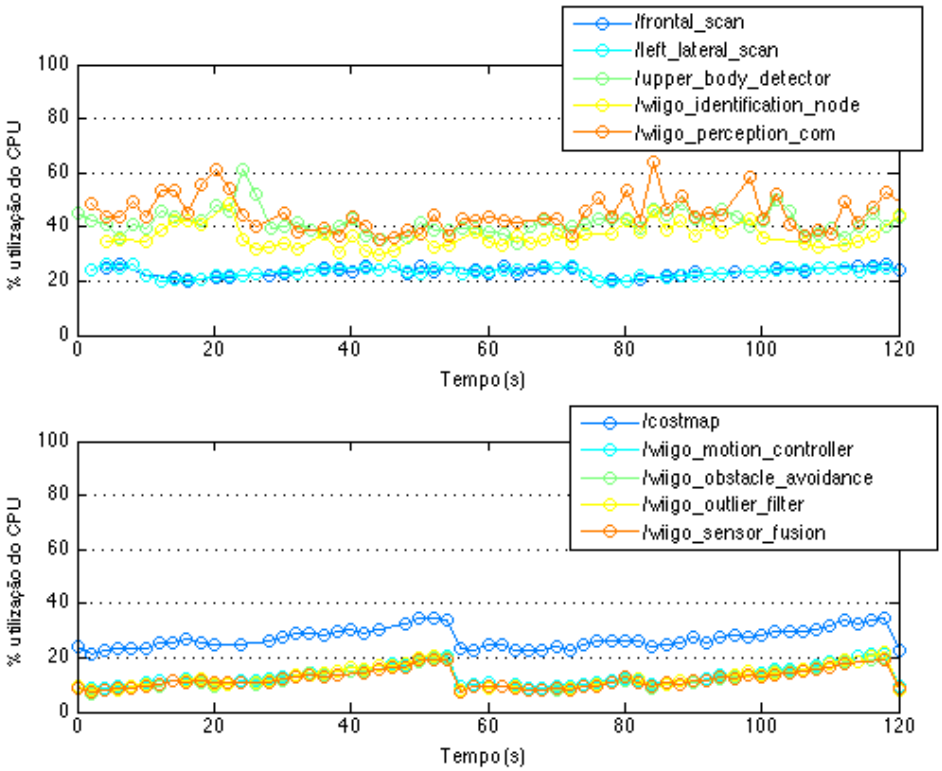


Figura 6.4: *Percentagem de processador usada por cada um dos nodes, gráfico superior I5-4460 e inferior I5-4258U, usando o pacote multimaster.*

6.3.2 Frequência dos tópicos

Nesta situação usou-se para comparação a configuração usada na experiência para medição de performance na arquitetura 2 com um único *master*. Foram escolhidos estes resultados de comparação para provar que o pacote *multimaster fki* tem um impacto reduzido na performance do sistema. Como se poder ver na Figura 6.5 a diferença entre as frequências médias dos tópicos com um único *master* em relação à situação com o uso do pacote *multimaster* é muito reduzida.

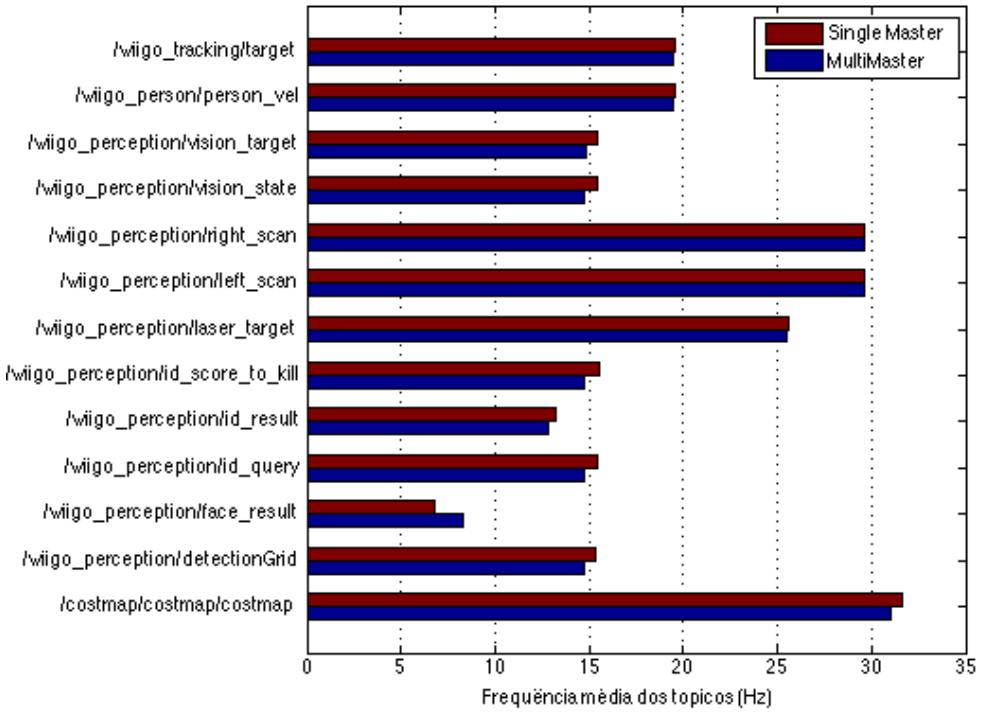


Figura 6.5: *Comparação das frequências dos tópicos na arquitetura 2 com um roscore vs dois roscore usando o MultiMaster.*

6.3.3 Mensagens Perdidas

Tabela 6.1: *Percentagem media e desvio padrão das mensagens perdidas nos tópicos.*

	% média de mensagens perdidas	σ
/person_stopped	37.499	0.326
/scan	14.172	11.680
/vision/ground_plane	31.598	3.897
/wiigo_active_camera/tilt_current_pos	20.596	17.234
/wiigo_odometry/odom	33.164	0.546

6.4 Arquitetura 1

Nesta secção vão ser apresentados os resultados correspondentes a cada um dos processadores nas experiências que utilizaram a arquitetura 1.

6.4.1 Perceção no I5-4258U

Devido aos problemas gerados pelo bug na ferramenta de estatísticas do ROS o sistema ia abaixo quando eram executados todos os *nodes* de perceção no segundo computador.

Para tentar resolver este problema foi decidido trocar os dois *nodes* que regularmente iam abaixo para a maquina principal. Deste modo a experiência foi realizada com os *nodes /wiigo_identification_node* e */wiigo_perception_com* no computador do wiigo (I5-4460) ao contrario do que foi anteriormente definido na arquitetura 2. Este problema foi mais evidente no portátil equipado com este processador (I5-4258U).

Utilização do processador do sistema

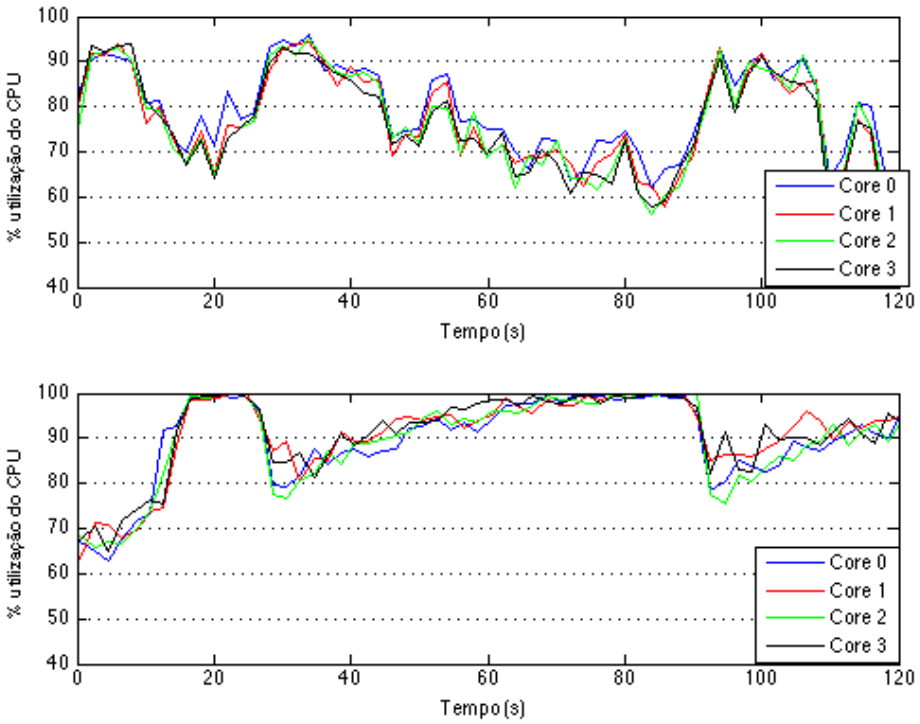


Figura 6.6: *Percentagem de carga em cada um dos núcleos do processador, gráfico superior I5-4258U e inferior I5-4460.*

Utilização do processador pelos *nodes*

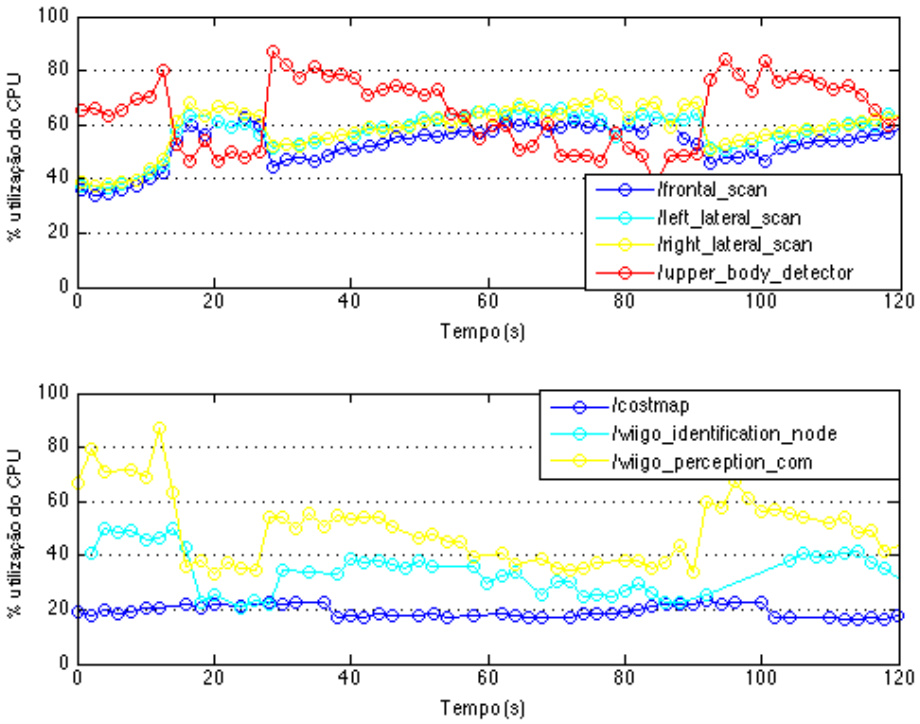


Figura 6.7: *Percentagem de processador usada por cada um dos nodes, gráfico superior I5-4258U e inferior I5-4460.*

Frequência dos tópicos

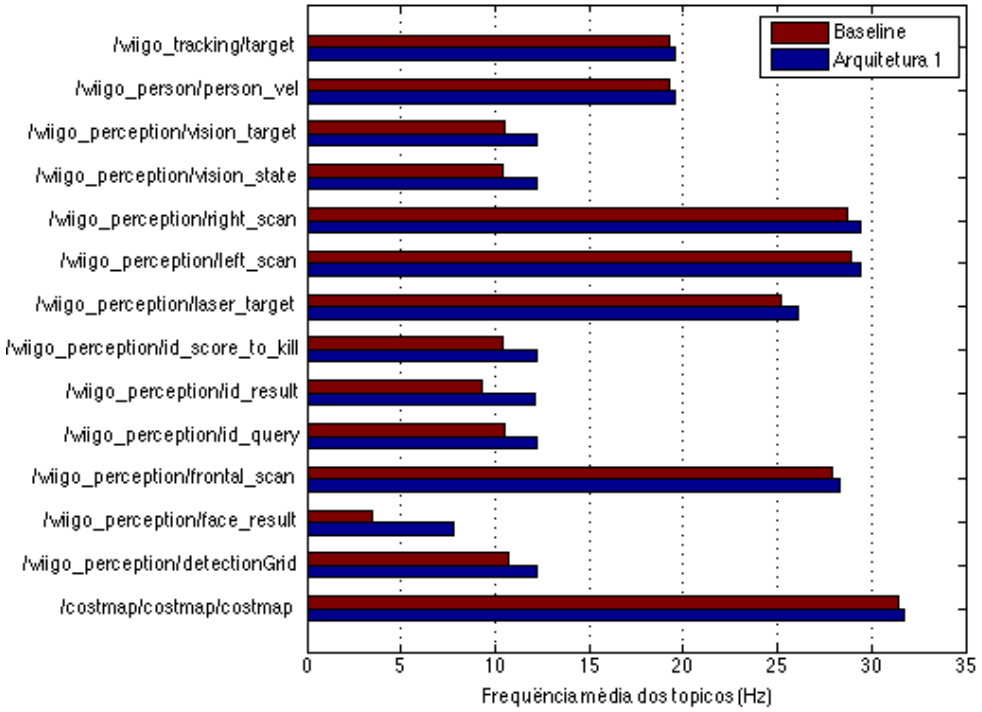


Figura 6.8: *Comparação das frequências em modo de seguimento com a baseline.*

Mensagens Perdidas

Tabela 6.2: *Percentagem media e desvio padrão das mensagens perdidas nos tópicos.*

	% média de mensagens perdidas	σ
/person_stopped	37.491	0.310
/scan	18.912	7.376
/vision/ground_plane	23.053	7.722
/wiigo_active_camera/tilt_current_pos	21.665	15.266
/wiigo_odometry/odom	33.219	0.412

6.4.2 Perceção no I3-6100U

Com este processador especifico foi possível efetuar os *logs* de dados sem o sistema ir abaixo.

Utilização do CPU do sistema

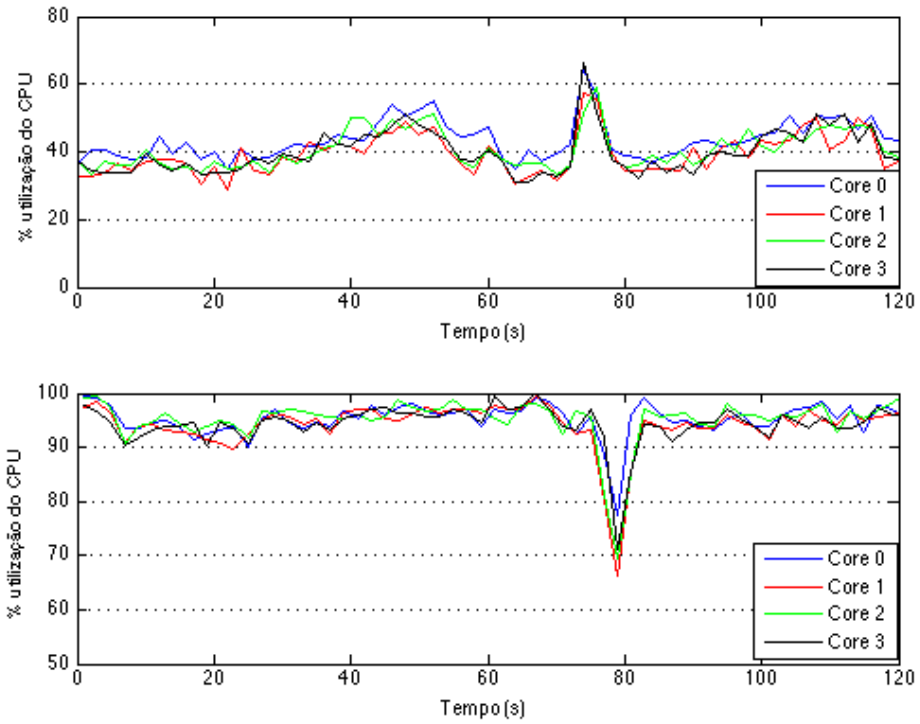


Figura 6.9: *Percentagem de carga em cada um dos núcleos do processador, gráfico superior I5-4460 e inferior I3-6100U.*

Utilização do CPU pelos *nodes*

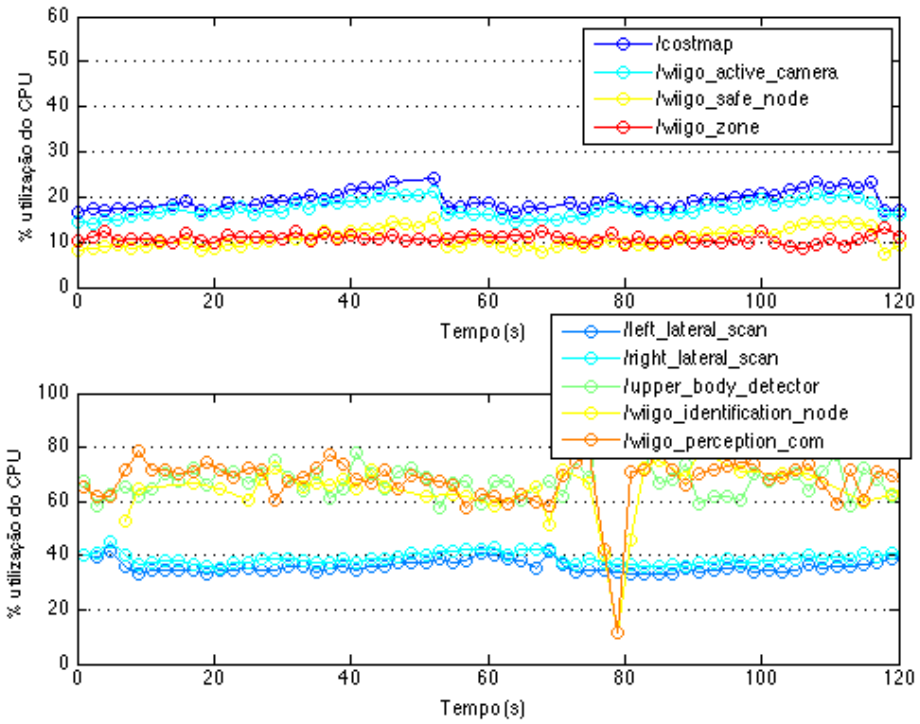


Figura 6.10: *Percentagem de processador usada por cada um dos nodes, gráfico superior I5-4460 e inferior I3-6100U.*

Frequência dos tópicos

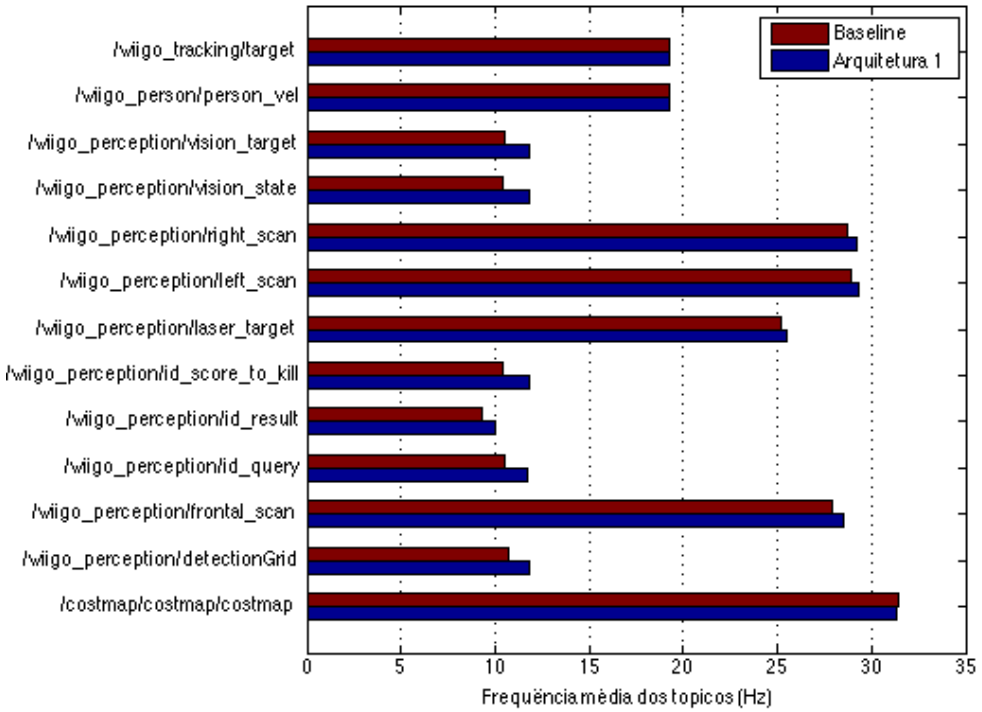


Figura 6.11: *Comparação das frequências em modo de seguimento com a baseline.*

Mensagens perdidas

Tabela 6.3: *Percentagem média de mensagens perdidas e desvio padrão*

	% média de mensagens perdidas	σ
/person_stopped	37.501	0.275
/scan	19.150	7.402
/vision/ground_plane	33.617	5.435
/wiigo_active_camera/tilt_current_pos	20.535	18.599
/wiigo_odometry/odom	33.006	0.709

6.5 Arquitetura 2

Nesta secção vão ser apresentados os resultados correspondentes a cada um dos processadores nas experiências que utilizaram a arquitetura 2.

6.5.1 Navegação no I5-4258U

Utilização do CPU do sistema

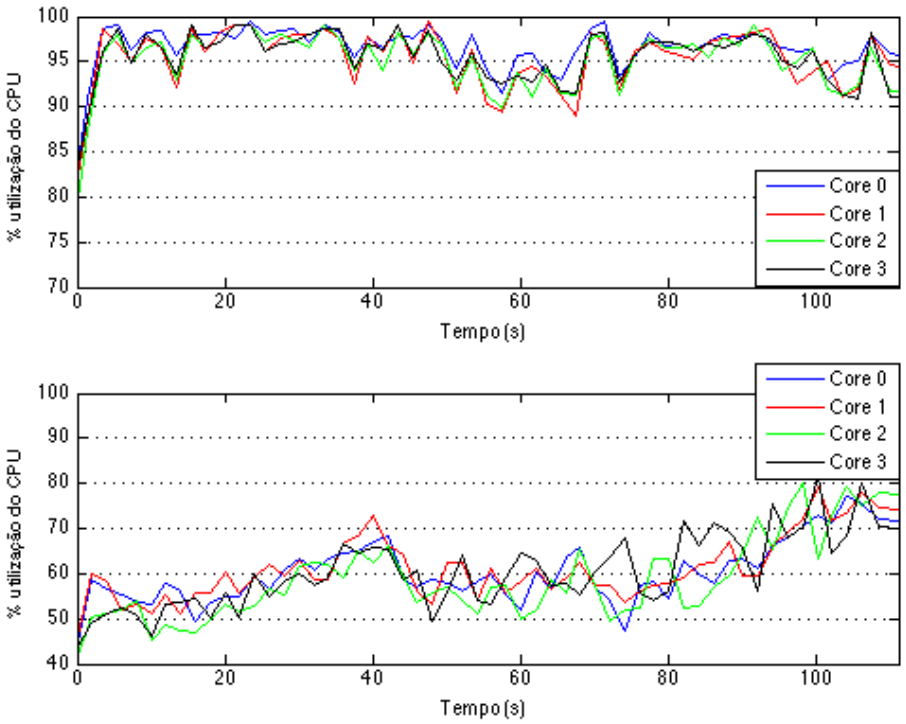


Figura 6.12: *Percentagem de carga em cada um dos núcleos do processador, gráfico superior I5-4258U e inferior I5-4460.*

Utilização do CPU pelos *nodes*

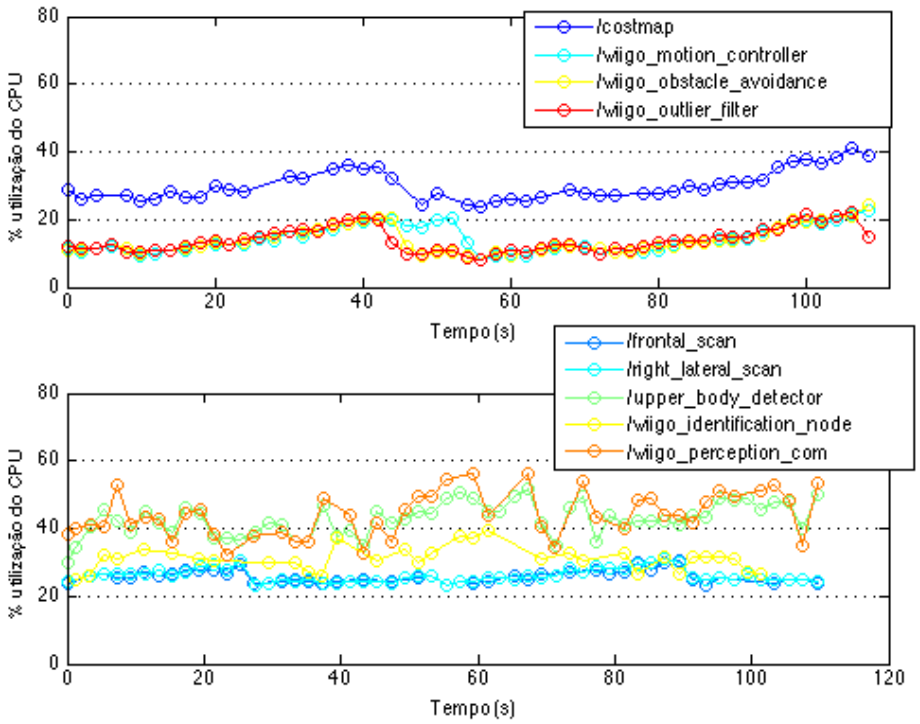


Figura 6.13: *Percentagem de processador usada por cada um dos nodes, gráfico superior I5-4258U e inferior I5-4460.*

Frequência dos tópicos

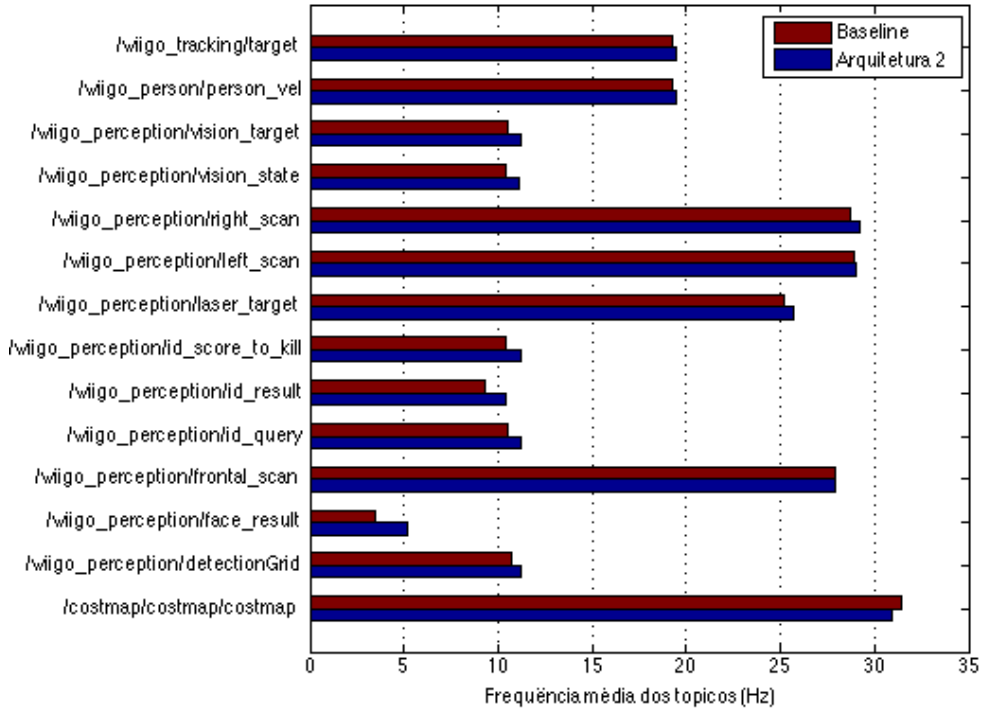


Figura 6.14: *Comparação das frequências em modo de seguimento com a baseline.*

Mensagens Perdidas

Nesta situação e em relação à *baseline*, a percentagem de mensagens perdidas foi significativamente menor, como se pode observar na tabela 6.4.

Tabela 6.4: *Percentagem media e desvio padrão das mensagens perdidas nos tópicos.*

	% média de mensagens perdidas	σ
/person_stopped	19.066	18.830
/scan	10.700	11.541
/vision/ground_plane	22.440	16.769
/wiigo_active_camera/tilt_current_pos	17.667	17.357
/wiigo_odometry/odom	22.219	15.659

6.5.2 Navegação no I3-6100U

Utilização do CPU do sistema

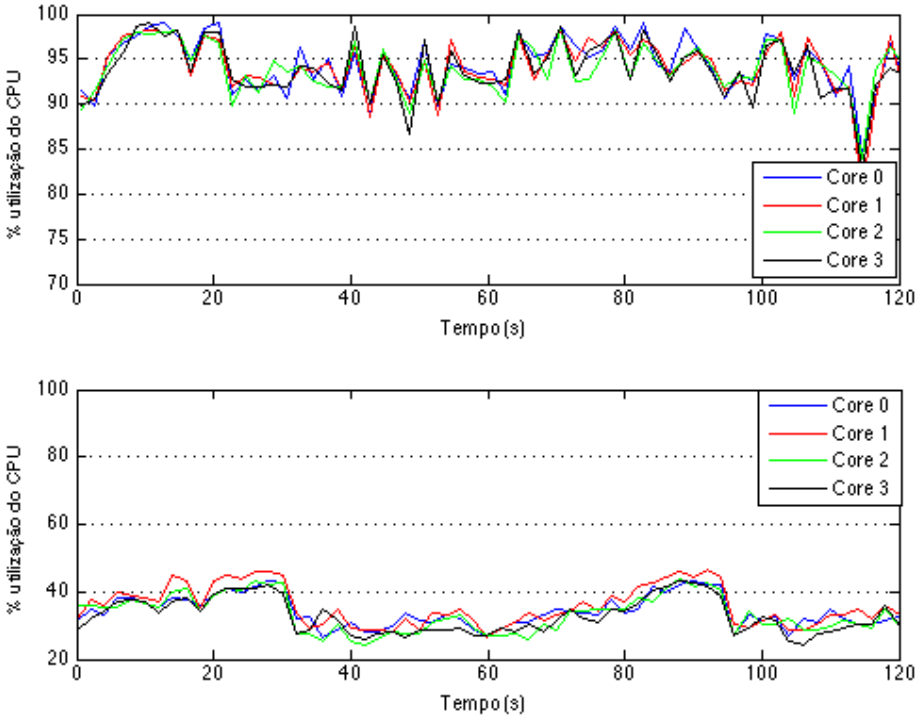


Figura 6.15: *Percentagem de carga em cada um dos núcleos do processador, gráfico superior I5-4460 e inferior I3-6100U.*

Utilização do CPU pelos *nodes*

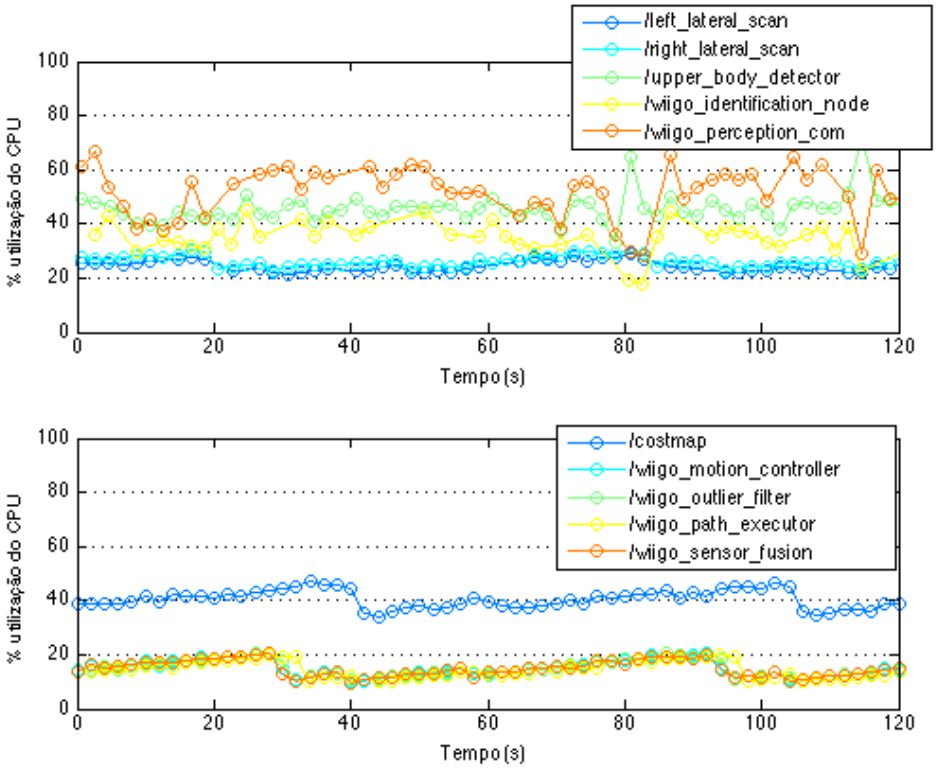


Figura 6.16: *Percentagem de processador usada por cada um dos nodes, gráfico superior I5-4460 e inferior I3-6100U.*

Frequência dos tópicos

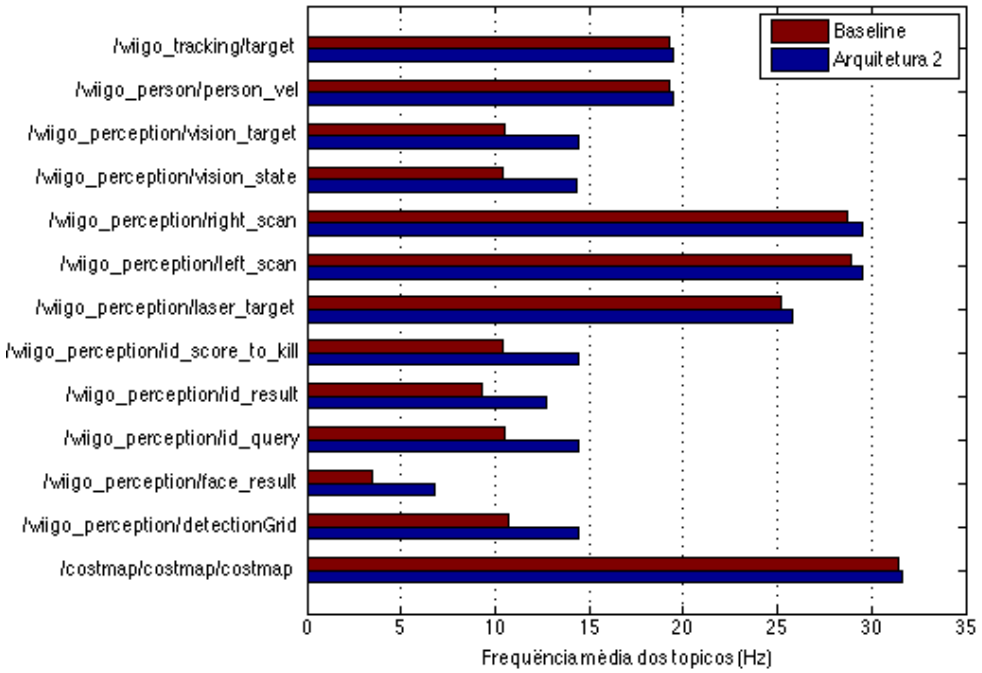


Figura 6.17: *Comparação das frequências em modo de seguimento com a baseline.*

Mensagens perdidas

Tabela 6.5: *Percentagem media de mensagens perdidas e desvio padrão*

	% média de mensagens perdidas	σ
/person_stopped	37.487	0.326
/scan	13.084	11.959
/vision/ground_plane	32.251	5.174
/wiigo_active_camera/tilt_current_pos	24.949	17.744
/wiigo_odometry/odom	33.225	0.435

6.5.3 Navegação no Intel Celeron J1900

Nesta experiência foi testada a viabilidade da utilização de um processador de mais baixa gama para processar a parte da navegação do robô. Não foi possível executar a perceção neste computador devido à sua baixa capacidade de processamento.

Utilização de CPU do sistema

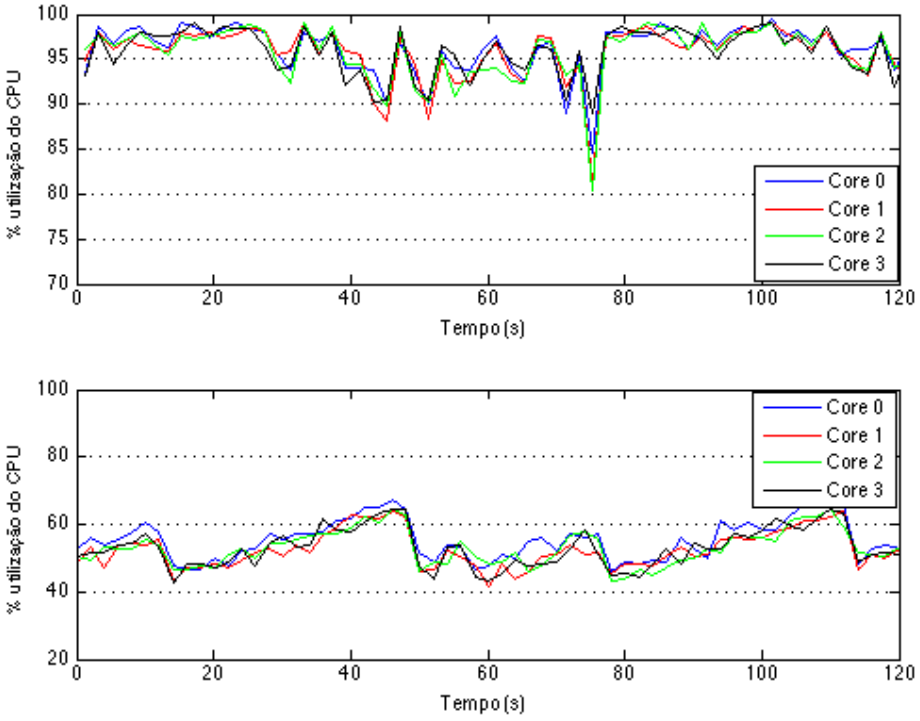


Figura 6.18: *Percentagem de carga em cada um dos núcleos do processador, gráfico superior I5-4460 e inferior Celeron J1900.*

Utilização do CPU pelos *nodes*

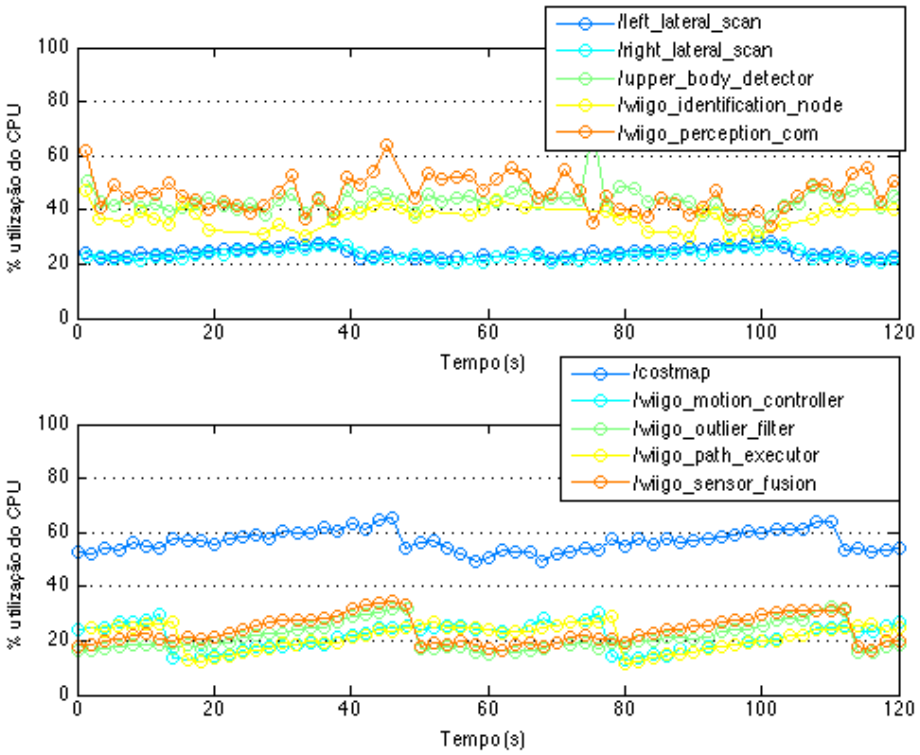


Figura 6.19: *Percentagem de utilização do CPU em cada um dos nodes, gráfico superior I5-4460 e inferior Celeron J1900.*

Frequência dos tópicos

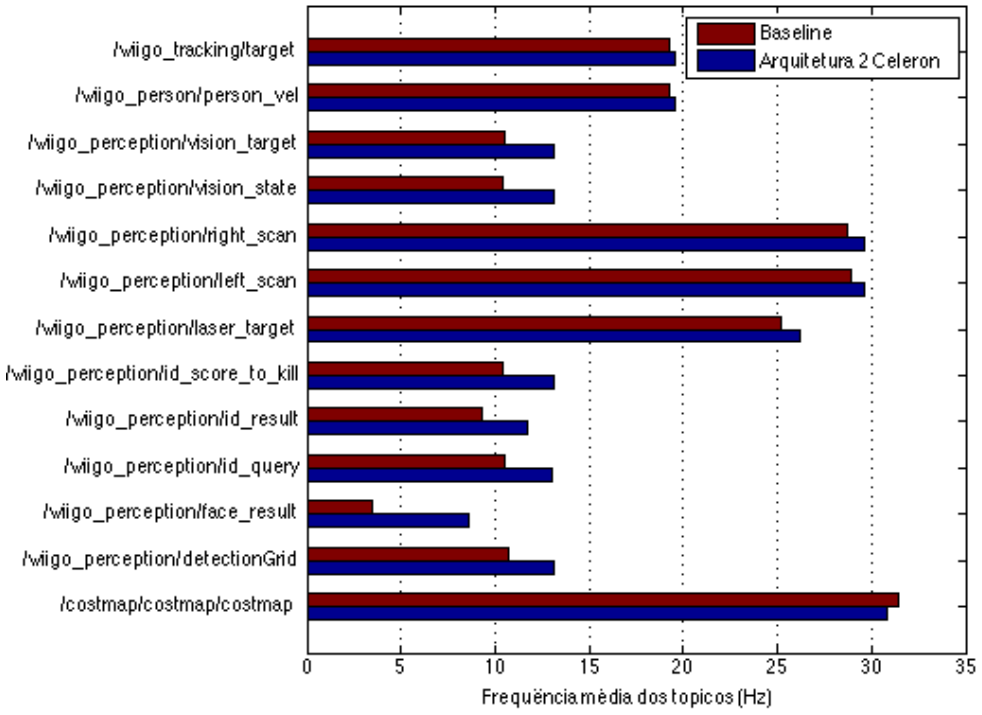


Figura 6.20: *Comparação das frequências em modo de seguimento com a baseline.*

6.5.4 Mensagens perdidas

Tabela 6.6: *Percentagem media de mensagens perdidas e desvio padrão*

	% média de mensagens perdidas	σ
/person_stopped	37.491	0.272
/scan	12.492	11.972
/vision/ground_plane	33.342	5.133
/wiigo_active_camera/tilt_current_pos	25.087	17.829
/wiigo_odometry/odom	33.183	0.440

6.5.5 Navegação no I5-4258U com comunicação por DDS

Nesta arquitetura cada uma das maquinas é independente e não há comunicação TCP ROS entre elas. Isto significa que para registar os logs foi necessário guardar os dados individualmente em cada maquina. Como nesta situação existem dois *roscore* independentes vão existir também dois tópicos */statistics* e duas instâncias do *rosprofiler*.

O *script matlab* para este caso teve de ser ligeiramente modificado para poder lidar com o dobro de ficheiros e sincronizar os tempos entre eles.

Utilização de CPU do sistema

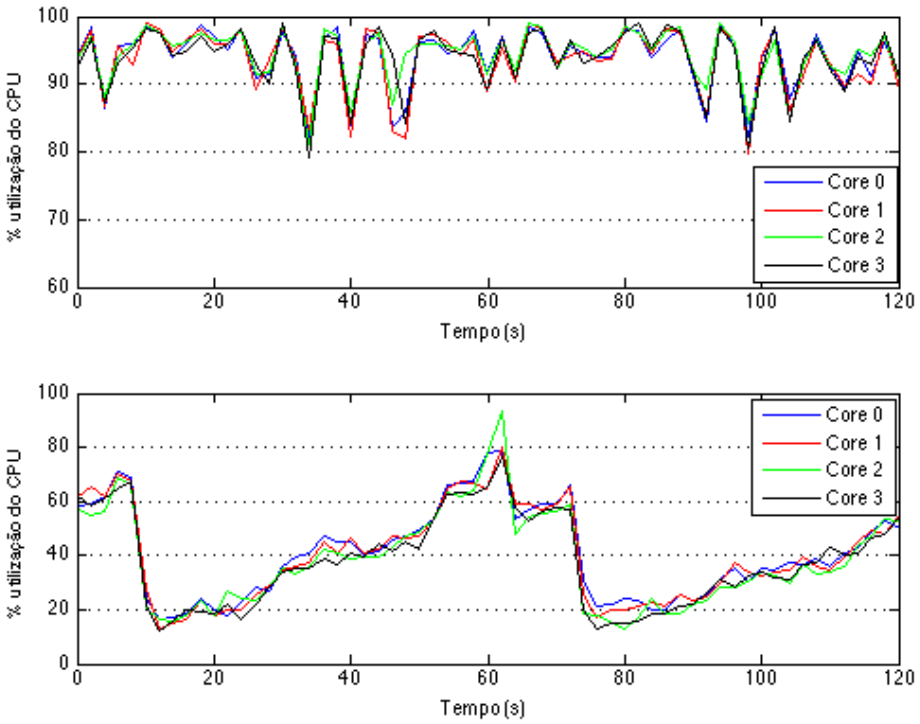


Figura 6.21: *Percentagem de utilização de processador na arquitetura com DDS, gráfico superior I5-4460 e gráfico inferior I5-4258U.*

Utilização do CPU pelos *nodes*

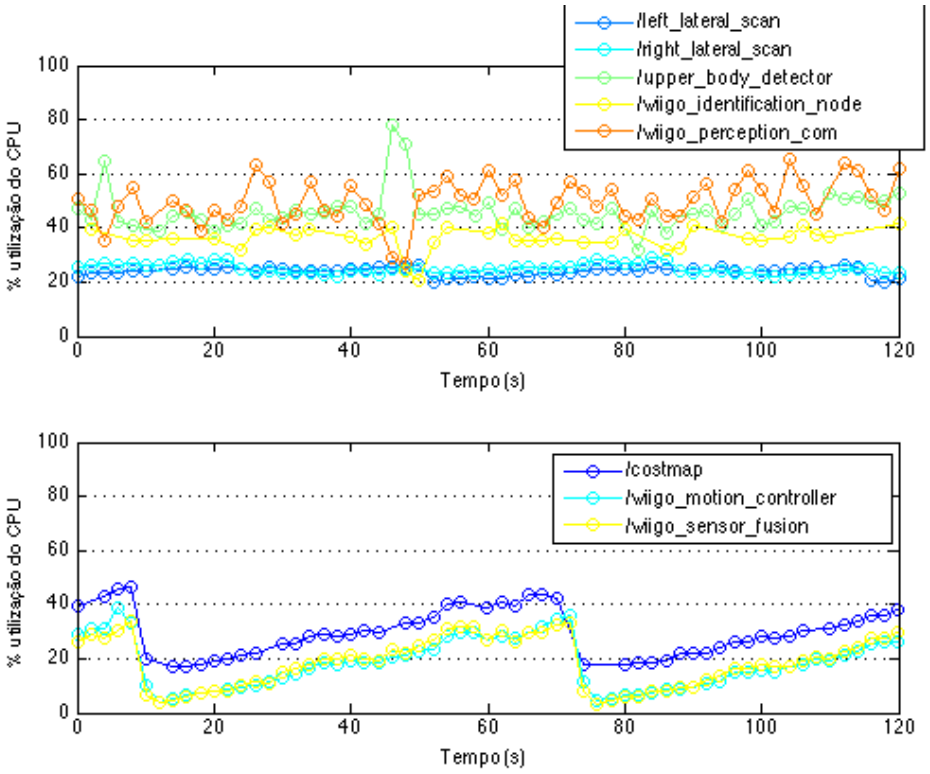


Figura 6.22: *Percentagem de processador usada por cada um dos nodes, gráfico superior I5-4460 e gráfico inferior I5-4258U.*

Frequência dos tópicos

A arquitetura 2 com DDS foi comparada com a arquitetura 2 que usa o mesmo processador secundário *I5-4258U*

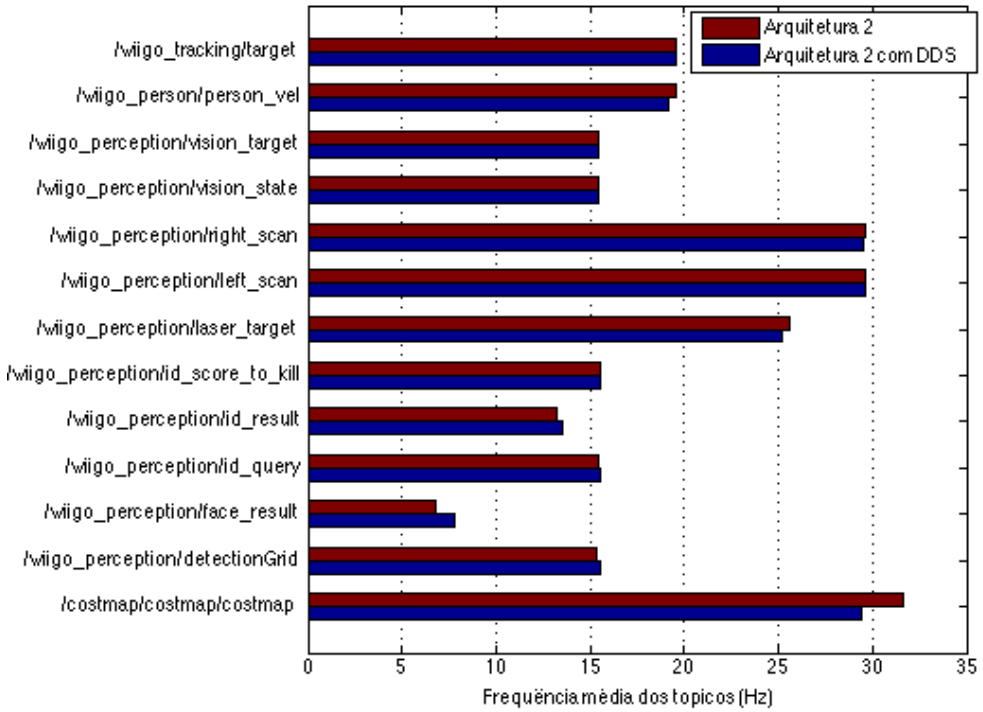


Figura 6.23: *Frequências dos tópicos na arquitetura 2 com DDS e sem DDS.*

6.6 Teste final

Como o sistema estava demasiado sobrecarregado não foi possível executar o *rosprofiler* para medir a utilização do processador. Por este motivo vão ser apresentados apenas os resultados do tópico *statistics*.

Nesta experiência notou-se uma melhoria significativa da performance quando a estatísticas do ROS foram desligadas. Sem as estatísticas ligadas foi possível ao robô seguir o utilizador normalmente com apenas mais algumas perdas do que o normal.

Com as estatísticas ligadas é possível observar uma degradação da performance com mais perdas do utilizador. Isto deve-se ao fato de que os processadores dos dois computadores estarem sobrecarregados quando o robô se encontra em modo de seguimento.

6.6.1 Frequências

Observando o gráfico da figura é possível observar uma diminuição significativa da frequência do *costmap*.

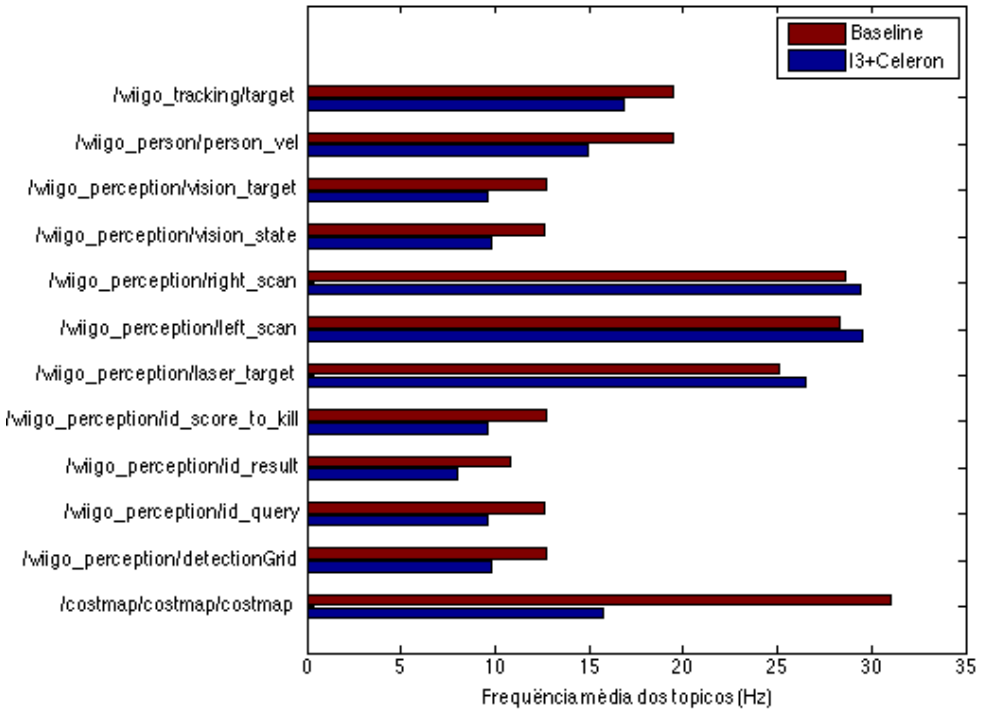


Figura 6.24: *Comparação das frequências dos tópicos nesta configuração com a montagem original do Wiigo.*

Mensagens Perdidas

O numero de mensagens perdidas 6.7 é ligeiramente superior ao original.

Tabela 6.7: *Percentagem média de mensagens perdidas e desvio padrão.*

	% média de mensagens perdidas	σ
/person_stopped	37.571	1.815
/scan	12.071	11.244
/vision/ground_plane	34.270	6.491
/wiigo_active_camera/tilt_current_pos	25.549	17.540
/wiigo_odometry/odom	35.915	4.393
/wiigo_velocity_control/cmd_vel	21.891	6.674

6.7 Proposta de solução final

Após o estudo dos resultados relativos à performance de cada um dos processadores testados chegou-se à conclusão que a melhor opção passa por uma versão modificada da

arquitetura 1, semelhante à configuração que foi abordada no capitulo X.

No entanto o volume livre disponível na cabeça do wiigo é relativamente reduzido o que limita o conjunto de computadores compatíveis.

Após uma pesquisa aprofundada de todos os *mini-pc* disponíveis no mercado decidiu-se que a gama *Intel NUC* oferece uma boa relação qualidade/preço. Os *NUC* estão disponíveis em tamanhos reduzidos havendo até a opção de comprar apenas a placa sem caixa externa.

Para ficar responsável por todo o processamento de imagem a máquina escolhida foi o *Intel NUC KIT NUC7i3BNK* [48] este mini-pc possui um processador *I3-7100U* ligeiramente superior ao do *asus VivoPC* usado na experiência de laboratório. Este computador tem disponíveis quatro portas *USB 3.0*, uma porta Ethernet, modem wi-fi integrado e possui dimensões de apenas 115 mm x 111 mm x 35 mm.

De maneira a avaliar se de facto este computador poderia ser instalado na cabeça do robô foi criado um modelo 3D do espaço livre disponível. Com recurso aos desenhos 3D disponibilizados pela *Intel* [49] dos computadores NUC foi possível encontrar a posição ideal para a sua instalação no wiigo, Figura 6.25. Numa implementação prática seria depois necessário desenvolver um suporte que fixava à estrutura da cabeça do wiigo e que encaixava nas fixações da base do NUC.

Na experiência prática o processador *Celeron* foi o elo mais fraco, diminuindo significativamente a frequência do *costmap*. Para evitar este problema foi escolhido um processador mais recente *Celeron J3455* que tem cerca de 62 % mais performance *multi-core* do que o *Celeron J1900*. Este processador vem integrado na *motherboard Gigabyte J3455N-D3H*, esta placa-mãe (*mini-itx*) foi escolhida por ser compatível com o encaixe e com a alimentação existentes no wiigo. Outra das vantagens desta placa é a existência de duas portas *ethernet*, o o que permite eliminar a necessidade de um *switch* ou adaptador para ligar o LIDAR.

As especificações técnicas e resultados retirados do site do *geekbench* destes dois processadores podem ser encontradas na Tabela 6.10. Nas varias experiências verificou-se que a memoria RAM ocupada pelo sistema nunca chegava aos 4 GB. Isto permitiu que a RAM escolhida para cada um dos dois computadores fosse de apenas 4 GB.

No final fez-se uma comparação dos custos desta nova configuração com os custos atuais do hardware de processamento do wiigo. Estes preços foram retirados à data de escrita desta tese de uma das lojas de informáticas mais conhecidas do país. Tal como se pode ver na tabela xxx a configuração com dois computadores custa mais 261.9 €do que a configuração original do wiigo. No entanto o *TDP* dos dois processadores juntos (25 W) é significativamente inferior ao *TDP* do processador principal do wiigo (84 W).

Componentes	preço
Processador Intel I5-4460	171.90 €
Motherboard Asus H81I-PLUS	84.90 €
RAM DDR3 Crucial 8GB	57.90 €
SSD Kingston V300 120GB	61.90 €
TP-Link Wireless N USB	17.90 €
Total	**394.5 €**

Tabela 6.8: *Lista de componentes da configuração atual do Wiigo.*

Componentes	preço
Mini-PC Intel Nuc i3-7100U	364.90 €
RAM DDR4 Crucial 4GB	41.90 €
SSD M.2 Western Digital 120GB	59.90 €
Motherboard Gigabyte J3455N-D3H	96.90 €
RAM DDR3L G.SKILL Ripjaws 4GB	32.90 €
SSD Kingston A400 120GB	59.90 €
Total	**656.4 €**

Tabela 6.9: *Lista de componentes para a configuração modular proposta.*

Tabela 6.10: *Especificações dos processadores e correspondentes resultados no geekbench*

	i3-7100U	**Intel Celeron J3455**
Nº Núcleos	2	4
Nº Threads	4	4
Clock base	2.4 GHz	1.5 GHz
Clock Turbo	-	2.3 GHz
Memoria Cache	3 MB	2 MB
Litografia	14 nm	14 nm
TDP Máx.	15 W	10 W
Gráficos	Intel HD Graphics 620	Intel HD Graphics 500
Data de Lançamento	Q3'2016	Q3'2016
Single core Score	2641	1404
Multi-Core Score	5499	4607

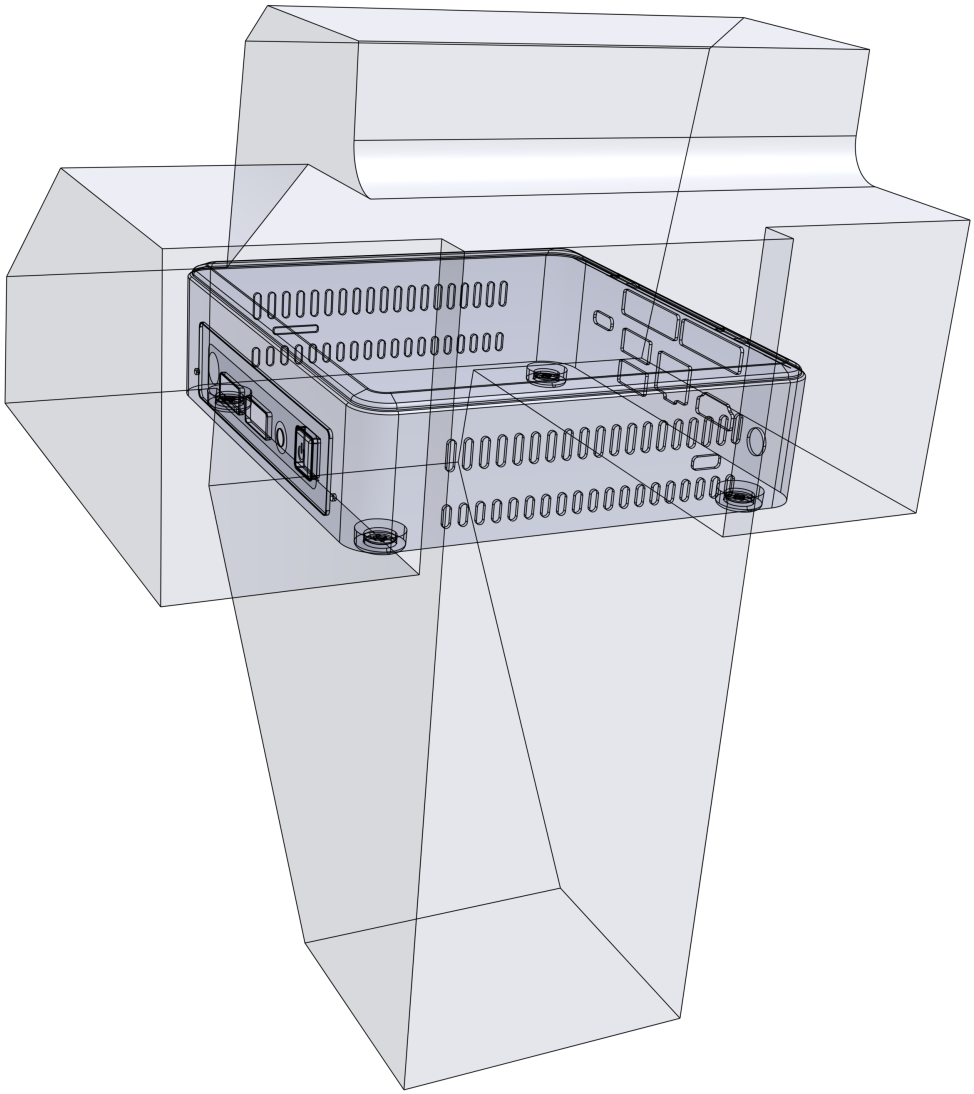


Figura 6.25: *Intel NUC7i3BNK dentro do volume vazio que se encontra atrás do ecrã localizado na cabeça do Wiigo (visto de frente).*

7

Conclusões e trabalho futuro

Uma parte importante deste trabalho foi o estudo dos robôs de serviço existentes no mercado e o estado da arte deste mesmo tipo de robôs. Com o estado da arte dos robôs de serviço foi possível retirar ideias para criação e implementação da arquitetura proposta. Para alem de enriquecer o estado da arte a informação dos robôs de serviço existentes no mercado serviu também como estudo da concorrência para a *Follow Inspiration*.

Foi proposta uma nova arquitetura com duas unidades de processamento e comunicação CAN. Embora o CAN seja um meio de comunicação mais robusto do que o USB (atualmente em uso para todos os sensores do wiigo) a sua implementação iria trazer custos adicionais, nomeadamente no conversor CAN-USB para ligar ao computador e nas mudanças no PCB da *sensor board* que seriam necessárias. A parte do CAN não foi executada porque à altura da implementação do trabalho a empresa não esteve interessada em seguir esta solução. No futuro será interessante implementar a rede de comunicação CAN e comparar a performance com a configuração atual.

O uso das ferramentas de medição de performance ROS permitiu simplificar o possesso de medição do desempenho nas diferentes arquiteturas propostas. Verificou-se que o ARNI é das ferramentas mais completas e de mais fácil utilização. Esta ferramenta tem uma interface gráfica intuitiva e possibilita a definição de contra-medidas associadas a eventos, por exemplo quando a frequência de um tópico se encontra abaixo de um limite pre-definido. No entanto devido a um bug numa biblioteca *python* de que este pacote faz uso as frequências dos tópicos apareceram constantemente todas a zero. Para contornar este problema foi então usado o *rostopic statistics* em conjunto com o *rosprofiler*. O *node_manager* que

vem com o pacote *multimaster fkie* mostrou também ter utilidade para sistemas ROS complexos como é caso do wiigo devido à flexibilidade de funcionar em qualquer configuração e a sua funcionalidade de gestão de *nodes*. Apesar destas ferramentas terem algum impacto na carga do processador, mostraram ser uma boa forma de medir a performance de um sistema ROS. Penso que à medida que o software do wiigo cresce e se torna mais complexo algumas destas ferramentas poderão vir a ser úteis e facilitar a gestão da performance do robô.

Este projeto tinha como objetivo principal a modularização do processamento do wiigo, isto é a divisão do processamento por dois computadores independentes. Com base nos resultados e como seria de esperar verificou-se um ligeiro aumento das frequências dos tópicos ROS para as configurações com o processamento dividido.

Nas pontuações dos processadores obtidas no *geekbench* o I3 e o I5 tinham performance semelhante, como foi depois comprovado nos resultados finais.

Na arquitetura 1 deve-se ao fato dos drivers de aquisição das três câmaras estarem a ser executados no computador do Wiigo o que significa que todas estas imagens estão a ser enviadas pela rede para o segundo PC. Isto resulta numa taxa de transferência de cerca de 45 MB/s entre computadores, que por sua vez tem um impacto negativo na performance. Esta separação dos drivers foi necessária porque as câmaras estão ligadas fisicamente por USB ao wiigo e por motivos de logística não foi possível ligar-las a todos os computadores testados.

Nesta arquitetura a separação dos *nodes* dos drivers de aquisição da imagem do resto do *perception* poderá ter sido uma das razões pela qual os *nodes identification* e *perception_com* iam abaixo quando as estatísticas do ROS estavam ligadas. Mesmo com a sincronização do *chrony* é provável que houvesse situações em que os relógios estavam ligeiramente desfasados o que despoletava o bug no *rostopic statistics*.

Outro problema desta configuração é que mesmo se todos os drivers e o *perception* estiverem no segundo computador todas as imagens continuam a ser transferidas pela rede porque são subscritas pelo tópico *diagnostics* e *UI*.

Na arquitetura 2 apesar de um dos *nodes* mais computacionalmente exigentes (*costmap*) ser passado para o segundo computador continuam a haver poucas mudanças na performance do computador principal, mantendo-se a utilização do CPU à volta dos 98%. Isto deve-se ao facto do processamento mais complexo estar na parte de processamento de imagem e dos *nodes* movidos para o segundo computador terem um custo computacional reduzido.

O *chrony* provou ser uma boa solução para a sincronização dos relógios dos computadores tendo diminuído significativamente os *warnings* de desfasamentos de tempo entre as *tf* ROS do robô.

Verificou-se que para a mesma configuração de hardware e arquitetura o desempenho do sistema com comunicação por DDS foi muito próximo do sistema com comunicação ROS.

Observou-se também que a implementação escolhida não foi a mais rápida devido ao processo de serialização e deserialização efetuado aquando o envio de cada mensagem. Uma implementação mais optimizada passaria pela criação das estruturas das mensagens ROS na linguagem descritiva IDL do DDS. Desta forma as mensagens ROS poderiam ser passadas diretamente para as estruturas de dados DDS sem ser necessario o processo de serialização.

Concluiu-se que nesta situação onde ambos os computadores estão ligados fisicamente por ethernet não existe uma grande vantagem do seu uso para alem do eventual aumento da robustez do sistema a falhas. As situações onde o DDS realmente se destaca são em configurações com ligações de baixa qualidade (por exemplo wi-fi) ou em situações com um elevado numero de sistemas envolvidos.

No futuro o DDS poderá ser implementado no wiigo para comunicação entre robôs. Outra das vantagens deste sistema é o fato de ser multi-plataforma o que permitiria por exemplo a comunicação com uma estação base windows a executar um programa em java que monitoriza em tempo real várias leituras dos robôs em operação. Graças ao *Opensplice mobile* este conceito poderá ser entendido a uma aplicação android que permitiria o wiigo interagir com múltiplos utilizadores ao mesmo tempo.

O estudo do DDS permitui também obtenção de conhecimentos desta tecnologia que é relativamente recente e que será implementada no novo ROS 2.0.

O pacote *multimaster fkie* provou ser também uma solução com pouco impacto na performance do sistema. Devido à facilidade de configuração e à documentação existente online este pacote provou ser uma boa alternativa, pelo facto de permitir múltiplos *roscore*, aumentando a robustez de um sistema ROS com $n + 1$ computadores

Dado que o processador atual se encontra perto do limite da sua capacidade o futuro do wiigo poderá passar por uma atualização do seu CPU principal ou pelo acrescento de um segundo computador. Em termos de custo a solução com dois computadores é significativamente mais cara, tendo como vantagem um consumo energético mais reduzido e não haver a necessidade de utilização de extensões USB. A solução mais simples e mais barata para aumentar o poder de processamento do wiigo passa por fazer uma atualização do processador I5 para por exemplo um I7. A instalação de um segundo computador só será justificada quando se esgotar a capacidade dum processador de alta gama ou se houver uma necessidade especifica de localizar o processamento na parte superior do robô.

No teste final voltou a ser encontrado um problema com os USB das câmaras que é também recorrente no wiigo. De vez em quando uma das câmaras não inicia sendo a

única solução desligar e voltar a ligar-la da porta USB. Na minha opinião isto poderá ser um problema com o controlador USB da *motherboard* que está a trabalhar no limite da sua largura de banda. Uma possível solução poderá passar por comprar uma placa PCI de expansão com entradas USB, desta forma não ficam todas as câmaras e apoiadas no mesmo controlador.

Em suma este projeto permitiu-me aumentar os meus conhecimentos de robótica, programação, redes, ROS e DDS. Ao longo do estágio foram obtidos também conhecimentos sobre a metodologia de desenvolvimento ágil *SCRUM* e do sistema de controlo de versão *git*.

A

Comparação de tempos

Node	Samples	NEW Samples	Wall Time Avg (ms)	NEW Wall Time Avg (ms)	Cpu Time Avg (ms)	NEW Cpu Time Avg (ms)
wiigo_acquisition_astra(acquisition_COM::callback	3241	1500	2.089	1.856	2.291	1.786
wiigo_ground_plane	2047	1500	0.004	0.156	0.003	0.160
Perception_identificationResultsCallback	569	1500	0.004	0.002	0.002	0.002
Perception_imageCallback_info	512	1500	0.005	0.003	0.004	0.002
Perception_lasertrackerCallback	259	1500	0.003	0.003	0.002	0.002
Perception_processa_frames	3669	1500	16.515	29.555	27.187	15.555
wiigo_lateral_scan :: scan_transform	765	1500	0.003	0.047	0.002	0.012
wiigo_scan_fusion :: scan_transform	2107	1500	0.226	0.174	0.234	0.175
wiigo_detection_filter	686	1500	0.106	0.156	0.090	0.160
wiigo_detection_filter_imageCallback_rgb	1124	1500	0.156	0.194	0.158	0.195
wiigo_laser_tracker	4435	1500	2.103	1.853	2.122	1.808
wiigo_odometry	2191	1500	0.019	0.030	0.018	0.031
wiigo_behavior	2246	1500	0.063	0.062	0.061	0.058
wheel_distance_log	415	1500	0.014	0.021	0.013	0.020
wiigo_laser_calibration	470	1500	0.014	0.039	0.012	0.039
scan_to_scan_filter_chain_Callback	2217	1500	1.691	0.972	1.691	0.974
wiigo_maneuver	1198	1500	0.034	0.056	0.033	0.054
wiigo_safe	5436	1500	0.063	0.189	0.063	0.063
wiigo_system_monitor	255	200	1185.107	1179.270	14.682	1.096
path_executor	2772	1500	0.062	0.047	0.063	0.049
wiigo_obstacle_avoidance	2752	1500	0.572	1.564	0.549	0.830
Costmap2d_mapUpdateLoop	1184	1500	8.943	16.276	9.224	11.125
wiigo_outlier_filter	655	1500	0.282	0.100	0.291	0.101
wiigo_zone_sonar	1307	1500	0.022	0.026	0.000	0.025
wiigo_zone_laser	3673	1500	4.172	1.591	4.295	1.418
wiigo_sensors	2040	1500	52.768	47.641	0.186	0.314
roboteq_driver	2030	1500	4.810	5.510	0.070	0.125
wiigo_active_camera	3673	1500	4.172	2.170	4.295	1.378

B

Hardware wiigo

					Sensor board STM32F072x8					
	Orbbec Astra 3D câmara	**Hokuyo UST-10LX**	**Roboteq SDC2130**	**HEDM-5500 Encoder**	**SRF05 Sonar**	**Button**	**Current Sensor ACS711E**	**Emergency Button**	**Dynamixel AX-12A**	**USB2AX**
Alimentação	5V	12V	14V-10V	5V	5V	5V	5V	5V	12 V	5V
Interface de dados	USB 2.0	Ethernet	**USB 2.0**/RS232/CAN	Digital	Digital	Digital	Analog	Digital	RS485	USB 2.0
Resolução	640*480 @ 30FPS	angular 0.25°	-	1024 P/R (4096?)	0.3cm	-	68 mV/A with Vcc = 5 V	-	300° / 1024	-
Field of view	60°H x 49.5°V	270°	-	-	15°	-	-	-	-	-
Alcançe de deteção	0.6m a 8.0m	0.06m a 10m	-	-	2cm a ~4.5m	-	-	-	-	-
Frequencia de aquisição	30 Hz	40 Hz	50 Hz	100 KHz	40 KHz	-	100 KHz	-	-	-
Accuracy	2-3 cm (1cm a 2m)	±40mm	-	-	-	-	±4%	-	-	-
					17 Hz					
Data Rate	3 * 43,946 MB/s	310 KB/s	115.2 KB/s		16 bytes	1 byte	16 bytes	1 byte	57.6 KB/s	1 MB/s
Total	149,838 MB/s	310 KB/s	115.2 KB/s		360 bytes/s				1 MB/s	

	Motherboard Asus H81I-PLUS		**Intel I5-4460 Quad-Core @3.2 GHz**
Dimenções	Formato Mini ITX (17 cm x 17 cm)	*Nº Núcleos*	4
Chipset	Intel H81	*Nº Threads*	4
Memória	2 x U-DIMM, Max. 16GB, DDR3 1600/1333/1066 MHz Non-ECC, Un-buffered Dual chanel	*Clock base*	3.2 GHz
Slot de expansão	1 x PCIe 2.0 x16	*Clock Turbo*	3.4 GHz
Armazenamento	2 x Porta(s) SATA 6Gb/s + 2 x Porta(s) SATA 3Gb/s,	*Memoria Cache*	6 MB
LAN	Realtek® RTL8111G, Gigabit ethernet	*Instruções/Extens*	SSE 4.1/4.2 AVX 2.0
Áudio	Realtek® ALC887 8-Channel High Definition Audio	*Litografia*	22 nm
Portas USB	4 x USB 3.0 port(s) (2 no painel traseiro, 2 na placa) *1x xHCI controller*	*TDP Máx.*	84 W
	8 x USB 2.0/1.1 port(s) (4 no painel traseiro, 4 na placa) 2x *EHCI controller*	*Gráficos*	Intel® HD Graphics 4600
Portas I/O painel traseiro	1 x Porta(s) PS/2 combo teclado/rato		
	1 x DVI-D		
	1 x D-Sub	SSD crucial 120 GB	
	1 x HDMI		
	1 x Porta(s) LAN (RJ45)		
	4 x USB 2.0		
	2 x USB 3.0		
	3 x Audio jack(s)		
I/O Interno	1 x Conector USB 3.0 suporta mais 2 USB 3.0		
	2 x USB 2.0 Conectores suportam mais 4 USB 2.0		
	2 x Conectores SATA 6Gb/s		
	1 x TPM header		
	2 x Conector(es) SATA 3Gb/s		
	1 x Conector(es) CPU Fan (1 x 4 -pin)		
	1 x Conector(es) Chassis de ventoinha (1 x 4 -pin)		
	1 x Saída S/PDIF header(s)		
	1 x Conector(es) 24-pin EATX		
	1 x Conector(es) 4-pin ATX 12V		
	1 x Conector(es) no painel frontal		
	1 x Conector(es) de colunas internas		
	1 x Painel de Sistema		
	1 x Clear CMOS jumper(s)		
BIOS	64 Mb Flash ROM, AMI BIOS, PnP, DMI2.0, WfM2.0, SM BIOS 2.7, ACPI 2.0a		

C

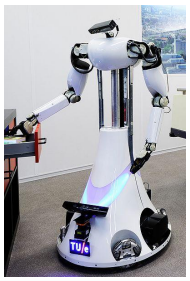
Lista de comparação de robôs

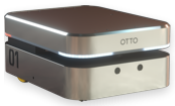
					
Rating Similaridade com wGO[0-10]	5	4	6	3	7
Fabricante	Panasonic	Knightscope Inc.	Fetch Robotics	Fraunhofer IPA	Savioke
Modelo	AP-3020A003	K5	hmishelf15	Care-O-bot 4	Relay
Dimensão (LxWxH) [mm]	630 x 725 x 1386	813 x 914 x 1524	559 x 559 x 359	720 x 720 x 1580	510 x 510 x 920
Peso (Com baterias) [Kg]	170	136	68	140	40.8
Autonmia [h]	9	?	8-10	?	4 (continous)
Tipo de bateria	lead Batery	?	Sealed Lead Acid	Li-Ion 48V	?
Tempo de carga [h]	4,5	?	3.5 (90%)	?	?
Volume de carga (LxWxH) [mm]	350 x 440 x 390	x	508 x 559 x 939.8	x	21 litros
Capacidade de carga [Kg]	20	x	80.7	x	4.5 Kg
v_max [m/s]	1	1.34	2.0	1.1	0.7
Rotação da cabeça	Sim	Não	Não	Sim	Não
Camaras laterais	Sim	Sim	Não	Não	Não
Camara central	Sim	Sim	Sim RGB-D	Sim	Sim
Nº total de Camaras	~4	~3	1	3	2
Tipo de tração	Diferencial	Diferencial	Diferencial	Omnidirectional	Diferencial
Nº LIDAR	1	2	1	3	1
Modelo Lidar	?	Velodyne VLP-16	SICK TIM571	?	?
IMU (Inertial Measurement Unit)	?	Sim	Sim	?	Sim
Ultrasonic Sensors	Não	Sim	Não	Não	Sim
Microfones	Sim	Sim	Não	Sim	Não
Monitor	Sim	nao	sim (opcional)	Sim (touch)	Sim (touch)
Nº de PC's	?	?	1	4-6	1
Outros Sensores	Fingertip sensor	Camara Termografica, GPS	x	x	Safety bumper
Áreas de aplicação	Hospitais	Retalho	Industria	Domestico	Retalho

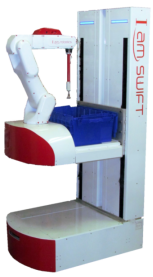
	Referencias
Panasonic AP-3020A003	http://news.panasonic.com/global/topics/2015/44009.html
Knightscope K5	https://en.wikipedia.org/wiki/Knightscope
Fetch Robotics hmishelf15	http://fetchrobotics.com/wp-content/uploads/2017/02/Fetch_spec_download_2016.pdf
Care-O-bot 4	http://www.care-o-bot.de/en/care-o-bot-4/technical-data.html
Savioke Relay	https://www.hartrobotics.com/relay/#tab-1452592393685-99252-8383
Aethon TUG T3	http://www.aethon.com/wp-content/uploads/2015/09/DatasheetT3_1.pdf
Locus Robotics LocusBot	http://www.locusrobotics.com/features/autonomous-robots/
Keonn AdvanRobot	https://www.keonn.com/systems/view-all-2/inventory-robots.html
PAL StockBot	https://www.pal-robotics.com/en/products/stockbot/
Fellow Navii	http://www.fellowrobots.com/#NAVii™
MetraLabs TORY	http://www.metralabs.com/shopping-rfid-robot
Simbe Robotics Tally	http://www.simberobotics.com/
Cobalt Robotics	https://www.cobaltrobotics.com/
5 Elements Robotics	http://Selementsrobotics.com/dash-robotic-shopping-cart/
Toyota HSR	http://www.toyota-global.com/innovation/partner_robot/family_2.html
AMIGO	http://roboticopenplatform.org/wiki/AMIGO
Piaggio Gita	http://piaggiofastforward.com/gita
Metra Labs WeRobot	http://www.attraktion.com/sites/default/files/WeRobots_English.pdf
Starship	https://www.starship.xyz/
Marble	https://www.marble.io/
VGo	https://www.vecna.com/vgo-telepresence/
Beam Max	http://awabot.com/en/telepresence-robots/beam-plus-max
OTTO 100	https://www.ottomotors.com/otto100
Arkrobot	http://www.arkrobot.com/
Chuck	http://6river.com/
MIR100	http://www.mobile-industrial-robots.com/products/mir100
GP8	https://seegrid.com/wp-content/uploads/2016/09/gp8_product_spec_brochure.pdf
Swift	https://www.iamrobotics.com/solutions?hsCtaTracking=6396be5d-3ad8-4563-b373-ff0acdd9fcf8%7C811a7b12-3996-4069-b577-409b7a63c10b#swi
Lynx	http://www.adept.com/products/mobile-robots/mobile-platforms/lynx/general
Peper	http://doc.aldebaran.com/2-0/family/juliette_technical/index_juliette.html
SPENCER	http://www.spencer.eu/project.html
Loomo	http://www.segwayrobotics.com/product
OSHbot	http://www.lowesinnovationlabs.com/innovation-robots/
Effibot	http://www.effidence.com/effibot

						
Rating Similaridade com wGO[0-10]	5	6	7	7	7	7
Fabricante	Aethon	Locus Robotics	Keonn	PAL Robotics	Fellow Robots	MetraLabs
Modelo	TUG T3	LocusBot	AdvanRobot	Stockbot	NAVii	TORY
Dimensão (LxWxH) [mm]	1164 x 570 x1200	?	500 x 500 x 2082.75	500 x 500 x 1900	457.2 x 685.8 x 1524	500 x 500 x 1500
Peso (Com baterias) [Kg]	x	36	64	?	90.71	60
Autonmia [h]	10	10	12	8	?	18
Tipo de bateria	Valve-Regulated Lead-Acid	?	Li-Ion 24V	Li-Ion	?	?
Tempo de carga [h]	x	0.5-1	x	4	?	6
Volume de carga (LxWxH) [mm]	Modular	?	x	x	x	x
Capacidade de carga [Kg]	453	18-45	x	x	x	x
v_max [m/s]	0.762	2	1	1	?	1
Rotação da cabeça	Não	Não	Não	não	Não	Não
Camaras laterais	Não	Não	Não	não	Sim	Não
Camara central	Não	Sim	Sim	sim	sim	Sim
Nº total de Camaras	0	1	1	2	3	1
Tipo de tração	Omnidirectional	Diferencial	Diferencial	Diferencial	Diferencial	Diferencial
Nº LIDAR	1	1	1	1	1	1
Modelo Lidar	?	?	?	?	?	?
IMU (Inertial Measurement Unit)	?	?	não	não	?	?
Ultrasonic Sensors	Sim	não	sim	sim	não	?
Microfones	Não	Não	não	não	não	não
Monitor	Não	Sim (ipad)	não	não	sim (touch)	não
Nº de PC's	1	?	?	?	?	?
Outros Sensores	Bar code scanner, RFID, Fingerprint	x	RFID antenas	RFID Antenas	x	RFID Antenas e NFC
Áreas de aplicação	Hospitais	Industria	Retalho	Retalho	Retalho	Retalho

ift

						
Rating Similaridade com wGO[0-10]	7	4	8	3	3	9
Fabricante	Simbe Robotics	Cobalt Robotics	5 Elements Robotics	Toyota	Eindhoven University of Technology	Piaggio
Modelo	Tally	Cobalt robo-guard	DASH Robotic Shopping Cart	HSR	AMIGO	Gita
Dimensão (LxWxH) [mm]	H 965.2	?	?	430 x 430 x (1005 - 1350)	650 x 650 x (1000-1035)	H 660.4
Peso (Com baterias) [Kg]	13.6	?	?	37	80	?
Autonmia [h]	?	?	?	?	0.5	8
Tipo de bateria	?	?	?	?	Makita Ni-MH (BH2433)	?
Tempo de carga [h]	?	x	?	x	x	?
Volume de carga (LxWxH) [mm]	x	x	?	x	x	32.7 L
Capacidade de carga [Kg]	x	?	?	1.2	1.5	19
v_max [m/s]	?	?	?	0.22	1	9.72
Rotação da cabeça	Não	Não	Não	Sim	Sim	Não
Camaras laterais	Não	Sim	Não	Não	Não	Não
Camara central	Sim	Sim	Sim	Sim	Sim	Sim
Nº total de Camaras	?	4	1	5	2	~4
Tipo de tração	Diferencial	Difrencial	Diferencial	Omnidirectional	Omnidirectional	Diferencial
Nº LIDAR	1	1	1	1	1	0
Modelo Lidar	?	?	?	?	Hokuyo UTM-30LX	x
IMU (Inertial Measurement Unit)	?	?	?	Sim	Não	?
Ultrasonic Sensors	não	Não	Sim	Não	Não	?
Microfones	não	Sim	Não	Sim	Sim	Não
Monitor	Sim	Sim (touch)	Sim (touch)	Sim (touch)	Não	sim
Nº de PC's	?	?	?	?	3	?
Outros Sensores	x	RFID	x	Deteção de fita magnetica	x	Par stereo no utilizador, GPS
Áreas de aplicação	Retalho	Retalho/ escritórios	Retalho	Doméstico	R&D	Uso pessoal

						
Rating Similaridade com wGO[0-10]	5	4	4	5	4	5
Fabricante	Metra Labs	Starship Robots	Marble Robots	Otto Motors	ARKRobot	6 River Systems
Modelo	WeRobot	Starship	Marble	OTTO 100	Small Arkrobot	Chuck
Dimensão (LxWxH) [mm]	?	685.8 x 558.8 x 558.8	?	750 x 550 x 304	?	1016 x 635 x 1244
Peso (Com baterias) [Kg]	?	18	?	127	75	?
Autonmia [h]	12	2-2.5	6-8	8	6	?
Tipo de bateria	?	?	?	?	lithium ion	?
Tempo de carga [h]	6	?	?	?	?	?
Volume de carga (LxWxH) [mm]	x	406.4 x 342.9 x 330.2	?	Variavel	500 x 325 x 200	?
Capacidade de carga [Kg]	x	10	?	100	250	72.57
v_max [m/s]	?	1.67	1.78	2	1.17	?
Rotação da cabeça	Não	Não	Não	Não	Não	Não
Camaras laterais	Sim	Sim	Sim	Não	Não	Não
Camara central	Não	Sim	Sim	Sim	?	Sim
Nº total de Camaras	~1	9	~4	2	?	1
Tipo de tração	Diferencial	6 wheel drive	ackerman	Diferencial	Diferencial	Diferencial
Nº LIDAR	1	0	2	1	1	1
Modelo Lidar	?	-	Velodyne VLP-16 / SICK	safety rated	?	?
IMU (Inertial Measurement Unit)	?	Sim	Sim	?	Não	?
Ultrasonic Sensors	Sim	Sim	Sim	Não	Não	não
Microfones	Sim	Não	Não	Não	Não	Não
Monitor	Sim (touch)	Não	Não	Não	Não	Sim
Nº de PC's	?	? Tegra K1	? Jetson TX1 AI supercomputers	?	?	?
Outros Sensores	RFID, Active Bumper	GPS, 3x time-of-flight cameras	GPS	-	-	-
Áreas de aplicação	Entretenimento/Promoção	Retalho	Retalho	Industrial	Industrial	Industrial

						
Rating Similaridade com wGO[0-10]	6	4	3	6	5	8
Fabricante	MIR	Seegrid	IAM Robotics	Adept	SoftBank Robotics	
Modelo	MIR100	GP8 Pallet Truck	Swift	Lynx	Peper	SPENCER
Dimensão (LxWxH) [mm]	890 x 580 x 352	2275 x 917 x 1981	1107 x 704 x 1981		425 x 485 x 1200	
Peso (Com baterias) [Kg]	62.5	?	181.4	60	28	250
Autonmia [h]	10	?	?	13	12	
Tipo de bateria	Li-NMC, 24 V	? 24V	Lithium Ion	LiFePO4	Lithium-ion	
Tempo de carga [h]	3	?	?	3.5	?	
Volume de carga (LxWxH) [mm]	600 x 800	pallets		?	-	
Capacidade de carga [Kg]	100	3 628	6.8 (arm)	60	?	
v_max [m/s]	1.5	1.78	0.91	1.8	0.89	1.5
Rotação da cabeça	Não	Não	Não	Não	Sim	Sim
Camaras laterais	Não	Sim	Sim	Não	Não	Não
Camara central	Sim	Sim	Sim	Sim	Sim	sim
Nº total de Camaras	1	~4	~3	1	3	6
Tipo de tração	Diferencial	?	Diferencial	Diferencial	Omnidirectional	Diferencial
Nº LIDAR	2	1	?	1	0	3
Modelo Lidar	SICK S300	?	?	?	-	SICK LMS 500 + Velodyne 16
IMU (Inertial Measurement Unit)	?	?	?	?	Sim	Não
Ultrasonic Sensors	Sim	Não	Sim	Sim	Sim	Não
Microfones	Não	Não	Não	Não	Sim	Sim
Monitor	Não	Sim	Não	Não	Sim	Sim
Nº de PC's	?	?	?	?	?	5
Outros Sensores	-	-	-	-	Laser, Bump, touch sensors	Bump, touch sensors
Áreas de aplicação	Industrial	Industrial	Industrial	Industrial	Retalho/Doméstico/ Entretenimento	Aeroportos

D

Small form factor PC boards

	COM Express				
Brand	Eurotech	Eurotech	Eurotech	AAEON	AAEON
Model	CPU-162-22	CPU-161-17	Adbc8037	NanoCOM-SKU	COM-SKHB6
CPU	Intel® 6th Gen Core i5 6442EQ 1.9GHz (2.7GHz), 4 Cores/ i5 6442EQ 1.9GHz (2.7GHz)/ 4 Cores, Celeron G3902E 1.6GHz	Intel® 6th Gen Core i5 6300U 2.4GHz (3.4GHz), 2 Cores/ i7 6600U 2.6GHz (3.4GHz), 2 Cores/ Celeron 3955U 2.0GHz	Intel® 3rd Gen Intel Core i7-3615QE(Quad Core / 2.3 GHz) / i7-3555LE (2 cores, 2.5GHz) /i7-3517UE (2 cores, 1.7GHz)	Intel® 6th Gen U series CPU, i7-6600U/ i5-6300U / i3-6100U	Intel® 6th Gen i7-6820EQ (2.8 GHz) / i5-6440EQ (2.6 GHz) / i3-6100E (2.7 GHz)
Chipset	QM170	Intergrated into SoC	Mobile Intel® QM57 Express	Onboard	PCH QM170 or CM236
RAM	2x SODIMM Sockets - Up to 32GB DDR4 2133MHz	2x SODIMM Sockets - Up to 16GB DDR3L 1600MHz	Direct-mounted DDR3L-1600 8GB, ECC supported	Non-EEC DDR4-2133, 4GB	DDR4 SODIMM x 2 up to 32 GB
SATA	4x SATA 3.0	3x SATA 3.0	2x SATA-300 + 2x SATA-600	2x SATA3 (6.0Gb/s)	4x SATA3 (6.0Gb/s)
Ethernet	1xGigabit Ethernet (iAMT Support)	1x Gigabit Ethernet (iAMT Support)	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet
USB	4x USB 3.0 8x USB 2.0	4x USB 3.0 8x USB 2.0	2x USB 3.0 2x USB 2.0	2x USB 3.0 2x USB 2.0	4x USB 3.0 8x USB 2.0
Serial	2x Serial (TX/RX)	2x Serial (TX/RX)	-	2x 2-wire UART	2x 2-wire UART
Video	1x VGA, 1 x LVDS, 1x DDI (HDMI)	1x VGA, 1 x LVDS, 1x DDI (HDMI)	3x DDI (HDMI), 1x VGA	1x LVDS/eDP, DDI	2x VGA, LVDS/eDP, DDI
PCI Express	1x PCIe x16 or 2x PCIe x8 (Gen 3)	up to 5 Devices and 8 Lanes, Gen 3	1x x16 or 2x x8	1x PCIe x 4,	1x PCIe x 4, 1x PCIe x 16
Power Input	12V, 5VSB,(ATX Mode); 12V, (AT Mode)	12V, 5VSB,(ATX Mode); 12V, (AT Mode)	DC 12V (VCC_12V)	DC 12V	AT/ATX 12 V
Power Consumption	25W (CPU TDP)	10.28 Watt (typ), 16.46W (max)	45W/ 25W /17W (CPU TDP)	18 W	45W (CPU TDP)
Complaiance	PICMG COM Express® R2.1, Type 6	PICMG COM Express® R2.1, Type 6	PICMG COM Express® R2.0, Type 6 & Type 2	COM Express Type 10	COM Express Type 6 Basic
Dimensions (WxL)	95x125mm - COM Express Basic	95x95mm - COM Express Compact	95 x 125 mm	Mini Size, 84mm x 55mm	95 x 125 mm
Link	http://www.eurotech.com/en/products/CPU-162-22	http://www.eurotech.com/en/products/CPU-161-17	http://www.eurotech.com/en/products/Adbc8037	http://www.aaeon.com/en/p/com-express-modules-nar	http://www.aaeon.com/en/p/com-express-modules-co

	Mini-ITX Industrial				
Brand	Eurotech	Adlink	Adlink	AAEON	AAEON
Model	CPU-521-17	AmITX-SL-G	AmITX-HL-G	EMB-H110B	EMB-Q170A
CPU	Onboard Celeron 3955U 2.0GHz, 2 Cores /i5 6300U 2.4 GHz, 2 Cores/ i7 6600U 2.6GHz, 2 Cores	Socket LGA1151 supporting 6th/7th generation Intel® Core™ i7/i5/i3	Socket H3 LGA1150 supporting 4th generation Intel® Core™ i7/i5/i3	Socket LGA1151 supporting Intel® 6th Generation Core™ i7/i5/i3	Socket LGA1151 supporting Intel® 6th Generation Core™ i7/i5/i3
Chipset	Intergrated into SoC	Intel® Q170/H110	Intel Q87/H81	Intel® H110	Intel® Q170
RAM	2x SODIMM Sockets - Up to 16GB DDR3L 1600MHz	Up to 32 GB dual channel DDR4 at 2133MHz	Up to 16 GB dual SODIMM DDR3/DDR3L at 1666/1333 M	DDR4 2133/1867 MHz SODIMM x 2, Up to 32GB	DDR4 2133/1867 MHz SODIMM x 2, Up to 32GB
SATA	2x SATA 3.0	3x SATA 6 Gbps ports	3x SATA 6 Gbps ports	2x SATA3 (6.0Gb/s)	2x SATA3 (6.0Gb/s)
Ethernet	2x RJ-45 Gbit/s (I210AT + I219LM)	2x RJ-45 Gbit/s (i219-LM + i211AT)	1x RJ-45 Gbit/s (i218LM)	2x RJ-45 Gbit/s (realtek 8111G)	2x RJ-45 Gbit/s (219LM + i211AT)
USB	4x USB 3.0 4x USB 2.0 (Header)	4x USB 3.0 and 4x USB 2.0 on rear I/O 2x USB 3.0 onboard header (H110: USB 2.0) 1x USB 3.0 on vertical connector	4x USB 3.0 and 4x USB 2.0 on rear I/O 2x USB 2.0 on front panel header + 2x USB 2.0 (header) 1x USB 2.0 on vertical connector	4x USB 3.0 2x ports support additional 4 USB	4x USB 3.0 (rear panel) 2x USB 2.0 2x ports support additional 4 USB 2.0
Serial	2x RS-232/422/485	1x RS-232/422/485, 3x RS-232	1x RS-232/422/485, 3x RS-232	1x RS-232/422/485, 1x RS-232	1x RS-232/422/485, 1x RS-232
Video	1x DP++, 1 x LVDS, 1x HDMI	3x DisplayPort, 1x LVDS	3x DisplayPort, 1x LVDS	2x HDMI, 1x LVDS, 1x eDP	1x VGA , 3x DP
PCI Express	1x Full/Half Size Mini PCIe (PCIe/USB/mSATA) 1x Full/Half Size Mini PCIe (PCIe/USB), 1x PCIe x4 Gen 3	1 PCIe x16 Gen3 + 1 PCIe x1 Gen2 1 Mini PCIe (full size slot) 1 Mini PCIe (half size slot)	1 PCIe x16 Gen3 + 1 PCIe x1 Gen2 1 Mini PCIe (full size slot) 1 Mini PCIe (half size slot)	1 PCIe x1 Gen2 1 Mini PCIe (full size slot) 1 Mini PCIe (half size slot)	1 PCIe x16
Power Input	12V (DC-in, Screw-lock Jack)	Standard Input: ATX: 12V	Standard Input: ATX: 12V	12V AUX Power Connector x 1	12V AUX Power Connector x 1
Power Consumption	12V @1.58A (18.96W) Max	Up to 65 W (CPU)	Up to 65 W (CPU)	Up to 65 W (CPU)	Up to 65 W (CPU)
Complaiance	Mini-ITX (mobilie CPU)	Mini-ITX	Mini-ITX	Mini-ITX	Mini-ITX
Dimensions	170x170mm- Height: 3.5/1.6/16.5mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm
Link	http://www.eurotech.com/en/products/CPU-521-17	http://www.adlinktech.com/PD/web/PD_detail.php?cKind	http://www.adlinktech.com/PD/web/PD_detail.php?cKind=	http://www.aaeon.com/en/p/mini-itx-motherboards-eml	http://www.aaeon.com/en/p/mini-itx-motherboards-em

	3.5" SubCompact Boards				
Brand	AAEON	AAEON	Axiomtek	Axiomtek	Axiomtek
Model	GENE-SKU6	GENE-APL5	CAPA500	CAPA880	CAPA881
CPU	Intel® 6th Generation Core™ i7-6600U/i5-6300U SoC Processor	Intel® Pentium® N4200/ Celeron® N3350 Processor SoC	LGA1151 socket 7th/6th generation Intel® Core™ i7/i5/i3	LGA1150 socket 4th gen Intel® Core™ i7/i5/i3 & Celeron®	4th/5th Generation Intel® Core™ i7-4700EQ/ i7-5850EQ / i5-4402E/ i5-4422E/ i3-4100E/ i3-4110E
Chipset	Intergrated into SoC	Intergrated into SoC	Intel® H110 or Q170	Intel® H81	Intel® HM86 or QM87
RAM	Up to 1x 16 GB DDR4 at 2133MHz	Up to 1x 8 GB DDR3L 1867MHz, SODIMM	1 x SO-DIMM DDR4-1867, up to 16GB	1 x SO-DIMM DDR3-1600, up to 8 GB	
SATA	1x SATA3 (6.0Gb/s)	1x SATA3 (6.0Gb/s)	1x SATA3 (6.0Gb/s)	1x SATA3 (6.0Gb/s)	
Ethernet	2x RJ-45 Gbit/s (i210/i211) (support etherCAT)	2x RJ-45 Gbit/s (i210/i211)	2x RJ-45 Gbit/s (i219LM + i211AT)	2x RJ-45 Gbit/s (i211AT + i218LM)	
	4x USB 3.0	2x USB 3.0	3x USB 3.0	2x USB 3.0	
USB	2x USB 2.0	4x USB 2.0	3x USB 2.0	4x USB 2.0	
	onboard touch screen controller		1x USB 2.0 expansion	1x USB 3.0 expansion	
Serial	3x RS-232/422/485, 1x RS-232	2x RS-232/422/485, 2x RS-232	1 x RS-232/422/485	1 x RS-232/422/485	
Video	1x LVDS, 1x DVI-I, 1x DP	1x VGA, 1x LVDS	1 x VGA, 1 x LVDS, 1 x HDMI	1 x VGA, 1 x LVDS, 1 x HDMI	
	1x mSATA	1x mSATA	1 x Full-size PCI Express Mini	1 x Full-size PCI Express Mini	
PCI Express	1x Mini-Card	1x Mini-Card	1x PCIe x1, 1 x LPC, 1 x SMBus	1x PCIe x1, 1 x LPC, 1 x SMBus	
Power Input	AT/ ATX 12V	AT/ ATX 12V	12V DC-in only	12V DC-in only	
Power Consumption	-	-	CPU TDP Up to 35W		
Dimensions	146mm x 101.7mm	146mm x 101.7mm	146 x 104 mm	146 x 104 mm	146 x 104 mm
Link	http://www.aaeon.com/en/p/embedded-single-board-compu	http://www.aaeon.com/en/p/embedded-single-board-com	http://www.axiomtek.com/Default.aspx?MenuId=Products/	http://www.axiomtek.com/Default.aspx?MenuId=Produ	http://www.axiomtek.com/Default.aspx?MenuId=Prod

	COM-Express Carrier Boards				
Brand	Connect Tech	Connect Tech	Connect Tech	Connect Tech	Connect Tech
Model	COM Express® Type 6 Ultra Lite Carrier Board	COM Express® Type 6 Rugged Ultra Lite Carrier Board	COM Express® Type 6 Stacking Carrier	COM Express® Type 6 104e	COM Express® Type 6 PMC/XMC Lite Carrier
Compatibility	COM Express® Type 6	Type 6, PCIMG COM Express® COM.0 R2.0	Type 6, PICMG COM Express COM.0 R2.1	COM Express® Type 6	COM Express® Type 6
SD Card	1x microSD	-	1x microSD	-	1x microSD
Mini PCIe/mSATA	2x Mini PCIe	1x SIM Card Expansion	3x Mini-PCIe/mSATA Full Size	2x Mini-PCIe/mSATA	2 slots (with PCIe, USB and SATA connections)
Storage	2x mSATA, 2x on-board connectors	2x External Vertical Locking SATA Connector, 2x mSAT/	4x SATA	1x External SATA	-
USB	2x USB 2.0, 4x USB 3.0	8x USB 2.0	2x USB 3.0, 3x USB 2.0 (Mini-PCIe), 4x USB 2.0	4x USB 3.0	2x USB 2.0, 4x USB 3.0
Network	2x Gigabit (10/100/1000) Ports	-	4x Gigabit Ethernet	2x Gigabit ethernet	2x Gigabit ethernet
Display	1 x LVDS interface, 1x DisplayPort, 1x HDMI, 1x VGA	1 x LVDS interface, 1x VGA, 2x DDI	1x LVDS, 1x VGA, 2x DDI	1x LVDS, 1x VGA	1x LVDS, 1x VGA, 2x DP/HDMI
Serial	1x RS-232 from COM Express, 2x RS-232, 2x RS-485	1x Console RS -232 port (TX/RX), 2x RS-232, 2x RS-422/485	2x RS-232/485	2x RS-232/485	1x RS-232
Power Requirements	12V DC Input Only (DC Barrel)	Single +12V input	12V Input	12V Input	12V to +48V DC
Operating Temperature	-40°C to +85°C	-40°C to +85°C	-40°C to +85°C	-40°C to +85°C	?
Dimensions	125mm x 95mm	125mm x 95mm	125mm x 95mm	125mm x 95mm	195mm x 190mm
Link	http://connecttech.com/product/com-express-type-6-ultra-li	http://connecttech.com/product/com-express-type-6-rugg	http://connecttech.com/product/com-express-type-6-stacki	http://connecttech.com/product/com-express-type-6-1(	http://connecttech.com/product/com-express-type-6-p

	Nano-ITX
Brand	AAEON
Model	NITX-BD1
CPU	Onboard Intel® 5th gen. 14nm ULT i7-5650U/ i5-5350U/ i3-5010U
Chipset	Intergrated into SoC
RAM	2x SODIMM, max. 16GB, DDR3L 1600 MHz
SATA	1x SATA 6.0 Gb/s port, 1x M.2 slot for mSATA
Ethernet	2x RJ-45 Gbit/s (2x Realtek 8111G)
USB	2x USB 3.0 port (Rear) 2x USB 3.0 port (front), 2x USB 2.0 port (front) 1x USB header support USB 2.0 port
Serial	
Video	2x DP
PCI Express	1x M.2 slot (M-key) half size
Power Input	12V (DC-in Jack)
Power Consumption	15 W (CPU)
Complaiance	Nano-itx
Dimensions	120mm x 120mm
Link	http://www.aaeon.com/en/p/nano-itx-motherboards-nitx-bd

	MINI-PC				
Brand	Intel®	Intel®	Intel®	MSI	MSI
Model	NUC 6i5SYK	NUC 5i7RYH	NUC 5i5RYH	CUBI 011BEU	Cubi 2 012BEU
CPU	Intel® 6th Generation Core™ i5-6260U 2x 1.8 GHz com Turbo até 2.9 GHz	Intel® 5th Generation Core™ I7 5557U 2x 3.1 GHz com Turbo até 3.4 GHz	Intel® 5th Generation Core™ i5-5250U 1.6 GHz com Turbo até 2.7 GHz	Intel® 5th Generation Core™ I5-5200U 2x 2.2 GHz com Turbo até 2.7 GHz	Intel® 6th Generation Core™ i5-7200U Dual-Core, 2.5 GHz com Turbo até 3.1
Chipset					Intel® H110
RAM	2x DDR4 SO-DIMM, dual PC4-17000S, max. 32GB	2x SODIMM Sockets - Up to 16GB DDR3L	2x SO-DDR-3 RAM 1333/1600Mhz até 16GB	2x SO-DDR-3 RAM 1333/1600Mhz até 16GB	2x DDR4 SO-DIMM, dual PC4-17000S, max. 32GB
SATA	1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA
Ethernet	1x RJ-45 Gbit/s	1x RJ-45 Gbit/s	1x RJ-45 Gbit/s	1x RJ-45 Gbit/s	1x RJ-45 Gbit/s
USB	4x USB 3.0 2x headers USB 2.0	4x USB 3.0 2x headers USB 2.0	4x USB 3.0 2x headers USB 2.0	4x USB 3.0	3x USB 3.0 1x 1 x USB 3.1 Type-C
Video	1x full-size HDMI, 1x Mini DisplayPort	1x mini HDMI, 1x Mini DisplayPort	1x mini HDMI, 1x Mini DisplayPort	1xHDMI, 1x Mini DisplayPort	1xHDMI, 1x Mini DisplayPort
PCI Express	1x M.2/M-Key (PCIe 3.0 x4)	1x M.2/M-Key (PCIe 3.0 x4)	1x M.2/M-Key (PCIe 3.0 x4)	1x mSATA	1x mSATA
Power Input	12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)
Power Consumption	36.6 W MAX, 10-18 TYP	?	?	30 W	?
Dimensions	115mm x 111mm x 32mm	115mm x 111mm x 49mm	115 x 111 x 34.5mm	115 x 111 x 35mm	115.2 x 112.2 x 32.6
Link	http://micro-xpc.com/items/barebones~tabletpc-touch-pc-/	http://micro-xpc.com/items/barebones~tabletpc-touch-pc	http://micro-xpc.com/items/barebones~tabletpc-touch-pc-/	https://www.msi.com/Desktop/Cubi.html#hero-overvie	https://www.pcdiga.com/mini-pc-msi-cubi-2-012beu-i5
Preço	386.45 € (barebones)	486.55 € (barebones)	357.32 € (barebones)	350€	379€

Lojas em Portugal

http://micro-xpc.com/

http://www.lusomatrix.pt/

AAEON	AAEON	CONGATEC	CONGATEC	CONGATEC	CONGATEC
COM-SKUC6	COM-SKUC6	conga-TS175	conga-TS170	conga-TS97	conga-TS87
Intel® 6th Gen i7-6600U, (2.6 GHz) / i3-6100U (2.3 GHz)	Intel i7- 3555LE (2.5 GHz)/ i7- 3517UE (1.7 GHz) / i3- 3217UE (1.6 GHz)/ i7- 3555LE/2.5 GHz	Intel 7th gen Xeon® E3-1505M/ E3-1505L, Core i7-7820EQ/ i5-7440EQ / i5-7442EQ/ i3-7100E/ i3-7102E	Intel 6th gen Xeon® E3-1578LV5/ E3-1558LV5/ E3-1515MV5 /E3-1505MV5 /E3-1505LV5, Core i7-6820EQ/ i5-6440EQ/ i5-6442EQ/ i3-6100E	Intel 5th gen Xeon® E3-1278LV4/ E3-1258LV4, Intel® Core™ i7-5850EQ / i7-5700EQ	Intel 4th gen Core i7-4700EQ / i5-4 i3-4100E / i3-4102E
Intergrated into SoC	Intel® QM77/HM76	Mobile Intel® 100 Series Chipset	Mobile Intel® 100 Series Chipset	Mobile Intel® 8 Series: Intel® QM87 for Intel® Core™	Mobile Intel® 8 Series: Intel® QM8
DDR3L 1333/1600, SODIMM up to 8 GB	DDR3 1333/1600 MHz SODIMM x 2, up to 16 GB	2 Sockets, SO-DIMM DDR4 up to 32GByte	2 Sockets, SO-DIMM DDR4 up to 32GByte	2 Sockets, SO-DIMM DDR3L up to 32GByte	2 Sockets, SO-DIMM DDR3L up to
3x SATA3 (6.0Gb/s)	4x SATA2	4x SATA3	4x SATA3	4x SATA3	4x SATA3
1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet
4x USB 3.0	4x USB 3.0	4x USB 3.0	4x USB 3.0	4x USB 3.0	4x USB 3.0
8x USB 2.0	8x USB 2.0	8x USB 2.0	8x USB 2.0	8x USB 2.0	8x USB 2.0
2x 2-wire UART	2x 2-wire UART	2x 2-wire UART	2x 2-wire UART	2x 2-wire UART	2x 2-wire UART
1x VGA, LVDS/eDP, DDI	2x HDMI, 2x DVI, 2x DP	3x TMDS (HDMI) / DisplayPort	3x TMDS (HDMI) / DisplayPort	3x TMDS (HDMI) / DisplayPort	3x DS (DVI | HDMI) / DisplayPort
8x PCIe x 1	7x PCIe [x1], 1x PCIe [x16]	8x PCIe [x1], 1 x PEG x16 Gen 3	8x PCIe [x1], 1 x PEG x16 Gen 3	7x PCIe [x1], 1 x PEG x16 Gen 3	7x PCIe [x1]
AT/ATX 12 V	AT/ATX 12 V	AT/ATX 12 V	AT/ATX 12 V	AT/ATX 12 V	AT/ATX 12 V
13 W	27 W	45W /25W /45W /45W /25W /35W /25W	45W /45W /45W /45W /25W /45W /25W /45W/ 25W	47W	47W/ 37W / 25W/ 37W/ 25W
COM Express Type 6 Compact	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Basic
95 x 125 mm	95 x 125 mm	95 x 125 mm	95 x 125 mm	95 x 125 mm	95 x 125 mm
http://www.aaeon.com/en/p/com-express-modules-c	http://www.aaeon.com/en/p/com-express-modules-c	http://www.congatec.com/products/com-express-typ	http://www.congatec.com/products/com-express-typ	http://www.congatec.com/products/com-express-type6/	http://www.congatec.com/products

AAEON	AAEON	AAEON	AAEON	AAEON	AAEON
EMB-Q170B	EMB-Q170C	EMB-Q87A	EMB-Q87B	EMB-H81A	EMB-QM87A
Socket LGA1151 supporting Intel® 6th Generation Core™ i7/i5/i3	Socket LGA1151 supporting Intel® 6th Generation Core™ i7/i5/i3	Socket LGA1150 supporting Intel® 4th Generation Core™ i7/i5/i3	Socket LGA1150 supporting Intel® 4th Generation Core™ i7/i5/i3	Socket LGA1150 supporting Intel® 4th Generation Core™ i7/i5/i3	Onboard Intel® 4th Generation Co 4400E / i3-4100E
Intel® Q170	Intel® Q170	Intel® Q87	Intel® Q87	Intel® H81	Intel® QM87
DDR4 2133/1867 MHz SODIMM x 2, Up to 32GB	DDR4 2133/1867 MHz SODIMM x 2, Up to 32GB	DDR3 1333/1600 MHz SODIMM x 2, Up to 16GB	DDR3 1333/1600 MHz SODIMM x 2, Up to 16GB	DDR3 1333/1600 MHz SODIMM x 2, Up to 16GB	DDR3 1333/1600 MHz SODIMM x
4x SATA3 (6.0Gb/s)	6x SATA3 (6.0Gb/s)	4x SATA3 (6.0Gb/s)	4x SATA3 (6.0Gb/s)	2x SATA3 (6.0Gb/s), 2x SATA2 (3.0Gb/s)	4x SATA3 (6.0Gb/s), 2x SATA2 (3
2x RJ-45 Gbit/s (219LM + 211AT)	2x RJ-45 Gbit/s (219LM + 211AT)	2x RJ-45 Gbit/s (I217LM + I210AT)	2x RJ-45 Gbit/s (I217LM + I210AT)	2x RJ-45 Gbit/s (2x8111F)	2x RJ-45 Gbit/s (2x 8111E)
4x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)
2x USB 2.0	2x USB 2.0	2x USB 2.0	3x ports support additional 6 USB 2.0	2x USB 2.0 (rear panel)	2x USB 3.0 ports, 4x USB 2.0 port
3x ports support additional 6 USB 2.0	1x ports support additional 2 USB 3.0	3x ports support additional 6 USB 2.0		1x ports support additional 2 USB 2.0	
1x RS-232/422/485, 1x RS-232	1x RS-232/422/485, 1x RS-232	1x RS-232/422/485, 1x RS-232	1x RS-232/422/485, 1x RS-232	1x RS-232/422/485, 2x RS-232(2x COM Port in rear)	1x RS-232/422/485, 4x RS-232
2x DP ,1x LVDS, 1x eDP	1x VGA, 2x HDMI	1x DP, 1x HDMI	3x DP	1x VGA, 1x LVDS, 1x DVI-I	1x VGA, 3x HDMI
1 PCIe x4	1 PCIe x8	1 PCIe x4		1 PCIe x4	1 PCIe x16
	1 Mini PCIe (full size slot)	1 Mini PCIe (full size slot)	1 Mini PCIe (full size slot)	1 Mini PCIe (full size slot)	1x COM port RS-232/422/485
	1 Mini PCIe (half size slot)	1 Mini PCIe (half size slot)	1 Mini PCIe (half size slot)		1 Mini PCIe (half size slot)
12V AUX Power Connector x 1	12V AUX Power Connector x 1	12-24V AUX Power Connector x 1	12-24V AUX Power Connector x 1	12V AUX Power Connector x 1, 24-pin EATX Power co	1x 8-pin connector
Up to 65 W (CPU)	Up to 65 W (CPU)	Up to 65 W (CPU)	Up to 65 W (CPU)	Up to 65 W (CPU)	Up to 65 W (CPU)
Mini-ITX	Mini-ITX	Mini-ITX	Mini-ITX	Mini-ITX	Mini-ITX (moblile CPU)
170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm
http://www.aaeon.com/en/p/mini-itx-motherboards-e	http://www.aaeon.com/en/p/mini-itx-motherboards-e	http://www.aaeon.com/en/p/mini-itx-emb-q87a	http://www.aaeon.com/en/p/mini-itx-motherboards-e	http://www.aaeon.com/en/p/mini-itx-emb-h81a	http://www.aaeon.com/en/p/mini-itb

Connect Tech	Connect Tech	Connect Tech	SECO	Kontron	Kontron
COM Express® Type 6 PMC/XMC Ultra Lite Carrier	COM Express® Type 10 PC Connector Carrier	COM Express® Type 10 Stacking Carrier	CCOMe-965	COMe Ref.Carrier–i T6 TMI	COMe Ref.Carrier-i T10 TNI
COM Express® Type 6	Type 10 (PICMG COM Express® COM.0 R2.1)	Type 10, PICMG COM Express COM.0 R2.1	COM Express® Type 6	COM Express® Type 6	COM Express® Type 6
-	-	1x microSD	-	-	-
2 slots (with PCIe, USB and SATA connections)	2 x half size cards OR 1 x full length card	1x Mini-PCIe Half Size with SIM Card Slot, 1x Mini-F	PCIE x16 slot, 1 x PCI-e x4 ,2 x miniPCI-e slot Full /	1x Mini-PCIe Half Size with SIM Card Slot, 1x Mini-PCI	?
2x External SATA, 2x mSATA	1x mSATA	1x SATA 2.0 Port	2 x SATA, 1 x mSATA	?	?
2x USB 2.0, 4x USB 3.0	2x USB 3.0, 6x USB 2.0	2x USB 2.0 (Mini-PCIe), 2x USB 3.0, 4x USB 2.0	4 x USB 3.0 , 2 x USB 2.0 (header)	2 x USB 3.0 , 2 x USB 2.0	2 x USB 3.0 , 2 x USB 2.0
2x Gigabit ethernet	2x Gigabit ethernet	2x Gigabit ethernet	2x Gigabit ethernet	2x Gigabit ethernet	2x Gigabit ethernet
1x LVDS, 1x VGA, 2x DDI	1x HDMI, 1x LVDS	1 x DDI, 1x LVDS	1 x LVDS, 1 x eDP, 2 x combo DP / HDMI, 1 x VGA	1x DisplayPort, 2x MiniDisplayPort	1x HDMI
		2x RS-232/485			
1x RS-232	2x RS-232/422/485		2x RS-232	2x RS-232	2x RS-232
12V to +48V DC	12V Input	12V Input	ATX 12V	12V ATX	12V ATX
-40°C to +85°C	-40°C to +85°C	-40°C to +85°C	0°C to 60°C	-40°C to +85°C	-40°C to +85°C
170mm x 165mm	84mm x 73.415mm	84mm x 55mm	170 x 170 mm	170 x 170 mm	120mm x 120mm
http://connecttech.com/product/com-express-type-6/	http://connecttech.com/product/com-express-type-10	http://connecttech.com/product/com-express-type-1(	http://www.seco.com/prods/eu/ccome-965.html	http://www.kontron.com/products/boards-and-standard-	http://www.kontron.com/products/b

ASUS	Intel®	Intel®	Intel®	Asus	MSI
VivoPC UN65U-BM009M	NUC D34010WYKH	NUC 5i3RYH	NUC 7i3BNH	VivoMini UN65U	Cubi 2 011BEU
Intel® 6th Generation Core™ i5-7200U Dual-Core, 2.50 GHz com Turbo até 3.10 GHz	Intel 4th Generation Core i3-4010U CPU 2x 1.7Ghz QM87	Intel 5th Generation Core i3-5010U 2x 2.1Ghz	7ª Geração Intel® Core™ i3-7100U Dual-Core, 2.40 GHz	7ª Geração Intel® Core™ i3-7100U Dual-Core, 2.40 GHz	7ª Geração Intel® Core™ i3-7100U GHz
2x DDR4 SO-DIMM, dual PC4-17000S, max. 32GB	2x SO-DDR-3 RAM 1333/1600Mhz até 16GB	2x SO-DIMM DDR3L RAM	2x DDR4 SO-DIMM, dual PC4-17000S, max. 32GB	2x DDR4 SO-DIMM, dual PC4-17000S, max. 32GB	2x DDR4 SO-DIMM, dual PC4-170
1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA	1x SATA3, 1x 2.5" SATA
1x RJ-45 Gbit/s	1x RJ-45 Gbit/s	1x RJ-45 Gbit/s	1x RJ-45 Gbit/s	1x RJ-45 Gbit/s	1x RJ-45 Gbit/s
4x USB 3.0	4x USB 3.0	4x USB 3.0	4x USB 3.0	4x USB 3.0	3x USB 3.0
					1x 1 x USB 3.1 Type-C
1xHDMI, 1x DisplayPort	1xHDMI, 1x DisplayPort	1xHDMI, 1x DisplayPort	1xHDMI, 1x DisplayPort	1xHDMI, 1x DisplayPort	1xHDMI, 1x Mini DisplayPort
1x mSATA	1x mSATA, 1x PCIe Mini Card (full-size)	1x mSATA	1x mSATA	1x mSATA	1x mSATA
	1x PCIe Mini Card (half-size)				
12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)	12-19V DC Input (DC Barrel)
?	?				
131 x 131 x 52	131 x 131 x 49.5	115 x 111 x 34	115 mm x 111 mm x 51 mm	131 x 131 x 52 mm	131 x 131 x 52
https://www.pcdiga.com/pc-asus-vivopc-un65u-bm0(	http://micro-xpc.com/items/barebones~tabletpc~touc	http://micro-xpc.com/items/barebones~tabletpc~touc	https://www.pcdiga.com/barebone-intel-nuc-i3-7100	https://www.pcdiga.com/mini-pcs-windows/pc-asus-viv	https://www.pcdiga.com/mini-pc-m
387€	€238.40	€299.55	€365.9	€329	299

	ADLINK	ADLINK	ADLINK	ADLINK	ADLINK	ADLINK
	Express-KL/KLE	Express-SL/SLE	Express-BL	Express-HLE	cExpress-KL	cExpress-SL
i400E / i5-4402E /	Intel 7th gen Xeon® E3-1505M/ E3-1505L, Core i7-7820EQ/ i5-7440EQ/ i5-7442EQ/ i3-7100E/ i3-7102E	Intel 6th gen Xeon® E3-1515M/ E3-1505M/ E3-1505L, Core i7-6820EQ/ i7-6822EQ/ i5-6440EQ/ i5-6442EQ/ i3-6100E/ i3-6102E	Intel 5th gen Xeon® E3-1278L/ E3-1258L, Core i7-5850EQ/ i7-5700EQ	Intel 4th gen Core i7-4700EQ /i5-4400E /i5-4402E /i3-4100E /i3-4102E /i7-4860EQ	Intel 7th gen Core i7-7600U / i5-7300U/ i3-7100U	Intel 6th gen Cor
i7 for Intel® Core™	Intel® QM175/HM175/CM238	Intel QM170/HM170/CM236	Intel® QM87	Intel® QM87 Express	Intergrated into SoC	Intergrated into S
16GByte	2 Sockets, SO-DIMM DDR4 up to 32GByte	2 Sockets, SO-DIMM DDR4 up to 32GByte	2 Sockets, SO-DIMM DDR3L up to 12GByte	Up to 16GB Dual Channel ECC DDR3L at 1600MHz	2 Sockets, SO-DIMM DDR4 up to 32GByte	2 Sockets, SO-DI
	4x SATA3	4x SATA3	4x SATA3	4x SATA3	3x SATA3	3x SATA3
	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethern
	4x USB 3.0	4x USB 3.0	4x USB 3.0	4x USB 3.0	4x USB 3.0	4x USB 3.0
	8x USB 2.0	8x USB 2.0	8x USB 2.0	4x USB 2.0	4x USB 2.0	4x USB 2.0
	2x 2-wire UART	2x 2-wire UART	2x wire UART	2x wire UART	2x wire UART	2x wire UART
	3x DDI, 1x LVDS	3x DDI, 1x LVDS	3x DDI, 1x VGA	3x DDI, 1x VGA, 1x LVDS	1x DDI, 1x VGA, 1x LVDS	1x DDI, 1x LVDS
	8x PCIe [x1], 1 x PEG x16 Gen 3	PCIe x16 or 2 PCIe x8 or 1 PCIe x8 with 2 PCIe x4	1 PCIe x16, or 2 PCIe x8, or 1 PCIe x8 and 2 PCIe x4	7x PCIe [x1], 1 x PEG x16 Gen 3	6x PCIe [x1]	6x PCIe [x1]
	AT/ATX 12 V	AT/ATX 12 V	AT/ATX 12 V	AT/ATX 12 V	AT/ATX 12 V	AT/ATX 12 V
	45W/ 25W / 45W/ 45W/ 25W/ 35W /25W	45W/ 45W / 25W/ 45W/ 25W/ 45W /25W/ 35W/ 25W	47 W	45W/ 25W / 45W/ 45W/ 25W/ 35W /25W	15W	15W
	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Compact	COM Express Ty
	95 x 125 mm	95 x 125 mm	95 x 125 mm	95 x 125 mm	95x95mm	95x95mm
/com-express-type6/c	http://www.adlinktech.com/PD/web/PD_detail.php?c	http://www.adlinktech.com/PD/web/PD_detail.php?cl	http://www.adlinktech.com/PD/web/PD_detail.php?cKin	http://www.adlinktech.com/PD/web/PD_detail.php?cl	http://www.adlinktech.com/PD/web/PD_detail.php?c	http://www.adlink

	AAEON	CONGATEC	CONGATEC	CONGATEC	CONGATEC	Kontron
	EMB-H81B	conga-IC175	conga-IC170	conga-IC97	conga-IC87	mITX-KBL-H-CM
re i7-4700EQ / i5-	Intel® 4th Gen. Core™ i Series 22nm LGA1150 socket Processor	7th Generation Intel Core™ mobile i7-7600U / i5-7300U /i3-7100U / Celeron® 3965U (dual cores)	6th Generation Intel Core mobile i7-6600U/ i5-6300U / i3-6100U /Celeron® 3955U	5th Generation Intel Core mobile i7-5650U/ i5-5350U / i3-5010U /Celeron® 3765U	4th Generation Intel Core mobile i7-4650U/ i5-4300U / i3-4010U / Celeron® 2980U	supporting 7th ge i7/i5/i3, Xeon® E
	Intel® H81	Intergrated into SoC	Intergrated into SoC	Intergrated into SoC	Intergrated into SoC	CM238
2, Up to 16GB	DDR3 1333/1600 MHz SODIMM x 2, Up to 16GB	2x SO DIMM (dual channel DDR4 up to 2x 16 GB)	2x SO DIMM (dual channel DDR4 up to 2x 16 GB)	2x SO DIMM (dual channel DDR3L up to 16GB)	2x SO DIMM (dual channel DDR3L up to 16GB)	2x DDR4 SODIM
.0Gb/s)	2x SATA3 (6.0Gb/s), 1x SATA2 (3.0Gb/s)	3x SATA3 (6.0Gb/s)	3x SATA3 (6.0Gb/s)	3x SATA3 (6.0Gb/s)	3x SATA3 (6.0Gb/s)	4x SATA3 (6.0Gb
	2x RJ-45 Gbit/s (2x 8111G)	2x RJ-45 Gbit/s (i219LM + i211)	2x RJ-45 Gbit/s (i219LM + i211)	2x RJ-45 Gbit/s (i218LM + i210/i211)	2x RJ-45 Gbit/s (i218LM + i210/i211)	4x RJ-45 Gbit/s
	2x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)	4x USB 3.0 (rear panel)	2x USB 3.0 (Inter
s	2x USB 2.0 (rear panel)	4x USB 2.0	4x USB 2.0	4x USB 2.0	4x USB 2.0	2x USB 3.0 (Rea
	3 ports support additional 6 USB 2.0					2x USB 2.0 (Fron
	1x RS-232/422/485, 1x RS-232	2x Serial Port COM2	2x Serial Port COM2	2x Serial Port COM2	2x Serial Port COM2	1xRS232; 1xRS4
	1x DP, 1x HDMI	2x DP, 1x LVDS	2x DP, 1x LVDS	2x DP, 1x LVDS	2x DP, 2x LVDS	3x DP
	1 PCIe x16	1x PCIe x4	1x PCIe x4	1x PCIe x4	1x PCIe x4	1x PCIe x16 (Ger
	1 Mini PCIe (full size slot)	1x Full/Half-size Mini PCIe slot	1x Full/Half-size Mini PCIe slot	1x Full/Half-size Mini PCIe slot	1x Full/Half-size Mini PCIe slot	1x Mini-PCIe
	1 Mini PCIe (half size slot)	1x mSATA	1x mSATA	1x mSATA	1x mSATA	
	1x 12V AUX Power Connector	1x DC-in 12V	1x DC-in 12V	1x DC-in 12V	1x DC-in 12V	12-24 V DC
	Up to 65 W (CPU)	15 W (CPU)	15 W (CPU)	15 W (CPU)	15 W (CPU)	-
	Mini-ITX	Mini-ITX (mobilie CPU)	Mini-ITX (mobilie CPU)	Mini-ITX (mobile CPU)	Mini-ITX (mobile CPU)	Mini-ITX (mobilie
	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 m
c-emb-qm87a	http://www.aaeon.com/en/p/mini-itx-motherboards-e	http://www.congatec.com/en/products/mini-itx-single	http://www.congatec.com/en/products/mini-itx-single-bo	http://www.congatec.com/en/products/mini-itx-single	http://www.congatec.com/en/products/mini-itx-single	http://www.kontro

	Kontron
	COMe Ref.Carrier T6
	COM Express® Type 6
	-
	1x PCI Express x4, 1x miniPCIe , 1x Express Card
	3x SATA ,1x eSATA
	2 x USB 3.0 , 4 x USB 2.0
	1x Gigabit ethernet
	DVI-I, DisplayPort, HDMI, LVDS
	2x RS-232
	12V ATX
	0° to 60°C
	170 x 170 mm
oards-and-standard-l	http://www.kontron.com/products/boards-and-standa

	GIGABYTE
	Brix GB-BSi3H-6100
J Dual-Core, 2.40	6ª Geração Intel® Core ™ i3-6100U Dual core 2.3 GHz
00S, max. 32GB	1600 MHz Max. 16GB 2 x SO-DIMM slots de DDR3l
	1x SATA3, 1x 2.5" SATA
	1x RJ-45 Gbit/s
	4x USB 3.0
	1xHDMI, 1x Mini DisplayPort
	1x mSATA
	12-19V DC Input (DC Barrel)
	112,6x 119,4 x 46,8
si-cubi-2-011beu-i3-7	https://www.pccomponentes.pt/gigabyte-brix-gb-bsi3
	314.11

	SECO	SECO	Kontron	Kontron	Kontron
	COMe-B09-BT6	COMe-953-BT6	COMe-bKL6	COMe-bBD6	COMe-bSL6
e i7-6600U/ i5-6300U / i3-6100U	Intel 6th gen Xeon® E3-1535M/ E3-1505M, Core i7-6820EQ/ i7-6822EQ/ i5-6440EQ/ i5-6442EQ/ i3-6100E/ i3-6102E	Intel 4th gen Core i7-4700EQ/ i5-4402E/ i5-4400E/ i3-4102E/ i3-4100E	Intel 7th gen Xeon® E3-1505M/ E3-1505L, Core™ i7-7820EQ/ i5-7440EQ/ i5-7442EQ/ i3-7100E/ i3-7102E	Intel 6th gen Xeon® D-1559 (12C)/D-1539 (8C)/ D-1548 (8C)/ D-1537, (8C)/ D-1528, (6C)/ D-1527, (4C)	Intel 6th gen Xeon® E3-1515M/ E3-1505M/ E3-1505L, Core™ i7-6820EQ/ i7-6822EQ/ i5-6440EQ/ 5-6442EQ/ i3-6100E/ i3-6102E,
oC	Intel® QM170, HM170 or CM236 PCH	Intel® QM87 Chipset	Intel® Mobile CM238 / QM175	Integrated in SoC	Intel® Mobile CM236 / QM170
MM DDR4 up to 32GByte	2 Sockets, SO-DIMM DDR4 up to 32GByte	Up to 16GB DDR3L-1600 on two SO-DIMM slots	2x DDR4-2400 SO-DIMM up to 2x 16 GByte	2x DDR4-2400 SO-DIMM up to 2x 16 GByte	2x DDR4-2400 SO-DIMM up to 2x 16 GByte
	4x SATA3	4x SATA3	4x SATA3	4x SATA3	4x SATA3
et	1x Gigabit Ethernet (I219-LM)	1x Gigabit Ethernet	1x Gigabit Ethernet	2x Gigabit Ethernet	1x Gigabit Ethernet
	4 x USB 3.0	4 x USB 3.0	4 x USB 3.0	4 x USB 3.0	4 x USB 3.0
	8 x USB 2.0	8 x USB 2.0	4x USB 2.0	-	4x USB 2.0
	2 x UART	2 x UART	2 x UART	2 x UART	2 x UART
	3x DDI, 1x LVDS, VGA	3 x HDMI / DVI, 1x eDP	3 x DP, 1x VGA, 1x LVDS	-	3 x DP, 1x VGA, 1x LVDS
	8 x PCI-e x1 Gen3	7 x PCI Express x1	7 x PCI Express x1,	16x PCIE 3.0, 8x PCIE2.0	8x PCIe x1, 1x PEG x16
	12V DC	12V DC	12V DC	ATX 12V	ATX 12V
	45W/ 45W/ 45W/ 25W/ 45W/ 25W/ 35W/ 25W	47W/ 25W/ 37W/ 25W/ 37W	45W/ 25W/ 45W/ 45W/ 25W/ 35W/ 25W	45W/ 35W/ 45W/ 35W/ 35W/ 35W	
pe 6 Compact	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Basic	COM Express Type 6 Basic
	95 x 125 mm	95 x 125 mm	95 x 125 mm	95 x 125 mm	95 x 125 mm
tech.com/PD/web/PD_detail.php?c	http://www.seco.com/prods/eu/category/com-expres	http://www.seco.com/prods/eu/category/com-expres	http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa

	Kontron	Kontron	Kontron	Kontron	Kontron
238	mITX-SKL-S-C236	mITX-SKL-S-H110	mITX-SKL-H	mITX-BDW-U	KTQM87/mITX
neration Intel® mobile Core™ 3	Socket LGA1151 supporting 6th generation Intel® Core™ i7/i5/i3, Xeon® E3	Socket LGA1151 supporting 6th generation Intel® Core™ i7/i5/i3, Xeon® E3	Intel® 5th gen Xeon® E3-1505M, Core i7-6820EQ / i5-6440EQ / i3-5010U, Celeron G3900E	Intel® 5th gen Core i7-5650U / i5-5350U / i3-5010U	Intel® 4th gen Core i7-4700EQ/ i7-4860EQ / i5-4400E /i3-4100E
	C236	H110	CM236	Broadwell PCH-LP	Embedded QM87
M 2133 MHz (up to 32 GByte)	2x DDR4 SODIMM 2133 MHz (up to 32 GByte)	2x DDR4 SODIMM 2133 MHz (up to 32 GByte)	2x DDR4 SODIMM 2133 MHz (up to 32 GByte)	up to 16 GB SO-DIMM Socket DDR3L-1333/1066	Up to 8 GByte DDR3L/DDR3, 2x (max. 16 GByte)
s/s)	4x SATA3 (6.0Gb/s)	3x SATA3 (6.0Gb/s)	4x SATA3 (6.0Gb/s)	2x SATA3 (6.0Gb/s)	4x SATA3 (6.0Gb/s)
	3x RJ-45 Gbit/s	2x RJ-45 Gbit/s	4x RJ-45 Gbit/s	2x RJ-45 Gbit/s	2x RJ-45 Gbit/s
rnal)+ 1x USB3.0 (Client)	2x USB 3.0 (Rear)+ 2x USB 2.0 (Rear)	4x USB 3.0 (Rear)	2x USB 3.0 (Internal) + 1x USB 3.0 (Client)	4x USB 3.0 in Rear I/O	4x USB3.0
r)+ 2x USB 2.0 (Rear)	2x USB 2.0 (Front Header) + 1x USB 2.0 (mPCIe)	2x USB 2.0 (Front Header) + 1x USB 2.0 (mPCIe)	2x USB 3.0 (Rear I/O) + 2x USB 2.0 (Rear I/O)	2x USB 2.0 on front panel	4x USB2.0 in Rear IO
t) + 1x USB 2.0 (mPCIe)			2x USB 2.0 (Front Header) + 1x USB 2.0 (mPCIe)	1x on mPCIe	USB 2.0 2x Internal Header
85	3xRS232; 1xRS485	1xRS-232/422/485, 1 x RS-232	1x RS232; 1x RS485	2x RS232	2x RS232
	3x DP, 2DP+LVDS	1x DP, 1x LVDS/VGA	2x DP; 1x LVDS	2x DP	3x DP
n 3)	1x PCIe x16 (Gen 3)	1x PCIe x16 (Gen 3)	1x PCIe x16 (Gen 3)		1x PCIe x16 (Gen 3)
	1x Mini-PCIe (Half size)	1x Mini-PCIe (Half size)	1x Mini-PCIe (Half size)	1x PCIe x2	2x Mini-PCIe/ mSATA
				1x mini-PCI Express half/full	2x PCIe sockets
	12 V ATX	12 V ATX	12-24 V DC	12-24 V DC	12-24 V DC
	-		45W/ 45W / 45W / 45W/ 35W	15W	47W / 37W / 37W
CPU)	Mini-ITX	Mini-ITX	Thin Mini-ITX (moblile CPU)	Mini-ITX low profile (moblile CPU)	Mini-ITX (mobilie CPU)
m	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm
n.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa

Kontron	Kontron	Kontron
COMe-bBL6	COMe-cKL6	COMe-cSL6
Intel 5th gen Xeon® E3-1278L/ E3-1258L, Core™ i7-5850EQ/ i7-5700EQ	Intel 7th gen Core™ i7-7600U/ i5-7300U/ i3-7100U	Intel 6th gen Core™ i7-6600U/ i5-6300U / i3-6100U
Intel® Mobile QM87	Integrated in SoC	Integrated in SoC
2x non-ECC DDR3L-1600 up to 2x 16 GByte	1x DDR4 SO-DIMM up to 16 GByte	8GB DDR4 memory down & up to 16GB DDR4
3x SATA3, 1x SATA2	2x SATA3	2x SATA3
1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet
4 x USB 3.0	4 x USB 3.0	4 x USB 3.0
4x USB 2.0	4x USB 2.0	4x USB 2.0
2 x UART	2 x UART	2 x UART
3 x DP, 1x VGA, 1x LVDS	3 x DP, 1x VGA, 1x LVDS	3 x DP, 1x VGA, 1x LVDS
7x PCIe x1, 1x PEG x16	5x PCIe x1, 4x PCIE3.0 on PEG Lanes	5x PCIE3.0, 4x PCIE3.0 on PEG Lanes
ATX 12V	ATX 12V	ATX 12V
	15 W	15 W
COM Express Type 6 Basic	COM Express Type 6 Compact	COM Express Type 6 Compact
95 x 125 mm	95 x 95 mm	95 x 95 mm
http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa

Kontron	Kontron	Kontron
KTH81/mITX	KTQ87/mITX	KTA70M/mITX
Intel® 4th Gen. Core™ Series 22nm LGA1150 socket	Intel® 4th Gen. Core™ Series 22nm LGA1150 socket	AMD R-Series R-460L(4 cores) / R-452L(4 cores)/ R-252F (2 cores)
H81	Q87	AMD Fusion Controller Hub A70M
Up to 16 GB DDR3 (max. 2x 8 GB)	Up to 16 GB DDR3 (max. 2x 8 GB)	Up to 16 GB DDR3 (max. 2x 8 GB)
3x SATA3 (6.0Gb/s)	5x SATA3 (6.0Gb/s)	5x SATA3 (6.0Gb/s)
2x RJ-45 Gbit/s (i218V + i218AT)	2x RJ-45 Gbit/s (i218LM + i211AT)	2x RJ-45 Gbit/s
2x USB 3.0 (rear)	4x USB 3.0 (rear)	4x USB3.0 (rear)
8x USB 2.0 (2 rear)	6x USB 2.0	9x USB2.0 (2 rear)
2x RS232	2x RS232	2x RS232
2x DP	2x DP	3x DP
1x PCIe x16 (Gen 3)	1x PCIe x16 (Gen 3)	1x PCIe x8
1x mSATA	1x mSATA	1x PCIe x4
1x mini-PCI Express		
12 V ATX	12 V ATX	12 V ATX
-	-	-
Mini-ITX	Mini-ITX	Mini-ITX
170 mm x 170 mm	170 mm x 170 mm	170 mm x 170 mm
http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa	http://www.kontron.com/products/boards-and-standa

Bibliografia

[1] IFR, "Service Robots - Definition and Classification WR 2016," pp. 9–12 (2012).

[2] R. C. A. for Europe, "Updated Market study on European Robotics," (2016).

[3] F. Mauch, A. Roennau, G. Heppner, and T. Buettner, "Service Robots In The Field : The BratWurst Bot," pp. 13–19 (2017).

[4] Q. Ma, Y. Zou, and T. Zhang, "Study of service robot architecture based on middleware and abstract environment," 2012 IEEE International Conference on Robotics and Biomimetics, ROBIO 2012 - Conference Digest pp. 1200–1205 (2012).

[5] "Panasonic Autonomous Delivery Robots - HOSPI - Aid Hospital Operations at Changi General Hospital | Panasonic Newsroom Global,", http://news.panasonic.com/global/topics/2015/44009.html, (Accessed on 09/06/2017).

[6] "Savioke Relay | HartRobotics,", https://www.hartrobotics.com/relay/#tab-1452592393685-99252-8383, (Accessed on 09/06/2017).

[7] "Care-O-bot 4 - Technical Data,", https://www.care-o-bot.de/en/care-o-bot-4/technical-data.html, (Accessed on 09/06/2017).

[8] "Home • Knightscope,", http://www.knightscope.com/, (Accessed on 09/06/2017).

[9] "Fellow Robots® – Fellow is at the forefront of bringing innovation to retail,", http://www.fellowrobots.com/#Intro, (Accessed on 09/07/2017).

[10] "Stockbot - PAL-ROBOTICS,", http://pal-robotics.com/en/products/stockbot/, (Accessed on 09/07/2017).

[11] J. J. M. Lunenburg, S. V. D. Dries, J. Elfring, and R. J. M. Janssen, "Tech United Eindhoven Team Description 2012," pp. 1–8 (2012).

[12] "Stanford Personal Robotics Program,", http://personalrobotics.stanford.edu/, (Accessed on 09/19/2017).

[13] "Hardware Specs | Willow Garage,", http://www.willowgarage.com/pages/pr2/specs, (Accessed on 04/05/2017).

[14] W. Garage, "PR2 User Manual," (2012).

[15] X. Chen, D. Lu, K. Chen, Y. Chen, and N. Wang, "KeJia: The Intelligent Service Robot for RoboCup@ Home 2014," (2014).

[16] N. Hendrich, H. Bistry, and J. Zhang, "PEIS, MIRA, and ROS: Three frameworks, one service robot - A tale of integration," 2014 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2014 pp. 1749–1756 (2014).

[17] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch & Freight : Standard Platforms for Service Robot Applications," pp. 2–7 .

[18] R. Triebel *et al.*, "SPENCER: A socially aware service robot for passenger guidance and help in busy airports," Springer Tracts in Advanced Robotics **113,** 607–622 (2016).

[19] T. Linder and K. O. Arras, in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, A. Koubaa, ed., (Springer International Publishing, Cham, 2016), pp. 187–213.

[20] Spencer, "SPENCER : Social situation-aware perception and action for cognitive robots Software specification of the robot architecture Due," **12,** 1–29 (2014).

[21] A. Bordallo, V. Nagarajan, S. Ramamoorthy, and S. Vijayakumar, "Automatic Configuration of ROS Applications for Near-Optimal Performance," pp. 2217–2223 (2016).

[22] M. Matamoros, J. Savage, and J. L. Ortega-Arjona, "A comparison of two software architectures for general purpose mobile service robots," Proceedings - 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2015 pp. 131–136 (2015).

[23] A. Elkady and T. Sobh, "Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography," Journal of Robotics **2012,** 1–15 (2012).

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," Icra **3,** 5 (2009).

[25] M. Quigley, E. Berger, and A. Y. Ng, "STAIR : Hardware and Software Architecture," .

[26] R. Hartanto and M. Eich, "Reliable, cloud-based communication for multi-robot systems," IEEE Conference on Technologies for Practical Robot Applications, TePRA (2014).

[27] "roslaunch - ROS Wiki,", `http://wiki.ros.org/roslaunch`, (Accessed on 07/11/2017).

[28] "Topics - ROS Wiki,", `http://wiki.ros.org/Topics`, (Accessed on 07/20/2017).

[29] "rosprofiler - ROS Wiki,", `http://wiki.ros.org/rosprofiler`, (Accessed on 07/19/2017).

[30] "arni - ROS Wiki,", `http://wiki.ros.org/arni`, (Accessed on 08/10/2017).

[31] A. Bihlmaier and H. Wörn, "Increasing ROS Reliability and Safety Through Advanced Introspection Capabilities," In *Proceedings of the INFORMATIK 2014*, pp. 1319–1326 (2014).

[32] A. Tiderko, F. Hoeller, and T. Röhling, in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, A. Koubaa, ed., (Springer International Publishing, Cham, 2016), pp. 629–650.

[33] "GitHub - fkie/multimaster_fkie: ROS stack with FKIE packages for multi-robot (discovering, synchronizing and management GUI),", `https://github.com/fkie/multimaster_fkie`, (Accessed on 08/30/2017).

[34] "multimaster_fkie - ROS Wiki,", `http://wiki.ros.org/multimaster_fkie`, (Accessed on 08/30/2017).

[35] "roscpp/Overview/Callbacks and Spinning - ROS Wiki,", `http://wiki.ros.org/roscpp/Overview/Callbacks%20and%20Spinning`, (Accessed on 09/12/2017).

[36] "roscpp/Overview/Publishers and Subscribers - ROS Wiki,", `http://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers#Transport_Hints`, (Accessed on 08/21/2017).

[37] B. S. Peterson L, Larry and Davie, *Computer Networks a Systems Approach* (2003), pp. 396–397.

[38] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the Performance of ROS2," Emsoft pp. 0–9 (2016).

[39] "Object Management Group,", `http://www.omg.org/`, (Accessed on 06/07/2017).

[40] "chrony – Introduction,", `https://chrony.tuxfamily.org/`, (Accessed on 06/28/2017).

[41] "GDB: The GNU Project Debugger,", `https://www.gnu.org/software/gdb/`, (Accessed on 09/12/2017).

[42] R. Bosch, "CAN Specification," (1991).

[43] S. I. Em, "Protocolo de comunicação CAN," pp. 37–61 (1986).

[44] A. Neves *et al.*, "Functionalities and requirements of an autonomous shopping vehicle for people with reduced mobility," VEHITS 2017 - Proceedings of the 3rd International Conference on Vehicle Technology and Intelligent Transport Systems (2017).

[45] "Geekbench 3 - Cross-Platform Processor Benchmark - Geekbench,", `https://www.geekbench.com/geekbench3/`, (Accessed on 09/13/2017).

[46] "Duration is out of dual 32-bit range when enable_statistics = true - ROS Answers: Open Source Q&A Forum,", `https://answers.ros.org/question/220094/duration-is-out-of-dual-32-bit-range-when-enable_statistics-true/`, (Accessed on 10/17/2017).

[47] "GitHub - xuefengchang/micros_swarm_framework,", https://github.com/xuefengchang/micros_swarm_framework, (Accessed on 06/29/2017).

[48] "Intel® NUC Kit NUC7i3BNK,", https://www.intel.com/content/www/us/en/products/boards-kits/nuc/kits/nuc7i3bnk.html, (Accessed on 10/13/2017).

[49] "Mechanical Drawings and Custom Chassis Information for Intel® NUC,", https://www.intel.com/content/www/us/en/support/articles/000006820/boards-and-kits/intel-nuc-kits.html, (Accessed on 10/13/2017).