UNIOESTE - UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE ENGENHARIAS E CIÊNCIAS EXATAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Desenvolvimento de um sistema multiusuários para edifícios inteligentes

Victor Hugo de Almeida Alicino

FOZ DO IGUAÇU
2023

Victor Hugo de Almeida Alicino

Desenvolvimento de um sistema multiusuários para edifícios inteligentes

Monografia submetida à Universidade Estadual do Oeste do Paraná, Curso de Ciência da Computação - Campus de Foz do Iguaçu, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Dr. Antonio Marcos Massao Hachisuca

FOZ DO IGUAÇU

2023

Victor Hugo de Almeida Alicino

Desenvolvimento de um sistema multiusuários para edifícios inteligentes

---

Dr. Antonio Marcos Massao Hachisuca
Orientador(a)

Titulação Nome 1º membro da banca
Membro

Titulação Nome 2º membro da banca
Membro

Dedico este trabalho....

Agradecimentos

Deus, Pais, orientador, instituição, colegas... A todos que de alguma forma, contribuiram no processo

“xxx.”
(Autor)

Resumo

Escreva aqui um breve resumo do projeto de TCC Introdução sobre o assunto, proposta e resultados.

Palavras-chaves:

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Esse capítulo deveria ser composto por: introdução, estado da arte, proposta, justificativa, objetivos e organização da monografia. Uma introdução, como o próprio nome já diz, tem a função de introduzir o leitor ao tema, isto é, a partir dela tem-se uma visão total do trabalho de forma sucinta e objetiva.

É mais comum escrever este capítulo após a finalização de toda a pesquisa. Entretanto se esse capítulo for escrito juntamente com o início das pesquisas, é provável que sejam necessárias alterações no decorrer do desenvolvimento do Trabalho de Conclusão de Curso (`TCC`) .

Observações/Dicas:

1. expor as ideias de forma clara, concisa e convidativa: o leitor precisa se interessar pelo tema;
2. manter a coerência: redija um texto com início, meio e fim;
3. referenciar todo o conteúdo pesquisado;
4. evitar escrever em primeira pessoa (singular/plural);
5. ler o texto em voz alta, isso normalmente ajuda a identificar problemas de semântica e normalmente facilita a organização do conteúdo;
6. a primeira vez que usar uma sigla não esquecer de acrescentar sua descrição e, além disso, indicar que a mesma deve ir para a lista de siglas, conforme exemplo localizado no segundo parágrafo deste capítulo;
7. o editor `Microsoft Word` possui corretores sintático e semântico. Sugestão: copiar o conteúdo do capítulo para o `Word`, passar o corretor, então copiar de volta. Somente esse trabalho já elimina boa parte dos erros que são mais frequentes em uma monografia;
8. é importante a leitura dos exemplos localizados no próximo capítulo.

Para o melhor entendimento, essa monografia foi organizada da seguinte forma:

- No Capítulo (**??**) são apresentados alguns detalhes do LaTeX que deverão ajudar na elaboração do trabalho de graduação;
- Por fim, no Apêndice....

2 Edifício Inteligente

O conceito de edifícios com algum tipo de autonomia humana é chamado de edifício inteligente, esse conceito surge nos Estados Unidos em meados da década de 80 junto com os sistemas de automação de segurança e iluminação para edifícios [NEVES; CAMARGO, 2002]. Esses edifícios chamados de edifícios inteligentes têm se tornado populares nos últimos anos, mas os limites e requisitos do que um edifício precisa ter para ser considerado inteligente é algo nebuloso. O conceito de edifício inteligente em português origina de dois conceitos pouco distintos na língua inglesa, *Intelligent Buildings* e *Smart Buildings*, "Intelligent" segundo o *Oxford Learner's Dictionaries* é aquele que é bom de aprendizado [INTELLIGENT, 2023], enquanto "Smart" é aquele que é inteligente [SMART, 2023], o significado de ambas as palavras não diferem muito entre si, assim como os conceitos de *Intelligent Buildings* e *Smart Buildings*, ambos carregam as mesmas primícias, de um edifício com algum nível de automação, minimizando a interação humana [WONG; LI; WANG, 2005].

2.1 Intelligent Building

Intelligent Buildings vem sendo pesquisados e desenvolvidos há pelo menos três décadas [BUCKMAN; MAYFIELD; BECK, 2014] e existem no mínimo 30 definições diferentes para esse conceito [WIGGINTON; HARRIS, 2002], em 1990, Powell traz uma definição de edifício inteligente (Intelligent Building) comentando o que Stubbings diz em 1988,

> At present the term 'intelligent building' is normally taken to mean 'a building which totally controls its own environment' [STUBBINGS, 1986]. This seems to imply that it is the technical control of heating and air conditioning, lighting, security, fire protection, telecommunication and data services, lifts and other similar building operations that is important - a control typically given over to a management computer system. Such a definition for a conventionally intelligent building does not suggest user interaction at all. [POWELL, 1990]

Stubbings diz que um edifício inteligente é aquele edifício capaz de controlar seu próprio ambiente e Powell expande essa ideia, definindo que um edifício inteligente é aquele capaz de controlar seus sistemas de Aquecimento, Ventilação e Ar-Condicionado (AVAC), iluminação, segurança, combate a incêndio etc. através de um sistema de gerenciamento de edifício (Building Management System). Clements-Croome em 2009 define um edifício

inteligente como aquele que é responsivo aos requisitos dos seus ocupantes, organizações e sociedade, sendo sustentável no consumo de água e energia, gerando pouca poluição e sendo funcional de acordo com as necessidades do usuário [CLEMENTS-CROOME, 2011] e Brooks sugere que *Intelligent Building* e seu sistema de gestão (Building Management System) são essencialmente a mesma coisa [BUCKMAN; MAYFIELD; BECK, 2014], um sistema de controle que abrange todo o edifício, que conecta, controla e monitora a planta fixa e os equipamentos da instalação [BROOKS, 2012].

2.2 Smart Building

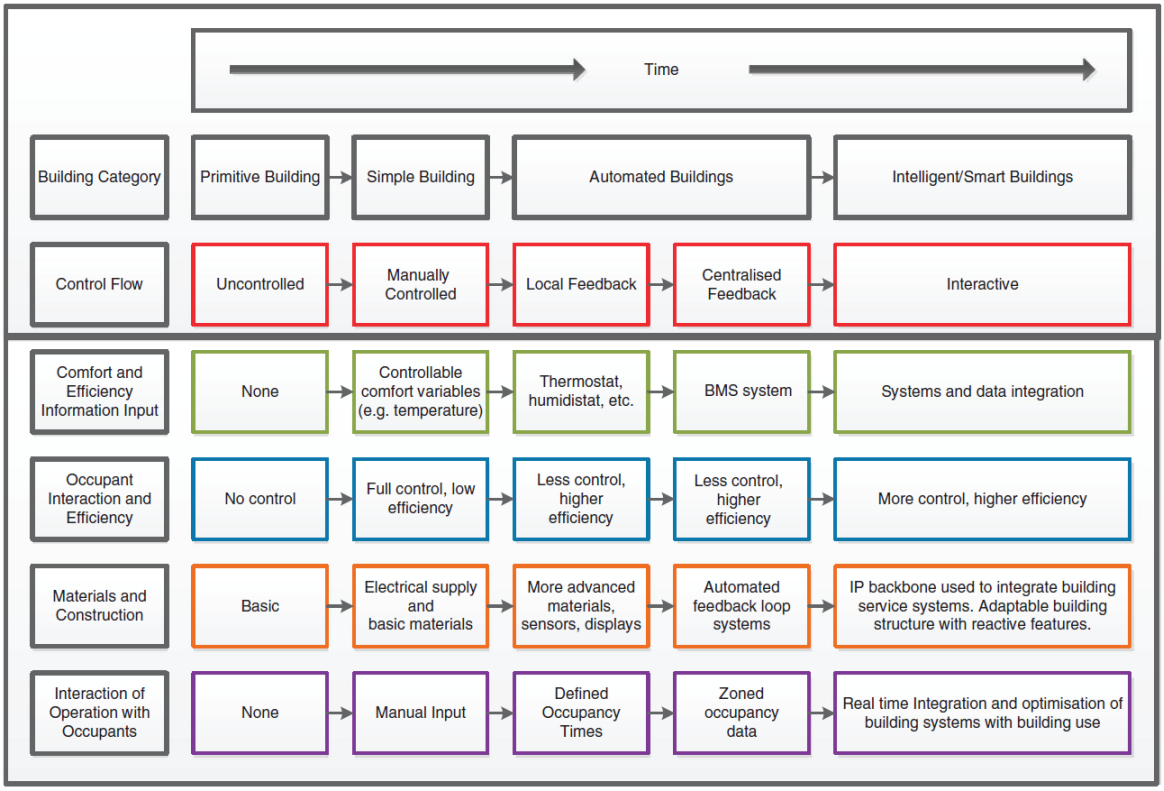

Figura 1 – Progressão da automação em edifícios

Fonte: Buckman, Mayfield e Beck [2014]

Na figura 1, Buckman, Mayfield, Beck mostram uma linha do tempo, partindo de um edifício primitivo com nenhum controle do seu ambiente até um edifício inteligente. Segundo sua pesquisa, sugerem que existem três principais pontos que desenvolvem um edifício até o estado de edifício inteligente, são eles:

1. Longevidade;
2. Energia e Eficiência; e
3. Conforto e Satisfação.

Para atingir o estado de edifício inteligente nos três pontos, existem quatro aspectos que variam o nível do edifício de primitivo a inteligente, são eles:

1. Inteligência, a forma como são coletadas informações das operações do edifício e sua resposta;
2. Controle, a interação entre ocupantes e o edifício;
3. Materiais e Construção, a forma física do edifício; e
4. Empresa, a forma como são coletadas informações e usadas para melhorar a performance dos ocupantes.

A evolução desses métodos em um edifício partindo do primitivo ao inteligente é mostrada na figura 1. De acordo com a pesquisa, Buckman, Mayfield, Beck sugerem que em um *Smart Building*, os quatro aspectos que variam o nível do edifício são desenvolvidos lado a lado, usando a informação de um na operação do outro, diferindo de um *Intelligent Building* que desenvolve a "Inteligência" citada acima de forma independente do outros três aspectos. *Smart Buildings* usam tanto o controle humano, quanto a automação para atingir os quatro aspectos apresentados acima. O aspecto de "Controle", o mais importante para esse trabalho, deve apresentar informações do aspecto de "Inteligência" para os ocupantes do edifício, para que os mesmo possam se adaptar ao edifício assim como o edifício se adapta a eles[BUCKMAN; MAYFIELD; BECK, 2014].

Sensores inteligentes, que podem ser adicionados a qualquer momento da vida de um edifício [KAMAL et al., 2021], frutos da internet das coisas (IoT, do inglês Internet of Things) permitem uma rápida implementação de instalações para edifícios inteligentes, como gerenciamento do sistema de AVAC, sistemas de segurança, monitoramento por câmeras, alertas para eventos como incêndio, vazamento de gás e monitoramento da integridade estrutural do edifício [BELLINI; NESI; PANTALEO, 2022]. A adoção de IoT promove a conectividade entre sensores, dispositivos e sistemas do edifício a nuvem, tal conexão promove o uso de aplicações que usaram os dados coletados [BERKOBEN; KAED; SODORFF, 2020].

2.3 Sistema de Gerenciamento de Edifício

Sistemas de Gerenciamento de Edifício, ou Building Management Systems (BMS) em inglês são controladores inteligentes baseados em microprocessadores instalados na rede para monitorar e controlar os aspectos técnicos de um edifício e seus serviços [APPLIED RISK, 2019]. Um BMS pode controlar componentes com protocolos de mais baixo nível, como BACnet, Modbus e etc. [BERKOBEN; KAED; SODORFF, 2020] os subsistemas do BMS ligam as funcionalidades individuais desses equipamentos para que eles

possam operar como um único sistema [APPLIED RISK, 2019]. A figura 2 mostra um

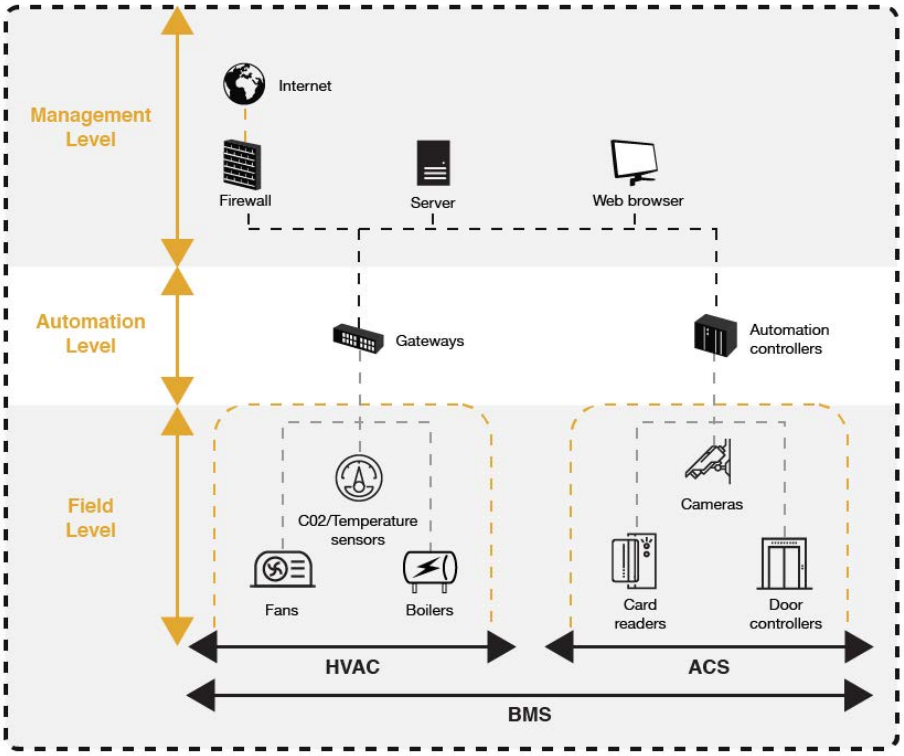

Figura 2 – Relação entre os componentes do BMS e sistemas

Fonte: APPLIED RISK [2019]

exemplo da relação entre os componentes de um BMS.

2.4 Sistema de Controle e Automação de Edifício

Quando um Sistema de Gerenciamento de Edifício passa a satisfazer os requisitos da ISO 16484 [International Organization for Standardization, 2020], tais como suporte a BACnet ele passa a ser um Sistema de Controle e Automação de Edifício (Building Automation Control System, BACS) [European Committee for Standardization, 2006]. Os conceitos de BMS e BACS possuem mais similaridades do que diferenças sendo difícil encontrar na literatura materiais que os diferenciem bem, sendo assim, para este trabalho, Sistemas de Gerenciamento de Edifícios (BMS) e Sistemas de Controle e Automação de Edifícios (BACS) serão considerados sinônimos.

Assim como o Sistema de Gerenciamento de Edifício, o Sistema de Controle e Automação de Edifício também é responsável por controlar e monitorar os aspectos técnicos de um edifício e seus serviços, o BACS também é divido em três camadas, como representado na figura 2, sendo elas:

- Camada de Campo (*Field Layer*);

- Camada de Automação (*Automation Layer*); e
- Cada de Gerenciamento (*Management Layer*).

A camada de campo é a camada mais baixa, sendo nela encontrados os sensores e atuadores que fazem a interação com o ambiente. A camada de automação é onde os dados são processados, loops de controle são executados e alarmes ativados. Por último, a camada de gerenciamento é a responsável por apresentar os dados do sistema, criar logs de acontecimentos e intermediar o controle do usuário com o sistema. Sistemas mais modernos tendem a separar a lógica da interface gráfica para o usuário com o objetivo de criar acessos mais flexíveis aos BACS [DOMINGUES et al., 2016].

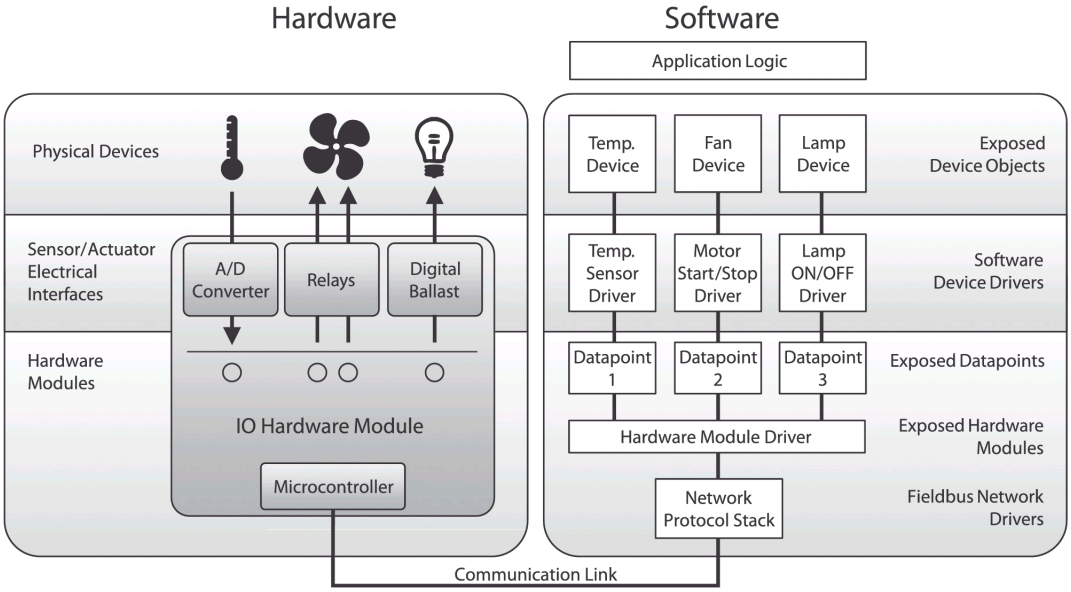

Figura 3 – Pilhas de Hardware e Software envolvidos no Sistema de Controle e Automação de Edifício

Fonte: Domingues et al. [2016]

O caminho da informação em um BACS tem a forma representada pela figura 3, sensores e atuadores interagem com dispositivos físicos através de módulos em hardware, controlados por um microcontrolador responsável por enviar os dados até o BACS através de uma conexão de comunicação (também chamada de barramento de campo ou *fieldbus* em inglês) através de um dos protocolos de baixo nível já citados, como BACnet, KNX, LonWorks, Modbus, ZigBee ou EnOcean. Os dados são recebidos por módulos de hardware que expõem esses dados para o software através de *datapoints* (ou também chamados de *endpoints*, *tags* ou *points*) que interagem com o software para realizar a ação necessária [DOMINGUES et al., 2016].

3 Internet das Coisas

Enquanto a internet foi desenvolvida com dados criados por pessoas, a Internet das Coisas é sobre os dados criados por objetos [NORD; KOOHANG; PALISZKIEWICZ, 2019]. Madakam, Ramaswamy e Tripathi definem Internet das Coisas como uma rede de objetos inteligente aberta que tem a capacidade de se auto-organizar, compartilhar informações, dados e recursos [MADAKAM; RAMASWAMY; TRIPATHI, 2015].

Internet das Coisas é muitas vezes referido pela sigla `IoT` que vem do seu nome em inglês *Internet of Things*. No nome dessa tecnologia temos as duas partes que desempenham os principais papéis: *Internet* e *Coisas*. *Internet* aqui, se refere a mesma internet usadas por bilhões de pessoas ao redor do mundo, já *Coisas*, se refere aos dispositivos com capacidades computacionais atrelados a sensores ou atuadores conectados à rede, o que permite a eles trocarem e consumirem informações.

Em uma rede `IoT`, dispositivos (muitas vezes chamados de *inteligentes*) tem a habilidade de se comunicarem para monitorar o ambiente em que estão, ou alterar este ambiente. Essas ações podem ser configuradas previamente pelo usuário ou serem definidas na hora com a possibilidade de ser acionadas fora da rede local onde esses dispositivos estão conectados através da internet. Como por exemplo, verificar a temperatura de uma sala sem ter chego nela e solicitar ao sistema que acione as unidades AVAC para que a sala esteja na temperatura desejada ao chegar.

Apesar de ocorrer leves diferenças nos modelos de arquitetura, geralmente um sistema IoT é formado por três camadas, sendo elas:

- Camada Física: responsável por perceber o ambiente físico com sensores, atuadores ou qualquer tipo de interface para perceber e modificar o ambiente físico;
- Camada de Rede: camada de transmissão dos dados através das inúmeras formas de conexões disponíveis como redes sem fio, redes móveis e a internet, a fim de fornecer os dados da Camada Física para a Camada de Aplicação;
- Camada de Aplicação, que oferece os serviços chamados *inteligentes* para os usuários finais.

[YAN; ZHANG; VASILAKOS, 2014].

Segundo a empresa especializada em coleta de dados, Statista, a quantidade de dispositivos `IoT` conectados no mundo passa de 15 bilhões em 2023 com projeção para

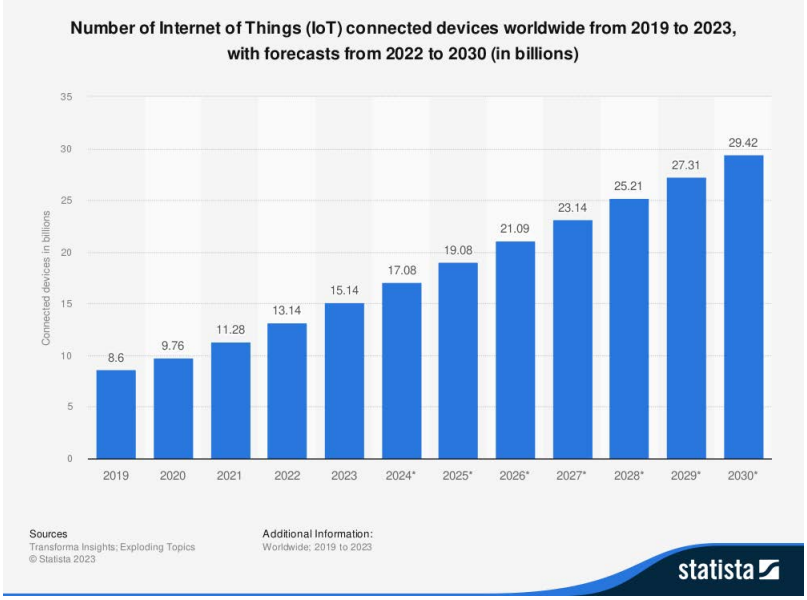

Figura 4 – Números de dispositivos IoT conectados no mundo de 2019 à 2023

Fonte: Statista [2023]

quase o dobro deste número até o fim da década [STATISTA, 2023] como mostrado na figura 4.

Isso se deve muito ao fato de que sensores e microcontroladores ficaram mais baratos, o preço médio dos sensores caiu 200% entre 2004 e 2018 [MICROSOFT, 2019] o que significa mais produtos finais voltado para `IoT` já que ficou possível embarcar sensores em itens do dia a dia com custos acessíveis.

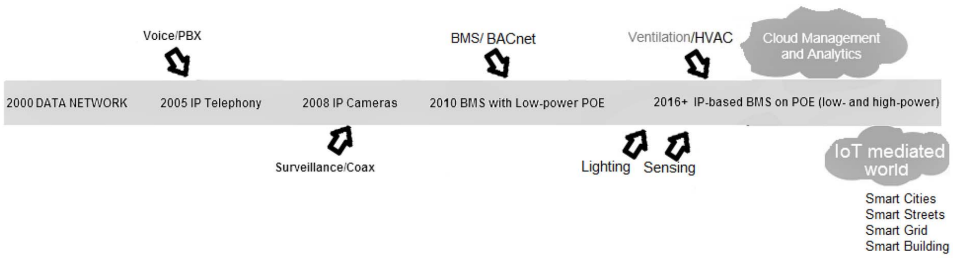

Figura 5 – Convergência da tecnologia de sistemas inteligentes para edifícios nos últimos anos

Fonte: Minoli, Sohraby e Occhiogrosso [2017]

Alguns anos atrás, mesmo em edifícios com BACS, não era incomum encontrar diferentes protocolos, redes e cabos, o que gerava ineficiência tanto na implantação como nos sistemas de gerenciamento. Dessa forma algumas soluções migraram para o uso de cabo e conjunto de protocolos comuns entre as aplicações, como o cabo de par trançado categoria 6 e o protocolo TCP/IP. Com a convergência para protocolos como TCP/IP facilitou a entrada do `IoT` em BACS, como mostrado na figura 5 [MINOLI; SOHRABY; OCCHIOGROSSO, 2017].

Já é possível encontrar equipamentos preparados para uso em BACS com suporte a protocolos `IoT`, como o PFC200 da WAGO que suporta o protocolo MQTT (citado na seção 4.2).

Figura 6 – Controlador PFC200

Fonte: WAGO [2023]

4 Tecnologias

4.1 BACnet

Protocolo de Rede de Automação e Controle de Edifícios (Building Automation and Control Network Protocol) ou BACnet é um protocolo mantido e desenvolvido pela ASHRAE com o objetivo de atender as necessidades de comunicações dos sistemas de controle e automação dos edifícios com seu sensores e atuadores. O protocolo fornece meios para que equipamentos troquem informações não importando o serviço que eles realizem [ASHRAE, 2016]. A forma com que o BACnet funciona, permite que usuários não fiquem presos a sistemas proprietários, criando um padrão de comunicação entre os dispositvos como sistemas de AVAC, alarmes de incêndio e etc. uma das suas grandes vantagens é a habilidade de se adaptar a novas tecnologias de rede [BUSHBY; NEWMAN, 2002]. o BACnet define um modelo de informação, criando objetos para os dispositivos conectados, esses objetos representam seus serviços, assim como suas entradas e saídas. Esses objetos chamados de *device object* definem as propriedades do dispositivo como: nome do modelo, fabricante, status e a lista de outros objetos BACnet associados ao dispositivo [DOMINGUES et al., 2016].

4.2 MQTT

MQTT ou Message Queuing Telemetry Transport [International Organization for Standardization, 2016] é um protocolo de transporte para mensagens no formato *publicar/assinar*, ele é simples, leve, aberto e projetado para ser fácil de implementar, seu foco de uso são ambientes onde recursos são geridos com cuidado, como comunicação *Machine to Machine* (M2M) e Internet das Coisas [OASIS, 2019]. O foco principal do projeto desse protocolo é minimizar o consumo de banda e recursos do dispositivo, sendo assim, ele é capaz de transmitir dados com uma banda baixa e conexões instáveis. O protocolo funciona sobre TCP/IP ou outros protocolos de rede que possam prover as mesmas capacidades do TCP/IP como conexões bi-direcionais. O MQTT oferece três qualidades de serviços (*Quality of Service*, ou QoS) na entrega de mensagens.

- QoS 0: “No máximo uma vez”, mensagens são entregues no máximo uma vez.
- QoS 1: “Pelo menos uma vez”, mensagens são garantidas de serem entregues pelo menos uma vez, pode ocorrer duplicatas.
- QoS 2: “Exatamente uma vez”, mensagens são entregues exatamente uma vez.

Uma estrutura que suporte o protocolo MQTT deve ter pelo menos três componentes,

1. *Publisher* ou Produtor, um cliente MQTT
2. *Broker*, o servidor MQTT que gerencia o recebimento e entrega das mensagens
3. *Subscriber* ou Consumidor, um cliente MQTT

No padrão *publicar/assinar*, o produtor cria uma mensagem e envia para o servidor, consumidores então leem essa mensagem. No MQTT um produtor publica mensagens nos tópicos do *broker*, consumidores interessados em receber essas mensagens se inscreverão nesse tópico e o *broker* se encarregará de entregá-las [MISHRA; KERTESZ, 2020].

4.3 KNX

KNX ou Konnex é um padrão aberto desenvolvido para ser utilizado em automação de edifícios ou residências [SAPUNDZHI, 2020], da mesma forma que o BACnet, o KNX integra sensores, atuadores e controladores dentro de uma rede, os dispositivos são controlados através de um barramento e são chamados de *Bus Access Unit* ou BAU, sendo possível controlar mais de 65 mil BAUs. No KNX, cada fabricante pode implementar sua própria especificação nos dispositivos. BAUs podem ser separadas em grupos menores chamados de de *areas* que podem ser separadas em *lines* de dispositivos. As mensagens trocadas pelos dispositivos KNX pode ser transmitida através de cabos de par trançado, cabos de energia, frequências de rádio ou internet, sendo a forma mais comum o KNX.TP que usa os cabos de par trançado [KRAUS; VIERTEL; BURGERT, 2020]. Um ponto chave do KNX é seu método de comunicação, pouco similar ao MQTT, o KNX utiliza o padrão *observador* na hora de trocar informações, neste padrão múltiplos dispositivos "observadores" serão notificados através de uma única mensagem em *multicast* de quando os dados em uma fonte sofrerem alterações, dessa forma fica mais fácil a criação de relacionamentos um para muitos (1:N). [DOMINGUES et al., 2016].

4.4 ZigBee

ZigBee é um dos mais populares padrões de redes *mesh* sem fio. O protocolo é aberto e baseado em pacotes, projetado para ser de fácil uso, baixo consumo e seguro [TOMAR, 2011]. O modelo ZigBee é principalmente definido em três camadas.

- *Application Support Sublayer*: Responsável por vincular *endpoints*, transmistir mensagens entre os dispositivos e o gerenciamento de grupos.

- *ZigBee Device Object*: Responsável pelo gerenciamento de dispositivos; definir o modo de operação do dispositivo; descobrir novos dispositivos e quais serviços de aplicação ele provê; lidar com os pedidos de vinculamento de outros dispositivos.
- *Application Framework*: Abriga as aplicações dos dispositivos.

[DOMINGUES et al., 2016]

4.5 Tasmota

Tasmota é um software de código aberto que atua como firmware para SoC's (System-on-a-Chip) ESP32 da Espressif e possibilita controle e comunicação personalizado com alguns dispositivos comerciais que são baseados nesse SoC, como interruptores inteligentes, lâmpadas inteligentes ou controles infravermelho inteligentes. A comunicação com o firmware é realizada através do protocolo MQTT, permitindo integrar o hardware comercial a qualquer sistema com suporte ao protocolo [TASMOTA, 2020].

5 Proposta

5.1 Objetivo Geral

Este trabalho tem como objetivo criar um sistema mínimo que possa testar a viabilidade de um sistema de edifício inteligente com múltiplos usuários.

5.2 Objetivos Específicos

Para atingir o objetivo geral, uma pequena arquitetura foi proposta, esta arquitetura é composta de quatro componentes principais, sendo eles:

- **Servidor Local**
- **Servidor Remoto**
- **Aplicativo Móvel**
- **Aplicativo Web**

5.2.1 Servidor Local

O *Servidor Local* é responsável por criar uma camada de tradução entre dispositivos IoT de diferentes fabricantes, que muitas vezes utilizam diferentes protocolos de comunicação. Ele se comunica com esses dispositivos e traduz comandos recebidos pelos usuários do sistema para os dispositivos usando seus respectivos protocolos, o *Servidor Local* também é responsável manter o que cada usuário pode acessar e controlar.

5.2.2 Servidor Remoto

O servidor remoto é responsável por autenticar e armazenar usuários, manter uma relação de todos os *Servidores Locais* conectados e disponíveis e servir como uma ponte de acesso para um usuário acessar um *Servidor Local*.

5.2.3 Aplicativo Móvel

O aplicativo móvel é a interface principal do usuário com o sistema, nele um usuário pode interagir com os dispositivos disponíveis no *Servidor Local*, como por exemplo, ligar e desligar um ar condicionado.

5.2.4 Aplicativo Web

O aplicativo web é a interface pensada para o administrador do *Servidor Local*, nela o administrador pode gerenciar os edifícios, andares e salas disponíveis para aquele *Servidor Local*, além de poder adicionar novos dispositivos e usuários.

É esperado que ao desenvolver essa arquitetura, seja possível responder a problemática inicial.

6 Arquitetura

A arquitetura proposta é ilustrada na figura 7.

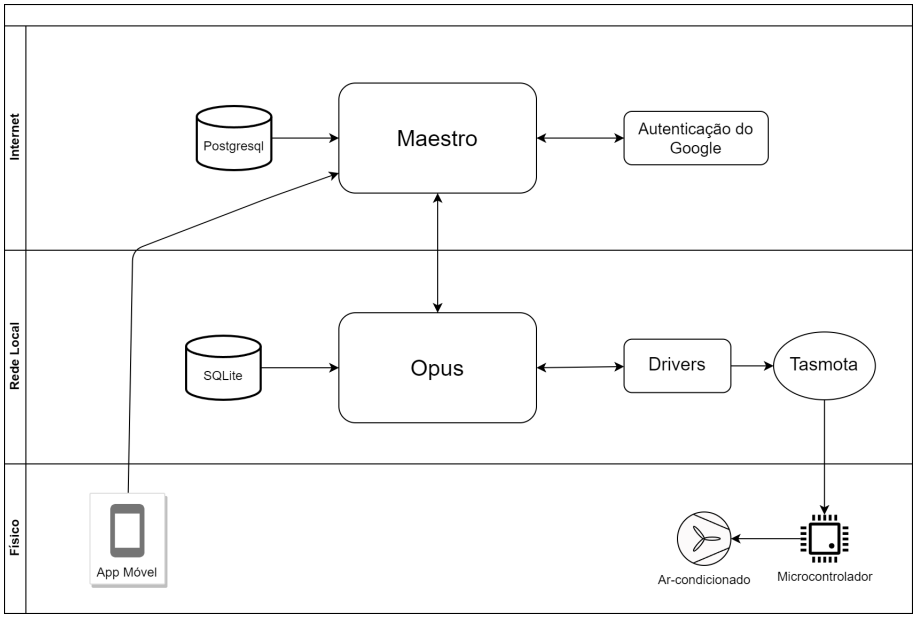

Figura 7 – Arquitetura completa

Fonte: Elaborado pelo Autor (2024)

6.1 Nomeando os Componentes

Como uma homenagem à música erudita, os componentes do sistema foram nomeados com algumas palavras recorrentes no meio musical. A seguir, uma breve descrição de cada componente e o motivo de seu nome.

6.1.1 Servidor Local: **Opus**

A palavra *opus* vem do latim e quer dizer *obra* [OPUS, 2024]. Na música erudita o termo é utilizado para identificar uma obra específica de um compositor [WHAT..., 2024], como o caso da *Op. 67* de Beethoven (Op. é a abreviação de *opus*), esta composição é conhecida por vários nomes como *Sinfonia nº 5* ou *Sinfonia do Destino* [HOW..., 2024], então para identificá-la é usado o *Opus Number*, ou número da obra.

O *Servidor Local* leva este nome pois é nele que encontramos os dispositivos, quase como instrumentos em uma orquestra, os dispositivos são parte de uma obra cujo o objetivo é a coordenação de um edifício.

6.1.2 Aplicação Web e Móvel: **Conductor**

A aplicação web e móvel levam o nome em inglês de *Conductor*, que significa *Condutor*, o condutor é o responsável por dirigir a orquestra ou banda durante a música.

6.1.3 Servidor Remoto: **Maestro**

O *Servidor Remoto* leva o nome de *Maestro* sem muito motivo especial, a palavra vem do italiano e significa *mestre* [MAESTRO, 2024], maestro é um título de respeito atingido por um condutor de orquestra. Apenas por uma questão hierárquica, o *Servidor Remoto* foi nomeado de *Maestro*.

6.2 Camada Física

Camada física é o nome dado por este trabalho a tudo que tem por característica principal, seu hardware, como é o caso de um microcontrolador e um smartphone.

Nesta camada estão dispostos os dispositivos controlados pelo *Opus* e as interfaces gráficas de usuário (GUI) como o aplicativo móvel e o aplicativo web.

6.2.1 Dispositivos Controlados

O sistema *Opus* é feito para ser possível controlar qualquer dispositivo que o desenvolvedor projetar um *driver* para ele. Esses dispositivos em sua forma física normalmente são constituídos de um microcontrolador com acesso à internet e um ou mais atuadores.

6.2.2 Interfaces Gráficas de Usuário

As interfaces gráficas de usuário são os meios que o usuário tem para interagir com o sistema. Nesta arquitetura temos duas interfaces gráficas de usuário: o aplicativo móvel e o aplicativo web. O aplicativo móvel é feito para servir de ponto principal de acesso para o usuário, podendo interagir com os *Dispositivos Controlados* pelo *Opus* já o aplicativo web é a forma que o administrador tem de interagir com o *Maestro*.

6.3 Rede Local

A rede local refere-se a conexão local dentro de um ambiente, como a rede interna de um edifício. Ela pode ou não estar conectada a internet já que a maioria dos protocolos de comunicação utilizado pelos *Dispositivos Controlados* não requerem conexão com a internet, apenas à uma rede local com DHCP.

6.3.1 Banco de Dados

Dentro da rede local deve estar situado o banco de dados que o *Opus* utiliza para armazenar a estrutura do edifício e os dados dos *Dispositivos Controlados*. Devido ao fato do *Opus* ser toda uma aplicação contida dentro de si mesma, sem dependências além das bibliotecas utilizadas na sua concepção e das escolhidas pelo usuário como os *Drivers*, o banco de dados escolhido foi o SQLite3, assim não é necessário que o usuário precise instalar outro software para que o *Opus* funcione.

6.3.2 Opus

O *Opus* é o núcleo do sistema, a aplicação propriamente dita, responsável por se comunicar com os *Dispositivos Controlados* e manter as estruturas do edifício. O *Opus*

requer comunicação constante com o *Maestro* para receber comandos do usuário, logo, apesar de estar situado na Rede Local, que por sua vez não necessita de conexão com a internet, o *Opus* necessita de uma.

6.3.3 Drivers

Os *Drivers* são as partes do *Opus* que podem ser estendidas por outros programadores, *Drivers* são responsáveis por se comunicar com os *Dispositivos Controlados*, traduzindo os comandos do *Opus* para o protocolo do dispositivo, eles são desenvolvidos como módulos do Python, e são carregados dinamicamente pelo *Opus*.

6.3.4 Tasmota

Para provar a viabilidade do sistema, foi escrito um *Driver* para o *Opus*, um driver para comunicação com o firmware Tasmota 4.5.

6.4 Internet

Na camada de internet está o *Maestro*, o servidor central na nuvem, onde ficam armazenados os usuários e os servidores *Opus* cadastrados.

6.4.1 Banco de Dados

O banco de dados da camada de internet é o responsável por armazenar os usuários e os servidores *Opus* cadastrados, este banco de dados não necessariamente precisa estar na mesma máquina que o *Maestro*, sendo assim, existe a possbilidade de configurar um banco de dados externo. A escolha do banco de dados foi o PostgreSQL, por ser um banco de dados de código aberto e versátil.

6.4.2 Autenticação do Google

Inspirado no modelo *Zero Trust* ou *Confiança Zero* [KANG et al., 2023], o sistema não tenta implementar autenticação de usuários e sim delega essa responsabilidade para uma entidade externa que pode verificar a autenticidade do usuário, aqui foi escolhido o serviço de autenticação da empresa Google.

6.4.3 Maestro

O *Maestro* é o servidor central do sistema, responsável por manter os usuários e os servidores registrados, além de ser o responsável por intermediar a comunicação entre os servidores *Opus* e os usuários.

7 Opus

A aplicação que gerencia os dispositivos, a abstração do edifício no sistema e o nível de acesso de usuários a dispositivos específicos é chamada de *Opus*, nela estão contido os *drivers* que são responsáveis por se comunicar com os dispositivos.

O *Opus* foi desenvolvido em Python, especificamente na versão 3.12.2 e faz uso de diversas bibliotecas externas para seu funcionamento, sendo algumas das mais importantes:

- getmac;
- paho-mqtt;
- PyYAML;
- SQLAlchemy;

A aplicação é organizada em quatro módulos principais, como ilustrado na figura 8.

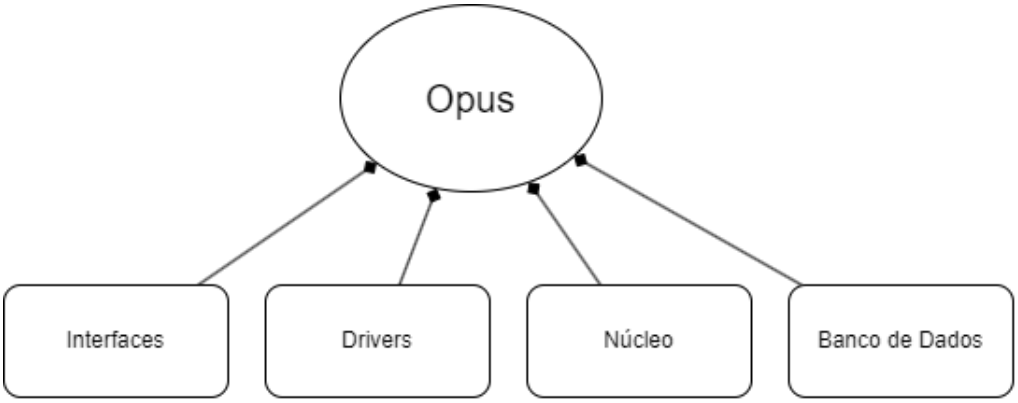

Figura 8 – Módulos do Opus

Fonte: Elaborado pelo Autor (2024)

Alguns módulos podem ser alterados conforme for descrito no arquivo de configuração do *Opus*, o arquivo de configuração é um arquivo texto escrito em YAML, onde o usuário pode alterar o comportamento do *Opus* sem a necessidade de alterar o código fonte. Um exemplo de arquivo de configuração é mostrado na figura 9.

Módulos como *drivers* e *interfaces* terão seus objetos carregados no ambiente de execução do *Opus* através de um dicionário, onde a chave é o nome do módulo e o valor é objeto. Sendo assim para acessar um *driver* ou *interface* basta acessar o dicionário com o nome do módulo como no exemplo a seguir:

```yaml
# Interfaces
interfaces:
# Interfaces are expected to be configurabl
e, so don't forget the colon at the end of t
he line
  - mqtt<local>:
      log_id: "mqtt<local>"
      host: "ip-do-broker-mqtt-local"
      port: 1883
      username: ""
      password: ""
      client_id: "opus-server"

  - mqtt<maestro>:
      log_id: "mqtt<maestro>"
      host: "ip-do-broker-do-maestro"
      port: 1883
      username: ""
      password: ""
      client_id: "opus-server"

# Drivers
drivers:
  - sonoff
  - tasmota
```

Figura 9 – Exemplo de arquivo de configuração

Fonte: Elaborado pelo Autor (2024)

```
interfaces['nome_da_interface'].metodo_desejado()
```

7.1 Interfaces

O módulo de interfaces contém as implementações para comunicações externas do *Opus* utilizando outros protocolos, alguns *drivers* podem requesitar que uma ou mais interfaces específicas estejam ativadas para que ele funcione corretamente. O programa principal irá carregar cada interface cujo o nome do script estiver listado no arquivo de configuração, esse processo se dará através de uma função "def initialize () −> Any" que deve ser implementada em todas as interfaces, essa função deverá retornar o objeto principal da classe desta interface, para que ele seja adicionado no ambiente de execução do *Opus*.

7.1.1 Interface MQTT

A interface MQTT é implementada por padrão no *Opus* uma vez que a comunicação do *Opus* com o servidor *Maestro* é feita via MQTT, logo, ela é essencial para o funcionamento do sistema.

A implementação da interface MQTT segue uma abordagem simples para um script Python, apenas uma classe feita no padrão de projeto Singleton, isso não se fez tão necessário já que interfaces são carregadas para o ambiente de execução do *Opus* por apenas um objeto que é compartilhado entre toda a aplicação, mas foi adotado por boas práticas.

Os seguintes métodos foram implementados na class MQTT:

- def begin(self , config : dict) −> bool: inicializa o Client MQTT, assim que o client efetua a conexão com o servidor MQTT especificado no arquivo de configurações, ele se inscreve em um tópico raiz que é o ID do client, assim ele receberá todas as mensagens enviadas para este tópico;
- def start_thread (self),: inicia a Thread do Cliente MQTT, isso é necessário para que o cliente MQTT possa receber mensagens de forma assíncrona;
- def on_connect(self , client , userdata, flags , reason_code, properties)
- def on_message(self, client , userdata, msg)
- def connect(self , host: str , port: int = 1883)
- def publish(self , topic : str , payload: str): publica o texto da variável payload no tópico especificado;
- def register_callback (self , topic : str , callback): registra uma função como callback para um tópico específico, extremamente útil para que outros módulos do *Opus* possam especificar o que querem receber do MQTT e onde.

- def subscribe(self , topic : str), se inscreve no tópico especificado.

7.2 *Drivers*

Os *drivers* são responsáveis por traduzir os comandos do sistema para os dispositivos, de forma que cada dispositivo no *Opus* está associado a um *driver* específico, desta forma, quando o usuário pedir que um determinado dispositivo seja ligado, por exemplo, esse comando chamará o método correspondente da classe do dispositivo que utilizará seu *driver* para enviar o comando para o hardware.

Usando o Tasmota como exemplo, um dos comandos que é possível usar, é o de ligar um AVAC, para isso é necessário enviar uma mensagem MQTT em uma estrutura de JSON para o tópico: "cmnd/TOPICO MQTT DO DISPOSITIVO/IRHVAC", com o seguinte conteúdo:

```
{
"Vendor": O fabricante do AVAC,
"Power": AVAC ligado ou desligado,
"Mode":  Modo do AVAC, como frio, calor, ventilação, etc,
"FanSpeed": Velocidade do ventilador,
"Temp": Temperatura desejada
}
```

A função do driver é abstrair detalhes como estes e expor para aplicação apenas os comandos que ela espera ter, como ligar, desligar, etc. Para ligar um AVAC Tasmota no *Opus* utilizando este driver, a aplicação apenas chama o método "on()" do objeto. O método "on()" está ilustrado na figura 10.

```python
def on(self) -> None:
    """Turn the HVAC on"""
    log.info("Turning on HVAC %s @ %s", self.name, self.room_id
)   self.mqtt_link.publish(
        topic=f"cmnd/{self.mqtt_name}/IRHVAC",
        payload=json.dumps({
            "Vendor": self.vendor,
            "Power": "On",
            "Mode": self.mode,
            "FanSpeed":3,
            "Temp": self.temperature
        })
    )
    self.power_state = "On"
```

Figura 10 – Método on() de dispositivos AVAC com o *driver* Tasmota

Fonte: Elaborado pelo Autor (2024)

7.3 Núcleo

O módulo *Core*, ou Núcleo, é o módulo principal do *Opus*, nele estão as classes bases para os dispositivos suportados e os *Gerenciadores* que fazem o tráfego das informações no sitema, são exemplos deles:

- DeviceManager, que gerencia os dispositivos;
- LocationManager, que gerencia a estrutura do edifício no sistema;
- CloudManager, que gerencia a comunicação com o *Maestro*;
- UserManager, que gerencia os usuários do *Opus* em questão.

7.3.1 DeviceManager

ah meu Deus eu não aguento mais

7.4 Banco de Dados

8 Resultados

9 Conclusão

10 Trabalhos Futuros

Referências

APPLIED RISK. *I Own Your Building (Management System)*. 2019. Citado 2 vezes nas páginas 4 e 5.

ASHRAE. *BACnet™ - A Data Communication Protocol for Building Automation and Control Networks*. Georgia, US, 2016. Disponível em: ⟨https://www.ashrae.org/technical-resources/standards-and-guidelines/standards-addenda/standard-135-2016-bacnet-a-data-communication-protocol-for-building-automation-and-control-net⟩ Citado na página 10.

BELLINI, P.; NESI, P.; PANTALEO, G. Iot-enabled smart cities: A review of concepts, frameworks and key technologies. *Applied Sciences (Switzerland)*, MDPI, v. 12, 2 2022. ISSN 20763417. Citado na página 4.

BERKOBEN, K.; KAED, C.; SODORFF, T. *A Digital Buildings Ontology for Google's Real Estate*. 2020. Disponível em: ⟨https://github.com/google/digitalbuildings⟩. Citado na página 4.

BROOKS, D. Intelligent buildings: an investigation into current and emerging security vulnerabilities in automated building systems using an applied defeat methodology. 03 2012. Citado na página 3.

BUCKMAN, A. H.; MAYFIELD, M.; BECK, S. B. What is a smart building? *Smart and Sustainable Built Environment*, Emerald Publishing, v. 3, p. 92–109, 9 2014. ISSN 20466102. Citado 3 vezes nas páginas 2, 3 e 4.

BUSHBY, S. T.; NEWMAN, H. M. *BACnet Today Significant New Features And Future Enhancements*. 2002. Citado na página 10.

CLEMENTS-CROOME, D. Sustainable intelligent buildings for people: A review. *Intelligent Buildings International*, v. 3, p. 67–86, 2011. ISSN 17508975. Citado na página 3.

DOMINGUES, P. et al. *Building automation systems: Concepts and technology review*. [S.l.]: Elsevier B.V., 2016. 1-12 p. Citado 4 vezes nas páginas 6, 10, 11 e 12.

European Committee for Standardization. *Energy performance of buildings — Impact of Building Automation Control and Building Management*. [S.l.], 2006. Disponível em: ⟨http://www.cres.gr/greenbuilding/PDF/prend/set4/WI_22_TC-approval_version_prEN_15232_Integrated_Building_Automation_Systems.pdf⟩. Citado na página 5.

HOW did Beethoven's Symphony No.5 become so famous? 2024. Disponível em: ⟨https://www.aco.com.au/news/2023-november/why-is-beethoven-symphony-no-5-so-famous⟩. Citado na página 16.

INTELLIGENT. *In: Oxford Learner's Dictionaries*. 2023. Disponível em: ⟨https://www.oxfordlearnersdictionaries.com/definition/american_english/intelligent⟩. Acesso em: 18 de fevereiro de 2023. Citado na página 2.

International Organization for Standardization. *Information technology — Message Queuing Telemetry Transport (MQTT) v3.1.1*. Geneva, CH, 2016. Disponível em: ⟨https://www.iso.org/standard/69466.html⟩. Citado na página 10.

International Organization for Standardization. *Building automation and control systems (BACS) — Part 6: Data communication conformance testing*. Geneva, CH, 2020. Disponível em: ⟨https://www.iso.org/standard/79630.html⟩. Citado na página 5.

KAMAL, S. et al. *Smart and Intelligent Buildings Achieving Architectural Concepts*. 2021. 155-163 p. Citado na página 4.

KANG, H. et al. *Theory and Application of Zero Trust Security: A Brief Survey*. [S.l.]: Multidisciplinary Digital Publishing Institute (MDPI), 2023. Citado na página 18.

KRAUS, N.; VIERTEL, M.; BURGERT, O. Control of knx devices over ieee 11073 service-oriented device connectivity. In: . [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2020. p. 421–424. ISBN 9781728163895. Citado na página 11.

MADAKAM, S.; RAMASWAMY, R.; TRIPATHI, S. Internet of things (iot): A literature review. *Journal of Computer and Communications*, Scientific Research Publishing, Inc„ v. 03, p. 164–173, 2015. ISSN 2327-5219. Citado na página 7.

MAESTRO. *In: Cambridge Dictionary*. 2024. Disponível em: ⟨https://dictionary.cambridge.org/dictionary/english-italian/maestro⟩. Acesso em: 06 de junho de 2024. Citado na página 16.

MICROSOFT. 2019 manufacturing trends report. 2019. Disponível em: ⟨https://info.microsoft.com/rs/157-GQE-382/images/EN-US-CNTNT-Report-2019-Manufacturing-Trends.pdf⟩. Citado na página 8.

MINOLI, D.; SOHRABY, K.; OCCHIOGROSSO, B. Iot considerations, requirements, and architectures for smart buildings-energy optimization and next-generation building management systems. *IEEE Internet of Things Journal*, Institute of Electrical and Electronics Engineers Inc., v. 4, p. 269–283, 2 2017. ISSN 23274662. Citado na página 8.

MISHRA, B.; KERTESZ, A. The use of mqtt in m2m and iot systems: A survey. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 201071–201086, 2020. ISSN 21693536. Citado na página 11.

NEVES, R. P. A. de A.; CAMARGO, A. R. *Espaços Arquitetônicos de Alta Tecnologia: Os Edifícios Inteligentes*. 2002. Citado na página 2.

NORD, J. H.; KOOHANG, A.; PALISZKIEWICZ, J. *The Internet of Things: Review and theoretical framework*. [S.l.]: Elsevier Ltd, 2019. 97-108 p. Citado na página 7.

OASIS. *MQTT Version 5.0*. [S.l.], 2019. Disponível em: ⟨https://mqtt.org/mqtt-specification/⟩. Citado na página 10.

OPUS. *In: The Latin Dictionary*. 2024. Disponível em: ⟨http://latindictionary.wikidot.com/noun:opus⟩. Acesso em: 06 de junho de 2024. Citado na página 16.

POWELL, J. A. *Intelligent Design Teams Design Intelligent Buildings*. [S.l.]: Habitat International, 1990. 83-94 p. Citado na página 2.

SAPUNDZHI, F. A survey of knx implementation in building automation. *TEM Journal*, UIKTEN - Association for Information Communication Technology Education and Science, v. 9, p. 144–148, 2 2020. ISSN 22178333. Citado na página 11.

SMART. *In: Oxford Learner's Dictionaries*. 2023. Disponível em: ⟨https://www.oxfordlearnersdictionaries.com/definition/american_english/smart_1⟩. Acesso em: 18 de fevereiro de 2023. Citado na página 2.

STATISTA. *Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030*. 2023. Disponível em: ⟨https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/⟩. Citado na página 8.

STUBBINGS, M. Intelligent buildings. Contracts, 1986. Citado na página 2.

TASMOTA. 2020. Disponível em: ⟨https://tasmota.github.io/docs/About⟩. Citado na página 12.

TOMAR, A. *Introduction to Zigbee Technology*. [S.l.]: element14, 2011. Citado na página 11.

WAGO. *Controlador PFC200; 2 geração; 2 x ETHERNET, RS-232/-485*. 2023. Disponível em: ⟨https://www.wago.com/br/controlador/controlador-pfc200/p/750-8212⟩. Citado na página 9.

WHAT is an opus number? 2024. Disponível em: ⟨https://www.abc.net.au/listen/classic/read-and-watch/music-reads/what-is-an-opus-number/11228928⟩. Citado na página 16.

WIGGINTON, M.; HARRIS, J. *Intelligent skins*. [S.l.]: Butterworth-Heinemann, 2002. 176 p. ISBN 0750648473. Citado na página 2.

WONG, J. K.; LI, H.; WANG, S. W. Intelligent building research: A review. *Automation in Construction*, Elsevier, v. 14, p. 143–159, 2005. ISSN 09265805. Citado na página 2.

YAN, Z.; ZHANG, P.; VASILAKOS, A. V. A survey on trust management for internet of things. *Journal of Network and Computer Applications*, Academic Press, v. 42, p. 120–134, 2014. ISSN 10958592. Citado na página 7.