UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

DANIEL AUGUSTO CARNEIRO DE SOUZA

Implementação de Comunicação no Padrão OPC UA em Microcontroladores de Baixo Custo

Sorocaba

2022

DANIEL AUGUSTO CARNEIRO DE SOUZA

IMPLEMENTAÇÃO DE COMUNICAÇÃO NO PADRÃO OPC UA EM MICRONCONTROLADORES DE BAIXO CUSTO

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Eduardo Paciência Godoy

Sorocaba

2022

Souza, Daniel Augusto Carneiro de
Implementação de Comunicação no Padrão OPC UA em Microcontroladores de Baixo Custo / Daniel Augusto Carneiro de Souza. -- Sorocaba, 2022
65 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Eduardo Paciência Godoy

1. Engineering. 2. Redes industriais. 3.Protocolo OPC UA
. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

IMPLEMENTAÇÃO DE COMUNICAÇÃO NO PADRÃO OPC UA EM MICROCONTROLADORES DE BAIXO CUSTO

DANIEL AUGUSTO CARNEIRO DE SOUZA

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Profº. Drº. Maurício Becerra Vargas
Coordenador

BANCA EXAMINADORA:

Prof. Dr. EDUARDO PACIÊNCIA GODOY
Orientador/UNESP-Campus de Sorocaba

Profª. Drª. MARILZA ANTUNES DE LEMOS
UNESP- Campus de Sorocaba

Me. PABLO FELIPE SOARES DE MELO
UNESP – Campus de Sorocaba

Maio de 2022

Agradecimentos

Queria deixar um agradecimento especial a minha mãe, Rosa Maria, e ao meu falecido pai, Odilon Junior, que sempre estiveram comigo nessa caminhada, que me ensinaram valores que contribuíram para minha formação como pessoa e como profissional, e que fizeram de tudo para me proporcionar a oportunidade de estudar em uma universidade pública de qualidade.

Aos meus irmãos, Mariana, Thiago e Maria Beatriz, os quais tenho o prazer de dividir essa jornada, que estiveram sempre por mim e que são exemplos para mim.

Aos meus familiares, que sempre me apoiaram, incentivaram e demonstraram interesse pela minha formação, me acolhendo com muito respeito, amor e carinho.

Aos meus amigos do Colégio São José, os quais estiveram comigo na formação do infantil ao ensino médio, os quais levo para a vida.

Ao Prof. e Orientador Eduardo Paciência Godoy, pelo apoio, conhecimento ao longo da graduação, direcionamentos passados para a realização desse projeto, cobranças, e pela preocupação com qualidade do trabalho, representando com maestria a figura da instituição UNESP.

Aos amigos que ganhei ao longo dessa jornada, a turma XIII, E11+P12 que sempre proporcionaram momentos de alegrias, parcerias, ensinamentos os quais levarei para a vida toda. As instituições do campus: GeraBixo, Dinâmica Eng. Jr., PET-ECA, WECA, Ao Vivo e Em Cores, AAAUSO, que contribuíram com experiências extracurriculares para minha formação.

A minha namorada, Fernanda Borges, que tive o prazer de encontrar nessa caminhada, obrigado pela parceria de vida, pelo apoio nos bons e maus momentos, por estar sempre ao meu lado, incentivando e contribuindo principalmente para meu desenvolvimento pessoal, é um prazer dividir essa vida ao seu lado.

Por fim, deixo um agradecimento a todos que contribuíram direta ou indiretamente para que eu pudesse chegar e concluir mais essa etapa da minha vida.

SOUZA, D. A. C. **Implementação de Comunicação no Padrão OPC UA em Microcontroladores de Baixo Custo.** Trabalho de Graduação (Engenharia de Controle e Automação) – Instituto de Ciência e Tecnologia de Sorocaba, UNESP - Universidade Estadual Paulista, Sorocaba, 2022.

Resumo

A transformação digital tem permitido o desenvolvimento de indústrias mais inteligentes, eficientes e seguras. Dentro desse cenário, a automação industrial possui um grande papel nesta transformação. Uma vez que a padronização e velocidade de dados têm possibilitado que a Indústria 4.0 se torne uma realidade, barreiras de interface e interoperabilidade, que ainda existem no modelo atual da indústria, vêm sendo rompidas. O protocolo OPC UA consiste num padrão independente de plataforma e vem sendo usado para promover a padronização da comunicação industrial. A arquitetura de comunicação OPC UA é baseada no modelo Cliente/Servidor. Alinhado a essa vertente, este trabalho tem por objetivo estudar e utilizar a biblioteca de código aberto open62541, que se trata de uma biblioteca do OPC UA em linguagem C/C++, para a implementação da comunicação no padrão OPC UA em microcontroladores de baixo custo ESP32. A aplicação dessa biblioteca nos microcontroladores ESP32 visou realizar a leitura e o comando de suas entradas e saídas digitais e analógicas. Nesse trabalho o microcontrolador ESP32 funciona como um servidor OPC UA para compartilhamento de suas variáveis e a comunicação é realizada por meio de uma rede Wi-Fi. Todo o desenvolvimento, bem como conceitos necessários para entendimento das aplicações desenvolvidas, está descrito de maneira didática, visando a disponibilização do trabalho como um modelo para desenvolvimento de outras aplicações baseadas na biblioteca open62541 em microcontroladores de baixo custo como o ESP32. A validação foi realizada através da comunicação OPC UA entre o servidor desenvolvido na ESP32 e clientes em softwares comerciais.

Palavras-chave: OPC UA, microcontrolador, ESP32, servidor, cliente.

SOUZA, D. A. C. **Implementation of OPC UA Standard Communication in Low Cost Microcontrollers.** Graduation Project (Bachelor degree in Control and Automation Engineering) – Instituto de Ciência e Tecnologia de Sorocaba, UNESP - Universidade Estadual Paulista, Sorocaba, 2022.

Abstract

Digital transformation has allowed the development of smarter, efficient and safer industries. Within this scenario, industrial automation plays a major role in this transformation. Since standardization and data communication have enabled Industry 4.0 to become a reality, interface and interoperability barriers, which still exist in the current industry model, have been disrupted. The OPC UA protocol consists of a platform-independent standard and has been used to promote the standardization of industrial communication. The OPC UA communication architecture is based on the Client/Server model. In line with this subject, this work aims to study and use the open source library open62541, which is an OPC UA library in C/C++ language, for the implementation of communication in the OPC UA standard in low-cost ESP32 microcontrollers. The application of this library in the ESP32 microcontrollers aimed to perform the reading and command of its digital and analog inputs and outputs. In this work the ESP32 microcontroller functions as an OPC UA server for sharing its variables and communication is performed over a Wi-Fi network. All development, as well as concepts necessary to understand the applications developed, is described in a didactic way, aiming at making the work available as a model for the development of other applications based on the open62541 library in low-cost microcontrollers such as ESP32. The validation was performed through OPC UA communication between the server developed at ESP32 and clients in commercial software.

Keywords: OPC UA, microcontrollers, ESP32, server, client.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas, Siglas e Símbolos

OPC –	OLE for Process Control
OPC UA –	Open Platform Communication Unified Architecture
OLE –	Object Linking and Embedding
IoT –	Internet of Things
Wi–Fi –	Wireless Fidelity
UNESP –	Universidade Estadual Paulista
DCOM –	Distributed Component Object Model
OPC DA –	OLE for Process Control Data Access
OPC A&E –	OLE for Process Control Alarm & Events
OPC HDA –	OLE for Process Control Historical Data Access
COM –	Component Object Model
XML –	Extensible Markup Language
.NET –	.Network
SOA –	Service Oriented Architecture
ERP –	Enterprise Resource Planning
IP –	Internet Protocol
Mac OS –	Macintosh Operating System
API –	Application Programming Interface
URL –	Uniform Resource Locator
URI –	Uniform Resource Identifier
ID –	Identity
TCP –	Transmission Control Protocol
SOAP –	Simple Object Access Protocol
HTTPS –	Hypertext Transfer Protocol Secure
SQL –	Structured Query Language
CSV –	Comma Separated Values
RIC –	Redes Industriais de Comunicação
PLC –	Programmable Logic Controller

Sumário

1. INTRODUÇÃO

1.1 Justificativa

Empresas tem cada vez mais buscado a excelência operacional, visando sempre a redução de custos. Deste modo, embarcam em jornadas de transformação por meio da digitalização, inovação nos processos, utilizam-se de recursos IoTs, conectividade, Big Data, entre outras inúmeras ferramentas e metodologias que contribuem para esse processo de transformação e construção de uma Indústria 4.0.

Assim, quando se pensa em aplicações relacionadas a Industria 4.0, existem dois principais pontos que devem ser definidos visando a melhor sinergia possível, o custo-benefício para as aplicações e suas funcionalidades, sendo esses a forma com que os equipamentos vão se comunicar e o dispositivo que será utilizado para controlar o processo.

Por parte de departamentos de pesquisa e desenvolvimento tecnológico, grandes esforços e investimentos estão criando padrões, equipamentos e softwares, permitindo adentrar cada vez mais para a Indústria 4.0.

A transformação digital tem permitido uma indústria mais inteligente, eficiente, barata e segura. Dentro desse cenário, a automação industrial tem grande papel nesta transformação. Uma vez que convergência, padronização e velocidade de dados têm possibilitado que a Indústria 4.0 se torne uma realidade, barreiras de interface, que hoje existem no modelo atual da indústria, vêm sendo rompidas.

Sobre esse contexto, o presente projeto pretende abordar tópicos como:

- Comunicação de dados na indústria
- Padronização de tecnologias
- Evolução da conectividade
- Controle/Comunicação wireless

A tecnologia mais moderna atualmente na padronização de comunicação, é o OPC UA e, para entender sua origem, é necessário entender que para se fazer uma comunicação de dados digitais, antes de meados de 1990, eram necessários “drives” proprietários. Na prática, isso significava que cada fabricante desenvolvia o seu, com suas particularidades e características. Com o desenvolvimento do sistema operacional Microsoft Windows e a adoção desta plataforma na automação industrial, surgiu a tecnologia OPC (Clássica).

Através de um modelo de coleta de dados padronizados, utilizando os recursos do próprio sistema operacional, foi possível comunicar com sistemas e hardwares industriais. Sendo assim, as limitações e novas demandas, levaram ao desenvolvimento do OPC UA, que tem recebido um grande destaque na indústria 4.0, pois não só suporta como também cumpre uma série de requisitos relacionados a aplicações de IoT, como segurança, mecanismos de transporte de dados diferentes, suporte a aplicações web em nuvem, protocolo orientado a serviços, entre outros.

Atualmente, o desenvolvimento do OPC UA é disponibilizado por meio de várias formas e bibliotecas, sendo até mesmo comercializado por empresas. Vale ressaltar que muitos projetos o disponibilizam de forma open source, por ser um protocolo independente de linguagem e plataforma, tendo assim, uma gama de disponibilidade de desenvolvimentos e aplicações.

Na parte de open source é possível encontrar bibliotecas para diferentes tipos de plataformas, embarcados como o Raspberry PI, PC ou até mesmo microcontroladores. E, nesse contexto de soluções open source tem havido uma busca por desenvolver e portar tais tipos de bibliotecas para soluções de baixo custo, com microcontroladores mais simples como Arduíno e ESP32 que será melhor abordado no decorrer do projeto.

1.2 Objetivos

Alinhado a essa vertente, este trabalho tem por objetivo a elaboração de um projeto que consiste na implementação da comunicação no padrão OPC UA no microcontrolador ESP32, bem como estudar e implementar a biblioteca "*open62541*", que se trata de uma biblioteca do OPC UA em linguagem C, visando realizar o controle de suas entradas e saídas, tendo potencial para simular diversas aplicações industriais, utilizando conceitos adquiridos ao longo da disciplina de Redes Industriais de Comunicação (RIC) do curso de Engenharia de Controle e Automação, para enfim, criar um documento base que possa ser adaptado para diferentes aplicações.

A arquitetura de comunicação OPC UA proposta será composta por um servidor e seu respectivo cliente, possibilitando o envio e recebimento de informações do microcontrolador ESP32 (servidor), para isso será adotado um software open source que possibilite estabelecer essa conexão OPC UA (cliente).

1.3 Estrutura do Trabalho

O presente trabalho foi estruturado em seis capítulos, sendo este primeiro uma introdução sobre a temática abordada, bem como as motivações e objetivos para o desenvolvimento do trabalho.

No capítulo 2 (dois), será apresentada uma revisão bibliográfica sobre protocolos de comunicação com foco nos principais conceitos do OPC UA. Dentro dessas contextualizações são apresentadas as mais relevantes pesquisas que já foram publicadas com base no objeto de estudo proposto.

No capítulo 3 (três), será explicado a metodologia utilizada para os desenvolvimentos, bem como os materiais, hardware e software, necessários para concretização do projeto, sendo eles o Prosys OPC UA Client Mobile, o UA Expert Client, o microcontrolador ESP32 e a Espressif, e os componentes eletrônicos como o sensor DHT22, o motor DC e a Ponte H L298N.

O capítulo 4 (quatro) apresentará os desenvolvimentos realizados e resultados obtidos com tais desenvolvimentos, evidenciando o funcionamento das aplicações propostas por meio da implementação do circuito eletrônico e utilização da biblioteca open 62541.

Por fim, no capítulo 5 (cinco) serão abordadas conclusões sobre o projeto realizado, melhores práticas, projetos futuros e no Capítulo 6 (seis) as referências bibliográficas utilizadas para embasamento do trabalho.

2. REVISÃO BIBLIOGRÁFICA

2.1 Protocolo de Comunicação

Todo dispositivo eletrônico digital tem um protocolo funcionando internamente, assim, os protocolos nada mais são do que maneiras nas quais os dispositivos eletrônicos, softwares, equipamentos tem de conversarem entre si, no qual um envia uma mensagem e outro recebe. Deste modo, o protocolo surge para estabelecer regras, sintaxes, semântica e sincronização dessa comunicação, bem como os possíveis métodos de recuperação de erros.

De modo geral, sistemas de comunicação usam formatos bem definidos para trocar várias mensagens. Cada mensagem deve ter significado exato destinado a extrair uma resposta para aquela situação específica. Os protocolos de comunicação precisam ser acordados pelas partes envolvidas e, para chegar a um acordo, ele pode ser desenvolvido seguindo um padrão técnico (LONGEN, 2019).

Assim, protocolos podem descrever inúmeras formas de uma única comunicação, isso porque a aplicação de tais protocolos podem ser diferentes, já que existem diversos elementos que determinam a qualidade de um protocolo em detrimento de outro olhando do ponto de vista de uma determinada aplicação. Vale ressaltar que a tecnologia está em constante evolução, e certos protocolos não eram possíveis de serem implementados com a tecnologia do passado, por isso, surgem novos protocolos, com características focadas em velocidade, segurança e economia de energia. Deste modo, existem protocolos que são mais seguros que outros, principalmente no mundo de Internet das Coisas (IoT), onde é muito importante que os protocolos sejam seguros, já que tudo se conectando.

O presente trabalho visa aplicar o protocolo OPC UA, por meio do uso da biblioteca de código aberto *open62541* em microcontroladores de baixo custo, focado na implementação da comunicação entre servidor e cliente, com o propósito de disponibilizar uma documentação guia para tal implementação. Tal protocolo tem sido capaz de atender as mais diversas necessidades dentro do meio industrial, sendo muito utilizado na comunicação de um dispositivo supervisório com o seu respectivo módulo de controle. Ele pode ser implementado nas mais diversas áreas, tendo como exemplo sistemas de informação, sistemas industriais, telecomunicação, produção de bens em geral, transporte, saúde, entre outros.

2.2 Padrão de Comunicação OPC

No cenário industrial de algumas décadas atrás, repleto de incompatibilidades, observou-se a que era necessário que um novo padrão industrial fosse desenvolvido a fim de garantir a comunicação entre sistemas baseados na arquitetura Cliente/Servidor. Nessa arquitetura, o processamento da informação é dividido em processos distintos: um processo é responsável pela manutenção da informação (servidores) e outros responsáveis pela obtenção dos dados (os clientes). Com o intuito de desenvolver um padrão de comunicação para acesso em tempo real à dados dentro do sistema operacional *Windows*, a plataforma mais utilizada para automação industrial nessa época, diversas empresas se reuniram na metade da década de 90. (BURKE, 2017).

Como consequência da reunião entre essas diversas empresas, foi criada a *OPC Foundation*, uma fundação que conta com mais de trezentos membros ao redor do mundo, dentre eles os maiores fornecedores de sistemas de controle e instrumentação. Isso apenas foi possível pois notou-se que a comunicação entre os instrumentos e equipamentos no chão de fábrica é de fundamental importância para o bom desempenho e funcionamento dos processos industriais. Contudo, o fato de que cada fabricante desenvolvia seus próprios drivers e protocolos de comunicação (drivers proprietários) tornava essa tarefa bastante complexa (FONSECA, 2002).

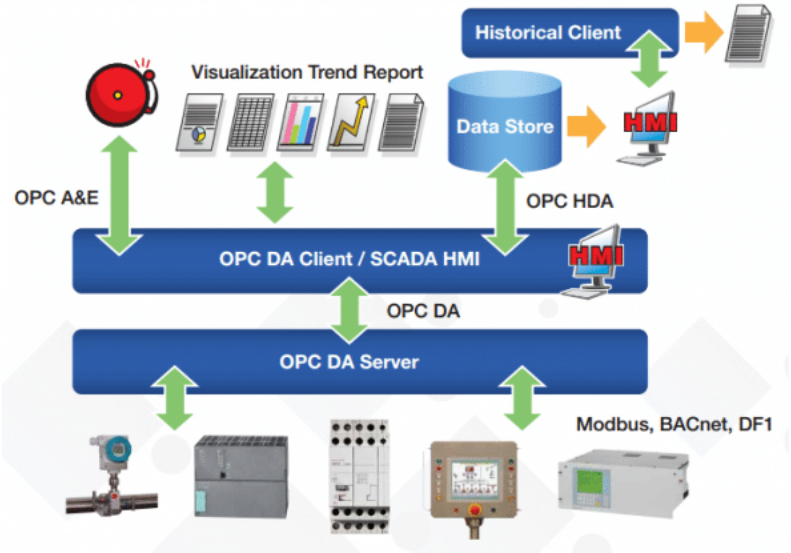
Desta forma surgiu a tecnologia OPC, desenvolvida com a finalidade de suprir a necessidade de se integrar diferentes aplicações, solucionando os problemas de incompatibilidades entre instrumentos, equipamentos e sistemas de controle das mais variadas empresas.

2.3 OPC CLÁSSICO

A *OPC Foundation*, responsável pelo desenvolvimento e manutenção do OPC, define este, como sendo um padrão de interoperabilidade para a troca segura e confiável de dados no espaço de automação industrial. Ao longo dos anos, o acrônimo OPC sofreu algumas variações. Inicialmente, a sigla OPC significava "*OLE for Process Control*", ou "*Object Linking and Embedding for Process Control*", porém devido ao seu estado atual, o acrograma OPC corresponde à "*Open Platform Communications*" (FONSECA, 2002).

O OPC Clássico fornece as seguintes especificações: OPC DA (Data Access), OPC A&E (*Alarm & Event*) e OPC HDA *(Historical Data Access*). A Figura 1 ilustra a arquitetura do OPC Clássico.

Figura 1 – Arquitetura do OPC Clássico



Fonte: FONSECA, 2002

O OPC DA foi designado para padronizar o mecanismo de comunicação de inúmeras fontes de dados, como, dispositivos de entradas e saídas de hardware no chão da planta ou banco de dados em salas de controle. Para a efetivação do OPC DA, os clientes devem selecionar explicitamente as variáveis, também chamadas de itens OPC as quais se deseja ler, escrever ou monitorar no servidor, obtendo acesso aos dados em tempo real (MELO, 2019).

Já o OPC HDA disponibiliza acesso aos dados já armazenados. Em se tratando dos modos de funcionamento do OPC HDA, o cliente durante a conexão com o servidor, cria um objeto denominado OPCHDA *Server*, sendo que este oferece todas as interfaces e métodos para a leitura e atualização de dados históricos. Também há um outro objeto, chamado de OPC HDA *Browser* que é responsável pela navegação no espaço de endereços do servidor HDA (MELO, 2019).

Por fim, o OPC A&E fornece notificações de alarmes e eventos sob demanda. Dentro dessas notificações estão inclusos: alarmes de processos, ações do operador, mensagens informativas e mensagens de rastreamento/auditoria. Para transmitir alarmes e eventos do processo, o cliente OPC A&E precisa estabilizar uma conexão com o servidor, através de uma funcionalidade denominada *Subscription* (MELO, 2019).

Mesmo o OPC Clássico sendo bem aceito na indústria, pelo fato do padrão ser baseado na tecnologia COM (*Component Object Model*) / DCOM da *Microsoft*, algumas restrições tornam-se aparentes como a não utilização do protocolo em plataformas independentes (apenas para plataformas da *Microsoft*), pacotes gerados pelo DCOM são em geral

complexos e difíceis de serem enviados pela internet devido ao firewall e um difícil gerenciamento dos servidores independentes (FONSECA, 2002).

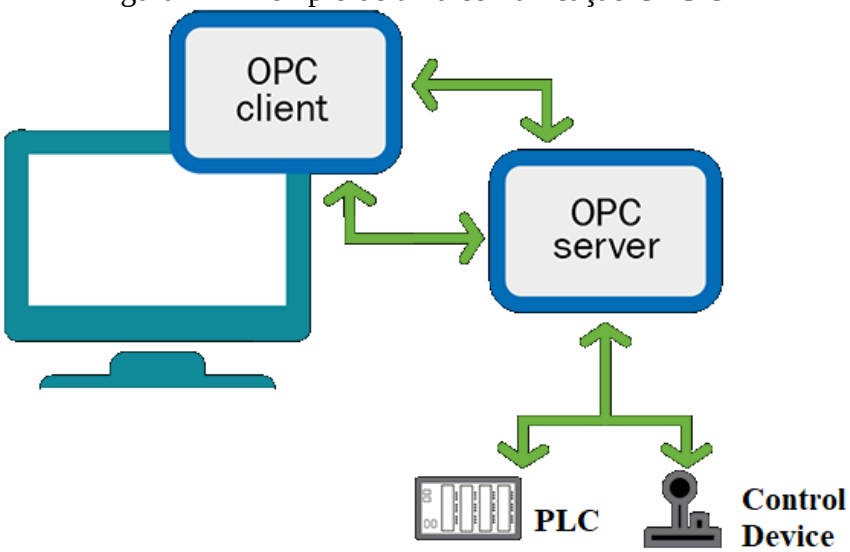
Para resolver estas limitações, a *OPC Foundation* aproveitou algumas tecnologias como exemplo XML (*Extensible Markup Language*) e .NET (.*Network*) para a elaboração de um novo padrão de objeto e arquitetura unificada, completamente orientada a serviços, visando o modelo de fabricação empresarial. Tal elaboração acarretou na criação do OPC UA (*Unified Architecture*) que unifica todas as especificações do seu antecessor e as leva para o estado da arte usando SOA (*Service-Oriented Architecture*) (FONSECA, 2002).

2.4 OPC UA

O OPC UA é um protocolo para comunicação industrial também definido como um padrão independente de uma determinada plataforma mantendo, assim, uma arquitetura baseada em padrões de tecnologias web. Em função disso, o protocolo busca assegurar a interoperabilidade entre seus componentes e apresenta como principal característica a definição de poderosos modelos de informações (ANDRADE, 2019).

Com ilustrado na Figura 2, baseia-se no princípio de cliente-servidor e permite uma comunicação contínua de diversos tipos de dispositivos individuais como módulos de controle ou PLCs (*Programmable Logic Controller*) até o sistema de ERP (*Enterprise Resource Planning*) ou a nuvem (ANDRADE, 2019).

Figura 2 – Exemplo de uma comunicação OPC UA



Fonte: ANDRADE, 2019

Além disso, possui um modelo orientado a serviços para o controle de processos. Esta camada de interoperabilidade contribui com a unificação da troca de informações e fornece uma interface comum (ANDRADE, 2019).

Neste sentido, o OPC UA faz a conexão entre o mundo baseado em IP (*Internet Protocol*) e o chão de fábrica. Com o desenvolvimento deste protocolo, interfaces e gateways, a perda associada de informação passou a ser superada. Ou seja, não há mais a necessidade de sistemas tradicionais de aquisição de dados em tempo real a nível chão de fábrica. Com o OPC UA, toda a informação desejada está disponível para cada aplicação e pessoa autorizada a qualquer momento e em qualquer lugar. Esta função é independente do fabricante do qual as aplicações originam, da linguagem de programação em que foram desenvolvidos ou do sistema operacional (ANDRADE, 2019).

Algumas características do OPC UA que o tornam único:

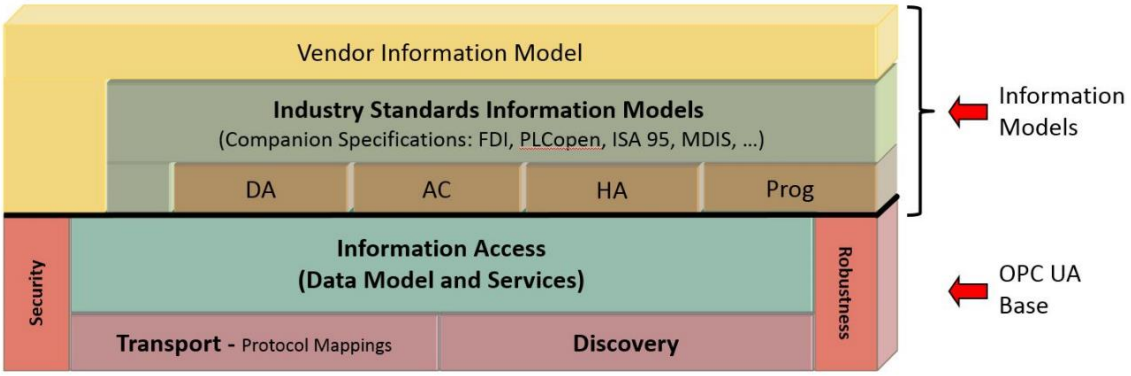
- Disponível gratuitamente (não está limitado a COM/DCOM da *Microsoft*);
- Utilização em plataformas não-Windows como: *Linux, Android* e *Mac OS* (tal desenvolvimento só é possível pelo fato de que a OPC Foundation disponibiliza alguns *Software Development Kits* estruturados em linguagens de programação, como, C/C++, C# e Java);
- Alto Desempenho em comunicação via *Web Services*;
- Integração de todas as ferramentas (dados, alarmes e eventos) em um modelo unificado;
- Suporte a estruturas complexas de dados (dispõe modelos de informações meta dados que podem se estender facilmente);
- Diversos mecanismos de segurança (autenticação, certificado e criptografia) para a comunicação;
- Compatibilidade com o firewall e aumento da proteção contra acessos não autorizados (modelo flexível de segurança que pode ser utilizado nos diferentes níveis de automação, atendendo às exigências para cada ambiente).

2.4.1 Arquitetura e Modelagem OPC UA

A estrutura de modelagem de informações do OPC UA transforma dados em informações. Com recursos completos orientados a objetos, até mesmo as estruturas de vários níveis mais complexas podem ser modeladas e estendidas. Tipos de dados e estruturas são definidos em perfis. Por exemplo, as especificações existentes do OPC Classic foram

modeladas em perfis UA que também podem ser estendidos por outras organizações como mostra a Figura 3 (MELO, 2019).

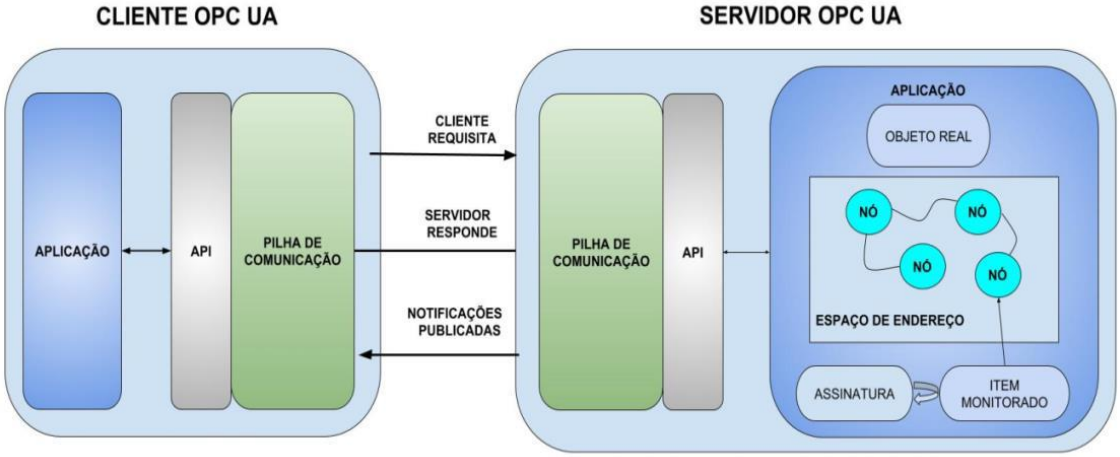
Figura 3 – Modelagem de informações do OPC UA



Fonte: MELO, 2019

Dentro de suas especificações a OPC Foundation afirma que a arquitetura do protocolo OPC UA modela clientes e servidores fazendo-os interagir como parceiros. Dentro de cada sistema pode conter diversos clientes e servidores, sendo que cada cliente pode relacionar-se com um ou mais servidores e o mesmo se aplica para o servidor. Essa combinação de cliente e servidor está ilustrada na Figura 4 (BURKE, 2017).

Figura 4 – Arquitetura do sistema OPC UA



Fonte: MELO, 2019

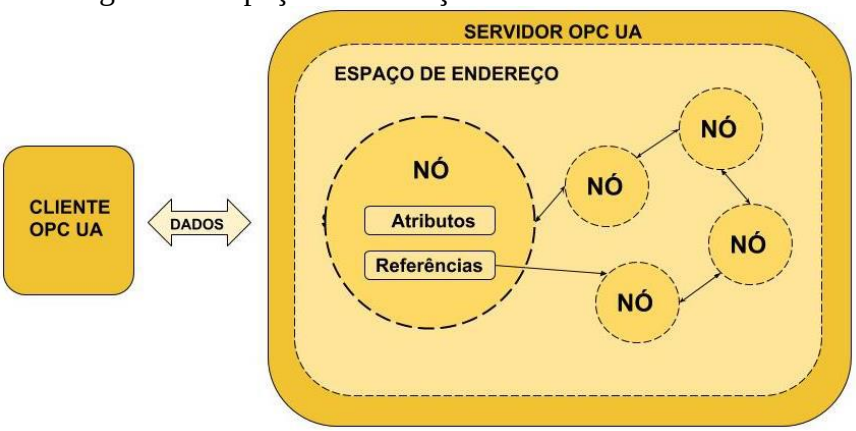
A arquitetura do cliente OPC UA modela o ponto final da integração entre cliente/servidor, resultando assim, na implementação de um cliente que é realizada por meio de uma pilha de comunicação. Para acessar essa pilha, o cliente utiliza uma API. Essa API tem a função de enviar/receber serviços, respostas e requisições do servidor. É importante notar que ela isola o código da aplicação da pilha de comunicação (BURKE, 2017).

Já a arquitetura do servidor OPC UA modela o ponto de extremidade da integração entre cliente/servidor. A aplicação é o código que executa o servidor, usando uma API apropriada para enviar e receber mensagens dos clientes. De forma análoga ao cliente, a API isola o código da aplicação da pilha de comunicação (BURKE, 2017).

2.4.2 Espaço de Endereço OPC UA

Segundo especificação da OPC Foundation, um conjunto de objetos que um servidor OPC UA disponibiliza aos clientes através de dados que representam um sistema em tempo real é chamado de Espaço de Endereço. Como pode ser visto na Figura 5 espaço de endereço pode ser representado por um conjunto de Nós, sendo que esses são descritos por Atributos e interconectados com outros Nós por meio de Referências (MAHNKE; LEITNER; DAMM, 2009).

Figura 5 – Espaço de endereço de um servidor OPC UA



Fonte: MELO, 2019

Cada Nó pode deter um conjunto fixo de atributos. Esses atributos podem ser obrigatórios ou opcionais. Os Atributos são definidos como elementos de dados que descrevem os Nós. Com isso, os clientes OPC UA acessam valores de atributos através de serviços. As Referências são responsáveis por interligar e descrever a relação entre dois Nós (MAHNKE; LEITNER; DAMM, 2009).

2.4.3 Segurança de Dados

A camada de segurança é essencial para execução de uma comunicação segura, sendo responsável por assinar, codificar e decodificar mensagens trocadas entre clientes e servidores OPC UA. A *OPC Foundation* propõe um modelo de segurança, incluindo

diferentes mecanismos e parâmetros. Antes de abrir um canal seguro, o servidor e o cliente devem concordar com o mecanismo de segurança usado para a sessão de comunicação. Para isso, são definidas quatro políticas de segurança para a encriptação dos dados (BURKE, 2017):

- *None* (nenhuma): sem nenhuma configuração de segurança;
- *Basic128Rsa15*: configurações médias de segurança;
- *Basic256*: configurações médias para altas de seguranças;
- *Basic256Sha256*: configurações altas de segurança.

Outro ponto a ser considerado diz respeito à autenticação dos dados. A *OPC Foundation* disponibiliza as seguintes formas para troca de *tokens* de identidade do usuário (BURKE, 2017):

- Anônimo: não é necessária nenhuma identidade de usuário;
- Usuário e Senha: combinações de nomes e senhas para identificar usuário;
- Certificado: par de chaves público/privado para identidade do usuário.

2.4.4 Transporte de Dados

Responsável pelo transporte dos dados entre clientes e servidores, o OPC UA suporta dois protocolos de comunicação:

- Protocolo de transporte TCP (*Transmission Control Protocol*): este protocolo de baseia no TCP para transporte do dado, permitindo um canal *full-dulplex* entre o Servidor e o Cliente, no modelo *Socket* (soquete), o canal se comunica com pacotes binários, permitindo canal seguro (soquete seguro) (VENURELLI, 2019);
- Protocolo SOAP/HTTPS (*Simple Object Access Protocol / Hypertext Transfer Protocol Secure*): Este protocolo trabalha com mensagens estruturadas no modelo SOAP (XML), que são transmitidas via HTTPS. Estas mensagens são conjuntos de dados que são utilizados no nível de informação entre diversos tipos dispositivos que suportam XML, podendo trabalhar com modo seguro, criptografia e certificados digitais, sendo acessíveis por *Firewall* e *Internet*, o que permite um controle de acesso local ou via *Internet* (VENURELLI, 2019).

2.5 Bibliotecas Open Source OPC UA

Atualmente, existem diversas soluções para implementação do OPC UA utilizando código aberto, seja com linguagem de programação em Python, Java, Javascript, C, C++. É necessário entender qual a biblioteca de implementação que melhor combina com os pré-requisitos definidos para o desenvolvimento de um projeto.

2.5.1 Open62541

O projeto open62541 é uma implementação escrita em linguagem C/C++ e licenciado sobre o Mozilla Public License v2.0. A biblioteca do open62541 é utilizável com boa parte dos principais compiladores, além de fornecer as ferramentas necessárias para criação de aplicações de clientes e servidores OPC UA (PROFANTER, 2018). O open62541 é escalável, suportando arquitetura Multi-Threaded, em que cada conexão ou sessão pode ser operada por threads individuais (KOZAR, 2016). Destaca-se que o open62541 é portável, pois suporta POSIX, o que torna o projeto habilitado em diferentes plataformas, como, Windows, Linux, MacOs, Android e Sistemas Embarcados.

O open62541 está na sua terceira versão (v0.3) e disponibiliza uma extensa e excelente documentação para suportar a criação de clientes e servidores OPC UA. A respeito da utilização do open62541, Contrenas (2017) expõe as características essenciais que permitem que um sistema de fabricação seja adaptado para se tornar um aplicativo da I4.0. Neste trabalho a parte de comunicação é implementada usando o OPC UA. Johansson (2017) discute como uma simulação interativa de modelos de sistemas físicos para o OpenModelica pode ser projetada usando um cliente OPC UA.

O estudo proposto por Muller (2017), apresenta um servidor OPC UA funcionando através de uma placa Arduino Yun.

2.5.2 Eclipse Milo

O Eclipse Milo foi apresentado durante a EclipseCon Europe em 2016 (Eclipse, 2016). O Eclipse Milo, parte do Eclipse IoT Project, é uma implementação de código aberto do OPC UA baseado na linguagem de programação Java. O projeto é desenvolvido pela Fundação Eclipse que fornece uma boa documentação e licenciado sob o EPL -1.0. O Eclipse

Milo oferece uma pilha e SDK em que há um núcleo de código comum compartilhado entre cliente e servidor (REIMANN, 2018).

A respeito da utilização do Eclipse Milo, a Eclipse (2018) apresenta o conceito de um adaptador de dispositivos que permite que o dispositivo seja facilmente conectado a sistemas de produção usando o OPC UA. O projeto apresentado por Profanter (2017), avalia algumas implementações de código aberto e certifica a Eclipse Milo como uma escolha perfeita para um sistema de produção multinível Plug & Produce.

2.5.3 ASNeg

O ASNeg (Automation Service Next Generation – Serviço de Automação de Próxima Geração) é uma implementação desenvolvida em C++ e licenciado sob o Apache 2.0. Além de fornecer soluções relacionadas a clientes e servidores OPC UA, o ASNeg conta com 9 repositórios para outras aplicações, incluindo OPCUADB (Base de dados do Servidor), OPCUaWebServer e OPCUARaspberry (HUEBL, 2017).

2.5.4 UA.Net

Desenvolvido diretamente por membros da OPC Foundation, o UA.NET é uma implementação desenvolvida em C# usando o .NET Standard Library (MOLDOVEN; BARNSTED; OURSEL, 2018). O UA.NET fornece uma pilha OPC UA contendo uma série de aplicativos de amostras. O UA.NET é licenciado sobre o GLP e RCL. O RCL permite que os membros da OPC Foundation implementem suas aplicações usando a pilha UA.NET sem que seja obrigatório divulgar o código do aplicativo. Enquanto isso, os não membros devem divulgar o código de suas aplicações ao usar a pilha UA.NET (OPC FOUNDATION, 2016). O .Net Standard Library permite que se desenvolva aplicativos que funcionem em todas as plataformas comuns disponíveis, incluindo Linux, iOS, Android (via Xamarin) e Windows 7/8/8.1/10 (incluindo edições embarcadas/IoT), sem requerer modificações específicas da plataforma. Além disso, também são suportados aplicativos e serviços em nuvem (como ASP.Net, DNX, Azure Web, Azure Webjobs, Azure Nano Server e Azure Service Fabric) (MOLDOVEN; BARNSTED; OURSEL, 2018).

2.5.5 FreeOPCUA

O FreeOPCUA é um projeto composto por um conjunto de bibliotecasdo OPC UA, construídas em C++ e Python. O FreeOPCUA conta com uma biblioteca LGPL C++ para desenvolvimento de aplicações de clientes e servidores OPC UA, utilizando a linguagem de programação C++ que é denominada “freeopcua”. Outra possibilidade é pela aplicação da biblioteca escrita em Python, fornecida através do repositório no GitHub, chamado “phyton opcua” (RYKONANOV, 2018). Além disso, o FreeOPCUA fornece duas soluções de GUI (Graphical User Interface): opcua-client-gui e opcua-modeler. A primeira implementa as funcionalidades básicas, incluindo a alteração de dados e eventos, escrita de valores nas variáveis, listagem de atributos e referências. A segunda é usada para projetar espaços de endereços usando XML, o que permite que o XML produzido seja portável para qualquer SDK (RYKONANOV, 2018).

A respeito da utilização do FreeOPCUA, Moldoven, Barnsted e Oursel (2018) usam o OPC UA num modelo proposto para implantar o conceito de metrologia para a Industria 4.0. Akerman, Berglund e Ekered (2016) apresentam soluções que exemplificam a integração vertical e horizontal de sistemas usando OPC UA. Dentro dessas soluções é implementado um sistema RFID combinado com a implementação FreeOPCUA.

3. MATERIAIS E MÉTODOS

3.1 Proposta do Trabalho

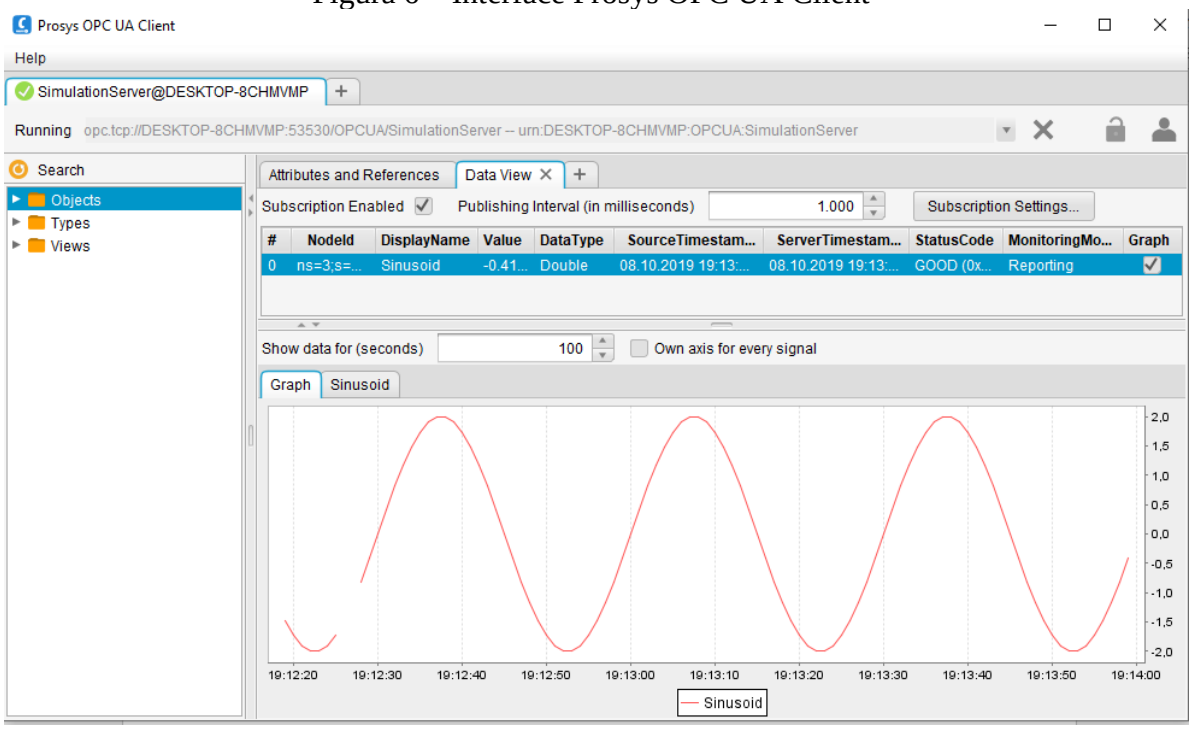
A proposta de trabalho, consiste na implementação de um servidor OPC UA em um microcontrolador de baixo custo, no caso, o microcontrolador ESP32, utilizando a biblioteca Open62541 em linguagem C, de modo que seja possível realizar o controle de suas entradas e saídas digitais e analógicas, se comunicando com um cliente por meio dos softwares Prosys OPC UA Client Mobile e o UA Expert Client.

3.2 Prosys

3.2.1 Prosys OPC UA Client

O Prosys OPC Client (PROSYS OPC, 2022) consiste em uma ferramenta que visa ajudar a solucionar problemas de conexões OPC UA, bem como realizar testes, sendo possível ler e gravar informações, navegar entre os espaços do OPC UA e monitorar os valores captados via microcontrolador em tempo real, com pode ser visto na Figura 6. É uma plataforma rápida, leve e de fácil utilização.

Figura 6 – Interface Prosys OPC UA Client

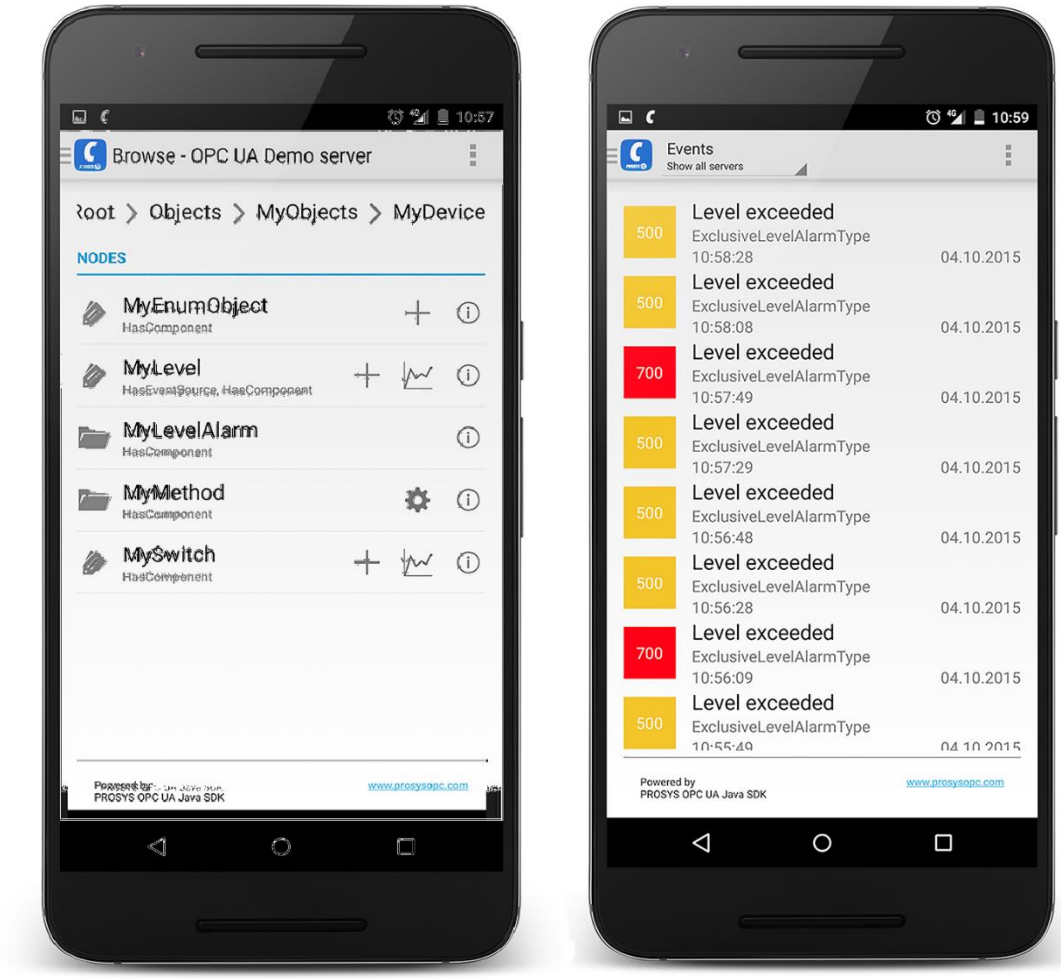


Fonte: Autor Desconhecido

3.2.2 Prosys OPC UA Client Mobile

O Prosys OPC UA Client Mobile para Android (PROSYS OPC, 2022) trata-se de um aplicativo otimizado para dispositivos móveis que compreende recursos essenciais do aplicativo de desktop Prosys OPC UA Client. Nele também é possível navegar no espaço de endereços do OPC UA, ler e gravar dados e receber notificações de eventos. Ele apresenta uma interface simples e de fácil utilização como pode ser visto na Figura 7.

Figura 7 – Interface Prosys OPC UA Client Mobile



Fonte: PROSYS OPC, 2022

3.3 UA Expert

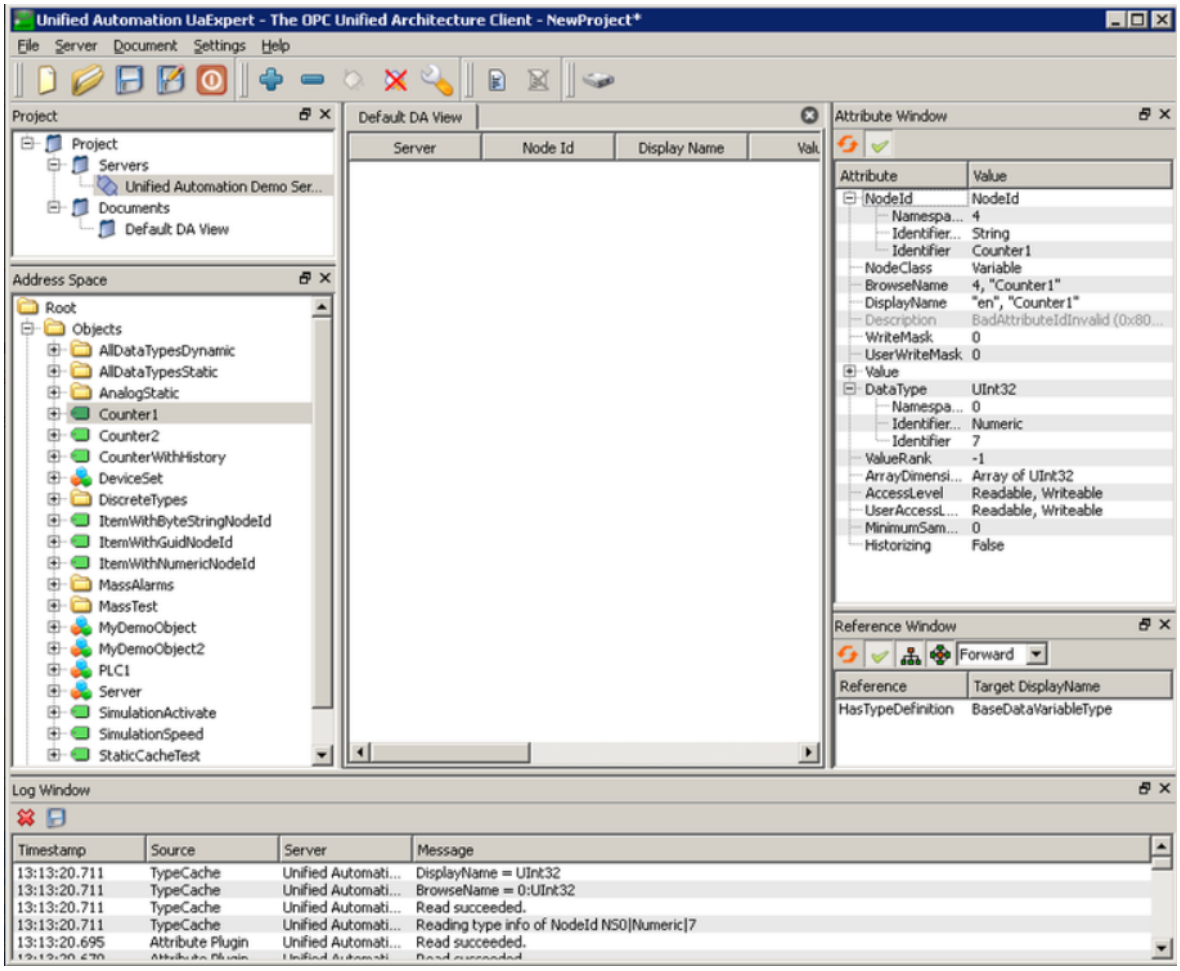
3.3.1 UA Expert Client

O UaExpert (UA EXPERT, 2022) trata-se de um cliente OPC UA projetado como um cliente de teste de uso geral que suporta recursos OPC UA como *DataAccess*, *Alarms* & *Conditions*, *Historical Access* e chamada de Métodos UA, vale ressaltar que é um cliente de teste OPC UA multiplataforma programado em C++, usando a biblioteca GUI QT da Nokia formando a estrutura básica que é extensível por Plugins. A versão gratuita do UaExpert vem com os seguintes plugins:

- Visualização de Acesso a Dados OPC UA
- Visualização de Alarmes e Condições do OPC UA
- Visualização de tendências históricas do OPC UA
- Visualização de diagnóstico do servidor
- Plugin CSV do Registrador de Dados Simples
- Plug-in de desempenho OPC UA
- Plugin de modelo push GDS
- XMLNodeSet-Export View (requer licença)
- O UaExpert está disponível para Windows e Linux.

A estrutura básica do UaExpert (UA EXPERT, 2022) inclui funcionalidades gerais como manipulação de certificados, descoberta de servidores UA, conexão com servidores UA, navegação no modelo de informações, exibição de atributos e referências de nós UA específicos. A Figura 8, a seguir, mostra um exemplo de um projeto OPC UA utilizando a UA Expert:

Figura 8 – Interface UA Expert



Fonte: UA EXPERT, 2022

O painel *Project* mostra os servidores UA conectados e os plug-ins de documentos abertos. O painel *Address Space* mostra o modelo de informações dos Servidores UA. Dependendo do *Node* selecionado, os painéis *Attribute Window* e *Reference Window* mostram o atributo do *Node* selecionado e suas referências dentro da rede em malha do espaço de endereço dos servidores.

3.4 Microcontrolador

Quando se fala em aplicações de automatização em uma escala pequena, sendo estas aplicações que não demandam alta velocidade de processamento, alta potência ou um número considerável de entradas/saídas, nomes relacionados a família Arduíno, a família dos Raspberry PI, Beaglebone, Galileu, ESP32, entre outros, sempre vêm à tona pois, apesar de serem microcontroladores iniciais, como mostra a Figura 9 abaixo, possuem um grande

potencial para diversas aplicações justamente por possuírem alta versatilidade, somados ao seu baixo custo, além de possuírem ambientes de desenvolvimento gratuitos.

Figura 9 – Variedades de microcontroladores, na imagem: Arduino UNO e NANO, Raspberry PI, ESP32, ESP8266.

Fonte: Autor Desconhecido

3.4.1 ESP32

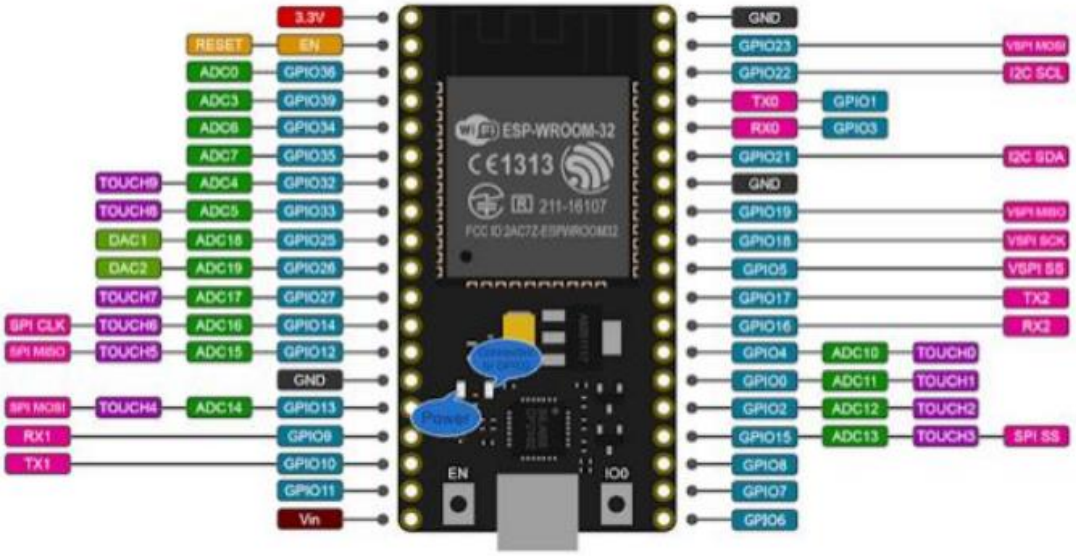
Dentre os microcontroladores mencionados, para este projeto será utilizado o ESP 32, que pode ser observado na Figura 10 abaixo, um dispositivo IoT (Internet das Coisas) que consiste em um microprocessador de baixa potência dual core Tensilica Xtensa 32-bit LX6 com suporte embutido à rede WiFi, Bluetooth e memória flash integrada. Essa arquitetura permite que ele possa ser programado de forma independente, sem a necessidade de outras placas microcontroladores como o Arduino, por exemplo.

Dentre as principais características deste dispositivo, podemos citar: baixo consumo de energia, alto desempenho de potência, amplificador de baixo ruído, robustez, versatilidade e confiabilidade.

Assim, pode-se dizer que a ESP32, segue a mesma linha do Arduíno, já que pode apresentar o mesmo ambiente de desenvolvimento, o Arduino IDE, linguagem de programação C/C++, baixo custo, entre outras características, bem como suas diferenças, já que possui uma memória interna maior, o fato de já vir com placa Wi-Fi e Bluetooth

integrados, o que acaba facilitando sua aplicação em Internet das Coisas (Internet of Things – IoT).

Figura 10 – Pinagem do ESP 32



Fonte: Autor Desconhecido

Dentre suas principais especificações têm-se:

– CPU: Xtensa® Dual-Core 32-bit LX6

– ROM: 448 Kbytes

– RAM: 520 Kbytes

– Flash: 4 MB

– Clock máximo: 240MHz

– Wireless padrão 802.11 b/g/n

– Conexão Wi-Fi 2.4Ghz (máximo de 150 Mbps)

– Antena embutida

– Conector micro-usb

– Wi-Fi Direct (P2P), P2P Discovery, P2P Group Owner mode e P2P Power Management 14

– Modos de operação: STA/AP/STA+AP

– Bluetooth BLE 4.2

– Portas GPIO: 11

– GPIO com funções de PWM, I2C, SPI, etc

– Tensão de operação: 4,5 ~ 9V

– Taxa de transferência: 110-460800bps
– Suporta Upgrade remoto de firmware
– Conversor analógico digital (ADC)
– Distância entre pinos: 2,54 mm
– Dimensões: 52 mm x 28 mm x 5 mm (desconsiderando os pinos)

3.4.2 Espressif

A Espressif Systems (688018.SH) consiste em uma empresa pública de semicondutores, fundada em 2008, focados no desenvolvimento de soluções de comunicação sem fio de ponta, baixo consumo de energia e AIoT. Criador das populares séries de chips, módulos e placas de desenvolvimento ESP8266, ESP32, ESP32-S, ESP32-C e ESP32-H.

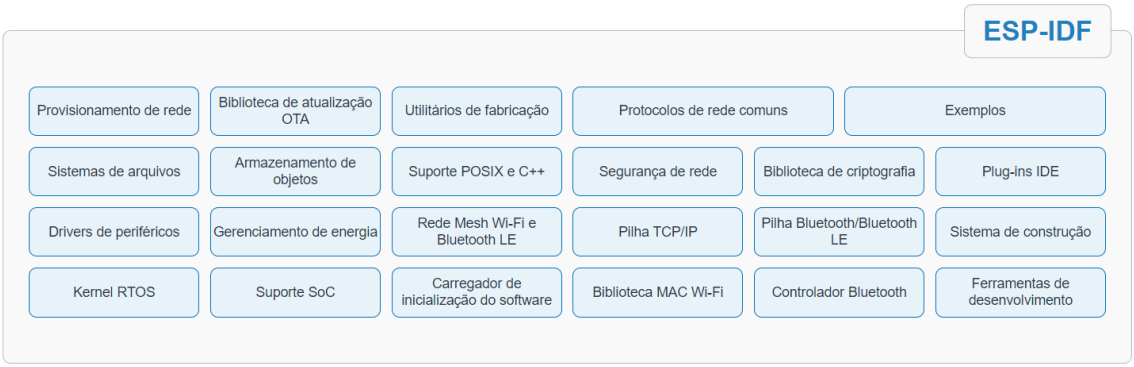
Fornecem softwares de código aberto, para construção de aplicativos AIoT, como o Espressif's IoT Development Framework ESP-IDF, Audio Development Framework ESP-ADF, Mesh Development Framework ESP-MDF, Device Connectivity Platform ESP RainMaker, Facial Recognition Development Framework ESP-WHO e Smart Voice Assistant ESP -Skainet.

3.2.2.1 ESP-IDF

A ESP-IDF (ESPRESSIF SYSTEMS, 2022) trata-se da estrutura de desenvolvimento de IoT oficial da Espressif para as séries de SoCs ESP32, ESP32-S e ESP32-C. Fornece um SDK autossuficiente para o desenvolvimento de aplicativos genéricos nessas plataformas, usando linguagens de programação como C e C++. Está disponível de forma gratuita no GitHub e a maioria dos seus componentes está disponível na forma de origem sob a licença Apache 2.0.

É capaz de suportar um grande número de componentes de software, incluindo RTOS, drivers periféricos, pilha de rede, várias implementações de protocolo e auxiliares para casos de uso de aplicativos comuns, que contribuem para o foco na lógica de processo, enquanto o SDK fornece a maioria dos blocos de construção necessários para aplicativos típicos, como mostra a Figura 11.

Figura 11 – Componentes ESP-IDF



Fonte: ESPRESSIF SYSTEMS, 2022

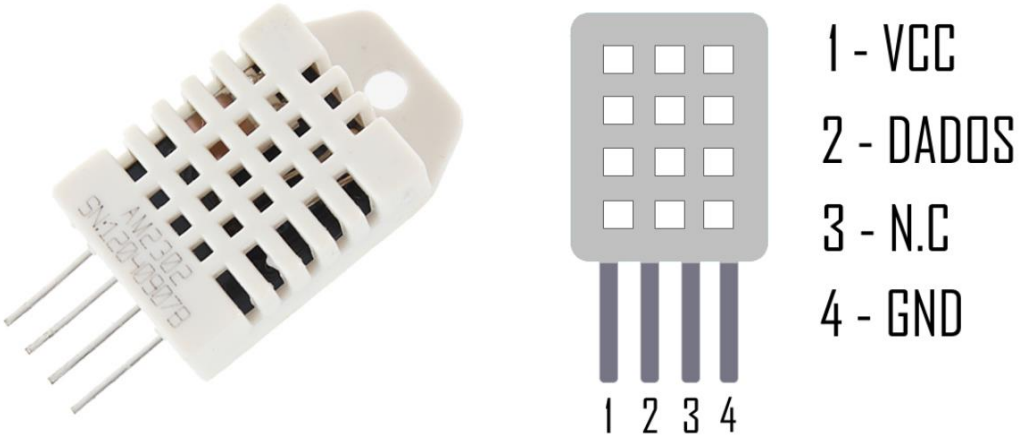
A ESP-IDF contém diversos exemplos explicando o uso de seus componentes, bem como seus periféricos e recursos de hardware. Tais exemplos forneceram um excelente ponto de partida para o desenvolvimento do projeto.

3.5 Componentes eletrônicos

3.5.1 DHT22

O DHT22, representado na Figura 12, é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre -40 a +80 graus Celsius e umidade entre 0 a 100%, e possui apenas 1 pino com saída digital. Este sensor é formado por um sensor de umidade capacitivo e um termistor para medir o ar ao redor, enviando no pino de dados um sinal digital (Aosong (Guangzhou) Electronics Co.,Ltd., 2022).

Figura 12 – Sensor de Temperatura e Umidade DHT22



Fonte: Autor Desconhecido

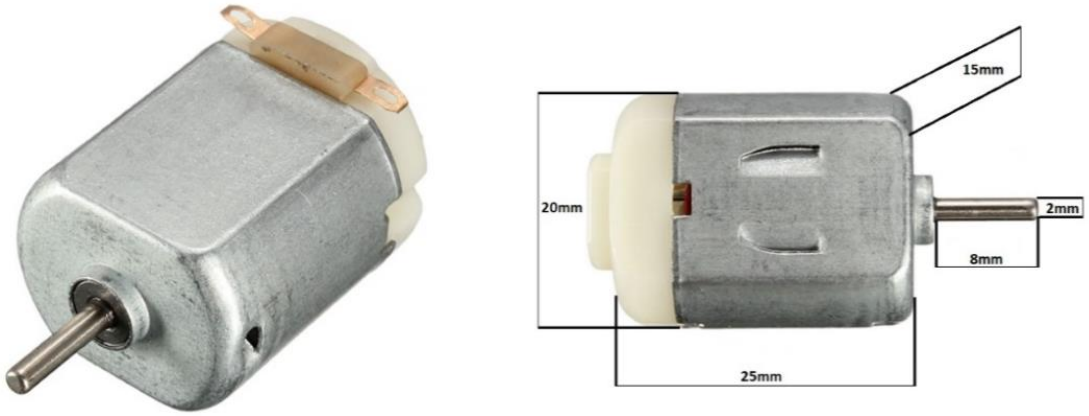
Especificações:

– Modelo: AM2302 (datasheet) REF: 9SS25
– Tensão de operação: 3-5VDC (5,5VDC máximo)
– Faixa de medição de umidade: 0 a 100% UR
– Faixa de medição de temperatura: -40º a +80ºC
– Corrente: 2,5mA max durante uso, em stand by de 100uA a 150 uA
– Precisão de umidade de medição: ± 2,0% UR
– Precisão de medição de temperatura: ± 0,5 ºC
– Resolução: 0,1
– Tempo de resposta: 2s
– Dimensões: 25 x 15 7mm (sem terminais)

3.5.2 Motor DC

Foi utilizado um mini motor DC 3V, recomendado para projetos compactos com microcontroladores, como Arduino, ESP32, e que necessitam de alta rotação, como mostra na Figura 13 abaixo:

Figura 13 – Motor DC 3V



Fonte: Autor Desconhecido

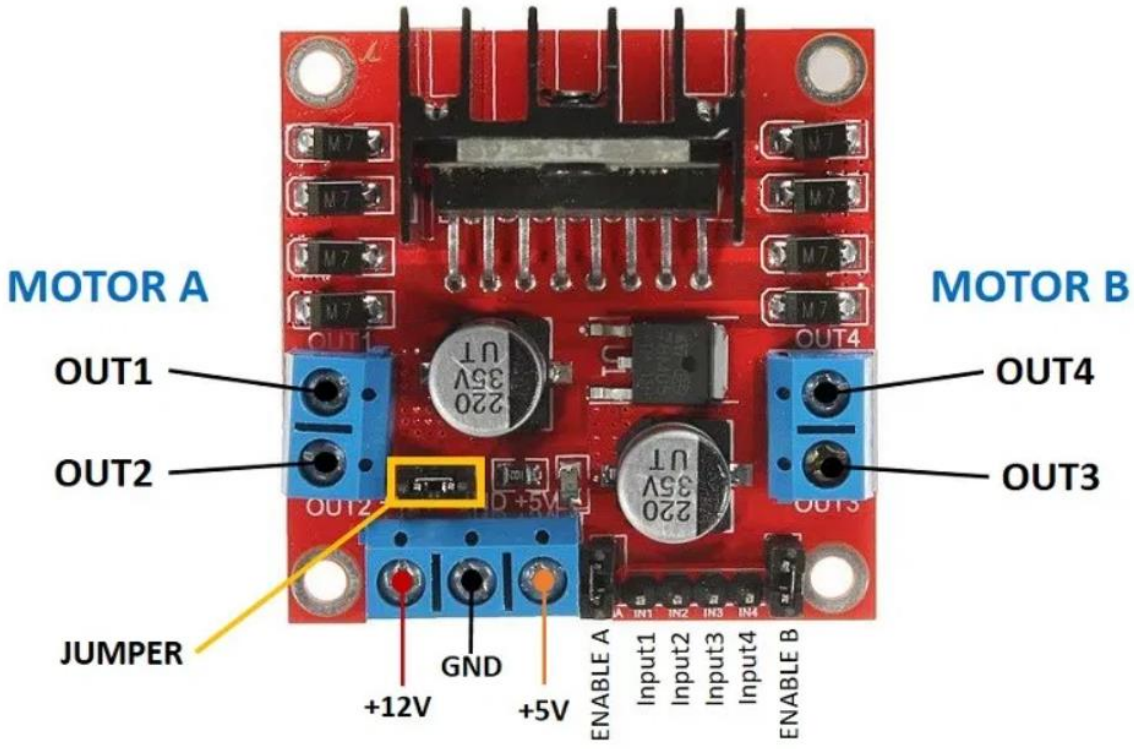
Especificações:

– Tensão de operação: 3-6V
– Diâmetro do eixo: 2mm
– Comprimento do eixo: 9mm
– Dimensões: 38 x 20 x 15mm
– Peso: 14g
– SKU: 7MS73

3.6 Driver Motor Ponte H – L298N

Para acionamento do motor, foi utilizado o driver ponte H, este driver é baseado no chip L298N, construído para controlar cargas indutivas como relés, solenoides, motores DC e motores de passo, nele é possível controlar independentemente a velocidade e rotação de 2 motores DC ou 1 motor de passo, e também possui terminais parafusáveis para realizar as conexões, como mostra a Figura 14 abaixo:

Figura 14 – Driver Ponte H L298N



Fonte: SANTOS, 2018

Especificações:

– Tensão de Operação: 4~35v
– Chip: ST L298N (datasheet)
– Controle de 2 motores DC ou 1 motor de passo
– Corrente de Operação máxima: 2A por canal ou 4A max
– Tensão lógica: 5v
– Corrente lógica: 0~36mA
– Limites de Temperatura: -20 a +135°C
– Potência Máxima: 25W
– Dimensões: 43 x 43 x 27mm
– Peso: 30g

O driver possui dois blocos com os terminais OUT1 - OUT2 e OUT3 - OUT4 sendo:

- OUT1: Motor DC A + terminal
- OUT2: Motor DC A - terminal
- OUT3: Motor DC B + terminal
- OUT4: Motor DC B - termina

A parte inferior possui um bloco de três terminais com +12V, GND, e +5V. O +12V é utilizado para ligar os motores. O terminal +5V é utilizado para ligar o chip L298N. No entanto, se o jumper estiver conectado, o chip é alimentado usando a fonte de alimentação do motor e não é necessário fornecer 5V por meio do terminal de +5V. (SANTOS, 2018)

No canto inferior direito tem-se quatro pinos *input* e dois terminais de *enable*. Os pinos *inputs* são usados para controlar a direção dos motores CC e os pinos *enable* são usados para controlar a velocidade de cada motor, sendo eles:

- IN1: *Input* 1 para Motor A
- IN2: *Input* 2 para Motor A
- IN3: *Input* 1 para Motor B
- IN4: *Input* 2 para Motor B
- EN1: *Enable* 1 para Motor A
- EN2: *Enable* 2 para Motor B

Assim, acionando os pinos *input*, é possível controlar o sentido de rotação do motor, como mostra a Tabela 1 a seguir:

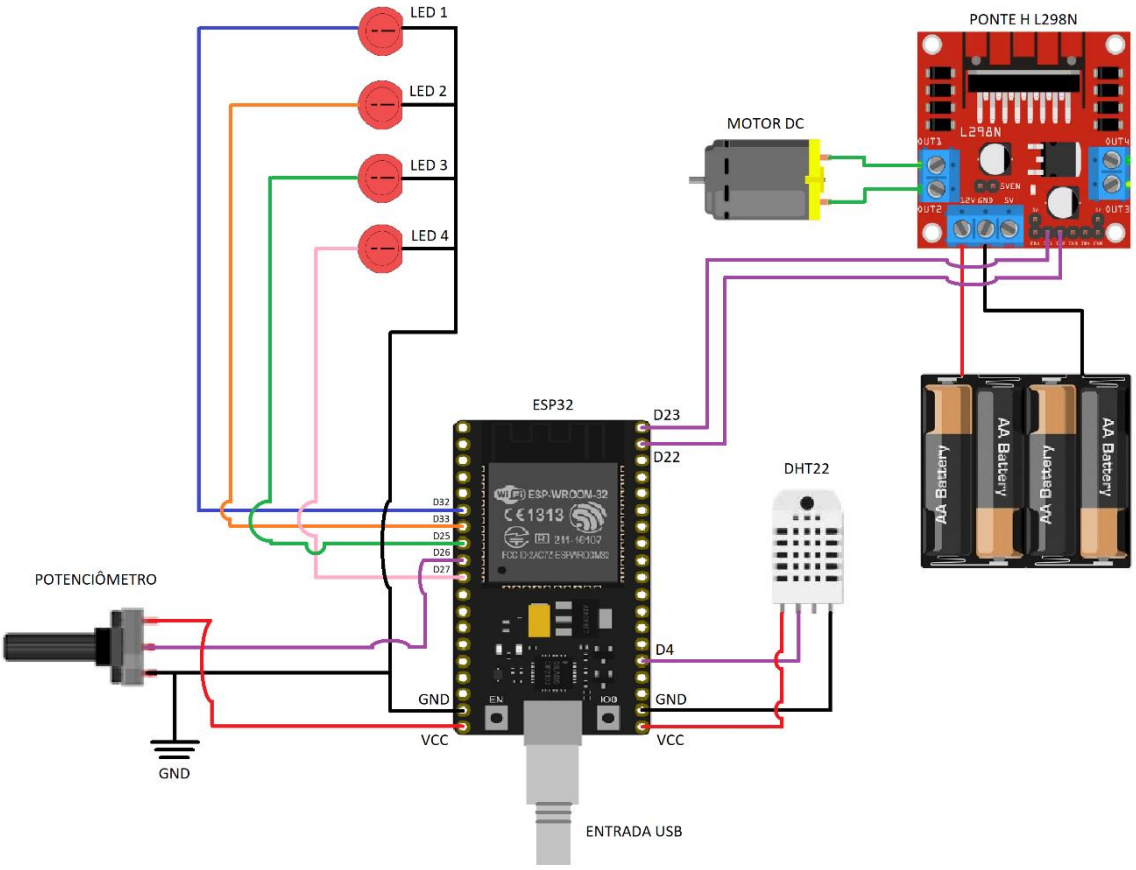
Tabela 1 – Acionamento do Motor DC utilizando driver Ponte H L298N

Sentido	**IN1**	**IN2**	**IN3**	**IN4**
Horário	0	1	0	1
Anti-Horário	1	0	1	0
Parado	0	0	0	0

3.7 Montagem do circuito

Com os devidos componentes eletrônicos selecionados, levando em consideração suas especificações e condições de uso, foi realizada a montagem do circuito a ser implementado. O microcontrolador ESP32 é a peça central e os componentes vão se conectando em suas portas I/O. Assim, foi construído o seguinte diagrama representado na Figura 15 abaixo:

Figura 15 – Projeto Circuito Eletrônico



Fonte: Autoria Própria

A entrada USB do ESP32 é do tipo Micro USB, e é dessa conexão a qual o microcontrolador é alimentado, bem como o sensor DHT22, o potenciômetro e as trilhas dos LEDs. As pilhas são necessárias para o funcionamento do motor DC e do driver Ponte H L298N, que necessitam de uma tensão maior de operação.

3.8 Open62541

A open62541 (http://open62541.org) é uma implementação gratuita e de código aberto do OPC UA escrita no subconjunto comum das linguagens C e C++. A biblioteca pode ser usada com todos os principais compiladores e fornece as ferramentas necessárias para implementar clientes e servidores OPC UA dedicados ou para integrar a comunicação baseada em OPC UA em aplicativos existentes, tratando-se de uma biblioteca independente de plataforma. Todas as funcionalidades específicas da plataforma são implementadas através de plugins intercambiáveis, tais implementações são fornecidas para os principais sistemas operacionais.

A biblioteca está disponível em fonte padrão e formato binário. Além disso, a distribuição de origem de arquivo único mescla toda a biblioteca em um único arquivo .c e .h que pode ser facilmente adicionado a projetos existentes. Tem como principais características:

1) Comunicação Stack:

- Protocolo binário OPC UA;
- Codificação OPC UA JSON;
- Comunicação segura com mensagens criptografadas;
- Camada de rede intercambiável (plugin) para usar APIs de rede personalizadas (por exemplo, em destinos incorporados);
- Suporte para gerar tipos de dados a partir de definições XML padrão.

2) Servidor:

- Suporte para todos os tipos de nó OPC UA;
- Controle de acesso para nós individuais;
- Suporte para gerar modelos de informações do lado do servidor a partir de definições XML padrão (conjuntos de nós);
- Suporte para adicionar e remover nós e referências também em tempo de execução;
- Suporte para herança e instanciação de tipos de objetos e variáveis (construtor/destruidor personalizado, instanciação de nós filhos);
- Suporte para assinaturas (notificações e eventos de alteração de dados).

3) Client:

- Todos os serviços OPC UA suportados;
- Solicitações de serviço assíncronas;
- Tratamento de assinaturas em segundo plano.

4) Publicar/Assinar:

- Protocolo binário UADP com comunicação UDP-multicast ou Ethernet;
- Codificação PubSub JSON.

5) Segurança:

- Servidor de dispositivo micro incorporado
- Faceta do servidor de método
- Gerenciamento de nós
- Políticas de segurança:
    - Básico128Rsa15
    - Básico 256
    - Básico256Sha256
- Tokens de usuário:
    - Faceta anônima
    - Nome de usuário/senha do servidor

3.9 Código Base – Github

Para desenvolvimento do projeto, foi utilizado um código fonte base (público), que pode ser encontrado como um projeto no *GitHub* como "*Pro/open62541-esp32*" ou como "*OPC UA on a ESP32 Microcontroller*". O projeto *git* consiste na implementação de um servidor OPC UA em um microcontrolador ESP32 utilizando a biblioteca de código aberto open62541. Neste projeto, o autor realiza o controle de dois LEDs, um sensor de temperatura DHT22 e o controle do LED interno da placa. Deste modo, o código fonte foi adaptado para o contexto da proposta do presente trabalho, trazendo funcionalidades mais abrangentes, de modo que fosse possível controlar entradas e saídas analógicas e digitais do ESP32, bem como realizar um controle PWM de um motor DC, aplicado a um modelo educacional, no qual o trabalho possa ser utilizado como base para outros tipos de desenvolvimentos.

4. DESENVOLVIMENTO E COMUNICAÇÃO

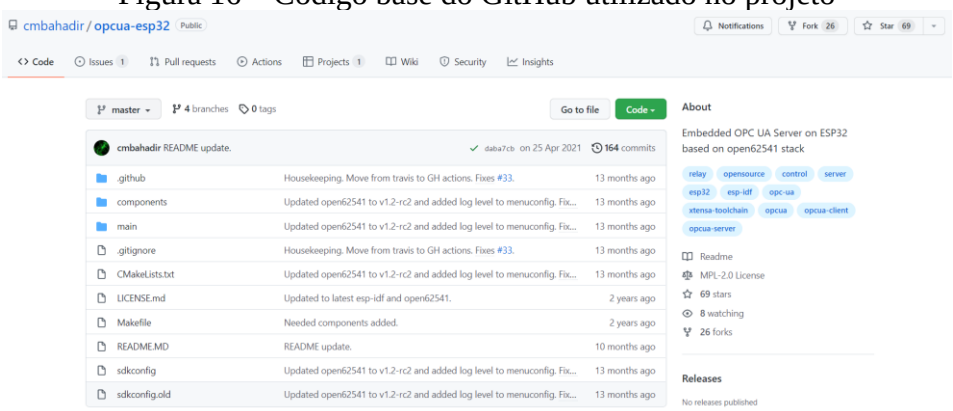
Como apresentado nos capítulos acima, o propósito deste trabalho foi implementar um servidor OPC UA dentro do ESP32. A proposta focou em disponibilizar o acesso, via um servidor OPC UA, de portas de entradas e portas de saída do ESP 32. Deste modo, foram criados recursos para acessar portas de entrada digital, portas de entrada analógica, portas de saída digital, portas de saída analógica e controle PWM, como mostra a Tabela 2 abaixo. Para isso, foi necessário entender o uso dos métodos de comunicação para implementar cada recurso, tomando como base a documentação fornecida pela Espressif.

Tabela 2 – Funcionalidades implementadas

	Digital		*Analógica*		*PWM*
	Entrada	Saída	Entrada	Saída	Saída
Acender LEDs		X			
Ler Temperatura DHT22	X				
Ler Tensão Potenciômetro			X		
Luminosidade LED				X	
Velocidade Motor DC					X
Sentido de Rotação Motor DC		X			
Piscar LED Interno		X			

Inicialmente foi realizado as configurações e instalações dos softwares utilizados, listados no capítulo 3, bem como o download do código open source (OPCUA-ESP32, GitHub) que foi utilizado como base, como mostra a Figura 16 abaixo. Para controle mobile, foi realizado a instalação do aplicativo Prosys OPC UA Cliente e para controle via PC, foi utilizado o OPC UA Expert.

Figura 16 – Codigo base do GitHub utilizado no projeto



Fonte: Autoria Própria

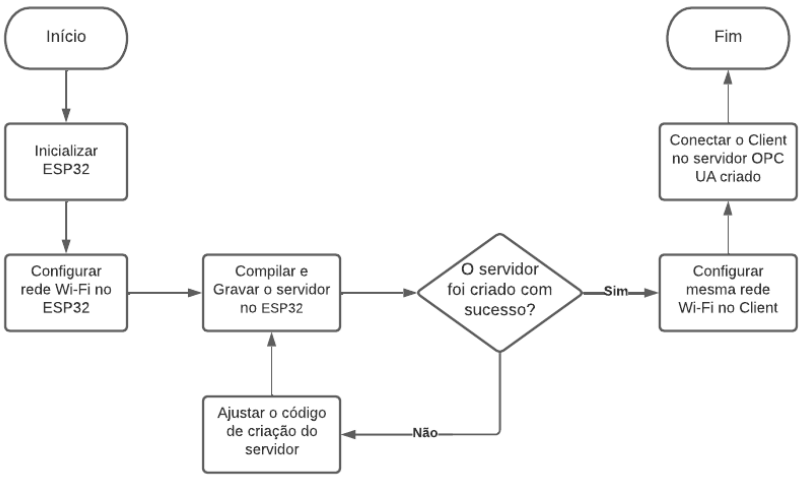
Para a implementação das novas funcionalidades propostas no trabalho, foi alterado o código base, utilizando o VS Studio Code (MICROSOFT, 2020). Deste modo, foram alterados os arquivos "*opcua_esp32.c*" localizado na pasta "*main*" de modo a adicionar a chamada das novas funções implementadas, e os arquivos "*model.c*" e "*model.h*" localizados na pasta "*open62541lib*" com os códigos das funções novas. Vale ressaltar que as funcionalidades novas foram declaradas nas seguintes portas do ESP32 descritas na Tabela 3 abaixo:

Tabela 3 – I/O das funcionalidades implementadas

	Porta ESP32
LED Interno da placa	D2
Sensor de Temperatura	D4
Motor INPUT 2	D22
Motor INPUT 1	D23
LED 3 (DAC)	D25
Potenciômetro	D26
LED 4	D27
LED 1	D32
LED 2	D33

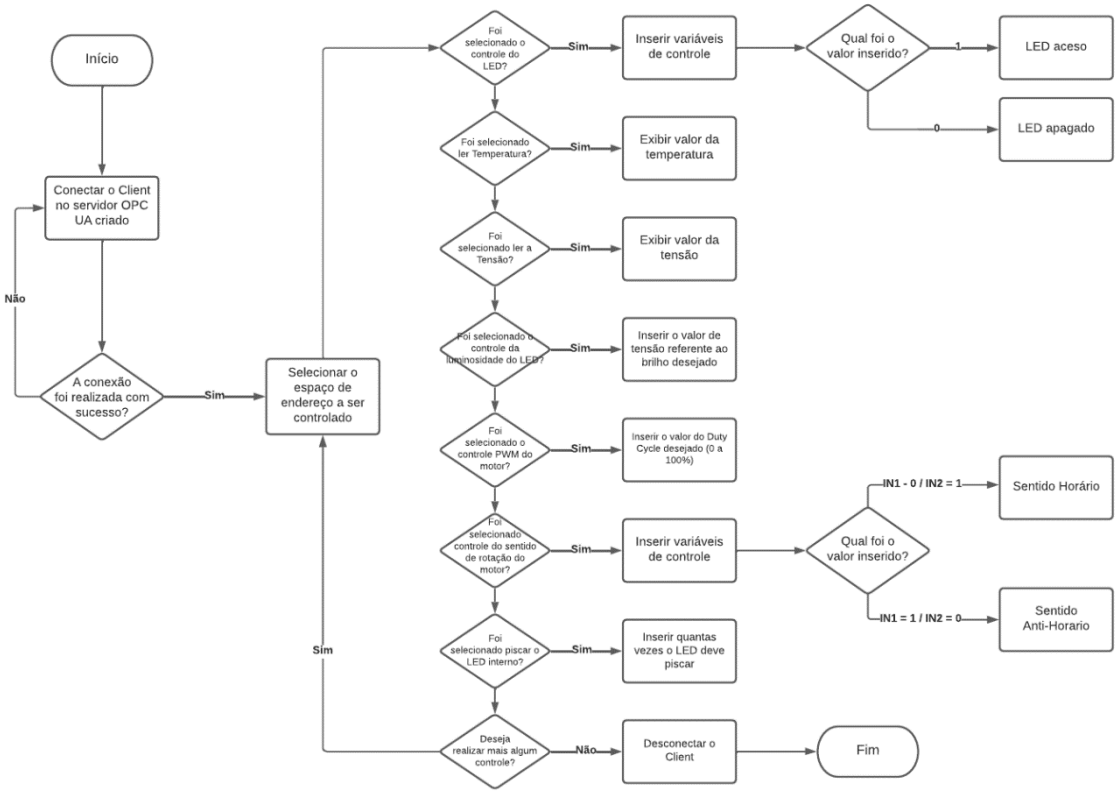
Assim, montou-se os fluxogramas representados nas Figura 17 e Figura 18 a seguir para exemplificar o fluxo de conexão e o fluxo de controle do servidor OPC UA e suas funcionalidades, respectivamente:

Figura 17 – Fluxo de conexão com o servidor OPC UA criado



Fonte: Autoria Própria

Figura 18 – Fluxo de controle das funcionalidade do servidor OPC UA criado



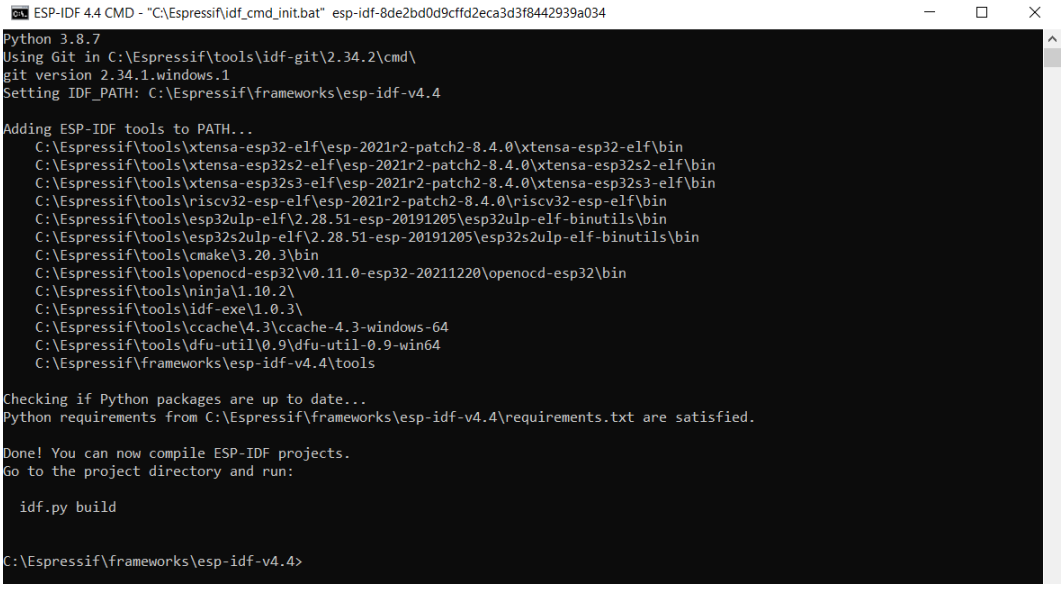
Fonte: Autoria Própria

O código desenvolvido para controle das funcionalidades começa a funcionar no momento em que o servidor é inicializado e, assim que o espaço de endereço é selecionado para controle, o código chama a função atribuída naquele espaço de endereço do servidor OPC UA e executa o controle programado ou aguarda inserir as variáveis de controle conforme a aplicação desejada.

4.1 Implementação do Servidor OPC UA no ESP32

Para a implementação do servidor OPC UA no ESP32 foi realizada a instalação da ESP-IDF, que pode ser encontrada no site da ESPRESSIF (ESPRESSIF, ESP-IDF). Após concluir o processo de instalação, é necessário abrir a ESP-IDF. A interface é representada na Figura 19 a seguir:

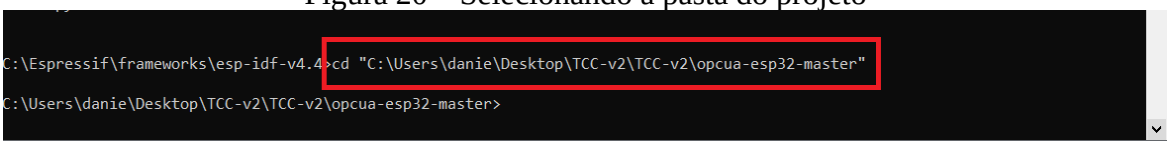
Figura 19 – Exemplo Interface ESP-IDF



Fonte: Autoria Própria

Com a interface aberta, é necessário selecionar o caminho onde está o código do projeto. Deste modo, utilizou-se o comando: ***cd "endereço do arquivo"***, como mostra a Figura 20 abaixo:

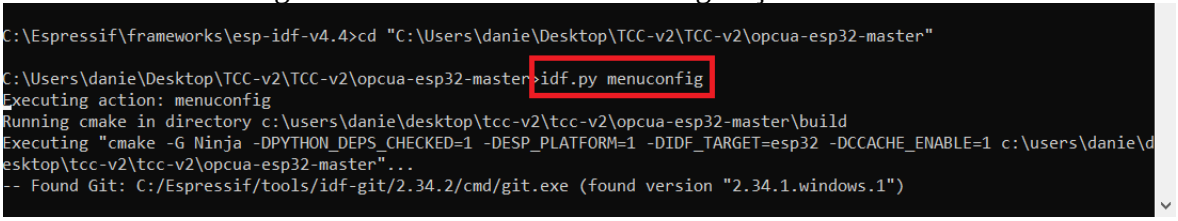
Figura 20 – Selecionando a pasta do projeto



Fonte: Autoria Própria

Após acessar a pasta do projeto, deve-se realizar as configurações do ESP32. Elas são realizadas no menu de configuração, onde é possível realizar as configurações de conexão Wi-Fi. Para acessá-la foi utilizado o comando: ***idf.py menuconfig***, representado na Figura 21 a seguir:

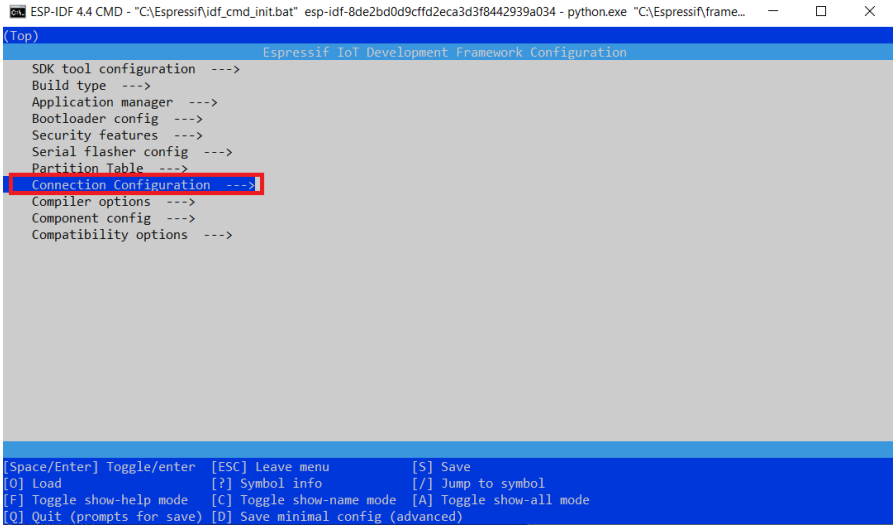
Figura 21 – Acessar menu de configuração do ESP32



Fonte: Autoria Própria

Com o menu de configuração aberto, é necessário conectar o ESP32 na mesma rede Wi-Fi na qual o device (celular ou PC) está conectado. Para isso, acessou-se o menu *Connection Configuration*, Figura 22, onde foi inserido as credenciais da rede.

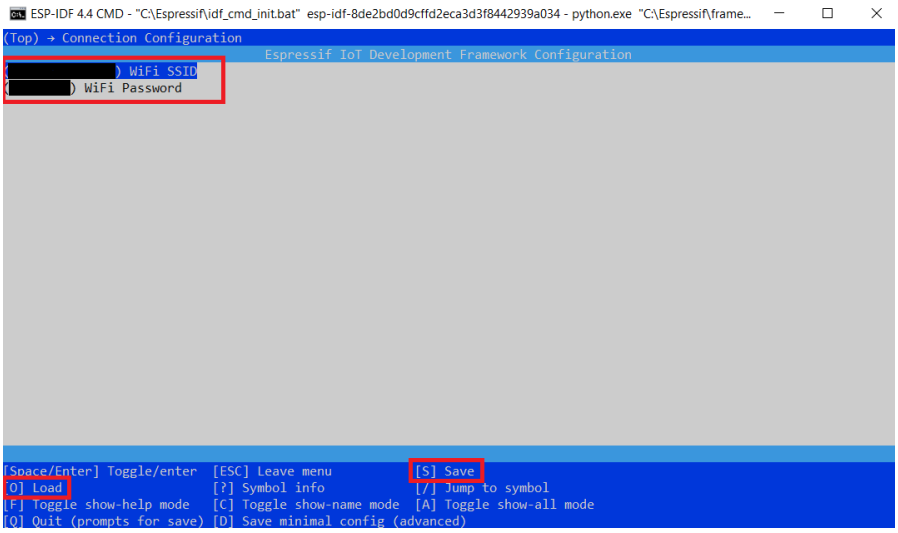
Figura 22 – *Espressif IoT Development Framework Configuration*



Fonte: Autoria Própria

Após inserir as credenciais da rede, deve-se carregar as informações no ESP32, deste modo, foi selecionado os comandos [O] Load, para carregar os dados no ESP32 e então salvar as configurações, utilizando o comando [S] Save, como mostra a
Figura 23 abaixo:

Figura 23 – Credenciais Wi-Fi, carregando e salvando informações

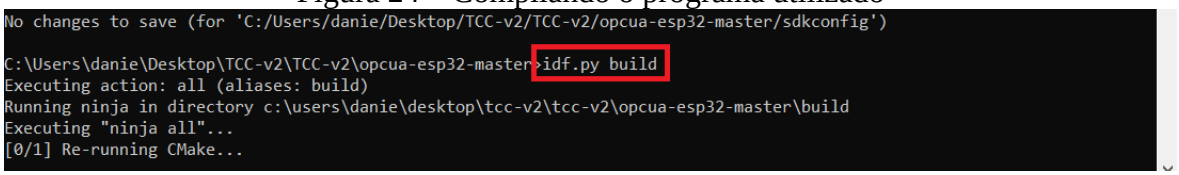


Fonte: Autoria Própria

Com as configurações gravadas no ESP32, é necessário sair do menu de configuração e voltar para a interface de comandos da ESP-IDF utilizando o comando ***[ESC] Leave menu*** duas vezes.

Então, foi compilado o programa para enfim, gravá-lo no ESP32. Para compilar, é necessário utilizar o comando: ***idf.py build***, como indicado na Figura 24 a seguir:

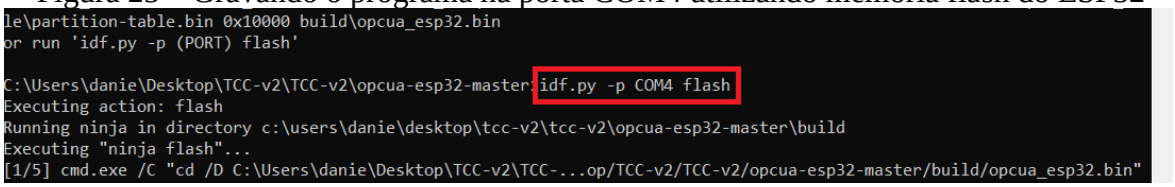
Figura 24 – Compilando o programa utilizado



Fonte: Autoria Própria

Por fim, foi utilizado a memória flash do ESP32 para gravar o programa na porta COM4. Assim, utilizou-se o comando: ***idf.py -p COM4 flash***, representado na Figura 25 abaixo. Vale ressaltar que é possível utilizar a porta COM3 do ESP32 também para gravar o programa.

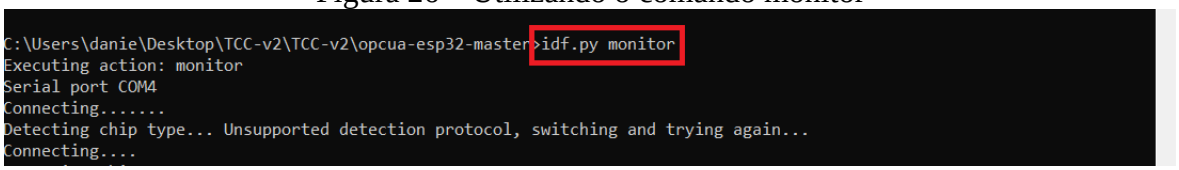
Figura 25 – Gravando o programa na porta COM4 utilizando memória flash do ESP32



Fonte: Autoria Própria

Então, para visualizar se todas as configurações deram certo, se o servidor foi criado e qual o IP do servidor, utilizou-se o comando: ***idf.py monitor***, como indicado na Figura 26 abaixo:

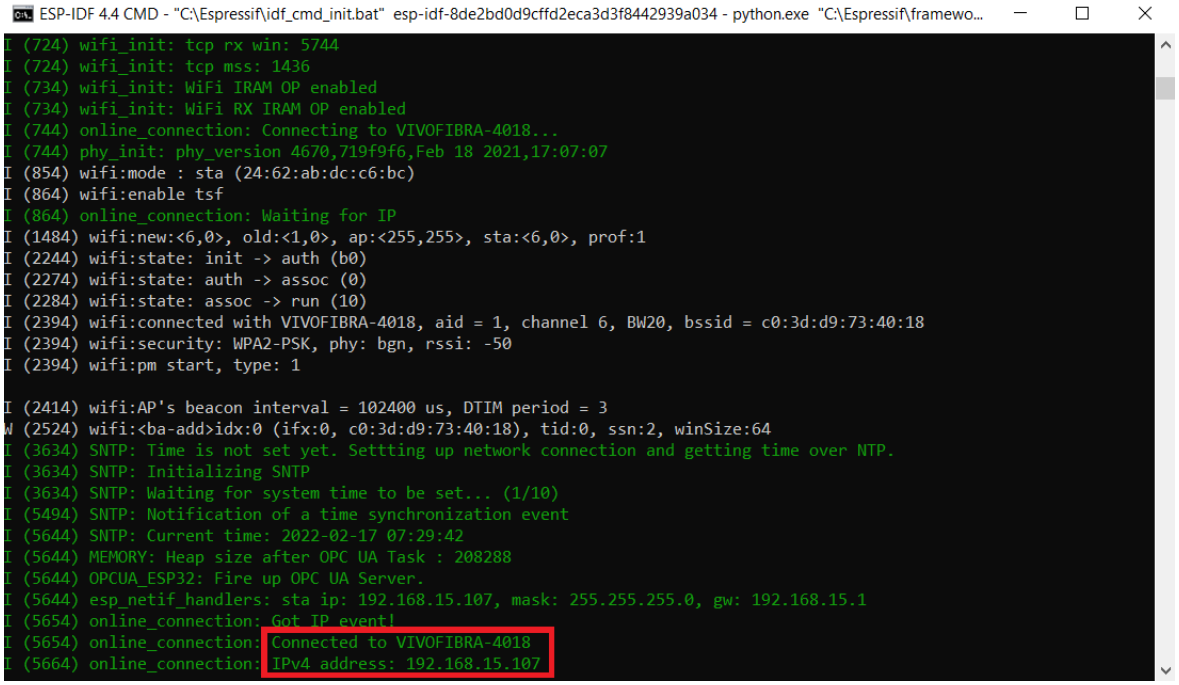
Figura 26 – Utilizando o comando monitor



Fonte: Autoria Própria

Com este comando, é possível visualizar todas as configurações do servidor, informações da aplicação, tempo de compilação, tamanho, versão, entre outras informações, como mostra a Figura 27 a seguir:

Figura 27 – Configurações do servidor, informações da aplicação, IPv4



Fonte: Autoria Própria

Assim, foi implementado o servidor OPC UA no ESP32 com sucesso, onde o endereço de IP obtido, o qual deverá ser utilizado para a conexão dos clientes, foi **192.168.15.107**, como mostra a Figura 27 acima.

4.2 Controle das entradas/saídas analógicas/digitais

Com todas as configurações acima realizadas junto a montagem do circuito proposto, é possível conectar no servidor OPC UA criado por meio das ferramentas Prosys OPC UA Client Mobile e OPC UA Expert (PC), descritas no capítulo 3. Deste modo, foi realizado as devidas conexões para enfim, controlar as portas de entrada e saída do ESP32.

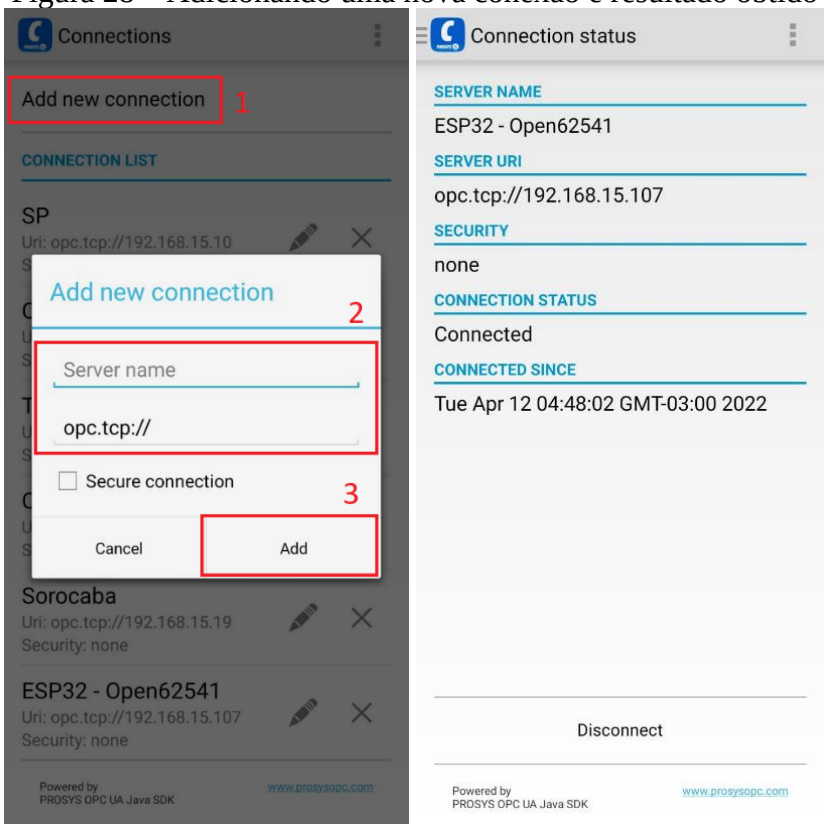
A próxima seção (4.2.1) detalhará os resultados obtidos utilizando o Prosys OPC UA Client Mobile e a seção 4.2.2 detalhará os resultados obtidos com o OPC UA Expert.

4.2.1 Prosys OPC UA Client Mobile

Inicialmente, foi necessário conectar-se ao servidor criado adicionando uma nova conexão no aplicativo, para isso, utilizou-se o endereço de IP obtido, no caso do trabalho, o endereço: opc.tcp://192.168.15.107.

Deste modo, foi possível conectar-se ao servidor criado. A Figura 28 a seguir, mostra a tela para se conectar e o status de conexão com o servidor, nela é possível ver o nome do servidor criado, bem como o endereço, informações de segurança, o status da conexão e quando foi realizada a conexão.

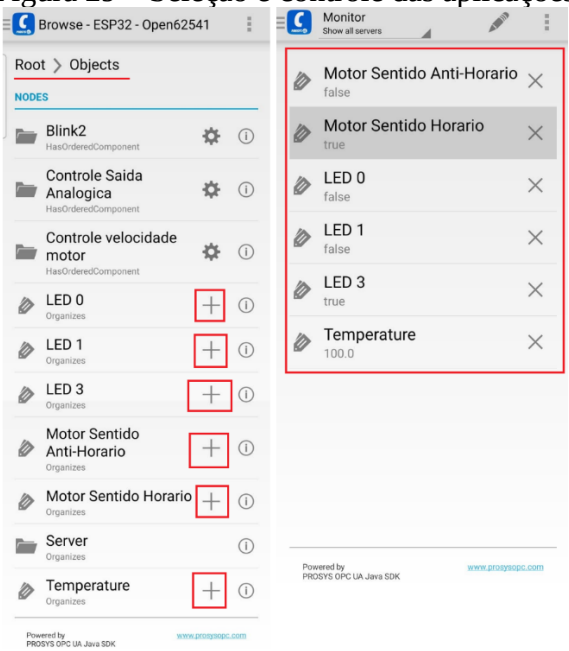
Figura 28 – Adicionando uma nova conexão e resultado obtido



Fonte: Autoria Própria

Com o dispositivo conectado ao servidor, foi possível realizar o controle das portas de entrada e saída do ESP32 por meio do aplicativo. Assim, acessou-se a tela de "*Objects*", para encontrar os nodes criados para a realização do controle das portas. Foram adicionadas algumas aplicações para controle, que podem ser observadas e controladas na tela de "*Monitor*" do app, como mostra a Figura 29 a seguir:

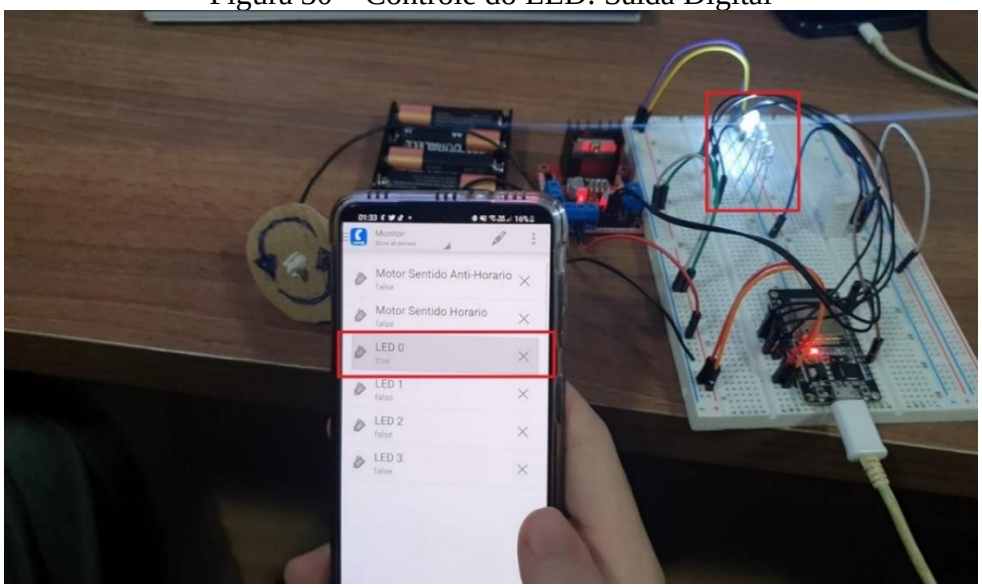
Figura 29 – Seleção e controle das aplicações



Fonte: Autoria Própria

Assim, foi realizado o primeiro controle, no qual foi possível acender o primeiro LED. Como ele foi programado para ser uma saída digital, ele recebe valores de 0 e 1 para ligar e desligar. O aplicativo também entende os valores como "*true*" = 1 e "*false*" = 0. A Figura 30 mostra o funcionamento da aplicação junto ao comando utilizado.

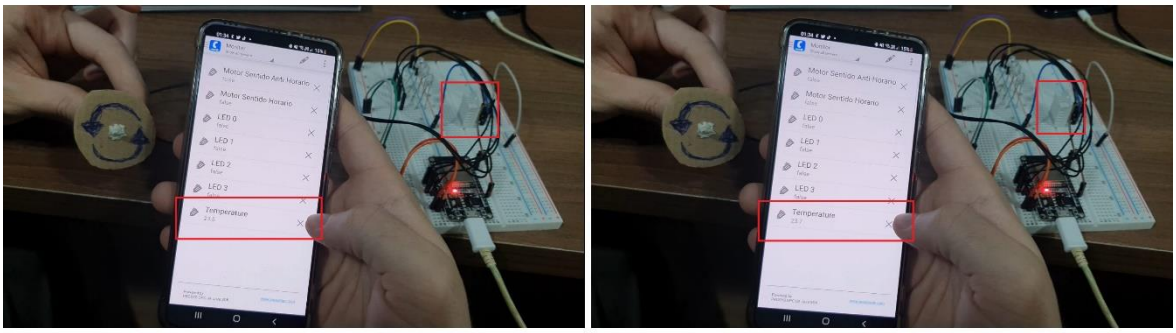
Figura 30 – Controle do LED: Saída Digital



Fonte: Autoria Própria

Então, realizou-se o teste para leitura da temperatura obtida pelo sensor DHT22, representando uma entrada digital. O valor de leitura do sensor é exibido na própria tela de monitoramento, como mostra a Figura 31, nela foi possível acompanhar a variação em tempo real da temperatura. Vale ressaltar que a leitura do sensor é realizada em alta frequência, o que fez com que alguns valores não fossem medidos/exibidos com precisão absoluta.

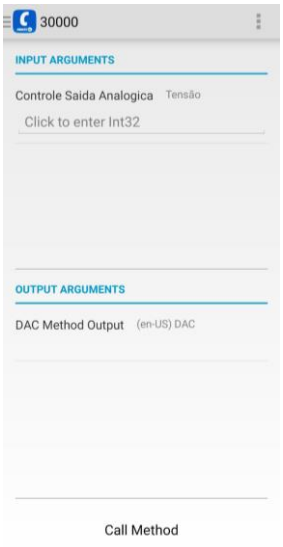
Figura 31 – Leitor de Temperatura DHT22: Entrada Digital



Fonte: Autoria Própria

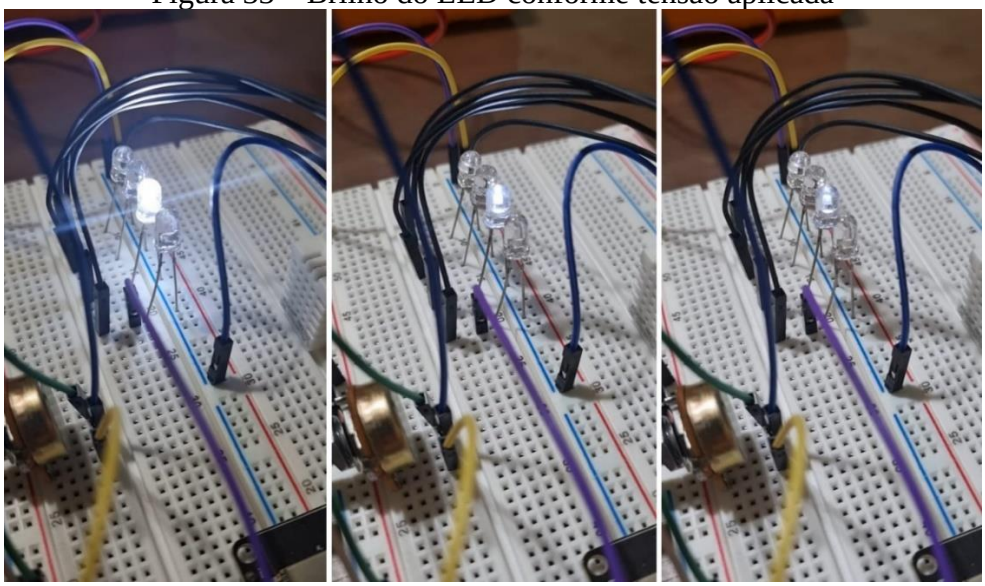
Foi realizado também o controle da saída analógica, variando o valor da tensão que era colocado na porta, fazendo com que variasse o brilho de um dos LEDs. Deste modo, foi criado um método no qual era possível digitar o valor de tensão, sendo 0V o valor mínimo, no qual o LED permanecia apagado e 3V o valor máximo, no qual o LED ficava com o brilho máximo. Na Figura 32 é possível observar a tela na qual se inseriu o valor da tensão, e na Figura 33 os brilhos do LED conforme os valores de tensões aplicados na porta.

Figura 32 – Controle de Tensão: Saída Analógica



Fonte: Autoria Própria

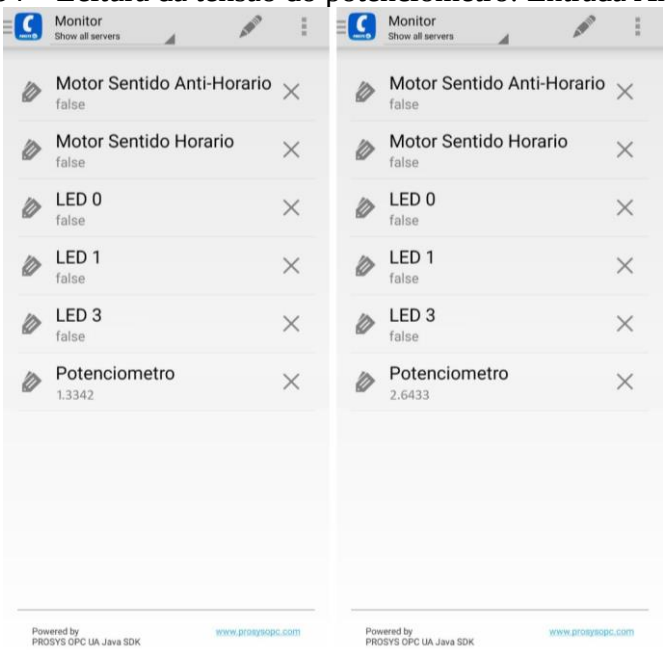
Figura 33 – Brilho do LED conforme tensão aplicada



Fonte: Autoria Própria

Então, foi realizado um teste de leitura de um valor de tensão dado por meio de um potenciômetro, de modo a testar uma entrada de valor analógico. Foi aplicado valores de 0V e 3,3V nos terminais das extremidades do potenciômetro, sendo assim, era possível obter valores dentro desse intervalo conforme fosse variando o cursor do potenciômetro, bem como a sua resistência. A Figura 34 mostra alguns dos resultados obtidos:

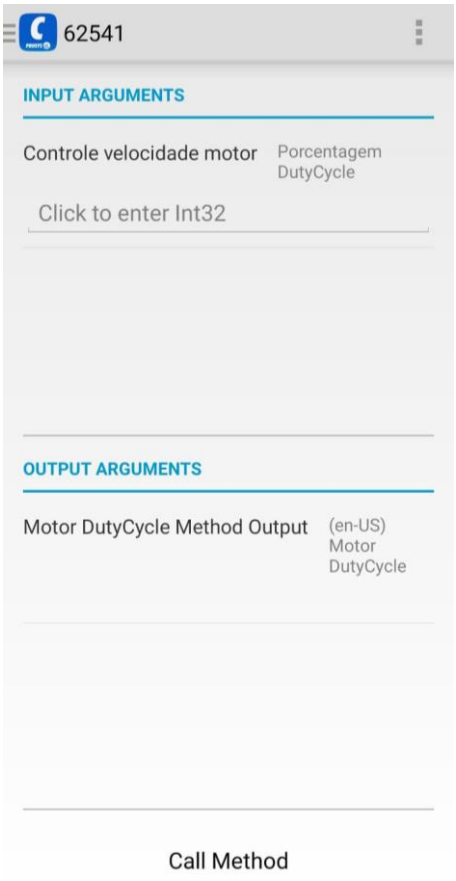
Figura 34 – Leitura da tensão do potenciômetro: Entrada Analógica



Fonte: Autoria Própria

Testou-se também o controle da velocidade de um motor DC por meio do controle PWM, bem como o sentido de rotação desse motor, conforme explicado na seção 3.6 do capítulo de Materiais e Métodos. O valor da velocidade variava conforme o Duty Cycle inserido no método de controle de velocidade, podendo assumir valor de 0 a 100%, sendo 0% o motor parado e 100% o motor com velocidade máxima. A Figura 35 mostra a tela do método de controle do Duty Cycle inserido.

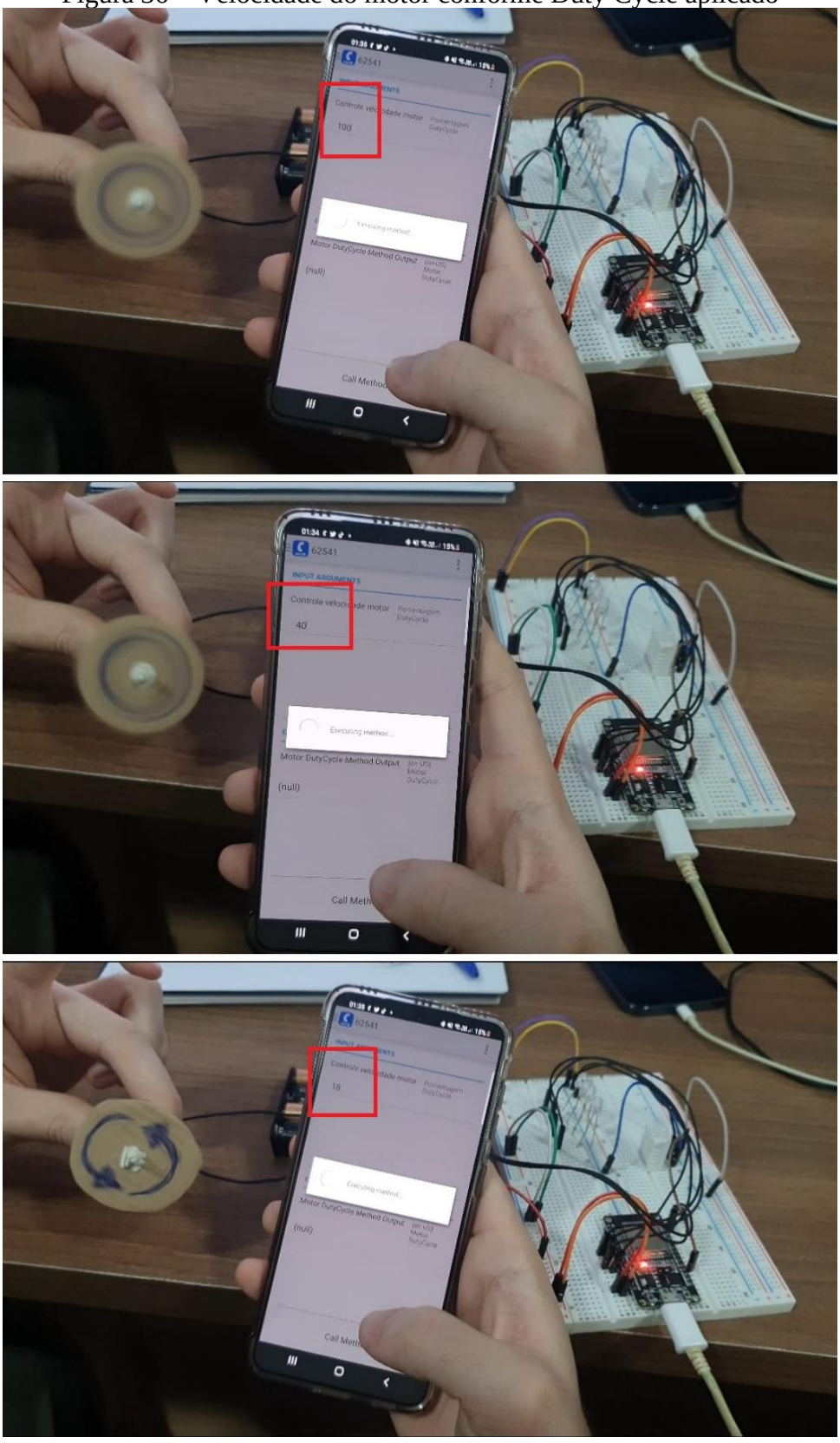
Figura 35 – Controle da Velocidade do Motor DC: Controle PWM



Fonte: Autoria Própria

Assim, conforme era inserido o valor do Duty Cycle, a velocidade do motor DC variava. A Figura 36 mostra o comportamento do motor, sendo os valores inseridos de 100%, 40% e 18% respectivamente.

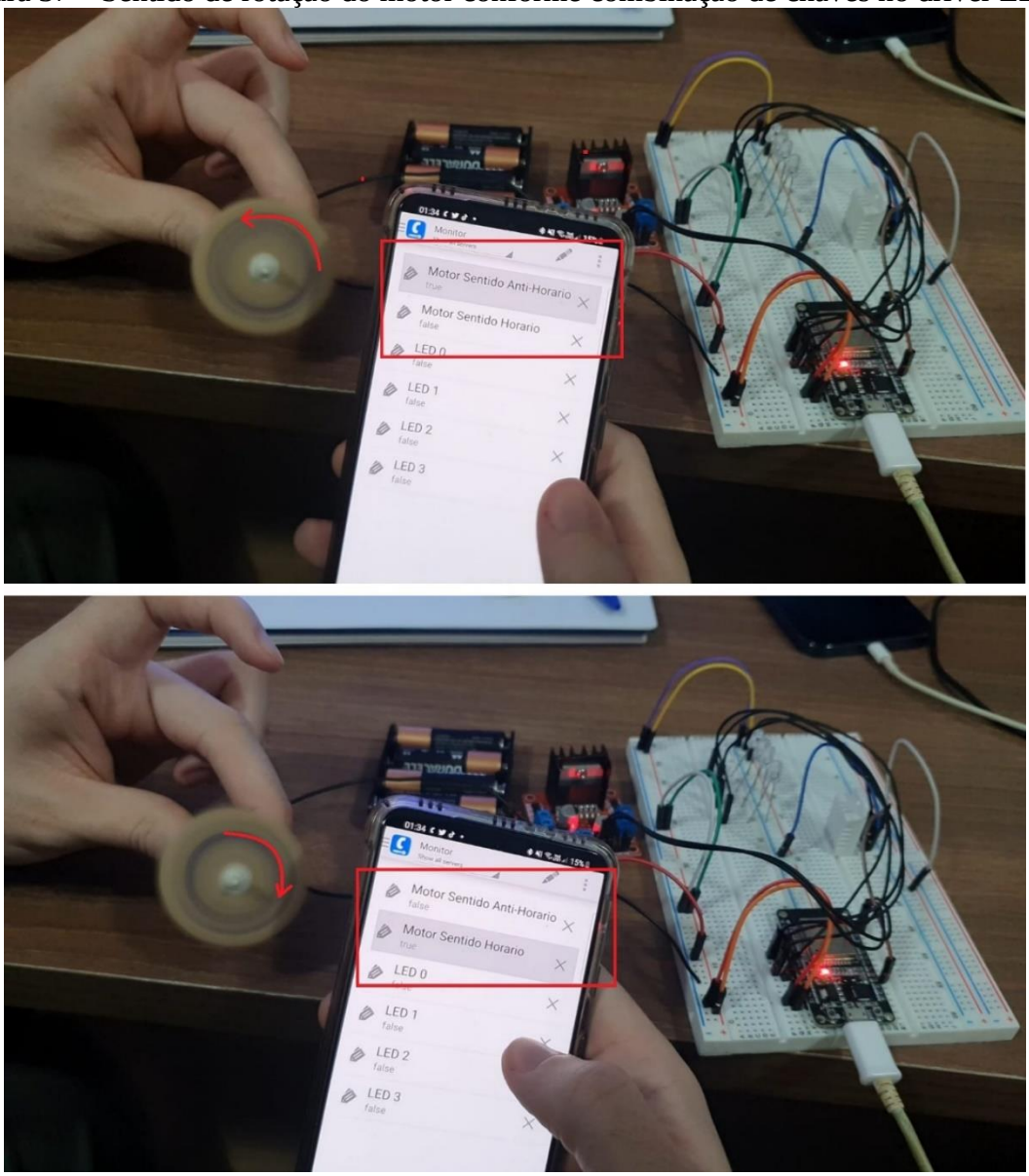
Figura 36 – Velocidade do motor conforme Duty Cycle aplicado



Fonte: Autoria Própria

Também foi possível controlar o sentido de rotação do motor DC, conforme mostra a Figura 37 abaixo. O controle era feito por meio das entradas INPUT1 e INPUT2 do driver Ponte H L298n, conectadas aos terminais do motor. Assim, foram configuradas duas portas de saídas digital, de modo a realizar esse controle, sendo assim, o aplicativo recebia valores de 0 e 1 para poder setar os inputs, sendo possível também inserir os valores "*true*" e "*false*", representando os valores de 1 e 0 respectivamente.

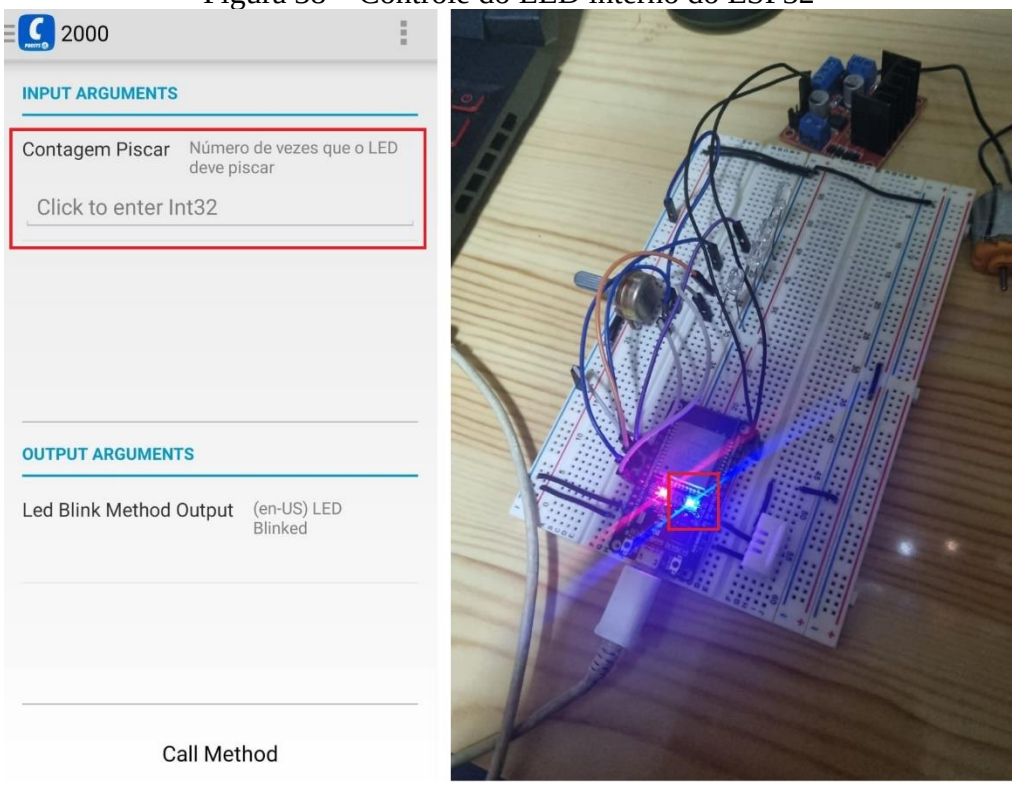
Figura 37 – Sentido de rotação do motor conforme combinação de chaves no driver L298n



Fonte: Autoria Própria

Por fim, foi realizado o controle da rotina que fazia com que o LED interno da placa piscasse conforme o número de vezes inserido no método, a Figura 38 mostra a funcionamento da aplicação, deste modo, fez-se o teste inserindo-se alguns valores e foi possível observar que o LED interno do ESP32 piscava conforme quantidade inserida.

Figura 38 – Controle do LED interno do ESP32



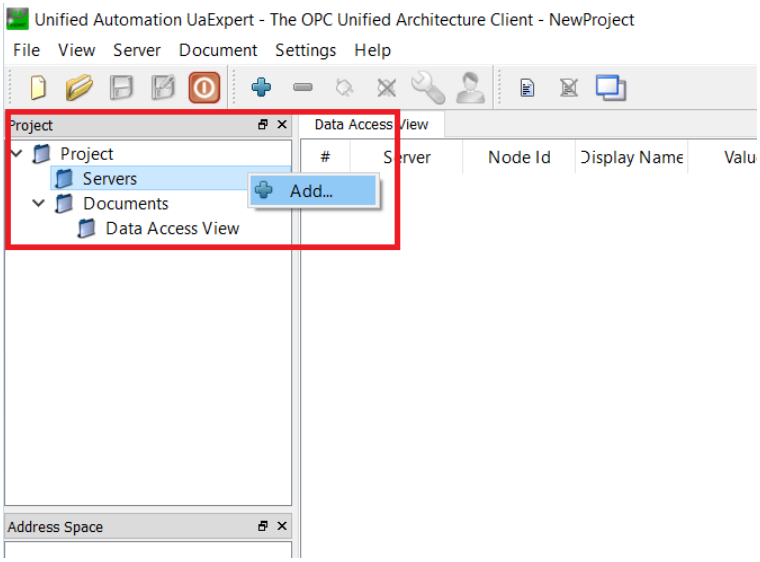
Fonte: Autoria Própria

4.2.2 Conexão com o UA Expert (PC)

Para conectar ao servidor OPC UA no ESP32 com o cliente OPC UA via PC foi utilizado o software UA Expert, o qual foi descrito com maiores detalhes no capítulo 3, na seção 3.3. O software é capaz de controlar as portas de entrada e saída do ESP32 e, para isso, é necessário criar a conexão com o servidor OPC UA criado bem como adicionar as variáveis de controle.

Deste modo, para adicionar uma conexão acessou-se a guia "*Project*" na tela inicial do UA Expert e em "*Servers*" foi possível adicionar a nova conexão como mostra a Figura 39 a seguir:

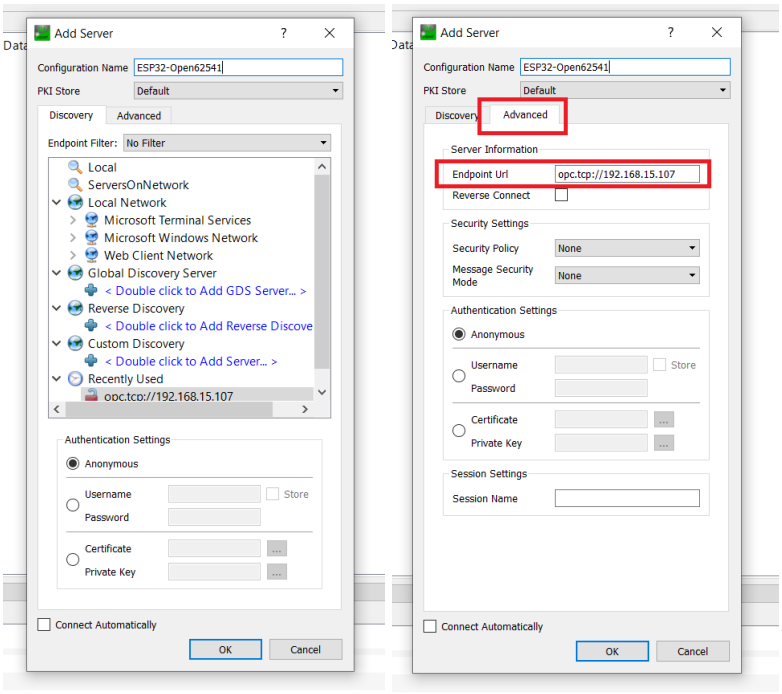
Figura 39 – Tela inicial do UA Expert



Fonte: Autoria Própria

Assim, para criar a conexão foi necessário nomear o servidor em "*Configuration Name*", e na aba "*Advanced*" inserir o *endpoint* do servidor, como mostra a Figura 40, no caso, foi utilizado o IP gerado inicialmente ao criar o servidor OPC UA no ESP32, tendo o seguinte endereço de conexão: opc.tcp://192.168.15.107

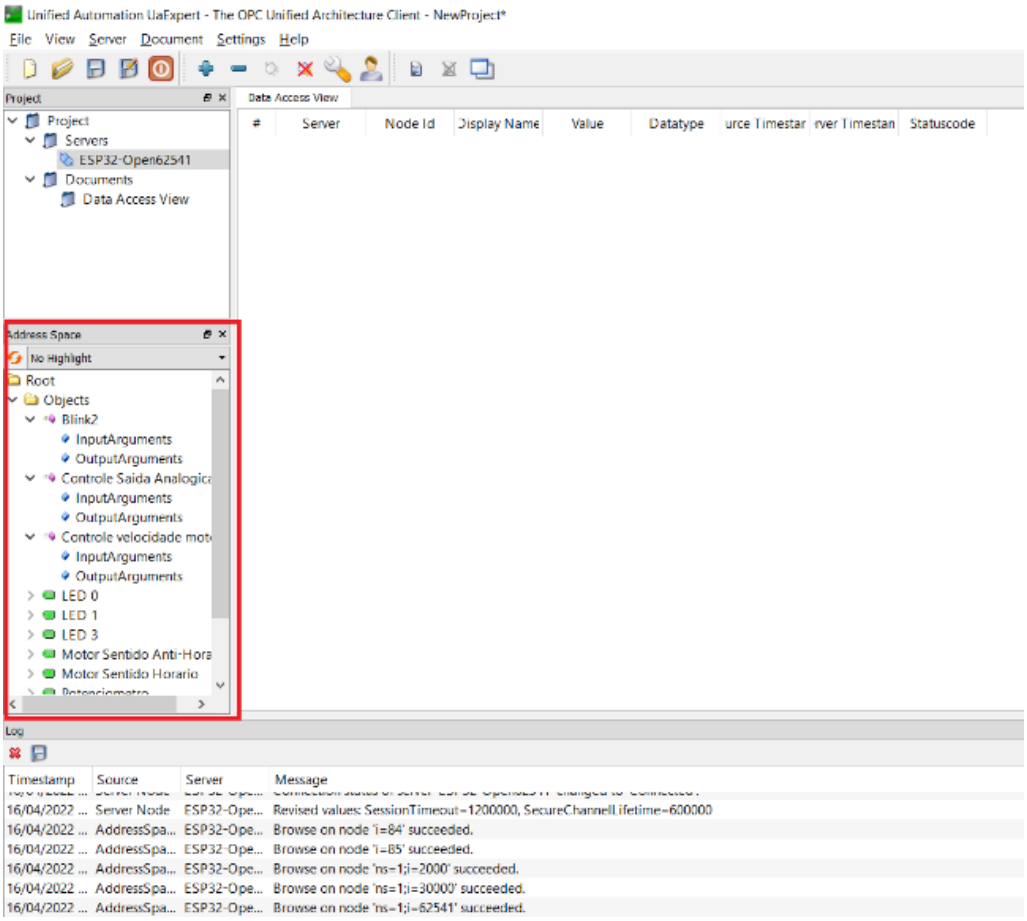
Figura 40 – Controle do LED interno do ESP32



Fonte: Autoria Própria

Com o cliente conectado ao servidor OPC UA, foi possível observar os nodes para controle das funcionalidades do circuito na aba de "*Address Space*", como mostra a Figura 41. Assim, para que fosse de fato realizado o controle, foi necessário adicionar o node desejado na aba de "*Data Access View*".

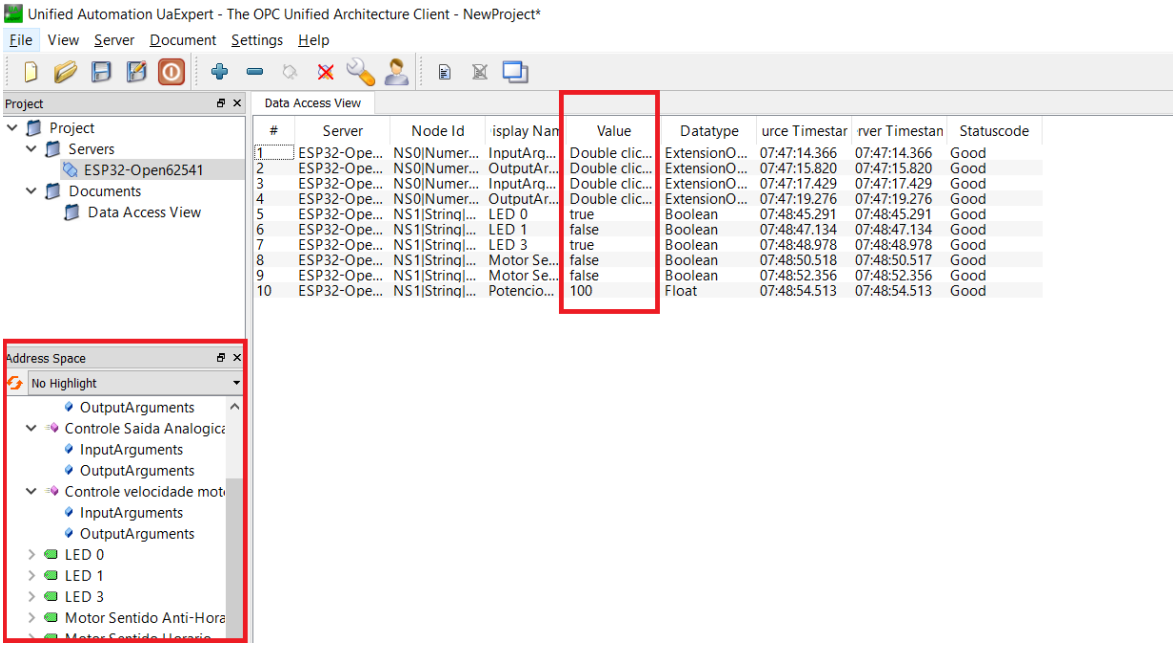
Figura 41 – Address Spaces UA Expert



Fonte: Autoria Própria

Com os nodes adicionados, foi possível realizar o controle das aplicações implementadas no servidor OPC UA no ESP32 e, para isso, alterou-se o valor das variáveis, como mostra a Figura 42, da mesma forma que foi realizado o controle utilizando o Prosys OPC UA Mobile apresentado na seção 4.2.1 acima.

Figura 42 – Controle dos nodes do servidor OPC UA



Fonte: Autoria Própria

Assim, foi possível realizar o controle das portas de entrada e saída digital e analógica, bem como o controle PWM da velocidade do motor DC no ESP32 utilizando o software UA Expert. Os resultados obtidos foram os mesmos apresentados na seção 4.2.1, sendo possível controlar os LEDs, acender/apagar e o brilho, fazer a leitura da temperatura utilizando o DHT22, fazer a leitura da tensão do potenciômetro, controlar o sentido de rotação do motor DC e controlar sua velocidade também pelo cliente no computador.

4.3 Considerações Finais e Trabalhos Futuros

Para o desenvolvimento do projeto, incialmente foi necessário a realização de um estudo sobre o a tecnologia OPC UA, bem como o conceito de servidores e clientes. Então foi realizado um estudo sobre a ESP-IDF, exemplos de programas e formas de comunicação. Assim, foi realizado testes de desenvolvimentos separados para cada funcionalidade proposta, sendo um exemplo de cada tipo de controle, entrada/saída e digital/analógica.

Com uma base conceitual consolidada, foi feita uma análise do programa *open source* (OPCUA-ESP32, GitHub) bem com um estudo da biblioteca *open62541* de modo entender como era possível realizar a implementação do servidor OPC UA no ESP32. A partir dessa análise, foi possível implementar novas funcionalidades no servidor, visando trazer

exemplos de desenvolvimentos que abrangessem a maior parte das funcionalidades do ESP32.

Por fim, foi realizado um estudo sobre as ferramentas de comunicação com o servidor OPC UA criado, como o Prosys OPC UA Client Mobile, para controle via celular e o UA Expert, para controle via PC.

Assim, como trabalhos futuros tem-se como primeira sugestão a melhoria nas implementações, montando circuitos mais robustos, levando em consideração interferências, ruídos, para obter resultados mais limpos e precisos, como no caso da leitura do DHT22 e do potenciômetro. Também se sugere que possam explorados outros tipos de softwares de comunicação com o servidor, como foram utilizados softwares gratuitos, estes possuem algumas limitações tornando seu uso um tanto quanto específico para fins acadêmicos. Por fim, estudar uma maneira mais pratica de se fazer o controle PWM do motor DC, de modo que ele possa ser implementado sem a necessidade de se chamar uma rotina de controle.

5. CONCLUSÃO

O padrão de comunicação utilizado, possibilitou o servidor OPC UA a compartilhar as informações com cliente OPC UA, de modo que fosse possível gerenciar e supervisionar os processos, acompanhando e monitorando o envio e recebimento de dados.

O ESP32 provou-se ser um microcontrolador de baixo custo com alto potencial de desenvolvimento e uma gama de funcionalidades disponíveis, tendo como uma das principais características, a versatilidade e a robustez, podendo tomar cada vez mais espaço nos desenvolvimentos de projetos envolvendo IoTs.

A biblioteca open source utilizada, a open62541 escrita em linguagem C/C++, forneceu as ferramentas necessárias para criação de aplicações de clientes e servidores OPC UA, com uma excelente documentação disponível para auxílio nos desenvolvimentos.

Em suma, com a implementação do servidor OPC UA no ESP32, foi possível observar os conceitos teóricos aplicados na prática, bem como fornecer exemplos de comunicação e controle das portas de entrada e saída digital e analógica do ESP32, de modo que possa ser possível adaptar os desenvolvimentos para outras necessidades de diversos projetos.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] - ANDRADE, F. **Falando sobre a comunicação OPC e OPC UA**. Disponível em: <https://automacaoecartoons.com/2018/02/12/comunicacao-opc-e-opc-ua/>. Acesso em 09 de janeiro de 2022.

[2] - BURKE, T.J. **OPC Unified Architecture: Interoperability for Industrie 4.0 and the Internet of Things.** OPC Foundation, 2017.

[3] - ECLIPSE. **Eclipse Milo 0.1.0 Release Review**. Disponível em: <https://projects.eclipse.org/projects/iot.milo/reviews/0.1.0-release-review> Acesso em: 03 de abril de 2022.

[4] - ESPRESSIF, ESP-IDF - **Windows Installer Download**. Disponível em: <https://dl.espressif.com/dl/esp-idf/> Acesso em: 20 de setembro de 2020

[5] - ESPRESSIF SYSTEMS. **Official IoT Development Framework**. Disponível em: < https://www.espressif.com/en/products/sdks/esp-idf>. Acesso em 09 de janeiro de 2022.

[6] - FONSECA, M. O. **Comunicação OPC – Uma abordagem prática**. VI Seminário de Automação de Processos, Associação Brasileira de Metalurgia e materiais, 2002.

[7] - HUEBL, K. **Willkommen beim ASNeG Git Repository**. Disponível em: <https://81.169.197.52:8443/repositories/>. Acesso em: 03 de abril de 2022.

[8] - KOZAR, S. **Integration of IEC 61499 with OPC UA**. In 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, pp. 1-7, 2016.

[9] - LONGEN, A.S. **Protocolos de Rede: O Que São, Como Funcionam e Tipos de Protocolos de Internet**. Disponível em: <https://www.weblink.com.br/ blog/tecnologia/conheca-os-principais-protocolos-de-internet/>. Acesso em 09 de janeiro de 2022.

[10] - MAHNKE, W.; LEITNER,S,H.; DAMM, M. **OPC Unified Architecture. In Springer**, 2009.

[11] - MELO, P. F. S. de. **Dispositivo de Controle para a Indústria 4.0 baseado no RAMI 4.0 e OPC UA**. Dissertação (Mestrado em Engenharia Elétrica) – Campus Experimental de Sorocaba, UNESP - Universidade Estadual Paulista, Sorocaba, 2019

[12] - MICROSOFT. **Visual Studio Code**. Disponível em: < https://code.visualstudio.com/?wt.mc_id=vscom_downloads> Acesso em: 20 de setembro de 2020

[13] - MOLDOVEN, A; BARNSTED, E; OURSEL, E. **OPC UA.Net Standard Stack and Samples**. Disponível em: https://github.com/OPCFoundation/UA-. NETStandard. Acesso em: 03 de abril de 2022.

[14] - OPCUA-ESP32. **OPC UA Server on ESP32**. Disponível em: <https://github.com/cmbahadir/opcua-esp32> Acesso em 20 de setembro de 2020

[15] - OPC FOUNDATION. **Unified Architecture, 2017**. Disponível em: <https://opcfoundation.org/developer-tools/specifications-unified-architecture/>. Acesso em: 03 de abril de 2022.

[16] - PROFANTER, S.; DOROFEEV, K.L; ZOITL, A.; KNOLL, A. **OPC UA for Plug & Produce: Automatic Device Discovery using LDS-ME**. 2017 22 nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 2017, pp. 1-8.

[17] - PROFANTER, S. **List of Open Source OPC UA Implementations**. Disponivel em: <https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC UA-Implementations&gt> Acesso em: 03 de abril de 2022.

[18] - PROSYS OPC. **OPC UA - Client application running on Android**. Disponível em: < https://www.prosysopc.com/products/opc-ua-client-for-android/>. Acesso em 09 de janeiro de 2022.

[19] - PROSYS OPC. **OPC UA & Classic Products**. Disponível em: < https://www.prosysopc.com/products/>. Acesso em 09 de janeiro de 2022.

[20] - REIMANN, J. **OPC UA solutions with Eclipse Milo**, 2017. Disponível em: <https://dentrassi.de/2017/09/14/creating-opc-ua-solutions-eclipse-milo/> Acesso em: 03 de abril de 2022.

[21] - RYKONANOV, A. **FreeOPCUA Library**. Disponivel em:<https://github.com/FreeOPCUA>. Acesso em: 03 de abril de 2022.

[22] - SANTOS, R. **ESP32 with DC Motor and L298N Motor Driver – Control Speed and Direction**, 2018. Disponível em: <https://randomnerdtutorials.com/esp32-dc-motor-l298n-motor-driver-control-speed-direction/> Acesso em: 20 de setembro de 2020

[23] - UA EXPERT. UaExpert - **A Full-Featured OPC UA Client**. Disponível em: <https://www.unified-automation.com/products/development-tools/uaexpert.html>. Acesso em 09 de janeiro de 2022.

[24] - VENURELLI, M. **OPC UA na Automação Industrial**. Disponível em: <https://www.automacaoindustrial.info/opc-ua-na-automacao-industrial/>. Acesso em 09 de janeiro de 2022.

7. APÊNDICE A - Tutorial de Comunicação criado para a disciplina Redes Industriais de Comunicação (RIC)