

PROJETO E DESENVOLVIMENTO DE INTEGRAÇÃO DE IOT PARA CONTROLE DE MÃO ROBÓTICA IMPRESSA EM 3D

**PROJECT AND DEVELOPMENT OF IOT INTEGRATION FOR CONTROL OF A
3D PRINTED ROBOTIC HAND**

Ciências Exatas e da Terra, Engenharias • 08/10/2026

REGISTRO DOI: [10.70773/revistatopicos/791258867](https://doi.org/10.70773/revistatopicos/791258867)

Roberto Consentins Torma¹

André Weizmann²

RESUMO

O uso de controle via internet para dispositivos eletrônicos é razoavelmente recente, conhecido como a Internet das Coisas (IoT), permite que equipamentos conectados à internet sejam monitorados e controlados remotamente. Isso abre portas para novas aplicações, como automatizações baseadas em dados em tempo real e a indústria 4.0, por exemplo. Com isso, este trabalho tem como objetivo principal a construção de um protótipo impresso em 3D para que o mesmo possa ser controlado remotamente, utilizando protocolos de IoT, secundariamente, objetivou-se o desenvolvimento de uma pesquisa sobre IoT, bem como visar o conhecimento a respeito do ESP32 e finalmente foi realizado um estudo a respeito das técnicas necessárias para integração destes recursos com componentes eletrônicos para seu pleno funcionamento. Para isso foi utilizado o ESP32 como componente central tanto no emissor quanto no receptor de informações, motores tipo servo para realização de movimentos e sensores flexíveis para leitura do posicionamento da mão. Um ponto crítico no desenvolvimento do trabalho foi a escolha do protocolo responsável pelo tráfego de informações, principalmente pela sua documentação escassa.

Palavras-chave: Internet das coisas; ESP32; robótica.

ABSTRACT

The use of internet control for electronic devices is reasonably recent, known as the Internet of Things (IoT), it allows equipment connected to the internet to be monitored and controlled remotely. This opens doors to new applications, such as automation based on real-time data and industry 4.0, for example. Therefore, this work's main objective is to build a 3D printed prototype so that it can be controlled remotely, using IoT protocols. regarding the ESP32 and

finally a study was carried out regarding the techniques necessary to integrate these resources with electronic components for their full functioning. For this purpose, the ESP32 was used as a central component in both the transmitter and receiver of information, servo-type motors to carry out movements and flexible sensors to read the positioning of the hand. A critical point in the development of the work was the choice of the protocol responsible for information traffic, mainly due to its scarce documentation.

Keywords: Internet of Things; ESP32; robotics.

1. INTRODUÇÃO

O uso de IoT permite que praticamente quaisquer componentes sejam conectados à internet, facilitando o monitoramento remoto em tempo real de seu desempenho e condições, podendo enviar dados sobre o movimento, desgaste e condições de operação, permitindo seu controle à distância.

Conforme Alves e Rosa (2021), dispositivos IoT podem ser integrados a um ecossistema maior de dispositivos conectados, isso permitiria ao usuário, por exemplo, controlar dispositivos para interagir com outros sistemas em seu ambiente, efetuar leituras das condições de sensores e afins, possibilitando conceitos da indústria 4.0, por exemplo.

Graças ao IoT, neste caso a prótese poderá ser controlada de qualquer lugar do mundo, sem limitações geográficas, isso permitiria, por exemplo, que profissionais de saúde, enviem ajustes ou orientações à distância, ou até mesmo assumam o controle temporariamente em situações críticas, podendo inclusive auxiliar na adaptação ao uso de próteses, bem como auxiliar no processo de

fisioterapia, com pequenas alterações na modelagem da prótese, ensinando como realizar as atividades de recuperação de movimentos, por exemplo, auxiliando assim a qualidade de vida dos usuários, ao integrar essas tecnologias de maneira segura, eficiente e acessível.

Sendo assim o artigo se permeia no desenvolvimento de um protótipo de prótese de mão, que é controlado via internet através de IoT, cujo processo de realização pode servir para difusão do conhecimento a respeito da Internet das coisas e o seu resultado final pode servir como ponto inicial para uso destas tecnologias aplicadas à saúde.

Para que possa ser possível obter os resultados esperados foram utilizados o ESP32 como componente de processamento central, o protocolo MQTT, que realiza o gerenciamento de mensagens baseado em padrões para comunicação entre dispositivos IoT e sensores e atuadores para captura de informações e realização de movimentos físicos, respectivamente.

No artigo a seguir será abordado os conceitos básicos sobre a Internet das coisas, sobre o ESP32, o protocolo MQTT bem como componentes e interligação dos mesmos perante o projeto.

Sendo assim o presente artigo, basicamente, se divide em 5 seções, sendo a segunda seção a apresentar a base teórica, que trata de toda a fundamentação dos conceitos no qual o trabalho está baseado. Na terceira seção aborda-se a metodologia utilizada. Já a quarta seção é composta pela discussão dos resultados obtidos, além da exibição deles. E por fim, mas não menos importante, a

quinta seção que apresenta as considerações finais do artigo, além das possíveis abordagens em eventuais trabalhos futuros.

2. TÓPICOS FUNDAMENTAIS

2.1. Internet das Coisas

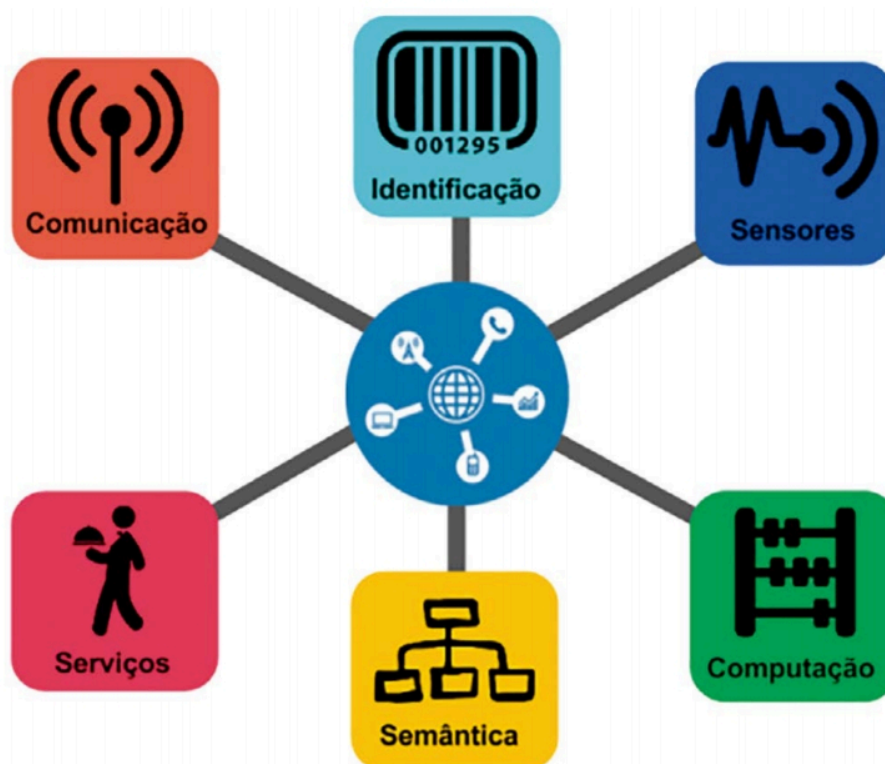
A IoT surgiu graças aos avanços das tecnologias de miniaturização de componentes eletrônicos e das tecnologias de comunicação sem fio. O pa-radigma IoT estabelece que diversos itens do cotidiano podem ter acesso à internet, o que contribui para uma diversidade de benefícios à população. A previsão é que bilhões de “coisas” sejam capazes de conectar-se à internet para prover os mais diversos tipos de informações aos usuários (Pacheco, 2018).

Segundo Maschietto (2021), a Internet das Coisas (IoT ou Internet of Things), refere-se à conexão de objetos físicos à internet, permitindo que eles colem, troquem e processem dados entre si ou com sistemas centralizados, sem a necessidade de intervenção humana direta. Esses objetos podem ser uma vasta gama de dispositivos, desde eletrodomésticos, sensores industriais, sistemas de segurança, veículos, até dispositivos médicos, como próteses e equipamentos hospitalares.

Conceitualmente, podemos definir a internet das coisas como a associação de diversos componente, como pode ser visto na figura 1, abaixo, onde tem-se a representação gráfica deste conjunto de operações, No bloco da Identificação estão as tecnologias relacionadas às tecnologias empregadas para identificar os objetos (RFID, NFC, entre outros). Já os sensores e atuadores retratam os componentes capazes de coletar ou agir sobre o contexto onde se encontram. A comunicação abrange diversas técnicas usadas para

conectar objetos inteligentes como conexões de rede sem fio e *Bluetooth*. Na sequência temos a computação que apresentam as unidade de processamento como microcontroladores e processadores, executando algoritmos locais nos objetos inteligentes. Os serviços englobam serviços de identificação, de agregação de dados, de colaboração e inteligência e de ubiquidade, possibilitando a Semântica que é a habilidade de extração de conhecimento dos objetos na IoT (Santos et al., 2016).

Figura 1. Blocos básicos da IoT.



Fonte: Santos et al. (2016, p. 6).

A internet das coisas funciona conectando dispositivos equipados com sensores, atuadores, e conectividade à internet. Esses dispositivos capturam informações sobre o ambiente ou sobre o próprio dispositivo (como temperatura, movimento, localização, desempenho), e essas informações são transmitidas a um servidor central, a uma nuvem, ou diretamente para outros dispositivos. Em seguida, as informações são processadas e podem desencadear

ações automatizadas ou fornecer informações para tomada de decisão em tempo real (Maschietto, 2021).

A IoT constitui um novo paradigma que combina aspectos de tecnologias de computação, redes de sensores sem fio, protocolos de comunicação da internet, tecnologias de sensoriamento, comunicação e dispositivos com tecnologias incorporadas. As cidade inteligentes avançam para um ambiente integrado e inteligente, onde a IoT é usada para interconectar, interagir, controlar e fornecer insights sobre os vários sistemas fragmentados dentro das cidades. Um grande número de dispositivos interconectados, bem como uma quantidade significativa de dados gerados por eles, oferece oportunidades sem precedentes para enfrentar os desafios urbanos. Tais tecnologias são mescladas a sistemas urbanos para formar um ambiente onde os mundos real e digital se encontram e estão continuamente em interação sinérgica (JOÃO, SOUZA e SERRALVO, 2019).

2.1.1. Protocolos de IoT

Segundo Sinclair (2018), protocolos são conjuntos de regras e padrões que governam como dispositivos se comunicam entre si em uma rede. Eles especificam a forma como os dados são transmitidos, recebidos e interpretados, garantindo que diferentes dispositivos, sistemas e plataformas possam interagir de maneira eficiente e segura. Na Internet das Coisas, a comunicação eficiente entre dispositivos é fundamental, já que a IoT envolve um grande número de dispositivos conectados, desde sensores até sistemas

complexos. Para garantir que esses dispositivos possam interagir uns com os outros e com servidores ou serviços na nuvem, são necessários protocolos adequados para diferentes aspectos da comunicação, como conectividade, segurança, gerenciamento de dados e interoperabilidade.

2.1.1.1. Protocolo MQTT

Segundo Barros (2020), o protocolo MQTT (*Message Queuing Telemetry Transport*) é um protocolo projetado para dispositivos com largura de banda e energia limitadas. Muito usado em dispositivos de baixa potência, como sensores, para transmitir dados para a nuvem.

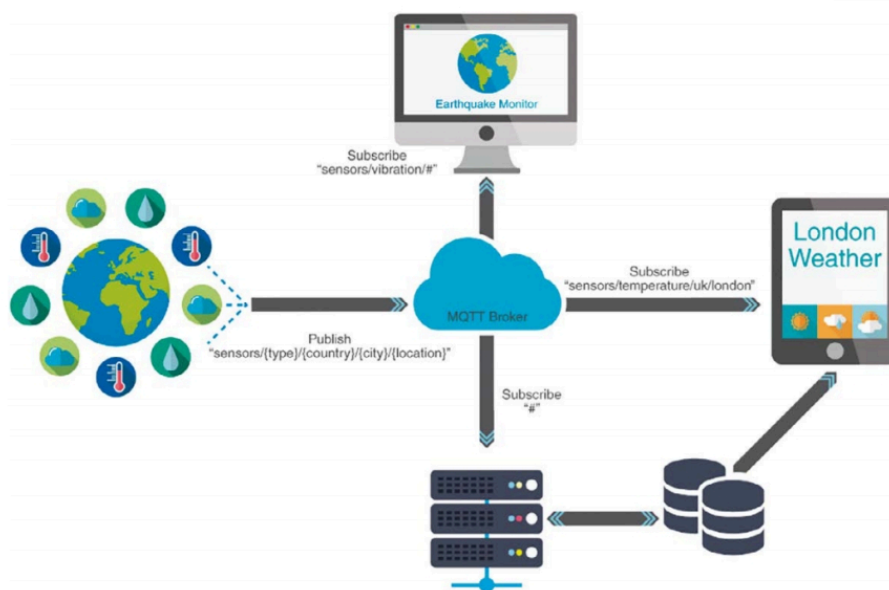
O objetivo central do protocolo MQTT é permitir a comunicação entre objetos da IoT e definir o papel de cada objeto a partir de um modelo de operação. Entre as operações fundamentais está a definição do papel de cada dispositivo, a ordem de envio de mensagens e o formato das mensagens. O MQTT foi criado para ser empregado em dispositivos com baixa largura de banda, capacidade computacional reduzida e sem garantia de conectividade (Maschietto, 2021, p. 89).

Em 2014, com a publicação do RFC 72523, o CoAP se tornou o protocolo-padrão para a troca de informações entre dispositivos com restrições computacionais, restrições de energia e em redes com baixa largura de banda (Martins; Zem, 2015). Mais recentemente, porém, a tecnologia que apresenta maior adesão pelas empresas do ramo de IoT é o MQTT. A Amazon, por exemplo,

adotou MQTT, WebSocket e Hypertext Transfer Protocol (HTTP) como protocolos-padrão em sua plataforma AWS IoT. Já o Facebook, mesmo fora do ramo de IoT, adotou em suas comunicações um sistema de instant messaging que utiliza o protocolo MQTT. Dessa forma, fica claro que o protocolo MQTT é bastante relevante no cenário atual (Torres; Rocha; Souza, 2016).

Segundo Maschietto (2021), como recurso para troca de mensagens, o protocolo MQTT apresenta um servidor de aplicação chamado Broker. A função do Broker é servir como um sistema intermediário nas trocas de mensagens. A Figura 2 ilustra mensagens sendo publicadas em um servidor MQTT Broker por diversos dispositivos de diferentes finalidades ao redor do mundo.

Figura 2. Servidor MQTT Broker.



Fonte: Maschietto (2021, p 90).

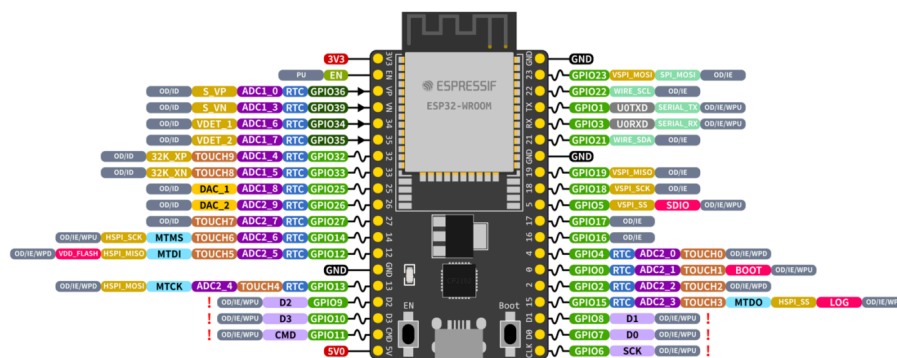
2.2. ESP32

Segundo o fabricante (ESPRESSIF, 2024), o ESP32 é um microcontrolador de baixo custo e que tem integrado interfaces Wi-Fi e Bluetooth integradas. É da mesma família do ESP8266, ambos

desenvolvidos pela empresa chinesa Espressif Systems. O ESP32 também conta com um processador dual-core, suporte a comunicação, baixo consumo de energia e recursos como criptografia, o que o torna ideal para aplicações de automação residencial, dispositivos vestíveis, redes de sensores e controle industrial.

O módulo possui a saída de tensão em 3,3 volts, possui também 30 ou 38 pinos dependendo de sua configuração. Possui também interface USB para alimentação e comunicação, além de dois botões; *Enable (En)* e *Boot*. Os pinos *General Purpose Input/Output* (GPIO) são responsáveis por receber ou enviar informações digitais a outros pinos. Os pinos *ADC (Analog to Digital Converter)* permitem também receber ou enviar informações a outros pinos, porém de entrada ou saída analógicas.(ESPRESSIF, 2024). Na Figura 3 dispõe-se da pinagem do ESP32.

Figura 3. Pinagem do ESP32.



Fonte: Usinainfo, 2024.

2.3. Sensores, Atuadores e Demais Componentes

2.3.1. Servomotor

Segundo Nussey (2019), Os servomotores têm um conjunto de circuito na própria placa responsável pela verificação sua rotação ou

posição. Os servomotores são frequentemente utilizados para operações de precisão, como a abertura controlada de válvulas ou o movimento de robôs.

Um servomotor é composto de um motor e um dispositivo chamado codificador ou sensor de velocidade que consegue acompanhar a rotação do motor. Os servomotores são usados para movimentos precisos, movimentando-se em inúmeros graus para um local exato. Ao utilizar seu Arduino, você pode dizer ao servomotor em que grau quer que ele se mova, e ele se movimentará a partir de sua posição atual. A maioria dos servomotores pode se movimentar apenas em 180 graus, contudo, você pode usar uma engrenagem para ampliar esse intervalo (Nussey, 2019, p. 133).

Figura 4. Imagem de um servomotor.



Fonte: Usinainfo, 2024.

2.3.2. Sensor Flexível

Segundo Eletrodex (2024), É um sensor cuja resistência varia conforme o sensor é flexionado. Quando maior a força aplicada,

maior é a resistência na saída do sensor, cujos valores de referência são de resistência mínima de 30K Ohms e máxima de 70K Ohms.

Figura 5. Imagem de um sensor flexível



Fonte: Eletrodex, 2024.

2.3.3. Resistor

Segundo Junior (2021), Os resistores são dispositivos eletrônicos simples e passivos, cuja principal finalidade é transformar energia elétrica em outro tipo de energia. Por exemplo, no chuveiro elétrico, o resistor transforma a energia elétrica em calor ou energia térmica; nos motores elétricos, ele transforma a energia elétrica em movimento ou energia mecânica; e nas lâmpadas, ele transforma a energia elétrica em luz. Os resistores podem ser classificados em dois tipos: fixos e variáveis. Os resistores fixos possuem valor de resistência teoricamente fixos, já os resistores variáveis, variam de acordo com certas condições, como temperatura, quantidade de luminosidade ou fisicamente.

Figura 6. Imagem de um resistor



Fonte: Junior (2021, p. 186).

3. METODOLOGIA

3.1. Metodologia de Pesquisa

O método que foi utilizado para o desenvolvimento deste trabalho tem sua natureza como uma pesquisa aplicada, pois tem como objetivo gerar conhecimento a partir de testes práticos. A forma de abordagem do problema é o de pesquisa qualitativa, já que a ideia do trabalho é validar a alternativa apresentada para controle de prótese utilizando conceitos de internet das coisas. Do ponto de vista dos objetivos é uma pesquisa explicativa, com o intuito de mostrar como se aplica algo. Já do ponto de vista dos procedimentos técnicos é uma pesquisa bibliográfica e experimental, visto que toda base do trabalho é feita a partir de uma revisão bibliográfica, para então essa teoria ser aplicada na prática sendo possível sua validação.

3.2. Materiais e Métodos

Para a criação deste projeto, foram usados dois microcontroladores EPS32, sendo um para o emissor de informações, no caso, a luva com sensores flexíveis, e o segundo para o módulo receptor, que irá traduzir as informações recebidas em valores de ângulo para posicionamento de cada dedo da prótese. A prótese foi impressa em 3D, utilizando-se uma impressora Creality-3D, modelo ender-3v2, o modelo escolhido foi disponibilizado gratuitamente no site thingiverse.com. A prótese foi dotada de movimentação através do uso de cinco unidades do servomotor cento e oitenta graus modelo

MG996, controlados com auxílio de um driver de motor PCA9685. A luva emissora de posições foi construída utilizando-se a associação de resistores de 30K Ohm em série com o sensor flexível de 45mm, criando um divisor de tensão, necessário para leitura da alteração dos valores de resistência dos sensores de cada dedo.

4. DESENVOLVIMENTO E RESULTADOS

4.1. Conceito Global

Antes de explicar o funcionamento e interligação dos componentes específicos, é importante conceituar o modelo em sua totalidade. Será utilizado como protótipo para a realização destes estudo, um modelo 3D de mão robótica, dotado de articulações baseadas em uma mão humana, o modelo foi disponibilizado online, gratuitamente, pelo seu desenvolvedor, a impressão aconteceu utilizando-se uma impressora 3D, com filamento PLA (polylactic acid ou ácido poliláctico, em tradução livre). O modelo encontra-se sobre uma base que permitisse a instalação tanto dos motores, para realização dos movimentos simulados, quanto da parte lógica do sistema.

Para que fosse possível a movimentação das articulações, foi utilizado um sistema com fios de nylon e material elástico, sendo que este possibilitara o retorno a posição inicial enquanto o fio de nylon executa a função de tração para simular o fechamento da mão, conforme figura 7 abaixo.

Figura 7. Modelo da mão robótica impressa em 3D.



Fonte: O Autor, 2024.

Este módulo, doravante chamado de receptor, será quem irá receber as informações do servidor para atuação dos motores e execução dos movimentos previstos.

Ainda em tempo, cabe abordar o módulo emissor, responsável pela captura dos movimentos da mão de um operador humano, através de um sistema de sensores flexíveis, que serão enviados para o servidor *broker* através do ESP32.

4.2. Módulo Emissor

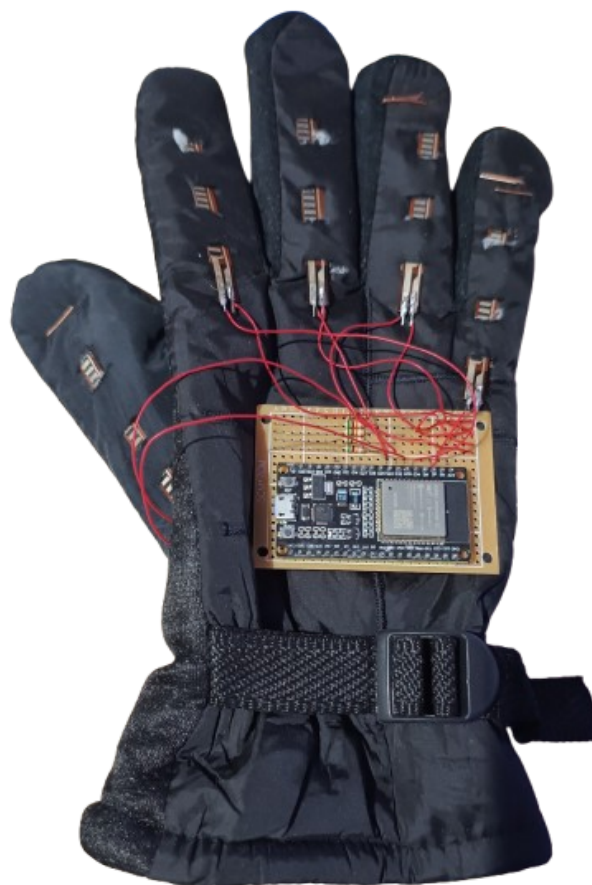
O módulo emissor foi construído utilizando-se como base física uma luva comum, visando a usabilidade do operador, que irá calçá-la como qualquer item de vestimenta simples, capturando com máxima eficiência e exatidão possível os seus movimentos.

Para cada dedo foi anexado um sensor flexível (fitas marrons em cada dedo da luva) associado em série a um resistor de 30K Ohms (30 mil ohms), para que fosse possível a leitura da diferença de tensão nos pinos do ESP32, causado pela variação de resistência em decorrência da flexão física dos sensores flexíveis.

Quanto à parte lógica do sistema, o ESP32 foi carregado com as bibliotecas de Wifi e a PubSubClient, para que pudesse se conectar à rede sem fio, bem como aos serviços do protocolo MQTT. O código fonte será comentado em partes principais na sequência.

Abaixo segue a figura 8 que apresenta o resultado final da implementação do protótipo.

Figura 8. Luva com módulo emissor.



Fonte: O Autor, 2024.

O trecho de código a seguir apresenta a parte inicial do código-fonte, desde a importação das bibliotecas citadas anteriormente bem como a declaração em constantes dos valores de SSID e senha de rede para conexão *wifi* e posteriormente acesso à internet.

Figura 9. Trecho de código inicial.

```
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3
4  const char* SSID = "Fernanda";
5  const char* PASSWORD = "21072020";
6  WiFiClient wifiClient;
```

Fonte: O Autor, 2024.

Posteriormente torna-se necessário a declaração de dois valores, sendo o primeiro o endereço do *broker* MQTT, que ofertará os serviços de troca de mensagens entre emissor e receptor, além do *integer* contendo a porta de conexão. Depois de apontar os dados para conexão ao servidor, é necessário criar um nome para o *publisher*, ou publicador, que efetuará o *payload*, que são os pacotes com informações, no caso concreto ficou criado o “BCI01”, nome criado aleatoriamente, cabe salientar que é salutar a criação de um nome com nome não tão comum, evitando conflito com outras identificações do servidor. Neste momento é possível criar o objeto, instanciando a classe através do *client* MQTT, conforme imagem 10.

Figura 10. Trecho de código da conexão com broker.

```
8  //MQTT Server
9  const char* BROKER_MQTT = "mqtt.eclipseprojects.io";
10 int BROKER_PORT = 1883;
11
12 #define ID_MQTT "BCI01"
13 PubSubClient MQTT(wifiClient);
14
```

Fonte: O Autor, 2024.

A função *setup*, conforme imagem 11, limita-se a chamar a função de conexão *wifi* e instanciar, utilizando as configurações declaradas

anteriormente, o servidor MQTT.

Figura 11. Trecho de código da função setup.

```
28 void setup() {
29     Serial.begin(115200);
30     conectaWiFi();
31     MQTT.setServer(BROKER_MQTT, BROKER_PORT);
32 }
33
```

Fonte: O Autor, 2024.

A sequência de repetição programada é reponsável pela manutenção das conexões de rede, em caso de perda de conexão, ele tentará efetuar a reconexão, após executa a chamada de função para envio de valores, conforme figura 12 abaixo:

Figura 12. Trecho de código da função enviaValores.

```
84 void enviaValores() {
85     static int sensorde1, sensorind, sensormed, sensorane, sensormin;
86     sensorde1= analogRead(36); //G35
87     sensorind= analogRead(32); //G34
88     sensormed= analogRead(33); //G23
89     sensorane= analogRead(34); //G32
90     sensormin= analogRead(35); //SP
91     itoa(sensorde1, de1, 10);
92     itoa(sensorind, de2, 10);
93     itoa(sensormed, de3, 10);
94     itoa(sensorane, de4, 10);
95     itoa(sensormin, de5, 10);
96     MQTT.publish("DED01", de1);
97     MQTT.publish("DED02", de2);
98     MQTT.publish("DED03", de3);
99     MQTT.publish("DED04", de4);
100    MQTT.publish("DED05", de5);
101 }
```

Fonte: O Autor, 2024.

A função “enviaValores” é responsável pela leitura dos valores dos sensores, foi criado as variáveis, uma para cada dedo, conforme linhas 86 a 90, no comentário de cada linha está o pino de conexão física do ESP32. Para que fosse possível o envio da mensagem, tornou-se necessário a conversão dos valores inteiros lidos para valores literais, através do comando “itoa”, no primeiro argumento é chamado o valor a converter, no segundo a *string* de resultado e no terceiro argumento é a base de conversão, no caso, decimal. Então é possível a publicação dos valores através da função *publish* do

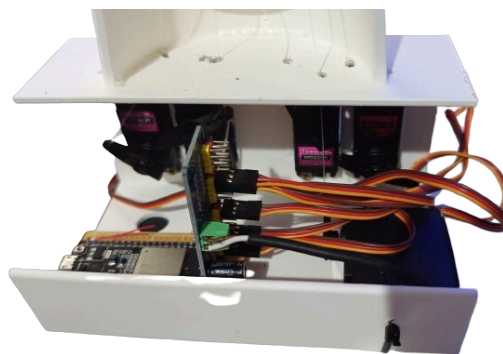
objeto MQTT, para cada dedo foi criado um tópico, evitando a necessidade de tratamentos posteriores da mensagem. Em cada tópico é postado o valor em *string*, correspondente ao valor lido pelo sensor presente na luva.

A partir deste momento os valores lidos pelos sensores já encontram-se disponíveis no servidor na nuvem, ou seja, em qualquer lugar do mundo, com conexão à internet, é possível a coleta das informações.

4.3. Módulo Receptor

Ao módulo receptor, configurado como *Subscriber* (assinante), ficou delegado a função de conexão e coleta das informações para convertê-las em movimento do protótipo, para tal foi instalado um segundo ESP32, o qual controla um driver de motores modelo PCA9685, facilitando a operação dos cinco motores servo MG996, de cento e oitenta graus, que finalmente criarão movimento das articulações. Na imagem abaixo é possível verificar o resultado final da implementação realizada.

Figura 13. Implementação do módulo receptor.



Fonte: O Autor, 2024.

O código fonte inicia com a importação das bibliotecas Wifi, para conexão à rede e internet, PubSubClient para utilização do protocolo

MQTT, Wire para utilização do driver de motores em arranjo I2C, que é um protocolo de comunicação serial que permite a troca de dados entre dispositivos e a biblioteca Adafruit_PWMServoDriver para efetivo controle do módulo de driver de motores modelo PCA9685. Na sequência foi instanciado o objeto chamado de pca9685, orientando a conexão I2C através do endereço 0x40 e criado uma variável global chamada pwm0 para utilização futura.

Figura 14. Código fonte inicial do módulo receptor.

```
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3  #include <Wire.h>
4  #include <Adafruit_PWMServoDriver.h>
5  Adafruit_PWMServoDriver pca9685 = Adafruit_PWMServoDriver(0x40);
6  int pwm0;
```

Fonte: O Autor, 2024.

Posteriormente será necessário a criação de constantes responsáveis pelas informações de validação da rede wifi e referentes ao endereço e porta do servidor broker. O próximo passo é a definição da identidade do dispositivo na rede MQTT, no caso concreto ficou definido o nome de “BCI02”, diferente do módulo emissor, evitando conflitos e por fim, a criação do objeto de conexão, instanciando a classe através do client MQTT.

Figura 15. Código fonte da conexão do módulo receptor.

```
9  const char* SSID = "Fernanda";
10 const char* PASSWORD = "21072020";
11 WiFiClient wifiClient;
12
13 const char* BROKER_MQTT = "mqtt.eclipseprojects.io";
14 int BROKER_PORT = 1883;
15
16 #define ID_MQTT "BCI02"
17 PubSubClient MQTT(wifiClient);
```

Fonte: O Autor, 2024.

A função *Setup* do módulo receptor fica responsável pela inicialização do objeto pca9685 e definição da frequência de pwm

em 50 (cinquenta), valor suficiente para o modelo de servomotor utilizado. Ele também iniciará a conexão com a rede, efetuará a conexão ao servidor MQTT e chamará a função para receber pacotes do servidor, sendo que este ESP32 está funcionando no modo *Subscriber*, ou seja, ele apenas irá receber os dados do emissor, ou seja, do *Publisher*.

Figura 16. Função Setup do módulo receptor.

```
40 void setup() {  
41     Serial.begin(115200);  
42     pca9685.begin();  
43     pca9685.setPWMFreq(50);  
44     Serial.begin(115200);  
45     conectaWiFi();  
46     MQTT.setServer(BROKER_MQTT, BROKER_PORT);  
47     MQTT.setCallback(recebePacote);  
48 }
```

Fonte: O Autor, 2024.

No trecho abaixo é apresentado a função de conexão ao MQTT, inicialmente, enquanto a condição de não conectado permanecer, ele tentará novamente conectar o broker, assim que ele realizar a conexão ele poderá então assinar os tópicos criados pelo módulo emissor onde estarão os valores de cada um dos sensores do módulo emissor.

Figura 17. Código fonte da função conectaMQTT do módulo receptor.

```
83 void conectaMQTT() {  
84     while (!MQTT.connected()) {  
85         Serial.print("Conectando ao Broker MQTT: ");  
86         Serial.println(BROKER_MQTT);  
87         if (MQTT.connect(ID_MQTT)) {  
88             Serial.println("Conectado ao Broker com sucesso!");  
89             MQTT.subscribe("DED01");  
90             MQTT.subscribe("DED02");  
91             MQTT.subscribe("DED03");  
92             MQTT.subscribe("DED04");  
93             MQTT.subscribe("DED05");  
94         }  
95         else {  
96             Serial.println("Nao foi possivel se conectar ao broker.");  
97             Serial.println("Nova tentatica de conexao em 10s");  
98             delay(2000);  
99         }  
100     }  
101 }
```

Fonte: O Autor, 2024.

Por fim temos a coleta e tratamento dos valores lidos, a função `recebePacote` fica encarregada de coletar o *payload* do *broker* e ao mesmo tempo efetuar a escrita do valor convertido ao seu respectivo motor, atrelado a cada dedo.

Inicialmente o *char* topic* é convertido para uma *string* simples visando a comparação com uma outra *string*. Caso o *topic* recebido seja algum dos tópicos esperados (DEDO1, DEDO2, até DEDO5), ele entrará no laço condicional para que cada tópico possa ser associado ao seu dedo correspondente. É criado então uma variável que irá receber o texto da mensagem do tópico, o laço de repetição irá criar, conforme o tamanho da mensagem, uma matriz de caracteres, na mesma sequência dos dados recebidos via *payload*, ou seja, ao final deste laço, teremos a variável “msg” contendo exatamente o texto original enviado pelo emissor, porém dados literais não podem ser utilizados como dados numéricos, devendo ser convertidos através da função “*toInt*”.

Na posse dos valores numéricos referentes ao valor dos sensores, pode ser realizado um mapeamento deste valor, para convertê-los em valor de ângulo dos servomotores, para tal a função “*map*” é chamada, no primeiro argumento é passado a variável que contém a informação a ser mapeada, nos dois próximos argumentos são passados os valores mínimos e máximos que esta variável pode ter, e por fim, os dois valores de ângulos, correspondentes ao mínimo e ao máximo valores, respectivamente.

Por fim este valor angular, proporcional ao lido pelo sensor, é enviado para o servomotor correspondente a este tópico, movendo o dedo correspondente. A função “*writeServos*”, em seu primeiro argumento, possui a porta do driver PCA9685 e no segundo

argumento o valor angular a ser escrito no mesmo, movendo-o em uma posição específica, conforme imagem 18, abaixo.

Esta sequência é realizada mais quatro vezes, um para cada tópico, ou seja, uma para cada dedo do protótipo.

Figura 18. Código fonte da função `recebePacote` do módulo receptor.

```
103 void recebePacote(char* topic, byte* payload, unsigned int length)
104 {
105     String topicStr = topic;
106     if (topicStr == "DED01"){
107         String msg;
108         static long ded1;
109         for(int i = 0; i < length; i++)
110         {
111             char c = (char)payload[i];
112             msg += c;
113         }
114         ded1=msg.toInt();
115         pwm0 = map(ded1, 2600, 1800, 10, 170);
116         writeServos(0, pwm0);
117     }
```

Fonte: O Autor, 2024.

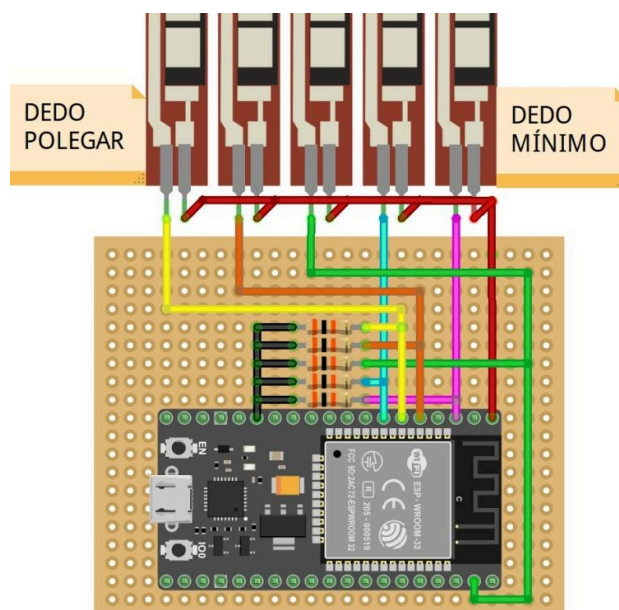
4.4. Diagrama Elétrico

Visando facilitar o entendimento e replicação do protótipo para futuras aplicações e melhorias, será apresentado o diagrama elétrico das interligações dos componentes.

4.4.1. Diagrama Elétrico do Módulo Emissor

Para a leitura de cada dedo, um conjunto em série de um sensor flexível e um resistor é necessário, no caso prático, cada resistor foi conectado ao pino GND do ESP32 (fios pretos), da mesma forma que um dos polos do sensor flexível ao pino com saída de 3,3V (fio vermelho). cada nó entre o resistor e o sensor flexível deve ser ligado aos pinos de leitura do microcontrolador, para que o mesmo possa realizar a leitura conforme a deformação física aplicada ao sensor (fios amarelo, laranja, verde, ciano e rosa).

Figura 19. Diagrama elétrico do módulo emissor.



Fonte: O Autor, 2024

4.4.2. Diagrama Elétrico do Módulo Receptor

Para que seja possível o controle dos motores, será utilizado uma fonte externa de 5VDC (cinco volts em corrente contínua) que deve ser plugada no borne superior, a qual será responsável também pela alimentação do módulo PCA9685, através do jumper interligando os pinos de V+ e VCC, conforme fio rosa.

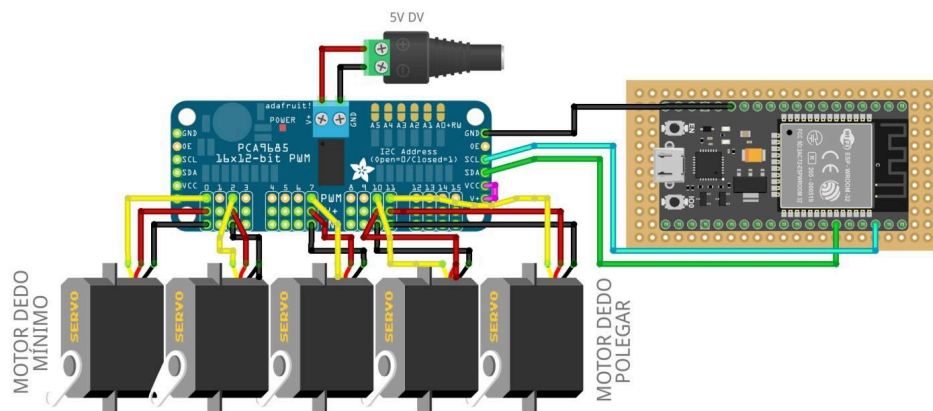
Foi utilizado as portas 00, 02, 07, 10 e 11 para conexão dos cinco servomotores, por escolha pessoal, podendo ser utilizado quaisquer portas, desde que acompanhe o código programado. As pinagens do servomotor são de sinal para o fio amarelo, tensão positiva para o fio vermelho e GND para o fio preto.

A conexão I2C utiliza os pinos SCL e SDA do módulo driver, conectados aos pinos 22 e 21, respectivamente.

De forma comum a qualquer interligação de componentes eletrônicos, torna-se necessário o compartilhamento do GND de

todos circuitos, garantindo assim a referência de tensão necessária ao seu bom funcionamento, representado aqui pelo fio preto que conecta o módulo PCA9685 e o ESP32.

Figura 20. Diagrama elétrico do módulo receptor.



Fonte: O Autor, 2024

4.5. Resultados Obtidos

Com a finalização do protótipo controlado por IoT, foi possível validar o funcionamento e a eficácia da integração entre os dispositivos conectados e a rede de comunicação estabelecida. O sistema demonstrou uma performance consistente na execução das tarefas propostas, destacando-se pela estabilidade da conexão e pela precisão nas respostas aos comandos enviados via interface de controle.

Os sensores integrados ao protótipo capturaram dados em tempo real, proporcionando um monitoramento contínuo dos parâmetros desejados. A latência da comunicação entre o controlador central e os dispositivos IoT foi mínima, cerca de um segundo entre comando e ação, garantindo uma experiência de uso fluida e sem interrupções significativas. Além disso, o sistema mostrou-se escalável, permitindo a adição de novos dispositivos e sensores sem comprometer o desempenho geral.

A interface de controle desenvolvida, demonstrou-se intuitiva e eficaz, permitindo que o usuário interagisse de forma prática com o protótipo, o que agrega grande valor ao seu potencial de utilização em cenários variados.

Em termos de consumo energético, o protótipo operou de forma eficiente, corroborando com a expectativa de baixo consumo, fundamental em aplicações IoT.

Esses resultados demonstram que o protótipo desenvolvido é uma solução viável para aplicações IoT, sendo capaz de oferecer controle remoto confiável, monitoramento contínuo e escalabilidade. Isso o posiciona como uma base para futuros aprimoramentos e implementações em ambientes reais.

5. CONCLUSÃO

O desenvolvimento e implementação do protótipo controlado por IoT alcançaram os objetivos propostos, demonstrando a viabilidade de integração de dispositivos conectados para controle remoto em tempo real. Os resultados obtidos evidenciaram um sistema eficiente em termos de conectividade, latência, segurança e consumo energético, reforçando o potencial dessa solução em diversas aplicações práticas.

Do ponto de vista dos objetivos específicos, foram realizados os estudos acerca da internet das coisas, sobre o ESP32, bem como dos demais componentes necessários para esta construção, logo tendo sido atingidos ao longo da construção do trabalho.

A arquitetura desenvolvida mostrou-se escalável e adaptável, o que facilita a adição de novos dispositivos. Além disso, a interface

intuitiva de controle remoto proporcionou uma experiência de usuário positiva, permitindo a operacionalização do sistema de maneira simples e eficaz.

Diante disso, o protótipo serve como uma base promissora para futuras inovações e melhorias. As possibilidades de expansão incluem a adoção de tecnologias emergentes, como redes 5G e inteligência artificial. Não foi criado estratégias de proteção ou de segurança da informação, tema que também pode ser objeto de novas pesquisas.

Em relação à importância do tema, pode-se apontar que o modelo de aplicação dos protocolos e utilização do ESP32 como microcontrolador central trazem inúmeras possibilidades de automações e controles através da internet das coisas com poucas alterações no projeto inicial. A simples aplicação de outros sensores e atuadores, junto a uma lógica específica podem dar origem a diversos projetos relevantes.

Dessa forma, o projeto se alinha às tendências de inovação tecnológica, oferecendo uma plataforma flexível e segura para o avanço da Internet das Coisas.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, David; PEIXOTO, Mario; ROSA, Thiago. **Internet das Coisas (IoT): Segurança e privacidade de dados pessoais**. Rio de Janeiro: Editora Alta Books, 2021. E-book. ISBN 9786555202793. Disponível em:

<https://integrada.minhabiblioteca.com.br/#/books/9786555202793/>.

Acesso em: 23 de set. de 2024.

BARROS, Marcelo. **MQTT — protocolos para IoT. In: EMBARCADOS.** [S. l.: s. n.], 2015. Disponível em: <https://www.embarcados.com.br/mqtt-protocolos-para-iot/>. Acesso em: 23 de set. de 2024.

ESPRESSIF. **ESP32: A diferente lot and Power Performance.** 2014. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acessado em 23 de set. de 2024.

FILIPEFLOP. **Módulo WiFi ESP32 Bluetooth.** Disponível em <https://www.filipeflop.com/produto/modulo-wifi-esp32-bluetooth/>. Acessado em 24 de set. de 2024.

JOÃO, Belmiro do Nascimento.; SOUZA, Crisomar Lobo de; SERRALVO, Francisco Antônio. **Revisão sistemática de cidades inteligentes e internet das coisas como tópico de pesquisa.** Cadernos EBAPE.BR, v. 17, no. 4, jan. 2020.

JÚNIOR, José Marcelo de Assis Wendling; MARTIN, Andrea Acunha; MARGOTI, Luciana Margoti. **Eletricidade.** Porto Alegre: Grupo A, 2021. E-book. ISBN 9786556901787. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9786556901787/>. Acesso em: 23 de set. de 2024.

MARTINS, Ismael Rodrigues.; ZEM, José Luís. **Estudo dos protocolos de comunicação MQTT e COaP para aplicações machine-to-machine e Internet das coisas.** Revista Tecnológica da Fatec Americana, v. 3, n. 1, 2015. Disponível em: <https://fatecbr.websiteseuro.com/revista/index.php/RTecFatecAM/article/view/41/50>. Acesso em: 23 de ago. de 2024.

MASCHIETTO, Luís Gustavo; VIEIRA, Anderson Luis Nogueira; TORRES, Fernando Esquírio; et al. **Arquitetura e Infraestrutura de IoT**. Porto Alegre: Grupo A, 2021. E-book. ISBN 9786556901947. Disponível em:

<https://integrada.minhabiblioteca.com.br/#/books/9786556901947/>.

Acesso em: 23 de set. de 2024.

NUSSEY, John. **Arduino Para Leigos**. Rio de Janeiro: Editora Alta Books, 2019. E-book. ISBN 9788550808383. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788550808383/>.

Acesso em: 23 de set. de 2024.

PACHECO, Luis Alberto Belém. **Arquitetura para privacidade na integração de internet das coisas e computação em nuvem**.

Dissertação (Mestrado em Informática) — Departamento de Ciências da Computação, Universidade de Brasília, Brasília, 2018.

Disponível em: https://repositorio.unb.br/bitstream/10482/32137/1/2018_LuisAlbertoBelemPacheco.pdf. Acesso em: 24 de set. de 2024.

SANTOS, Bruno; LUCAS, Silva, CLAYSON, Celes, João B. BORGES, João; et al. **Internet das coisas: da teoria à prática**. 2016. Disponível em: <https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf>. Acesso em: 24 de set. de 2024.

SINCLAIR, Bruce. **IoT: como usar a internet das coisas para alavancar seus negócios**. São Paulo: Grupo Autêntica, 2018. E-book. ISBN 9788551303559. Disponível em:

<https://integrada.minhabiblioteca.com.br/#/books/9788551303559/>.

Acesso em: 23 de set. de 2024.

USINAINFO, **ESP32 Doit DevKitC V4 WROOM-32D**. Disponível em: <https://www.usinainfo.com.br/esp32/esp32-doit-devkitc-v4-wroom-32d-8682.html>. Acesso em 24 de set. de 2024.

USINAINFO, **Servo Motor MG996R Tower Pro 180° 11kgf.cm de Posição com Engrenagens Metálicas**. Disponível em: <https://www.usinainfo.com.br/servo-motores/servo-motor-mg996r-tower-pro-180-11kgfcm-de-posicao-com-engrenagens-metalicas-4850.html>. Acesso em 24 de set. de 2024.

¹ Docente do curso de Bacharelado em Engenharia de Software, Ugv - Centro Universitário. Bacharel em Sistemas de Informação e Especialista em Didática e Docência do Ensino Superior pelo Centro Universitário do Vale do Iguaçu. E-mail: [acesse o artigo original para visualizar o e-mail](#)

² Professor Mestre e Coordenador do Curso de Engenharia de Software, do Ugv - Centro Universitário. E-mail: [acesse o artigo original para visualizar o e-mail](#)