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

ANNA JÚLIA CAMARGO DIAS

SISTEMA EMBARCADO PARA CONTROLE DE MICROPLÁSTICOS

Sorocaba
2024

ANNA JÚLIA CAMARGO DIAS

SISTEMA EMBARCADO PARA CONTROLE DE MICROPLÁSTICOS

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

Orientador: Prof. Dr. Ivando Severino Diniz

Sorocaba

2024

D541s

Dias, Anna Júlia Camargo
Sistema embarcado para controle de microplásticos / Anna Júlia Camargo Dias. -- Sorocaba, 2024
46 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Ivando Severino Diniz

1. Automação. 2. Sistemas embarcados (Computadores). 3. Internet das coisas. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Universidade Estadual Paulista (UNESP), 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

SISTEMA EMBARCADO PARA CONTROLE DE MICROPLÁSTICOS

ANNA JÚLIA CAMARGO DIAS

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

Prof. Dr. Everson Martins
Coordenador

BANCA EXAMINADORA:

Prof. Dr. IVANDO SEVERINO DINIZ
Orientador/UNESP - Campus de Sorocaba

Prof. Dr. EVERSON MARTINS
UNESP- Campus de Sorocaba

Prof. DANILO RUY GOMES
UNESP – Campus de Sorocaba

Sorocaba
11 de junho de 2024

Dedico este trabalho, e todos os outros que virão, à M, S, H, Y, SG, o Barbudo e ao B, que estão sempre comigo, me ensinando e me ajudando a recordar a ser quem sou, mesmo que eu não entenda 70% do que dizem, e me irritando em vários momentos da minha jornada.

Também o dedico à Luciana e ao Jotíssimo. Sem o apoio deles, tanto o curso quanto este trabalho não seriam concluídos.

AGRADECIMENTOS

Ao meu professor orientador Prof. Dr. Ivando Severino Diniz por todo o conhecimento transmitido, ajuda e paciência para a realização deste trabalho, e ao Prof. Dr. Walter Ruggeri Waldman pelas conversas sobre a integração entre automação e microplásticos, que em muito contribuíram para a ideia deste trabalho.

Aos meus professores da graduação por todo o ensinamento me auxiliando a formar a minha carreira profissional.

À meus pais e irmão, que estão sempre ao meu lado, e me surpreendendo a cada dia pelo apoio e amor imensurável.

Gostaria de agradecer aos meus amigos Blenda, Bianca, Johnnatan e Fernanda, por essa grande jornada juntos. Foram momentos incríveis que eu levarei comigo sempre. Um destaque especial ao Johnnatan, que em momentos difíceis, não só me auxiliou como amigo, mas também ensinou os mais diversos assuntos.

À Janaína por toda ajuda durante a minha graduação. Suas palavras e apoio foram fundamentais para a minha continuidade no curso. Gratidão eterna.

RESUMO

A necessidade de ambientes controlados é rotina em muitos laboratórios de pesquisa. No caso de laboratórios de microplásticos, é muito importante que a quantidade de partículas no ar esteja dentro do limite permitido, de forma que não interfiram nos resultados obtidos mediante os experimentos realizados. Diante deste problema, foi desenvolvido um sistema automático que mede a quantidade de partículas no ambiente, por meio do sensor de poeira GP2Y1010AU0F da Sharp, e envia os dados ao microcontrolador ESP-32 DEV KIT V1. Este, o qual foi programado na IDE do Arduino versão 2.3.2, envia os dados à rede pelo protocolo de comunicação MQTT, podendo ser acessados tanto pelo computador (brokers online), quanto pelo aplicativo IoT MQTT Panel, e aciona o ventilador e o LED, caso a quantidade de particulados seja maior que o limite estipulado. Ademais, o código elaborado permite que o usuário do aplicativo torne o controle do sistema manual, possibilitando-o de ligar e desligar o ventilador a seu critério. Também é possível visualizar um gráfico da densidade de poeira em função do tempo, para futuras análises. Assim, com o desenvolvimento deste trabalho foi possível a diminuição da densidade de partículas próximas ao sistema.

Palavras-chave: microplásticos; sensor de poeira; ESP-32; MQTT; IoT MQTT Panel.

ABSTRACT

The need for controlled environments is routine in many research laboratories. In the case of microplastics laboratories, it is very important that the amount of particles in the air is within the permitted limit, so that they do not interfere with the results obtained through the experiments carried out. Faced with this problem, an automatic system was developed that measures the amount of particles in the environment, using Sharp's GP2Y1010AU0F dust sensor, and sends the data to the ESP-32 DEV KIT V1 microcontroller. The ESP-32, which was programmed in the Arduino IDE, sends data to the network through the MQTT communication protocol, which can be accessed both by the computer (online brokers) and by the IoT MQTT Panel app, and activates the fan and the LED, if the amount of particulates is greater than the stipulated limit. Furthermore, the code created allows the app user to control the system manually, enabling them to turn the fan on and off at their will. It is also possible to view a graph of dust density as a function of time, for future analysis. Thus, with the development of this work it was possible to reduce the density of particles close to the system.

Keywords: microplastics; dust sensor; ESP-32; MQTT; IoT MQTT Panel.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Plásticos, microplásticos e sistema de limpeza automático

Os plásticos são provenientes de resinas derivadas do petróleo, pertencentes ao grupo dos polímeros. Seu nome deriva da língua grega, e significa ‘aquilo que pode ser moldado’. O primeiro plástico sintético foi criado no início do século XX, sendo rapidamente desenvolvido a partir de 1920. Atualmente é um dos principais materiais utilizados pela sociedade, principalmente em embalagens, e, devido a seu descarte incorreto, tem se tornado um sério problema ambiental, pois sua degradação acarreta no surgimento dos microplásticos (ECOMARAPENDI, 2020).

Os microplásticos são partículas de plástico com um tamanho máximo 5 mm, sendo provenientes de resíduos industriais, residenciais e de transporte marítimo quando de origem primária. Já de origem secundária, são provenientes de processos de degradação dos plásticos. Os polímeros que geram tais micropartículas são: polietileno tereftalato (PET), polipropileno (PP), poliestireno (PS), poliuretano (PU), policloreto de vinila (PVC) e náilon (PA).

No início do século XXI foi descoberta a presença de microplásticos tanto em ambientes de água doce, quanto marinhos, devido ao descarte incorreto de materiais plásticos, e diversos estudos têm sido feitos para analisar seu impacto nestes e em outros meios. Os resultados destes estudos apontaram que a presença de microplásticos em ambientes de água doce e marinho acarretaram em sua ingestão por animais aquáticos, causando danos físicos, como asfixia, lesão nos órgãos internos e bloqueio do trato gastrointestinal, como também danos tóxicos. Logo, o crescimento dos microplásticos têm prejudicado drasticamente diversos seres vivos (SARRIA-VILLA; GALLO-CORREDOR, 2016).

Diante dos crescentes problemas gerados pelos microplásticos, vários estudos têm sido realizados envolvendo a temática microplásticos a fim de minimizar os efeitos gerados no ambiente. Neste contexto, é necessário que os laboratórios, os quais desenvolvem pesquisas sobre tais partículas, apresentem atmosferas limpas.

O desenvolvimento da tecnologia tem possibilitado a criação de sistemas automatizados que permitem melhorias em diversos setores, como os de pesquisas científicas.

1.2 Justificativa e Objetivos

Com o intuito de auxiliar nos estudos e pesquisas dos microplásticos realizados em laboratórios, o presente trabalho tem como objetivo apresentar um sistema que mede e registra a quantidade de particulados no ar, e, caso esta seja maior que um valor tolerável, seja ligado um ventilador para retirar os particulados presentes na atmosfera. Isto porque a análise de microplásticos deve ser realizada em ambientes limpos, de forma que os resultados de um determinado experimento não sejam afetados pela própria presença de microplásticos no ar, como por exemplo, sua liberação proveniente das fibras têxteis das roupas utilizadas pelo pesquisador.

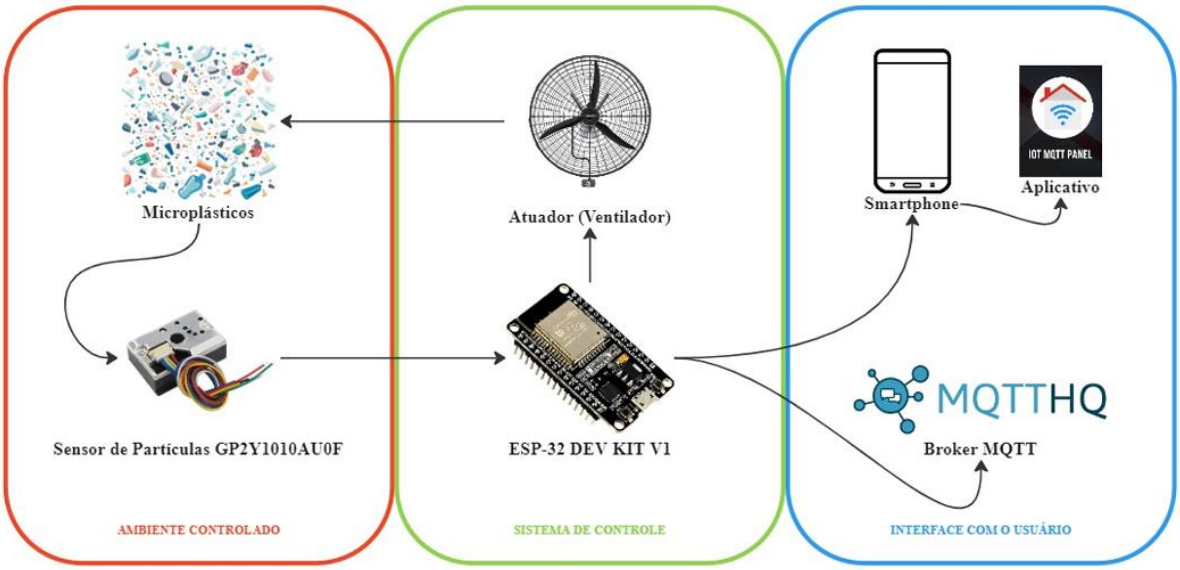
Com os recursos disponíveis foi criada uma montagem com os seguintes componentes: um sensor de partículas (GP2Y1010AU0F), um capacitor de $220\,\mu F$, dois resistores (um de $150\,\Omega$ e outro de $220\,\Omega$), um LED vermelho, um microcontrolador (ESP-32), um módulo relé, uma bateria de $5\,V$, um ventilador e um celular.

O sensor GP2Y1010AU0F mede a quantidade de partículas presentes e envia seus valores (em forma de tensão) ao microcontrolador. Este, pelo protocolo de comunicação MQTT, encaminha os dados a um aplicativo (IoT MQTT Panel), para que o usuário possa visualizá-los em tempo real. Além disso, o aplicativo foi configurado para possibilitar duas opções de sistema: automático e manual. No primeiro caso, o ventilador é acionado automaticamente a partir do instante em que a quantidade de microplásticos atinge um valor pré-estabelecido. Já no segundo caso, o usuário determina se os filtros devem ou não ser ligados (pressionando um botão switch). O estado do ventilador (ligado ou desligado) pode também ser observado através de um LED no circuito (acendendo quando o ventilador é ligado), para facilitar sua visualização.

Ademais, o sistema possibilita a observação dos dados em forma de um gráfico (densidade de partículas em função do tempo), permitindo que sejam feitas análises de quais períodos do dia há um aumento de microplásticos presentes no ar, seja em momentos em que estão sendo realizados experimentos, ou quando a porta do laboratório é aberta, por exemplo.

O sistema automatizado pode ser dividido, de forma simplificada, em três partes: ambiente controlado, sistema de controle e interface com o usuário (Figura 01). O primeiro é composto pela substância medida e pelo sensor; o segundo pela placa microcontroladora e pelo atuador; e o terceiro pelo celular, o qual possui o aplicativo IoT MQTT Panel, e pelo broker MQTT.

Figura 1 - Diagrama do sistema automatizado desenvolvido para controle de microplásticos no ambiente.



Fonte: Autoria própria.

2 DESCRIÇÃO TEÓRICA

2.1 Revisão da literatura

O sistema automatizado desenvolvido neste trabalho possui uma ampla aplicabilidade nos setores residenciais, agrícolas e industriais, conforme observado durante a revisão da literatura.

2.1.1 Setor Residencial

No setor residencial a automação tem sido amplamente utilizada para as mais diversas funcionalidades, destacando-se a automação de interiores. Os sistemas para controle de parâmetros como temperatura e umidade do ar, por exemplo, têm sido cada vez mais viáveis e eficazes (MARTINS, 2019).

Um sistema para o controle de temperatura permite um maior conforto nas residências, pois evita alterações bruscas desta variável. Já no caso da umidade, seu controle permite grandes benefícios à saúde humana, especialmente para pessoas com problemas respiratórios.

Martins (2019) teve por objetivo a aplicação de um sistema capaz de monitorar a temperatura, umidade e alarme de segurança de cômodos como cozinha e escritório de uma residência. Para isso utilizou microcontroladores ESP32, ESP8266 e a ferramenta Node-RED, os quais permitem o acionamento de alarmes e de saídas representadas por LED’s. Nesse estudo foi possível verificar que a aplicação dessas ferramentas está tornando a automação mais acessível, pois procura auxiliar cada vez mais o encontro de um equilíbrio entre o conforto, a economia e a sustentabilidade, evidenciando que esse tipo de automação permite economias de energia, auxiliando e minimizando impactos ambientais (MARTINS, 2019).

2.1.2 Setor Agrícola

No setor agrícola observa-se que tanto o controle quanto a automação são os protagonistas da quarta revolução agrícola (Agricultura 4.0).

A primeira revolução agrícola foi simultânea à revolução industrial e se caracterizou pelo desenvolvimento e utilização de arados mecanizados e tratores. A segunda revolução foi caracterizada pela utilização de fertilizantes, sendo considerada uma revolução química. Já a terceira revolução se deu por meio do desenvolvimento da engenharia genética, permitindo o crescimento de diferentes cultivares em várias estações do ano, fato que anteriormente não era possível devido a sua falta de resistência a determinada época do ano, principalmente em culturas sazonais (CANTÚ; MONTEZ, 2020).

Assim, nessa quarta revolução agrícola (Agricultura 4.0) destacam-se as técnicas computacionais associadas a dispositivos inteligentes atuais, os quais mostraram-se promissores para o desenvolvimento agrícola, associados à necessidade de minimização de custos e desperdícios e ao aumento da produtividade (CANTÚ; MONTEZ, 2020).

Além das evidências no desenvolvimento e aprimoração na agricultura no campo, destacam-se também sistemas de cultivo indoor. Como o nome já diz, trata-se do desenvolvimento agrícola em um ambiente fechado e controlado, sendo independente de fatores externos como luz solar e precipitações. O desenvolvimento e utilização dessa técnica traz inúmeros benefícios, tais como maior controle das condições de cultivo, menor exposição às pragas e consequentemente minimização do desenvolvimento de doenças (JATÍ, 2023).

Neste contexto, Jatí (2023) desenvolveu um sistema para o cultivo de plantas utilizando o princípio da internet das coisas (IoT). A programação deste sistema foi feita na IDE do Arduino, com a inserção de bibliotecas para um funcionamento adequado e completo dos componentes, sendo estes sensores e atuadores que juntos controlam a temperatura e umidade do solo, bem como a iluminação do ambiente. Os dados deste sistema são transmitidos via protocolo de comunicação MQTT para um monitoramento mobile. Este projeto mostrou-se eficiente dentro da proposta oferecida, pois evidenciou o ‘low cost’, já que foram utilizados dispositivos que podem ser obtidos em lojas de eletrônicos acessíveis, bem como softwares e programas gratuitos, disponíveis online (JATÍ, 2023).

2.1.3 Setor Industrial

Análogo ao que ocorre no setor agrícola (item 2.2), o setor industrial também tem se aprimorado com o que é nomeado de quarta revolução industrial (Indústria 4.0). Esse conceito refere-se à automação industrial, associada à tecnologia da informação, sendo seu principal objetivo a melhoria dos processos industriais e o aumento da produtividade com minimização de desperdícios desnecessários promovendo a sustentabilidade. Justamente o que torna a quarta revolução industrial tão única e distinta das demais é devido à associação de diferentes tecnologias (TEIXEIRA FILHO, 2022).

A primeira revolução industrial foi caracterizada pelo processo de mecanização, a segunda pela produção em massa e a terceira pela automação. Além da integração entre as tecnologias, a quarta revolução industrial é sustentada principalmente pelo pilar da descentralização lógica dos processos, tendo cada etapa sua autonomia, além da modularidade e a interoperabilidade (TEIXEIRA FILHO, 2022).

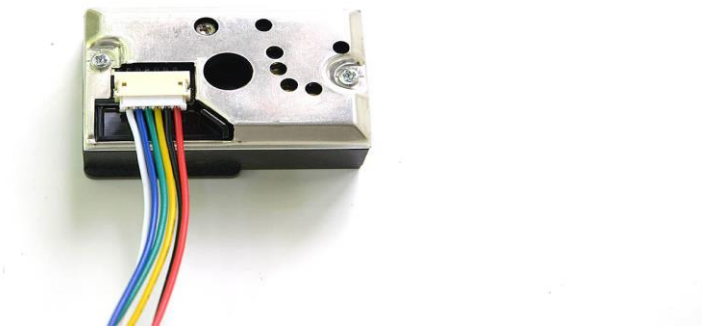
Assim, diversos são os trabalhos desenvolvidos com a aplicação industrial baseados na indústria 4.0. Como exemplo, pode-se citar o trabalho realizado por Teixeira Filho (2022) que buscou apresentar uma solução em automação para um sistema eletropneumático baseado em código aberto, utilizando o conceito de IoT (Internet of Things). Assim, foi desenvolvida uma nova forma de comunicação em relação aos sistemas eletropneumáticos, criando uma rede entre sensores e válvulas que utilizam o ZigBee e o MQTT como protocolo de troca de mensagens, a fim de facilitar a atual sequência de movimentação baseada nos métodos Ladder e Grafcet. Os sensores dos atuadores enviam mensagens ZigBee para as válvulas, que por sua vez enviam mensagens para um broker MQTT (Mosquitto), sendo o ponto de comunicação entre os atuadores. O autor pôde demonstrar a aplicabilidade da indústria 4.0, pois evidenciou a integração entre a informática e as tecnologias de automação, permitindo a supervisão facilitada dos processos (TEIXEIRA FILHO, 2022).

2.2 Sensores e Sensor de Óptico de Poeira GP2Y1010AU0F

O sensor é um dispositivo que detecta uma determinada entrada, seja ela luz, calor, movimento, ou qualquer outro estímulo físico, e a converte em um sinal o qual pode ser transformado em outra grandeza física.

O sensor utilizado foi o GP2Y1010AU0F da Sharp (Figura 02). Trata-se de um sensor óptico de poeira / qualidade do ar de saída analógica (seis pinos) que funciona com base no princípio de dispersão de laser, muito eficaz na detecção de partículas muito pequenas, sendo amplamente utilizado em sistemas de purificação do ar.

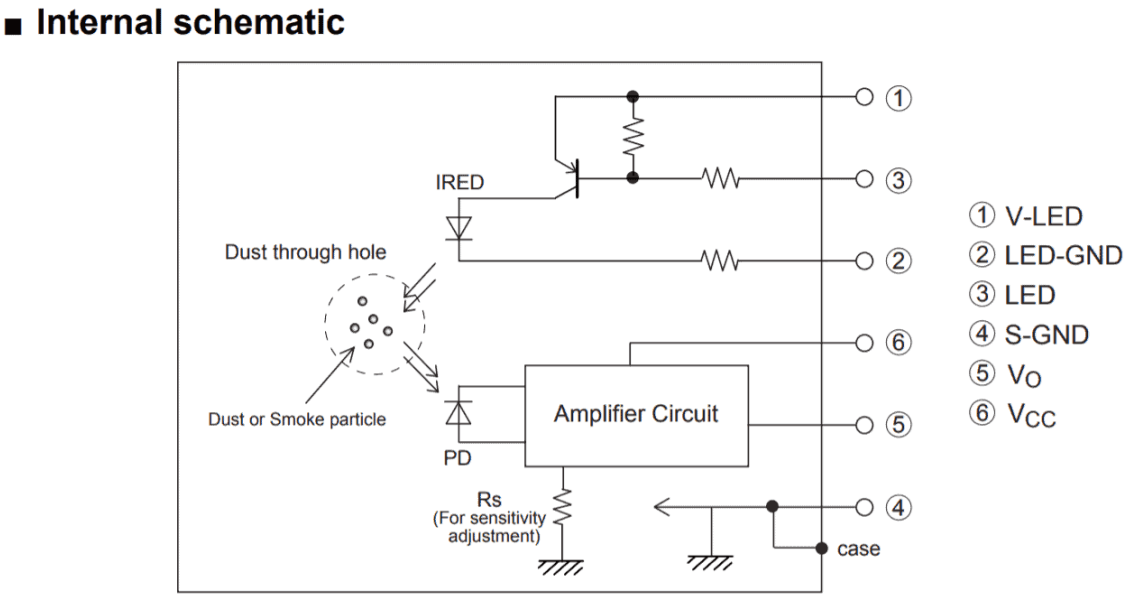
Figura 2 - Foto do sensor GP2Y1010AU0F da Sharp.



Fonte: Pieters (2018).

Este sensor consiste basicamente em um diodo emissor de infravermelho e um fototransistor, funcionando a partir de uma tensão de alimentação nominal entre $4.5\,V_{DC}$ e $5.5\,V_{DC}$. Normalmente opera com uma tensão de $5.0\,V_{DC}$ e requer uma corrente máxima de $20\,mA$. O esquema interno deste sensor pode ser observado na Figura 03, e a descrição de seus pinos na tabela 01.

Figura 3 - Esquema interno do sensor GP2Y1010AU0F da Sharp.



Fonte: EINSTRONIC ([2018?]).

Tabela 1 - Nome e descrição dos pinos do sensor GP2Y1010AU0F da Sharp.

Nº do pino	Nome do Pino	Descrição
1	$V-LED$	Pino V_{CC} do LED. É conectado a uma tensão de $5\,V$ através de um resistor de $150\,\Omega$.
2	$LED-GND$	Pino GND do LED. Conectado ao terra.
3	LED	Utilizado para ligar ou desligar o LED. Conectado a qualquer pino digital do ESP 32.
4	$S-GND$	Pino de aterramento do sensor. Conectado ao GND do ESP 32.
5	V_O	Pino de saída analógica do sensor. Conectado a qualquer pino analógico do ESP 32.
6	V_{CC}	Pino de alimentação. Conectado a uma tensão de

		alimentação de $5\ V$.

Fonte: Optical [...] (2020).

Este sensor possui um intervalo de detecção de densidade de poeira de 0 a $600\ \mu g/m^3$, conforme especificado na documentação fornecida pela Sharp.

2.3 Arduino e ESP

O Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software de simples entendimento e utilização. As placas Arduino são desenvolvidas para ler entradas, sejam elas dados de sensores ou mensagens, e convertê-las em uma determinada saída, como o acionamento de um motor, a ligação de um LED ou enviando uma mensagem na rede. Para isso, deve ser elaborado um conjunto de instruções em linguagem de programação Arduino no software Arduino (IDE), o qual é enviado para a placa (MAKIYAMA, 2022).

A família de microcontroladores ESP, criada e desenvolvida pela empresa chinesa Espressif Systems, foi estruturada de forma muito semelhante ao do Arduino. A principal diferença entre eles é que os ESP's possuem conexões wireless inerentes, enquanto que os microcontroladores Arduino necessitam de um aparato específico (Shield) para tal fim. Ademais, a família ESP é capaz interpretar linguagens de programação além de C++, como JavaScript, Python e C (TORELLA, 2020).

Com exceção a tais pontos, estas duas famílias de microcontroladores (ESP e Arduino) funcionam de maneira muito parecida, além de serem programadas através da mesma IDE (TORELLA, 2020).

2.4 ESP-32 DEV KIT V1

A placa utilizada foi o modelo DEV KIT V1, sendo uma evolução do modelo ESP8266, tendo uma maior capacidade de processamento e bluetooth BLE 4.2 embutido. A placa possui o chip ESP-32 com antena, interface usb-serial, um regulador de tensão de 3.3 V e 4 MB de memória flash, o que possibilita o desenvolvimento de diversos projetos de IoT, webservers e dataloggers, entre outros. Algumas de suas principais especificações técnicas estão apresentadas na tabela 02, e sua pinagem, na Figura 04.

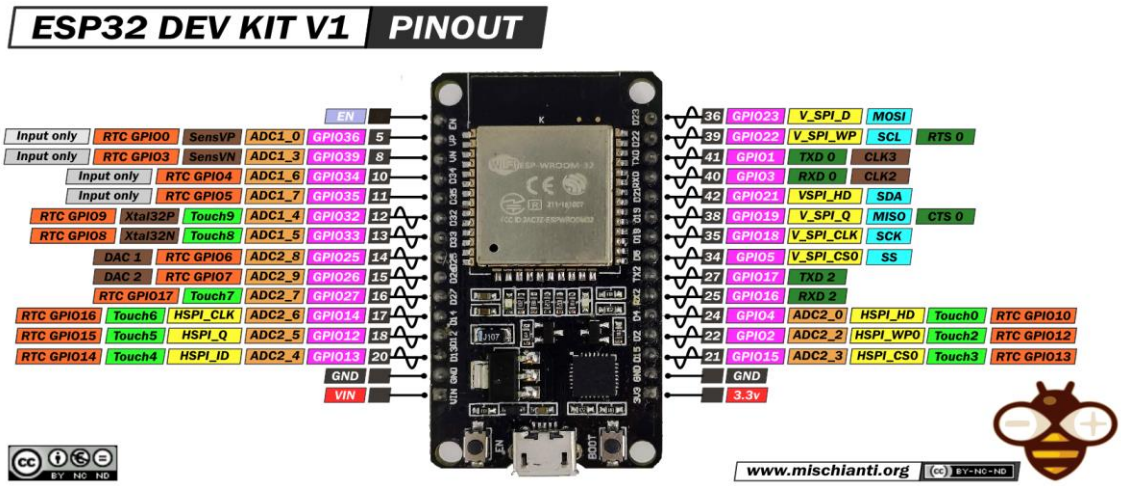
Tabela 2 - Especificações técnicas do ESP-32 modelo DEV KIT V1.

Qtde. de Pinos	30

Controlador	ESP-WROOM-32
CPU	Xtensa® Dual-Core 32-bit LX6
ROM	448 KBytes
RAM	520 KBytes
FLASH	4 MB
Clock Máximo	240 MHz
Atributos	Wireless padrão 802.11 b/g/n; conexão Wifi 2.4Ghz (máximo de 150 Mbps) e conversor analógico digital (ADC).
Conector	Micro-USB
Bluetooth	v4.2 BR / EDR e BLE (Bluetooth Low Energy)
Portas GPIO	25 com funções de PWM, I2C, SPI.
Tensão de Operação	$4.5\ V \sim 9\ V$
Corrente de Operação (Típica)	$80\ mA$

Fonte: Eletrogate ([202-?]).

Figura 4 - Pinagem da placa ESP-32 DEV KIT V1.

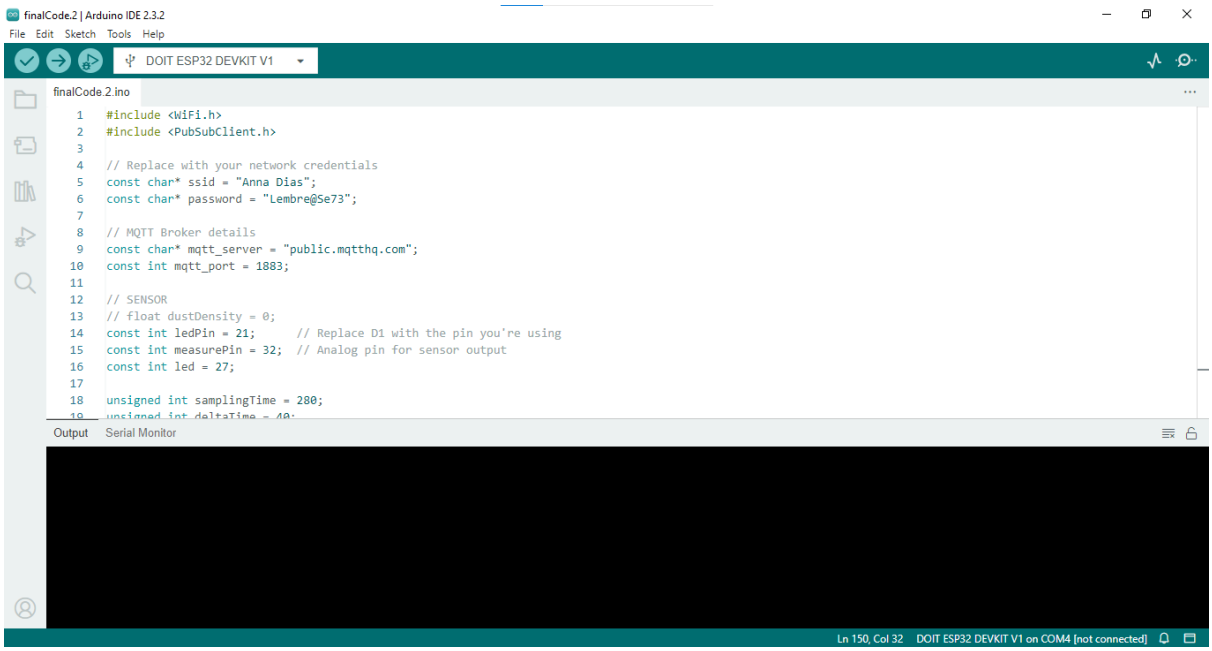


Fonte: Mischianti (2021).

2.5 Software do Arduino

A programação do ESP-32 pode ser feita através de um software obtido no site do Arduino, sendo uma IDE a qual permite a criação de sketches com uma linguagem própria baseada na linguagem C e C++. Esta IDE é gratuita, e sua interface é dividida em três partes principais: a toolbar no topo, a Sketch Window (onde é escrito o código) no centro e uma janela de mensagens na base, como pode ser observado na Figura 05.

Figura 5 - Interface da IDE do Arduino.



Fonte: Autoria própria.

A programação da placa pode ser feita utilizando os seguintes passos: conexão da placa a uma porta USB do computador; elaboração de um sketch contendo os comandos necessários para o funcionamento do projeto; realização do upload do sketch para a placa via comunicação USB; e aguardo da reinicialização, que ocorre após a execução do sketch.

Após o upload, não é mais necessário o uso do computador, salvo seja necessária a modificação do código, porém é necessário a conexão com uma fonte de alimentação.

A Toolbar da interface apresenta uma ou mais guias, cada qual contendo o nome do sketch. No topo há uma barra com diversos menus, contendo as opções de File, Edit, Sketch, Tools e Help. Há também alguns ícones de atalho: verify, que analisa se há ou não algum erro no código; upload, o qual compila e grava o código na placa; start debugging, que encontra e corrige erros (bugs) no código; serial plotter, o qual permite que informações variadas sejam apresentadas em forma de gráfico; e serial monitor, que abre o monitor serial.

Na primeira vez que o programa é executado é necessário que seja selecionada a placa utilizada (no menu Tools), bem como a COM que ela foi atribuída.

O sketch possui duas funções básicas iniciais, a setup e a loop. A função setup é executada na inicialização do programa, sendo responsável pelas configurações iniciais do microcontrolador, tais como a definição dos pinos de I/O, inicialização da comunicação serial, entre outras. Na função loop ocorre o laço infinito da programação, ou seja, é onde deve ser inserido o código que será executado continuamente pelo microcontrolador (SOUZA, 2013).

2.6 MQTT

2.6.1 Message Queuing Telemetry Transport

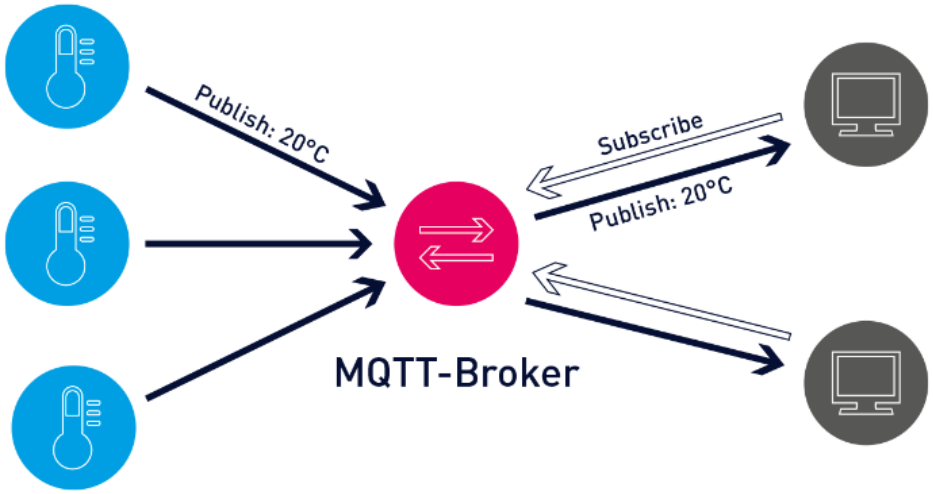
O Message Queuing Telemetry Transport (MQTT) é um protocolo de comunicação utilizado para otimizar a troca de informações entre dispositivos e sistemas distribuídos. Como o MQTT funciona permitindo que dispositivos enviem mensagens para que outros dispositivos possam monitorar determinada situação de maneira eficiente, este protocolo tornou-se uma das principais opções para a internet das coisas (PAESSLER, [201-?]).

2.6.2 Arquitetura MQTT

O MQTT possui dois tipos de sistemas em sua arquitetura: clients e brokers. Um broker é um servidor com o qual os clientes se comunicam. Ele recebe informações de clientes e as envia para outros clientes. Os clientes não se comunicam diretamente entre eles, e sim através do broker. Cada cliente pode ser um publisher, um subscriber, ou ambos.

Este protocolo é orientado a eventos. Isto significa que não transmissões de dados de forma contínua nem periódica. Um cliente apenas publica algo quando há uma informação a ser enviada, e um broker apenas envia dados a clientes quando há alguma mensagem a ser enviada (PAESSLER, [201-?]). A Figura 06 ilustra esta arquitetura.

Figura 6 - Ilustração representando a arquitetura do protocolo MQTT.



Fonte: Paessler ([201-?]).

2.6.3 Tópicos

As mensagens MQTT são publicadas como tópicos (topics). Estes são estruturas as quais formam uma hierarquia, indicadas pelo uso de barras (/). Uma estrutura como, por exemplo, sensors/OilandGas/Pressure/ permite que um assinante (subscriber) especifique que devem ser enviados apenas dados de clientes que publicam no tópico Pressure, ou, de forma mais ampla, todos os dados de clientes que publicam no tópico sensors/OilandGas

Os tópicos não são necessariamente criados no MQTT. Se um broker recebe um dado publicado em um tópico inexistente, este simplesmente é criado, e a partir de então os clientes podem se inscrever nele (PAESSLER, [201-?]).

2.6.4 Mensagens MQTT

Para que o protocolo seja conciso, são possíveis apenas quatro ações em qualquer comunicação: publish, subscribe, unsubscribe, ou ping.

A ação publish envia um conjunto de dados com uma determinada mensagem. Estes dados podem ser indicações de ligado ou desligado, valores de um determinado sensor, como temperatura ou pressão, entre outros. Conforme supracitado, caso o tópico em que tais dados foram publicados não exista, ele é automaticamente criado no broker.

A ação subscribe transforma um cliente em um assinante (subscriber) de um tópico. Na inscrição, um cliente envia um pacote SUBSCRIBE e recebe um pacote SUBACK. Caso haja uma mensagem retida no tópico, o novo inscrito também a recebe.

Na ação unsubscribe, uma mensagem ‘disconnect’ é enviada ao broker, informando-o que o cliente não irá mais receber mensagens nem deste broker, nem do publisher. Este procedimento permite que o cliente reconecte-se utilizando a mesma identidade.

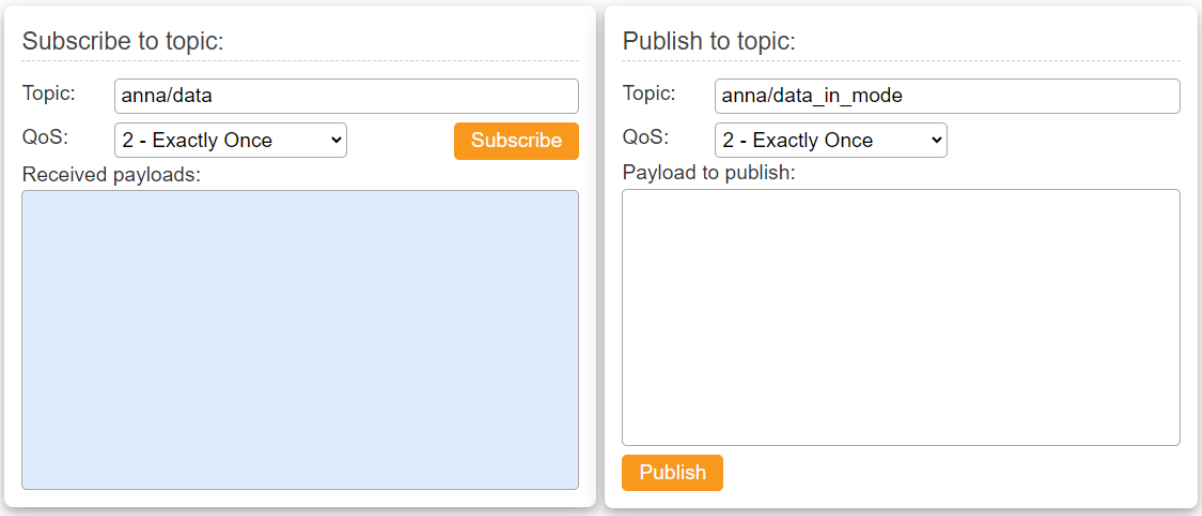
Por fim, na ação ping um pacote ‘PINGREQ’ é enviado pelo assinante (subscriber) e uma mensagem ‘PINGRESP’ é enviada como resposta. Este processo verifica se a conexão ainda está funcionando e que a conexão TCP não foi interrompida por outro equipamento (PAESSLER, [201-?]).

2.7 MQTTHQ

O MQTTHQ é um broker gratuito que permite o envio e recebimento de mensagens MQTT, sendo projetado para ser estável e suportar conexões TCP e WebSocket.

Acessando URL public.mqtthq.com do broker, é possível que seja feita a inscrição a um tópico para que sejam recebidos dados enviados por ele, e também publicações em outro tópico, para que sejam enviadas informações ao ESP-32 (MQTTHQ..., [2020?]). A Figura 07 apresenta a interface deste site.

Figura 7 - Interface do broker gratuito MQTTHQ para inscrição e publicação de mensagens MQTT.



Fonte: MQTTHQ [...] ([2020?]).

2.8 IoT MQTT Panel

O IoT MQTT Panel é um aplicativo que permite o gerenciamento e visualização de projetos IoT, baseados no protocolo MQTT. Com ele é possível visualizar e controlar facilmente várias entradas e saídas do aplicativo IoT, estando disponível para plataformas Android e iOS.

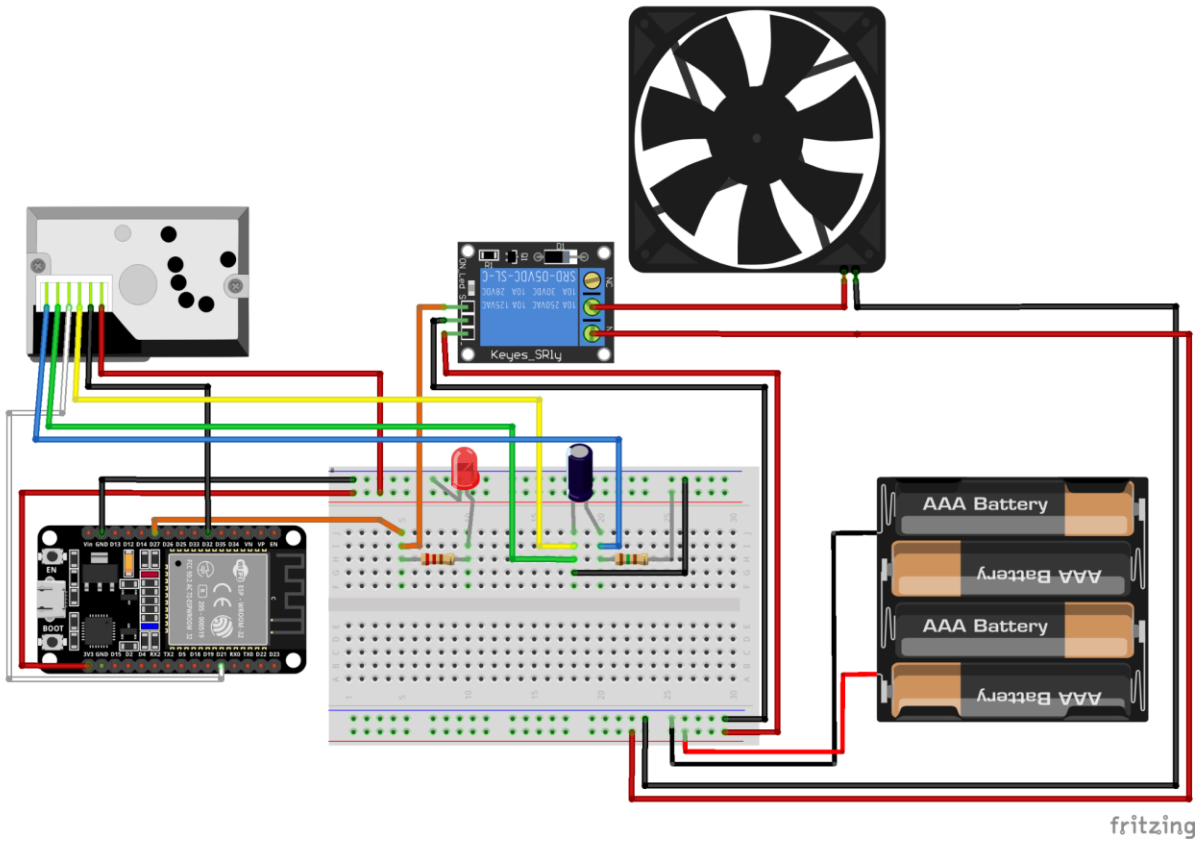
Ele oferece um grande número de painéis que podem atender a todos os tipos de casos de uso de uso geral, suportando SSL para comunicação segura. O recurso de caminho JSON permite várias leituras de sensores com um único *payload*. O compartilhamento das configurações com diversos dispositivos é simples, pois cada um dos paineis oferece várias opções para personalizá-los (KUNDU, 2017).

3 MATERIAIS E MÉTODOS

3.1 Hardware (Circuito)

O hardware do circuito elaborado pode ser observado na Figura 08, utilizando-se o software Fritzing, um hardware de código aberto para desenvolvimento de protótipos de circuitos eletrônicos (FRITZING, [201-]).

Figura 8 - Circuito elaborado no software Fritzing para o sistema automático do laboratório de microplásticos.



Fonte: Autoria própria.

O sensor GP2Y1010AU0F teve seu fio vermelho (V-LED) conectado a um lado do resistor e ao pino de tensão do ESP-32 ($3.3\,V$); o fio preto (LED-GND) conectado ao pino analógico 32; o fio amarelo (LED) e verde (V_O) ao catodo do capacitor, o branco (S-GND) ao pino 21 do ESP-32 e o azul ao ânodo do capacitor e ao outro lado do resistor. O catodo é ligado ao fio terra.

Para realizar a conexão entre o ESP-32 e o ventilador, foi utilizado um módulo relé, o qual fornece um valor de corrente maior ao motor do ventilador para que seja possível seu

acionamento, além de funcionar como uma chave, que fecha quando o número de partículas no ambiente torna-se maior que o configurado ($300\ \mu g/m^3$).

O pino ‘sinal’ do módulo relé foi conectado ao pino 27 do microcontrolador, sendo este mesmo pino 27 conectado a um resistor, que por sua vez foi ligado ao LED. Os pinos positivo e negativo do relé foram ligados a uma fonte de alimentação de $5\ V$ e seu comum ao terminal negativo da pilha do ventilador; o contato normalmente aberto (NA) foi ligado ao motor, e foi feita uma conexão através de um outro fio entre o motor e o terminal positivo da pilha do ventilador.

3.2 Software (Código)

Inicialmente foram importadas as bibliotecas ‘WiFi.h’ e ‘PubSubClient.h’, sendo a primeira utilizada para conexão com a rede wi-fi e a segunda para a comunicação MQTT. Foram criadas variáveis (‘ssid’ e ‘password’) as quais contém as credenciais da rede utilizada, e também variáveis com o endereço (‘mqtt_server’) e porta (‘mqtt_port’) do broker.

Foram definidas as variáveis utilizadas para o funcionamento do projeto. As variáveis ‘fanControl’ e ‘fan’ são utilizadas para controlar o ventilador e o LED; a primeira determina se o controle do sistema será automático (‘fanControl = 1’) ou manual (‘fanControl’ = 0), e a segunda se o LED deverá ser ligado (‘fan = 1’) ou desligado (‘fan = 0’).

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Anna Dias";
const char* password = "Lembre@Se73";

const char* mqtt_server = "public.mqtthq.com";
const int mqtt_port = 1883;

const int ledPin = 21;
const int measurePin = 32;
const int led = 27;

unsigned int samplingTime = 280;
unsigned int deltaTime = 40;
unsigned int sleepTime = 9680;

float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;
```

```
int fanControl = 1;
int fan = 0;
```

Foram criadas instâncias (‘WiFiClient’ e ‘PubSubClient’); ‘espClient’ é utilizado para a conexão com a rede e ‘client’ para tratar do protocolo MQTT. Também foram criados protótipos das funções as quais serão utilizadas posteriormente.

```
WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi();
void callback(char* topic, byte* message, unsigned int length);
void reconnect();
void sensor();
```

Na função setup foi inicializada a comunicação serial a uma taxa de transmissão de 115200 (‘Serial.begin(115200)’); é chamada a função para a conexão com o wi-fi (‘setup_wifi()’); ‘client.setServer(mqtt_server, mqtt_port)’ conFigura o endereço e a porta MQTT; e ‘client.setCallback(callback)’ conFigura a função a ser chamada quando uma mensagem chegar.

```
void setup() {
  Serial.begin(115200);

  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);  // Ensure LED is off
  pinMode(led, OUTPUT);
}
```

Na função loop a estrutura condicional verifica se o cliente está conectado com o broker e, caso não esteja, a função ‘reconnect’ é chamada; ‘client.loop()’ trata do recebimento e envio das mensagens MQTT; também é chamada a função ‘sensor’.

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  sensor();
}
```

A função ‘setup_wifi’ inicializa a conexão wi-fi (‘WiFi.begin(ssid, password)’); a estrutura de repetição ‘while’ aguarda até o ESP-32 conectar-se ao wi-fi; é impresso o endereço IP atribuído ao ESP-32.

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Na função ‘callback’ é impresso o tópico da mensagem recebida (‘Serial.print("Message arrived on topic: ")’), é criada uma variável temporária para armazenar a mensagem recebida; a primeira estrutura de repetição ‘for’ lê a mensagem byte

por byte e a armazena na variável ‘messageTemp’. A primeira estrutura condicional verifica se a mensagem está no tópico ‘anna/data_in_mode’ inscrito, e, caso a mensagem seja ‘manualControl’, é atribuído 0 à variável ‘fanControl’, caso seja ‘automaticControl’, é atribuído 1 à esta variável. A segunda estrutura condicional verifica se a mensagem está no tópico ‘anna/data_in_state’; se estiver, há outra estrutura condicional que analisa se a mensagem é ‘on’ e se a variável ‘fanControl’ é nula, e, caso essas condições sejam verdadeiras, é atribuído 1 à variável ‘fanControl’, e impressa a mensagem ‘Ventilador Ligado’; já no caso de serem falsas (e a mensagem for ‘off’), a variável ‘fanControl’ continua tendo o valor 0 e é impressa a mensagem ‘Ventilador Desligado’.

```
      void callback(char* topic, byte* message, unsigned int length)
{
        Serial.print("Message arrived on topic: ");
        Serial.print(topic);
        Serial.print(". Message: ");
        String messageTemp;

        for (unsigned int i = 0; i < length; i++) {
          Serial.print((char)message[i]);
          messageTemp += (char)message[i];
        }
        Serial.println();

        if (String(topic) == "anna/data_in_mode") {

          if (messageTemp == "manualControl") {
            fanControl = 0;
          }
          if (messageTemp == "automaticControl") {
            fanControl = 1;
          }
        }

        if (String(topic) == "anna/data_in_state") {

          if ((messageTemp == "on") && (fanControl == 0)) {
```

```
      Serial.println("Ventilador Ligado");
      fan = 1;
    } else if (messageTemp == "off") {
      Serial.println("Ventilador Desligado");
      fan = 0;
    }
  }
}
```

A função ‘reconnect’ foi elaborada para que o ESP-32 sempre esteja conectado ao broker. Ela possui um laço de repetição (‘while’) que constantemente tenta conectar-se a ele, e imprime uma mensagem alertando sobre as tentativas de conexão (‘Attempting MQTT connection…’). O laço também possui uma condicional que verifica se o broker está conectado ao ID ‘ESP32Client’, e, se estiver, é impresso ‘connected’ e o cliente é inscrito nos tópicos ‘anna/data_in_mode’ e ‘anna/data_in_state’; se não estiver, é impresso o motivo da falha de conexão (‘client.state()’), e um comando para que seja aguardado cinco segundos para a próxima tentativa de conexão.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32Client")) {
      Serial.println("connected");
      client.subscribe("anna/data_in_mode");
      client.subscribe("anna/data_in_state");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Por fim, a função ‘sensor’ inicialmente liga o LED interno do sensor GP2Y1010AU0F para que seja medida as partículas no ar (‘digitalWrite(ledPin, LOW);’). É esperado um

período para que a leitura seja estável. ‘voMeasured = analogRead(measurePin);’ lê a tensão analógica oferecida pelo sensor. É aguardado mais um período através da variável ‘deltaTime’ para a obtenção de um valor estável, e posteriormente o LED é desligado (‘digitalWrite(ledPin, HIGH);’). A variável ‘sleepTime’ tem a função de fazer com que o ESP-32 aguarde 9680 ms até a próxima medição.

```
void sensor() {
  digitalWrite(ledPin, LOW);
  delayMicroseconds(samplingTime);

  voMeasured = analogRead(measurePin);

  delayMicroseconds(deltaTime);
  digitalWrite(ledPin, HIGH);
  delayMicroseconds(sleepTime);
```

Posteriormente, o valor da tensão ADC medido pelo ESP-32 é transformado em um valor de tensão proporcional, e a partir desta, é calculada a densidade de partículas. Caso esta seja negativa devido a um pequeno desvio, é atribuído 0 a variável que guarda o valor da densidade de partículas por meio de uma estrutura condicional. Os resultados obtidos são impressos.

```
calcVoltage = voMeasured * (3.3 / 4095.0);
  dustDensity = (0.17*voMeasured) - 0.1;

  if (dustDensity < 0) {
    dustDensity = 0.00;
  }

  delay(1000);

  Serial.print("Sensor Value: ");
  Serial.print(voMeasured);
  Serial.print(" - Voltage: ");
  Serial.print(calcVoltage);
  Serial.print("V - Dust Density: ");
```

```
        Serial.print(dustDensity);
        Serial.println(" ug/m^3");
```

É feita uma análise através de outra condicional para o acionamento do ventilador e do LED: caso a densidade de partículas seja superior a $300\ \mu g/m^3$ e o modo do sistema esteja no automático (representado por 'fanControl' com valor 1), ou o sistema esteja no modo manual ('fanControl = 0') e tenha sido pressionado o botão para ligar ('fan = 1'), o LED e o ventilador são acionados; caso estas duas condições sejam falsas, o LED permanece apagado (ventilador desligado).

```
        if (((dustDensity > 300) && fanControl == 1) || (fanControl
== 0 && fan == 1)) {
          digitalWrite(led, HIGH);
        } else {
          digitalWrite(led, LOW);
        }

        client.publish("anna/data", String(dustDensity).c_str());
      }
```

3.3 IoT MQTT Panel

Foi criado um painel com o nome 'clean_lab' para o controle do laboratório (Figura 09). Nele, há um gauge denominado 'Dust Density' que recebe os dados (valor da densidade de partículas) publicados no tópico 'anna/data', variando de *0* até $600\ \mu g\ /\ m^3$, entre os valores *200* e *300* o gauge foi configurado para tornar-se amarelo, abaixo do primeiro, verde, e acima do segundo, vermelho.

Há um botão chamado 'Controle Automático' inscrito no tópico 'anna/data_in_mode', que envia a mensagem 'automaticControl' caso seja pressionado, e faz com que o sistema entre no modo automático. Outro botão foi colocado ('Controle Manual') que, ao ser pressionado, envia a mensagem 'manualControl' para o mesmo tópico, de forma que o sistema entra no modo manual quando ele é apertado. Foi acrescida uma caixa de texto que apresenta a mensagem que foi enviada a este tópico ('automaticControl' ou 'manualControl').

Um switch denominado 'Desligar / Ligar Ventilador' foi acrescido ao painel para que o ventilador seja ligado (o ícone torna-se verde), ou desligado (o ícone torna-se vermelho), caso o sistema esteja no modo manual.

Foi inserido um gráfico para apresentar os valores de densidade de partículas em função do tempo, de forma que o usuário possa fazer uma análise das partículas em diferentes períodos. Por exemplo, com ele é possível verificar se há um aumento na quantidade de microplásticos no ambiente quando algum cientista estiver realizando experimentos no laboratório.

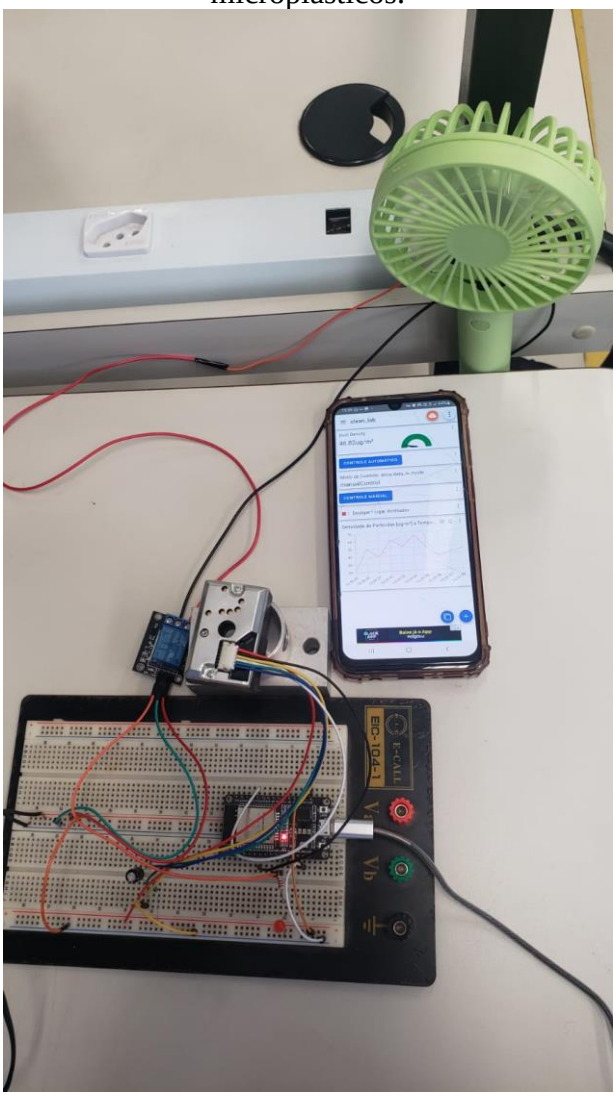
Figura 9 - Interface do painel de controle elaborado no IoT MQTT Panel.

Fonte: Autoria própria.

4 RESULTADOS E DISCUSSÃO

O circuito elaborado pode ser observado na Figura 10.

Figura 10 - Foto do circuito elaborado para o sistema automático do laboratório de microplásticos.



Fonte: Autoria própria.

Após realizadas todas as conexões, e feito o upload do código no Arduino, o monitor serial apresentou os seguintes resultados (Figura 11):

Figura 11 - Valores apresentados no serial monitor da IDE do Arduino quando o sensor encontrava-se em ambiente natural.

```
Sensor Value: 572.00 - Voltage: 0.46V - Dust Density: 97.14 ug/m^3
Sensor Value: 903.00 - Voltage: 0.73V - Dust Density: 153.41 ug/m^3
Sensor Value: 886.00 - Voltage: 0.71V - Dust Density: 150.52 ug/m^3
Sensor Value: 677.00 - Voltage: 0.55V - Dust Density: 114.99 ug/m^3
Sensor Value: 741.00 - Voltage: 0.60V - Dust Density: 125.87 ug/m^3
Sensor Value: 720.00 - Voltage: 0.58V - Dust Density: 122.30 ug/m^3
Sensor Value: 689.00 - Voltage: 0.56V - Dust Density: 117.03 ug/m^3
Sensor Value: 797.00 - Voltage: 0.64V - Dust Density: 135.39 ug/m^3
Sensor Value: 718.00 - Voltage: 0.58V - Dust Density: 121.96 ug/m^3
Sensor Value: 853.00 - Voltage: 0.69V - Dust Density: 144.91 ug/m^3
Sensor Value: 880.00 - Voltage: 0.71V - Dust Density: 149.50 ug/m^3
Sensor Value: 853.00 - Voltage: 0.69V - Dust Density: 144.91 ug/m^3
Sensor Value: 794.00 - Voltage: 0.64V - Dust Density: 134.88 ug/m^3
Sensor Value: 754.00 - Voltage: 0.61V - Dust Density: 128.08 ug/m^3
Sensor Value: 908.00 - Voltage: 0.73V - Dust Density: 154.26 ug/m^3
```

Fonte: Autoria própria.

Observa-se que os valores obtidos estão aproximadamente entre 100 e $150\ \mu g\ /\ m^3$. Para testar seu funcionamento, foi queimado um pedaço de papel próximo ao sensor, de forma que a fumaça e as partículas emitidas devido à combustão fossem detectadas por ele. Observou-se um aumento significativo nos valores da densidade de partículas no ar (Figura 12).

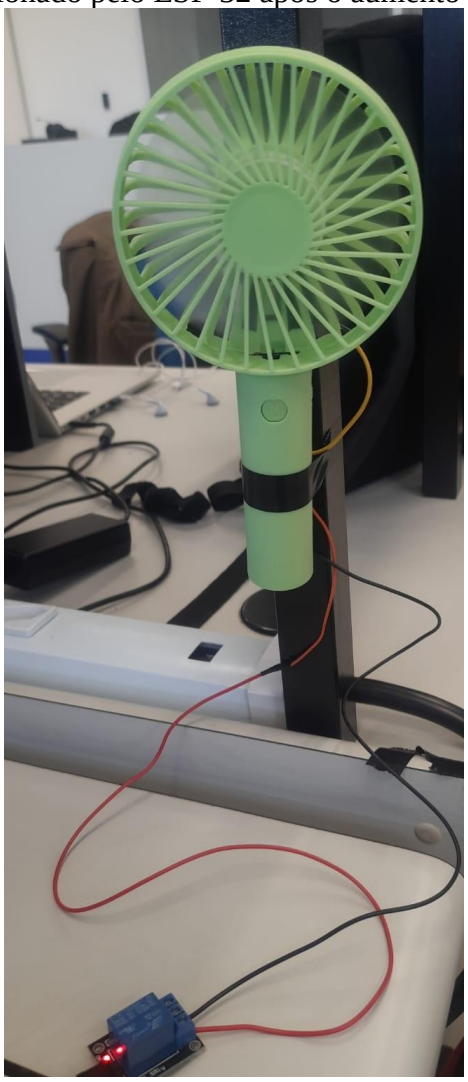
Figura 12 - Valores apresentados no serial monitor da IDE do Arduino quando o sensor encontrava-se em ambiente em que foi feita a combustão de um papel.

```
Sensor Value: 2512.00 - Voltage: 2.02V - Dust Density: 426.94 ug/m^3
Sensor Value: 2387.00 - Voltage: 1.92V - Dust Density: 405.69 ug/m^3
Sensor Value: 2226.00 - Voltage: 1.79V - Dust Density: 378.32 ug/m^3
Sensor Value: 2202.00 - Voltage: 1.77V - Dust Density: 374.24 ug/m^3
Sensor Value: 2416.00 - Voltage: 1.95V - Dust Density: 410.62 ug/m^3
Sensor Value: 2405.00 - Voltage: 1.94V - Dust Density: 408.75 ug/m^3
Sensor Value: 2421.00 - Voltage: 1.95V - Dust Density: 411.47 ug/m^3
Sensor Value: 2453.00 - Voltage: 1.98V - Dust Density: 416.91 ug/m^3
Sensor Value: 2222.00 - Voltage: 1.79V - Dust Density: 377.64 ug/m^3
Sensor Value: 2469.00 - Voltage: 1.99V - Dust Density: 419.63 ug/m^3
Sensor Value: 2401.00 - Voltage: 1.93V - Dust Density: 408.07 ug/m^3
Sensor Value: 2306.00 - Voltage: 1.86V - Dust Density: 391.92 ug/m^3
Sensor Value: 2283.00 - Voltage: 1.84V - Dust Density: 388.01 ug/m^3
Sensor Value: 2551.00 - Voltage: 2.06V - Dust Density: 433.57 ug/m^3
```

Fonte: Autoria própria.

Com o aumento da densidade de partículas (maiores que o valor de $300\ \mu g/m^3$), o ESP-32 acionou o ventilador para a limpeza do ar do ambiente, com o sistema configurado no modo automático no painel do IoT MQTT Panel, como pode ser observado na Figura 13.

Figura 13 - Ventilador acionado pelo ESP-32 após o aumento de partículas no ambiente.



Fonte: Autoria própria.

Foi testado o modo manual: quando o botão ‘Controle Manual’ foi pressionado, mesmo com os valores estando acima do limite estipulado, o ventilador desligou. Só foi ligado novamente quando foi pressionado o botão switch ‘Desligar / Ligar Filtro’.

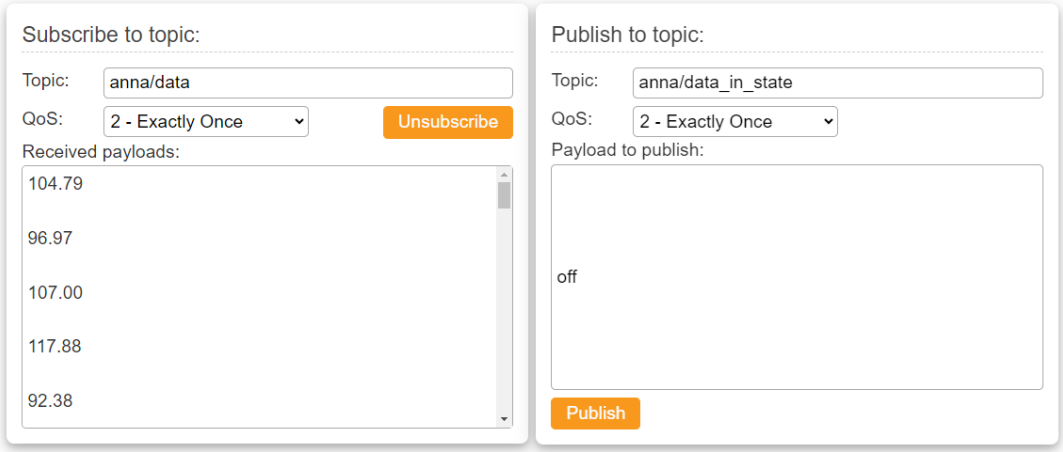
Também observou-se em tempo real o gráfico da densidade de partículas em função do tempo (Figura 14). Estes dados podem ser recolhidos para que sejam feitas análises dos momentos em que a quantidade de microplásticos no laboratório torna-se mais elevada; quando está sendo realizado algum experimento, por exemplo.

Figura 14 - Gráfico da densidade de partículas em µg/m^3 em função do tempo.

Fonte: Autoria própria.

Ademais, foi aberto o broker MQTTHQ para verificar o recebimento dos dados através dele. Como pode ser visto na Figura 15, este procedimento também foi bem-sucedido, pois os valores que apareciam no monitor serial e no aplicativo eram os mesmos recebidos pelo broker.

Figura 15 - Broker MQTTHQ recebendo e publicando dados nos tópicos anna/data e anna/data_in_state, respectivamente.



Fonte: Autoria própria.

5 CONCLUSÕES

A crescente necessidade de estudar os microplásticos devido aos problemas cada vez mais evidentes gerados por eles faz com que sejam necessários desenvolvimentos de novos sistemas e tecnologias que facilitem e aprimorem tais estudos. Com isto, o trabalho elaborado teve como finalidade criar um sistema automático para a medição da densidade de partículas e limpeza do ambiente durante o estudo de microplásticos, logo, conforme apresentado nos resultados, foi bem-sucedido, visto que quando o ESP-32 recebeu os dados detectados pelo sensor, o qual mediu uma maior quantidade de particulados no ar ao ser produzida fumaça por meio da combustão de um papel, ligou o ventilador, fazendo com que o laboratório fosse limpo.

Além disso, o envio de dados pelo protocolo de comunicação MQTT também foi bem-sucedido, pois os dados puderam ser visualizados tanto pelo broker online (MQTTHQ) quanto pelo painel desenvolvido no IoT MQTT Panel. Ainda, foi possível controlar o acionamento do ventilador quando pressionado o botão ‘Controle Manual’ do painel, de forma que o usuário teve a opção de escolha: que o sistema ligasse o ventilador automaticamente (sendo ligado após a quantidade de partículas atingir o valor estabelecido), ou ligasse a seu critério (pressionando o botão switch ‘Desligar / Ligar Ventilador’).

Com isto, espera-se que o presente trabalho estimule e seja uma referência para que outros sistemas sejam elaborados, para a obtenção de novas soluções para os problemas que os microplásticos têm acarretado.

REFERÊNCIAS

CANTÚ, E.; MONTEZ, C. B. Protocolos, Tecnologias, Ferramentas e Laboratórios para Aplicações de Internet das Coisas. In: CONGRESSO LATINO-AMERICANO DE SOFTWARE LIVRE E TECNOLOGIAS ABERTAS (LATINOWARE), 17., 2020, Online. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2020. p. 1-15.

ECOMARAPENDI. **Plástico:** história, composição, tipos, produção e reciclagem. [*S. l.*]: Recicloteca, 2020. Disponível em: https://www.recicloteca.org.br/material-reciclavel/plastico/. Acesso em: 05 jun. 2023.

EINSTRONIC. **GP2Y1010AU0F PM2.5 Dust Particle Sensor.** [*S. l.*], [2018?]. Disponível em: https://einstronic.com/product/gp2y1010au0f-pm2-5-dust-particle-sensor-module/. Acesso em: 24 maio 2024.

ELETROGATE. **Módulo WiFi ESP32 Bluetooth 30 pinos.** Belo Horizonte: Eletrogate, [202-?]. Disponível em: https://www.eletrogate.com/modulo-wifi-esp32-bluetooth-30-pinos. Acesso em: 27 maio 2024.

FRITZING. **[Página inicial].** [Berlin]: Fritzing, [201-]. Disponível em: https://fritzing.org/. Acesso em: 10 jun. 2024.

JATÍ, J. M. N. **Sistema para o cultivo de plantas utilizando princípios de Internet das Coisas.** 2023. 57 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Campus Manaus Distrito Industrial, Manaus, 2023.

KUNDU, R. **IOT MQTT Panel FAQ and feedback.** [*S. l.*]: SNR LAB, 15 ago. 2017. Disponível em: https://blog.snrlab.in/iot/iot-mqtt-panel-user-guide/. Acesso em: 13 jun. 2024.

MAKIYAMA, M. **O que é Arduino:** para que serve, benefícios e projetos [Exemplos]. [*S. l.*]: Victor Vision, 29 nov. 2022. Disponível em: https://victorvision.com.br/blog/o-que-e-arduino/. Acesso em: 27 maio 2024.

MARTINS, Victor Ferreira. **Automação residencial usando protocolo MQTT, Node-RED e Mosquitto Broker com ESP32 e ESP8266.** 2019. 53 p. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) – Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia, 2019.

MISCHIANTI, R. **DOIT ESP32 DEV KIT v1 high resolution pinout and specs.** [*S. l.*], 17 fev. 2021. Disponível em: https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/. Acesso em: 28 maio 2024.

MQTTHQ Web Client. [*S. l.*]: MQTTHQ, [2020?]. Disponível em: https://mqtthq.com/client. Acesso em: 5 jun. 2024.

OPTICAL Dust Sensor - GP2Y1010AU0F. [*S. l.*]: Components101, 23 dez. 2020. Disponível em: https://components101.com/sensors/optical-dust-sensor-pinout-features-applications-working-datasheet. Acesso em: 24 maio 2024.

PAESSLER. **What is MQTT? Definition and Details.** Nuremberg: Paessler, [201-?]. Disponível em: https://www.paessler.com/it-explained/mqtt. Acesso em: 1 jun. 2024.

PIETERS, A. **GP2Y1010AU0F - PM2.5 Optical Particle Sensor.** [*S. l.*]: Studio Pieters, 22 maio 2018. Disponível em: https://www.studiopieters.nl/gp2y1010au0f-pm2-5-optical-particle-sensor/#google_vignette. Acesso em: 24 maio 2024.

SARRIA-VILLA, R. A.; GALLO-CORREDOR, J. A. La gran problemática ambiental de los residuos plásticos: microplásticos. **Journal de Ciencia e Ingeniería**, [*S. l.*], v. 8, n. 1, p. 21–27, ago. 2016. Disponível em: https://jci.uniautonoma.edu.co/2016/2016-3.pdf. Acesso em: 24 maio 2024.

SOUZA, F. **Introdução ao Arduino:** primeiros passos na plataforma. [*S. l.*]: Embarcados, 06 nov. 2013. Disponível em: https://embarcados.com.br/arduino-primeiros-passos/#Software. Acesso em: 1 jun. 2024.

TEIXEIRA FILHO, Rubens Rodrigues. **Internet das coisas aplicada em sistemas eletropneumáticos no controle de atuadores utilizando os protocolos ZigBee e MQTT.** 2022. 72 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Mecatrônica) - Universidade Federal de Uberlândia, Uberlândia, 2022.

TORELLA, M. **Arduino ou ESP:** Descubra a melhor opção! [*S. l.*]: Lobo da Robótica, [201-?]. Disponível em: https://lobodarobotica.com/blog/arduino-ou-esp-descubra-a-melhor-opcao/. Acesso em: 27 maio 2024.

APÊNDICE A - CÓDIGO DO ESP-32

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Anna Dias";
const char* password = "Lembre@Se73";

const char* mqtt_server = "public.mqtthq.com";
const int mqtt_port = 1883;

const int ledPin = 21;
const int measurePin = 32;
const int led = 27;

unsigned int samplingTime = 280;
unsigned int deltaTime = 40;
unsigned int sleepTime = 9680;

float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;

int fanControl = 1;  // 1 -> automatic control
int fan = 0;

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi();
void callback(char* topic, byte* message, unsigned int length);
void reconnect();
void sensor();

void setup() {
  Serial.begin(115200);

  setup_wifi();
```

```
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
  pinMode(led, OUTPUT);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  sensor();
}

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* message, unsigned int length) {
```

```
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (unsigned int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();

  // Handle message
  if (String(topic) == "anna/data_in_mode") {

    if (messageTemp == "manualControl") {
      fanControl = 0;
    }

    if (messageTemp == "automaticControl") {
      fanControl = 1;
    }
  }

  if (String(topic) == "anna/data_in_state") {

    if ((messageTemp == "on") && (fanControl == 0)) {
      Serial.println("Ventilador Ligado");
      fan = 1;
    } else if (messageTemp == "off") {
      Serial.println("Ventilador Desligado");
      fan = 0;
    }
  }
}

void reconnect() {
  while (!client.connected()) {
```

```
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32Client")) {
      Serial.println("connected");
      // Subscribe to a topic
      client.subscribe("anna/data_in_mode");
      client.subscribe("anna/data_in_state");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void sensor() {
  digitalWrite(ledPin, LOW);
  delayMicroseconds(samplingTime);

  voMeasured = analogRead(measurePin);

  delayMicroseconds(deltaTime);
  digitalWrite(ledPin, HIGH);
  delayMicroseconds(sleepTime);

  calcVoltage = voMeasured * (3.3 / 4095.0);
  dustDensity = (0.17*voMeasured) - 0.1;

  if (dustDensity < 0) {
    dustDensity = 0.00;
  }

  delay(1000);

  Serial.print("Sensor Value: ");
  Serial.print(voMeasured);
```

```
  Serial.print(" - Voltage: ");
  Serial.print(calcVoltage);
  Serial.print("V - Dust Density: ");
  Serial.print(dustDensity);
  Serial.println(" ug/m^3");

  if (((dustDensity > 300) && fanControl == 1) || (fanControl == 0
&& fan == 1)) {
    digitalWrite(led, HIGH);
  } else {
    digitalWrite(led, LOW);
  }

  client.publish("anna/data", String(dustDensity).c_str());
}
```