UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO DE BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

MÁRCIO PEREIRA DA SILVA

SISTEMA DE IRRIGAÇÃO AUTOMATIZADO CONTROLADO POR APLICATIVO

PAU DOS FERROS
2021

MÁRCIO PEREIRA DA SILVA

SISTEMA DE IRRIGAÇÃO AUTOMATIZADO CONTROLADO POR APLICATIVO

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência e Tecnologia.

Orientador: Pedro Thiago Valério de Souza, Prof. Dr.

PAU DOS FERROS

2021

S586s Silva, Márcio Pereira da.
Sistema de irrigação automatizado controlado por aplicativo / Márcio Pereira da Silva. - 2021.
43 f. : il.

Orientador: Pedro Thiado Valério de Souza.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência e Tecnologia, 2021.

1. Automação. 2. Arduino. 3. ESP-01S. 4. Aplicação Móvel. I. Souza, Pedro Thiado Valério de, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 1115

MÁRCIO PEREIRA DA SILVA

SISTEMA DE IRRIGAÇÃO AUTOMATIZADO CONTROLADO POR APLICATIVO

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência e Tecnologia.

Defendida em: _____ / _____ / 2 _____.

BANCA EXAMINADORA

PEDRO THIAGO VALERIO DE SOUZA:08294430455 Digitally signed by PEDRO THIAGO VALERIO DE SOUZA:08294430455 Date: 2021.11.27 13:04:16 -03'00'

Pedro Thiago Valério de Souza, Prof. Dr. (UFERSA)
Presidente

FRANCISCO CARLOS GURGEL DA SILVA SEGUNDO:06062106444 Dados: 2021.11.27 11:19:21 -03'00'

Francisco Carlos Gurgel da Silva Segundo, Prof. Dr. (UFERSA)
Membro Examinador

VINICIUS SAMUEL VALERIO DE SOUZA:05083241498 Assinado de forma digital por VINICIUS SAMUEL VALERIO DE SOUZA:05083241498 Dados: 2021.11.27 11:18:04 -03'00'

Vinícius Samuel Valério de Souza, Prof. Dr. (UFERSA)
Membro Examinador

Dedico este trabalho aos meus pais que sempre estiveram ao meu lado e lutaram comigo para que a minha graduação fosse possível.

AGRADECIMENTOS

Agradeço a Deus por me proporcionar a oportunidade de realizar este sonho de cursar o curso que desejei, e por me dar forças e confiança para seguir até a conclusão do mesmo.

Aos meus pais Joana Darc e Cosme Pereira por sempre estarem ao meu lado me dando forças, acreditando que meu sonho poderia ser realizado e por dedicarem esforços para que ele se tornasse realidade.

Ao meu orientador, professor doutor Pedro Thiago, por ter aceito este desafio, por toda a sua disponibilidade para atender às minhas solicitações e me orientar neste trabalho de conclusão de curso.

Aos meus melhores amigos, Alison Cavalcante e Deise Silveira, por sempre estarem ao meu lado, confiarem no meu potencial e me apoiarem ao longo do desenvolvimento deste trabalho. Vocês são demais.

A Banca Examinadora, pelo interesse e disponibilidade.

E a todos que direta ou indiretamente, contribuíram para que este trabalho fosse realizado.

RESUMO

Este trabalho foi idealizado a partir da observação de um sistema de irrigação manual. Observou-se que o sistema, com controle manual, apresenta diversas limitações e proporciona a ocorrência de alguns problemas, como a queima da bomba d'água caso a mesma seja ligada e não haja água no reservatório. O objetivo geral deste trabalho é projetar a automação do sistema de irrigação, com o propósito de eliminar limitações e solucionar problemas encontrados pelo controle manual. Além de permitir ao usuário controlar o sistema de forma local ou remota por meio de uma aplicação móvel. Para que tal automação fosse possível, foram utilizados dispositivos como Arduino, ESP-01S, Módulos Relés, Válvulas Solenóides e Chaves Boia. Ao fim do trabalho, foi possível observar uma maior autonomia do usuário no controle do sistema, melhor administração do funcionamento do sistema como um todo, redução dos riscos de queima de dispositivos e redução de trabalhos repetitivos.

Palavras-chave: Automação; Arduino; ESP-01S; Aplicação Móvel.

ABSTRACT

This work was conceived from the observation of a manual irrigation system. It was observed that the system, with manual control, has several limitations and causes the occurrence of some problems, such as the burning of the water pump if it is turned on and there is no water in the reservoir. The general objective of this work is to design the automation of the irrigation system, with the purpose of eliminating limitations and solving problems encountered by manual control. In addition to allowing the user to control the system locally or remotely through a mobile application. For this automation to be possible, devices such as Arduino, ESP-01S, Relay Modules, Solenoid Valves and Float Switches were used. At the end of the work, it was possible to observe greater user autonomy in controlling the system, better administration of the system's operation as a whole, reduced risk of burning devices and reduced repetitive work.

Keywords: Automation; Arduino; ESP-01S; Mobile Application.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

USB	Universal Serial Bus
IDE	Integrated Development Environment
GPIO	General Purpose Input/Output
VIN	Tensão de Entrada
VOUT	Tensão de Saída
JSON	JavaScript Object Notation
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
APP	Aplicativo

SUMÁRIO

1 INTRODUÇÃO

Com o advento da automação, o homem vem desenvolvendo técnicas e dispositivos que visam uma maior e melhor produção através de mecanismos e processos automatizados que substituem, de forma parcial ou integral, as atividades manuais exercidas por seres humanos. A revolução industrial, iniciada na Inglaterra na segunda metade do século XVIII, foi um grande marco no desenvolvimento tecnológico. Ela proporcionou uma aceleração no processo de produção de mercadorias através da substituição da produção manual por máquinas.

Lamb (2015) define Automação como o uso de comandos lógicos programáveis e de equipamentos mecanizados para substituir as atividades manuais que envolvem tomadas de decisão e comandos-resposta de seres humanos. É notável a importância da automação para a otimização de processos manuais visando o ganho de produtividade e evitando problemas ocasionados pela atividade manual. A automação de sistemas é uma ferramenta poderosa para redução do esforço humano e é bastante aplicada nas diversas áreas de produção industrial.

O presente projeto foi idealizado após observações em um sistema de irrigação controlado de forma manual. O mesmo é composto de uma bomba d'água responsável por levar água de uma cisterna a um reservatório que abastece a residência, ou levar água até os canteiros de verduras. Ou seja, a bomba d'água só pode ser ligada para um único propósito por vez, abastecer a caixa d'água ou irrigar os canteiros. Este controle é feito de forma manual através de um registro esfera e necessita de uma pessoa para fazê-lo.

Dentre os problemas e limitações que o controle manual do sistema apresenta, podemos destacar:

- Danificação da bomba d'água, caso a mesma seja ligada e não haja água na cisterna;
- Necessidade de uma pessoa para abrir ou fechar o registro esfera que irá direcionar a água para os canteiros ou reservatório;
- Desperdício de água, caso a bomba seja ligada e não haja um monitoramento do nível d'água do reservatório ou caso a irrigação dos canteiros fique ligada por um tempo superior ao necessário.
- Abertura/fechamento manual do registro esfera, processo repetitivo.

- Necessidade de uma pessoa no local para ligar/desligar a bomba d'água e controlar o registro esfera.
- Risco dos canteiros não serem irrigados ou do reservatório ficar vazio, caso a pessoa responsável esqueça de ligar a bomba d'água.

Neste contexto, o presente trabalho tem como objetivo geral a projeção da automação do sistema em questão, permitindo que o usuário tenha controle de forma local ou remota do sistema, através de um aplicativo móvel. Assim como também consiga agendar a execução de comandos, como ligar ou desligar a bomba em determinado horário. Com este propósito, no decorrer do trabalho serão apresentados dispositivos e soluções para o desenvolvimento do projeto de automação.

Como objetivos específicos, destacam-se:

- Desenvolvimento de um circuito, utilizando Arduino e ESP-01S e outros componentes;
- Desenvolvimento de um servidor de comandos, que cuidará de toda a lógica de execução de comandos;
- Desenvolvimento de uma aplicação móvel, que proporcionará ao usuário uma melhor administração do sistema.

Deste modo, no Capítulo 2 é apresentada a arquitetura do projeto e as etapas de desenvolvimento do trabalho.

No Capítulo 3 são apresentados os passos de desenvolvimento do projeto, os componentes que compõem o circuito da automação e o motivo da utilização de cada um deles. Assim como também, é explicada a lógica por trás do servidor de comandos.

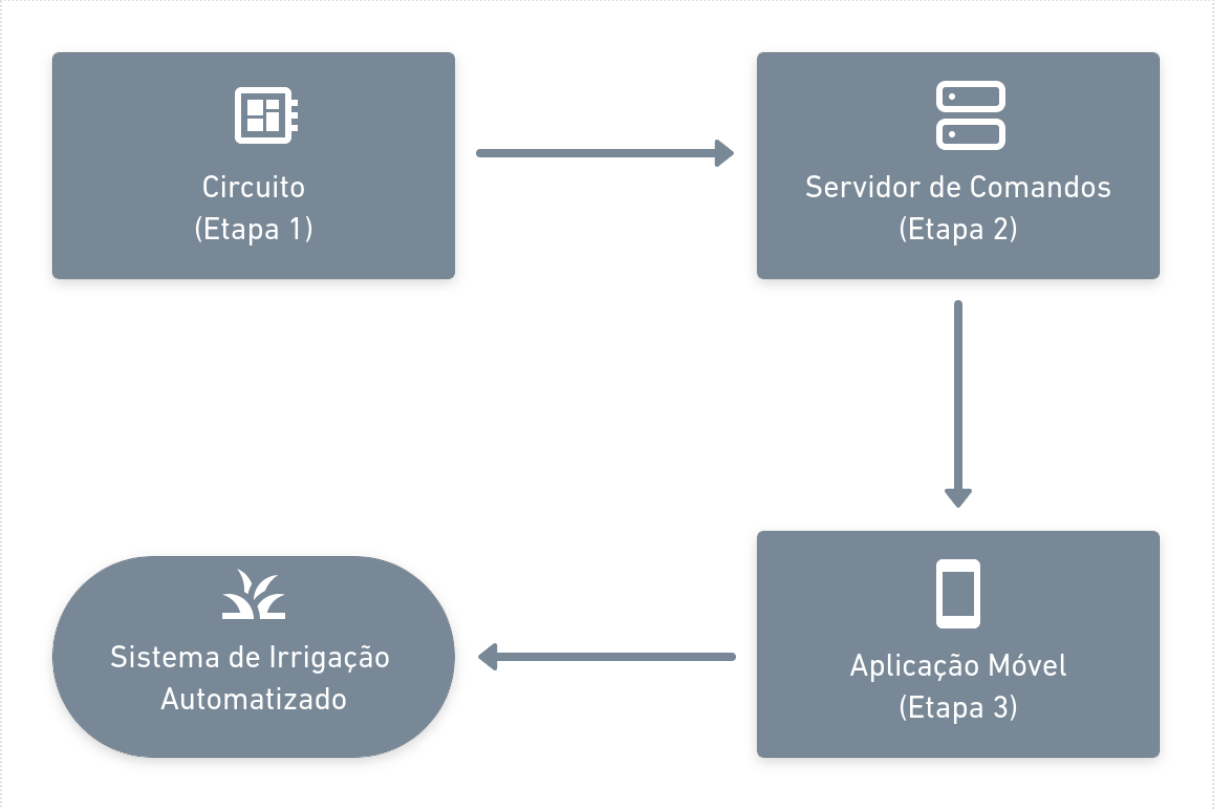
O Capítulo 4 destina-se a resultados e discussões, e apresentação da aplicação móvel desenvolvida.

E por último, no Capítulo 5, são consolidadas as considerações finais.

2 ARQUITETURA DO PROJETO

Este Capítulo tem como objetivo apresentar a estrutura do projeto e descrever de forma resumida cada etapa e sua respectiva responsabilidade. Sendo assim, para um melhor entendimento, o desenvolvimento do projeto foi dividido em três etapas principais: Circuito, Servidor de Comandos e Aplicação Móvel. Como é esquematizado na Figura 1. As etapas são descritas abaixo.

Figura 1 - Esquema das etapas do projeto



Fonte: Autor (2021).

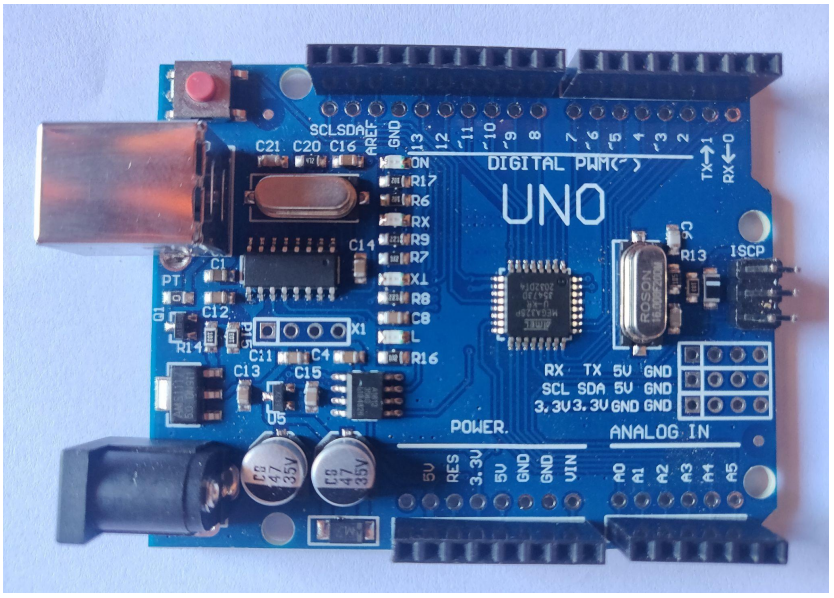
2.1 CIRCUITO

Na primeira etapa do projeto será desenvolvido o circuito, “cérebro” da aplicação. O mesmo será desenvolvido utilizando componentes eletrônicos e o microcontrolador Arduino. Assim como também, Módulos ou Shields que possibilitam complementar as funcionalidades do Arduino.

Monk (2017) define o Arduino como uma pequena placa microcontroladora que possui um plugue de conexão USB (Universal Serial Bus), que permite a ligação com um computador, além de contar com um conjunto de pinos de conexão, que torna possível a ligação de dispositivos eletrônicos externos, como motores, relés, sensores e outros. Os Arduinos podem ser energizados pelo computador através de um cabo USB, por uma bateria de 9V ou por uma fonte de alimentação. Eles são programados inicialmente pelo computador, através de um software denominado Arduino IDE e sem seguida podem ser desconectados para trabalharem independentemente do computador .

O Arduino é um projeto de código aberto, denominado “open source". Isto possibilita que qualquer pessoa possa utilizar o seu código original e desenvolver a sua própria placa modificada. Com isso, há vários modelos de placas Arduino disponíveis no mercado. Para este projeto foi utilizada a placa microcontroladora Arduino UNO, ilustrada na Figura 2, uma placa de baixo custo que nos possibilita desenvolver diversos tipos de automações. Este microcontrolador, em conjunto com outros componentes, tem a capacidade de ligar ou desligar dispositivos, desde um simples led a uma bomba d’água. O Arduino pode ser programado através de um software, chamado Arduino IDE (do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado), que utiliza a linguagem de programação C.

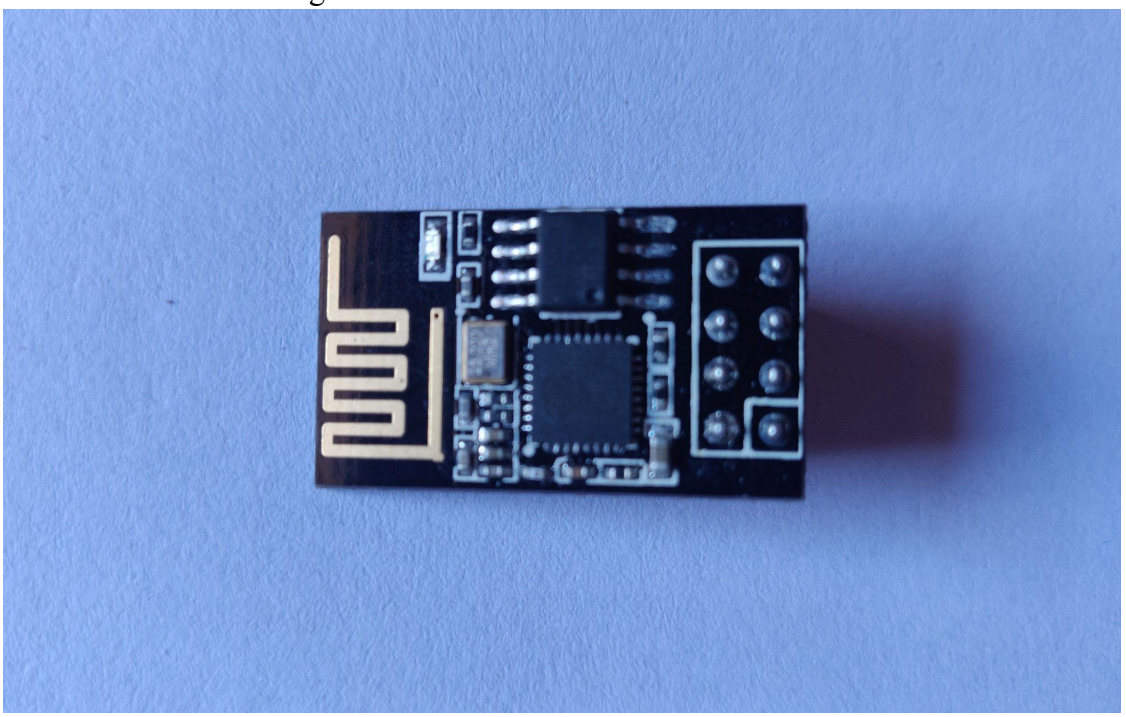
Figura 2 - Arduino UNO



Fonte: Autor (2021).

A placa Arduino UNO não possui conexão com a internet, por esta razão deve-se utilizar um Módulo ou Shield que possibilite a conexão à internet. Módulos ou Shields são componentes que tem como objetivo complementar as funcionalidades do Arduino. Para que seja possível conectar o Arduino UNO a internet deve-se utilizar um Shield Ethernet ou um Módulo WiFi. O Shield Ethernet possibilitará a conexão cabeada e o Módulo WiFi, a conexão sem fio. Visando uma melhor mobilidade do circuito, será utilizado o Módulo WiFi ESP-01S, ilustrado na Figura 3. Este módulo contém o chip ESP8266.

Figura 3 - Módulo WiFi ESP8266 ESP-01S



Fonte: Autor (2021).

Para Monk (2017) o ESP8266 é um sistema WiFi com microcontrolador contido em um chip. Um único chip que faz praticamente tudo que um Arduino Uno faz, incluindo a capacidade de se conectar à internet via WiFi. Ele contém alguns pinos GPIO[1] e uma entrada analógica que pode ser programada a partir do IDE do Arduino, como se fosse uma placa oficial de Arduino.

O circuito que será desenvolvido na primeira etapa do projeto, ficará responsável por receber e executar todos os comandos vindos do servidor de comandos. No próximo capítulo

[1] General Purpose Input/Output (GPIO) são portas de entrada e saída de dados que podem ser programadas com o objetivo de possibilitar uma interface entre periféricos e os microcontroladores.

será abordado e descrito em detalhes o desenvolvimento de tal circuito, assim como também a forma de conexão entre o Arduino UNO, o ESP-01S, e os demais componentes, elencando a responsabilidade de cada componente do circuito.

2.2 SERVIDOR DE COMANDOS

Mas adiante será visto que é possível configurar um servidor Web no ESP-01S e possibilitar a consulta e controle os estados dos pinos do arduino via internet, ou seja, somente com o Arduino e o ESP-01S, e sem a necessidade de um servidor de comandos, já seria possível controlar todo o sistema através de uma aplicação móvel. Porém, para prover uma maior segurança ao sistema e um maior controle, será implementado um servidor de comandos que intermediará a aplicação móvel e o circuito. Este servidor é uma máquina com sistema operacional Linux e com um back-end desenvolvido na linguagem de programação PHP juntamente com o framework Laravel. O mesmo terá como principais atribuições:

- Adicionar uma camada de segurança ao sistema;
- Consultar o estado dos pinos do arduino em tempo real;
- Permitir que o usuário consiga atribuir nomes aos pinos do arduino, facilitando o entendimento do funcionamento do sistema como um todo;
- Receber, analisar e enviar para o arduino os comandos advindos da aplicação móvel;
- Armazenar em um banco de dados o histórico de comandos executados, a fim de gerar relatórios;
- Prover a possibilidade de agendar a execução de comandos;
- Controlar o fluxo de execução de comandos, em outras palavras, não permitir a execução de um comando que já está em execução.

2.3 APLICAÇÃO MÓVEL

Na terceira etapa do projeto, será desenvolvida uma aplicação móvel. Esta proverá uma melhor experiência de utilização do sistema para o usuário. E terá como responsabilidades:

- Enviar e coletar informações do sistema de comandos;
- Mostrar em tempo real os comandos que estão sendo executados e possibilitar o cancelamento destas execuções;

- Permitir que o usuário consulte, cadastre, edite e remova dispositivos.
- Permitir que o usuário consulte, cadastre, edite, remova e execute comandos;
- Permitir que o usuário consulte, cadastre e cancele agendamentos;
- Permitir que o usuário consulte um relatório de execução de comandos;

3 DESENVOLVIMENTO

Neste Capítulo serão apresentados os passos de desenvolvimento do circuito do projeto e a sua lógica de funcionamento, assim como também a lógica de funcionamento por trás do servidor de comandos. Além disso, serão mencionadas as linguagens de programação utilizadas, os componentes utilizados na confecção do circuito e o motivo de utilizá-los.

3.1 CIRCUITO

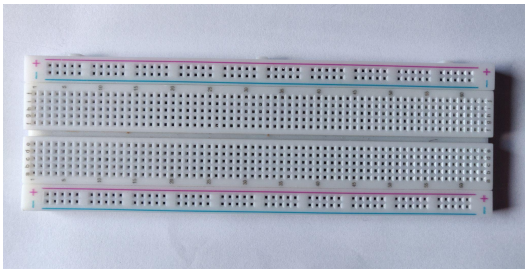
O projeto é composto por três etapas principais: Circuito, Servidor de Comandos e Aplicação Móvel. Na primeira etapa será desenvolvido o circuito do projeto, que terá como principais componentes o Arduino e o ESP-01S. Abaixo são apresentados os materiais e componentes utilizados para a confecção do circuito.

3.1.1 Materiais Utilizados

Para a confecção do circuito serão necessários os seguintes componentes:

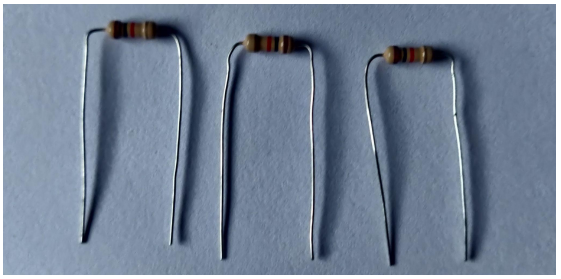
- Uma placa Arduino Uno (Figura 2);
- Um módulo WiFi ESP8266 ESP-01S (Figura 3);
- Uma Protoboard (Figura 4);
- Três resistores de 1kΩ (Figura 5);
- Um módulo Relé de 8 canais (Figura 6);
- Uma chave bóia (Figura 7);
- Alguns jumpers (Figura 8);

Figura 4 - Protoboard



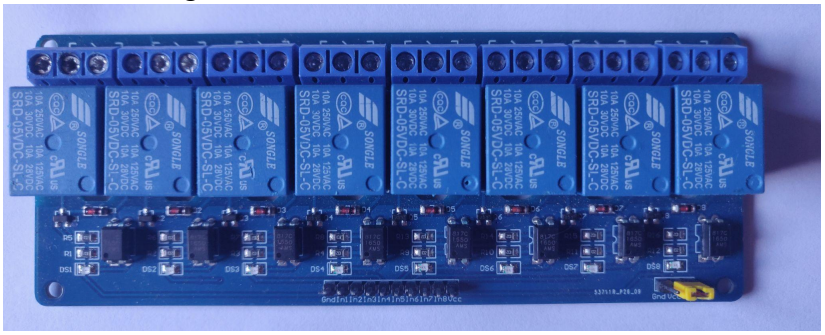
Fonte: Autor (2021).

Figura 5 - Resistores de 1kΩ



Fonte: Autor (2021)

Figura 6 - Módulo de Relés de 8 Canais

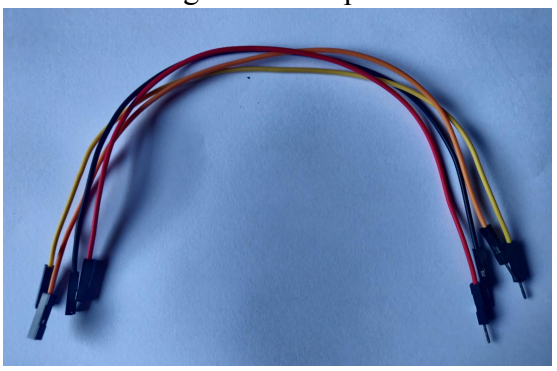


Fonte: Autor (2021).

Figura 7 - Chave bóia

Fonte: Autor (2021).

Figura 8 - Jumpers



Fonte: Autor (2021)

3.1.2 Montagem

A placa Arduino UNO e o ESP-01S serão os componentes principais do circuito. O ESP-01S será responsável por possibilitar a conexão à internet do arduino. Os dados irão trafegar entre os dois componentes através da comunicação serial. Nesta comunicação os dados são enviados em série, bit a bit, de forma sequencial num canal de comunicação ou barramento. Para estabelecer essa comunicação entres os dois componentes, será necessário conectar as portas TX e RX do Arduino às portas RX e TX do ESP-01S, respectivamente. Conectar a porta TX do ESP-01S diretamente à porta RX do Arduino não trará problema ao funcionamento do circuito, porém conectar a porta TX do Arduino diretamente à porta do ESP-01S poderá acarretar problemas, pois as portas do Arduino trabalham em uma tensão de 5V e o ESP-01S trabalha em 3,3V. Para resolver este problema, pode ser utilizado um conversor de nível lógico 3,3-5V bidirecional ou um divisor de tensão constituído de dois resistores em série, o primeiro de 1,2kΩ e o segundo de 2,2kΩ. O cálculo do divisor de tensão é apresentado abaixo:

$Vout = Vin\frac{R2}{(R1+R2)}$, onde:

Vin é a tensão de entrada

$Vout$ é a tensão de saída

$R1$ e $R2$ são os resistores.

$$Vout = 5v\frac{2{,}2k\Omega}{(1{,}2k\Omega+2{,}2k\Omega)} \simeq 3{,}24V$$

Devido a falta de resistores com as especificações mostradas anteriormente, serão utilizados 3 resistores de 1kΩ. O primeiro resistor ocupará o lugar do resistor de 1,2kΩ e os outros dois ficarão no lugar do resistor de 2,2kΩ. Desta forma, teremos o seguinte cálculo para o divisor de tensão:

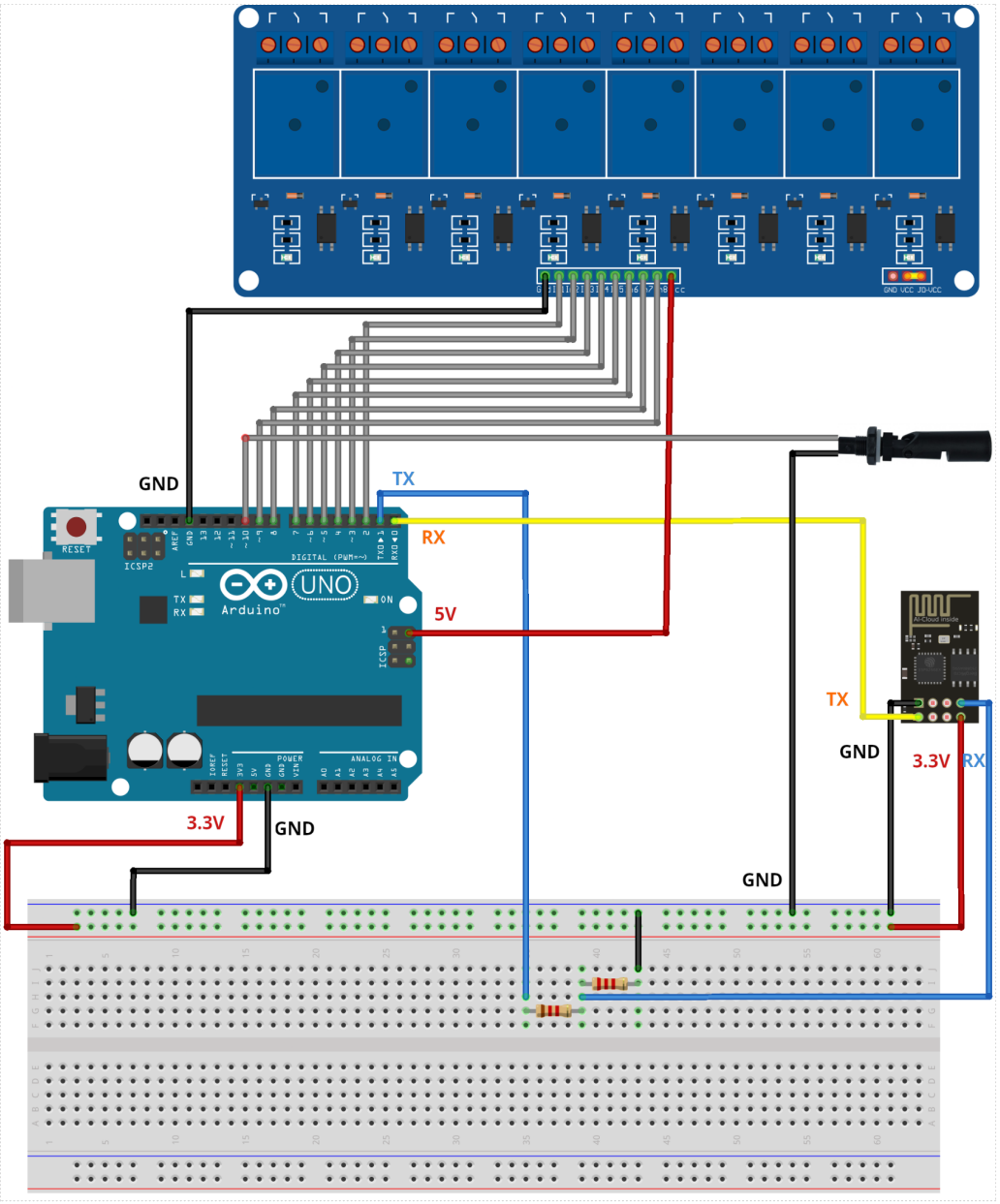
$$Vout = 5v\frac{(1k\Omega + 1k\Omega)}{[1k\Omega+(1k\Omega + 1k\Omega)]} = 5v\frac{2k\Omega}{3k\Omega} \simeq 3{,}33V$$

Feito isso, para conectar o Arduino e o ESP-01S é bem simples, basta conectar a porta TX do ESP-01S diretamente a porta RX do Arduino, em seguida conectar a porta TX do Arduino ao divisor de tensão e conectar a porta RX do ESP-01S entre os resistores que compõem o divisor, como mostra a figura 9. Além disso, será necessário alimentar o ESP-01S conectando-o a uma tensão de 3,3V e ao terra do circuito. Para conectar o módulo relé ao Arduino, é necessário conectá-lo a uma porta de 5V e ao terra. O arduino possui pinos digitais numerados de 2 a 13, por esses pinos é possível enviar ou receber os sinais de nível lógico alto ou baixo. Na programação do arduino há a possibilidade de definir um pino digital como INPUT, entrada de dados, ou OUTPUT, saída de dados. Para que seja possível acionar os relés, será necessário conectar um fio dos pinos digitais, definidos como OUTPUT, do arduino para os relés.

A protoboard permitirá expandir e melhorar a organização das conexões. Os jumpers serão utilizados para conectar os componentes. O módulo relé é um componente que possui uma bobina e três contatos chamados, comum, normalmente aberto e normalmente fechado. Quando a bobina não está energizada, o contato comum fecha o circuito normalmente fechado e possibilita a passagem de corrente. Por outro lado, quando a bobina é energizada, o contato comum fecha o circuito normalmente aberto. Desta forma, é possível controlar, através do Arduino, a energização ou não da bobina do módulo relé e assim controlar o ligamento ou

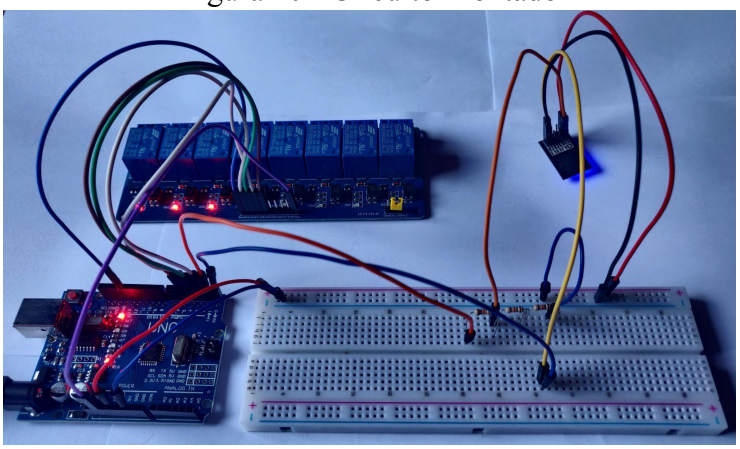
desligamento de dispositivos conectados a ele. Este é o motivo da utilização deste componente no circuito. O módulo relé utilizado para este projeto é de 8 canais, com isso é possível controlar o ligamento ou desligamento de até 8 dispositivos. A chave bóia será utilizado para checar o nível d'água do reservatório onde se encontra a bomba d'água. Com ele será possível evitar que a bomba seja ligada quando não há água do reservatório, evitando problemas na mesma. Para facilitar o entendimento da conexão entre os componentes, nas figuras 9 e 10 são mostrados o sketch do circuito e sua montagem, respectivamente.

Figura 9 - Sketch do Circuito



Fonte: Autor, Ilustrada no software Fritzing (2021).

Figura 10 - Circuito Montado



Fonte: Autor (2021).

3.1.3 Funcionamento do Circuito

Sabe-se que o Arduino e o ESP-01S trocam informações via comunicação serial, para facilitar esta comunicação entre estes dois componentes foi implementada a troca de dados no formato JSON. JSON (JavaScript Object Notation), é um padrão de serialização de dados, de formato leve, para a troca de dados entre sistemas. Atualmente este padrão é bastante utilizado na comunicação entre sistemas e é bem simples de ser lido. Além disso, o JSON possibilita uma maior velocidade no transporte de dados. Um exemplo de dados no formato JSON é apresentado na figura 11.

Figura 11 - Dados no formato JSON

```
[
  {
    "pin": 1,
    "on": 0
  },
  {
    "pin": 2,
    "on": 1
  }
]
```

Fonte: Autor (2021).

O circuito foi programado com o objetivo de permitir que os status dos pinos do Arduino sejam consultados e alterados (somente os pinos de saída) via internet através do

ESP-01S, que funcionará como um servidor web. Para isso, foi necessário configurar uma API no módulo WiFi. API (do inglês Application Programming Interface ou em português Interface de Programação de Aplicações), é um conjunto de padrões e protocolos utilizados por um software para possibilitar a integração entre aplicações, desenvolvidas na mesma linguagem de programação ou não. Geralmente utiliza o formato JSON para a troca de informações. Através de uma API é possível integrar aplicações desenvolvidas em linguagens diferentes, sem a necessidade de uma aplicação saber como a outra foi desenvolvida e implementada. A API desenvolvida no ESP-01S possibilita a consulta e modificação dos status dos pinos do arduino através de requisições HTTP.

Basicamente, o HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação que permite a obtenção de recursos, como documentos HTML ou dados no formato JSON, e fácil de se usar. Ele é um protocolo cliente-servidor, ou seja, as requisições são realizadas inicialmente pelo cliente, geralmente um navegador Web, para um servidor. Em outras palavras, o cliente requisita recursos e o servidor o responde. O HTTP possui métodos de requisições, geralmente é um verbo como GET, POST, DELETE, PUT, entre outros. Estes verbos dizem ao servidor qual operação o cliente deseja realizar. Todas as respostas das requisições possuem um código (status) de retorno que é definido pelo servidor, pode ser um código de erro ou sucesso, por exemplo, uma requisição de consulta bem sucedida recebe um código de retorno 200 (OK)[2]. Obviamente, o servidor deve estar previamente programado para receber tais requisições e retornar uma resposta para o cliente. A seguir são descritos os principais verbos, mencionados anteriormente:

- **GET**: solicita recursos ao servidor, geralmente uma página HTML ou dados no formato JSON.
- **POST**: submete informações para o servidor. Geralmente utilizado para registrar um novo recurso, como por exemplo, cadastrar um novo usuário.
- **DELETE**: utilizado para remover um recurso específico.
- **PUT**: utilizado para atualizar as informações de um recurso específico.

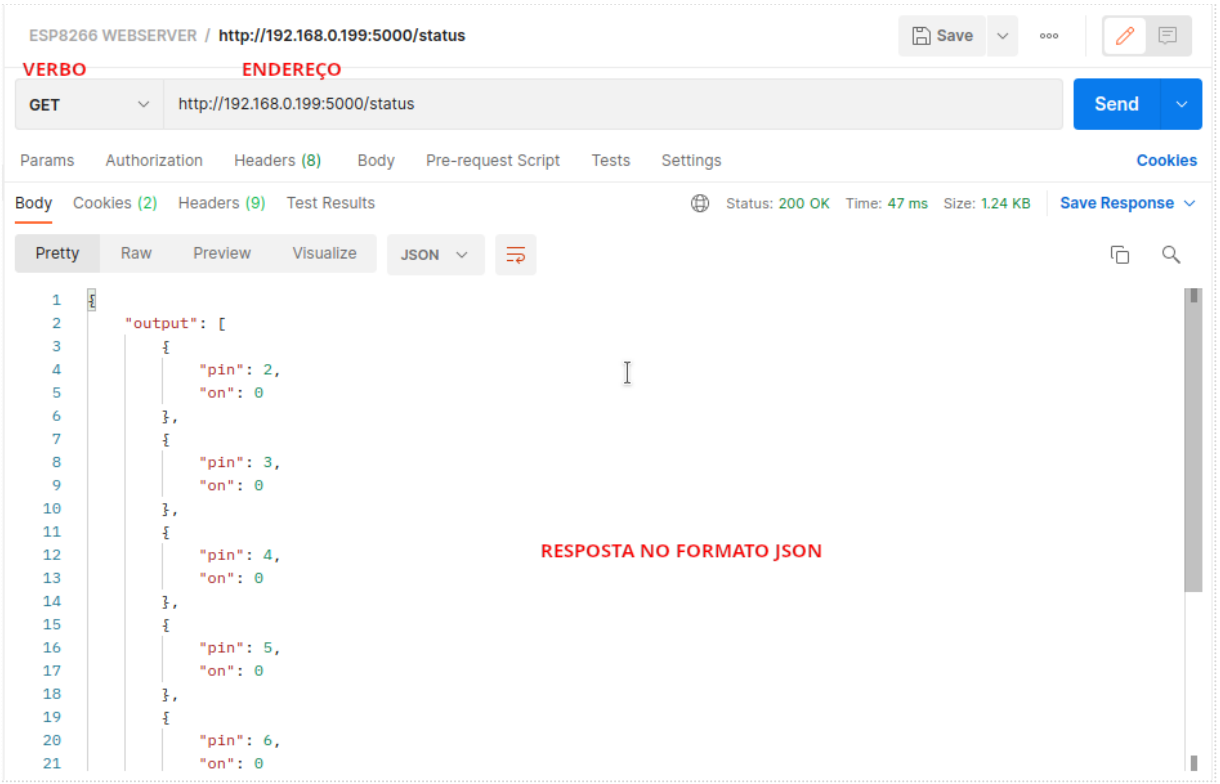
Diante disto, para consultar o status de todos os pinos do arduino basta enviar uma requisição GET para *http://IPDOESP[3]:PORTA[4]/status* da API, e para alterar os status dos

[2] Mais informações podem ser encontradas em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status>.
[3] IP do ESP-01S na rede.
[4] PORTA onde está configurada a API do ESP-01S.

pinos de saída, uma requisição POST, para o mesmo endereço, contendo as informações dos pinos e os status a serem setados. Ao receber uma requisição, GET ou POST, o ESP-01S irá se comunicar com o Arduino, via serial, para consultar ou setar os status dos pinos e ficará aguardando por uma resposta do Arduino, se não a receber retornará erro para o cliente que realizou a requisição na API. Se o Arduino receber a solicitação do ESP-01S, irá tratá-la e logo em seguida responderá o ESP-01S com os status atuais de todos os pinos. Ao receber a resposta do Arduino, o ESP-01S responderá a requisição do cliente. Os testes de consumo da API do ESP-01S foram realizados através do POSTMAN, software que fornece um ambiente de testes de consumo de API's.

Na figura 12, é possível verificar uma requisição GET realizada para a API do EPS-01S. Esta requisição é realizada com o propósito de coletar os status dos pinos do Arduino no momento da requisição. Na figura abaixo, é possível verificar o verbo utilizado, o endereço da requisição e o formato da resposta do ESP-01S.

Figura 12 - Requisição GET para a API do ESP-01S

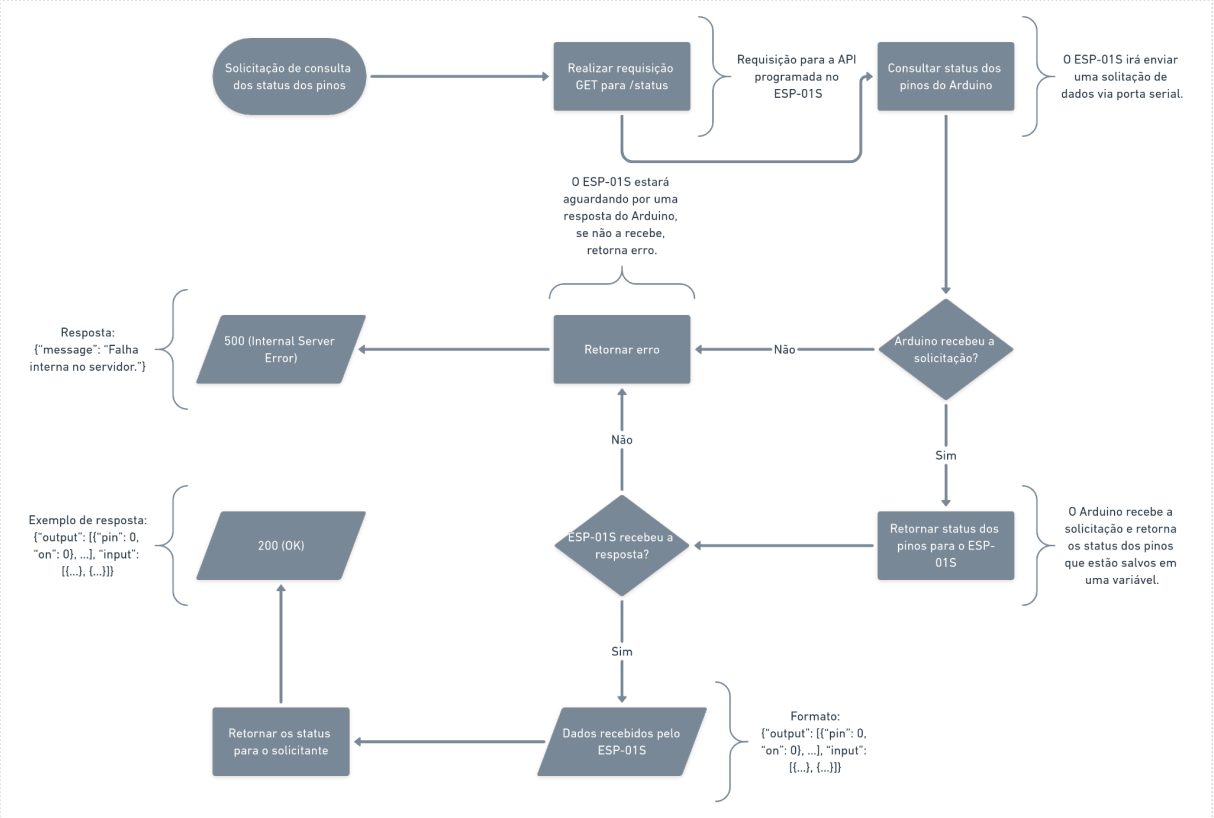


Fonte: Autor (2021).

Ao receber uma requisição do tipo GET no endpoint */status*, o ESP-01S enviará uma requisição para o Arduino, via comunicação serial, solicitando os status dos pinos e ficará

aguardando por uma resposta, se não a receber irá responder o cliente da requisição com uma mensagem de erro. Se o Arduino receber a solicitação do ESP-01S, o mesmo irá consultar os status dos pinos que estão salvos em uma variável e irá responder o ESP-01S com os status. Ao receber a resposta do Arduino, o ESP-01S responderá o cliente com as informações sobre os pinos. Este processo pode ser verificado no fluxograma da figura 13.

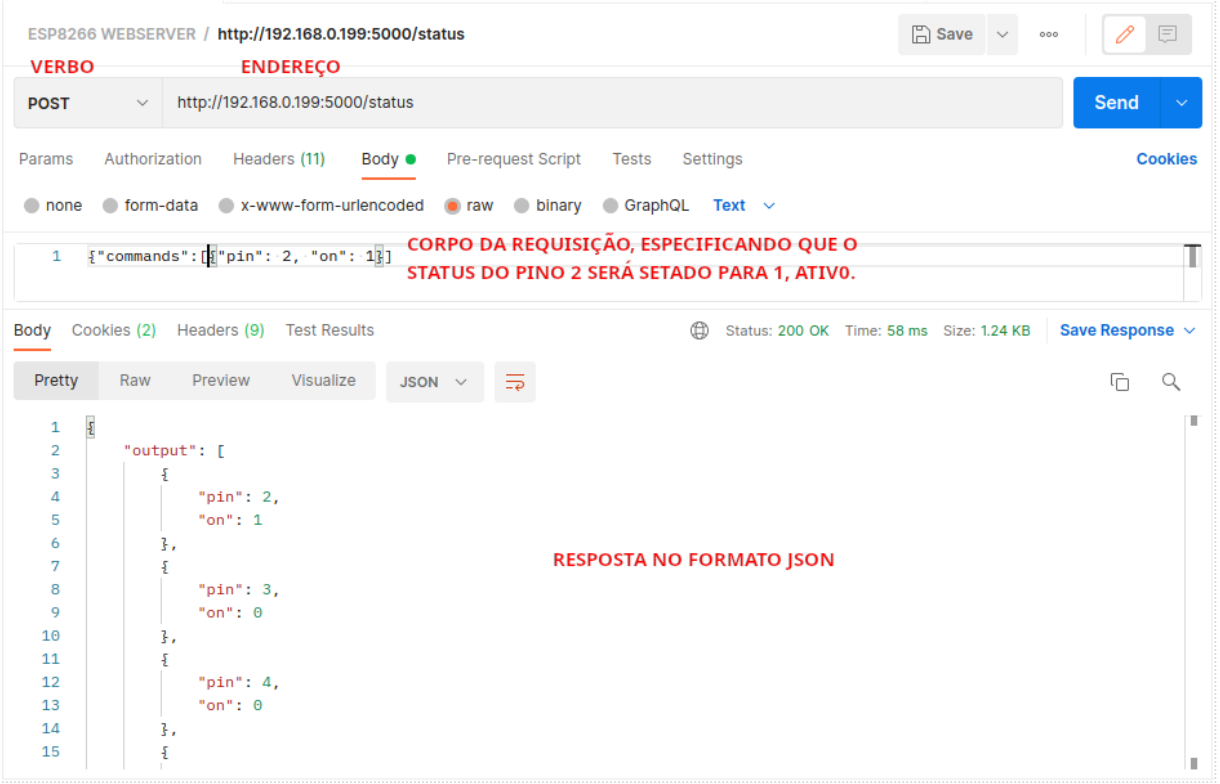
Figura 13 - Fluxograma de consulta dos status dos pinos do Arduino



Fonte: Autor (2021).

A API do ESP-01S também possibilita a modificação dos status dos pinos do Arduino. A figura 13 mostra uma requisição POST para a API. Esta requisição, diferentemente da requisição anterior, deve enviar em seu corpo dados que especificam os pinos e os seus respectivos status a serem setados no Arduino. Da mesma forma que a requisição de consulta dos status, nesta requisição o ESP-01S também retornará os status dos pinos do Arduino.

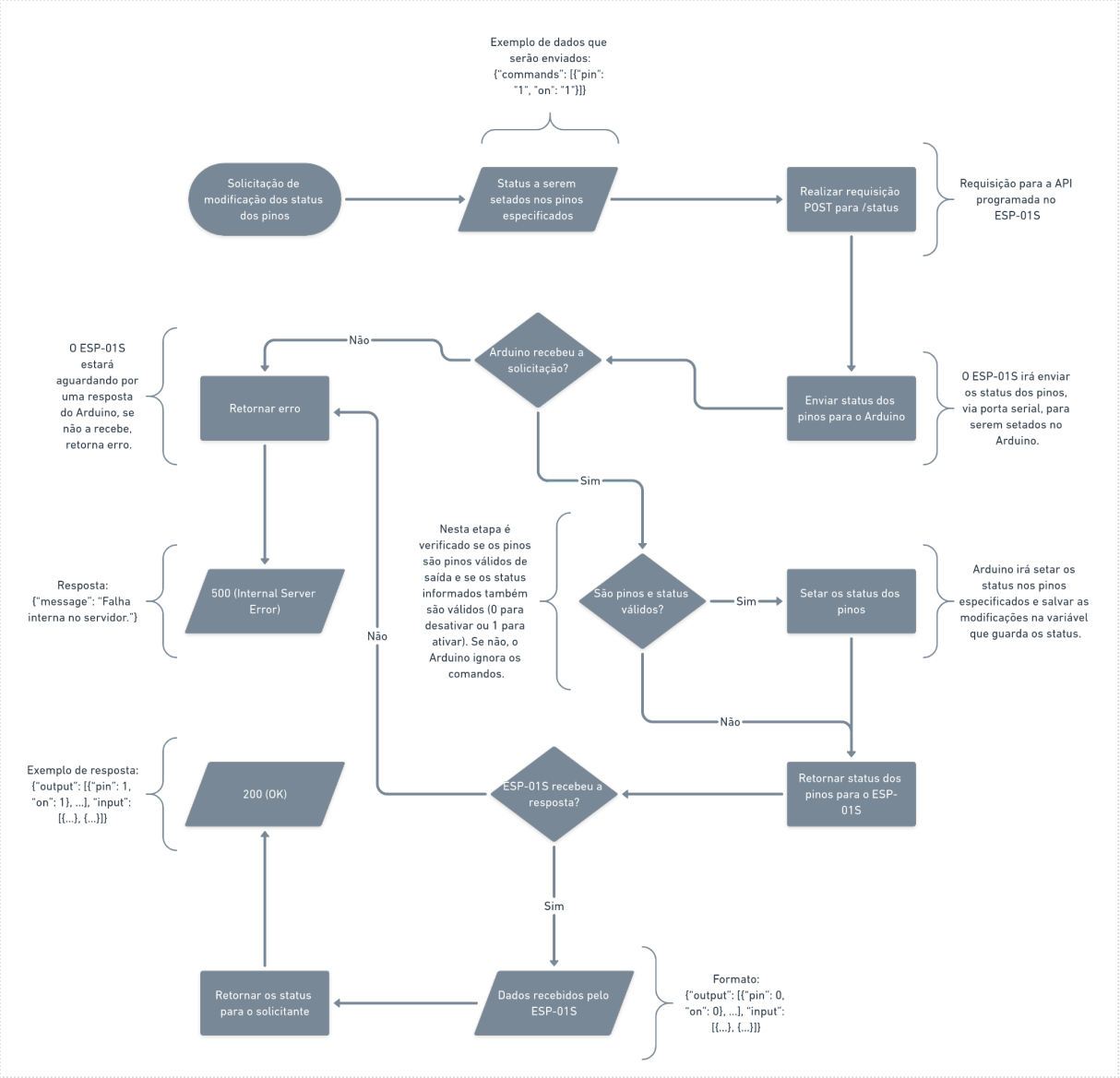
Figura 14 - Requisição POST para a API do ESP-01S



Fonte: Autor (2021).

Ao receber uma requisição do tipo POST no endpoint */status,* o ESP-01S enviará uma requisição, via comunicação serial, para o Arduino contendo os pinos e status a serem setados e ficará aguardando por uma resposta, se não a receber irá responder o cliente da requisição com uma mensagem de erro. Se o Arduino receber a solicitação do ESP-01S, o mesmo irá verificar se os pinos e status são válidos. Se sim, irá realizar a modificação nos status dos pinos e logo em seguida responderá o ESP-01S com os status atuais. Ao receber a resposta do Arduino, o ESP-01S responderá o cliente com as informações sobre os pinos. Este processo é ilustrado no fluxograma da figura 15.

Figura 15 - Fluxograma de modificação dos status dos pinos do Arduino



Fonte: Autor (2021).

Como visto no capítulo anterior, por questões de segurança e um melhor controle, a API do ESP-01S não será consumida diretamente pela aplicação móvel. Haverá um servidor de comandos que intermediará esta API e a aplicação móvel. Nesta perspectiva, o servidor de comandos será um cliente que consumirá a API do ESP-01S.

3.2 SERVIDOR DE COMANDOS

Como já mencionado, umas das atribuições do servidor de comandos é prover uma maior segurança para o sistema de automação. Desta forma, a API do ESP-01S somente

poderá ser consumida pelo servidor de comandos de forma local, não será aberta para a internet. Foi implementada uma segunda API neste servidor, utilizando a linguagem de programação PHP e o framework Laravel. Esta será aberta para a internet e será consumida pela aplicação móvel. A mesma possui autenticação e só permite que usuários autorizados tenham acesso às informações do sistema. Para facilitar o entendimento do funcionamento do servidor, a seguir serão apresentados detalhes sobre a base de dados que armazena as informações do sistema, bem como a lógica de funcionamento por trás do servidor.

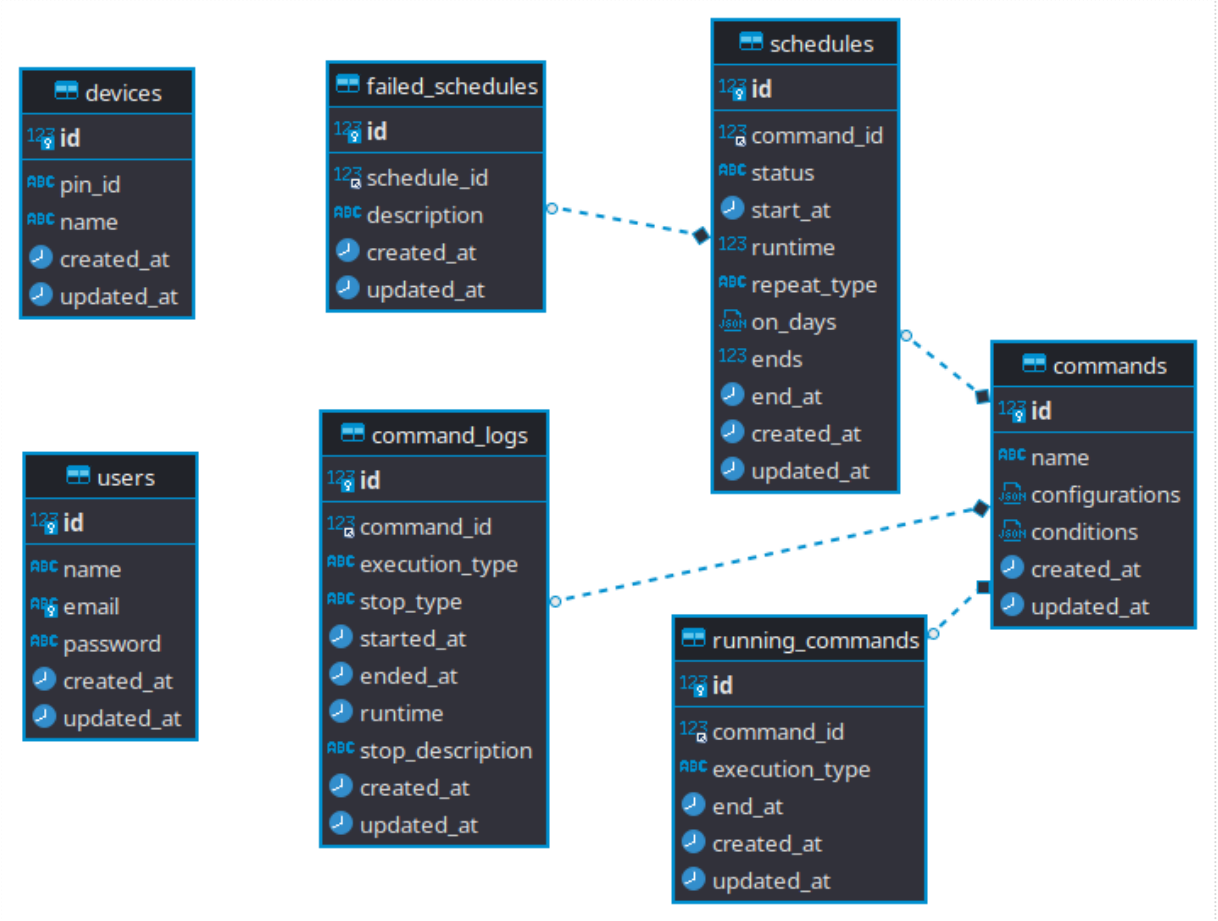
3.2.1 Banco de dados

Foi projetado um banco de dados que irá armazenar todas as informações relacionadas ao sistema de automação. Nesta base de dados foram criadas as entidades: Usuário, Dispositivo, Comando, Comando em Execução, Log de Comando, Agendamento e Agendamentos Falhados. A seguir serão apresentados os papéis de cada entidade para o funcionamento do sistema.

- **Usuário**: representará os usuários da aplicação.
- **Dispositivo:** representará os dispositivos cadastrados pelo usuário. O usuário do sistema poderá dar nomes aos pinos do Arduino, isso facilitará a sua utilização da aplicação.
- **Comando:** representará os comandos que serão executados envolvendo os dispositivos. O usuário poderá cadastrar um comando que liga os dispositivos X e Y ao mesmo tempo.
- **Comando em Execução:** representará os comandos que estão em execução no momento.
- **Log de Comando:** representará todos os registros de execuções de comandos.
- **Agendamento:** representará os agendamentos de comandos. O usuário poderá agendar a execução de comandos.
- **Agendamentos Falhados:** representará as falhas de execução de agendamentos. Todas as falhas e seu motivo serão registrados.

Na figura 16, é mostrado o Diagrama de Relacionamento de Entidade, nele é possível visualizar todos os atributos das entidades e os relacionamentos com outras entidades.

Figura 16 - Diagrama de Relacionamento de Entidade



Fonte: Autor (2021).

A API do servidor de comandos permitirá que o usuário, através da aplicação móvel, execute as seguintes tarefas:

- Listar, cadastrar, editar e deletar dispositivos;
- Listar e cancelar comandos em execução;
- Listar, cadastrar, editar, deletar e executar comandos;
- Listar, cadastrar e cancelar agendamentos;
- Consultar o histórico de comandos executados.

3.2.2 Lógica de funcionamento

A maioria das tarefas realizadas pela aplicação irão necessitar dos status dos pinos do Arduino em tempo real. Diante disto, foi configurado um serviço, no servidor de comandos, que é responsável por consultar periodicamente os status dos pinos do Arduino através da API

do ESP-01S e armazená-los em um banco de dados Redis[5]. Banco de dados de alta velocidade que armazena dados em memória, sendo este o motivo de sua utilização neste projeto. Com isso, o servidor de comandos terá acesso aos status dos pinos em qualquer momento e assim poderá tomar decisões, que dependem dessas informações, com maior precisão.

Como vimos na figura 16, os comandos possuem os atributos: nome, configurações e condições. As configurações de um comando representam as modificações que serão feitas nos pinos do arduino, e as condições, condições de execução do comando. Por exemplo, o comando descrito na figura 17, só ligará os dispositivos A e B, pinos 2 e 3 do Arduino, se o sensor 1, pino 9, estiver desligado.

Figura 17 - Exemplo de comando registrado no banco de dados

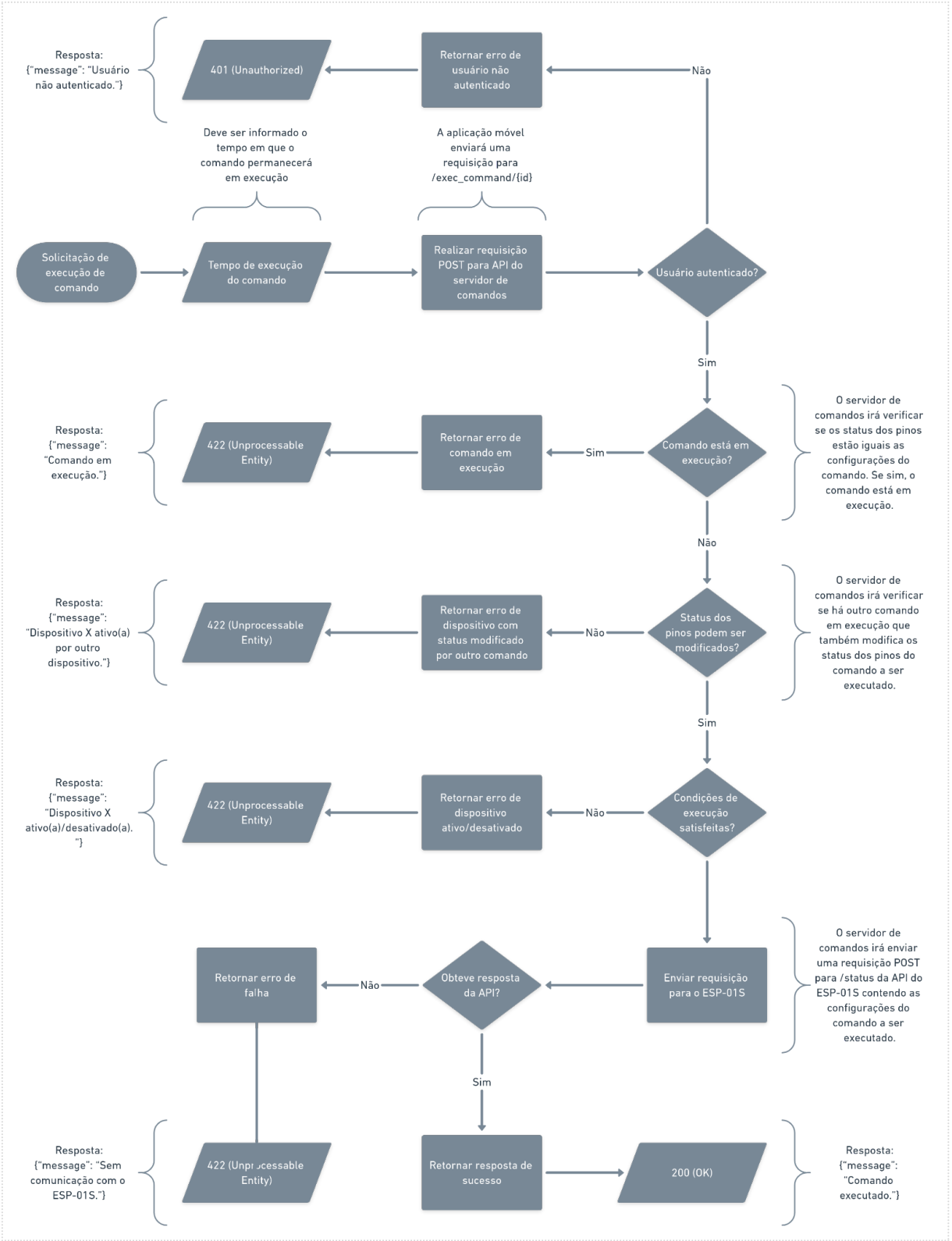
id	name	script	conditions	created_at	updated_at
91	Ligar dispositivos A e B	[{"on": "1", "pin": "2"}, {"on": "1", "pin": "3"}]	[{"on": "0", "pin": "9"}]	2021-10-31 21:15:17	2021-10-31 21:20:40

Fonte: Autor (2021).

Para executar o comando da figura 17, será necessário enviar uma requisição POST para a API do servidor de comandos no endpoint */exec_command/91*. Ao receber a requisição, o servidor irá verificar se o usuário requisitante está autenticado, se não estiver será retornado um erro informado que o usuário não está autenticado. Se estiver autenticado, o servidor irá verificar se o comando já está em execução, se estiver, retornará uma mensagem de erro informando que o comando está em execução. Se não estiver, irá verificar se há algum comando em execução que também modifica os status dos pinos 2 ou 3, se houver, o servidor retornará uma mensagem informando que o comando não pode ser executado. Se não houver, dará continuidade e irá verificar se as condições de execução do comando estão satisfeitas, se não estiverem, retornará erro informando que as condições de execução não estão satisfeitas. Se estiverem, enviará uma requisição para a API do ESP-01S para modificar os status dos pinos do Arduino. Este processo é ilustrado na figura 18.

[5] Mais informações sobre o banco de dados Redis podem ser encontradas em <https://redis.io/>.

Figura 18 - Fluxograma de execução de comando



Fonte: Autor (2021).

Foi configurado um serviço no servidor de comandos que é responsável por verificar periodicamente durante a execução de um comando, se suas condições de execução continuam satisfeitas. Se em algum momento da execução as condições deixarem de ser satisfeitas, o servidor de comandos enviará uma requisição para o ESP-01S para que o comando seja cancelado. Esta é uma medida que promove uma execução de comandos de forma segura e previne problemas. Por exemplo, foi executado um comando que liga uma bomba d'água e a condição de execução deste comando é que o Chave Bóia X (chave bóia que verifica se o nível d'água do reservatório está em um nível seguro para ligar a bomba) esteja em nível lógico baixo, desativado. Quando o usuário solicitou a execução deste comando, o servidor de comandos verificou que as condições de execução do mesmo estavam satisfeitas, desta forma o comando foi executado. Porém, durante a execução do comando o status do Chave Bóia X foi modificado para nível lógico alto, o que indica que o nível d'água do reservatório já não é mais seguro para permanecer com a bomba ligada, sendo assim o servidor de comandos identifica esta mudança no status do Chave Bóia X e imediatamente cancela a execução do comando e previne danos a bomba d'água.

Sempre que um comando é executado é inserido um novo registro na tabela de comandos em execução com as informações:

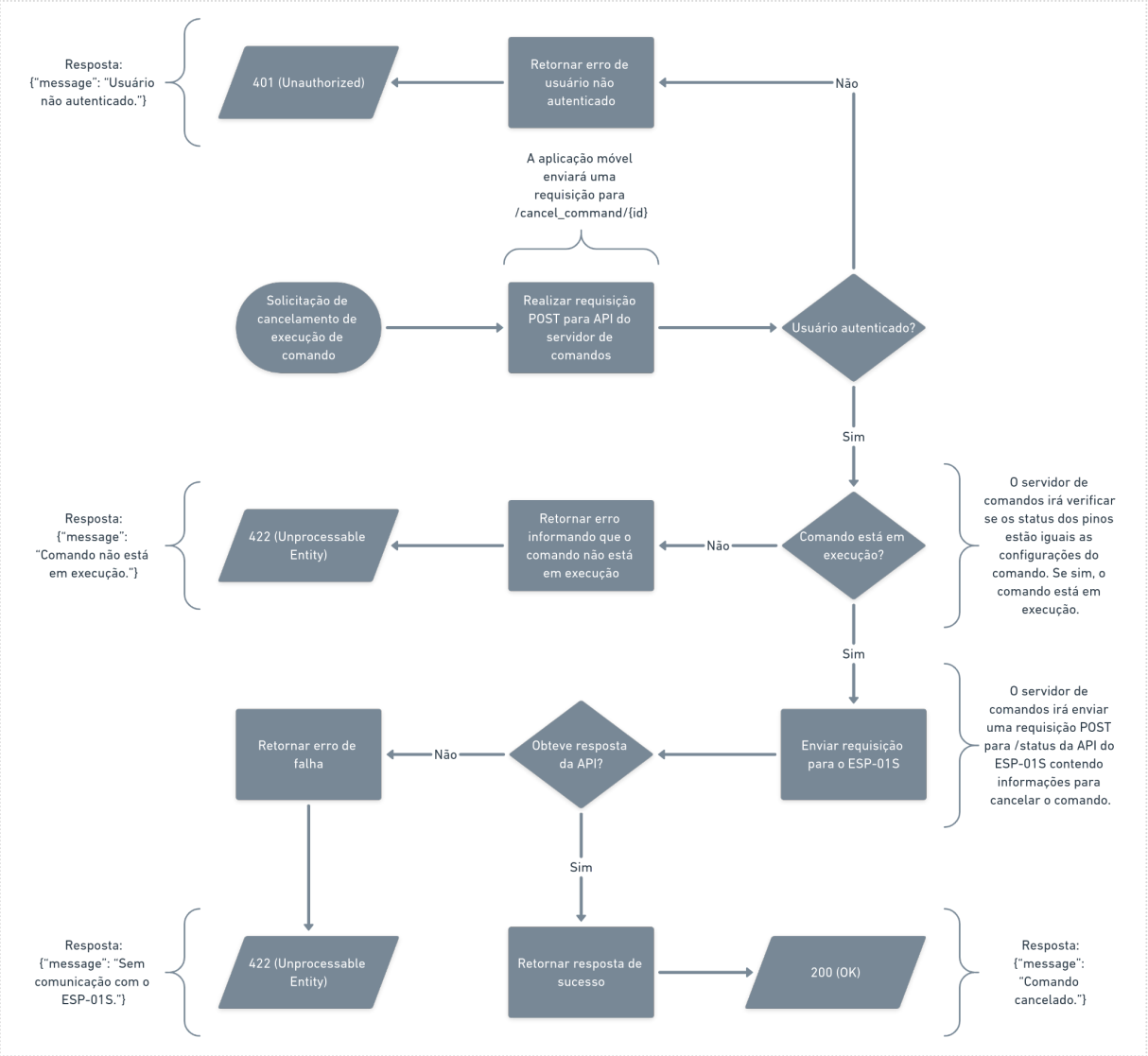
- Id do comando executado;
- Tipo de execução (se foi uma solicitação do usuário ou um agendamento);
- Horário de encerramento, este é definido com base no tempo de execução que o usuário informa na solicitação de execução de um comando.

Há um outro serviço do servidor que verifica periodicamente os comandos em execução e verifica se o horário de encerramento já passou, se sim a execução do comando é encerrada. Este serviço possibilita uma medida de segurança para evitar que o comando fique em execução por um longo período caso o usuário esqueça de encerrá-lo.

Por mais que o usuário defina um tempo de execução do comando, o mesmo pode desejar encerrá-lo antes que este tempo se esgote. Pensando nisso, foi implementada a possibilidade do usuário solicitar o cancelamento de um comando em execução. Ao receber a solicitação de cancelamento de comando em execução, o servidor irá verificar se o comando realmente está em execução, se não, retornará a mensagem informando que o comando não

está em execução. Se sim, o servidor solicitará o cancelamento do comando. A figura 19, apresenta um fluxograma de cancelamento de comando.

Figura 19 - Fluxograma de cancelamento de execução de comando



Fonte: Autor (2021).

Com o intuito de gerar relatórios, sempre que um comando é cancelado ou encerrado é inserido um novo registro na tabela de logs de comandos com as informações:

- Id do comando executado;
- Tipo de execução (solicitado pelo usuário ou agendamento);
- Tipo de parada (cancelado pelo usuário, interrompido ou tempo esgotado);
- Data de início da execução;

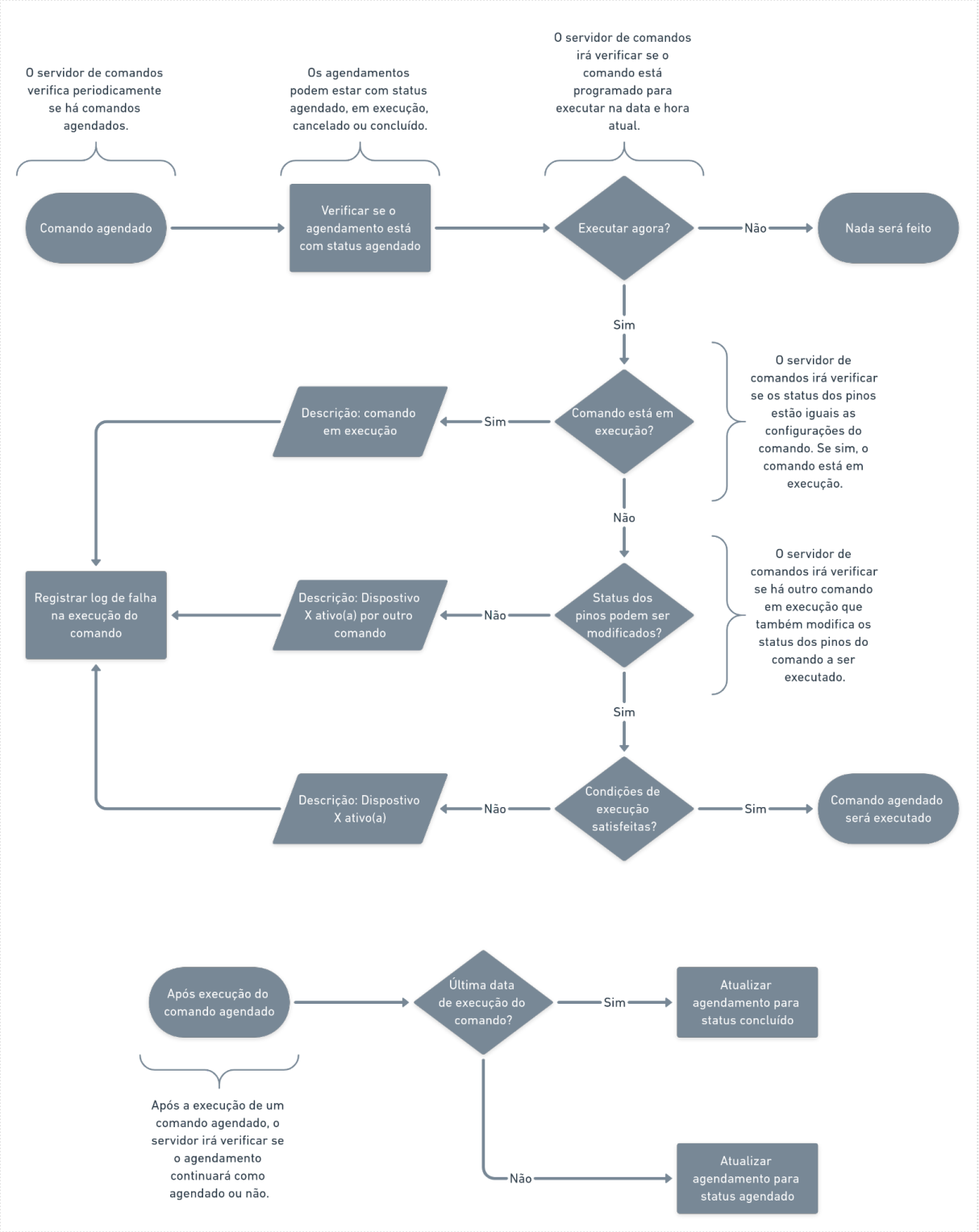
- Data de encerramento da execução;
- Tempo em execução;
- Descrição do encerramento.

Já vimos como funcionam as solicitações de execução e cancelamento de um comando pelo usuário. Por fim, veremos que o servidor de comandos também possibilita que o usuário programe a execução de comandos. É possível programar a execução das seguintes formas:

- Em dias específicos;
- Todos os dias, com ou sem uma última data de execução;
- Toda semana em dias específicos da semana, com ou sem uma última data de execução;
- Todos os meses em dias específicos do mês, com ou sem uma última data de execução;

Há tambeḿ um serviço no servidor que é responsável por verificar periodicamente se há execuções de comandos agendadas. Se houver, é verificado se o comando está programado para executar na data e hora atual. Se sim, o servidor irá verificar se as condições de execução estão satisfeitas, se não, irá inserir um novo registro na tabela de agendamentos falhados com a data, horário e descrição da falha. Se sim, irá seguir o procedimento padrão de execução de comandos e atualizará o agendamento para o status *em execução*. Após a execução de um comando agendado, o servidor irá verificar se aquele foi o último dia de execução do comando agendado, se sim, atualizará o agendamento para *concluído*, se não, atualizará o agendamento para *agendado* novamente. A figura 20, mostra o fluxograma de execução de um comando agendado.

Figura 20 - Fluxograma de execução de comando agendado



Fonte: Autor (2021).

4 RESULTADOS E DISCUSSÕES

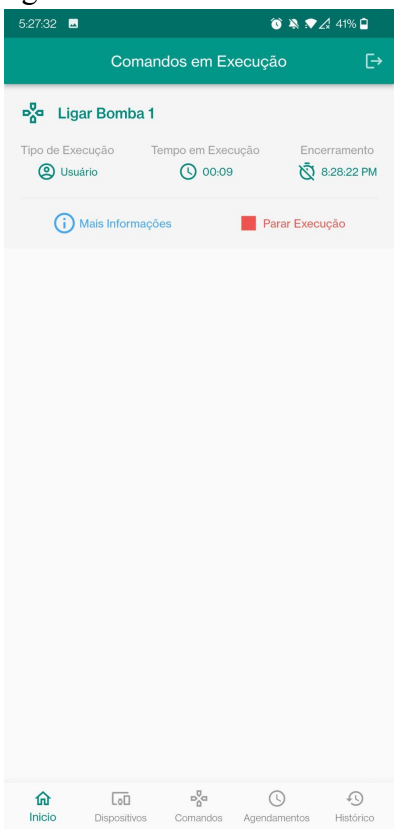
Este capítulo tem a finalidade de apresentar a aplicação móvel desenvolvida e os resultados obtidos na realização dos testes práticos.

4.1 APLICAÇÃO MÓVEL

Desenvolveu-se uma aplicação móvel com intuito de facilitar a utilização do sistema e dar mais autonomia ao usuário. A aplicação foi desenvolvida utilizando a linguagem de programação Dart e o framework Flutter. Através dela é possível gerenciar dispositivos, comandos e agendamentos.

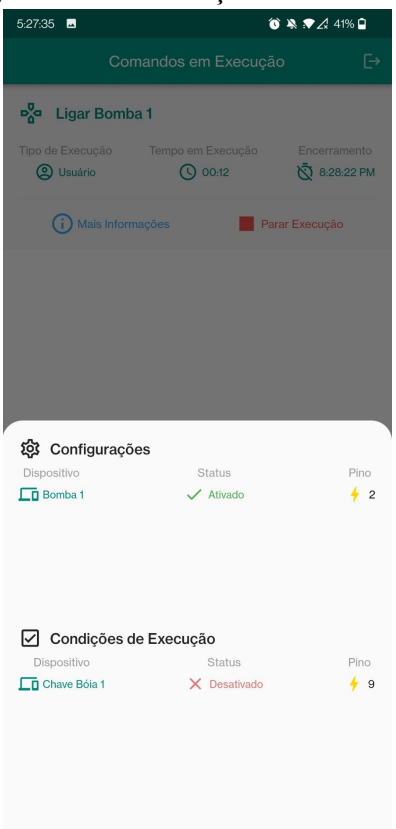
Na tela inicial, figuras 21 e 22, o usuário poderá acompanhar em tempo real os comandos que estão sendo executados no momento. Acerca dos comandos serão apresentadas informações como o tipo de execução do comando (usuário ou agendamento), tempo em que o comando está em execução e horário de encerramento. O usuário poderá verificar as configurações e condições de execução do comando e também poderá parar a execução.

Figura 21 - Tela inicial do APP



Fonte: Autor (2021).

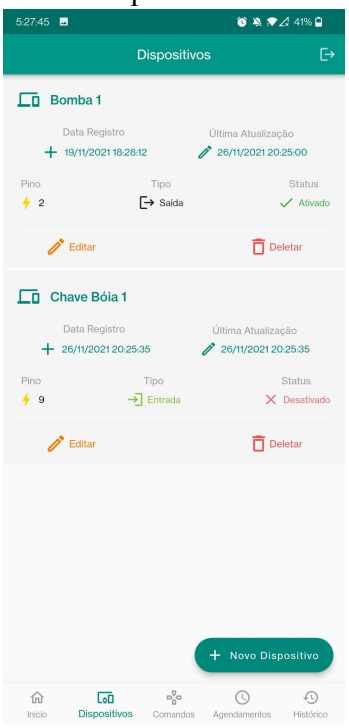
Figura 22 - Informações do comando



Fonte: Autor (2021).

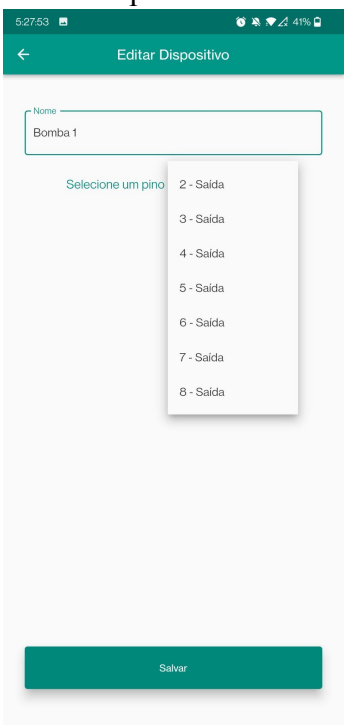
Nas telas das figuras 23, 24 e 25, o usuário poderá gerenciar os dispositivos cadastrados no sistema. É possível cadastrar, editar e deletar dispositivos. Como já informado anteriormente, especificar os dispositivos que estão conectados aos pinos do Arduino facilitará o entendimento do funcionamento do sistema. Na listagem de dispositivos são mostradas informações como a numeração do pino em que o dispositivo está conectado, o tipo do pino (entrada ou saída) e o status atual do dispositivo (ativo ou desativado) que é atualizado em tempo real.

Figura 23 - Listagem de dispositivos



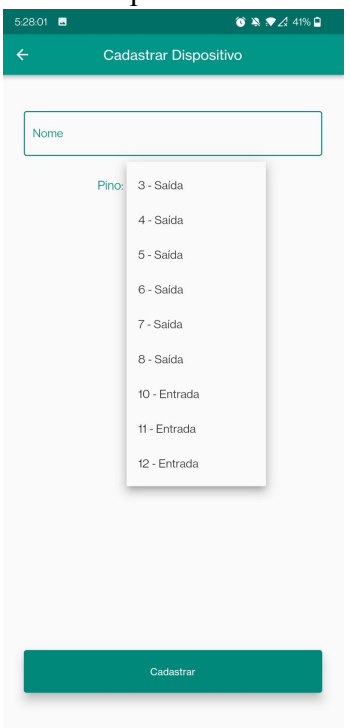
Fonte: Autor (2021).

Figura 24 - Edição de dispositivos



Fonte: Autor (2021).

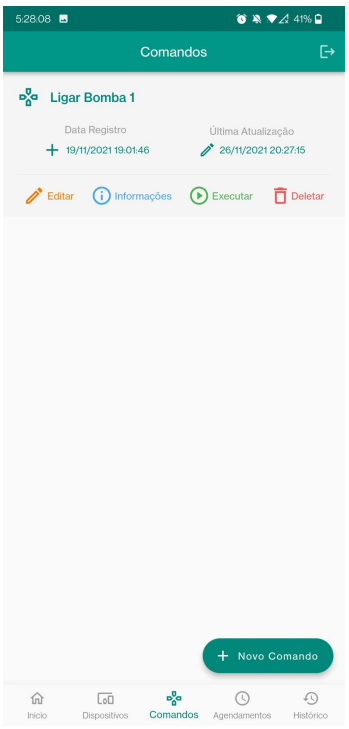
Figura 25 - Cadastro de dispositivos



Fonte: Autor (2021).

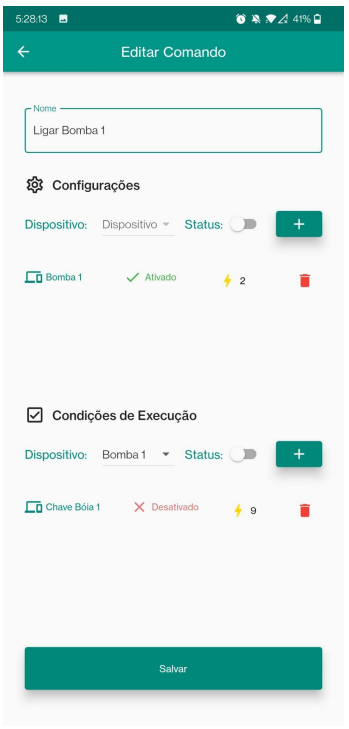
As figuras numeradas de 26 a 30, apresentam as telas de gerenciamento de comandos. A partir destas páginas é possível consultar todos os comandos cadastrados, assim como também verificar informações sobre o comando, editá-lo, executá-lo e realizar o cadastro de um novo comando. Na figura 28, podemos notar que é possível verificar as configurações do comando, ou seja, saber as alterações que determinado comando irá realizar nos pinos do Arduino. Também é possível verificar as condições de execução do comando de forma simples e prática.

Figura 26 - Listagem de comandos



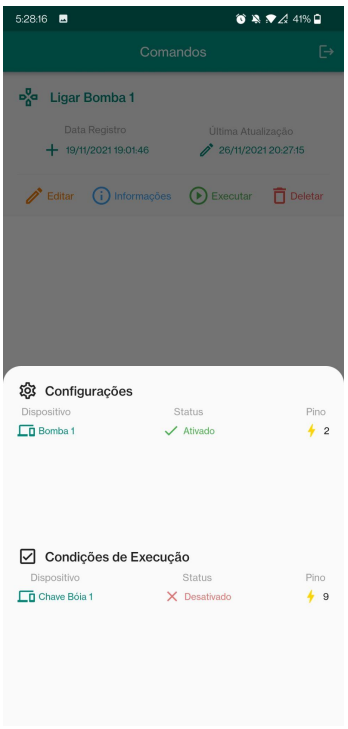
Fonte: Autor (2021).

Figura 27 - Edição de comando



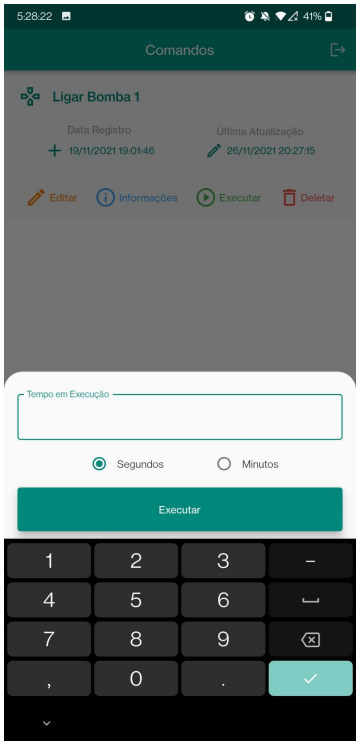
Fonte: Autor (2021).

Figura 28 - Informações do comando



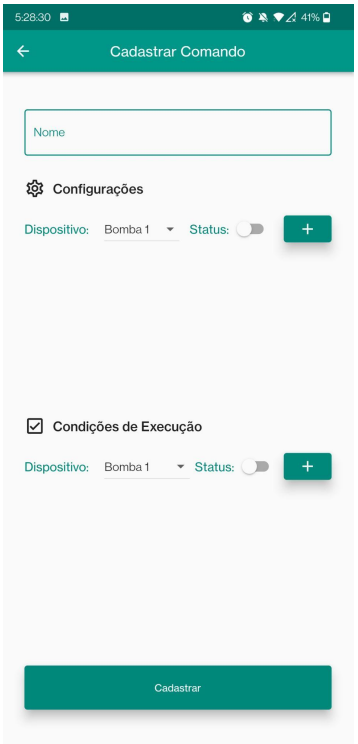
Fonte: Autor (2021).

Figura 29 - Executar de comando



Fonte: Autor (2021).

Figura 30 - Cadastro de comando



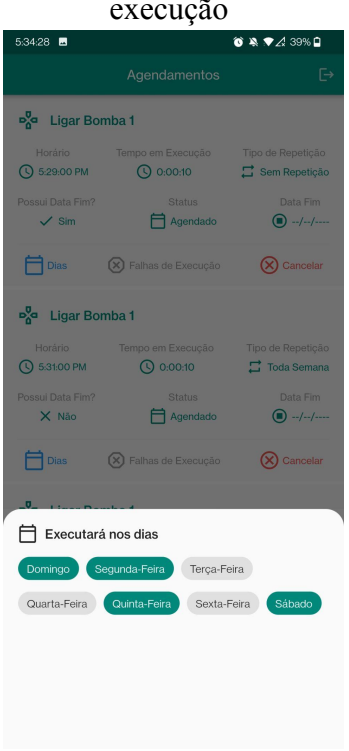
Fonte: Autor (2021).

As figuras 31, 32 e 33, apresentam as telas de gerenciamento de agendamentos. O usuário poderá acompanhar todos os agendamentos e informações como horário de execução do comando agendado, tempo de execução, tipo de repetição do agendamento (sem repetição, todos os dias, toda semana ou todo mês), se o agendamento possui ou não data fim, o status do agendamento (agendado, em execução, cancelado ou concluído) e a data fim, caso o agendamento possua uma data fim. Ainda sobre agendamentos, é possível realizar o cadastro de um novo agendamento e realizar um filtro de agendamentos, por status, que o usuário deseja acompanhar, como mostram as figuras 34 e 35 abaixo.

Figura 31 - Listagem de agendamentos

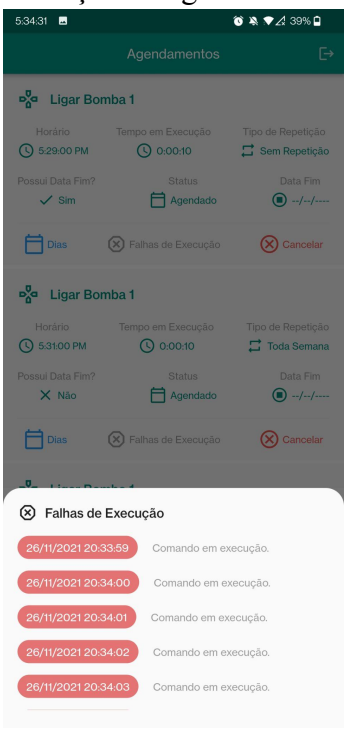
Fonte: Autor (2021).

Figura 32 - Dias de execução



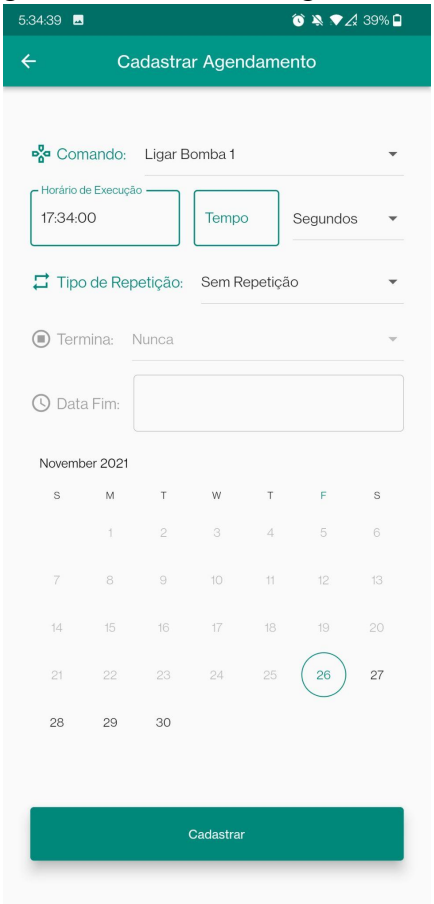
Fonte: Autor (2021).

Figura 33 - Falhas de execução do agendamento



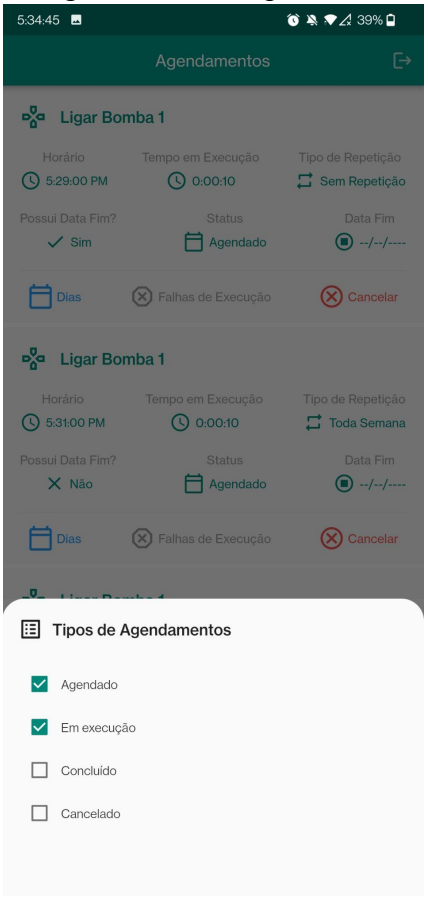
Fonte: Autor (2021).

Figura 34 - Cadastro de agendamento



Fonte: Autor (2021).

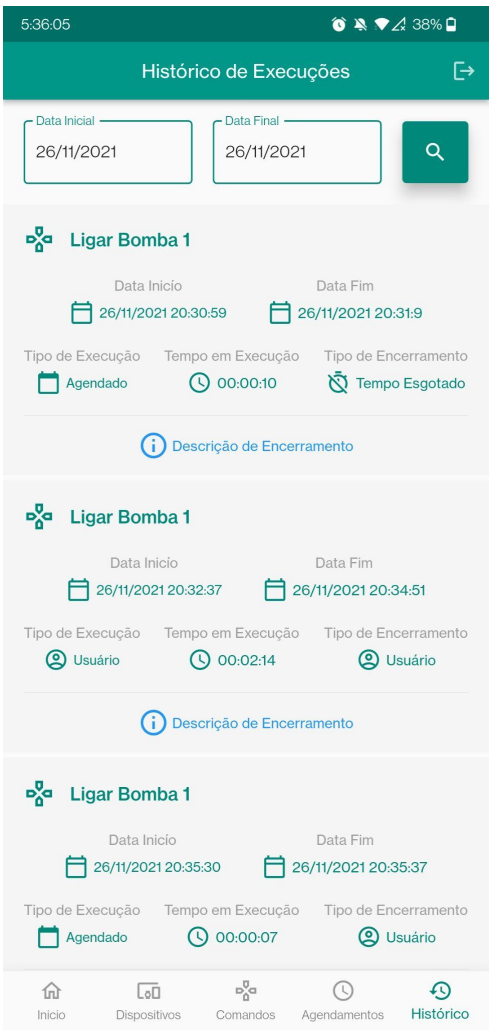
Figura 35 - Filtro por status



Fonte: Autor (2021).

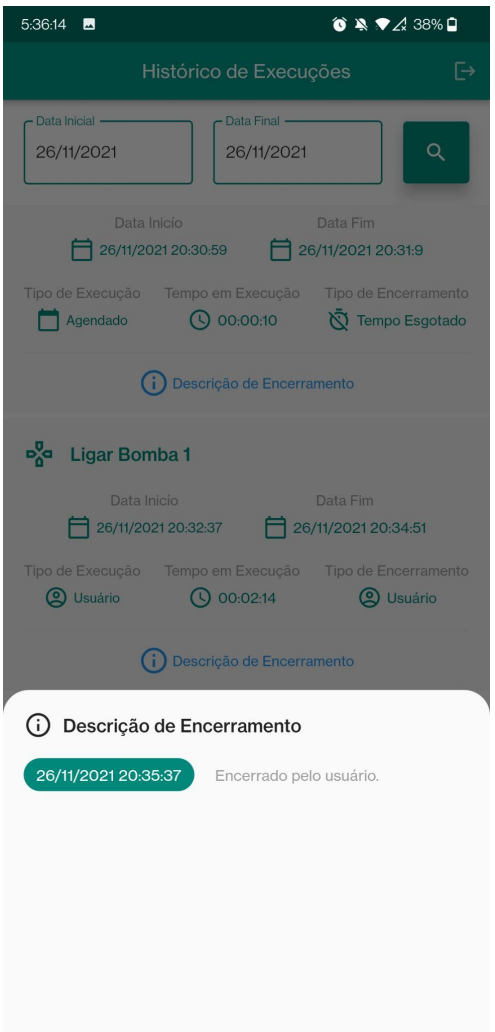
Por fim, a aplicação também possibilita que o usuário tenha acesso a um relatório contendo todas as execuções de comandos. Este relatório contém informações como nome do comando, data início e fim da execução do comando, tempo em execução, tipo de execução e tipo de encerramento, se foi encerrado pelo usuário, por tempo esgotado ou interrompido durante a execução. É possível filtrar a data que deseja verificar o relatório e consultar uma descrição de encerramento do comando. As figuras 36 e 37 ilustram a página de relatórios.

Figura 36 - Listagem de comandos executados



Fonte: Autor (2021).

Figura 37 - Descrição de encerramento do comando



Fonte: Autor (2021).

Todas as figuras mostradas neste capítulo foram reproduzidas durante a utilização prática do sistema de automação desenvolvido neste trabalho.

5 CONSIDERAÇÕES FINAIS

Atualmente, a automação é cada vez mais implementada em sistemas que ainda não são automatizados. A mesma provê várias vantagens, como a redução de trabalhos repetitivos, ganho de tempo, produtividade e otimização de processos. O presente trabalho foi desenvolvido com o objetivo de projetar a automação de um sistema de irrigação a partir da utilização de microcontroladores (Arduino e ESP-01S) e uma aplicação móvel.

A automação desenvolvida neste projeto possibilita ao usuário uma maior autonomia e melhor administração de todo o sistema. Assim como também, provê a prevenção de problemas e danos aos dispositivos utilizados, como por exemplo, o não acionamento de uma bomba d'água quando o reservatório está vazio. Através da aplicação móvel, também é possível acompanhar um relatório de todas as atividades executadas no sistema.

Um problema encontrado, durante o desenvolvimento do projeto, foi o aumento do tempo de resposta, em alguns momentos, aos comandos executados pela aplicação, isto devido ao circuito estar conectado à internet através do ESP-01S e sofrer interferências na rede WiFi.

Portanto, para trabalhos futuros, sugere-se a utilização de um Shield Ethernet, ao invés do ESP-01S, para conectar o arduino à internet. Esta medida resolverá o problema do aumento do tempo de resposta, na execução de comandos, devido a interferências na rede WiFi. Por se tratar de um sistema que necessita de dados em tempo real, aconselha-se a utilização de Websockets ao invés de requisições HTTP para consultar os status dos pinos do Arduino. Assim como o HTTP, o Websocket é um protocolo de transferência de dados, porém ao invés do servidor encerrar a conexão após a requisição do cliente ser respondida, a mesma é contínua e o cliente fica ouvindo todos os eventos disparados pelo servidor. Esta tecnologia é muito utilizada em chats de mensagens.

Por fim, os resultados obtidos durante os testes práticos mostraram que o objetivo do trabalho foi alcançado.

REFERÊNCIAS

Códigos de status de respostas HTTP. Developer Mozilla, 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status. Acesso em 27 out. 2021.

Comunicação serial e paralela. IFPR, 2021.Disponível em: http://wiki.foz.ifpr.edu.br/wiki/index.php/Comunica%C3%A7%C3%A3o_serial_e_paralela. Acesso em 25 out. 2021.

JSON. Json, 2021. Disponível em: https://www.json.org/json-en.html. Acesso em 26 out. 2021.

LAMB, Frank. Automação industrial na prática. Porto Alegre: Grupo A, 2015. 9788580555141. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788580555141/. Acesso em: 01 nov. 2021.

MONK, Simon. Programação com Arduino. Porto Alegre: Grupo A, 2017. 9788582604472. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788582604472/. Acesso em: 25 out. 2021.

Uma visão geral do HTTP. Developer Mozilla, 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview. Acesso em 27 out. 2021.