PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

ILUMINAÇÃO AUTOMATIZADA PARA INTEGRAÇAO DO AMBIENTE RESIDENCIAL

GOIÂNIA
2022

ROGÉRIO MELO LOPES

ILUMINAÇÃO AUTOMATIZADA PARA INTEGRAÇÃO DO AMBIENTE RESIDENCIAL

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Marcelo Antonio Adad de Araújo

GOIÂNIA
2022

ROGÉRIO MELO LOPES

ILUMINAÇÃO AUTOMATIZADA PARA INTEGRAÇAO DO AMBIENTE RESIDENCIAL

Este Trabalho de Conclusão de Curso julgado adequado para obtenção o Título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola Politécnica, da Pontifícia Universidade Católica de Goiás, em ___/___/_____.

_____

Prof. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de conclusão de curso

Banca examinadora:

_____

Orientador: Prof. Marcelo Antonio Adad de Araújo, M.E.E.

_____

Prof. Carlos Alexandre Ferreira de Lima, M.E.E.

_____

Prof. Nilson Cardoso Amaral, PhD.

GOIÂNIA
2022

RESUMO

O presente trabalho apresenta um estudo de automação e controle aplicados à iluminação de um ambiente residencial, fazendo uso de um sistema microcontrolado e ambiente Web. Para tanto, é produzido um protótipo de um circuito com o Arduino UNO, LEDs e o módulo ESP-01, que são conectados à internet através do Wi-Fi e controlados por um programa na Web ou aplicativo mobile. Os ambientes da Adafruit IO e blynk são utilizados para armazenar os dados e para realizar a comunicação entre o circuito microcontrolado e o ambiente da internet. Esse programa é utilizado pelo usuário para controlar a iluminação de um ambiente a partir de perfis predefinidos ou criados pelo usuário. Tudo isso, visa apontar uma maior acessibilidade e disponibilidade desses sistemas para pessoas comuns, além da importância da capacidade de controle da luz do ambiente e do seu impacto no usuário, e por consequência, aprofundar os estudos da *Internet of Things.*

Palavras-chave: Iluminação; automação; Arduino; comunicação; ambiente*, Internet of Things.*

ABSTRACT

The present work presents a study of automation and control applied to the lighting of a residential environment, making use of a microcontrolled system and Web environment. Therefore, a prototype of a circuit is produced with the Arduino UNO, LEDs and the ESP-01 module, which are connected to the internet through Wi-Fi and controlled by a web program or mobile application. The Adafruit IO and blynk environments are used to store data and to carry out communication between the microcontroller circuit and the internet environment. This program is used by the user to control the lighting of an environment from predefined profiles or created by the user. All this aims to point out a greater accessibility and availability of these systems for ordinary people, in addition to the importance of the ability to control the ambient light and its impact on the user, and consequently, deepen the studies of the Internet of Things.

Keywords: *Lighting; automation; Arduino; communication; environment, Internet of Things.*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

M.E.E.	Mestre em engenharia elétrica
Ma.	Mestra
Prof.	Professor
PUC	Pontifícia Universidade Católica
PhD	*Philosophiae Doctor*

LISTA DE SIGLAS

API	*Application Programming Interface*
cm	Centímetros
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
g	Gramas
GHz	Gigahertz
GND	*Ground*
IBM	International Business Machines Corporation
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
IoT	*Internet of Things*
IP	*Internet Protocol*
kB	Kilobytes
kHz	Quilohertz
LED	*light-emitting diode*
lm	Lúmens
lux	Luminância
mA	Miliampere
mm	Milímetros
MQTT	*Message Queuing Telemetry Transport*
nm	Nanômetros
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
RISC	*Reduced Instruction Set Computer*
SPI	*Serial Peripheral Interface*
SSID	*Service Set Identifier*
TCP	*Transmission Control Protocol*
THz	Terahertz
UART	*Universal Asynchronous receiver/transmitter*
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*
V	Volts
WLAN	*Wireless Local Area Networking*

WPA *WiFi Protected Access*

SUMÁRIO

1 INTRODUÇÃO

O espaço é uma dimensão física que, em conjunto com os diversos elementos que o influenciam e o transformam, constituem o ambiente em que se vive, como um escritório, quarto, cozinha etc. Um dos fatores mais importantes para a composição de um ambiente é a iluminação, que pode influenciar o usuário psicologicamente e fisiologicamente (ALEXANDRE, 2020).

O avanço constante da tecnologia ligada à comunicação sem fio, possibilita o progresso da *Internet Of Things* (IoT), proporcionando assim, uma maior variedade de dispositivos que são capazes de conectar com a internet e usá-la para comunicação com o usuário e outros dispositivos (VENKATRAMAN; OVERMARS; THONG, 2021).

A iluminação de um ambiente está muito ligada ao dia a dia das pessoas, logo, há uma busca por maior conforto e controle dela. Com o surgimento e desenvolvimento da automação em conjunto com a IoT surgiram dispositivos capazes de funcionar sem a interação humana direta. Junto a isso, a grande disseminação do uso da internet, torna o controle da iluminação uma realidade desejada e possível.

Smartphones e microcontroladores, com fácil acesso e configuração, permitem a fácil personalização dessa automação ao gosto do usuário, conferindo inteligência ao ambiente. Com isso, é possível ter economia de energia, aumentar o conforto e dar funcionalidade ao ambiente, através da automação da iluminação (ALEXANDRE, 2020).

O segundo capítulo busca descrever os principais conceitos ligados a iluminação e sua automatização, para ajudar nos estudos e desenvolvimento.

O terceiro capítulo descreve a solução proposta para o presente trabalho, com o uso da plataforma Arduino e internet sem fio (Wi-Fi) junto à um dispositivo, para automatizar a iluminação.

Conclusão e perspectiva futura descreve os resultados das análises realizadas de forma a finalizar e direcionar o presente trabalho.

Diante deste contexto, este trabalho pretende responder à questão se é possível propiciar melhor qualidade e conforto de um ambiente a partir do controle de sua iluminação.

1.1 Objetivos

Neste tópico são abordados os objetivos gerais e específicos do projeto.

1.1.1 Objetivo geral

Esse trabalho tem como objetivo a compreensão e criação de uma iluminação automatizada residencial de custo acessível, integrada à rede da internet e um ambiente mobile.

1.1.2 Objetivos Específicos

- Estudar os efeitos da iluminação e suas benécias para o ser humano.
- Programar o uso do microcontrolador para controle dos leds.
- Elaborar um circuito para controle dos leds.
- Elaborar uma aplicação mobile para controle da iluminação.
- Criar perfis específicos de usuários para o aplicativo.
- Empregar o uso de rede sem fio (Wi-Fi) para controle do microcontrolador.

1.2 Justificativa

O acesso à automação residencial possui um grande potencial que é pouco disseminado, devido à preconcepção de que há um alto custo e difícil implementação. Entretanto, devido ao crescimento desse ramo, é possível adaptar tecnologias com custo acessível. Agregando isso à IoT, e à ampla disseminação do uso de *Smartphones* e da internet, é possível aumentar o alcance para potenciais novos usuários, cujas necessidades podem ser atendidas pelas vantagens da automação.

O presente trabalho se justifica na busca por melhor acessibilidade a automação da luz residencial, realizando o uso de hardwares e softwares de baixo custo, em conjunto com a internet e ambientes mobile para integrar o ambiente e propiciar conforto e economia, visando a simplificação de ações do dia a dia para as pessoas comuns.

1.3 Métodos

Este trabalho segundo sua natureza, é uma apresentação de uma aplicação da automação para área da iluminação, baseado em teorias já construídas com uma implicação prática em sua realização. Para tal, em seus procedimentos técnicos, foi utilizado de pesquisas bibliográficas e experimental.

Inicialmente foi realizado um estudo sobre o estado da arte de assuntos relacionados à área de iluminação e automação, fornecendo conceitos e definições necessários para a compreensão do tema, utilizando fontes confiáveis de artigos, livros, apostilas e periódicos.

A seguir foi montado um circuito com os hardwares Arduino UNO ligado a LEDs, e programado no Arduino IDE para controle da iluminação. Logo após, foi aplicado um software mobile e web para IoT com conexão Wi-Fi com o intuito de controlar remotamente o Arduino.

1.4 Cronograma

No presente trabalho o cronograma é apresentado em duas fases, sendo elas:

Fase 1:

- Definição do conteúdo
- Pesquisa bibliográfica.
- Revisão do conteúdo pesquisado.

Fase 2:

- Montagem do circuito Arduino.
- Programação do IDE Arduino.
- Configuração da interface Homem-Máquina
- Implementação do sistema.
- Análise e interpretação dos resultados obtidos.
- Escrita dos resultados em uma monografia.

Tabela 1 - Cronograma do trabalho

	2022										
Atividade	**JAN**	**FEV**	**MAR**	**ABR**	**MAI**	**JUN**	**JUL**	**AGO**	**SET**	**OUT**	**NOV**
Definição do conteúdo	X	X	X	X	X	X	X	X	X	X	
Pesquisa bibliográfica		X	X	X	X	X	X	X	X	X	
Revisão do conteúdo pesquisado			X	X	X	X	X	X	X	X	
Montagem do circuito Arduino						X	X	X	X		
Programação do IDE Arduino							X	X	X		
Configuração da Interface Homem-Máquina							X	X	X	X	
Implementação do sistema.					X	X	X	X	X	X	X
Análise e interpretação dos resultados						X	X	X	X	X	X
Escrita da análise dos resultados em uma monografia.			X	X	X	X	X	X	X	X	X

Fonte: Elaborado pelo Autor, 2022

1.5 Resultados Obtidos

Pretende-se, com os resultados previstos para este trabalho, usar a iluminação e a automação do ambiente residencial e sua interação com o usuário para melhorar sua qualidade do conforto e da estética, utilizando da conexão com um ambiente mobile e a internet, de forma acessível e difundindo assim, o uso da automação, auxiliando os estudos e avanços da IoT.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo abrange o referencial teórico relevante e necessário para a composição desse trabalho, através de conceitos e definições utilizadas para melhor compreensão dessa aplicação.

2.1 Iluminação

Na criação de um ambiente, existem vários fatores que o valorizam e tornam a experiência e a estadia nele mais agradável, em que a luz é um dos mais importantes. A percepção humana do espaço é exercida, em sua maioria, através da luz, portanto, uma iluminação bem executada pode proporcionar uma estética adequada, assim como conforto e funcionalidade ao ambiente. Deste modo, com certas cores e intensidades, é possível ter funcionalidade na luz e consequentemente criar ambientes especializados para variadas situações (MANAIA, 2010).

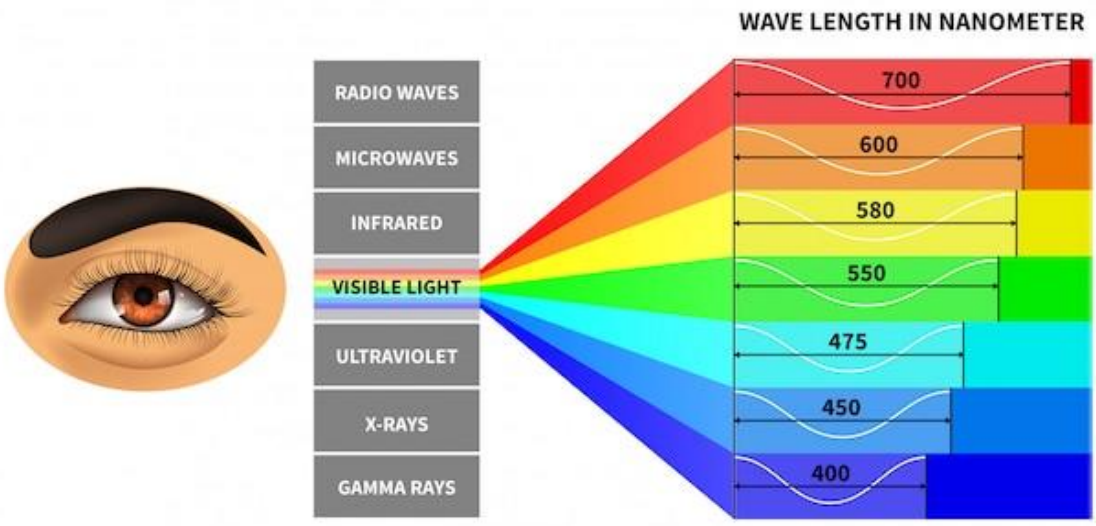
Britannica (2017) define iluminação como o uso de uma fonte de luz para iluminar um ambiente. Quando em um ambiente residencial, essa iluminação será realizada com luz artificial com o uso de LEDs ou lâmpadas. Desta forma, é necessário considerar e analisar os fatores que influenciam essa iluminação, como o fluxo luminoso e a iluminância.

A iluminância representa a intensidade de luz da superfície dividida pela sua área aparente, que, dependendo também das características de reflexão da superfície, cria a sensação de claridade provocada no olho, podendo ser medida com um luxímetro (unidade: lux) (MARTINS; BELENKI; SANCHES,2019). Já o fluxo luminoso representa a radiação total emitida por uma fonte luminosa, por segundo, em todas as direções, produzindo estímulos visuais através da luz, com unidade de medida em lúmens(lm) (LOSS, 2013).

2.1.1 Classificação das cores

As cores, como conhecidas, são percepções visuais transmitidas para o cérebro, sendo elas o efeito da luz na retina produzido após ela ser refletida ou emitida de um objeto, assim, na incidência da luz sobre um objeto ela será refletida em diferentes comprimentos de onda do espectro eletromagnético. Em outros termos, é a sensação provocada pela ação da luz sobre o órgão da visão (RAMBAUSKE, s.d., p.70). Sua manifestação é condicionada à luz (objeto físico, agindo como estímulo) e o olho (aparelho receptor, funcionado como decifrador do fluxo luminoso) (RAMBAUSKE, s.d., p.70). O olho humano consegue apenas enxergar um limitado espectro da luz, o chamado espectro visível, que se encontra entre o infravermelho e o ultravioleta.

Figura 1 - Espectro eletromagnético

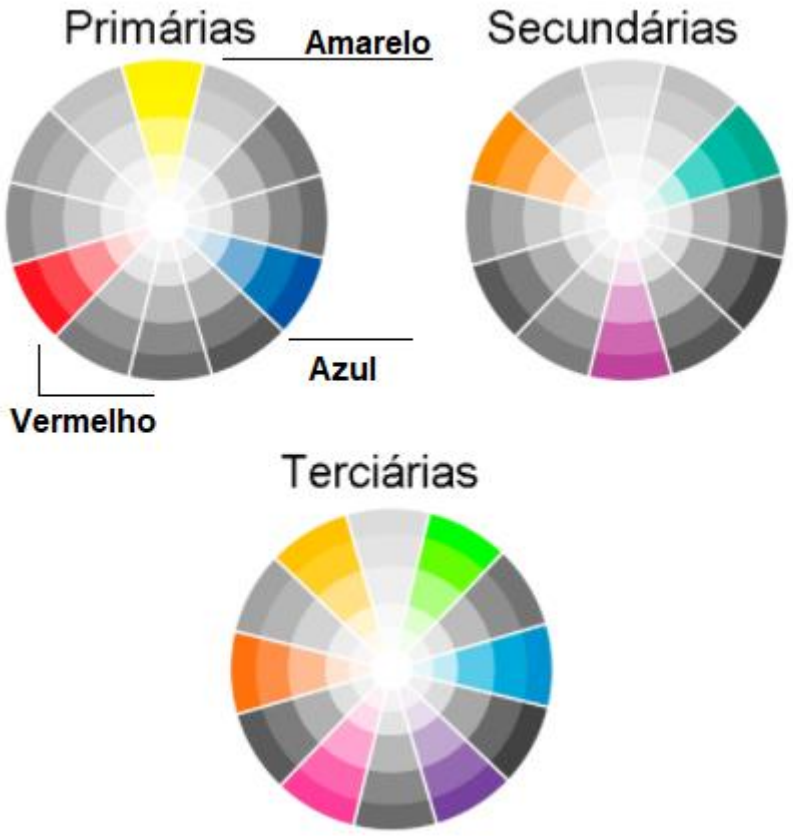


Fonte: Freepik, 2022

As cores do espectro visível, são representadas pelo círculo cromático que é composto por cores primárias, ou seja, cores que não podem ser obtidas mediante mistura de outras cores, sendo elas o amarelo, vermelho e azul, cores secundárias, que são criadas a partir da mistura de duas cores primárias e as terciárias, feitas a partir da mistura das secundárias. (SOARES, 2019)

O círculo cromático é retratado na Figura 2.

Figura 2 - Círculo cromático



Fonte: Claudio Soares, 2019

As cores possuem três propriedades que nos permite diferenciá-las, o Matiz, a saturação e o brilho. O Matiz se refere à cor em seu estado puro que ligada ao seu comprimento de onda gera a sensação de tom da cor. A saturação diz respeito a proporção de cinza no matiz da cor, com isso, se diz que uma cor é pura quando ela tem o maior grau de saturação. O brilho é a intensidade que a cor possui, influenciando a quantidade de energia física usada na luz, dizendo respeito à intensidade do brilho da luz sobre a superfície. (SOARES, 2019)

A figura 3 retrata as propriedades das cores.

Figura 3 – Propriedades das cores

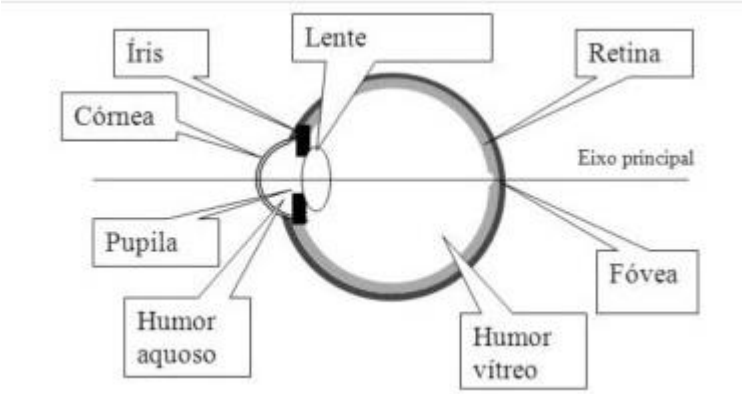
Fonte: Claudio Soares, 2019

2.1.2 Olho humano

O globo ocular humano é formado pela córnea, que é uma lente transparente, pela íris, responsável pelo controle da abertura da entrada luminosa, pela pupila, a lente antigamente denominada cristalino, e pela retina que se encontra no fundo do olho. Sua região interna é preenchida por dois materiais transparentes. Entre a córnea e a lente há um líquido chamado humor aquoso, e após a lente continua o outro líquido até a retina, denominado humor vítreo. Ambos os humores têm índices de refração de 1,34, próximos da água que é 1,33. O globo ocular é uma elipse quase esférica, com raio perto de 1,2 cm. (HELENE, O.; HELENE, A, 2011).

A estrutura simplificada do olho é apresentada na figura 4.

Figura 4 - Componentes ópticos do olho humano



Fonte: HELENE, O.; HELENE, A. 2011.

A pupila é uma pequena abertura, logo após a córnea, onde ocorre a entrada da luz, podendo ter diâmetro variável de 2mm a 6mm, dependendo da iluminação.

A íris é uma estrutura circular cuja cor da parte externa pode variar (marrom, azul ou verde são as cores mais comuns), e que controla a abertura da pupila, através da musculatura da própria íris. (HELENE, O.; HELENE, A. 2011).

A córnea é um tecido transparente que protege a entrada do olho, além de funcionar como uma lente, ajudando moderadamente no foco da luz. Ela possui um índice de refração da ordem de 1,38 e um raio de curvatura por volta de 0,8cm na parte interior e 0,65cm na parte exterior. Além disso, sua parte central possui 0,06cm de espessura (HELENE, O.; HELENE, A. 2011).

A lente interior do olho, também conhecida como cristalino, tem raios de curvatura variáveis, já que precisa mudar de foco dependendo da distância das imagens. O raio de curvatura anterior é próximo de 0,8cm e o posterior de 0,65cm e seu índice de refração é de 1,42 (HELENE, O.; HELENE, A. 2011).

A retina do olho é composta por células fotorreceptoras: bastonetes, que reconhecem a presença ou ausência de luz e alguns tons intermediários, e cones, que percebem as cores (NISHIDA, 2012).

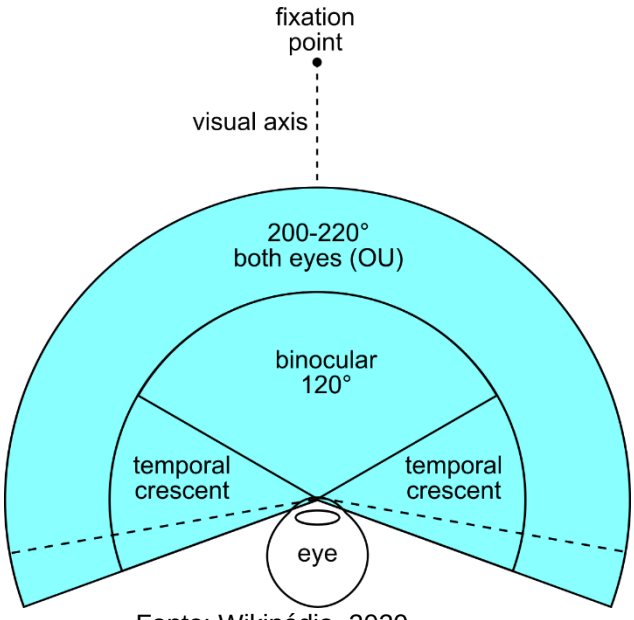
A imagem deve estar sobre a retina para ser formada pelo sistema óptico do olho, em especial em uma região muito densa de células sensíveis à luz, a fóvea. (HELENE, O.; HELENE, A. 2011). O nervo óptico é responsável por transportar as informações visuais para o cérebro (NISHIDA, 2012).

A estrutura ocular, tem uma restrição no que é visível devido a composição das partes externas, assim, a extensão perceptível da visão (quando o olho está parado) é denominada campo de visão (NISHIDA, 2012). No caso do olho humano, o campo de visão vertical e horizontal, é composto pelo ângulo ótimo ou central, e o ângulo total ou limite (FIELD, 2020).

O ângulo central é responsável por delimitar a visão binocular humana, na qual é possível distinguir as imagens observadas, sendo ele 120 graus na horizontal e 60 graus na vertical (NISHIDA, 2012). Já o ângulo total é a delimitação da visão total humana, no qual, após o ângulo central, há a visão periférica, que é a extensão máxima do campo de visão, em que a presença e movimento das imagens são percebidos, mas são pouco nítidos e sem discriminação cromática. A visão periférica horizontal varia de 40 a 50 graus na esquerda e direita, já a visão periférica vertical varia de 25 a 30 graus em cima e 30 a 40 graus em baixo (FIELD, 2020).

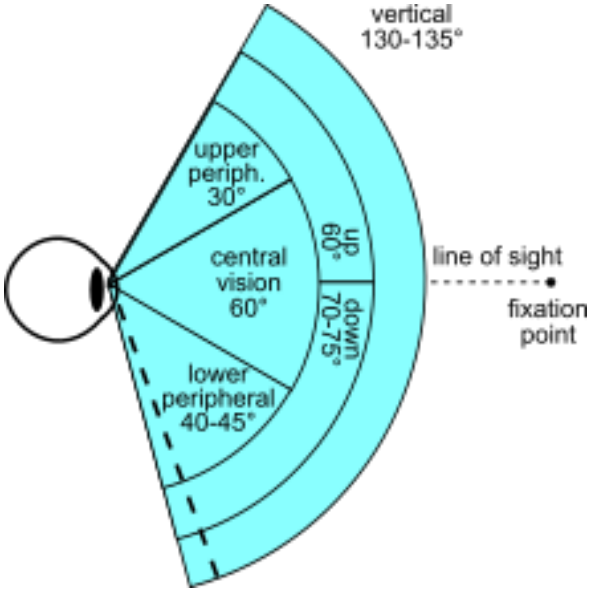
O campo de visão do olho humano é representado pelas figuras 5 e 6.

Figura 5 - Campo de visão horizontal



Fonte: Wikipédia, 2020

Figura 6 - Campo de visão vertical



Fonte: Wikipédia, 2020

2.1.4 Efeitos psicológicos da luz

A teoria ou psicologia das cores é o estudo sobre as cores e sua influência no comportamento do ser humano, fazendo o uso dela para transmitir sensações e propósitos (RAMBAUSKE, s.d., p.59).

Foram constatados certos padrões de comportamento e de emoções na presença de cores específicas, portanto, o impacto psicológico da luz está ligado à sua cor ou comprimento onda. Agregado a isso, é possível transmitir sensações de frio e calor. Cores quentes são mais estimulantes e ligadas a emoções fortes, como o vermelho, o laranja e o amarelo (MENEZES, 2020).

O vermelho é relacionado ao amor e a raiva, ao calor e ao fogo, sendo assim, ela é considerada a cor dominante em relação as atitudes positivas e a um ambiente mais vívido (HELLER, 2007).

Já o amarelo, além de ser a cor mais associada à iluminação, é ligado ao intelecto e a mente, representando otimismo e espontaneidade, ideal para ambientes de criatividade e inovação (HELLER, 2007).

O laranja é a junção do vermelho com o laranja, e sua relação é com emoções fortes e otimistas, criando um ambiente de encorajamento e autoconfiança (HELLER, 2007).

Cores frias possuem propriedades mais suaves, sendo ligadas à natureza e à calma, como o verde, azul e o roxo (MENEZES, 2020).

O azul é relacionado à paz e à tranquilidade, ampliando assim o senso de tempo e até mesmo a produtividade, ideal para ambientes de trabalho como um escritório (HELLER, 2007).

O verde é a cor ligada a natureza, assim ela é relacionada ao equilíbrio e o amor à natureza, transformando o ambiente em um lugar calmo e relaxante (HELLER, 2007).

O roxo por sua vez é associado à vaidade e à riqueza, podendo criar ambientes bons para demostrar fartura e abundância (HELLER, 2007).

Além disso, há também o branco, que é relacionada à simplicidade e à igualdade e até mesmo à perfeição e proporciona um ambiente equilibrado e simples, sem muito destaque e emoção (HELLER, 2007).

2.1.5 Efeitos fisiológicos da luz nos olhos

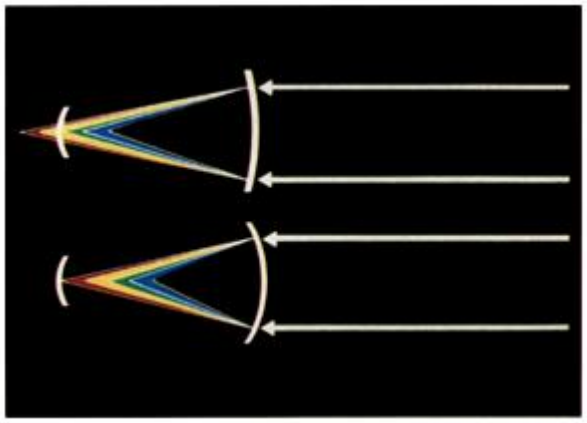
Junto da ação psicológica, existe também a ação fisiológica que a luz tem sobre o indivíduo, demostrada na interação de diferentes comprimentos de onda com o sistema nervoso, juntamente com as partes componentes dos olhos.

Nos cones do olho, há três tipos de células fotorreceptoras, sendo elas sensíveis a determinados comprimentos. Assim, cada uma consegue identificar cores baseadas em seus comprimentos de onda: Verde (530 nm - 540 nm), azul (430 nm - 440 nm) e vermelho (560 nm – 580 nm) (Murch, 1984). Logo, uma luz com 470 nm será formada como um azul forte com alguns componentes mínimos de verde e vermelho. Cada cone possui um dos três tipos de células, entretanto, elas não são distribuídas uniformemente, já que 64% dos cones contêm o pigmento vermelho, 32% contêm o pigmento verde, e apenas 2% possuem o pigmento azul (Murch, 1984).

A lente do olho é utilizada para a focar a luz externa na retina, tal como uma lente usada em óculos ou quaisquer dispositivos ópticos. Deste modo, diferentes comprimentos de onda possuem diferentes distâncias de foco. Assim, para diferentes matizes deve se ajustar o cristalino para que a luz seja focada corretamente. Para a lente humana, comprimentos de onda maiores terão uma distância focal maior, ou seja, o vermelho tem a maior distância focal e o azul a menor (Murch, 1984).

A figura 5 apresenta como a lente humana se adapta a diferentes comprimentos de onda.

Figura 7 - Foco da lente com diferentes comprimentos de onda



Fonte: Much, 1984

No caso de uma alternância frequente da luz entre vermelho e azul, a lente ficará em constante mudança para ajustar a distância focal, e isso poderá causar um cansaço visual. Essa lente também absorve luz, sendo que o azul do espectro da cor é duas vezes mais absorvido, o que resulta em uma sensibilidade maior a maiores comprimentos de onda e menor a menores comprimentos de onda, e isso se agrava com a idade. Assim, ao envelhecer, a lente fica amarelada, filtrando mais cores de menor comprimento de onda, o que faz com que maiores comprimentos de ondas sejam ainda mais absorvidos, logo, quanto maior a idade, menor a sensibilidade a menores comprimentos (ciano a azul) (G. Murch,1984).

2.2 Automação

Antigamente, automação era definida como uma máquina capaz de realizar uma série de operações lógicas e de controle, sem a intervenção humana (AUTOMAÇÃO, 2022). Atualmente ela se tornou um campo de conhecimento.

Esse conceito foi elaborado quando se percebeu a necessidade de modernizar os processos operacionais, com o objetivo de agilizar e melhorar a capacidade de diversos processos. A automação é subdividida em automação industrial, residencial ou domótica e comercial, devido à grande expansão apresentada por essa área de conhecimento ao longo do tempo.

2.2.1 Automação industrial

A automação, como diversos outros recursos que hoje são desenvolvidos e inseridos no dia a dia, foi pensada inicialmente para a indústria. De acordo com Araújo Júnior et al (2003), as estratégias de automatização de produção criadas pelas indústrias eram um meio de aumentar sua competitividade, tanto em termos de custo e disponibilidade quanto em termos de inovação e qualidade, buscando assim, produzir um produto melhor com menor custo e mais agilidade na produção.

A automação de um sistema industrial se refere ao meio pelo qual as operações realizadas por funcionários humanos são substituídas por um conjunto de elementos tecnológicos que não precisam de intervenção humana, com exceção da programação, de reparos e inspeções (SILVEIRA, 2011).

As motivações da automatização da indústria estão atreladas à sua linha de produção, tendo em vista que, com o uso da automação pode-se obter um aumento na qualidade dos produtos, pela maior precisão da máquina, além de uma redução nos custos de pessoal, já que ocorre a substituição da mão de obra humana (ARAÚJO JÚNIOR et al.,2003). Junto a isso, é possível melhorar as condições de trabalho dos funcionários, por colocar a máquina para realizar trabalhos de alto risco (SILVEIRA, 2011).

2.2.2 Automação comercial

Após já estar bastante difundida no meio industrial, a automação começou a ser implementada em outros meios, sendo o primeiro deles o setor comercial. Com objetivos similares a automação industrial, de aumentar a eficiência das operações e reduzir os custos de operação, com a diferença de também querer aumentar a satisfação do cliente (FAGUNDES, 2020).

A integração do comércio com os processos de automação permite ao comerciante melhorar o controle de estoques, os pedidos de compra e até mesmo as obrigações fiscais, trazendo assim uma maior flexibilidade e controle sobre seus produtos e loja. Além disso, essa integração traz rapidez e conveniência no atendimento, tornando a estadia em lojas físicas mais agradáveis e sites de compras se tornam mais fáceis de se usar (FAGUNDES, 2020).

2.2.3 Domótica

Gerenciar o controle de acesso, de comunicação e iluminação por meio de um sistema automatizado proporciona, além de um esperado aumento na produtividade, um conforto e até mesmo segurança para o ambiente utilizado (DIAS, PIZZOLATO, 2004). A partir disso, foi notado que o uso da automação vai além de apenas aumentar a produtividade, conseguindo dar conforto e controle sobre o ambiente.

Com isso, surgiu a domótica, um termo que caracteriza o uso e integração de mecanismos automatizados no meio residencial. Advindo do francês *Domotique* que é a junção de *Domus* “casa” e *Immotique* “automático” (ELIENE, 2011).

O conforto e a praticidade derivados do avanço tecnológico aplicados na domótica concedem ao usuário uma maior comodidade dentro de sua própria residência (METTEDE, 2020). Assim, é possível ter um controle sobre a operação dos recursos de uma casa, como no caso do uso de um software mobile.

O ponto negativo na aplicação da automação residencial, é a necessidade de um projeto com investimento de alto custo para implementá-la (ELIENE, 2011). Entretanto, a criação de uma rede integrada, além de fornecer conforto e praticidade, também valoriza o imóvel, aumenta sua segurança, e aumenta o controle e supervisão sobre ele (METTEDE, 2020), sendo assim, é um investimento de longo prazo para a melhoria de qualidade de vida.

Entre os variados usos e aplicações da automação residencial estão:

- Controlar as luzes, como tonalidade, cor e intensidade.
- Controlar temperatura ambiente
- Desligar luzes por controle, seja por meio de aparelho ou até de voz
- Gestão do gasto de energia

2.3 *Internet of things*

É um momento que se vive na época da internet e da conexão, na qual tecnologias são amplamente difundidas e criadas. Isso caracteriza uma situação em que os aparelhos também se conectam à internet, criando uma forma de comunicação entre pessoas e coisas (BANDYOPADHYAY; SEN, 2011).

De acordo com Bharti, Saxena e Kumar (2017), o termo *Internet of things* se refere a recursos heterogêneos, os quais percebem, processam e interpretam informações via estruturas conectadas à internet. Esses recursos podem consistir em softwares ou hardwares. Com esse conceito é possível pensar em aplicações de casas inteligentes, carros inteligentes, até mesmo serviços inteligentes como o de assistência médica (SEN, 2018).

A IoT é uma área ainda emergente, mas com vastas áreas de estudo e aplicações que facilitariam a troca de dados, a automatização de sistemas e comunicações entre sistemas (BHARTI; SAXENA; KUMAR, 2017).

2.3.1 conectividade

A conectividade é representada pelo uso de tecnologias com e sem fio para fins de interconectar dados, informações e de uma forma geral indivíduos.

2.3.1.1 Bluetooth

A tecnologia Bluetooth foi pensada quando, após a invenção dos telefones celulares, foi constatada a necessidade de uma comunicação sem fio entre os telefones móveis e outros dispositivos. Sua primeira versão foi publicada em 1999, por um grupo de companhias telefônicas: Ericsson, IBM, Nokia e Toshiba (AGARWAL, 2021). A partir deste momento, essa tecnologia foi mais desenvolvida e abrangendo novos aparelhos, como computadores, fones e outros dispositivos de rede.

A primeira versão do Bluetooth tinha uma frequência de transmissão de dados de 1Mbps. As últimas versões, 4 e 5, possuem uma banda de frequência de 2,45 GHz, um aumento de mais de mil vezes (Alecrim, 2021).

A rede de conexão Bluetooth consiste em um conjunto de dispositivos, contendo no mínimo dois e um máximo de oito dispositivos conectados via Bluetooth. Essa rede é composta por um dispositivo mestre, sendo esse o que inicia a comunicação, e até oito dispositivos escravos, que respondem ao dispositivo mestre após a solicitação para sincronizarem. (AGARWAL, 2021).

O Bluetooth é utilizado para troca de informações que podem ser documentos, arquivos e sua vantagem está ligada ao pouco uso de energia para seu funcionamento (Alecrim, 2021).

Em relação à sua segurança, o Bluetooth necessita de medidas externas dos aparelhos para controlar seu acesso, caso contrário o seu acesso é irrestrito por qualquer outro dispositivo com a mesma frequência (AGARWAL, 2021). Logo, em dispositivos que possuem a tecnologia Bluetooth, também são implementadas medidas de restrição, como a certificação da conexão, limitação de dispositivos pareados e, até mesmo, limitar as atividades permitidas durante a conexão (Zoom, 2021).

Junto à IoT, o Bluetooth começou a ser implementado em diversos aparelhos, englobando fones de ouvido, impressoras, câmeras, teclados, gps, até mesmo aparelhos eletrodomésticos mais sofisticados, tudo isso a fim de melhorar e facilitar o acesso aos aparelhos e suas informações e funções.

2.3.1.2 Wi-Fi

Wi-Fi é o termo comercial para o IEEE 802.11 conhecido como *wireless local area networking* (WLAN), uma tecnologia de rede sem fio amplamente utilizada na IoT, que foi criado para prover outra forma de conexão à internet, que, sem o uso de cabos, concedesse aos dispositivos, o acesso à internet (PAHLAVAN; KRISHNAMURTHY, 2021).

Na década de 80, quando os computadores pessoais começaram a ser comercializados e popularizados nas indústrias, surgiu a necessidade de melhorar a conexão entre os computadores com um esforço mínimo. Assim, surgiu as primeiras implementações da tecnologia WLAN, com o uso de um *modem* primitivo. Com o passar do tempo, essa tecnologia foi implementada em escritórios, e por volta dos anos 2000, começou a adentrar o ambiente residencial (PAHLAVAN; KRISHNAMURTHY, 2021). A aplicação do Wi-Fi para o uso pessoal começou com os celulares e laptops, mas agora também é usado extensivamente em outros dispositivos como eletrodomésticos e automóveis (AGARWAL, 2020), permitindo assim a implementação da IoT.

Inicialmente, essa tecnologia utilizava uma frequência de 2,4 GHz para comunicação, entretanto, com o avanço tecnológico, ela foi expandida para 5 GHz que possui uma transmissão de dados mais rápida. Além disso, a frequência 6 GHz já está em desenvolvimento (SHAW, 2020).

O Wi-Fi é um meio de transmissão de internet e rede sem fio, que usa ondas de rádio para transmitir dados de um dispositivo do usuário para um ponto de acesso (roteador) ou para outro usuário (SHAW, 2020). Devido a isso, o alcance dessas ondas é dependente da estruturação do ambiente em que se encontra, já que superfícies como paredes podem diminuir esse alcance, e outros aparelhos como o micro-ondas podem interferir nas ondas (AGARWAL, 2020).

Para a implementação de uma rede Wi-Fi, é necessário que o dispositivo tenha um adaptador Wi-Fi para conseguir fazer a comunicação, e um ponto de acesso remoto como um modem e um roteador, que se conecta à rede e distribui o sinal de internet a outros dispositivos (SHAW, 2020).

Apesar de ter vantagens como o não uso de cabos e simplicidade de conexão, sua implementação é mais cara. Além disso, sua proteção deve ser bem implementada, com o uso de firewalls, e protocolos de segurança, como o WPA3, que implementa o uso de senhas criptografadas (SHAW, 2020).

2.3.1.4 Infravermelho

O infravermelho é um tipo de radiação eletromagnética, com frequência menor que a luz vermelha, assim, não faz parte da parte visível do espectro eletromagnético (Figura 1) (JÚNIOR, 2022). Ela possui um comprimento de onda na faixa de 700 nm até 1mm e uma frequência na faixa de 300GHz a 430THz, sendo largamente usada para detectar temperatura em objetos, como em fotografias térmicas (*COMMUNICATION*, 2017).

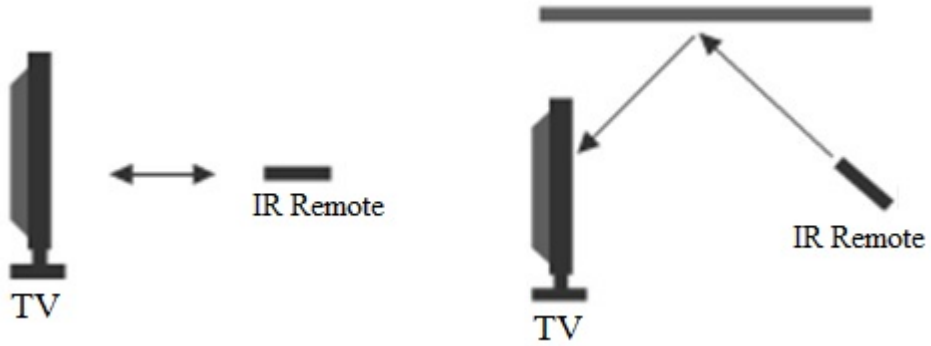
Outro uso importante do infravermelho é sua implementação em um sistema de comunicação entre dispositivos, em que há dois tipos de comunicação infravermelha, ponto a ponto e ponto difuso.

A base da comunicação infravermelha é formada por um transmissor, que é o dispositivo responsável por emitir uma luz infravermelha a cada pulso elétrico ou informação, como o botão de um controle remoto sendo apertado, e um receptor como em uma televisão, ou quaisquer dispositivos que contenham um fotodetector, gerando um sinal elétrico quando acionado. Para essa comunicação funcionar sem interferência de outras fontes de infravermelho, o transmissor trabalha com sinais modulados de frequência entre 36 e 46 kHz, chamada frequência portadora, e o receptor possui um filtro que descarta todas as frequências acima ou abaixo da portadora (*COMMUNICATION*, 2017).

A comunicação ponto a ponto requer uma linha de visão entre os dispositivos para que se comuniquem, não pode existir, entre o transmissor e o receptor do sinal, quaisquer obstáculos. Já na comunicação difusa, pode haver obstáculos entre o receptor e o transmissor, dado que a comunicação entre eles é mantida pela reflexão ou ricocheteamento pelas superfícies, como o teto e as paredes (*COMMUNICATION*, 2017).

Ambos os tipos de transmissão são apresentados na figura 8.

Figura 8 - Tipos de comunicação infravermelha



Fonte: rfwireless, 2012

2.4 Microcontrolador

Os microcontroladores são comumente empregados em sistemas embarcados, isto é, sistemas computacionais que consigam atender uma demanda específica (IEEE, 2020). Eles têm a vantagem de serem programáveis, logo é possível atribuir dados e algoritmos a sistemas e, assim, ter total controle sobre eles.

Sua composição pode ser considerada um pequeno computador, com memórias voláteis e não voláteis, entradas e saídas de dados e, um microprocessador, que é um chip programável (IEEE, 2020). Seu propósito é controlar dispositivos eletrônicos a partir de algoritmos programados no microcontrolador, de forma automática, ou seja, sem a intervenção humana. Seu uso varia de sensores, como o ultrassônico, a LEDs e atuadores como os elétricos ou pneumáticos (PRÓSPERO, 2020).

2.4.1 Arduino

Para este projeto, foi utilizado o Arduíno, por sua flexibilidade e simplicidade, além de ser *open source*, ou seja, de código aberto.

O Arduino é uma plataforma aberta de prototipação eletrônica baseada em softwares e hardware de fácil uso (*INTRODUCTION*, 2018). O objetivo dessa plataforma é permitir que qualquer pessoa consiga melhorar sua qualidade de vida a partir de uma tecnologia eletrônica simples e poderosa por um preço acessível (*ABOUT*, 2021).

A plataforma Arduino é composta por um ambiente de desenvolvimento (software) e uma placa controladora ou microcontrolador (hardware). Por possuir um software de fácil compreensão para usuários iniciantes e com uma complexidade alta para usuários avançados, o Arduino é utilizado para facilitar prototipagens e emulação de sistemas interativos residenciais e comerciais (ARDUINO, 2022).

O uso do Arduino oferece diversas vantagens em relação a outros microcontroladores. Seu software é multiplataforma, funcionando em Linux, Windows e Mac, além disso, por ser *open source,* é possível expandi-lo com o uso de bibliotecas C++. Já seu hardware é barato e acessível, e por também ser *open source* é possível expandi-lo com módulos personalizados (*ABOUT*, 2021).

Existe uma grande variedade de produtos Arduino, abrangendo diversas demandas possíveis, partindo de hardwares de nível de entrada, como o Arduino *Nano*, que são usados para projetos simples. A partir disso, produtos com recursos avançados, como o Arduino *Due*, para projetos com funcionalidades complexas e avançadas ou performances mais rápidas. Além disso, também existem produtos com propósitos mais específicos, como o Arduino MKR1000, sendo relacionado à IoT, possuindo conexão com a Arduino *Cloud* e permitindo a coleta de dados e uso de aplicativos mobile através de uma conexão à internet (*PRODUCT*, 2018).

3 PROPOSTA DE SOLUÇÃO

Este capítulo vai descrever a proposta de solução.

3.1 Componentes de Hardware

Neste item serão demonstrados os componentes físicos usados na composição da prototipação deste trabalho.

3.1.1 Arduino Uno

Para este projeto, a plataforma selecionada é o Arduino UNO, sendo um arranjo simples e completo, com conectividade USB e programação acessível, possuindo pequenas dimensões, medindo 2,7'' x 2,1'', que equivalem a 68,6 mm por 53,4 mm e pesando 25 g, sendo composta pela interface de 14 pinos digitais e 6 pinos analógicos para entrada e saída de dados, 7 pinos de alimentação para *Shields* (expansões) e módulos, além de uma conexão USB e uma conexão de energia. Para a alimentação externa da placa, são necessários de 7 a 12 V de energia, pois apresenta um regular de tensão integrado, e a tensão de funcionamento interno é de 5V (ARDUINO UNO, 2021). Essa placa não possui um botão de ligar e desligar, assim para interromper seu ciclo de trabalho é necessário a desconexão do cabo da fonte de energia.

Seu componente principal é o microcontrolador ATMEL ATMEGA328, que possui 32 kB de memória flash, 2 kB de memória RAM (memória de acesso aleatório) e 1 kB de EEPROM (*Electrically Erasable Programmable Read-Only Memory*). A placa Arduino UNO opera em 16 MHz, conta com um comparador analógico interno, alguns *timers* e 6 PWMs (Modulação por Largura de Pulso) (SOUZA, 2013).

Para os pinos digitais, que são os pinos de 0 a 13, sua operação é feita em 5V, sendo que cada pino pode propiciar ou receber até 40 mA de corrente. Os pinos 3,5,6,9,10 e 11 são pinos que podem ser usados como PWMs de 8 bits (função *analogWrite* ()). Já os pinos 0 e 1 podem servir para a comunicação serial, e os 2 e 3 podem ser configurados para interrupção externa (função *attachInterrupt* ()). Os pinos analógicos, A0 a A5, possuem resolução de 10 bits (SOUZA, 2013).

Os outros pinos são os conectores de alimentação para módulos e *Shields*, que quando ligados à placa Arduino, ampliam ou adicionam funcionalidades. São eles, o pino IOREF responsável por fornecer tensão de referência para *Shields*, que funcionam em 3,3V, e possam funcionar em 5V. O pino RESET, utilizado para um reset externo do Arduino. Os pinos 3,3V e 5V que fornecem essa tensão elétrica para *Shields* e módulos externos, sendo que o 3,3V possui corrente máxima de 50 mA. Os pinos GND que são pinos de aterramento e o pino VIN, com o propósito de alimentar a placa Arduino por meio dos *Shields* ou módulos externos (SOUZA, 2013).

Os pinos desta placa são apresentados na figura 9.

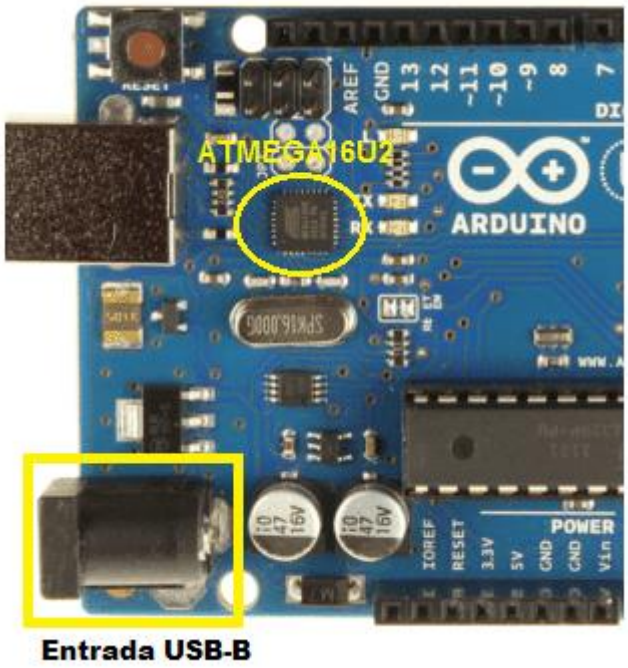
Figura 9 - Pinos da placa Arduino UNO

Fonte: Souza, 2013

Para a comunicação do Arduino com o computador é usado a entrada USB em conjunto com o microcontrolador ATMEL ATMEGA16U2. Essa conexão permite o upload do código compilado para o Arduino (SOUZA, 2013).

O microcontrolador e a entrada USB-B do Arduino UNO são apresentados na figura 10.

Figura 10 - Microcontrolador USB e entrada USB-B do Arduino UNO



Fonte: Souza, 2013

3.1.2 Módulo Wi-Fi ESP8266 Versão ESP-01

Os módulos ESP8266 são alguns dos vários módulos existentes que surgiram para explorar a IoT e conectar sistemas a internet. Para tanto, esses módulos contam com uma arquitetura RISC de 32 bits e um SoC (*System-on-a-chip* ou sistema em um *chip*) com Wi-Fi embutido. Além disso, esses modelos também possuem conexão serial, entradas ADC, saída PWM e sensor interno de temperatura (CURVELLO, 2015)

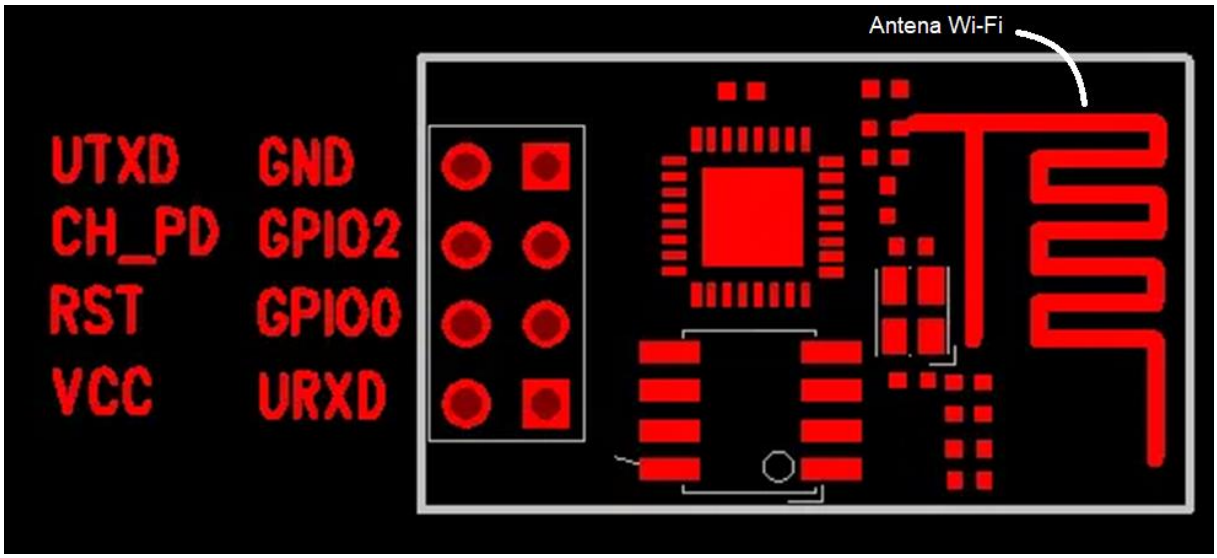
Existem, oficialmente, 12 modelos desse módulo, numerados de ESP-01 a ESP-12. Para o protótipo do trabalho, será usado o módulo ESP-01. Esse módulo, junto ao módulo ESP-10 são utilizados para como ponto serial Wi-Fi. Em outras palavras, eles interagem com a rede Wi-Fi por meio de conexões TCP/UDP após receberem comandos via UART(Serial) (CURVELLO, 2015).

A versão ESP-01 conta com os padrões de rede IEEE 802,11b, 802,11g e 802,11n, empregando a rede Wi-Fi em uma frequência de 2,4 GHz e suportando os padrões de segurança WPA e WPA2.Também possui consumo baixo de energia, um alcance de 90 metros aproximadamente e comunicação via UART, o que permite a interação com outros microcontroladores, como o Arduino UNO (MODULO, 2017).

Possui dimensões pequenas, de 24 mm de comprimento, 14 mm de largura e 3 mm de altura e pesa 7 g, possui uma alimentação de 3,3 V e corrente de 80 mA, uma memória RAM de 50 KB aproximadamente e uma interface serial UART (Tx e Rx) (MODULO, 2017). Por ser pequeno e leve, seu manuseio é facilitado.

O ESP-01 e sua pinagem é apresentado na figura 11:

Figura 11 - ESP-01 e sua pinagem



Fonte: Curvello,2015

O pino GND serve para o sinal de terra, o VCC para a tensão de alimentação, os pinos Rx e Tx são os sinais de módulos a serem conectados respectivamente aos Tx e Rx do microcontrolador conectado a ele e o pino de reset é o RST. Para entrada e saída de dados há os pinos GPIO0 e GPIO2, que são controlados pelo firmware, sendo que o firmware é gravado utilizando o pino CH_PD (CURVELLO, 2015).

Esse módulo é programado para servir de ponto de acesso, recebendo e enviando dados do microcontrolador Arduino para o programa na internet com uma conexão Wi-Fi.

3.1.3 Módulo alternativo:NodeMCU-32S ESP32

Como uma alternativa ao uso de módulos separados, o ESP32, assim como seu antecessor o ESP8266, o módulo ESP32, visa possibilitar e facilitar os avanços da IoT. Este modelo tem grandes vantagens por ter mais recursos, mas sua aquisição necessita de um investimento maior. Esse dispositivo tem uma maior independência quando comparada com os módulos antigos, possuindo diversos pinos, conexão Wi-Fi e Bluetooth, conseguindo (OLIVEIRA,2017).

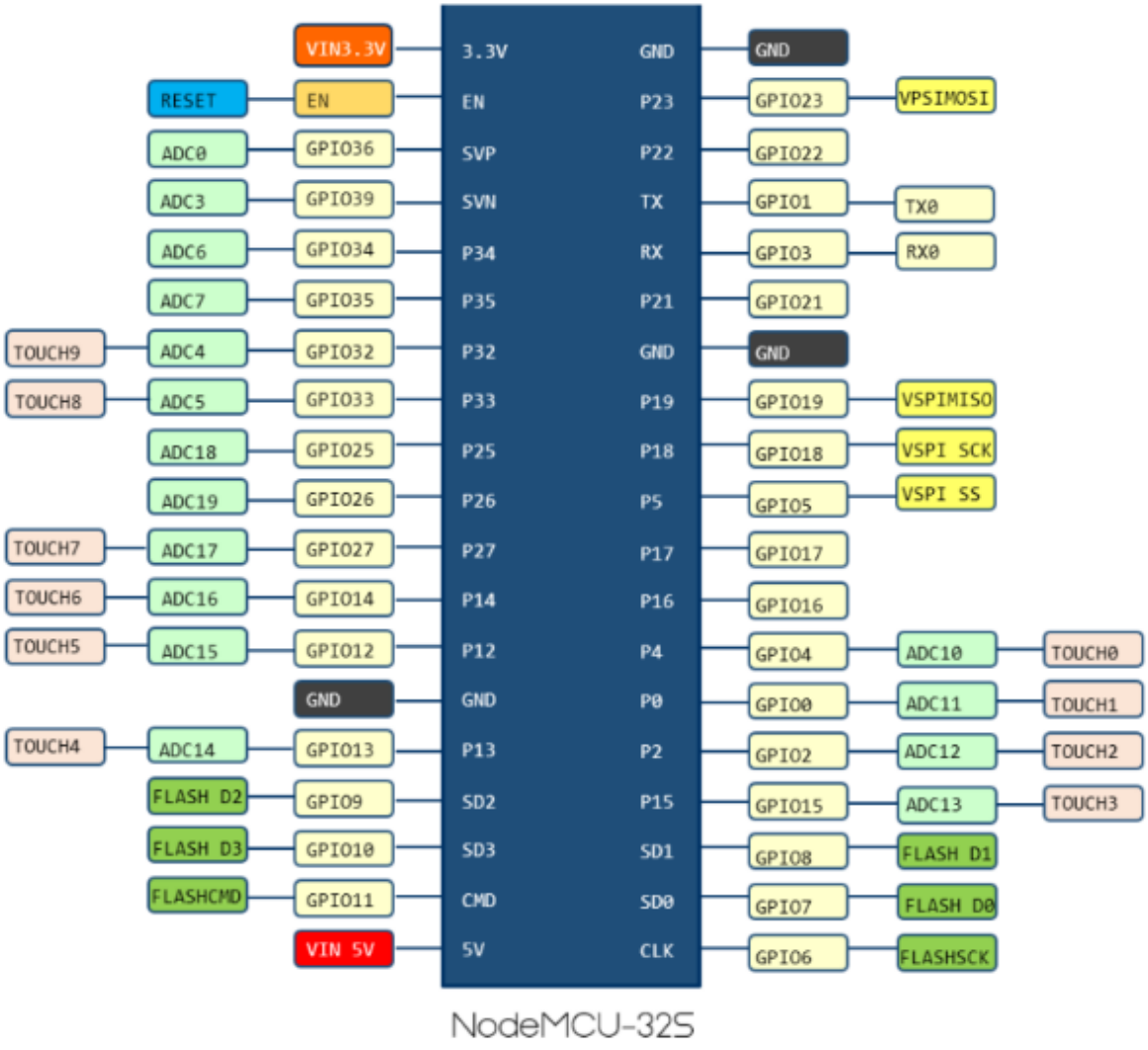
Figura 12 – ESP32

Fonte: Oliveira, 2017

O ESP32 conta com microprocessador ESP-WROOM-32 e opera na faixa de 2,2V e 3,6V, com nível lógico operante apenas em 3,3V. Possui conversor analógico digital (ADC) e digital analógico (DAC), incorpora um bluetooth v4,2 e conta com os padrões 802,11b, g e n, utilizando o Wi-Fi entre 2,4 GHz e 2,5GHz e suportando os padrões de segurança WPA, WPA2, WPA2-Enterprise e WPS. Como complemento, no Wi-Fi, ainda possui opções de criptografia AES / RSA / ECC / SHA. (OLIVEIRA,2017).

Possuindo 30 ou 36 pinos GPIOs, com Rx e Tx para comunicação serial, em que alguns possuem função PWM e SPI (Interface Serial Periférica) e compatibilidade com a IDE Arduino. Sendo maior que os ESP8266, mas com tamanho robusto e grande versatilidade. (OLIVEIRA,2017). A pinagem do ESP32 é apresentada na figura 13.

Figura 13 - Pinagem do ESP32



Fonte: Oliveira, 2017

3.1.4 Acessórios

Para a montagem do circuito do protótipo são necessários, além do microcontrolador Arduino UNO e o módulo ESP-01, alguns componentes para complementar a comunicação dos outros componentes do circuito, sendo eles, os jumpers, resistores e a *protoboard*, e conectados a eles lâmpadas LEDs controladas pelo microcontrolador.

LEDs são fontes de luz artificiais baseadas em semicondutores que convertem corrente elétrica em luz (LOSS, 2013). Os jumpers são cabos de cobre responsáveis por fazer a ligação entre os conectores serial dos microcontroladores Arduino Uno e ESP-01 e entre as LEDs e o microcontrolador, já o resistor será responsável por limitar a corrente elétrica. A *protoboard* ou placa de ensaio é uma placa com furos, em que sua superfície é uma matriz de contato de plástico com vários orifícios onde os componentes ou cabos são inseridos. No seu inferior há contatos metálicos soldados que interligam eletricamente os componentes que foram inseridos na matriz (PLACA, 2021).

3.2 Componentes de Software

Neste item serão demonstrados os programas de softwares usados para compor a prototipação deste trabalho.

3.2.1 Arduino IDE

Para a programação do Arduino Uno e do módulo ESP-01 é utilizado o ambiente de desenvolvimento integrado do Arduino, o Arduino IDE, pois ela conecta ao Arduino, fazendo o upload dos programas e permitindo a comunicação com o microcontrolador (OVERVIEW, 2022).

A programação é feita por meio de um editor de código, dispondo também de um console, uma barra de ferramentas, uma área de notificação e vários menus para melhor uso da IDE. Além disso, também possui meios de aumentar seus recursos, como uma vasta quantidade de bibliotecas, aumentando as funções da IDE e junto a isso, também tem suporte para hardwares externos, possibilitando sua adição aos, já amplos, diretórios (OVERVIEW, 2022).

Para isso, o Arduino utiliza de uma linguagem de programação C++/C, com algumas modificações para facilitar a programação de microcontroladores, como o *setup*() e o *loop*(), que são funções específicas para microcontroladores. Sua estrutura então, conta com uma lógica de programação completa (CHAVIER, 2017).

3.2.2 Adafruit IO

A Adafruit é responsável pela implementação ou gerenciamento do banco de dados e do controle do hardware, através da utilização do seu próprio servidor web.

Adafruit é uma empresa de venda de componentes de hardware, que conta com diversos produtos voltados para a área de IoT. Para complementar esses hardwares, ela também oferece vários recursos de aprendizagem, como vídeos sobre diferentes placas. Assim, para o uso desses produtos e a interação deles com a internet, também desenvolveram a Adafruit IO, que é uma plataforma open-source grátis que exibe e interage com os dados de um projeto através da internet (RUBELL, 2013a).

Para tanto, a plataforma conta com login e senha para manter as informações seguras, e um serviço cloud, ou seja, em nuvem. Logo, essa plataforma possibilita o controle de um projeto via internet, como um sensor ou rodas, mostrando os dados em tempo real (RUBELL, 2013b).

Essa plataforma também conta com uma integração com hardwares, como o ESP32 ou algumas variações do ESP 8266, hardwares próprios e diversos outros. Também possui bibliotecas para integração com softwares como Arduino e Python, aumentando o escopo de programação e aumentando os tipos de placas para serem utilizadas, como o Arduino UNO e o ESP-01 (RUBELL, 2013b).

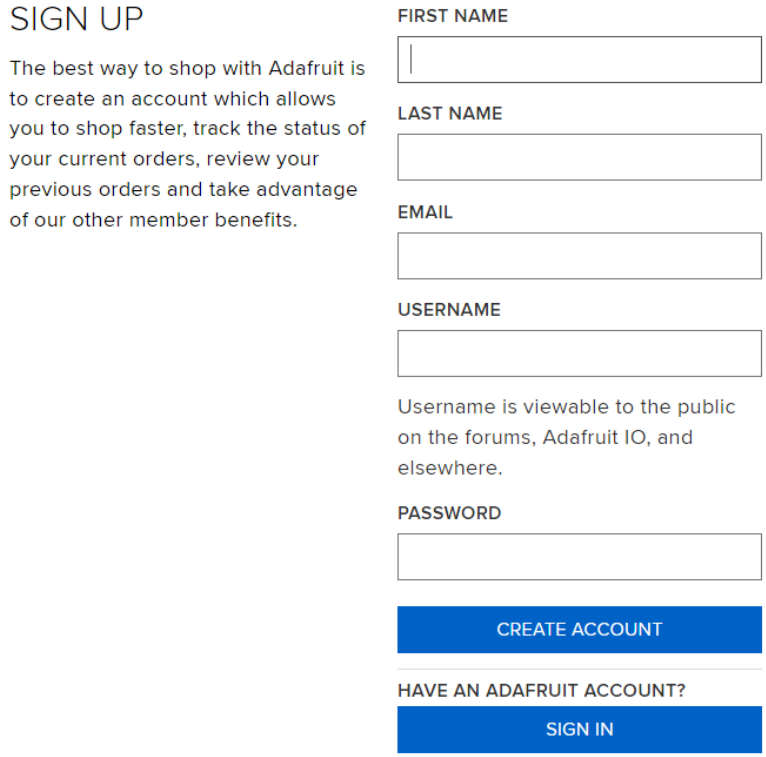
Para a montagem de um painel de controle para o projeto, a Adafruit oferece diversos tipos de controle, que vão de um simples botão de apetar ou de ON/OFF, até gráficos e mapas para controle de dados (RUBELL, 2013b).

O site da Adafruit é divido em várias sessões, contendo um blog da empresa, uma loja de hardwares e itens fabricados pela Adafruit, um guia de exemplos e aplicações e uma API (*Application Programming Interface*), sendo ela uma interface de usuário programável para a implementação da IoT, chamada io.adafruit.

3.2.2.1 Manuseio da API da Adafruit IO

Para inicializar o uso da API, é necessário criar um usuário e login, para que com isso, as informações aplicadas e manipuladas fiquem seguras, conforme a figura 14.

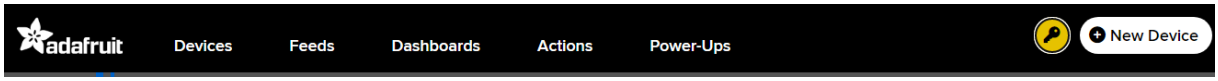
Figura 14 – Inscrição no site da Adafruit



Fonte: Modificada de io.adafruit, 2022

Com a conta Adafruit criada, são disponibilizados vários recursos para a criação e modificação da API de IoT. É possível vincular dispositivos produzidos pela Adafruit, onde, no próprio site, é disponibilizado o download do WipperSnapper (versão beta). Esse *firmware* possibilita a conexão direta de um microcontrolador com a plataforma WEB, possibilitando o manuseio plena do software pelo Adafruit sem a necessidade de escrever códigos ou utilizar programas externos, conforme apresentado na figura 15.

Figura 15 – Recursos da adafruit IO

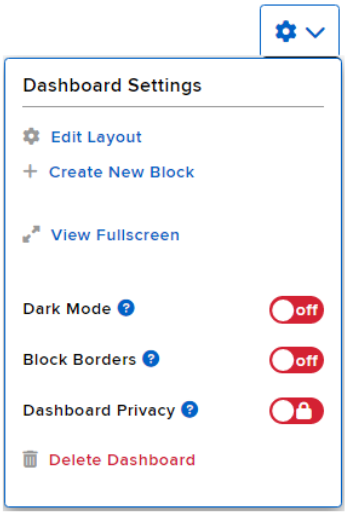


Fonte: Modificada de io.adafruit, 2022

Os dispositivos que conseguem utilizar o WipperSnapper estão disponibilizados em “*Devices*” ou em “+*New Devices*” como o ESP32-S2 Feather (há 17 placas compatíveis atualmente).

Na aba *Dashboards* são criadas as funcionalidades da API. Eles são os painéis em que será construída as funções da API, conforme a figura 16.

Figura 16 – Configurações do *Dashboard*



Fonte: Modificada de io.adafruit, 2022

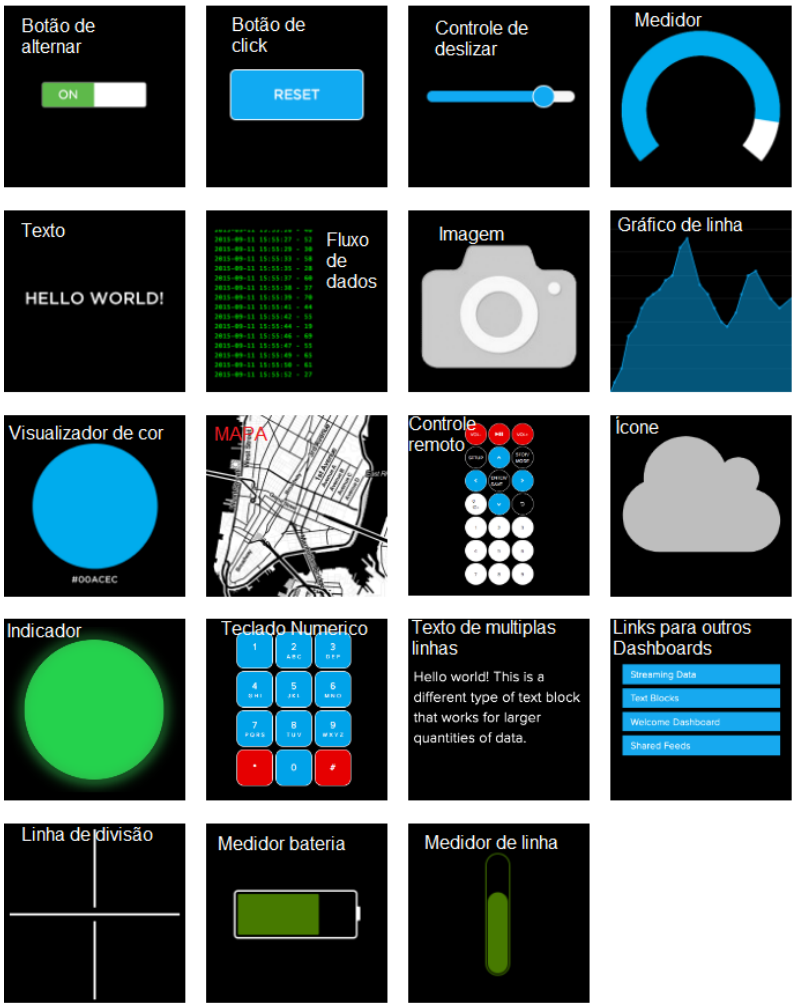
Para a criação e montagem da API, são utilizados diversos blocos, com funcionalidades diversas, criados com “+ *Create New Block*”. Com o comando “*Edit Layout*” é possível alterar a posição e tamanho de todos os componentes de um bloco.

Os blocos disponíveis são:

- Botão de alternar e Botão de click.
- Controle de deslizar, medidor, medidor de bateria e medidor de linha.
- Texto e Texto de múltiplas linhas: Texto pequeno e Grande.
- Fluxo de dados: Histórico de múltiplos *Feeds*.
- Imagem: Visualizar imagens em base64.
- Gráfico de linha e Visualizador de cor.
- Mapa: Rastreia os locais advindo do *Feed*, usado para comunicar com microcontroladores com GPS (Sistema de Posicionamento global).
- Controle remoto e Teclado Numérico.
- Indicador: Lâmpada indicadora Liga/Desliga baseada em condições.
- Ícone: Um ícone que pode ser utilizado para Design ou indicar o estado de um Feed.
- Links para *Dashboards*: Links para o acesso de quaisquer *Dashboards* feitos pelo usuário.
- Linha de divisão: Linha de divisão do *Dashboard*, utilizado para o melhorar o design.

Os blocos são apresentados como na figura 17.

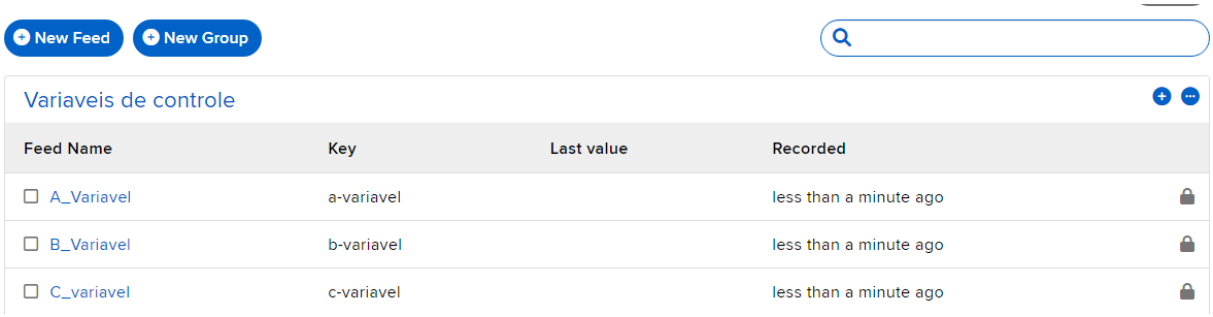
Figura 17 – Blocos do *Dashboard*



Fonte: Modificada de io.adafruit, 2022

Cada um dos blocos deve ser associado a um *Feed*, e utiliza as *Keys* (variáveis) geradas nele para interagir com os blocos. Assim, quando for necessário mandar ou receber informações de algum bloco para o hardware, a *Key* deve ser usada como uma variável de acesso na programação. Essas variáveis do Adafruit ficam salvas e podem ser acessadas através da aba “*Feeds*” (Figura 15). Esses *Feeds* podem ser separados em grupos, para facilitar seu manuseio entre vários *Dashboards*. Eles são apresentados na figura 18.

Figura 18 – Controle dos *Feeds*

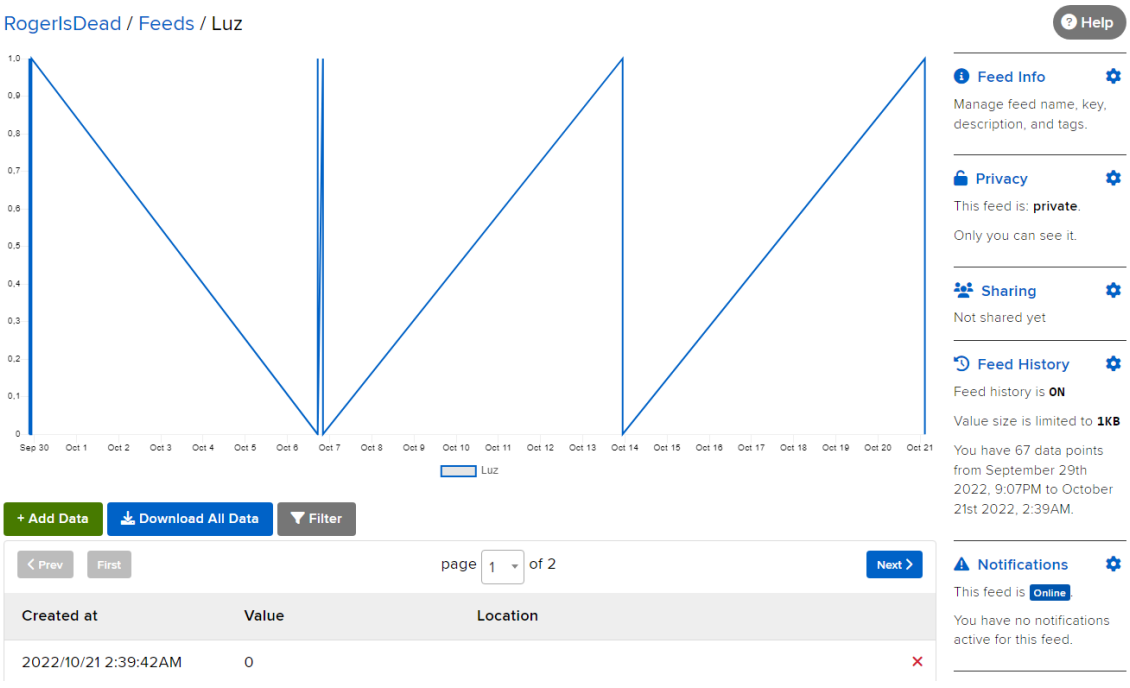


Fonte: Modificada de io.adafruit, 2022

Cada *Feed* guarda as mudanças de informações no servidor da Adafruit, e essas informações, que podem ser acessadas em cada *Feed*, apresentam a hora, o valor e, se configurado, até o local da mudança de valores de um *Feed*, no qual é montado um gráfico para melhor visualização dessas informações.

A figura 19 apresenta esses dados.

Figura 19 – Informações de um *Feed*



Fonte: Modificada de io.adafruit, 2022

Na aba “Actions”, é possível criar ações ou reações especificas dentro da API. Essas ações são acionadores opcionais para notificar o usuário de algum evento ocorrido na API. Essa mensagem pode ser repassada por e-mail, mensagem de celular, mensagem de *webhooks* (para um servidor web diferente) ou uma mensagem dentro da própria API.

São disponibilizadas ações reativas, programadas e cronometradas, como apresentado nas figuras 20, 21, 22 e 23.

Figura 20 – Exemplo de ações

Fonte: Modificada de io.adafruit, 2022

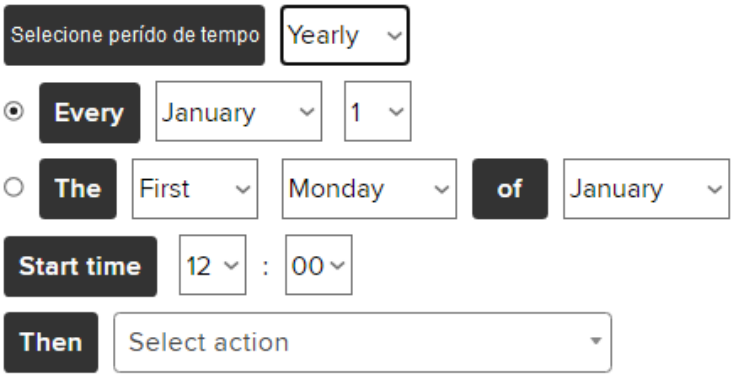
- Ação reativa: Compara um *Feed* com outro *Feed* ou valor, e, com resposta positiva, manda a mensagem para o usuário, em um determinado intervalo de tempo.

Figura 21 – Ação reativa

Fonte: Modificada de io.adafruit, 2022

- Ação programada: Programa um intervalo de tempo para executar a mensagem ao usuário. Abrange um intervalo de minutos, horas, dias, semanas, meses e anos.

Figura 22 - Ação programada



Fonte: Modificada de io.adafruit, 2022

- Ação cronometrada: Assim como a ação reativa, também compara um *Feed* com outro *Feed* ou valor, mas, com a resposta positiva, começa a cronometrar o tempo definido para depois executar a mensagem.

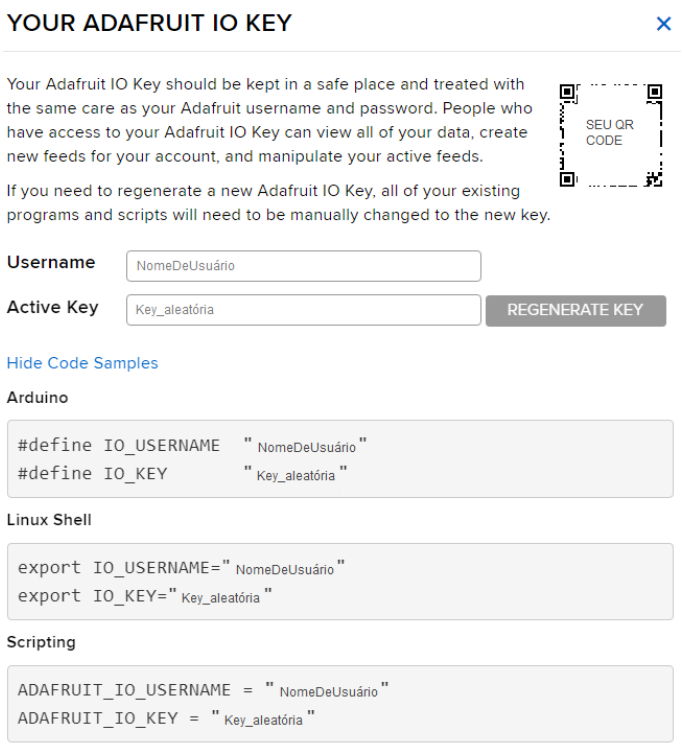
Figura 23 - Ação cronometrada

Fonte: Modificada de io.adafruit, 2022

Na aba “Power-Ups”, é possível integrar a API da Adafruit com outras funcionalidades, provenientes de serviços conectados a Adafruit, como a Zapier e a IFTTT. Além de serviços de conectar a relógios, ao clima e um gerador de dados aleatórios.

Os componentes mais importantes para o uso dessa plataforma em conjunto com outras, são as chaves de acesso Adafruit, simbolizados pela chave envolta por amarelo na Figura 15. É nesse local que são guardados o nome de usuário Adafruit e a chave Adafruit, utilizados na programação dos microcontroladores. São disponibilizados ainda, exemplos de código para implementação. A figura 24 apresenta esses dados.

Figura 24 – Chave e nome de usuário Adafruit

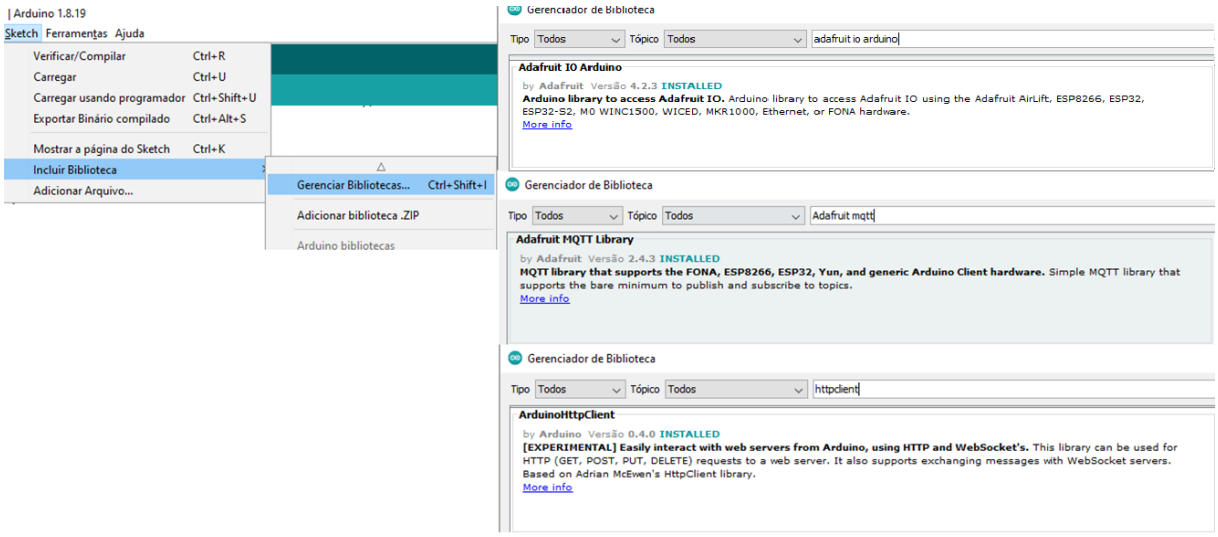


Fonte: Modificada de io.adafruit, 2022

Para poder utilizar a IDE Arduino para programar uma conexão com a Adafruit, é necessária a instalação das bibliotecas Adafruit IO Arduino, Adafruit MQTT Library e ArduinoHttpClient. Essas bibliotecas possuem as funções necessárias para o acesso do servidor web da adafruit e da conexão Wi-Fi de alguns módulos. Essas bibliotecas são importantes para fazer a conexão MQTT entre o servidor web e o microcontrolador.

A figura 25 mostra como instalar as bibliotecas necessárias.

Figura 25 – Instalação das bibliotecas para conectar no Wi-Fi e Adafruit



Fonte: Elaborada pelo autor, 202

O protocolo MQTT, é um protocolo de mensagens otimizado para pequenos dispositivos móveis e sensores, no qual a troca de mensagens é leve e pouco complexa. Assim, esse protocolo é ideal para aplicações IoT, tendo que seus princípios arquitetônicos buscam minimizar o uso de banda de rede e uso de recursos dos equipamentos, enquanto garante confiabilidade e um certo nível de economia de energia (MQTT, 2022).

3.2.3 Blynk IO

Blynk, assim como a Adafruit, é uma plataforma especializada em IoT, que, além de poder utilizar um site como servidor web, ele também utiliza um aplicativo mobile. Esse aplicativo é uma ótima opção para uso pessoal em celulares, tornando o acesso mais rápido e direto, sem a necessidade de um navegador para controle (BLYNK, 2022a).

O Blynk é um conjunto de softwares necessários para prototipar, implantar e controlar remotamente dispositivos eletrônicos conectados à internet. Essa plataforma foca na conexão de servidor web, iOS e Android, sem a criação de códigos, para analisar os dados em tempo real dos dispositivos IoT (BLYNK, 2022a).

As aplicações desse software são pretendidas para o uso em desenvolvimento ou para o uso de usuários finais, ou seja, é possível que outra pessoa, com o Blynk, instalado, consiga acessar o seu software. Para isso, o software possui configurações de usuários, que permitem a definição de usuários com permissões diferentes, no qual esses usuários terão acesso restrito aos dados (BLYNK, 2022a).

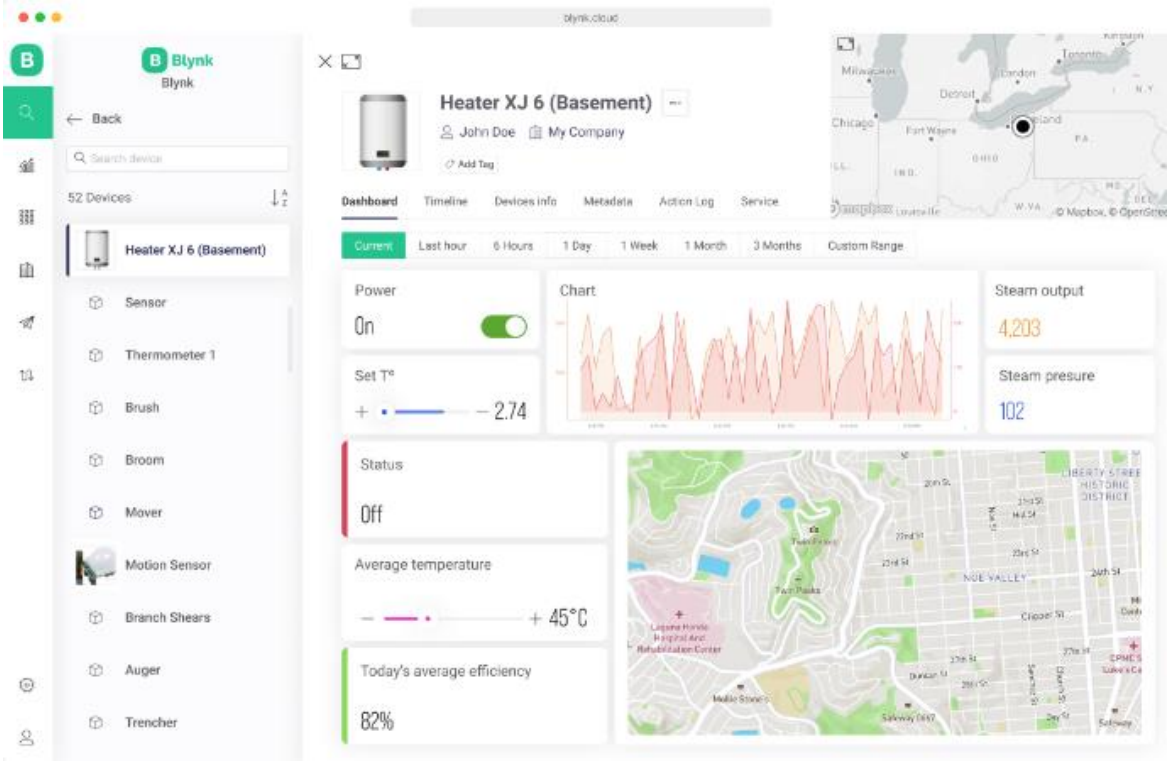
Essa plataforma é bastante utilizada por diversos usuários únicos, mas também por empresas, e por isso, eles possuem níveis de acesso a recursos. Esses níveis estão divididos em planos, que incluem, o grátis, o *Plus*, o *PRO* e o *Business*, que ficam respectivamente mais caros e com mais recursos. Esses recursos incluem aumento de possíveis dispositivos conectados, aumento do número e tipos de *Widgets* (todos inclusos a partir do *Plus*), aumento de armazenamento, aumento de usuários e suas permissões e até aumento de segurança (BLYNK, 2022a).

Na parte de hardwares, a Blynk suporta 7 tipos de microcontroladores com tokens dinâmicos, ou seja, tokens de identificação que são alterados com o passar do tempo para o aumento de segurança, sendo eles o ESP32, ESP8266, Arduino MKR WiFi 1010, Arduino Nano 33 IoT, Arduino MKR1000, Seeed Wio Terminal e TI CC3220. Entretanto, para tokens estáticos, a plataforma suporta mais de 400 tipos de dispositivos suportados diferentes, que incluem vários tipos de Arduino, como o Arduino UNO, módulos ESP, Raspberry PI, computadores com Linux/Windows/OS X, entre inúmeros outros. (BLYNK, 2022a).

Além disso, junto à conexão Wi-Fi, também consegue-se fazer a comunicação por meio de cabos USB (serial) e cabos de ethernet para os módulos que os suportam.

Na parte de software, a Blynk possui o “Blynk.Console”, uma aplicação web da plataforma, que pode ser acessada por vários tipos de usuário e que faz o monitoramento e gerenciamento de dados e do controle de dispositivos, além proporcionar, através dela, a configuração de como conectar os dispositivos e a aplicação de suas configurações (BLYNK, 2022b). A figura 26 apresenta um exemplo dessa aplicação.

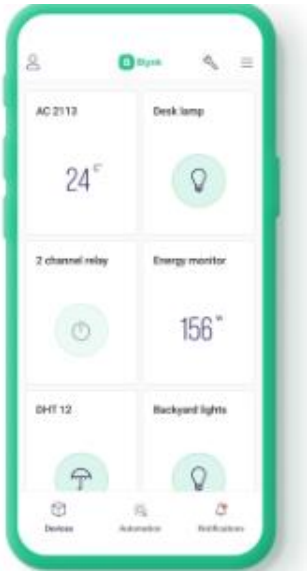
Figura 26 - Exemplo Blynk.Console



Autor: Blynk, 2022b

Também possui o “Blynk.Apps”, sendo sua aplicação mobile, que possui os mesmos recursos da aplicação web embutidas no ambiente mobile, como apresentado na figura 27.

Figura 27 – Exemplo Blynk.Apps



Autor: Blynk,2022b

Para a programação dos dispositivos de controle, existe o “Blynk.Edgent”, que é a biblioteca responsável por controlar e rodar os diversos modelos de hardware compatíveis. Uma versão simplificada é apresentada na figura 28.

Figura 28 – Exemplo Blynk.Edgeny

Fonte: Blynk, 2022b

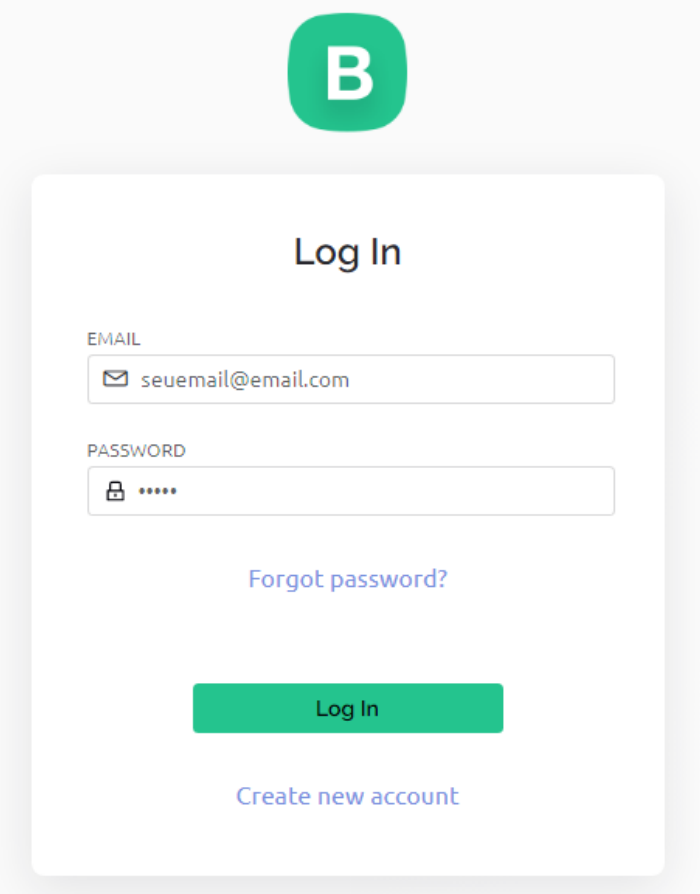
É necessário baixar e instalar a biblioteca da Blynk, disponibilizada no *Gethub* da empresa, para usar essa função na IDE de programação.

Por fim, para vincular todas essas funcionalidades, existe a “Blynk.Cloud”, uma infraestrutura de servidor responsável pela comunicação e armazenamento de dados entre todos os componentes.

3.2.3.1 Manuseio da API da Blynk IO

Primeiramente, para poder ter acesso aos recursos da API, é necessário criar uma conta na plataforma Blynk. Com a conta criada o acesso aos recursos é disponibilizado. A figura 29 apresenta a tela de *Log In*.

Figura 29 – Blynk *Log In*

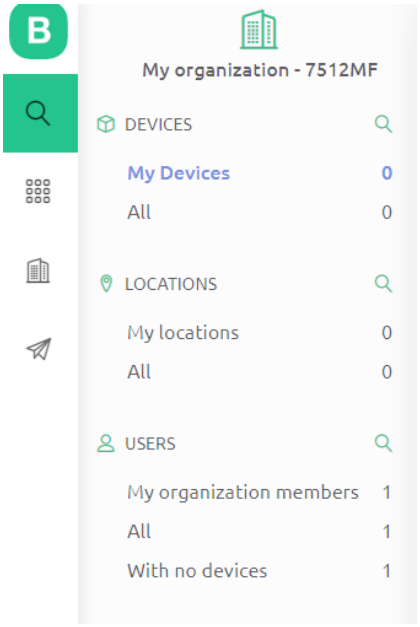


Fonte: Modificada de blynk.cloud, 2022

Com o acesso a plataforma, há diversos recursos disponíveis. Primeiro, há o recurso de busca, que serve para visualizar e alterar os dispositivos, localizações e usuários cadastrados na sua plataforma de forma rápida e eficaz.

A figura 30 mostra o recurso de busca.

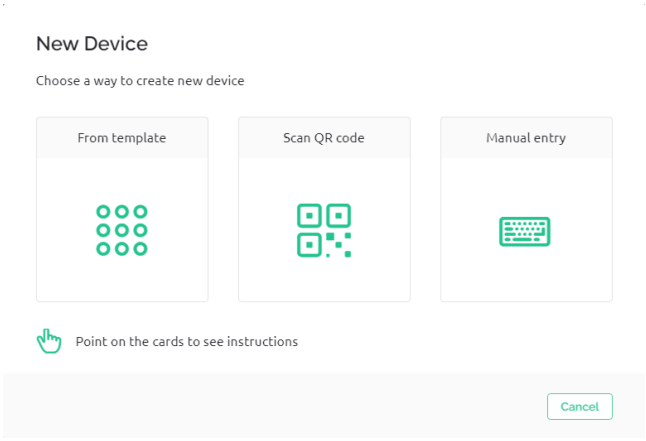
Figura 30 – Recurso de busca Blynk



Fonte: Modificada de blynk.cloud, 2022

Os dispositivos também podem ser adicionados pelo aplicativo mobile ou pelo servidor web, e, sua adição, serve para conexão e controle direto deles. Isso é realizado utilizando uma entrada manual, ou de escaneamento de código ou de um *"Template"*. Como apresentado na figura 31.

Figura 31 – Adicionar dispositivos na Blynk

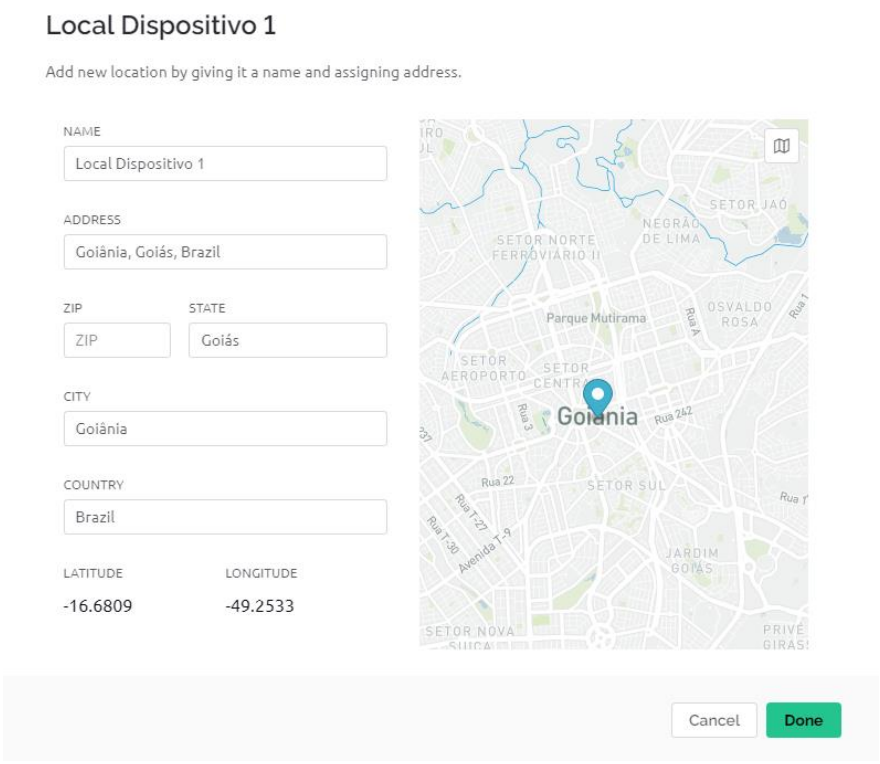


Fonte: Modificada de blynk.cloud, 2022

A opção "*locations*" serve para marcar locais no mapa, onde se encontram os dispositivos, sendo bastante útil quando os dispositivos se encontram muito distantes.

A figura 32 apresenta a adição de uma localização

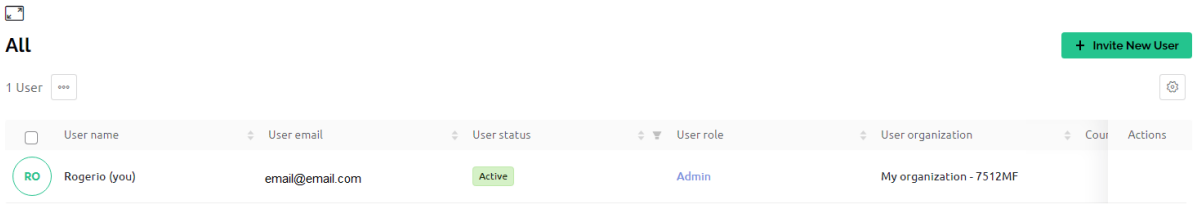
Figura 32 – Adicionar localização na Blynk



Fonte: Modificada de blynk.cloud, 2022

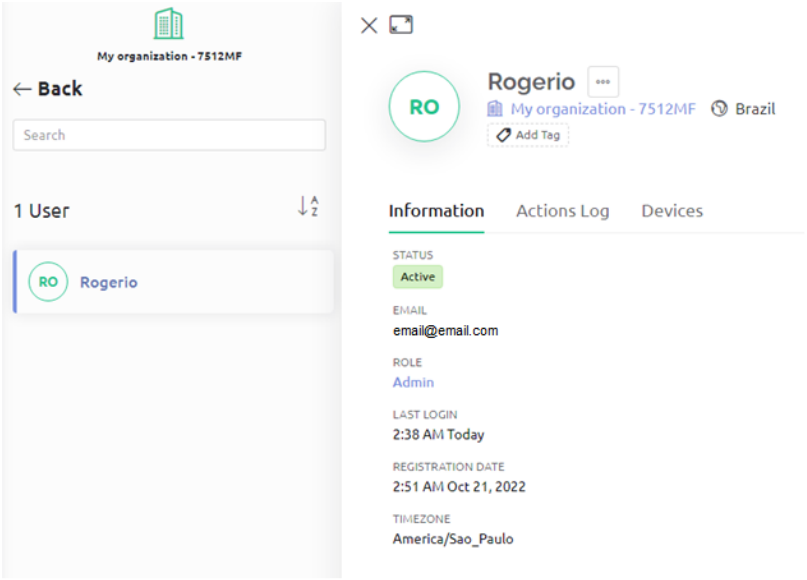
E, nos usuários, é possível visualizar os nomes, e-mails, *status*, função, país e várias outras informações de todos os usuários inclusos, além de permitir o convite de novos usuários. As figuras 33 e 34 apresentam essas informações.

Figura 33 – Usuário Blynk



Fonte: Modificada de blynk.cloud, 2022

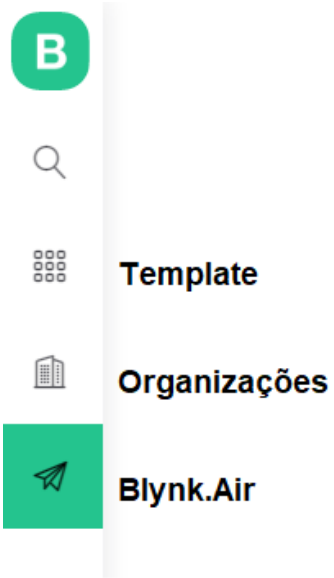
Figura 34 – Informações de um usuário Blynk



Fonte: Modificada de blynk.cloud, 2022

As aplicações de Organizações e Blynk.Air são aplicações avançadas da plataforma. A aba “Organizações” só é disponibilizada a partir do plano *Plus* e tem o propósito de montar uma organização sobre os usuários da aplicação simulando uma empresa, com uma determinada hierarquia. Já a aba Blynk.Air serve para mantar o software atualizado através da plataforma. Ambas são apresentadas na figura 35.

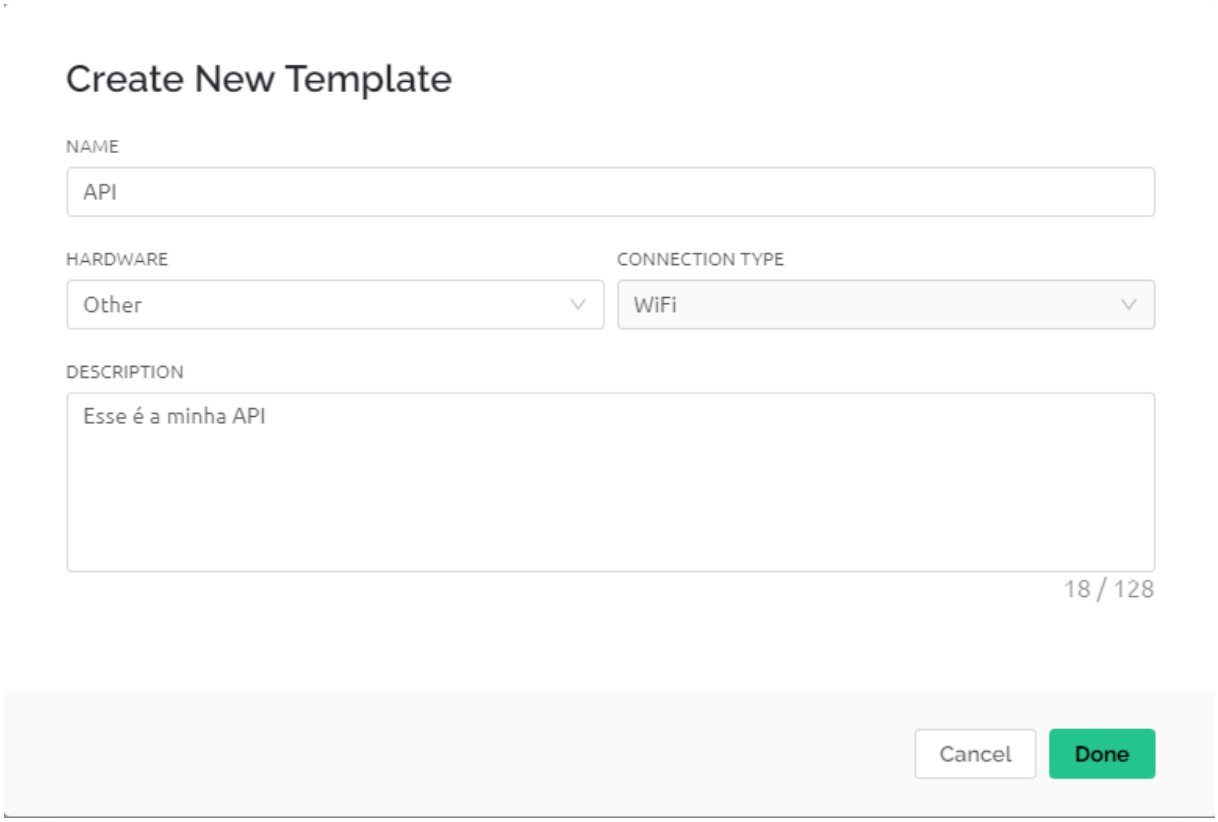
Figura 35 – Aplicações avançadas



Fonte: Modificada de blynk.cloud, 2022

A aplicação usada para a criação da API de controle, é o “*Template* ”. Nele, é criado o nome o dispositivo usado para a API criada, como apresentado na figura 36.

Figura 36 – Criando um *template* blynk

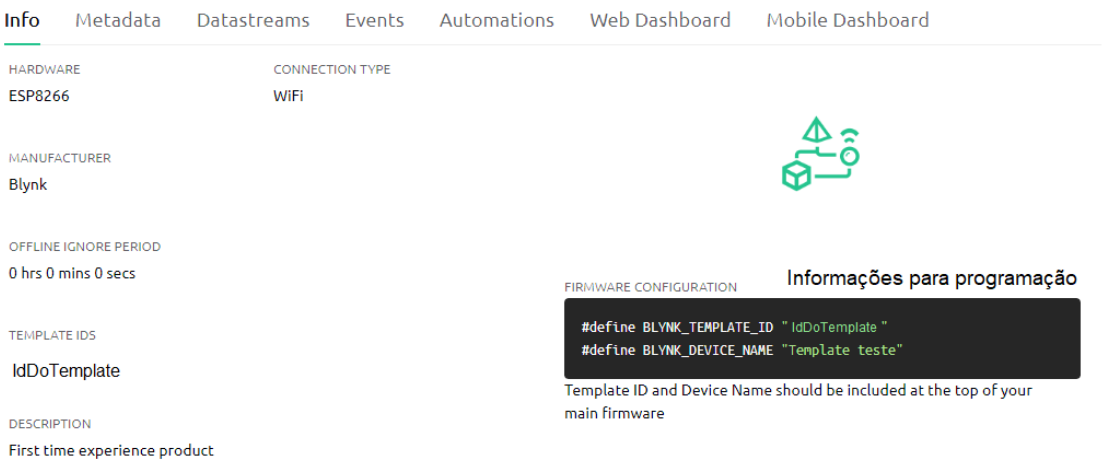


Fonte: Modificada de blynk.cloud, 2022

Com o *template* criado, tem-se acesso a API e a 7 abas de manuseio

A aba “Info” é de informações do *template*, e dentro dela, serão fornecidas primeiramente suas informações, na aba “Info”, sendo que entre elas, estará o identificador do *template* e seu nome, que são utilizados para fazer a configuração da programação, como apresentado na figura 37.

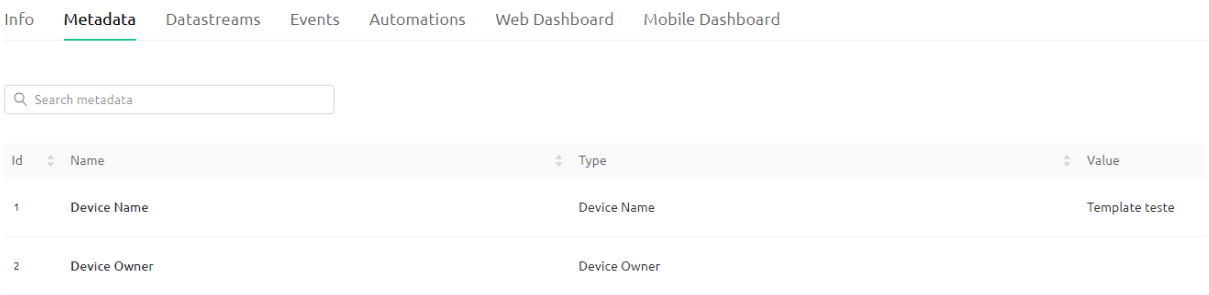
Figura 37 – Informações do *template* Blynk



Fonte: Modificada de blynk.cloud, 2022

A aba “Metadata” é utilizada para salvar e acessar os dispositivos conectados, e seus donos no *template* criado, como apresentado na figura 38.

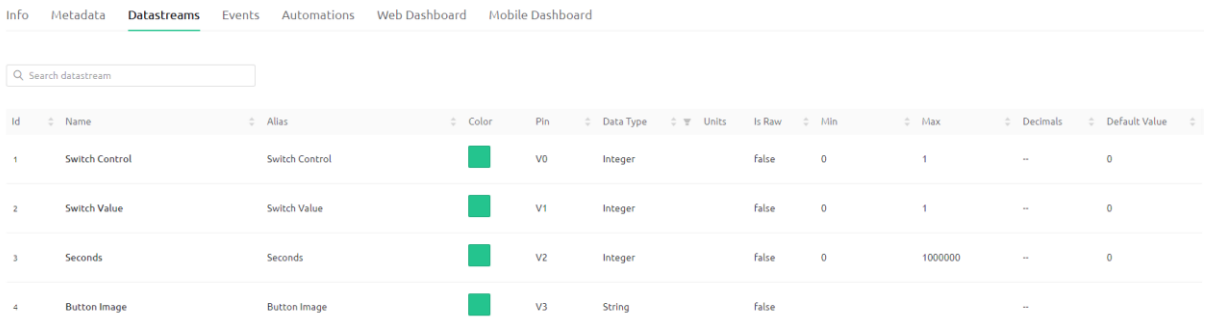
Figura 38 – Dispositivos conectados em “metadatas”



Fonte: Modificada de blynk.cloud, 2022

Já a aba “Datastreams” é responsável pelo manuseio das variáveis de controle, ou seja, as variáveis utilizadas para comunicar e controlar os dispositivos conectados através dos protocolos Blynk, utilizados nos dispositivos. Assim, ela é utilizada para ver as informações de uma variável ou alterá-las. As variáveis são apresentadas na figura 39.

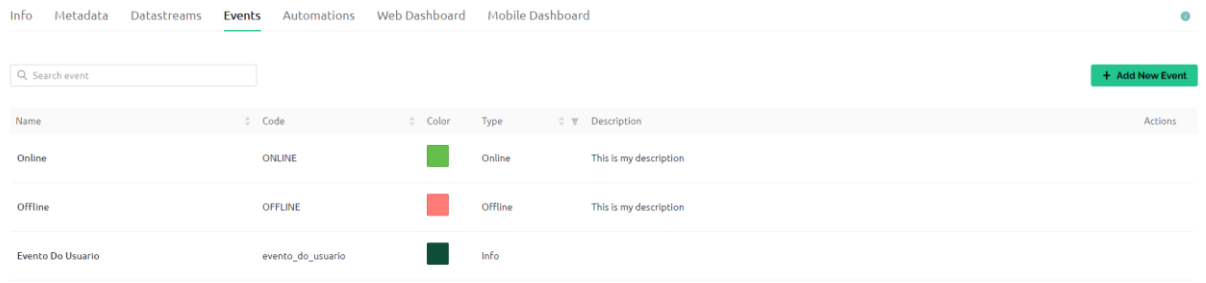
Figura 39 – Variáveis de controle “Datastreams” Blynk



Fonte: Modificada de blynk.cloud, 2022

Utiliza-se a aba “Events” para criar eventos, usados para rastrear, registrar e trabalhar com eventos importantes que ocorrem no dispositivo,usados para mandar e-mails, mensagens no celular ou no aplicativo. Existem três tipos de eventos, o evento de sistema, que é um evento padrão do Blynk e que não pode ser desativado, apenas configurado, o evento personalizado, ou seja, evento criado pelo usuário e eventos de conteúdo, que são eventos informativos e mostrados separadamente no aplicativo. A aba “Events” é apresentada na figura 40.

Figura 40 – Eventos Blynk



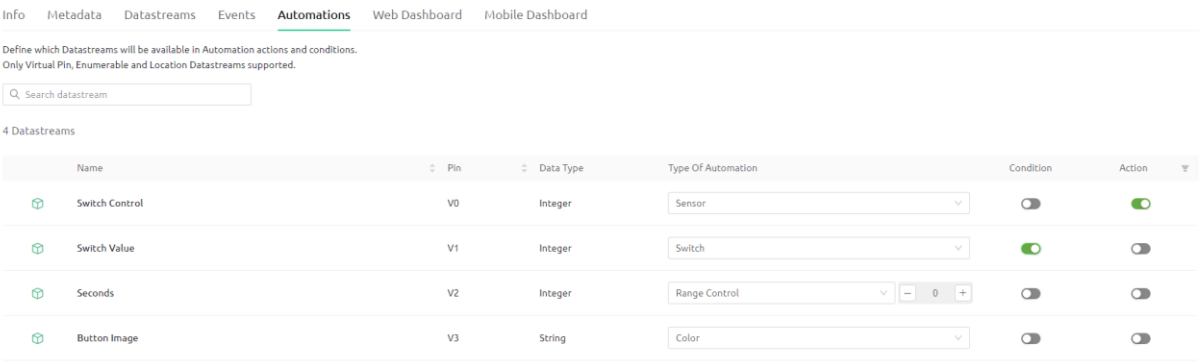
Fonte: Modificada de blynk.cloud, 2022

A aba de automação possui os cenários de automação criados pelo usuário, permitindo ao usuário criar e modificar ações do dispositivo baseadas em condições estabelecidas. Essas condições podem ser de cronograma, baseada na chegada de uma hora ou data específica para acionar um evento, de nascer e pôr do sol, na qual o evento é ativado apenas antes ou depois do pôr ou do nascer do sol, de estado do dispositivo, em que o evento é ativado baseado em um valor de uma variável “datastream”, e por fim, de cena, que aciona manualmente um cenário de automação do aplicativo móvel Blynk.App ou do aplicativo Web Blynk.Console.

As ações disponíveis para essas condições são: mandar uma notificação no celular, encaminhar dados do dispositivo para alguma outra variável do dispositivo, mandar um e-mail, definir um atraso e mudar o valor de uma variável “datastream”.

A figura 41 apresenta a aba “Automations”.

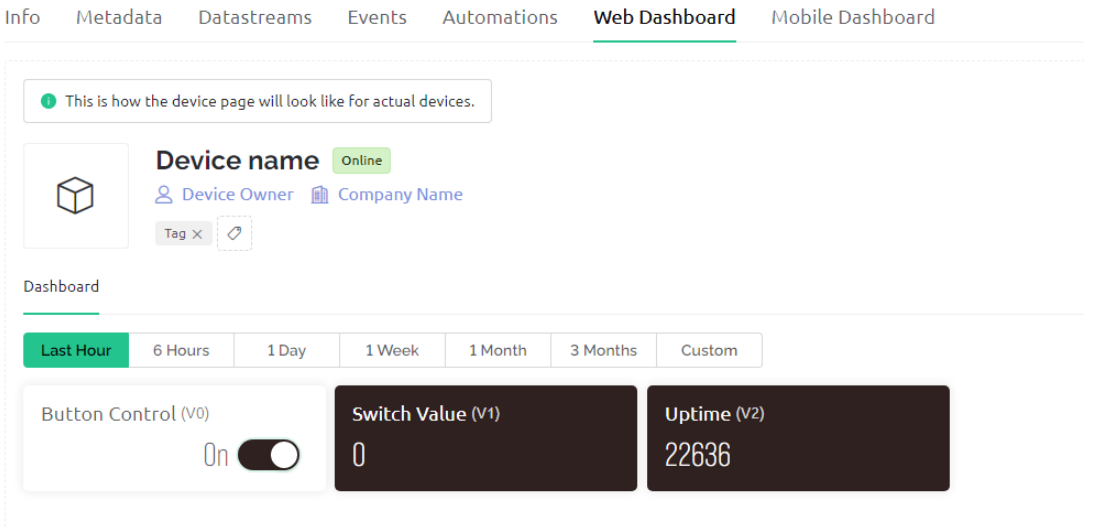
Figura 41 – Automações Blynk



Fonte: Modificada de blynk.cloud, 2022

A aba “Web Dashboard” é onde está localizada a interface da API web que será construída. Nela são disponibilizados os *Widgets*, ou seja, blocos de função para controle da API. Com a versão gratuita são disponibilizados 8 deles, com um limite de 30 na API. A figura 42 mostra um exemplo de uma API web.

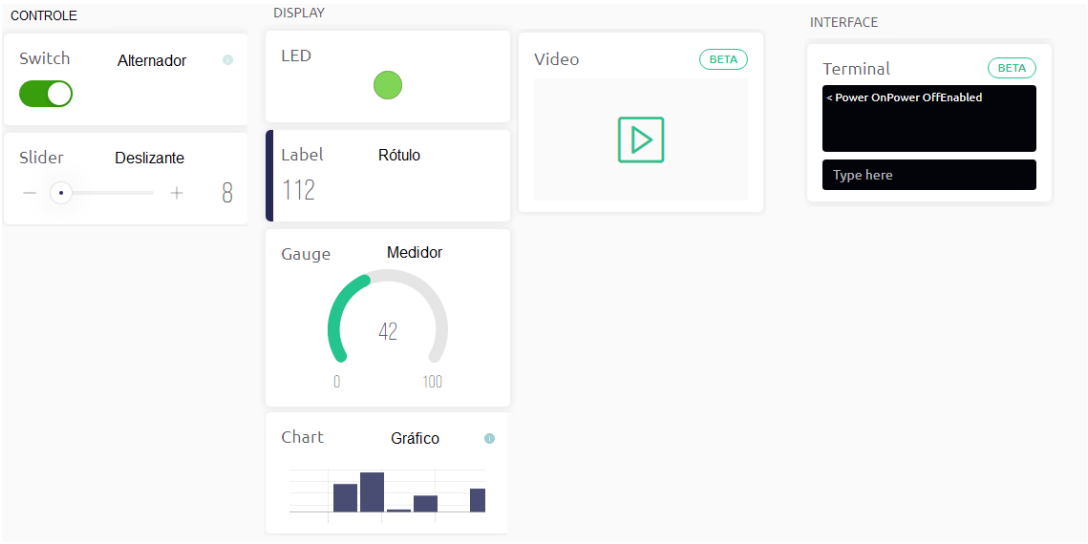
Figura 42 – API web Blynk



Fonte: Modificada de blynk.cloud, 2022

Para a aplicação web, são disponibilizados 8 blocos, divididos em *widgets* de controle que são o alternador e controle deslizante, *widget* de display, que são LED, Rótulo, medidor, gráfico e vídeo, e *widget* de interface que é o Terminal. Esses blocos são apresentados na figura 43.

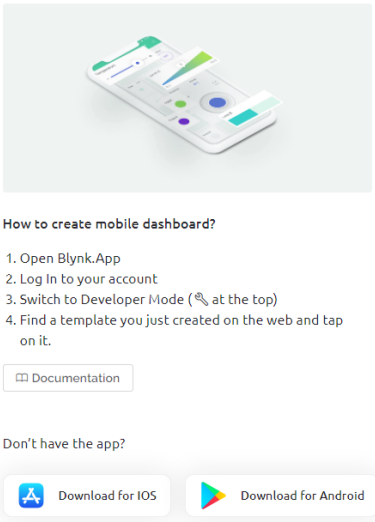
Figura 43 – *Widgets* API web



Fonte: Modificada de blynk.cloud, 2022

Por último, a aba “Mobile Dashboard” não é utilizada no navegador já que essa aba serve apenas para indicar como acessar o ambiente da API mobile, como apresentado pela figura 44.

Figura 44 – Aba de direcionamento para API mobile

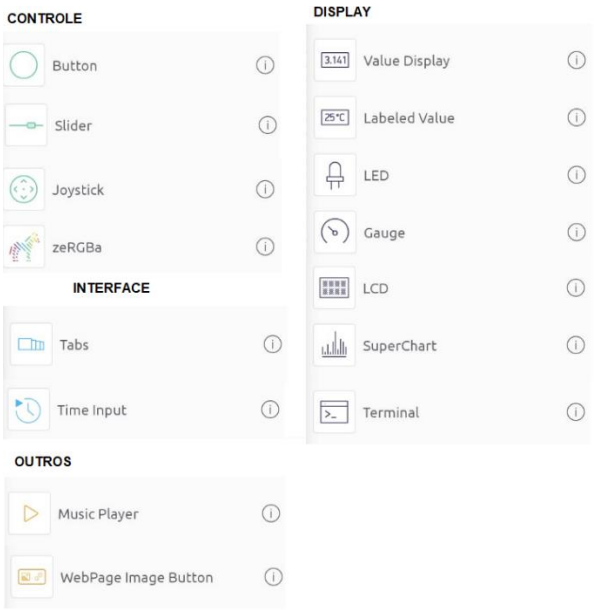


Fonte: Modificada de blynk.cloud, 2022

Assim, para criar a API mobile, é necessário o fazer o *login* no aplicativo e acessar o *Developer Mode* (modo desenvolvedor) e selecionar o mesmo *template* criado na API web. Com isso, será disponibilizado o acesso à edição da API mobile.

Essa API é totalmente diferente da API web, mesmo que seja limitada pelo plano grátis, ela possui mais blocos para personalização, contanto com quatro *widgets* de controle: Botão, deslizante, *joystick* (controle de vídeo game) e zeRGBa (controle de cor RBG). Sete *widgets* de display: Display de valor, display rotulado, LED, medidor, LCD, super gráfico e terminal. Dois *Widgets* de interface: *Tabs* (expandir o espaço do projeto) e entrada de tempo (manda valor de tempo para o hardware). E dois outros: *Music Player* (simula a interface de um tocador de música) e *WebPage Image Button* (Botão de imagem da página da Web). Os *Widgets* da API mobile são apresentados pela figura 45.

Figura 45 – *Widgets* API mobile



Fonte: Modificada de blynk.cloud, 2022

Por último, para fazer o uso API em conjunto com o microcontrolador, a Blynk também disponibiliza um gerador de códigos de exemplo, chamando “examples.blynk”, no qual, a partir da inserção de informações de microcontrolador, placa ou módulo para conexão à internet (Wi-Fi, *Ethernet*, bluetooth, serial ou cellular), ID do *template*, nome do dispositivo e o token de autenticação, é possível gerar exemplos de códigos feitos. Com isso, é necessário apenas fazer algumas alterações para encaixar o código em um projeto. A figura 46 mostra o gerador de código.

Figura 46 - Gerador de códigos Blynk

Fonte: Modificada de blynk.cloud, 2022

3.3 Interações de hardware e software

É demonstrado aqui, os meios pelos quais os softwares e os hardwares identificados podem interagir de modo a criar um ambiente IoT.

3.3.1 ESP-01 e a internet

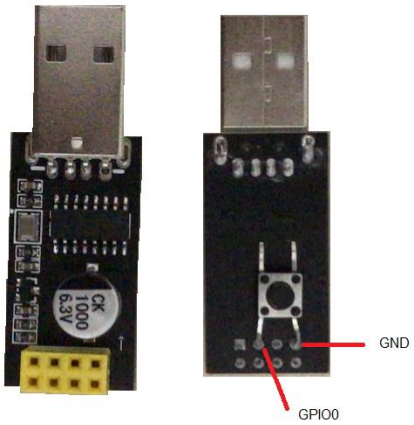
O módulo de Wi-Fi ESP-01, por possuir um microcontrolador embutido, não necessita de outros componentes para conseguir fazer uma conexão com a internet. Por causa de sua arquitetura, é possível utilizar o ESP-01 independentemente para construir uma aplicação IoT. Com os requisitos de energia e algumas conexões, esse módulo é capaz de controlar até 4 saídas (Pinos Tx, Rx, GPIO0 e GPIO2), enquanto faz a conexão com a internet (Pimenta, 2021).

Para programar um módulo ESP-01 é necessária apenas uma IDE, para que o programa possa ser elaborado e carregado. Quando montado em um circuito conectado a 3,3 V e conectado a uma lâmpada, por exemplo, ele vai controlá-la.

Esse módulo necessita de algumas implementações para ser aproveitado, sendo elas um botão ligando o pino GPIO0 e o GND, para que ele entre em modo de leitura e o programa possa ser carregado, e um adaptador USB/Serial, para fazer a ligação com o dispositivo que possui a IDE.

Para tanto, existe um gravador USB específico na utilização do ESP-01, no qual é possível fazer a conexão de leitura (GPIO0 e GND) com a solda de um botão, como é apresentado na figura 47.

Figura 47 – Gravador do ESP-01 com botão de modo de leitura



Fonte: Elaborada pelo autor, 2022

A figura 48 mostra como a conexão entre o gravador USB e o ESP-01 deve ser feita.

Figura 48 – Conexão ESP-01 e gravador USB

Fonte: Elaborada pelo autor, 2022

Após a montagem de um circuito, a programação do ESP-01 pode ser feita utilizando a IDE Arduino. Para tanto, é necessário preparar o ambiente de programação, instalando as versões ESP8266 nas placas da IDE, o drive de reconhecimento do módulo (drive CH340) e adicionando o pacote json do ESP nas preferências da IDE.

A programação para conectar o ESP na internet é objetiva, necessitando do SSID (Nome da rede) e da *Password* (senha da rede) para a conexão com Wi-Fi. Para conectar diretamente com um servidor web, existe a biblioteca “WiFiEsp.h”, que faz a conexão direta com um servidor IP disponível.

Para uma implementação mais acessível, é possível utilizar a biblioteca da Adafruit IO (3.2.2) chamada “AdafruitIO_WiFi”, que também funciona com outros dispositivos. Utilizada o SSID e o *password para* conectar à internet, mas também utiliza Adafruit *username* (nome de usuário) e *key* (chave de acesso) para conectar ao site da AdafruitIO e ter acesso as variáveis do site, para fazer o controle do módulo.

3.3.2 ESP-01 e Arduino

Um ponto forte do ESP-01 é a sua acessibilidade, pois, além de ter um preço reduzido, também possui um tamanho pequeno, facilitando sua aquisição e manuseio. Entretanto, seu ponto fraco é possuir poucas portas, que dificulta uma maior complexidade de circuitos. O Arduino, como o Arduino UNO, pode ser usado para contornar essa restrição, pois, com o Arduino recebendo as informações do ESP-01, é possível ter todas as diversas portas do Arduino para o controle do resto do circuito.

A comunicação entre ambas as peças pode ser feita com uma comunicação serial, ou seja, através da porta Tx e Rx do Arduino e ESP-01, pode ser feita a troca de informações mais complexas, ou com uma comunicação direta, na qual, através das portas analógicas ou digitais, é possível fazer a troca de níveis lógicos entre os microcontroladores.

Para que a combinação funcione, é preciso o uso de um uma fonte de 3,3 V para o ESP-01 e uma fonte de 7 a 12 V para o Arduino. Também é necessário, na comunicação entre as portas, um conversor de 3,3 V e 5 V, para ter as tensões certas em cada peça.

A conexão à internet ainda será feita pelo ESP-01, com as bibliotecas “WiFiEsp.h” e “AdafruitIO_WiFi” para os diferentes meios de implementação, e o circuito, conectado ao Arduino UNO, pode ter diversos outros módulos acoplados. E fazendo uso das bibliotecas disponibilizadas pela IDE Arduino, é possível, com mais investimento, utilizar sensores, motores ou até mesmo mais luzes para incrementar o projeto.

3.4 Implementações

Nesse tópico será demonstrada a implementação prática de circuitos de automação de luz, conectados à internet, para o controle e integração de ambientes.

3.4.1 Implementação com ESP-01

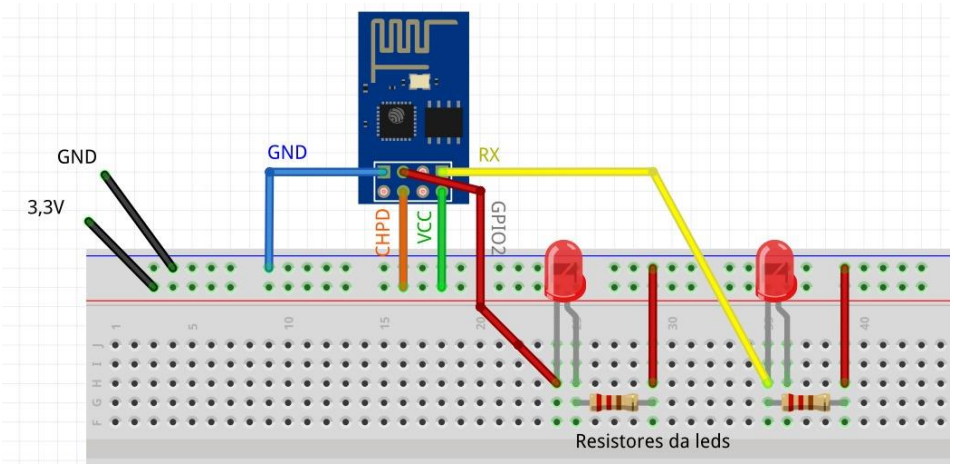
A implementação com menor número de componentes e menor investimento, é a que utiliza apenas o módulo ESP-01 para conectar-se direto com a internet. Em contrapartida, o uso exclusivo desse módulo é limitado a apenas as suas quatro portas de I/O.

3.4.1.1 Circuito do ESP-01

Para a aplicação única do ESP-01, é implementado um circuito com alimentação 3,3 V para o ESP-01, conectando na porta VCC (alimentação) e na porta CHPD (alimentação do chip), onde, as portas GPIO0, GPIO2, RX e TX, podem conectar-se à uma luz (led) que será controlada através da conexão com a Adafruit IO.

O circuito montado com duas led conectadas em RX e GPIO2 é apresentado na figura 49

Figura 49 – Circuito ESP-01

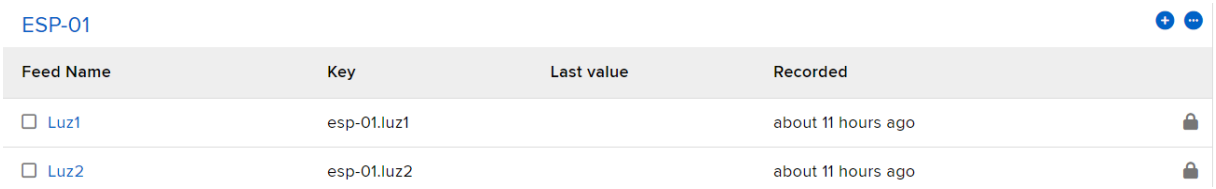


Fonte: Elaborada pelo autor, 2022

3.4.1.2 Ambiente de controle Adafruit

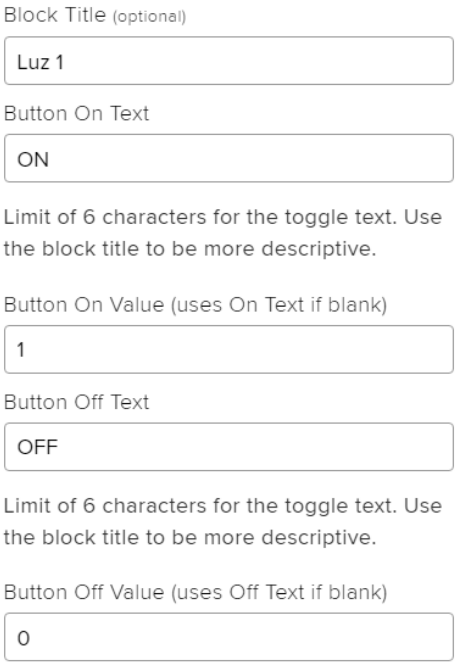
Utilizando do ambiente disponibilizado pela Adafruit, é possível construir um ambiente web para o controle do circuito. Após a criação do *Dashboard* denominado ESP-01, serão criados os *Feeds* de controle chamados ‘Led1’ e ‘Led2’. Esses *Feeds* foram separados no grupo ESP-01 para melhor idenficá-los. Ligados aos *Feeds*, os botões de controle das LEDs serão criados, os quais terão valores de ‘0’ para desligado e ‘1’ para ligado e mostrarão os estados de ‘ON’ para ligado e ‘OFF’ para desligado. Esses “*Feeds*” são apresentados na figura 50 e as configurações dos botões são apresentadas na figura 51.

Figura 50 – *Feeds* Adafruit ESP-01



Fonte: Modificada de io.adafruit, 2022

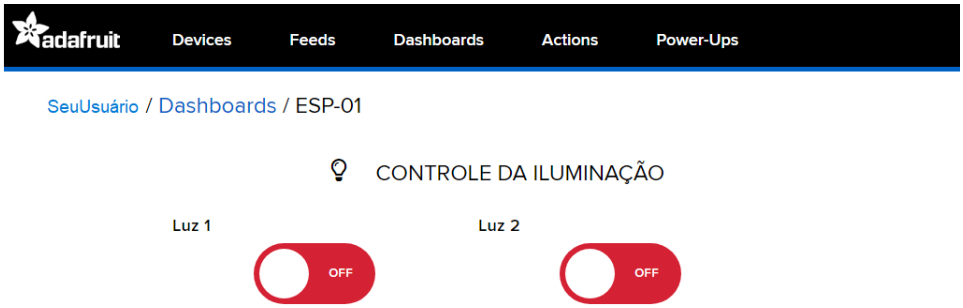
Figura 51 – Botões Adafruit ESP-01



Fonte: Modificada de io.adafruit, 2022

Como o ESP-01 consegue apenas mandar estados digitais de *HIGH* e *LOW* para suas portas, o controle será apenas de ligar e desligar as luzes. O ambiente de controle com o Adafruit é apresentado na figura 52.

Figura 52 – Ambiente de controle Adafruit ESP-01



Fonte: Modificada de io.adafruit, 2022

Após a criação da API de controle do Adafruit, é feita a programação do ESP-01 através da IDE Arduino. Isso servirá para conectar o módulo à internet e à API criada na Adafruit e controlar as LEDs.

Começando com a inclusão da biblioteca que se usa, feita pela Adafruit, com '#include "AdafruitIO_WIFI.h"'. Para fazer a conexão ao servidor web da Adafruit e ao Wi-Fi, é preciso criar um objeto dessa biblioteca, no qual esse objeto necessita das informações do nome de usuário e *key* da Adafruit, além do nome e senha da rede Wi-Fi. A definição dessas informações e a criação do objeto chamado 'io' é apresentada na figura 53.

Figura 53 – Objeto AdafruitIO_WiFi e definição de suas informações

```
#include "AdafruitIO_WiFi.h"

// CONFIGURAÇÃO DA ADAFRUITIO
#define IO_USERNAME  "NomeDeUsuario"
#define IO_KEY "Key_aleatória"

// CONFIGURAÇÃO DO WIFI
#define WIFI_SSID "NomeDaRedeWiFi"
#define WIFI_PASS "SenhaDaRedeWiFi"

AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
```

Fonte: Elaborada pelo autor, 2022

Após isso, cria se as variáveis dos *Feeds* com “AdafruitIO_Feed *Nome da variável”. A variável luz1 ficará receberá os dados do primeiro LED, e a variável luz2 os dados do segundo LED. Além disso, também é definido quais pinos receberão os leds, no caso a 3 (RX) e 2 (GPIO2). A figura 54 mostra essas declarações.

Figura 54 – Variáveis de feed e de pinos

```
// INSTANCIANDO OBJETOS
AdafruitIO_Feed *luz1 = io.feed("luz1");
AdafruitIO_Feed *luz2 = io.feed("luz2");

//DEFININDO PINOS DE SAIDA DAS LEDS
#define LED1 2
#define LED2 3
```

Fonte: Elaborada pelo autor, 2022

Com essas informações criadas e definidas, é possível criar o setup do programa. Primeiramente, utiliza-se o comando “io.connect()” para fazer a conexão MQTT com o site io.adafruit.com e, junto a isso, os pinos dos leds são definidos como pinos de saída, como apresentado na figura 55.

Figura 55 – Inicialização do setup

```
void setup() {
  // Inicia a conexão mqtt com io.adafruit.com
  io.connect();

  //Define os pinos das leds como pinos de saida
  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
```

Fonte: Elaborada pelo autor, 2022

Por último, usa-se a função “Feed->onMessage(função)” para acionar uma função quando o Feed utilizado for acionado e, junto a isso, também usa as funções “Feed->get()”, utilizado para pegar os valores das variáveis quando o ESP-01 iniciar. A figura 56 apresenta esses dados.

Figura 56 – Finalização do setup

```
//Chama as funções mensagemRecebia e mensagemRecebida2 quando os Feeds luz1 e luz2 trocarem estados
luz1->onMessage(mensagemRecebida);
luz2->onMessage(mensagemRecebida2);

//Pega o valores dos feeds luz1 e luz2 quando se inicia o ESP-01
luz1->get();
luz2->get();
```

Fonte: Elaborada pelo autor, 2022

Na criação do loop, é utilizada apenas a função “io.run()”, que mantém o cliente (ESP-01) conectado ao “io.adafruit.com” e processa quaisquer dados que são gerados na comunicação. Assim, quando há uma interação com a API do Adafruit, e algum feed tem seu valor alterado, ele é processado de acordo com as funções “onMessage”. Pode-se utilizar da função “delay (tempo)” para impor intervalos entre as conexões. Essas interações são apresentadas pela figura 57.

Figura 57 – Criação do loop

```
void loop() {
 //Mantém o cliente conectado ao site da adafruite
 //processa os dados da comnunicação
 io.run();
 //delay(500);
}
```

Fonte: Elaborada pelo autor, 2022

Para finalizar o programa, são criadas as funções ligadas às chamadas feitas por “onMessage”. São elas as funções “mensagemRecebida” e “mensagemRecebida2”. Essas funções recebem os dados da Adafruit, que estão em formato “AdafruitIO_data” e salvam na variável “data”. A partir disso, com a função “data->value”, é possível usar o valor do *Feed* e, com uma função condicional, acionar o led para ligá-lo ou desligá-lo. Essas funções são apresentadas na figura 58.

Figura 58 – Implementação das funções

```
// IMPLEMENTO DE FUNÇÕES
void mensagemRecebida(AdafruitIO_Data *data) {

  if(data->value()) //Se o valor do Feed for positivo(1)
  {
    digitalWrite(LED1,HIGH); //Liga a led1
  }
  else //se o valor nao for positivo, ou seja, é negativo(0)
  {
    digitalWrite(LED1,LOW); //Desliga a led1
  }

}


void mensagemRecebida2(AdafruitIO_Data *data) {

  if(data->value()) //Se o valor do Feed for positivo(1)
  {
    digitalWrite(LED2,HIGH); //Liga a led2
  }
  else //se o valor nao for positivo, ou seja, é negativo(0)
  {
    digitalWrite(LED2,LOW); //Desliga a led2
  }

}
```

Fonte: Elaborada pelo autor, 2022

Com essas implementações, o ESP-01 vai conseguir se conectar à internet e, a partir dos valores dos *Feeds* do Adafruit, controlar os LEDs.

3.4.2 Implementação com Arduino UNO e ESP-01

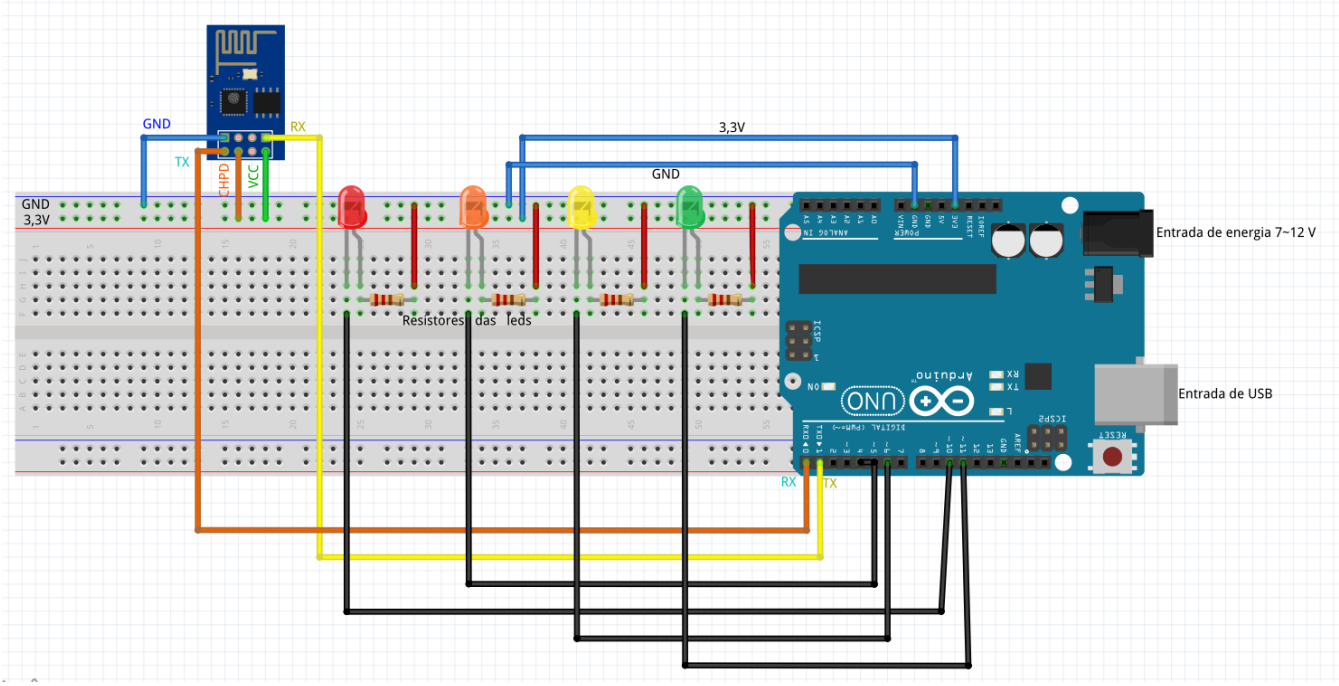
Essa implementação, aumenta o escopo de controle da implementação, enquanto também aumenta sua complexidade e investimento. Com a utilização da comunicação serial, TX Arduino -> RX ESP-01 e TX ESP-01 -> RX Arduino, é possível realizar troca de dados entre ambos os componentes, onde o ESP-01 se torna responsável apenas por enviar e receber dados da internet para o Arduino, sendo que este, fica responsável pelo controle dos LEDs. Logo, é possível aumentar o escopo de controle dos LEDs, aplicando as saídas PWM para controle da intensidade da luz.

3.4.2.1 Circuito Arduino Uno e ESP-01

Para a implementação com mais componentes e consequentemente mais complexidade e controle, utiliza-se o ESP-01, conectado em 3,3 V na porta VCC e na porta CHPD, e tendo suas portas RX e TX conectadas, respectivamente, nas portas TX e RX do Arduino UNO para a comunicação serial, onde a porta TX (*transmitter*) é a porta de transmissão de dados e a porta RX (*receiver*) é a porta de recepção de dados. Nas saídas PWM dos Arduino UNO, serão conectadas os LEDs.

O circuito montado com quatro LEDs conectados nas saídas PWM do Arduino UNO (5,6,10 e 11) e utilizando a comunicação serial RX/TX E TX/RX para comunicação Arduino UNO e ESP-01 é apresentado na figura 59.

Figura 59 – Circuito ESP-01 e Arduino UNO



Fonte: Elaborada pelo autor, 2022

3.4.2.2 Ambiente de controle com Adafruit

Utilizando o ambiente da Adafruit para criar a API de controle do projeto. Inicia-se a montagem do ambiente de controle com a criação do *Dashboard* denominado “Iluminação automatizada”. Essa API terá controle sobre o funcionamento de 4 luzes, sendo que, para cada uma, deve haver um botão deslizante, com função de controle de sua intensidade. As variáveis da ligar e desligar podem receber, respectivamente, os valores de 1 e 0. Já as variáveis de intensidade, recebem valores de 0 a 100, representando a porcentagem da intensidade.

Essas luzes serão controladas por oito *Feeds*, sendo que, quatro serão responsáveis por ligar e desligar as luzes, e os outros quatro responsáveis por controlar a intensidade. Eles são: “Luz Trabalho”, “Luz Banheiro”, “Luz Lazer”, “Luz Extra”, “IntensidadeTrabalho”, “IntensidadeBanheiro”, “IntensidadeLazer” e “IntensidadeExtra”. A figura 60 apresenta esses *Feeds*, assim como suas *Keys*, utilizadas para comunicação com a IDE de programação dos microcontroladores.

Figura 60 – *Feeds* de controle Adafruit

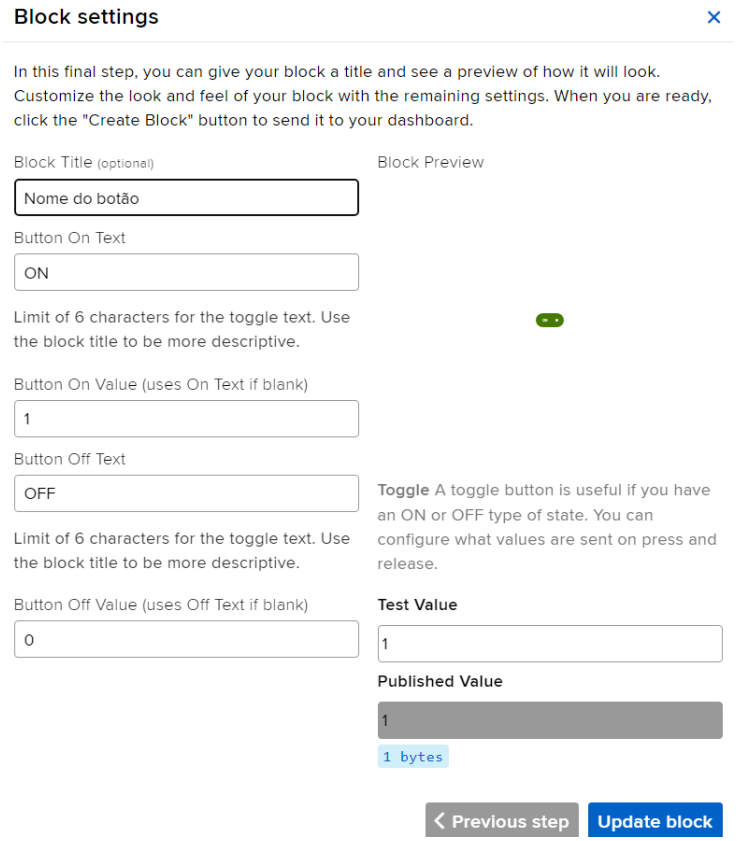
Variaveis de controle

Feed Name	Key	Last value	Recorded
IntensidadeBanheiro	intensidadebanheiro	0	6 minutes ago
IntensidadeExtra	intensidadeextra	0	6 minutes ago
IntensidadeLazer	intensidadelazer	0	6 minutes ago
IntensidadeTrabalho	intensidadetrabalho	0	6 minutes ago
Luz Banheiro	luz-banheiro	0	32 minutes ago
Luz Extra	luz-extra	0	28 minutes ago
Luz Lazer	luz-lazer	0	28 minutes ago
Luz Trabalho	luz-trabalho	0	28 minutes ago

Fonte: Modificada de io.adafruit, 2022

Com os *Feeds* criados, é possível criar os blocos de controle. Os primeiros blocos são os botões de liga e desliga. Os *Feeds* desses botões devem ter valor 1 para representar ligado e valor 0 para representar desligado. Suas configurações são apresentadas na figura 61.

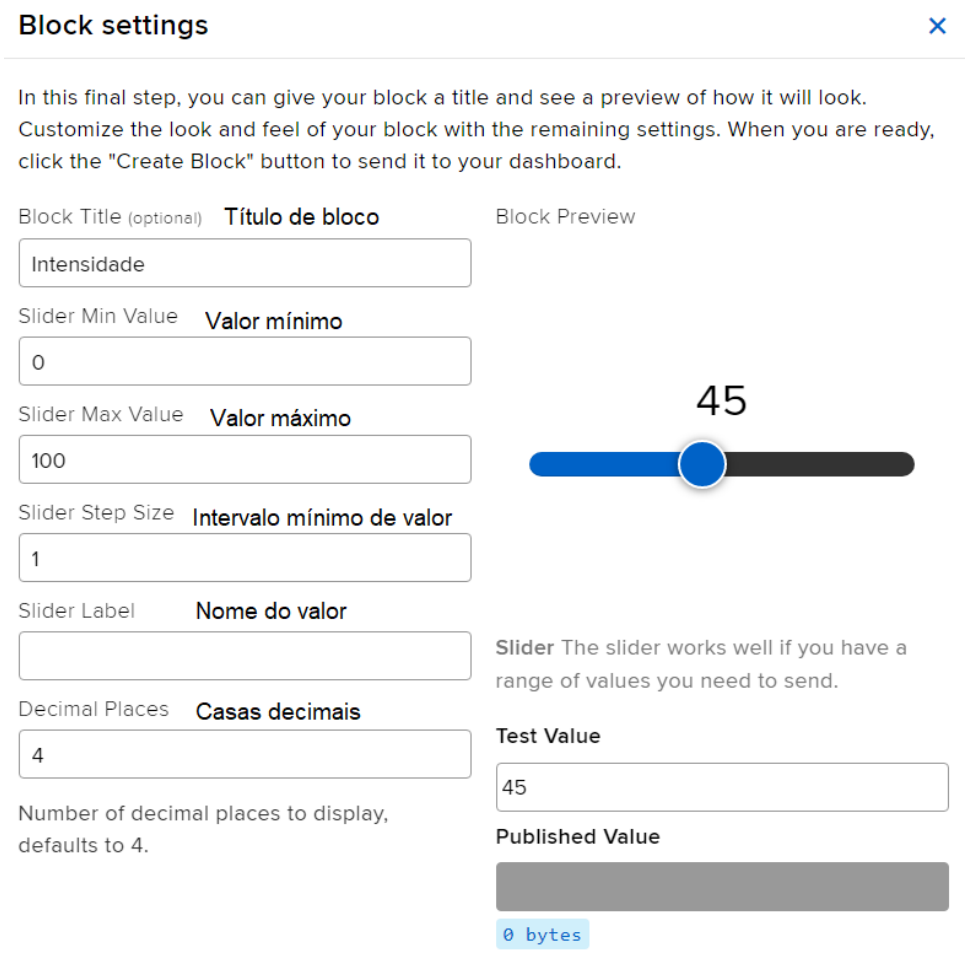
Figura 61 – Bloco de ligar e desligar luz Adafruit Arduino



Fonte: Modificada de io.adafruit, 2022

Para o controle da intensidade de cada luz, serão utilizados os blocos de deslizar, para escolher uma porcentagem de intensidade de luz. Esses blocos poderão alterar de valor entre 1 e 100, com um intervalo de 1. Sua configuração é apresentada na figura 62.

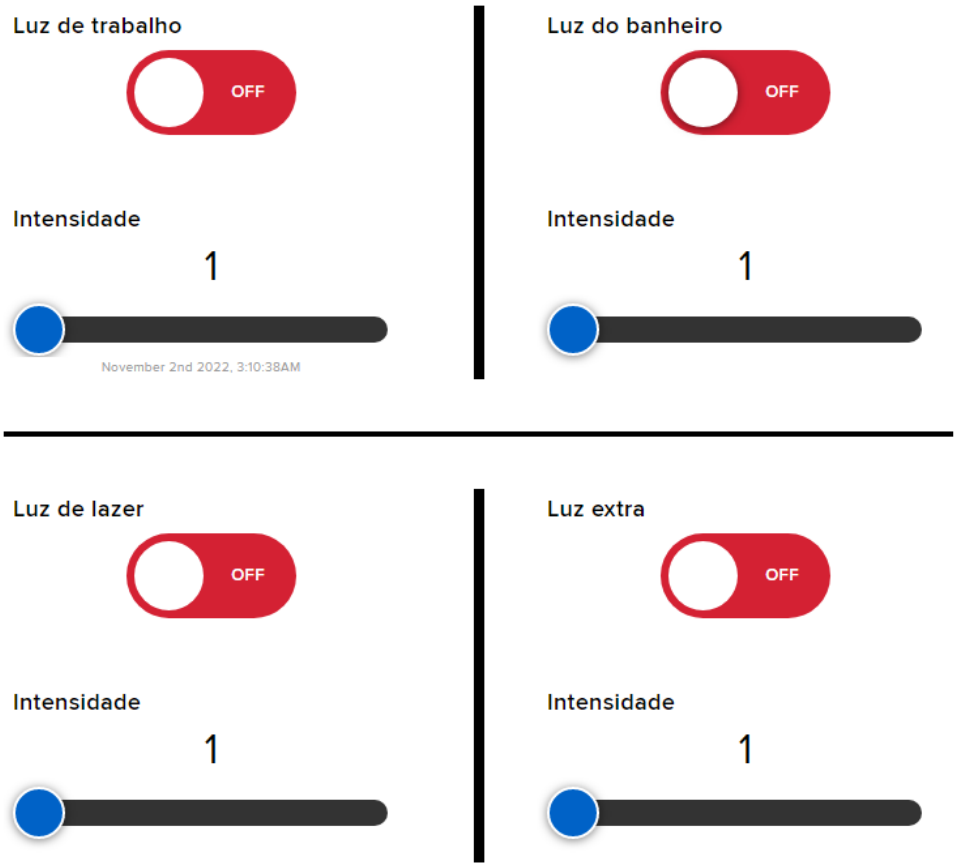
Figura 62 – Bloco de deslizar para intensidade da luz



Fonte: Modificada de io.adafruit, 2022

Após a criação dos blocos, é feito seu arranjo, utilizando dos blocos linha de divisão, para uma melhor visualização da API. A API final é apresentada na figura 63.

Figura 63 – API de controle web Adafruit Arduino



Fonte: modificada de io.adafruit, 2022

Com a conclusão da montagem da API, é necessário programar os microcontroladores, iniciando pelo ESP-01 e utilizando a IDE Arduino. O início da programação necessita da inclusão da biblioteca da Adafruit, sendo ela “AdafruitIO_WIFI.h”’. Seguido pela criação do objeto dessa biblioteca que faz a conexão entre Adafruit e módulo Wi-Fi. Essa configuração é padrão para utilizar o ESP-01, sendo apresentada pela figura 53 da aplicação ESP-01.

Com esse objeto criado, pode-se criar as variáveis responsáveis pelos valores dos *Feeds*. As oito variáveis de intensidade de estado são apresentadas na figura 64.

Figura 64 - Variáveis de controle do ESP-01 com Arduino

```
// INSTANCIANDO OBJETOS
AdafruitIO_Feed *luzBanheiro = io.feed("luz-banheiro");
AdafruitIO_Feed *luzTrabalho = io.feed("luz-trabalho");
AdafruitIO_Feed *luzLazer = io.feed("luz-lazer");
AdafruitIO_Feed *luzExtra = io.feed("luz-extra");
AdafruitIO_Feed *intensidadeBanheiro = io.feed("intensidadebanheiro");
AdafruitIO_Feed *intensidadeTrabalho = io.feed("intensidadetrabalho");
AdafruitIO_Feed *intensidadeLazer = io.feed("intensidadelazer");
AdafruitIO_Feed *intensidadeExtra = io.feed("intensidadeextra");
```

Fonte: Elaborada pelo autor, 2022

Após isso, cria-se o setup do programa. Ele começa com a inicialização da porta serial, que será utilizada para comunicação com o Arduino UNO, com a taxa de 9600 bits/s, e termina com a conexão com o Adafruit, através da função “io.connect”. Ambos apresentados na figura 65.

Figura 65 – Início do setup ESP-01 Arduino

```
void setup() {
  // Inicializando o Serial com 9600 bits de taxa de comunicação
  Serial.begin(9600);
  //Conecta-se ao Adafruit
  io.connect();
```

Fonte: Elaborada pelo autor, 2022

É feita então, a chamada das funções “nomeDoFeed->onMessage(função)”, que chamam a função quando o *Feed* é ativado no site da Adafruit. Cada variável vai chamar uma função diferente, com o objetivo de possibilitar a transmissão de mensagens claras ao Arduino. Também é feita uma chamada inicial de cada *Feed* no setup com a função “get()” e finaliza-se o setup. Ambas chamadas são apresentadas na figura 66.

Figura 66 – Funções “onMessage” e “get” do ESP-01 Arduino

```
// Chamada das funções que ocorrem quando determinado Feed é alterado
  luzBanheiro->onMessage(mensagemRecebidaBanheiro);
  luzTrabalho->onMessage(mensagemRecebidaTrabalho);
  luzLazer->onMessage(mensagemRecebidaLazer);
  luzExtra->onMessage(mensagemRecebidaExtra);
  intensidadeBanheiro->onMessage(mensagemRecebidaBanheiro2);
  intensidadeTrabalho->onMessage(mensagemRecebidaTrabalho2);
  intensidadeLazer->onMessage(mensagemRecebidaLazer2);
  intensidadeExtra->onMessage(mensagemRecebidaExtra2);

// Inicializa todos os valores uma vez no setup
  luzBanheiro->get();
  luzLazer->get();
  luzTrabalho->get();
  luzExtra->get();
  intensidadeBanheiro->get();
  intensidadeTrabalho->get();
  intensidadeLazer->get();
  intensidadeExtra->get();
```

Fonte: Elaborada pelo autor, 2022

Após o setup, será inicializado e finalizado o loop, com apenas a função “io.run()”, que é responsável por manter a conexão com o servidor web e processar os dados que forem trocados. Essa função é mostrada na figura 35.

Para finalizar são implementadas as oito funções que são acionadas dependendo de qual *Feed* for acionado, sendo elas: “mensagemRecebidaBanheiro”, “mensagemRecebidaBanheiro2”, “mensagemRecebidaTrabalho”, “mensagemRecebidaTrabalho2”, “mensagemRecebidaLazer”, “mensagemRecebidaLazer2”, “mensagemRecebidaExtra” e “mensagemRecebidaExtra2”. Elas são apresentadas pela figura 67.

Essas funções são responsáveis por mandar dados para o Arduino UNO através da porta serial, e por isso, é utilizado o comando “Serial.println(mensagem)”, para mandar esses dados, onde a ‘mensagem’ é o dado passado. Para criar a ‘mensagem’ que será enviada, foi criada uma variável String (variável de caracteres) em cada função.

O primeiro caractere dessa variável representará qual *Feed* foi acionado. Para o *Feed* de ‘luz trabalho’ será utilizado o caractere “T”, para “luz banheiro” será o “B”, para “luz lazer” será o “L” e para “luz extra” será o “E”.

O segundo caractere vai representar a função do *Feed* acionado. Sendo assim, quando for acionado um *Feed* de ligar e desligar, o caractere usado será “E”, para representar estado, e quando o *Feed* acionado for de intensidade será utilizado “I”.

O resto dos caracteres vai representar o valor advindo do *Feed*, logo, a função “data->value” que adquire o valor do *Feed*, será adicionada dentro da função String(), para que então, seu valor seja adicionado como os últimos caracteres da ‘mensagem’. O código do Arduino UNO será responsável por ler e identificar essa mensagem.
A figura 67 apresenta essas funções.

Figura 67 – Funções de acionamento e envio de *Feed* ESP-01 Arduino

```
// IMPLEMENTO DE FUNÇÕES

void mensagemRecebidaBanheiro(AdafruitIO_Data *data) {
  String banheiro;
  banheiro = "BE" + String(data->value());
  Serial.println(banheiro);
}
void mensagemRecebidaBanheiro2(AdafruitIO_Data *data) {
  String banheiro;
  banheiro = "BI" + String(data->value());
  Serial.println(banheiro);
}
void mensagemRecebidaTrabalho(AdafruitIO_Data *data) {
  String trabalho;
  trabalho = "TE" + String(data->value());
  Serial.println(trabalho);
}
void mensagemRecebidaTrabalho2(AdafruitIO_Data *data) {
  String trabalho;
  trabalho = "TI" + String(data->value());
  Serial.println(trabalho);
}
void mensagemRecebidaLazer(AdafruitIO_Data *data) {
  String lazer;
  lazer = "LE" + String(data->value());
  Serial.println(lazer);
}
void mensagemRecebidaLazer2(AdafruitIO_Data *data) {
  String lazer;
  lazer = "LI" + String(data->value());
  Serial.println(lazer);
}
void mensagemRecebidaExtra(AdafruitIO_Data *data) {
  String extra;
  extra = "EE" + String(data->value());
  Serial.println(extra);
}
void mensagemRecebidaExtra2(AdafruitIO_Data *data) {
  String extra;
  extra = "EI" + String(data->value());
  Serial.println(extra);
}
```

Fonte: Elaborada pelo autor, 2022

Após montagem do circuito, a montagem da API e a programação do ESP-01, é feita a programação do Arduino UNO. Começando com a definição dos pinos PWM dos LEDs que serão usados e a declaração das variáveis de estado e intensidade que serão utilizadas. Cada variável começa com seu identificador: “T” para trabalho, “B” para banheiro, “L” para lazer e “E” para extra. Como é apresentado na figura 68.

Figura 68 - Definição dos pinos e variáveis Arduino

```
// DEFINIÇÕES DE SAIDAS
#define ledTrabalho 5
#define ledLazer 6
#define ledBanheiro 10
#define ledExtra 11

// DECLARAÇÃO DE VARIAVEIS
int TEstado;
int LEstado;
int BEstado;
int EEstado;

double TIntensidade;
double LIntensidade;
double BIntensidade;
double EIntensidade;
int inAux;  //Variavel auxiliar
```

Fonte: Elaborada pelo autor, 2022

Logo após, é montado o setup do programa, que primeiramente inicializa as portas seriais, com sua taxa de transferência igual a 9600 bits, sendo a mesma das portas seriais do ESP-01. Junto a isso, também configura os pinos das leds como pinos de saída e inicializa as variáveis de estado e intensidade de todas as leds como ‘0’, para que elas estejam desligas. A criação do setup é demonstrada na figura 69.

Figura 69 – Setup do Arduino

```
void setup() {
  // Inicializando as portas seriaias do Arduino com a mesma taixa do ESP-01
  Serial.begin(9600);

  //Configurando os pinos das leds como saida
  pinMode(ledTrabalho, OUTPUT);
  pinMode(ledLazer, OUTPUT);
  pinMode(ledBanheiro, OUTPUT);
  pinMode(ledExtra, OUTPUT);

  //Inicializando todas as leds desligadas e intensidades 0
  TEstado = 0;
  BEstado = 0;
  LEstado = 0;
  EEstado = 0;
  TIntensidade = 0;
  BIntensidade = 0;
  LIntensidade = 0;
  EIntensidade = 0;

}
```

Fonte: Elaborada pelo autor, 2022

Com o setup preparado, é inicializado o loop com a leitura do pino serial, checando se ele foi acionado com a função “Serial.available()” dentro de um comando “while” para ativar apenas quando a porta serial for acionada. Dentro desse loop “while” é lido o último valor da porta serial, com a função “Serial.readString()”, que são salvos na variável “mensagem”. É utilizado a variável do tipo “String” para a ler os dados como textos, o que torna a troca de informações mais clara e precisa.

A inicialização do loop com a leitura do serial é apresentada na figura 70.

Figura 70 – Início do loop do Arduino

```
void loop() {

// recebendo informação do serial do ESP-01
  while (Serial.available()) {
    String mensagem = Serial.readString(); // lendo a mensagem como string
```

Fonte: Elaborada pelo autor, 2022

Após a aquisição dos valores que precisam ser alterados, eles passam por vários comandos condicionais (função if()), para determinar quais LEDs devem ter seus valores alterados. A lógica desses comandos será a mesma criada para a ‘mensagem’ do ESP-01. O comando “analogWrite(pino, valor)” será usado, devido a sua capacidade de simular uma saída analógica utilizando PWM, e consequentemente, simular uma alteração na tensão de saída dos pinos. Essa alteração permite controlar a intensidade das LEDs conectadas aos pinos.

A lógica será aplicada sobre a variável “mensagem”, já que ela possui os dados que devem ser executados, utilizando principalmente do comando “variável.chatAt(posição)”, que retorna o caractere da variável em determinada posição. Com isso, a primeira condicional vai verificar o primeiro caractere da “mensagem”, que fica na sua posição 0, com a função “mensage.charAt(0)”. Esse caractere tem o dado de qual *Feed* a informação foi obtida.

Se esse caractere for ‘T’, as variáveis usadas serão as de trabalho: “ledTrabalho”, “TEstado” e “TIntensidade”.

Se for ‘B’, as variáveis serão as de banheiro: “ledBanheiro”, “BEestado” e “BIntensidade”.

Se for ‘L’, as variáveis serão as de lazer: “ledLazer”, “LEestado” e “LIntensidade”.

Se for ‘E’, as variáveis serão as de extra: “ledExtra”, “EEestado” e “EIntensidade”.

A figura 71 apresenta identificação genérica do primeiro caractere.

Figura 71 – Condicional do primeiro caractere

```
// Identificar a primeira letra para identificar a led que será alterada
if(mensagem.charAt(0) == 'X') // X = T ou B ou E ou L
```

Fonte: Elaborada pelo autor, 2022

Após o primeiro caractere ser identificado, o segundo será analisado, com a função “mensagem.charAt(1)”. O dado desse caractere identifica qual função do *Feed* que foi acionado. Se era de estado(ligar/desligar), o caractere será um “E”, entretanto, se for de intensidade, o caractere será um “I”.

Quando o caractere for “E”, dentro de sua condicional, será identificado o último caractere da ‘mensagem’, que contém a informação do estado: 1 para ligar e 0 para desligar.

Logo, quando o valor foi 1, a led deverá ser ligada, utilizando o comando “analogWrite(led,IntensidadeDaLed)”. Se não houver valores de intensidade, ou seja, ela for zero, seu valor será alterado para o valor máximo, que é 255, para que a luz seja ligada no máximo. Junto a isso, a variável de estado da led terá seu valor alterado para 1.

Já, quando for 0, a led acionada deve ser desligada, utilizando o comando “analogWrite(led,0)”, além disso, a variável de estado da led também tem seu valor igual a 0. A figura 72 mostra o código de tratamento geral quando a informação é do estado da led.

Figura 72 – Identificando e tratando a alteração de estado

```
// Identificar a segunda letra para identificar o que está sendo alterado
if(mensagem.charAt(1) == 'E') //alterar o estado
{
    if(mensagem.charAt(2) == 'l') // Ligar a led
    {
      if(XIntensidade == 0) // Se a intensidade for 0 a lampada vai ligar e alterar intensidade para a maxima
      {
        XIntensidade = 255;
      }
      //Liga a LED de trabalho, com a intensidade ja estabelecida
      analogWrite(ledY,XIntensidade); // Y = Trabalho ou Banheiro ou Lazer ou Extra
      XEstado = 1;                    // X = T ou B ou E ou L
    }
    else //Desligar a led (== 0)
    {
      analogWrite(ledY,0); // Y = Trabalho ou Banheiro ou Lazer ou Extra
      XEstado = 0; // X = T ou B ou E ou L
    }
}
```

Fonte: Elaborada pelo autor, 2022

Em vez disso, quando o caractere identificado for “I”, dentro de sua condicional, é identificado os últimos caracteres da mensagem, que contenham a informação da porcentagem da intensidade. Nesse caso, é necessário utilizar uma estrutura de repetição “for” para a leitura dos últimos caracteres, já que eles podem ter 1,2 ou 3 caracteres de tamanho (0 a 9, 10 a 99, 100). Esse “for” será utilizado para pegar os caracteres “mensagem.chatAt(i)”, onde ‘i’ é um valor de 2 (3 caracteres) até o tamanho total da ‘mensagem’, que será descoberto com a função “mensagem.length()”.

Com esses valores salvos em uma String auxiliar, ela pode ser transformada e salva em uma variável inteira. Essa variável inteira terá o valor em porcentagem da intensidade da led, logo, para aplicá-la na variável de intensidade, é necessário multiplicar o valor por 2,55 (valor/100 * 255), já que o valor da intensidade pode variar de 0 a 255 no pino da led.

Por último, é aplicado uma condicional para checar se a led usada está ligada. Se ela tiver ligada, a função “analogWrite(led,IntensidadeDaLed)” é aplicada, e com isso, o valor da intensidade da led é alterado. Caso a led esteja desligada, o valor apenas será salvo na variável de intensidade para futuros usos.

A figura 73 apresenta o modelo geral de tratamento de informações de “I”, ou seja, intensidade.

Figura 73 – Identificando e tratando a alteração de Intensidade

```
else// if(mensagem.charAt(1) == 'I')Alterar a intensidade
{
  String M = "";
  for(int i = 2; i<mensagem.length();i++) // usa esse for para contemplar os 3 tamanhos de intensidade, de 0a9, 11 a 99 e 100
  {
    M = M + mensagem.charAt(i); //M vai receber o valor da intensidade em um string
  }

  int inAux =  M.toInt();
  XIntensidade = inAux*2.55;   // Multiplicação por 2.55, queremos o valor resultante da porcentagem. Onde o valor máximo é 255.(valor/100 * 255)
  // X = T ou B ou E ou L

  if(XEstado){ //Se a luz estiver ligada, altera a sua intensidade, se nao, apenas salva o valor
  analogWrite(ledY,XIntensidade);// X = T ou B ou E ou L
                                 // Y = Trabalho ou Banheiro ou Lazer ou Extra
  }// Altera a intensidade da saida da led
}
```

Fonte: Elaborada pelo autor, 2022

A figura 74 apresentada o tratamento geral de uma informação adquirida pela porta serial.

Figura 74 – Tratamento da informação advinda do ESP-01 para o Arduino

```
//exemplo: TE0
//exemplo: BI50

// Identificar a primeira letra para identificar a led que será alterada
if(mensagem.charAt(0) == 'X') // X = T ou B ou E ou L
 {// Identificar a segunda letra para identificar o que está sendo alterado
  if(mensagem.charAt(1) == 'E') //alterar o estado
  {
      if(mensagem.charAt(2) == '1') // Ligar a led
      {
        if(XIntensidade == 0) // Se a intensidade for 0 a lampada vai ligar e alterar intensidade para a maxima
        {
          XIntensidade = 255;
        }
        //Liga a LED de trabalho, com a intensidade ja estabelecida
        analogWrite(ledY,XIntensidade); // Y = Trabalho ou Banheiro ou Lazer ou Extra
        XEstado = 1;                    // X = T ou B ou E ou L
      }
      else //Desligar a led (== 0)
      {
        analogWrite(ledY,0); // Y = Trabalho ou Banheiro ou Lazer ou Extra
        XEstado = 0; // X = T ou B ou E ou L
      }
  }
  else// if(mensagem.charAt(1) == 'I')Alterar a intensidade
  {
    String M = "";
    for(int i = 2; i<mensagem.length();i++) // usa esse for para contemplar os 3 tamanhos de intensidade, de 0a9, 11 a 99 e 100
    {
      M = M + mensagem.charAt(i); //M vai receber o valor da intensidade em um string
    }

    int inAux =  M.toInt();
    XIntensidade = inAux*2.55;   // Multiplicação por 2.55, queremos o valor resultante da porcentagem. Onde o valor máximo é 255.(valor/100 * 255)
    // X = T ou B ou E ou L

    if(XEstado){ //Se a luz estiver ligada, altera a sua intensidade, se nao, apenas salva o valor
    analogWrite(ledY,XIntensidade);// X = T ou B ou E ou L
                                   // Y = Trabalho ou Banheiro ou Lazer ou Extra
    }// Altera a intensidade da saida da led
  }
}
```

Fonte: Elaborada pelo autor, 2022

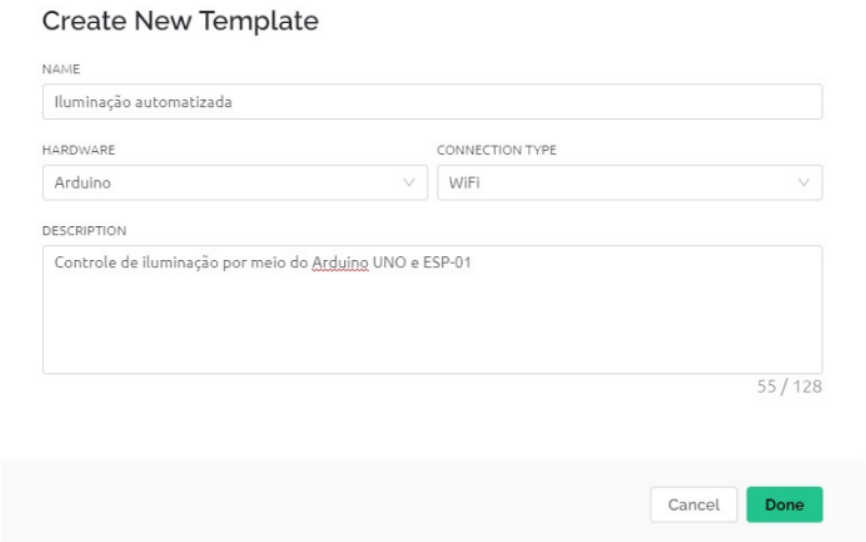
Com a API montada e a programação do ESP-01 e do Arduino UNO finalizada, e uma conexão Wi-Fi, é possível fazer o controle dos LEDs, a partir da plataforma web, de uma forma rápida e acessível.

É importante ressaltar que, caso as portas RX(0) e TX(1) do Arduino UNO, estejam conectadas ao circuito durante a gravação do programa, ele poderá gerar erros, já que ambas são utilizadas para a comunicação com a própria placa.

3.4.2.3 Ambiente de controle com Blynk

O ambiente Blynk foi utilizado para a criação um ambiente de controle. Inicia-se com a criação de um *template* denominado “Iluminação automatizada”, no qual a API será criada. Ele fará uso do hardware “Arduino”, e terá a conexão do tipo “WiFi”, a partir do ESP-01. A figura 75 apresenta a sua criação.

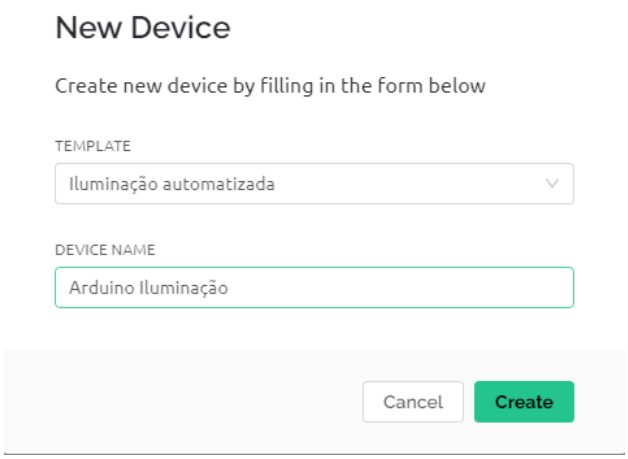
Figura 75 – Criação do *template* Iluminação automatizada



Fonte: Modificada de blynk.cloud, 2022

É necessário adicionar um dispositivo nesse *template*, para que, a partir disso, seja criado o *token* de autenticação, que será inserido, junto com o nome e Id do dispositivo, no gerador de códigos para o microcontrolador. Portanto, para criar um dispositivo, será acessado o menu de busca e usada a opção de criar um dispositivo a partir de um *template* como apresentado na Figura 31. A figura 76 apresenta a configuração desse dispositivo

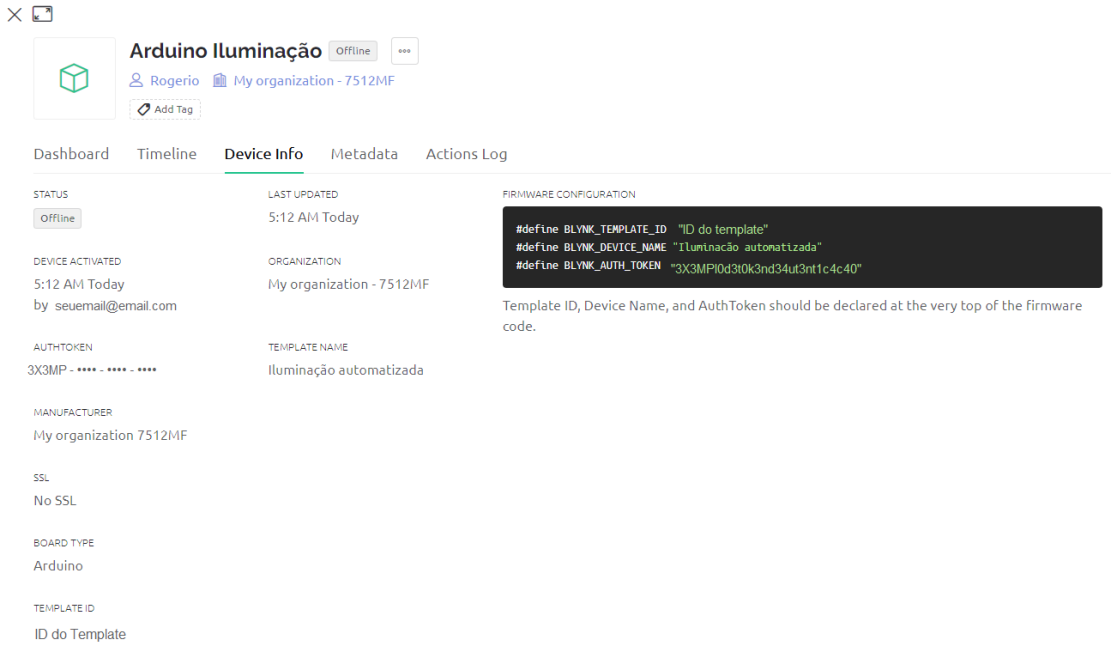
Figura 76 – Configurando o novo dispositivo



Fonte: Modificada de blynk.cloud, 2022

Com isso, os códigos de Id e nome do dispositivo, e o *token* de autenticação serão criados, e disponibilizados na aba “Device Info” do dispositivo criado, como apresentado na figura 77.

Figura 77 – Informações do dispositivo adicionado

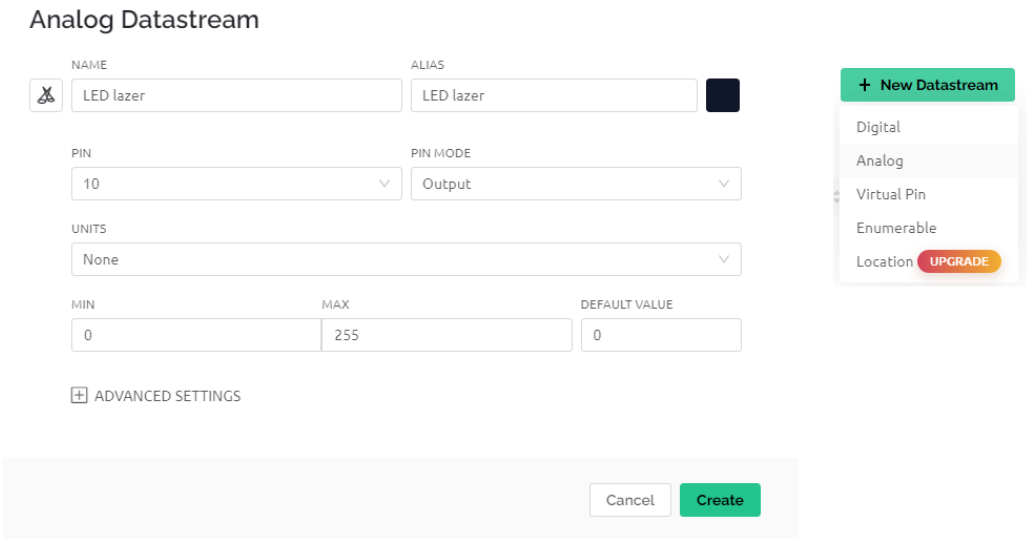


Fonte: Modificada de blynk.cloud, 2022

Para personalizar a API, edita-se o *template*, inicializando com a criação das “datastreams”, ou seja, as variáveis que controlarão o microcontrolador. Elas serão criadas como analógicas, já que controlam os pinos digitais PWM simulando uma saída analógica, com variância de valores, que consequentemente, controlará a intensidade das LEDs. Os pinos utilizados serão os pinos PWM 5, 6, 10 e 11, com variação de 0 a 255, que são, respectivamente, o mínimo (desligada) e o máximo de tensão dos pinos.

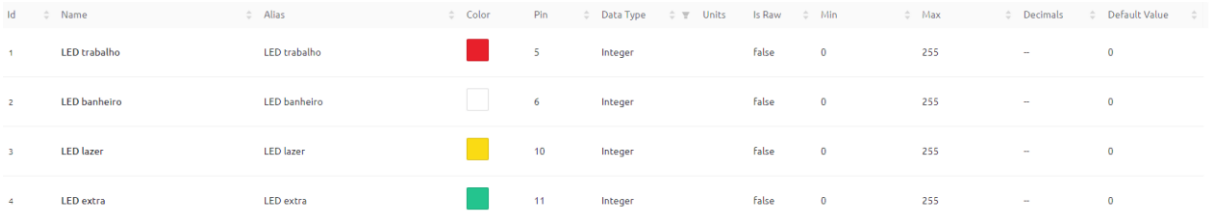
A figura 78 mostra a criação das variáveis “datastreams” e a figura 79 todas as “datastreams”.

Figura 78 – Criando “Datastream” da LED lazer



Fonte: Modificada de blynk.cloud, 2022

Figura 79 - Todas “Datastreams” criadas

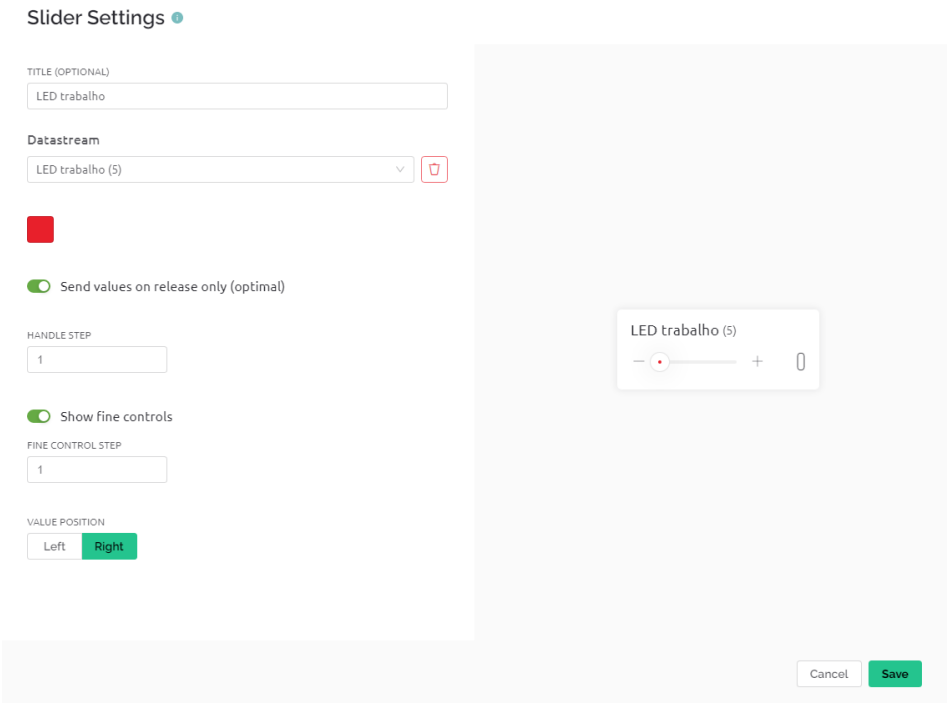


Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Max	Decimals	Default Value
1	LED trabalho	LED trabalho		5	Integer		false	0	255	–	0
2	LED banheiro	LED banheiro		6	Integer		false	0	255	–	0
3	LED lazer	LED lazer		10	Integer		false	0	255	–	0
4	LED extra	LED extra		11	Integer		false	0	255	–	0

Fonte: Modificada de blynk.cloud, 2022

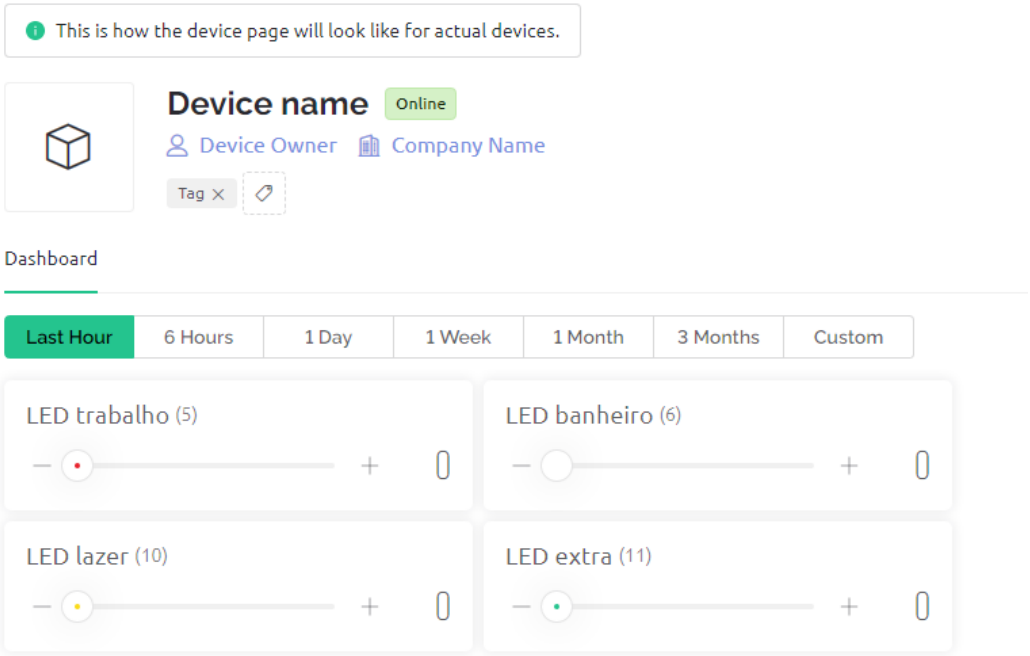
Com as variáveis criadas, podemos criar a API web. Ela terá, para controle da intensidade de todas as LEDs, quatro *widgets* de deslizar, cada um ligado ao seu “datastream” correspondente. Como demostrado nas figuras 80 e 81

Figura 80 – Criação do bloco deslizante



Fonte: Modificada de blynk.cloud, 2022

Figura 81 – API web de controle

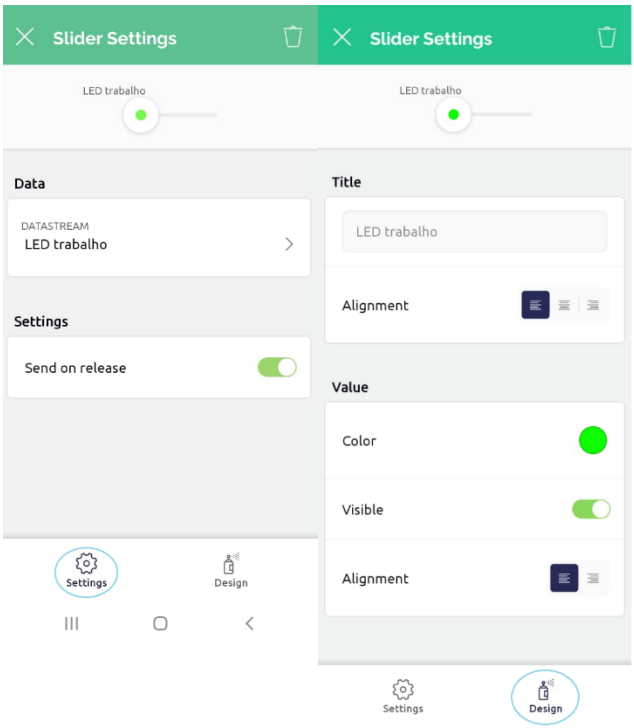


Fonte: Modificada de blynk.cloud, 2022

Essa API web, permite o controle do microcontrolador por meio do navegador.

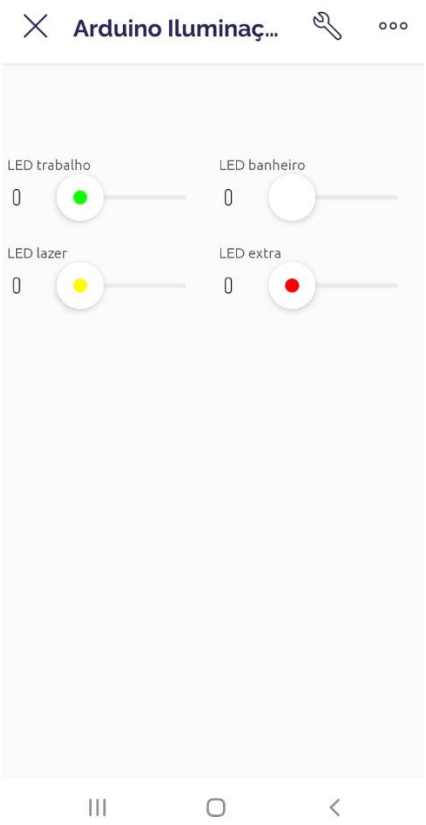
Após a criação das variáveis e da API de controle web, cria-se a API de controle mobile. Após o download do aplicativo e *Log In* no usuário, entra-se no modo de desenvolvimento, para acessar o *template* criado. Com isso, a inserção dos blocos deslizantes é possibilitada. Assim como na API web, a API mobile também usará as "datastream" criadas para os pinos 5, 6, 10 e 11. A figura 82 apresenta as configurações dos *widgets* deslizantes de controle, enquanto a figura 83 apresenta a API mobile.

Figura 82 – Configuração *widgets* API mobile



Fonte: Modificada do aplicativo Blynk, 2022

Figura 83 – API de controle mobile

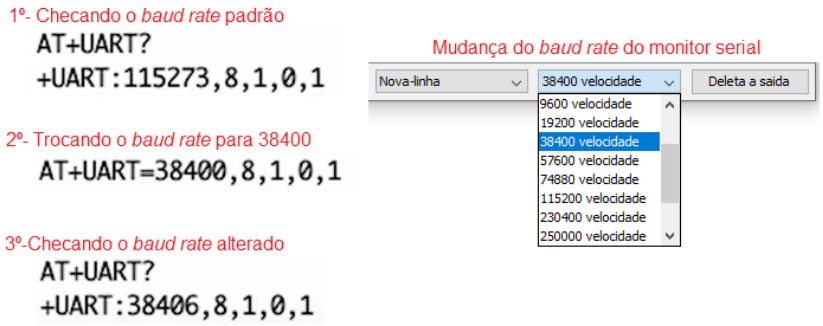


Fonte: Modificada do aplicativo Blynk, 2022

Após a finalização dos ambientes web e mobile, os dispositivos estarão disponíveis em ambos os ambientes para o controle. Então, para que possa ser feita a interação dos ambientes com o microcontrolador, é necessário o upload do código pela Arduino IDE diretamente no Arduino UNO.

Entretanto, é necessário ajustar a taxa de comunicação (*baud rate*) do ESP-01, para 38440, que é o padrão do Blynk, para prevenir erros de comunicação entre os dispositivos. Para isso, conecta-se o ESP-01 em modo normal, e, com a IDE Arduino, acessa-se o monitor serial. O *baud rate* padrão do ESP-01 é 155273, que pode ser visto com o comando “AT+UART?” no monitor serial. Para alterá-lo para 38400, utiliza-se o comando “AT+UART = 38400,8,1,0,1”. É importante também alterar o valor dessa velocidade no monitor serial. A imagem 84 apresenta os comandos para alterar e verificar o *baud rate* e a mudança da velocidade no monitor serial.

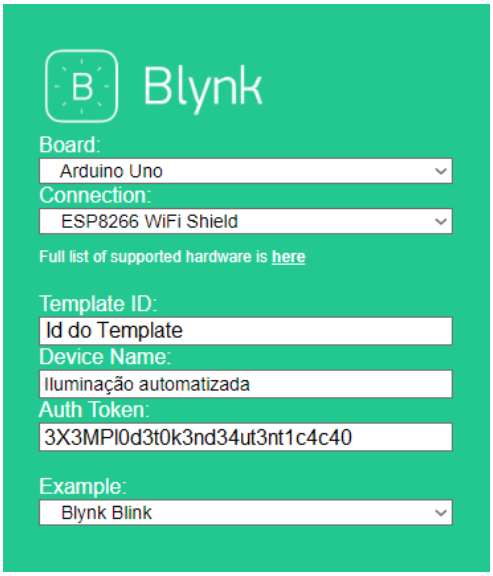
Figura 84 – Comandos e velocidade do monitor serial



Fonte: Elaborada pelo autor, 2022

Para montagem desse código, é possível utilizar o gerador de códigos disponibilizado pela Blynk, como base. Após a inserção dos dados necessários, e seleção do tipo de código (no caso o Blynk blink), um código será gerado, e a partir dele, será montado o código para o Arduino UNO, utilizando o ESP-01 (ESP8266) para conexão Wi-Fi. A figura 85 apresenta as configurações utilizadas.

Figura 85 – Configurações do gerador de códigos



Fonte: Modificada de examples.blynk, 2022

Com o código exemplo pronto, pode-se transpor ele para a IDE Arduino e editá-lo. As informações de Id do *template*, nome do dispositivo e o *token* de autenticação serão preenchidos automaticamente com as informações da figura 82. A figura 86 apresenta a definição dessas informações e da função de mensagem do monitor serial.

Figura 86 – Informações para conexão Blynk e mensagem do monitor serial

```
// Template ID, Nome dispositivo e Token de autenticação que são disponibilizados em Blynk.Cloud
// DEFINIÇÕES PARA CONEXÃO COM BLYNK.CLOUD
#define BLYNK_TEMPLATE_ID           "TMPLw88WmLZQ"
#define BLYNK_DEVICE_NAME           "Iluminação automatizada"
#define BLYNK_AUTH_TOKEN            "KSA6Nwtq6YqFrDlQBeF6nSPtY6eT8Rq9"


//BLYNK_PRINT É UTILIZADO PARA MOSTAR, NO MONITOR SERIAL, MENSAGENS DE CONEXÃO
#define BLYNK_PRINT Serial
```

Fonte: Elaborada pelo autor, 2022

Algumas das variáveis utilizadas na função de conexão com o Blynk são definidas, sendo elas o *token* de autenticação e nome e senha da rede Wi-Fi. Como apresentado na figura 87.

Figura 87 – Variáveis importantes para a conexão com o Blynk

```
//VARIÁVEIS UTILIZADAS NA FUNÇÃO DE CONEXÃO

char auth[] = BLYNK_AUTH_TOKEN;

// Suas informações de Wi-Fi.
// Use password como "" quando o Wi-Fi é aberto.
char ssid[] = "NomeDaRede";
char pass[] = "SenhaDaRede";
```

Fonte: Elaborada pelo autor, 2022

Para o ESP-01 conectar à internet, e comunicar-se com a placa Arduino UNO e com o servidor Blynk, são utilizadas bibliotecas advindas da Blynk, e definidos os parâmetros do ESP-01, como portas de comunicação serial e *baud rate*. A figura 88 apresenta essas definições.

Figura 88 – Bibliotecas e definições para o ESP-01 Blynk

```
//Bibliotecas usadas para conexão com o ESP-01 e blynk
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>

// Usado em Software Serial no Arduino Uno, Nano...
//Inclusão da bibliteca de comunicação com o ESP-01
#include <SoftwareSerial.h>

//Definição dos pinos serial utilizados para comunicação Arduino UNO e ESP-01
SoftwareSerial EspSerial(0, 1); // RX, TX

// ESP8266(ESP-01) baud rate, ou seja, velocidade de comunicação previamente adquirida:
#define ESP8266_BAUD 38400

//Conexão com o ESP8266(ESP-01) através das portas 0 e 1
ESP8266 wifi(&EspSerial);
```

Fonte: Elaborada pelo autor, 2022

Com todas as definições, o setup é inicializado, no qual são inicializados a comunicação serial do Arduino UNO com velocidade de 115200, a comunicação serial do ESP-01 com velocidade de 38400(*baud rate*) e a conexão com o servidor Blynk.

A função de conexão com o Blynk, “Blynk.begin(dados)” necessita do *token* de autenticação, a comunicação Wi-Fi (definida na comunicação com ESP-01) e o nome e senha do Wi-Fi. É possível definir diretamente qual servidor será conectado (pode ser automático), assim, pode conectar-se a um endereço IP específico ou, no caso desse projeto, ao servidor “blynk.cloud”. A figura 89 apresenta a função setup.

Figura 89 - Setup da programação do Arduino UNO Blynk

```
void setup()
{
  // Inicialiazanto as portas seriais do Arduino UNO com 115200 de velocidade.
  Serial.begin(115200);

  // Inicializanto a porta serial do ESP com o seu valor de baud rate
  EspSerial.begin(ESP8266_BAUD);
  delay(10);
  //Inicializando a conexão com o servidor Blynk
    Blynk.begin(auth, wifi, ssid, pass, "blynk.cloud", 80);

  //Pode-se também não especificar o server:
  //Blynk.begin(auth, wifi, ssid, pass);

  //pode ser usado para conectar-se a um IP específico:
  //Blynk.begin(auth, wifi, ssid, pass, IPAddress(192,168,1,100), 8080);
}
```

Fonte: Elaborada pelo autor, 2022

É possível, se necessário, obter o valor das variáveis de controle do Blynk, com a função “BLYNK_WRITE(Valor)”, na qual o “Valor” é adquirido e pode ser salvo em alguma variável do Arduino. Como apresentado na figura 90.

Figura 90 – Função de adquirir valor Blynk

```
/*
* Essa função é chamada sempre que o widget de datastream V1
* for acionado
*/
BLYNK_WRITE(V1)// V1 é o nome da datastream da qual o valor será obtido
{
  int ValorDoPino = param.asInt(); // Atribuindo o valor de V1 a uma variável ValorDoPino
  // Pode ser usada string ou double para alterar o tipo da variável no qual o valor será atribuido:
  // String i = param.asStr();
  // double d = param.asDouble();
}
```

Fonte: Elaborada pelo autor, 2022

Por último, no loop, apenas a função “Blynk.run()” estará presente para esse projeto. Essa função é responsável por manter a conexão da placa com o servidor Blynk e fazer a troca de informações com a API de controle.

A adição de mais funcionalidades ao projeto pode ser feita sem alteração do funcionamento da comunicação com Blynk.

A figura 91 apresenta a função loop.

Figura 91 – Loop da programação do Arduino UNO Blynk

```
void loop()
{
  Blynk.run();
  // Pode-se fazer a inserção de códigos sem alteração do funcionamento.
  // É recomendado evitar a função delay()!
}
```

Fonte: Elaborada pelo autor, 2022

Com o código completo e aplicado a placa, é possível fazer o controle das LEDs, no ambiente web configurado, utilizando do navegador para mudança dos dados, e no ambiente mobile, no aplicativo da Blynk. Assim, o acesso a esses dados é facilitado e ampliado, mantendo a segurança dos dados por meio do usuário criado na plataforma.

4 CONSIDERAÇÕES FINAIS

O presente trabalho apresenta como proposta, a integração de uma iluminação automatizada residencial com a internet, possuindo o objetivo de criar ambientes funcionais a partir dos gostos ou necessidades do usuário, atuando sobre o controle da luz. Para essa integração, foram utilizadas LEDs conectados a microcontroladores com acesso à internet e um ambiente mobile web, para permitir o fácil acesso e controle da iluminação.

Com base nos estudos realizados sobre luz e cores, e suas implicações no ser humano, constata-se a grande importância e impacto da constante exposição à luz, principalmente a luz artificial. Em razão disso, a implementação do controle automatizado, em conjunto à uma análise apropriada dos diversos casos de uso da iluminação, fará com que a interação, entre usuário e ambiente, seja mais confortável e satisfatória, aumentando sua comodidade e conforto.

A partir desses estudos também é possível idealizar a construção de perfis predefinidos para diferentes situações e necessidades, que quando incorporados aos perfis personalizados, oferecem liberdade de personalização e interação com a iluminação.

Entende-se que o usuário possa, a partir de um trabalho igual ou semelhante a esse, interagir de forma a que o ambiente fique com características que sejam pessoais, na composição das diversas aplicações em automação de forma pessoal, no presente caso, aplicado à iluminação.

Pelos estudos realizados, constantes no referencial teórico, é de inferência que seja possível desenvolver um sistema baseado em microcontrolador e ambiente Web, capaz de utilizar em computadores pessoais e ambientes mobiles diversos. Devido ao caráter do desenvolvimento web, é interessante racionalizar que qualquer ambiente que esteja conectado à internet, através de uma identificação do usuário, possa ter os parâmetros do sistema desenvolvido modificados.

O presente trabalho busca uma maior integração entre as áreas de arquitetura, design e engenharias elétrica e eletrônica com a engenharia da computação.

Com a montagem do protótipo neste trabalho, é possível inferir também, a acessibilidade e simplicidade de implementação da automação, em áreas específicas de uma residência.

Utilizando os estudos e aplicações IoT nos ambientes disponibilizados pela Adafruit e pela Blynk, é possível constatar a grande acessibilidade dos meios de comunicação da internet com dispositivos de controle residencial.

Com a utilização da Adafruit e seu servidor Web, é possível, através de diversos dispositivos com acesso à internet e um navegador Web, integrar a residência através do controle remoto de algumas de suas partes. Assim, com uma API amigável e sem nenhum custo, é possível a criação de um ambiente de controle próprio e personalizado que concede uma maior liberdade na sua concepção. Além disso, com as bibliotecas específicas, disponibilizadas pela própria empresa, a programação fica menos complexa, descomplicando a criação de todo o projeto e exigindo um menor investimento.

Com a implementação do Blynk, a API amigável e sem custo também está presente, possibilitando a criação de um ambiente de controle personalizado. Entretanto, esse ambiente é um aplicativo mobile, dispensando o uso do navegador para a API. Assim, apesar de perder alguns recursos, como compatibilidade com quase todos os tipos de sistemas, o aplicativo tem a vantagem de poder acessar recursos do celular, ampliando ainda mais a personalização do ambiente de controle.

A plataforma também oferece a programação integralizada, necessitando de informações básicas, como modelo do microcontrolador, como o ESP-8266 e Arduino UNO e informações da rede para gerar o programa do modulo que conecta à internet.

Caso o desenvolvedor não utilize ambientes pré-montados para a criação da API de controle, pode-se utilizar de uma conexão direta a web, ou seja, o envio direto de informações para o ambiente web(http) criado diretamente. Essa aplicação pode ser utilizada de duas maneiras. Primeiro, com a criação do servidor web diretamente pela programação do ESP-01, ou seja, após conectar o módulo a um IP, utiliza-se a porta serial para enviar o código HTML junto com o valor das variáveis. Segundo, para um servidor web já construído, é possível utilizar a porta serial para fazer requisições diretamente a esse servidor web. Necessitando apenas das bibliotecas “ESP8266WiFi.h” e “ESP8266HTTPClient.h”.

4.1 Trabalhos futuros

Para aplicações futuras deste trabalho segue-se:

- Acoplar novos módulos ao Arduino, para aplicar o Bluetooth ou o infravermelho, como modo de conexão e interação com os LEDs.
- Implementar controle de foco, distância e movimento da luz.
- Desenvolver um ambiente de controle que possibilite o uso de sistemas IOS.
- Aumentar o escopo de perfis.
- Aumentar o número de dispositivos controlados pelo sistema.
- Ampliação do uso dos recursos das API implementadas.
- Ampliar os estudos sobre softwares de controle, aplicando mais deles.
- Usar diferentes placas para o controle, como a ESP32.

Trabalhos semelhantes podem ser aplicados na expansão da automação residencial, aplicando-a no controle de temperatura, controle de persianas, portas e janelas, controle de eletrodomésticos, e outros dispositivos de controle geral.

Além disso, também podem aumentar o escopo de softwares IoT, que facilitem a implementação e a acessibilidade do uso dessa automação e do controle da luz.

Os programas desenvolvidos no presente trabalho, são apresentados nos apêndices, o termo de autorização de publicação de produção acadêmica encontra-se no anexo 1.

REFERÊNCIAS

ABOUT Arduino. **Arduino**, 2021. Disponível em: < https://www.arduino.cc/en/about >. Acesso em: 31 de mar. de 2022.

AGARWAl, Tarun. *Different Types of Wireless Communication with Applications*. **Elprocus**, 2020. Disponível em: < https://www.elprocus.com/types-of-wireless-communication-applications/ >. Acesso em: 24 de mar. de 2022.

Agarwal, Tarun. *What is Bluetooth: Architecture & It's Working*. **Elprocus**, 2021. Disponível em: < https://www.elprocus.com/how-does-bluetooth-work/ >. Acesso em: 24 de mar. de 2022

Alecrim, Emerson. Bluetooth: o que é, como funciona e versões. **Infowester**, 2008. Disponível em: <https://www.infowester.com/bluetooth.php>. Acesso em: 24 de mar. de 2022.

ALEXANDRE, Iluminação: a importância da luz nos ambientes. **Htec Multimídia**, 2020. Disponível em: <https://htecmultimidia.com.br/iluminacao-a-importancia-da-luz-nos-ambientes/ >. Acesso em: 25 de abr. 2022.

ARAÚJO JÚNIOR, Antônio Pereira de; CHAGAS, Christiano Vasconcelos das; FERNANDES, Raphaela Galhardo. Uma rápida análise sobre automação industrial. **Redes para automação Industrial**, Rio Grande do Norte, v. 40, n. 1, p. 85-98, jan./abr. 2003. Disponível em: <https://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_6.pdf>. Acesso em: 16 mar. 2022.

ARDUINO UNO R3. UNO R3. **Arduino**, 2021. Disponível em: < https://docs.arduino.cc/hardware/uno-rev3 >. Acesso em: 31 de mar. de 2022.

ARDUINO. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2022. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Arduino&oldid=62935520>. Acesso em: 31 mar. 2022.
AUTOMAÇÃO. *In*: DICIO, Dicionário Online de Português. Porto: 7Graus, 2022. Disponível em: <https://www.dicio.com.br/automacao/>. Acesso em: 16/03/2022

BANDYOPADHYAY, Debasis.; SEN, Jaydip. *Internet of things: applications and challenges in technology and standardization. Wireless Personal Communications*, v. 58, n. 1, p. 49-69, Mar. 2011. DOI:10.1007/s11277-011-0288-5. Acesso em: 18 mar. 2022.

BHARTI, Monika; SAXENA, Sharad; KUMAR, Rajesh. *Intelligent resource inquisition framework on internet-of-things. Computers & Electrical Engineering,* v. 58, p. 265-281, fev. 2017. DOI: https://doi.org/10.1016/j.compeleceng.2016.12.023. Acesso em: 16 mar. 2022.

BRITANNICA, *The Editors of Encyclopaedia*. lighting. ***Encyclopedia Britannica***, 27 jun. 2017. Disponível em: <https://www.britannica.com/technology/lighting>. Acesso: 5 mai. 2022.

CHAVIER, Luís Fernando. Programação para Arduíno - Primeiros Passos. **Professor Luzerna IFC**, 2017. Disponível em: <https://professor.luzerna.ifc.edu.br/marcelo-cendron/wp-content/uploads/sites/40/2017/03/Programação-para-Arduino-Primeiros-Passos-Conceitos-iniciais-de-programação-para-Arduino-Projeto-de-eletrônica-modular-com-Arduino-Circuitar.pdf>. Acesso em: 10 de mai. de 2022

COMMUNICATION Using Infrared Technology. **Elprocus**, 2017. Disponível em: < https://www.elprocus.com/communication-using-infrared-technology/ >. Acesso em: 24 de mar. de 2022.

CURVELLO, André. Apresentando o módulo ESP8266. **Embarcados**, 2015. Disponível em: < https://www.embarcados.com.br/modulo-esp8266/ >. Acesso em: 08 de mai. 2022

DIAS, César Luiz de Azevedo; PIZZOLATO, Nélio Domingues. Domótica: Aplicabilidade e Sistemas de Automação Residencial. **Revista Vértices**, v. 6, n. 3, p. 9-32, 2004. DOI: https://doi.org/10.5935/1809-2667.20040015. Acesso em: 18 de mar. 2022

DOMÓTICA. In: WIKIPÉDIA, a enciclopédia livre. Flórida: *Wikimedia Foundation*, 2021. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Dom%C3%B3tica&oldid=62301013>. Acesso em: 18 out. 2021.

ELIENE. Domótica. **Mundo Educação,** 2011. Disponível em: < https://www.infoescola.com/tecnologia/domotica/ >. Acesso em: 18 de mar de 2022.

FIELD of view. In: WIKIPÉDIA, a enciclopédia livre. Flórida: *Wikimedia Foundation,* 2020. Disponível em: < https://en.wikipedia.org/wiki/Field_of_view >. Acesso em: 27 mar. 2020.

HELENE, Otaviano; HELENE, André Frazão. Alguns aspectos do olho humano. Rev. Bras. Ensino Fis. Artigos gerais, 10 ago. 2011. DOI: https://doi.org/10.1590/S1806-11172011000300012. Acesso em: 14 fev. 2022.

HELLER, Eva, 1948 - 2008. A psicologia das cores: como as cores afetam a emoção e a razão / Eva Heller; [tradução Maria Lúcia Lopes da Silva]. -- 1 ed. -- São Paulo: Gustavo Gili, 2013.

IEEE RAS UFCG. O Que É Um Microcontrolador? **Capítulo Estudantil IEEE**, 2020. Disponível em: <https://edu.ieee.org/br-ufcgras/o-que-e-um-microcontrolador/ >. Acesso em: 18 de mar de 2022.

INTRODUCTION Arduino. *What is Arduino?* **Arduino,** 2018. Disponível em: < https://www.arduino.cc/en/Guide/Introduction >. Acesso em: 31 de mar. de 2022.

JÚNIOR, Joab Silas da Silva. O que é infravermelho? **Brasil Escola**, 2017. Disponível em: <https://brasilescola.uol.com.br/o-que-e/fisica/o-que-e-infravermelho.html>. Acesso em 24 de março de 2022.

LOSS, Juliana. **Iluminação artificial residencial: a percepção do usuário de Curitiba em ambientes de descanso**. Orientador: Prof. Dr. Aloísio Leoni Schmid. 2013. 97f. TCC (Graduação) – Engenharia de Construção Civil, Setor de Tecnologia, Universidade Federal do Paraná, Paraná, 2013. Disponível em: <https://acervodigital.ufpr.br/bitstream/handle/1884/34632/R%20-%20D%20-%20JULIANA%20LOSS.pdf;jsessionid=FCA5EA7399B462C1AFFC9EDB59B31176?sequence=1> Acesso em: 5 mai. 2022.

MANAIA, Mariele B. Luz, cor e percepção: A influência da iluminação no comportamento humano. **LUME Arquitetura**, Paraná, p. 45-71, 2010. Disponível em: < https://facnopar.com.br/conteudo-arquivos/arquivo-2019-08-28-15670309785134.pdf >. Acesso em: 5 mai. 2022.

MARTINS, Juliana C.; BELENKI, Samara A.; SANCHES Henrique L. C. Iluminação e sua influência nos usuários da edificação: O bom e o mau projeto. **FACNOPAR**, Paraná, n. 1, p. 45-71, 2019. Disponível em: < https://facnopar.com.br/conteudo-arquivos/arquivo-2019-08-28-15670309785134.pdf >. Acesso em: 5 mai. 2022.

METTEDE, Henrique. Domótica - O que é e quais suas vantagens. **Mundo da Eletrica,** 2020. Disponível em: < https://www.mundodaeletrica.com.br/domotica-o-que-e-quais-as-vantagens/>. Acesso em: 18 de mar de 2022.

MÓDULO WiFi ESP8266 – ESP-01. **Curto Circuito**, 2017. Disponível em: < https://www.curtocircuito.com.br/modulo-wifi-esp8266-esp-01.html/ >. Acesso em: 08 de mai. 2022

MURCH, M. Gerald. Physiological Principles for the effective Use of Colors.IEEE Computer Graphics and Applications, V 4. #11*,* nov. 1984*.*

NISHIDA, Silvia M. **Sentido da visão**. Botucatu, SP: Departamento de Fisiologia, Unesp –Botucatu, 2012. Disponível em <https://www.biologia.bio.br/curso/1º%20período%20Faciplac/08.sentido_visao.pdf> Acesso em: 27 mar. 2022.

OVERVIEW of Arduino IDE 1. **Arduino,** 2022. Disponível em:< https://docs.arduino.cc/software/ide-v1/tutorials/Environment >. Acesso em:10 mai. 2022

PLACA DE ENSAIO. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2021. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Placa_de_ensaio&oldid=60659265>. Acesso em: 16 mar. 2021.

PRODUCT Arduino. **Arduino**, 2018. *Products*. Disponível em: < https://www.arduino.cc/en/Guide/Introduction >. Acesso em: 31 de mar. de 2022.

PRÓSPERO, Israel. Microcontroladores: o que são e aplicações. **esscjr**, 2020. Disponível em: < https://eescjr.com.br/blog/microcontroladores-o-que-sao-e-aplicacoes/>. Acesso em: 17 de mar de 2022.

RAMBAUSKE, Ana Maria. Decoração e Design de Interiores: Teoria da Cor. s.d. Disponível em: <https://hosting.iar.unicamp.br/lab/luz/ld/Cor/teoria-da-cor.pdf>. Acesso em: 2.9 fev. 2022.

SEN, Jaydip. *Internet of Things - Technology, Applications and Standardization*. *IntechOpen*. 2018 134 p. DOI: https://doi.org/10.5772/intechopen.70907. Acesso em: 18 mar. 2022

SILVEIRA,Cristiano Bertulucci. O que é automação industrial. **Citi Systems**, 2011. Disponível em: <https://www.citisystems.com.br/o-que-e-automacao-industrial/#>. Acesso em: 17 de mar de 2022.

SOARES,Cláudio. Cores. **Claudio Soares Designer**, 2019. Disponível em: <https://claudiosoaresdesigner.com.br/cores/>. Acesso em: 12 de mar de 2022.

SOUZA, Fábio. Arduino UNO. **Embarcados**, 2013. Disponível em: <https://www.embarcados.com.br/arduino-uno/ >. Acesso em: 06 de abr. de 2022

VENKATRAMAN, S.; OVERMARS, A.; THONG, M. *Smart Home Automation—Use Cases of a Secure and Integrated Voice-Control System. **Systems***, v. 9, n. 4, p. 77, 28 out. 2021. DOI: https://doi.org/10.3390/systems9040077. Acesso em 28 abr. 2022.

WORTMEYER, Charles; FREITAS, Fernando; CARDOSO, Líuam. Automação Residencial: Busca de Tecnologias visando o Conforto, a Economia, a Praticidade e a Segurança do Usuário. **II Simpósio de Excelência em Gestão e Tecnologia - SEGeT**, Rio de Janeiro, n. 1, p. 1065-1067, 2005. Disponível em: <https://www.aedb.br/seget/arquivos/artigos05/256_SEGET%20-%20Automacao%20Residencial.pdf>. Acesso em: 18 abr. 2022.

FAGUNDES, Eduardo M. Automação comercial. **Efagundes,** 2020. Disponível em: < https://efagundes.com/artigos/automacao-comercial/>. Acesso em: 15 de jun de 2022.

PIMENTA, Rafael. Como programar ESP8266 ESP-01 como StandAlone com Arduino. **Eletrofun**, 2021. Disponível em: < https://www.electrofun.pt/blog/como-programar-esp8266-esp-01-como-standalone-com-arduino/>. Acesso em: 22 de set. 2022.

RUBELL, Brent. *Welcome to Adafruit IO: Overview*. ***Learn Adafruit***, 2013a. Disponível em: < https://learn.adafruit.com/welcome-to-adafruit-io/overview>. Acesso em: 7 set. 2022

RUBELL, Brent. *Welcome to Adafruit IO: What is Adafruit IO*? ***Learn Adafruit***, 2013b. Disponível em: < https://learn.adafruit.com/welcome-to-adafruit-io/what-is-adafruit-io>. Acesso em: 7 set. 2022

RUBELL, Brent. *Quickstart: Adafruit IO WipperSnapper*. ***Learn Adafruit*** ,2021. Disponível em: < https://learn.adafruit.com/quickstart-adafruit-io-wippersnapper >. Acesso em: 10 out. 2022

OLIVEIRA, Euler. Conhecendo o NodeMCU-32S ESP32. **MasterWalker** 2017. Disponível em: < https://blogmasterwalkershop.com.br/embarcados/esp32/conhecendo-o-nodemcu-32s-esp32>. Acesso em: 23 de set. 2022.

MQTT. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2022. Disponível em: <https://pt.wikipedia.org/w/index.php?title=MQTT&oldid=64055661>. Acesso em: 22 jul. 2022.

BLYNK, Introduction: Welcome to Blynk Documentation. **Blynk io,** 2022a. Disponível em: < https://docs.blynk.io/en/ >. Acesso em: 31 de mar. de 2022.

BLYNK, Plataform overview: Products. **Blynk io,** 2022b. Disponível em: < https://docs.blynk.io/en/platform-overview/products >. Acesso em: 31 de mar. de 2022

Anexo 1 - Termo de autorização de publicação de produção acadêmica

PUC GOIÁS

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR
Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO nº 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Rogério Melo Loes
do Curso de Engenharia de Computação, matrícula 2018.1.0033.0315-4,
telefone: (62)981250018 e-mail rogerlopes1999@hotmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Iluminação automatizada para intergração do ambiente residencial, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 29 de setembro de 2022.

Assinatura do autor: Rogério Melo Lopes

Nome completo do autor: Rogério Melo Lopes

Assinatura do professor–orientador:

Nome completo do professor-orientador: Marcelo Antonia Adad de Araújo

APÊNDICES

APÊNDICE A- Código ESP-01 sozinho

```
/*
    AUTOR:      Rogério Melo Lopes
    DATA:      29/08/2022
*/
//inclusão da biblioteca
#include "AdafruitIO_WiFi.h"

// CONFIGURAÇÃO DA ADAFRUITIO
#define IO_USERNAME  "NomeDeUsuário"
#define IO_KEY "Key_aleatória"

// CONFIGURAÇÃO DO WIFI
#define WIFI_SSID "NomeDaRedeWiFi"
#define WIFI_PASS "SenhaDaRedeWiFi"

AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);

// INSTANCIANDO OBJETOS
AdafruitIO_Feed *luz1 = io.feed("luz1");
AdafruitIO_Feed *luz2 = io.feed("luz2");

//DEFININDO PINOS DE SAIDA DAS LEDS
#define LED1 2
#define LED2 3


void setup() {
  // Inicia a conexão mqtt com io.adafruit.com
  io.connect();

  //Define os pinos das leds como pinos de saida
  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);

  //Chama as funções mensagemRecebia e mensagemRecebida2 quando os Feeds
luz1 e luz2 trocarem estados
  luz1->onMessage(mensagemRecebida);
  luz2->onMessage(mensagemRecebida2);

  //Pega os valores dos feeds luz1 e luz2 quando se inicia o ESP-01
  luz1->get();
  luz2->get();
}

void loop() {
 //Mantém o cliente conectado ao site da adafruite
```

```
 //processa os dados da comnunicação
 io.run();
 //delay(500);
}

// IMPLEMENTO DE FUNÇÕES

//Função acionada para o Feed da led 1
void mensagemRecebida(AdafruitIO_Data *data) {

  if(data->value()) //Se o valor do Feed for positivo(1)
  {
    digitalWrite(LED1,HIGH); //Liga a led1
  }
  else //se o valor nao for positivo, ou seja, é negativo(0)
  {
    digitalWrite(LED1,LOW); //Desliga a led1
  }

}

//Função acionada para o Feed da led 2
void mensagemRecebida2(AdafruitIO_Data *data) {

  if(data->value()) //Se o valor do Feed for positivo(1)
  {
    digitalWrite(LED2,HIGH); //Liga a led2
  }
  else //se o valor nao for positivo, ou seja, é negativo(0)
  {
    digitalWrite(LED2,LOW); //Desliga a led2
  }

}
```

APÊNDICE B – Código ESP-01, quando em conjunto com o Arduino

```
/*
    AUTOR:          Rogério Melo Lopes
    DATA:           29/08/2022
*/
#include "AdafruitIO_WiFi.h"

// CONFIGURAÇÃO DA ADAFRUITIO
#define IO_USERNAME  "NomeDeUsuário"
#define IO_KEY "Key_aleatória"

// CONFIGURAÇÃO DO WIFI
#define WIFI_SSID "NomeDaRedeWiFi"
#define WIFI_PASS "SenhaDaRedeWiFi"

AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);

// INSTANCIANDO OBJETOS
AdafruitIO_Feed *luzBanheiro = io.feed("luz-banheiro");
AdafruitIO_Feed *luzTrabalho = io.feed("luz-trabalho");
AdafruitIO_Feed *luzLazer = io.feed("luz-lazer");
AdafruitIO_Feed *luzExtra = io.feed("luz-extra");
AdafruitIO_Feed *intensidadeBanheiro = io.feed("intensidadebanheiro");
AdafruitIO_Feed *intensidadeTrabalho = io.feed("intensidadetrabalho");
AdafruitIO_Feed *intensidadeLazer = io.feed("intensidadelazer");
AdafruitIO_Feed *intensidadeExtra = io.feed("intensidadeextra");


// DECLARAÇÃO DE VARIÁVEIS

void setup() {
  // Inicializando o Serial com 9600 bits de taxa de comunicação
  Serial.begin(9600);
  //Conecta-se ao Adafruit
  io.connect();

// Chamada das funções que ocorrem quando determinado Feed é alterado
  luzBanheiro->onMessage(mensagemRecebidaBanheiro);
  luzTrabalho->onMessage(mensagemRecebidaTrabalho);
  luzLazer->onMessage(mensagemRecebidaLazer);
  luzExtra->onMessage(mensagemRecebidaExtra);
  intensidadeBanheiro->onMessage(mensagemRecebidaBanheiro2);
  intensidadeTrabalho->onMessage(mensagemRecebidaTrabalho2);
  intensidadeLazer->onMessage(mensagemRecebidaLazer2);
  intensidadeExtra->onMessage(mensagemRecebidaExtra2);

// Inicializa todos os valores uma vez no setup
  luzBanheiro->get();
  luzLazer->get();
```

```
  luzTrabalho->get();
  luzExtra->get();
  intensidadeBanheiro->get();
  intensidadeTrabalho->get();
  intensidadeLazer->get();
  intensidadeExtra->get();
}

void loop() {
  //Mantém o ESP-01 conectado ao adafruit e aciona as funções "onMessage"
quando um Feed é alterado
   io.run();


//Caso o arduino for mandar alguma informação para o AdaFruit
//  if(Serial.available()){
//    String valorRecebido = Serial.readString();
//    Variavel->save(valorRecebido);
//  }

}

// IMPLEMENTO DE FUNÇÕES

//Acionada para alterações de estado na Luz banheiro
void mensagemRecebidaBanheiro(AdafruitIO_Data *data) {

  String banheiro;
  banheiro = "BE" + String(data->value());
  Serial.println(banheiro);

}
//Acionada para alterações de intensidade da luz banheiro
void mensagemRecebidaBanheiro2(AdafruitIO_Data *data) {

  String banheiro;
  banheiro = "BI" + String(data->value());
  Serial.println(banheiro);

}


//Acionada para alterações na Luz trabalho
void mensagemRecebidaTrabalho(AdafruitIO_Data *data) {

  String trabalho;
  trabalho = "TE" + String(data->value());
  Serial.println(trabalho);

  }
```

```
//Acionada para alterações de intensidade da luz trabalho
void mensagemRecebidaTrabalho2(AdafruitIO_Data *data) {

  String trabalho;
  trabalho = "TI" + String(data->value());
  Serial.println(trabalho);


}
//Acionada para alterações de estado na Luz lazer
void mensagemRecebidaLazer(AdafruitIO_Data *data) {

  String lazer;
  lazer = "LE" + String(data->value());
  Serial.println(lazer);


}
//Acionada para alterações de intensidade da luz lazer
void mensagemRecebidaLazer2(AdafruitIO_Data *data) {

  String lazer;
  lazer = "LI" + String(data->value());
  Serial.println(lazer);


}

//Acionada para alterações de estado na Luz extra
void mensagemRecebidaExtra(AdafruitIO_Data *data) {

  String extra;
  extra = "EE" + String(data->value());
  Serial.println(extra);


}

//Acionada para alterações de intensidade da luz extra
void mensagemRecebidaExtra2(AdafruitIO_Data *data) {

  String extra;
  extra = "EI" + String(data->value());
  Serial.println(extra);

}
```

APÊNDICE C – Código Arduino, em conjunto com o ESP-01

```
/*
    AUTOR: Rogério Melo Lopes
    DATA:  29/08/2022
*/

// DEFINIÇÕES DE SAIDAS
#define ledTrabalho 5
#define ledLazer 6
#define ledBanheiro 10
#define ledExtra 11

int TEstado;
int LEstado;
int BEstado;
int EEstado;

double TIntensidade;
double LIntensidade;
double BIntensidade;
double EIntensidade;
int inAux;  //Variavel auxiliar

void setup() {
  // Inicializando as portas seriais do Arduino com a mesma taixa do ESP-01
  Serial.begin(9600);

  //Configurando os pinos das leds como saida
  pinMode(ledTrabalho, OUTPUT);
  pinMode(ledLazer, OUTPUT);
  pinMode(ledBanheiro, OUTPUT);
  pinMode(ledExtra, OUTPUT);


  TEstado = 0;
  LEstado = 0;
  BEstado = 0;
  EEstado = 0;
  TIntensidade = 0;
  LIntensidade = 0;
  BIntensidade = 0;
  EIntensidade = 0;
}

void loop() {

// recebendo informação do serial do ESP-01
  while (Serial.available())
  {
```

```
   String mensagem = Serial.readString(); // lendo a mensagem como string

   // Tratando casos para idenficar qual led acender e sua intensidade
   // Mensagem = IdentificadorDaLed + EstadoDaLed
   // IdentificadorDaLed = T(ledTrabalho),L(ledLazer),B(ledBanheiro),E(ledExtra)
   // Tipo de alteração = E: Estado, 1 (ligar) ou 0 (desligar)
   //               e    I: Intensidade, 0 a 100%
   // Exemplos: TE1 = ledTrabalho,alterar estado, ligar
   //          BI50 = ledBanheiro,alterar intensidade, 50%


   // Identificar a primeira letra para identificar a led que será alterada

   //----------------------------------LEDTRABALHO---------------------
   if(mensagem.charAt(0) == 'T') //T = led de trabalho
    {// Identificar a segunda letra para identificar o que está sendo alterado
     if(mensagem.charAt(1) == 'E') //alterar o estado
     {
        if(mensagem.charAt(2) == '1') // Ligar a led
        {
          if(TIntensidade == 0) // Se a intensidade for 0 a lampada vai ligar e alterar
intensidade para a maxima
          {
            TIntensidade = 255;
          }
          //Liga a LED de trabalho, com a intensidade ja estabelecida
          analogWrite(ledTrabalho,TIntensidade);
          TEstado = 1;
        }
        else //Desligar a led (== 0)
        {
          analogWrite(ledTrabalho,0);
          TEstado = 0;
        }
     }
     else//Alterar a intensidade
     {
       String M = "";
       for(int i = 2; i<mensagem.length();i++) // usa esse for para contemplar os 3
tamanhos de intensidade, de 0a9, 11 a 99 e 100
       {
         M = M + mensagem.charAt(i); //M vai receber o valor da intensidade em uma
string
       }

       int inAux =  M.toInt();
       TIntensidade = inAux*2.55;   // Multiplicação por 2.55, queremos o valor
resultante da porcentagem. Onde o valor máximo é 255.(valor/100 * 255)
```

```
      if(TEstado){ //Se a luz estiver ligada, altera a sua intensidade, se nao, apenas
salva o valor
      analogWrite(ledTrabalho,TIntensidade);
      }// Altera a intensidade da saida da led}
    }
  }

  //----------------------------------LEDBANHEIRO---------------------
  else if(mensagem.charAt(0) == 'B') //
  {
    if(mensagem.charAt(1) == 'E') //alterar o estado
    {
       if(mensagem.charAt(2) == '1') // Ligar a led
       {
         if(BIntensidade == 0) // Se a intensidade for 0 a lampada vai ligar e alterar
intensidade para a maxima maxima
          {
           BIntensidade = 255;
          }
          //Liga a LED de trabalho, com a intensidade ja estabelecida
          analogWrite(ledBanheiro,BIntensidade);
          BEstado = 1;
       }
       else //Desligar a led (== 0)
       {
         BEstado = 0;
         analogWrite(ledBanheiro,0);
       }
    }
    else //Alterar a intensidade
    {
      String M = "";
      for(int i = 2; i<mensagem.length();i++) // usa esse for para contemplar os 3
tamanhos de intensidade, de 0a9, 11 a 99 e 100
      {
        M = M + mensagem.charAt(i); //M vai receber o valor da intensidade em uma
string
      }

      inAux =  M.toInt();
      BIntensidade = inAux*2.55;

      if(BEstado){ //Se a luz estiver ligada, altera a sua intensidade, se nao, apenas
salva o valor
      analogWrite(ledBanheiro,BIntensidade); // Altera a intensidade da saida da led
      }
    }
  }

  //----------------------------------LEDLAZER-----------------------
```

```
   else if(mensagem.charAt(0) == 'L') //
   {
     if(mensagem.charAt(1) == 'E') //alterar o estado
     {
        if(mensagem.charAt(2) == '1') // Ligar a led
        {
          if(LIntensidade == 0) // Se a intensidade for 0 a lampada vai ligar e alterar
intensidade para a maxima maxima
          {
            LIntensidade = 255;
          }
          //Liga a LED de trabalho, com a intensidade ja estabelecida
          analogWrite(ledLazer,LIntensidade);
          LEstado = 1;
        }
        else //Desligar a led (== 0)
        {
          analogWrite(ledLazer,0);
          LEstado = 0;
        }
     }
     else //Alterar a intensidade
     {
       String M = "";
       for(int i = 2; i<mensagem.length();i++) // usa esse for para contemplar os 3
tamanhos de intensidade, de 0a9, 11 a 99 e 100
       {
         M = M + mensagem.charAt(i); //M vai receber o valor da intensidade em uma
string
       }

       inAux =  M.toInt();
       LIntensidade = inAux*2.55;
       if(LEstado){ //Se a luz estiver ligada, altera a sua intensidade, se nao, apenas
salva o valor
       analogWrite(ledLazer,LIntensidade);// Altera a intensidade da saida da led
       }
     }
   }

   //----------------------------------LEDEXTRA-----------------------
   else if(mensagem.charAt(0) == 'E') //
   {
     if(mensagem.charAt(1) == 'E') //alterar o estado
     {
        if(mensagem.charAt(2) == '1') // Ligar a led
        {
          if(EIntensidade == 0) // Se a intensidade for 0 a lampada vai ligar e alterar
intensidade para a maxima maxima
          {
```

```
            EIntensidade = 255;
          }
          //Liga a LED de trabalho, com a intensidade ja estabelecida
          analogWrite(ledExtra,EIntensidade);
          EEstado = 1;
        }
        else //Desligar a led (== 0)
        {
          analogWrite(ledExtra,0);
          EEstado = 0;
        }
      }
      else //Alterar a intensidade
      {
        String M = "";
        for(int i = 2; i<mensagem.length();i++) // usa esse for para contemplar os 3
tamanhos de intensidade, de 0a9, 11 a 99 e 100
        {
          M = M + mensagem.charAt(i); //M vai receber o valor da intensidade em uma
string
        }

        inAux =  M.toInt();
        EIntensidade = inAux*2.55;

        if(TEstado){ //Se a luz estiver ligada, altera a sua intensidade, se nao, apenas
salva o valor
        analogWrite(ledExtra,EIntensidade); // Altera a intensidade da saida da led
        }
      }
    }
  }
 }
```

APÊNDICE D – Código do Arduino UNO para conectar-se ao blynk

```
/*************************************************************
  Autor : Rogério, modificado de examples.blynk

  Necessário:
   - Blynk IoT app (download da App Store ou Google Play)
   - Placa Arduino Uno
   - Conexão Wi-Fi para acessar o Blynk

 *************************************************************/

// Template ID, Nome dispositivo e Token de autenticação que são disponibilizados em Blynk.Cloud
// DEFINIÇÕES PARA CONEXÃO COM BLYNK.CLOUD
#define BLYNK_TEMPLATE_ID           "Id do Template"
#define BLYNK_DEVICE_NAME           "Iluminação automatizada"
#define BLYNK_AUTH_TOKEN            "3X3MPl0d3t0k3nd34ut3nt1c4c40"



//BLYNK_PRINT É UTILIZADO PARA MOSTAR, NO MONITOR SERIAL, MENSAGENS DE CONEXÃO
#define BLYNK_PRINT Serial


//VARIÁVEIS UTILIZADAS NA FUNÇÃO DE CONEXÃO

char auth[] = BLYNK_AUTH_TOKEN;

// Suas informações de Wi-Fi.
// Use password(pass) como "" quando o Wi-Fi é aberto.
char ssid[] = "NomeDaRede";
char pass[] = "SenhaDaRede";

//Bibliotecas usadas para conexão com o ESP-01 e blynk
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>

// Usado em Software Serial no Arduino Uno, Nano...
//Inclusão da bibliteca de comunicação com o ESP-01
#include <SoftwareSerial.h>

//Definição dos pinos seriais utilizados para comunicação Arduino UNO e ESP-01
SoftwareSerial EspSerial(0, 1); // RX, TX

// ESP8266(ESP-01) baud rate, ou seja, velocidade de comunicação previamente adquirida:
#define ESP8266_BAUD 38400

//Conexão com o ESP8266(ESP-01) através das portas 0 e 1
```

```
ESP8266 wifi(&EspSerial);

void setup()
{
  // Inicialiazanto as portas seriais do Arduino UNO com 115200 de velocidade.
  Serial.begin(115200);

  // Inicializanto a porta serial do ESP com o seu valor de baud rate
  EspSerial.begin(ESP8266_BAUD);
  delay(10);
  //Inicializando a conexão com o servidor Blynk
    Blynk.begin(auth, wifi, ssid, pass, "blynk.cloud", 80);

  //Pode-se também não especificar o servidor:
  //Blynk.begin(auth, wifi, ssid, pass);

  //pode ser usado para conectar-se a um IP específico:
  //Blynk.begin(auth, wifi, ssid, pass, IPAddress(192,168,1,100), 8080);
}

void loop()
{
  Blynk.run();
  // Pode-se fazer a inserção de códigos sem alteração do funcionamento.
  // É recomendado evitar a função delay()!
}
```