**UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”**

**FACULDADE DE ENGENHARIA**

**CAMPUS DE ILHA SOLTEIRA**

**IGOR DE OLIVEIRA ANTUNES HOLTZ**

# DESENVOLVIMENTO DE APLICATIVO BASEADO EM ARDUINO PARA DOMÓTICA

**Ilha Solteira**

**2022**

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

**IGOR DE OLIVEIRA ANTUNES HOLTZ**

# DESENVOLVIMENTO DE APLICATIVO BASEADO EM ARDUINO PARA AUTOMAÇÃO RESIDENCIAL

Trabalho de Graduação apresentado à Faculdade de Engenharia de Ilha Solteira – UNESP como parte dos requisitos para obtenção do título de Bacharel em Engenharia Elétrica.

Prof. Dr. Carlos Antônio Alves

**Orientador**

Ilha Solteira

2022

FICHA CATALOGRÁFICA

Desenvolvido pelo Serviço Técnico de Biblioteca e

Documentação

H758d

Holtz, Igor de Oliveira Antunes.
Desenvolvimento de aplicativo baseado em arduino para domótica / Igor de Oliveira Antunes Holtz. -- Ilha Solteira: [s.n.], 2022
65 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2022

Orientador: Carlos Antônio Alves
Inclui bibliografia

1. App inventor. 2. Arduino. 3. Automação residencial.

Raiane da Silva Santos
Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação
CRB/8 - 9999

# ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos dois dias do mês de março do ano de dois mil e vinte e dois, o discente ***Igor de Oliveira Antunes Holtz,*** matriculada sob o nº 141050187, tendo como banca examinadora o seu orientador, o *Prof. Dr. Carlos Antonio Alves*, o *Prof. Dr. Jean Marcos de Souza Ribeiro, e o Pós. Doutorando. Ricardo Taoni Xavier*, apresentou o Trabalho de Graduação intitulado **"DESENVOLVIMENTO DE APLICATIVO BASEADO EM ARDUINO PARA AUTOMAÇÃO RESIDENCIAL",** obtendo a nota 9,5 (nove e meio) e conceito APROVADO.

Prof. Dr. Carlos Antonio Alves
- Orientador -

Igor de Oliveira Antunes
- Discente -

Prof. Dr. Jean Marcos de Souza Ribeiro
- Membro da Banca -

Pós. Doutorando. Ricardo Taoni Xavier
- Membro da Banca -

Faculdade de Engenharia – Departamento de Engenharia Elétrica
Avenida Professor José Carlos Rossi, 1370 - CEP 15.385-000, Ilha Solteira - SP - BrasilTel (18) 37431150 - Fax (18) 3742-2735 - www.feis.unesp.br

## AGRADECIMENTOS

A todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho, enriquecendo o meu processo de aprendizado. Ao professor Carlos Antônio Alves por ter sido meu orientador e ter desempenhado tal função com dedicação, paciência e amizade. A banca avaliadora pela cordialidade e esmero. Aos amigos de jornada: Nicholas Garcia, Luis *Espeto*, Fernando *Estavare*, Pedro *Melges*, Alisson *Cedimar* e Ruan. Aos grupos IEEE e Engenheiros Sem Fronteiras. Em especial a minha namorada Mayara Carvalho pela prestimosidade, parceria e paciência. Ao eterno primo Felipe Luna e irmãos de vida Maurício *Voodo,* Murilo *Mubs* e Yuri *Kwheltz* por inúmeras conversas, aprendizados e batalhas. E por último e mais importante a maior guerreira de todas, Nancy Antunes, minha mãe.

## RESUMO

Este trabalho propõe um projeto de automação residencial de baixo custo e fácil implementação para controle sem fio via módulo bluetooth e placa Arduino, a partir de aplicativo de celular desenvolvido na plataforma gratuita App Inventor. Em específico, apresenta a metodologia utilizada para unir diversas tecnologias independentes afim de realizar o acionamento de lâmpadas e leitura de sensores de maneira e, apresentando estrutura modular conforme as necessidades e desejos do usuário.



**Palavras-chave:** APP Inventor. Arduino. Automação Residencial.

## ABSTRACT

This work provies a low-cost and easy-to-implement home automation Project, for a bluetooth remote control of an Arduino board using a software application developed in App Inventor plataform. In particular, it presents a methodology used to combine several independent technologies in order to control the activation of lamps and reading of sensors of remotely and presents a high capability of expansion according to the needs of the user.



**Keywords:** APP Inventor. Arduino. Home Automation.

## LISTA DE SIGLAS

| | |
|---|---|
| APP | Aplicativo |
| dBm | Decibel miliwatt |
| DHT | *Digital humidity and temperature* |
| GFSK | Gaussian frequency-shift keying |
| GHz | Giga-hertz |
| GPS | *Global Positioning System* |
| iOS | *Iphone Operational System* |
| KB | Kilobyte |
| MB | Megabyte |
| Mbps | Megabits por segundo |
| PCB | *Printed circuit board* |
| PNG | *Portable Graphics Format* |
| JPG | Joint Photographic Group |
| USB | *Universal Serial Bus* |
| VCC | Tensão de corrente contínua |

## LISTA DE FIGURAS

**SUMÁRIO**

# 1 INTRODUÇÃO

Segundo Muratori e Dal Bó (2011), a automação residencial ou domótica, define-se pela utilização da tecnologia para cuidar de atividades e interesses cotidianos tais como comunicação, conforto, praticidade, segurança e gestão de energia. Portanto, refere-se à automatização e controle da residência em todos os âmbitos possíveis. De forma geral, os processos de automação residencial necessitam desde redes de comunicações até a implementação de sistemas interligados e assistidos. A utilização de sistemas remotos, possibilitam que o usuário seja capaz de analisar e controlar situações em tempo real à distância. A grande vantagem da utilização de sistemas como estes é a possibilidade de expansão, acrescentando recursos e funcionalidades ao sistema, fazendo uso de controladores lógicos programáveis (CASTRUCCI e MORAES,2007).

Contudo, o investimento necessário para a automação de uma residência é variável de acordo com o projeto, e muitas vezes imagina-se que seja limitada ao público de alta classe (LIMA; NOBRE; ALENCAR, 2015). Segundo reportagem do jornal G1 (2020), na região de Sorocaba, um projeto completo de automação pode custar de R$ 5 mil a R$ 45 mil dependendo da complexidade da construção, da instalação e da quantidade de pontos físicos necessários para atender o desejo do cliente. Diante desde cenário, alternativas podem ser desenvolvidas com a utilização de diferentes tecnologias, ferramentas de baixo custo e relativa facilidade de implementação.

Arduino é uma plataforma *open-source* de prototipagem eletrônica que integra flexibilidade visando facilitar o uso do *hardware* e do *software*. É constituído por uma placa única com suporte de entrada/saída, a qual permite captar informações do ambiente através de sensores conectados as portas de entrada e também integra atuadores com o meio externo. A placa do Arduino possui um microcontrolador de 8 bits, Atmel AVR programado usando fundamentalmente linguagem C/C++ para enviar os comandos. Projetos do Arduino podem ser *stand-alone*, com o código já compilado em seu chip ou podem comunicar com *software* executado em um computador (ARDUINO, 2018). É utilizado para criar objetos ou ambientes interativos. Através dele pode-se adquirir informações do estado do ambiente que o cerca por meio da

recepção de sinais de sensores e interagir com os seus arredores, controlando luzes, motores e outros atuadores (AFONSO; PEREIRA; PEREIRA, 2015).

A computação permeia diversas áreas de estudo, e entender seus conceitos tem se tornado cada vez mais importante para atuar na sociedade (WILENSKY; BRADY; HORN, 2014). Isso tem motivado o ensino da computação a alunos da educação básica no mundo todo, que tipicamente é realizado ensinando-se algoritmos e programação utilizando linguagens de programação baseadas em blocos (GROVER; PEA, 2013). Nessas linguagens, criam-se programas arrastando e encaixando blocos visuais, o que diminui a carga cognitiva quando comparado ao uso de linguagens de programação baseadas em texto (LYE; KOH, 2014). Uma das alternativas para o ensino de computação é o desenvolvimento de aplicativos móveis utilizando o APP Inventor.

O APP Inventor é um ambiente de programação baseado em blocos usado para criar aplicativos móveis para dispositivos Android. Em 2019 obteve mais de 400.000 usuários ativos de várias partes do mundo (MIT APP INVENTOR, 2019). Os aplicativos do APP Inventor consistem em componentes visuais como botões e imagens, com componentes não visuais como câmera e sensor de GPS, além de um conjunto de blocos compondo o código que controla esses componentes. Assim, o processo de desenvolvimento de um aplicativo inclui tanto o design de interface de usuário quanto a programação da funcionalidade.

Este trabalho propõe uma alternativa de baixo custo para um projeto de automação residencial utilizando principalmente duas tecnologias: sistema Arduino e plataforma App Inventor. Ambas possuem ampla variedade de aplicações, muitos materiais educacionais de acesso gratuito na internet e intuitivas linguagens de programação, possibilitando serem utilizadas por usuários de diversos níveis de conhecimento. Este projeto foca em controlar o acionamento de lâmpadas e realizar a leitura de sensor de temperatura e umidade de maneira remota através da comunicação via *bluetooth* entre Arduino, em conjunto ao módulo bluetooth, e o aplicativo de celular desenvolvido. Apesar de completo, o projeto não se limita ao que foi realizado e pode ser expandido para utilizar um número maior de sensores e acionar um número maior de dispositivos, utilizando uma metodologia similar, conforme desejo do usuário.

# 2 AUTOMAÇÃO RESIDENCIAL

A automação residencial, também denominada domótica, é o conjunto de serviços proporcionados por sistemas tecnológicos integrados com o objetivo de proporcionar segurança, comunicação, gestão energética e conforto de uma residência (MURATORI; DAL BÓ, 2011; PRUDENTE, 2013). Alguns termos relacionados à automação residencial são casa inteligente, residência inteligente.

A automação se realiza mediante o uso de equipamentos com capacidade de se comunicar interativamente entre si e que seguem instruções de um programa previamente estabelecido pelo usuário da residência, com possibilidades de alterações conforme seus interesses (MURATORI; DAL BÓ, 2011).

O controle de processos residenciais pode reduzir a necessidade de intervenção humana no gerenciamento de equipamentos eletroeletrônicos, a partir da utilização de sistemas de controle. As informações sobre o ambiente são coletadas por sensores e analisadas automaticamente para a tomada de decisões, as quais podem disparar ações que podem alterar o estado de atuadores, modificando condições específicas do ambiente (BOLZANI, 2010). Desta forma, a automação residencial é capaz de auxiliar em tarefas diárias que demandam tempo e permite otimizá-las reduzindo a necessidade de ação do usuário ou em sistemas mais robustos, eliminando a necessidade dela.

Nas próximas seções serão apresentadas as principais tecnologias utilizadas neste projeto de automação residencial,

## 2.1 ARDUINO

Atualmente, existem diversos modelos desde o mais básico aos mais completos, cada um com sua particularidade, diferenciando no tipo de microcontrolador utilizado, quantidade de pinos para entradas e saídas digitais e analógicas, quantidade de memórias entre outras características, conforme mostra a Figura 1.

Figura 1 - Especificações dos diferentes modelos de Arduino

| Boards | Microcontroller | Operating Voltage/s (V) | Digital I/O Pins | PWM Enabled Pins | Analog I/O Pins | DC per I/O (mA) | Flash Memory (KB) | SRAM (KB) | EEPROM (KB) | Clock (MHz) | Length (mm) | Width (mm) | Cable | Native Network Support |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Uno | ATmega328 | 5 | 14 | 6 | 6 | 20 | 32 | 2 | 1 | 16 | 68.6 | 53.4 | USB A-B | None |
| Leonardo | ATmega32u4 | 5 | 20 | 7 | 12 | 40 | 32 | 2.5 | 1 | 16 | 68.6 | 53.3 | micro-USB | None |
| Micro | ATmega32u4 | 5 | 20 | 7 | 12 | 40 | 32 | 2.5 | 1 | 16 | 48 | 18 | micro-USB | None |
| Nano | ATmega328 | 5 | 22 | 6 | 8 | 40 | 32 | 2 | 0.51 | 16 | 45 | 18 | mini-B USB | None |
| Mini | ATmega328 | 5 | 14 |  | 6 | 20 | 32 | 2 | 1 | 16 | 30 | 18 | USB-Serial | None |
| Due | Atmel SAM3X8E ARM Cortex-M3 CPU | 3.3 | 54 | 12 | 12 | 800 | 512 | 96 | × | 84 | 102 | 53.3 | micro-USB | None |
| Mega | ATmega2560 | 5 | 54 | 15 | 16 | 20 | 256 | 8 | 4 | 16 | 102 | 53.3 | USB A-B | None |
| M0 | Atmel SAMD21 | 3.3 | 20 | 12 | 6 | 7 | 256 | 32 | × | 48 | 68.6 | 53.3 | micro-USB | None |
| Yun Mini | ATmega32u4 | 3.3 | 20 | 7 | 12 | 40 | 32 | 2.5 | 1 | 400 | 71.1 | 23 | micro-USB | Ethernet/Wifi |
| Uno Ethernet | ATmega328p | 5 | 20 | 4 | 6 | 20 | 32 | 2 | 1 | 16 | 68.6 | 53.4 | Ethernet | Ethernet |
| Tian | Atmel SAMD21 | 5 | 20 | 12 | 0 | 7 | 16000 | 64000 | × | 560 | 68.5 | 53 | micro-USB | Ethernet/Wifi |
| Mega ADK | ATmega2560 | 5 | 54 | 15 | 16 | 40 | 256 | 8 | 4 | 16 | 102 | 53.3 | USB A-B | None |
| M0 Pro | Atmel SAMD21 | 3.3 | 20 | 12 | 6 | 7 | 256 | 32 | × | 48 | 68.6 | 53.3 | micro-USB | None |
| Industrial 101 | ATmega32u4 | 5 | 7 | 2 | 4 | 40 | 16000 | 64000 | 1 | 400 | 51 | 42 | micro-USB | Ethernet/Wifi |
| Uno Wifi | ATmega328 | 5 | 20 | 6 | 6 | 20 | 32 | 2 | 1 | 16 | 68.6 | 53.4 | USB A-B | Wifi |
| Leonardo Ethernet | ATmega32u4 | 5 | 20 | 7 | 12 | 40 | 32 | 2.5 | 1 | 16 | 68.6 | 53.3 | USB A-B | Ethernet |
| MKR1000 | Atmel SAMD21 | 3.3 | 8 | 12 | 7 | 7 | 256 | 32 | × | 48 | 64.6 | 25 | micro-USB | Wifi |

Fonte: Core Eletronics, 2018

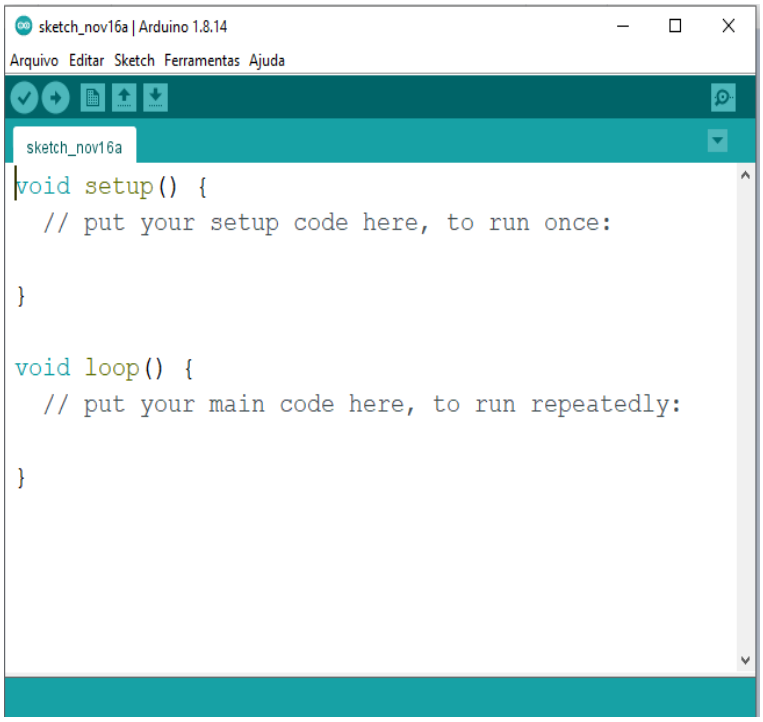
Para o projeto foi utilizado o modelo Uno, um dos modelos de ampla disponibilidade no mercado e baixo custo relativo de aquisição. Suas características serão exploradas com mais detalhes na subseção 2.1.2 do trabalho.

### 2.1.1 IDE e Linguagem de Programação

O ambiente de desenvolvimento integrado (*Integrated Development Environment* – IDE) é um programa especial executado no computador que permite a criação de *sketches* para a placa Arduino em uma linguagem simples derivada essencialmente da linguagem C/C++. Uma das grandes vantagens é a disponibilidade de uma grande quantidade de bibliotecas de programas, de acesso livre, que usadas como sub-rotinas facilitam a comunicação com os mais diferentes tipos de sensores. (SANTOS; AMORIM; DERECZYNSK, 2017).

No momento do *upload* do *sketch* para a placa, o código escrito é traduzido e transmitido para o compilador avr-gcc, importante *software* de código aberto que realiza a tradução final de seus comandos, agora para uma linguagem que pode ser compreendida pelo microcontrolador. Assim o Arduino simplifica ao máximo as complexidades inerentes à programação de microcontroladores (BANZI, 2012).

Figura 2 - Interface inicial do programa de IDE




Fonte: Elaborado pelo autor

Na Figura 2 é apresentada a interface inicial da IDE do Arduino, a qual fornece todas as funcionalidades necessárias para programação, incluindo vários exemplos de programas ou *sketches* que demonstram a conexão com o microcontrolador e comunicação com alguns dispositivos comuns, tais como LEDs, LCDs e sensores. Assim como o *hardware*, o *software* para o Arduino é de código aberto e pode ser baixado gratuitamente. As versões do IDE estão disponíveis para os sistemas operacionais *Windows*, Mac OS X e Linux (EVANS; NOBLE; HOCHENBAUM, 2013).

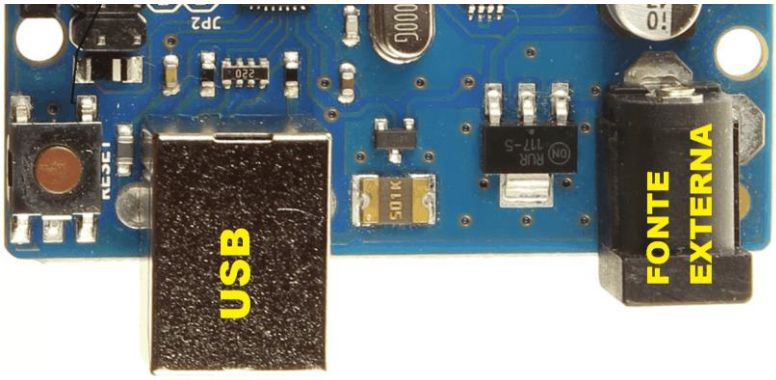
### 2.1.2 Arduino Uno R3

O Arduino Uno R3 possui como microcontrolador o chip ATmega328 e apresenta 14 pinos digitais operando a 5 V, dos quais 6 podem ser usados com saída PWM. Possui também 6 entradas analógicas, um cristal oscilador de 16 MHz, uma conexão USB, conector de energia e um botão de reset.

A placa pode ser alimentada tanto pela conexão USB como por uma fonte de alimentação externa através do conector Jack com positivo no centro, onde os valores máximo e mínimo de tensão são, respectivamente, 6 V a 20 V, porém se alimentada

com uma tensão abaixo de 5 V pode ficar instável e quando alimentada com tensão acima de 12 V, o regulador de tensão da placa pode sobreaquecer e danificar a placa. Os conectores USB e Jack podem ser observados na Figura 3.

Figura 3 - Conectores de alimentação




Fonte: Embarcados, 2013

Dessa forma, é recomendado para tensões de fonte externa valores de 7 V a 12 V. Já para conexão USB, o Arduino possui um circuito de proteção que faz com que a tensão não precisa ser estabilizada pelo regulador de tensão (SOUZA, 2013).

O modelo UNO R3 possui 8 conectores de alimentação que fornecem tensão tanto para *shields*, como para outros componentes conectados a eles, conforme mostra a Figura 4.

Figura 4 - Conectores de alimentação Arduino UNO R3


Fonte: Souza, 2013

- IOREF – Fornece uma tensão de referência para que *shields* possam selecionar o tipo de interface apropriada, dessa forma *shields* que funcionam com a placas Arduino que são alimentadas com 3,3 V podem se adaptar para ser utilizados em 5 V e vice-versa.
- RESET – pino conectado ao pino de RESET do microcontrolador. Pode ser utilizado para um *reset* externo da placa Arduino.
- 3,3V. – Fornece tensão de 3,3 V para alimentação de *shields* e módulos externos. Corrente máxima de 50 mA.
- 5V – Fornece tensão de 5 V para alimentação de *shields* e circuitos externos.
- GND – pinos de referência, terra.
- VIN – pino para alimentar a placa através de *shield* ou bateria externa. Quando a placa é alimentada pelo conector *Jack*, a tensão da fonte estará nesse pino.

Para a comunicação com os componentes externos a placa Arduino UNO possui 14 pinos de entrada e saídas digitais, 6 pinos de entradas analógicas, conforme mostra a Figura 5.

Figura 5 - Pinos de entrada e saída no Arduino UNO R3


Fonte: Souza, 2013

Cada um dos 14 pinos digitais podes fornecer ou receber uma corrente máxima de 40 mA. Os pinos 3, 5, 6, 9, 10 e 11 podem ser usados como saídas PWM de 8 bits. Os pinos 0 e 1 podem ser utilizados para comunicação serial. Deve-se observar que estes pinos são ligados ao microcontrolador responsável pela comunicação USB com o PC. Já os pinos 2 e 3 podem ser configurados para gerar uma interrupção externa. Por fim, os 6 pinos de entrada analógica identificados por A0, A1, A2, A3, A4 e A5 possuem uma resolução de 10 bits cada. Por padrão a referência do conversor analógico-digital está ligada internamente ao pino de 5 V, ou seja, quando a entrada estiver com 5 V o valor da conversão analógica digital será 1023. O valor da referência pode ser mudado através do pino AREF (SOUZA, 2013).

## 2.2 BLUETOOTH

A tecnologia *Bluetooth* facilita a comunicação de dados e voz entre vários dispositivos eletrônicos através de uma conexão sem fio, baseado em baixo custo, baixa potência, e curta distância, utilizando sinais de frequência de rádio. Para haver uma conexão entre dois dispositivos devem conter um rádio *Bluetooth*, sendo este um pequeno chip de computador que estabelece e gerencia as conexões individuais com baixo consumo de energia. O padrão *Bluetooth* determina que todos os dispositivos se conectem por meio de um conjunto específico de frequências de rádio e também determina os protocolos precisos para transmitir e receber dados, sendo que todos os dispositivos *Bluetooth* devem usar os mesmos protocolos para que todos falem a mesma linguagem eletrônica. Quando dois dispositivos *Bluetooth* fazem a conexão, é formada uma piconet, podendo ser de até oito dispositivos. Várias piconets podem ser conectadas formando a *scatternet* (MILLER, 2001).

Na comunicação entre dispositivos *Bluetooth*, dois nomes surgem com frequência: “mestre” e “escravo”. O dispositivo ou processo que tem controle sobre outros dispositivos ou processos é chamado de mestre, enquanto escravo o dispositivo ou processo controlado por outro dispositivo, denominado mestre.

### 2.2.1 Módulo HC-05

O módulo *Bluetooth* HC-05 é ideal para todo tipo de projetos em que seja necessária uma conexão sem fio confiável e simples de utilizar. É configurado por

comando AT e tem a possibilidade de funcionar tanto em modo mestre como escravo. Com isso é possível conectar dois módulos juntos, conectar um robô a um celular ou criar uma pequena rede de sensores intercomunicados com um mestre e vários escravos. As principais especificações técnicas são:

- Protocolo *Bluetooth*: v1.1 / 2.0.
- Frequência: banda ISM de 2,4 GHz
- Modulação: GFSK
- Potência de transmissão: menos de 4 dBm Classe 2
- Sensibilidade: Menos de -84 dBm no 0,1 % BER
- Razão assíncrona: 2.1 Mbps (Max) / 160 Kbps
- Síncrono: 1 Mbps / 1 Mbps
- Porta Serial *Bluetooth* (Mestre e Escravo)
- Alimentação 3,3 VCC 50 mA (suporta de 3,3 V a 6 V)
- Temperatura de operação: -5 °C a 45 °C

Um importante detalhe são os valores de tensão que os pinos TX e RX operam, no caso, 3,3 V, conforme consta no módulo, ilustrado pela Figura 6.

Figura 6 - Módulo *Bluetooth* HC-05

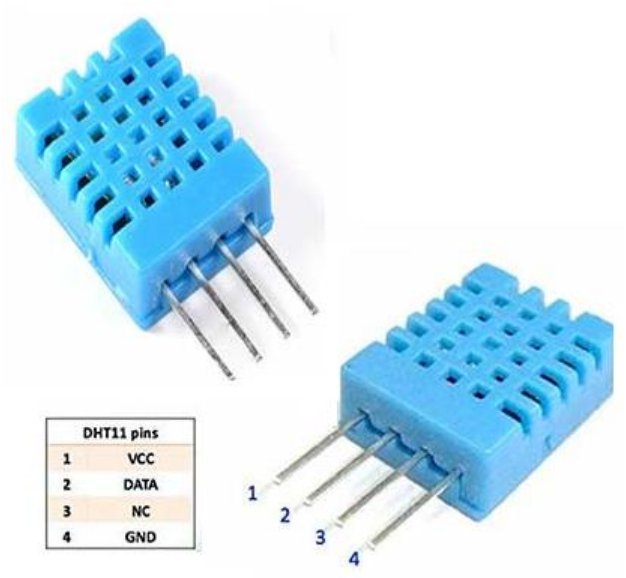

Fonte: Elaborado pelo autor

Se o pino RX for conectado diretamente a placas que forneçam 5 V, o módulo pode ser danificado, portanto a conexão deverá ser feita a utilizando resistores como divisores de tensão ou um conversor de nível lógico. O módulo possui também um botão para entrar em modo de comandos sendo possível seu acesso por *software* utilizando o pino EN. Há também um LED incorporado que indica o estado da conexão (MULTILÓGICA-SHOP, 2021).

## 2.3 SENSOR DE TEMPERATURA E UMIDADE DHT11

O uso de sensores elétricos é muito comum em pesquisas nos dias atuais, principalmente àquelas que envolvem dados meteorológicos. Os sensores elétricos de temperatura transformam a grandeza física de temperatura do ar em sinais elétricos em forma de tensão ou resistência elétrica. O sensor DHT11 faz a leitura em tempo real da temperatura e umidade relativa do ar, tendo como característica a técnica de aquisição de sinal digital, e a possibilidade de comunicação com microcontroladores de 8-bits de alta *performance* (D-ROBOTICS, 2010). Comercialmente pode ser encontrado na cor azul e possui 4 pinos para conexão, conforme ilustra a Figura 7.

Figura 7 - Sensor DHT11 e sua pinagem




Fonte: Modificado de Pinouts.net, 2019

Dentre suas características está a capacidade de realizar medições de temperatura na faixa de 0 ºC a 50 ºC, com acurácia de ± 2 ºC e umidade relativa entre 20 % e 95 %, com acurácia de ± 5 % e deve ser alimentado por uma tensão entre 3 V a 5,5 V.

## 2.4 MIT APP INVENTOR

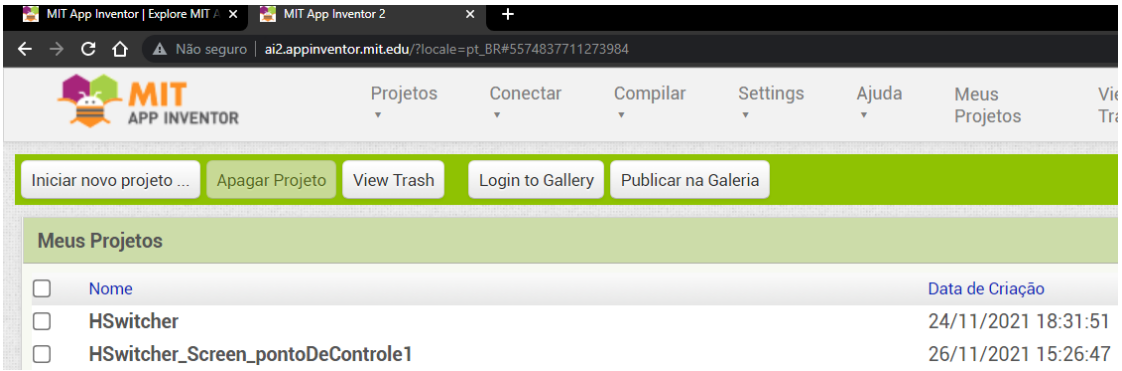
MIT APP Inventor é um ambiente de programação visual intuitivo que permite criar aplicativos totalmente funcionais para *smartphones* e *tablets* Android e iOS. Até

mesmo pessoas que nunca programaram um aplicativo de celular podem programar um aplicativo simples, instalar e vê-lo em funcionamento em menos de 30 minutos. A ferramenta baseada em blocos facilita a criação de aplicativos complexos e de alto impacto em muito menos tempo do que os ambientes de programação tradicionais (MIT APP INVENTOR, 2019).

A ferramenta pode ser acessada diretamente pelo navegador de internet ao selecionar a opção "*Create Apps!*", (MIT APP INVENTOR, 2019). Em seguida o usuário deverá realizar *login* a uma conta *Google*, na ficarão salvos todos os projetos de aplicativos realizados na plataforma.

Ao realizar o login, a página é redirecionada e exibe as opções de iniciar um novo projeto, apagar projeto, selecionar projetos salvos e entre outras, conforme ilustra a Figura 8.

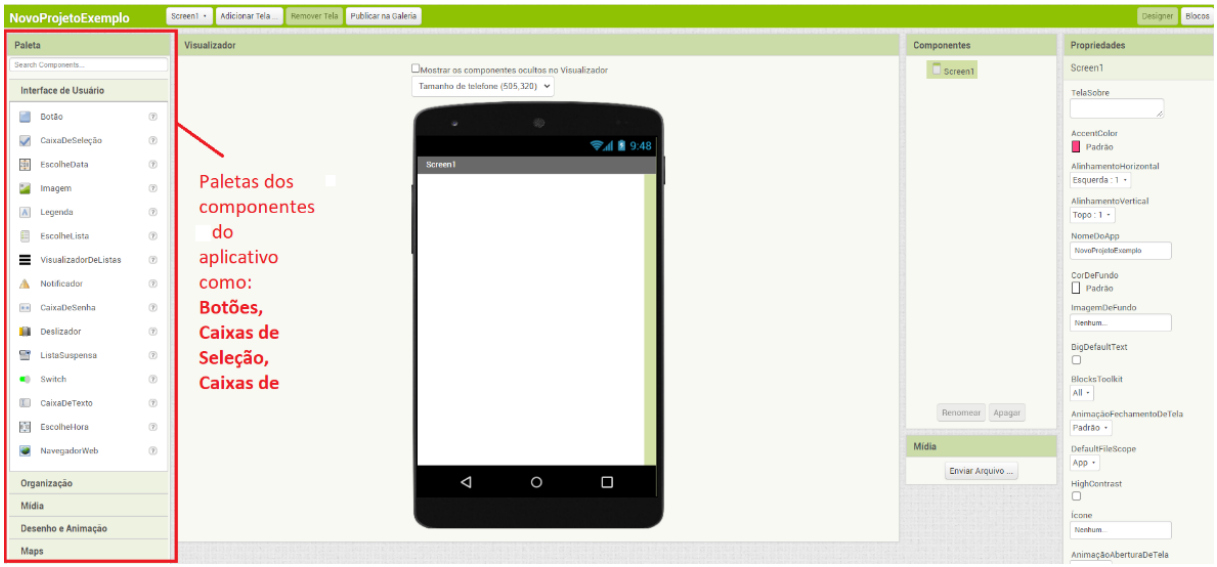
Figura 8 - Tela exibida ao realizar *login*




Fonte: Elaborado pelo autor

Após selecionar a opção de iniciar novo projeto, será necessário nomeá-lo e em seguida as opções de acesso ao ambiente estarão disponíveis. É importante destacar que a ferramenta divide o processo de programação em duas etapas: *Designer* e Blocos. Cada etapa pode ser acessada pelas opções presentes na porção superior direita da tela. Na parte esquerda da seção *designer,* são encontrados os componentes na seção "paletas", divididos em categorias, que serão posteriormente programados para executar funções, conforme ilustra a Figura 9.

Figura 9 - Localização da paleta de componentes




Fonte: Elaborado pelo autor

Devido ao grande número de componentes, esta seção explicará de forma geral somente os que foram necessários para o projeto. No capítulo 4 é possível encontrar informações sobre como cada componente foi configurado para o projeto.

Além das paletas de componentes, a ferramenta exibe, a seção “Visualizador” e “Componentes”. O primeiro representa a tela do celular do usuário e recebe os componentes das paletas, o segundo exibe em lista os componentes que foram adicionados ao visualizador. A última seção da aba *designer*, “Propriedades” permite que o usuário altere as características de cada componente, como o formato e cor por exemplo.

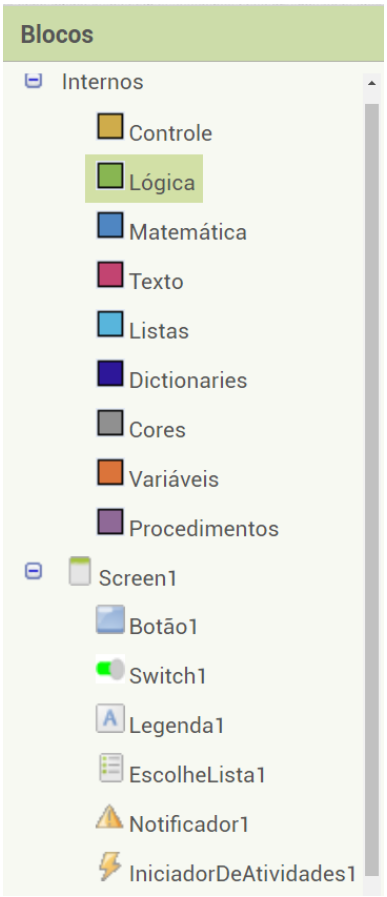
### 2.4.1 Principais Componentes

- Botão: Detecta toques do usuário, contudo volta ao seu estado inicial quando o toque é cessado. Sua aparência pode ser alterada para uma imagem em formato PNG ou JPG, assim como o texto vinculado a ele, tanto pela seção “Propriedades” como por programação de bloco.
- *Switch:* Também detecta toques do usuário, contudo seu estado se mantém, como um interruptor de lâmpada, até que receba outro toque do usuário. Seu texto vinculado e cores podem ser alterados também na seção “Propriedades”.

- Legenda: Utilizado para exibir texto na tela do aplicativo. Na seção “Propriedades” é possível alterar seu texto inicial e outras opções de formatação. Esse componente será responsável por exibir os valores de temperatura e umidade no projeto.
- Escolhe Lista: Exibe ao usuário uma lista e permita a escolha de algum elemento dela. Inicialmente esse componente não apresenta elementos salvos ou atribuídos a ele. Esse componente será responsável por armazenar os dispositivos *bluetooth* disponíveis para conexão.
- Notificador: Componente invisível ao visualizador e tem a função de exibir ao usuário uma mensagem personalizada de notificação temporária. Seu texto, tempo de exibição e ativação são determinados na aba de blocos.
- Iniciador de Atividades: Componente invisível ao visualizador e tem a função de chamar eventos do sistema operacional do celular. Será utilizado para chamar o evento de ativar o *bluetooth* do dispositivo.
- Cliente *Bluetooth*: Componente invisível ao visualizador que permite conectar o dispositivo a outros dispositivos via *Bluetooth.* Será utilizado para conexão com o módulo HC-05.
- Temporizador: Componente invisível ao visualizador que utiliza o relógio do celular para realizar eventos após determinado intervalo de tempo. Será utilizado para atualizar a legenda os valores de temperatura e umidade.

Para controlar o comportamento do aplicativo, é necessário realizar a programação em blocos dos eventos ativados com a interação do usuário com os componentes inseridos na aba *designer*. Ao selecionar a aba “blocos”, duas seções: Blocos e Visualizador estão disponíveis. Na primeira se encontram todos blocos de comandos e funções para controle do aplicativo. A segunda recebe todos os blocos de funções escolhido pelo usuário. Um importante detalhe é que a organização dos tipos de blocos é feita por categorias e suas respectivas cores. Outro detalhe é a divisão entre blocos internos, disponíveis independentemente de quais componentes estão em seu projeto, e blocos específicos de cada componente adicionado na aba *designer,* com seu próprio conjunto de eventos, métodos e propriedades. A Figura 10 ilustra a seção de blocos separados por categoria.

Figura 10 - Seção Blocos e suas categorias



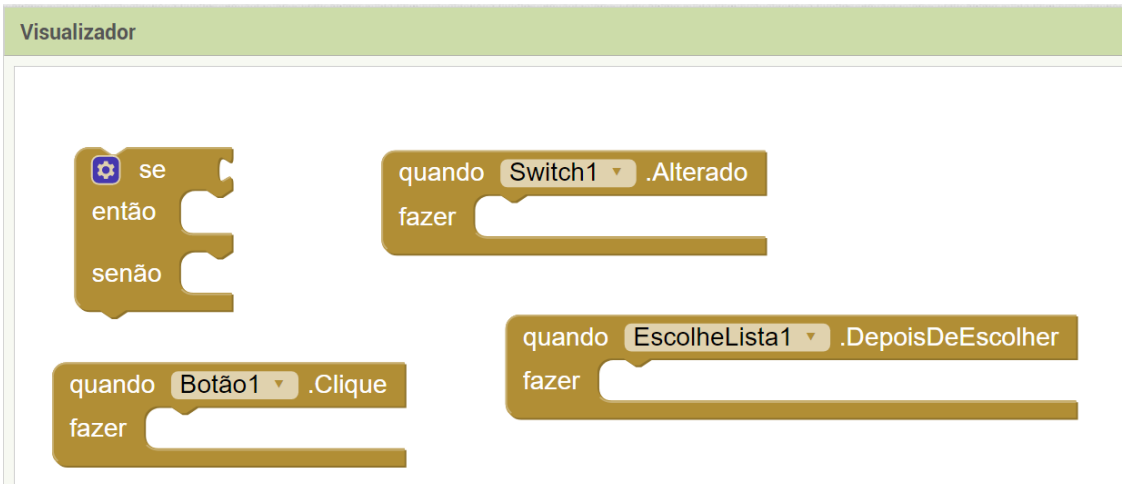

Fonte: Elaborado pelo autor

Devido ao grande número de blocos, esta seção explicará de forma geral somente os que foram necessários para o projeto. No capítulo 4 é possível encontrar mais informações.

### 2.4.2 Principais blocos

Blocos de Controle: Iniciam um evento após uma condição ser satisfeita, como por exemplo: Botão pressionado. É possível observar na Figura 11 os blocos de controle específicos dos componentes adicionados na aba *designer*.

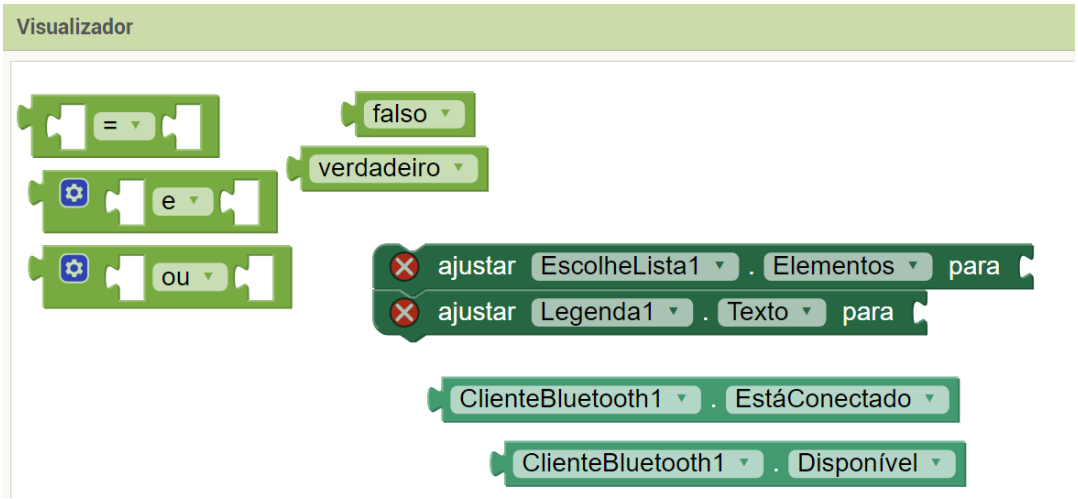
Figura 11 - Blocos de controle específicos




Fonte: Elaborado pelo autor

Blocos de Lógica: Além de realizar a comparação *booleana* entre diferentes condições, também podem defini-las. No projeto esses blocos serão responsáveis por verificar o estado da conexão *bluetooth*, por exemplo. Na Figura 12 é possível observar alguns exemplos dos blocos de lógica utilizados no projeto.

Figura 12 - Exemplo de blocos de lógica

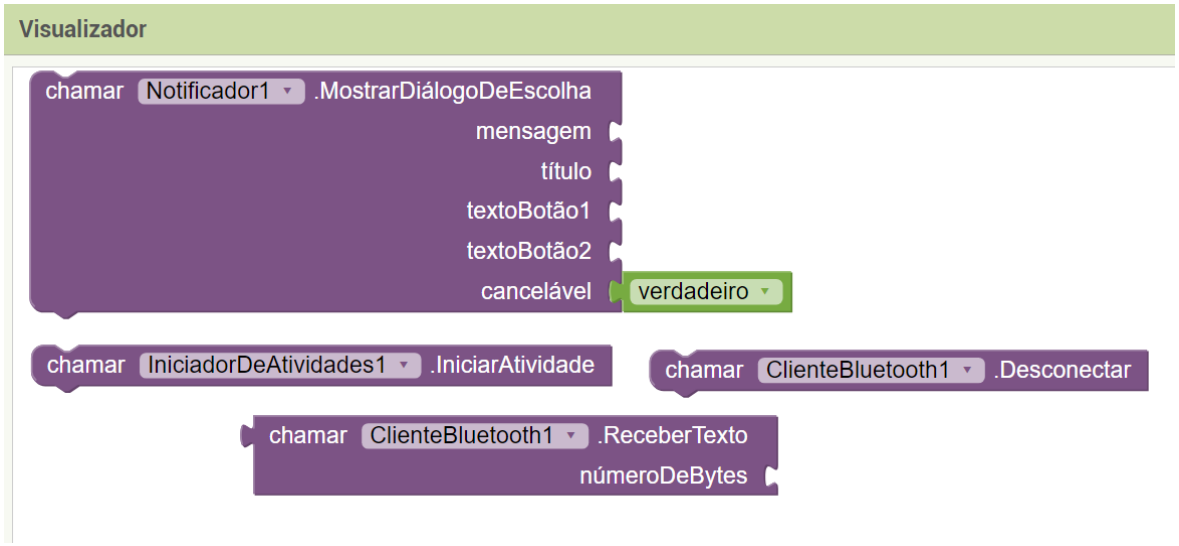



Fonte: Elaborado pelo autor

Blocos de Texto: Responsáveis por realizar a manipulação de variáveis de texto no aplicativo. No projeto vão compor as informações exibidas nos componentes legenda e notificação, por exemplo.

Blocos de Procedimentos: São funções que necessitam ser chamadas por blocos de controle ou lógica. Os componentes: Notificador, Cliente *Bluetooth* e Iniciador de atividades são ativados por essa categoria de blocos. A Figura 13 ilustra alguns exemplos de blocos de procedimentos do projeto.

Figura 13 - Exemplo de blocos de procedimentos

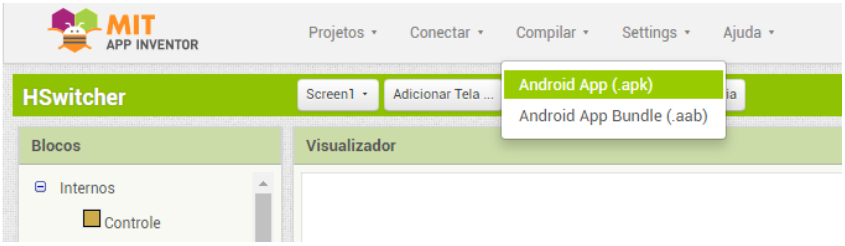



Fonte: Elaborado pelo autor

### 2.4.3 Compilação do programa

Com os componentes da interface e os blocos programados o usuário pode compilar o programa para gerar o arquivo de instalação do aplicativo. Para isso é necessário selecionar a opção “Compilar”, presente na parte superior da tela, e *“Android App* (.apk)”, conforme ilustra a Figura 14.

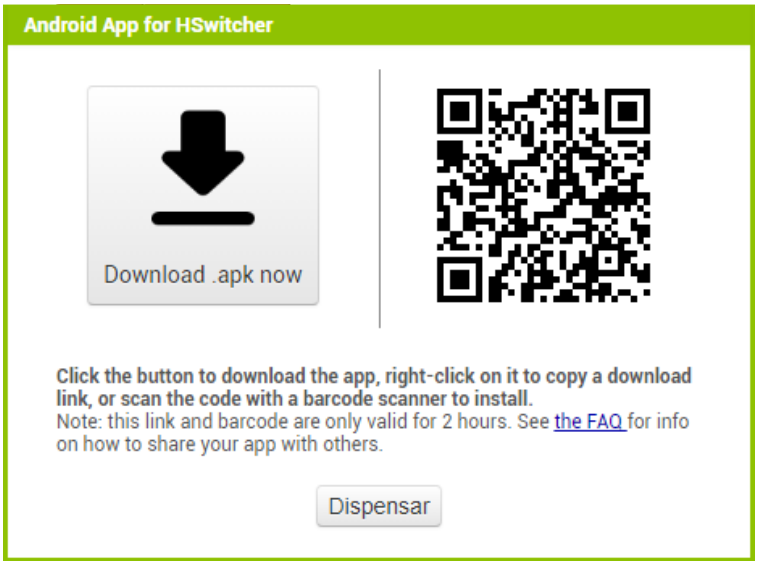
Figura 14 - Opção de compilar o programa




Fonte: Elaborado pelo autor

Após a compilação, a plataforma apresenta duas opções ao usuário: realizar download do aplicativo pelo navegador ou escanear o QR Code com o celular, para que o download possa ser feito diretamente para o dispositivo, conforme mostra a Figura 15.

Figura 15 - Opções exibidas após compilar o programa




Fonte: Elaborado pelo autor

Feita a transferencia do aplicativo, é necessário realizar sua instalação. É comum o usuário realizar diversas compilações do aplicativo, conforme o desenvolvimento do projeto para testar seu funcionamento e verificar a necessidade de correções, porém, antes da instalação de uma nova versão do aplicativo, é recomendado desinstalar a versão antiga.

## 2.5 PROTEUS DESIGN SUITE

Proteus Design Suite é um *software* usado para montagem de circuitos eletrônicos, simulação de circuitos baseados em microprocessadores e para projetos de placa de circuito impresso (PCB) (LABCENTER, 2021).

As vantagens do uso do simulador de circuitos é que ele pode ser acessado de forma mais fácil e mais rápida, sem a necessidade de um laboratório estruturado, ou da compra de materiais para a construção dos circuitos. Outro fator positivo é que, numa mesma simulação, é possível trabalhar com situações diferentes, podendo incluir muitos elementos como: resistores, capacitores ou lâmpadas; permitindo assim

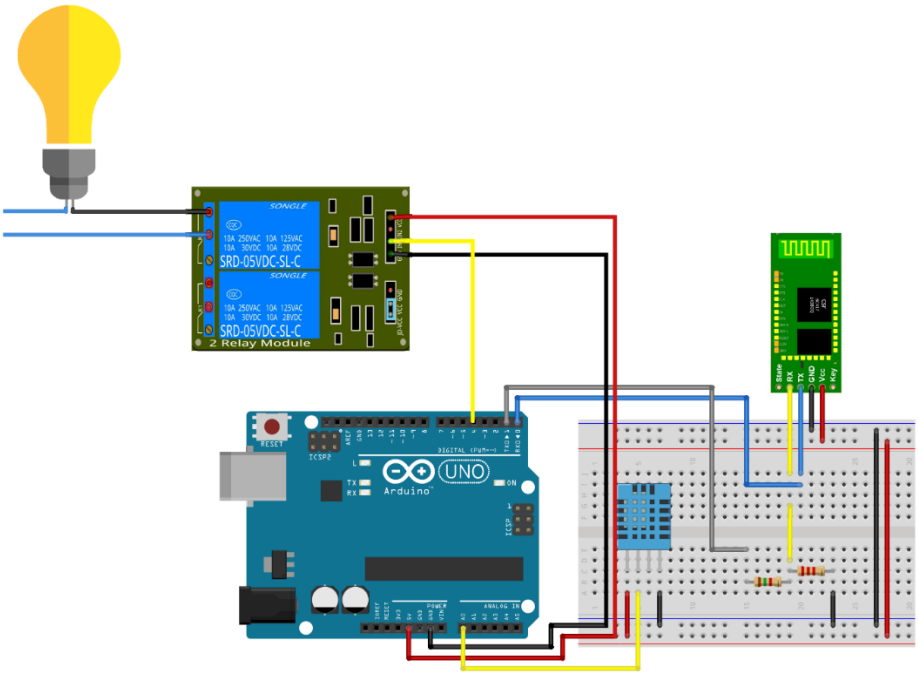
um nível maior de exploração e interação. O uso do simulador também diminui a possibilidade de acidentes ou queima de aparelhos. Contudo as simulações também possuem suas limitações e geralmente um sistema físico real é mais complexo e as simulações que o descrevem são sempre baseadas em modelos que contêm necessariamente simplificações.

O *software* Proteus foi escolhido para o projeto pois é um dos poucos simuladores de circuitos que possui em sua biblioteca de componentes o módulo HC-05 e o Arduino.

# 3 DESENVOLVIMENTO DO CIRCUITO

Neste capítulo será apresentado o esquema de conexão entre os componentes físicos necessários para o projeto e as considerações realizadas para poder integrar as diversas tecnologias apresentadas anteriormente. A Figura 16 ilustra o circuito geral utilizado.

Figura 16 - Circuito Geral




Fonte: Elaborado pelo autor

Os componentes operam em 5 V e podem ser alimentados pelo próprio Arduino, eliminando assim a necessidade de fonte de alimentação externa e individual para cada um deles. Outra questão considerada é a de que o Arduino não fornece tensão e potência suficiente para energizar lâmpadas residenciais, portanto, o Arduino irá controlar especificamente o interruptor que as aciona, sendo a tensão e a potência de alimentação delas fornecida pela própria rede elétrica da residência.

Para operar como interruptor foram utilizados relés. Esse componente permite que o fluxo de corrente da lâmpada seja interrompido, ou liberado, nos terminais de alta tensão conforme o sinal de entrada fornecido em seus terminais de baixa tensão.

A Figura 17 ilustra a pinagem dos conectores de baixa e alta tensão de um módulo com 2 relés.

Figura 17 - Pinagem do módulo relé duplo


Fonte: Elaborado pelo autor

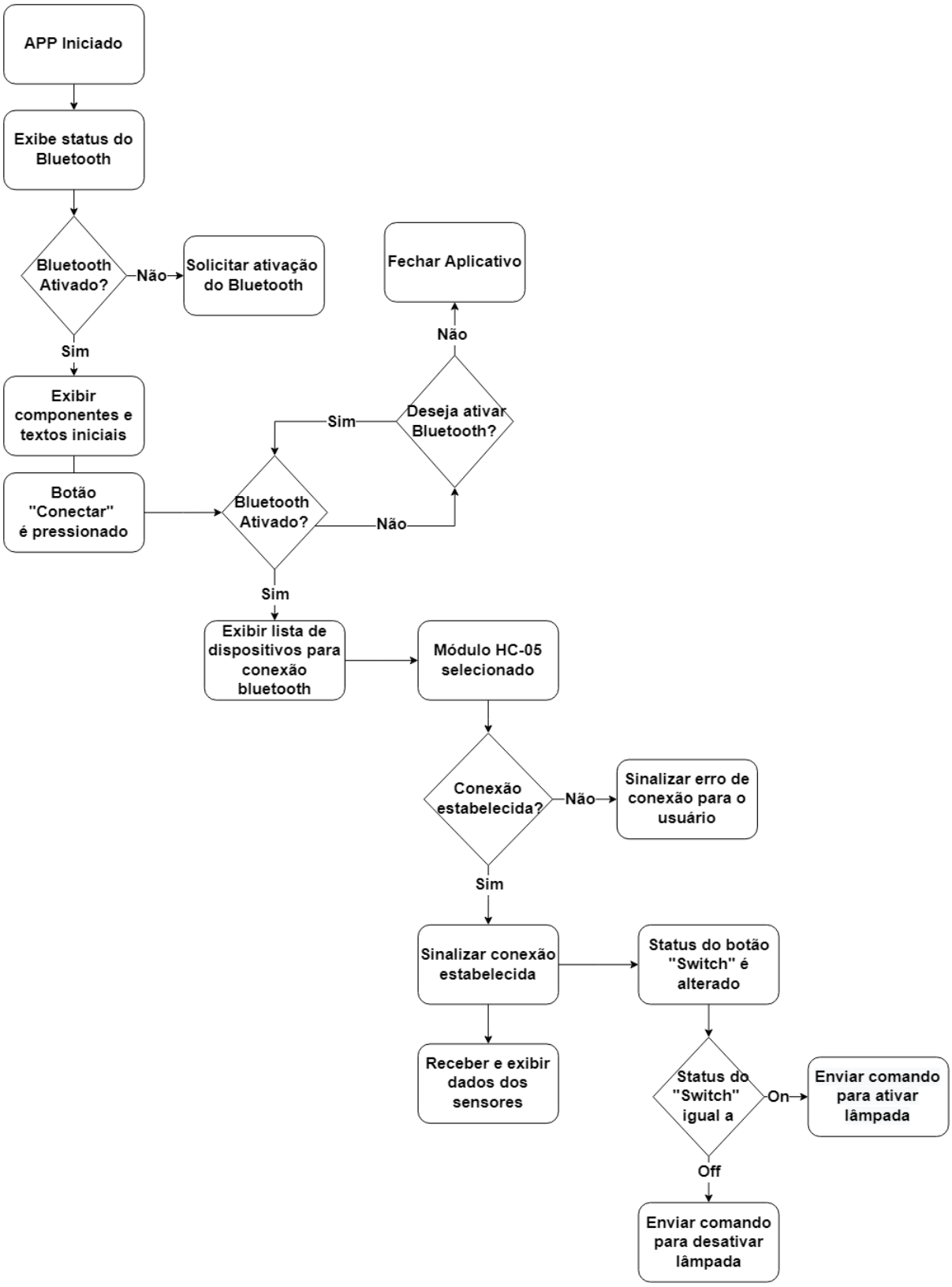
Para realizar a ativação do relé, foi escolhida a porta digital 4 do Arduino, conectada no pino “IN1” do relé. O pino analógico A0 foi conectado ao pino de dados do sensor de temperatura e pressão. O pino TX do módulo *Bluetooth*, responsável por enviar os dados para o Arduino, foi conectado no pino digital 0 do Arduino. Para conectar o pino 1 do Arduíno ao pino RX do módulo *Bluetooth,* foi utilizado um divisor de tensão para que a tensão no pino RX seja de 3,3 V. Para o divisor foi utilizado um resistor de 1 KΩ ligado em série com dois resistores de 1 KΩ ligados entre si em paralelo.

# 4 DESENVOLVIMENTO DO APLICATIVO

Com o circuito estabelecido o próximo passo do projeto é o desenvolvimento do aplicativo. Como o acesso a plataforma e visão geral dos componentes e blocos já foram apresentados na seção 2.4, este capítulo foca no algoritmo do aplicativo e componentes da plataforma utilizados especificamente no projeto.

Antes de iniciar a programação na plataforma, foi determinado o funcionamento do aplicativo de forma geral. Para isso o algoritmo foi organizado em um diagrama de blocos, facilitando assim a visualização das etapas a serem seguidas. A Figura 18 mostra o diagrama final.

Figura 18 - Diagrama de blocos do algoritmo




Fonte: Elaborado pelo autor

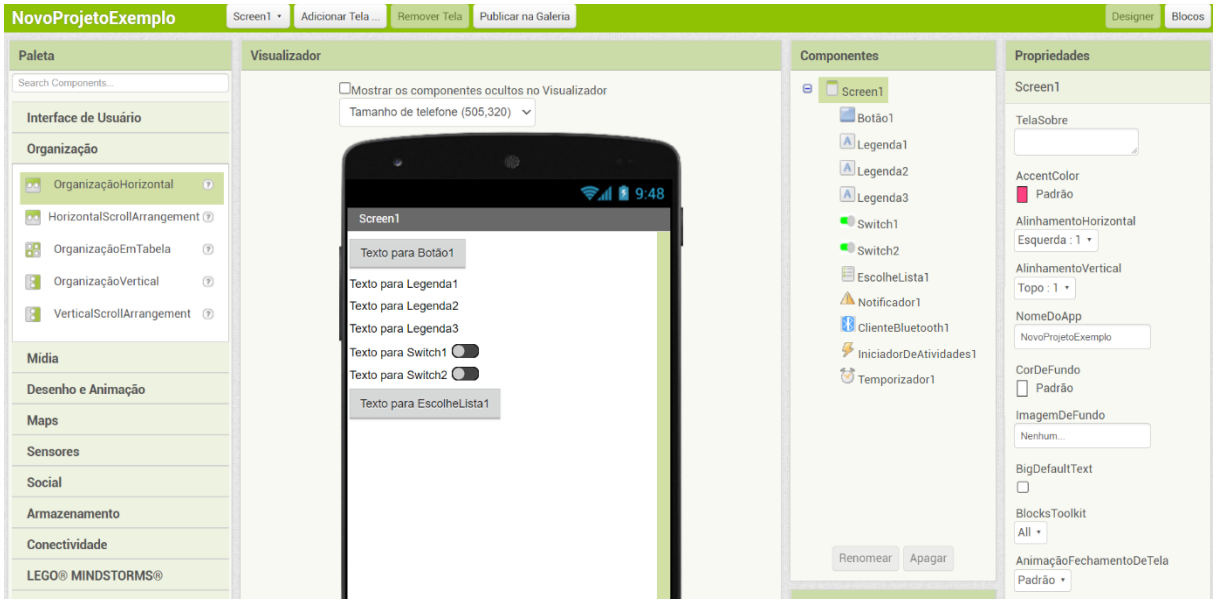
O primeiro evento ocorre assim que o aplicativo for iniciado, será verificado se o *bluetooth* do celular do usuário está ativo. Caso esteja, será exibido o botão de conectar e os textos iniciais. Caso não esteja, será solicitado para que o usuário o ative. O segundo evento ocorre no momento em que o botão com símbolo do *bluetooth* é pressionado. O APP irá verificar se o *bluetooth* do dispositivo está ativo, se não estiver, será solicitado ao usuário para ativar. Se já estiver ativo ou for ativado na etapa anterior, será exibida uma lista de dispositivos *bluetooth* para conexão. O terceiro evento ocorre ao final do segundo, no momento em que o módulo HC-05 é selecionado. Caso a conexão não seja estabelecida com sucesso, o usuário será notificado. Se ocorrer sucesso, será exibido uma mensagem de aviso e o aplicativo irá exibir os valores de temperatura e humidade na tela. O quarto evento envia o status do componente "*Switch*" para ativar ou desativar as lâmpadas.

Seguindo o algoritmo, foi iniciado a construção do aplicativo na plataforma e seu processo dividido em duas etapas.

### 4.1 Etapa 1: Design

Nessa etapa os componentes a serem programados devem ser acrescentados a área do visualizador, arrastando-os. Alguns componentes não possuem uma imagem de visualização na tela do "visualizador", como o "Cliente*Bluetooth*1" e o "Notificador1", porém podem ser observados e configurados na aba "componentes" localizado na lateral direita da tela. A Figura 19 ilustra os componentes adicionados ao visualizador.

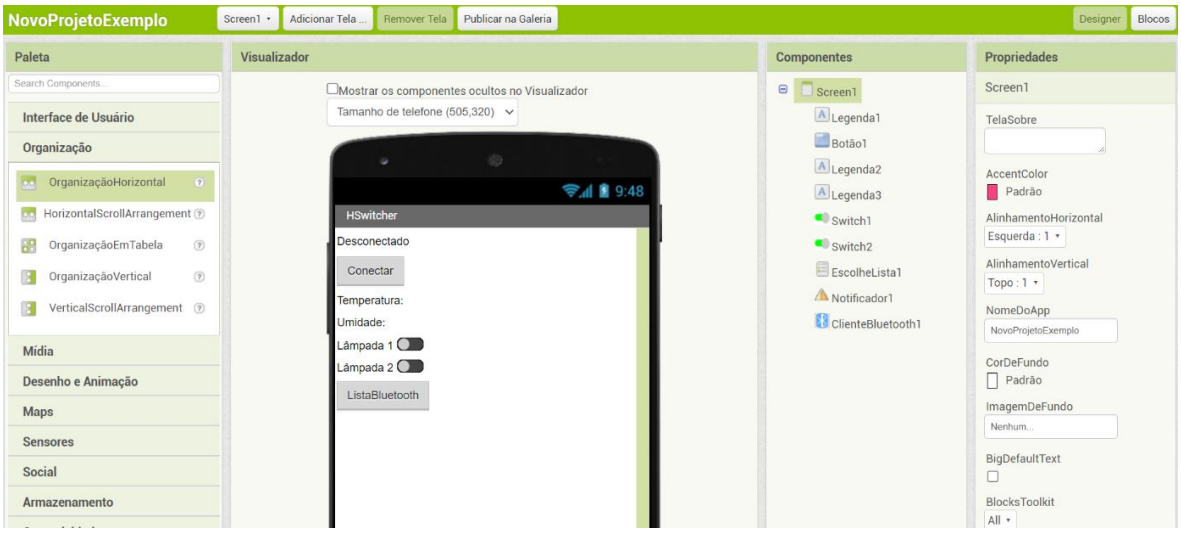
Figura 19 - Visualizador após adição dos componentes do projeto




Fonte: Elaborado pelo autor

Em seguida, foram alterados os textos dos componentes na aba “Propriedades”, localizado na lateral direita da tela conforme mostra a Figura 20.

Figura 20 - Texto dos componentes após serem renomeados

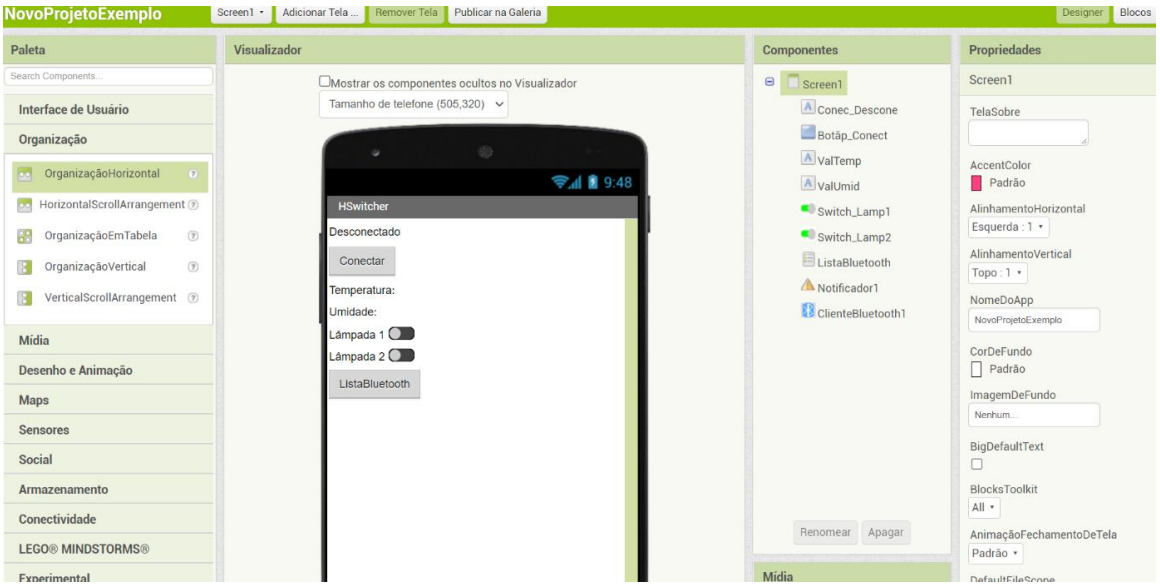



Fonte: Elaborado pelo autor

Na plataforma, cada componente é uma variável e seu nome de variável é diferente do nome exibido ao usuário, por exemplo: o componente “Botão1” é exibido para o usuário como “Desconectado”, porém a variável é chamada de “Botão1”. Os

componentes foram renomeados, facilitando sua identificação durante a programação de blocos, conforme ilustra a Figura 21.

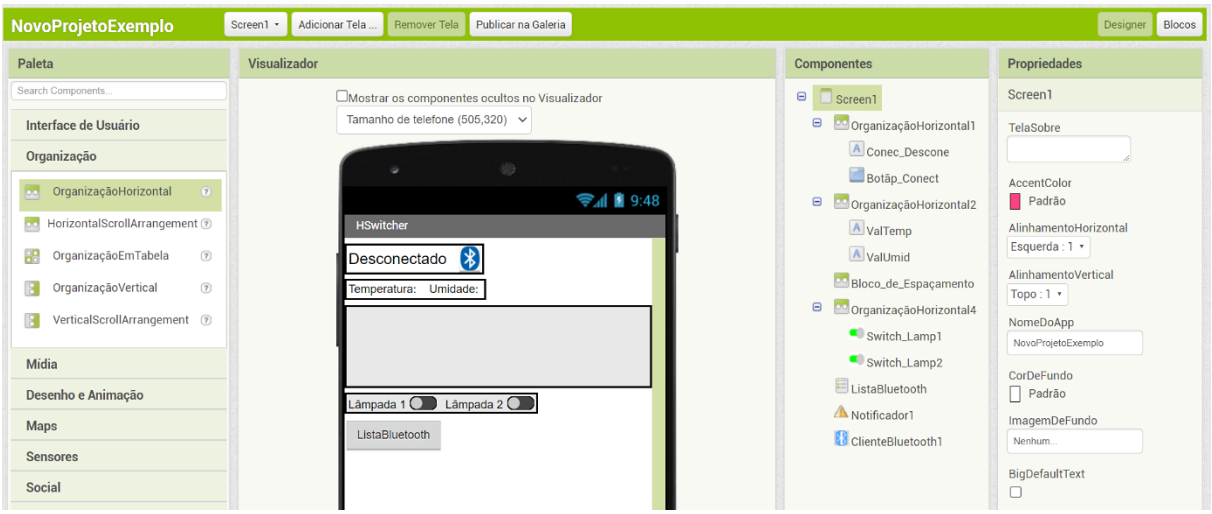
Figura 21 - Variáveis dos componentes após serem renomeadas




Fonte: Elaborado pelo autor

Para organizar os elementos na tela do aplicativo, é necessário utilizar o recurso de “organização”. Com esse recurso, é possível posicionar os componentes lado a lado, por exemplo, tornando a visualização mais agradável. A Figura 22 ilustra a localização do recurso na paleta de componentes.

Figura 22 - Paleta de Organização e componentes organizados

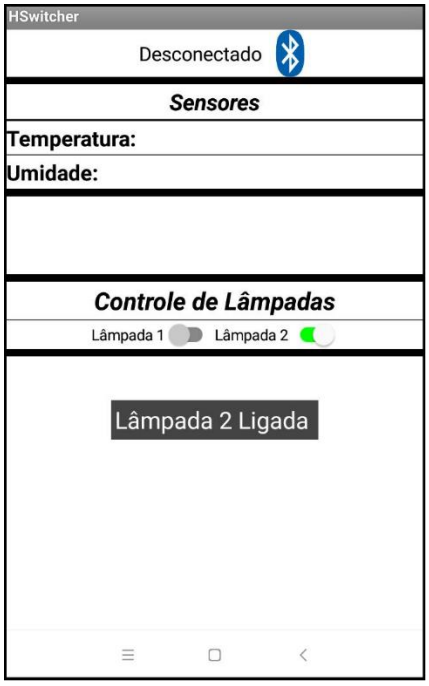



Fonte: Elaborado pelo autor

Para o botão responsável por chamar a função de conectar e desconectar o aplicativo, foi atribuído uma imagem em formato JPEG com o simbolo da tecnologia *bluetooth* e removendo seu texto anterior. Ajustes de redimensionamento foram necessários e realizados alterando suas propriedades. Para dar início a programação em blocos, o componente “Lista*Bluetooth*” foi ocultado da tela do aplicativo, pois é um componente que será utilizado em conjunto com a função atribuída ao botão “Conectar”. A dimensão e espaçamento dos componentes pode variar dependendo da resolução da tela do dispositivo instalado.

Por fim, alguns ajustes de tamanho de fonte dos textos e localização dos componente foram feitos, bem como a ocultação do componente “Lista*Bluetooth*”, para tornar o visual mais agradável. A Figura 14 ilustra a interface do aplicativo para o usuário.

Figura 23 - Tela inicial da versão final do aplicativo



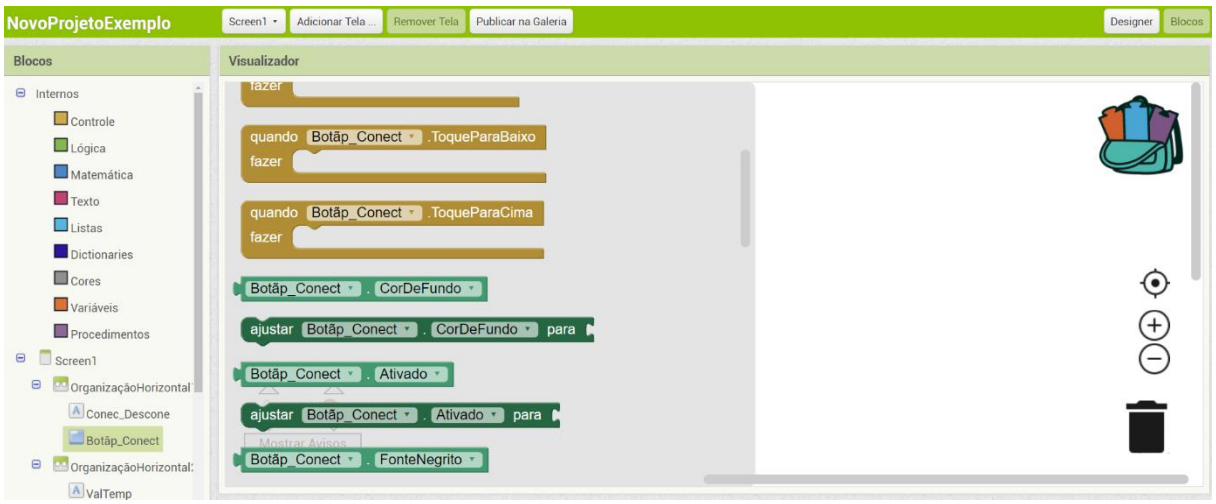

Fonte: Elaborado pelo autor

## 4.2 Etapa 2: Blocos

Agora é possível identificar os componentes adicionados na aba “Designer”, entre outras funções, na aba “Blocos”. Ao selecionar um componente, é exibido na

lateral esquerda todas as funções de programação relacionadas a ele, conforme mostra a Figura 24.

Figura 24 - Parte de funções disponíveis ao selecionar o componente “Botão”




Fonte: Elaborado pelo autor

Ao iniciar o aplicativo no celular, o programa irá salvar na variável “Lista *Bluetooth*” os dispositivos que foram pareados anteriormente com o celular e verificará se o *bluetooth* está ativo. Caso não esteja ativo, exibirá um aviso interativo para sua ativação. Esse conjunto de funções é demonstrado na Figura 25.

Figura 25 - Funções a serem executadas quando o aplicativo é iniciado


Fonte: Elaborado pelo autor

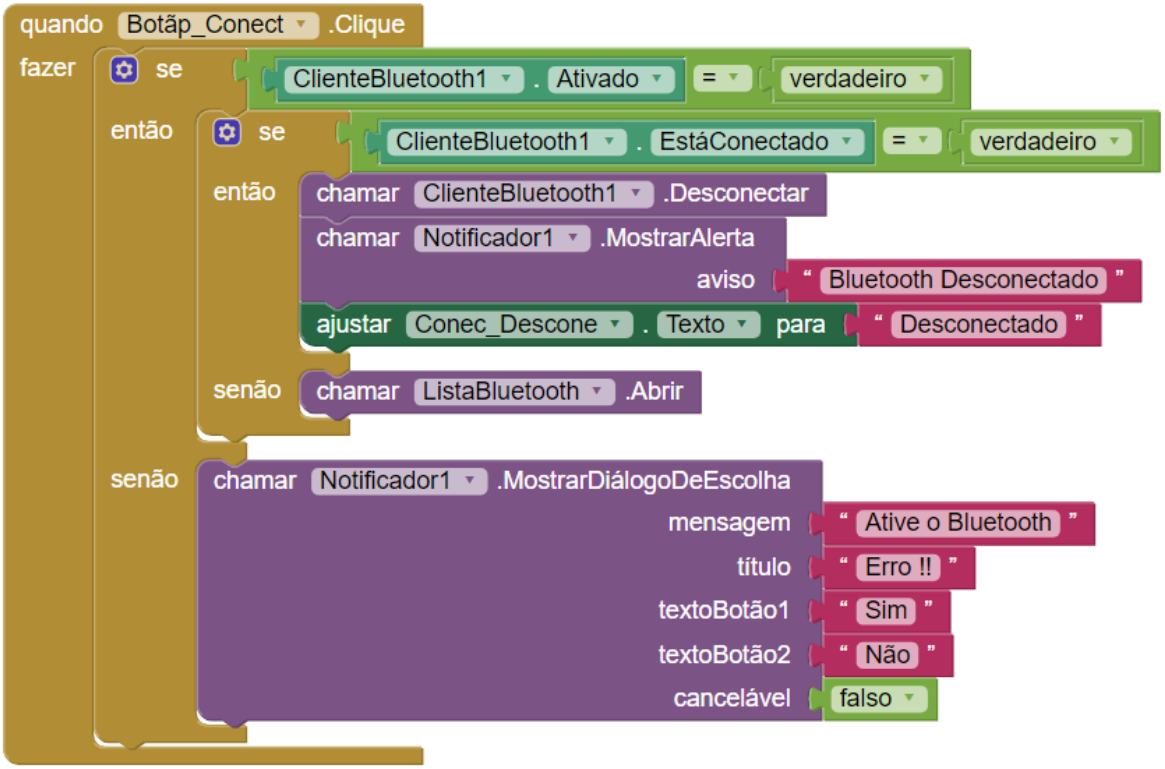
Para que o componente “IniciadorDeAtividades1 possa gerar uma notificação interativa, é necessário inserir no campo “ação” do componente, na aba designer, a seguinte linha de texto: *android.bluetooth.adapter.action.REQUEST_ENABLE* (DEVELOPERS, 2022), conforme ilustra a Figura 26.

Figura 26 - Ação do componente “IniciadorDeAtividades1” configurada


Fonte: Elaborado pelo autor

Para o aplicativo se conectar ao módulo HC-05, o componente “Botão” foi programado para que, quando pressionado, exiba a lista de dispositivos *bluetooth* disponíveis para se conectar. Caso o *bluetooth* esteja inativo no celular, um erro será exibido ao usuário. Caso o aplicativo esteja conectado, a função do botão será de desconectar o aplicativo e exibir uma notificação com o texto “*Bluetooth* Desconectado”, conforme pode ser observado na Figura 27.

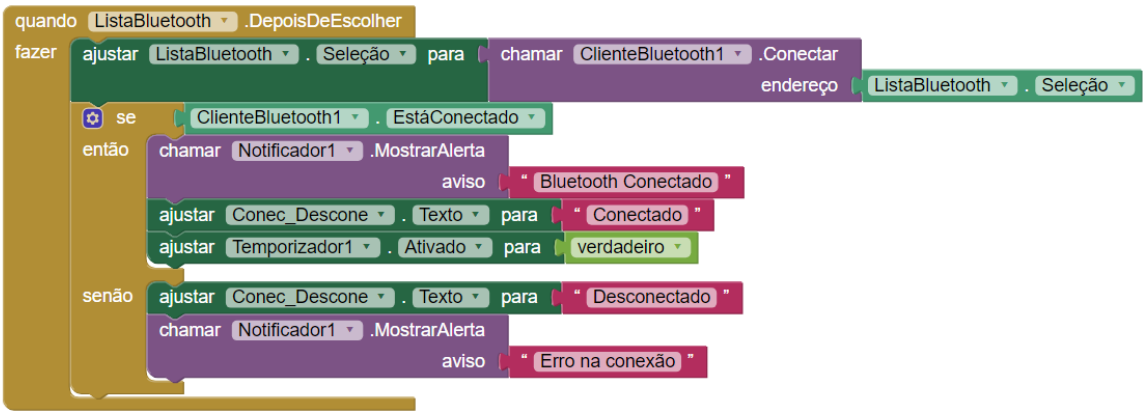
Figura 27 - Funções a serem executadas ao pressionar o componente “Botão”




Fonte: Elaborado pelo autor

Após o usuário escolher um dispositivo da lista, o programa chama a função que realiza a conexão. Caso o aplicativo falhe em se conectar será exibida uma notificação sinalizando que ocorreu a falha. Caso o aplicativo se conecte com sucesso, será exibida uma notificação sinalizando o sucesso, conforme ilustra a Figura 28.

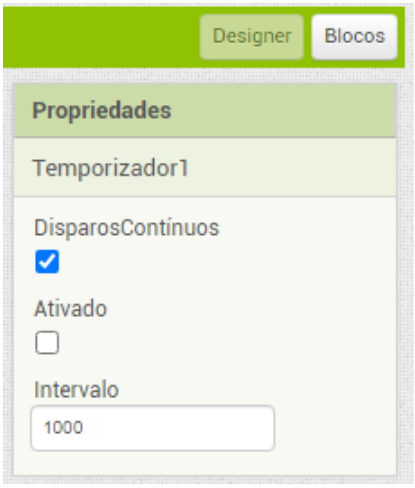
Figura 28 - Funções a serem executadas após a seleção de dispositivo




Fonte: Elaborado pelo autor

O componente “Temporizador” possui duas propriedades que foram alteradas: o intervalo, em que as ações contidas nessa função são executadas novamente, foi configurado para a 1000 ms; e o campo “ativado” foi deselecionado, para que o componente seja executado somente após a conexão ser bem sucedida, conforme ilustra a Figura 29.

Figura 29 - Propriedades do componente temporizador



Fonte: Elaborado pelo autor

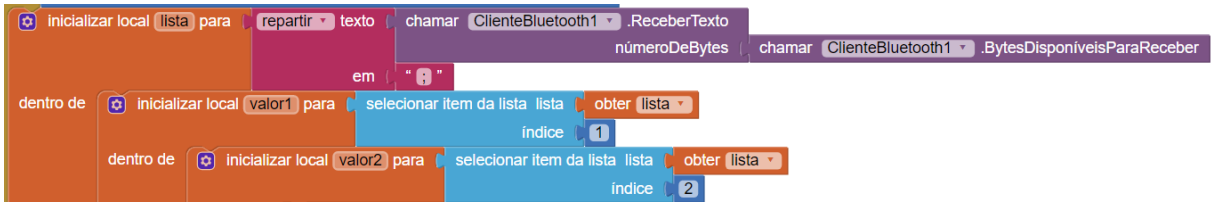
Quando a função do Temporizador é ativa, é verificado se o aplicativo está conectado ao módulo *bluetooth*. Caso não esteja, o Temporizador é desativado, como pode ser observado na Figura 30.

Figura 30 - Função de desativar o Temporizador caso *bluetooth* esteja desconectado


Fonte: Elaborado pelo autor

Caso o componente “Cliente *Bluetooth*” esteja conectado e recebendo dados do módulo HC-05, uma variável local chamada “lista” irá receber os dados de temperatura e umidade. Esse valores são enviados pelo Arduino de forma conjunta. Por exemplo, se a temperatura for 23 °C e a umidade 90 %, o valor enviado é 9023. Dessa forma, deve ser inserido um caracter separador, no caso foi escolhido o ponto e vírgula, “*;*” , para que a plataforma possa repartir o texto “90;23” e armazenar esses valores nas variáveis “valor1” e “valor2”, para então serem armazenadas na variável “lista”, nos indices 1 e 2 respectivamente, conforme mostra a Figura 31.

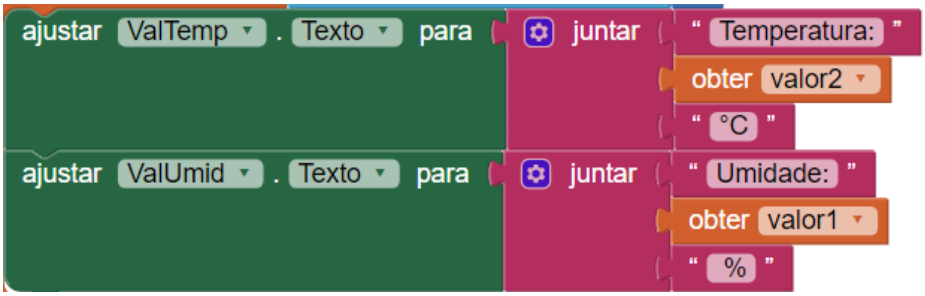
Figura 31 - Função de repartir o valor recebido e armazenar na “lista”




Fonte: Elaborado pelo autor

Com os valores armazenados, os componentes de legenda presentes na aba *Designer*, anteriormente nomeados de “Temperatura” e “Umidade”, são renomeados para o texto (“Temperatura” + “valor2” + “°C”) e (“Umidade” + “valor1” + “%”), respectivamente, conforme ilustra a Figura 32.

Figura 32 - Funções de exibir a temperatura e umidade na tela do aplicativo

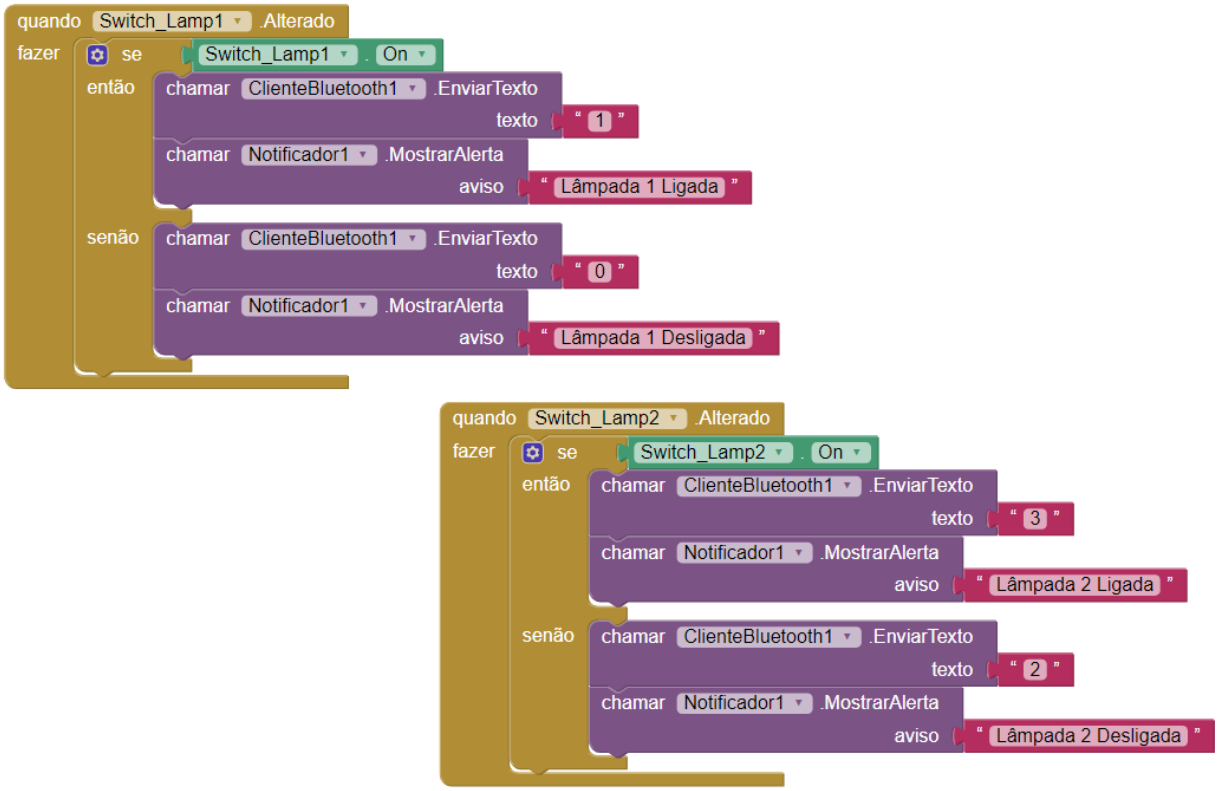



Fonte: Elaborado pelo autor

Como forma de acionamento das lampadas foi escolhido o componente "*Switcher*" ao invés do "Botão". Esse componente funciona como um interruptor de dois estágios: *On* – Ligado e *Off* – Desligado.

O conjunto de funções que determinam, quando o usuário interage com o interrruptor, que seja enviado via *bluetooth* o caractere, "0" e "2" para desativar as lâmpadas 1 e 2 respectivamente e, "1" e "3" para ativar as lâmpadas 1 e 2, respectivamente pode ser observado na Figura 33.

Figura 33 - Funções de enviar texto de ativação das lâmpadas 1 e 2




Fonte: Elaborado pelo autor

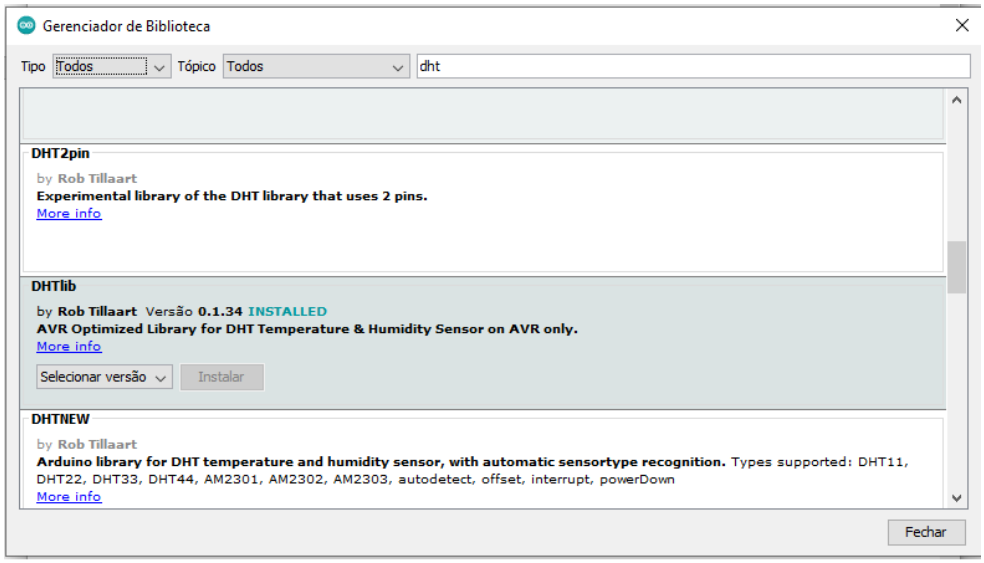
Esse caractere será lido e interpretado pelo Arduino realizar o processo de ativação e desativação dos relés de cada lâmpada. Esse é o ultimo passo da etapa de criação do aplicativo pela plataforma, em seguida o programa foi compilado e instalado no celular do usuário.

# 5 CODIFICAÇÃO DO ARDUINO

Neste capítulo será apresentado a construção do *sketch* a ser carregado no Arduino para realizar a integração entre aplicativo de celular, o módulo *bluetooth* e o sensor DHT11.

Apesar da IDE do Arduino possuir diversas funções e bibliotecas nativas instaladas, pode ser necessário realizar a instalação de novas bibliotecas. Elas possuem funções já programadas, podendo tornar o código mais eficiente. Para realizar a aquisição dos dados do sensor DHT11, foi utilizado a biblioteca “DHTlib” que pode ser adicionada a IDE do Arduino pressionando as teclas de atalho “Ctrl+Shift+i”, no *Windows*, e pesquisando por “dht” no campo de busca conforme ilustra a Figura 34.

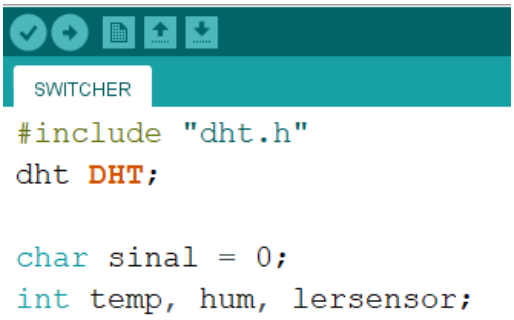
Figura 34 - Gerenciador de Biblioteca do Arduino




Fonte: Elaborado pelo autor

Para a programação, a biblioteca foi incluída no código *sketch* que será carregado ao Arduino utilizando a função “#include”. Em seguida, a linha “dht DHT” foi inserida para permitir utilizar as funções da biblioteca DHT possam ser chamadas com o prefixo “DHT”. A Figura 35 exibe o código *sketch* contendo a inclusão da biblioteca.

Figura 35 - Inclusão da biblioteca DHTlib e declaração de variáveis




Fonte: Elaborado pelo autor

Para o projeto, foi utilizado uma variável do tipo “char” para receber o comando dos componentes “swtcher” do aplicativo, e variáveis do tipo “int” para receber os valores de temperatura, umidade do sensor.

Na função *void setup*() do *sketch* foram declarados como saída os pinos 4 e 5 para que possam ativar os relés, conforme mostra a Figura 36. Nessa função, a leitura serial foi iniciada para que o dado enviado pelo aplicativo, através do modulo *bluetooth* possa ser lido.

Figura 36 - Comandos da função void setup()

```
void setup() {

  Serial.begin(9600);

  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
}
```

Fonte: Elaborado pelo autor

Na primeira parte da função *void loop*(), o valor da porta analógica referente ao sensor DHT11, é lido com a função da biblioteca anteriormente incluída: “DHT.read11(A0)”. Os valores de umidade e temperatura são armazenados nas variáveis “hum” e “temp”, respectivamente. A variável “sinal” recebe o valor enviado pelo módulo HC-05. A Figura 37 ilustra os comandos para aquisição de dados do sensor pelo Arduino.

Figura 37 - Aquisição de dados do sensor e do modulo *bluetooth*

```
void loop() {

  DHT.read11(A0);

  hum = DHT.humidity;
  temp = DHT.temperature;

  sinal =  Serial.read();
```

Fonte: Elaborado pelo autor

Para enviar os valores de temperatura e umidade ao mesmo tempo, foi necessário utilizar um caractere de separação, no caso o “;”. A Figura 38 mostra as linhas de comandos responsáveis pelo envio da umidade, caractere separador e temperatura, respectivamente.

Figura 38 - Envio de dados de temperatura e umidade

```
Serial.print(hum); //envia a umidade para o app
Serial.print(";"); //separador de valores
Serial.println(temp); //envia a temperatura para o app
```

Fonte: Elaborado pelo autor

Os níveis lógicos das portas 4 e 5 são alterados, conforme o valor recebido pela variável “sinal”. A Figura 39 ilustra as condições para alteração dos níveis lógicos conforme o valor armazenado na variável “sinal”.

Figura 39 - Condição para variação dos níveis lógicos das portas 4 e 5

```
  if (sinal == '1') {
    digitalWrite(4, LOW);

  } if (sinal == '0') {
    digitalWrite(4, HIGH);

  } if (sinal == '3') {
    digitalWrite(5, HIGH);

  } if (sinal == '2') {
    digitalWrite(5, LOW);

  }
  delay (1000);
}
```

Fonte: Elaborado pelo autor

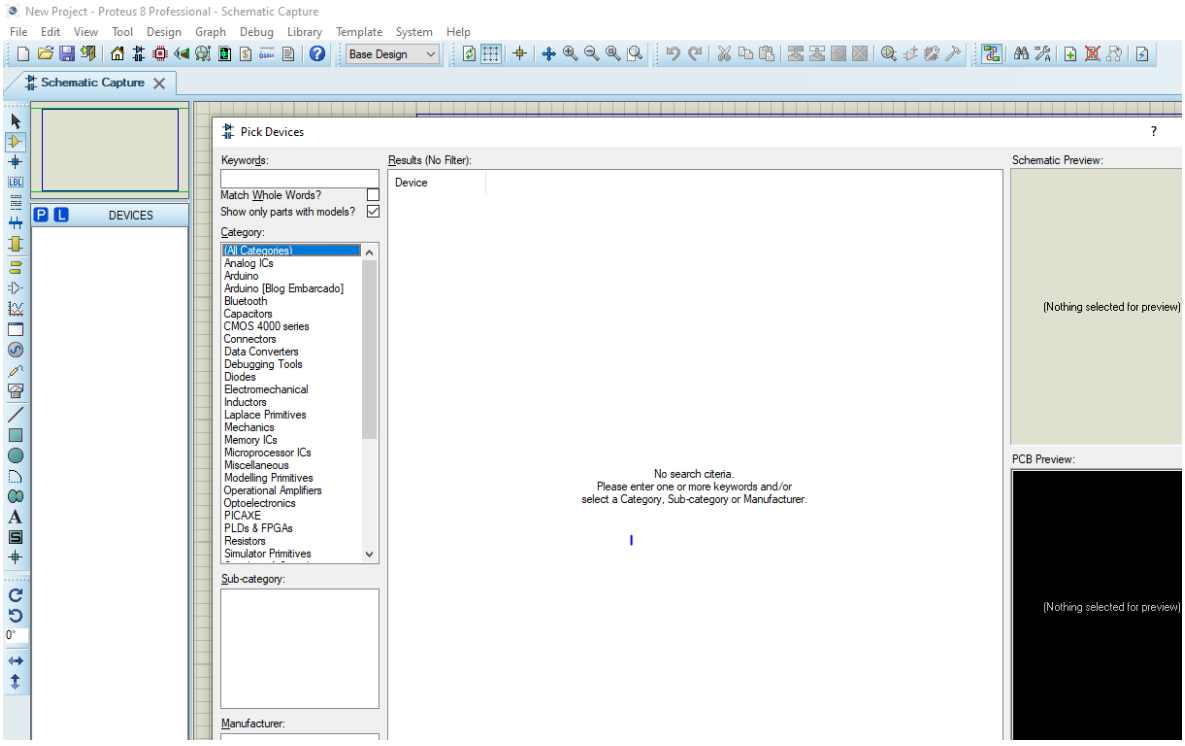
Ao final do código foi configurado um intervalo de 1 segundo, utilizando o comando “*delay* (1000)”, para a reexecução da função *void loop*().

# 6 SIMULAÇÃO DO CIRCUITO FINAL

O *software* Proteus possui diversas funções e ferramentas para simular até mesmo projetos robusto. Porém este capítulo irá descrever de forma geral as principais opções do programa e como elas foram utilizadas para o projeto.

Ao dar início a um projeto novo, é exibido diversas opções na parte superior e na lateral esquerda, e uma área central a qual os componentes do circuito são inseridos. Para escolher os componentes o usuário deve realizar um duplo clique na área branca, na lateral esquerda ou pressionar a tecla P. Uma janela irá surgir e dezenas de componentes eletrônicos serão exibidos divididos por categorias. Caso o usuário deseje, possível procurar pelo nome do componente através do campo de busca, conforme mostra a Figura 40.

Figura 40 - Menu de seleção de componentes

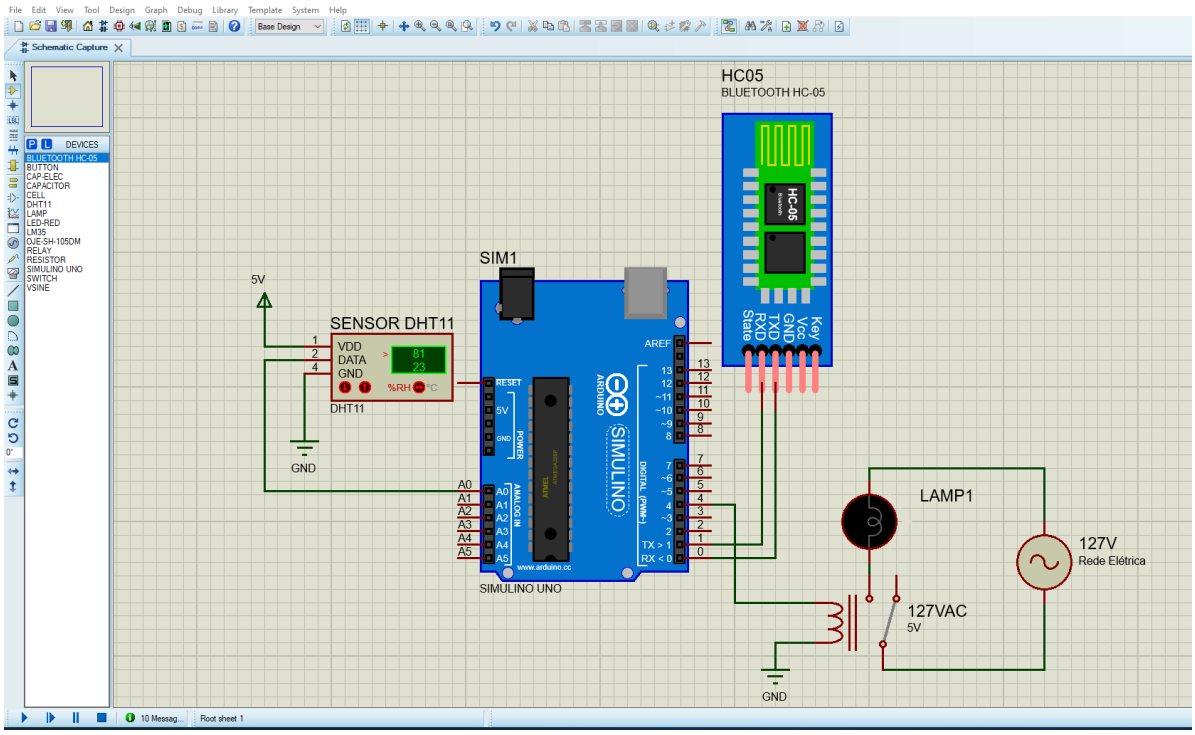



Fonte: Elaborado pelo autor

Para seguir com o projeto, foram utilizados os mesmos componentes do circuito exibido no capítulo 2, com exceção do divisor de tensão, que não se fez necessário. O módulo HC-05 é conectado nas portas digitais 0 e 1, a porta digital 4, configurada como *Output* na IDE é conectada ao terminal de ativação do relé. Foi utilizado uma

fonte de tensão externa de 127 V para energizar a lâmpada. A Figura 41 ilustra o circuito montado para a simulação.

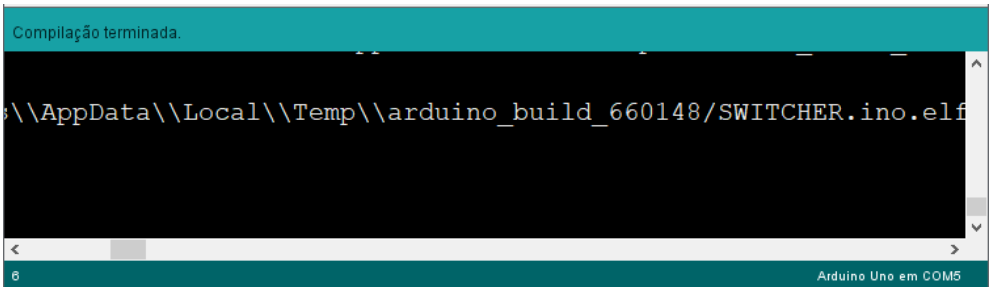
Figura 41 - Diagrama esquemático do circuito simulado




Fonte: Elaborado pelo autor

Para carregar o *sketch* programado no Arduino simulado é necessário verificar o local onde o arquivo é salvo. A IDE do Arduino, após compilar o código exibe na parte inferior de sua janela o caminho do arquivo, conforme mostra a Figura 42.

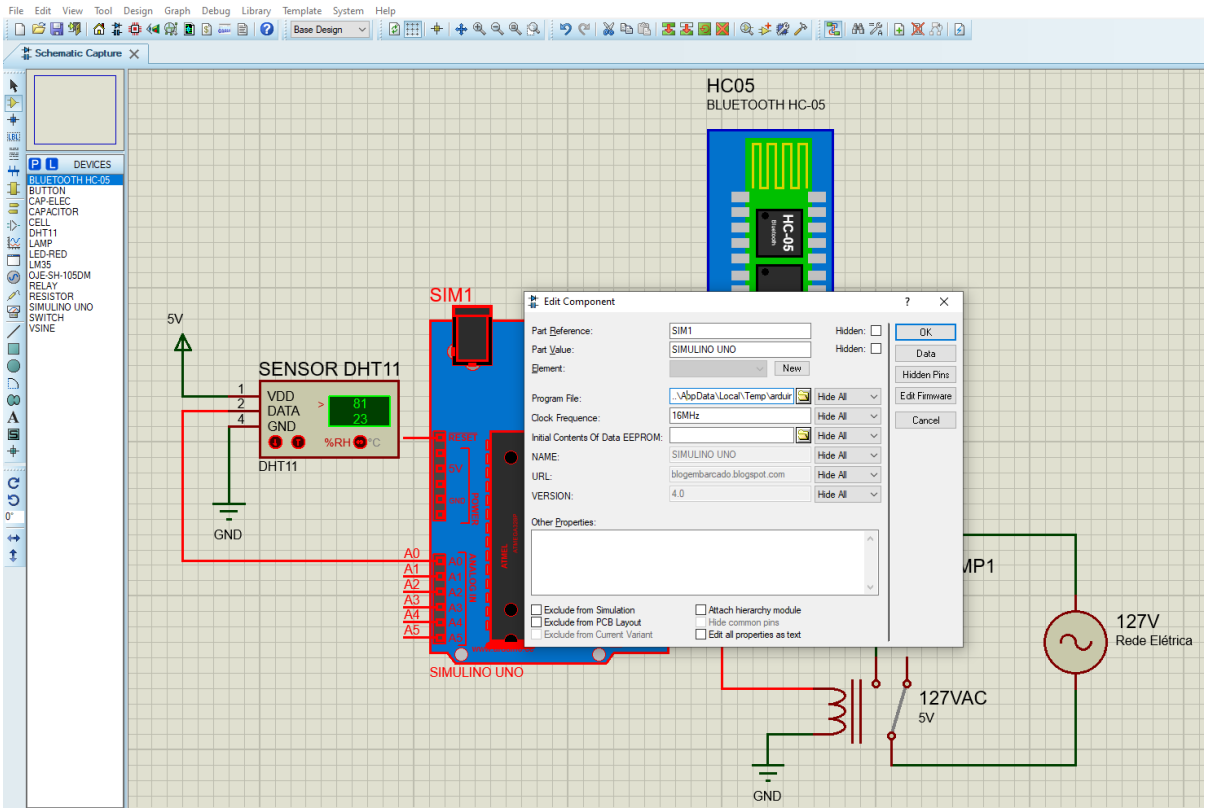
Figura 42 - IDE exibindo o caminho




Fonte: Elaborado pelo autor

O endereço deve ser inserido no campo “*Program File*”, ao selecionar a opção de editar componente. A Figura 43 mostra a janela exibida ao selecionar a opção de editar componente com o campo “*Program File*” realçado.

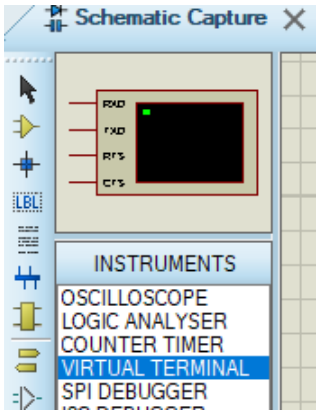
Figura 43 - Opções de edição do componente Arduino




Fonte: Elaborado pelo autor

Para verificar o sinal enviado e recebido pelo modulo HC-05, foi utilizado o instrumento “*Virtual Terminal*”. A Figura 44 ilustra o símbolo e sua localização no programa.

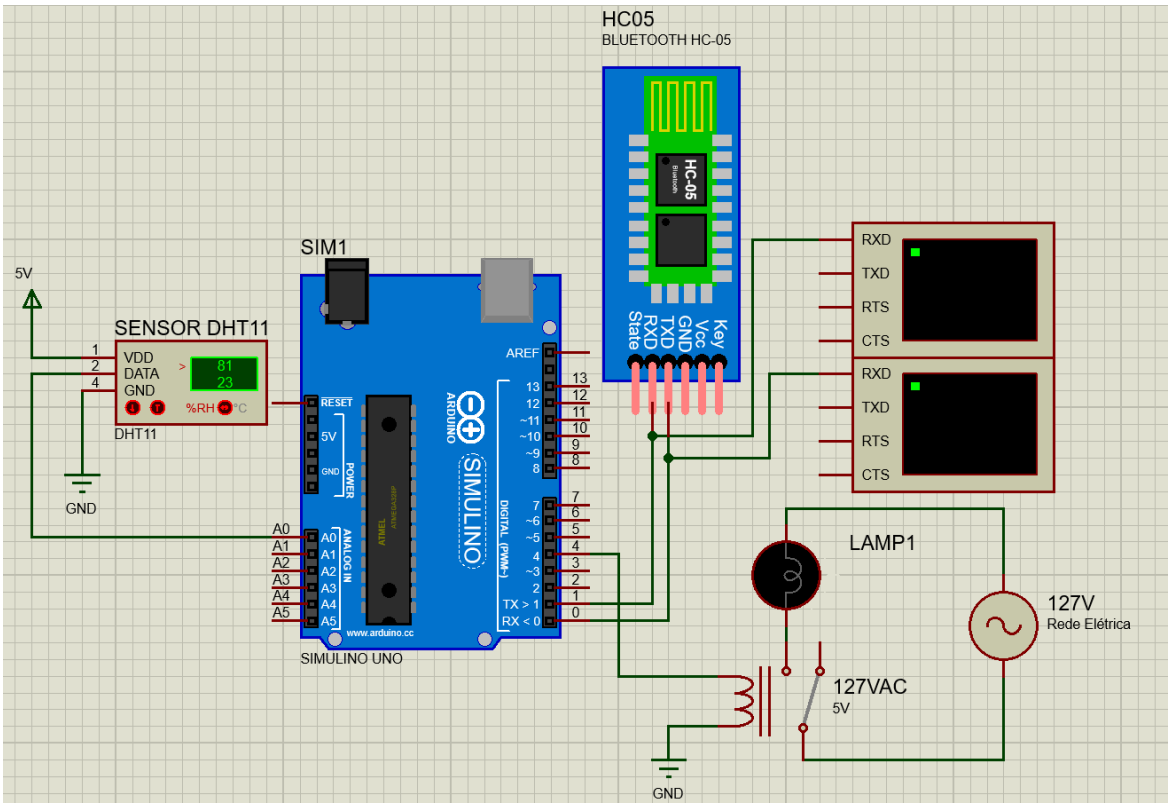
Figura 44 - Símbolo e localização do instrumento “Virtual Terminal”




Fonte: Elaborado pelo autor

Foram utilizados dois instrumentos “*Virtual Terminal*” para a entrada RX e a saída TX do módulo *bluetooth* HC-05. Em seguida o celular com o aplicativo instalado foi conectado, via *bluetooth*, ao *bluetooth* do *notebook* que executa o programa Proteus 8. A Figura 45 ilustra o circuito simulado e o esquema de conexão de ambos “*Virtual Terminal*”.

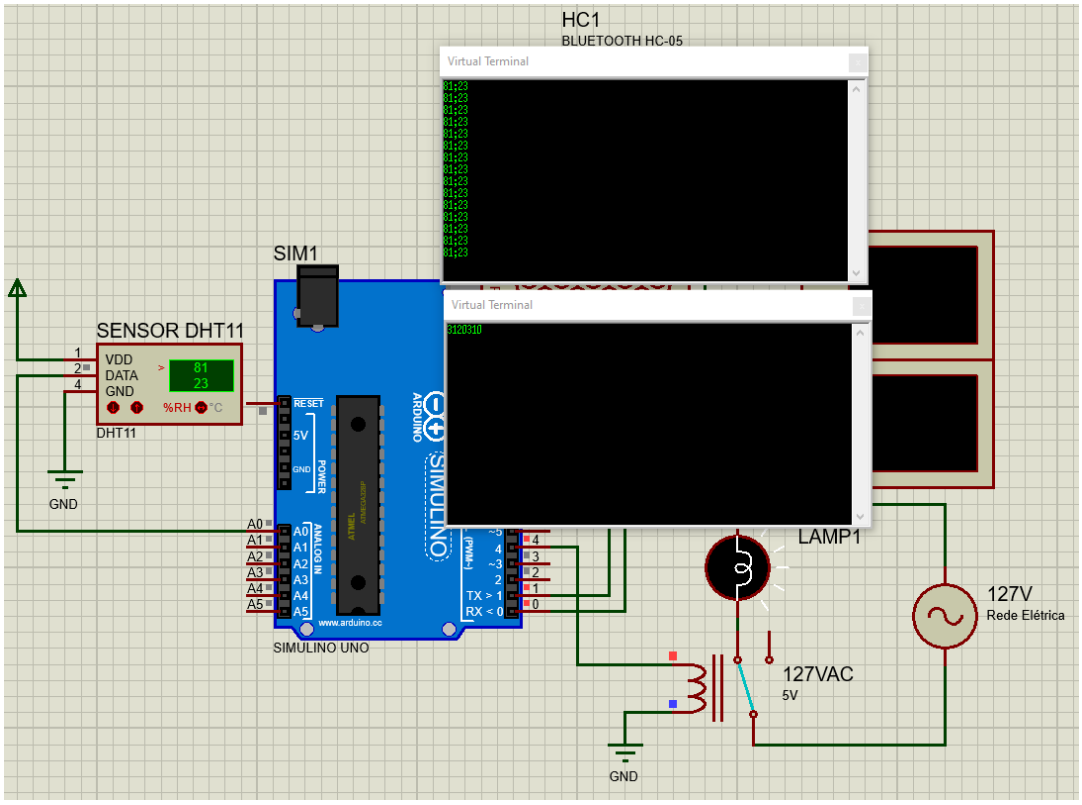
Figura 45 - Esquema de conexão de ambos Virtual Terminal




Fonte: Elaborado pelo autor

Com todas as conexões realizadas é possível dar início a simulação. Ao iniciar a simulação, as janelas de leitura dos instrumentos surgem e os valores de umidade e temperatura do sensor podem ser observados. Isso significa que o programa escrito e o Arduino simulado reconhecem o sensor DHT11, identificam o dado enviado por ele para o módulo HC-05, conforme ilustra a Figura 46.

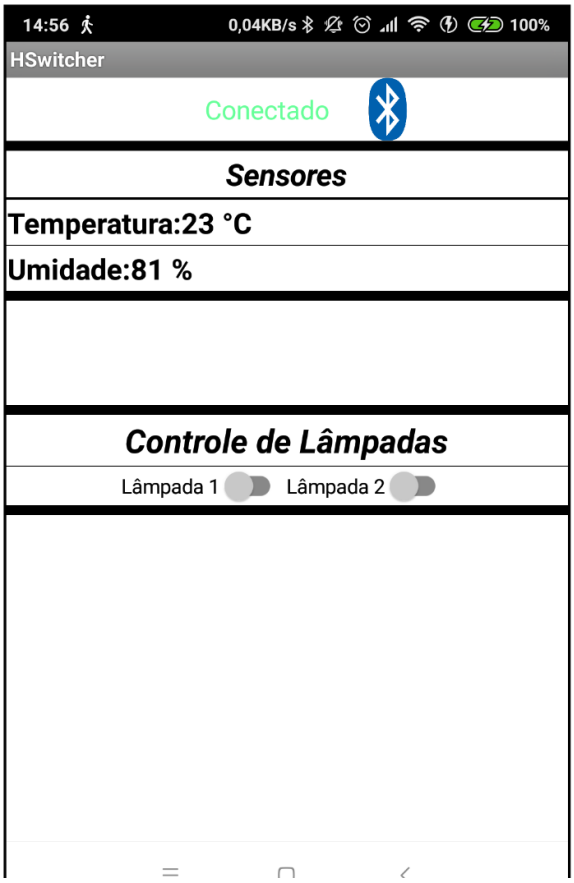
Figura 46 - Valores enviados e recebidos pelo módulo simulado




Fonte: Elaborado pelo autor

Em seguida, foi realizado o pareamento entre o celular com o aplicativo instalado e o *notebook* cuja simulação está ocorrendo. Os valores de temperatura e umidade são exibidos na tela do aplicativo e atualizados a cada 1 segundo, conforme determinado anteriormente. A Figura 47 mostra o estado de conexão do aplicativo e os valores de temperatura e umidade fornecidos pelo DHT11 simulado.

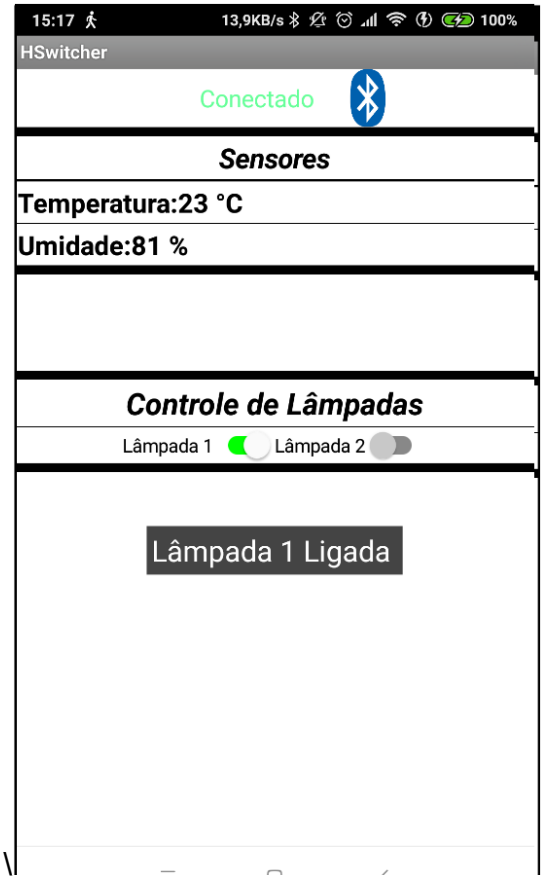
Figura 47 - Valores de temperatura e umidade no aplicativo




Fonte: Elaborado pelo autor

Ao ativar o componente *Switcher* referente â lâmpada 1, o aplicativo exibe uma notificação flutuante sinalizando o acionamento da lâmpada, conforme mostra a Figura 48.

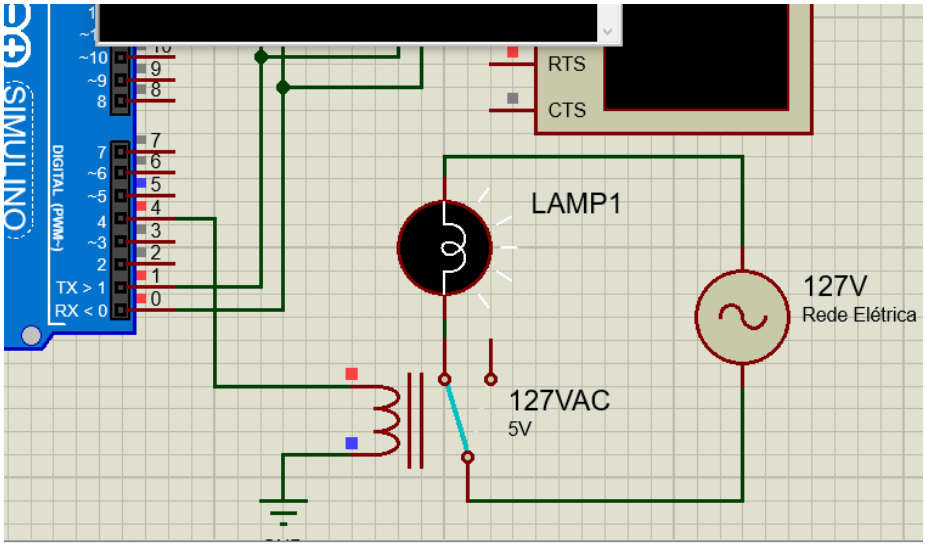
Figura 48 - Notificação após acionamento do *Switcher*




Fonte: Elaborado pelo autor

Nesse momento, o aplicativo envia via bluetooth o *byte* associado ao estado do componente, podendo ser observado na janela do *Virtual Terminal*. Também é possível verificar ainda que o Arduino ativa a saída digital 4 conectada ao relé e ascende a lâmpada da simulação, conforme ilustra a Figura 49.

Figura 49 - Notificação após acionamento do *Switcher*



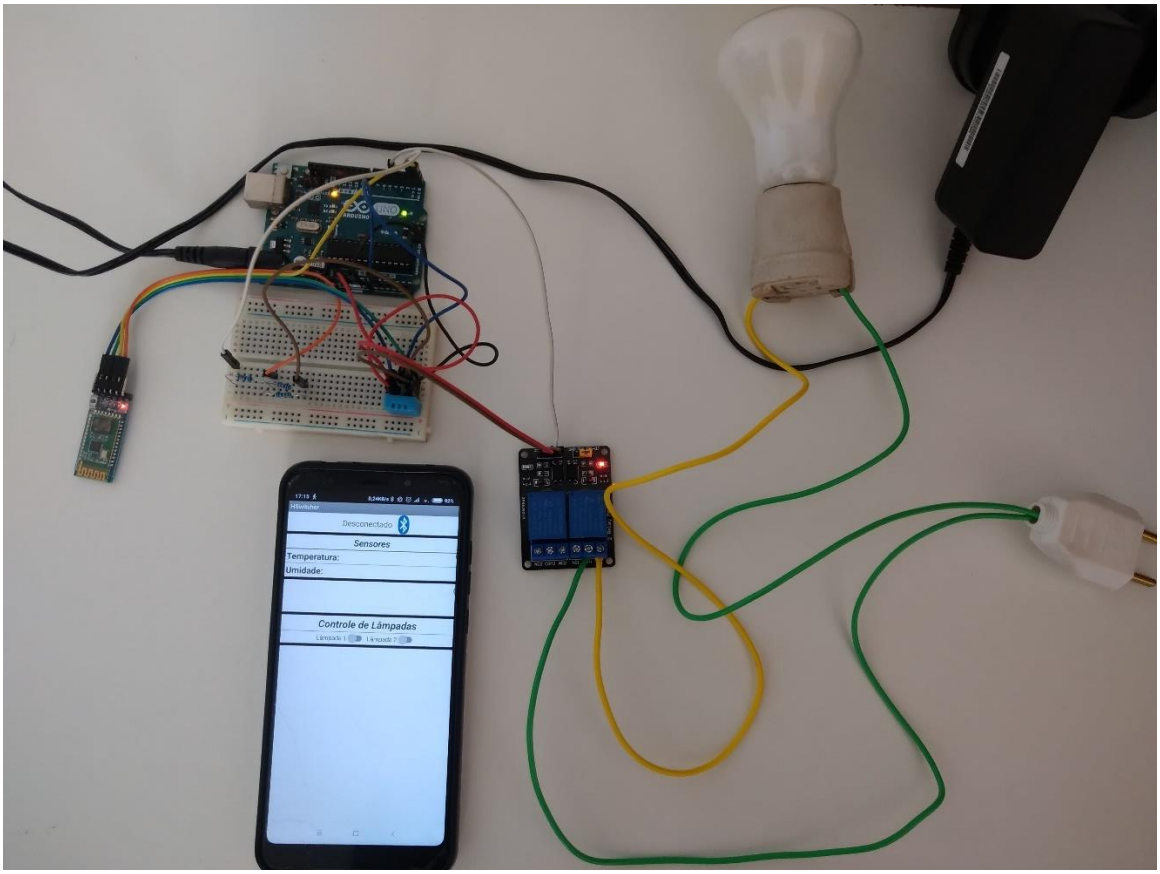

Fonte: Elaborado pelo autor

Ao final da simulação foi possível validar o funcionamento do aplicativo, o código utilizado no Arduino e o funcionamento do circuito proposto, tornando possível realizar o controle de iluminação. O projeto, apesar de completo, não se limita ao que foi proposto e pode ser ampliado para realizar o controle de um número maior de lâmpadas e até mesmo outros dispositivos conectados à rede elétrica, realizando as devidas adequações no aplicativo, acrescentando relés ao circuito e modificando o código do Arduino para utilizar um número maior de portas digitais.

# 7 TESTE DE BANCADA

Neste capítulo será apresentado o teste de bancada realizado após a simulação bem sucedida. Inicialmente foi montado o Circuito Geral da Figura 16, apresentado no Capitulo 3, com componentes físicos, conforme ilustra a Figura 50.

Figura 50 - Circuito Geral de Bancada




Fonte: Elaborado pelo autor

O Arduino foi conectado via USB ao computador para que o *Sketch* possa ser transferido, pelo *software* da IDE, e em seguida a placa foi desconectada do computador e alimentada com uma fonte 12 V e 1 A. A alimentação dos componentes foi realizada pela porta 5 V do próprio Arduino. Para o divisor de tensão, foi utilizado um resistor de 1 KΩ ligado em série com dois resistores de 1 KΩ, ligados em paralelo entre si. Os pinos TX e RX do HC-05 foram conectadas as portas seriais RX e TX do Arduino, respectivamente. Para bancada foi utilizado somente uma lâmpada, porém tanto o aplicativo como o Arduino estão programados para poder ativar duas lâmpadas, conectando ao pino IN2 do módulo relé a porta digital 5 do Arduino.

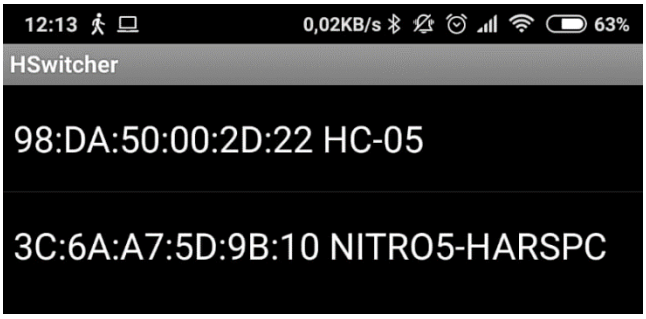
Com o circuito montado e conectado a alimentação, o aplicativo foi iniciado no celular e, como o *bluetooth* do aparelho se encontrava desativado, o App exibe uma notificação ao usuário, sugerindo sua ativação conforme ilustra a Figura 51.

Figura 51 - Notificação para ativar *Bluetooth*


Fonte: Elaborado pelo autor

Ao selecionar a opção “Permitir”, o *bluetooth* do celular é ativado e a conexão é realizada ao pressionar o botão com o símbolo do *bluetooth* e selecionando o dispositivo HC-05 na lista exibida pelo aplicativo, conforme ilustra a Figura 52.

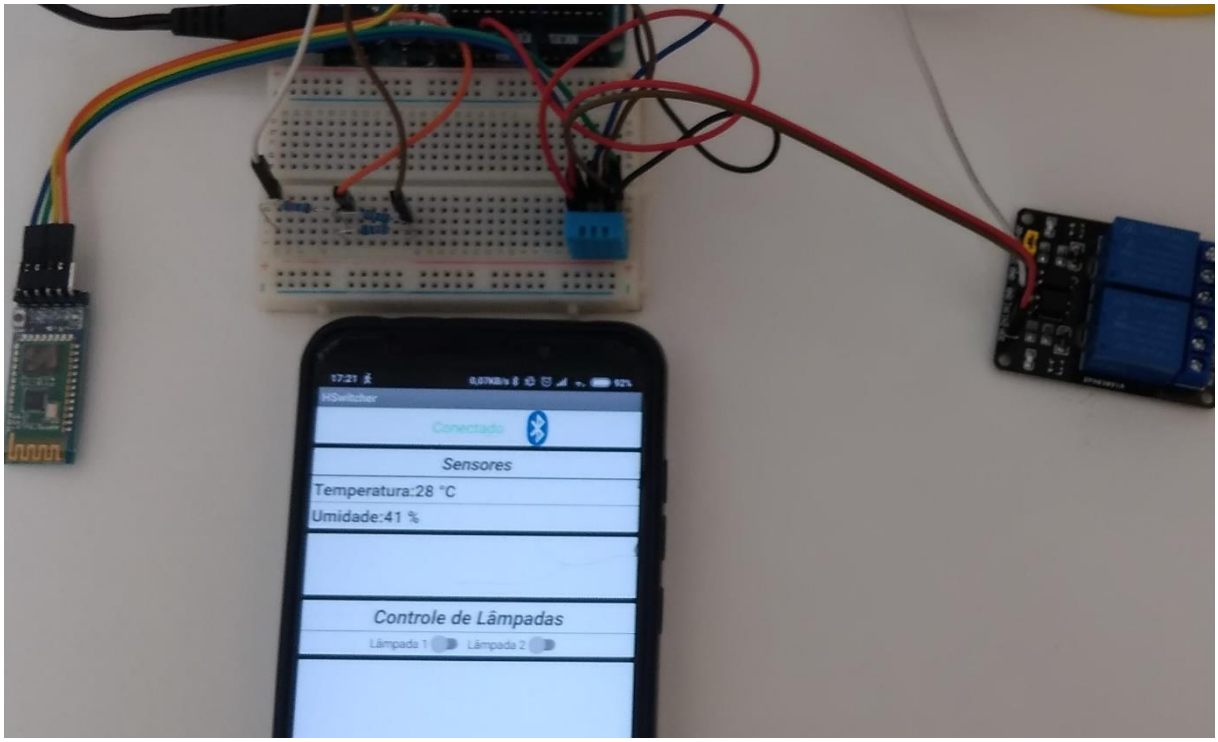
Figura 52 - Lista de dispositivos para conexão




Fonte: Elaborado pelo autor

Com a conexão bem sucedida, o texto ao lado do botão é alterado para “Conectado”, sua cor alterada para verde e é possível visualizar a leitura de temperatura e umidade relativa do ar, conforme mostra a Figura 53.

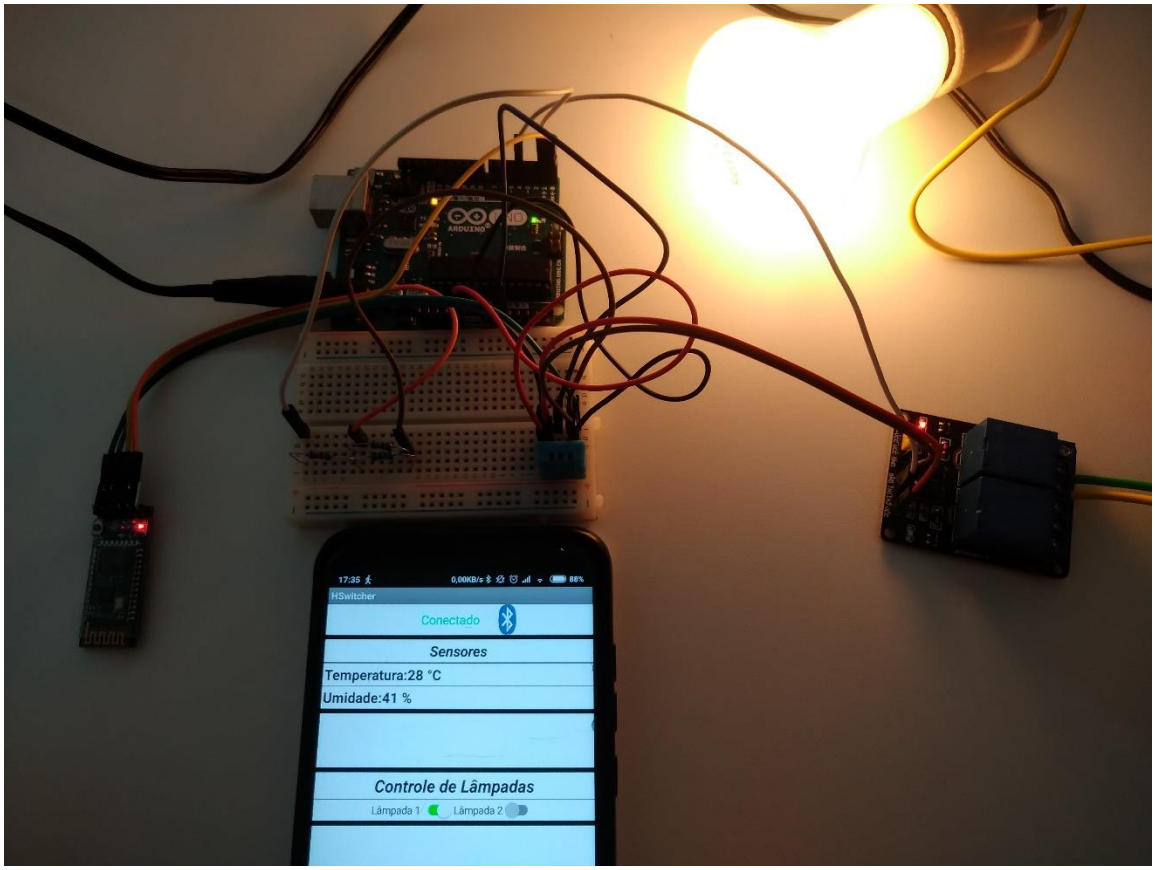
Figura 53 - Aplicativo conectado ao módulo *bluetooth*




Fonte: Elaborado pelo autor

Por fim, foi realizado o acionamento da lâmpada, alterando o estado do componente “Switch”. Com isso a o relé é ativado permitindo o acionamento da lâmpada, conforme ilustra a Figura 54.

Figura 54 - Lâmpada acionada pelo aplicativo




Fonte: Elaborado pelo autor

Por fim o circuito foi mantido ligado e o aplicativo executado, durante 1 hora, e o sistema se mostrou estável e funcional. Com o teste de bancada foi possível verificar que tanto o circuito como a programação do aplicativo e do Arduino funcionam.

Caso o usuário deseje expandir o projeto, deve ser realizado a adequação tanto do código do aplicativo, como do Arduino. Um número maior de relés deve ser utilizado para acionamento de novas cargas independentes. O modelo Uno apresenta 14 portas digitais, sendo que duas necessitam estar conectadas ao módulo HC05 e, portanto, somente 12 portas digitais estão disponíveis e caso o usuário necessite controlar um número maior de dispositivos, deve ser escolhido outro modelo de Arduino.

# 7 CONCLUSÕES

Foi apresentado um projeto de automação residencial baseado na conexão entre Arduino e aplicativo de celular desenvolvido utilizando a plataforma gratuita APP Inventor, o qual mostrou-se funcional, dada confiabilidade da simulação e do teste de bancada, e de baixo custo que permite ao usuário personalizar suas funções conforme sua necessidade. A variedade de componentes presentes tanto na plataforma APP Inventor, como para Arduino, em conjunto com a suas intuitivas linguagens de programação, tornam-nas ferramentas de projeto com potencial de expansão. Contudo, caso o desenvolvedor não antecipe possíveis necessidades ou alterações do projeto, é necessário realizar a reprogramação do aplicativo e do Arduino e com isso aumentar seu tempo de desenvolvimento.

## REFERÊNCIAS

AFONSO, B. S.; PEREIRA, R. B.; PEREIRA, M. F. Utilização da Internet das Coisas para o desenvolvimento de miniestação de baixo custo para monitoramento de condições do tempo em áreas agrícolas. *In:* Anais da Escola Regional de Informática da Sociedade Brasileira de Computação (SBC) – Regional de Mato Grosso, 6., 2015, Mato Grosso. **Anais** [...] Mato Grosso: ERI, 2015. p. 183-189.

ARDUINO. What is Arduino? 2018. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em 2021.

AUTOMAÇÃO residencial auxilia moradores desde tarefas simples até aumentar a segurança do imóvel. **G1 - Mercado Imobiliário do Interior**, 2020. Disponível em: <https://g1.globo.com/sp/sao-jose-do-rio-preto-aracatuba/mercado-imobiliario-do-interior/noticia/2020/04/01/automacao-residencial-auxilia-moradores-desde-tarefas-simples-ate-aumentar-a-seguranca-do-imovel.ghtml>. Acesso em 2021.

BANZI, M. **Primeiros Passos com Arduino**. São Paulo: Novatec, 2012. 152 p. ISBN 978-85-7522-290-4.

BLUETOOTH Adapter. **Developers,** 2022. Disponível em: <https://developer.android.com/reference/android/*bluetooth*/*Bluetooth*Adapter>. Aceso em 2022.

BOLZANI, C. A. M. **Análise de Arquiteturas e Desenvolvimento de uma Plataforma para Residências Inteligentes**. 2010. 155f. Tese (Doutorado em Engenharia Elétrica) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2010.

CORE ELECTRONICS. Arduino Boards, Compared. 2018. Disponível em: <https://core-electronics.com.au/tutorials/compare-Arduino-boards.html>. Acesso em 2021.

D-ROBOTICS. DHT11 Humidity & Temperature Sensor. 2010. Disponível em <https://datasheetspdf.com/pdf-file/785590/D-Robotics/DHT11/1>. Acesso em 2021.

EVANS, M; NOBLE, J; HOCHENBAUM, J. **Arduino em Ação**. São Paulo: Novatec, 2013. p. 25- 26. ISBN 978-85-7522-373-4.

GROVER, S.; PEA, R. Computational thinking in K–12: A review of the state of the field. **Educational Researcher**, v. 42, n. 1, p. 38-43, 2013.

LABCENTER. About Labcenter. Disponível em: <https://www.labcenter.com/about/>. Acesso em 2021

LIMA. S. M. E.; NOBRE. M. Y. A.; ALENCAR. E. A. R. Automação Residencial de Baixo Custo com Arduino Mega e Ethernet Shield. Centro Universitário Estácio do Ceará, Ceará, 17 p., 2015.

LYE, S. Y.; KOH, J. H. L. Review on teaching and learning of computational thinking through programming: What is next for K-12? **Computers in Human Behavior**, v. 41, p. 51-61, 2014.

MILLER, M. **Descobrindo *Bluetooth*.** Rio de Janeiro: Campus, 2001. 289 p.

MIT APP INVENTOR. About Us. 2019. Disponível em: <https://APPinventor.mit.edu/about-us>. Acesso em 2021.

MORAES, C. C.; CASTRUCCI, P. **Engenharia de automação industrial**. Rio de Janeiro: LTC Livros Técnicos e Científicos S. A., 2007. 361p

MULTILÓGICA-SHOP. Módulo blueooth HC-05. Disponível em: <https://multilogica-shop.com/modulo-*bluetooth*-hc-05>. Acesso em 2021.

MURATORI, J. R.; DAL BÓ, P.H. Automação residencial: histórico, definições e conceitos. **O Setor Elétrico**, v. 62, 2011.

PINOUTS.NET**.** Humidity and Rain Detection Sensor Module PinOuts and Tutorials. 2019. Disponível em: <https://pinouts.net/humidity-and-rain-detection-sensor-module-pinouts-and-tutorials/>. Acesso em 2021.

PRUDENTE, F. **Automação Predial e Residencial: Uma introdução**. Rio de Janeiro: LTC, 2013.

RIOS, J. *et al*. Introdução ao Arduino. Universidade Federal do Mato Grosso do Sul: Faculdade de Computação, Mato Grosso do Sul, 2012. Disponível em: <http://www.academia.edu/5229813/Introdu%C3%A7%C3%A3o_ao_Arduino>. Acesso em 2021.

SANTOS, A.; AMORIM, H.; DERECZYNSKI, C. Investigação do fenômeno ilha de calor urbana através da utilização da placa Arduino e de um sítio oficial de meteorologia. **Revista Brasileira de Ensino de Física**, v. 39, n. 1, e1505, 2017.

SOUZA, F. Arduino Uno. **Embarcados**, 2013. Disponível em: <https://www.embarcados.com.br/Arduino-uno/#Conclusao>. Acesso em 2021.

WILENSKY, U.; BRADY, C. E.; HORN, M. S. Fostering computational literacy in Science classrooms. **Communications of the ACM**, v. 57, n. 8, p. 24-28, 2014.