FACULDADE DE TECNOLOGIA DE AMERICANA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDERSON HENRIQUE DE SÁ SILVA

SAMUEL VICTOR VICENTE

AUTOMAÇÃO RESIDENCIAL - CASA INTELIGENTE

AMERICANA

2018

FACULDADE DE TECNOLOGIA DE AMERICANA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDERSON HENRIQUE DE SÁ SILVA

SAMUEL VICTOR VICENTE

AUTOMAÇÃO RESIDENCIAL - CASA INTELIGENTE

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Americana como parte dos requisitos para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

AMERICANA
2018

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

S578aSILVA, Anderson Henrique de Sá

Automação residencial: casa inteligente. / Anderson Henrique de Sá Silva, Samuel Victor Vicente. – Americana, 2018.

90f.

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Clerivaldo José Roccia

1 Domótica 2. Android – aplicativos I. VICENTE, Samuel Victor II. ROCCIA, Clerivaldo José III. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.518

Faculdade de Tecnologia de Americana

Anderson Henrique de Sá Silva
Samuel Victor Vicente

Automação Residencial – Casa Inteligente

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana.
Área de concentração: Projeto.

Americana, 05 de dezembro de 2018.

Banca Examinadora:

Clerivaldo José Roccia (Presidente)
Doutor
FATEC-Americana

Antonio Alfredo Lacerda (Membro)
Especialista
FATEC-Americana

Diógenes de Oliveira (Membro)
Mestre
FATEC-Americana

Resumo

Esse trabalho de conclusão de curso tem como objetivo o desenvolvimento de um software para automatizar uma residência, isso visa o auxílio de processos diários, como ligar uma luz e ventilador, também auxiliando na economia de água e energia. A sustentabilidade vem crescendo e sendo adotada a cada dia, com ela é possível cuidar do nosso meio ambiente e tentar dar uma vida melhor para as gerações futuras. Tanto a energia quanto a água devem ser economizados para que não ocorra a falta dos mesmos. O software proposto nesse trabalho de conclusão de curso auxiliará o usuário a realizar tarefas do dia a dia de forma fácil e ágil, também dando mais controle sobre o quanto de gasto em energia e/ou água está ocorrendo no banho, auxiliando pessoas que não estão dispostas a levantar ou pessoas com deficiência, a apagar a luz que está longe e/ou ligar e desligar um ventilador.

Palavras Chaves: Android – aplicativos, Arduino, Desenvolvimento de software

Abstract

This course completion work has the objective of developing a software to automate a residence, this is the aid of daily processes, such as turning on a light and a fan, also helping to save water and energy. Sustainability is growing and being adopted every day, with it it is possible to take care of our environment and try to give a better life for future generations. Both energy and water must be saved so that there is no shortage. The software proposed in this course completion work will help the user to perform day-to-day tasks in an easy and agile manner, also giving more control over how much energy and / or water expenditure is taking place in the bath, assisting people who are not willing to lift or disabled people, to turn off the light that is away and / or turn on and off a fan.

Keywords: Android – applications, Arduino, Software development.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1) INTRODUÇÃO

1.1) CONTEXTUALIZAÇÃO

Com o avanço da tecnologia na nossa atualidade, tornar as coisas mais fáceis define quem lidera o mercado, ou seja, inovar de forma eficiente se tornou mais do que necessário para aqueles que planejam ter produtos cobiçados.

Sabemos que a tecnologia é algo indispensável nessa era de inovação, junto com a tecnologia um dos grandes desafios do século XXI é melhorar de forma eficiente os processos tecnológicos das mais diversas áreas.

Uma das formas de processos tecnológicos que vem ganhando espaço no mundo da tecnologia é a automação, sendo ela residencial ou industrial.

1.2) PROBLEMA

Desenvolver um software de automação residencial, que possa também ajudar pessoas com deficiências físicas e auxiliar nos processos diários como o de ligar e desligar uma luz.

1.3) HIPÓTESE

A implementação de um software para gerenciar de forma inteligente e automatizada os principais recursos de uma residência como luz, caixa d'água, banho através de uma placa controladora Arduino e uma aplicação Android.

1.4) MOTIVAÇÃO

Analisando o ambiente interno residencial, a ideia de implementar um software gerenciador de recursos veio à tona, tendo como base a nova era da tecnologia. O software irá controlar uma placa Arduino por meio de uma aplicação Android, trazendo de forma simples e eficiente o controle interno de recursos de uma residência que poderá ser controlada por um smartphone ou tablet. A vantagem e comodidade que a junção entre a placa Arduino com a aplicação Android irá proporcionar para o usuário, é transformar processos manuais em automáticos. Analisando esses processos, reparamos que as coisas simples do dia a dia podem ser automatizadas, sem ter um custo absurdamente alto, como por exemplo, acender e apagar uma lâmpada, ligar um ventilador e até mesmo obter informações climáticas internas como temperatura e umidade.

1.5) JUSTIFICATIVA

O sistema também auxiliaria no processo de economia de água e energia. Conta com o recurso de monitoração em tempo real do reservatório de água, mostrando a quantidade de água presente no reservatório, disponibilizando ao usuário o acesso a informação de consumo diário, semanal e mensal.

Com a possibilidade de controle sobre a iluminação de diferentes pontos da casa, usando o aplicativo em seu smartphone, o usuário poderá acender e apagar as luzes, através de um botão presente no aplicativo. Essa possibilidade e facilidade de controlar equipamentos elétricos, traz de forma clara e objetiva a visualização de quais equipamentos estão sendo utilizado, e indiretamente faz com que o usuário se policie no uso dos equipamentos, desligando quando necessário e resultando na economia de energia.

Um dos recursos presente no aplicativo, é o acesso a temperatura interna da residência. A aplicação de forma automática, irá ligar o ventilador quando o sensor presente no Arduino perceber que houve aumento de temperatura que o usuário considere calor, mas também oferecerá ao usuário através de um botão ou comando de voz, a possibilidade de ligar o ventilador com um simples clique no botão.

1.6) OBJETIVOS

1.6.1) OBJETIVOS GERAIS

Desenvolver um software de automação residencial.

1.6.2) OBJETIVOS ESPECÍFICOS

- Desenvolver diagrama de caso de uso
- Requisitos funcionais e não funcionais
- Desenvolver diagrama de classe
- Desenvolver o diagrama de sequência
- Desenvolver o MER
- Desenvolvimento do aplicativo de controle
- Desenvolvimento do software de controle do Arduino

2) FUNDAMENTAÇÃO TEÓRICA

2.1) A Tecnologia no mundo contemporâneo

A utilização de tecnologia de forma geral se tornou muito comum no mundo contemporâneo, está presente tanto em meios empresariais quanto em nossa vida pessoal.

Uma definição sucinta sobre tecnologia da informação se dá pelo conjunto de hardware[1] e software[2] Cf. Boniati,2014, p.25-44 que irá desempenhar uma ou várias tarefas que serão coletadas a partir de informações e processadas. Como coleta, transmissão, estoque, recuperação, manipulação e exibição de dados. (FILHO, 1994).

2.1.1) Automação

Um problema que no passado a indústria sofria em fabricações dos mais variados produtos, era o processo manual e demorado, causando lentidão para o produto chegar no mercado e ser comercializado.

Com os avanços tecnológicos a automação chegou na indústria, trazendo inestimáveis benefícios para várias empresas. A partir do conceito de automação industrial, surge uma nova forma de automatizar processos, trazendo de forma simples e eficiente a automação integrada para o uso doméstico.

Segundo MURATORI e DAL BÓ a definição de Automação residencial integrada é:

> É um processo que, usando diferentes soluções e equipamentos, possibilita ao usuário usufruir o máximo de qualidade de vida na sua habitação. (MURATORI e DAL BÓ, 2013, p.15).

2.1.2) Domótica

A Domótica é a nova forma de se referir à automação residencial, originalizada com a junção da palavra latina Domus (casa) com termo Robótica (controle automatizado).

O princípio da Domótica é tornar processos manuais em automáticos, utilizando a tecnologia como principal recurso para automatizar tarefas, e facilitar o controle interno de uma casa.

A automação residencial traz recursos de muita importância para os habitantes de uma residência, com uma vasta possibilidade de automatização integrada, gerando de forma simplificada conforto, segurança, economia e controle interno. Trentin ressalta que:

> [...] permite a usuários o controle de suas casas, como fechar janelas, alterar temperatura do ar condicionado, dependendo da hora do dia, ligar e desligar luzes, abrir e fechar portões, entre outras possibilidades. (Trentin, 2012, p. 9)

O Conforto proporcionado pela AR(Automação Residencial) faz com que tarefas comuns do dia a dia como por exemplo, ligar ou deligar luz, ar-condicionado, ventilador, abrir cortina, portão eletrônico, portas, tarefas que são feita manualmente, podem ser automatizada de forma que o morador através de um smartphone ou controle remoto, faça isso com poucos cliques, ou programar cada tarefa para que seja executada automaticamente, definindo previamente cada processo de acordo com suas necessidades.

Com a tecnologia presente nas casas, é possível tornar tudo mais seguro. A possibilidade de implementação de câmeras de segurança, alarmes e sensores de presença, traz para o morador uma garantia amais de segurança ao seu patrimônio. Tendo monitoramento em tempo real das câmeras de segurança através de seu smartphone, o usuário do sistema de automação poderá ativar alarmes, ligar para a polícia e manda a localização para que a intervenção ocorra de forma rápida e precisa, e ainda melhor, sem precisar estar em casa. O smartphone contando com um aplicativo gerenciador do sistema de automatização, pode notificar o usuário de

invasões, incêndio, e até mesmo vazamento de água e gás, porém algum destes assuntos não serão abordados neste trabalho de conclusão de curso.

> A possibilidade da substituição de atos e decisões humanas, por atos e decisões processados por computadores, devidamente alimentados de informações, no comando de determinados dispositivos, geralmente em processos repetitivos, ou que exijam esforços físicos, reduzindo a possibilidade de erros nas decisões sujeitas à emoção, cansaço, dúvida, tempo e inexperiência, entre outras razões. (BARBOSA; QUALHARINI, 2004, p.3).

2.2) Requisitos

A análise de requisitos se torna um fato importante para a arquitetura de software, para que seja desenvolvido um produto de qualidade e com estrutura. Segundo o dicionário online Michaelis a definição de requisito se dá por: “Condição ou exigência imprescindível a que se deve satisfazer para alcançar determinado fim” (Michaelis, 2018). Sendo assim, é essencial para a definição do projeto ser analisado os requisitos que o mesmo deve exercer.

2.2.1) Requisitos funcionais

Os requisitos funcionais devem deixar explícito o que o sistema deve realizar, como diz uma definição do Sommerville:

> Os requisitos funcionais de um sistema descrevem o que o sistema deve fazer. Esses requisitos dependem do tipo do software que está sendo desenvolvido, dos usuários a que o software se destina e da abordagem geral considerada pela organização ao redigir os requisitos. Quando expressos como requisitos de usuário, eles são geralmente descritos de forma bastante abstrata. No entanto, os requisitos funcionais descrevem a função do sistema detalhadamente, suas entradas e saídas, exceções etc. (SOMMERVILLE, 2007, p.81).

Logo abaixo segue uma tabela de exemplo de requisitos funcionais

Tabela 1 – Requisitos Funcionais

Requisitos funcionais

IItem	Requisito
11	O usuário deve ser capaz de fazer uma busca em todo o conjunto inicial do banco de dados ou selecionar um subconjunto com base nele
12	O sistema deve fornecer telas apropriadas para o usuário ler os documentos no repositório de documentos
33	Para cada pedido, deve ser alocado um único identificador (ORDER_ID), o qual o usuário deve ser capaz de copiar para a área de armazenamento permanente da conta

Fonte: Sommerville,2007

2.2.2) Requisitos não funcionais

Os requisitos não funcionais são o inverso dos funcionais, ou seja, não estão diretamente relacionados a funções específicas como diz o autor citado acima:

> Os requisitos não funcionais, como o nome sugere, são aqueles não diretamente relacionados às funções específicas fornecidas pelo sistema. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e espaço de armazenamento. (…) (SOMMERVILLE, 2007, p.82).

Segue um exemplo de requisitos não funcionais logo abaixo

Tabela 2 – Requisitos não funcionais

Requisitos não funcionais

Propriedade	**Medida**
Velocidade	Transações processadas/segundo Tempo de resposta de usuário/evento Tempo de atualização da tela
Tamanho	Kbyte Número de chips de RAM
Facilidade de uso	Tempo de treinamento Número de frames de ajuda
Confiabilidade	Tempo médio de falha Probabilidade de indisponibilidade Taxa de ocorrência de falhas Disponibilidade
Robustez	Tempo para reiniciar após falha - Porcentagem de eventos que causam falhas - Probabilidade de corrupção de dados por falhas
Portabilidade	Porcentagem de declarações dependentes do sistema-alvo Número de sistema-alvo

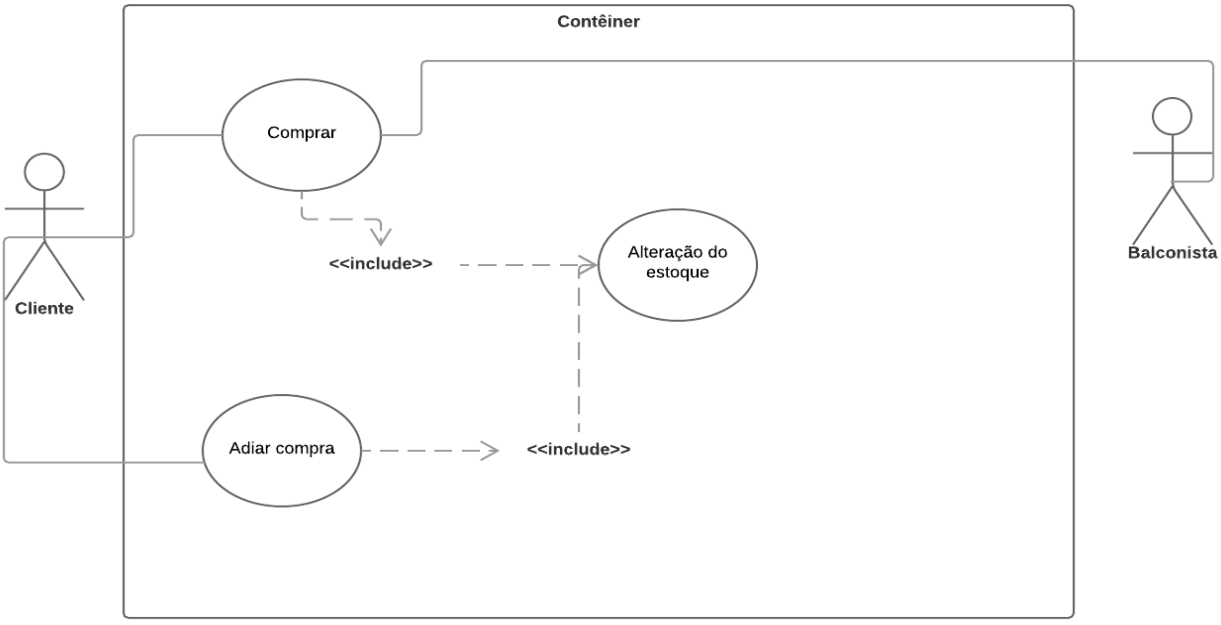
Fonte: Sommerville,2007

2.3) O Diagrama de Caso de Uso

Segundo o site MSDN da Microsoft:

> Um diagrama de caso de uso age como um foco para a descrição dos requisitos do usuário. Ele descreve as relações entre requisitos, os usuários e os principais componentes. Ele descreve os requisitos em detalhes; eles podem ser descritos em diagramas separados ou em documentos que podem ser vinculados a cada caso de uso (MSDN, 2016).

Figura 1 – Exemplo de diagrama de caso de uso



Fonte: Própria (2018)

Esse diagrama facilita a comunicação entre os analistas e os clientes, fazendo o cliente enxergar de forma mais clara um cenário com as funcionalidades do sistema. Você pode estar verificando um exemplo do diagrama de caso de uso na figura 1.

Uma breve descrição sobre o diagrama de caso de uso é abordada a seguir:
Ator: O ator (1) segundo o site MSDN da Microsoft é definido como:

“Uma classe de organização, pessoa, dispositivo ou componente de software externo que interage com o seu sistema. Atores de exemplo são clientes, restaurante, Sensor de temperatura, autorizador de cartão de crédito” (MSDN, 2016).

Caso de uso: “Um *caso de uso* (2) representa as ações executadas por um ou mais atores em busca de uma meta específica. Casos de uso de exemplo são refeição ordem, o Menu de atualização, processo de pagamento.” (MSDN, 2016).

Associação: Em um diagrama de caso de uso, o ator será associado com o ou os casos de usos relativos as ações dele.

Segue uma breve definição da utilidade do sistema no caso de uso:

> O *sistema (4)* é tudo o que você está desenvolvendo. Pode ser um componente de software pequeno, cujos atores são apenas outros componentes de software; ou pode ser um aplicativo completo; ou pode ser um grande conjunto distribuído dos aplicativos implantados em vários computadores e dispositivos. Subsistemas de exemplo são refeição pedidos site, empresa de entrega de refeição, versão 2 do site (MSDN, 2016).

2.3.1) Diagrama de Classe

O diagrama de classe é bem usado quando se fala em documentação do sistema, ele visa detalhar classes, métodos e atributos para auxiliar na definição do desenvolvimento. Segundo o autor Alberto Silva:

> Os diagramas de classes são usados para modelar a estrutura de um sistema. Estes modelos são também designados por "vista do desenho estático do sistema" e são usados tipicamente em três situações: (1) para modelar o vocabulário de um sistema; (2) para modelar colaborações simples; e (3) para modelar o desenho de um esquema de uma base de dados (SILVA, 2001, p.186).

Com o diagrama de classe desenvolvido, fica mais fácil o desenvolvimento do software pois as classes, atributos e métodos da mesma, já foram, portanto, esclarecidos e documentados.

2.3.2) Classe

Uma definição perfeita para classe é a do dicionário online michaelis com a seguinte definição: "Grupo de pessoas, animais ou coisas com atributos semelhantes.".

Ou seja, é considerado classe, um grupo seja ele de pessoas, animais, objetos. Mas é de extrema importância que eles contenham atributos semelhantes, como por exemplo a classe carro, um carro é identificado pelos seus atributos, como quantidade de rodas, motor, aparência entre outros atributos que dão a entender que aquele objeto é um carro.

2.3.3) Objeto

Um objeto na programação é uma instancia de uma classe, ou seja, o objeto é dar a vida a ela de certa forma, quando um objeto é criado ele herda as funções e atributos da mesma. A melhor definição encontrada no dicionário virtual Michaelis é:

“Qualquer coisa (física ou mental) para a qual uma ação, um pensamento ou sentimento se dirige” (MICHAELIS, 2018).

Explicando a partir do exemplo anterior, temos a classe carro, mas podemos ter um carro do tipo fusca, esse fusca contém uma série de atributos e funções que fazem identificarmos que ele pertence à classe carro. Dito isso, o fusca é um objeto da classe carro.

2.3.4) Métodos

Uma classe pode ou não conter métodos, também conhecidos como funções que ela exercerá quando necessário. Ainda com o exemplo do carro, o carro tem a função de andar, freiar etc. Segundo o dicionário virtual Michaelis um método se da por “Emprego de procedimentos ou meios para a realização de algo, seguindo um planejamento; rumo.” (MICHAELIS, 2018).

Os métodos em softwares podem ter retornos, portanto, podem devolver algum valor assim que exercem suas funções, esses valores podem vir em forma de texto normalmente definidos como Strings, podem vir como números podendo ser com números quebrados normalmente chamados de Double ou Float ou números inteiros chamados de Int, também podem vir como verdadeiro ou falso chamados de boolean, ou até sem nenhum retorno chamado de Void, esses são os principais porém existem outros tipos de retorno.

É importante ressaltar que esses tipos de retornos não são somente utilizados em retornos de métodos, podem ser definidos como tipos de atributo também. Um exemplo seria o atributo idade esse atributo é um tipo numérico inteiro, sendo assim ele seria do tipo Int.

2.3.5) Encapsulamento

Tanto métodos, quanto classes e objetos podem conter diferentes tipos de visibilidades ou qualificadores, as visibilidades podem ser publicas, privadas ou protegidas. Percebe-se que o encapsulamento é utilizado para restringir acessos a esses itens, uma definição de encapsulamento segundo Victorio Albani de Carvalho é:

> Encapsulamento é uma técnica utilizada para restringir o acesso a variáveis (atributos), métodos ou até à própria classe. Os detalhes da implementação ficam ocultos ao usuário da classe, ou seja, o usuário passa a utilizar os mé- todos de uma classe sem se preocupar com detalhes sobre como o método foi implementado internamente (CARVALHO, 2012, p.51).

Para melhorar o entendimento vamos reutilizar o exemplo do carro, um carro em seu aspecto básico tem a funcionalidade de andar, porém o motorista que o dirige não precisa saber como o carro faz isso, ou seja, ele não precisa saber como o motor reage e outras funcionalidades necessárias para que o carro ande, além das habilidades para dirigir.

Os tipos de visibilidades são explicados a seguir:

Publica(public): O atributo, classe e/ou método pode ser utilizado por qualquer classe, ou seja, qualquer um pode utiliza-lo. A identificação para publico no diagrama de classe se da pelo sinal “+”.

Privado(private): O atributo, classe e/ou método pode ser acessado somente pela própria classe. A identificação para o privado no diagrama de classe se da pelo sinal “-”.

Protegidos(protected): O atributo, classe e/ou método pode ser acessado somente pela própria classe, classes do mesmo pacote ou classes da mesma hierárquia. A identificação para protegido no diagrama de classe da pelo sinal “#”.

2.3.6) Exemplo de diagrama de classes

Segue abaixo uma imagem de exemplo de um diagrama de classe, sendo a primeira sessão de cada quadrado (classe) os atributos e a segunda sessão os métodos que aquela classe compõe.

Figura 2 – Diagrama de classe

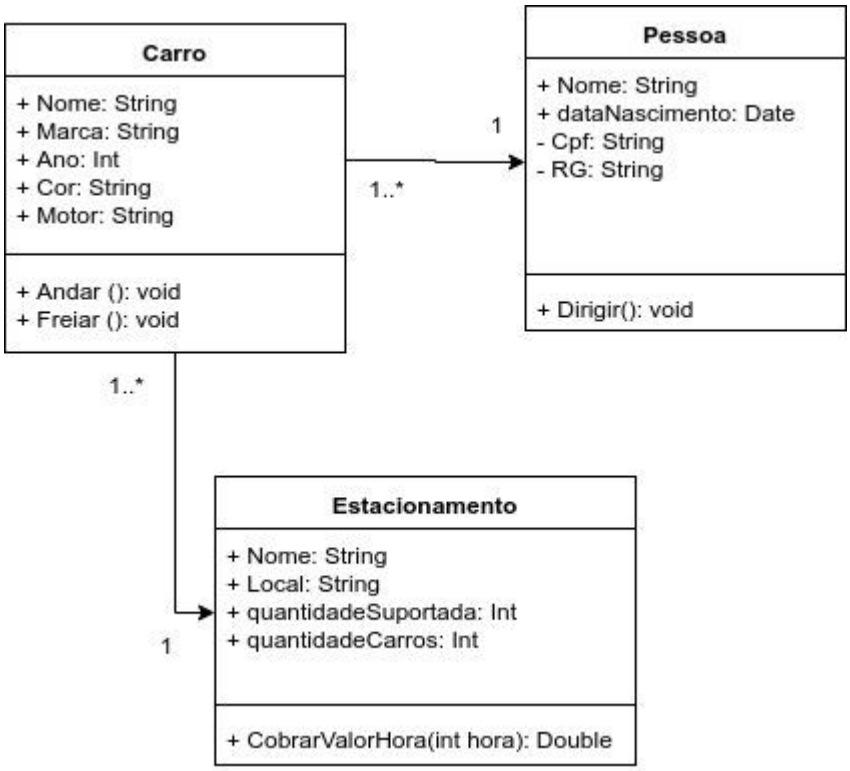


Figura: Elaborada pelo autor (2018)

2.4) A plataforma Arduino

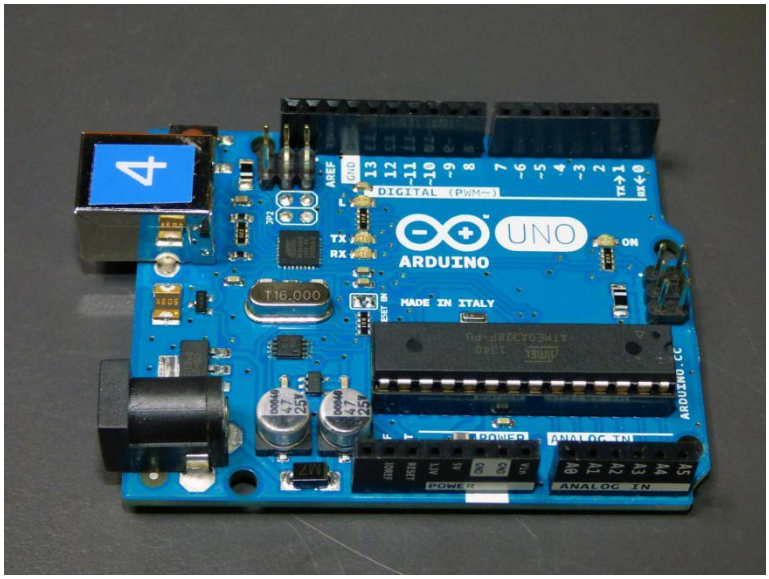
Arduino é uma plataforma open source[1] de hardware, software e conteúdo, ela também conta com uma comunidade global. O fato de ter uma grande comunidade é interessante pois existem facilidades em resolver problemas encontrados, ou de encontrar componentes e dicas sobre arduino.

Basicamente o Arduino se consiste em uma placa com vários componentes eletrônicos, que visam fazer entrada e saída de comandos. Esses comandos são enviados em forma de pulsos elétricos que podem ser ligados ou desligados.

As comunicações com os hardwares não são realizadas com linguagem comum, ou seja, linguagem de humanos. Essa comunicação é realizada por meio de linguagem de máquina, que converte os comandos para esses pulsos de ligado e desligado fazendo com que a máquina reaja de tal forma.

No projeto será utilizado o modelo Arduino UNO (figura 3).

Figura 3 – Arduino UNO



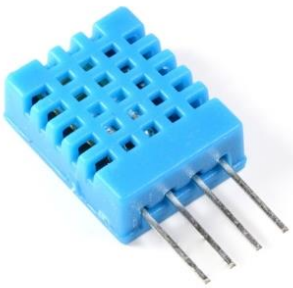
Fonte: https://pixabay.com/pt/circuito-integrado-computador-441289/

2.4.1) Sensores utilizados

2.4.1.1) DHT11

Esse componente é utilizado com o objetivo de captar a temperatura e umidade do local em que estiver instalado. Segue uma imagem a seguir:

Figura 4 – Sensor DHT11



Fonte:https://www.filipeflop.com/blog/monitorando-temperatura-e-umidade-com-o-sensor-dht11/

2.4.1.2) Relé

O componente relé funciona como um interruptor e é utilizado para eletrônicos 110V ou 220V, que precisa receber comando de um dispositivo 5V, assim impedindo que o dispositivo queime, e ao mesmo tempo execute os comandos que o dispositivo envia.

Figura 5 – Relé

Fonte: https://www.filipeflop.com/produto/rele-5v-songle-srd-05vdc-sl-c/

2.4.1.3) HC06

O sensor HC06 é um modulo bluetooth compatível com o Arduino, que é usado para enviar e receber dados e comandos.

Figura 6 – HC06

Fonte: https://www.embarcados.com.br/modulos-bluetooth-hc-05-e-hc-06/

2.4.1.4) Motor

Motor de 3V de potência utilizado para rotacionar algo que se deseja rodar ou mover através de uma correia ou corda e outros afins.

Figura 7 – Motor

Fonte: https://www.filipeflop.com/produto/mini-motor-dc-3v/

2.4.1.5) Lâmpada

Luz branca de LED 9W, será utilizada no projeto para iluminar o ambiente do mesmo.

Figura 8 – Lâmpada

Fonte: https://www.magazineluiza.com.br/lampada-de-led-compacta-9w-branca-fria-6500k-bivolt-ol-iluminacao-/p/6651825/cj/lald/

2.4.1.6) Sensor ultrassônico

Sensor ultrassônico emite ondas sonoras que assim que é repelido em um objeto ou parede a mesma volta para ele, a distância é calculada a partir do tempo em que a onda demorou para retornar ao sensor.

Figura 9 – Sensor ultrassônico

Fonte: https://www.filipeflop.com/produto/sensor-de-distancia-ultrassonico-hc-sr04/

2.4.1.7) Resistor

O resistor serve para limitar o fluxo de energia em um circuito elétrico.

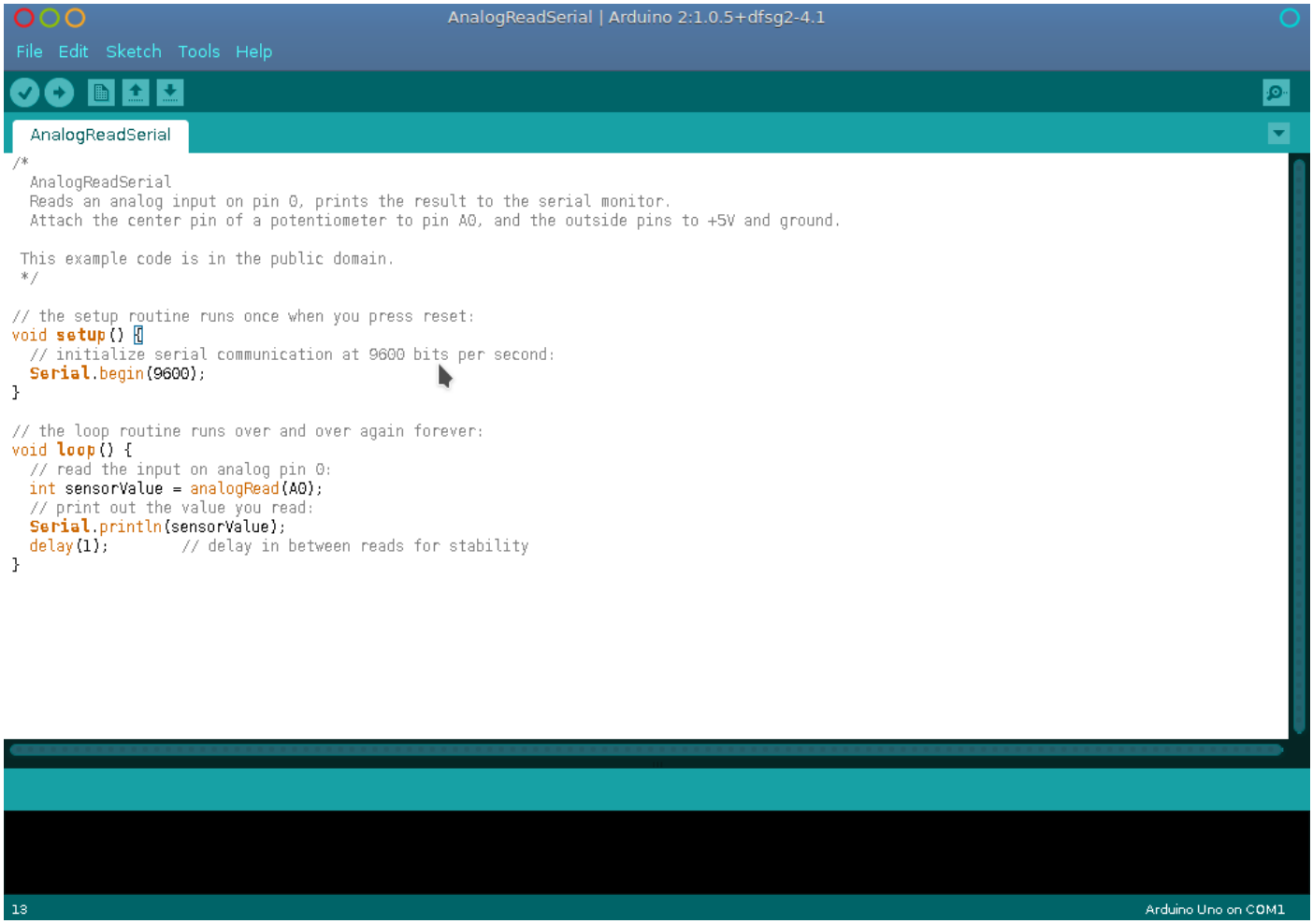
Figura 10 – Resistor

Fonte: http://www.milinfohobby.com.br/resistor-de-carbono-3,3m-ohm-3m3-1-4w-5-cr25-p5081

2.4.2) A plataforma de desenvolvimento do Arduino

Para desenvolvimento das funcionalidades do arduino utilizaremos a IDE do arduino que pode ser feito o download em https://www.arduino.cc/en/Main/Software, essa IDE nos permite que nosso código (série de comandos) seja passado para o arduino. A IDE do arduino é mostrada abaixo:

Figura 12 – A IDE do Arduino



Fonte: Própria (2018).

A linguagem de programação utilizada pelo arduino é uma mesclagem da linguagem C e da linguagem C++. Ambas as linguagens são bem usadas no ramo da eletrônica.

Figura 13 – Linguagem do Arduino

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

Fonte: Própria (2018).

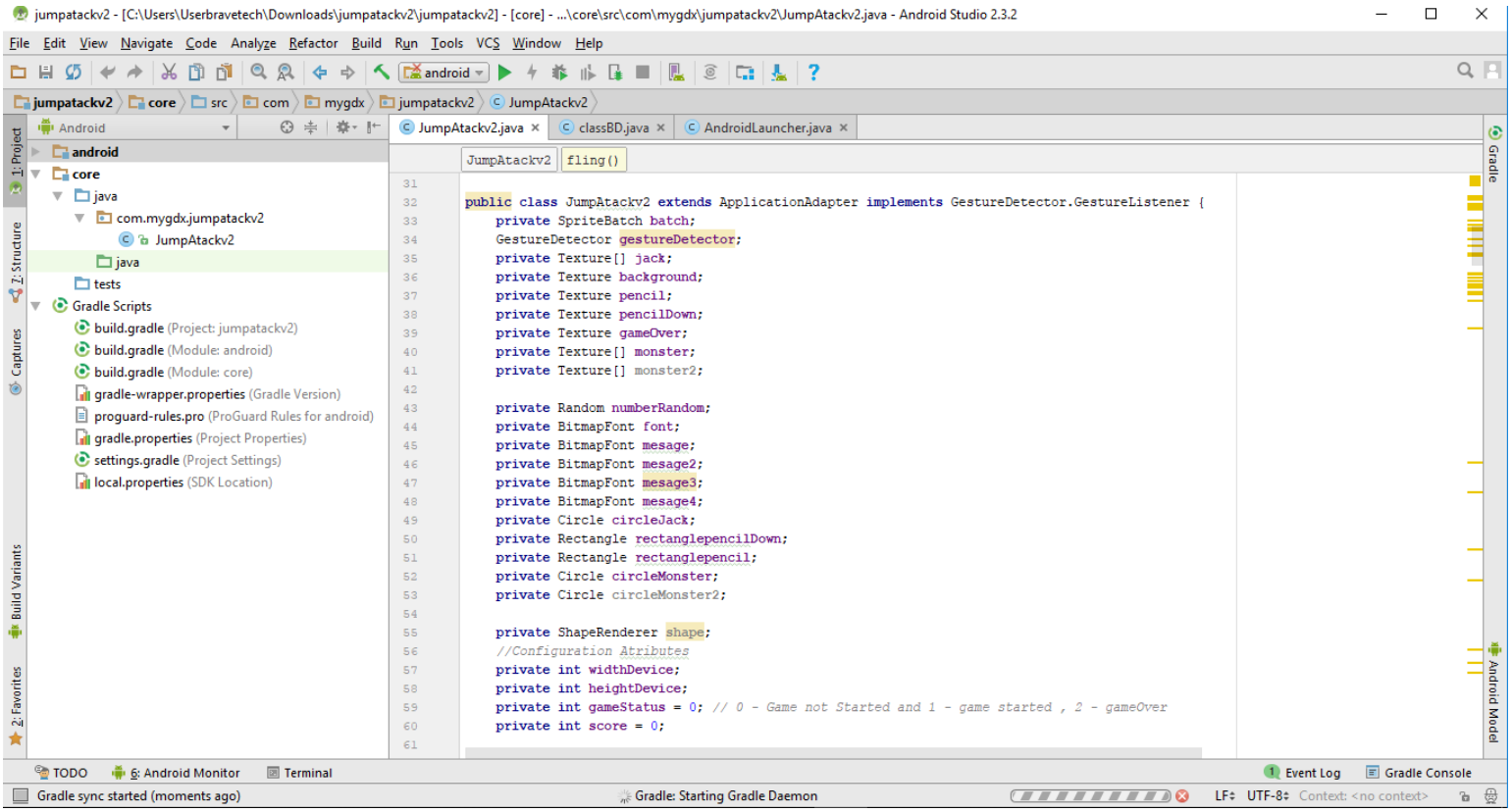
2.5) O android studio

Segundo a google developers (área especifica para desenvolvedores) para android a definição para o android studio se da por: “O Android Studio é o IDE oficial do Android. Criado especificamente para o Android, ele acelera o desenvolvimento e ajuda a criar aplicativos da mais alta qualidade para todos os dispositivos Android” (DEVELOPERS, 2018).

Eles também acrescentam que: “Ele oferece ferramentas personalizadas para desenvolvedores do Android, incluindo ferramentas avançadas para edição, depuração, testes e geração de perfis de código” (DEVELOPERS, 2018).

O nosso projeto será inicialmente desenvolvido para android utilizando o android studio, futuramente poderá ser implantado para IOS e desktops também, segue abaixo uma imagem da IDE do android studio.

Figura 14 – Android Studio



Fonte: Própria (2018).

2.6) O diagrama de sequência

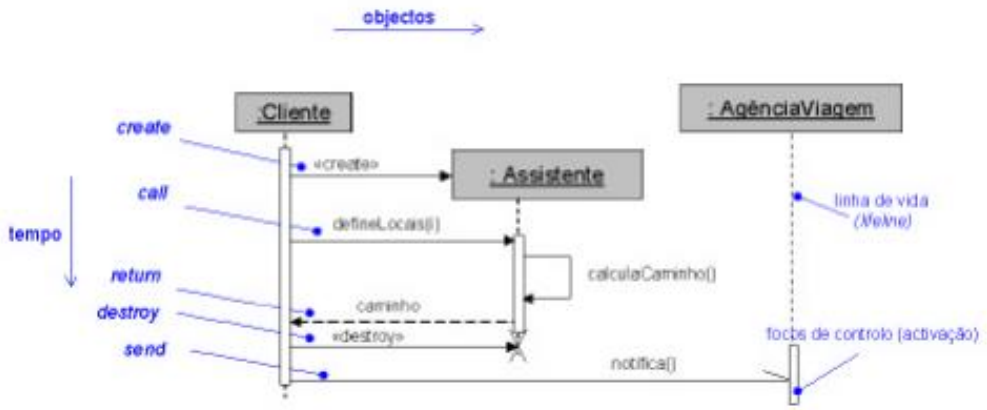
Um diagrama de sequência conforme explicado por Alberto Manuel Rodrigues da Silva é:

> Um diagrama de sequência ilustra uma interacção segundo uma visão temporal. Um diagrama de sequência é representado através de duas dimensões: a dimensão horizontal, que representa o conjunto de objectos intervenientes; e a dimensão vertical que representa o tempo. A apresentação destas dimensões pode ser invertida, se for conveniente. Não existe qualquer significado na ordenação horizontal dos objectos intervenientes, ou seja, na sua disposição relativa (SILVA, 2001, p.204).

Basicamente ele explica a ordem da sequência do software, ou seja, a sequência que ele exercerá a partir da ação que lhe foi requisitada pelo usuário. Esse tipo de diagrama define uma linha cronológica para o software também facilitando para o entendimento do cliente.

Abaixo um exemplo de um diagrama de sequência retirado do livro do Alberto Manuel Rodrigues

Figura 15 – Exemplo de um diagrama de sequência



Fonte: Silva (2001).

No diagrama da figura 15 podemos ver uma ordem cronológica do sistema que se inicia com um objeto cliente e em seguida criando um objeto assistente.

Logo depois, o software faz uma *call* (Chamada a um método) que chama o método defineLocais(), lembre-se que a definição do que é um método e um objeto já foi dada, assim como a definição de retorno.

O ciclo do software continua, até chamar um método que faz uma notificação e a partir disso não podemos ver a continuação do diagrama.

2.7) O diagrama MER

O diagrama MER visa facilitar a criação do banco de dados, fazendo com que os erros sejam cometidos antes de serem codificados, isso ajuda para que o software tenha uma boa estrutura do banco de dados para que diminua as chances de erros futuros

Segundo Joel Rodrigues do Dev Media o modelo de entidade e relacionamento (MER) se dá por:

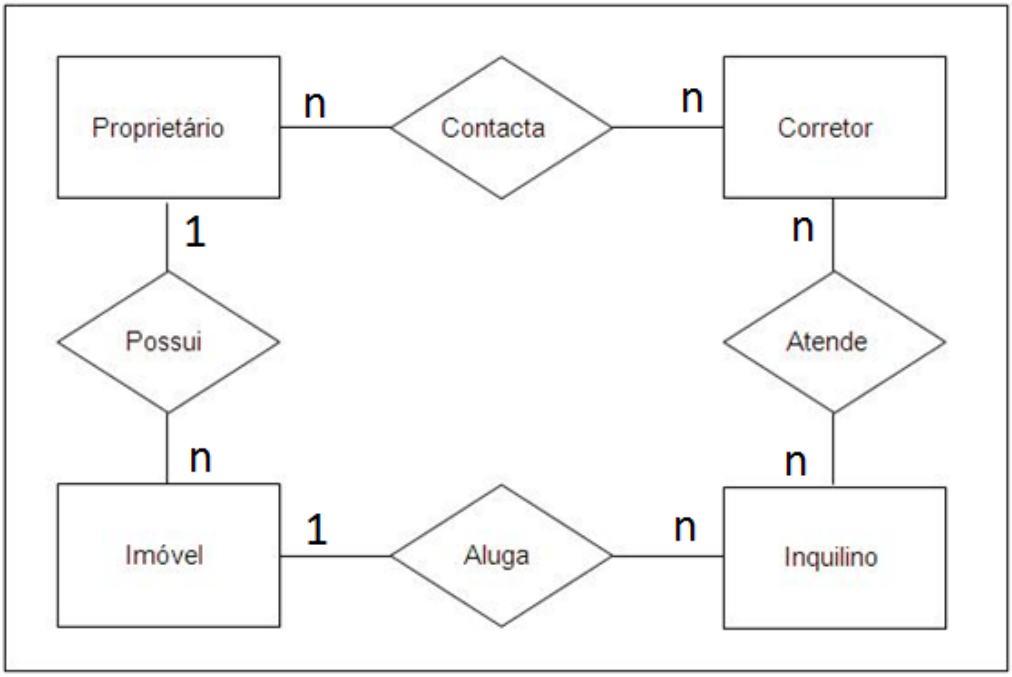
> O Modelo Entidade Relacionamento (também chamado Modelo ER, ou simplesmente MER), como o nome sugere, é um modelo conceitual utilizado na Engenharia de Software para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos) (RODRIGUES, 2014).

Ele também acrescenta que:

> Em geral, este modelo representa de forma abstrata a estrutura que possuirá o banco de dados da aplicação. Obviamente, o banco de dados poderá conter várias outras entidades, tais como chaves e tabelas intermediárias, que podem só fazer sentido no contexto de bases de dados relacionais (RODRIGUES, 2014).

Segue abaixo uma imagem de exemplo de um MER assim como a explicação de seus itens.

Figura 16 – Exemplo diagrama MER



Fonte: https://arquivo.devmedia.com.br/artigos/Joel_Rodrigues/mer/image001.png

O modelo entidade relacionamento mostrado na figura 16 é simples, porém contém coisas que precisamos saber como entidades, relacionamentos, atributos etc.

2.7.1) Entidades

As entidades são objetos ou partes envolvidas em um domínio, podendo ser classificados físicos ou lógicos. Joel Rodrigues define da seguinte forma:

> Os objetos ou partes envolvidas um domínio, também chamados de entidades, podem ser classificados como físicos ou lógicos, de acordo sua existência no mundo real. Entidades físicas: são aquelas realmente tangíveis, existentes e visíveis no mundo real, como um cliente (uma pessoa, uma empresa) ou um produto (um carro, um computador, uma roupa). Já as entidades lógicas são aquelas que existem geralmente em decorrência da interação entre ou com entidades físicas, que fazem sentido dentro de um certo domínio de negócios, mas que no mundo externo/real não são objetos físicos (que ocupam lugar no espaço). São exemplos disso uma venda ou uma classificação de um objeto (modelo, espécie, função de um usuário do sistema) (RODRIGUES, 2014).

Simplificando, cada quadrado no diagrama será uma entidade que conterá atributos e ela também pode ser forte, fraca ou associativa.

Entidade fortes: São entidades que não dependem de outras entidades, você verá mais adiante que tem entidades que contém partes de outras entidades.

Entidade fracas: São entidades que dependem de outras entidades para existir, por exemplo uma entidade de venda, precisaria de um produto para que ela exista.

Entidade associativa: Esse tipo de entidade serve para fazer a junção de duas entidades, de certo modo, simplificando, essa entidade serve para quando o relacionamento não é nem de uma entidade nem de outra, mas são das duas juntas você adicionará esse tipo de relacionamento.

2.7.2) Relacionamentos

No modelo entidade relacionamento como o nome dá a entender existem relacionamentos, relacionamentos servem para identificar qual a ligação de uma entidade com outra, isso ajuda a definir qual a ligação de uma tabela com outra no desenvolvimento do banco de dados.

Os tipos de relacionamento são 1..1 (um para um), 1..n ou 1..*(um para muitos) e n..n ou *..* (muitos para muitos), esses já citados serão definidos abaixo:

Relacionamento 1..1: Esse tipo de relacionamento define que uma entidade referência obrigatoriamente somente uma de outra entidade. Um exemplo é o cpf, 1 cpf deve ser somente referenciado a 1 pessoa. Ou seja, não pode mais de uma pessoa ter o mesmo CPF ou o mesmo CPF ter várias pessoas.

Relacionamento 1..n: Esse tipo de relacionamento define que uma entidade referência muitas unidades de outra entidade e muitas unidades de outra entidade referenciam apenas uma unidade de outra. Um exemplo é uma turma de escola de ensino médio, que não mude de sala de aula, uma sala de aula pode conter muitos alunos, porém, muitos alunos poderão conter somente uma sala de aula.

Relacionamento n..n: Nesse tipo de relacionamento muitas entidades podem referenciar várias unidades de outras e vice-versa. Por exemplo, disciplina e aluno, muitos alunos podem ter muitas disciplinas e muitas disciplinas podem ter muitos alunos.

2.7.3) Atributos

Os atributos é quase o mesmo sentido dos atributos de classe como já explicado, eles podem ser classificados como atributos descritivos, nominativos e referenciais.

Descritivos: Uma boa definição também feita por Joel Rodrigues é que os atributos descritivos: “representam característica intrínsecas de uma entidade, tais como nome ou cor.” (RODRIGUES, 2014).

Nominativos: Também são descritivos, porém eles deixam claro a identificação e definição de um objeto como Nome, código etc.

Referenciais: Demonstra a ligação de uma entidade com outra a partir do relacionamento delas. Eles também contêm estruturas que podem ser definidas como estruturas simples e compostas.

Simples: Quando somente uma informação é o suficiente como por exemplo peso, nome, cpf.

Composto: Quando precisa de mais de uma informação, como por exemplo o endereço, o endereço necessita de bairro, número, cidade, rua, estado. É de extrema importância informar que há o atributo que definimos como **primary key** (chave primaria), esse atributo não pode em hipótese alguma ser repetido, pois ele é o identificador para recuperar algum tipo de informação específica.

2.8) A linguagem java

Como já foi mencionado o projeto será feito no android-studio e é utilizado a linguagem java para desenvolvimento. Com o java podemos criar uma diversidade de softwares tanto para desktops, quanto para celulares e outros equipamentos.

É comum achar que java e javascript são a mesma coisa ou que simplesmente uma se estende de outra, mas não, segundo o site oficial do java:

"O Java é uma tecnologia usada para desenvolver aplicações que tornam a Web mais divertida e útil. O Java não é a mesma coisa que o javascript, que é uma tecnologia simples usada para criar páginas Web e só é executado no seu browser."

O java também é próprio para POO (programação orientada a objetos), como já foi especificado no diagrama de classe, o java trabalha usando objetos, classes, atributos etc.

Figura 17 – A linguagem Java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package foodfast;

/**
 *
 * @author Anderson
 */
public class Foodfast {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Principal principal = new Principal();
        principal.setVisible(true);
    }

}
```

Fonte: Própria (2018).

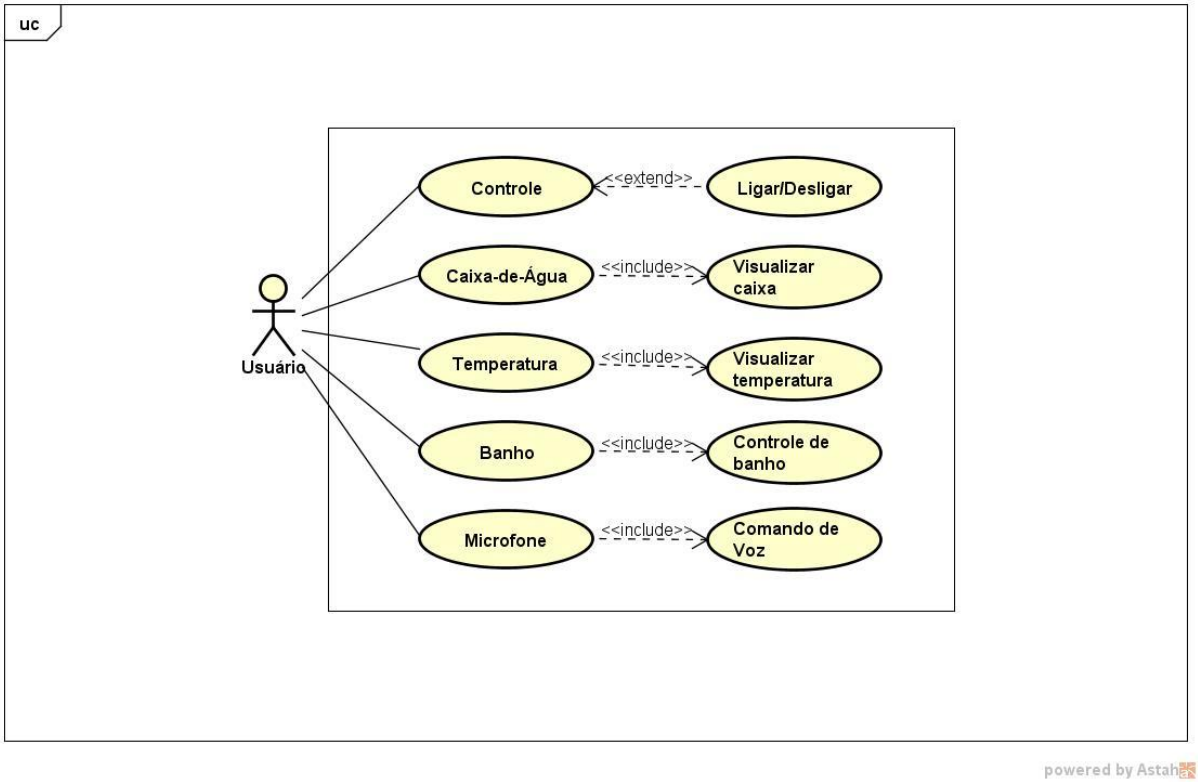
3) O desenvolvimento do projeto

Esse capítulo irá abordar o desenvolvimento do aplicativo e software do Arduino, para que seja entendido é extremamente necessário o leitor ter visto a fundamentação teórica para darmos uma base para isso. O aplicativo desenvolvido tem o objetivo inicial em tornar algumas tarefas manuais em automáticas, também ajudando com a economia de água e energia.

3.1) Diagrama de caso de uso

O diagrama de caso de uso tem uma visão do usuário das opções e modo de agir que ele tem disponível dentro do sistema, como já explicado na fundamentação teórica.

Figura 18 - Diagrama de caso de uso



Fonte: Elaborado pelo autor

Uma breve explicação do diagrama de caso de uso é vista a seguir:

Tabela 3 – Caso de controle

Nome	Controle
Descrição	O usuário tem a opção de por acionamento de botões apagar ou ligar luz e/ou ventilador.

Fonte: Elaborado pelo autor

Tabela 4 – Caso da caixa d'água

Nome	Caixa d'água
Descrição	O usuário poderá ver o nível em que o recipiente de água ou caixa d'água se encontra.

Fonte: Elaborado pelo autor

Tabela 5 – Caso da temperatura

Nome	Temperatura
Descrição	O usuário pode checar a temperatura local em que se encontra a sua residência.

Fonte: Elaborado pelo autor

Tabela 6 – Caso do banho

Nome	Banho
Descrição	O usuário poderá checar o tempo gasto no banho e o quanto de valor aquilo consome, também pode checar a quantidade gasta.

Fonte: Elaborado pelo autor

Tabela 7 – Caso do microfone

Nome	Microfone
Descrição	O usuário pode através de comandos de voz acionar comandos como ligar luz, consultar horário, ligar ventilador.

Fonte: Elaborado pelo autor

3.2) Requisitos funcionais e não funcionais

Tabela de requisitos funcionais

Tabela 8 – Requisitos funcionais do android

Identificação	Requisito Funcional	Categoria	Prioridade
RF001	O aplicativo deve ativar o bluetooth	Programação	Muito Alta
RF002	O aplicativo deve tentar se conectar a todo momento com o bluetooth caso ocorra uma falha	Programação	Muito alta
RF003	O aplicativo deve solicitar que selecione o dispositivo bluetooth que será conectado	Programação	Muito alta
RF004	O aplicativo deve armazenar depois da primeira conexão as configurações da mesma.	Programação	Alta
RF005	O aplicativo deve informar os comandos possíveis caso um inválido seja informado	Programação	Alta

Fonte: Elaborado pelo autor

Tabela 9 – Requisitos funcionais do arduino

Identificação	Requisito Funcional	Categoria	Prioridade
RF001	O software deve verificar o tipo de requisição feita	Programação	Muito alta
RF002	O software deve receber os dados vindo do android	Programação	Muito alta
RF003	O software deve enviar comandos para o android quando necessário	Programação	Alta
RF004	O Arduino tem que ser alimentado com energia	Hardware	Muito alta

Fonte: Elaborado pelo autor

Tabela de requisitos não funcionais

Tabela 10 – Requisitos não funcionais do android

Identificação	Requisito não funcional	Categoria	Prioridade
RNF001	A versão do android precisa ser a partir da 6.0	Software	Muito alta
RNF002	O dispositivo necessita ter reconhecimento de voz	Software	Alta
RNF003	O dispositivo deve possuir bluetooth	Hardware	Muito alta
RNF003	Permissões de bluetooth e reconhecimento de voz devem ser concedidas	Software	Muito alta

Fonte: Elaborado pelo autor

Tabela 11 – Requisitos não funcionais do arduino

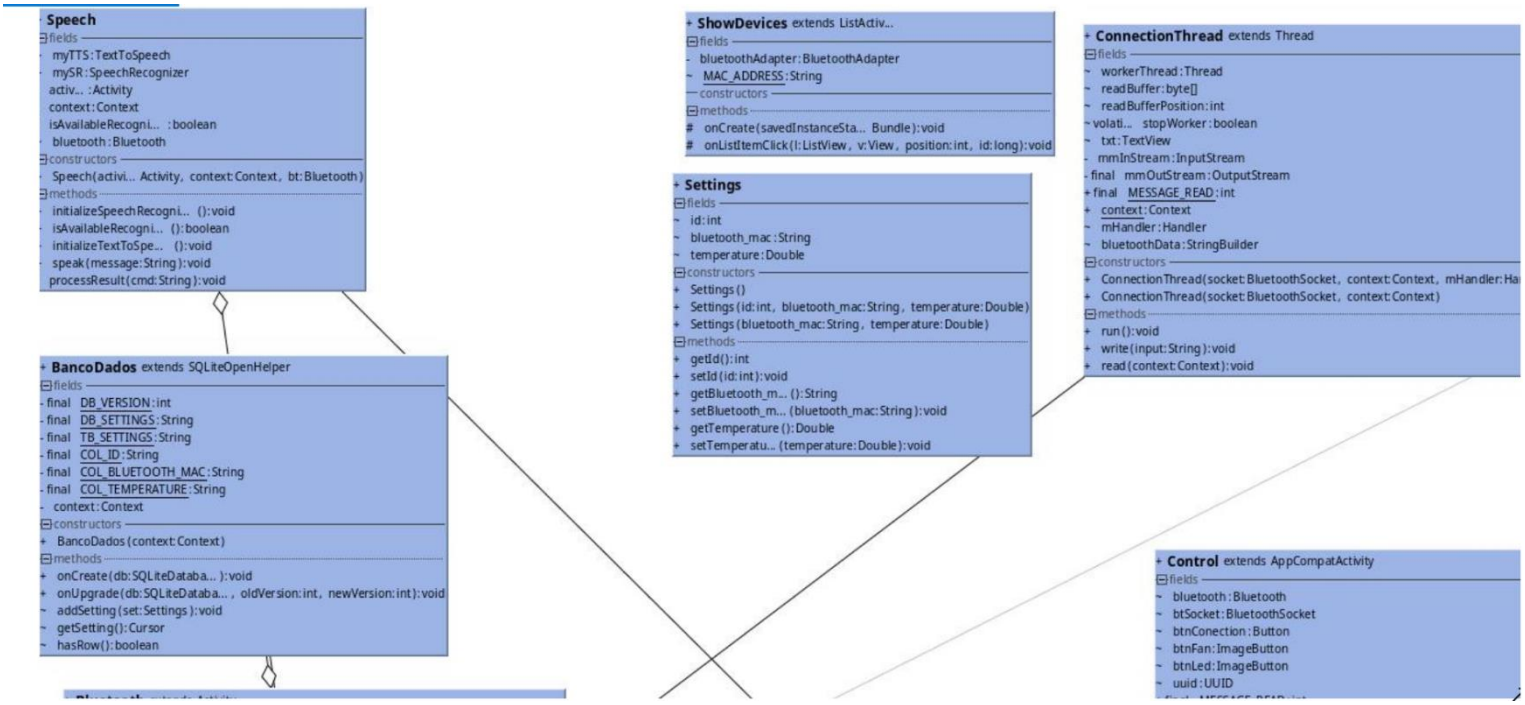
Identificação	Requisito não funcional	Categoria	Prioridade
RNF001	Somente um dispositivo poderá se conectar	Hardware	Muito alta
RNF002	O software deve ser mantido no Arduino e não pode ser substituído	Programação	Muito alta

Fonte: Elaborado pelo autor

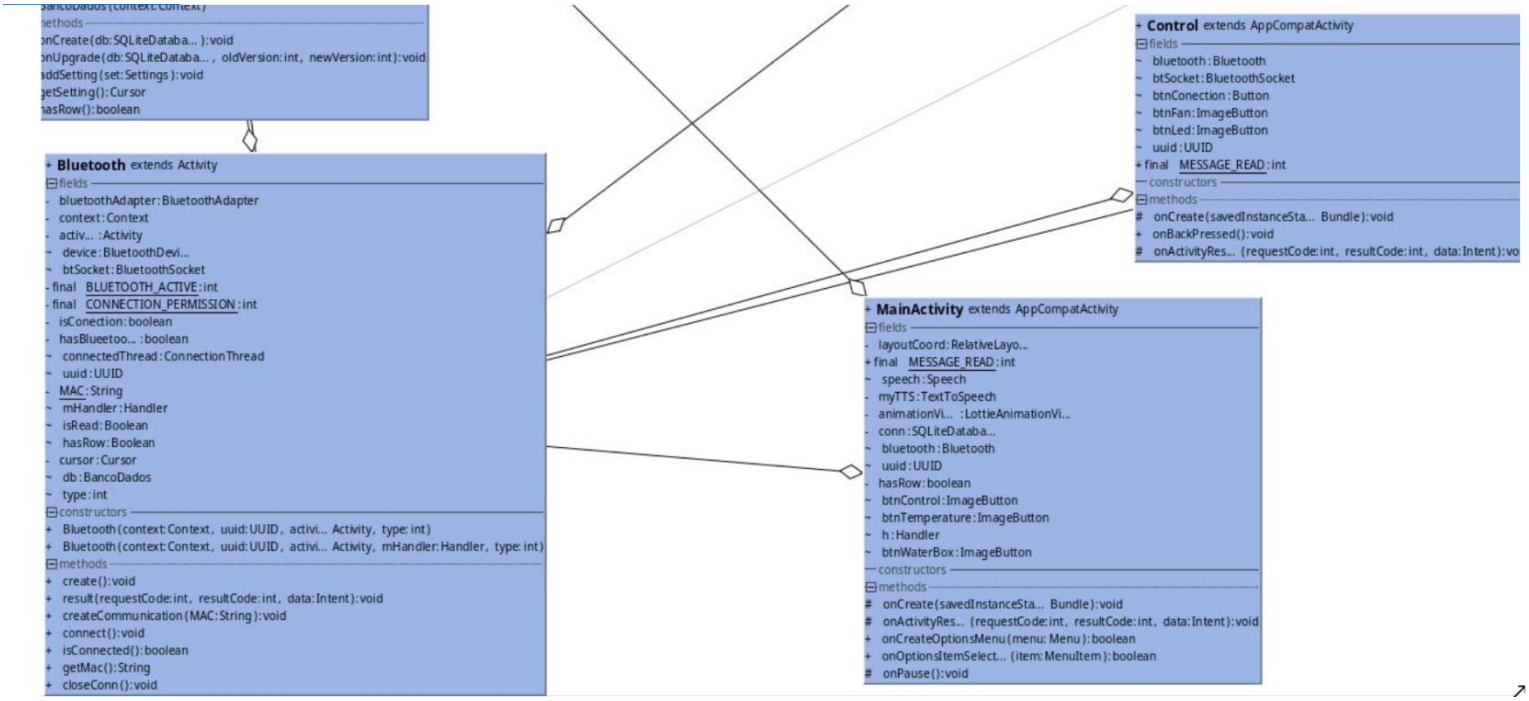
3.3) Diagrama de classe

O diagrama de classe especifica as classes que o software contém, assim como seus tributos métodos e encapsulamentos. Esse assunto já foi abordado na fundamentação teórica caso necessário saber o que é cada coisa.

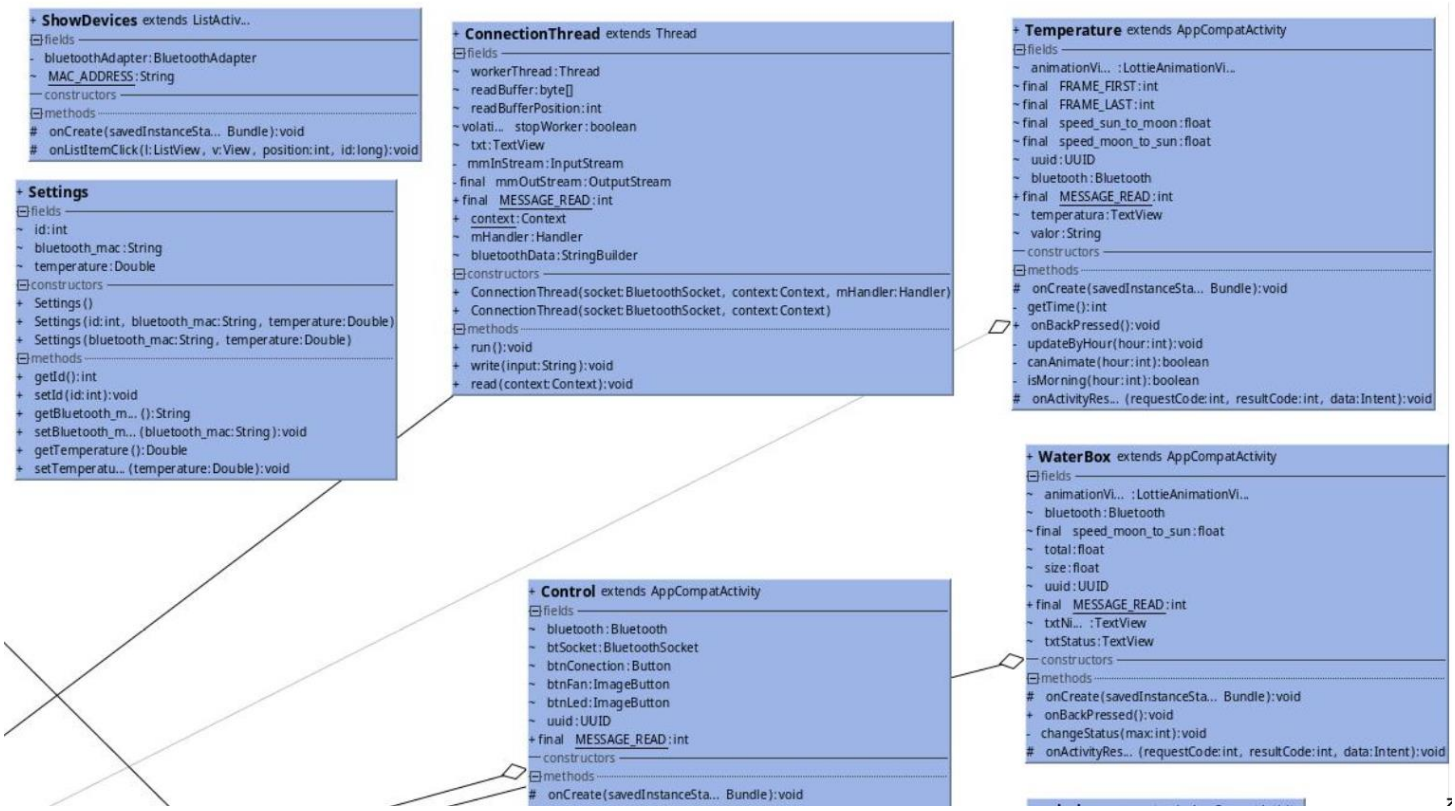
Figura 19 – Diagrama de classe



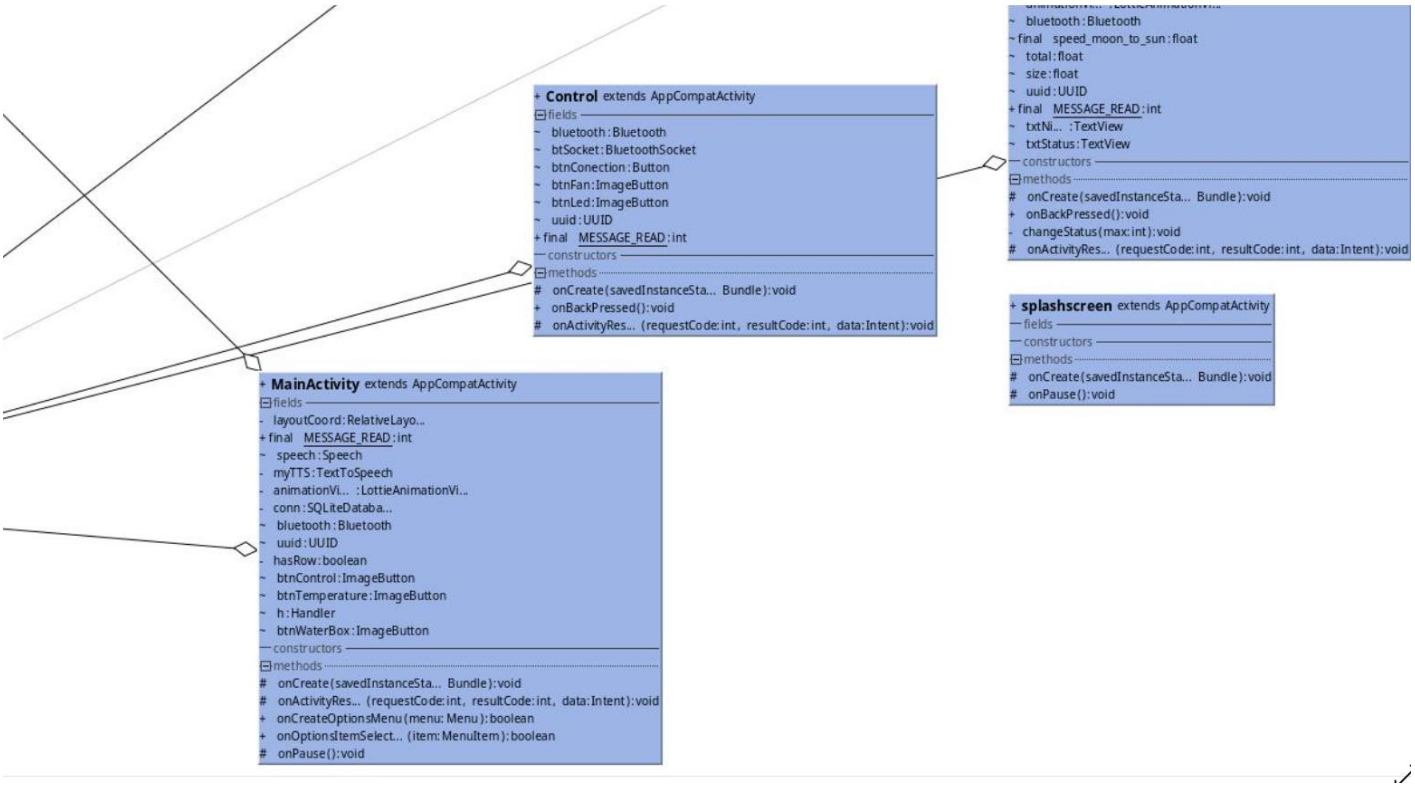
Fonte: Elaborado pelo autor



Fonte: Elaborado pelo autor



Fonte: Elaborado pelo autor



Fonte: Elaborado pelo autor

Aqui se encontra as classes utilizadas no aplicativo do android, não foi criada classe no Arduino, portanto o diagrama só aborda o aplicativo do android, temos as classes Bluetooth, MainActivity, Control, Splashscreen, BancoDados, Settings, WaterBox, Speech, ShowDevices, ConnectionThread, Temperature e Shower. Essas serão abordadas detalhadamente a seguir.

Tabela 12 – Classe principal

Nome da classe	MainActivity
Resumo	É a classe principal que gerencia a tela principal, ela é responsável por dar o start do programa assim como as demais funcionalidades a partir de instancias.
Método	**Descrição**
onCreate	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
onActivityResult	Responsável por receber algum resultado de uma intente ou algo do tipo e tratar esse resultado da forma desejada. Essa nesse caso é utilizada para bluetooth, para saber se o mesmo está ativado, se o dispositivo tem bluetooth e se está conectado.
onCreateOptionsMenu	Serve para modificação do menu
onOptionsItemSelected	Pega a opção selecionada.

Tabela 13 – Classe de controle

Nome da classe	Control
Resumo	Responsável por gerenciar a atividade de controle, essa é a tela onde o usuário pode ativar e desativar luz e ventilador pelo botão (sem comando de voz).
Método	**Descrição**
onCreate	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
onBackPressed	Ativada quando o botão de voltar é pressionado.
onActivityResult	Responsável por receber algum resultado de uma intente ou algo do tipo e tratar esse resultado da forma desejada. Essa nesse caso é utilizada para bluetooth, para saber se o mesmo está ativado, se o dispositivo tem bluetooth e se está conectado.

Tabela 14 – Classe de temperatura

Nome da classe	Temperature
Resumo	Responsável por gerenciar as atividades de temperatura, essa é a tela onde o usuário vê a temperatura atual (local).
Método	**Descrição**
onCreate	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
getTime	Pega a hora atual para determinar se é dia ou se é noite.
onBackPressed	Ativada quando o botão de voltar é pressionado.
updateByHour	Realiza a animação trocando de sol para lua ou vice-versa
canAnimate	Verifica se pode ser realizado a animação
isMorning	Verifica se é dia ou se é noite utilizando a hora retornada do método getTime
onActivityResult	Responsável por receber algum resultado de uma intente ou algo do tipo e tratar esse resultado da forma desejada. Essa nesse caso é utilizada para bluetooth, para saber se o mesmo está ativado, se o dispositivo tem bluetooth e se está conectado.

Tabela 15 – Classe da caixa d'água

Nome da classe	WaterBox
Resumo	Responsável por gerenciar as atividades da caixa d'água, essa é a tela onde o usuário vê o nível de água atual.
Método	**Descrição**
onCreate	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
onBackPressed	Ativada quando o botão de voltar é pressionado.
changeStatus	Muda o status da caixa d'água, como o nível e a animação.
onActivityResult	Responsável por receber algum resultado de uma intente ou algo do tipo e tratar esse resultado da forma desejada. Essa nesse caso é utilizada para bluetooth, para saber se o mesmo está ativado, se o dispositivo tem bluetooth e se está conectado.

Tabela 16 – Classe do banho

Nome da classe	Shower
Resumo	Responsável por gerenciar as atividades da tela de banho, essa é a tela onde o usuário vê o tempo gasto no banho assim como o valor gasto tanto de água quanto de eletricidade(caso desejado).
Método	**Descrição**
onCreate	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
getTime	Pega a hora atual para determinar se é dia ou se é noite.
onBackPressed	Ativada quando o botão de voltar é pressionado.

Tabela 17 – Classe do bluetooth

Nome da classe	Bluetooth
Resumo	Responsável por criar a comunicação bluetooth e gerenciar a mesma.
Método	**Descrição**
create	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
createCommunication	Cria a comunicação entre o bluetooth do android e o HC06 do arduino
Connect	Conecta ao dispositivo bluetooth.
isConnected	Retorna se está conectado ou não
getMac	Pega o mac address do dispositivo bluetooth em que vai se conectar.
result	Responsável por receber algum resultado de uma intente ou algo do tipo e tratar esse resultado da forma desejada. Essa nesse caso é utilizada para bluetooth, para saber se o mesmo está ativado, se o dispositivo tem bluetooth e se está conectado.

Tabela 18 – Classe de banco de dados

Nome da classe	BancoDados
Resumo	Responsável por gerenciar a comunicação do aplicativo com o banco de dados
Método	**Descrição**
onCreate	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
onUpgrade	Responsável por fazer o upgrade caso seja feita uma nova versão
addSetting	Salva as configurações no banco de dados
getSetting	Pega as configurações criadas
hasRow	Retorna se algo foi retornado do banco de dados

Tabela 19 – Classe de mostrar dispositivos

Nome da classe	ShowDevices
Resumo	Responsável por abrir uma lista com os dispositivos bluetooth disponíveis no ambiente.
Método	**Descrição**
onCreate	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
onListItemClick	Ativada quando selecionado o dispositivo

Tabela 20 – Classe do reconhecimento de voz

Nome da classe	Speech
Resumo	Responsável por realizar o reconhecimento de voz e também por converter texto para voz.
Método	**Descrição**
initializeSpeechRecognizer	Inicia o reconhecimento de voz
isAvailableRecognizer	Verifica se o dispositivo tem o reconhecimento de voz.
initializeTextToSpeech	Inicia a função de transformar texto em voz
speak	Reproduz em voz o texto passado
processResult	Varias configurações do reconhecimento de voz.

Tabela 21 – Classe da thread de conexão com arduino

Nome da classe	ConnectionThread
Resumo	Responsável por gerenciar a thread que monitora a leitura e escrita do Arduino.
Método	**Descrição**
Run	Método que fica sempre sendo executado quando a thread é iniciada.
Write	Envia dados para o Arduino
Read	Lê os dados do Arduino

Tabela 22 – Classe das configurações

Nome da classe	Settings
Resumo	Responsável por gerenciar os gets (pegar) e sets (adicionar) da classe de configuração.
Método	**Descrição**
getId	Pega o id da configuração no banco de dados
setId	Adiciona um id para uma determinada configuração.
getBluetooth_mac	Pega o mac address(Endereço) do dispositivo bluetooth.
setBluetooth_mac	Adiciona um mac address para um dispositivo bluetooth no banco de dados.
getTemperature	Pega a temperatura que o usuário considera quente
setTemperature	Adiciona a temperatura que o usuário considera quente

Tabela 23 – Classe da tela inicial (Não a principal)

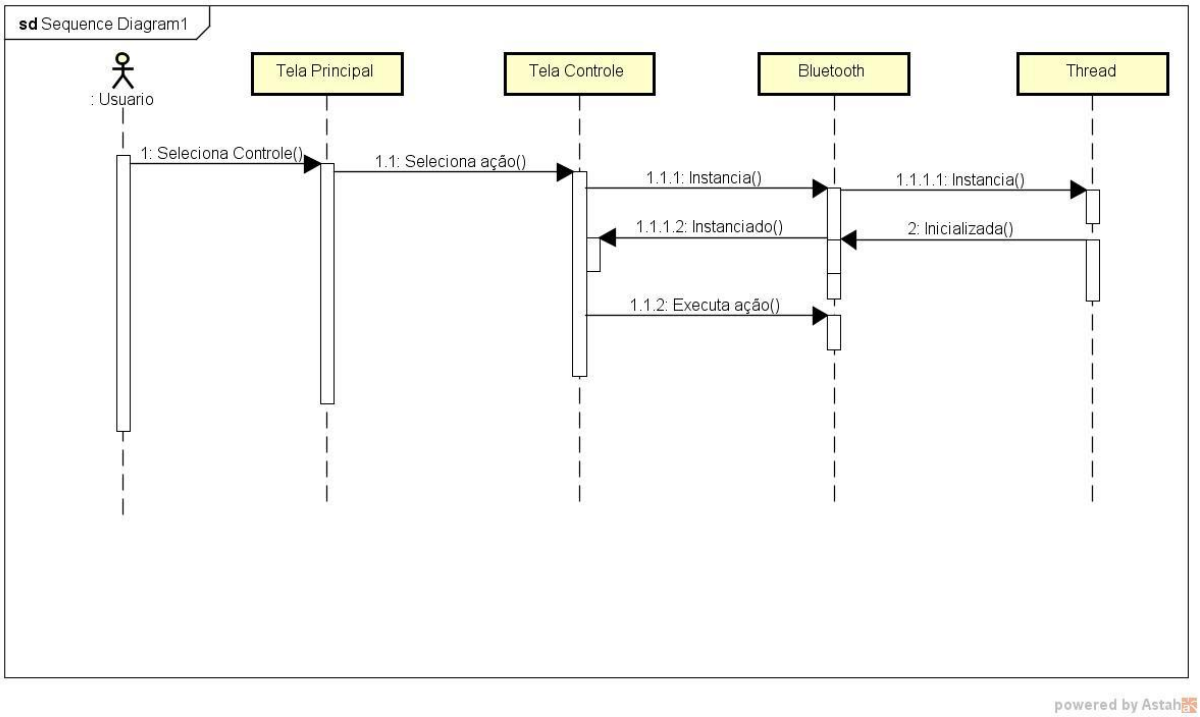
Nome da classe	Splashscreen
Resumo	Responsável por mostrar a tela inicial com o logo do aplicativo e depois mudar para a tela principal.
Método	**Descrição**
onCreate	Responsável por iniciar as funcionalidades assim que essa classe é instanciada.
onPause	Ativada quando é pausado a atividade.

Nome da classe	Consumo
Resumo	Responsável por pegar e adicionar os dados de consumo.
Método	**Descrição**
getId	Retorna o id do determinado consumo
setId	Adiciona o id do consumo
getTipo	Retorna o tipo do determinado consumo, 1 para água, 2 para energia
setTipo	Adiciona o tipo para o determinado consumo
getData	Retorna a data do determinado consumo
setData	Adiciona a data do determinado consumo
getGastoTotal	Retorna o gasto do determinado consumo
setGastoTotal	Adiciona o gasto do determinado consumo.

3.4) Diagrama de sequência

O diagrama de sequência mostra as etapas de como e o que será realizado no decorrer da execução do software, para mais detalhes leia mais sobre, na fundamentação teórica.

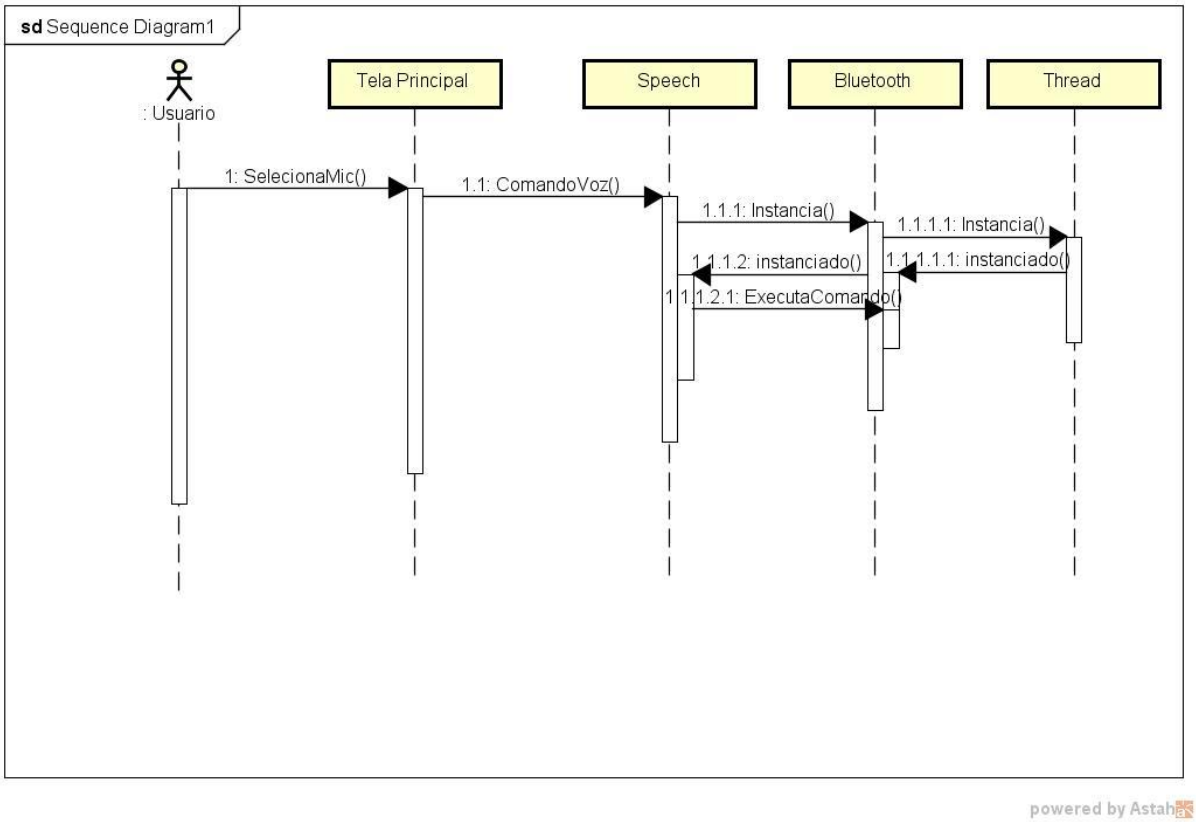
Figura 20 – Diagrama de sequência para função de controle



Fonte: Elaborado pelo autor

O usuário na tela principal seleciona o botão de controle e com isso é levado a tela de controle, na tela de controle ele tem dois botões com a opção de poder acionar ou desligar luz e/ou ventilador, para isso quando clicado é instanciado a classe de bluetooth e a classe de bluetooth inicia uma classe com uma *thread.*

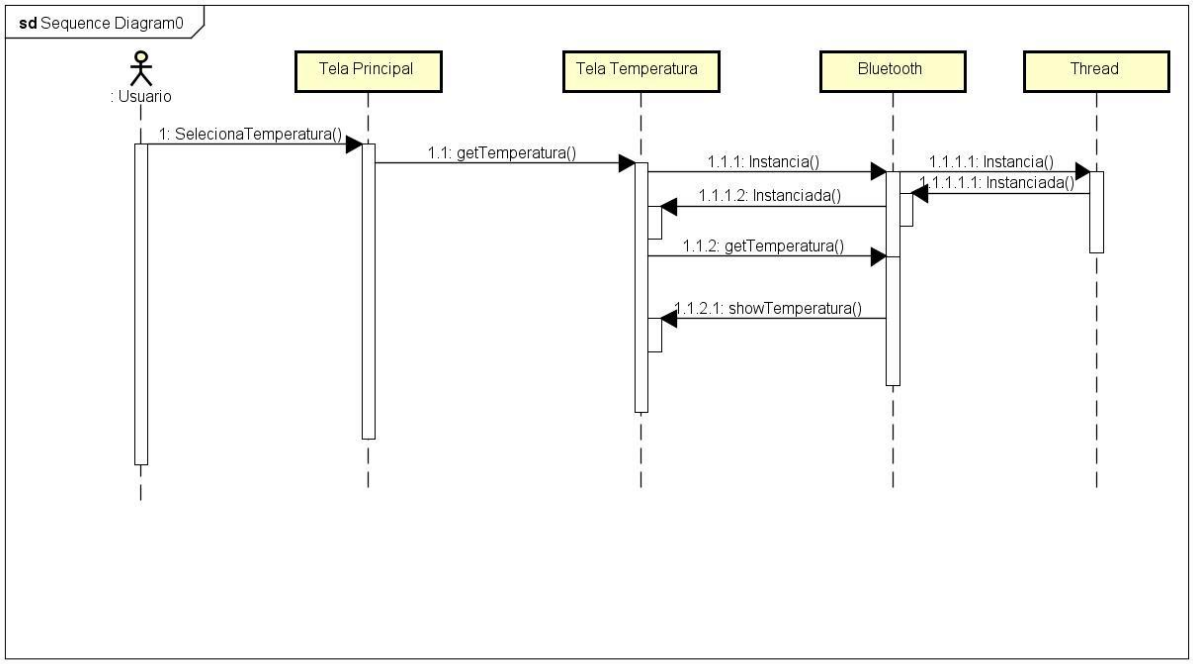
Figura 21 – Diagrama de sequência para a função de microfone



Fonte: Elaborado pelo autor

O usuário na tela principal aperta o botão de microfone, logo após isso é chamado o *speech recognizer*(reconhecimento de voz), isso ouve o que o usuário fala, transforma para texto e chama a classe de bluetooth, a mesma como na etapa passada instancia uma classe de thread e com isso escreve para o Arduino.

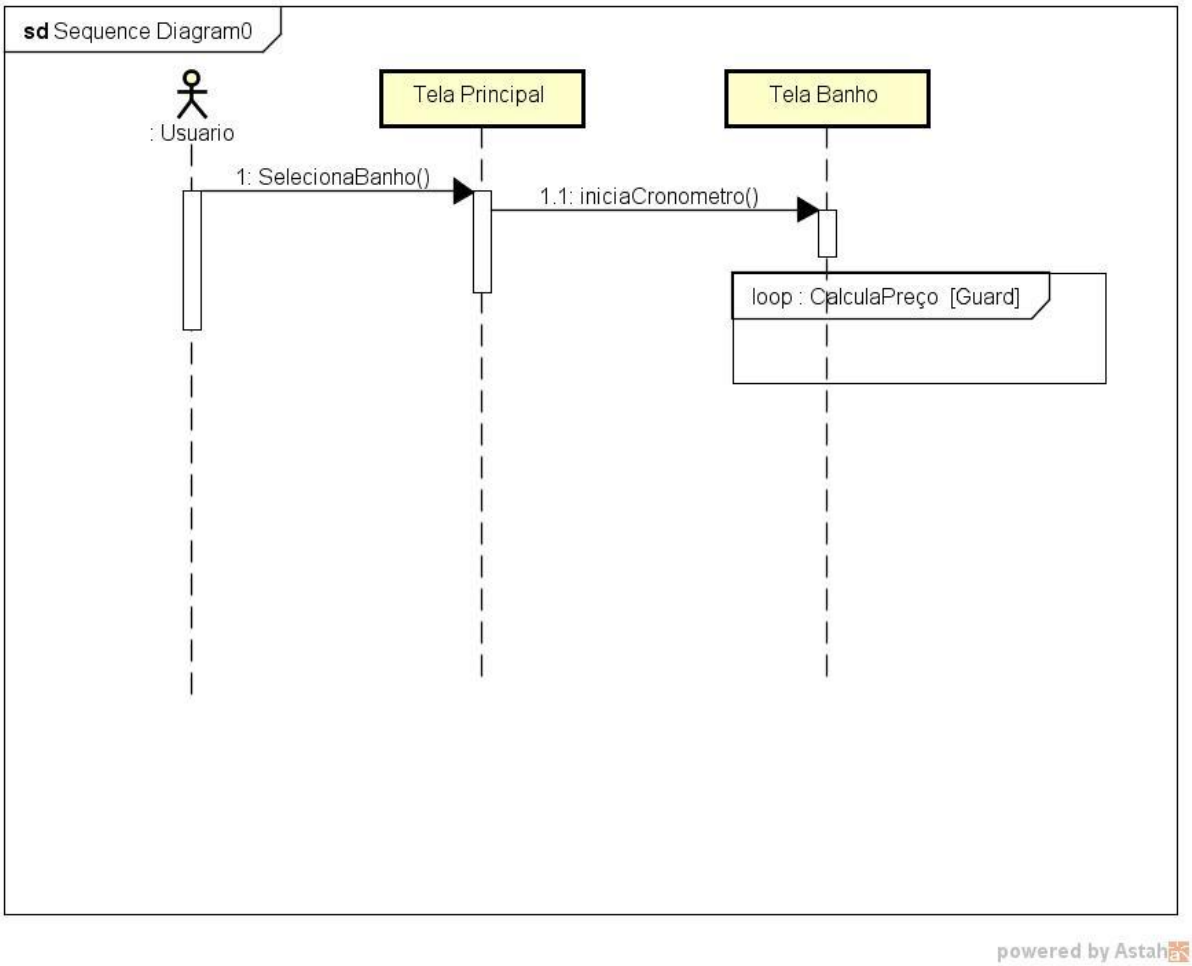
Figura 22 – Diagrama de sequência para a temperatura



Fonte: Elaborado pelo autor

O usuário seleciona o botão de temperatura, logo após isso a tela é modificada para a tela de temperatura, que inicia o bluetooth e assim como nos outros processos chama uma *thread* essa *thread* lê o que vem do Arduino e retorna ao bluetooth que manda para a tela de temperatura.

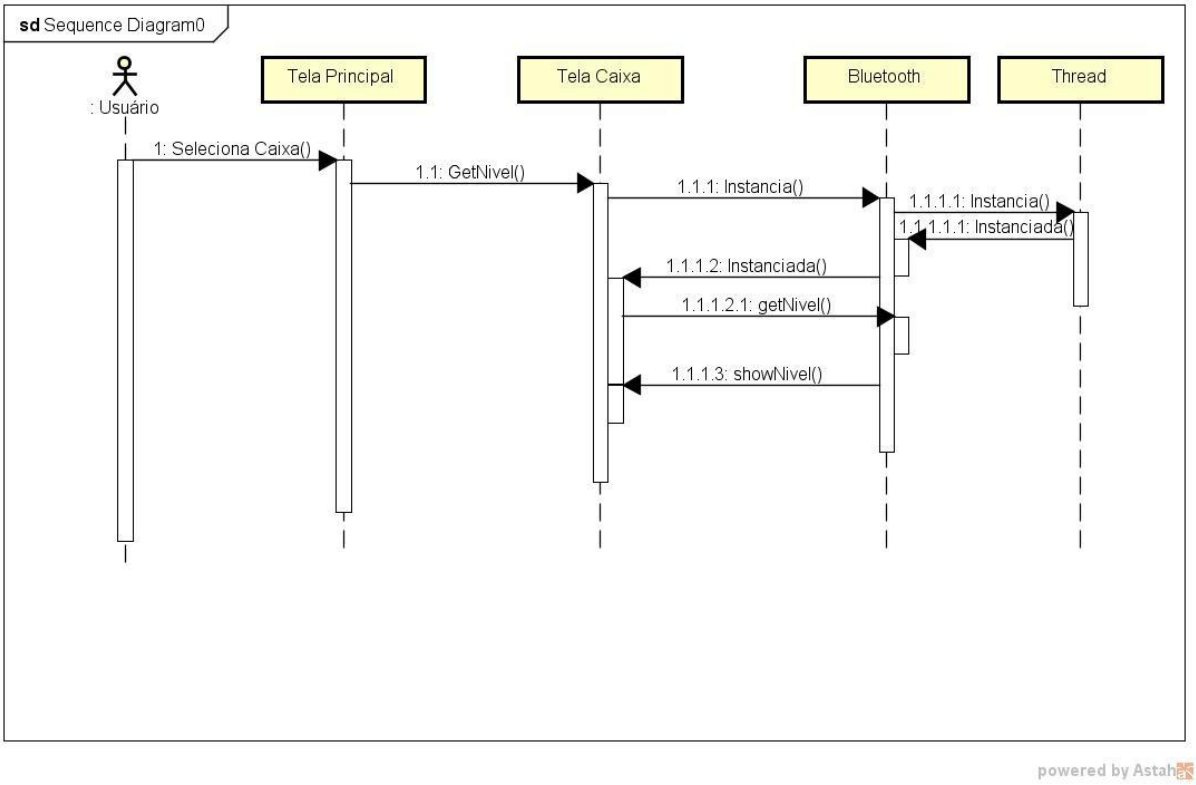
Figura 23 – Diagrama de sequência para a função de banho



Fonte: Elaborado pelo autor

O usuário seleciona a tela de banho e é movido para a mesma, logo após isso ele é recebido com dois botões de iniciar e parar, quando clicado em iniciar, o cronometro começa a contar e calcular o preço, caso a opção de chuveiro seja acionada ele também calcula o preço gasto no chuveiro. Também é mostrado a quantidade de litros consumida.

Figura 24 – Diagrama de sequência para a função da caixa d'água



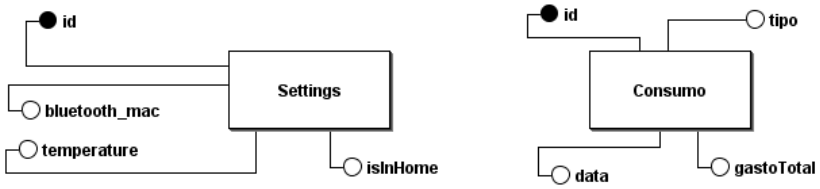
Fonte: Elaborado pelo autor

O usuário seleciona o botão de caixa d'água e logo em seguida é redirecionado para a tela de caixa d'água, a mesma instancia a classe de bluetooth que inicia a thread para ler o nível da caixa. Isso é retornado para a tela de caixa mostrando ao usuário.

3.5) DER do banco de dados do aplicativo

Segue a estrutura de banco de dados do aplicativo, as tabelas por não ter um usuário ou algo para *linkagem* não tem relacionamento entre elas, para mais informação consulte o modelo DER na fundamentação teórica.

Figura 25 – Diagrama DER do banco de dados da aplicação android



Fonte: Elaborado pelo autor

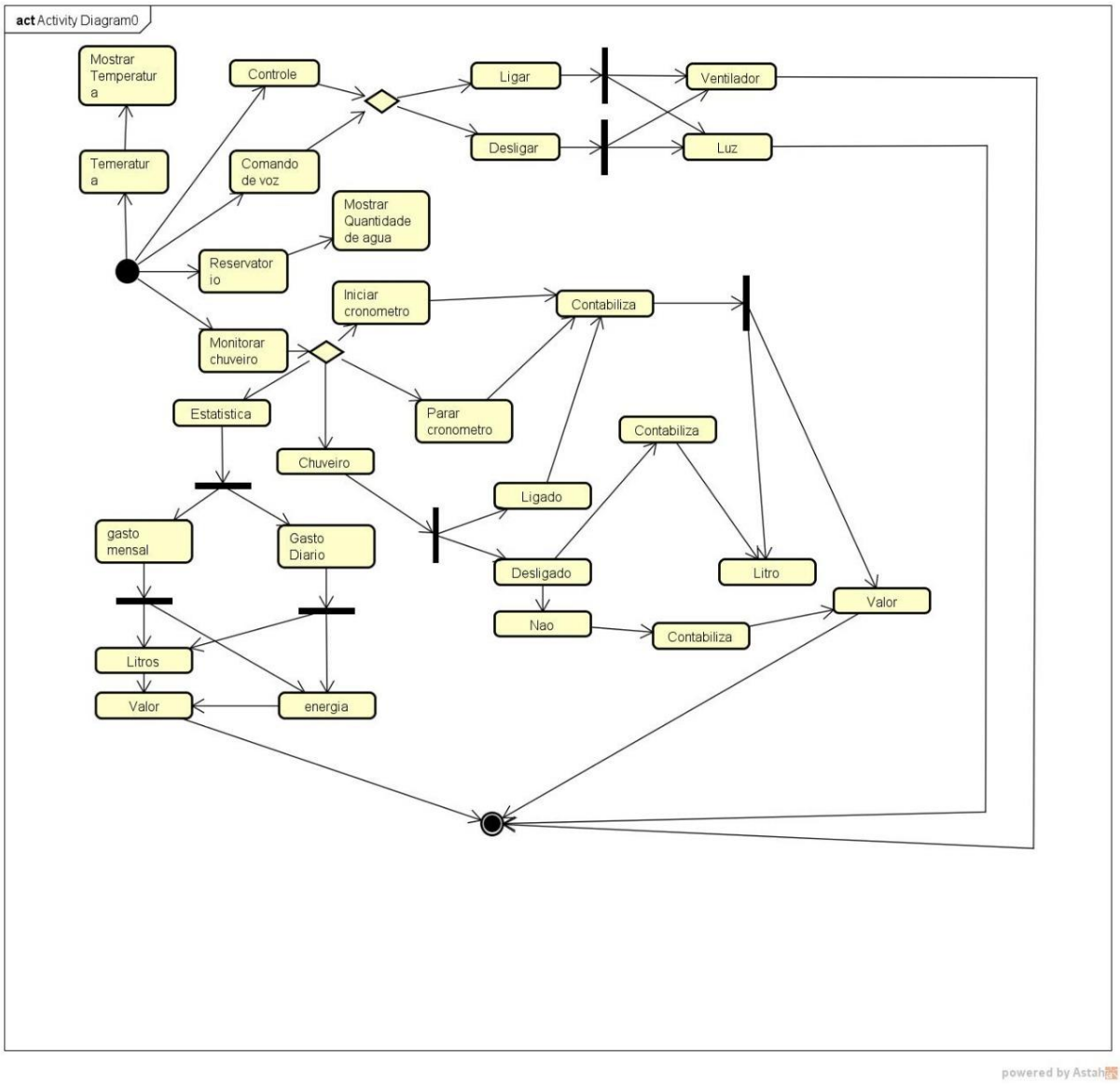
No diagrama temos a tabela de configuração (Settings) nessa é armazenado o *mac address* (endereço de identificação) do dispositivo bluetooth, em temperature é armazenado a temperatura padrão que o usuário considera muito calor, isso é usado para ligar o ventilador sozinho quando chega nessa determinada temperatura, o isInHome serve para saber se o usuário está saindo de casa, pois caso ele esteja fora de casa o ventilador não liga automaticamente.

A tabela de consumo (retângulo da figura 25), contém o tipo de consumo que foi feito podendo ser de água ou energia para o banho, o gasto total armazena o total gasto no dia e a data armazena a data atual para podermos contabilizar no mês quanto foi gasto.

3.6) Diagrama de atividade

O diagrama de atividade representa um fluxo de processamento, ou seja os processos que serão executados no decorrer do aplicativo. Abaixo segue o diagrama de atividades do nosso software.

Figura 26 – Diagrama de atividade



Fonte: Elaborado pelo autor

Na funcionalidade de controle o usuário pode escolher entre acender/apagar uma luz ou ventilador, em comandos de voz o usuário pode acionar e desativar luz e ventilador assim como outros comandos como hora, utilizando a voz, em temperatura o usuário será redirecionado para uma tela onde mostra a temperatura

local, caso clicado em reservatório (no menu principal, ícone de água) será mostrado o nível da água do recipiente e em monitorar chuveiro o usuário pode checar quantos está gastando tanto de dinheiro quanto de litros de água no chuveiro assim como monitorar o tempo de banho.

3.7) A codificação

O processo de codificação foi feito em duas linguagens de programação, foi utilizado para o desenvolvimento mobile o java e para o Arduino C/C++. Dentre a codificação que está disponível no github (checar apêndice) para quem quiser utilizar. Partes que merecem destaques serão mencionadas.

3.7.1) Codificação do Arduino

A codificação do Arduino foi feita com casos de testes a partir de comandos recebidos, esses comandos são concatenados em uma variável e checado qual foi recebido, a partir disso, o módulo HC-06 (Figura 6) envia comando para o android para executar uma determinada ação, caso o arduino receba “led1” ele acende a luz e caso receba “fan” acende o ventilador. Os tipos “type” são definidos a partir da tela utilizada no android, eg. Tela de controle aciona o tipo 1.

Segue um trecho do código logo abaixo, referente ao Arduino:

Figura 27 – Código de recebimento de comando

```
while (bluetooth.available()) {
  char caracter = bluetooth.read();
  command += caracter;
  delay(10);
}

if(command.indexOf("temperature") >= 0){
  type = 1;
}

if(command.indexOf("cmds") >= 0){
  type = 0;
}

if(command.indexOf("caixa") >= 0){
  type = 2;
}
```

Fonte: Elaborado pelo autor

Figura 28 – Código de testes de comandos recebidos

```
if(type == 0){
  if (command.indexOf("led1") >= 0) {
    digitalWrite(pinLed, !digitalRead(pinLed));
  }
  if (command.indexOf("fan") >= 0) {
    digitalWrite(fanPin, !digitalRead(fanPin));
  }
   if(command.indexOf("alloff") >= 0){
    digitalWrite(pinLed, HIGH);
    digitalWrite(fanPin, LOW);
   }
   if(command.indexOf("allon") >= 0){
      digitalWrite(pinLed, LOW);
      digitalWrite(fanPin, HIGH);
   }
}
if(type == 1){
  bluetooth.println("{");
  bluetooth.print(temperatura);
  bluetooth.println(" ºC");
  bluetooth.println("}");
}
if(type == 2){
  bluetooth.println("{");
  bluetooth.print(sonar.ping_cm());
  bluetooth.println(" %");
  bluetooth.println("}");
  Serial.println(sonar.ping_cm());
 }
```

Fonte: Elaborado pelo autor

3.7.2) A codificação android

O desenvolvimento do android foi mais árduo, por o Arduino ser uma programação em *loop* (fica se repetindo), era necessário uma thread para receber valor do Arduino constantemente em telas de leitura.

Também é necessário criar a comunicação bluetooth entre o android e o HC-06 (figura 6). Para isso era necessário consultar os dispositivos disponíveis e então adquirir o seu *mac address* para que fosse possível a conexão, também foi

necessário checar se o dispositivo tinha bluetooth e se o mesmo estava ativado, caso não estivesse teria que ser solicitado a ativação.

Segue abaixo imagens de códigos da conexão bluetooth utilizado

Figura 29 – Código da criação da classe bluetooth

```
public void create(){
    bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    db = new BancoDados(context);
    //db.getWritableDatabase();
    //db.addSetting(new Settings("FC:A8:9A:00:20:BA", 30.00));
    hasRow = db.hasRowSetting();

    //db.addSetting(new Settings("FC:A8:9A:00:20:BA", 30.00));
    if(bluetoothAdapter == null){
        Toast.makeText(context, text: "Seu dispositivo não possui bluetooth", Toast.LENGTH_LONG).show();
    }else if(!bluetoothAdapter.isEnabled()){
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        activity.startActivityForResult(enableBtIntent, BLUETOOTH_ACTIVE);
        hasBlueetooth = true;
    }
    else{
        cursor = db.getSetting();
        MAC = cursor.getString( columnIndex: 1);
        createCommunication(MAC);
    }
}
```

Fonte: Elaborado pelo autor

Figura 30 – Código de verificação de ativação

```
public void result(int requestCode, int resultCode, Intent data){

    switch(requestCode){

        case BLUETOOTH_ACTIVE:
            if(resultCode == Activity.RESULT_OK){
                Toast.makeText(context, text: "O bluetooth foi ativado", Toast.LENGTH_LONG).show();
                if(!hasRow)
                    connect();
                else{
                    cursor = db.getSetting();
                    MAC = cursor.getString( columnIndex: 1);
                    createCommunication(MAC);
                }
            }
            else {
                Toast.makeText(context, text: "O bluetooth não foi ativado, o app será encerrado", Toast.LENGTH_LONG)
                        .show();
                activity.finish();
            }
            break;
        case CONNECTION_PERMISSION:
            if(resultCode == activity.RESULT_OK){
                MAC = data.getExtras().getString(ShowDevices.MAC_ADDRESS);
                db.addSetting(new Settings(MAC, temperature: 30.00, isInHome: true));
                createCommunication(MAC);

            }else{
                Toast.makeText(context, text: "Falha ao obter o endereço MAC", Toast.LENGTH_LONG).show();
            }

    }
}
```

Fonte: Elaborado pelo autor

Figura 31 – Código da comunicação / conexão com modulo

```
public void createCommunication(String MAC){
    try{
        //btSocket.close();
        device = bluetoothAdapter.getRemoteDevice(MAC);
        //Create a communication socket (channel)
        btSocket = device.createRfcommSocketToServiceRecord(uuid);

        btSocket.connect();
        isConection = true;

        if(isRead) {
            //Instancia da thread, que receberá leitura
            connectedThread = new ConnectionThread(btSocket, context, mHandler);
            connectedThread.start();
        }
        else {
            //Instancia da thread
            connectedThread = new ConnectionThread(btSocket, context);
        }

        Toast.makeText(context, text: "Conectado com: " + MAC, Toast.LENGTH_LONG).show();
        if(type == 0){
            connectedThread.write( input: "cmds");
        }
        if(type == 1){
            connectedThread.write( input: "temperature");
        }
        if(type == 2){
            connectedThread.write( input: "caixa");
        }
    }catch(IOException error){
        isConection = false;
        Toast.makeText(context, text: "Falha ao conectar com: " + MAC, Toast.LENGTH_LONG).show();
    }
}
```

Fonte: Elaborado pelo autor

3.7.2.1) Reconhecimento de voz

O reconhecimento de voz foi utilizado a *API Speech Recognizer* (Reconhecimento de voz) e a *Text to Speech* (Texto para Voz), ambas são da Google, as mesmas convertem voz para texto e texto para voz, isso foi utilizado para receber os comandos de áudio.

Figura 32 – Reconhecimento de voz e reprodução

```
        else if(cmd.indexOf("desligar ventilador") != -1) {
            speak("desligando ventilador");
            bluetooth.connectedThread.write( input: "fan");
        }
        else if(cmd.indexOf("ligar ventilador") != -1){
            speak("ligando ventilador");
            bluetooth.connectedThread.write( input: "fan");
        }
        else if(cmd.indexOf("desligar luz") != -1){
            speak("desligar luz");
            bluetooth.connectedThread.write( input: "led1");
        }
        else if(cmd.indexOf("ligar luz") != -1) {
            speak("ligando luz");
            bluetooth.connectedThread.write( input: "led1");
        }
        else if(cmd.indexOf("desligar tudo") != -1){
            speak("desligando tudo");
            bluetooth.connectedThread.write( input: "alloff");
        }
        else if(cmd.indexOf("ligar tudo") != -1){
            speak("ligando tudo");
            bluetooth.connectedThread.write( input: "allon");
        }
```

Fonte: Elaborado pelo autor

3.7.2.2) A thread

Como já mencionado uma thread foi utilizada para receber comunicação do Arduino, para isso foi criado *handlers* que servem para requisitar algo de outra classe e então retornar o resultado para a tela desejada e realizar a ação.

Os trechos de imagens abaixo servem para a classe da thread e para o recebimento do *handlers.*

Figura 33 – A codificação da thread

```
public void run() {
    byte[] buffer = new byte[1024];  // buffer store for the stream
    int bytes; // bytes returned from read()

    // Keep listening to the InputStream until an exception occurs
    while (true) {
        try {
            // Read from the InputStream
            bytes = mmInStream.read(buffer);
            String btData = new String(buffer,  offset: 0, bytes);

            // Send the obtained bytes to the UI activity
            mHandler.obtainMessage(MESSAGE_READ, bytes,  arg2: -1, btData)
                    .sendToTarget();
        } catch (IOException e) {}
    }
}
    /* Call this from the main activity to send data to the remote device */
    public void write(String input) {
        byte[] buffer = input.getBytes();

        try {
            mmOutStream.write(buffer);
        } catch (IOException e) { }
    }
```

Fonte: Elaborado pelo autor

Figura 34 – A codificação do handler

```
final Handler mHandler = handleMessage(msg) → {
        if(msg.what == MESSAGE_READ){
            String receiveds = (String) msg.obj;

            int infoEnd = receiveds.indexOf("}");
            if(infoEnd > 0){
                String compData = receiveds.substring(0, infoEnd);
                int infoSize = compData.length();

                if(receiveds.charAt(0) == '{'){
                    String finalData = receiveds.substring(1, infoSize);
                    if(finalData.contains("C")){
                        temperatura.setText(finalData.replaceAll( regex: "(\\r|\\n)",  replacement: ""));
                    }
                    Log.d( tag: "CHEGANDO", finalData);
                }
                Log.d( tag: "CHEGANDO  2", receiveds);
                //btData.delete(0, btData.length());
            }

        }
};
```

Fonte: Elaborado pelo autor

3.8) A aplicação android

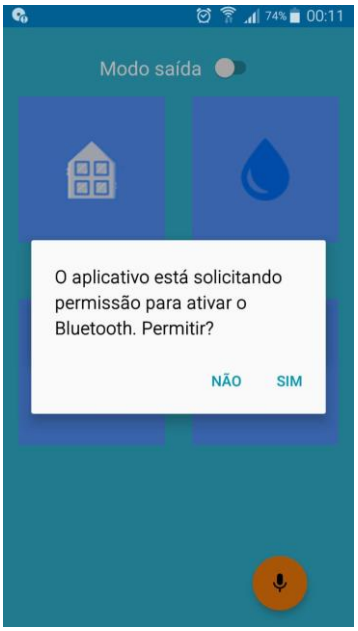
Nessa sessão será abordada as telas assim como suas funcionalidades da aplicação android. O aplicativo desenvolvido foi batizado com o nome SmartHouse em português Casa Inteligente, dando enfoque nas funcionalidades presente no aplicativo. O logo foi pensado nos benefícios em que o aplicativo trás, juntando a economia de energia com a economia de água. Assim que o aplicativo e iniciado, a tela mostra o logo, nome e versão, após iniciado, a tela principal do aplicativo é revelada. A tela de principal é responsável pelos botões de cada funcionalidade, sendo eles, o botão de controle, reservatório, temperatura, monitoramento do banho e comando de voz. Abaixo é demostrado a tela inicial e logo a seguir a principal.

Figura 35 - Tela inicial do aplicativo

Fonte: Elaborado pelo autor

Essa tela aparece assim que inicia o aplicativo, mostrando o logo, nome, e a versão.

Figura 36 – Conexão Bluetooth



Fonte: Elaborado pelo autor

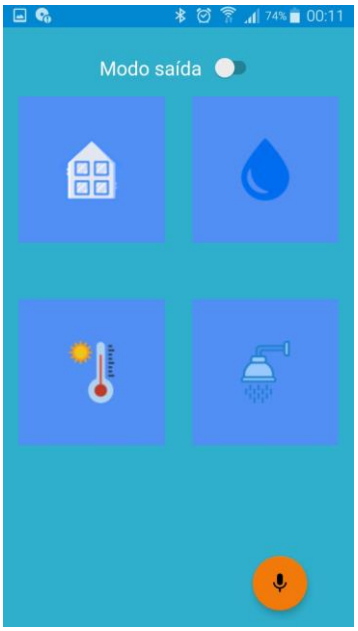
O aplicativo solicita ao usuário a permissão para ativar o Bluetooth.

Figura 37 – Conexão com o Arduino

Fonte: Elaborado pelo autor

Conectar com o módulo Bluetooth HC06.

Figura 38 – Principal



Fonte: Elaborado pelo autor

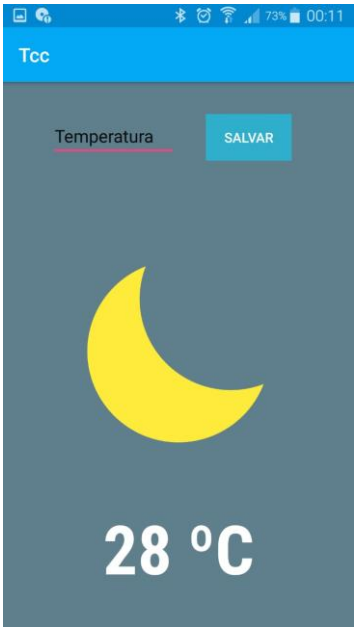
Essa tela é a que dá início a tudo, dando as opções para o usuário entre controle, nível de água, temperatura e banho.

Figura 39 – Caixa da Água

Fonte: Elaborado pelo autor

Permite a visualização da quantidade de água presente no reservatório.

Figura 40 - Temperatura



Fonte: Elaborado pelo autor

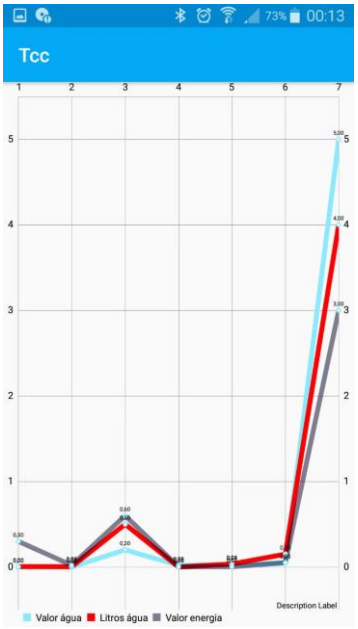
Essa tela mostra a temperatura captada pelo sensor de forma local, não consultando a internet para isso, ou seja, do ambiente local e não da cidade/estado. Na parte superior tem possibilidade de inserir a temperatura desejada e salvar no banco de dados, assim podendo ligar o ventilador de forma automática mesmo que não esteja presente na sua casa.

Figura 41 - Monitoramento de banho

Fonte: Elaborado pelo autor

Permite o usuário cronometrar seu banho, mostrando quantos litros de água foi gasto, e quantos reais e centavos gastou no banho levando em conta seu consumo de água e energia. Se o botão switch estiver desativado, não irá contabilizar o gasto de energia do chuveiro, tendo seus gastos sendo mostrado instantaneamente logo após que iniciar o cronometro. Na mesma tela está presente botão de estatísticas do consumo diário e mensal, mostrando quantos reais ou centavos gastou com água e energia, e a quantidade de água gasto em litros.

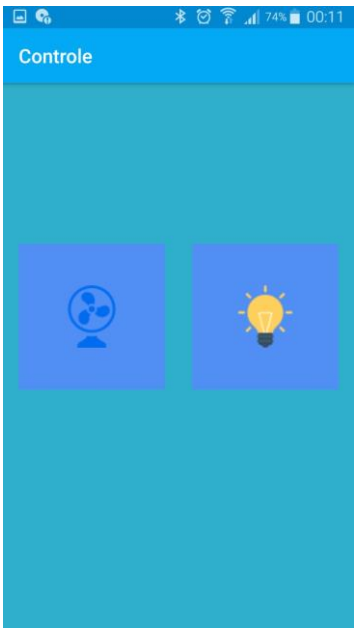
Figura 42 - Gráfico de consumo de água e energia



Fonte: Elaborado pelo autor

Essa função é acessada no botão “Gráfico” presente na tela de monitoramento de banho. Sua função é pegar os dados obtidos do monitoramento e gerar um gráfico com fácil visualização do seu consumo.

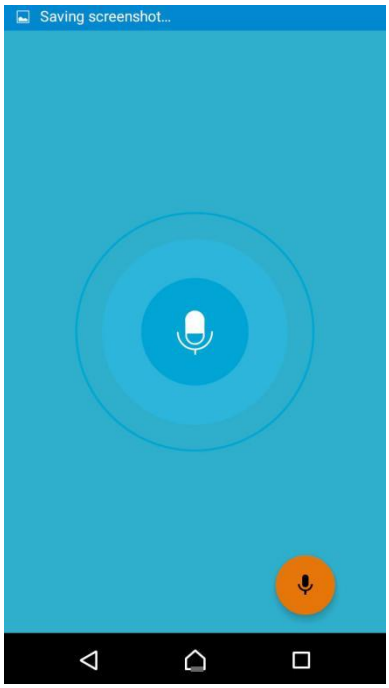
Figura 43 – Controle



Fonte: Elaborado pelo autor

Os botões da tela de controle permitem ao usuário ligar ou desligar a Luz e o Ventilador, com apenas um toque no botão com o ícone (desenho) do ventilador ou da luz.

Figura 44 - Comando de voz



Fonte: Elaborado pelo autor

O botão tem a funcionalidade de receber comandos do usuário através da voz, permitindo ligar e desligar o ventilador ou a luz, assim como perguntar a hora e comandos interativos, para isso o usuário deve falar “Ligar luz” ou “Ligar ventilador” caso o comando não seja reconhecido é mostrado para o usuário os comandos disponíveis.

3.9) O software em Arduino

Nessa sessão será abordado o software do Arduino, esse software basicamente fica esperando um comando vir do android, dependendo do comando recebido ele liga ou desliga um componente do Arduino como a luz, ventilador. Também pode enviar comandos como pegar a temperatura através do sensor dht11 e enviá-la para o android, também é possível com o sensor ultrassônico receber a distância em que a água se encontra no recipiente e envia-lo para o android.

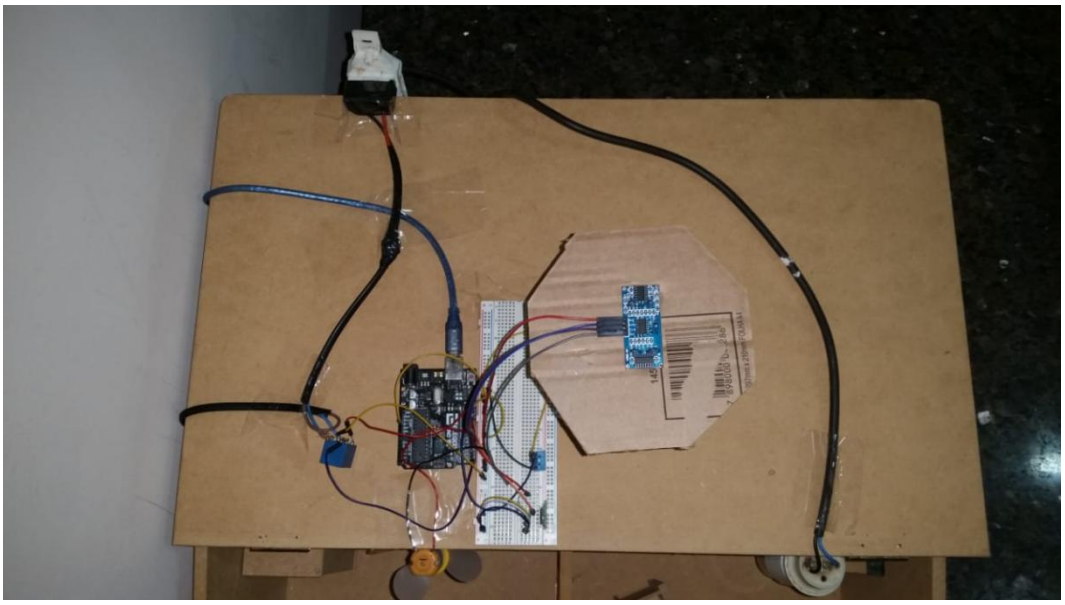
3.9.1) A maquete

Para demonstração das funcionalidades foi utilizado uma maquete de madeira, com ventilador simulado utilizando um motor, uma lâmpada led e a placa controladora Arduino, a foto da maquete é exibida a seguir.

Figura 45 – A maquete de frente

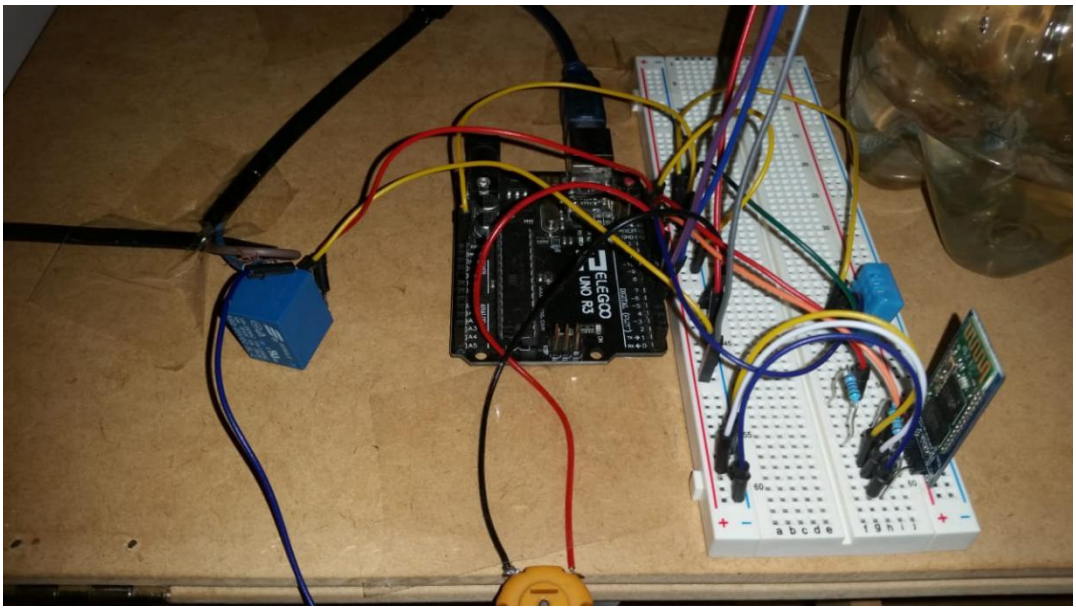
Fonte: Elaborado pelo autor

Figura 46 – Foto da maquete de cima



Fonte: Elaborado pelo autor

Figura 47 – Foto da placa controladora Arduino



Fonte: Elaborado pelo autor

4) Considerações finais

O projeto desenvolvido tem como foco trazer o entendimento e a proporção no quesito Automação Residencial para automatizar algumas tarefas comuns do dia a dia.

Logo após concluído, podemos dizer que seus objetivos foram todos atingidos, levando em consideração algumas limitações que a placa Arduino pode ter para ampliação, seu custo baixo faz com que o conjunto software e hardware tenha um ótimo custo benefício. Todas as funcionalidades propostas foram implementadas com sucesso, trazendo uma interface android fácil e amigável para qualquer usuário utilizar, facilitando o uso para idosos que costumam ter dificuldade com tecnologia em especial smartphones, e pessoas com deficiência física com funcionalidades que podem ser usadas pelo comando de voz. Não só a interface foi implementada com excelência, mas também o objetivo geral foi atingido, levando em conta a proposta inicial de sustentabilidade e economia consciente, as funções presentes no aplicativo desenvolvido atenderam todos os quesitos propostos.

Para o projeto ser concluído envolveu muito estudo e dedicação na linguagem de programação da plataforma Arduino, e consequentemente ao estudo de eletrônica, visto que a placa controladora tem que ser manipulada de forma correta para que todos os componentes e circuitos sejam ligados. A programação envolvida na aplicação android por mais que a linguagem utilizada seja java, tivemos que expandir os estudos, e aprender a utilizar a linguagem Kotlin, que é utilizado para facilitar a escrita da linguagem java.

Analisando o cenário geral, podemos concluir que o projeto tem potencial para ser expandido e melhorado, mesmo existindo algumas limitações. Adicionando novas funcionalidades como por exemplo, a comunicação do aplicativo com o arduino pode ser feita pela rede e internet, trazendo mais conforto e usabilidade.

A demonstração desse projeto torna claro a proporção da automação residencial e como ela pode ser importante para as pessoas. Todo o estudo necessário para o desenvolvimento do projeto contribuiu para nossa formação

profissional, visando que no mundo atual a automação está cada vez mais sendo utilizada por pessoas comuns e empresas.

REFERÊNCIAS

ANDROID DEVELOPERS. Recursos do Android Studio. Disponível em: <https://developer.android.com/studio/features?hl=pt-br> Acesso em: 10 maio 2018.

COSTA, Alexandre Aprato Ferreira da; LIMA, Paulo Ricardo Barbieri Dutra. automação residencial com foco no consumo consciente de energia elétrica. Artigo, Rio Grande do Sul.

BERALDI, Lairce Castanhera; FILHO, Edmundo Escrivão. Impacto da tecnologia de informação na gestão de pequenas empresas. Artigo, Brasilia, p.46, 2000. Disponivel em: <http://www.scielo.br/pdf/ci/v29n1/v29n1a5> Acesso em: 30 abr. 2018.

BLOG da Eletro energia. Como calcular o consumo de energia das lâmpadas de casa ?, 2015. Disponível em: <http://www.eletroenergia.com.br/como-calcular-o-consumo-de-energia-das-lampadas-de-casa/> Acesso em: 13 jun. 2018.

BONIATI, Bruno Batista; PREUSS, Evandro; FRANCISCATTO, Roberto. Introdução à Informática. 1° ed. Rio Grande do Sul: Colégio Agrícola de Frederico Westphalen, 2014. p. 25-44. Disponível em:

<http://estudio01.proj.ufsm.br/cadernos/cafw/tecnico_agroindustria/introducao_informatica.pdf> Acesso em: 30 abr. 2018.

CAMPOS FILHO, Maurício Prates de. Os Sistemas da Informação e as Modernas Tendências da Tecnologia e dos Negócios. Artigo, São Paulo/SP, p.36, 1994. Disponível em: <http://www.scielo.br/pdf/rae/v34n6/a05v34n6.pdf> Acesso em: 2 maio 2018.

CARVALHO, Victorio Albani de; TEIXEIRA, Giovany Frossard. Programação Orientada a Objetos. 1°.ed. Colatina/ES : Instituto Federal Espirito Santo, 2012. p.51. Disponível em: <http://jocivan.com.br/portal/wp-content/uploads/2016/04/APOSTILA_TS_DESENVOLVIMENTO_DE_SISTEMAS_Java.pdf> Acesso em: 2 maio 2018.

DARAYA, Vanessa. 9 fatos que vão fazer você economizar água a partir de agora. Exame, São Paulo, 15 maio 2014. Disponível em: <https://exame.abril.com.br/tecnologia/9-fatos-que-vao-fazer-voce-economizar-agua-a-partir-de-agora-2/>. Acesso 25 abr. 2018.

EPE, Empresa de Pesquisa Energética. Balanço energético nacional. Disponível em:

<https://ben.epe.gov.br/downloads/Relatorio_Final_BEN_2008.pdf> Acesso em: 05 de novembro 2018.

JAVA. O que é Java ?. Disponível em: <https://www.java.com/pt_BR/about/whatis_java.jsp?bucket_value=desktop-chrome66-linux&in_query=no> Acesso em: 13 maio 2018.

MICROSOFT. Diagramas de caso de uso UML: referência. Msdn, abril de 2016. Disponível em: <https://msdn.microsoft.com/pt-br/library/dd409427.aspx> Acesso em 2 maio 2018.

MICROSOFT. Diagramas de caso de uso UML: referência. Msdn, abril de 2016. Disponível em: <https://msdn.microsoft.com/pt-br/library/dd409432.aspx> Acesso em 2 maio 2018.

MURATORI, José Roberto; BÓ, Paulo Henrique Dal. Automação residencial conceitos e aplicações. 2°. ed. Belo Horizonte/MG : Educere, 2014. p. 17. Disponível em: <https://docs.google.com/viewerng/viewer?url=http://www.editoraeducere.com.br/images//degustacao/automacao_2_edicao.pdf&hl=pt-BR>

REZENDE, Denis Alcides. Tecnologia da Informação Aplicada a Sistemas de Informação Empresariais: O Papel Estratégico da Informação e dos Sistemas de Informação nas Empresas. 8°. ed. São Paulo/SP: Atlas, 2011. p.41.

RODRIGUES, Joel. Modelo Entidade Relacionamento (MER) e Diagrama de Entidade Relacionamento (DER). Disponível em: <https://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332> Acesso em: 10 maio 2018.

SOUZA, SANDRO FERREIRA DE. a contribuição da automação residencial na solução de problemas de acessibilidade no cotidiano do idoso. Artigo, Minas Gerais

SENAI. Desenho Elétrico. Disponível em: <http://www.abraman.org.br/arquivos/23/23.pdf> Acesso em: 13 maio 2018.SILVA, Alberto Manuel Rodrigues da; VIDEIRA, Carlos Alberto Escaleira. UML, Metodologias e Ferramentas Case. 1°.ed. Porto/Lisboa : Centro Atlântico, 2001. p.186 (Capítulo 6°)

SMIDT, Andre Crepaldi Geiger. Implementação de uma plataforma robótica controlada remotamente utilizando o Arduino. 2013. TG (Trabalho de Conclusão de Curso de Engenharia Elétrica e Computação), Escola de Engenharia de São Carlos, São Carlos, 2013. Disponível em: <http://www.tcc.sc.usp.br/tce/disponiveis/97/970010/tce-19112013-093717/publico/Smidt_Andre_Crepaldi_Geiger.pdf>. Acesso em: 20 abr. 2018.

SOMMERVILLE, Ian. Engenharia de Software. Selma Shin Shimizu Melnikoff; Reginaldo Arakaki; Edílson de Andrade Barbosa. 8°. ed. São Paulo/SP : Pearson, 2007. p.81

SOMMERVILLE, Ian. Engenharia de Software. Selma Shin Shimizu Melnikoff; Reginaldo Arakaki; Edílson de Andrade Barbosa. 8°. ed. São Paulo/SP : Pearson, 2007. p.82

WCED, World Commission on Environment and Development. Our Common Future. Oxford, U.K. : Oxford University Press, 1987. p. 383.