UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

SISTEMAS DE INFORMAÇÃO

MICHEL MIOLA

Desenvolvimento de componentes para App Inventor e servidor para dispositivos IoT

Florianópolis - SC

2019

MICHEL MIOLA

Desenvolvimento de componentes para App Inventor e servidor para dispositivos IoT

Trabalho de Conclusão de Curso para a obtenção do grau de Bacharelado no Curso de Sistemas de Informação na Universidade Federal de Santa Catarina.

Orientador: Jean Carlo Rossa Hauck

Florianópolis - SC

2019

MICHEL MIOLA

DESENVOLVIMENTO DE COMPONENTES PARA APP INVENTOR E SERVIDOR PARA DISPOSITIVOS IOT

Este Trabalho de Conclusão de Curso foi julgado para a obtenção do Título de "Bacharel em Sistemas de Informação", e aprovado em sua forma final pelo Curso de Bacharelado em Sistemas de Informação.

Florianópolis – Santa Catarina, julho de 2019.

Orientador:

_____

Prof. Dr. Jean Carlo Rossa Hauck
Orientador
Universidade Federal de Santa Catarina

Banca Examinadora:

_____

Prof. Dr. Ricardo Azambuja Silveira
Avaliador
Universidade Federal de Santa Catarina

_____

Prof. Dr. Cristian Koliver
Avaliador
Universidade Federal de Santa Catarina

Resumo

No cenário atual da sociedade, é necessário que o ensino da computação seja explorado já nos primórdios do ensino básico, tamanha sua importância. A introdução da computação utilizando dispositivos IoT torna o ensino mais lúdico e interessante para alunos da educação básica. Portanto, este trabalho tem por finalidade efetuar a pesquisa acadêmica e desenvolvimento de componentes na plataforma App Inventor, para comunicação com dispositivos IoT. No App Inventor, foi desenvolvida uma extensão que permite a comunicação com os dispositivos IoT por meio de um servidor, que recebe todas as requisições realizadas pelo App Inventor. Tal pesquisa tem por objetivo o campo educacional, ensinando a programação e suas estruturas, utilizando a programação em bloco junto a dispositivos IoT, para alunos de escolas do ensino básico. A avaliação realizada com o público-alvo, levanta indícios de que os componentes e o servidor desenvolvido são úteis para o aprendizado de computação e de princípios básicos de IoT.

Palavras chaves: IoT, programação, App Inventor, servidor, requisições.

Abstract

In the current scenario of society, it is necessary that the teaching of computing be explored already in the beginnings of basic education, for being so importat. The introduction of computing using IoT devices makes teaching more playful and interesting for students of basic education. Therefore, the purpose of this work is to perform the academic research and development of components in the App Inventor platform, for communication with IoT devices. In App Inventor, an extension was developed that allows communication with the IoT devices through a server, which receives all the requests made by App Inventor. This research aims at the educational field, teaching the programming and its structures, using block programming with IoT devices, for students of elementary schools. The evaluation conducted with the target audience raises indications that the components and the developed server are useful for learning computing and basic IoT principles.

Keywords: IoT, programming, App Inventor, server, requisitions.

Lista de ilustrações

Figura 1 – Eixos da Computação (SBC 2017) 17
Figura 2 – Área de "*Designer*" - App Inventor 21
Figura 3 – Área "*Block Editor*" - App Inventor 21
Figura 4 – Microserviço Flask 47
Figura 5 – Diagrama de classe componente Interruptor 49
Figura 6 – Diagrama de classe componente Temperatura 50
Figura 7 – Diagrama de classe componente Umidade 50
Figura 8 – Diagrama de classe componente Musica 51
Figura 9 – Diagrama de sequência componente Temperatura 52
Figura 10 – Aplicação no celular 54
Figura 11 – Blocos App Inventor 55
Figura 12 – Dispositivo IoT 56
Figura 13 – Arquivo de Boot da placa Raspberry Pi 57
Figura 14 – IP da placa Raspberry Pi 58
Figura 15 – Serviço REST relay 59
Figura 16 – Serviço REST temperatura 60
Figura 17 – Serviço REST umidade 61
Figura 18 – Serviço REST execução de música 62
Figura 19 – Serviço REST parar música 62
Figura 20 – Componente Interruptor 64
Figura 21 – Configuração dos parâmetros componente Interruptor 64
Figura 22 – Classe Interruptor 65
Figura 23 – Componente Temperatura 66
Figura 24 – Configuração dos parâmetros componente Temperatura 67
Figura 25 – Classe Temperatura 67
Figura 26 – Componente Umidade 68
Figura 27 – Configuração dos parâmetros componente Umidade 69
Figura 28 – Classe Umidade 70

Figura 29 – Componente Música 71
Figura 30 – Configuração do parâmetro componente Música 71
Figura 31 – Classe Musica 72
Figura 32 – Oficina Instituto Federal de Santa Catarina 75
Figura 33 – Slides oficina 76

Lista de tabelas

Tabela 1 – Sinônimos e traduções dos termos de busca 30
Tabela 2 – Expressão de busca. 31
Tabela 3 – Primeiro resultado bibliográfico 33
Tabela 4 – Segundo resultado bibliográfico 34
Tabela 5 – Terceiro resultado bibliográfico 36
Tabela 6 – Quarto resultado bibliográfico 38
Tabela 7 – Quinto resultado bibliográfico 39
Tabela 8 – Requisitos funcionais 42
Tabela 9 – Requisitos não-funcionais 43
Tabela 10 – Parâmetro do serviço relay 59
Tabela 11 – Parâmetro do serviço temperatura 60
Tabela 12 – Parâmetro do serviço umidade 61
Tabela 13 – Parâmetro do serviço play musica 62
Tabela 14 – Fatores de qualidade a serem avaliados 74
Tabela 15 – Avaliação da utilidade dos componentes IoTs 78
Tabela 16 – Avaliação da adequação funcional dos componentes IoTs 79
Tabela 17 – Avaliação da usabilidade dos componentes IoTs 80
Tabela 18 – Resultado da aplicação do questionário SUS 82

Lista de abreviaturas e siglas

APP	Aplicativos
℃	Grau Celsius
CIT	Stichting Mathematisch Centrum
CSTA	Associação de professores de ciência da computação
GB	Gigabyte
GPIO	General-purpose input/output
GPS	Global Position System (USA)
GSM	Global System for Mobile Communications
HTTP	Hypertext Transfer Protocol
ID	Identificação Digital
IoT	Internet of Things
IP	Internet Protocol
JVM	Java Virtual Machine
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MIT	Massachusetts Institute of Technology
MP3	MPEG-1/2 Audio Layer 3
RAM	Random Access Memory - Memoria de Acesso Aleatorio
REST	Representational State Transfer
RFID	Identificação por radiofrequência

RGB	Red, Green, Blue
S.O.	Sistema Operacional
SOAP	Simple Object Access Protocol
System Usability Scale	System Usability Scale
TCC	Trabalho de conclusão de curso
TCP/IP	Transmission Control Protocol/Internet Protocol
TI	Tecnologia da informação
UFSC	Universidade Federal de Santa Catarina
UML	Unified Modeling Language
WSGI	Web Server Gateway Interface

Sumário

1 INTRODUÇÃO 13

1.1 OBJETIVOS 14

1.1.1 OBJETIVO GERAL **14**

1.1.2 OBJETIVOS ESPECÍFICOS **15**

2 FUNDAMENTAÇÃO TEÓRICA 16

2.1 ENSINO DE COMPUTAÇÃO PARA CRIANÇAS E JOVENS 16

2.2 APP INVENTOR 18

2.2.1 LINGUAGEM VISUAL DE “BLOCOS” **22**

2.3 ENSINO DE IOT PARA CRIANÇAS/JOVENS 23

3 ESTADO DA ARTE 28

3.1 DEFINIÇÃO DO PROTOCOLO DE REVISÃO 28

3.1.1 CRITÉRIOS DE INCLUSÃO/EXCLUSÃO **28**

3.1.2 CRITÉRIO DE QUALIDADE **29**

3.1.3 DADOS A SEREM EXTRAÍDOS **29**

3.1.4 STRING DE BUSCA **30**

3.2 EXECUÇÃO DA BUSCA 31

3.2.1 PRIMEIRA E SEGUNDA ITERAÇÃO **31**

3.3 EXTRAÇÃO DAS INFORMAÇÕES ANÁLISE DOS RESULTADOS . 32

3.4 DISCUSSÕES 39

3.4.1 AMEAÇAS A VALIDADE DA REVISÃO DA LITERATURA **40**

4 ANLISE E DESENVOLVIMENTO 41

4.1 ANÁLISE DE REQUISITOS 41

4.2 TECNOLOGIAS UTILIZADAS 43

4.2.1 APP INVENTOR **43**

4.2.2 RASPBERRY PI **44**

4.2.3 PYTHON **45**

4.2.3.1 FLASK 46
4.2.4 JAVA 47
4.3 MODELAGEM 48
4.3.1 DIAGRAMA DE CLASSE 48
4.3.1.1 COMPONENTE INTERRUPTOR 49
4.3.1.2 COMPONENTE TEMPERATURA E UMIDADE 49
4.3.1.3 COMPONENTE MÚSICA 51
4.3.2 DIAGRAMA DE SEQUÊNCIA 51
4.4 DESENVOLVIMENTO 52
4.4.1 APLICAÇÃO NO CELULAR 53
4.4.2 BLOCOS APP INVENTOR 54
4.4.3 DISPOSITIVO IOT 55
4.4.4 IMPLEMENTAÇÃO SERVIDOR RASPBERRY PI 56
4.4.4.1 INICIALIZAÇÃO DISPLAY DE LED E SERVIDOR REST 57
4.4.4.2 ENDEREÇO IP PLACA RASPBERRY PI 57
4.4.4.3 SERVIÇO RELAY 58
4.4.4.4 SERVIÇO TEMPERATURA 60
4.4.4.5 SERVIÇO UMIDADE 60
4.4.4.6 SERVIÇO MÚSICA 61
4.4.5 COMPONENTES APP INVENTOR 63
4.4.5.1 COMPONENTE INTERRUPTOR 63
4.4.5.2 COMPONENTE TEMPERATURA 65
4.4.5.3 COMPONENTE UMIDADE 68
4.4.5.4 COMPONENTE MÚSICA 70

5 APLICAÇÃO E AVALIAÇÃO 73
5.1 DEFINIÇÃO DA AVALIAÇÃO 73
5.1.1 DEFINIÇÃO DA AVALIAÇÃO COM USUÁRIOS 74
5.2 EXECUÇÃO DA AVALIAÇÃO 76
5.3 ANÁLISE DOS DADOS 77

5.3.1 ANÁLISE DOS DADOS DAS AVALIAÇÕES COM USUÁRIOS **77**
5.3.2 PONTOS FORTES **82**
5.3.3 SUGESTÕES DE MELHORIA **82**
5.3.4 AMEAÇAS À VALIDADE **83**

6 CONCLUSÃO 84

Referências 87

1 INTRODUÇÃO

A iniciativa “Computação na Escola[1]” desenvolvida no Departamento de Informática e Estatística da Universidade Federal de Santa Catarina, tem o intuito de aumentar o ensino da computação nas escolas de nível básico (fundamental e médio). A iniciativa tem por objetivo instruir melhor as crianças e jovens para a área da computação e mostrar que a computação pode ser ensinada para todas as pessoas.

A computação pode ir além do “software” ou do “hardware”, podendo ser utilizada como ferramenta de apoio no ensino de outras matérias que estão indiretamente ligadas à computação, tornando-a uma área interdisciplinar que envolve biologia, física, química e matemática. Como observaram (BARR; STEPHENSON, 2011, p. 49):

> “[. . .] a comunidade de educação em informática pode desempenhar um papel importante no destaque da resolução de problemas algorítmicos práticos e aplicações de computação em todas as disciplinas, e ajudar a integrar a aplicação de métodos computacionais e ferramentas em diversas áreas de aprendizagem. ”

A “Computação na Escola” utiliza algumas ferramentas para facilitar assimilação, sendo a ferramenta App Inventor[2] um dos recursos utilizados. A ferramenta App Inventor foi desenvolvida pelo Instituto de Tecnologia de Massachusetts (MIT). Dentre os múltiplos recursos que a ferramenta App Inventor disponibiliza, inclusive o suporte ao ensino da programação e suas estruturas, a ferramenta utiliza uma abordagem visual no ensino da programação para computadores, possibilitando assimilação de maneira intuitiva, mesmo para crianças. Através disso, a ferramenta permite o ensino das estruturas de programação, utilizando o paradigma de programação em bloco, que facilita a criação de aplicativos complexos.

Nos últimos anos, tenta-se disseminar sistemas que possam se comunicar pela rede com aplicações embarcadas, podendo tanto coletar como transmitir dados, conceito conhecido como Internet das Coisas ou em inglês “Internet of Things“ (IoT). Definição de IoT segundo CASAGRAS (2009, p. 13):

[1] “Computação na Escola” - http://www.computacaonaescola.ufsc.br
[2] “App Inventor” - http://appinventor.mit.edu/

> "[...] uma infraestrutura de rede global, que interconecta fisicamente e virtualmente objetos, com o objetivo de explorar dados capturados e suas capacidades de comunicação. Essa infraestrutura inclui e envolve a Internet e as redes de comunicação, ela necessita de identificação única de objetos, sensores e capacidade de conexão, como base para o desenvolvimento independente de serviços e aplicações. Ela é caracterizada pelo alto grau de captura autônoma de dados, transferência de eventos de rede, conectividade e interoperabilidade."

A IoT hoje está presente no dia a dia sem que se perceba o quanto ela é importante para o funcionamento dos processos e das atividades recorrentes. Pode-se tomar como exemplo: sensor de GPS presente no celular, ou no carro para verificar a trajetória ou tráfego; sensores de trânsito, câmeras e camadas de informação mostrando o fluxo de automóveis para possibilitar análise do trânsito; o controle de aparelhos eletrodomésticos por interface de acessos chamados atuadores que, por exemplo, controlam o funcionamento de um ar condicionado. Segundo Pereira et al. (2016, p. 1):

> "[...] um atuador responsável por realizar as ações de: acionar, desativar, aumentar e diminuir a temperatura dos aparelhos de ar-condicionado gerenciados. O atuador em questão foi implementado sobre um Raspberry Pi, um microcomputador programável de baixo custo, o dispositivo foi equipado com um led emissor infravermelho utilizado para emitir comandos (ligar, desligar, aumentar e diminuir a temperatura) para os aparelhos de ar condicionado."

Atualmente a ferramenta App Inventor não permite o ensino de conceitos de computação utilizando estruturas físicas junto a rede TCP/IP e interação com o mundo real, o que pode ser interessante para alunos da Educação Básica.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

O objetivo principal deste trabalho é realizar o desenvolvimento de novos componentes junto à ferramenta App Inventor, para comunicação via rede TCP/IP com dispositivos IoT. E, através disso, permitir o desenvolvimento de aplicativos para celular junto à ferramenta App Inventor, com interação com dispositivos IoT. Com isso, espera-se aumentar o incentivo aos alunos da Educação Básica, no contexto da

iniciativa “Computação na Escola”, que utilizam a ferramenta App Inventor para o desenvolvimento de aplicativos.

Dessa maneira, a conclusão do projeto disponibiliza novos componentes na ferramenta App Inventor, que possibilitam a comunicação via protocolo TCP/IP com os dispositivos IoT.

O desenvolvimento junto à ferramenta App Inventor possibilita que pessoas com pouca experiência no campo da computação desenvolvam aplicativos que se comuniquem via TCP/IP com sensores externos à aplicação desenvolvida.

1.1.2 OBJETIVOS ESPECÍFICOS

- Analisar a literatura e o estado da arte relacionados às ferramentas de programação visual com IoT;
- Criar um novo módulo de componentes na ferramenta App Inventor para comunicação com IoT;
- Desenvolver e testar componentes para App Inventor permitindo a comunicação com dispositivos IoT, para apoio ao ensino de programação;

2 FUNDAMENTAÇÃO TEÓRICA

No presente capítulo são abordados os principais conceitos teóricos necessários para a compreensão deste trabalho. Primeiramente, é abordado o ensino da programação de computadores para crianças e jovens. Em sequência, é explanado sobre a ferramenta App Inventor, utilizando o conceito de programação em blocos. E, por fim, são apresentados os conceitos de IoT e suas aplicações junto ao campo educacional para crianças e jovens.

2.1 ENSINO DE COMPUTAÇÃO PARA CRIANÇAS E JOVENS

O ensino, conforme Libâneo (2006), tem como objetivo importante lapidar o aprendiz, desenvolvendo o valor psíquico e lógico para tomadas de decisões que acontecem ao longo da sua vida. Para que assim o aprendiz possa desenvolver a capacidade de análise humana (LIBÂNEO, 2006, p. 8). A abordagem a novos conhecimentos que o ensino proporciona, de uma maneira formal ou informal, aprimora as habilidades para superar problemas recorrentes do cotidiano.

Para Sundmaeker et al. (2010), a computação é vista como sendo um englobamento de conhecimento tanto teórico, prático e experimental. No atual cenário educacional público, é visto como sendo necessário o ensino de computação somente em cursos de graduação como Ciência da Computação, Sistema de Informação, entre outros (SUNDMAEKER et al., 2010, p. 2). O conhecimento da computação no século XXI é fundamental para a resolução de muitos dos problemas que são recorrentes na vida cotidiana, visto que grande parte das profissões na atualidade trabalham de maneira direta ou indireta com a área da computação. Dessa maneira, alguns autores trabalham com o conceito de “Pensamento Computacional” - uma forma de pensamento característica dos cientistas da computação, mas universalmente aplicável, que envolve um conjunto de atitudes e habilidades, tais como o uso da recursividade, abstração e decomposição na solução de problemas, tanto técnico-científicos quanto da vida cotidiana (WING, 2006). A Figura 1 mostra os eixos na computação e a importância do

pensamento computacional. Como citado por Raabe et al. (2017):

> "O Pensamento Computacional se refere à capacidade de sistematizar, representar, analisar e resolver problemas. Apesar de ser um termo recente, vem sendo considerado como um dos pilares fundamentais do intelecto humano, junto com leitura, escrita e aritmética, pois como estes, serve para descrever, explicar e modelar o universo e seus processos complexos (RAABE et al., 2017, p. 3)."

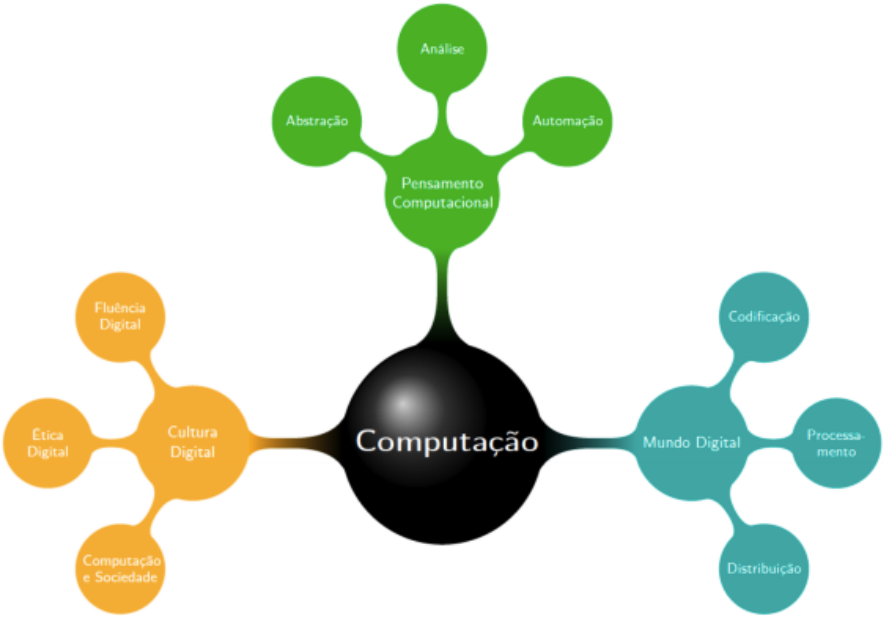

Figura 1 – Eixos da Computação (SBC 2017)
Sociedade Brasileira da Computação

Como visto, a importância da computação na atual sociedade, é praticamente fundamental para todas as áreas de trabalho. Tendo em vista que, no cenário atual, os computadores são manuseados por diversas e diferentes pessoas, nas profissões não poderia ser diferente. De acordo com o MEC (2011), toda profissão precisa do apoio da computação para se manter e se destacar no mercado atual, assim como se atualizar e atender com mais agilidade e qualidade suas demandas:

> "Não é um exagero dizer que a vida das pessoas depende de sistemas de computação e de profissionais que os mantêm, seja para dar segurança na estrada e no ar ou ajudar médicos a diagnosticar e tratar problemas de saúde, seja com um papel fundamental no desenvolvimento de novas drogas (MEC, 2011, p. 4)."

Dessa forma, Raabe et al. (2017) definem que é essencial o ensino da computação ao longo do ensino básico, para o aprendizado e o aprimoramento de conceitos importantes da computação. O aprendizado da computação pode ser iniciado na educação infantil trabalhando com conceitos lúdicos, com alguns pilares da computação. Nos anos seguintes - ensino fundamental - pode ser trabalhado com conceitos mais teóricos, aplicados dentro do campo da computação, permitindo já uma familiarização e experiência com o conteúdo. E por fim, quando o aluno esta na reta final de sua educação básica, com o conceito mais formado sobre a computação, pode se aprimorar a capacidade de resolução de problemas, através da realização de projetos e do desenvolvimento de habilidades relacionadas à análise crítica (RAABE et al., 2017, p. 7).

Nesse sentido, a CSTA K-12, uma associação dos professores de computação, define um currículo de ensino de Ciência da Computação em todo o Ensino Básico. O currículo da associação CSTA K-12 tem 3 níveis que se enquadram nos três níveis do ensino básico brasileiro. Para (SEEHORN et al., 2011), o ensino infantil do módulo “K-6 Ciência da Computação e Eu” foca no ensino introdutório sobre a Ciência da Computação, integrando habilidades básicas em tecnologia com simples ideias sobre o pensamento computacional. No nível 2 “Grade 6-9 - Ciência da Computação“ que é equivalente ao Ensino Fundamental, os alunos começam a usar o pensamento computacional como uma ferramenta de solução de problemas. E, por fim, no nível 3 ”Grade 9-12 - Aplicando conceitos e criando soluções do mundo real“ que é equivalente ao Ensino Médio, os alunos trabalham com conceitos mais avançados no campo da computação, e dessa forma, podem aplicar estes conceitos no desenvolvimento de trabalhos virtuais para a solução de problemas da vida real (SEEHORN et al., 2011, p. 8).

2.2 APP INVENTOR

De acordo com Falkembach et al. (2003), o ensino no âmbito da lógica de programação contém alguns desafios muito comuns para o aprendiz que dificultam sua

iniciação na programação de computadores:

- A interpretação, por parte do aluno em relação aos problemas que devem ser solucionados (FALKEMBACH et al., 2003, p.12)**;**
- Grau de abstração sobre a melhor maneira de abordar o problema proposto (FALKEMBACH et al., 2003, p.12)**;**
- Consenso sobre qual a esperada solução para um dado problema (FALKEMBACH et al., 2003, p.12);

Essas dificuldades são amenizadas na ferramenta . De acordo com Gomes e Melo (2013):

> "O App Inventor para Android, objeto de estudo do presente trabalho, é um ambiente visual de programação em blocos, o qual permite o desenvolvimento de aplicativos para dispositivos móveis Android de uma maneira consideravelmente simples, principalmente se comparada às linguagens de programação tradicionais. Seu ambiente gráfico possibilita o ensino de conceitos de lógica de programação de uma forma atraente e motivadora para estudantes do ensino médio e superior (GOMES; MELO, 2013, p. 3) ."

A ferramenta App Inventor é uma ferramenta com iniciativa de código aberto, possibilitando o desenvolvimento de aplicativos para dispositivos móveis. Ela tem o sua ambiente de desenvolvimento como grande diferencial, pois utiliza a estrutura de blocos para o ensino de programação. Outro diferencial é sua possibilidade de integração com plataformas externas como redes sociais, serviços disponibilizados pela web, entre outras integrações possíveis de serem realizadas. As aplicações geradas sobre a ferramenta são compiladas para rodarem sobre um ambiente Android, facilitando o desenvolvimento de aplicativos, uma vez que a grande maioria dos *smartphones* e *tablets* utilizam o Android (LECHETA, 2016) como Sistema Operacional. A ferramenta App Inventor é totalmente web e pode ser acessada online - ferramenta disponibilizado por sua organização - ou caso tenha que ser acessado de maneira *offline* pode ser instalado localmente.

De acordo com Morelli et al. (2011), a ferramenta foi criada em 2009 junto a Google, gerenciada pela equipe de Hal Abelson e atualmente é mantida pelo MIT. A ferramenta foi criada para disseminar e facilitar o ensino de programação de computadores para todas as escalas da sociedade, porém, com um foco maior no ensino para crianças e adolescentes. Com a crescente popularidade do uso de *smartphones* na atual sociedade - em especial entre os mais jovens - o App Inventor tem um grande potencial para a disseminação do pensamento computacional (MORELLI et al., 2011, p. 3).

O projeto do App Inventor está obtendo êxito no treinamento de professores para difundir o conceito de programação visual ao redor do mundo. Os treinamentos são ministrados por instrutores com certificados adquiridos junto ao MIT (APP INVENTOR, 2018).

O App Inventor é uma ferramenta que tem basicamente duas áreas de trabalho para o usuário final. Uma das áreas é chamada de "*Designer*", onde se encontram todos os componentes visuais como: caixa de texto aberto, botões entre outros componentes de tela, tal como é visto na Figura 2. Nesta área, é possível realizar o desenvolvimento da parte visual da aplicação elaborando estilos e montando a identidade da aplicação. A área "*Designer*" tem o objetivo de uma interface intuitiva e amigável, para o usuário não ter problemas na construção da interface gráfica da aplicação.

Figura 2 – Área de "*Designer*" - App Inventor
Plataforma online - App Inventor

A ferramenta App Inventor conta com uma segunda área de trabalho, onde é montada toda a lógica de programação da aplicação, chamada de "*Blocks Editor*". Essa área é responsável por interagir com os componentes criados na área de trabalho "*Designer*", dando ação aos elementos gráficos necessários. O ambiente "*Block Editor*", mostrado na Figura 3, disponibiliza para o usuário os blocos para encaixe, manipulados com ação de arrastar e soltar, que ao fim vão criar os comportamentos dos componentes de interface gráfica.

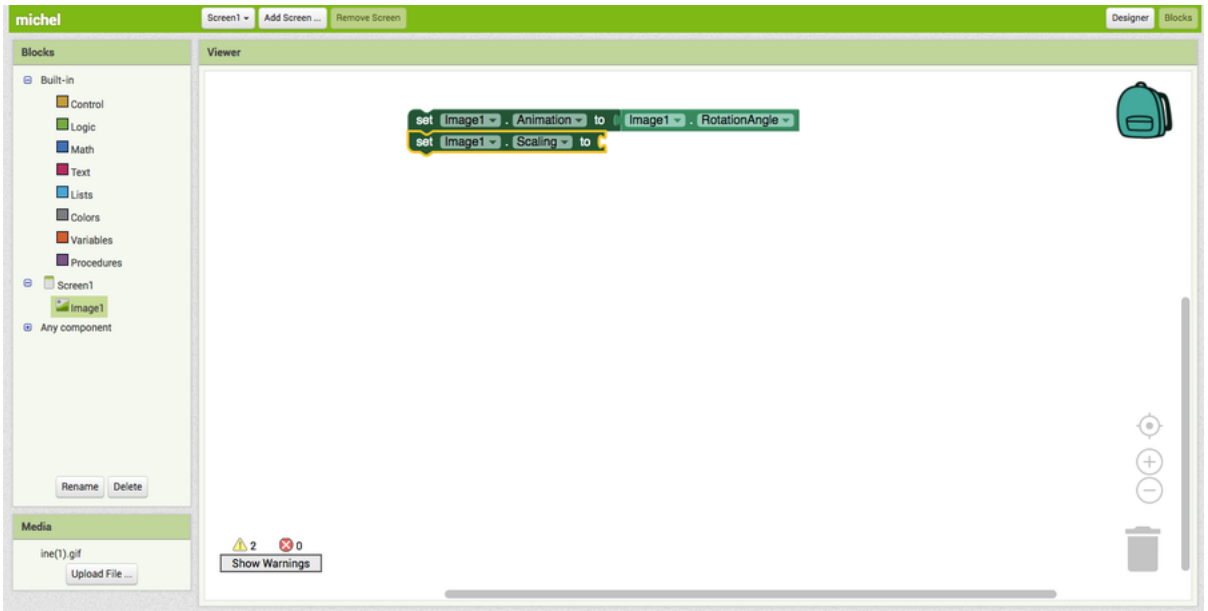

Figura 3 – Área "*Block Editor*" - App Inventor
Plataforma online - App Inventor

2.2.1 LINGUAGEM VISUAL DE “BLOCOS”

O conceito de linguagem visual que utiliza a aplicação de blocos para a montagem das estruturas de programação facilita a assimilação para essência do ensino da linguagem de programação (SUDOL, 2009)**.** Através disso, incentiva os jovens a ingressar no campo da programação para computadores. Para esses, fica mais fácil a assimilação do conteúdo de programação utilizando uma ferramenta de desenvolvimento visual, já que no começo eles não apresentam uma familiaridade com esse conteúdo. Dessa maneira, os alunos não precisam ter o conhecimento de determinada linguagem de programação para a construção de aplicações. Segundo Marcon Júnior e Boniati (2015):

> “Com o Blockly é mais difícil cometer um erro de sintaxe ou mesmo ambiguidades devido ao uso inadequado de parênteses. Adicionalmente, pessoas que nunca tiveram contato com programação podem achar uma tela branca com um cursor piscando amedrontadora, porém utilizando Blockly elas podem navegar por menus e ir conectando blocos de forma intuitiva e experimentar a partir de exemplos como as coisas funcionam (MARCON JÚNIOR; BONIATI, 2015, p. 66) “.

A programação visual permite modelar todos os conceitos de programação: estrutura de condição, laços de repetição, variáveis entre outras estruturas. Esses itens são apresentadas separadamente em blocos para serem arrastados e encaixados, montando funções que resolvem determinado problema. Todos os blocos já estão criados e visíveis aos usuários para serem utilizados na construção de algoritmos. Dessa maneira, a criação dos algoritmos é muito intuitiva para o usuário, sendo mostrados *feedbacks* quando um bloco encaixado não é compatível ao bloco que o antecede.

Uma linguagem que esta difundida no contexto de programação visual é a linguagem Blockly [1] que foi criada pela Google. A linguagem de programação visual Blockly foi desenvolvida indiretamente para melhorar a interface de *frontend* da ferramenta App

[1] blockly - https://developers.google.com/blockly/

Inventor, que foi construída, num primeiro momento, na linguagem de programação Java. Posteriormente, foi desenvolvido um Kit de ferramentas - Blockly - que pode ser integrado a outras plataformas para usar o conceito de programação em bloco. Ao realizar a refatoração no projeto Blockly, optou-se por uma linguagem que não necessita de um container para a sua execução. Por esse motivo, foi escolhida a linguagem de programação Javascript, que no cenário atual é uma das mais difundidas no mercado.

A biblioteca Blockly esta sendo administrada pelo Google junto à Equipe Scratch do MIT Media Lab, que está aprimorando o conceito de programação visual. A programação visual está em alta e atraindo muitas pessoas para terem um pensamento computacional. Um exemplo da alta do pensamento computacional é a organização Hour of Code[2] que organizou mais de 70 mil eventos e foi testado com mais de 95 milhões de pessoas desde alunos até mesmo chefes de estado como citado por Ibanez (2015).

2.3 ENSINO DE IOT PARA CRIANÇAS/JOVENS

O mundo está extremamente voltado ao campo tecnológico. Em 2011 a população chegou a 7 bilhões de pessoas no planeta e o número de aparelhos conectados chegou a 13 bilhões. Em 2015, o número de aparelhos superou 3 vezes o número da população mundial e a tendência é de aumentar ainda mais ao longo dos próximos anos. A internet nos tempos atuais é muito difundida e necessária para toda a sociedade. Dessa forma, ela tornou-se um instrumento importante para o dia a dia, e nos dias atuais é um recurso essencial para tarefas diárias. Pode ser vista como uma das maiores invenções já feitas pela humanidade, como citado por Castells (2003):

> "Se a tecnologia da informação é hoje o que a eletricidade foi na Era Industrial, em nossa época a Internet poderia ser equiparada tanto a uma rede elétrica quanto ao motor elétrico, em razão de sua capacidade de distribuir a força da informação por todo o domínio da atividade humana.(CASTELLS, 2003, p. 7) ".

Com a disseminação da internet ao longo dos anos, novos paradigmas que

[2] Hour of Code - https://hourofcode.com/us

interagem com o meio e com outros dispositivos surgiram. A partir disso surgiu o conceito de Internet das Coisas (IdC, ou do inglês IoT), para Castells (2003):

> "Para mim, a Internet das Coisas é como o vento. Você vê as coisas se movendo. Você sabe que algo as está movendo, mas não sabe exatamente o que é. Em alguns lugares, árvores chacoalham e prédios são destruídos; em outros, está tudo calmo e tranquilo. Até certo ponto é previsível, mas muitas vezes não.(CASTELLS, 2003, p. 2)".

O termo utilizado para descrever IoT foi citado a primeira vez no ano de 1999 pelos pesquisadores do Auto-ID da universidade MIT para demostração de objetos conectados na rede por meio da tecnologia RFID, como citado por Sundmaeker et al. (2010). Dessa forma, a IoT tem um propósito de interfaceamento dos dispositivos que são muito comuns na nossa rotina, como *smartphones*, geladeiras, e televisores, na interação destes dispositivos com o meio e outros dispositivos. Dessa forma, fica claro o conceito primordial do propósito da IoT. De acordo com Sundmaeker et al. (2010):

> "Mesmo que a primeira definição de IoT dada pelo laboratório Auto-ID estivesse relacionada com RFID, na verdade, a IoT está ligada não apenas à identificação por radiofrequência, mas também a outras soluções tecnologias: redes de sensores, TCP / IP, tecnologias móveis e, em geral, a todas as tecnologias de identificação que permitem identificar objetos, coletar e processar informações sobre eles e ligar o mundo físico ao virtual (SUNDMAEKER et al., 2010, p. 105)".

Todo objeto que está compartilhado na rede, consumindo ou gerando algum tipo de informação, pode se aplicar dentro do conceito de IoT. E, dessa forma, interligar-se a uma rede permitindo que todos os dispositivos se comuniquem sem qualquer intervenção dos seres humanos, fazendo troca de dados entre os próprios dispositivos.

Com o advanço da IoT, foi possível realizar várias fusões com outras tecnologias já existentes e consolidadas, criando com isso novos serviços como GSM, Bluetooth, GPS, entre outros, segundo Sundmaeker et al. (2010). Assim, é possível uma coleta de dados e informação com uma granularidade menor, e, através disso, obter-se uma exatidão maior nas informações coletadas.

Um dos grandes propósitos da IoT é a otimização de vários processos que são feitos de uma maneira manual, e muitos já se tornaram realidade na vida real. Dessa

forma, uma das aplicações teóricas que são muito comuns atualmente é a análise na parte logística do transporte público, que possibilita a verificação em tempo real de onde determinado ônibus está localizado, podendo otimizar o tempo que as pessoas precisam esperar no ponto de ônibus.

O ensino de computação para crianças e jovens é uma tendência pela importância que a computação tem no cotidiano, preparando as crianças a lidarem com a computação de uma maneira natural. Por algumas áreas da computação serem muito abstratas, o ensino para crianças fica um pouco mais difícil de ser assimilado. Sendo assim, uma abordagem de ensino voltada à IoT, despertaria um interesse maior por parte das crianças, que não ficariam presas a interações somente via periféricos normais do computador (mouse, teclado e tela). Para tal, poderiam ser mostradas outras formas de computação que interagem com o nosso meio, como sensores e atuadores.

O ensino de computação voltado ao campo do IoT é abordado em alguns países junto ao calendário escolar, contemplando até mesmo algumas competições para incentivar ainda mais as crianças neste ramo da computação, como a RoboCupJunior, citada por Sklar, Eguchi e Johnson (2002):

> "[. . .] RoboCup, usando robôs construídos e programados com o LEGO Mindstormskit para jogar futebol [8]. Desde então, o RoboCupJunior evoluiu para um evento internacional onde equipes de jovens estudantes constroem robôs para competir em um dos três desafios: futebol, resgate e dança .
>
> [. . .] Crianças, com idades 13-16, de 8 escolas formaram 12 equipes para jogar futebol cara-a-cara (SKLAR; EGUCHI; JOHNSON, 2002, p. 239). "

A computação com dispositivos IoT apresenta algumas dificuldades para ser adotada nas escolas brasileiras. Segundo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira. Brasil (2010-2014), há um grande déficit de professores especializados na área de ensino da computação, e não há materiais como sensores e microcomputadores, utilizados para o interfaceamento entre os sensores (INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA. Brasil, 2010-2014).

Outro ponto que deve ser levado em consideração é que a abordagem de

uma forma direta com as placas de microcontroladores - como Arduino, Raspberry Pi e GogoBoard - vai exigir um conhecimento na parte de eletrônica e computacional por parte dos alunos, algo que não é de fácil assimilação para grande maioria das crianças e adolescentes. Uma barreira que também seria gerada ao trabalhar diretamente com microcomputadores é a linguagem de programação que, na maioria das vezes, é de baixo nível. Dessa maneira, o uso de linguagem com uma abordagem visual - como já discutido na seção 2.2.1 - facilitaria a compreensão por parte de uma criança ou adolescente, que não precisa ter conhecimento da sintaxe de uma linguagem de programação para a implementação de aplicações com interação com microprocessadores. A programação utiliza abordagem visual, tornando o ensino para as crianças ou adolescentes mais atrativo, por conseguir trabalhar com imagens que são melhor assimiladas. De acordo com Costa (2005), as imagens apresentam um caráter intuitivo muito maior do que a linguagem verbal/escrita, pois, elas são mais universais do que as linguagens verbais e sonoras. Assim, a utilização da imagem pode ser útil como um recurso didático, pois esse caráter intuitivo da linguagem visual pode facilitar a aprendizagem dos estudantes.

Segundo (MEDEIROS FILHO; GONÇALVES, 2008), o governo brasileiro apresenta poucas ou até mesmo nenhuma iniciativa para a inclusão do ensino de IoT na educação básica.

> "Apesar de muitas pesquisas indicarem a robótica educacional como sendo uma ferramenta que envolve questões multidisciplinares, portanto rica pedagogicamente, ela, infelizmente não faz parte do cotidiano das escolas brasileiras. A explicação para tal fato, passa pela dificuldade na aquisição do equipamento. Essa dificuldade reside, principalmente, no momento de sua compra, pois seu custo, ainda é proibitivo (MEDEIROS FILHO; GONÇALVES, 2008, p. 265)."

Porém, alguns países apresentam currículos de estudos na área da computação bem consolidadas, como o "Curriculum Guidelines for K-12 Computing Education", já comentado na Seção 2.1. Esse currículo auxiliaria na introdução de conceitos fundamentais para a computação, com definições importantes sobre microcomputadores, sinalizadores, sensores e atuadores que são importantes para área de IoT.

Para Ruzzenente et al. (2012), o ensino de computação na área da Internet das Coisas, torna-se interdisciplinar, por trabalhar com alguns conceitos de disciplinas tradicionais previstas em matemática, física, programação de computadores e eletrônica, mas também filosofia, desenvolvimento da linguagem, história, e assim, a alfabetização de currículo do nível primário, secundário, de graduação e de pós-graduação. Isso pode, portanto, incentivar a continuar os estudos tanto no campo computacional como em áreas subjacentes a computação, tendo em vista que se almeja que grande parte dos alunos do Ensino Básico tenha a oportunidade para explorar o pensamento computacional e aflorar o lado criativo com o auxílio da computação.

3 ESTADO DA ARTE

Neste capitulo é levantado o estado da arte atual de pesquisas relacionadas a área de ensino de computação física/robótica/IoT com App Inventor. A análise do estado da arte foi realizada seguindo o método de revisão sistemática definida por Kitchenham (2004).

3.1 DEFINIÇÃO DO PROTOCOLO DE REVISÃO

Para identificar o estado da arte, foi definida uma revisão sistemática da literatura, com objetivo de analisar e sintetizar as literaturas existentes sobre a pergunta “O que está sendo feito atualmente na área de ensino de computação física/robótica/IoT com App Inventor? “.

Entre as alternativas de ferramentas para realizar a pesquisa sobre a literatura existente, foi escolhido o Google Scholar[1], que é abrangente para busca em fontes bibliográficas digitais. Porém, em virtude do App Inventor conter um repositório de bibliografias, foram realizadas pesquisas também em seu repositório. Dentre as ferramentas para pesquisas bibliográficas, o Google Scholar possui o recurso de pesquisas dentro do idioma da língua portuguesa do Brasil, além de outros idiomas, o que facilitou a pesquisa dos termos nos idiomas propostos pelo trabalho. Para realizar a pesquisa foram utilizados termos na língua portuguesa do Brasil e no inglês dentro do período de 2014 a 2018.

3.1.1 CRITÉRIOS DE INCLUSÃO/EXCLUSÃO

Foram consideradas as bibliografias que cumprem o requisito:

- Aplicação da computação no campo educacional, com crianças e adolescentes;
- Ensino de IoT no campo educacional;

[1] Google Scholar - https://scholar.google.com.br/

- Utilizar a ferramenta da Google App Inventor para ensino de computação a crianças e adolescentes;

3.1.2 CRITÉRIO DE QUALIDADE

- Priorizar as bibliografias que utilizam ferramentas com abordagem visual para o ensino de computação;
- Importante que a pesquisa tenha um foco voltado às crianças ou aos adolescentes com faixa etária do Ensino Básico;
- Considerar artigos que abordaram o ensino de computação na área de IoT;
- A profundidade em que o material apresenta detalhes na implementação da integração com as ferramentas;

3.1.3 DADOS A SEREM EXTRAÍDOS

Os dados extraídos dos trabalhos selecionados, de forma a procurar responder à pergunta de pesquisa, são:

- Descrição da estratégia de ensino de computação para crianças e adolescentes utilizada;
- Descrição da estratégia de ensino de computação física/robótica/IoT para crianças e adolescentes utilizada;
- Arquitetura e tecnologias utilizadas para o ensino de computação física/robótica/IoT;
- Como foi utilizada a ferramenta App Inventor;
- Resultados da avaliação da aplicação no ensino;

3.1.4 STRING DE BUSCA

Os termos pesquisados na ferramenta Google Scholar são derivados da pergunta proposta e são mostrados na tabela 1:

Termo	**Sinônimo**	**Tradução (inglês)**
“Ensino de Programação”	“Ensino de lógica de programação”, “ensino de algoritmos”, “ensino de construção de algoritmos”, ”ensino de computação“	“Programming education”, “programming logic education”, “algorithm education”, “algorithm building education”, “computer education”
“App Inventor”	“programação visual”, “programação em bloco”	“visual programming”, “block programming”
“Internet das Coisas”	“computação física”, “robótica”, “IoT”	“physical computing”, “Robotics”, “IoT”

Tabela 1 – Sinônimos e traduções dos termos de busca
elaborado pelo autor

A *String* de busca gerada a partir das palavras derivadas é mostrada na tabela 2, com o total de resultados retornados:

Idioma	**Expressão de Busca**	**Total de resultados**
Português	(“App Inventor” OR “programação visual” OR “programação em bloco”) AND (“Internet das Coisas” OR “computação física” OR “robótica” OR “IoT”) AND (“Ensino de Programação” OR “Ensino de lógica de programação” OR “ensino de algoritmos” OR “ensino de construção de algoritmos” OR “ensino de computação”) PERÍODO (2014 a 2018)	1439
Inglês	(“App Inventor” OR “visual programming” OR “block programming”) AND (“Internet of Things” OR “physical computing” OR “Robotics” OR “IoT”) AND (“Programming Teaching” OR “Programming education” OR “programming logic education” OR “algorithm educational” OR “algorithm building education” OR “computer education”) AND (“App Inventor” OR “visual programming” OR “block programming”) PERÍODO (2014 a 2018)	5835

Tabela 2 – Expressão de busca.
elaborado pelo autor

3.2 EXECUÇÃO DA BUSCA

As buscas foram feitas em maio de 2018. No total foram realizadas 8 iterações para concluir as buscas, que vão ser apresentadas na sequência.

3.2.1 PRIMEIRA E SEGUNDA ITERAÇÃO

Foi realizada a pesquisa dos termos separados pelo idioma - português e inglês - pois a ferramenta de busca Google Scholar limita a *String* de busca. No idioma

português, a ferramenta retornou um total de 1439 resultados e no inglês retornou um total de 5835 resultados. Porém, a *String* de busca continuou sendo limitada pela ferramenta do Google Scholar. Após a leitura dos 7274 primeiros resultados, foram selecionados 6 resultados que atendem aos objetivos desta pesquisa.

3.3 EXTRAÇÃO DAS INFORMAÇÕES ANÁLISE DOS RESULTADOS

Nesta seção são apresentados os resultados extraídos das execuções das buscas. Os resultados obtidos são:

Primeiro resultado (QUEIROZ; SAMPAIO; SANTOS, 2016)	
Estratégia de ensino de computação para crianças e adolescentes.	[. . .] vem adotando a estratégia de associar a Robótica Educacional e ambientes de Programação Visual em Blocos no desenvolvimento de atividades pedagógicas relacionadas ao currículo dos anos iniciais do Ensino Fundamental I (QUEIROZ; SAMPAIO; SANTOS, 2016).
Estratégia de ensino de computação física/robótica/IoT para crianças e adolescentes.	[. . .] caixinha batizada de “caixinha mágica” que permite às crianças programarem simultaneamente um conjunto de atuadores e sensores (QUEIROZ; SAMPAIO; SANTOS, 2016).
Arquitetura e tecnologias utilizadas para ensino de computação física/robótica/IoT.	Placa Arduino contendo os seguintes componentes: 1 display LCD, um display de 7 segmentos, 1 buzzer, 1 servo motor, 1 motor DC, 4 LEDs, 1 LED RGB, 1 sensor de distância, 1 sensor de luz e 1 sensor de temperatura (QUEIROZ; SAMPAIO; SANTOS, 2016).

Primeiro resultado (QUEIROZ; SAMPAIO; SANTOS, 2016)	
Como foi utilizada a ferramenta App Inventor.	Não usa App Inventor. Foi utilizado DuinoBlocks4Kids.
Resultados da avaliação da aplicação no ensino.	Uma avaliação parcial do aproveitamento dos alunos já nos permite constatar que entre as crianças dos 3º e 4º anos, metade conseguiu, ao final do curso, apresentar um bom domínio dos conteúdos trabalhados: sequenciamento de comandos, estruturas de repetição (contada e condicional) e reconhecimento e entendimento dos sensores e atuadores utilizados nos experimentos (QUEIROZ; SAMPAIO; SANTOS, 2016).

Tabela 3 – Primeiro resultado bibliográfico
elaborado pelo autor

Segundo resultado (HEINEN, 2015)	
Estratégia de ensino de computação para crianças e adolescentes.	[. . .] projeto é proposto como ferramenta para realização de oficinas de robótica paralelas à lições teóricas de programação de computadores (HEINEN, 2015).
Estratégia de ensino de computação física/robótica/IoT para crianças e adolescentes.	[. . .] componentes eletrônicos conectados a um minicomputador Raspberry Pi, de maneira que o aluno observe o resultado do experimento por meio do comportamento apresentado pelo modelo robótico (HEINEN, 2015).

Segundo resultado (HEINEN, 2015)	
Arquitetura e tecnologias utilizadas para ensino de computação física/robótica/IoT.	[. . .] componentes eletrônicos conectados ao Raspberry Pi, tais como motores de corrente direta, sensores de proximidade, botões e LEDs, constituem esta camada da arquitetura (HEINEN, 2015).
Como foi utilizada a ferramenta App Inventor.	Não usa App Inventor. Foi utilizado Blockly.
Resultados da avaliação da aplicação no ensino.	Estimula a exploração, permitindo que o aluno observe o resultado de cada instrução nas reações do modelo robótico (HEINEN, 2015).

Tabela 4 – Segundo resultado bibliográfico
elaborado pelo autor

Terceiro resultado (AMORIM et al., 2016)	
Estratégia de ensino de computação para crianças e adolescentes.	[. . .] integração dos processos pedagógicos, as tecnologias, e os princípios da computação, tendo a robótica como instrumento motivador pode oferecer de fato, segundo, um ambiente de aprendizagem que prioriza a forma de aprender de cada indivíduo na sua diversidade, oferecendo múltiplos estímulos como a visão, a audição e o tato simultaneamente (AMORIM et al., 2016).
Estratégia de ensino de computação física/robótica/IoT para crianças e adolescentes.	[. . .] objetivo de construção, programação e controle de um humanoide voltado para atividades pedagógicas, o trabalho descreve não só o processo com suas etapas e recursos, mas também uma análise de sua importância para construção de competências e habilidades pelos estudantes envolvidos diante das novas tecnologias (AMORIM et al., 2016).
Arquitetura e tecnologias utilizadas para ensino de computação física/robótica/IoT.	Placa Arduino que realiza o controle dos sensores do humanoide. [. . .] envio de comandos via porta serial para o Arduino; em seguida utilizando diferentes tecnologias embarcadas nos smartphones como sensores de equilíbrio e posicionamento GPS, finalizando com programas simples para controle de robôs com rodas (AMORIM et al., 2016).

Terceiro resultado (AMORIM et al., 2016)	
Como foi utilizada a ferramenta App Inventor.	[. . .] com o uso do App Inventor, na construção de aplicativos para a comunicação com o Arduino e controle das ações [. . .] (AMORIM et al., 2016).
Resultados da avaliação da aplicação no ensino.	Os resultados apresentados nesta experimentação, de fato contribuem para demonstrar o desenvolvimento de habilidades voltadas para a construção de algoritmos e programação com estudantes de ensino fundamental, assim como o desenvolvimento de aprendizagens necessárias como pré requisitos na construção de aplicativos produzidos através de linguagens de programação com criatividade e funcionalidade (AMORIM et al., 2016).

Tabela 5 – Terceiro resultado bibliográfico
elaborado pelo autor

Quarto resultado (TISSENBAUM et al., 2017)	
Estratégia de ensino de computação para crianças e adolescentes.	Ensinar os conceitos da computação através da linguagem de programação, porém, é uma barreira de entrada ter o conhecimento da sintaxe de uma linguagem de programação. Dessa maneira, muitos pesquisadores educacionais desenvolveram ambientes baseados em blocos. Esses ambientes aproveitam a metáfora de peças de quebra cabeça, em que os usuários montam programas juntando os "blocos" de código (TISSENBAUM et al., 2017).
Estratégia de ensino de computação física/robótica/IoT para crianças e adolescentes.	Realizar o ensino de IoT através da ferramenta App Invetor - programação em blocos - que realiza a comunicação com uma placa Arduino que controla alguns sensores (TISSENBAUM et al., 2017).
Arquitetura e tecnologias utilizadas para ensino de computação física/robótica/IoT.	Placa Arduino que controla alguns sensores e atuadores (TISSENBAUM et al., 2017).
Como foi utilizada a ferramenta App Inventor.	[..] desenvolvemos uma extensão para App Inventor que permite que os jovens criem aplicativos Mobile que possam enviar e receber dados do Arduino (TISSENBAUM et al., 2017).

Quarto resultado (TISSENBAUM et al., 2017)	
Resultados da avaliação da aplicação no ensino.	[. . .] Visto que, fornece oportunidades para as crianças jovens para criticamente explorar seu potencial para mudar seu relacionamento com o mundo ao seu redor, abrimos novos caminhos para os jovens transcenderem a visão de Papert de agentes intelectualmente empoderados (TISSENBAUM et al., 2017).

Tabela 6 – Quarto resultado bibliográfico
elaborado pelo autor

Quinto resultado (MERKOURIS; CHORIANOPOULOS; KAMEAS, 2017)	
Estratégia de ensino de computação para crianças e adolescentes.	Utilização de programação física para o ensino de computação com uma ferramenta de programação visual (MERKOURIS; CHORIANOPOULOS; KAMEAS, 2017).
Estratégia de ensino de computação física/robótica/IoT para crianças e adolescentes.	Os alunos foram convidados a conectar os atuadores e o sensor com o microcontrolador para tornar o robô capaz de se mover e sentir o ambiente. E, por fim, os alunos interagiram escrevendo alguns algoritmos junto a plataforma Modkit e Enchanting para comunicação com a placa montada por eles próprios (MERKOURIS; CHORIANOPOULOS; KAMEAS, 2017).

Quinto resultado (MERKOURIS; CHORIANOPOULOS; KAMEAS, 2017)	
Arquitetura e tecnologias utilizadas para ensino de computação física/robótica/IoT.	Utilizado Lego Mindstorms, controlado pela placa Arduino LilyPad (MERKOURIS; CHORIANOPOULOS; KAMEAS, 2017).
Como foi utilizada a ferramenta App Inventor.	Não usa App Inventor. Foi utilizado a plataforma Modkit e Enchanting.
Resultados da avaliação da aplicação no ensino.	[. . .] envolvimento emocional sugere que os alunos do grupo robótico estavam mais engajados do que os estudantes do grupo genérico, pois, relataram emoções (MERKOURIS; CHORIANOPOULOS; KAMEAS, 2017). No que diz respeito às atitudes, os alunos do grupo robótico eram mais propensos a se envolver na computação durante o tempo de lazer e acredita-se ter mais habilidades de programação do que aqueles no grupo genérico (MERKOURIS; CHORIANOPOULOS; KAMEAS, 2017).

Tabela 7 – Quinto resultado bibliográfico
elaborado pelo autor

3.4 DISCUSSÕES

A busca sobre as bibliografias, mostrou uma ampla iniciativa de trabalhos de pesquisa com o intuito de melhorar o ensino de computação com o auxílio da robótica para crianças e adolescentes.

Um fato discutido por alguns trabalhos levanta a questão dos custos para implantação das iniciativas de ensino de robótica nas escolas públicas brasileiras, em virtude de alguns componentes terem custo elevado. Dessa maneira, alguns trabalhos propõem soluções alternativas, de baixo custo, para o ensino de robótica. Por exemplo, utilizar componentes de hardware livre como minicomputadores Arduino[2] e

[2] Arduino - https://www.arduino.cc/

Raspberry Pi[3], que tem um custo inferior em relação a outros componentes como Lego Mindstorms[4].

Visto que, uma das grandes dificuldades apresentadas nas pesquisas bibliográficas é o custo elevado dos componentes de hardware para a implantação de projetos junto as escolas brasileiras, o presente trabalho implementado tem uma característica importante a ser considerada, pois a construção da placa teve um custo baixo em comparação com outros hardwares que são utilizados em trabalhos no ambiente escolar.

3.4.1 AMEAÇAS A VALIDADE DA REVISÃO DA LITERATURA

Na intenção de ter uma pesquisa abrangente com todos os termos relevantes ao trabalho proposto, e por uma limitação na *String* de busca que é inferida pela ferramenta Google Scholar, foi realizada a combinação de alguns termos, separados por idioma para não extrapolar o limite exigido pela ferramenta Google Scholar. Portanto, a quantidade de resultados total de bibliografias em cada idioma pode ter alguns trabalhos repetidos, por motivo de alguns trabalhos se encaixarem em mais de um *String* de busca.

[3] Raspberry Pi - https://www.raspberrypi.org/
[4] Lego Mindstorms - https://www.lego.com/en-us/mindstorms

4 ANLISE E DESENVOLVIMENTO

Este capítulo apresenta a análise dos requisitos dos componentes que foram implementados junto a ferramenta App inventor, e do desenvolvimento do servidor junto ao dispositivo IoT. Além disso, são detalhadas as tecnologias envolvidas, são discutidas as principais tomadas de decisões durante o processo de desenvolvimento e é descrita a implementação da solução.

4.1 ANÁLISE DE REQUISITOS

A partir das necessidades identificadas na análise do estado da arte e do estudo da literatura, são levantados os requisitos, identificando requisitos funcionais, apresentados na Tabela 8

ID	Descrição	Entrada	Saída
RF01.	Sistema deve permitir criar um conjunto de blocos de programação no App Inventor, que permitam interagir com o acionamento de um relay.	Informações de conexão da placa, GPIO, status relay.	Status code http.
RF02.	Sistema deve permitir criar um conjunto de blocos de programação no App Inventor para interagir com um sensor de temperatura do ambiente.	Informação da conexão da GPIO.	Informar temperatura.

ID	Descrição	Entrada	Saída
RF03	Sistema deve permitir criar um conjunto de blocos de programação no App Inventor para interagir com um sensor de umidade do ambiente.	Informação de conexão da GPIO	Informar a umidade.
RF04	Sistema deve permitir criar um conjunto de blocos de programação no App Inventor para executar/parar músicas.	Informar qual música executar/parar	Executar/parar a música
RF05.	O sistema deve permitir a conexão por meio de um serviço embarcado para ligar ou desligar relay.	GPIO, status relay.	Status code http.
RF06.	Criar serviço junto ao servidor da placa para informar temperatura.	GPIO	Temperatura do ambiente.
RF07.	Criar serviço junto ao servidor da placa para informar umidade.	GPIO	Umidade do ambiente
RF08.	Criar serviço junto ao servidor da placa para executar/parar música.	Informação para executar/parar e informar a música	Executar/parar música
RF09.	Mostrar IP do servidor	Identificação do endereço IP da rede	IP

Tabela 8 – Requisitos funcionais
elaborado pelo autor

E os requisitos não funcionais são apresentados na Tabela 9

ID	**Descrição**
RNF01.	Os componentes devem ser de extensão aix.
RNF02.	Os blocos criados devem ter baixa complexidade, não exigindo informação técnica por parte do usuário.
RNF03.	A extensão deve ser menor que 200 megabytes.
RNF04.	Os componentes criados devem adotar os padrões de projetos já utilizados pelo App Inventor.
RNF05.	O servidor deve ser acessado junto a rede local.
RNF06.	Usar a linguagem de programação Java, já utilizada pelo App Inventor nos componentes.
RNF07.	O sistema deve executar sobre rede TCP/IP.

Tabela 9 – Requisitos não-funcionais
elaborado pelo autor

4.2 TECNOLOGIAS UTILIZADAS

Nesta seção são apresentadas as tecnologias empregadas no desenvolvimento do componente para o dispositivo IoT, sendo elas: a ferramenta App Inventor, o minicomputador Raspberry Pi, linguagem de programação Python para o desenvolvimento do servidor no minicomputador Raspberry Pi e a linguagem de programação Java para o desenvolvimento dos componente na ferramenta App Inventor.

4.2.1 APP INVENTOR

Tendo em vista os recursos que os dispositivos móveis podem trazer para a educação e, através disso, resolver dificuldades no ensino de computação para as crianças e adolescentes, utilizou-se a ferramenta App Inventor neste trabalho. Ela é uma ferramenta gratuita e online, para a construção de aplicações Android, sem a necessidade de conhecimento prévio sobre uma determinada linguagem de programação.

O App Inventor é uma ferramenta open-source que pode ser extensível para a criação de novos componentes, que possibilitam o aprimoramento da ferramenta App

Inventor como citado em sua própria documentação oficial (MASSACHUSSETS. . . , 2012):

> "Qualquer um pode criar componentes de extensão. Isso requer familiaridade com o código-fonte do App Inventor (localizado no Github) e programação da extensão em Java. Componentes de extensão são empacotados como arquivos aix. Depois de criar um componente de extensão, qualquer pessoa poderá usá-lo em seus projetos do App Inventor. Os arquivos do componente de extensão aix podem ser armazenados em qualquer lugar na web. Os arquivos aix não precisam ser armazenados no MIT ou em qualquer outro lugar específico, embora o MIT hospede um repositório onde as pessoas possam disponibilizar arquivos aix para compartilhamento e uso público. (MASSACHUSSETS. . . , 2012)"

Dessa forma, foram desenvolvidos componentes extensivos para a comunicação com dispositivos embarcados, como é o caso do Raspberry Pi que é utilizado no desenvolvimento deste trabalho. Junto a ferramenta App Inventor, foram desenvolvidos quatro componentes extensíveis: interruptor, temperatura, umidade e música. A comunicação dos componentes desenvolvidos é feita através da rede TCP/IP, utilizando o protocolo HTTP, para que as informações possam ser transmitidas para as aplicações desenvolvidas pelos alunos.

4.2.2 RASPBERRY PI

Raspberry Pi, criado pela fundação Raspberry Pi Foundation, é um minicomputador criado com a intenção de estimular e facilitar o ensino de computação junto as universidades. Com um tamanho semelhante ao de um cartão de crédito, seus componentes são concentrados numa única placa de hardware, tendo ainda, um poder de processamento significativo. O Raspberry Pi é usado no desenvolvimento de projetos de aplicações Web, utilizando o Raspberry Pi como um micro servidor Web de aplicação, para ser utilizado como dispositivo IoT.

Uma das versões do dispositivo Raspberry Pi é o modelo Raspberry Pi 2 model B, que é construída sobre um processador ARM Cortex-A7 de 900MHz, e possui 1 GB de memória RAM. Tais configurações permitem a execução de um Sistema Operacional mais otimizado, como é o caso do Raspbian que foi criado para rodar

sobre uma arquitetura Raspberry Pi. Este é um dos grandes diferencias entre um minicomputador (Raspberry Pi) e uma placa controladora (Arduíno), que executa código compilado, oferecendo vantagens para aplicações em que o tempo de execução das instruções é um ponto crítico.

Raspberry Pi, contém em sua arquitetura GPIO (*General Purpose Input Output*), pinos digitais que oferecem a possibilidade de controle dos sensores digitais. Os pinos digitais são gerenciados pelo Sistema Operacional da Raspberry Pi, que funciona como um interruptor podendo ligar ou desligar determinado pino, transmitir ou receber dados e controlar a passagem de corrente elétrica.

Durante o desenvolvimento do trabalho, foi criado um servidor junto a placa Raspberry Pi, utilizando a arquitetura de microservice REST, com o auxílio da linguagem de programação Python - que gerencia as portas digitais da placa. Para facilitar a comunicação com o microservice REST que foi implementado na placa, foi adicionado um display digital, que ao inicializar o S.O. informa o IP local, possibilitando a troca de informação entre o componente criado no App Inventor e os sensores acoplados a placa Raspberry Pi.

4.2.3 PYTHON

Python é uma linguagem de programação de alto nível, que foi criada no ano de 1990 no Stichting Mathematisch Centrum(CIT) por Guido van Rossum e foi disponibilizado ao público geral no ano de 1991 (FOUNDATION, 2016). A linguagem é disponibilizada de forma gratuita, assim como, a grande maioria de seus tutorias e guias.

A linguagem de programação Python tem em suas características ser uma linguagem:

- Interpretada;
- Imperativa;

- Orientada a Objetos;
- Funcional;
- Tipagem dinâmica e forte;

Durante o processo de desenvolvimento do servidor na placa Raspberry Pi, optou-se, em um primeiro momento, pela implementação do servidor na linguagem de programação Node.js[1], porém, em virtude de poucos recursos da linguagem com GPIO e de pouca experiência com a linguagem de programação Node.js, essa opção foi descartada. Então, foi visto que a linguagem Python seria a melhor opção, visto que tem uma sintaxe simples, um farto número de bibliotecas disponíveis para a manipulação dos componentes da placa Raspberry Pi e um vasto número de guias para auxiliar no processo de desenvolvimento.

4.2.3.1 FLASK

Flask é um mini framework da linguagem de programação Python para criação de microserviços utilizando a arquitetura RESTful. Dessa forma, Flask segue dois pilares que são importantes:

- WerkZeug, padrão de desenvolvimento WSGI, que especifica a interface de aplicações Python com web service;
- Good Intentions, que segue um padrão de código Pythonico de alta qualidade por ter uma boa legibilidade e sem muitas configurações desnecessárias como segue na Figura 4.

[1] Node.js - https://nodejs.org/en/

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

Figura 4 – Microserviço Flask
Flask documentação - http://flask.pocoo.org/

No desenvolvimento do trabalho, foram criados quatro microserviços com Flask que são: gerenciamento do relay (interruptor), verificação da temperatura, verificação da umidade e o último serviço que gerencia a execução ou interrupção da música.

Para facilitar o processo de desenvolvimento, Flask tem embutido em sua biblioteca um servidor de aplicação, porém, para fins de um servidor com IP público sua documentação recomenda um servidor de aplicação separado.

> Se você executar o servidor, perceberá que o servidor só é acessível a partir do seu próprio computador, e não de qualquer outro na rede. Este é o padrão porque, no modo de depuração, um usuário do aplicativo pode executar código Python arbitrário em seu computador. (RONACHER, 2018)

Para o servidor implementado junto a este trabalho, foi decidido utilizar o servidor de aplicação que já é embutido na biblioteca, pois, não há a necessidade de utilização de um IP público no servidor da Raspberry Pi.

4.2.4 JAVA

A linguagem de programação Java surgiu no ano de 1991 na fundação Sun Microsytems, sob o comando do cientista da computação James Gosling. No primeiro momento, era parte do projeto chamado de Green Project, que tinha o objetivo de possibilitar a comunicação de equipamentos eletrônicos e eletrodomésticos. Posteriormente, foi realizado o aprimoramento da linguagem de programação para se adaptar a internet, no ano de 1995, que por sequência ganhou notoriedade.

Um dos grandes diferenciais da plataforma Java é sua execução, que é realizada sobre sua JVM (Java Virtual Machine), permitindo sua interpretação sobre qualquer tipo de hardware. Em 2008, a linguagem de programação Java foi comprada pela organização Oracle Corporation que é responsável pelo lançamento de novas *releases.*

A linguagem de programação Java tem em suas características ser uma linguagem:

- Orientada a Objetos;
- Imperativa;
- Compilada para execução na JVM;
- Tipagem estática e forte;
- Funcional;

4.3 MODELAGEM

Para o processo de desenvolvimento são apresentados os diagramas utilizando UML[2] - uma linguagem padrão para modelagem de projetos de software. Para a proposta deste trabalho, que consiste da criação de componentes do App Inventor junto a dispositivos IoT, foram elaborados diagramas de classe tanto para descrever os componentes criados para o App Inventor, como para o servidor de aplicação da placa Raspberry Pi. Dando prosseguimento a modelagem, é apresentado um diagrama de sequência que vai demonstrar a comunicação de um componente App Inventor com a placa Raspberry Pi.

4.3.1 DIAGRAMA DE CLASSE

Para os componentes criados na ferramenta App Inventor foi utilizado como base o componente Web, que é um componente que realiza requisições utilizando o

[2] UML - https://www.uml.org/

protocolo HTTP. Dessa maneira, as classes que representam os componentes criados são ao total quatro, sendo: Interruptor, Temperatura, Umidade e Música.

4.3.1.1 COMPONENTE INTERRUPTOR

Para o componente Interruptor, foram criados quatro métodos novos, além dos já existentes na classe Web. Os métodos criados foram: EnderecoServidor, Gpio, Ligar e Desligar. Todos os métodos da classe Interruptor estão no diagrama de classe que está representado na Figura 5.

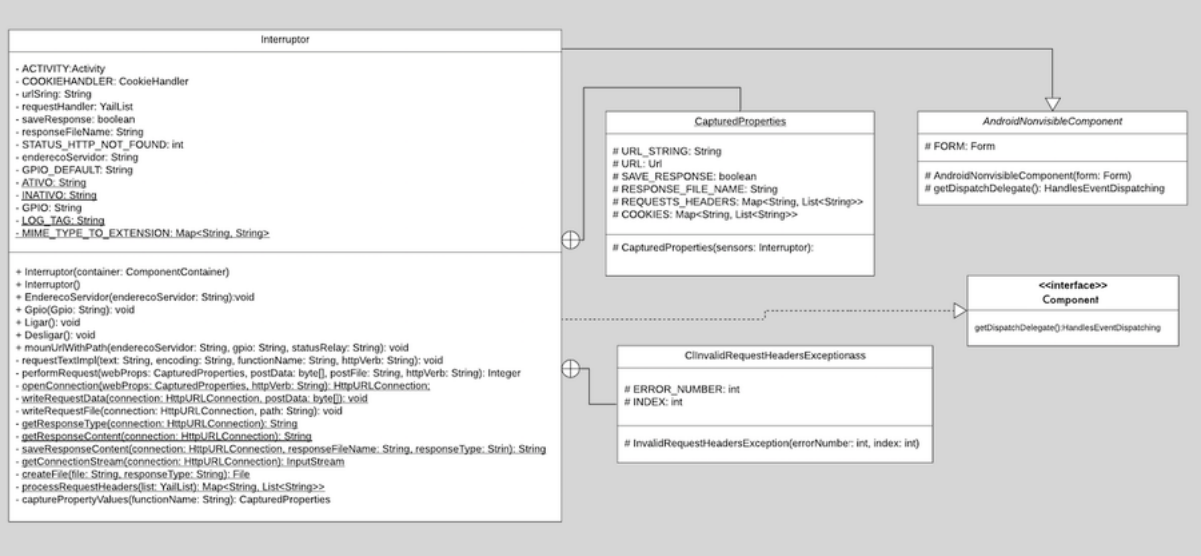

Figura 5 – Diagrama de classe componente Interruptor
elaborado pelo autor

4.3.1.2 COMPONENTE TEMPERATURA E UMIDADE

Os componentes Temperatura e Umidade foram criados contendo a mesma estrutura de métodos, porém, realizam requisições para serviços REST diferentes. Dessa forma, quatro métodos novos foram desenvolvidos nas classes Temperatura e Umidade, além dos já existentes na classe Web, sendo os novos métodos: EnderecoServidor, Gpio, Obter e Capturar. Todos os métodos das classes Temperatura e Umidade estão nos diagramas de classe representados respectivamente na Figura 6 e na Figura 7.

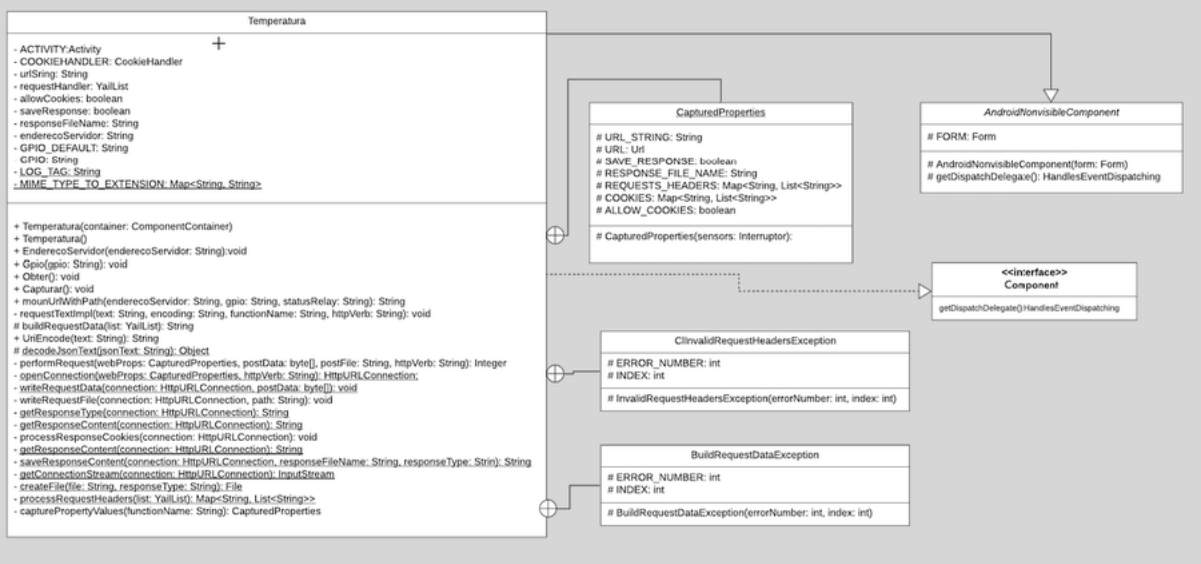

Figura 6 – Diagrama de classe componente Temperatura
elaborado pelo autor

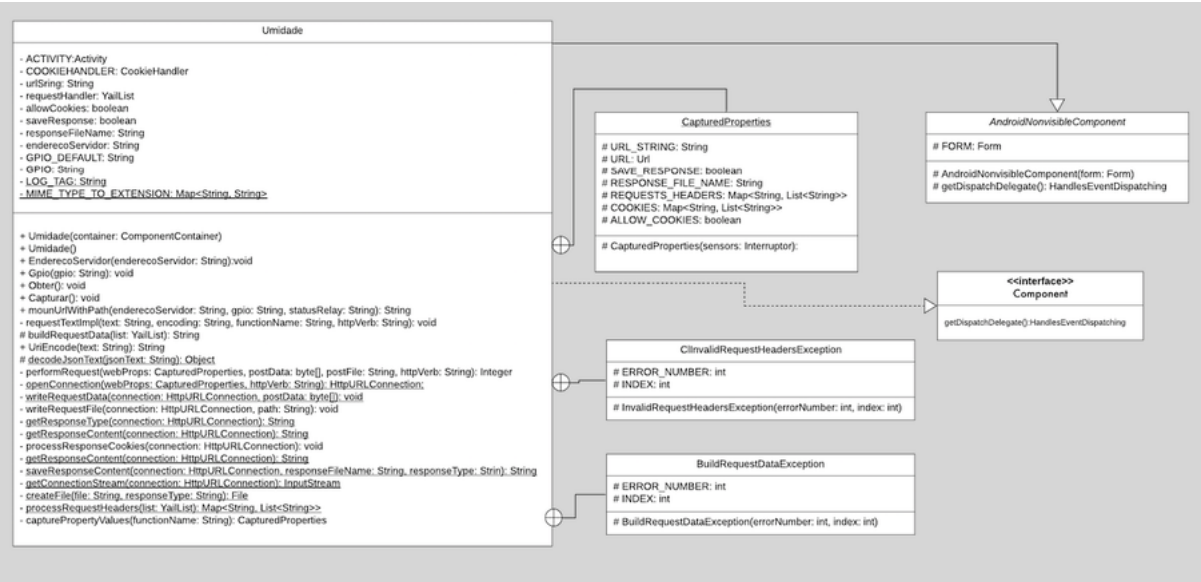

Figura 7 – Diagrama de classe componente Umidade
elaborado pelo autor

4.3.1.3 COMPONENTE MÚSICA

Para o componente Musica foram criados seis métodos novos, além dos já existentes na classe Web. Os métodos criados foram: EnderecoServidor, TocarAudio1, TocarAudio2, TocarAudio3, TocarAudio4 e PararAudio. Todos os métodos da classe Musica estão no diagrama de classe que está representado na Figura 8

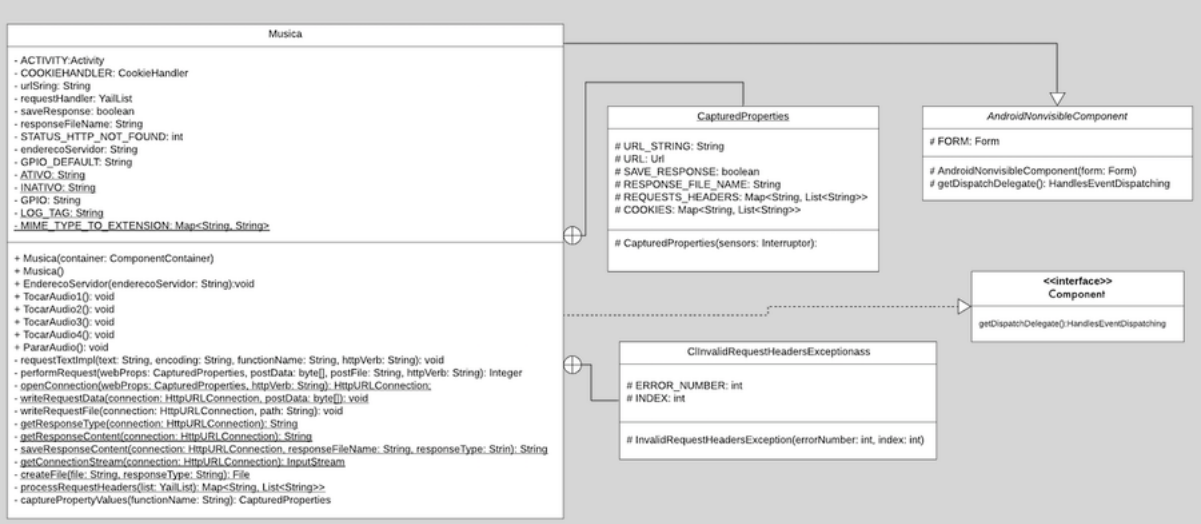

Figura 8 – Diagrama de classe componente Musica
elaborado pelo autor

4.3.2 DIAGRAMA DE SEQUÊNCIA

Por fim, na ilustração da comunicação via protocolo HTTP do componente junto ao servidor de aplicação, foi criado um diagrama de sequência para apenas um componente proposto pelo trabalho - que vai simular a comunicação dos outros componentes implementados pelo presente trabalho. A Figura 9 apresenta o diagrama de sequência das trocas de mensagens entre o componente do App Inventor e o servidor rodando na placa Raspberry Pi.

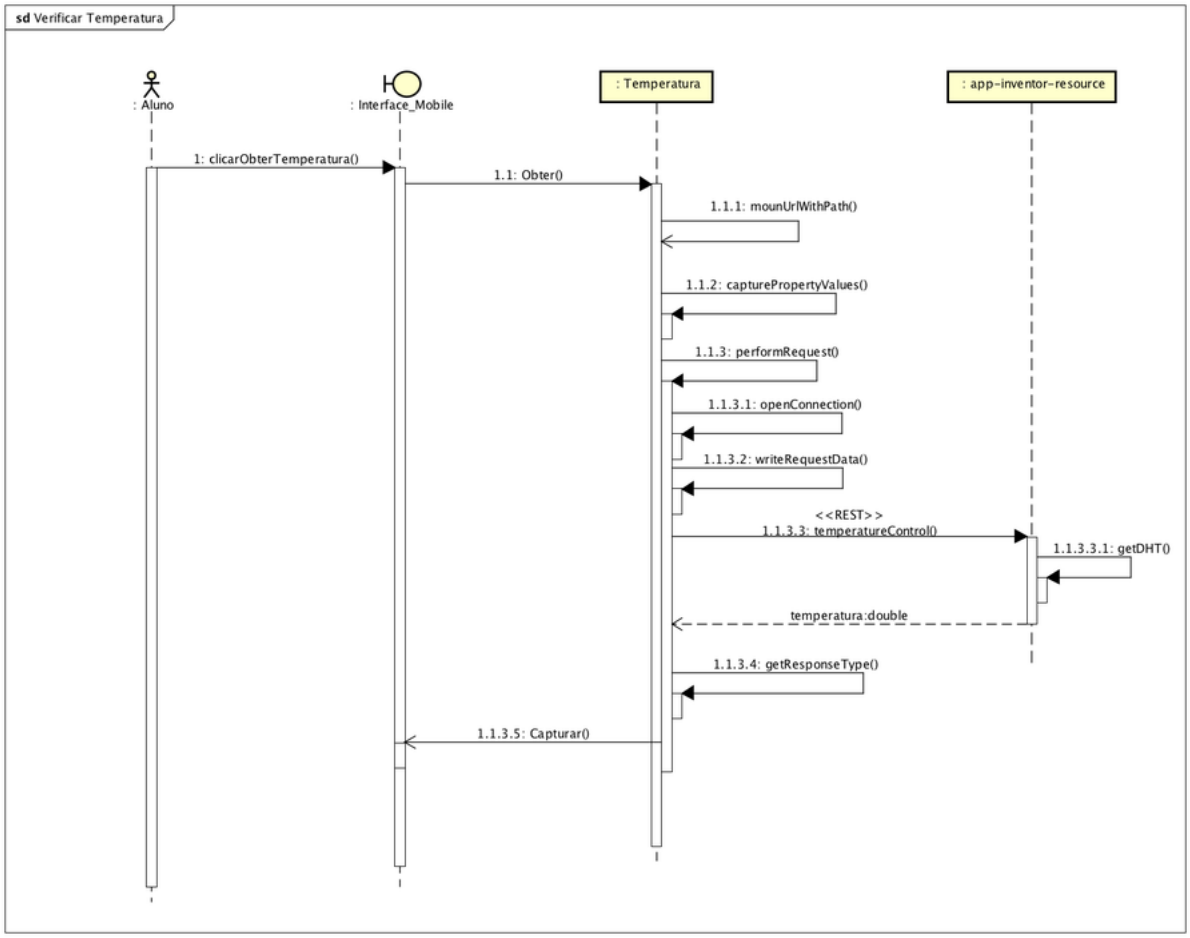

Figura 9 – Diagrama de sequência componente Temperatura
elaborado pelo autor

4.4 DESENVOLVIMENTO

Nesse trabalho foi criada uma solução de implementação para o ensino de computação utilizando dispositivo IoT com componentes extensíveis da ferramenta App Inventor, para o ensino mais lúdico e interessante no contexto de uma criança ou jovem que não tem experiência com linguagem de programação.

O trabalho foi desenvolvido utilizando a ferramenta App inventor para a criação de novos componentes extensíveis, os quais interagem com o servidor que foi implementado no minicomputador Raspberry Pi.

Esta seção apresenta as etapas do desenvolvimento de uma aplicação demostrativa, utilizando o componente desenvolvido na ferramenta App Inventor, e sua comunicação com o serviço - implementado na placa Raspberry Pi - para ligar ou desligar um relay.

4.4.1 APLICAÇÃO NO CELULAR

A primeira etapa consiste na construção da aplicação mobile na ferramenta App Inventor, onde são definidos os componentes da interface gráfica da aplicação. Os componentes adicionados são utilizados na grande maioria das vezes para interagir com os blocos da programação visual. Neste exemplo, como é demostrado na Figura 10, foi construído um *layout* onde o usuário informa qual instrução será transmitida para o dispositivo IoT. Para isso, foram utilizados dois componentes de Interruptor, que podem passar os comandos de ligar ou desligar seu respectivo relay.

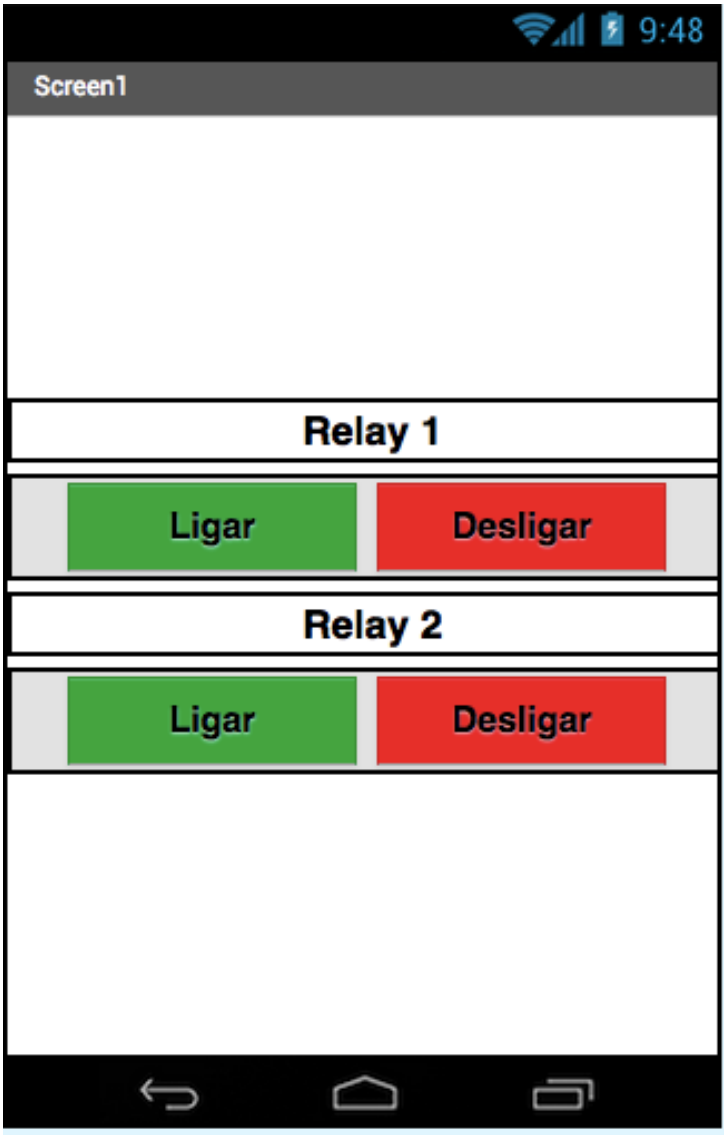

Figura 10 – Aplicação no celular
elaborado pelo autor

4.4.2 BLOCOS APP INVENTOR

Como segundo passo para o desenvolvimento da aplicação utilizando os componentes desenvolvidos neste trabalho, foram montados os blocos do componente Interruptor - desenvolvidos no presente trabalho - que neste exemplo são executados através de eventos de clique, disparados por uma ação no botão da interface gráfica.

No exemplo de programação em bloco descrito na Figura 11 são vistos quatro blocos, e cada bloco demonstra uma ação realizada pela tela, sendo eles: ligar ou desligar determinado relay. Os componentes que realizam as chamadas HTTP(s),

são responsáveis por passar as informações para a placa. Então, o servidor recebe uma *request* que foi gerada a partir do evento da interface gráfica, e que será interpretada e transformada numa instrução para a placa Raspberry Pi. Portanto, caso o evento de clique executado pelo aplicativo móvel for ligar, o circuito do relay será fechado ou, caso o evento for de desligar, será transmitida para placa a ação para abrir o circuito do relay.

O componente Interruptor recebe dois parâmetros de configuração, que são:

- EnderecoServidor - Que informa onde a placa se encontra na rede TCP/IP;
- Gpio - que é a porta digital onde está conectado o relay;

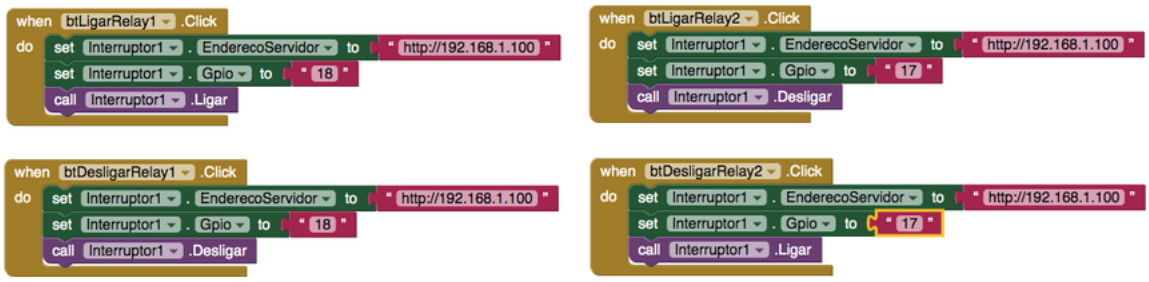

Figura 11 – Blocos App Inventor
elaborado pelo autor

4.4.3 DISPOSITIVO IOT

Na terceira etapa do desenvolvimento do aplicativo demostrativo, os blocos do componente Interruptor - da ferramenta App Inventor - que foram desenvolvidos, realizam a transmissão das informações. Os dados que são transmitidos através da rede TCP/IP pelo protocolo HTTP, e para tratar a *request* enviada pelo componente do App Inventor foi utilizada a arquitetura RESTful, que vai tratar este dado enviado e transformá-la numa instrução para a placa Raspberry Pi.

O minicomputador Raspberry Pi contém alguns componentes para que em seu todo possa atender as necessidades do trabalho em questão. Portando, junto a placa Raspberry Pi foram acoplados dois componentes que auxiliam neste exemplo,

como mostrado na Figura 12. Para facilitar a localização da placa Raspberry Pi na rede TCP/IP, foi conectada uma placa LCD que informa o IP local da placa na rede - esta informação é necessária para configuração dos parâmetros no componente Interruptor, quando utilizado na ferramenta App Inventor. O componente relay também foi adicionado na placa Raspberry Pi, para demostrar a interação do componente (extensão) desenvolvido na ferramenta App Inventor.

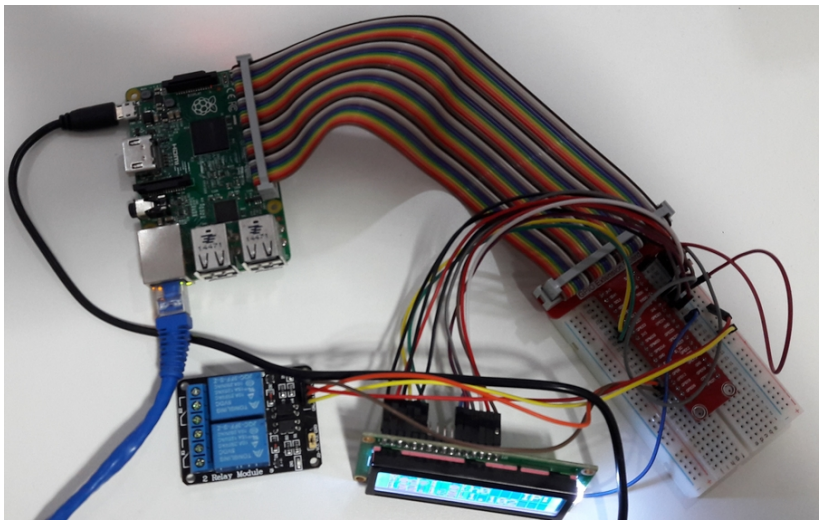

Figura 12 – Dispositivo IoT
elaborado pelo autor

4.4.4 IMPLEMENTAÇÃO SERVIDOR RASPBERRY PI

Nesta seção será abordada a implementação do servidor da placa Raspberry Pi. Portanto, será apresentada a parte de codificação com explicação de algumas decisões tomadas. Será indicada tanto a etapa de desenvolvimento do servidor de aplicação, como detalhamento do processo de inicialização do servidor na placa Raspberry Pi.

Para a implementação do servidor foi adotado o padrão para Web Services chamado RESTful - bem consolidado no mercado de desenvolvimento - visto que sua codificação é mais direta que o padrão SOAP - utilizado em Web Services de aplicações antigas.

Portanto, para a implementação do lado do servidor foram desenvolvidos microserviços - que é uma das premissas da arquitetura RESTFul. Dessa maneira, o servidor abstrai toda a comunicação realizada com o hardware da placa Raspberry Pi.

4.4.4.1 INICIALIZAÇÃO DISPLAY DE LED E SERVIDOR REST

Junto ao processo de inicialização do S.O. Raspbian é possível realizar a execução de arquivos shell (.sh), que podem realizar chamadas de códigos de linguagem de programação. Desse modo, junto a inicialização do S.O. é executado um arquivo shell que realiza a inicialização do servidor REST e do Display de LED - para informar o IP da placa - como mostra a Figura 13.

```
#!/bin/bash

python /home/pi/git/AppInventorCnEIoT/app-inventor-resource.py &

python /home/pi/git/AppInventorCnEIoT/display-ip.py &
```

Figura 13 – Arquivo de Boot da placa Raspberry Pi
elaborado pelo autor

4.4.4.2 ENDEREÇO IP PLACA RASPBERRY PI

O processo de comunicação dos componentes App Inventor com os sensores acoplados a placa ocorre via protocolo HTTP, sendo necessária a informação do IP da placa na configuração do componente da ferramenta App Inventor. Portanto, foi desenvolvido um algoritmo que escreve o endereço IP no Display de LED, como mostrado na Figura 14. Este código é executado quando o S.O. Raspbian é inicializado, como já explanado na seção 4.4.4.1.

```
#!/usr/bin/python
# Example using a character LCD connected to a Raspberry Pi or BeagleBone Black.
import time

import Adafruit_CharLCD as LCD
import Adafruit_GPIO as GPIO

from subprocess import *
from datetime import datetime
import socket
import fcntl
import struct

def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])


# Raspberry Pi pin configuration:
lcd_rs        = 27  # Note this might need to be changed to 21 for older revision Pi's.
lcd_en        = 22
lcd_d4        = 25
lcd_d5        = 24
lcd_d6        = 23
lcd_d7        = 18
lcd_backlight = 4

# Define LCD column and row size for 16x2 LCD.
lcd_columns = 15
lcd_rows    = 1

# Initialize the LCD using the pins above.
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_backlight)

# Print a two line message
lcd.message('Inicializando...!')

# Wait 25 seconds
time.sleep(25.0)

# Demo showing the cursor.
lcd.clear()

ipaddr = get_ip_address('eth0')
# print(ipaddr)
lcd.message('Rede: eth0   IP: \n')
lcd.message(ipaddr)
```

Figura 14 – IP da placa Raspberry Pi
https://github.com/ComputacaoNaEscola/Servidor-IoT-para-AppInventor/blob/master/ip_lcd.py

4.4.4.3 SERVIÇO RELAY

Para o acionamento do componente relay que foi desenvolvido neste projeto, através do microserviço REST, que recebe uma requisição HTTP para fechar ou abrir o circuito do relay, o serviço recebe dois parâmetros no path: portDigital e status como descrito na Tabela 10. Desse modo, a comunicação com a placa Raspberry Pi é intermediada através da GPIO[3], uma biblioteca que manda uma corrente elétrica para

[3] GPIO - https://github.com/adafruit/Adafruit_Python_GPIO

o acionamento das portas digitais da placa Raspberry Pi.

Como visto na Figura 15, o código tem um tratamento de *except* para KeyboardInterrupt - gerada quando usuário pressiona a tecla de interrupção (normalmente Control-C ou Delete) quando o código esta sendo executado, fazendo com que a requisição que foi realizada para a placa retorne o status code 400 (*Bad Request*) - quando não foi possível processar toda a requisição por alguma falha. Portanto, pode ser visto que o código não contém grande complexidade, tendo em vista que grande parte do código é para configurar a comunicação da biblioteca GPIO com a placa.

Parâmetro	**Tipo**	**Descrição**
portDigital	String	Identificar em qual pino digital o componente relay está conectado.
status	String	Comando para abrir ou fechar o circuito do relay

Tabela 10 – Parâmetro do serviço relay

```
15  @app.route('/relay/<int:portDigital>/status/<string:status>')
16  def relayControl(portDigital, status):
17      resp = Response();
18      try:
19          portDigital = int(portDigital)
20          GPIO.setmode(GPIO.BCM)
21          GPIO.setwarnings(False)
22          GPIO.setup(portDigital, GPIO.OUT)
23          GPIO.output(portDigital, GPIO.HIGH)
24          circuitRelay(status, portDigital)
25          resp.status_code = 200
26      except KeyboardInterrupt:
27          resp.status_code = 400
28      return resp
```

Figura 15 – Serviço REST relay
elaborado pelo autor

4.4.4.4 SERVIÇO TEMPERATURA

O serviço de temperatura que é mostrado na Figura 16 é chamado através de uma requisição HTTP e no path do serviço é passado somente um parâmetro: portDigital, onde o sensor DHT está conectado, como segue detalhado na Tabela 11. Para a resposta da requisição HTTP feita para o serviço, é adicionada ao corpo da resposta a temperatura em grau Celsius (℃) que foi capturada pelo sensor DHT.

Parâmetro	**Tipo**	**Descrição**
portDigital	String	Identificar em qual pino digital o componente temperatura está conectado.

Tabela 11 – Parâmetro do serviço temperatura

```
30 @app.route('/temperatura/<int:portDigital>')
31 def temperatureControl(portDigital):
32     resp = Response()
33     try:
34         portDigital = int(portDigital)
35         DHT = getDHT(portDigital)
36         resp = Response(str(DHT[TEMPERATURA]), status=200, mimetype='application/json')
37     except KeyboardInterrupt:
38         resp.status_code = 400
39     return resp
```

Figura 16 – Serviço REST temperatura
elaborado pelo autor

4.4.4.5 SERVIÇO UMIDADE

O serviço de umidade é semelhante ao serviço de temperatura, como é mostrado na Figura 17. Sua comunicação é feita através de uma requisição HTTP e no path do serviço também é passado o parâmetro portDigital, onde vai informar a porta digital do sensor DHT - o sensor coleta tando informação de temperatura como de umidade - como está detalhado na Tabela 12 . A requisição realizada coleta informação de umidade que, posteriormente, é escrita no corpo da resposta HTTP.

Parâmetro	Tipo	Descrição
portDigital	String	Identificar em qual pino digital o componente umidade está conectado.

Tabela 12 – Parâmetro do serviço umidade

```
@app.route('/umidade/<int:portDigital>')
def umidadeControl(portDigital):
    resp = Response()
    try:
        portDigital = int(portDigital)
        DHT = getDHT(portDigital)
        resp = Response(str(DHT[UMIDADE]), status=200, mimetype='application/json')
    except KeyboardInterrupt:
        resp.status_code = 400
    return resp
```

Figura 17 – Serviço REST umidade
elaborado pelo autor

4.4.4.6 SERVIÇO MÚSICA

O serviço de música também utiliza a comunicação via protocolo HTTP, porém, este serviço foi separado em dois, utilizando o padrão adotado pela arquitetura REST para microserviços. Na implementação deste serviço foi necessária a instalação de um player de música, para fazer o carregamento e execução/interrupção do áudio. Portanto, para este trabalho foi escolhido omxplayer[4], uma biblioteca que foi criada para execução/interrupção de músicas no S.O. Raspbian - Sistema O peracional da Raspberry Pi.

- Serviço execução de música

Na implementação do serviço, como mostra a Figura 18, que realiza a execução da música, foi criado um diretório na placa Raspberry Pi, que contém 4 músicas que podem ser executadas. No path do serviço é passado o parâmetro sound, informando qual a música que será executada na requisição que foi feita para o serviço, como

[4] Omxplayer - https://omxplayer.sconde.net/

está detalhado na Tabela 13. O padrão de nomenclatura adotado para os arquivos de música, contidos dentro da pasta, são: audio1, audio2, audio3 e audio4. O formato escolhido para as músicas foi o padrão MP3, que é suportado pela biblioteca omxplayer.

Parâmetro	**Tipo**	**Descrição**
sound	String	Identificar qual a música que será carregada do diretório para ser executada.

Tabela 13 – Parâmetro do serviço play musica

```
@app.route('/sound/<string:sound>/play')
def playSound(sound):
    resp = Response()
    try:
        subprocess.call('omxplayer --no-keys /home/pi/git/AppInventorCnEIoT/sounds/' + sound + '&', shell=True)
        resp = Response("Play sucesso",status=200, mimetype='application/json')
    except KeyboardInterrupt:
        resp.status_code = 400
    return resp
```

Figura 18 – Serviço REST execução de música
elaborado pelo autor

- Serviço parar música

Na implementação do serviço de interrupção, como mostra a Figura 19, é realizado o fim do processo da biblioteca omxplayer. Para finalizar o processo é executado o comando “pkill” - nativo das distribuições linux - e como parâmetro é informado o nome do processo que será finalizado, que neste caso é “omxplayer”. A mesma requisição HTTP é respondida com um status code 200 e no corpo o texto “Stop sucesso”.

```
@app.route('/sound/stop')
def stopSound():
    resp = Response()
    try:
        subprocess.call('pkill omxplayer', shell=True)
        resp = Response("Stop sucesso",status=200, mimetype='application/json')
    except KeyboardInterrupt:
        resp.status_code = 400
    return resp
```

Figura 19 – Serviço REST parar música
elaborado pelo autor

4.4.5 COMPONENTES APP INVENTOR

A ferramenta App Inventor segue um modelo de desenvolvimento “Open Source”, portanto, a implementação de novos componentes para a ferramenta torna-se muito simples e direto. Esta seção aborda os componentes que foram criados para na ferramenta, descrevendo a estrutura de métodos que foram desenvolvidos. Durante a implementação dos componentes para a ferramenta App Inventor algumas decisões foram tomadas, para o melhor processo de desenvolvimento possível.

O App Inventor é muito amplo na quantidade de componentes que já são embutidos na sua ferramenta, inclusive componentes que são muito úteis para o cenário da área de programação, como componentes que realizam conexão com Banco de Dados ou Bluetooth, e até mesmo que realizam chamadas utilizando o protocolo HTTP, muito útil para o cotidiano da programação. Para o desenvolvimento dos componentes novos, propostos por este trabalho, foi utilizado como base o componente nativo do App Inventor, que se chama Web - componente que realiza requisições utilizando o protocolo HTTP. Utilizando esse componente não foi necessário a implementação de métodos que realizam chamadas HTTP, economizando o tempo de desenvolvimento para os componentes que estão no presente trabalho.

4.4.5.1 COMPONENTE INTERRUPTOR

Simulando o funcionamento de um interruptor comum, utilizado na grande maioria dos ambientes domésticos da atualidade, foi criado o componente “Interruptor” - com o objetivo de abrir ou fechar um circuito elétrico. Entretanto, para aplicação desse projeto, o componente funciona como um interruptor que liga ou desliga uma lâmpada.

O componente criado é simples, contendo apenas dois blocos de funções dentro de sua estrutura: um para “Ligar” e outro para “Desligar” a lâmpada, como mostrado na Figura 20. Porém, dentro do componente é necessário realizar a configuração de alguns parâmetros, que são: endereço do servidor e gpio.

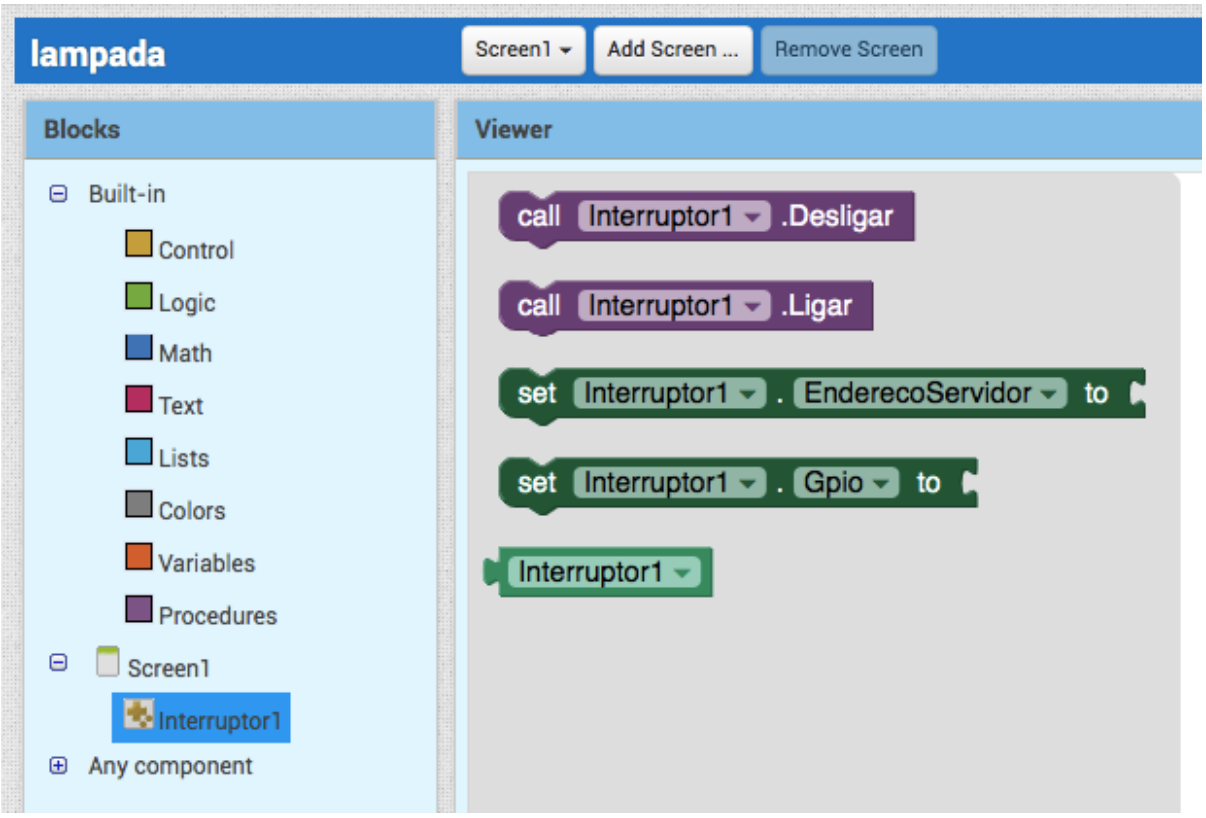

Figura 20 – Componente Interruptor
elaborado pelo autor

A configuração definitiva, tando do endereço do servidor como do Gpio, pode ser feita pela área de "*Designer*" do App Inventor, como mostrado na Figura 21. Dessa forma, todas as requisições feitas pelo componente, com a função ligar ou desligar, utilizam os parâmetros que são configurados na área de "*Designer*".

Figura 21 – Configuração dos parâmetros componente Interruptor
elaborado pelo autor

A classe Interruptor, criada junto ao projeto App Inventor, utilizou como base a classe nativa Web do próprio projeto, portanto, não foi necessária a criação de muitos métodos. Para a implementação do componente foram criados dois métodos que representam os blocos Ligar e Desligar, como é mostrado na Figura 22. Os métodos realizam chamadas HTTP para o serviços que foram implementados na placa Raspberry Pi, como mostra a seção 4.4.4.3 .

```
205⊖    @SimpleFunction
206     public void Ligar() {
207         if (this.GPIO == null || this.GPIO.isEmpty()) {
208             this.GPIO = this.GPIO_DEFAULT;
209         }
210         String urlWithPath = mounUrlWithPath(this.enderecoServidor, this.GPIO, ATIVO);
211         this.urlString = urlWithPath;
212         final CapturedProperties webProps = capturePropertyValues("Relay");
213         if (this.enderecoServidor == "" || this.GPIO == "") {
214             return;
215         }
216
217⊕         AsynchUtil.runAsynchronously(new Runnable() {
234     }
235
236⊖     @SimpleFunction
237     public void Desligar() {
238         if (this.GPIO == null || this.GPIO.isEmpty()) {
239             this.GPIO = this.GPIO_DEFAULT;
240         }
241         String urlWithPath = mounUrlWithPath(this.enderecoServidor, this.GPIO, INATIVO);
242         this.urlString = urlWithPath;
243         final CapturedProperties webProps = capturePropertyValues("Relay");
244         if (this.enderecoServidor == "" || this.GPIO == "") {
245             return;
246         }
247
248⊕         AsynchUtil.runAsynchronously(new Runnable() {
265     }
```

Figura 22 – Classe Interruptor
elaborado pelo autor

4.4.5.2 COMPONENTE TEMPERATURA

No atual cenário é normal a consulta para verificar a temperatura dentro de um ambiente, tanto doméstico como urbano. Dessa forma, foi desenvolvido o componente de temperatura para realizar requisições para a placa Raspberry Pi, que se comunica com o sensor de temperatura DHT11.

O componente foi criado utilizando como base outro componente, chamado Web, que é nativo da ferramenta App Inventor. Dessa maneira, para o desenvolvimento do componente temperatura, não foi necessário a codificação de chamadas HTTP. Porém,

foram encontradas algumas dificuldades para o entendimento na captura das respostas de requisições HTTP, pelo motivo de as requisições serem tratadas de maneira assíncrona. Foram criados dois blocos de função no componente, como mostrado na Figura 23, o primeiro bloco que é chamado Obter - realiza a chamada assíncrona - e o segundo bloco chamado de Capturar - recebe as informações que foram solicitadas pelo bloco Obter. Como já mostrado no componente lâmpada, é necessário realizar a configuração de dois parâmetros para o componente, que são: endereço do servidor e a gpio.

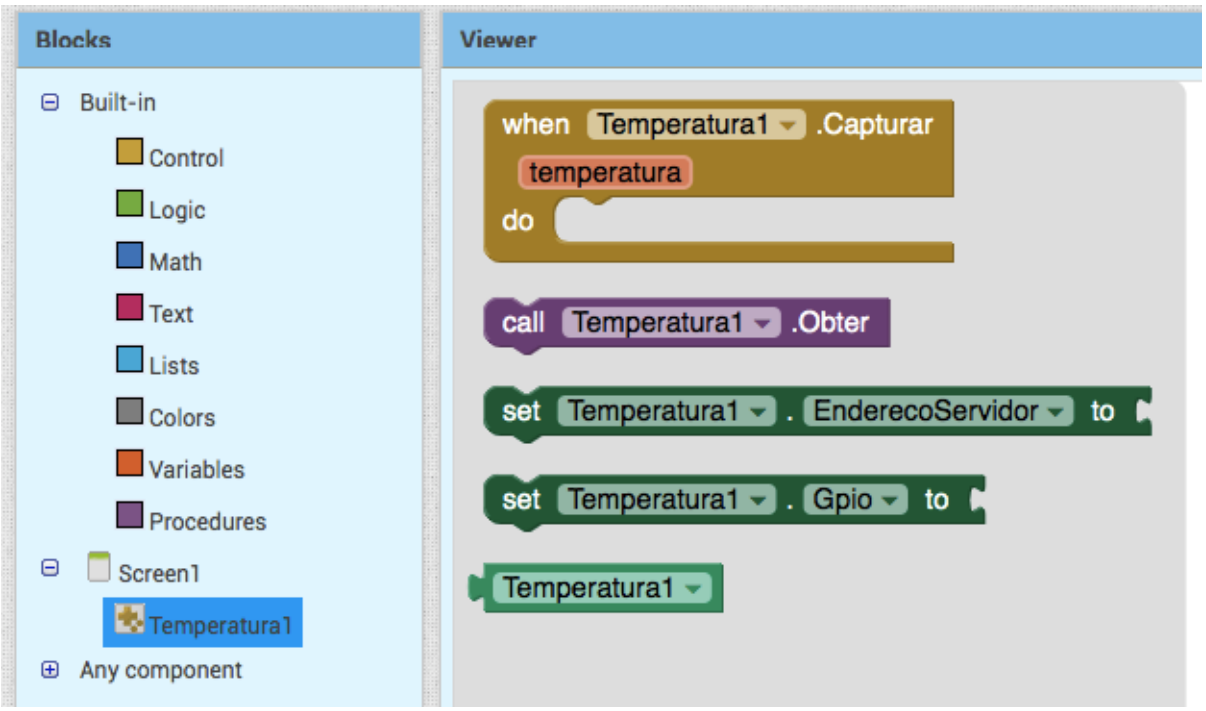

Figura 23 – Componente Temperatura
elaborado pelo autor

A configuração dos parâmetros do componente temperatura pode ser feita de maneira definitiva na área de "*Designer*", como mostrado na Figura 24. Dessa forma, todas as requisições que são realizadas pelo componente utilizam o endereço do servidor e a gpio que foi configurada na área de "*Designer*" do App Inventor.

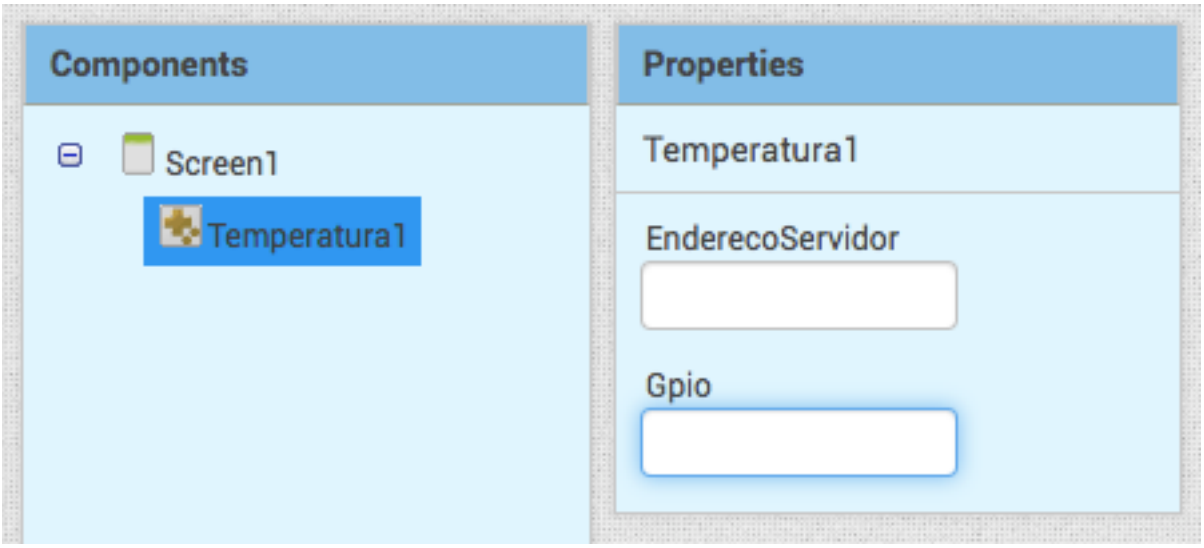

Figura 24 – Configuração dos parâmetros componente Temperatura
elaborado pelo autor

Para a classe Temperatura, que foi adicionada ao projeto App Inventor, foram implementados dois métodos que representam respectivamente as funções da tela, como mostrado na Figura 25: o método Obter - que realiza uma requisição HTTP para o serviço da placa Raspberry Pi, como comentado na seção 4.4.4.4, para solicitar a temperatura; e o método Capturar - que recebe as informações que foram solicitadas pelo bloco Obter e retornadas para o componente temperatura.

```
229⊕     * Performs an HTTP GET request using the Url property and retrieves the
237⊖    @SimpleFunction
238     public void Obter() {
239         if (this.GPIO == null || this.GPIO.isEmpty()) {
240             this.GPIO = this.GPIO_DEFAULT;
241         }
242         String urlWithPath = mounUrlWithPath(this.enderecoServidor, this.GPIO);
243         this.urlString = urlWithPath;
244         final CapturedProperties webProps = capturePropertyValues("Tempetaura");
245         if (this.enderecoServidor == "" || this.GPIO == "") {
246             return;
247         }
248
249⊕         AsynchUtil.runAsynchronously(new Runnable() {
263     }
264
266⊕     * Event indicating that a request has finished.
277⊖    @SimpleEvent
278     public void Capturar(String url, int responseCode, String responseType, String temperatura) {
279         // invoke the application's "GotText" event handler.
280         EventDispatcher.dispatchEvent(this, "Capturar", url, responseCode, responseCode, temperatura);
281     }
```

Figura 25 – Classe Temperatura
elaborado pelo autor

4.4.5.3 COMPONENTE UMIDADE

Sensores para controle de ambiente são comuns atualmente, como já citado na seção 4.4.5.2, e não seria diferente para o sensor de umidade. A aplicação do sensor de controle de umidade é válida em contextos que necessitam de controle mais refinado do ambiente, como por exemplo, a conservação de algumas bebidas que são sensíveis a umidade. Portando, esta seção vai detalhar a criação do componente na ferramenta App Inventor, com a demonstração de parte do código da ferramenta.

A extensão criada foi desenvolvida utilizando como base o componente nativo Web, da ferramenta App Inventor. O desenvolvimento não envolveu muita complexidade, pois a estrutura do código foi semelhante a implementação do componente temperatura. Desse modo, o componente quando executado realiza uma chamada HTTP para o serviço de umidade na placa Raspberry Pi, porém, a resposta é tratada de maneira assíncrona pela arquitetura da ferramenta App Inventor. Sendo assim, para o tratamento do assíncrono na requisição de umidade, foi necessária a criação de dois blocos dentro do componente, como mostrado na Figura 26. O componente umidade também necessita da configuração de dois parâmetros, que são: endereço do servidor e de gpio.

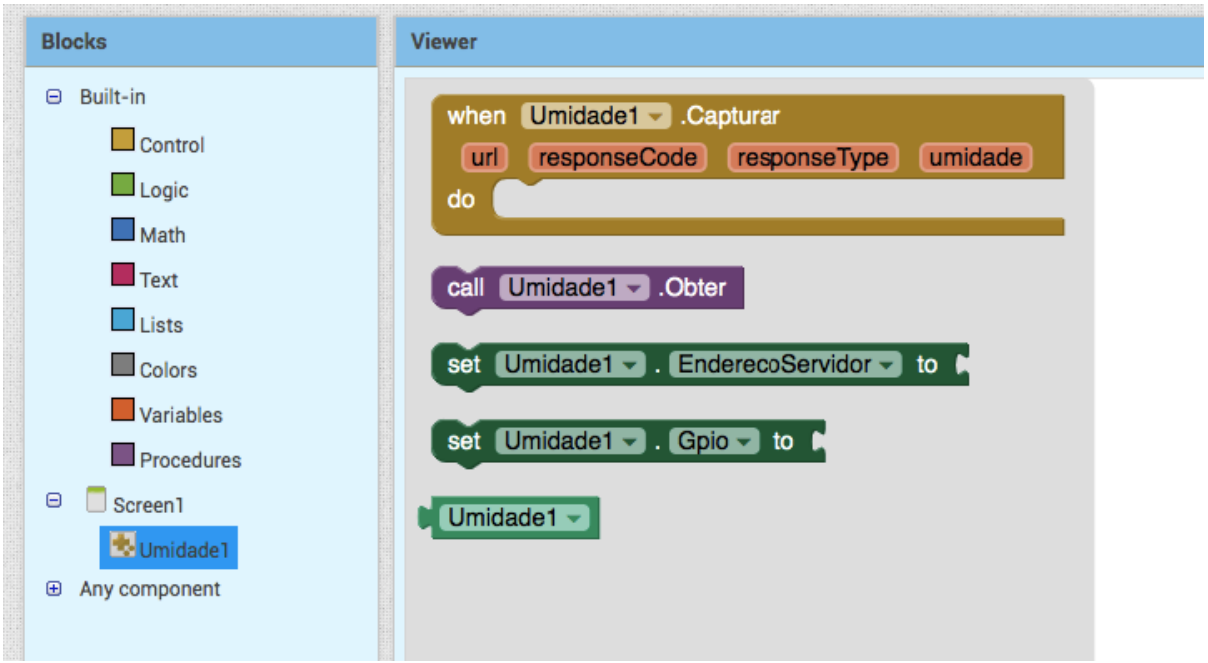

Figura 26 – Componente Umidade
elaborado pelo autor

A configuração dos parâmetros é semelhante a dos outros componentes já citados, podendo ser realizada de maneira única, sem a necessidade de configuração para cada bloco. A configuração é realizada na área de "*Designer*" do App Inventor, como mostrado na Figura 27.

Figura 27 – Configuração dos parâmetros componente Umidade
elaborado pelo autor

A classe Umidade, que foi implementada junto a ferramenta App Inventor, é muito semelhante à classe Temperatura. Portanto, contém dois métodos que representam respectivamente os blocos da tela, que são: Capturar e Obter, como mostra a Figura 28. Dessa forma, a execução dos métodos funciona da seguinte forma, o método Obter realiza uma requisição assíncrona para o serviço de umidade, como demostrado na seção 4.4.4.5. Por sequência, o método Capturar recebe as informações que foram solicitadas pelo método Obter, para em seguida ter a possibilidade de informá-las na tela da aplicação que estiver sendo desenvolvida.

```
228⊕     * Performs an HTTP GET request using the Url property and retrieves the..
236⊖    @SimpleFunction
237     public void Obter() {
238         if (this.GPIO == null || this.GPIO.isEmpty()) {
239             this.GPIO = this.GPIO_DEFAULT;
240         }
241         String urlWithPath = mounUrlWithPath(this.enderecoServidor, this.GPIO);
242         this.urlString = urlWithPath;
243         final CapturedProperties webProps = capturePropertyValues("Umidade");
244         if (this.enderecoServidor == "" || this.GPIO == "") {
245             return;
246         }
247
248⊕         AsynchUtil.runAsynchronously(new Runnable() {..
261     }
262
264⊕     * Event indicating that a request has finished...
275⊖    @SimpleEvent
276     public void Capturar(String url, int responseCode, String responseType, String umidade) {
277         // invoke the application's "GotText" event handler.
278         EventDispatcher.dispatchEvent(this, "Capturar", url, responseCode, responseType, umidade);
279     }
280
```

Figura 28 – Classe Umidade
elaborado pelo autor

4.4.5.4 COMPONENTE MÚSICA

O controle de música por meio aplicativos de celular é muito comum. Desse modo, foi implementado o componente música, junto à ferramenta App Inventor, para possibilitar assim a programação utilizando o conceito de desenvolvimento em bloco.

O desenvolvimento do componente música, seguindo o padrão de implementação dos outros componentes, utilizou como base o componente Web. Dessa maneira, a ferramenta App Inventor contém na área de programação a extensão que foi desenvolvida, composta por 5 blocos de funções, como mostra a Figura 29. Os blocos de função TocarAudio1 a TocarAudio4 realizam chamadas de serviço para a placa Rasbpberry Pi, onde serão executadas as músicas que estão pré estabelecidas no diretório da placa. Porém, além de realizar a execução das músicas, é possível também interromper a música que está sendo executada. O bloco ParарAudio segue o mesmo procedimento, realizando uma chamada de serviço para a placa, onde é realizada a interrupção do processo que está executando a música em questão. Para executar o componente música é necessário configurar o parâmetro endereço do servidor.

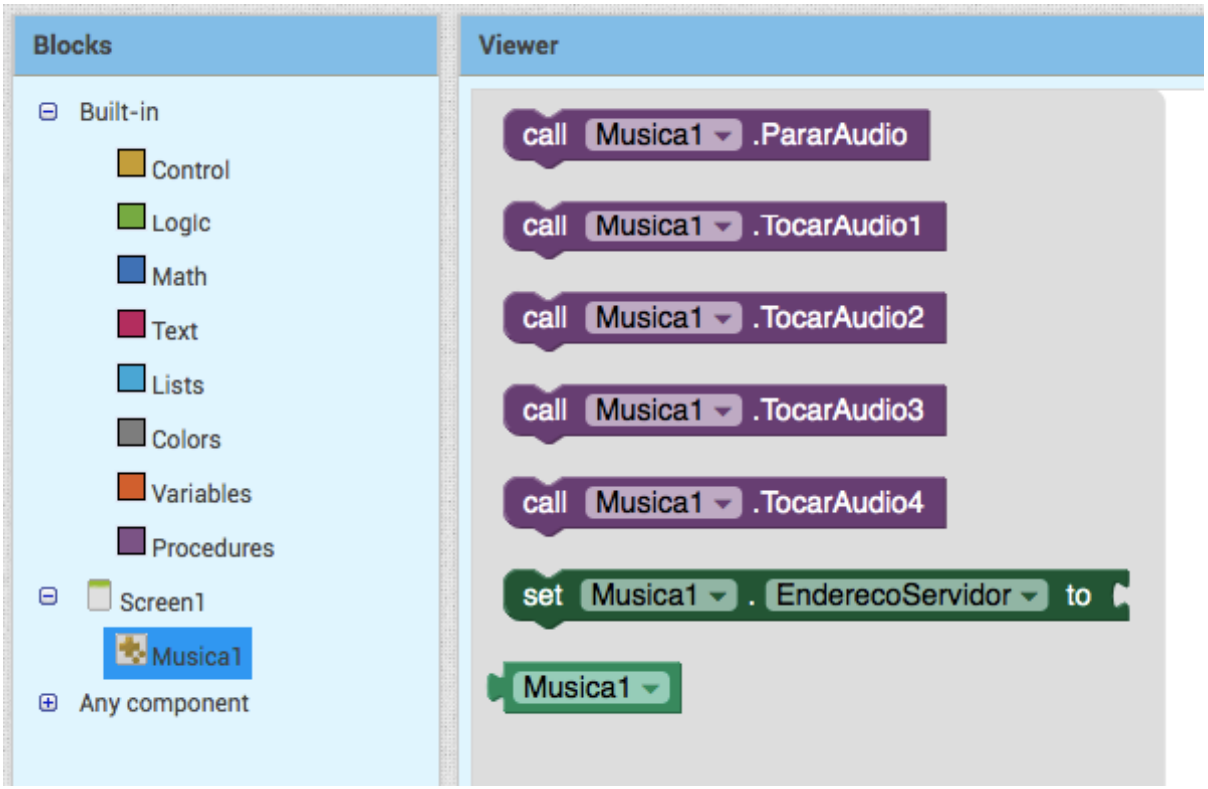

Figura 29 – Componente Música
elaborado pelo autor

A configuração do parâmetro endereço do servidor pode ser realizada de maneira definitiva, sem a necessidade de ser configurado na área de programação, como mostra a Figura 30.

Figura 30 – Configuração do parâmetro componente Música
elaborado pelo autor

O desenvolvimento da classe Musica, na ferramenta App Inventor, seguiu a mesma estratégia dos outros componentes implementados, tendo a classe Web como base para a programação. No decorrer do desenvolvimento foram encontradas algumas

dúvidas de como implementar o componente, pois a lista de músicas é um array de String fixo e, dessa forma, as músicas poderiam ser listadas no formato de um array, em um único bloco. Porém, não foi encontrada uma solução para a listagem de um array num bloco único, dentro do projeto App Inventor.

Sendo assim, foi definida a criação de métodos que representariam cada música separadamente. Desse modo, foram criados 5 métodos que seguem a representação dos blocos do App Inventor, como mostrado na Figura 31. Os métodos TocarAudio1 a TocarAudio4 realizam a chamada do mesmo serviço, que foram discutidos na seção 4.4.4.6, portanto, a única diferença de cada bloco é o parâmetro que representa a música que será executada em cada chamada de serviço. Já o método PararAudio realiza a chamada do serviço que se responsabiliza pela interrupção do processo que está executando a música.

```
216⊖    @SimpleFunction
217     public void TocarAudio1() {
218         String urlResult = "";
219         if (enderecoServidor != null && enderecoServidor != "") {
220             urlResult = enderecoServidor;
221         }
222         urlResult += "/sound/audio1.mp3/play";
223
224         this.urlString = urlResult;
225         final CapturedProperties webProps = capturePropertyValues("Musica");
226
227⊖         AsynchUtil.runAsynchronously(new Runnable() {
228⊖             @Override
229             public void run() {
230                 try {
231                     performRequest(webProps, null, null, "TOCAR_AUDIO1");
232                 } catch (FileUtil.FileException e) {
233                     form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio1", e.getErrorMessageNumber());
234                 } catch (Exception e) {
235                     Log.e(LOG_TAG, "ERROR_UNABLE_TO_GET", e);
236                     form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio1", ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
237                 }
238             }
239         });
240     }
241
243⊕     public void TocarAudio2() {
267
269⊕     public void TocarAudio3() {
293
295⊕     public void TocarAudio4() {
319
320⊖     @SimpleFunction
321     public void PararAudio() {
322         String urlResult = "";
323         if (enderecoServidor != null && enderecoServidor != "") {
324             urlResult = enderecoServidor + "/sound/stop";
325         }
326
327         this.urlString = urlResult;
328         final CapturedProperties webProps = capturePropertyValues("Musica");
329
330⊕         AsynchUtil.runAsynchronously(new Runnable() {
343     }
```

Figura 31 – Classe Musica
elaborado pelo autor

5 APLICAÇÃO E AVALIAÇÃO

Para avaliar a qualidade dos componentes que foram implementados junto à ferramenta App Inventor e analisar a qualidade, foi realizado um estudo de caso envolvendo uma turma de alunos da Educação Básica. A qualidade dos componentes e do servidor desenvolvidos foi avaliada em termos de utilidade, adequação funcional e usabilidade do ponto de vista de estudantes da Educação Básica.

5.1 DEFINIÇÃO DA AVALIAÇÃO

A avaliação das extensões que foram desenvolvidas para a ferramenta App Inventor foi baseada nos conceitos de qualidade definidos na norma ISO/IEC-25010 (STANDARDIZATION, 2016), sendo que os fatores de qualidade a serem avaliados são decompostos na Tabela 14.

Característica	**Subcaracterística**	**Avaliação do usuário**
		Item do questionário
		Questionário do aluno
Utilidade		Você acha a ferramenta componentes IoT do App Inventor útil na aprendizagem de programação.
Adequação funcional	Corretude funcional	Você observou algum erro (bug) em relação a funcionalidade dos componentes IoT do App Inventor.

Usabilidade	Satisfação	Você acha que a ferramenta componentes IoT auxilia no entendimento do conceito de IoT.
		Você acha que o ensino de programação utilizando sensores/placas torna o aprendizado mais interessante.
		Você achou fácil usar os componentes IoT da ferramenta App Inventor.
		Você observou algum erro (de ortografia/gramática) na interface dos componentes IoT do App Inventor.
		Você achou fácil usar a ferramenta na utilização do componente IoT.

Tabela 14 – Fatores de qualidade a serem avaliados

Instrumento de coleta de dados:

- Os dados foram coletados a partir de uma avaliação realizada com os alunos, obtendo dados sobre a eficácia e a qualidade das extensões que foram implementados;

5.1.1 DEFINIÇÃO DA AVALIAÇÃO COM USUÁRIOS

A avaliação realizada com usuários é, basicamente, um teste de usabilidade de forma sistemática dos componentes que foram implementados, analisando e coletando informações de usuários reais interagindo com os componentes e verificando possíveis dificuldades. Primeiramente, foi contextualizado o objetivo dos componentes que foram implementados e suas utilidades no cotidiano. Após essa apresentação inicial, foi executada uma tarefa predefinida (criar uma aplicação para controle de sensores que são comuns dentro de um ambiente doméstico).

A avaliação foi realizada no Instituto Federal de Santa Catarina (IFSC) - Campus Florianópolis - com turma da 7a. fase de Ensino Médio Integrado (Técnico em Eletrô-

nica), com apoio do professor Fernando Pacheco do IFSC e contou com a participação de 9 alunos, como mostrado na Figura 32.

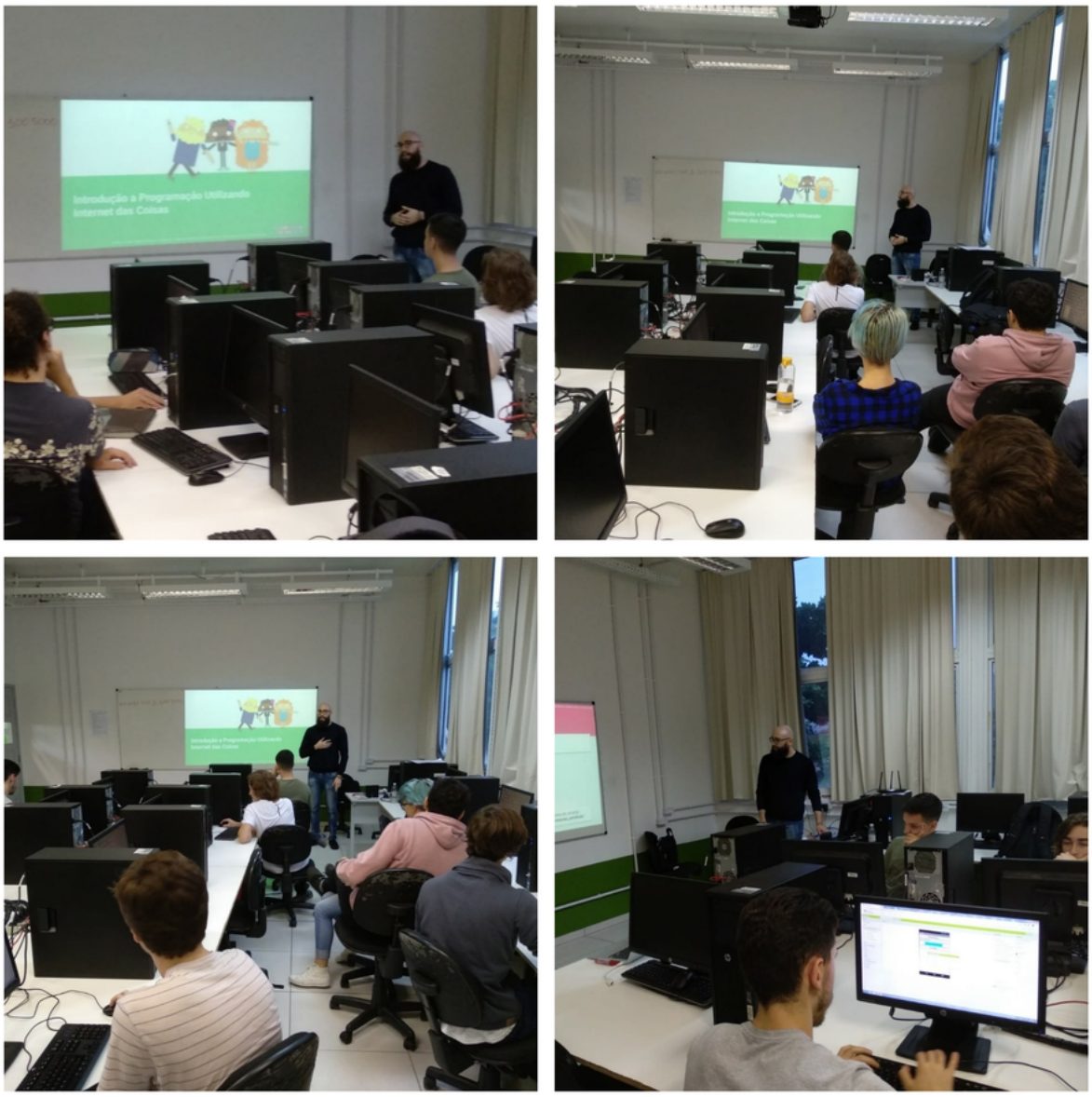

Figura 32 – Oficina Instituto Federal de Santa Catarina
elaborado pelo autor

Para realização da oficina foram preparados slides com o conteúdo a ser ministrado para os alunos da oficina, que foi realizada no IFSC. Foi abordada na oficina a criação de uma aplicação que controla dispositivos que são comuns ao ambiente doméstico, portanto, utilizam os quatro componentes que foram implementados no presente trabalho. Os participantes da oficina construíram a aplicação com o auxílio dos slides que são mostrados na Figura 33.

Figura 33 – Slides oficina
https://www.inf.ufsc.br/∼jean.hauck/outros/iot_cne/CnE_Oficina_IoT_PPT_Alunos.pdf

Ao final da oficina os participantes responderam um questionário (Apêndice A) que foi derivado da Tabela 14, utilizando fatores de qualidade que são abordados na engenharia de Software. Foi utilizada na construção do questionário uma abordagem de questões objetivas, porém, com algumas questões apresentando explicação para o caso de resultado negativo. Para a construção do questionário foi adotado também o método SUS (System Usability Scale), utilizado na avaliação global de usabilidade de uma interface, sendo considerado um mecanismo muito eficaz e barato, como citado por (JAMILLE et al.,). Junto ao questionário foram adicionadas questões discursivas para pontos fortes e fracos dos componentes desenvolvidos.

5.2 EXECUÇÃO DA AVALIAÇÃO

As avaliações foram realizadas durante o primeiro semestre de 2019 e ocorreram conforme definidas na seção 5.1.1. Dos nove participantes da avaliação de utilização dos componentes que foram implementados junto à ferramenta App Inventor, oito tinham experiência com estruturas de programação. Porém, não tinham familia-

ridade com a ferramenta App Inventor, mas já tinham realizado programação visual utilizando outras ferramentas (Codeblocks ou Scratch). Para avaliação, participaram um total de nove alunos do ensino médio técnico do IFSC (Instituto Federal de Educação), representando usuários da ferramenta App Inventor que utilizariam os componentes.

5.3 ANÁLISE DOS DADOS

Nesta seção é apresentada uma análise sobre os dados que foram coletados junto aos participantes da avaliação.

5.3.1 ANÁLISE DOS DADOS DAS AVALIAÇÕES COM USUÁRIOS

Os dados analisados têm o intuito de verificar o objetivo que foi proposto pela implementação dos componentes.

- **Os componentes IoT são úteis?**

Todos os usuários avaliaram a utilidade dos componentes que foram implementados junto à ferramenta App Inventor de maneira positiva. Assim, é possível inferir que os componentes implementados conseguiram exemplificar o conceito de “Internet das Coisas”, demostrando a interconectividade dos dispositivos que são comuns para o nosso cotidiano. Dessa forma, consideraram que a utilização dos componentes IoTs é mais lúdica e interessante para o aprendizado de programação. Os dados que foram coletados estão detalhados na Tabela 15.

Questão de análise	**Total de respostas**	
	Sim	**Não**
Você acha a ferramenta componentes IoT do App Inventor útil na aprendizagem de programação?	9 alunos	0 Alunos
Comentários		
“Funcionamento bom, intuitivo e didático.”		
“É intuitivo.”		
“Intuitiva.”		
“Intuitiva, prático e eficaz.”		
“Sim, pois o aluno não precisa se preocupar com sintaxe e apenas com a lógica.”		
“Apresenta de forma bastante elucidativa o ambiente de programação que muitas vezes é pouco convidativo.”		
“Porque é um pouco intuitivo para começar a aprender.”		
“Porque é um software mais objetivo e intuitivo, facilitando uma rápida aprendizagem.”		
“Pois todo conhecimento se agrega.”		

Tabela 15 – Avaliação da utilidade dos componentes IoTs

- **Os componentes IoTs são adequados funcionalmente?**

Em relação a adequação funcional, os componentes desenvolvidos receberam uma avaliação positiva dos alunos. Porém, foi levantado um ponto de melhoria no quesito de desempenho da aplicação, visto que quando muitos usuários estão utilizando o servidor da placa Raspberry Pi a(s) resposta(s) de alguns sensores acabam demorando ou não respondendo. Os dados que foram coletados estão detalhados na Tabela 16.

Questão de análise	**Total de respostas**	
	Sim	**Não**
Você observou algum erro (bug) em relação a funcionalidade dos componentes IoT do App Inventor?	7 alunos	2 alunos
Comentários		
“Lentidão causado pela quantidade de pessoas usando.”		
“Possui um certo limite para recepção de comandos em pouco tempo.”		

Tabela 16 – Avaliação da adequação funcional dos componentes IoTs

- **Os componentes IoTs tem boa usabilidade?**

Em relação a usabilidade, os componentes implementados receberam uma avaliação positiva dos participantes. Dessa forma, foi atingido o objetivo de abstrair toda a complexidade dos componentes, tornado sua manipulação simples e clara para os usuários. Os dados que foram coletados estão detalhados na Tabela 17.

Questão de análise	**Total de respostas**	
	Sim	**Não**
Você acha que a ferramenta componentes IoT auxilia no entendimento do conceito de IoT.	9 Alunos	0 Alunos
Você acha que o ensino de programação utilizando sensores/placas torna o aprendizado mais interessante.	9 Alunos	0 Alunos
Você achou fácil usar os componentes IoT da ferramenta App Inventor.	9 Alunos	0 Alunos

Você observou algum erro (de ortografia/gramática) na interface dos componentes IoT do App Inventor.	0 Alunos	9 Alunos
Você achou fácil usar a ferramenta na utilização do componente IoT.	9 Alunos	0 Alunos
Comentários		
“Mostra as associações, facilitando o entendimento.”		
“Pois é bem descrita.”		
“Exemplifica e demostra o funcionamento do IoT.”		
“Porque o aluno participa do processo de desenvolvimento.”		
“Mostra de forma prática a conectividade de diversos elementos.”		
“Porque é uma forma prática de demonstrar.”		
“Pois passa uma noção básica dos conceitos de IoT, abrindo as portas para algoritmos mais avançados.”		
“Pois pode interagir com smartphone.”		
“É mais didático pois é possível ver o funcionamento fora do computador.”		
“Pois, a utilização é prática.“		
“É mais atrativo visualmente.”		
“Torna tudo mais visível e compreensível.”		
“Sim, pois deixa o aprendizado mais próximo da realidade.”		
“Porque associa a programação á prática.”		
“Porque leva para a prática o que estudamos”		
“Aplicações que utilizam sensores mostrando aplicações reais, incentivando a pensar em coisas do dia a dia.“		
“Aproxima a teoria da prática.”		

Tabela 17 – Avaliação da usabilidade dos componentes IoTs

Durante a avaliação os usuários também responderam o questionário SUS. O questionário SUS (System Usability Scale) é um instrumento claro e rápido de coleta de informações que é utilizado ao final de um teste de interação. Desta forma, foi levantado

uma média da escala SUS de todos os participantes da oficina, sendo obtida a média 80,27 numa escala de 0 a 100. Desse modo, foi atingido um resultado satisfatório, levando em consideração que média abaixo de 50 é um sinal de que os investimentos em *design* e usabilidade precisam ser priorizados dentro de seu plano de negócios. O resultado da escala SUS de cada participante esta apresentada na Tabela 18.

Usuário	**Satisfação (SUS)**
Aluno 1	75,0
Aluno 2	87,5
Aluno 3	72,5
Aluno 4	82,5
Aluno 5	82,5
Aluno 6	85,0

Usuário	**Satisfação (SUS)**
Aluno 7	90,0
Aluno 8	82,5
Aluno 9	65,0
Média	**80,27**

Tabela 18 – Resultado da aplicação do questionário SUS

5.3.2 PONTOS FORTES

Os componentes que foram implementados junto à ferramenta App Inventor foram vistos como úteis no ensino dos conceitos de IoT. Como comentado pelos participantes, os componentes tornam a experiência mais lúdica e agradável em virtude de conseguirem ver uma saída mais real com o uso dos sensores. Desse modo, os alunos com pouca experiencia com algumas linguagens de programação não precisam se preocupar com problemas que são comuns inicialmente (como é o caso da sintaxe) na utilização dos componentes IoT, sendo que os blocos já são predefinidos. Os *feebacks* coletados no resultado das avaliações foram positivos, levantando indícios de que os componentes criados são intuitivos, práticos e eficazes para o que eles se propõem.

5.3.3 SUGESTÕES DE MELHORIA

Poucas sugestões foram levantadas pelos participantes da oficina. Porém, um ponto importante levantado por alguns participantes foi no quesito de desempenho do servidor de aplicação que roda na placa Raspberry Pi. Alguns alunos reclamaram

da demora na resposta de alguns sensores, e foi constatado que com muitos alunos acessando os mesmos sensores ocorre uma certa demora na resposta do servidor da placa Raspberry Pi. Além disso, alguns participantes sugeriram uma melhora no *design* de alguns componentes, para torná-los mais agradáveis visualmente.

5.3.4 AMEAÇAS À VALIDADE

Alguns fatores podem ser levantados como influenciadores ou ameaças à avaliação realizada junto aos alunos do IFSC. Devido a uma amostra pequena, houve baixa representatividade – visto que todos têm proximidade com a mesma instituição (IFSC). Visto que a grande maioria dos participantes não tinham familiaridade com a ferramenta App Inventor, encontraram dificuldade em localizar alguns componentes na interface gráfica. Os participantes utilizaram dispositivos móveis diferentes, em versões de Android que podiam não ter um desempenho agradável ao usuário, podendo assim alterar a sua perspectiva referente à avaliação.

6 CONCLUSÃO

O presente trabalho tem o objetivo de implementação de componentes IoT junto à ferramenta de código visual App Inventor, no intuito de tornar mais lúdico e atrativo o aprendizado de computação para alunos da Educação Básica, utilizando os componentes que foram desenvolvidos.

Desse modo, foi realizado um levantamento dos objetivos que seriam alcançados ao final do desenvolvimento do trabalho, com uma análise realizada através da fundamentação teórica sobre o aprendizado no ensino de computação utilizando o conceito de “Internet das Coisas”, com o auxílio de ferramentas de programação visual nas escolas de Ensino Básico, conforme pode ser visto no capítulo 2. Por consequente, foi levantado um estudo de mapeamento da literatura e, através disso, foram identificados poucos trabalhos que se enquadraram no contexto de programação visual utilizando a “Internet das Coisas” para o ensino de computação, conforme o capítulo 3.

Através de uma análise realizada a partir da fundamentação teórica e do estudo relacionando as literaturas disponibilizadas no estado da arte, foram desenvolvidas as extensões na ferramenta App Inventor que se comunicam com dispositivos IoTs, com a intenção de tornar o ensino de computação mais atratativo e interessante para crianças e adolescentes. As extensões desenvolvidas atendem principalmente a necessidade de usuários que são alunos de Educação Básica, porém, poderiam ser utilizadas no ensino de conceitos básicos nos primeiros anos de faculdades voltadas à computação.

Foi realizada a modelagem das extensões, conforme demonstra a seção 4.3, através de diagramas de UML, onde foram apresentadas as classes que foram criadas ao longo do processo de desenvolvimento. Foi também exposta uma simulação do processo de comunicação que é realizado pelo componente App Inventor com os sensores que estão conectados junto à placa Raspberry Pi. Em seguida, foi detalhado todo o processo de desenvolvimento ocorrido, com uma análise das decisões tomadas ao longo do desenvolvimento, levantando todo o processo de codificação, tanto da

camada na ferramenta App Inventor como da camada da placa Raspberry Pi, conforme mostrado na seção 4.4.

A avaliação dos componentes foi realizada junto a usuários reais, através de uma oficina aplicada a alunos do IFSC. Dessa forma, obteve-se dados sobre a eficácia no desenvolvimento das extensões bem como a qualidade percebida pelo ponto de vista dos participantes da oficina, analisando a utilidade, adequação funcional e usabilidade. Os resultados coletados das avaliações realizadas apresentou *feedback* positivo, sob o ponto de vista da clareza dos componentes e sua facilidade na utilização. Porém, a adequação funcional apresentou um ponto de melhoria no quesito desempenho do servidor da placa Raspberry Pi, que pode ser explorado em trabalhos futuros.

Através das pesquisas bibliográficas, foi levantada uma grande dificuldade no ponto de vista da implementação de projetos de ensino de computação com o auxílio de IoT, que é o quesito de custo, considerando que hardwares como LEGO Mindstormskit tem um alto valor, dificultando a implementação de projetos junto às escolas. Sendo assim, a proposta do trabalho apresenta um hardware mais acessível no quesito custo, levando em consideração que o Raspberry Pi e os componentes utilizados na montagem da placa não apresentaram um custo elevado para sua implementação, facilitando assim a implementação de projetos junto às escolas.

Espera-se com a implementação do presente trabalho que o ensino de computação tenha desempenho melhor no quesito aprendizado, tornando mais fácil a compreensão de algumas terminologias e tornando a área mais atrativa para os alunos da Educação Básica, para assim conseguir mais estudantes para faculdades no campo da computação.

Como trabalhos futuros, espera-se uma melhoria na parte do servidor de aplicação - quanto ao seu desempenho - , visto que, quando muitos usuários estão consumindo os serviços implementados, o servidor sofre uma sobrecarga, apresentando demora no retorno das requisições.

Ainda como ponto de melhoria, para aprimoramento das extensões, pode ser

implementado o processo de autenticação das requisições HTTP, o que evitaria que outro usuário alterasse o estado do sensor da placa.

Referências

AMORIM, J. et al. Integrando as Plataformas App Inventor e Arduino na Construção de um Humanoide. In: *Anais do Workshop de Informática na Escola*. [S.l.: s.n.], 2016. v. 22, n. 1.

APP INVENTOR. *Master Trainers*. 2018. Disponível em: http://appinventor.mit.edu/explore/master-trainers.html. Acesso em: 04/05/2018.

BARR, V.; STEPHENSON, C. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *Acm Inroads*, ACM, v. 2, n. 1, p. 48 – 54, 2011.

CASAGRAS, E. F. P. Casagras final report: rfid and the inclusive model for the internet of things. 2009.

CASTELLS, M. *A Galáxia Internet*: reflexões sobre a Internet, negócios e a sociedade. [S.l.]: Zahar, 2003.

COSTA, C. *Educação, imagem e mídias*. [S.l.]: Cortez, 2005.

FALKEMBACH, G. A. M. et al. Aprendizagem de Algoritmos: Uso da Estratégia Ascendente de Resolução de Problemas. *8\textordmasculine Taller Internacional de Software Educativo. Santiago, Chile*, 2003.

FOUNDATION, P. S. *Python*. 2016. Acesso em 31 Ago. 2016. Disponível em: https://docs.python.org/2/license.html.

GOMES, T. C.; MELO, J. C. de. App inventor for android: Uma nova possibilidade para o ensino de lógica de programaçao. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. [S.l.: s.n.], 2013. v. 2, n. 1.

HEINEN, E. *Raspiblocos*: ambiente de programação didático baseado em Raspberry Pi e Blockly. 2015. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná.

IBANEZ, L. *Blockly makes it easier to learn to code*. 2015. Disponível em: https://opensource.com/education/15/2/blockly-makes-easier-every-one-learn-code. Acesso em: 03/04/2018.

INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA. Brasil. Censo Escolar da Educação Básica. 2010–2014.

JAMILLE, N. de L. et al. USABILIDADE DE OBJETOS DE USO COTIDIANO: COMPARATIVO DE TÉCNICAS DE AVALIAÇÃO SUBJETIVA (SUS E DS) USABILITY OF OBJECTS OF USE EVERYDAY: COMPARISON OF TECHNIQUES FOR THE EVALUATION SUBJECTIVE (SUS E SD).

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele University*, v. 33, n. 2004, p. 1 – 26, 2004.

LECHETA, R. R. Android Essencial. *São Paulo, Editora NovaTec*, 2016.

LIBÂNEO, J. C. *A didática ea aprendizagem do pensar e do aprender*: a teoria histórico-cultural da atividade ea contribuição de Vasili Davydov. [S.l.]: SciELO Brasil, 2006.

MARCON JÚNIOR, R. P.; BONIATI, B. B. LogicBlocks: Uma Ferramenta para o Ensino de Lógica de Programação. *Anais do EATI-Encontro Anual de Tecnologia da Informação e Semana Acadêmica de Tecnologia da Informação*, 2015.

MASSACHUSSETS Institute of Technology - App Inventor for Android. 2012. Disponível em: www.appinventor.mit.edu. Acesso em: 12/03/2018.

MEC. Diretrizes Curriculares Nacionais. *Do Histórico da Computação, do Computador e dos Cursos*, 2011.

MEDEIROS FILHO, D. A.; GONÇALVES, P. C. Robótica Educacional de Baixo Custo: Uma Realidade para as Escolas Brasileiras. In: *Anais do Workshop de Informática na Escola*. [S.l.: s.n.], 2008. v. 1, n. 1.

MERKOURIS, A.; CHORIANOPOULOS, K.; KAMEAS, A. Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *ACM Transactions on Computing Education (TOCE)*, ACM, v. 17, n. 2, 2017.

MORELLI, R. et al. Can android app inventor bring computational thinking to k-12. In: *Proc. 42nd ACM technical symposium on Computer science education (SIGCSE'11)*. [S.l.: s.n.], 2011. p. 1 – 6.

PEREIRA, J. et al. Uma Solução de IoT para Uso Eficiente de Energia Elétrica em Prédios Inteligentes. 2016.

QUEIROZ, R.; SAMPAIO, F. F.; SANTOS, M. P. dos. DuinoBlocks4Kids: Ensinando conceitos básicos de programação a crianças do Ensino Fundamental I por meio da Robótica Educacional. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. [S.l.: s.n.], 2016. v. 5, n. 1.

RAABE, A. L. A. et al. Referenciais de Formação em Computação: Educação Básica. *Comissão de Educação e apresentado no CSBC 2017 durante as Assembleias do WEI e da SBC*, 2017.

RONACHER, A. Flask (a Python microframework). *Dosegljivo: http://flask. pocoo. org.[Dostopano: 20. 7. 2018]*, 2018.

RUZZENENTE, M. et al. A review of robotics kits for tertiary education. In: CITESEER, 2012. *Proceedings of International Workshop Teaching Robotics Teaching with Robotics*: Integrating robotics in school curriculum. [S.l.], 2012. p. 153 – 162.

SEEHORN, D. et al. CSTA K–12 Computer Science Standards: Revised 2011. ACM, 2011.

SKLAR, E.; EGUCHI, A.; JOHNSON, J. RoboCupJunior: learning with educational robotics. In: SPRINGER, 2002. *Robot Soccer World Cup*. [S.l.], 2002. p. 238 – 253.

STANDARDIZATION, I. O. for. *Systems and Software Engineering*: Systems and Software Quality Requirements and Evaluation (SQuaRE): Measurement of System and Software Product Quality. [S.l.]: ISO, 2016.

SUDOL, L. A. *Visual Programming Pedagogies and Integrating Current Visual Programming Language Features*. 2009. Tese (Doutorado) — Robotics Institute.

SUNDMAEKER, H. et al. Vision and challenges for realising the Internet of Things. *Cluster of European Research Projects on the Internet of Things, European Commision*, v. 3, n. 3, p. 34 – 36, 2010.

TISSENBAUM, M. et al. Off the Screen, and Into the World of Everyday Objects: Computational Thinking for Youth with the Internet of Things. *Siu-cheung KONG The Education University of Hong Kong, Hong Kong*, 2017.

WING, J. M. Computational thinking. *Communications of the ACM*, v. 49, n. 3, p. 33 – 35, 2006.

APÊNDICE A - Avaliação do componente de IoT da ferramenta App Inventor

Universidade Federal de Santa Catarina - Departamento de Informática e Estatística
Questionário do trabalho de conclusão de curso de Sistemas de Informação de Michel Miola

---

Avaliação do componente de **IoT da Ferramenta App Inventor**

Gostaríamos conhecer a sua opinião sobre a ferramenta componentes IoT do App Inventor. Os dados serão publicados somente de forma agregada e anônima.

Nome completo: _____

1- Você está em que ano do ensino médio?
( ) Ensino médio série 1ª
( ) Ensino médio série 2ª
( ) Ensino médio série 3ª
Outro:_____

2- Você já programou um programa de software?
( ) Nunca
( ) 1 - 2 vezes
( ) 3 - 5 vezes
( ) Mais de 5 vezes

3- Qual ambiente de programação você usou para programar? (Pode ser mais de uma resposta)
( ) App Inventor
( ) Scratch
( ) Snap!
( ) Outro: _____

4- Você acha a ferramenta componentes IoT do App Inventor útil no aprendizagem de programação?
( ) Sim
( ) Não
Explique, porque: _____

5- Você acha que a ferramenta componentes IoT auxilia no entendimento do conceito de IoT?
( ) Sim

( ) Não
Explique, porque: _____

6- Você acha que o ensino de programação utilizando sensores/placas torna o aprendizado mais interessante?
( ) Sim
( ) Não
Explique, porque: _____

7- Você observou algum erro (bug) em relação a funcionalidade dos componentes IoT do App Inventor?
( ) Sim
( ) Não
Se sim, qual(is)? _____

8- Você achou fácil usar os componentes IoT da ferramenta App Inventor?
( ) Sim
( ) Não
Se não, o que achou difícil?_____

9- Você observou algum erro (de ortografia/gramática) na interface dos componentes IoT do App Inventor?
( ) Sim
( ) Não
Se sim, qual?_____

10- Você achou fácil usar a ferramenta a utilização do componente IoT?
( ) Sim
( ) Não
Se não, o que achou difícil?_____

Para cada um dos itens a seguir, responda escolhendo entre: 1 (discordo totalmente), 2 , 3, 4, 5 (concordo completamente)

Universidade Federal de Santa Catarina - Departamento de Informática e Estatística
Questionário do trabalho de conclusão de curso de Sistemas de Informação de Michel Miola

---

Item	Afirmação	discordo totalmente 1	2	3	4	concordo completamente 5
1	Eu penso que usarei os componentes IoT do App Inventor com frequência.					
2	Acho os componentes IoT desnecessariamente complexos.					
3	Penso que os componentes IoT são fáceis de usar.					
4	Acho que vou precisar da ajuda de um técnico para usar os componentes IoT.					
5	Acho os componentes IoT bem integrados.					
6	Encontro muitas inconsistências nos componentes IoT.					
7	Imagino que as pessoas aprenderão rapidamente a usar os componentes IoT.					
8	Não acho os componentes IoT prática de usar.					

9	Senti-me confiante ao usar os componentes IoT.					
10	Precisei aprender muitas coisas antes de ser capaz de operar os componentes IoT.					

O que você mais gostou dos componentes IoT da ferramenta App Inventor?

Alguma sugestão de melhoria referente os componentes IoT da ferramenta App Inventor?

Mais algum comentário?

APÊNDICE B - Código fonte desenvolvido

```
#!/usr/bin/python
# Example using a character LCD connected to a Raspberry Pi or
BeagleBone Black.
import time

import Adafruit_CharLCD as LCD
import Adafruit_GPIO as GPIO

from subprocess import *
from datetime import datetime
import socket
import fcntl
import struct

def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,  # SIOCGIFADDR
        struct.pack('256s', ifname[:15])
    )[20:24])


# Raspberry Pi pin configuration:
lcd_rs        = 27  # Note this might need to be changed to 21 for
older revision Pi's.
lcd_en        = 22
lcd_d4        = 25
lcd_d5        = 24
lcd_d6        = 23
lcd_d7        = 18
lcd_backlight = 4

# Define LCD column and row size for 16x2 LCD.
lcd_columns = 15
lcd_rows    = 1

# Initialize the LCD using the pins above.
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6,
```

```
lcd_d7, lcd_columns, lcd_rows, lcd_backlight)

# Print a two line message
lcd.message('Inicializando...!')

# Wait 25 seconds
time.sleep(25.0)

# Demo showing the cursor.
lcd.clear()

ipaddr = get_ip_address('eth0')
# print(ipaddr)
lcd.message('Rede: eth0   IP: \n')
lcd.message(ipaddr)
```

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import Adafruit_DHT
from flask import Flask, Response
import logging
import pygame
import subprocess
import os
import json
app = Flask(__name__)

UMIDADE = 0
TEMPERATURA = 1

@app.route('/relay/<int:portDigital>/status/<string:status>')
def relayControl(portDigital, status):
      resp = Response();
      try:
            portDigital = int(portDigital)
            GPIO.setmode(GPIO.BCM)
            GPIO.setwarnings(False)
            GPIO.setup(portDigital, GPIO.OUT)
            GPIO.output(portDigital, GPIO.HIGH)
            circuitRelay(status, portDigital)
```

```
            resp.status_code = 200
      except KeyboardInterrupt:
            resp.status_code = 400
      return resp

@app.route('/temperatura/<int:portDigital>')
def temperatureControl(portDigital):
      resp = Response()
      try:
            portDigital = int(portDigital)
            DHT = getDHT(portDigital)
            resp = Response(str(DHT[TEMPERATURA]), status=200,
mimetype='application/json')
      except KeyboardInterrupt:
            resp.status_code = 400
      return resp

@app.route('/umidade/<int:portDigital>')
def umidadeControl(portDigital):
      resp = Response()
      try:
            portDigital = int(portDigital)
            DHT = getDHT(portDigital)
            resp = Response(str(DHT[UMIDADE]), status=200,
mimetype='application/json')
      except KeyboardInterrupt:
            resp.status_code = 400
      return resp

@app.route('/sound/<string:sound>/play')
def playSound(sound):
      resp = Response()
      try:
            subprocess.call('omxplayer --no-keys
/home/pi/git/AppInventorCnEIoT/sounds/' + sound + '&', shell=True)
            resp = Response("Play sucesso",status=200,
mimetype='application/json')
      except KeyboardInterrupt:
            resp.status_code = 400
      return resp

@app.route('/sound/stop')
def stopSound():
      resp = Response()
      try:
```

```
            subprocess.call('pkill omxplayer', shell=True)
            resp = Response("Stop sucesso",status=200,
mimetype='application/json')
      except KeyboardInterrupt:
            resp.status_code = 400
      return resp

@app.route('/sound/list')
def listSounds():
      resp = Response()
      try:
            nomesSound = []
            for root, dirs, files in
os.walk("/home/pi/git/AppInventorCnEIoT/sounds/"):
                for filename in files:
                    filenameSplit = filename.split(".")
                    nomesSound.append(filenameSplit[0])
            resp = Response(json.dumps(nomesSound) ,status=200,
mimetype='application/json')
      except KeyboardInterrupt:
            resp.status_code = 400
      return resp


def circuitRelay(status, portDigital):
      if (status == "ativo"):
            activeRelay(portDigital);
      elif (status == "inativo"):
            disableRelay(portDigital);

def getDHT(portDigital):
      sensor = Adafruit_DHT.DHT11
      GPIO.setmode(GPIO.BCM)
      GPIO.setwarnings(False)
      return Adafruit_DHT.read_retry(sensor, portDigital)

def activeRelay(portDigital):
      GPIO.output(portDigital, GPIO.LOW)

def disableRelay(portDigital):
      GPIO.output(portDigital, GPIO.HIGH)

logger = logging.getLogger('myapp')
hdlr = logging.FileHandler('/var/tmp/myapp.log')
formatter = logging.Formatter('%(asctime)s %(levelname)s %(message)s')
```

```
hdlr.setFormatter(formatter)
logger.addHandler(hdlr)
logger.setLevel(logging.DEBUG)
app.run(debug=True,host='0.0.0.0')
```

```
package edu.mit.cne.appinventor;

// -*- mode: java; c-basic-offset: 2; -*-

// Copyright 2009-2011 Google, All Rights reserved
// Copyright 2011-2012 MIT, All rights reserved
// Released under the Apache License, Version 2.0
// http://www.apache.org/licenses/LICENSE-2.0

import com.google.appinventor.components.annotations.DesignerComponent;
import com.google.appinventor.components.annotations.DesignerProperty;
import com.google.appinventor.components.annotations.PropertyCategory;
import com.google.appinventor.components.annotations.SimpleEvent;
import com.google.appinventor.components.annotations.SimpleFunction;
import com.google.appinventor.components.annotations.SimpleObject;
import com.google.appinventor.components.annotations.SimpleProperty;
import com.google.appinventor.components.annotations.UsesLibraries;
import com.google.appinventor.components.annotations.UsesPermissions;
import com.google.appinventor.components.common.ComponentCategory;
import com.google.appinventor.components.common.HtmlEntities;
import com.google.appinventor.components.common.PropertyTypeConstants;
import com.google.appinventor.components.common.YaVersion;
import
com.google.appinventor.components.runtime.AndroidNonvisibleComponent;
import com.google.appinventor.components.runtime.Component;
import com.google.appinventor.components.runtime.ComponentContainer;
import com.google.appinventor.components.runtime.EventDispatcher;
import
com.google.appinventor.components.runtime.HandlesEventDispatching;
import com.google.appinventor.components.runtime.Web;
import com.google.appinventor.components.runtime.collect.Lists;
import com.google.appinventor.components.runtime.collect.Maps;
import com.google.appinventor.components.runtime.util.AsynchUtil;
import com.google.appinventor.components.runtime.util.ErrorMessages;
import com.google.appinventor.components.runtime.util.FileUtil;
import com.google.appinventor.components.runtime.util.GingerbreadUtil;
import com.google.appinventor.components.runtime.util.JsonUtil;
import com.google.appinventor.components.runtime.util.MediaUtil;
import com.google.appinventor.components.runtime.util.SdkLevel;
import com.google.appinventor.components.runtime.util.YailList;
```

```
import android.app.Activity;
import android.text.TextUtils;
import android.util.Log;

import org.json.JSONException;
import org.json.JSONObject;
import org.json.XML;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.CookieHandler;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.List;
import java.util.Map;

/**
 * IoT components for connection
 *
 * @author michel.miola@ufsc.grad.br
 */
@DesignerComponent(version = YaVersion.WEB_COMPONENT_VERSION,
description = "Component lamp to turn on or off.", category =
ComponentCategory.EXTENSION, nonVisible = true, iconName =
"images/extension.png")
@SimpleObject(external = true)
@UsesPermissions(permissionNames = "android.permission.INTERNET," +
"android.permission.WRITE_EXTERNAL_STORAGE,"
            + "android.permission.READ_EXTERNAL_STORAGE")
@UsesLibraries(libraries = "json.jar")

public class Temperatura extends AndroidNonvisibleComponent implements
Component {
```

```
      private final Activity activity;
      private final CookieHandler cookieHandler;

      private String urlString = "";
      private boolean allowCookies;
      private YailList requestHeaders = new YailList();
      private boolean saveResponse;
      private String responseFileName = "";
      private String enderecoServidor = "";
      private final String GPIO_DEFAULT = "6";
      private String GPIO = "";

      private static class InvalidRequestHeadersException extends
Exception {
            /*
             * errorNumber could be:
ErrorMessages.ERROR_WEB_REQUEST_HEADER_NOT_LIST
             * ErrorMessages.ERROR_WEB_REQUEST_HEADER_NOT_TWO_ELEMENTS
             */
            final int errorNumber;
            final int index; // the index of the invalid header

            InvalidRequestHeadersException(int errorNumber, int index) {
                  super();
                  this.errorNumber = errorNumber;
                  this.index = index;
            }
      }

      /**
       * BuildRequestDataException can be thrown from buildRequestData.
It is
       * thrown if the list passed to buildRequestData contains an item
that is
       * not a list. It is thrown if the list passed to buildRequestData
contains
       * an item that is a list whose size is not 2.
       */
      // VisibleForTesting
      static class BuildRequestDataException extends Exception {
            /*
             * errorNumber could be:
             * ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT_LIST
             *
ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT_TWO_ELEMENTS
```

```
             */
            final int errorNumber;
            final int index; // the index of the invalid header

            BuildRequestDataException(int errorNumber, int index) {
                  super();
                  this.errorNumber = errorNumber;
                  this.index = index;
            }
      }

      /**
       * The CapturedProperties class captures the current property
values from a
       * Web component before an asynchronous request is made. This
avoids
       * concurrency problems if the user changes a property value after
       * initiating an asynchronous request.
       */
      private static class CapturedProperties {
            final String urlString;
            final URL url;
            final boolean allowCookies;
            final boolean saveResponse;
            final String responseFileName;
            final Map<String, List<String>> requestHeaders;
            final Map<String, List<String>> cookies;

            CapturedProperties(Temperatura temperatura) throws
MalformedURLException, InvalidRequestHeadersException {
                  urlString = temperatura.urlString;
                  url = new URL(urlString);
                  allowCookies = temperatura.allowCookies;
                  saveResponse = temperatura.saveResponse;
                  responseFileName = temperatura.responseFileName;
                  requestHeaders =
processRequestHeaders(temperatura.requestHeaders);

                  Map<String, List<String>> cookiesTemp = null;
                  if (allowCookies && temperatura.cookieHandler != null)
{
                        try {
                              cookiesTemp =
temperatura.cookieHandler.get(url.toURI(), requestHeaders);
                        } catch (URISyntaxException e) {
```

```
                        // Can't convert the URL to a URI; no
cookies for you.
                    } catch (IOException e) {
                        // Sorry, no cookies for you.
                    }
                }
                cookies = cookiesTemp;
            }
        }

        private static final String LOG_TAG = "Temperatura";

        private static final Map<String, String> mimeTypeToExtension;

        static {
            mimeTypeToExtension = Maps.newHashMap();
            mimeTypeToExtension.put("application/pdf", "pdf");
            mimeTypeToExtension.put("application/zip", "zip");
            mimeTypeToExtension.put("audio/mpeg", "mpeg");
            mimeTypeToExtension.put("audio/mp3", "mp3");
            mimeTypeToExtension.put("audio/mp4", "mp4");
            mimeTypeToExtension.put("image/gif", "gif");
            mimeTypeToExtension.put("image/jpeg", "jpg");
            mimeTypeToExtension.put("image/png", "png");
            mimeTypeToExtension.put("image/tiff", "tiff");
            mimeTypeToExtension.put("text/plain", "txt");
            mimeTypeToExtension.put("text/html", "html");
            mimeTypeToExtension.put("text/xml", "xml");
            // TODO(lizlooney) - consider adding more mime types.
        }

        /**
         * Creates a new Web component.
         *
         * @param container
         *            the Form that this component is contained in.
         */
        public Temperatura(ComponentContainer container) {
            super(container.$form());
            activity = container.$context();

            cookieHandler = (SdkLevel.getLevel() >=
SdkLevel.LEVEL_GINGERBREAD) ? GingerbreadUtil.newCookieManager() : null;
        }
```

```
	/**
	 * This constructor is for testing purposes only.
	 */
	protected Temperatura() {
		super(null);
		activity = null;
		cookieHandler = null;
	}

	@DesignerProperty(editorType =
PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue = "")
	@SimpleProperty
	public void EnderecoServidor(String enderecoServidor) {
		if (enderecoServidor != null) {
			this.enderecoServidor = enderecoServidor.trim();
		}
	}

	@DesignerProperty(editorType =
PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue = "6")
	@SimpleProperty
	public void Gpio(String Gpio) {
		if (Gpio != null) {
			this.GPIO = Gpio.trim();
		}
	}

	/**
	 * Performs an HTTP GET request using the Url property and
retrieves the
	 * response.<br>
	 * If the SaveResponse property is true, the response will be
saved in a
	 * file and the GotFile event will be triggered. The
ResponseFileName
	 * property can be used to specify the name of the file.<br>
	 * If the SaveResponse property is false, the GotText event will
be
	 * triggered.
	 */
	@SimpleFunction
	public void Obter() {
		if (this.GPIO == null || this.GPIO.isEmpty()) {
			this.GPIO = this.GPIO_DEFAULT;
		}
```

```
            String urlWithPath = mounUrlWithPath(this.enderecoServidor,
this.GPIO);
            this.urlString = urlWithPath;
            final CapturedProperties webProps =
capturePropertyValues("Tempetaura");
            if (this.enderecoServidor == "" || this.GPIO == "") {
                  return;
            }

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              performRequest(webProps, null, null,
"OBTER");
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Temperatura.this, "Obter",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_GET", e);

form.dispatchErrorOccurredEvent(Temperatura.this, "Obter",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET,
                                          webProps.urlString);
                        }
                  }
            });
      }

      /**
       * Event indicating that a request has finished.
       *
       * @param url
       *            the URL used for the request
       * @param responseCode
       *            the response code from the server
       * @param responseType
       *            the mime type of the response
       * @param temperatura
       *            the response content from the server
       */
      @SimpleEvent
      public void Capturar(String temperatura) {
            // invoke the application's "GotText" event handler.
```

```
            EventDispatcher.dispatchEvent(this, "Capturar", "", 200,
"JSON", temperatura);
      }

      public static String mounUrlWithPath(String enderecoServidor,
String gpio) {
            String urlResult = "";
            if (enderecoServidor != null && enderecoServidor != "") {
                  urlResult = enderecoServidor;
            }
            if (gpio != null && gpio != "") {
                  urlResult += "/temperatura/" + gpio;
            }

            return urlResult;
      }

      /*
       * Performs an HTTP GET, POST, PUT or DELETE request using the Url
property
       * and the specified text, and retrieves the response
asynchronously.<br>
       * The characters of the text are encoded using the given
encoding.<br> If
       * the SaveResponse property is true, the response will be saved
in a file
       * and the GotFile event will be triggered. The ResponseFileName
property
       * can be used to specify the name of the file.<br> If the
SaveResponse
       * property is false, the GotText event will be triggered.
       *
       * @param text the text data for the POST or PUT request
       *
       * @param encoding the character encoding to use when sending the
text. If
       * encoding is empty or null, UTF-8 encoding will be used.
       *
       * @param functionName the name of the function, used when
dispatching
       * errors
       *
       * @param httpVerb the HTTP operation to be performed: GET, POST,
PUT or
       * DELETE
```

```
       */
      private void requestTextImpl(final String text, final String
encoding, final String functionName,
                  final String httpVerb) {
            // Capture property values before running asynchronously.
            final CapturedProperties webProps =
capturePropertyValues(functionName);
            if (webProps == null) {
                  // capturePropertyValues has already called
                  // form.dispatchErrorOccurredEvent
                  return;
            }

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        // Convert text to bytes using the encoding.
                        byte[] requestData;
                        try {
                              if (encoding == null || encoding.length()
== 0) {
                                    requestData = text.getBytes("UTF-8");
                              } else {
                                    requestData =
text.getBytes(encoding);
                              }
                        } catch (UnsupportedEncodingException e) {

form.dispatchErrorOccurredEvent(Temperatura.this, functionName,

ErrorMessages.ERROR_WEB_UNSUPPORTED_ENCODING, encoding);
                              return;
                        }

                        try {
                              performRequest(webProps, requestData, null,
httpVerb);
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Temperatura.this, functionName,
e.getErrorMessageNumber());
                        } catch (Exception e) {

form.dispatchErrorOccurredEvent(Temperatura.this, functionName,
```

```
ErrorMessages.ERROR_WEB_UNABLE_TO_POST_OR_PUT, text,
webProps.urlString);
                        }
                    }
            });
      }

      /*
       * Converts a list of two-element sublists, representing name and
value
       * pairs, to a string formatted as
application/x-www-form-urlencoded media
       * type, suitable to pass to PostText.
       *
       * @param list a list of two-element sublists representing name
and value
       * pairs
       *
       * @throws BuildPostDataException if the list is not valid
       */
      // VisibleForTesting
      String buildRequestData(YailList list) throws
BuildRequestDataException {
            StringBuilder sb = new StringBuilder();
            String delimiter = "";
            for (int i = 0; i < list.size(); i++) {
                  Object item = list.getObject(i);
                  // Each item must be a two-element sublist.
                  if (item instanceof YailList) {
                        YailList sublist = (YailList) item;
                        if (sublist.size() == 2) {
                              // The first element is the name.
                              String name =
sublist.getObject(0).toString();
                              // The second element is the value.
                              String value =
sublist.getObject(1).toString();

sb.append(delimiter).append(UriEncode(name)).append('=').append(UriEncod
e(value));
                        } else {
                              throw new
BuildRequestDataException(ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT
_TWO_ELEMENTS,
                                          i + 1);
```

```
                        }
                  } else {
                        throw new
BuildRequestDataException(ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT
_LIST, i + 1);
                  }
                  delimiter = "&";
            }
            return sb.toString();
      }

      public String UriEncode(String text) {
            try {
                  return URLEncoder.encode(text, "UTF-8");
            } catch (UnsupportedEncodingException e) {
                  // If UTF-8 is not supported, we're in big trouble!
                  // According to Javadoc and Android documentation for
                  // java.nio.charset.Charset, UTF-8 is
                  // available on every Java implementation.
                  Log.e(LOG_TAG, "UTF-8 is unsupported?", e);
                  return "";
            }
      }

      /**
       * Decodes the given JSON encoded value.
       *
       * @param jsonText
       *            the JSON text to decode
       * @return the decoded object
       * @throws IllegalArgumentException
       *             if the JSON text can't be decoded
       */
      // VisibleForTesting
      static Object decodeJsonText(String jsonText) throws
IllegalArgumentException {
            try {
                  return JsonUtil.getObjectFromJson(jsonText);
            } catch (JSONException e) {
                  throw new IllegalArgumentException("jsonText is not a
legal JSON value");
            }
      }

      /*
```

```
       * Perform a HTTP GET or POST request. This method is always run
on a
       * different thread than the event thread. It does not use any
property
       * value fields because the properties may be changed while it is
running.
       * Instead, it uses the parameters. If either postData or postFile
is
       * non-null, then a post request is performed. If both postData
and postFile
       * are non-null, postData takes precedence over postFile. If
postData and
       * postFile are both null, then a get request is performed. If
saveResponse
       * is true, the response will be saved in a file and the GotFile
event will
       * be triggered. responseFileName specifies the name of the file.
If
       * saveResponse is false, the GotText event will be triggered.
       *
       * This method can throw an IOException. The caller is responsible
for
       * catching it and triggering the appropriate error event.
       *
       * @param webProps the captured property values needed for the
request
       *
       * @param postData the data for the post request if it is not
coming from a
       * file, can be null
       *
       * @param postFile the path of the file containing data for the
post request
       * if it is coming from a file, can be null
       *
       * @throws IOException
       */
      private void performRequest(final CapturedProperties webProps,
byte[] postData, String postFile, String httpVerb)
                  throws IOException {

            // Open the connection.
            HttpURLConnection connection = openConnection(webProps,
httpVerb);
            if (connection != null) {
```

```
                  try {
                        if (postData != null) {
                              writeRequestData(connection, postData);
                        } else if (postFile != null) {
                              writeRequestFile(connection, postFile);
                        }

                        // Get the response.
                        final int responseCode =
connection.getResponseCode();
                        final String responseType =
getResponseType(connection);
                        processResponseCookies(connection);

                        if (!saveResponse) {
                              final String temperatura =
getResponseContent(connection);

                              // Dispatch the event.
                              activity.runOnUiThread(new Runnable() {
                                    @Override
                                    public void run() {
                                          Capturar(temperatura);
                                    }
                              });
                        }

                  } finally {
                        connection.disconnect();
                  }
            }
      }

      /**
       * Open a connection to the resource and set the HTTP action to
PUT or
       * DELETE if it is one of them. GET would be the default, and POST
is set in
       * writeRequestData or writeRequestFile
       *
       * @param webProps
       *            the properties of the connection, set as properties
in the
       *            component
       * @param httpVerb
```

```
       *            One of GET/POST/PUT/DELETE
       * @return a HttpURL Connection
       * @throws IOException
       * @throws ClassCastException
       * @throws ProtocolException
       *             thrown if the method in setRequestMethod is not
correct
       */
      private static HttpURLConnection openConnection(CapturedProperties
webProps, String httpVerb)
                  throws IOException, ClassCastException,
ProtocolException {

            HttpURLConnection connection = (HttpURLConnection)
webProps.url.openConnection();

            if (httpVerb.equals("PUT") || httpVerb.equals("DELETE")) {
                  // Set the Request Method; GET is the default, and if
it is a POST,
                  // it will be marked as such
                  // with setDoOutput in writeRequestFile or
writeRequestData
                  connection.setRequestMethod(httpVerb);
            }

            // Request Headers
            for (Map.Entry<String, List<String>> header :
webProps.requestHeaders.entrySet()) {
                  String name = header.getKey();
                  for (String value : header.getValue()) {
                        connection.addRequestProperty(name, value);
                  }
            }

            // Cookies
            if (webProps.cookies != null) {
                  for (Map.Entry<String, List<String>> cookie :
webProps.cookies.entrySet()) {
                        String name = cookie.getKey();
                        for (String value : cookie.getValue()) {
                              connection.addRequestProperty(name, value);
                        }
                  }
            }
```

```
            return connection;
      }

      private static void writeRequestData(HttpURLConnection connection,
byte[] postData) throws IOException {
            // According to the documentation at
            //
http://developer.android.com/reference/java/net/HttpURLConnection.html
            // HttpURLConnection uses the GET method by default. It will
use POST if
            // setDoOutput(true) has
            // been called.
            connection.setDoOutput(true); // This makes it something
other than a
                                                      // HTTP GET.
            // Write the data.
            connection.setFixedLengthStreamingMode(postData.length);
            BufferedOutputStream out = new
BufferedOutputStream(connection.getOutputStream());
            try {
                  out.write(postData, 0, postData.length);
                  out.flush();
            } finally {
                  out.close();
            }
      }

      private void writeRequestFile(HttpURLConnection connection, String
path) throws IOException {
            // Use MediaUtil.openMedia to open the file. This means that
path could
            // be file on the SD card,
            // an asset, a contact picture, etc.
            BufferedInputStream in = new
BufferedInputStream(MediaUtil.openMedia(form, path));
            try {
                  // Write the file's data.
                  // According to the documentation at
                  //
http://developer.android.com/reference/java/net/HttpURLConnection.html
                  // HttpURLConnection uses the GET method by default.
It will use
                  // POST if setDoOutput(true) has
                  // been called.
                  connection.setDoOutput(true); // This makes it
```

```
something other than
                                                                  // a
HTTP GET.
                  connection.setChunkedStreamingMode(0);
                  BufferedOutputStream out = new
BufferedOutputStream(connection.getOutputStream());
                  try {
                        while (true) {
                              int b = in.read();
                              if (b == -1) {
                                    break;
                              }
                              out.write(b);
                        }
                        out.flush();
                  } finally {
                        out.close();
                  }
            } finally {
                  in.close();
            }
      }

      private static String getResponseType(HttpURLConnection
connection) {
            String responseType = connection.getContentType();
            return (responseType != null) ? responseType : "";
      }

      private void processResponseCookies(HttpURLConnection connection)
{
            if (allowCookies && cookieHandler != null) {
                  try {
                        Map<String, List<String>> headerFields =
connection.getHeaderFields();
                        cookieHandler.put(connection.getURL().toURI(),
headerFields);
                  } catch (URISyntaxException e) {
                        // Can't convert the URL to a URI; no cookies for
you.
                  } catch (IOException e) {
                        // Sorry, no cookies for you.
                  }
            }
      }
```

```
	private static String getResponseContent(HttpURLConnection
connection) throws IOException {
		// Use the content encoding to convert bytes to characters.
		String encoding = connection.getContentEncoding();
		if (encoding == null) {
			encoding = "UTF-8";
		}
		InputStreamReader reader = new
InputStreamReader(getConnectionStream(connection), encoding);
		try {
			int contentLength = connection.getContentLength();
			StringBuilder sb = (contentLength != -1) ? new
StringBuilder(contentLength) : new StringBuilder();
			char[] buf = new char[1024];
			int read;
			while ((read = reader.read(buf)) != -1) {
				sb.append(buf, 0, read);
			}
			return sb.toString();
		} finally {
			reader.close();
		}
	}

	private static String saveResponseContent(HttpURLConnection
connection, String responseFileName,
			String responseType) throws IOException {
		File file = createFile(responseFileName, responseType);

		BufferedInputStream in = new
BufferedInputStream(getConnectionStream(connection), 0x1000);
		try {
			BufferedOutputStream out = new
BufferedOutputStream(new FileOutputStream(file), 0x1000);
			try {
				// Copy the contents from the input stream to the
output stream.
				while (true) {
					int b = in.read();
					if (b == -1) {
						break;
					}
					out.write(b);
				}
```

```
                        out.flush();
                  } finally {
                        out.close();
                  }
            } finally {
                  in.close();
            }

            return file.getAbsolutePath();
      }

      private static InputStream getConnectionStream(HttpURLConnection
connection) {
            // According to the Android reference documentation for
            // HttpURLConnection: If the HTTP response
            // indicates that an error occurred, getInputStream() will
throw an
            // IOException. Use
            // getErrorStream() to read the error response.
            try {
                  return connection.getInputStream();
            } catch (IOException e1) {
                  // Use the error response.
                  return connection.getErrorStream();
            }
      }

      private static File createFile(String fileName, String
responseType) throws IOException, FileUtil.FileException {
            // If a fileName was specified, use it.
            if (!TextUtils.isEmpty(fileName)) {
                  return FileUtil.getExternalFile(fileName);
            }

            // Otherwise, try to determine an appropriate file extension
from the
            // responseType.
            // The response type could contain extra information that we
don't need.
            // For example, it might
            // be "text/html; charset=ISO-8859-1". We just want to look
at the part
            // before the semicolon.
            int indexOfSemicolon = responseType.indexOf(';');
            if (indexOfSemicolon != -1) {
```

```
                responseType = responseType.substring(0,
indexOfSemicolon);
            }
            String extension = mimeTypeToExtension.get(responseType);
            if (extension == null) {
                  extension = "tmp";
            }
            return FileUtil.getDownloadFile(extension);
      }

      /*
       * Converts request headers (a YailList) into the structure that
can be used
       * with the Java API (a Map<String, List<String>>). If the request
headers
       * contains an invalid element, an InvalidRequestHeadersException
will be
       * thrown.
       */
      private static Map<String, List<String>>
processRequestHeaders(YailList list)
                  throws InvalidRequestHeadersException {
            Map<String, List<String>> requestHeadersMap =
Maps.newHashMap();
            for (int i = 0; i < list.size(); i++) {
                  Object item = list.getObject(i);
                  // Each item must be a two-element sublist.
                  if (item instanceof YailList) {
                        YailList sublist = (YailList) item;
                        if (sublist.size() == 2) {
                              // The first element is the request header
field name.
                              String fieldName =
sublist.getObject(0).toString();
                              // The second element contains the request
header field
                              // values.
                              Object fieldValues = sublist.getObject(1);

                              // Build an entry (key and values) for the
                              // requestHeadersMap.
                              String key = fieldName;
                              List<String> values = Lists.newArrayList();

                              // If there is just one field value, it is
```

```
specified as a
                              // single non-list item (for
                              // example, it can be a text value). If
there are multiple
                              // field values, they are
                              // specified as a list.
                              if (fieldValues instanceof YailList) {
                                    // It's a list. There are multiple
field values.
                                    YailList multipleFieldsValues =
(YailList) fieldValues;
                                    for (int j = 0; j <
multipleFieldsValues.size(); j++) {
                                          Object value =
multipleFieldsValues.getObject(j);
                                          values.add(value.toString());
                                    }
                              } else {
                                    // It's a single non-list item. There
is just one field
                                    // value.
                                    Object singleFieldValue =
fieldValues;

values.add(singleFieldValue.toString());
                              }
                              // Put the entry into the
requestHeadersMap.
                              requestHeadersMap.put(key, values);
                        } else {
                              // The sublist doesn't contain two
elements.
                              throw new
InvalidRequestHeadersException(ErrorMessages.ERROR_WEB_REQUEST_HEADER_NO
T_TWO_ELEMENTS,
                                          i + 1);
                        }
                  } else {
                        // The item isn't a sublist.
                        throw new
InvalidRequestHeadersException(ErrorMessages.ERROR_WEB_REQUEST_HEADER_NO
T_LIST, i + 1);
                  }
            }
            return requestHeadersMap;
```

```
      }

      /*
       * Captures the current property values that are needed for an
HTTP request.
       * If an error occurs while validating the Url or RequestHeaders
property
       * values, this method calls form.dispatchErrorOccurredEvent and
returns
       * null.
       *
       * @param functionName the name of the function, used when
dispatching
       * errors
       */
      private CapturedProperties capturePropertyValues(String
functionName) {
            try {
                  return new CapturedProperties(this);
            } catch (MalformedURLException e) {
                  form.dispatchErrorOccurredEvent(this, functionName,
ErrorMessages.ERROR_WEB_MALFORMED_URL, urlString);
            } catch (InvalidRequestHeadersException e) {
                  form.dispatchErrorOccurredEvent(this, functionName,
e.errorNumber, e.index);
            }
            return null;
      }
}
```

```
package edu.mit.cne.appinventor;

// -*- mode: java; c-basic-offset: 2; -*-

// Copyright 2009-2011 Google, All Rights reserved
// Copyright 2011-2012 MIT, All rights reserved
// Released under the Apache License, Version 2.0
// http://www.apache.org/licenses/LICENSE-2.0

import com.google.appinventor.components.annotations.DesignerComponent;
import com.google.appinventor.components.annotations.DesignerProperty;
import com.google.appinventor.components.annotations.PropertyCategory;
import com.google.appinventor.components.annotations.SimpleEvent;
```

```
import com.google.appinventor.components.annotations.SimpleFunction;
import com.google.appinventor.components.annotations.SimpleObject;
import com.google.appinventor.components.annotations.SimpleProperty;
import com.google.appinventor.components.annotations.UsesLibraries;
import com.google.appinventor.components.annotations.UsesPermissions;
import com.google.appinventor.components.common.ComponentCategory;
import com.google.appinventor.components.common.HtmlEntities;
import com.google.appinventor.components.common.PropertyTypeConstants;
import com.google.appinventor.components.common.YaVersion;
import
com.google.appinventor.components.runtime.AndroidNonvisibleComponent;
import com.google.appinventor.components.runtime.Component;
import com.google.appinventor.components.runtime.ComponentContainer;
import com.google.appinventor.components.runtime.EventDispatcher;
import
com.google.appinventor.components.runtime.HandlesEventDispatching;
import com.google.appinventor.components.runtime.Web;
import com.google.appinventor.components.runtime.collect.Lists;
import com.google.appinventor.components.runtime.collect.Maps;
import com.google.appinventor.components.runtime.util.AsynchUtil;
import com.google.appinventor.components.runtime.util.ErrorMessages;
import com.google.appinventor.components.runtime.util.FileUtil;
import com.google.appinventor.components.runtime.util.GingerbreadUtil;
import com.google.appinventor.components.runtime.util.JsonUtil;
import com.google.appinventor.components.runtime.util.MediaUtil;
import com.google.appinventor.components.runtime.util.SdkLevel;
import com.google.appinventor.components.runtime.util.YailList;

import android.app.Activity;
import android.text.TextUtils;
import android.util.Log;


import org.json.JSONException;
import org.json.JSONObject;
import org.json.XML;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
```

```
import java.net.CookieHandler;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.List;
import java.util.Map;

/**
 * IoT components for connection
 *
 * @author michel.miola@ufsc.grad.br
 */
@DesignerComponent(version = YaVersion.WEB_COMPONENT_VERSION,
description = "Component lamp to turn on or off.", category =
ComponentCategory.EXTENSION, nonVisible = true, iconName =
"images/extension.png")
@SimpleObject(external = true)
@UsesPermissions(permissionNames = "android.permission.INTERNET," +
"android.permission.WRITE_EXTERNAL_STORAGE,"
            + "android.permission.READ_EXTERNAL_STORAGE")
@UsesLibraries(libraries = "json.jar")

public class Musica extends AndroidNonvisibleComponent implements
Component {

      private final Activity activity;
      private final CookieHandler cookieHandler;

      private String urlString = "";
      private boolean allowCookies;
      private YailList requestHeaders = new YailList();
      private boolean saveResponse;
      private String responseFileName = "";
      private String enderecoServidor = "";

      private static class InvalidRequestHeadersException extends
Exception {
            /*
             * errorNumber could be:
ErrorMessages.ERROR_WEB_REQUEST_HEADER_NOT_LIST
             * ErrorMessages.ERROR_WEB_REQUEST_HEADER_NOT_TWO_ELEMENTS
             */
```

```
            final int errorNumber;
            final int index; // the index of the invalid header


            InvalidRequestHeadersException(int errorNumber, int index) {
                  super();
                  this.errorNumber = errorNumber;
                  this.index = index;
            }
      }

      /**
       * BuildRequestDataException can be thrown from buildRequestData.
It is
       * thrown if the list passed to buildRequestData contains an item
that is
       * not a list. It is thrown if the list passed to buildRequestData
contains
       * an item that is a list whose size is not 2.
       */
      // VisibleForTesting
      static class BuildRequestDataException extends Exception {
            /*
             * errorNumber could be:
             * ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT_LIST
             *
ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT_TWO_ELEMENTS
             */
            final int errorNumber;
            final int index; // the index of the invalid header

            BuildRequestDataException(int errorNumber, int index) {
                  super();
                  this.errorNumber = errorNumber;
                  this.index = index;
            }
      }

      /**
       * The CapturedProperties class captures the current property
values from a
       * Web component before an asynchronous request is made. This
avoids
       * concurrency problems if the user changes a property value after
       * initiating an asynchronous request.
```

```
       */
      private static class CapturedProperties {
            final String urlString;
            final URL url;
            final boolean allowCookies;
            final boolean saveResponse;
            final String responseFileName;
            final Map<String, List<String>> requestHeaders;
            final Map<String, List<String>> cookies;

            CapturedProperties(Musica musica) throws
MalformedURLException, InvalidRequestHeadersException {
                  urlString = musica.urlString;
                  url = new URL(urlString);
                  allowCookies = musica.allowCookies;
                  saveResponse = musica.saveResponse;
                  responseFileName = musica.responseFileName;
                  requestHeaders =
processRequestHeaders(musica.requestHeaders);

                  Map<String, List<String>> cookiesTemp = null;
                  if (allowCookies && musica.cookieHandler != null) {
                        try {
                              cookiesTemp =
musica.cookieHandler.get(url.toURI(), requestHeaders);
                        } catch (URISyntaxException e) {
                              // Can't convert the URL to a URI; no
cookies for you.
                        } catch (IOException e) {
                              // Sorry, no cookies for you.
                        }
                  }
                  cookies = cookiesTemp;
            }
      }

      private static final String LOG_TAG = "Musica";

      private static final Map<String, String> mimeTypeToExtension;
      static {
            mimeTypeToExtension = Maps.newHashMap();
            mimeTypeToExtension.put("application/pdf", "pdf");
            mimeTypeToExtension.put("application/zip", "zip");
            mimeTypeToExtension.put("audio/mpeg", "mpeg");
            mimeTypeToExtension.put("audio/mp3", "mp3");
```

```
            mimeTypeToExtension.put("audio/mp4", "mp4");
            mimeTypeToExtension.put("image/gif", "gif");
            mimeTypeToExtension.put("image/jpeg", "jpg");
            mimeTypeToExtension.put("image/png", "png");
            mimeTypeToExtension.put("image/tiff", "tiff");
            mimeTypeToExtension.put("text/plain", "txt");
            mimeTypeToExtension.put("text/html", "html");
            mimeTypeToExtension.put("text/xml", "xml");
            // TODO(lizlooney) - consider adding more mime types.
      }
      /**
       * Creates a new Web component.
       *
       * @param container
       *            the Form that this component is contained in.
       */
      public Musica(ComponentContainer container) {
            super(container.$form());
            activity = container.$context();

            cookieHandler = (SdkLevel.getLevel() >=
SdkLevel.LEVEL_GINGERBREAD) ? GingerbreadUtil.newCookieManager() : null;
      }

      /**
       * This constructor is for testing purposes only.
       */
      protected Musica() {
            super(null);
            activity = null;
            cookieHandler = null;
      }

      @DesignerProperty(editorType =
PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue = "")
      @SimpleProperty
      public void EnderecoServidor(String enderecoServidor) {
             if (enderecoServidor != null) {
                   this.enderecoServidor = enderecoServidor.trim();
             }
       }

      @SimpleFunction
      public void TocarAudio1() {
            String urlResult = "";
```

```
            if (enderecoServidor != null && enderecoServidor != "") {
                  urlResult = enderecoServidor;
            }
            urlResult += "/sound/audio1.mp3/play";

            this.urlString = urlResult;
            final CapturedProperties webProps =
capturePropertyValues("Musica");

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              performRequest(webProps, null, null,
"TOCAR_AUDIO1");
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio1",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_GET", e);

form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio1",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                        }
                  }
            });
      }

      @SimpleFunction
      public void TocarAudio2() {
            String urlResult = "";
            if (enderecoServidor != null && enderecoServidor != "") {
                  urlResult = enderecoServidor;
            }
            urlResult += "/sound/audio2.mp3/play";

            this.urlString = urlResult;
            final CapturedProperties webProps =
capturePropertyValues("Musica");

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
```

```
                              performRequest(webProps, null, null,
"TOCAR_AUDIO2");
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio2",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_GET", e);

form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio2",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                        }
                  }
            });
      }

      @SimpleFunction
      public void TocarAudio3() {
            String urlResult = "";
            if (enderecoServidor != null && enderecoServidor != "") {
                  urlResult = enderecoServidor;
            }
            urlResult += "/sound/audio3.mp3/play";

            this.urlString = urlResult;
            final CapturedProperties webProps =
capturePropertyValues("Musica");

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              performRequest(webProps, null, null,
"TOCAR_AUDIO3");
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio3",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_GET", e);

form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio3",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                        }
                  }
```

```
            });
      }

      @SimpleFunction
      public void TocarAudio4() {
            String urlResult = "";
            if (enderecoServidor != null && enderecoServidor != "") {
                  urlResult = enderecoServidor;
            }
            urlResult += "/sound/audio4.mp3/play";

            this.urlString = urlResult;
            final CapturedProperties webProps =
capturePropertyValues("Musica");

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              performRequest(webProps, null, null,
"TOCAR_AUDIO4");
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio4",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_GET", e);

form.dispatchErrorOccurredEvent(Musica.this, "tocar_audio4",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                        }
                  }
            });
      }

      @SimpleFunction
      public void PararAudio() {
            String urlResult = "";
            if (enderecoServidor != null && enderecoServidor != "") {
                  urlResult = enderecoServidor + "/sound/stop";
            }

            this.urlString = urlResult;
            final CapturedProperties webProps =
capturePropertyValues("Musica");
```

```
            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              performRequest(webProps, null, null,
"PARAR_AUDIO1");
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Musica.this, "parar_audio1",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_GET", e);

form.dispatchErrorOccurredEvent(Musica.this, "parar_audio1",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                        }
                  }
            });
      }

      /*
       * Performs an HTTP GET, POST, PUT or DELETE request using the Url
property
       * and the specified text, and retrieves the response
asynchronously.<br>
       * The characters of the text are encoded using the given
encoding.<br> If
       * the SaveResponse property is true, the response will be saved
in a file
       * and the GotFile event will be triggered. The ResponseFileName
property
       * can be used to specify the name of the file.<br> If the
SaveResponse
       * property is false, the GotText event will be triggered.
       *
       * @param text the text data for the POST or PUT request
       *
       * @param encoding the character encoding to use when sending the
text. If
       * encoding is empty or null, UTF-8 encoding will be used.
       *
       * @param functionName the name of the function, used when
dispatching
       * errors
```

```
       *
       * @param httpVerb the HTTP operation to be performed: GET, POST,
PUT or
       * DELETE
       */
      private void requestTextImpl(final String text, final String
encoding, final String functionName,
                  final String httpVerb) {
            // Capture property values before running asynchronously.
            final CapturedProperties webProps =
capturePropertyValues(functionName);
            if (webProps == null) {
                  // capturePropertyValues has already called
                  // form.dispatchErrorOccurredEvent
                  return;
            }

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        // Convert text to bytes using the encoding.
                        byte[] requestData;
                        try {
                              if (encoding == null || encoding.length()
== 0) {
                                    requestData = text.getBytes("UTF-8");
                              } else {
                                    requestData =
text.getBytes(encoding);
                              }
                        } catch (UnsupportedEncodingException e) {

form.dispatchErrorOccurredEvent(Musica.this, functionName,

ErrorMessages.ERROR_WEB_UNSUPPORTED_ENCODING, encoding);
                              return;
                        }

                        try {
                              performRequest(webProps, requestData, null,
httpVerb);
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Musica.this, functionName,
e.getErrorMessageNumber());
```

```
                    } catch (Exception e) {

form.dispatchErrorOccurredEvent(Musica.this, functionName,

ErrorMessages.ERROR_WEB_UNABLE_TO_POST_OR_PUT, text,
webProps.urlString);
                    }
                }
            });
      }

      /*
       * Perform a HTTP GET or POST request. This method is always run
on a
       * different thread than the event thread. It does not use any
property
       * value fields because the properties may be changed while it is
running.
       * Instead, it uses the parameters. If either postData or postFile
is
       * non-null, then a post request is performed. If both postData
and postFile
       * are non-null, postData takes precedence over postFile. If
postData and
       * postFile are both null, then a get request is performed. If
saveResponse
       * is true, the response will be saved in a file and the GotFile
event will
       * be triggered. responseFileName specifies the name of the file.
If
       * saveResponse is false, the GotText event will be triggered.
       *
       * This method can throw an IOException. The caller is responsible
for
       * catching it and triggering the appropriate error event.
       *
       * @param webProps the captured property values needed for the
request
       *
       * @param postData the data for the post request if it is not
coming from a
       * file, can be null
       *
       * @param postFile the path of the file containing data for the
post request
```

```
       * if it is coming from a file, can be null
       *
       * @throws IOException
       */
      private void performRequest(final CapturedProperties webProps,
byte[] postData, String postFile, String httpVerb)
                  throws IOException {

            // Open the connection.
            HttpURLConnection connection = openConnection(webProps,
httpVerb);
            if (connection != null) {
                  try {
                        if (postData != null) {
                              writeRequestData(connection, postData);
                        } else if (postFile != null) {
                              writeRequestFile(connection, postFile);
                        }

                        // Get the response.
                        final int responseCode =
connection.getResponseCode();
                        final String responseType =
getResponseType(connection);

                        if (!saveResponse) {
                              final String musica =
getResponseContent(connection);

                              // Dispatch the event.
                              activity.runOnUiThread(new Runnable() {
                                    @Override
                                    public void run() {
                                          //Capturar(musica);
                                    }
                              });
                        }

                  } finally {
                        connection.disconnect();
                  }
            }
      }

      /**
```

```
       * Open a connection to the resource and set the HTTP action to
PUT or
       * DELETE if it is one of them. GET would be the default, and POST
is set in
       * writeRequestData or writeRequestFile
       *
       * @param webProps
       *            the properties of the connection, set as properties
in the
       *            component
       * @param httpVerb
       *            One of GET/POST/PUT/DELETE
       * @return a HttpURL Connection
       * @throws IOException
       * @throws ClassCastException
       * @throws ProtocolException
       *             thrown if the method in setRequestMethod is not
correct
       */
      private static HttpURLConnection openConnection(CapturedProperties
webProps, String httpVerb)
                  throws IOException, ClassCastException,
ProtocolException {

            HttpURLConnection connection = (HttpURLConnection)
webProps.url.openConnection();

            if (httpVerb.equals("PUT") || httpVerb.equals("DELETE")) {
                  // Set the Request Method; GET is the default, and if
it is a POST,
                  // it will be marked as such
                  // with setDoOutput in writeRequestFile or
writeRequestData
                  connection.setRequestMethod(httpVerb);
            }

            // Request Headers
            for (Map.Entry<String, List<String>> header :
webProps.requestHeaders.entrySet()) {
                  String name = header.getKey();
                  for (String value : header.getValue()) {
                        connection.addRequestProperty(name, value);
                  }
            }
```

```
            // Cookies
            if (webProps.cookies != null) {
                  for (Map.Entry<String, List<String>> cookie :
webProps.cookies.entrySet()) {
                        String name = cookie.getKey();
                        for (String value : cookie.getValue()) {
                              connection.addRequestProperty(name, value);
                        }
                  }
            }

            return connection;
      }

      private static void writeRequestData(HttpURLConnection connection,
byte[] postData) throws IOException {
            // According to the documentation at
            //
http://developer.android.com/reference/java/net/HttpURLConnection.html
            // HttpURLConnection uses the GET method by default. It will
use POST if
            // setDoOutput(true) has
            // been called.
            connection.setDoOutput(true); // This makes it something
other than a
                                                                  // HTTP GET.
            // Write the data.
            connection.setFixedLengthStreamingMode(postData.length);
            BufferedOutputStream out = new
BufferedOutputStream(connection.getOutputStream());
            try {
                  out.write(postData, 0, postData.length);
                  out.flush();
            } finally {
                  out.close();
            }
      }

      private void writeRequestFile(HttpURLConnection connection, String
path) throws IOException {
            // Use MediaUtil.openMedia to open the file. This means that
path could
            // be file on the SD card,
            // an asset, a contact picture, etc.
            BufferedInputStream in = new
```

```
BufferedInputStream(MediaUtil.openMedia(form, path));
            try {
                  // Write the file's data.
                  // According to the documentation at
                  //
http://developer.android.com/reference/java/net/HttpURLConnection.html
                  // HttpURLConnection uses the GET method by default.
It will use
                  // POST if setDoOutput(true) has
                  // been called.
                  connection.setDoOutput(true); // This makes it
something other than
                                                                  // a
HTTP GET.
                  connection.setChunkedStreamingMode(0);
                  BufferedOutputStream out = new
BufferedOutputStream(connection.getOutputStream());
                  try {
                        while (true) {
                              int b = in.read();
                              if (b == -1) {
                                    break;
                              }
                              out.write(b);
                        }
                        out.flush();
                  } finally {
                        out.close();
                  }
            } finally {
                  in.close();
            }
      }

      private static String getResponseType(HttpURLConnection
connection) {
            String responseType = connection.getContentType();
            return (responseType != null) ? responseType : "";
      }

      private static String getResponseContent(HttpURLConnection
connection) throws IOException {
            // Use the content encoding to convert bytes to characters.
            String encoding = connection.getContentEncoding();
            if (encoding == null) {
```

```
                  encoding = "UTF-8";
            }
            InputStreamReader reader = new
InputStreamReader(getConnectionStream(connection), encoding);
            try {
                  int contentLength = connection.getContentLength();
                  StringBuilder sb = (contentLength != -1) ? new
StringBuilder(contentLength) : new StringBuilder();
                  char[] buf = new char[1024];
                  int read;
                  while ((read = reader.read(buf)) != -1) {
                        sb.append(buf, 0, read);
                  }
                  return sb.toString();
            } finally {
                  reader.close();
            }
      }

      private static String saveResponseContent(HttpURLConnection
connection, String responseFileName,
                  String responseType) throws IOException {
            File file = createFile(responseFileName, responseType);

            BufferedInputStream in = new
BufferedInputStream(getConnectionStream(connection), 0x1000);
            try {
                  BufferedOutputStream out = new
BufferedOutputStream(new FileOutputStream(file), 0x1000);
                  try {
                        // Copy the contents from the input stream to the
output stream.
                        while (true) {
                              int b = in.read();
                              if (b == -1) {
                                    break;
                              }
                              out.write(b);
                        }
                        out.flush();
                  } finally {
                        out.close();
                  }
            } finally {
                  in.close();
```

```
            }

            return file.getAbsolutePath();
      }

      private static InputStream getConnectionStream(HttpURLConnection
connection) {
            // According to the Android reference documentation for
            // HttpURLConnection: If the HTTP response
            // indicates that an error occurred, getInputStream() will
throw an
            // IOException. Use
            // getErrorStream() to read the error response.
            try {
                  return connection.getInputStream();
            } catch (IOException e1) {
                  // Use the error response.
                  return connection.getErrorStream();
            }
      }

      private static File createFile(String fileName, String
responseType) throws IOException, FileUtil.FileException {
            // If a fileName was specified, use it.
            if (!TextUtils.isEmpty(fileName)) {
                  return FileUtil.getExternalFile(fileName);
            }

            // Otherwise, try to determine an appropriate file extension
from the
            // responseType.
            // The response type could contain extra information that we
don't need.
            // For example, it might
            // be "text/html; charset=ISO-8859-1". We just want to look
at the part
            // before the semicolon.
            int indexOfSemicolon = responseType.indexOf(';');
            if (indexOfSemicolon != -1) {
                  responseType = responseType.substring(0,
indexOfSemicolon);
            }
            String extension = mimeTypeToExtension.get(responseType);
            if (extension == null) {
                  extension = "tmp";
```

```
            }
            return FileUtil.getDownloadFile(extension);
      }

      /*
       * Converts request headers (a YailList) into the structure that
can be used
       * with the Java API (a Map<String, List<String>>). If the request
headers
       * contains an invalid element, an InvalidRequestHeadersException
will be
       * thrown.
       */
      private static Map<String, List<String>>
processRequestHeaders(YailList list)
                  throws InvalidRequestHeadersException {
            Map<String, List<String>> requestHeadersMap =
Maps.newHashMap();
            for (int i = 0; i < list.size(); i++) {
                  Object item = list.getObject(i);
                  // Each item must be a two-element sublist.
                  if (item instanceof YailList) {
                        YailList sublist = (YailList) item;
                        if (sublist.size() == 2) {
                              // The first element is the request header
field name.
                              String fieldName =
sublist.getObject(0).toString();
                              // The second element contains the request
header field
                              // values.
                              Object fieldValues = sublist.getObject(1);

                              // Build an entry (key and values) for the
                              // requestHeadersMap.
                              String key = fieldName;
                              List<String> values = Lists.newArrayList();

                              // If there is just one field value, it is
specified as a
                              // single non-list item (for
                              // example, it can be a text value). If
there are multiple
                              // field values, they are
                              // specified as a list.
```

```
                                if (fieldValues instanceof YailList) {
                                      // It's a list. There are multiple
field values.
                                      YailList multipleFieldsValues =
(YailList) fieldValues;
                                      for (int j = 0; j <
multipleFieldsValues.size(); j++) {
                                            Object value =
multipleFieldsValues.getObject(j);
                                            values.add(value.toString());
                                      }
                                } else {
                                      // It's a single non-list item. There
is just one field
                                      // value.
                                      Object singleFieldValue =
fieldValues;

values.add(singleFieldValue.toString());
                                }
                                // Put the entry into the
requestHeadersMap.
                                requestHeadersMap.put(key, values);
                          } else {
                                // The sublist doesn't contain two
elements.
                                throw new
InvalidRequestHeadersException(ErrorMessages.ERROR_WEB_REQUEST_HEADER_NO
T_TWO_ELEMENTS,
                                            i + 1);
                          }
                    } else {
                          // The item isn't a sublist.
                          throw new
InvalidRequestHeadersException(ErrorMessages.ERROR_WEB_REQUEST_HEADER_NO
T_LIST, i + 1);
                    }
              }
              return requestHeadersMap;
        }

        /*
         * Captures the current property values that are needed for an
HTTP request.
         * If an error occurs while validating the Url or RequestHeaders
```

```
property
       * values, this method calls form.dispatchErrorOccurredEvent and
returns
       * null.
       *
       * @param functionName the name of the function, used when
dispatching
       * errors
       */
      private CapturedProperties capturePropertyValues(String
functionName) {
            try {
                  return new CapturedProperties(this);
            } catch (MalformedURLException e) {
                  form.dispatchErrorOccurredEvent(this, functionName,
ErrorMessages.ERROR_WEB_MALFORMED_URL, urlString);
            } catch (InvalidRequestHeadersException e) {
                  form.dispatchErrorOccurredEvent(this, functionName,
e.errorNumber, e.index);
            }
            return null;
      }
}
```

```
// -*- mode: java; c-basic-offset: 2; -*-
// Copyright 2009-2011 Google, All Rights reserved
// Copyright 2011-2012 MIT, All rights reserved
// Released under the Apache License, Version 2.0
// http://www.apache.org/licenses/LICENSE-2.0

package edu.mit.cne.appinventor;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.CookieHandler;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
```

```
import java.net.ProtocolException;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.List;
import java.util.Map;

import org.json.JSONException;

import com.google.appinventor.components.annotations.DesignerComponent;
import com.google.appinventor.components.annotations.DesignerProperty;
import com.google.appinventor.components.annotations.PropertyCategory;
import com.google.appinventor.components.annotations.SimpleFunction;
import com.google.appinventor.components.annotations.SimpleObject;
import com.google.appinventor.components.annotations.SimpleProperty;
import com.google.appinventor.components.annotations.UsesLibraries;
import com.google.appinventor.components.annotations.UsesPermissions;
import com.google.appinventor.components.common.ComponentCategory;
import com.google.appinventor.components.common.PropertyTypeConstants;
import com.google.appinventor.components.common.YaVersion;
import
com.google.appinventor.components.runtime.AndroidNonvisibleComponent;
import com.google.appinventor.components.runtime.Component;
import com.google.appinventor.components.runtime.ComponentContainer;
import com.google.appinventor.components.runtime.collect.Lists;
import com.google.appinventor.components.runtime.collect.Maps;
import com.google.appinventor.components.runtime.util.AsynchUtil;
import com.google.appinventor.components.runtime.util.ErrorMessages;
import com.google.appinventor.components.runtime.util.FileUtil;
import com.google.appinventor.components.runtime.util.GingerbreadUtil;
import com.google.appinventor.components.runtime.util.JsonUtil;
import com.google.appinventor.components.runtime.util.MediaUtil;
import com.google.appinventor.components.runtime.util.SdkLevel;
import com.google.appinventor.components.runtime.util.YailList;

import android.app.Activity;
import android.text.TextUtils;
import android.util.Log;

/**
 * IoT components for connection
 *
 * @author michel.miola@ufsc.grad.br
 */
@DesignerComponent(version = YaVersion.WEB_COMPONENT_VERSION,
                              description = "Component lamp to turn on or
```

```
off.",
                              category = ComponentCategory.EXTENSION,
                              nonVisible = true,
                              iconName = "images/extension.png"
)
@SimpleObject(external = true)
@UsesPermissions(permissionNames = "android.permission.INTERNET,"
            + "android.permission.WRITE_EXTERNAL_STORAGE,"
            + "android.permission.READ_EXTERNAL_STORAGE")
@UsesLibraries(libraries = "json.jar")
public class Interruptor extends AndroidNonvisibleComponent implements
Component {

      private final Activity activity;
      private final CookieHandler cookieHandler;

      private String urlString = "";
      private YailList requestHeaders = new YailList();
      private boolean saveResponse;
      private String responseFileName = "";
      private final Integer STATUS_HTTP_NOT_FOUND = 404;
      private String enderecoServidor = "";
      private final String GPIO_DEFAULT = "19";
      private static final String ATIVO = "ativo";
      private static final String INATIVO = "inativo";
      private String GPIO = "";

      /**
      * InvalidRequestHeadersException can be thrown from
processRequestHeaders.
       * It is thrown if the list passed to processRequestHeaders
contains an item
       * that is not a list. It is thrown if the list passed to
       * processRequestHeaders contains an item that is a list whose
size is not
       * 2.
       */
      private static class InvalidRequestHeadersException extends
Exception {
            /*
             * errorNumber could be:
ErrorMessages.ERROR_WEB_REQUEST_HEADER_NOT_LIST
             * ErrorMessages.ERROR_WEB_REQUEST_HEADER_NOT_TWO_ELEMENTS
             */
            final int errorNumber;
```

```
            final int index; // the index of the invalid header

            InvalidRequestHeadersException(int errorNumber, int index) {
                  super();
                  this.errorNumber = errorNumber;
                  this.index = index;
            }
      }

      /**
       * The CapturedProperties class captures the current property
values from a
       * Web component before an asynchronous request is made. This
avoids
       * concurrency problems if the user changes a property value after
       * initiating an asynchronous request.
       */
      private static class CapturedProperties {
            final String urlString;
            final URL url;
            final boolean saveResponse;
            final String responseFileName;
            final Map<String, List<String>> requestHeaders;
            final Map<String, List<String>> cookies;

            CapturedProperties(Interruptor sensors) throws
MalformedURLException, InvalidRequestHeadersException {
                  urlString = sensors.urlString;
                  url = new URL(urlString);
                  saveResponse = sensors.saveResponse;
                  responseFileName = sensors.responseFileName;
                  requestHeaders =
processRequestHeaders(sensors.requestHeaders);

                  Map<String, List<String>> cookiesTemp = null;
                  if (sensors.cookieHandler != null) {
                        try {
                              cookiesTemp =
sensors.cookieHandler.get(url.toURI(), requestHeaders);
                        } catch (URISyntaxException e) {
                              // Can't convert the URL to a URI; no
cookies for you.
                        } catch (IOException e) {
                              // Sorry, no cookies for you.
                        }
```

```
                }
                cookies = cookiesTemp;
            }
        }

        private static final String LOG_TAG = "IOTCnE";

        private static final Map<String, String> mimeTypeToExtension;

        static {
            mimeTypeToExtension = Maps.newHashMap();
            mimeTypeToExtension.put("application/pdf", "pdf");
            mimeTypeToExtension.put("application/zip", "zip");
            mimeTypeToExtension.put("audio/mpeg", "mpeg");
            mimeTypeToExtension.put("audio/mp3", "mp3");
            mimeTypeToExtension.put("audio/mp4", "mp4");
            mimeTypeToExtension.put("image/gif", "gif");
            mimeTypeToExtension.put("image/jpeg", "jpg");
            mimeTypeToExtension.put("image/png", "png");
            mimeTypeToExtension.put("image/tiff", "tiff");
            mimeTypeToExtension.put("text/plain", "txt");
            mimeTypeToExtension.put("text/html", "html");
            mimeTypeToExtension.put("text/xml", "xml");
            // TODO(lizlooney) - consider adding more mime types.
        }

        /**
         * Creates a new Web component.
         *
         * @param container
         *            the Form that this component is contained in.
         */
        public Interruptor(ComponentContainer container) {
            super(container.$form());
            activity = container.$context();

            cookieHandler = (SdkLevel.getLevel() >=
SdkLevel.LEVEL_GINGERBREAD) ? GingerbreadUtil.newCookieManager() : null;
        }

        /**
         * This constructor is for testing purposes only.
         */
        public Interruptor() {
            super(null);
```

```
            activity = null;
            cookieHandler = null;
      }

      @DesignerProperty(editorType =
PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue = "")
      @SimpleProperty
      public void EnderecoServidor(String enderecoServidor) {
             if (enderecoServidor != null) {
                   this.enderecoServidor = enderecoServidor.trim();
             }
       }

      @DesignerProperty(editorType =
PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue = "19")
      @SimpleProperty
      public void Gpio(String Gpio) {
             if (Gpio != null) {
                   this.GPIO = Gpio.trim();
             }
       }

      @SimpleFunction
      public void Ligar() {
            if (this.GPIO == null || this.GPIO.isEmpty()) {
                  this.GPIO = this.GPIO_DEFAULT;
            }
            String urlWithPath = mounUrlWithPath(this.enderecoServidor,
this.GPIO, ATIVO);
            this.urlString = urlWithPath;
            final CapturedProperties webProps =
capturePropertyValues("Relay");
            if (this.enderecoServidor == "" || this.GPIO == "") {
                  return;
            }

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              Integer responseCode =
performRequest(webProps, null, null, "GET");
                              if (responseCode == STATUS_HTTP_NOT_FOUND){

form.dispatchErrorOccurredEvent(Interruptor.this, "Relay",
```

```
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                              }

                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Interruptor.this, "Relay",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_RELAY", e);

form.dispatchErrorOccurredEvent(Interruptor.this, "Relay",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                        }
                  }
            });
      }

      @SimpleFunction
      public void Desligar() {
            if (this.GPIO == null || this.GPIO.isEmpty()) {
                  this.GPIO = this.GPIO_DEFAULT;
            }
            String urlWithPath = mounUrlWithPath(this.enderecoServidor,
this.GPIO, INATIVO);
            this.urlString = urlWithPath;
            final CapturedProperties webProps =
capturePropertyValues("Relay");
            if (this.enderecoServidor == "" || this.GPIO == "") {
                  return;
            }

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              Integer responseCode =
performRequest(webProps, null, null, "GET");
                              if (responseCode == STATUS_HTTP_NOT_FOUND){

form.dispatchErrorOccurredEvent(Interruptor.this, "Relay",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                              }

                        } catch (FileUtil.FileException e) {
```

```
form.dispatchErrorOccurredEvent(Interruptor.this, "Relay",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_RELAY", e);

form.dispatchErrorOccurredEvent(Interruptor.this, "Relay",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                        }
                  }
            });
      }

      public static String mounUrlWithPath(String enderecoServidor,
String gpio, String statusRelay) {
            String urlResult = "";
            if (enderecoServidor != null && enderecoServidor != "") {
                  urlResult = enderecoServidor;
            }
            if (gpio != null && gpio != "") {
                  urlResult += "/relay/" + gpio;
            }
            if (statusRelay != null && statusRelay != "") {
                  urlResult += "/status/" + statusRelay;
            }
            return urlResult;
      }

      /*
       * Performs an HTTP GET, POST, PUT or DELETE request using the Url
property
       * and the specified text, and retrieves the response
asynchronously.<br>
       * The characters of the text are encoded using the given
encoding.<br> If
       * the SaveResponse property is true, the response will be saved
in a file
       * and the GotFile event will be triggered. The ResponseFileName
property
       * can be used to specify the name of the file.<br> If the
SaveResponse
       * property is false, the GotText event will be triggered.
       *
       * @param text the text data for the POST or PUT request
       *
       * @param encoding the character encoding to use when sending the
```

```
text. If
       * encoding is empty or null, UTF-8 encoding will be used.
       *
       * @param functionName the name of the function, used when
dispatching
       * errors
       *
       * @param httpVerb the HTTP operation to be performed: GET, POST,
PUT or
       * DELETE
       */
      private void requestTextImpl(final String text, final String
encoding, final String functionName, final String httpVerb) {
            // Capture property values before running asynchronously.
            final CapturedProperties webProps =
capturePropertyValues(functionName);
            if (webProps == null) {
                  // capturePropertyValues has already called
                  // form.dispatchErrorOccurredEvent
                  return;
            }

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        // Convert text to bytes using the encoding.
                        byte[] requestData;
                        try {
                              if (encoding == null || encoding.length()
== 0) {
                                    requestData = text.getBytes("UTF-8");
                              } else {
                                    requestData =
text.getBytes(encoding);
                              }
                        } catch (UnsupportedEncodingException e) {

form.dispatchErrorOccurredEvent(Interruptor.this, functionName,

ErrorMessages.ERROR_WEB_UNSUPPORTED_ENCODING, encoding);
                              return;
                        }

                        try {
                              performRequest(webProps, requestData, null,
```

```
httpVerb);
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Interruptor.this, functionName,
e.getErrorMessageNumber());
                        } catch (Exception e) {

form.dispatchErrorOccurredEvent(Interruptor.this, functionName,

ErrorMessages.ERROR_WEB_UNABLE_TO_POST_OR_PUT, text,
webProps.urlString);
                        }
                    }
                });
        }

        /*
         * Perform a HTTP GET or POST request. This method is always run
on a
         * different thread than the event thread. It does not use any
property
         * value fields because the properties may be changed while it is
running.
         * Instead, it uses the parameters. If either postData or postFile
is
         * non-null, then a post request is performed. If both postData
and postFile
         * are non-null, postData takes precedence over postFile. If
postData and
         * postFile are both null, then a get request is performed. If
saveResponse
         * is true, the response will be saved in a file and the GotFile
event will
         * be triggered. responseFileName specifies the name of the file.
If
         * saveResponse is false, the GotText event will be triggered.
         *
         * This method can throw an IOException. The caller is responsible
for
         * catching it and triggering the appropriate error event.
         *
         * @param webProps the captured property values needed for the
request
         *
         * @param postData the data for the post request if it is not
```

```
coming from a
       * file, can be null
       *
       * @param postFile the path of the file containing data for the
post request
       * if it is coming from a file, can be null
       *
       * @throws IOException
       */
      private Integer performRequest(final CapturedProperties webProps,
byte[] postData, String postFile, String httpVerb)
                  throws IOException {
            Integer responseCode = null;
            // Open the connection.
            HttpURLConnection connection = openConnection(webProps,
httpVerb);
            if (connection != null) {
                  try {
                        if (postData != null) {
                              writeRequestData(connection, postData);
                        } else if (postFile != null) {
                              writeRequestFile(connection, postFile);
                        }

                        // Get the response.
                        responseCode = connection.getResponseCode();
                        final String responseType =
getResponseType(connection);

                        if (saveResponse) {
                              final String path =
saveResponseContent(connection, webProps.responseFileName,
responseType);

                              // Dispatch the event.
                              activity.runOnUiThread(new Runnable() {
                                    @Override
                                    public void run() {
                                          // GotFile(webProps.urlString,
responseCode,
                                          // responseType, path);
                                    }
                              });
                        } else {
                              final String responseContent =
```

```
getResponseContent(connection);

                              // Dispatch the event.
                              activity.runOnUiThread(new Runnable() {
                                    @Override
                                    public void run() {
                                          // GotText(webProps.urlString,
responseCode,
                                          // responseType,
responseContent);
                                    }
                              });
                        }

                  } finally {
                        connection.disconnect();
                  }
            }
            return responseCode;
      }

      /**
       * Open a connection to the resource and set the HTTP action to
PUT or
       * DELETE if it is one of them. GET would be the default, and POST
is set in
       * writeRequestData or writeRequestFile
       *
       * @param webProps
       *            the properties of the connection, set as properties
in the
       *            component
       * @param httpVerb
       *            One of GET/POST/PUT/DELETE
       * @return a HttpURL Connection
       * @throws IOException
       * @throws ClassCastException
       * @throws ProtocolException
       *             thrown if the method in setRequestMethod is not
correct
       */
      private static HttpURLConnection openConnection(CapturedProperties
webProps, String httpVerb)
                  throws IOException, ClassCastException,
ProtocolException {
```

```
            HttpURLConnection connection = (HttpURLConnection)
webProps.url.openConnection();

            if (httpVerb.equals("PUT") || httpVerb.equals("DELETE")) {
                  // Set the Request Method; GET is the default, and if
it is a POST,
                  // it will be marked as such
                  // with setDoOutput in writeRequestFile or
writeRequestData
                  connection.setRequestMethod(httpVerb);
            }

            // Request Headers
            for (Map.Entry<String, List<String>> header :
webProps.requestHeaders.entrySet()) {
                  String name = header.getKey();
                  for (String value : header.getValue()) {
                        connection.addRequestProperty(name, value);
                  }
            }

            // Cookies
            if (webProps.cookies != null) {
                  for (Map.Entry<String, List<String>> cookie :
webProps.cookies.entrySet()) {
                        String name = cookie.getKey();
                        for (String value : cookie.getValue()) {
                              connection.addRequestProperty(name, value);
                        }
                  }
            }

            return connection;
      }

      private static void writeRequestData(HttpURLConnection connection,
byte[] postData) throws IOException {
            // According to the documentation at
            //
http://developer.android.com/reference/java/net/HttpURLConnection.html
            // HttpURLConnection uses the GET method by default. It will
use POST if
            // setDoOutput(true) has
            // been called.
```

```
            connection.setDoOutput(true); // This makes it something
other than a
                                                                  // HTTP GET.
            // Write the data.
            connection.setFixedLengthStreamingMode(postData.length);
            BufferedOutputStream out = new
BufferedOutputStream(connection.getOutputStream());
            try {
                  out.write(postData, 0, postData.length);
                  out.flush();
            } finally {
                  out.close();
            }
      }

      private void writeRequestFile(HttpURLConnection connection, String
path) throws IOException {
            // Use MediaUtil.openMedia to open the file. This means that
path could
            // be file on the SD card,
            // an asset, a contact picture, etc.
            BufferedInputStream in = new
BufferedInputStream(MediaUtil.openMedia(form, path));
            try {
                  // Write the file's data.
                  // According to the documentation at
                  //
http://developer.android.com/reference/java/net/HttpURLConnection.html
                  // HttpURLConnection uses the GET method by default.
It will use
                  // POST if setDoOutput(true) has
                  // been called.
                  connection.setDoOutput(true); // This makes it
something other than
                                                                  // a
HTTP GET.
                  connection.setChunkedStreamingMode(0);
                  BufferedOutputStream out = new
BufferedOutputStream(connection.getOutputStream());
                  try {
                        while (true) {
                              int b = in.read();
                              if (b == -1) {
                                    break;
                              }
```

```
                        out.write(b);
                    }
                    out.flush();
                } finally {
                    out.close();
                }
            } finally {
                in.close();
            }
        }

        private static String getResponseType(HttpURLConnection
connection) {
            String responseType = connection.getContentType();
            return (responseType != null) ? responseType : "";
        }

        private static String getResponseContent(HttpURLConnection
connection) throws IOException {
            // Use the content encoding to convert bytes to characters.
            String encoding = connection.getContentEncoding();
            if (encoding == null) {
                encoding = "UTF-8";
            }
            InputStreamReader reader = new
InputStreamReader(getConnectionStream(connection), encoding);
            try {
                int contentLength = connection.getContentLength();
                StringBuilder sb = (contentLength != -1) ? new
StringBuilder(contentLength) : new StringBuilder();
                char[] buf = new char[1024];
                int read;
                while ((read = reader.read(buf)) != -1) {
                    sb.append(buf, 0, read);
                }
                return sb.toString();
            } finally {
                reader.close();
            }
        }

        private static String saveResponseContent(HttpURLConnection
connection, String responseFileName,
                String responseType) throws IOException {
            File file = createFile(responseFileName, responseType);
```

```
            BufferedInputStream in = new
BufferedInputStream(getConnectionStream(connection), 0x1000);
            try {
                  BufferedOutputStream out = new
BufferedOutputStream(new FileOutputStream(file), 0x1000);
                  try {
                        // Copy the contents from the input stream to the
output stream.
                        while (true) {
                              int b = in.read();
                              if (b == -1) {
                                    break;
                              }
                              out.write(b);
                        }
                        out.flush();
                  } finally {
                        out.close();
                  }
            } finally {
                  in.close();
            }

            return file.getAbsolutePath();
      }

      private static InputStream getConnectionStream(HttpURLConnection
connection) {
            // According to the Android reference documentation for
            // HttpURLConnection: If the HTTP response
            // indicates that an error occurred, getInputStream() will
throw an
            // IOException. Use
            // getErrorStream() to read the error response.
            try {
                  return connection.getInputStream();
            } catch (IOException e1) {
                  // Use the error response.
                  return connection.getErrorStream();
            }
      }

      private static File createFile(String fileName, String
responseType) throws IOException, FileUtil.FileException {
```

```
            // If a fileName was specified, use it.
            if (!TextUtils.isEmpty(fileName)) {
                  return FileUtil.getExternalFile(fileName);
            }

            // Otherwise, try to determine an appropriate file extension
from the
            // responseType.
            // The response type could contain extra information that we
don't need.
            // For example, it might
            // be "text/html; charset=ISO-8859-1". We just want to look
at the part
            // before the semicolon.
            int indexOfSemicolon = responseType.indexOf(';');
            if (indexOfSemicolon != -1) {
                  responseType = responseType.substring(0,
indexOfSemicolon);
            }
            String extension = mimeTypeToExtension.get(responseType);
            if (extension == null) {
                  extension = "tmp";
            }
            return FileUtil.getDownloadFile(extension);
      }

      /*
       * Converts request headers (a YailList) into the structure that
can be used
       * with the Java API (a Map<String, List<String>>). If the request
headers
       * contains an invalid element, an InvalidRequestHeadersException
will be
       * thrown.
       */
      private static Map<String, List<String>>
processRequestHeaders(YailList list)
                  throws InvalidRequestHeadersException {
            Map<String, List<String>> requestHeadersMap =
Maps.newHashMap();
            for (int i = 0; i < list.size(); i++) {
                  Object item = list.getObject(i);
                  // Each item must be a two-element sublist.
                  if (item instanceof YailList) {
                        YailList sublist = (YailList) item;
```

```
                        if (sublist.size() == 2) {
                              // The first element is the request header
field name.
                              String fieldName =
sublist.getObject(0).toString();
                              // The second element contains the request
header field
                              // values.
                              Object fieldValues = sublist.getObject(1);

                              // Build an entry (key and values) for the
                              // requestHeadersMap.
                              String key = fieldName;
                              List<String> values = Lists.newArrayList();

                              // If there is just one field value, it is
specified as a
                              // single non-list item (for
                              // example, it can be a text value). If
there are multiple
                              // field values, they are
                              // specified as a list.
                              if (fieldValues instanceof YailList) {
                                    // It's a list. There are multiple
field values.
                                    YailList multipleFieldsValues =
(YailList) fieldValues;
                                    for (int j = 0; j <
multipleFieldsValues.size(); j++) {
                                          Object value =
multipleFieldsValues.getObject(j);
                                          values.add(value.toString());
                                    }
                              } else {
                                    // It's a single non-list item. There
is just one field
                                    // value.
                                    Object singleFieldValue =
fieldValues;

values.add(singleFieldValue.toString());
                              }
                              // Put the entry into the
requestHeadersMap.
                              requestHeadersMap.put(key, values);
```

```
                        } else {
                              // The sublist doesn't contain two
elements.
                              throw new
InvalidRequestHeadersException(ErrorMessages.ERROR_WEB_REQUEST_HEADER_NO
T_TWO_ELEMENTS,
                                          i + 1);
                        }
                  } else {
                        // The item isn't a sublist.
                        throw new
InvalidRequestHeadersException(ErrorMessages.ERROR_WEB_REQUEST_HEADER_NO
T_LIST, i + 1);
                  }
            }
            return requestHeadersMap;
      }

      /*
       * Captures the current property values that are needed for an
HTTP request.
       * If an error occurs while validating the Url or RequestHeaders
property
       * values, this method calls form.dispatchErrorOccurredEvent and
returns
       * null.
       *
       * @param functionName the name of the function, used when
dispatching
       * errors
       */
      private CapturedProperties capturePropertyValues(String
functionName) {
            try {
                  return new CapturedProperties(this);
            } catch (MalformedURLException e) {
                  form.dispatchErrorOccurredEvent(this, functionName,
ErrorMessages.ERROR_WEB_MALFORMED_URL, urlString);
            } catch (InvalidRequestHeadersException e) {
                  form.dispatchErrorOccurredEvent(this, functionName,
e.errorNumber, e.index);
            }
            return null;
      }
}
```

```java
package edu.mit.cne.appinventor;

// -*- mode: java; c-basic-offset: 2; -*-

// Copyright 2009-2011 Google, All Rights reserved
// Copyright 2011-2012 MIT, All rights reserved
// Released under the Apache License, Version 2.0
// http://www.apache.org/licenses/LICENSE-2.0

import com.google.appinventor.components.annotations.DesignerComponent;
import com.google.appinventor.components.annotations.DesignerProperty;
import com.google.appinventor.components.annotations.PropertyCategory;
import com.google.appinventor.components.annotations.SimpleEvent;
import com.google.appinventor.components.annotations.SimpleFunction;
import com.google.appinventor.components.annotations.SimpleObject;
import com.google.appinventor.components.annotations.SimpleProperty;
import com.google.appinventor.components.annotations.UsesLibraries;
import com.google.appinventor.components.annotations.UsesPermissions;
import com.google.appinventor.components.common.ComponentCategory;
import com.google.appinventor.components.common.HtmlEntities;
import com.google.appinventor.components.common.PropertyTypeConstants;
import com.google.appinventor.components.common.YaVersion;
import
com.google.appinventor.components.runtime.AndroidNonvisibleComponent;
import com.google.appinventor.components.runtime.Component;
import com.google.appinventor.components.runtime.ComponentContainer;
import com.google.appinventor.components.runtime.EventDispatcher;
import
com.google.appinventor.components.runtime.HandlesEventDispatching;
import com.google.appinventor.components.runtime.Web;
import com.google.appinventor.components.runtime.collect.Lists;
import com.google.appinventor.components.runtime.collect.Maps;
import com.google.appinventor.components.runtime.util.AsynchUtil;
import com.google.appinventor.components.runtime.util.ErrorMessages;
import com.google.appinventor.components.runtime.util.FileUtil;
import com.google.appinventor.components.runtime.util.GingerbreadUtil;
import com.google.appinventor.components.runtime.util.JsonUtil;
import com.google.appinventor.components.runtime.util.MediaUtil;
import com.google.appinventor.components.runtime.util.SdkLevel;
import com.google.appinventor.components.runtime.util.YailList;

import android.app.Activity;
import android.text.TextUtils;
import android.util.Log;
```

```
import org.json.JSONException;
import org.json.JSONObject;
import org.json.XML;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.CookieHandler;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.List;
import java.util.Map;

/**
 * IoT components for connection
 *
 * @author michel.miola@ufsc.grad.br
 */
@DesignerComponent(version = YaVersion.WEB_COMPONENT_VERSION,
description = "Component lamp to turn on or off.", category =
ComponentCategory.EXTENSION, nonVisible = true, iconName =
"images/extension.png")
@SimpleObject(external = true)
@UsesPermissions(permissionNames = "android.permission.INTERNET," +
"android.permission.WRITE_EXTERNAL_STORAGE,"
            + "android.permission.READ_EXTERNAL_STORAGE")
@UsesLibraries(libraries = "json.jar")

public class Umidade extends AndroidNonvisibleComponent implements
Component {

      private final Activity activity;
      private final CookieHandler cookieHandler;
```

```
      private String urlString = "";
      private boolean allowCookies;
      private YailList requestHeaders = new YailList();
      private boolean saveResponse;
      private String responseFileName = "";
      private String enderecoServidor = "";
      private final String GPIO_DEFAULT = "6";
      private String GPIO = "";

      private static class InvalidRequestHeadersException extends
Exception {
            /*
             * errorNumber could be:
ErrorMessages.ERROR_WEB_REQUEST_HEADER_NOT_LIST
             * ErrorMessages.ERROR_WEB_REQUEST_HEADER_NOT_TWO_ELEMENTS
             */
            final int errorNumber;
            final int index; // the index of the invalid header


            InvalidRequestHeadersException(int errorNumber, int index) {
                  super();
                  this.errorNumber = errorNumber;
                  this.index = index;
            }
      }

      /**
       * BuildRequestDataException can be thrown from buildRequestData.
It is
       * thrown if the list passed to buildRequestData contains an item
that is
       * not a list. It is thrown if the list passed to buildRequestData
contains
       * an item that is a list whose size is not 2.
       */
      // VisibleForTesting
      static class BuildRequestDataException extends Exception {
            /*
             * errorNumber could be:
             * ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT_LIST
             *
ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT_TWO_ELEMENTS
             */
            final int errorNumber;
```

```
            final int index; // the index of the invalid header

            BuildRequestDataException(int errorNumber, int index) {
                  super();
                  this.errorNumber = errorNumber;
                  this.index = index;
            }
      }

      /**
       * The CapturedProperties class captures the current property
values from a
       * Web component before an asynchronous request is made. This
avoids
       * concurrency problems if the user changes a property value after
       * initiating an asynchronous request.
       */
      private static class CapturedProperties {
            final String urlString;
            final URL url;
            final boolean allowCookies;
            final boolean saveResponse;
            final String responseFileName;
            final Map<String, List<String>> requestHeaders;
            final Map<String, List<String>> cookies;

            CapturedProperties(Umidade umidade) throws
MalformedURLException, InvalidRequestHeadersException {
                  urlString = umidade.urlString;
                  url = new URL(urlString);
                  allowCookies = umidade.allowCookies;
                  saveResponse = umidade.saveResponse;
                  responseFileName = umidade.responseFileName;
                  requestHeaders =
processRequestHeaders(umidade.requestHeaders);

                  Map<String, List<String>> cookiesTemp = null;
                  if (allowCookies && umidade.cookieHandler != null) {
                        try {
                              cookiesTemp =
umidade.cookieHandler.get(url.toURI(), requestHeaders);
                        } catch (URISyntaxException e) {
                              // Can't convert the URL to a URI; no
cookies for you.
                        } catch (IOException e) {
```

```
                                // Sorry, no cookies for you.
                        }
                    }
                    cookies = cookiesTemp;
                }
            }

            private static final String LOG_TAG = "Umidade";

            private static final Map<String, String> mimeTypeToExtension;
            static {
                mimeTypeToExtension = Maps.newHashMap();
                mimeTypeToExtension.put("application/pdf", "pdf");
                mimeTypeToExtension.put("application/zip", "zip");
                mimeTypeToExtension.put("audio/mpeg", "mpeg");
                mimeTypeToExtension.put("audio/mp3", "mp3");
                mimeTypeToExtension.put("audio/mp4", "mp4");
                mimeTypeToExtension.put("image/gif", "gif");
                mimeTypeToExtension.put("image/jpeg", "jpg");
                mimeTypeToExtension.put("image/png", "png");
                mimeTypeToExtension.put("image/tiff", "tiff");
                mimeTypeToExtension.put("text/plain", "txt");
                mimeTypeToExtension.put("text/html", "html");
                mimeTypeToExtension.put("text/xml", "xml");
                // TODO(lizlooney) - consider adding more mime types.
            }
            /**
             * Creates a new Web component.
             *
             * @param container
             *            the Form that this component is contained in.
             */
            public Umidade(ComponentContainer container) {
                super(container.$form());
                activity = container.$context();

                cookieHandler = (SdkLevel.getLevel() >=
SdkLevel.LEVEL_GINGERBREAD) ? GingerbreadUtil.newCookieManager() : null;
            }

            /**
             * This constructor is for testing purposes only.
             */
            protected Umidade() {
                super(null);
```

```
            activity = null;
            cookieHandler = null;
      }

       @DesignerProperty(editorType =
PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue = "")
       @SimpleProperty
       public void EnderecoServidor(String enderecoServidor) {
             if (enderecoServidor != null) {
                   this.enderecoServidor = enderecoServidor.trim();
             }
       }

       @DesignerProperty(editorType =
PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue = "6")
       @SimpleProperty
       public void Gpio(String Gpio) {
             if (Gpio != null) {
                   this.GPIO = Gpio.trim();
             }
       }

      /**
       * Performs an HTTP GET request using the Url property and
retrieves the
       * response.<br>
       * If the SaveResponse property is true, the response will be
saved in a
       * file and the GotFile event will be triggered. The
ResponseFileName
       * property can be used to specify the name of the file.<br>
       * If the SaveResponse property is false, the GotText event will
be
       * triggered.
       */
      @SimpleFunction
      public void Obter() {
            if (this.GPIO == null || this.GPIO.isEmpty()) {
                  this.GPIO = this.GPIO_DEFAULT;
            }
            String urlWithPath = mounUrlWithPath(this.enderecoServidor,
this.GPIO);
            this.urlString = urlWithPath;
            final CapturedProperties webProps =
capturePropertyValues("Umidade");
```

```
            if (this.enderecoServidor == "" || this.GPIO == "") {
                  return;
            }

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              performRequest(webProps, null, null,
"OBTER");
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Umidade.this, "Obter",
e.getErrorMessageNumber());
                        } catch (Exception e) {
                              Log.e(LOG_TAG, "ERROR_UNABLE_TO_GET", e);

form.dispatchErrorOccurredEvent(Umidade.this, "Obter",
ErrorMessages.ERROR_WEB_UNABLE_TO_GET, webProps.urlString);
                        }
                  }
            });
      }

      /**
       * Event indicating that a request has finished.
       *
       * @param url
       *            the URL used for the request
       * @param responseCode
       *            the response code from the server
       * @param responseType
       *            the mime type of the response
       * @param temperatura
       *            the response content from the server
       */
      @SimpleEvent
      public void Capturar(String temperatura) {
            // invoke the application's "GotText" event handler.
            EventDispatcher.dispatchEvent(this, "Capturar", "", 200,
"JSON", temperatura);
      }

      public static String mounUrlWithPath(String enderecoServidor,
String gpio) {
```

```
            String urlResult = "";
            if (enderecoServidor != null && enderecoServidor != "") {
                  urlResult = enderecoServidor;
            }
            if (gpio != null && gpio != "") {
                  urlResult += "/Umidade/" + gpio;
            }

            return urlResult;
      }

      /*
       * Performs an HTTP GET, POST, PUT or DELETE request using the Url
property
       * and the specified text, and retrieves the response
asynchronously.<br>
       * The characters of the text are encoded using the given
encoding.<br> If
       * the SaveResponse property is true, the response will be saved
in a file
       * and the GotFile event will be triggered. The ResponseFileName
property
       * can be used to specify the name of the file.<br> If the
SaveResponse
       * property is false, the GotText event will be triggered.
       *
       * @param text the text data for the POST or PUT request
       *
       * @param encoding the character encoding to use when sending the
text. If
       * encoding is empty or null, UTF-8 encoding will be used.
       *
       * @param functionName the name of the function, used when
dispatching
       * errors
       *
       * @param httpVerb the HTTP operation to be performed: GET, POST,
PUT or
       * DELETE
       */
      private void requestTextImpl(final String text, final String
encoding, final String functionName,
                  final String httpVerb) {
            // Capture property values before running asynchronously.
            final CapturedProperties webProps =
```

```
captureProperty Values(functionName);
            if (webProps == null) {
                  // capturePropertyValues has already called
                  // form.dispatchErrorOccurredEvent
                  return;
            }

            AsynchUtil.runAsynchronously(new Runnable() {
                  @Override
                  public void run() {
                        // Convert text to bytes using the encoding.
                        byte[] requestData;
                        try {
                              if (encoding == null || encoding.length()
== 0) {
                                    requestData = text.getBytes("UTF-8");
                              } else {
                                    requestData =
text.getBytes(encoding);
                              }
                        } catch (UnsupportedEncodingException e) {

form.dispatchErrorOccurredEvent(Umidade.this, functionName,

ErrorMessages.ERROR_WEB_UNSUPPORTED_ENCODING, encoding);
                              return;
                        }

                        try {
                              performRequest(webProps, requestData, null,
httpVerb);
                        } catch (FileUtil.FileException e) {

form.dispatchErrorOccurredEvent(Umidade.this, functionName,
e.getErrorMessageNumber());
                        } catch (Exception e) {

form.dispatchErrorOccurredEvent(Umidade.this, functionName,

ErrorMessages.ERROR_WEB_UNABLE_TO_POST_OR_PUT, text,
webProps.urlString);
                        }
                  }
            });
      }
```

```
      /*
       * Converts a list of two-element sublists, representing name and
value
       * pairs, to a string formatted as
application/x-www-form-urlencoded media
       * type, suitable to pass to PostText.
       *
       * @param list a list of two-element sublists representing name
and value
       * pairs
       *
       * @throws BuildPostDataException if the list is not valid
       */
      // VisibleForTesting
      String buildRequestData(YailList list) throws
BuildRequestDataException {
            StringBuilder sb = new StringBuilder();
            String delimiter = "";
            for (int i = 0; i < list.size(); i++) {
                  Object item = list.getObject(i);
                  // Each item must be a two-element sublist.
                  if (item instanceof YailList) {
                        YailList sublist = (YailList) item;
                        if (sublist.size() == 2) {
                              // The first element is the name.
                              String name =
sublist.getObject(0).toString();
                              // The second element is the value.
                              String value =
sublist.getObject(1).toString();

sb.append(delimiter).append(UriEncode(name)).append('=').append(UriEncod
e(value));
                        } else {
                              throw new
BuildRequestDataException(ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT
_TWO_ELEMENTS,
                                          i + 1);
                        }
                  } else {
                        throw new
BuildRequestDataException(ErrorMessages.ERROR_WEB_BUILD_REQUEST_DATA_NOT
_LIST, i + 1);
                  }
```

```
                delimiter = "&";
            }
            return sb.toString();
      }

      public String UriEncode(String text) {
            try {
                  return URLEncoder.encode(text, "UTF-8");
            } catch (UnsupportedEncodingException e) {
                  // If UTF-8 is not supported, we're in big trouble!
                  // According to Javadoc and Android documentation for
                  // java.nio.charset.Charset, UTF-8 is
                  // available on every Java implementation.
                  Log.e(LOG_TAG, "UTF-8 is unsupported?", e);
                  return "";
            }
      }

      /**
       * Decodes the given JSON encoded value.
       *
       * @param jsonText
       *            the JSON text to decode
       * @return the decoded object
       * @throws IllegalArgumentException
       *             if the JSON text can't be decoded
       */
      // VisibleForTesting
      static Object decodeJsonText(String jsonText) throws
IllegalArgumentException {
            try {
                  return JsonUtil.getObjectFromJson(jsonText);
            } catch (JSONException e) {
                  throw new IllegalArgumentException("jsonText is not a
legal JSON value");
            }
      }

      /*
       * Perform a HTTP GET or POST request. This method is always run
on a
       * different thread than the event thread. It does not use any
property
       * value fields because the properties may be changed while it is
running.
```

```
       * Instead, it uses the parameters. If either postData or postFile
is
       * non-null, then a post request is performed. If both postData
and postFile
       * are non-null, postData takes precedence over postFile. If
postData and
       * postFile are both null, then a get request is performed. If
saveResponse
       * is true, the response will be saved in a file and the GotFile
event will
       * be triggered. responseFileName specifies the name of the file.
If
       * saveResponse is false, the GotText event will be triggered.
       *
       * This method can throw an IOException. The caller is responsible
for
       * catching it and triggering the appropriate error event.
       *
       * @param webProps the captured property values needed for the
request
       *
       * @param postData the data for the post request if it is not
coming from a
       * file, can be null
       *
       * @param postFile the path of the file containing data for the
post request
       * if it is coming from a file, can be null
       *
       * @throws IOException
       */
      private void performRequest(final CapturedProperties webProps,
byte[] postData, String postFile, String httpVerb)
                  throws IOException {

            // Open the connection.
            HttpURLConnection connection = openConnection(webProps,
httpVerb);
            if (connection != null) {
                  try {
                        if (postData != null) {
                              writeRequestData(connection, postData);
                        } else if (postFile != null) {
                              writeRequestFile(connection, postFile);
                        }
```

```
                        // Get the response.
                        final int responseCode =
connection.getResponseCode();
                        final String responseType =
getResponseType(connection);
                        processResponseCookies(connection);

                        if (!saveResponse) {
                              final String temperatura =
getResponseContent(connection);

                              // Dispatch the event.
                              activity.runOnUiThread(new Runnable() {
                                    @Override
                                    public void run() {
                                          Capturar(temperatura);
                                    }
                              });
                        }

                  } finally {
                        connection.disconnect();
                  }
            }
      }

      /**
       * Open a connection to the resource and set the HTTP action to
PUT or
       * DELETE if it is one of them. GET would be the default, and POST
is set in
       * writeRequestData or writeRequestFile
       *
       * @param webProps
       *            the properties of the connection, set as properties
in the
       *            component
       * @param httpVerb
       *            One of GET/POST/PUT/DELETE
       * @return a HttpURL Connection
       * @throws IOException
       * @throws ClassCastException
       * @throws ProtocolException
       *             thrown if the method in setRequestMethod is not
```

```
correct
       */
      private static HttpURLConnection openConnection(CapturedProperties
webProps, String httpVerb)
                  throws IOException, ClassCastException,
ProtocolException {

            HttpURLConnection connection = (HttpURLConnection)
webProps.url.openConnection();

            if (httpVerb.equals("PUT") || httpVerb.equals("DELETE")) {
                  // Set the Request Method; GET is the default, and if
it is a POST,
                  // it will be marked as such
                  // with setDoOutput in writeRequestFile or
writeRequestData
                  connection.setRequestMethod(httpVerb);
            }

            // Request Headers
            for (Map.Entry<String, List<String>> header :
webProps.requestHeaders.entrySet()) {
                  String name = header.getKey();
                  for (String value : header.getValue()) {
                        connection.addRequestProperty(name, value);
                  }
            }

            // Cookies
            if (webProps.cookies != null) {
                  for (Map.Entry<String, List<String>> cookie :
webProps.cookies.entrySet()) {
                        String name = cookie.getKey();
                        for (String value : cookie.getValue()) {
                              connection.addRequestProperty(name, value);
                        }
                  }
            }

            return connection;
      }

      private static void writeRequestData(HttpURLConnection connection,
byte[] postData) throws IOException {
            // According to the documentation at
```

```
            //
http://developer.android.com/reference/java/net/HttpURLConnection.html
            // HttpURLConnection uses the GET method by default. It will
use POST if
            // setDoOutput(true) has
            // been called.
            connection.setDoOutput(true); // This makes it something
other than a
                                                        // HTTP GET.
            // Write the data.
            connection.setFixedLengthStreamingMode(postData.length);
            BufferedOutputStream out = new
BufferedOutputStream(connection.getOutputStream());
            try {
                  out.write(postData, 0, postData.length);
                  out.flush();
            } finally {
                  out.close();
            }
      }

      private void writeRequestFile(HttpURLConnection connection, String
path) throws IOException {
            // Use MediaUtil.openMedia to open the file. This means that
path could
            // be file on the SD card,
            // an asset, a contact picture, etc.
            BufferedInputStream in = new
BufferedInputStream(MediaUtil.openMedia(form, path));
            try {
                  // Write the file's data.
                  // According to the documentation at
                  //
http://developer.android.com/reference/java/net/HttpURLConnection.html
                  // HttpURLConnection uses the GET method by default.
It will use
                  // POST if setDoOutput(true) has
                  // been called.
                  connection.setDoOutput(true); // This makes it
something other than
                                                              // a
HTTP GET.
                  connection.setChunkedStreamingMode(0);
                  BufferedOutputStream out = new
BufferedOutputStream(connection.getOutputStream());
```

```
                try {
                        while (true) {
                                int b = in.read();
                                if (b == -1) {
                                        break;
                                }
                                out.write(b);
                        }
                        out.flush();
                } finally {
                        out.close();
                }
        } finally {
                in.close();
        }
    }

    private static String getResponseType(HttpURLConnection
connection) {
        String responseType = connection.getContentType();
        return (responseType != null) ? responseType : "";
    }

    private void processResponseCookies(HttpURLConnection connection)
{
        if (allowCookies && cookieHandler != null) {
                try {
                        Map<String, List<String>> headerFields =
connection.getHeaderFields();
                        cookieHandler.put(connection.getURL().toURI(),
headerFields);
                } catch (URISyntaxException e) {
                        // Can't convert the URL to a URI; no cookies for
you.
                } catch (IOException e) {
                        // Sorry, no cookies for you.
                }
        }
    }

    private static String getResponseContent(HttpURLConnection
connection) throws IOException {
        // Use the content encoding to convert bytes to characters.
        String encoding = connection.getContentEncoding();
        if (encoding == null) {
```

```
                  encoding = "UTF-8";
            }
            InputStreamReader reader = new
InputStreamReader(getConnectionStream(connection), encoding);
            try {
                  int contentLength = connection.getContentLength();
                  StringBuilder sb = (contentLength != -1) ? new
StringBuilder(contentLength) : new StringBuilder();
                  char[] buf = new char[1024];
                  int read;
                  while ((read = reader.read(buf)) != -1) {
                        sb.append(buf, 0, read);
                  }
                  return sb.toString();
            } finally {
                  reader.close();
            }
      }

      private static String saveResponseContent(HttpURLConnection
connection, String responseFileName,
                  String responseType) throws IOException {
            File file = createFile(responseFileName, responseType);

            BufferedInputStream in = new
BufferedInputStream(getConnectionStream(connection), 0x1000);
            try {
                  BufferedOutputStream out = new
BufferedOutputStream(new FileOutputStream(file), 0x1000);
                  try {
                        // Copy the contents from the input stream to the
output stream.
                        while (true) {
                              int b = in.read();
                              if (b == -1) {
                                    break;
                              }
                              out.write(b);
                        }
                        out.flush();
                  } finally {
                        out.close();
                  }
            } finally {
                  in.close();
```

```
            }

            return file.getAbsolutePath();
      }

      private static InputStream getConnectionStream(HttpURLConnection
connection) {
            // According to the Android reference documentation for
            // HttpURLConnection: If the HTTP response
            // indicates that an error occurred, getInputStream() will
throw an
            // IOException. Use
            // getErrorStream() to read the error response.
            try {
                  return connection.getInputStream();
            } catch (IOException e1) {
                  // Use the error response.
                  return connection.getErrorStream();
            }
      }

      private static File createFile(String fileName, String
responseType) throws IOException, FileUtil.FileException {
            // If a fileName was specified, use it.
            if (!TextUtils.isEmpty(fileName)) {
                  return FileUtil.getExternalFile(fileName);
            }

            // Otherwise, try to determine an appropriate file extension
from the
            // responseType.
            // The response type could contain extra information that we
don't need.
            // For example, it might
            // be "text/html; charset=ISO-8859-1". We just want to look
at the part
            // before the semicolon.
            int indexOfSemicolon = responseType.indexOf(';');
            if (indexOfSemicolon != -1) {
                  responseType = responseType.substring(0,
indexOfSemicolon);
            }
            String extension = mimeTypeToExtension.get(responseType);
            if (extension == null) {
                  extension = "tmp";
```

```
            }
            return FileUtil.getDownloadFile(extension);
      }

      /*
       * Converts request headers (a YailList) into the structure that
can be used
       * with the Java API (a Map<String, List<String>>). If the request
headers
       * contains an invalid element, an InvalidRequestHeadersException
will be
       * thrown.
       */
      private static Map<String, List<String>>
processRequestHeaders(YailList list)
                  throws InvalidRequestHeadersException {
            Map<String, List<String>> requestHeadersMap =
Maps.newHashMap();
            for (int i = 0; i < list.size(); i++) {
                  Object item = list.getObject(i);
                  // Each item must be a two-element sublist.
                  if (item instanceof YailList) {
                        YailList sublist = (YailList) item;
                        if (sublist.size() == 2) {
                              // The first element is the request header
field name.
                              String fieldName =
sublist.getObject(0).toString();
                              // The second element contains the request
header field
                              // values.
                              Object fieldValues = sublist.getObject(1);

                              // Build an entry (key and values) for the
                              // requestHeadersMap.
                              String key = fieldName;
                              List<String> values = Lists.newArrayList();

                              // If there is just one field value, it is
specified as a
                              // single non-list item (for
                              // example, it can be a text value). If
there are multiple
                              // field values, they are
                              // specified as a list.
```

```
                                if (fieldValues instanceof YailList) {
                                      // It's a list. There are multiple
field values.
                                      YailList multipleFieldsValues =
(YailList) fieldValues;
                                      for (int j = 0; j <
multipleFieldsValues.size(); j++) {
                                            Object value =
multipleFieldsValues.getObject(j);
                                            values.add(value.toString());
                                      }
                                } else {
                                      // It's a single non-list item. There
is just one field
                                      // value.
                                      Object singleFieldValue =
fieldValues;

values.add(singleFieldValue.toString());
                                }
                                // Put the entry into the
requestHeadersMap.
                                requestHeadersMap.put(key, values);
                          } else {
                                // The sublist doesn't contain two
elements.
                                throw new
InvalidRequestHeadersException(ErrorMessages.ERROR_WEB_REQUEST_HEADER_NO
T_TWO_ELEMENTS,
                                            i + 1);
                          }
                    } else {
                          // The item isn't a sublist.
                          throw new
InvalidRequestHeadersException(ErrorMessages.ERROR_WEB_REQUEST_HEADER_NO
T_LIST, i + 1);
                    }
              }
              return requestHeadersMap;
        }

        /*
         * Captures the current property values that are needed for an
HTTP request.
         * If an error occurs while validating the Url or RequestHeaders
```

```
property
       * values, this method calls form.dispatchErrorOccurredEvent and
returns
       * null.
       *
       * @param functionName the name of the function, used when
dispatching
       * errors
       */
      private CapturedProperties capturePropertyValues(String
functionName) {
            try {
                  return new CapturedProperties(this);
            } catch (MalformedURLException e) {
                  form.dispatchErrorOccurredEvent(this, functionName,
ErrorMessages.ERROR_WEB_MALFORMED_URL, urlString);
            } catch (InvalidRequestHeadersException e) {
                  form.dispatchErrorOccurredEvent(this, functionName,
e.errorNumber, e.index);
            }
            return null;
      }
}
```

APÊNDICE C - Artigo desenvolvido

Desenvolvimento de componentes para App Inventor e servidor para dispositivos IoT

Michel Miola[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

miola.michel@gmail.com

Abstract. *In the current scenario of society, it is necessary that the teaching of computing be explored already in the beginnings of basic education, for being so importat. The introduction of computing using IoT devices makes teaching more playful and interesting for students of basic education. Therefore, the purpose of this work is to perform the academic research and development of components in the App Inventor platform, for communication with IoT devices. In App Inventor, an extension was developed that allows communication with the IoT devices through a server, which receives all the requests made by App Inventor. This research aims at the educational field, teaching the programming and its structures, using block programming with IoT devices, for students of elementary schools. The evaluation conducted with the target audience raises indications that the components and the developed server are useful for learning computing and basic IoT principles.*

Resumo. *No cenário atual da sociedade, é necessário que o ensino da computação seja explorado já nos primórdios do ensino básico, tamanha sua importância. A introdução da computação utilizando dispositivos IoT torna o ensino mais lúdico e interessante para alunos da educação básica. Portanto, este trabalho tem por finalidade efetuar a pesquisa acadêmica e desenvolvimento de componentes na plataforma App Inventor, para comunicação com dispositivos IoT. No App Inventor, foi desenvolvida uma extensão que permite a comunicação com os dispositivos IoT por meio de um servidor, que recebe todas as requisições realizadas pelo App Inventor. Tal pesquisa tem por objetivo o campo educacional, ensinando a programação e suas estruturas, utilizando a programação em bloco junto a dispositivos IoT, para alunos de escolas do ensino básico. A avaliação realizada com o público-alvo, levanta indícios de que os componentes e o servidor desenvolvido são úteis para o aprendizado de computação e de princípios básicos de IoT.*

1. INTRODUÇÃO

A iniciativa “Computação na Escola” desenvolvida no Departamento de Estatística e Informática da Universidade Federal de Santa Catarina, tem o intuito de aumentar o ensino da computação nas escolas de nível básico (fundamental e médio). A iniciativa tem por objetivo instruir melhor as crianças e jovens para a área da computação e mostrar que a computação pode ser ensinada para todas as pessoas.

A computação pode ir além do “software” ou do "hardware", podendo ser utilizada como ferramenta de apoio no ensino de outras matérias que estão indiretamente ligadas à computação, tornando-a uma área interdisciplinar que envolve biologia, física, química e matemática.

A “Computação na Escola” utiliza algumas ferramentas para facilitar assimilação, sendo a plataforma App Inventor um dos recursos utilizados. A aplicação App Inventor foi desenvolvida pelo Instituto de Tecnologia de Massachusetts (MIT). Dentre os múltiplos recursos que a plataforma App Inventor disponibiliza, inclusive o suporte ao ensino da programação e suas estruturas, a plataforma utiliza uma abordagem visual no ensino da programação para computadores, possibilitando assimilação de maneira intuitiva, mesmo para crianças. Através disso, a ferramenta permite o ensino das estruturas de programação, utilizando o paradigma de programação em bloco, que facilita a criação de aplicativos complexos.

A IoT hoje está presente no dia a dia sem que se perceba o quanto ela é importante para o funcionamento dos processos e das atividades recorrentes. Pode-se tomar como exemplo: sensor de GPS presente no celular, ou no carro para verificar a trajetória ou tráfego; sensores de trânsito, câmeras e camadas de informação mostrando o fluxo de automóveis para possibilitar análise do tr*â*nsito; o controle de aparelhos eletrodomésticos por interface de acessos chamados atuadores que, por exemplo, controlam o funcionamento de um ar condicionado.

Atualmente a ferramenta App Inventor não permite o ensino de conceitos de computação utilizando estruturas físicas junto a rede TCP/IP e interação com o mundo real, o que pode ser interessante para alunos da Educação Básica.

As seções a seguir apresentam uma breve refencial teórico, seguido de uma análise de trabalhos relacionados, a descricão do modelo proposto e uma avaliação realizada dos componente propostos.

2. FUNDAMENTAÇÃO TEÓRICA

No presente capítulo são abordados os principais conceitos teóricos necessários para a compreensão deste trabalho. Primeiramente, é abordado o ensino da programação de computadores para crianças e jovens. Em sequência, é explanado sobre a ferramenta App Inventor, utilizando o conceito de programação em blocos. E, por fim, são apresentados os conceitos de IoT e suas aplicações junto ao campo educacional para crianças e jovens.

2.1. ENSINO DE COMPUTAÇÃO PARA CRIANÇAS E JOVENS

Para Sundmaeker, Harald et al. (2010), a computação é vista como sendo um englobamento de conhecimento tanto teórico, prático e experimental. No atual cenário educacional público, é visto como sendo necessário o ensino de computação somente em cursos de graduação como Ciência da Computação, Sistema de Informação, entre outros (Sundmaeker, Harald et al. (2010), p. 2). O conhecimento da computação no século XXI é fundamental para a resolução de muitos dos problemas que são recorrentes na vida cotidiana, visto que grande parte das profissões na atualidade trabalham de maneira direta ou indireta com a área da computação. Dessa maneira, alguns autores trabalham com o conceito de "Pensamento Computacional" - uma forma de pensamento característica dos cientistas da computação, mas universalmente aplicável, que envolve um conjunto de atitudes e habilidades, tais como o uso da recursividade, abstração e decomposição na solução de problemas, tanto técnico-científicos quanto da vida cotidiana (Wing, Jeannette M (2006)).

Nesse sentido, a CSTA K-12, uma associação dos professores de computação, define um currículo de ensino de Ciência da Computação em todo o Ensino Básico. O currículo da associação CSTA K-12 tem 3 níveis que se enquadram nos três níveis do ensino básico brasileiro. Para (SEEHORN et al., 2011), o ensino infantil do módulo "K-6 Ciência da Computação e Eu" foca no ensino introdutório sobre a Ciência da Computação, integrando habilidades básicas em tecnologia com simples ideias sobre o pensamento computacional. No nível 2 "Grade 6-9 - Ciência da Computação" que é equivalente ao Ensino Fundamental, os alunos começam a usar o pensamento computacional como uma ferramenta de solução de problemas. E, por fim, no nível 3 "Grade 9-12 - Aplicando conceitos e criando soluções do mundo real" que é equivalente ao Ensino Médio, os alunos trabalham com conceitos mais avançados no campo da computação, e dessa forma, podem aplicar estes conceitos no desenvolvimento de trabalhos virtuais para a solução de problemas da vida real (SEEHORN et al., 2011, p. 8).

2.2. APP INVENTOR

App Inventor, uma ferramenta desenvolvida pela MIT (Inventor, APP (2012)), que utiliza o conceito de programação visual para uma maior assimilação sobre a lógica no campo da computação, abstraindo toda a parte de sintaxe da linguagem de programação.

A plataforma App Inventor é um ambiente com iniciativa de código aberto, possibilitando o desenvolvimento de aplicativos para dispositivos móveis. Ela tem o seu ambiente de desenvolvimento como grande diferencial, pois utiliza a estrutura de blocos para o ensino de programação. Outro diferencial é sua possibilidade de integração com plataformas externas como redes sociais, serviços disponibilizados pela web, entre outras integrações possíveis de serem realizadas. As aplicações geradas sobre a plataforma são compiladas para rodarem sobre um ambiente Android, facilitando o desenvolvimento de aplicativos, uma vez que a grande maioria dos smartphones e tablets utilizam o Android (LECHETA, 2016) como Sistema Operacional. A plataforma App Inventor é totalmente web e pode ser acessada online - ambiente disponibilizado

por sua organização - ou caso tenha que ser acessado de maneira offline pode ser instalado localmente.

2.2.1. LINGUAGEM VISUAL DE "BLOCOS"

O conceito de linguagem visual que utiliza a aplicação de blocos para a montagem das estruturas de programação facilita a assimilação para essência do ensino da linguagem de programação (Sudol, Leigh Ann (2009)). Através disso, incentiva os jovens a ingressar no campo da programação para computadores. Para esses, fica mais fácil a assimilação do conteúdo de programação utilizando uma ferramenta de desenvolvimento visual, já que no começo eles não apresentam uma familiaridade com esse conteúdo. Dessa maneira, os alunos não precisam ter o conhecimento de determinada linguagem de programação para a construção de aplicações.

A programação visual permite modelar todos os conceitos de programação: estrutura de condição, laços de repetição, variáveis entre outras estruturas. Esses itens são apresentadas separadamente em blocos para serem arrastados e encaixados, montando funções que resolvem determinado problema. Todos os blocos já estão criados e visíveis aos usuários para serem utilizados na construção de algoritmos. Dessa maneira, a criação dos algoritmos é muito intuitiva para o usuário, sendo mostrados *feedbacks* quando um bloco encaixado não é compatível ao bloco que o antecede.

2.3. ENSINO DE IOT PARA CRIANÇAS/JOVENS

O mundo está extremamente voltado ao campo tecnológico. Em 2011 a população chegou a 7 bilhões de pessoas no planeta e o número de aparelhos conectados chegou a 13 bilhões. Em 2015, o número de aparelhos superou 3 vezes o número da população mundial e a tendência é de aumentar ainda mais ao longo dos próximos anos. A internet nos tempos atuais é muito difundida e necessária para toda a sociedade. Dessa forma, ela tornou-se um instrumento importante para o dia a dia, e nos dias atuais é um recurso essencial para tarefas diárias.

O ensino de computação para crianças e jovens é uma tendência pela importância que a computação tem no cotidiano, preparando as crianças a lidarem com a computação de uma maneira natural. Por algumas áreas da computação serem muito abstratas, o ensino para crianças fica um pouco mais difícil de ser assimilado. Sendo assim, uma abordagem de ensino voltada à IoT, despertaria um interesse maior por parte das crianças, que não ficariam presas a interações somente via periféricos normais do computador (mouse, teclado e tela). Para tal, poderiam ser mostradas outras formas de computação que interagem com o nosso meio, como sensores e atuadores.

Outro ponto que deve ser levado em consideração é que a abordagem de uma forma direta com as placas de microcontroladores - como Arduino, Raspberry Pi e GogoBoard - vai exigir um conhecimento na parte de eletrônica e computacional por parte dos alunos, algo que não é de fácil assimilação para grande maioria das crianças e adolescentes. Uma barreira que também seria gerada ao trabalhar diretamente com microcomputadores é a linguagem de programação que, na maioria das vezes, é de

baixo nível. Dessa maneira, o uso de linguagem com uma abordagem visual facilitaria a compreensão por parte de uma criança ou adolescente, que não precisa ter conhecimento da sintaxe de uma linguagem de programação para a implementação de aplicações com interação com microprocessadores. A programação utiliza abordagem visual, tornando o ensino para as crianças ou adolescentes mais atrativo, por conseguir trabalhar com imagens que são melhor assimiladas. De acordo com Costa, Cristina (2005), as imagens apresentam um caráter intuitivo muito maior do que a linguagem verbal/escrita, pois, elas são mais universais do que as linguagens verbais e sonoras. Assim, a utilização da imagem pode ser útil como um recurso didático, pois esse caráter intuitivo da linguagem visual pode facilitar a aprendizagem dos estudantes.

Para Ruzzenente, Marco et al. (2012) , o ensino de computação na área da Internet das Coisas, torna-se interdisciplinar, por trabalhar com alguns conceitos de disciplinas tradicionais previstas em matemática, física, programação de computadores e eletrônica, mas também filosofia, desenvolvimento da linguagem, história, e assim, a alfabetização de currículo do nível primário, secundário, de graduação e de pós-graduação. Isso pode, portanto, incentivar a continuar os estudos tanto no campo computacional como em áreas subjacentes a computação, tendo em vista que se almeja que grande parte dos alunos do Ensino Básico tenha a oportunidade para explorar o pensamento computacional e aflorar o lado criativo com o auxílio da computação.

3. ESTADO DA ARTE

Neste capitulo é levantado o estado da arte atual de pesquisas relacionadas a área de ensino de computação física/robótica/IoT com App Inventor. A análise do estado da arte foi realizada seguindo o método de revisão sistemática definida por Kitchenham, Barbara (2004) .

3.1. DEFINIÇÃO DO PROTOCOLO DE REVISÃO

Para identificar o estado da arte, foi definida uma revisão sistemática da literatura, com objetivo de analisar e sintetizar as literaturas existentes sobre a pergunta "O que está sendo feito atualmente na área de ensino de computação física/robótica/IoT com App Inventor? ".

Entre as alternativas de ferramentas para realizar a pesquisa sobre a literatura existente, foi escolhido o Google Scholar, que é abrangente para busca em fontes bibliográficas digitais. Porém, em virtude do App Inventor conter um repositório de bibliografias, foram realizadas pesquisas também em seu repositório. Dentre as ferramentas para pesquisas bibliográficas, o Google Scholar possui o recurso de pesquisas dentro do idioma da língua portuguesa do Brasil, além de outros idiomas, o que facilitou a pesquisa dos termos nos idiomas propostos pelo trabalho. Para realizar a pesquisa foram utilizados termos na língua portuguesa do Brasil e no inglês dentro do período de 2014 a 2018.

3.2. EXECUÇÃO DA BUSCA

De modo a extrair um melhor aproveitamento dos resultados obtidos na etapa de busca da revisão sistemática da literatura, alguns critérios foram previamente definidos como sendo indicadores de inclusão e exclusão de trabalhos. Após a aplicação dos critérios de inclusão e exclusão, a execução da busca ocorreu em maio de 2018.

No total foram realizadas 8 iterações para concluir as buscas, que vão ser apresentadas na sequência.

3.3. DISCUSSÕES

A busca sobre as bibliografias, mostrou uma ampla iniciativa de trabalhos de pesquisa com o intuito de melhorar o ensino de computação com o auxílio da robótica para crianças e adolescentes.

Um fato discutido por alguns trabalhos levanta a questão dos custos para implantação das iniciativas de ensino de robótica nas escolas públicas brasileiras, em virtude de alguns componentes terem custo elevado. Dessa maneira, alguns trabalhos propõem soluções alternativas, de baixo custo, para o ensino de robótica. Por exemplo, utilizar componentes de hardware livre como minicomputadores Arduino e Raspberry Pi, que tem um custo inferior em relação a outros componentes como Lego Mindstorms.

Visto que, uma das grandes dificuldades apresentadas nas pesquisas bibliográficas é o custo elevado dos componentes de hardware para a implantação de projetos junto as escolas brasileiras, o presente trabalho implementado tem uma característica importante a ser considerada, pois a construção da placa teve um custo baixo em comparação com outros hardwares que são utilizados em trabalhos no ambiente escolar.

3.4. PROPOSTA DE SOLUÇÃO

Os trabalhos estudados na etapa anterior, tornaram-se então a base para o estado da arte na implementação de novos componentes junto a ferramenta App Inventor que se comunicam com dispositivos IoT.

Esperava-se que tal esforço pudesse contribuir positivamente com a o ensino de computação nas escolas de nível básico, em especial com os alunos. O objetivo principal deste trabalho é realizar o desenvolvimento de novos componentes junto à ferramenta App Inventor, para comunicação via rede TCP/IP com dispositivos IoT. E, através disso, permitir o desenvolvimento de aplicativos para celular junto à ferramenta App Inventor, com interação com dispositivos IoT. Com isso, espera-se aumentar o incentivo aos alunos da Educação Básica, no contexto da iniciativa "Computação na Escola", que utilizam a ferramenta App Inventor para o desenvolvimento de aplicativos.

4. MODELAGEM E DESENVOLVIMENTO

Este capítulo apresenta a modelagem dos componentes dos componentes que foram implementados junto a ferramenta App inventor, e do desenvolvimento do servidor junto ao dispositivo IoT.

4.1. MODELAGEM

Para o processo de desenvolvimento são apresentados os diagramas utilizando UML - uma linguagem padrão para modelagem de projetos de software. Para a proposta deste trabalho, que consiste da criação de componentes do App Inventor junto a dispositivos IoT, foram elaborados diagramas de classe tanto para descrever os componentes criados para o App Inventor, como para o servidor de aplicação da placa Raspberry Pi. Dando prosseguimento a modelagem, é apresentado um diagrama de sequência que vai demonstrar a comunicação de um componente App Inventor com a placa Raspberry Pi.

4.1.1. DIAGRAMA DE CLASSE

Para os componentes criados na ferramenta App Inventor foi utilizado como base o componente Web, que é um componente que realiza requisições utilizando o protocolo HTTP. Dessa maneira, as classes que representam os componentes criados são ao total quatro, sendo: Interruptor, Temperatura, Umidade e Música.

4.1.1.1. COMPONENTE INTERRUPTOR

Para o componente Interruptor, foram criados quatro métodos novos, além dos já existentes na classe Web. Os métodos criados foram: EnderecoServidor, Gpio, Ligar e Desligar. Todos os métodos da classe Interruptor estão no diagrama de classe.

Figura 5 – Diagrama de classe componente Interruptor
elaborado pelo autor

4.1.1.2. COMPONENTE TEMPERATURA E UMIDADE

Os componentes Temperatura e Umidade foram criados contendo a mesma estrutura de métodos, porém, realizam requisições para serviços REST diferentes. Dessa forma, quatro métodos novos foram desenvolvidos nas classes Temperatura e Umidade, além dos já existentes na classe Web, sendo os novos métodos: EnderecoServidor, Gpio, Obter e Capturar.

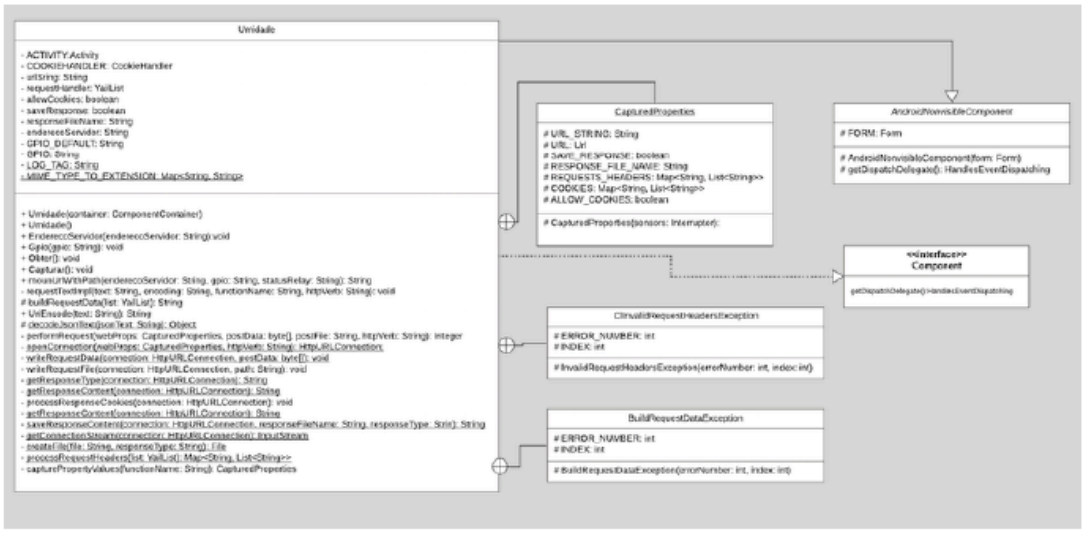

Figura 7 – Diagrama de classe componente Umidade
elaborado pelo autor

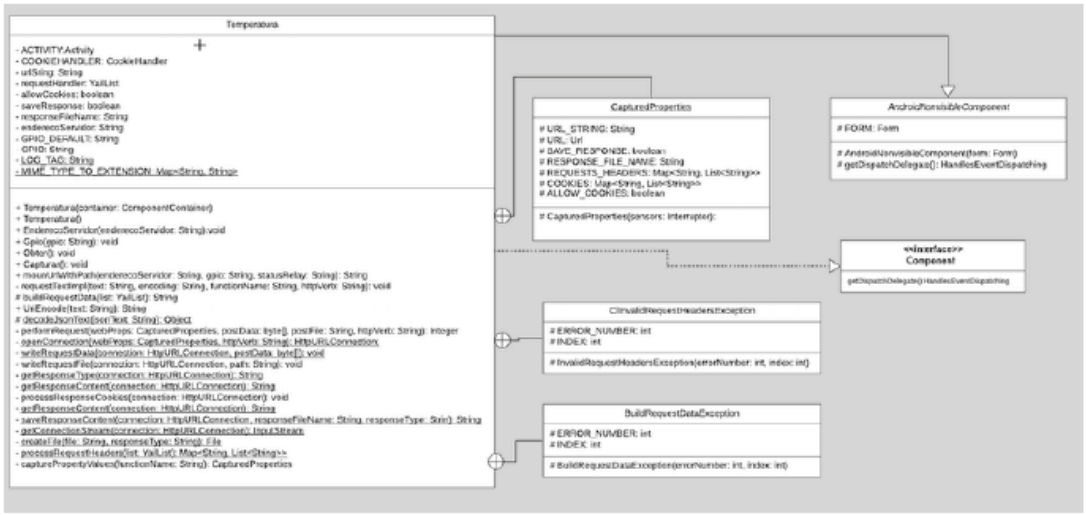

Figura 6 – Diagrama de classe componente Temperatura
elaborado pelo autor

4.1.1.3. COMPONENTE MÚSICA

Para o componente Musica foram criados seis métodos novos, além dos já existentes na classe Web. Os métodos criados foram: EnderecoServidor, TocarAudio1, TocarAudio2, TocarAudio3, TocarAudio4 e ParararAudio.

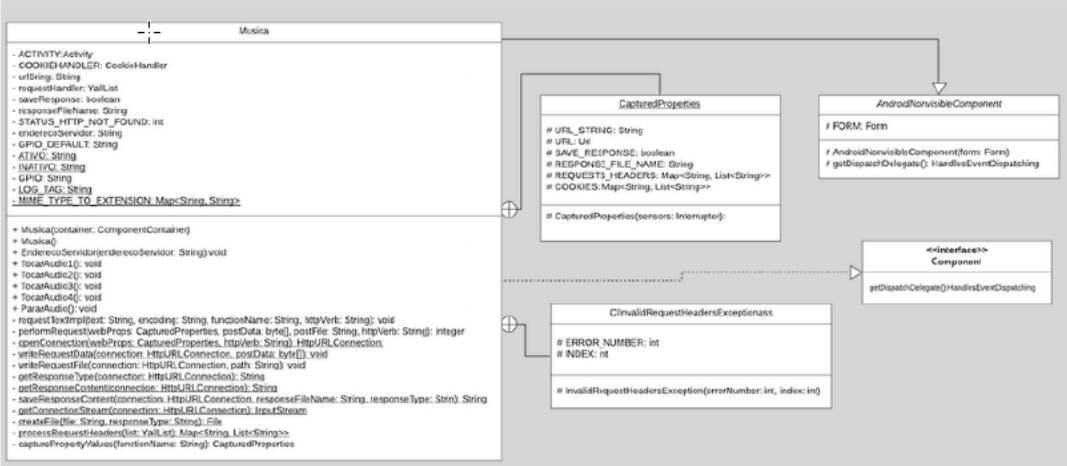

Figura 8 – Diagrama de classe componente Musica
elaborado pelo autor

4.1.2. DIAGRAMA DE SEQUÊNCIA

Por fim, na ilustração da comunicação via protocolo HTTP do componente junto ao servidor de aplicação, foi criado um diagrama de sequência para apenas um componente proposto pelo trabalho - que vai simular a comunicação dos outros componentes implementados pelo presente trabalho. Apresenta o diagrama de sequência das trocas de mensagens entre o componente do App Inventor e o servidor rodando na placa Raspberry Pi.

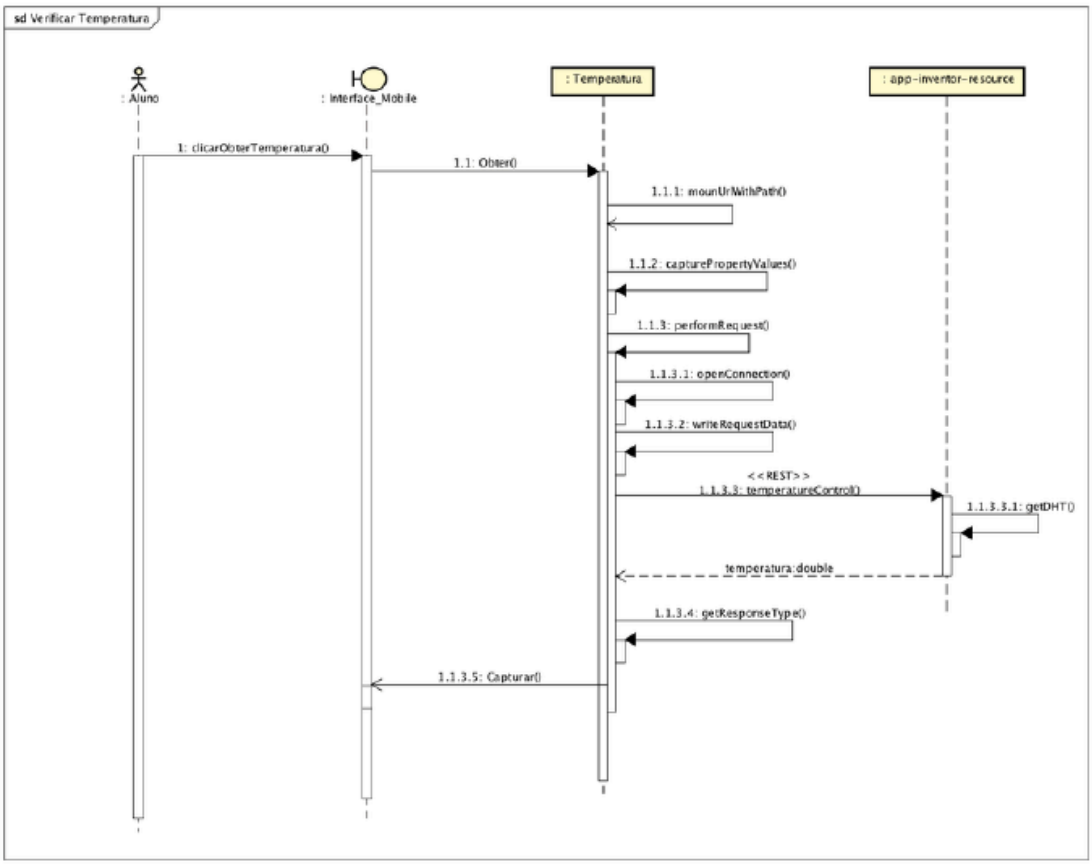

Figura 9 – Diagrama de sequência componente Temperatura
elaborado pelo autor

4.2. DESENVOLVIMENTO

Nesse trabalho foi criada uma solução de implementação para o ensino de computação utilizando dispositivo IoT com componentes extensíveis da ferramenta App Inventor, para o ensino mais lúdico e interessante no contexto de uma criança ou jovem que não tem experiência com linguagem de programação.

O trabalho foi desenvolvido utilizando a ferramenta App inventor para a criação de novos componentes extensíveis, os quais interagem com o servidor que foi implementado no minicomputador Raspberry Pi.

4.2.1. COMPONENTE INTERRUPTOR

O componente criado é simples, contendo apenas dois blocos de funções dentro de sua estrutura: um para "Ligar" e outro para "Desligar" a lâmpada. Porém, dentro do componente é necessário realizar a configuração de alguns parâmetros, que são: endereço do servidor e gpio.

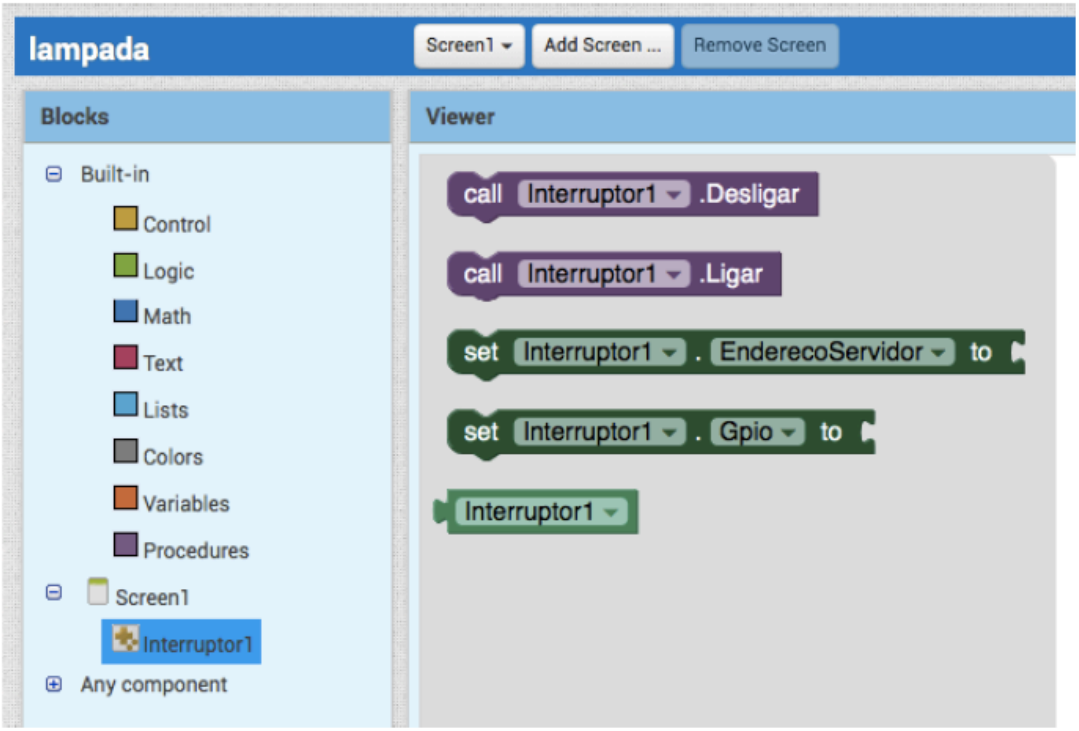

Figura 20 – Componente Interruptor
elaborado pelo autor

4.2.2. COMPONENTE TEMEPRATURA

Foram criados dois blocos de função no componente, o primeiro bloco que é chamado Obter - realiza a chamada assíncrona - e o segundo bloco chamado de Capturar - recebe as informações que foram solicitadas pelo bloco Obter. Como já mostrado no componente lâmpada, é necessário realizar a configuração de dois parâmetros para o componente, que são: endereço do servidor e a gpio.

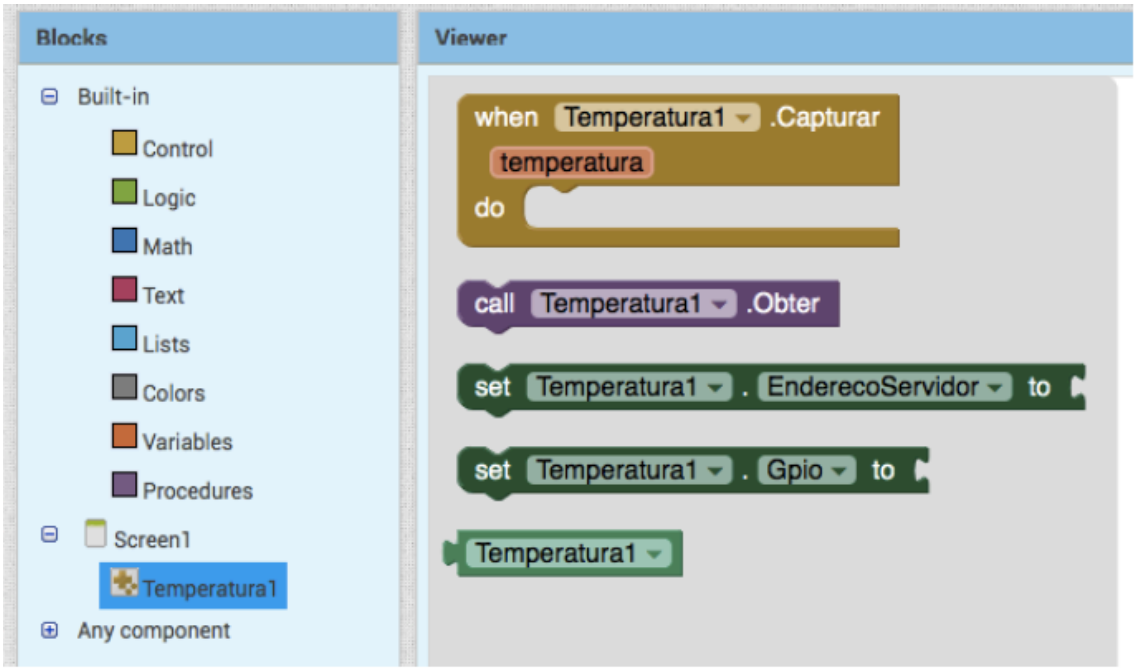

Figura 23 – Componente Temperatura
elaborado pelo autor

4.2.3. COMPONENTE UMIDADE

O desenvolvimento não envolveu muita complexidade, pois a estrutura do código foi semelhante a implementação do componente temperatura. Desse modo, o componente quando executado realiza uma chamada HTTP para o serviço de umidade na placa Raspberry Pi, porém, a resposta é tratada de maneira assíncrona pela arquitetura da plataforma App Inventor. Sendo assim, para o tratamento do assíncrono na requisição de umidade, foi necessária a criação de dois blocos dentro do componente. O componente umidade também necessita da configuração de dois parâmetros, que são: endereço do servidor e de gpio.

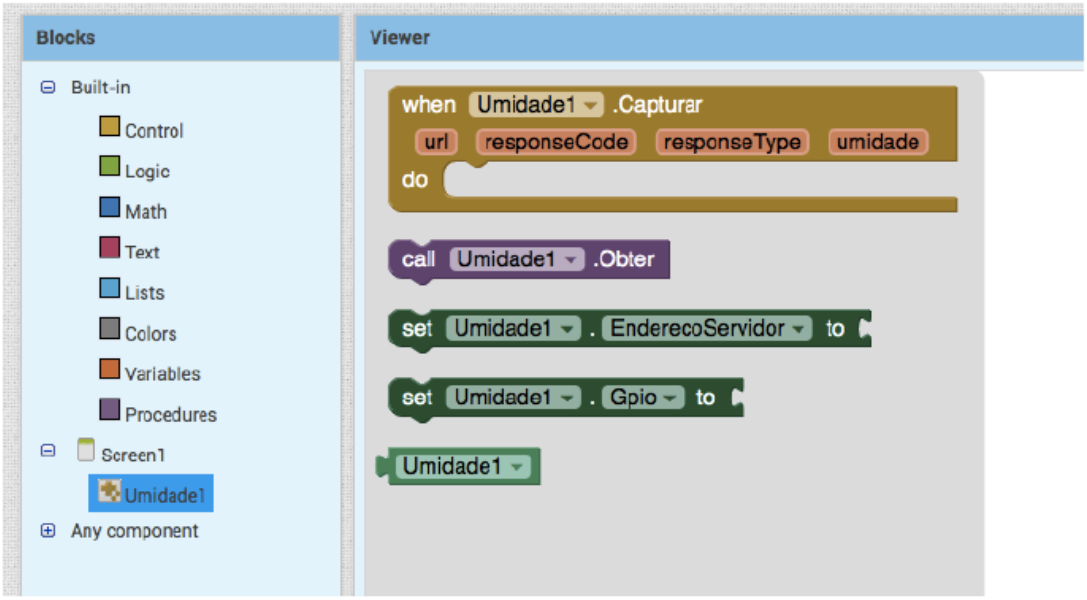

Figura 26 – Componente Umidade
elaborado pelo autor

4.2.4. COMPONENTE MÚSICA

Foram criado os blocos de função TocarAudio1 a TocarAudio4 realizam chamadas de serviço para a placa Rasbpberry Pi, onde serão executadas as músicas que estão pré estabelecidas no diretório da placa. Porém, além de realizar a execução das músicas, é possível também interromper a música que está sendo executada. O bloco ParararAudio segue o mesmo procedimento, realizando uma chamada de serviço para a placa, onde é realizada a interrupção do processo que está executando a música em questão. Para executar o componente música é necessário configurar o parâmetro endereço do servidor.

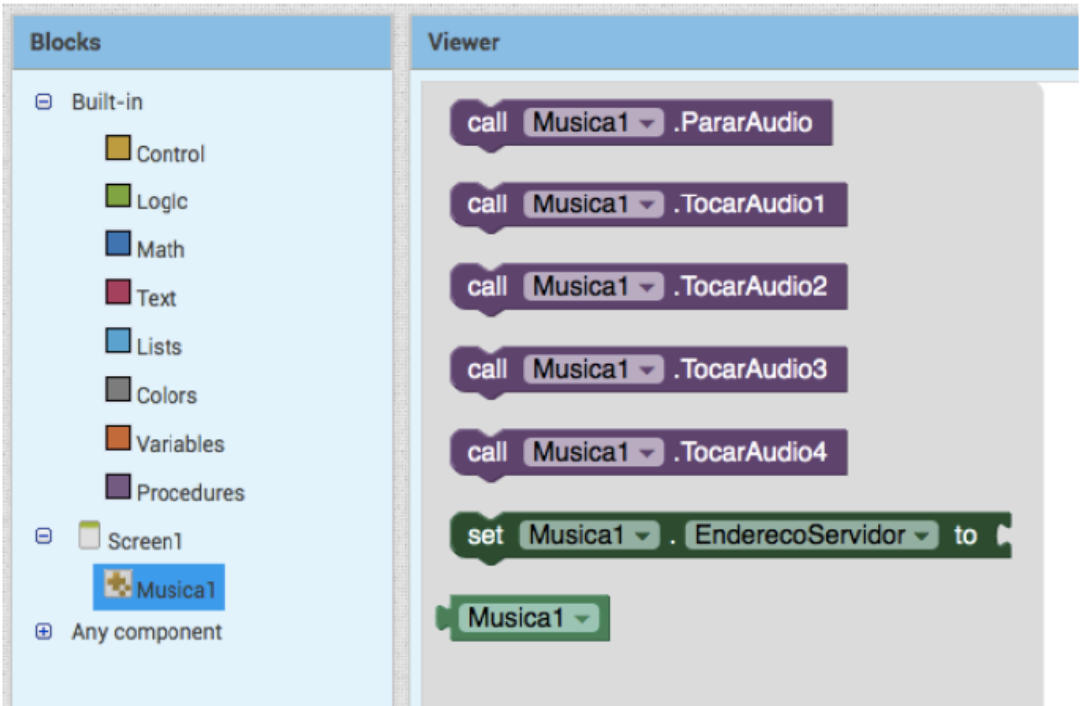

Figura 29 – Componente Música
elaborado pelo autor

5. AVALIAÇÃO

Para avaliar a qualidade dos componentes que foram implementados junto à ferramenta App Inventor e analisar a qualidade, foi realizado um estudo de caso envolvendo uma turma de alunos da Educação Básica. A qualidade dos componentes e do servidor desenvolvidos foi avaliada em termos de utilidade, adequação funcional e usabilidade do ponto de vista de estudantes da Educação Básica.

5.1. DEFINIÇÃO DA AVALIAÇÃO

A avaliação das extensões que foram desenvolvidas para a ferramenta App Inventor foi baseada nos conceitos de qualidade definidos na norma ISO/IEC-25010 (International Organization for Standardization (2016)), sendo que os fatores de qualidade a serem avaliados são decompostos a seguir.

Característica	Subcaracterística	Avaliação do usuário
		Item do questionário
		Questionário do aluno
Utilidade		Você acha a ferramenta componentes IoT do App Inventor útil na aprendizagem de programação.
Adequação funcional	Corretude funcional	Você observou algum erro (bug) em relação a funcionalidade dos componentes IoT do App Inventor.
Usabilidade	Satisfação	Você acha que a ferramenta componentes IoT auxilia no entendimento do conceito de IoT.
		Você acha que o ensino de programação utilizando sensores/placas torna o aprendizado mais interessante.
		Você achou fácil usar os componentes IoT da ferramenta App Inventor.
		Você observou algum erro (de ortografia/gramática) na interface dos componentes IoT do App Inventor.
		Você achou fácil usar a ferramenta na utilização do componente IoT.

Tabela 14 – Fatores de qualidade a serem avaliados

5.2. EXECUÇÃO DA AVALIAÇÃO

As avaliações foram realizadas durante o primeiro semestre de 2019. Dos nove participantes da avaliação de utilização dos componentes que foram implementados junto à ferramenta App Inventor, oito tinham experiência com estruturas de programação. Porém, não tinham familiaridade com a ferramenta App Inventor, mas já tinham realizado programação visual utilizando outras ferramentas (Codeblocks ou Scratch). Para avaliação, participaram um total de nove alunos do ensino médio técnico do IFSC (Instituto Federal de Educação), representando usuários da ferramenta App Inventor que utilizariam os componentes.

5.3. ANÁLISE DOS DADOS

Nesta seção é apresentada uma análise sobre os dados que foram coletados junto aos participantes da avaliação.

5.3.1. ANÁLISE DOS DADOS DAS AVALIAÇÕES COM USUÁRIOS

Os dados analisados têm o intuito de verificar o objetivo que foi proposto pela implementação dos componentes.

- **Os componentes IoT são úteis?**

Todos os usuários avaliaram a utilidade dos componentes que foram implementados junto à ferramenta App Inventor de maneira positiva. Assim, é possível inferir que os componentes implementados conseguiram exemplificar o conceito de "Internet das Coisas", demostrando a interconectividade dos dispositivos que são comuns para o nosso cotidiano. Dessa forma, consideraram que a utilização dos componentes IoTs é mais lúdica e interessante para o aprendizado de programação.

- **Os componentes IoTs são adequados funcionalmente?**

Em relação a adequação funcional, os componentes desenvolvidos receberam uma avaliação positiva dos alunos. Porém, foi levantado um ponto de melhoria no quesito de desempenho da aplicação, visto que quando muitos usuários estão utilizando o servidor da placa Raspberry Pi a(s) resposta(s) de alguns sensores acabam demorando ou não respondendo.

- **Os componentes IoTs tem boa usabilidade?**

Em relação a usabilidade, os componentes implementados receberam uma avaliação positiva dos participantes. Dessa forma, foi atingido o objetivo de abstrair toda a complexidade dos componentes, tornado sua manipulação simples e clara para os usuários.

Durante a avaliação os usuários também responderam o questionário SUS. O questionário SUS (System Usability Scale) é um instrumento claro e rápido de coleta de informações que é utilizado ao final de um teste de interação. Desta forma, foi levantado uma média da escala SUS de todos os participantes da oficina, sendo obtida a média 80,27 numa escala de 0 a 100. Desse modo, foi atingido um resultado satisfatório, levando em consideração que média abaixo de 50 é um sinal de que os investimentos em *design* e usabilidade precisam ser priorizados dentro de seu plano de negócios.

5.4. DISCUSSÃO

Os componentes que foram implementados junto à ferramenta App Inventor foram vistos como úteis no ensino dos conceitos de IoT. Como comentado pelos participantes, os componentes tornam a experiência mais lúdica e agradável em virtude de conseguirem ver uma saída mais real com o uso dos sensores. Desse modo, os alunos com pouca experiencia com algumas linguagens de programação não precisam se preocupar com problemas que são comuns inicialmente (como é o caso da sintaxe) na utilização dos componentes IoT, sendo que os blocos já são predefinidos. Os *feebacks* coletados no resultado das avaliações foram positivos, levantando indícios de que os componentes criados são intuitivos, práticos e eficazes para o que eles se propõem.

6. CONCLUSÃO

A avaliação dos componentes foi realizada junto a usuários reais, através de uma oficina aplicada a alunos do IFSC. Dessa forma, obteve-se dados sobre a eficácia no desenvolvimento das extensões bem como a qualidade percebida pelo ponto de vista dos participantes da oficina, analisando a utilidade, adequação funcional e usabilidade. Os resultados coletados das avaliações realizadas apresentou *feedback* positivo, sob o ponto de vista da clareza dos componentes e sua facilidade na utilização. Porém, a adequação funcional apresentou um ponto de melhoria no quesito desempenho do servidor da placa Raspberry Pi, que pode ser explorado em trabalhos futuros.

Através das pesquisas bibliográficas, foi levantada uma grande dificuldade no ponto de vista da implementação de projetos de ensino de computação com o auxílio de IoT, que é o quesito de custo, considerando que hardwares como LEGO Mindstormskit tem um alto valor, dificultando a implementação de projetos junto às escolas. Sendo assim, a proposta do trabalho apresenta um hardware mais acessível no quesito custo, levando em consideração que o Raspberry Pi e os componentes utilizados na montagem da placa não apresentaram um custo elevado para sua implementação, facilitando assim a implementação de projetos junto às escolas.

Espera-se com a implementação do presente trabalho que o ensino de computação tenha desempenho melhor no quesito aprendizado, tornando mais fácil a compreensão de algumas terminologias e tornando a área mais atrativa para os alunos da Educação Básica, para assim conseguir mais estudantes para faculdades no campo da computação.

Como trabalhos futuros, espera-se uma melhoria na parte do servidor de aplicação - quanto ao seu desempenho - , visto que, quando muitos usuários estão consumindo os serviços implementados, o servidor sofre uma sobrecarga, apresentando demora no retorno das requisições.

Ainda como ponto de melhoria, para aprimoramento das extensões, pode ser implementado o processo de autenticação das requisições HTTP, o que evitaria que outro usuário alterasse o estado do sensor da placa.

REFÊNCIAS

SUNDMAEKER, H. et al. Vision and challenges for realising the Internet of Things. Cluster of European Research Projects on the Internet of Things, European Commision, v. 3, n. 3, p. 34 – 36, 2010.

WING, J. M. Computational thinking. Communications of the ACM, v. 49, n. 3, p. 33 – 35, 2006.

SEEHORN, D. et al. CSTA K–12 Computer Science Standards: Revised 2011. ACM, 2011.

INVENTOR, A. What is MIT App Inventor,[online] Available from: http://www. appinventor. mit. edu. Accessed 23April, 2012.

LECHETA, R. R. Android Essencial. São Paulo, Editora NovaTec, 2016.

SUDOL, L. A. Visual Programming Pedagogies and Integrating Current Visual Programming Language Features. 2009. Tese (Doutorado) — Robotics Institute.

COSTA, C. Educação, imagem e mídias. [S.l.]: Cortez, 2005.

RUZZENENTE, M. et al. A review of robotics kits for tertiary education. In: CITESEER,
2012. Proceedings of International Workshop Teaching Robotics Teaching with Robotics: Integrating robotics in school curriculum. [S.l.], 2012. p. 153 – 162.

STANDARDIZATION, I. O. for. Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): Measurement of System and Software Product Quality. [S.l.]: ISO, 2016.