Universidade de Santo Amaro

Bacharelado em Sistemas de Informação

Edileuza Batista de Oliveira

Jefferson Siqueira de Paulo

Priscila da Costa Souza

Tácio Veiga Ferreira

Arduino: Automação Residencial

São Paulo

2013

Edileuza Batista de Oliveira
Jefferson Siqueira de Paulo
Priscila da Costa Souza
Tácio Veiga Ferreira

Arduino: Automação Residencial

Trabalho de Conclusão de Curso apresentado ao Curso de Sistema de Informação, da Universidade de Santo Amaro – UNISA como requisito parcial à obtenção do grau de bacharel em Sistema de Informações.

Orientador: Prof. Eugenio Nassu
Co-orientador: Prof. Marcos Antonio

Edileuza Batista de Oliveira
Jefferson Siqueira de Paulo
Priscila da Costa Souza
Tácio Veiga Ferreira

Arduino: Automação Residencial

Trabalho de Conclusão de Curso apresentado ao Curso de Sistema de Informação, da Universidade de Santo Amaro – UNISA como requisito parcial à obtenção do grau de bacharel em Sistema de Informações.

Data de Defesa: 18 de Novembro de 2013.
Resultado:_____ .

BANCA EXAMINADORA

Eugenio Akihiro Nassu
da Universidade de Santo Amaro – UNISA
Nota Atribuída:

Itsche Baran
da Universidade de Santo Amaro – UNISA
Nota Atribuída:

Marco Antonio Gomes
da Universidade de Santo Amaro – UNISA
Nota Atribuída:

AGRADECIMENTOS

Primeiramente agradecemos ao nosso Deus, por tudo que somos e tudo que conquistamos.

A nossa família, por ter nos dado todo apoio, e por ter suportado durante todos esses anos nossas angustias, tristezas, incertezas, dúvidas, ausências e também as alegrias, amizades, confraternizações, comemoração e a vitória.

Aos amigos e colegas de sala, pela intensa convivência do dia a dia, colaboração com o material que por diversas vezes foram compartilhado quando não podemos estar presente por força maior.

Ao nosso orientador Eugenio Nassu pela paciência, disposição e dedicação e por acreditar e confiar na nossa capacidade.

Ao nosso co-orientador Marco Antonio, que vamos levar grandes aprendizados e ensinamentos.

E a todos os professores que colaboram para o nosso desenvolvimento intelectual, profissional, moral e ético.

Dedicamos este trabalho de conclusão de curso ao nosso Deus sem rosto e sem imagem alguma, que pela nossa fé cremos Nele, por ter nos dado forças, sabedoria, entendimento e conhecimento para realizar este trabalho, nos concedendo a vitória de poder chegar onde estamos hoje.

Toda disciplina, com efeito, no momento não parece ser motivo de alegria, mas de tristeza; ao depois, entretanto, produz fruto pacífico aos que têm sido por ela exercitados, fruto de justiça.

Hebreus 12:11.

RESUMO

Este trabalho foi desenvolvido com a finalidade de criar uma conexão entre Android e Arduino através do bluetooth, a fim de demostrar às diversas formas de uso de aplicação, dentro do ambiente residencial, para sua automação. Nele o leitor, poderá encontrar com detalhes a forma de elaboração e toda fonte de pesquisa, com o intuito de reprodução por qualquer ser humano. Ao final deste trabalho um protótipo em perfeito funcionamento, será apresentado, mostrando ser possível sua reprodução.

Palavras chaves: Arduino, automação, residencial, conexão e bluetooth

ABSTRACT

This project was developed with the intent to create a connection between Android and Arduino using Bluetooth to demonstrate the variety of applications in a residential environment, for its automation. Here the reader can find details about how to make it and all source of the research, in order to reproduce by any person. At the end of this work a prototype running perfectly, will be presented, showing that it is possible to reproduce it.

Keywords: Arduino, automation, residential, connection and bluetooth

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A tecnologia vem crescendo de maneira rápida. A cada dia nos deparamos com novidades tanto na parte de hardware quanto de software.

Em algum momento vimos reportagens ou ouvimos falar sobre a “casa do futuro”, com lâmpadas que são acionadas pela voz ou um controle universal que é compatível com todos os aparelhos eletrônicos da residência.

Para muitos, isso parece um sonho distante de ser alcançado, podemos pensar “- isso é apenas para os ricos!”, mas essa realidade pode estar muito mais perto que as pessoas imaginam.

Visando essas novas tendências, vimos por meio deste trabalho a oportunidade de apresentar uma “placa eletrônica” que aliada aos programas de desenvolvimento já existentes podem nos proporcionar novos conhecimentos e instigar a curiosidade para desenvolvermos nossos programas e/ou aplicativos que serão usados no nosso dia a dia, de maneira que sua aplicação não seja tão difícil de se realizar.

Trata-se de uma placa eletrônica *open source* conhecida como Arduino (e seus derivantes) que em conjunto com outros *hardwares* e a aplicação de *softwares* compatíveis com seu desenvolvimento sejam capazes de interagir com um dispositivo que pode se comunicar com os aparelhos e ambientes de uma residência, através de conexão bluetooth, rede ou até mesmo por comando de voz.

Temos diversos controles-remoto em nossa casa, para aparelhos de som, televisores, videogames, portão da garagem e etc. Mas se tudo isso estivesse em um único dispositivo eletrônico que está sempre conosco seria muito mais fácil. Podemos transformar o nosso smartphone ou tablet nesse controle universal.

O Arduino tem muitas funcionalidades no meio eletrônico que são exploradas diariamente, mas o nosso objetivo é apenas demonstrar a sua utilização em algo comum no nosso dia, como por exemplo, acender ou apagar uma lâmpada. Vamos utilizar um dispositivo móvel com o sistema operacional Android, que irá agir como um controle que irá acender/apagar uma lâmpada.

Esta funcionalidade que iremos desenvolver trará uma autonomia para pessoas de todos os tipos de classe social, com um toque de modernidade e tecnologia acessível a todos.

Outra particularidade do uso do Arduino é a segurança promovida ao usuário final, sem grandes investimentos e diversidade de operação, além de ser um hardware aberto, o que faz seu uso extremamente interessante e oportunista.

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

Temos como objetivo, criar uma conexão com o sistema Android e o Arduino, através de uma implementação de um protótipo onde utilizaremos uma aplicação de software e conexão de rede ou bluetooth, para que possamos automatizar tarefas rotineiras de suas residências.

1.1.2 Objetivos Específicos

Realizar pesquisas relacionadas a viabilidade x comercialização do produto final.

Desenvolver um protótipo de software para comunicação com Arduino x Android.

Desenvolver um protótipo de hardware para comunicação de Arduino x a rede elétrica (circuito integrado).

Desenvolver um software para execução dos comandos de controles internos do Arduino.

Analisar um plano de vantagens e desvantagens sobre o uso da aplicação.

Além dos itens acima, também serão utilizados os equipamentos logo abaixo:

- Arduino Uno REV 3
- Módulo Bluetooth Shield
- Lâmpada de 127v
- Leds

- Celular com sistema operacional Android com conexão bluetooth (Modelo: Motorola Defy – MB525 e Samsung Galaxy SII GT-I9100)

Os programas utilizados:

- App Inventor
- Arduino IDE

1 REVISÃO DA LITERATURA

2.1 Automações Residenciais

Ao se tratar de automação residencial, logo vem à mente o sinônimo de tecnologia, casa inteligente, inovação e altos custos, no desenvolver deste capítulo, serão abordados elementos relevantes que contribuíram para progressão e o sucesso da automação residencial, através de métodos e principais metodologias e sistemas utilizados.

A Automação Residencial é um recurso que abrange cada vez mais as expectativas do ser humano, hoje podemos abrir o jornal, uma enquete de um site qualquer ou até mesmo uma conversa com amigos, o assunto é abordado com êxtase. E é através do “poderoso” celular smartphone, que esta conquista esta cada vez mais real na vida dos seres humanos. Com plataformas de aplicação de desenvolvimento cada vez mais acessível e com interfaces amigáveis ao usuário, é possível trazer esta realidade, para dentro de nossas casas. É o sucesso e o avanço da automação residencial através de pequenos dispositivos eletrônicos, acessíveis hoje a uma grande parte da população. Podemos então utilizar estes pequenos dispositivos (celular smartphone), para realizar grandes feitos em nossa casa.

2.1.1 História

A Automação Residencial originou-se através da automação industrial, aproximadamente na década de 60, com os controles de CLPs (Controladores Lógicos Programáveis), e graças à microeletrônica, as empresas puderam mudar

seu foco, voltando-se para a automação residencial.

Na década de 70 teve seu grande marco, foi no EUA onde foram lançados os primeiros módulos inteligentes chamados X-10. O protocolo X-10 utilizava uma tecnologia PLC (*Power Line Carrier*), trata-se de uma comunicação através da rede elétrica com dispositivo de automação.

Mais tarde na década de 80, pode se pensar na centralização da automação, com a chegada dos computadores, a desvantagens era na falha total e no comprometimento do funcionamento de todo sistema. E foi pensando nisto, que entrou os microprocessadores e micro controladores.

O mundo não parou por ai, enquanto isso, ao mesmo tempo, novas tecnologias iam se desenvolvendo paralelamente tais como, infravermelhos, controles remotos, radiofrequência entre outras.

A internet teve um papel todo especial, tornando possível o monitoramento e controle, quebrando barreiras das localidades, pois agora seria possível o controle e o monitoramento de qualquer parte do mundo onde houvesse disposição do serviço.

Com a chegada do século XXI, veio também os tablets, celulares e smartphones o que intensificou a tecnologia juntamente com a internet, telefonia, banda larga, músicas, filmes, monitoramento, controle residencial, etc. juntos formou um cenário extremamente favorável a absorção tecnológica no Brasil.

Porém os preços altamente elevados tornou uma grande barreira para a maioria da população. Mas hoje isso está mudando, com a mudança econômica mundial e os avanços tecnológicos cada vez mais acessíveis, cabe ao usuário à opção de escolha entre o recurso e programação, que melhor lhe atenda.

2.1.2 AURESIDE

AURESIDE - Associação Brasileira de Automação Residencial é uma associação fundada no ano de 2000, com registro legal e sede em São Paulo.
Por se caracterizar uma associação, tem como objetivo, regulamentar normas de padronização da indústria, bem como orientar, treinar profissionais da área, certificação de profissionais, oferecer cursos de capacitação de desenvolvimento profissional, reduzir custos, proteger consumidores, aumentar vantagens competitivas entre outros.

A necessidade de sua existência deu-se, devido ao grande volume de

profissionais envolvido na área de automação residencial. Podemos dizer que a área de automação residencial é composta por uma vasta categoria de profissionais de diversas áreas de atuação do mercado, como engenheiros, projetista, programadores, designer de interiores, arquitetos, técnicos, instaladores e toda demanda tecnológica envolvida. Daí, a necessidade de planejamento de estruturação de métodos e metodologias de gerenciamento, deste novo ramo que surgiu graças à evolução tecnológica.

2.1.3 Características do Mercado Atual

Em alguns anos atrás, pouco se ouvia ou sabia o que era automação residencial ou a “casa inteligente”, porém hoje o mercado de automação residencial está cada vez mais aquecida, graças aos grandes investidores que a cada dia tem buscado maiores tecnologias e funcionalidades nas infraestruturas dos novos imóveis.

Podemos dizer também que, as facilidades na obtenção de bens tecnológicos, são fatores que impulsionam ainda mais o mercado consumidor, pois quanto mais o consumidor pode obter estes bens tecnológicos, maior se torna a vontade de automatizá-los, e o mercado promete crescer ainda mais.

De acordo com AURESIDE - Associação Brasileira de Automação Residencial revelam um crescimento das automações residenciais no Brasil em 80%. Segundo pesquisa feita pela própria Associação, existem aproximadamente no Brasil 60 milhões de imóveis de todos os níveis sociais e, apenas 300 mil imóveis são automatizados e, os indicadores revelando a triplicação destes crescimentos.

Somente no Brasil, são mais de 250 mil empresas especializadas neste ramo de atividade, e devido ao amplo conhecimento de diversas áreas que envolve a automação residencial, muitos integradores acabam por se especializar em dois ou três sistemas que envolve o sistema, para assim poder atender a demanda que promete grandes propagações futuras.

2.1.4 Características do Mercado Consumidor

Apesar dos valores ainda não fazer parte da grande realidade dos brasileiros, a competitividade entre fornecedores e pequenas empresas que entram nesta briga e investem na automação residencial, para obter um mercado mais acirrado e competitivo, proporcionam melhores preços de negociação com objetivo de atender melhor seus potentes e crescentes clientes.

Fica claro que, as pessoas deixaram de ver a automação como algo surreal e luxuoso, passando a ver como algo essencial, onde possa proporcionar além de conforto mais segurança, valorização do imóvel, agilidade e economia em energia principalmente, uma prova disto é que já existe disponível no mercado, empreendimentos de edifícios, que dispõe essa tecnologia para seus clientes.

A automação residencial tem como objetivo deixar a casa ou o imóvel além de mais moderna e inteligente, também possibilitar o fácil acesso de controle aos equipamentos eletrônicos como, ligar televisores de outra parte da casa, gerenciar ar condicionado, programando o horário de sua chegada ou saída, gerenciar câmeras nos ambientes de longa distancias, programar micro-ondas, acender e apagar as luzes quando não estiver, além de programa-los para determinado período de tempo e ser acionados a longa distancia qualquer tipo de dispositivo eletrônico.

Como por exemplo, quando estiver viajando, poderá programar que as luzes se acendam e apaguem em determinado horário, para que os vizinhos e vizinhança possam pensar que há alguém em casa, evitando assim possíveis assaltos ou quaisquer outros imprevistos ao imóvel, como invasão de domicilio.

O Empresário Valdiberto Rodrigues de Araújo deixa bem claro e, expressa com objetividade, resume em poucas palavras do que automação capaz.

> “A gente costuma brincar que o céu é o limite. Tudo que ele tem de equipamento eletrônico na sua residência, toda parte de iluminação, ar condicionado, irrigação de jardim a gente consegue agregar, colocar tudo numa mesma interface para que ele tenha controle total de sua residência em um único ponto”, afirma Araújo.

Depois deste breve comentário, fico sem palavras para descrever de melhor forma o que a automação residencial pode fazer em nossas vidas.

2.2 Arduino

2.2.1 Arduino

A palavra "Arduino" é nome próprio italiano de origem germânica, o Arduino trata-se de uma plataforma de desenvolvimento open-source é uma placa com dispositivos eletrônicos, composta de hardware e software flexível de fácil utilização, o micro controlador Atmel AVR é usado para o desenvolvimento da programação.

Seu público alvo é direcionado aos técnicos, engenheiros, designers, artistas e/ou toda sorte de pessoas que desejam ou querem aplicar tecnologias de interação aos ambientes ou objetos, a fim de torna-los mais atrativos ao mercado consumidor.

O Arduino como já foi dito, é uma placa eletrônica composta principalmente com um controlador, linhas de conexão de entrada e saídas digital e analógica, onde serão conectados os devidos dispositivos para controle, uma entrada UBS para se conectar ao computador onde serão realizadas as interversões de programação e diversos componentes eletrônicos a fim de garantir a integridade do funcionamento de todo o conjunto.

Ele tem também como característica, sua própria linguagem e, seu próprio ambiente de desenvolvimento podendo ser comprado em lojas ou até mesmo fabrica-los para os mais adeptos ao sistema. Por conta destas características seu uso torna-o de fácil utilização aos amadores.

Alguma das vantagens de se usar o Arduino é porque ele é fácil de programar, pois possui código enxuto, seu compilador é conhecido, executa de forma rápida, leve e segura, pode ser agregado funcionalidade adicionais de baixo custo entre outros.

2.2.2 História

Com o intuito de proporcionar menores custos de orçamento em projetos escolares, comparados ao daquela época, um microcontrolador custava em média 100 euros, iniciou-se então, o projeto na cidade de Ivre, Itália, no ano de 2005 no Instituto de Design de Interação. Criado e desenvolvido por Massimo Banzi, David

Cuartielles, Tom Igoe, Gianluca Martino e David Mellis, formaram também um time de suporte ao Arduino. As placas foram produzidas em apenas dois dias e, tinha uma tecnologia, acessível, eficiente e compatível com Windows, Mac e Linux, e foi ao desenvolver o software para conversar com o computador, que surgiu o desejo de produzir de forma profissional.

Logo após a menção honrosa na categoria Comunidades Digitais em 2006, pela Prix Arts Eletrônica (Prêmio de Artes Eletrônicas), iniciou seu sucesso onde pode atribuir além do nome da marca, mais de 50 mil placas foram vendidas até o mês de outubro de 2008, estima-se que no final de 2012 foram comercializadas mais de meio milhão de placas de todos os modelos.

Seu hardware atualmente é composto pelo microcontrolador Atmel AVR, no entanto, ele pode ser facilmente modificado através de agregação de placas para expansões, com o objetivo de adaptar-se as necessidades particulares de cada projeto e, assim recebem nomes exclusivos. O sistema do Arduino também pode ser montado pelo próprio usuário, que implicaria em menores custos de fabricação, além de dispor um serviço de venda de pré-montados e distribuidores oficiais, o que facilita ainda mais sua produção.

Hoje o Arduino pode ser facilmente encontrado na internet ou em lojas físicas, tem um custo médio de a partir de 100 reais a placa, podendo ser personalizada dependendo do tipo de aplicação que será utilizada ficando assim ao gosto de freguês.

2.2.3 Versões

Com o conceito de *software* livre, dá-se a grande gama de “variedades” de Arduinos no mercado. Como o nome Arduino é uma marca patenteada, a nomenclatura dos protótipos muda de acordo com seus idealizadores, como por exemplo, Seeduino, Feeduino, Garagino, entre outros, podendo inclusive ser feito em casa, seguindo o esquema do projeto original, descritos em site oficial www.arduino.cc. O site está em inglês, lá é possível encontrar a descrição dos Arduinos “oficias”, com seus respectivos esquemas elétrico.

A diferença entre eles está no ATmega (microcontrolador) utilizado, a quantidade de portas analógicas e digitais e até mesmo em alguns “apetrechos” que

podem compô-lo, como placa wifi integrada, placa ethernet, que obviamente deixa o custo do produto mais caro para ser adquirido, principalmente aqui no Brasil. Mas, em geral, todos tem a mesma finalidade para o desenvolvimento dos projetos. Quando não tem esses itens embutidos na sua placa, pode-se utilizar placas separadas que podem ser acopladas ao Arduino, são as chamadas Shields.

Segue abaixo algumas das placas mais populares comercializadas:

Figura 1- Arduino Uno R3

Fonte: www.arduino.cc

Figura 2- Arduino Leornado

Fonte: www.arduino.cc

Figura 3 - Arduino Ethernet Shield

Fonte: www.arduino.cc

Figura 4 - Arduino Due

Fonte: www.farnellnewark.com.br

Figura 5 – Arduino Mega ADK

Fonte: www.farnellnewark.com.br

Figura 6 - Arduino Micro

Fonte: www.arduino.cc

Figura 7 - Arduino Mega

Fonte: www.arduino.cc

Figura 8 - Arduino Motor Escudo

Fonte: www.arduino.cc

Figura 9 - Arduino GSM Escudo

Fonte: www.arduino.cc

Figura 10 - Arduino Robot

Fonte: www.arduino.cc

2.2.4 Especificações Técnicas Uno R3 (Modelo usado no projeto)

O Arduino que usaremos em nosso campo de estudo será o Arduino Uno R3, sendo ele um dos modelos considerado mais básico de toda linha dos Arduino, por conta de não conter nenhum tipo de acessório adicional de série. Essa escolha foi feita exatamente pelo fato que, ao decorrer do projeto poderíamos ir agregando e personalizando ao nosso trabalho de acordo com as nossas necessidades. Diga-se de passagem, que a escolha no modelo a ser utilizado para o desenvolvimento do

nosso trabalho, gerou grandes discursões, então através de um comum acordo optamos pelo modelo mais simples, o que nos daria maior autonomia para elaboração do trabalho desenvolvido.

Tabela 1. Especificações Arduino Uno

Microcontrolador	ATmega328
Flash Memory	32 KB (ATmega328) dos quais 0,5KB são utilizados pelo bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Pinos de entrada analógica	6
Pinos E/S digitais	14 (dos quais 6 podems ser saídas PWM)
Corrente CC por pino E/S	40 mA
Corrente CC para o pino 3,3V	50 mA
Velocidade de Clock	16 MHz
Voltagem Operacional	5V
Voltagem de entrada (recomendada)	7-12V
Voltagem de entrada (limites)	6-20V
Conexão USB	1
Alimentação	Conexão ICSP
Botão de Reset	1

Fonte: Tabela de especificação elaborada pelo autor

Como dito anteriormente, existem vários tipos de Arduino. Utilizaremos o Arduino Uno R3, que significa “Um” em italiano, uma das versões mais recentes e de baixo custo além de ser recomendada para quem está começando, assim como nós.

Os modelos existentes possuem uma estrutura semelhante, então, vendo as características básicas de um, vemos as demais.

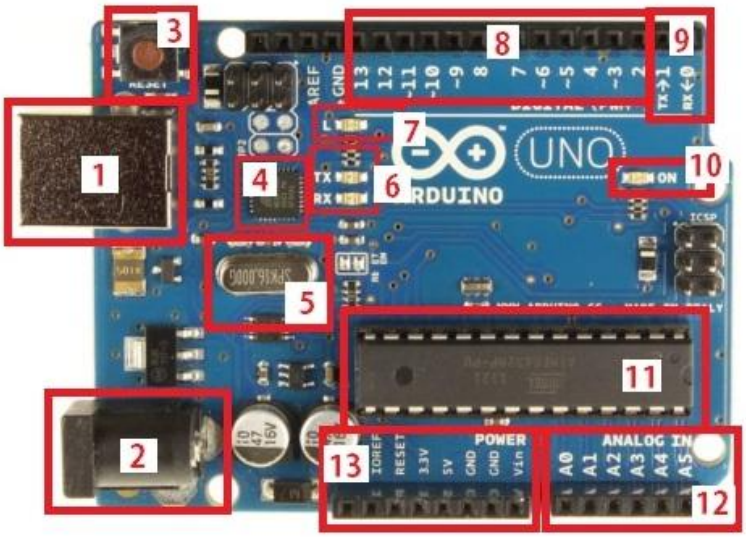
Figura 11 - Arduino Uno R3, frente e costa

Arduino Uno R3 Front *Arduino Uno R3 Back*

Fonte: www.arduino.cc

A figura abaixo mostra as principais partes de um Arduino:

Figura 12 - Arduino Uno R3



Fonte: www.arduino.cc (editado pelo grupo1)

[1] A imagem foi extraída do site oficial www.arduino.cc. A edição da imagem foi baseada num blog de Alex Giuliano Martins, que dá a descrição da placa.

1- Entrada USB: Faz comunicação direta com o computador que provém energia à placa para seu funcionamento.

2 - Alimentação: Utilizada quando o Arduino não está conectado no computador. A alimentação pode ser feita por uma bateria de 9 volts ou utilizando uma fonte AC/VC de 7 a 12 volts na tomada.

3 - Botão Reset: Reinicia a programação do Arduino.

4 - Chip Atmel: Não é o nome específico dele, mas é ele quem faz a comunicação com o computador.

5 - Cristal de 16 MHz: Realiza a frequência do microcontrolador.

6 - TX/RX (leds): Quando a placa está conectada ao computador, esses leds indicam que estão transmitindo (TX) e recebendo (RX) os dados.

7 - Led: Muito utilizado para pequenos testes. Está ligado ao pino 13. Com ele é possível saber se a placa está se comunicando de maneira adequada ao computador sem precisar de algum componente externo. Um exemplo disso é mostrar através do “Monitor Serial” (no programa), mandar um comando “*Blink*” para o led ficar piscando.

8 - Pinos Digitais: São as portas digitais de entrada/saída que vão de 0 a 13, sendo que as portas que possuem o símbolo “~” (til) são as chamadas PWM, conseguem “modular” a frequência entre 0V e 5V.

9 - TX/RX (Pinos): Toda ligação externa utiliza esses dois pinos para se comunicarem.

10 - On (led): Indica que a placa está ligada.

11 - Microcontrolador Atmega328: Fabricado pela Atmel é o microcontrolador principal, onde gravamos os programas criados. Nele está uma das diferenças entre os variados tipos de arduinos, pois sua capacidade varia de acordo com os alguns modelos. O Uno, por exemplo, possui 8bits com 32KB de memória, já o Mega, possui 256KB.

12 - Pinos Analógicos: São as portas de entrada. Diferente das portas digitais, as analógicas podem apenas receber, ou seja, só podem “ler” os dados.

13 - Power: Servem como alimentação para outros dispositivos (3.3V, 5V e Gnd = “terra” de 0V).

2.2.5 Segurança e Viabilidades de Uso

Com citamos anteriormente, o uso do Arduino torna-se viável, devido a diversos fatores, tanto por meios técnicos como por meios econômicos.

Open Source (software e hardware): por ser um hardware e software livre, é possível encontrar diversos artigos disponíveis gratuitamente no mercado, o que facilita também um vasto suporte e orientações de diversos usuários. Ele também possui um IDE (Integrated Development Environment – Ambiente de desenvolvimento integrado), permitindo ser adquiro gratuitamente em vários sites ou no www.arduino.cc.

IDE: a programação poderá se desenvolvida no próprio IDE, o que facilita, pois este ambiente de desenvolvimento tem a vantagem de simplicidade de utilização de usuários leigos no assunto.

Multiplataformas: por ser compatível com os principais sistemas operacionais como Windows, Macintosh e Linux, tornando viável o seu uso e sua aplicação, o que implica em maior segurança e menores custos, pois não será necessária aquisição de softwares e hardwares específicos para seu uso.

Fácil Manutenção: por ser uma placa com sistema elétrico bem acessível, podendo até mesmo ser fabricada pelo próprio usuário, sua manutenção por sua vez torna-se simples e de baixo custo, pois a troca dos componentes tem custo pequeno.

Programação: sua programação é feita da porta serial USB, o que hoje em dia é muito comum encontrar em um computador desktop, netbook, notebook e até os tablets.

Eletrônica: a parte eletrônica do dispositivo é muito simples e não requer grandes conhecimentos dos envolvidos, para sua execução. Uma pessoa com boa vontade e disposição para ler e seguir alguns tutoriais, será capaz de projetar um circuito eletrônico sem grandes dificuldades.

Fonte de informação: possui uma vasta fonte de informações de diversas partes do mundo, por ser um software livre, seus adeptos a esta ideologia, compartilham diversos materiais de apoio aos iniciantes e até mesmo para os mais experientes que encontram novos problemas.

2.2.6 Acessórios

Os acessórios são essenciais dependo da aplicação que será usada no Arduino, eles são conhecidos como Shields em inglês, que traduzindo significa escudo. Os Shields são placas de circuitos eletrônicos que são adaptáveis ao modulo do Arduino através de terminais para conexão dos mesmos, com a finalidade de expansão ou agregação de funcionalidades, tais como, comunicação com rede de computadores (placa Ethernet), comunicação bluetooth (modulo de bluetooth), controle de motores e alta tensão (Motor Shield L293D) entre outros.

Dependo do que o usuário necessita, é possível encontrar a venda nos mercados, placas de Arduinos com estas funcionalidades já embutidas na própria placa, sabendo-se que isto implicará em maiores custos. Também é necessário que se faça uma boa pesquisa de valores de comercialização, para chegar a um bom senso, se é mais vantajoso comprar uma excelente placa com diversas funções ou ir expandindo conforme a necessidade.

Vale ressaltar que também será necessário alguns componentes de eletrônica, como resistores, leds, jumper, relés, capacitores, diodos, push button e toda sorte de componentes utilizados na montagem de circuitos eletrônicos. Estes componentes são usados para auxiliar a conexão entre o Arduino e o meio onde quer se estabelecer a conexão, tem um papel especial de estabilizar, direcionar e controlar a função desejada.

Veja abaixo a lista de algumas placas de Shield:

Shields Extensores
Placa de Relé
Arduino Ethernet Shield
Liquidware TouchShield
Liquidware InputShield
O XBee Shield
Arduino Motor L293D
Módulo Bluetooth

2.3 Android

2.3.1 Android

Podemos definir Android como um sistema operacional criado para dispositivos móveis, como tablet's e celulares. Mas para entendermos a função desse sistema, precisamos primeiramente saber a sua real importância.

O sistema operacional nada mais é que uma coleção de programas iniciados com o computador, programas esses responsáveis pelo bom funcionamento do hardware. É através dele que os programas são gerenciados.

Quando abrimos um simples editor de texto como o Word, utilizado para fazer esse documento, é aberto também um processo no SO (vulgo Sistema Operacional), que controla o funcionamento do programa, sua prioridade de execução, as tarefas geradas por ele, entre outros e nos disponibiliza diversas opções, entre elas o fechamento desse aplicativo em caso de mau-funcionamento, algo que algumas vezes é necessário e gera certa dor de cabeça para o usuário.

Atualmente existem diversos SO's, alguns bastante consagrados como o onipresente Windows, que está em mais de 90% dos computadores pessoais como notebooks e desktops, o Macintosh, famoso sistema da gigante Apple e o Linux, que é gerado por diversos desenvolvedores em todo o mundo. O Google também já se aventurou no mundo dos desenvolvedores de SO's, com o seu Chrome OS, que ainda não deslanchou.

Figura 13 – Sistemas Operacionais

Fonte: http://www.tecnologia.culturamix.com

Para dispositivos móveis, além do ator principal desse documento, existem outros muito conhecidos como o IOS da Apple, o Windows Phone e o Symbian, um sistema criado pela Nokia, mas que se encontra em processo de extinção por ser apenas um coadjuvante no cenário atual em relação aos citados anteriormente, entre outros.

Tendo como base essas informações, não nos restam dúvidas de que ele é o principal programa de um equipamento, sem ele o Hardware se limitaria a um monte de lixo eletrônico.

2.3.2 História

Tudo começou em 2003, no mês de outubro, em uma cidade chamada Palo Alto, (Califórnia), quando 4 profissionais de TI fundaram a Android Inc., empresa com foco em desenvolvimento de sistemas operacionais para celulares provenientes do Linux.

Figura 14 – Palo Alto, EUA

Fonte: www.google.com.br

O gigante de buscas Google, desejando lançar aparelhos com serviços baseados em localização, não possuía uma plataforma necessária para isso, então viu nessa pequena empresa a oportunidade de colocar seus projetos em prática. Sendo assim, em agosto de 2005 a adquiriu e começou a corrida para desenvolver um sistema que fosse flexível e de fácil migração para os fabricantes.

Até então, o ingresso do Google nesse mercado não passava de mera especulação, mas em novembro de 2007, foi anunciado por essa multinacional a criação da *Open Handset Alliance (OHA)*, um conselho com mais de 33 empresas parceiras (entre elas Motorola, HTC, Dell, Samsung , LG, Nvidia e muito mais), que resolveram definir uma plataforma única e aberta, assim sendo, anunciaram que o Android passaria a ser open source (um software de código livre), para quem quisesse desenvolver e aprimorá-lo, adicionando novas funcionalidades ou até mesmo corrigindo falhas.

Os objetivos dessa aliança são os de melhorar a vida das pessoas através do desenvolvimento da tecnologia, deixar os consumidores mais satisfeitos com o produto final e ajudar a eles mesmos através da criação de aplicações corporativas.

Um ano após esse acontecimento, foi lançado o primeiro aparelho Android, o Dream G1, da marca HTC, que possuía ótimos recursos para a época, como integração com o webmail do Google(Gmail) , janelas de notificações, e uma central

de download, o Android Market, onde foram adicionados milhares de aplicativos e acabou sendo substituído pelo Google Play.

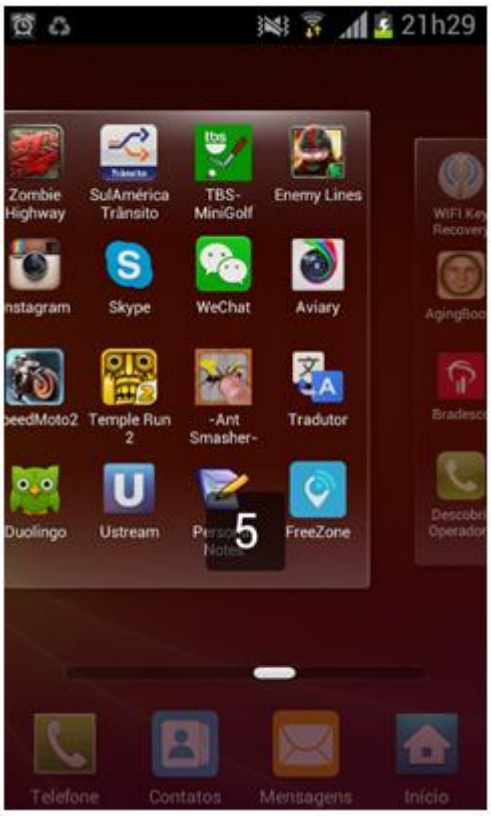
2.3.3 Conhecendo um pouco mais sobre Android

Qualquer celular nos dias atuais não é um simples aparelho para efetuarmos e recebermos ligações, mas uma verdadeira “caixa de ferramentas”, isso desde o mais simples até os incríveis Smartphones, que nos permitem ter em um único dispositivo câmeras de alta resolução, GPS, acesso à internet, entre outras infinitas opções.

Sendo assim, podemos diferenciar o Android dos diversos sistemas existentes para dispositivos móveis.

Seu design fascinante, customizável e estiloso (como podemos ver na imagem abaixo), unido a sua capacidade de integrar diversos aplicativos para facilitar nossas vidas, faz com que ele se torne uma excelente opção. Além disso, por ser um sistema open source, conforme mencionado anteriormente, é também constantemente melhorado, o que só gera benefícios aos seus usuários.

Figura 15 – Screenshot Tela de Menu



Fonte: Elaborada pelo autor

2.3.4 Google Play

Atualmente, o Android conta com o aplicativo Google Play, em substituição ao Android Market. Através dele podemos efetuar o download de aplicativos, jogos, livros e filmes, sendo que uma enorme vantagem deste sistema é possuir aplicativos grátis em sua grande maioria.

Vale ressaltar que, apesar de ser o aplicativo oficial, o Google Play não é o único. Alguns aparelhos usam uma loja de apps da própria fabricante, como por exemplo a Amazon, com a Amazon App Store.

Outro dado interessante é que, segundo o Google, a loja Google Play tem mais de 700 mil aplicativos e mais de 20 bilhões de downloads já foram feitos - seja de apps pagos ou gratuitos.

Figura 16 – Screenshot Google Play

Fonte: Elaborada pelo autor

2.3.5 Maps

No Google Maps, temos diversas funcionalidades que são bastante úteis no dia-a-dia.

Como por exemplo, saber como anda o transito ao seu redor.

Figura 17- Screenshot Google Maps

Fonte: Elaborada pelo autor

Além disso, podemos traçar uma rota através do GPS, encontrar locais de interesse como parques, restaurantes, bares, etc.

Enfim, se fossemos nos aprofundar nas funcionalidades desse sistema, facilmente escreveríamos um livro, tamanha a quantidade de utilidades.

2.3.6 Versões

A primeira versão do Android desenvolvida pelo Google juntamente com a OHA e que foi comerciada, foi criada em setembro de 2008.

Desde 2009, as versões Android desenvolvidas possuem um codinome em ordem alfabética: Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich e Jelly Bean.

Ele já é o sistema operacional móvel mais utilizado no mundo, com mais de 75% de participação no mercado, e a versão mais utilizada é a Gingerbread (2.3), rodando em aproximadamente 45% dos dispositivos.

Em relação aos dispositivos, a Samsung lidera com folga o ranking, a cada 10 aparelhos utilizados no mundo, 8 são da fabricante sul-coreana.

2.4 Software de Aplicação

2.4.1 App Inventor

O App Inventor é um aplicativo disponibilizado em uma página de internet onde os usuários podem criar aplicações de Android sem ter noções de qualquer linguagem de programação específica, podendo colocar suas aplicações no Google Play disponibilizando para outros usuários de forma gratuita ou não.

Foi criado pelos desenvolvedores da empresa Google por solicitações de usuários do sistema operacional Android. O projeto App Inventor foi administrado pelo Google por um período que durou da data de disponibilização em 12 de Julho de 2010, com o lançamento oficial no dia 15 de dezembro de 2010 até o dia 31 de dezembro 2011.

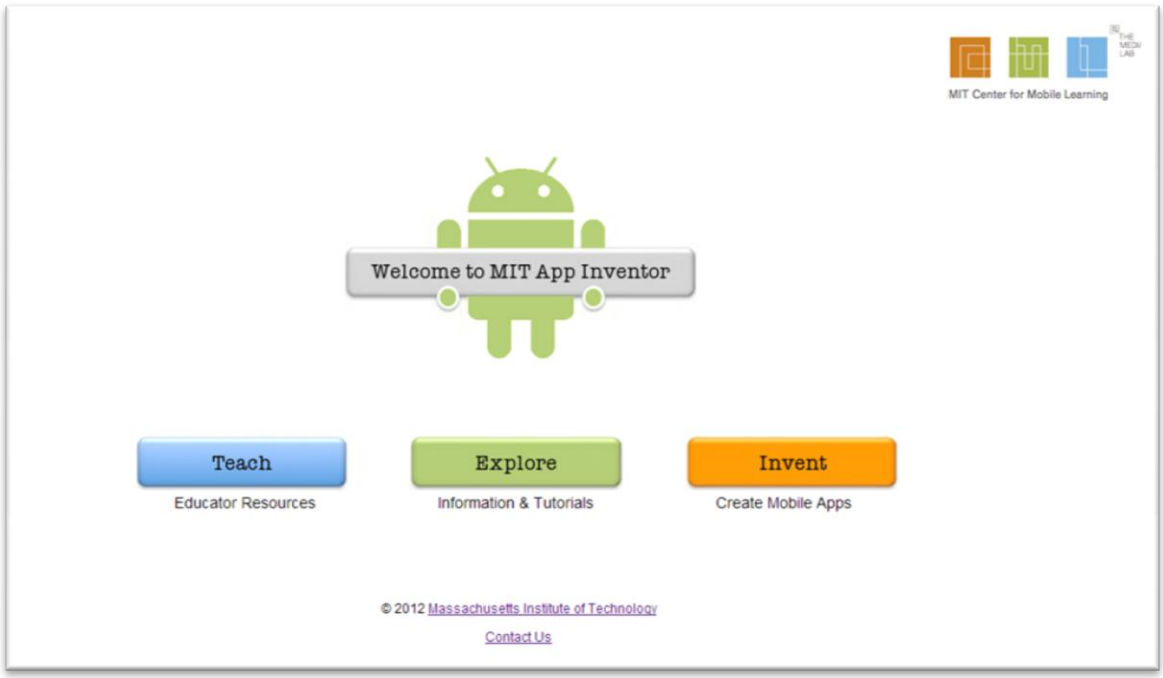
Após essa data o Centro do MIT (*Massachussets Institut of Technology - "Instituto de Tecnologia de Massachusetts")*, uma das melhores universidades de desenvolvimento no ambiente tecnológico do mundo, tomou a frente do projeto mudando seu nome para MIT App Inventor, que consiste em um ambiente de desenvolvimento online com a programação beseada em *Open Source* como o StarLogo TNG, através de *OpenBlocks*, desenvolvido em Java.

Apesar de estar no mercado há algum tempo, este aplicativo ainda está em sua versão Beta em constante desenvolvimento, a cada seis meses, mais ou menos, tem uma atualização disponível.

Para utilizar o programa, basta ter uma conta do G-mail e a partir disso ir seguindo o passo a passo para a instalação do software no computador. A parte lógica do programa é desenvolvida através do encaixe de “blocos de comandos”, então, como peças de quebra-cabeças, vai-se encaixando os blocos sem a necessidade de inserir linhas de comando, como a maioria dos programas, tornando assim mais fácil a concepção do aplicativo.

O software *open source* pode ser instalado nos sistemas operacionais Windows, Linux e Mac.

Figura 18 – Página inicial do MIT App Inventor



Fonte: http://appinventor.mit.edu/

2.4.2 Vantagens e Desvantagens

Vantagens

- É um método diferente para se programar, instigando a criatividade.
- A aparência é atrativa, os itens são fáceis de serem vistos.
- Apesar de estarem em inglês, os tutoriais são bem explicativos. Indo desde um nível fácil até um mais elaborado.
- Métodos diferentes para a simulação. Quem não tem um smartphone ou tablete em mãos, pode usar o simulador online.

Desvantagens

- Está um pouco instável. As atualizações, às vezes, travam os novos projetos.
- O site está todo em inglês, e a tradução online não ajuda tanto. Como tudo é traduzido ao pé da letra, alguns comandos também são traduzidos, dificultando o entendimento da sintaxe.
- Não são todos os itens que podem ser emulados online.
- O software possui restrição de tamanho de desenvolvimento, o que impossibilita o uso de grandes desenvolvimentos de aplicativos.

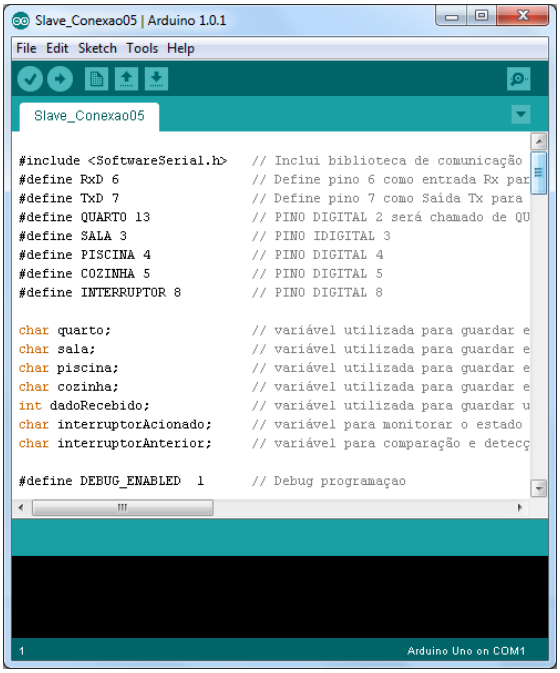
2.5 Plataformas de Programação

2.5.1 IDE

A figura abaixo mostra a plataforma de desenvolvimento do Arduino, é nela que são gerados os códigos de implementação de um projeto, as bibliotecas são

adicionadas e, através de comandos específicos são referenciando as bibliotecas para executarem conforme a necessidade.

Figura 19 – Arduino IDE



Fonte: Elaborada pelo autor

2.5.2 Linguagem de Programação

A linguagem de programação do Arduino é baseada na Linguagem C/C++. Trata-se de uma linguagem estruturada na qual dividimos basicamente em 2 blocos de funções: setup() e loop().

Uma Função em Linguagem de Programação nada mais é do que uma declaração de procedimentos, rotinas e ações que o programa irá realizar. A mesma deve ser criada antes de qualquer chamada de outras funções que referencie á ela. Deve-se atribuir um tipo á função, um nome seguido de parênteses na qual se passam parâmetros á função. No caso da função não receber parâmetros, os parênteses podem ficar vazios.

A função SETUP é executada uma única vez, assim que a placa é ligada ou reiniciada. Abaixo o exemplo da estrutura SETUP do Arduino.

```
void setup(){
}
```

Para deixar mais claro, quem já programou em qualquer outra linguagem, o setup seria semelhante ao *main* da Linguagem C e C++.

A função LOOP é executada de forma indefinida, sendo o principal bloco de desenvolvimento do seu programa. O bloco Loop é escrito da seguinte forma:

```
void loop(){
}
```

2.6 Eletrônica

2.6.1 Composição Eletrônica

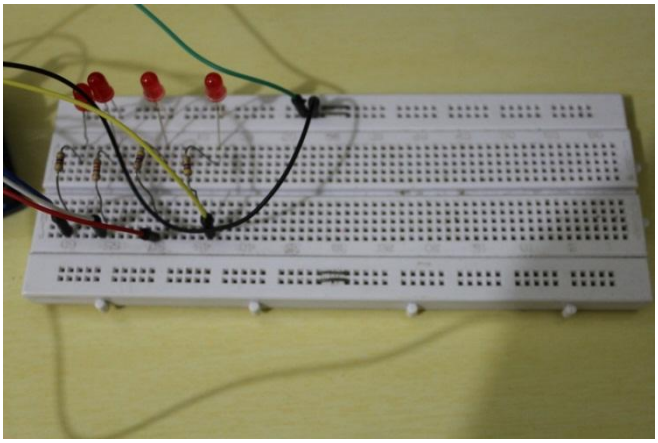
Para realizamos a implementação deste projeto de automação residencial, foi necessário um conhecimento mínimo de eletrônica, nada que um leigo no assunto não possa realizar. Inicialmente foi necessária a montagem do projeto em uma Protoboard, que nada mais é uma placa onde são realizados os protótipos de placas eletrônicas, antes que as mesmas sejam confeccionadas em definitivo. A Protoboard serve para garantir a integridade e funcionamento total do circuito antes mesmo que sejam produzidas oficialmente, pois elas possuem terminais horizontais e verticais que proporciona conexões entre si. Os componentes são sobreposto a placa sem que haja a necessidade de solda-los, abaixo encontramos uma figura ilustrativa.

Figura 20 - Protoboard

Fonte: www.pares.com.br

Também foram utilizados uma serie de componentes para estabilização do circuito, que neste caso, foi usado resistores comuns 330ohm, resistor de pull-up 10K, leds (diodo emissor de luz), chave liga/desliga, lâmpada de 127volts, placa para confecção do circuito definitivo, fios para conexão e, um multímetro para medição e verificação de tensões, corrente, continuidade, e resistência do circuito, as imagens abaixo representa cada componente utilizado:

Figura 21 – Esquema elétrico do circuito usado



Fonte: Elaborada pelo autor

2.6.2 Componentes Utilizados

Resistor de 330omh – usado para controlar a corrente que é chegada até o led (diodo emissor de luz), a fim de evitar que sua luminosidade perca em um curto espaço de tempo.

Resistor pull-up 10K – utilizado para controlar o tempo

Leds – são diodos emissores de luzes, usaremos para simular uma lâmpada, trazendo iluminação a alguns ambientes da casa. Sua escolha se deu, devido as características eletrônicas possuídas, pois trabalha com baixa voltagem, ou seja corrente continua DC.

Lâmpada de 127v – como previsto, nosso objetivo é desenvolver projeto para residência e isto implica em usarmos a corrente alternada, para que seja mostrado a elaboração do nosso trabalho, através da aplicação de corrente AC usada na lâmpada de 127v.

Fios – usaremos fios flexíveis comuns, para realizar as ligações necessárias de todo circuito.

Placa de circuito impresso – um circuito devera ser desenvolvido de forma a ser definitiva, uma vez que fora testado na protoboard, esta seria a parte de finalização, confeccionar a placa oficial do projeto.

Multímetro Digital – por ser uma das ferramentas mais utilizadas na área de eletrônica, seu uso em nosso trabalho se faz indispensável, pois com ele podemos verificar e medir tudo que estiver relacionado as grandezas elétricas, como resistência, continuidade do circuito, corrente entre outros.

Placa de relé – devo citar que esta placa de relé, apesar de ser um acessório, ele poderia ter sido reproduzida pelo grupo, porém devido ao tempo hábil para seu desenvolvimento, optamos por fazer a compra da mesma, do mesmo modo se dá a uma serie de placas e circuito pré-montados, que estão disponíveis para comercialização no mercado.

3. MATERIAIS E METÓDOS

3.1 Áreas de estudo

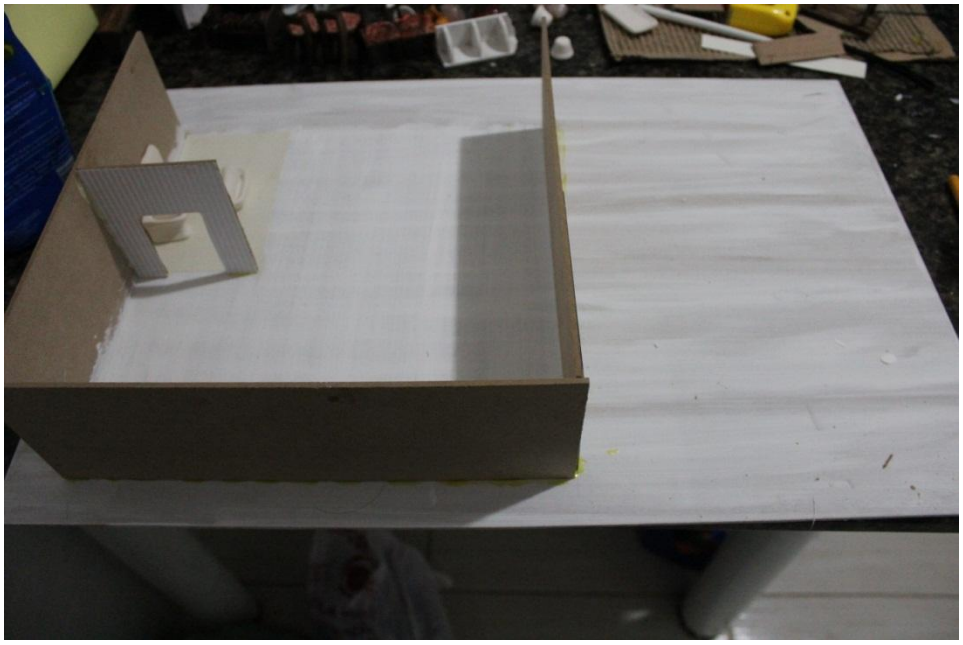
Visando reproduzir e viabilizar a automação residencial, fazendo com que esta realidade chegue o mais próximo possível da realidade de muitos de nós, é que

tivemos a ideia de automatizar uma residência, a fim que até nós mesmo pudéssemos ser beneficiados com o desenvolvimento deste recurso no futuro próximo.

Elaboramos um modelo de uma maquete residencial, onde podemos acompanhar seu desenvolvimento logo abaixo. Nela é possível visualizar os devidos ambientes onde serão aplicadas as instalações elétricas para o controle que iremos desenvolver no decorrer deste trabalho.

Ela é uma das partes principais deste trabalho, pois é através dela que poderemos visualizar os comandos e testes que realizaremos a seguir.

Figura 22 – Fase inicial da maquete da casa



Fonte: Elaborada pelo autor

Figura 23 – Fase de modelagem dos ambientes da casa

Fonte: Elaborada pelo autor

Feito isto, foi necessário reproduzi-la oficialmente, para que as instalações pudessem ser feitas. Como podemos observar logo abaixo todo seu desenvolvimento, seu passo a passo e enfim sua finalização.

Figura 24 – Maquete finalizada

Fonte: Elaborada pelo autor

Outra etapa que estaria sendo efetuada em paralelo a construção da maquete é a programação do Arduino juntamente com o desenvolvimento do App Inventor. Estes passos são fundamentais, pois todos eles ao final deverão ser integrados uns aos outros.

3.1.2 Instalação e Configurações de Softwares

Antes mesmo de começarmos a realizar qualquer feito, é necessária a verificação e a instalação caso seja necessária de alguns programas e software, para garantir a perfeita execução de todas as funcionalidades do sistema.

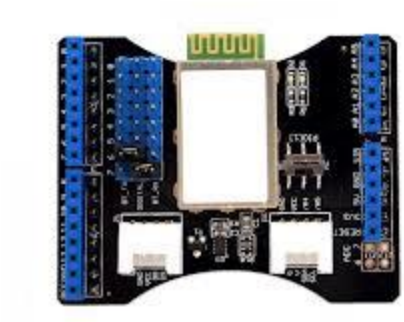
O Java é um destes programas que deve estar atualizado com a versão mais recente em uso e, neste caso em especial não iremos demostrar o passo a passo de sua instalação devido sua simplicidade de instalação, em alguns computadores ele é atualizado automaticamente.

Figura 25 – Tela de atualização do Java

Fonte: http://pt.kioskea.net

Para o módulo de Bluetooth Shield, também foi necessário programá-lo para que o Arduino o reconhecesse e, outras tratativas foram tratadas dentro da programação do Arduino, para que se estabilizasse o módulo de bluetooth.

Figura 26 – Módulo de Bluetooth Shield



Fonte: www.labdegaragem.org

3.1.3 Instalação do Aplicativo App Inventor

Para utilizar o aplicativo não é difícil. Seguindo o passo a passo, mesmo o conteúdo do site estando em inglês, dá para instalar sem maiores problemas. Aqui está descrita a explicação de sua instalação:

Ao clicar no logo da página inicial ou no botão “Explorer”, aparecerá a página com o conteúdo disponível do site. Essa página contém várias guias, onde podemos explorar para saber mais informações.

Clique na guia “Setup”.

Figura 27 – Acessando a tela de setup

Fonte: Print da tela App inventor elaborada pelo autor

Aparecerá uma imagem com os passos a seguir para instalação.

Figura 28 – Instalação do aplicativo

Fonte: Print da tela de instalação elaborada pelo autor

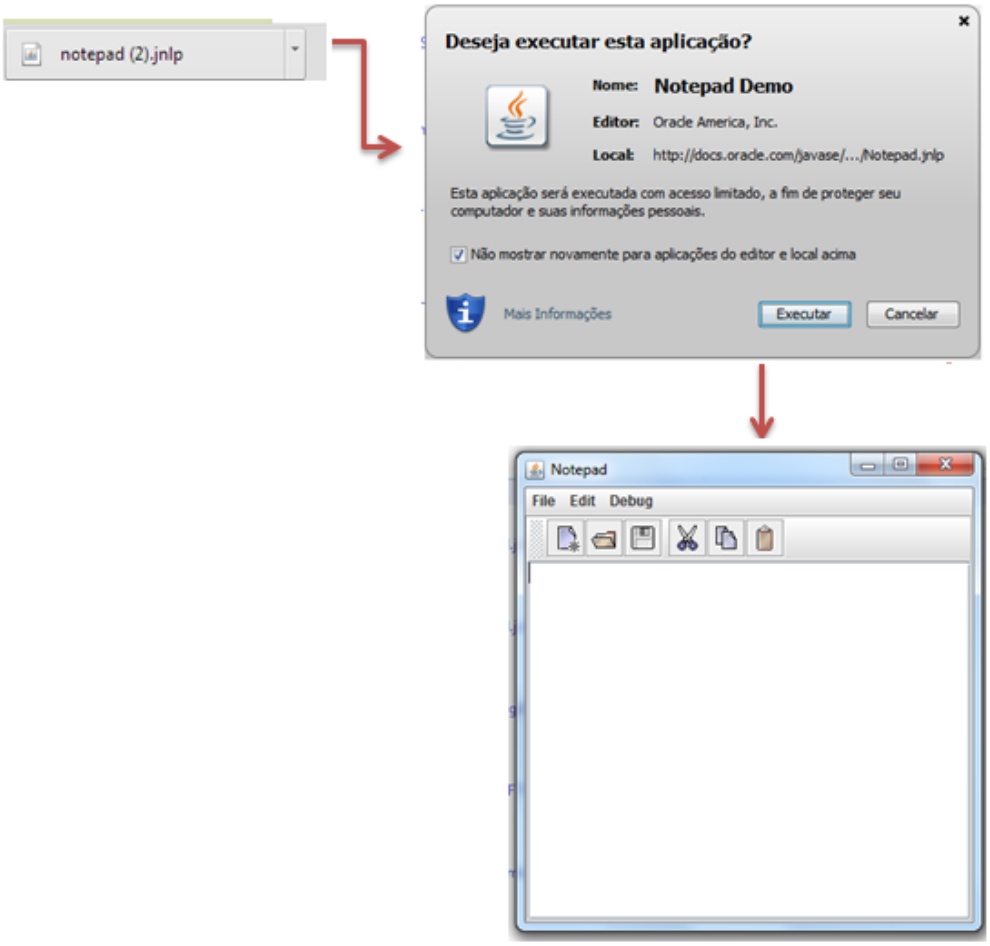
Ao clicar no item “1.Prepare Your System Java”, será direcionado a uma página que fará a verificação se o Java está instalado. Caso não tenha certeza se o mesmo está instalado ou não é verificado através do botão “Launch”.

Figura 29 – Ilustração do botão

Fonte: Print do botão elaborada pelo autor

Será baixado um arquivo “notepad.jnlp”, que dependendo do sistema operacional utilizado pode começar automaticamente ou pode ser aberto manualmente.

Figura 30 – Execução da aplicação instalada

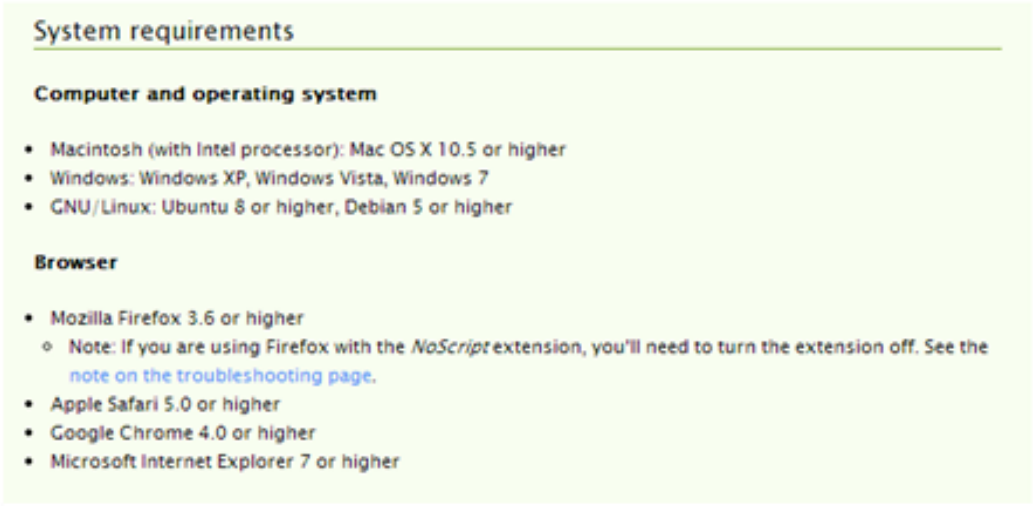


Fonte: Print da tela de execução elaborada pelo autor

Ao clicar em executar e o Notepad aparecer significa que o Java está instalado corretamente. Caso não apareça, é mostrada algumas possíveis soluções como desabilitar o firewall do computador, reinstalar o programa entre outros.

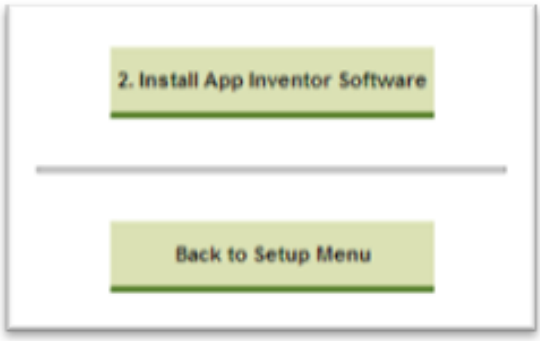
Na mesma página é mostrado os requisitos do sistema e em seguida o próximo passo.

Figura 31 – Requisitos necessários para instalação



Fonte: Print da tela de requisitos elaborada pelo autor

Figura 32 –Tela de instalação



Fonte: Print da tela de instalação elaborada pelo autor

A página de instalação mostra os sistemas operacionais que deve ser escolhido:

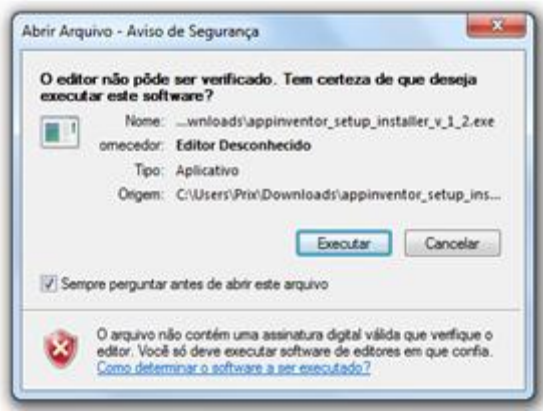
Figura 33 – Escolha do sistema operacional

Fonte: Print da tela de requisitos elaborada pelo autor

O sistema operacional para o desenvolvimento do projeto foi o Windows. Ao clicar em “Download”, o instalador é baixado. Ao baixar, deve ser localizado na pasta de destino, com o nome **AppInventor_Setup_Installer_v_1_2.exe (~ 92 MB).** Depois é só instalar.

Clique em Executar, como na figura abaixo e, logo abrirá uma caixa de diálogo, então só apertar na tecla “Next” para prosseguir, conforme relação das figuras mais abaixo.

Figura 34 – Aceitação de instalação



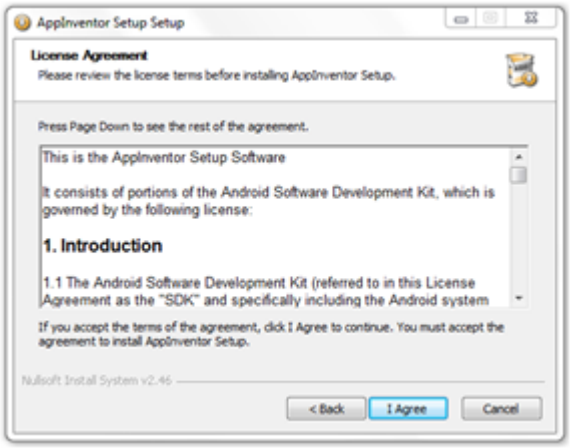
Fonte: Print da tela de aceitação da instalação elaborada pelo autor

Figura 35 – Prosseguir instalação

Fonte: Print da tela de prosseguir instalação elaborada pelo autor

Clique em “I Agree” para aceitar os termos da licença e continuar

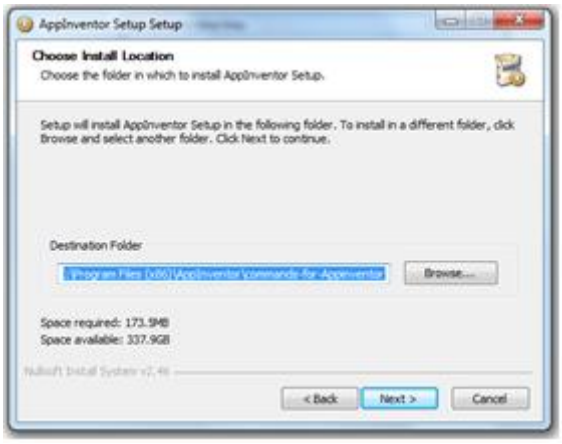
Figura 36 – Aceitação dos termos de uso



Fonte: Print da tela de aceitação de termos de uso elaborada pelo autor

É escolhido o local para gravar o programa, geralmente fica em C:\Program Files (X86), nos arquivos de programas, mas pode ser redirecionado para outro lugar através da opção “Browse”. Depois de escolhido o destino, clique em “Next” novamente.

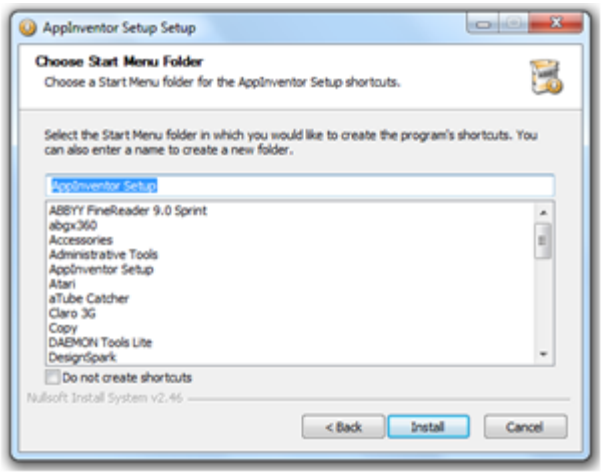
Figura 37 – Local de instalação



Fonte: Print da tela do local de instalação elaborada pelo autor

Aqui se tem a opção de criar ou não um atalho no menu iniciar. A opção será invalidada se for marcado em “Do not create shortcuts” (Não criar atalho).

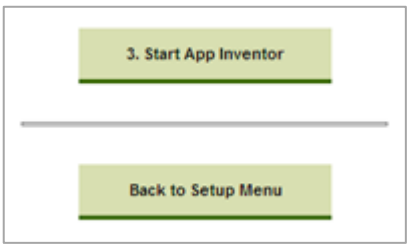
Figura 38 – Opções de escolha de atalhos



Fonte: Print da tela de opções de escolhas de atalhos elaborada pelo autor

Clicando em “Install” ele finaliza com a instalação do programa.
Após o término da instalação o próximo passo é utilizar a aplicação no site.

Figura 39 – Start App Inventor



Fonte: Print da tela de Start elaborada pelo autor

3.1.4 Instalando e configurando o Arduino

Para começar, acesse o site http://arduino.cc/en/main/software e faça o download do software em formato Zip, atualmente a última versão disponível é a 1.0.5.

A instalação é relativamente simples, vamos seguir o passo-a-passo, vale ressaltar que estamos utilizando o Windows 7 como Sistema Operacional.

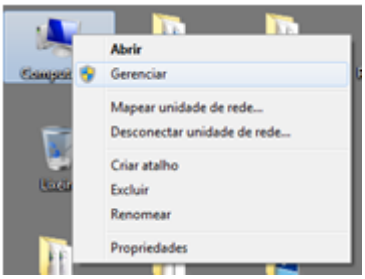
Figura 40 – Download do Aplicativo

Fonte: Print Screen elaborado pelo autor

Após o download, extraia o arquivo no seu computador.

Em primeiro lugar conecte o dispositivo, em seguida clique com o botão direito do mouse em ‘Computador’ e selecione ‘Gerenciar’.

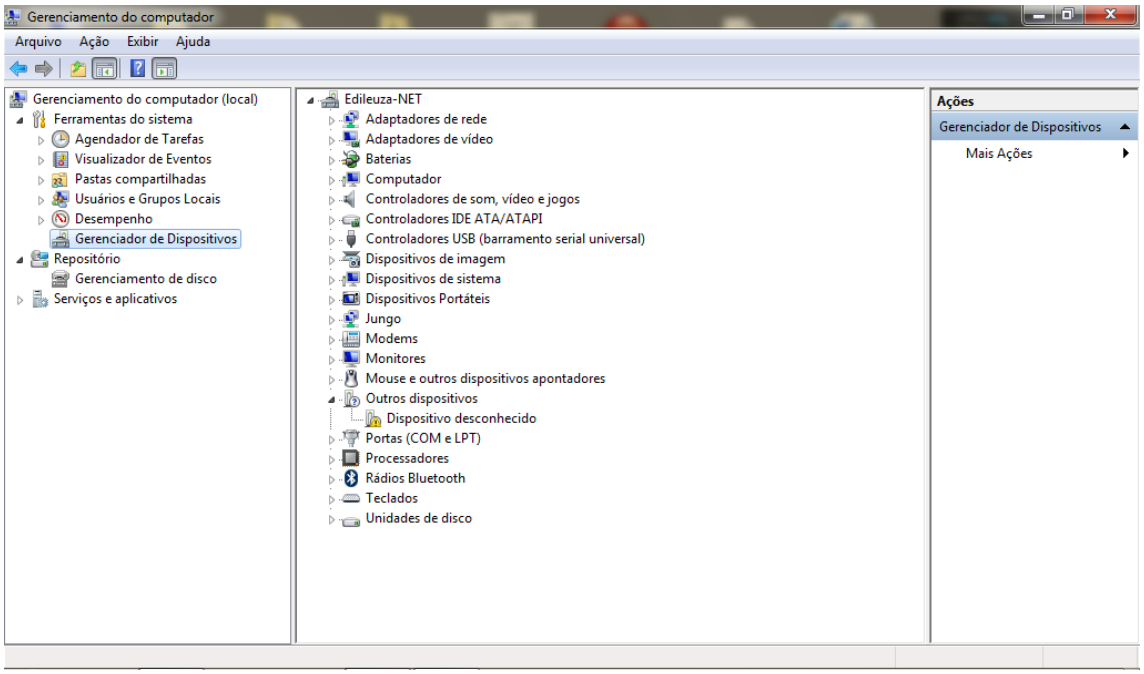
Figura 41 – Acessando gerenciamento do computador



Fonte: Print Screen elaborado pelo autor

Em seguida, selecione a opção ‘Gerenciador de dispositivos’.

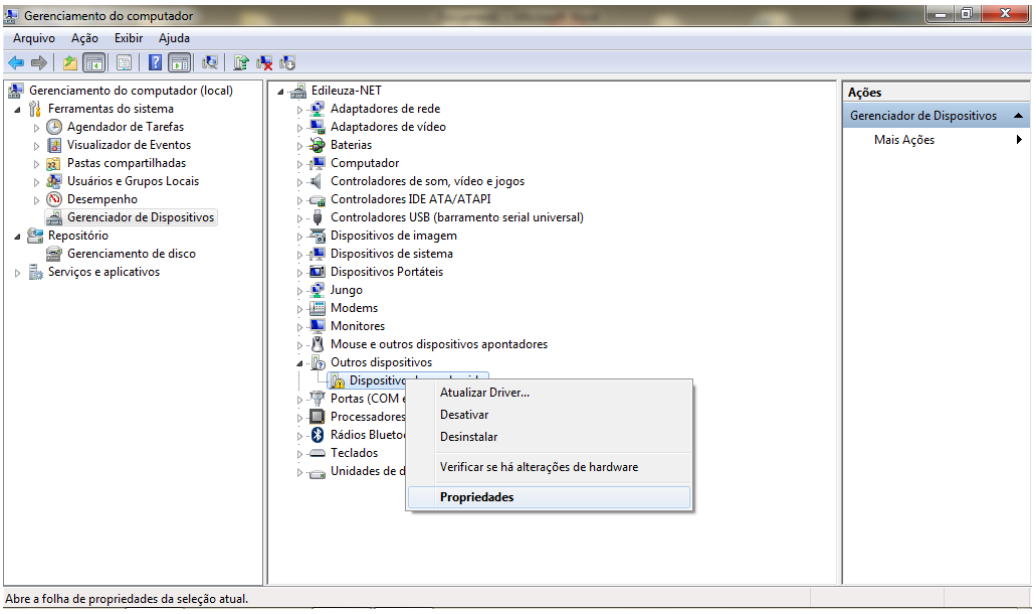
Figura 42 – Acessando gerenciador de dispositivos



Fonte: Print Screen elaborado pelo autor

Logo quando selecionamos o gerenciador, vemos um dispositivo com um sinal de exclamação, indicando que não foi reconhecido pelo Windows. Clique com o botão direito do mouse e selecione ‘Propriedades’.

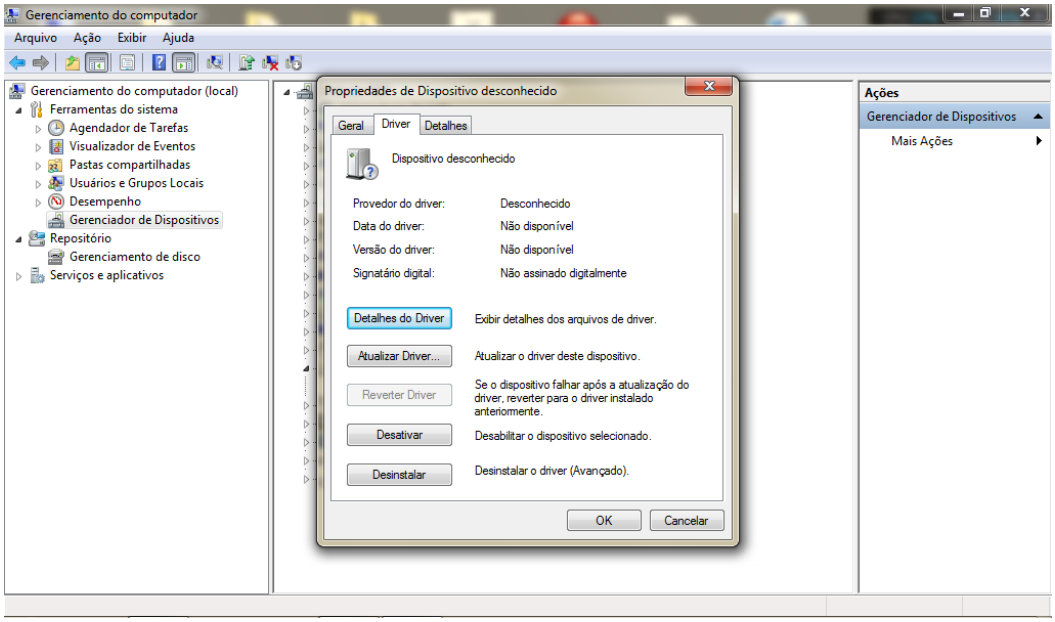
Figura 43 – Selecionando dispositivo



Fonte: Print Screen elaborado pelo autor

Será aberta uma janela com as opções do dispositivo, é necessário clicar na segunda guia (Driver), em seguida na opção ‘Atualizar Driver’.

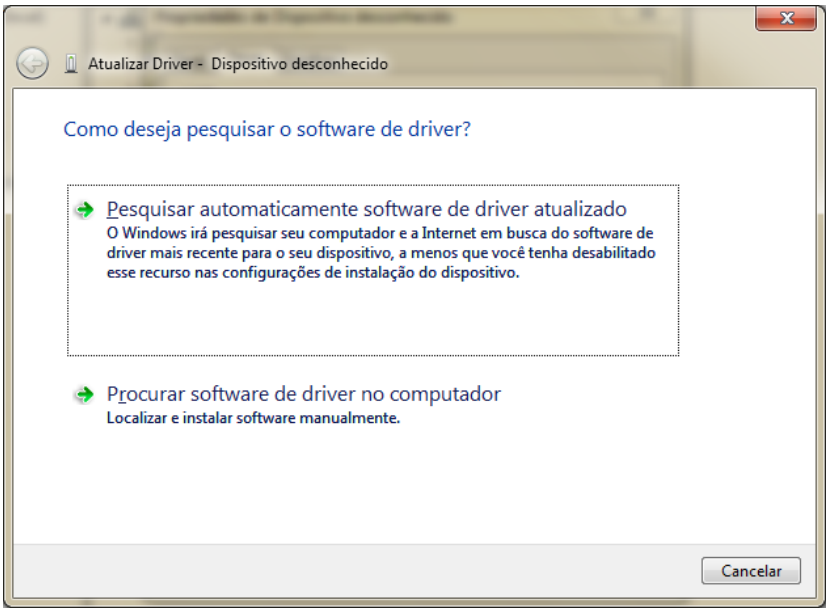
Figura 44 – Atualizando driver do dispositivo



Fonte: Print Screen elaborado pelo autor

Veremos a opção da imagem abaixo, devemos selecionar a opção ‘Procurar software de driver no computador’.

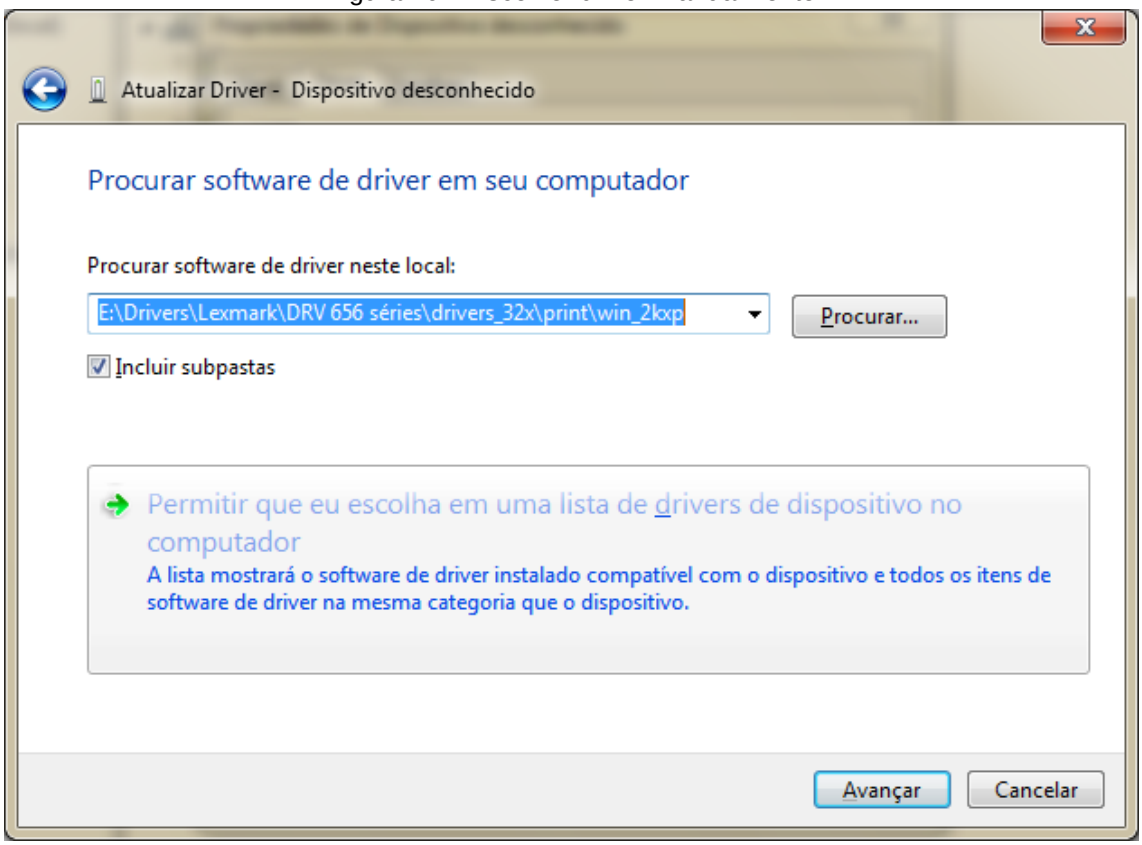
Figura 45 – Atualizando driver do dispositivo



Fonte: Print Screen elaborado pelo autor

Devemos selecionar ‘Permitir que eu escolha em uma lista de drivers de dispositivo no computador’.

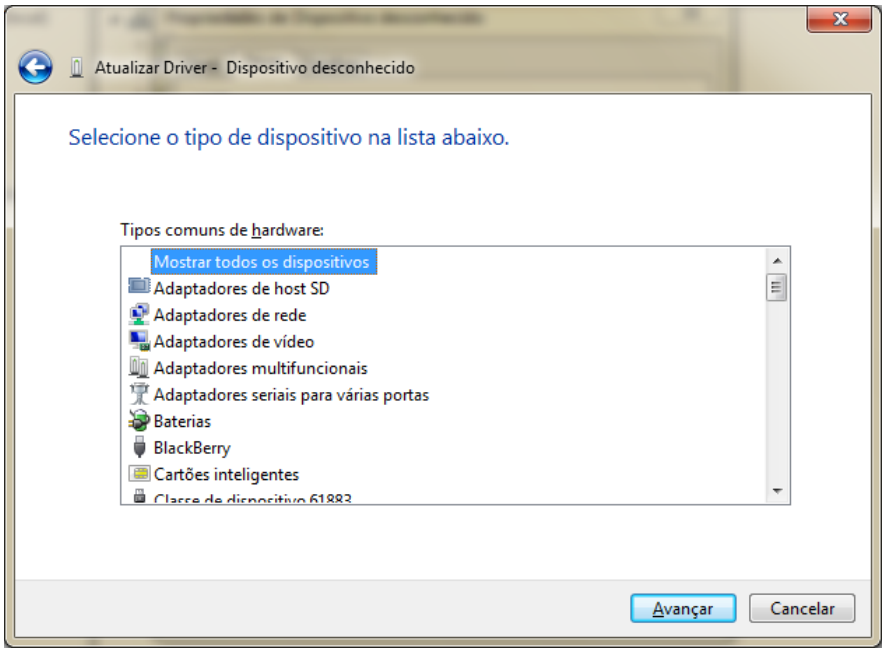
Figura 46 – Escolher driver manualmente



Fonte: Print Screen elaborado pelo autor

Clique em ‘Mostrar todos os dispositivos’.

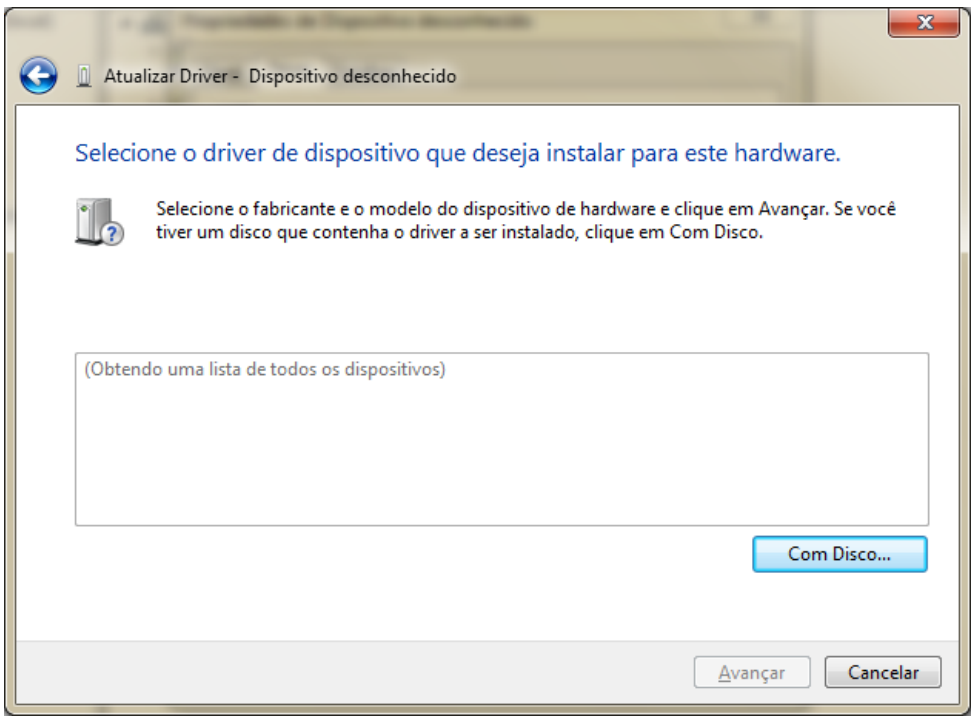
Figura 47 – Mostrar dispositivos



Fonte: Print Screen elaborado pelo autor

Selecione ‘Com Disco’.

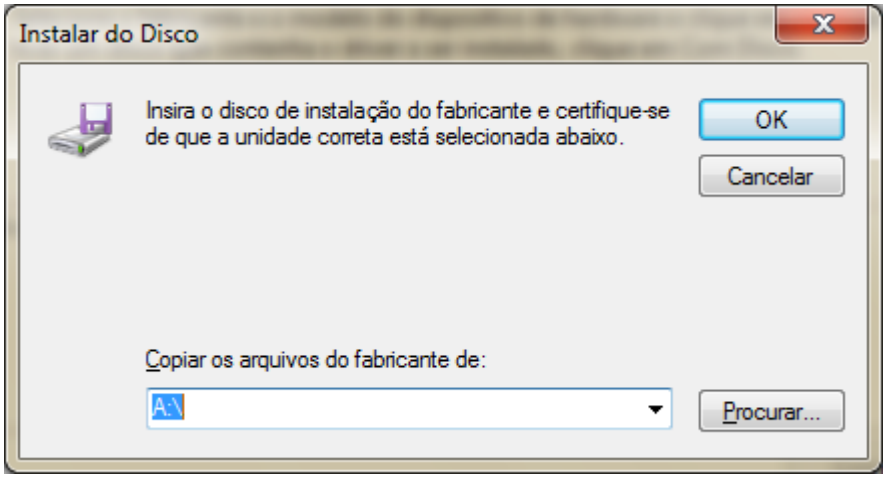
Figura 48 – Selecionar forma de instalação de driver



Fonte: Print Screen elaborado pelo autor

Clique em ‘Procurar’.

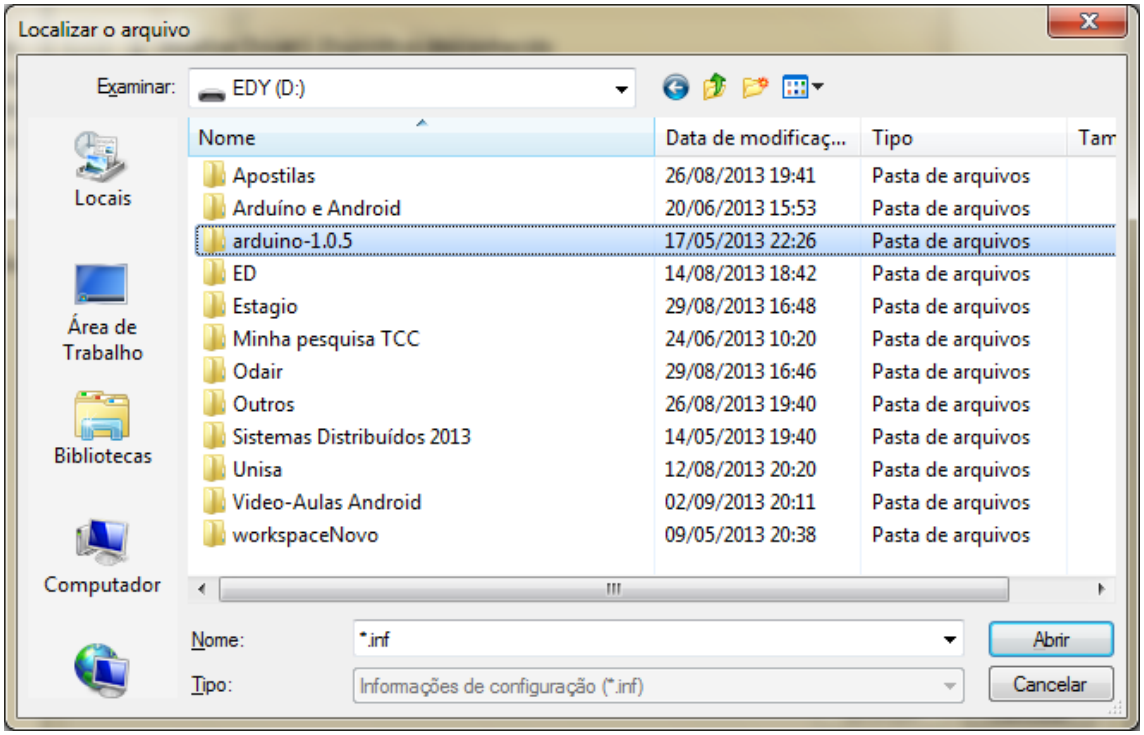
Figura 49 – Selecionar local de driver



Fonte: Print Screen elaborado pelo autor

Quando clicamos em procurar, é aberta a opção de localização dos drivers no Windows, no nosso caso, extraímos o software em um dispositivo removível.

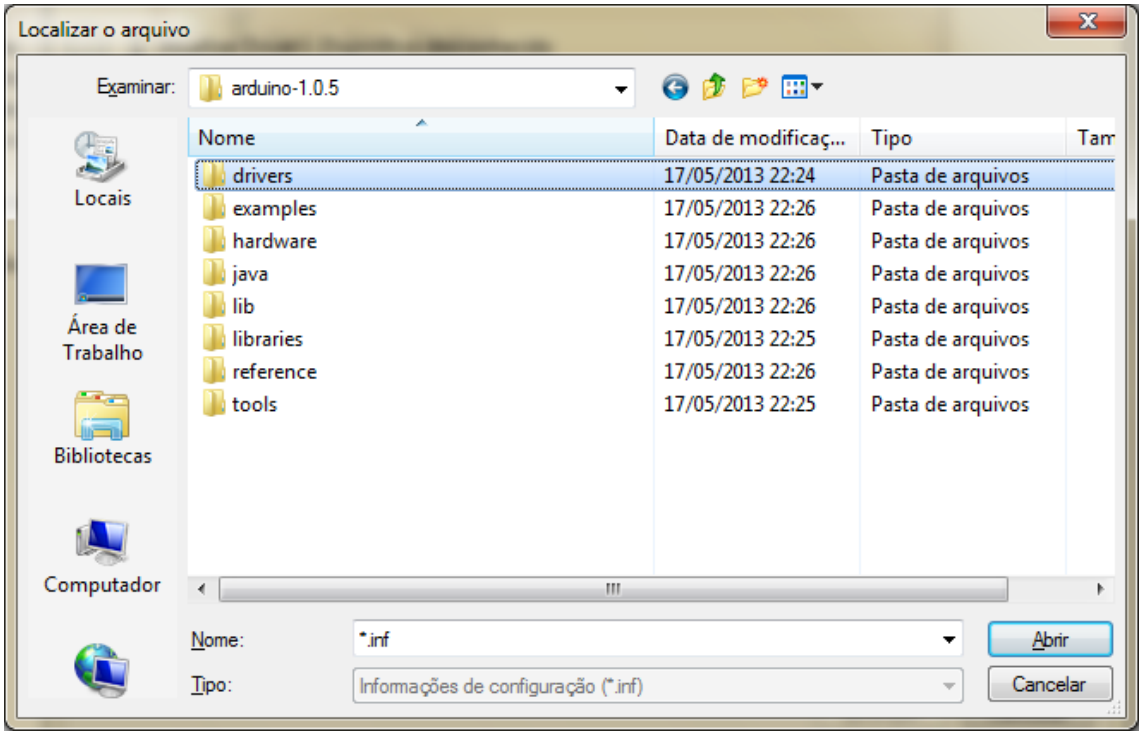
Figura 50 – Selecionar pasta do Arduino



Fonte: Print Screen elaborado pelo autor

Abra a pasta ‘drivers’.

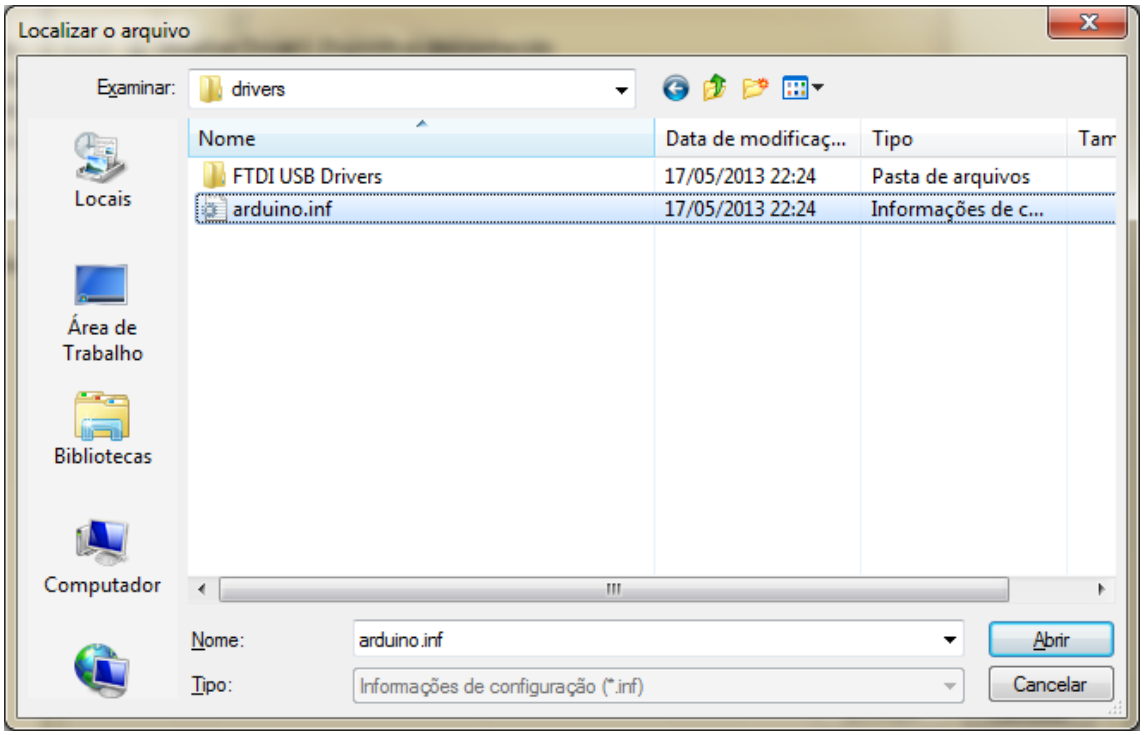
Figura 51 – Selecionar pasta driver



Fonte: Print Screen elaborado pelo autor

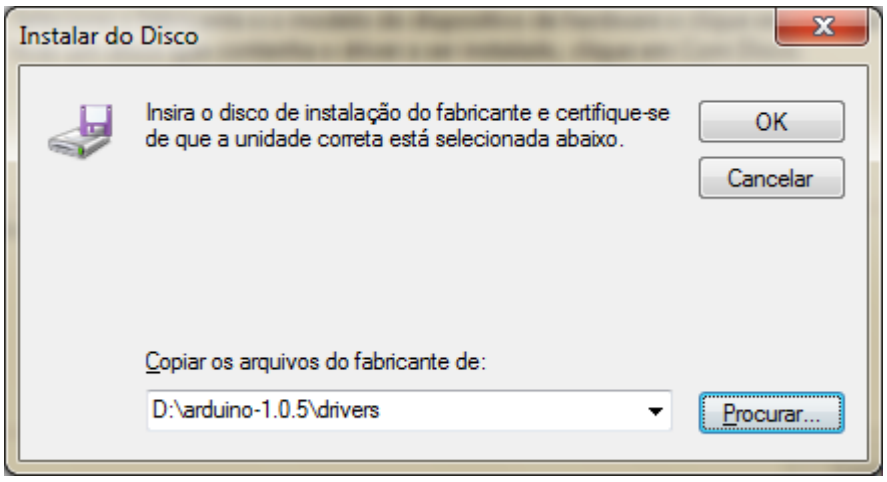
Clique duas vezes no arquivo abaixo, em seguida clique em ok.

Figura 52 – Selecionar driver



Fonte: Print Screen elaborado pelo autor

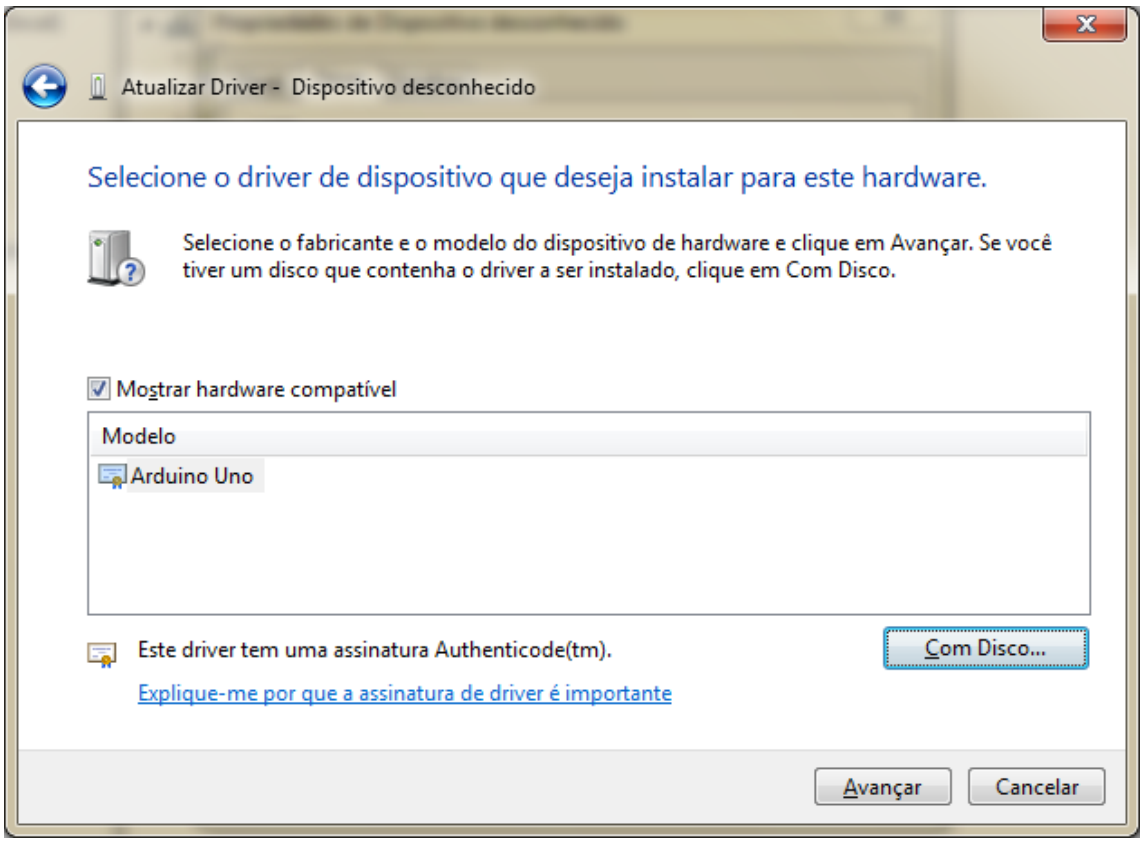
Figura 53 – Confirmar driver



Fonte: Print Screen elaborado pelo autor

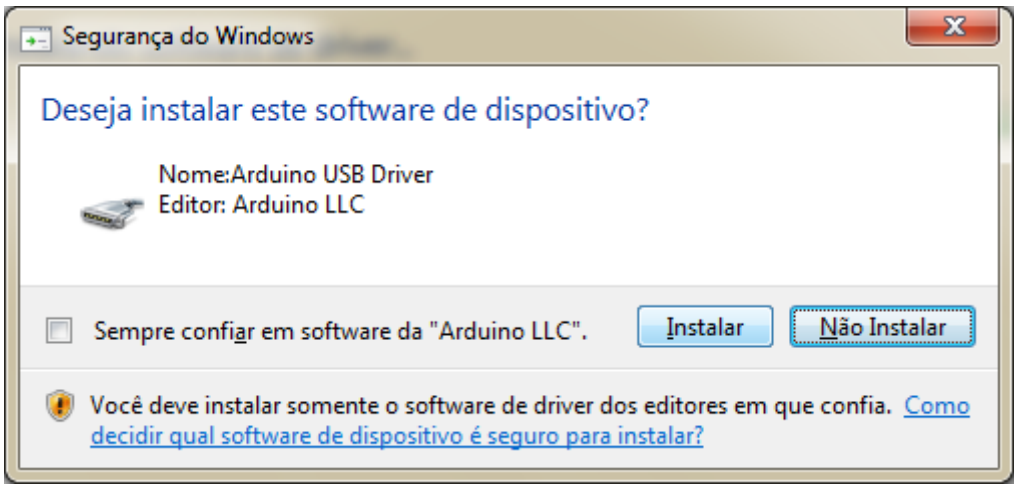
Avance e clique em ‘Instalar’.

Figura 54 – Confirmando driver



Fonte: Print Screen elaborado pelo autor

Figura 55 – Instalando driver

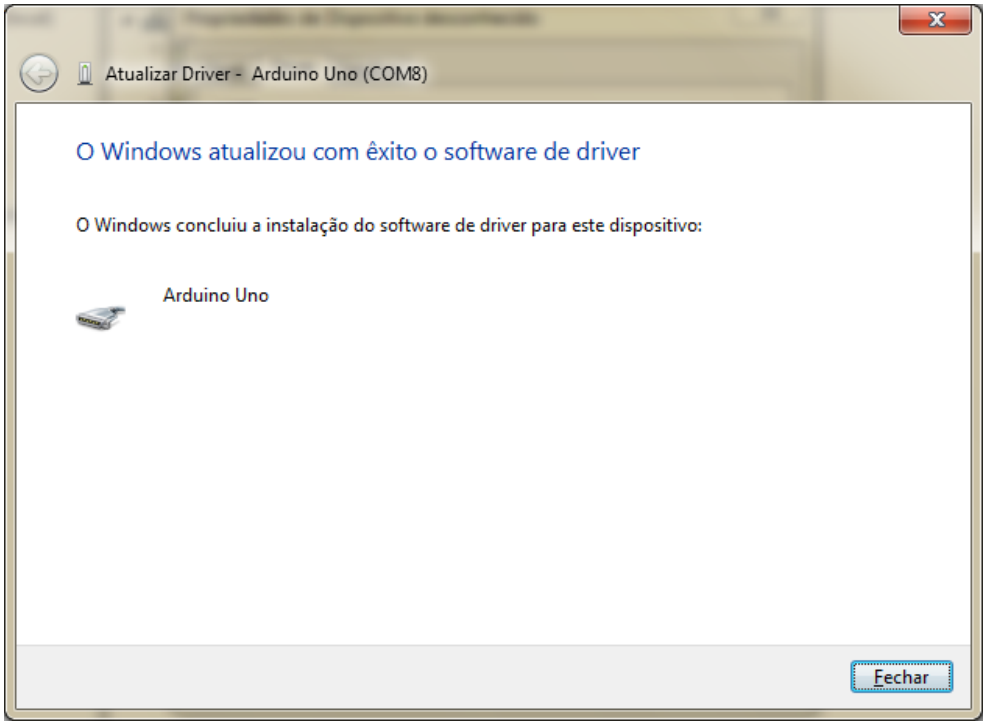


Fonte: Print Screen elaborado pelo autor

Quando o processo for finalizado, basta clicar em ‘fechar’.

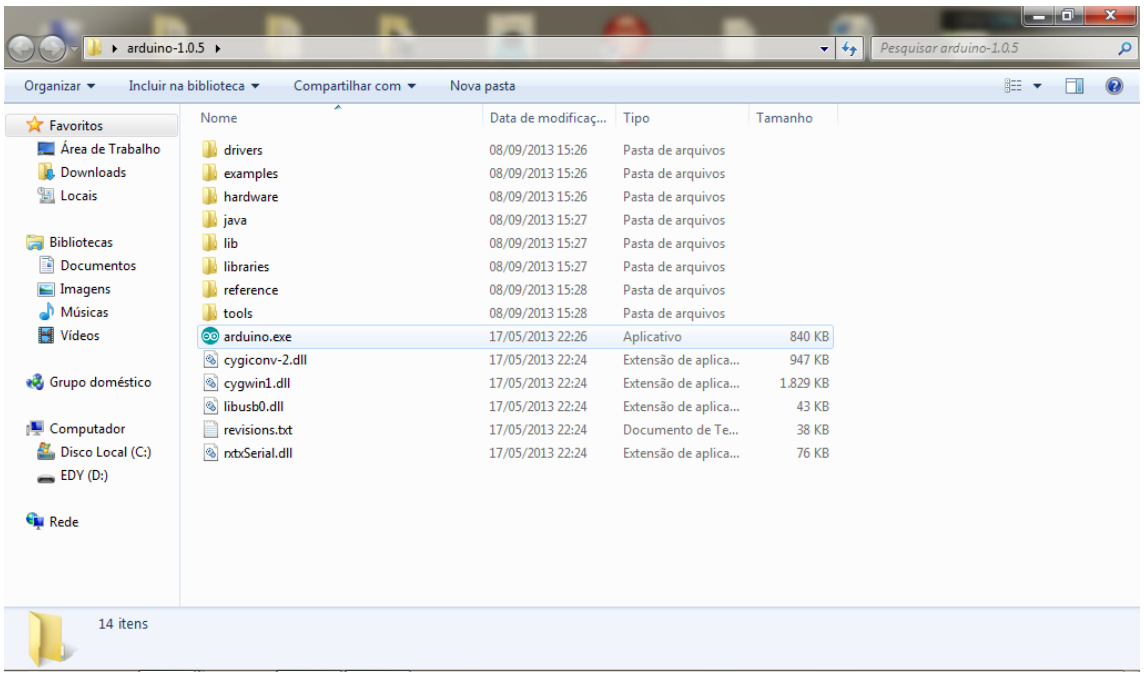
Pronto! O ambiente de desenvolvimento do Arduino já está instalado no seu computador. Basta abrir o aplicativo e começar a utilizar!

Figura 56 – Confirmação de instalação



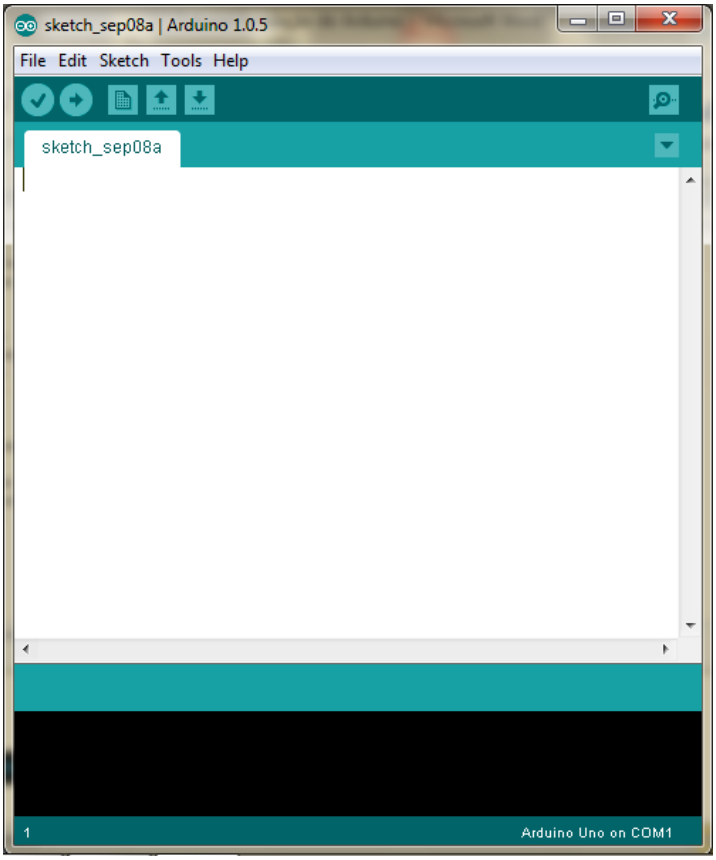
Fonte: Print Screen elaborado pelo autor

Figura 57 – Abrindo a IDE do Arduino



Fonte: Print Screen elaborado pelo autor

Figura 58 – Tela principal da IDE



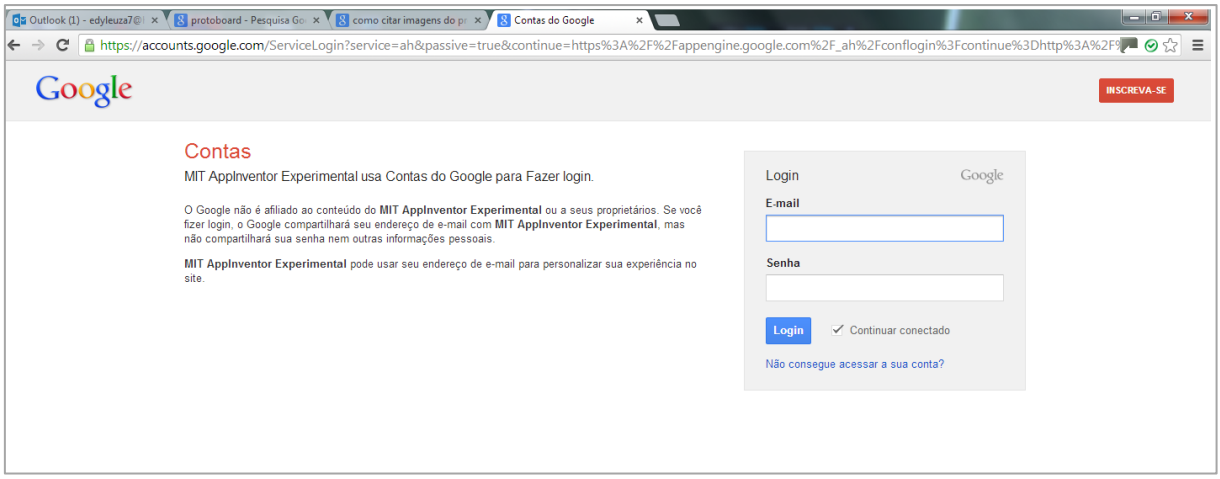
Fonte: Print Screen elaborado pelo autor

3.1.5 Desenvolvimento do aplicativo para os smartphone

Feito a instalação e logo após clicar em Start, você será direcionado à página com instruções de como usar o aplicativo pela primeira vez. É necessário ter acesso a internet e uma conta do G-mail, pois é através dela que se realiza o login para acessar os projetos que serão criados, caso esta não exista há um link que leva para a página de criação da conta do e-mail.

Com o e-mail criado, é preciso acessar a página http://beta.appinventor.mit.edu/ que irá direcionar para que seja inserido login e senha como mostra a figura abaixo.

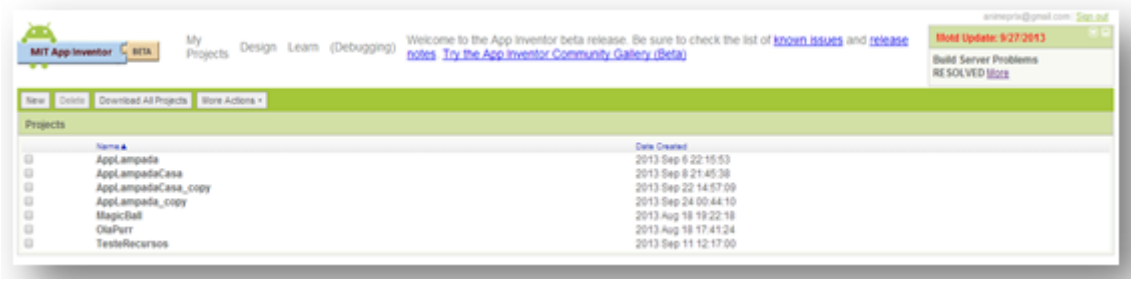
Figura 59 – Acesso à conta do Gmail



Fonte: Print da tela de acesso do gmail elaborada pelo autor

Ao entrar, já visualizamos o local onde ficarão os projetos criados.

Figura 60 – Lista de projetos criados



Fonte: Print da tela de projetos elaborada pelo autor

Utilizando o Aplicativo

No geral, pode-se dizer que a criação do projeto é dividida em três etapas: a escolha do nome do projeto (que é a primeira tela), o desenvolvimento do conteúdo e a ligação dos blocos para fazer o aplicativo funcionar, ou seja, a parte lógica. Mas antes é preciso conhecer a utilização de cada uma das partes.

A tela inicial, como dito anteriormente, é a parte da escolha do nome do projeto. Além disso, vemos outros recursos descritos na tabela abaixo.

Figura 61 – Visão geral dos recursos

Fonte: Print da tela da visão geral dos recursos elaborada pelo autor

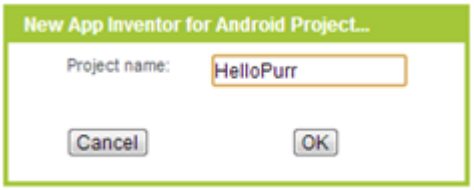
Tabela 2 – Lista de legendas de recursos

New	Botão para criar o projeto.
Delete	Ao selecionar um ou mais projetos, pode-se excluí-lo.
Download All Projects	Baixa todos os projetos criados para o computador.
More Actions	Faz o download ou upload de um projeto
Projects	Projetos listados em ordem alfabética *(name)*
Date Created	Data de criação. Os projetos podem também serem listados por ordem de criação.
My Projects	Mostra a tela inicial onde estão listados os projetos.
Design	Abre o último projeto visto
Learn	Leva à guia de Tutoriais
(Debbuging)	Mostra mensagens de desenvolvimento do projeto.

Fonte: Print da tela da lista de legendas elaborada pelo autor

No início, o principal item dessas descrições é o “New”. Ao acioná-lo, aparece a seguinte tela:

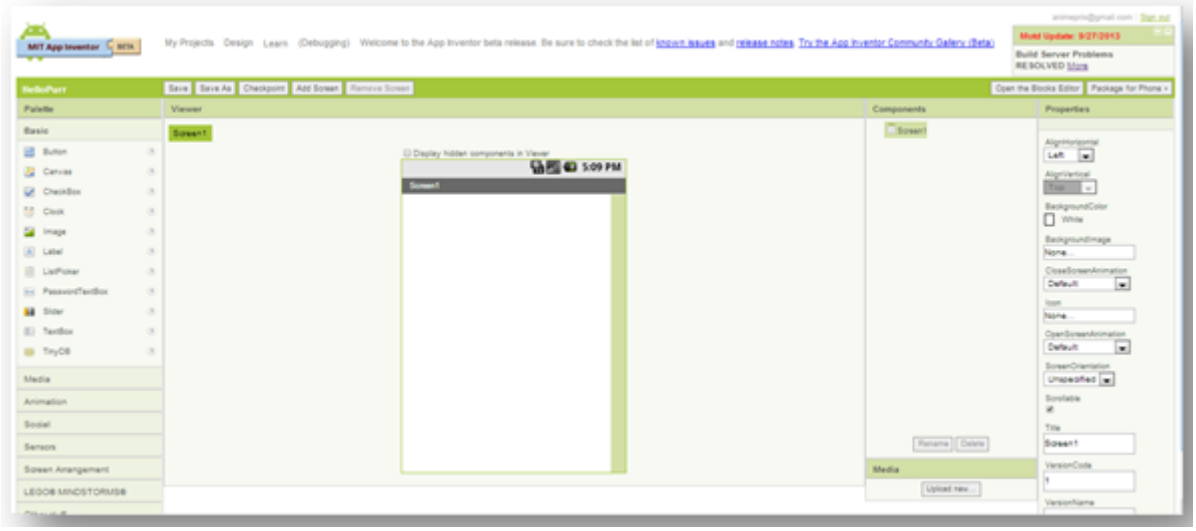
Figura 62 – Criando um novo projeto



Fonte: Print da tela de novo projeto elaborada pelo autor

O nome do projeto deve começar com uma letra. Pode conter letras, números e *underline.* Ao clicar em “OK” automaticamente é direcionada a tela de desenvolvimento do projeto.

Figura 63 – Ambiente de desenvolvimento do novo projeto



Fonte: Print da tela do ambiente de desenvolvimento elaborada pelo autor

Nessa página, chamada *Designer,* são inseridos os objetos que darão “vida” ao aplicativo. Também sem a necessidade de inserir linhas de comando, apenas precisa escolher o item a ser colocado e arrastá-lo para o simulador da tela do celular *(Viewer → Screen1)*. Como mostra a figura abaixo.

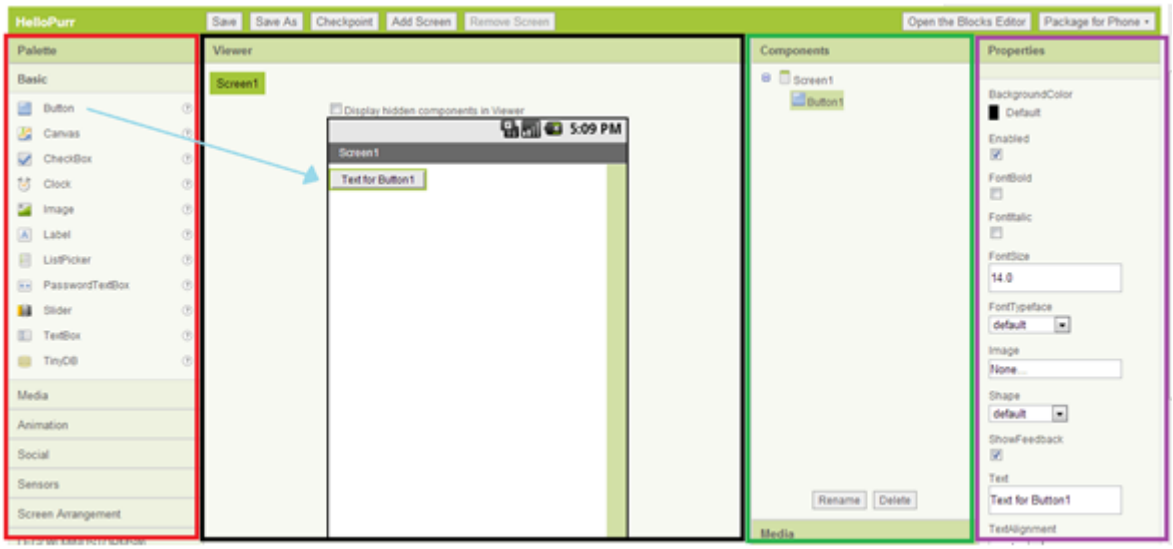
Tabela 3 – Legenda de Designer

Palette	Todos os objetos que podem ser utilizados se encontram nesta guia, que é dividida em: *Basic, Media, Animation, Social, Sensors, Screen arregiment,Lego® Mindstorms®, Other stuff, Not ready for prime time*. Cada item tem uma função específica como botões, caixas de texto, conexão com web, entre outros.
Viewer	Simula a tela do celular
Components	Todos os itens que são arrastados para a "tela", mesmo os que não ficam visíveis, como música ou bluetooth, por exemplo, ficam alocados neste elemento. Assim fica fácil ter uma visão geral dos objetos utilizados, bem como se conectam a outros objetos. Também é possível alterar o nome ou excluir um objeto.
Properties	É a parte de *layout*, onde são definidos os tamanhos, cores, conteúdos, tipos de letra, etc.

Fonte: Print da tela de legenda de Designer elaborada pelo autor

A seta azul mostra que o objeto foi arrastado para a tela do celular

Figura 64 – Utilizando ferramentas



Fonte: Print da tela de uso de ferramentas elaborada pelo autor

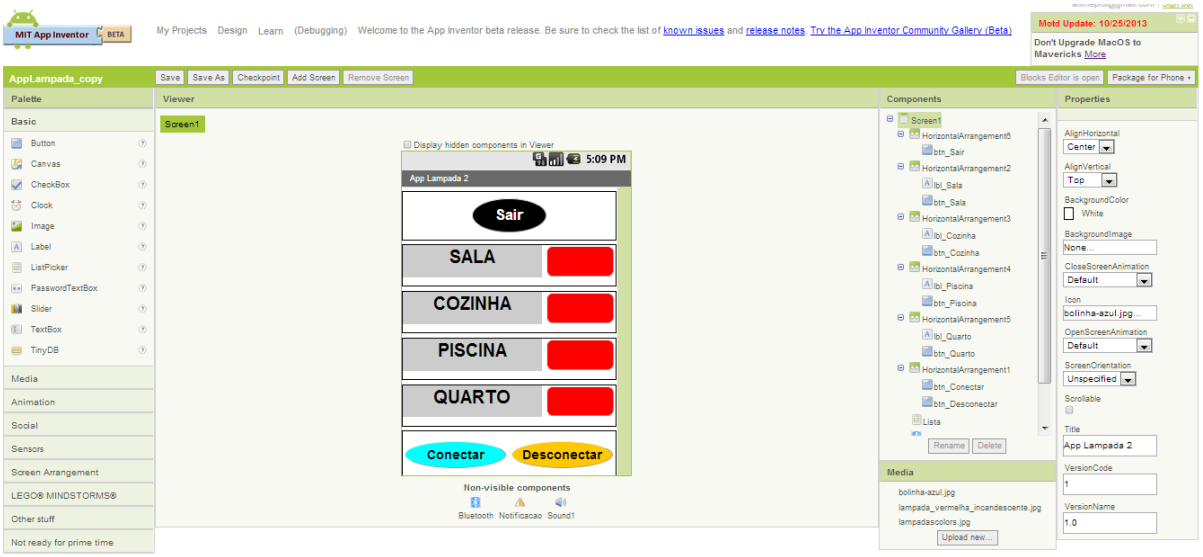
Partindo deste princípio, pode-se fazer muitas coisas com os objetos que estão disponíveis para a criação de aplicativos, iremos dar início ao nosso desenvolvimento na próxima etapa, após este processo de elaboração do trabalho, poderemos observar a criação dos aplicativos utilizados.

4. RESULTADOS

4.1 Elaboração do Aplicativo

Nesta primeira etapa iremos acompanhar todo processo de desenvolvimento definitivo do software que irá “rodar” nos celulares que demostraremos logo mais adiante, no decorrer deste trabalho, bem como configurações e ajustes que foram necessários para o sucesso à aplicação.

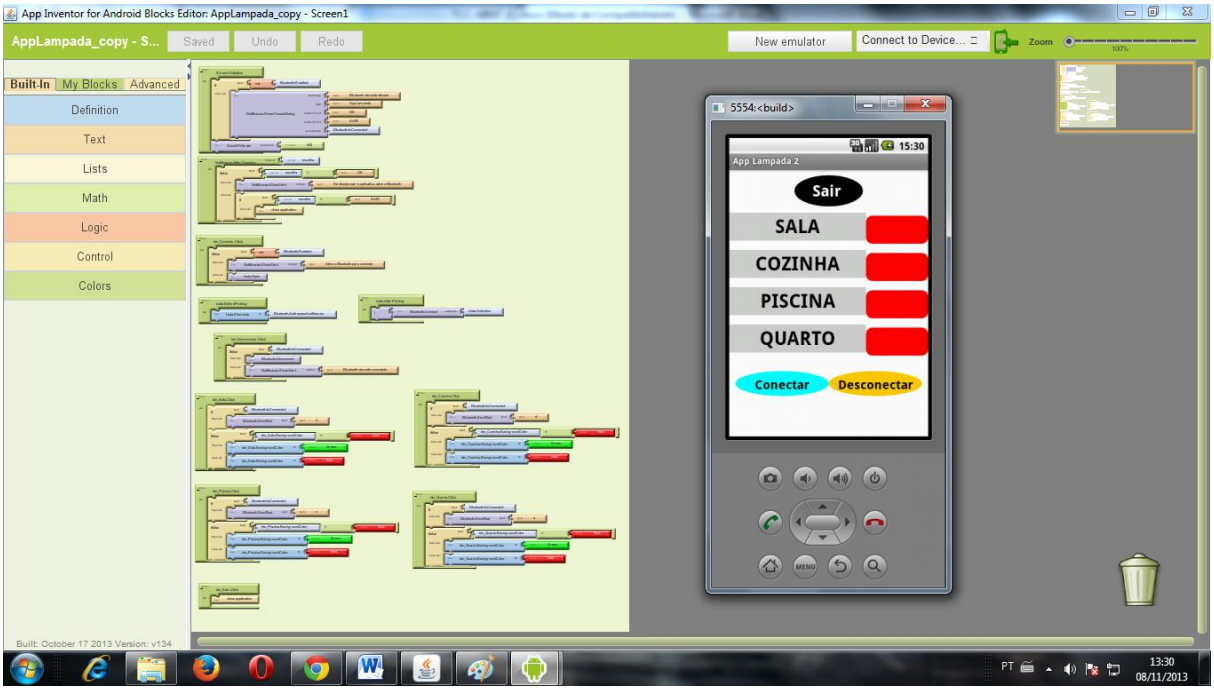
Figura 65 – Tela de Designer



Fonte: Print MIT App Inventor- Tela Designer (elaborada pelo autor)

Tela inicial onde desenvolvemos o *layout* do programa. Utilizando “Buttons”, “Images”, itens de conexão e etc. Define-se os nomes dos objetos e dá as características aos itens como tamanho da imagem, cor do botão, plano de fundo, entre outros.

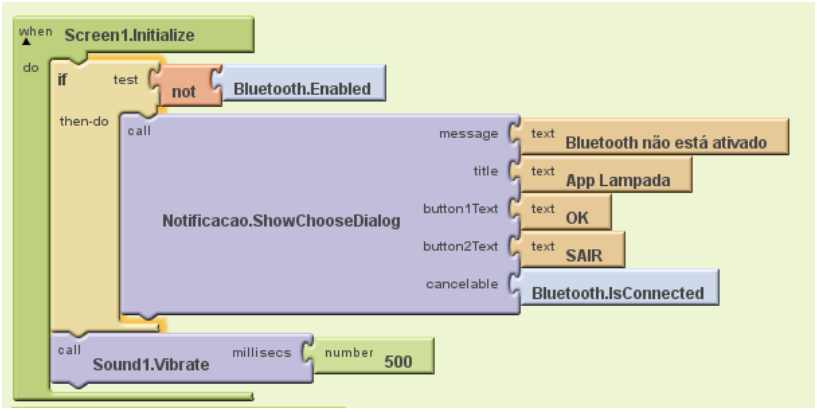
Figura 66 – Blocos de comandos



Fonte: Print da tela do MIT App Inventor (elaborada pelo autor)

Visão do Editor de Blocos onde é desenvolvida a parte lógica do programa. Monta-se a estrutura do programa (minimizado para uma melhor visualização) utilizando os “blocos” de comando para desenvolver os eventos e métodos que farão com que o programa funcione. Ao lado o emulador do celular com o programa ativo dá ao usuário uma experiência mais interativa (apesar que esses comandos com o bluetooth não puderam ser testados no emulador, pois ele não realiza a função de conexão).

Figura 67 – Editor de Blocos



Fonte: MIT App Inventor – Editor de Blocos (elaborada pelo autor)

O *"**Screen1.Initialize**"* faz a verificação da conexão do bluetooth do aparelho. Basicamente ele pergunta **se** o bluetooth está ou não conectado.

Se ele vê que não há conexão, mostra-se uma notificação com a seguinte mensagem "O Bluetooth não está ativado", título "App Lampada" dá as opções "OK" e "SAIR".

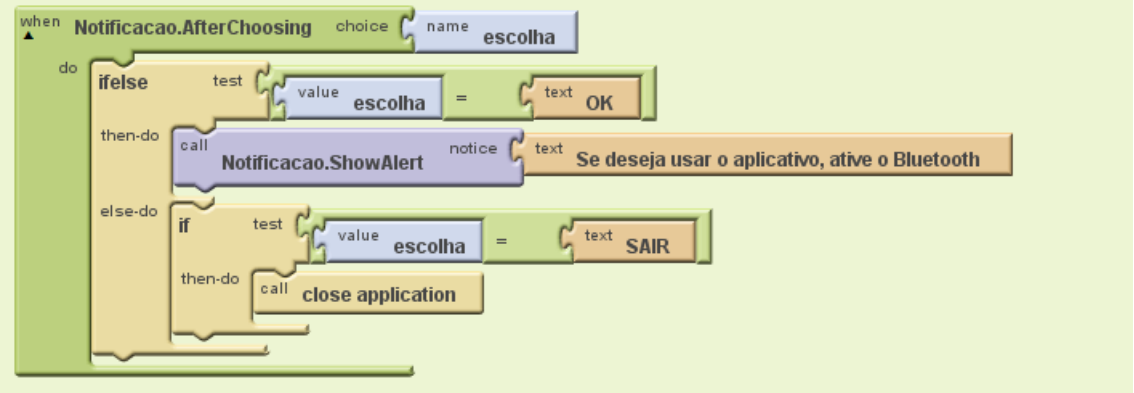
Se ele enxerga que o Bluetooth está ativo *(Bluetooth.IsConnected)*, não é mostrada nenhuma notificação. Quando se inicia o aplicativo o aparelho vibra *(Sound1.Vibrate)*.

Figura 68 – Emulador do App Inventor

Fonte: Print Screen elaborado pelo autor

Essa é a imagem da notificação no emulador.

Figura 69 – Notificação no emulador



Fonte: Print Screen elaborado pelo autor

No evento ***"Notificacao.AfterChoosing"*** o valor "escolha" vai definir o que acontecerá na sequência. Se a escolha for igual a "OK", ao pressioná-lo o aplicativo fica "aberto", mas não funciona. Deve-se procurar no dispositivo a opção para acionar o bluetooth. Já o botão SAIR fecha o aplicativo.

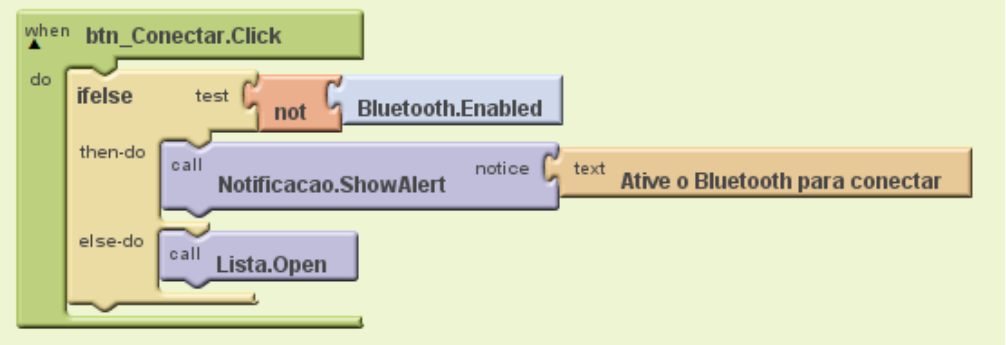
O aparecimento da notificação "Se deseja usar o aplicativo, ative o Bluetooth", só será mostrada se for detectado que o bluetooth não está ativo no aparelho, caso contrário é ignorado.

Visão da notificação no emulador (em destaque).

Figura 70 - Emulador

Fonte: Print Screen elaborado pelo autor

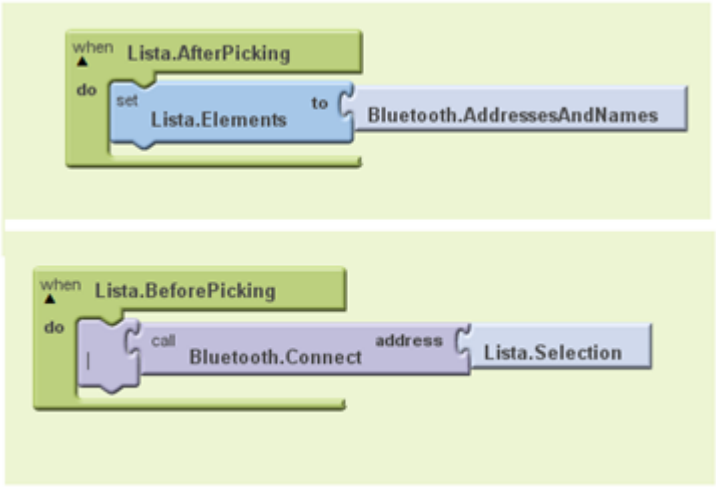
Figura 71 – Botão Conectar



Fonte: Print Screen elaborado pelo autor

Ao clicar o botão conectar também há uma verificação para informar se a conexão está estabelecida. O teste é se o bluetooth não está conectado mostre a mensagem “Para usar o aplicativo, ative o Bluetooth”, senão chame a lista onde será conectado o dispositivo que fará a conexão com o aparelho celular. No caso o item que será conectado é o Módulo Bluetooth que está ligado ao Arduino. O módulo é o elo entre o Smartphone e a placa Arduino.

Figura 72 – Estabelecendo conexão

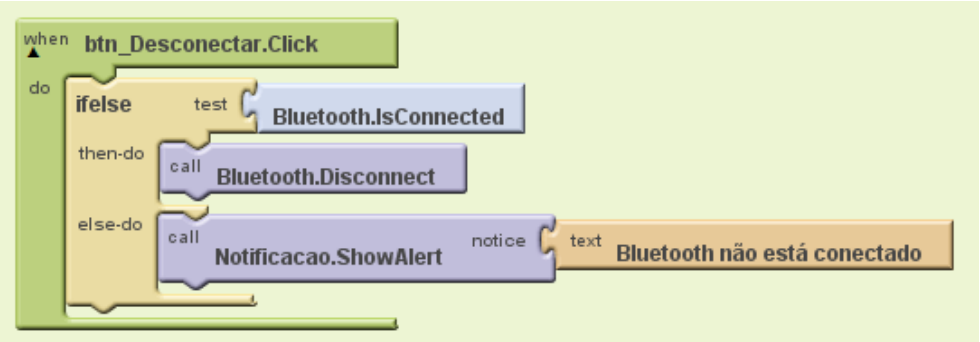


Fonte: Print Screen elaborado pelo autor

Lista.AfterPicking é aqui que ele entra na lista onde estão os dispositivos com bluetooth que podem ser conectados *(Bluetooth.AddressesAndNames)*

Lista.BeforePicking, “aceita” o endereço do dispositivo desejado formalizando a conexão.

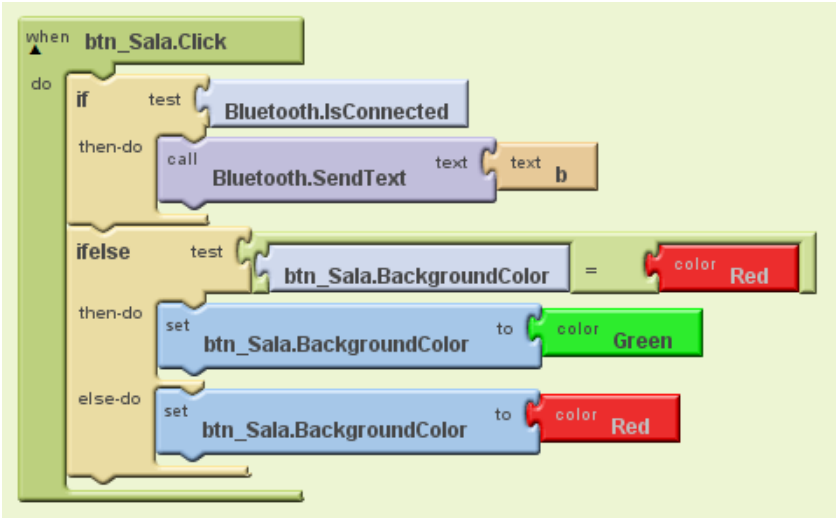
Figura 73 – Desconectando Bluetooth



Fonte: Print Screen elaborado pelo autor

O botão **btn_Desconectar.Click**, ao ser clicado, faz a seguinte verificação: testa se o bluetooth está conectado, se não estiver, desconecte. Se ele não estiver conectado mostre a mensagem: “O Bluetooth não está conectado”.

Figura 74 – Botão de acionamento da Sala



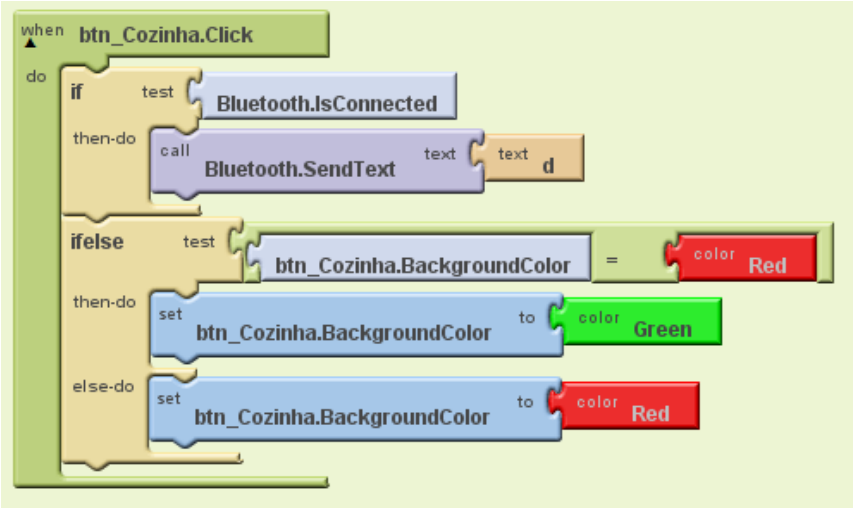
Fonte: Print Screen elaborado pelo autor

btn_Sala.Click = Botão que liga a lâmpada da sala.

Quando clicado testa se o bluetooth está conectado e envia para o módulo ligado ao Arduino o comando “b” que será lido como o botão de ação para ligar e desligar a Lâmpada (ou qualquer outro equipamento que estiver ligado na porta estipulada)

O ifelse fará a alteração das cores. Inicializa com o botão “Red” (vermelho) e quando clicado muda para a cor “Green” (verde), clicado novamente volta para vermelho.

Figura 75 – Botão de acionamento da Cozinha



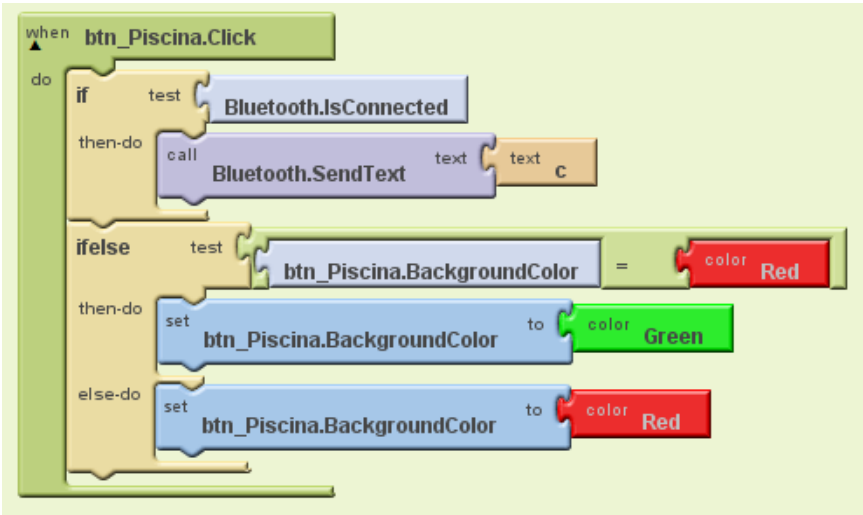
Fonte: Print Screen elaborado pelo autor

btn_Cozinha.Click = Botão que liga a lâmpada da cozinha.

Quando clicado testa se o bluetooth está conectado e envia para o módulo ligado ao Arduino o comando “d” que será lido como o botão de ação para ligar e desligar a Lâmpada (ou qualquer outro equipamento que estiver ligado na porta estipulada)

O ifelse fará a alteração das cores. Inicializa com o botão “Red” (vermelho) e quando clicado muda para a cor “Green” (verde), clicado novamente volta para vermelho.

Figura 76 – Botão de acionamento da Piscina e Lâmpada de 127v.



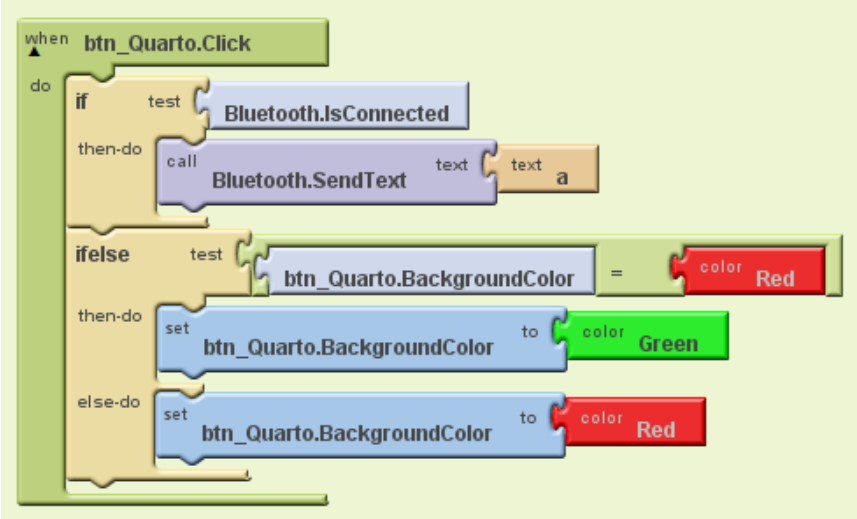
Fonte: Print Screen elaborado pelo autor

btn_Piscina.Click = Botão que liga a lâmpada da piscina (representada pelos leds) e a lâmpada de 127v, que pode ser acionada tanto pelo interruptor quanto pelo aplicativo.

Quando clicado testa se o bluetooth está conectado e envia para o módulo ligado ao Arduino o comando “c” que será lido como o botão de ação para ligar e desligar a Lâmpada (ou qualquer outro equipamento que estiver ligado na porta estipulada)

O ifelse fará a alteração das cores. Inicializa com o botão “Red” (vermelho) e quando clicado muda para a cor “Green” (verde), clicado novamente volta para vermelho.

Figura 77 – Botão de acionamento do Quarto



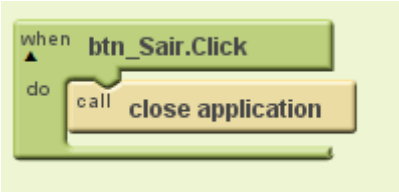
Fonte: Print Screen elaborado pelo autor

btn_Quarto.Click = Botão que liga a lâmpada do quarto.

Quando clicado testa se o bluetooth está conectado e envia para o módulo ligado ao Arduino o comando “a” que será lido como o botão de ação para ligar e desligar a Lâmpada (ou qualquer outro equipamento que estiver ligado na porta estipulada)

O ifelse fará a alteração das cores. Inicializa com o botão “Red” (vermelho) e quando clicado muda para a cor “Green” (verde), clicado novamente volta para vermelho.

Figura 78 – Encerramento do aplicativo



Fonte: Print Screen elaborado pelo autor

O botão **btn_Sair.Click** solicita o encerramento do aplicativo e desconecta do módulo bluetooth.

4.2 Transferência de dados do App Inventor

Aqui poderemos acompanhar como foi feita a configuração dos três aparelhos usados neste projeto para transferência de dados do aplicativo do App Inventor. Devemos ressaltar que para o perfeito funcionamento dos dispositivos usados percebemos que era necessária uma determinada configuração específica para cada um deles, é o que iremos ver logo abaixo:

4.2.1 Motorola Defy MB525

O primeiro aparelho que foi utilizado para realizar os testes do projeto. Versão do Android 2.2.1.

Não foi necessário modificar as configurações do aparelho, apenas ter baixado o MIT AICompanion e um leitor de *barcode*. Neste aparelho foi feito o download do aplicativo QR-code Scaner.

O aplicativo AppLampada_copy – nome dado a segunda versão do aplicativo que criamos – foi baixado tanto via WiFi, quanto passado para o aparelho através do cabo USB obtendo êxito nos dois meios.

Foi direcionado para o cartão de memória do aparelho e instalado através da requisição “Instalador do pacote”, criando um ícone no menu do celular pronto para o uso.

Figura 79 – Aparelho celular Mortorola Defy MB525[2]

Fonte: Elaborado pelo autor

4.2.2 Galaxy SII GT-I9100

Este aparelho possui características bem diferentes dos outros dois usados também no projeto possui processador de 1GHz, 16GB de memória interna e android 4.0.

Por ser o mais “parrudo” da turma não deu nenhum tipo de trabalho, logo que foi conectado ao computador, seu sistema atualizou automaticamente o que era preciso e transferiu os dados sem maiores problemas, tanto por via USB como por leitura de barcode.

[2] Na última foto da imagem, o botão “Lâmpada” foi substituído para “Quarto”, presente no aplicativo atual.

Figura 80 – Celular Galaxy SII

Fonte: Imagem elaborada pelo autor

4.2.3 LG L3 E405f

Este aparelho em especial foi o que deu um pouco mais de trabalho para configurar, ele possui versão do Android 2.3.

Foi necessária a habilitação depuração de USB, para se habilitar esta opção é preciso que se vá até, as configurações do dispositivo – aplicativos – desenvolvimento e habilitar esta opção. Lembre-se que o aparelho deverá está conectado previamente ao seu computador e ao escolher o modo de conexão, escolha a opção por vínculo USB.

Dessa forma prossiga com a transferência de dados como foi efetuado nos demais, o aplicativo poderá ser acessado dentro do local padrão, dos aplicativos baixados no celular com esta plataforma.

Figura 81 – Celular LG L3

Fonte: Imagem elaborada pelo autor

4.3 Desenvolvendo a Programação

Antes de começarmos a desenvolver, assim como na linguagem C e C++, podemos incluir bibliotecas que vamos precisar durante o desenvolvimento da aplicação. Chamamos de Diretivas de Compilação, que são comandos que fazem o compilador executar determinadas funções antes mesmo de iniciar a compilação do programa em si.

Utilizamos assim o *#include* para incluirmos as bibliotecas que vamos utilizar no decorrer do desenvolvimento.

Em nosso caso, incluímos a biblioteca *SoftwareSerial.h*, que nos dá suporte á comunicação Serial do Arduino. Abaixo a sintaxe da nossa inclusão da biblioteca no código:

#include <SoftwareSerial.h>

A comunicação Serial permite que o Arduino possa se comunicar com outros dispositivos, tais como Computador ou Celular. As placas Arduino possuem, no mínimo, uma porta serial conhecida como UART ou USART. Essa comunicação acontece através dos pinos digitais RX: 0 e TX : 1.

Podemos definir alguns nomes para identificarmos os pinos digitais da placa do Arduino. São 14 pinos, numerados de 0 a 13 que se encontram na placa em duas barras contendo 8 pinos cada. Nesses pinos, podemos ligar nossos Led's para as iluminações de cada região de nossa casa automatizada.

Dessa forma, definimos nomenclaturas para identificarmos esses pinos durante a compilação do código. Utilizamos o *#define* para isso.

Abaixo uma imagem contendo a linha de código do nosso programa que define os nomes aos seus respectivos pinos:

Figura 82 – Definindo nomes para os Pinos Digitais

```
#define RxD 6                 // Define pino 6 como entrada Rx para comunicação com módulo Bluetooth
#define TxD 7                 // Define pino 7 como Saída Tx para comunicação com módulo Bluetooth
#define QUARTO 13             // PINO DIGITAL 13
#define SALA 3                // PINO IDIGITAL 3
#define PISCINA 4             // PINO DIGITAL 4
#define COZINHA 5             // PINO DIGITAL 5
#define INTERRUPTOR 8         // PINO DIGITAL 8
```

Fonte: Print da tela de desenvolvimento do IDE do Arduino

No nosso código, renomeamos os pinos conforme os ambientes em nossa maquete. A numeração é justamente os números dos pinos na placa. Dessa forma conseguimos manipular esses ambientes no código pelos nomes que definimos.

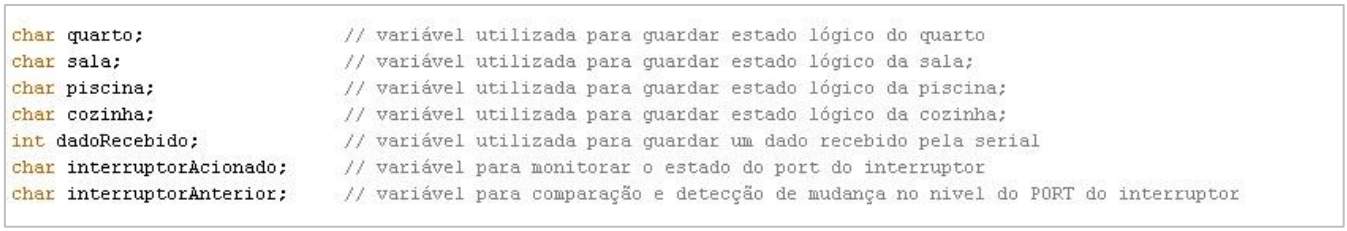
Em seguida, declaramos as variáveis que vamos utilizar no decorrer do código. É necessário que essa variável possua um tipo de dado. Existem diversos tipos de dados que possamos utilizar no Arduino. Segue abaixo os principais tipos de dados:

- Boolean: Verdadeiro ou Falso
- Char: Caracter
- Byte: Armazena um número de 8 bits
- Int: Armazenamento de números inteiros
- Long: Armazenamento numérico capaz de armazenar até 32 bits.
- Float: Armazenamento numérico com Ponto Flutuante. Números com ponto decimal.

- Double: Armazenamento numérico com Ponto Flutuante maior que o Float. Mais preciso em diferentes cenários.
- String: Array de caracteres (char)
- Array: Uma espécie de coleção de variáveis, com um índice numérico representando sua posição na coleção.

No nosso caso, declaramos algumas variáveis para utilizarmos no código. Assim conseguimos, por exemplo, ligar ou desligar um dos leds, verificar se os mesmos estão ligados ou desligados, gravar informações de estados anteriores dos leds, entre outras funções. Veja na figura abaixo:

Figura 83 – Declarando variáveis



```
char quarto;                // variável utilizada para guardar estado lógico do quarto
char sala;                  // variável utilizada para guardar estado lógico da sala;
char piscina;               // variável utilizada para guardar estado lógico da piscina;
char cozinha;               // variável utilizada para guardar estado lógico da cozinha;
int dadoRecebido;           // variável utilizada para guardar um dado recebido pela serial
char interruptorAcionado;   // variável para monitorar o estado do port do interruptor
char interruptorAnterior;   // variável para comparação e detecção de mudança no nivel do PORT do interruptor
```

Fonte: Print da tela de desenvolvimento do IDE do Arduino

Declaramos também uma variável capaz de manipularmos as funções do Software Serial. Nesse caso declaramos o *blueToothSerial.* Segue abaixo o exemplo da declaração:

SoftwareSerial blueToothSerial (RxD, TxD);

Setup do Arduino

O Setup é executado somente uma única vez. Ele é executado após a placa Arduino ser ligada ou reiniciada. Nela você pode configurar os dados dos pinos, declarar bibliotecas, manipular dados iniciais das variáveis, entre outras funções. Abaixo, o Setup de nosso projeto:

Figura 84 – Declarando Setup

```
void setup()
{
  Serial.begin(9600);            // Inicializa comunicação serial com Baud Rate em 9600 bps
  pinMode(RxD, INPUT);           // configura pino RxD do arduino como entrada digital
  pinMode(TxD, OUTPUT);          // configura pino TxD do arduino como saída digital
  setupBlueToothConnection();    // Chama rotina para configurar o módulo bluetooth
  pinMode (QUARTO, OUTPUT);      // define QUARTO como saida
  pinMode (SALA, OUTPUT);        // define SALA como saída
  pinMode (PISCINA, OUTPUT);     // define PISCINA como saida
  pinMode (COZINHA, OUTPUT);     // define COZINHA como saida
  pinMode (INTERRUPTOR, INPUT);  // define INTERRUPTOR como entrada
  interruptorAcionado = digitalRead(INTERRUPTOR);      // aqui faz-se uma verificação do estado do interruptor
  interruptorAnterior = interruptorAcionado;           // iguala variável para posterior comparação
}
```

Fonte: Print da tela de desenvolvimento do IDE do Arduino

No nosso caso acima, temos algumas novidades. O Serial.begin(int Velocidade) está inicializando o Serial passando uma taxa de transmissão em 9600 bps (Bits por Segundo).

Já o *pinMode* apenas configura o comportamento do pino específico para Entrada (Input) ou Saída (Output).

O *digitalRead* lê o que se pega no INTERRUPTOR, que definimos como representante do pino 8, apenas realiza uma leitura no pino digital obtendo seu estado atual.

Nota que é chamado uma função chamada *setupBluetoothConnection()*. Essa função é criada mais abaixo, conforme print:

Figura 85 – Função setupBluetoothConnection

```
void setupBlueToothConnection()
{
  blueToothSerial.begin(38400); //Define BluetoothBee BaudRate a taxa de transmissão padrão 38400
  blueToothSerial.print("\r\n+STWMOD=0\r\n"); //define o trabalho de bluetooth em modo escravo
  blueToothSerial.print("\r\n+STNA=AutomaCasa\r\n"); //definir o nome do bluetooth como "AutomaCasa"
  blueToothSerial.print("\r\n+STOAUT=1\r\n"); // Permitir dispositivo emparelhado me conectar
  blueToothSerial.print("\r\n+STAUTO=0\r\n"); // Conexão automática deve ser proibida aqui
  blueToothSerial.print("\r\n +STPIN=5555\r\n");
  delay(2000); // Delay de 2 segundos
  blueToothSerial.print("\r\n+INQ=1\r\n"); //make the slave bluetooth inquirable
  Serial.println("The slave bluetooth is inquirable!");
  delay(2000); // Delay de 2 segundos
  blueToothSerial.flush();
}
```

Fonte: Print da tela de desenvolvimento do IDE do Arduino

Essa função define todas as configurações para o funcionamento do Bluetooth.

4.3.1 Loop do Arduino

Desenvolvemos o outro bloco de comando chamado LOOP. O Loop é o principal bloco de comando de seu código onde serão executadas indefinidas vezes.

Assim como as demais Linguagens de Programação, a programação do Arduino permite inserir comandos de estruturas de controles, tais como: *if, else, for, while, switch, return, goto, etc.*

Segue abaixo a nossa sintaxe:

Figura 86 – Declarando Loop

```
void loop()           // rotina principal do programa: loop infinito
{
    char recvChar;    // declara variável

    //verifica se há qualquer dado enviado remoto Bluetooth.
    if(blueToothSerial.available()){
      recvChar = blueToothSerial.read();
      Serial.print(recvChar);
    }

    //verifica se há quaisquer dados enviados do terminal serial local, você pode adicionar outros aplicativos aqui.
    if(Serial.available()){
      recvChar  = Serial.read();
      blueToothSerial.print(recvChar);
    }
```

Fonte: Print da tela de desenvolvimento do IDE do Arduino

Em nosso caso acima, declaramos uma variável para trabalharmos durante todo o decorrer do bloco.
O *bluetoothSerial.available()* ou *Serial.available()* obtém o número de caracteres ou bytes, que estão disponíveis para leitura através da porta.
O *Serial.print(data)* envia os dados pela porta Serial.
O *bluetoothSerial.read()* ou *Serial.read()* lê os dados que estão entrando pela porta serial.

Continuando no bloco de comandos Loop, precisamos tratar todas as condições de entrada de dados pela Serial, ou seja, sempre que vim comandos de

um dispositivo externo, Celular ou Computador, o programa deve tratar e manipular esses dados.

Para isso utilizamos o SWITCH, conforme imagem abaixo:

Figura 87 – Tratando dados recebidos

```
switch(recvChar/*dadoRecebido*/){     // quando recebe algo pela serial entra aqui para verificar se deve executar alguma tarefa

  case 'a':                           // Quando recebe 1 em dadoRecebido entra aqui
    if (quarto == 1){                // se PORT já está acionado
      quarto = 0;
      digitalWrite(QUARTO, LOW);    // PORT assume nível lógico 0
    }else{                            // senão
      quarto = 1;
      digitalWrite(QUARTO, HIGH);   // PORT assume nível lógico 1
    }
    break;

  case 'b':                           // Quando recebe 1 em dadoRecebido entra aqui
    if (sala == 1){                // se PORT já está acionado
      sala = 0;
      digitalWrite(SALA, LOW);    // PORT assume nível lógico 0
    }else{                            // senão
      sala = 1;
      digitalWrite(SALA, HIGH);   // PORT assume nível lógico 1
    }
    break;

  case 'c':                           // Quando recebe 1 em dadoRecebido entra aqui
    if (piscina == 1){              // se PORT já está acionado
      piscina = 0;
      digitalWrite(PISCINA, LOW);    // PORT assume nível lógico 0
    }else{                            // senão
      piscina = 1;
      digitalWrite(PISCINA, HIGH);   // PORT assume nível lógico 1
    }
    break;
```

Fonte: Print da tela de desenvolvimento do IDE do Arduino

Visando o controle de fluxo do programa, criamos o Switch a fim de que o programa possa checar cada caso possível, e executar o seu comando específico. A ideia do Switch é eliminarmos a ideia de adicionarmos diversas linhas com IF/ELSE. No nosso caso acima, cada *case* do Switch é uma possibilidade de condição. Nota-se que a cada caso, é tratado um ambiente específico.

O *digitalWrite* apenas escreve valor HIGH (Alto) ou LOW (Baixo) em um pino digital específico. Seria a mesma ideia de True ou False, 0 ou 1.

Após o Switch, suspendemos a execução do programa utilizando um delay de 300 milisegundos. Segue a forma de inserir esse delay: *delay (300);*

Agora vamos trabalhar com os interruptores.

Há um certo cuidado ao trabalhar com interruptores pois os mesmos podem alterar o estado rapidamente devido ao mecanismo do click do interruptor. Pode ser que haja uma falha no mecanismo “físico” do interruptor, e envie de forma rápida o estado de ligado e desligado em questão de milisegundos.

Para suprir nossa necessidade de inserirmos um interruptor no projeto, desenvolvemos a lógica abaixo para funcionar, sem que exista esse tipo de variação rápida de estado.

Segue abaixo uma imagem mostrando a sintaxe:

Figura 88 – Tratando dados recebidos

```
  interruptorAcionado = digitalRead(INTERRUPTOR);      // aqui faz-se uma verificação do estado do interruptor

  if (interruptorAcionado == interruptorAnterior)     // caso o estado do interruptor tenha sido alterado deve-se verificar estado do PORT
    return;
  else {
    delay (50);
    interruptorAcionado = digitalRead(INTERRUPTOR);
    if (interruptorAcionado == interruptorAnterior)
      return;
    else{
      interruptorAnterior = interruptorAcionado;
      if (quarto == 1){             // se PORT já está acionado
        quarto = 0;
      digitalWrite(QUARTO, LOW);    // PORT assume nível lógico 0
      }else{                            // senão
        quarto = 1;
        digitalWrite(QUARTO, HIGH);   // PORT assume nível lógico 1
    }
  }
}
}
```

Fonte: Print da tela de desenvolvimento do IDE do Arduino

O *digitalRead(INTERRUPTOR)* lê o estado atual do pino digital. Abaixo realizamos as validações necessárias para o funcionamento do interruptor. Se o valor acionado for igual ao valor anterior, ou seja, ele quer ligar um led que já está aceso, ele não executa nenhum procedimento. Nota que no else damos um delay com 50 milisegundos, justamente para não cairmos numa possível falha de mecanismo do interruptor.
Após o delay, ele realiza a mesma validação e caso o valor acionado seja diferente do valor anterior, ele realiza a alteração de estado do pino (Liga ou Desliga o Led, por exemplo).

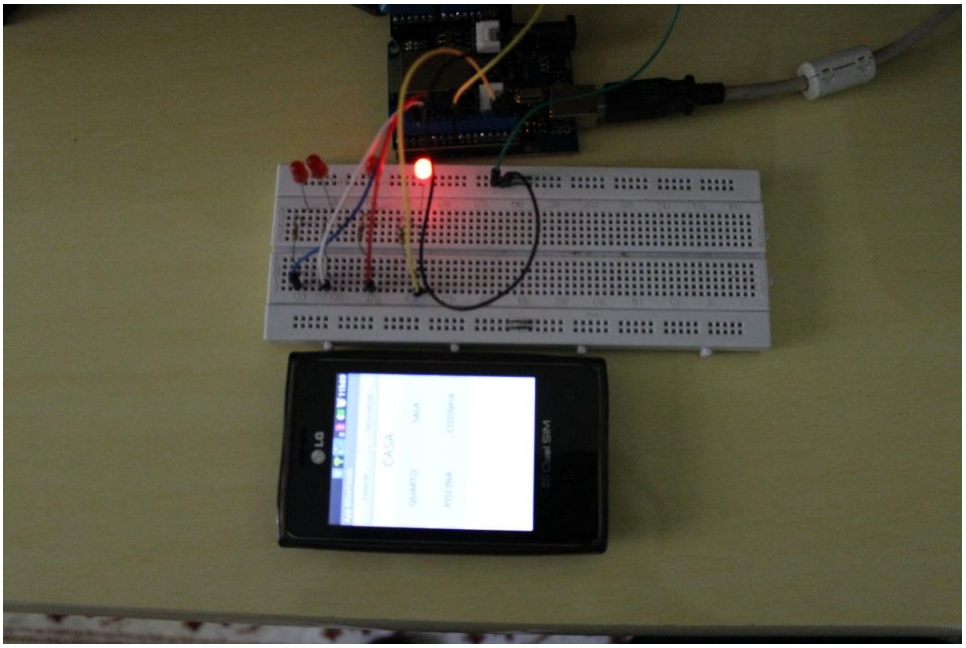
5. CONCLUSÃO

5.1 Testes Finais

As imagens a seguir mostra a operação dos comandos em perfeito funcionamento, acionando e alternando entre os ambientes e também pela chave liga/desliga.

Devemos ressaltar que durante este processo de desenvolvimento inúmeros ajustes de programação foram efetuados, a fim de garantir este resultado final.

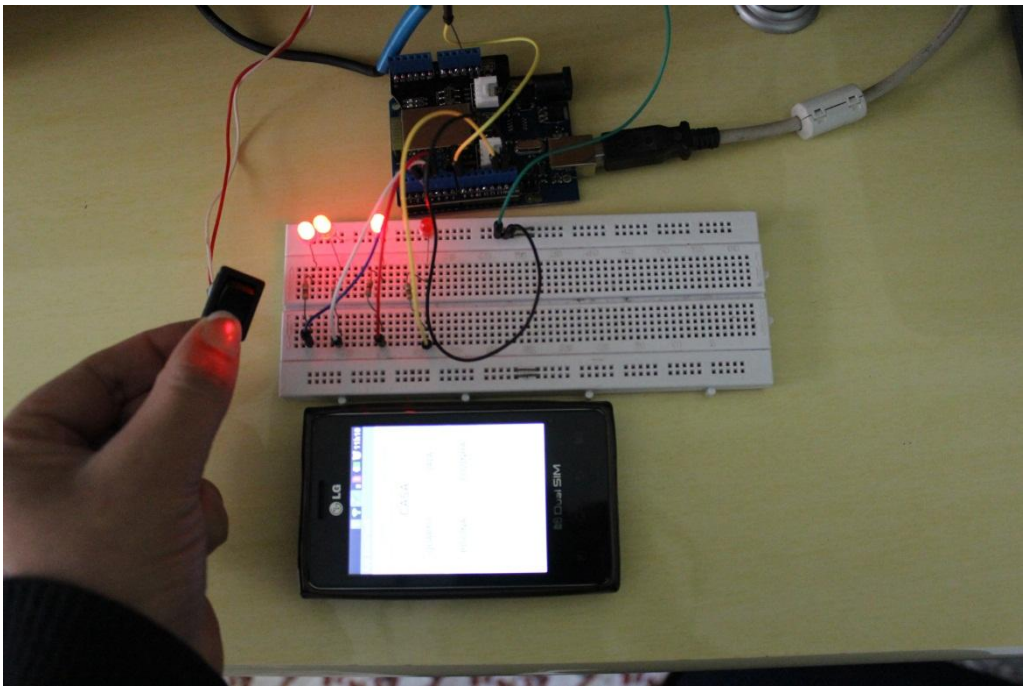
Figura 89 – Primeiro ambiente sendo acionado



Fonte: Imagem elaborada pelo autor

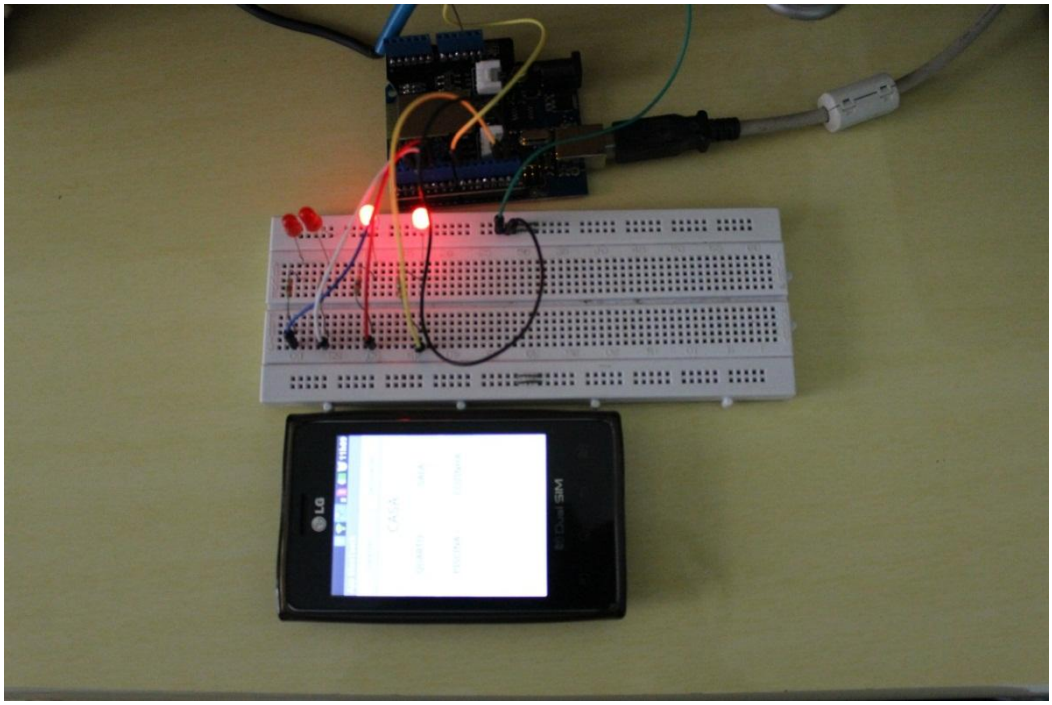
Nesta etapa o importante é frisar, que na ausência do celular o usuário poderá manipular o ambiente de forma convencional, com a chave liga e desliga ou com o interruptor como queira chamar.

Figura 90 – Mesmo ambiente sendo desligado através da chave



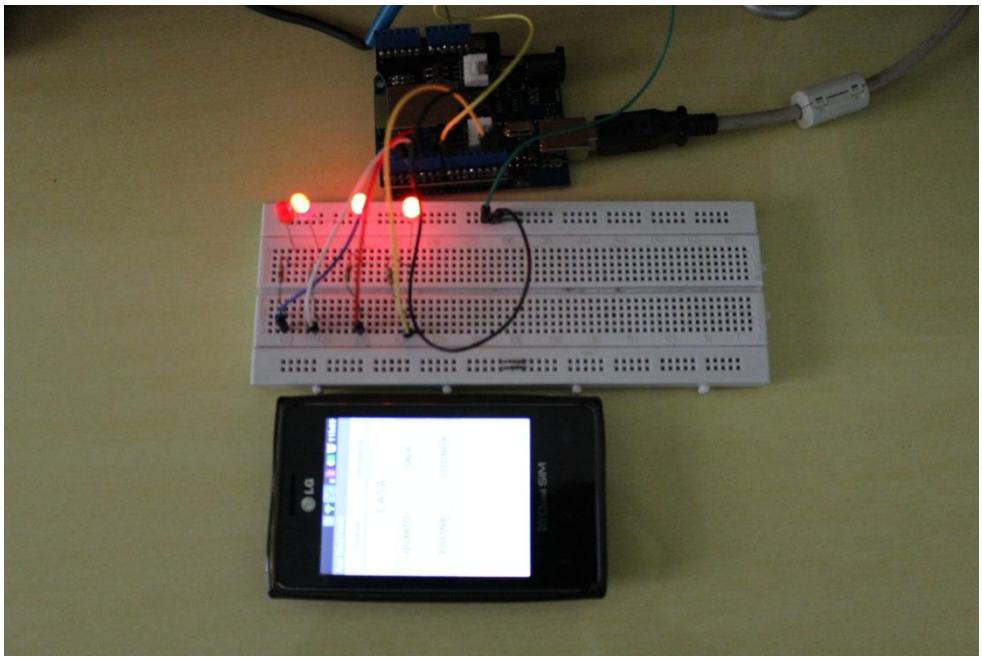
Fonte: Imagem elaborada pelo autor

Figura 91 – Segundo ambiente acionado



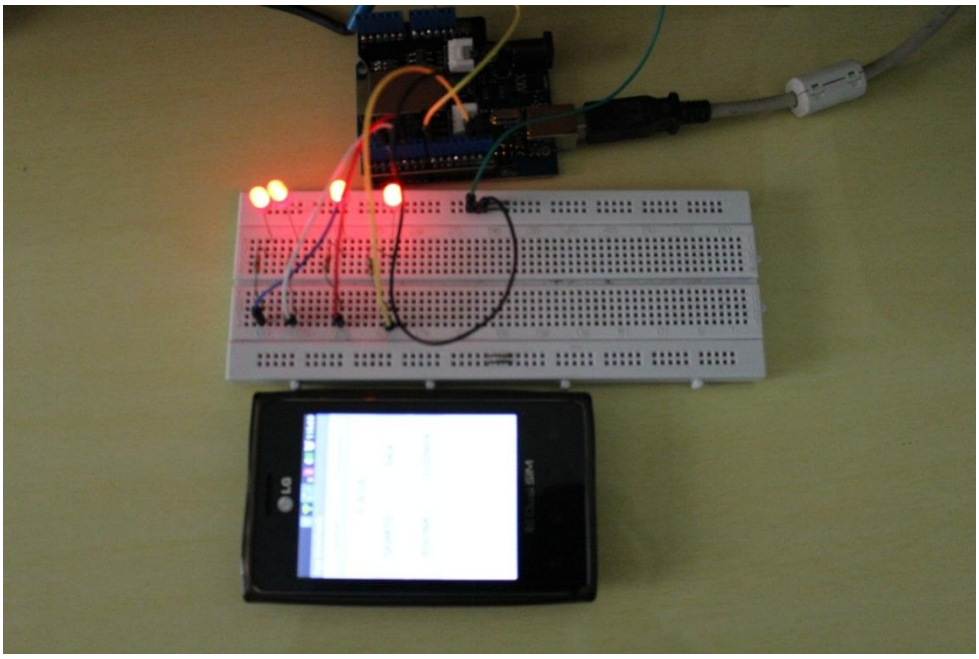
Fonte: Imagem elaborada pelo autor

Figura 92 – Terceiro ambiente acionado



Fonte: Imagem elaborada pelo autor

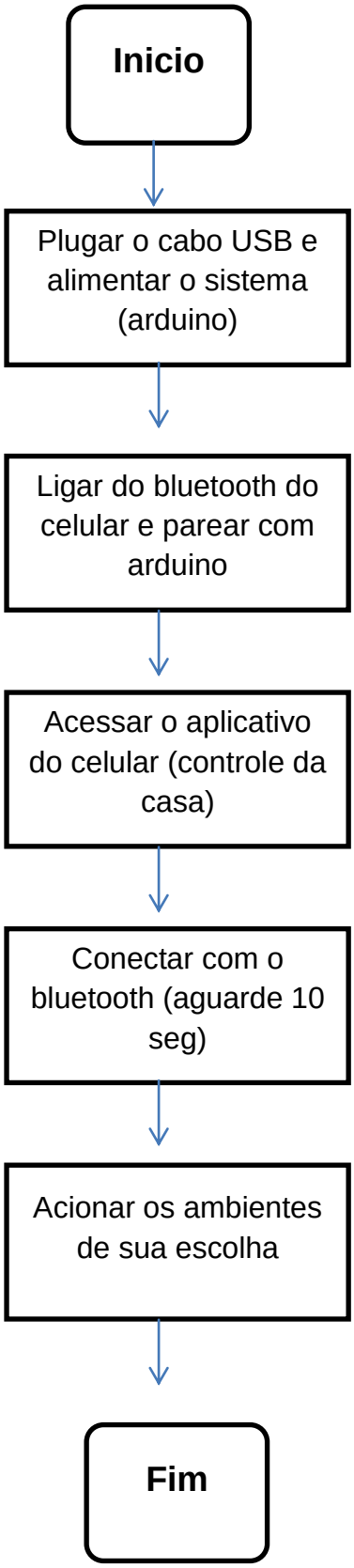
Figura 93 – Quarto ambiente acionado



Fonte: Imagem elaborada pelo autor

Infelizmente nesta etapa desse processo, não houve como documentar o funcionamento da lâmpada de 127v, devido a inconveniência da estruturação, porém iremos demonstrar adiante no projeto final.

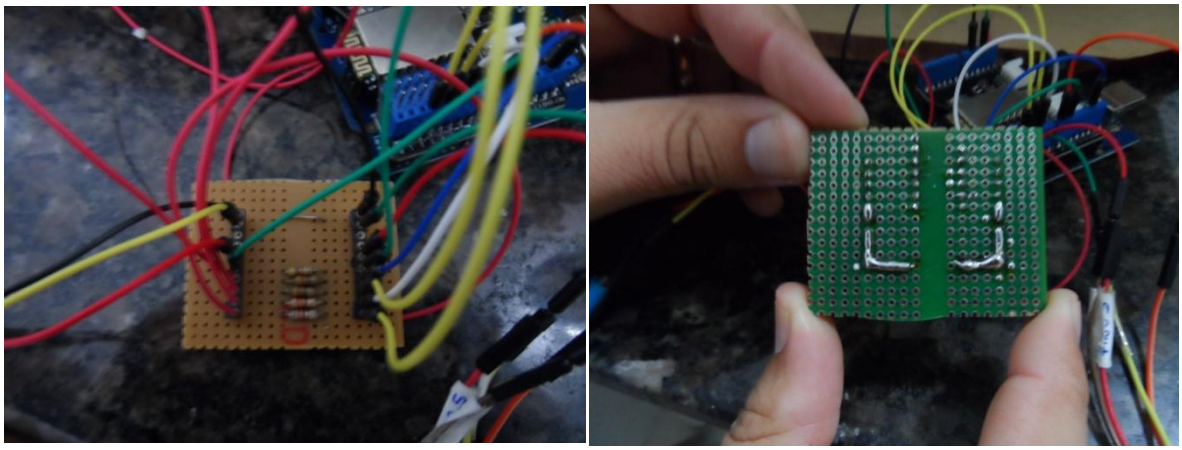
5.2 Fluxo do processo de execução



5.3 Finalização do Protótipo

A imagem em frente verso logo abaixo mostra o circuito impresso, elaborado com exclusividade pelo grupo para que garanta o perfeito funcionamento do projeto.

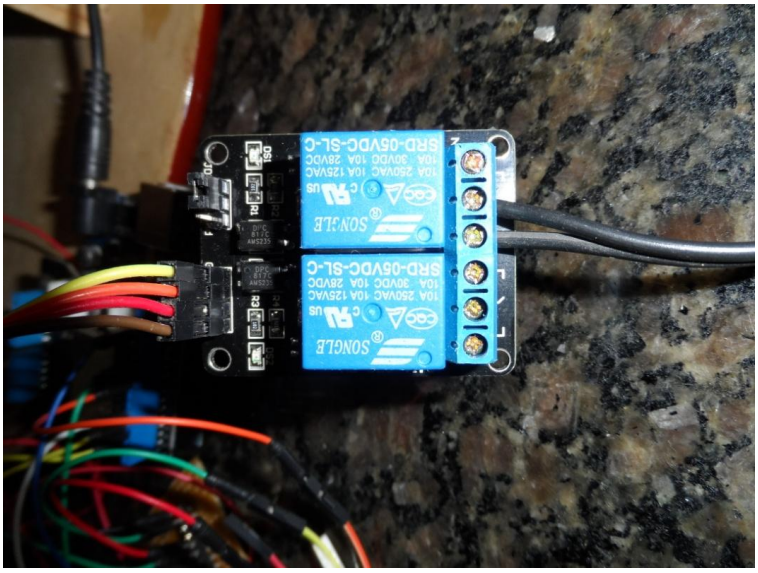
Figura 94 – Circuito Elétrico



Fonte: Imagem elaborada pelo autor

Também podemos observar a placa de relé, como mostra a figura, nela foi instalada uma lâmpada de 127 v. É através dessa placa onde ocorre todo o controle da corrente alterna AC .

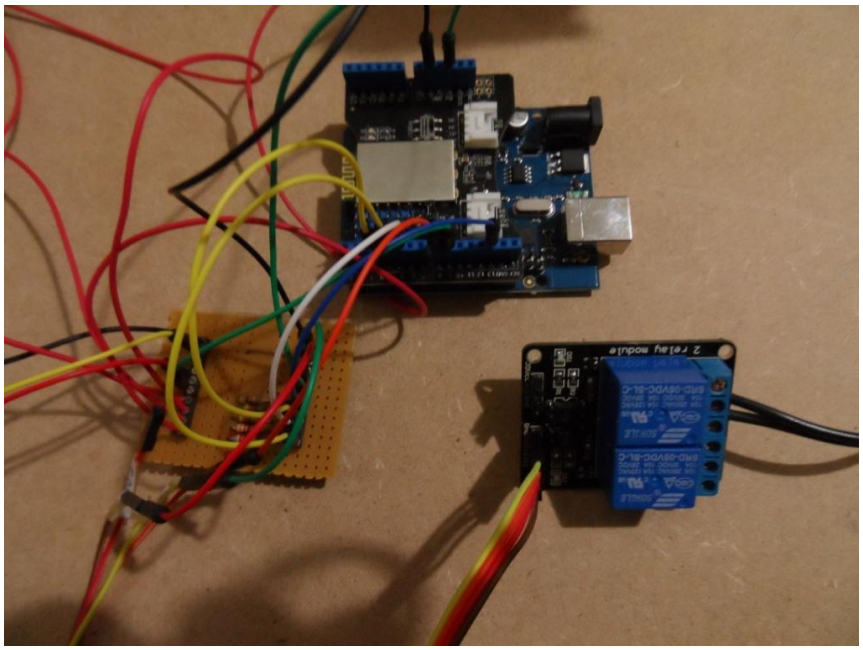
Figura 95 – Placa de Relé



Fonte: Imagem elaborada pelo autor

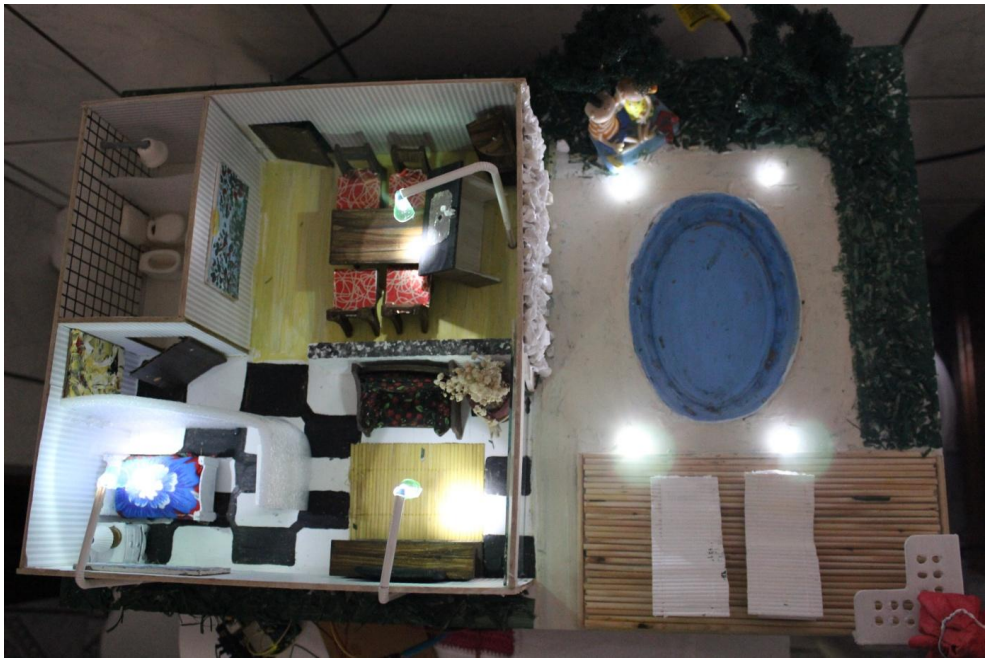
A imagem abaixo mostra como deve ser feita as ligações elétricas entre Arduino, placa de circuito elétrico e a placa de relé para que o projeto funcione de forma correta.

Figura 96 – Ligação elétrica do circuito



Fonte: Imagem elaborada pelo autor

Figura 97 – Maquete acionada



Fonte: Imagem elaborada pelo autor

5.4 Considerações Finais

No decorrer deste trabalho inúmeras dificuldades foram encontradas, tais como: montagem do circuito eletrônico, que por sua vez, poucos do grupo tinham habilidade de compreensão, dificultando o processo de montagem, que por sua vez houve um empenho grande para que todos pudessem se interar do assunto.

O mesmo aconteceu com o App Inventor, esta plataforma de desenvolvimento, fez com que houvesse grande mobilidade na leitura de tutoriais para sua elaboração, o que se tornou em uma grande estafa mental, porém os objetivos foram alcançados.

Embora a preocupação maior fosse com o Arduino que por sua vez linguagem de programação é C, surpreendeu a todos, pois devido as bibliotecas contidas e seu uso é feito de forma referencial.

O custo total do projeto chegou aproximadamente em torno de R$ 450,00 reais, porém este valor poderia ser reduzido, com a fabricação própria do Arduino e de algumas placas. Que devido ao tempo desprendido no projeto e, também pela comodidade e praticidade nos levou a optar por comprá-las, caso não fosse feito isto, necessitaríamos de um tempo muito maior, para desenvolvê-las.

Pontos positivos também foram bastante comuns, como interação de todos os integrantes em todos os processos de desenvolvimento deste trabalho, com isso houve maior aprendizado de um todo. As habilidades manuais ao desenvolver a maquete, construção do circuito eletrônico e todo conteúdo intelectual adquirido na elaboração do trabalho, até mesmo na parte escrita, pode-se agregar valores imensuráveis a cada um dos participantes.

Algumas modificações poderiam ser feitas como o uso da Ethernet, uma placa que se comunica pela rede ao invés do uso do bluetooth, também poderíamos optar o uso do comando por voz para acionar as luzes.

Até mesmo poderíamos acionar outros dispositivos não somente luzes como rádios, micro-ondas, TVs etc. É obvio que para se realizar tudo isto seria necessário um tempo muito maior e muito mais conhecimentos.

Referência Bibliográfica

McROBERTS, Michael - **Arduino Básico** / Michael McRoberts ; [tradução Rafael Zanolli]. - São Paulo : Novatec - Editora, 2011.

Arduino™ 2011/2013. Disponível em: <http://www.arduino.cc/>. Acesso em: 16, out. 2013.

MARTINS, Alex Giuliano. Um blog sobre programação, Arduino – Conhecendo o Arduino UNO. 3, ago. 2012.
Disponível em: <http://alexmartins.net/blog/category/eletronica/>. Acesso em: 8, out. 2013.

Arduino Projetos. Automação, Hobby, Programação e Projetos. Disponível em: <http://www.arduino-projetos.com.br/>. Acesso em: 30, set. 2013.

MIT App Inventor. © 2012-2013 Massachusetts Institute of Technology. Disponível em: <http://appinventor.mit.edu/explore>. Acesso em: 29, set. 2013.

MIT App Inventor. © 2012-2013 Massachusetts Institute of Technology. Disponível em: <http://appinventor.mit.edu/>. Acesso em 8, nov. 2013.

REVISTA FINESTRA (Arquitetura/Tecnologia/Ecoeficiência) - Casa Do Futuro Já É Realidade No Mercado. Ed. 68, Maio/Junho de 2011. Disponível em: <http://www.sinduscon-rio.com.br/sindusletter/sindusletter_030811/n33.htm>. Acesso em: 5 set. 2013.

Dsconto ...e você nunca perderá uma oferta. App Inventor: android, 2012, Brasil, saiba tudo aqui. 2012. Disponível em: <http://www.dsconto.com/app-inventor-android-2012-brasil-saiba-tudo-aqui/>. Acesso em: 16, out. 2013.

Programma Computação Científica. 2013. Disponível em: <http://www.programma.com.br/index.php?option=com_content&view=article&id=18&Itemid=29>. Acesso em: 16, out. 2013.

Wikipédia, A enciclopédia Livre. 2013. Disponível em: <http://pt.wikipedia.org/wiki/Android>. Acesso em: 7, nov. 2013.

REVISTA ABRIL SUPER INTERESSANTE. Conheça a história do Android, o sistema operacional mobile da Google. 16, ago. 2011. Disponível em: <http://super.abril.com.br/galerias-fotos/conheca-historia-android-sistema-operacional-mobile-google-688822.shtml#0>. Acesso em: 8, nov. 2013.

ANEXO - CÓDIGO FONTE DO PROGRAMA

```
#include <SoftwareSerial.h>   // Inclui biblioteca de comunicação serial
#define RxD 6                 // Define pino 6 como entrada Rx para comunicação com módulo Bluetooth
#define TxD 7                 // Define pino 7 como Saída Tx para comunicação com módulo Bluetooth
#define QUARTO 13             // PINO DIGITAL 2 será chamado de QUARTO, assim, quando quiser utilizar esse pino, basta digitar seu nome no programa
#define SALA 3                // PINO DIGITAL 3
#define PISCINA 4             // PINO DIGITAL 4
#define COZINHA 5             // PINO DIGITAL 5
#define LAMPADA 8             // PINO DIGITAL 8
#define INTERRUPTOR 9         // PINO DIGITAL 9

char quarto;                  // variável utilizada para guardar estado lógico do quarto
char sala;                    // variável utilizada para guardar estado lógico da sala;
char piscina;                 // variável utilizada para guardar estado lógico da piscina;
char cozinha;                 // variável utilizada para guardar estado lógico da cozinha;
int dadoRecebido;             // variável utilizada para guardar um dados recebido pela serial
char interruptorAcionado;     // variável para monitorar o estado do port do interruptor
char interruptorAnterior;     // variável para comparação e detecção de mudança no nivel do PORT do interruptor

#define DEBUG_ENABLED  1      // Debug programaçao

SoftwareSerial blueToothSerial (RxD, TxD);    // Refere a comunicação serial padrão dos pinos 0 e 1 do arduino para os pinos definidos Rx e Tx para comunicação com Bluetooth

void setup()
```

```
{
  Serial.begin(9600);           // Configura comunicação serial com Baud Rate em 9600
bps
  pinMode(RxD, INPUT);          // configura pino RxD do arduino como entrada digital
  pinMode(TxD, OUTPUT);         // configura pino TxD do arduino como saída digital
  setupBlueToothConnection();    // Chama rotina para configurar o módulo bluetooth
  pinMode (QUARTO, OUTPUT);      // define QUARTO como saida
  pinMode (SALA, OUTPUT);        // define SALA como saída
  pinMode (PISCINA, OUTPUT);     // define PISCINA como saida
  pinMode (COZINHA, OUTPUT);     // define COZINHA como saida
  pinMode (LAMPADA, OUTPUT);     // define LAMPADA como saida
  pinMode (INTERRUPTOR, INPUT);  // define INTERRUPTOR como entrada
  interruptorAcionado = digitalRead(INTERRUPTOR);      // aqui faz-se uma
verificação do estado do interruptor
  interruptorAnterior = interruptorAcionado;          // iguala variável para posterior
comparação
  digitalWrite(LAMPADA, HIGH);    // PORT assume nível lógico 1 pois o módulo rele
utilizado tem por padrão ficar ativo em 0, ou seja, para acionar o rele, deve-se
assumir nivel 0 no port
}

void loop()          // rotina principal do programa: loop infinito
{
  char recvChar;    // declara varíavel

  if(blueToothSerial.available()){          //verifica se há dados enviados pelo bluetooth
    recvChar = blueToothSerial.read();
    Serial.print(recvChar);
  }
  if(Serial.available()){                  //verifica se há dados enviados pela serial do
arduino
    recvChar  = Serial.read();
    blueToothSerial.print(recvChar);
  }
```

```
 switch(recvChar/*dadoRecebido*/){     // quando recebe algo pela serial entra aqui para verificar se deve executar alguma tarefa

   case 'a':                    // Quando recebe 1 em dadoRecebido entra aqui
    if (quarto == 1){             // se PORT já está acionado
      quarto = 0;
      digitalWrite(QUARTO, LOW);    // PORT assume nível lógico 0
    }else{                       // senão
      quarto = 1;
      digitalWrite(QUARTO, HIGH);   // PORT assume nível lógico 1
    }
    break;

   case 'b':                    // Quando recebe 1 em dadoRecebido entra aqui
    if (sala == 1){               // se PORT já está acionado
      sala = 0;
      digitalWrite(SALA, LOW);      // PORT assume nível lógico 0
    }else{                       // senão
      sala = 1;
      digitalWrite(SALA, HIGH);     // PORT assume nível lógico 1
    }
    break;

   case 'c':                    // Quando recebe 1 em dadoRecebido entra aqui
    if (piscina == 1){            // se PORT já está acionado
      piscina = 0;
      digitalWrite(PISCINA, LOW);    // PORT assume nível lógico 0
      digitalWrite(LAMPADA, HIGH);   // PORT assume nível lógico 1 - rele desligado
    }else{                       // senão
      piscina = 1;
      digitalWrite(PISCINA, HIGH);   // PORT assume nível lógico 1
      digitalWrite(LAMPADA, LOW);    // PORT assume nível lógico 0 - rele acionado
    }
```

```
    break;

  case 'd':                    // Quando recebe 1 em dadoRecebido entra aqui
    if (cozinha == 1){           // se PORT já está acionado
      cozinha = 0;
      digitalWrite(COZINHA, LOW);    // PORT assume nível lógico 0
    }else{                       // senão
      cozinha = 1;
      digitalWrite(COZINHA, HIGH);   // PORT assume nível lógico 1
    }
    break;
 }
 delay (300);                    // Conta 1 segundo para continuar o programa

 interruptorAcionado = digitalRead(INTERRUPTOR);      // aqui faz-se uma
verificação do estado do interruptor

 if (interruptorAcionado == interruptorAnterior)      // caso o estado do interruptor
tenha sido alterado deve-se verificar estado do PORT
   return;
 else {
   delay (50);
   interruptorAcionado = digitalRead(INTERRUPTOR);
   if (interruptorAcionado == interruptorAnterior)
     return;
   else{
     interruptorAnterior = interruptorAcionado;
     if (piscina == 1){           // se PORT já está acionado
       piscina = 0;
       digitalWrite(PISCINA, LOW);    // PORT assume nível lógico 0
       digitalWrite(LAMPADA, HIGH);   // PORT assume nível lógico 1 - desliga rele
     }else{                       // senão
       piscina = 1;
       digitalWrite(PISCINA, HIGH);   // PORT assume nível lógico 1
```

```
        digitalWrite(LAMPADA, LOW);    // PORT assume nível lógico 0 - aciona rele
      }
    }
  }
}

void setupBlueToothConnection()                    //Rotina para configuração do
módulo bluetooth shield
{
  blueToothSerial.begin(38400);                    //Configura Baud Rate (velocidade)
para do Bluetooth Shield para 38400
  blueToothSerial.print("\r\n+STWMOD=0\r\n");       //Configura módulo Bluetooth
para operar como escravo (Slave)
  blueToothSerial.print("\r\n+STNA=AutomaCasa\r\n");    //Configura nome para o
identificação do módulo
  blueToothSerial.print("\r\n+STOAUT=1\r\n");       //Permite disposito pareado ligar
o bluetooth
  blueToothSerial.print("\r\n+STAUTO=0\r\n");       //Desabilitar conexão
automática
  blueToothSerial.print("\r\n +STPIN=5555\r\n");    //Seta código PIN para 5555
  delay(2000);                                     //Tempo de espera para configuração
  blueToothSerial.print("\r\n+INQ=1\r\n");          //Deixar o bluetooth slave receber
dados
  Serial.println("Modulo bluetooth configurado!");    //Envia a mensagem
  delay(2000);                                     //Tempo de espera para configuração
  blueToothSerial.flush();                         //Inicia rotina de comunicação do módulo
Bluetooth com serail Arduino
}
```